VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

SQL edukacinio modelio modeliavimas

Atliko:
Agnė Čeponkutė parašas

Vadovas:
dr. Agnė Brilingaitė

Vilnius
2018

Turinys

Santrauka **4**

Summary **5**

Įvadas **6**

1. Panašių darbų analizė **8**
1.1. Bendro pobūdžio mokymosi platformos 8
1.2. DBVS mokymosi platformos 9
1.3. Automatinis užduočių tikrinimas ir vertinimas 13
1.3.1. Gautų atsakymų pagal įvestą SQL sakinį vertinimas 13
1.3.2. Įvestų SQL sakinių vertinimas 15

2. Edukacinis modelis ir algoritmai **17**
2.1. Skirtingų tipų sakinių struktūros 17
2.2. Algoritmų veikimo modelis 19
2.3. Algoritmai ir modelyje naudojami apibrėžimai 20
2.3.1. SQL sakinių palyginimas 20
2.3.2. Atsakymų palyginimas 30
2.4. SQL sakinių ir pagal juos gautų atsakymų vertinimas 36
2.4.1. SELECT tipo sakinio ir gauto atsakymo vertinimas 36
2.4.2. CREATE tipo sakinio ir gauto rezultato vertinimas 38
2.4.3. INSERT tipo sakinio ir gauto rezultato vertinimas 40
2.4.4. UPDATE ir DELETE tipų sakinių ir gautų atsakymų vertinimas 41

3. Algoritmų įgyvendinimas **41**

4. Algoritmų testavimas **42**
4.1. SELECT tipo sakinių testavimo rezultatai 43
4.2. CREATE tipo sakinių testavimo rezultatai 44
4.3. INSERT tipo sakinių testavimo rezultatai 47
4.4. UPDATE tipo sakinių testavimo rezultatai 48
4.5. DELETE tipo sakinių testavimo rezultatai 50

Išvados ir rekomendacijos **52**

Ateities tyrimų planas **53**

Literatūros šaltiniai **54**

Priedai **56**

A. SELECT tipo sakinio vertinimo rezultatai **57**

B. CREATE tipo sakinio vertinimo rezultatai **58**

C. INSERT tipo sakinio vertinimo rezultatai **60**

D. UPDATE tipo sakinio vertinimo rezultatai **61**

E. DELETE tipo sakinio vertinimo rezultatai **61**

Santrauka

SQL žinios jau tampa bendrosiomis kompetencijomis ne tik informacinių technologijų srityje dirbantiems žmonėms. Dėl šios priežasties įmonėms reikia žmonių, kurie išmanytų duomenų bazių valdymo sistemas (DBVS). Dėl to nė vienas IT ir kitų sričių specialistas negali atsiriboti nuo duomenų bazių valdymo sistemų bei dirbti, neturint DBVS žinių ir praktinių įgūdžių.

SQL kompetencijų įgavimui ir tobulinimui yra sukurtas ne vienas įrankis. Kiekvienas įrankis yra skirtingas ir turi skirtingas funkcijas. Įrankiai naudoja įvairius algoritmus, kurie optimizuoja ir automatizuoja mokymosi procesą. Bet koks vertinimo ir grįžtamojo ryšio suteikimas yra puiki pagalba mokymosi procese.

Šiame darbe yra siūlomas pagrindinių SQL sakinių ir pagal juos gautų atsakymų vertinimo modelis. Pagrindiniuose SQL sakiniuose yra vertinama sakinių sintaksė, pagal sakinius gautas rezultatas, jeigu sakinys yra sintaksiškai teisingas. Algoritmuose yra naudojamos ir sisteminės duomenų bazės lentelės, kurios reikalingos įvertinti pagal tam tikrus sakinius gautiems rezultatams. Algoritmų testavimui yra naudojami realūs studentų SQL sakiniai ir dėstytojo pateikti įvertinimai. Šių algoritmų naudojimas DBVS mokymosi platformoje palengvina rankinį darbą dėstytojams bei pagreitina grįžtamąjį ryšį studentams. Algoritmai įgyvendinti *C#* programavimo kalba, naudojama *Microsoft SQL Server* duomenų bazė.

Summary

SQL Educational Model Modeling

Knowledge about SQL is becoming the main competence for not just people working with information technologies (IT). Thus, companies need employees who can use database management systems (DBMS) efficiently. For this reason, there is a need to prepare more and more IT professionals, but almost none of the IT professionals can distance him/herself from DBMS and work without the knowledge and practical skills.

These days when technologies and DBMS have already been developed for a while there is more than one SQL learning and teaching tool. Each tool is different and has various functions. Many of those tools use different algorithms which optimize learning process. Any evaluation and response system is perfect for learning process.

In this document there are algorithms for automatically generated main types SQL sentences and algorithms for responses given by the entered sentence which compare those sentences. In the main SQL clauses the syntax of sentences is evaluated. The result is also evaluated if the sentence is syntactically correct. Algorithms are also used system database tables that are needed to evaluate the results obtained by certain sentences. Algorithm testing uses real students SQL statements and evaluations provided by the lecturer. Usage of these algorithms in DBMS learning platform makes it easier for lecturers and increases response time with students. All this gives incentive to learn more about databases because you need less and less time for preparation. Algorithms written in *C#* and using *Microsoft SQL Server* databases.

Įvadas

Pagal 2017 m. skelbiamas *INFOBALT* naujienas [5] informacinių technologijų srityje siūlomų darbo vietų kiekis yra didelis ir vis didėja. Pagal atliktą asociacijos *INFOBALT* ir Mokslo ir studijų stebėsenos ir analizės centro (MOSTA) tyrimą [6] prognozuojama, kad augs ne tik programuotojų, bet ir testuotojų, sąsajų kūrėjų, tinklų inžinierių, duomenų bazių is saugumo administratorių, informacinių sistemų analitikų ir architektų bei kitų informacinių ir ryšių technologijų specialybių darbuotojų poreikis. Siekiant išvengti pasirengimo lygio smukimo, reikia gerinti ir technologijas, kurios padeda įgyti būsimiems informacinių technologijų specialistams reikiamas kompetencijas. Viena iš svarbių ir reikalingų kompetencijų, norint būti informacinių ir ryšių technologijų srities specialistu – duomenų bazių valdymo sistemų (DBVS) išmanymas.

Kaip ir beveik visiems studijų dalykams, taip ir duomenų bazių dalykams, studijų programose dažniausiai būna numatytas tik vienas studijų semestras, todėl, norint spėti įgyti kuo daugiau ir kuo aukštesnių bei kokybiškesnių kompetencijų šioje srityje, yra reikalingas įrankis, kuris padėtų pagreitinti ir optimizuoti mokymosi procesą. Įrankis turėtų padėti įgyti ir gilinti praktines DBVS žinias ir kompetencijas, padėti pritaikyti ir naudoti įgytas teorines žinias praktikoje. Be abejo, norint greitai ir kokybiškai mokytis, grįžtamasis ryšys besimokančiajam taip pat turi būti suteikiamas kaip įmanoma greičiau. Norint suteikti greitą grįžtamąjį ryšį, šis procesas turėtų būti automatizuotas, todėl naudojamame įrankyje turėtų būti naudojamas automatinis užduočių tikrinimas ir vertinimas.

Informacinėse technologijose DBVS mokymosi problema nėra nauja, todėl šiuo metu jau yra sukurtas ir naudojamas ne vienas įrankis, skirtas šiai problemai spręsti. Kiekvienas įrankis yra skirtingas, turi skirtingas funkcijas, kai kurie įrankiai yra internetinės aplikacijos ir pasiekiamos bet kuriam vartotojui, tiek studijuojančiam, tiek norinčiam savarankiškai mokytis, kai kurios yra autonominės ir pasiekiamos tik tam tikros mokymosi įstaigos ar kitos bendruomenės nariams. Taip pat dalis įrankių yra orientuoti į visiškai savarankišką mokymąsi, įgyjant tiek teorines, tiek praktines žinias savarankiškai, o kai kurie įrankiai reikalauja ir mokytojo įsikišimo bei yra orientuoti tik į praktinių įgūdžių įgyjimą ir ugdymą. Žiūrint į bendrą mokytojų ir besimokančiųjų poreikį, abu dalykus – galimybę efektyviai mokytis savarankiškai bei efektyviai tikrinti ir vertinti užduotis – būtų naudinga naudoti viename įrankyje, panaudojant kaip įmanoma tikslesnius užduočių atsakymų tikrinimo ir vertinimo algoritmus.

Kylant poreikiui įgyti jau bendrosiomis tampančias SQL kompetencijas, kyla poreikis sukurti SQL edukacinį modelį, kuriame būtų naudojami automatinio tikrinimo ir vertinimo algoritmai. Norint pasiekti šį tikslą, reikia išanalizuoti kitas naudojamas SQL mokymosi platformas, jose naudojamus tikrinimo bei vertinimo algoritmus bei ir juose naudojamas metodikas. Taip pat reikia šanalizuoti dažniausius naudojamus SQL sakinius, naudojamus užduočių apibrėžimus ir dažniausius reikalavimus atliekamoms užduotims. Išanalizavus, būtina sukurti SQL ekukacinį modelį, kuris naudotų tinkamas metodikas ir skaičiavimus, vertinant įvestus SQL sakinius ir pagal juos gaunamus rezultatus, kad algoritmo ir vertintojo įvertinimo skirtumai būtų kaip įmanoma mažesni, o įvertinimai būtų logiški.

Panašių darbų analizėje apžvelgiamos bendro pobūdžio mokymosi platformos, tokios kaip *Moodle*, *VILLE*, *NoobLab*, SQL mokymosi platformos *SQL-Tutor*, *SQLT-Web*, *SQLify*, *SQLator* bei žaidimu grįsto mokymosi platforma, taip pat apžvelgiami platformose *AsseSQL*, *SQLPlus* naudojami SQL sakinių vertinimo algoritmai bei platformose *XData*, *aSQLg* naudojami rezultatų vertinimo algoritmai. Išanalizavus sistemas ir algoritmus buvo nuspręsta kuriamame modelyje naudoti tiek SQL sakinių, tiek pagal juos gautų rezultatų vertinimą, kad būtų galima kaip įmanoma tiksliau

įvertinti atliekamas užduotis.

Siekiant greitai ir efektyviai įgyti praktinių DBVS žinių bei tobulinti kompetencijas, šiame darbe yra siūlomas SQL edukacinis modelis, skirtas pagrindinių SQL sakinių ir pagal juos gautų rezultatų vertinimui. Darbe yra įgyvendinti du algoritmai, skirti pagrindinių SQL sakinių vertinimui. Sakiniai yra vertinami pagal sintaksę bei pagal įvestus sakinius gautą rezultatą. Vertinant pagal įvestus sakinius gautą rezultatą yra naudojamos ir sisteminės duomenų bazės lentelės, kurios yra skirtos įvertinti duomenų bazėje sukurtiems duomenims, kurių negalima gauti iš karto paprastu virtualios lentelės pavidalu. Realizuoti algoritmai galėtų būti integruoti į mokymosi platformą ir turėtų pagreitinti bei palengvinti DBVS mokymosi procesą. Darbe pristatomi sukurti automatinio užduočių tikrinimo ir vertinimo pagal įvestą SQL sakinį ir pagal gautą atsakymą algoritmai.

Panašių darbų analizė pateikta 1 skyriuje. Edukacinis modelis, automatiniai palyginimo ir vertinimo algoritmai detaliai apžvelgiami 2 skyriuje. Algoritmų įgyvendinimas ir panaudojimas apžvelgiamas 3 skyriuje. Algoritmų testavimo rezultatai apžvelgiami 4 skyriuje.

Šis darbas remiasi mokslo tiriamuoju darbu, kuris buvo atliktas antrame magistro studijų semestre. Lyginant su mokslo tiruamuoju darbu, šiame darbe yra praplėsta panašių sistemų ir modelių analizė. Patobulinti automatinio tikrinimo ir vertinimo algoritmai, kurie naudoja jau nebe vieną, o dvi metodikas sintaksiniam sakinių vertinimui. Taip pat algoritmas papildytas ir patobulintas, kad būtų galimybė vertinti ne tik vieno tipo sakinius, bet ir visus pagrindinių tipų SQL sakinius. Algoritmai buvo integruoti į bakalauriniame darbe [1] sukurtą edukacinę platformą *SQLintojas*, o algoritmai testuoti, naudojant realius studentų SQL sakinius bei dėstytojo įvertinimus.

1. Panašių darbų analizė

Darbe analizuojami panašūs darbai – bendro pobūdžio mokymosi platformos, specializuotos DBVS mokymosi platformos, naudojančios automatinio tikrinimo ir/ar vertinimo algoritmus bei patys SQL sakinių automatinio tikrinimo ir/ar vertinimo algoritmai. Analizuojant DBVS mokymosi platformas, bus atsižvelgiama į kelis skirtingus aspektus: įrankio suteikiamas galimybes dėstytojui ir besimokančiajam, įrankyje naudojamus automatinius užduočių tikrinimo ir/ar vertinimo algoritmus, besimokančiajam suteikiamo grįžtamojo ryšio tikslumą ir greitį.

1.1. Bendro pobūdžio mokymosi platformos

Vilniaus universitete yra naudojama mokymosi priemonė *Moodle*, kuri yra skirta lengvesniam komunikavimui tarp dėstytojų ir studentų. Dėstytojai, naudodamiesi Vilniaus universiteto virtualia mokymosi aplinka [19], kuria įvairias užduotis atsiskaitymams, savarankiškam pasiruošimui, įkelia kitą su mokymusi susijusią informaciją ir elektroninę medžiagą. Šis įrankis suteikia galimybę vykdyti ir įvairius elektroninius užduočių atsiskaitymus, kurie gali būti testinio pobūdžio arba atvirų klausimų pobūdžio. Naudojant testinio pobūdžio atsiskaitymą, studentams grįžtamasis ryšys yra suteikiamas iš karto. Esant atviro pobūdžio klausimų atsiskaitymams, kiekvienas studentas turi laukti, kol grįžtamasis ryšys bus suteiktas asmeniškai iš dėstytojo, todėl tai gali užtrukti ilgiau nei norima. Ši platforma yra bendro pobūdžio, todėl nėra pritaikyta specifiškai programavimo kalbų ar DBVS mokymuisi, todėl joje nėra naudojami SQL sakinių automatinio vertinimo algoritmai, tačiau tokius algoritmus būtų galima integruoti ir į tokio pobūdžio mokymosi platformą.

Edukacinė platforma, skirta mokytis programavimo kalbų pagrindams, apžvelgiama Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila ir Tapio Slakoski straipsnyje [14]. Straipnyje apžvelgiamas įrankis, pavadinimu *VILLE*, kuris yra nepriklausomas nuo kalbos vizualizavimo įrankis, teikiantis abstraktų programavimo vaizdą. Įrankis gali būti naudojamas tiek užsiėmimų su dėstytoju metu, tiek savarankiškam mokymuisi. Įrankio privalumai – nėra pririšta prie vienos konkrečios programavimo kalbos mokymosi, yra galimybė vienu metu matyti pavyzdžius skirtingomis programavimo kalbomis, o vykdant programinį kodą ir naudojant derinimo (angl. *debug*) režimą, galima sekti pirmyn ir atgal tarp pažymėtų ribų (angl. *breakpoints*) ir matyti konkrečioje vietoje gaunamą rezultatą. Paskutinysis paminėtas privalumas taip pat nėra standartinis, mokantis ar naudojant įvairius programavimo įrankius. *VILLE* įrankis skirtas pradedantiesiems vartotojams, nes šios mokymo priemonės tikslas yra padėti suvokti, kaip veikia įvairūs programavimo procesai, ciklai. Taip pat įrankį galima naudoti ir kaip atsiskaitymo priemonę, kurioje galima kurti testinio tipo klausimus ir, naudojant iššokančius langus, užduoti studentams klausimus kodo veikimo metu. Kadangi klausimai yra testinio tipo, grįžtamasis ryšys studentui yra suteikiamas iškart. Dar vienas privalumas – kiekviena kodo eilutė yra paaiškinama, todėl mokymosi procesas yra palengvinamas, nes studentas iš karto žino, kodėl yra naudojamas vienoks ar kitoks žymėjimas. Kodo eilučių paaiškinimai yra generuojami automatiškai, todėl dėstytojui taip pat nereikia papildomai dirbti prie kiekvienos užduoties.

VILLE programa apima mokymosi sritį, kai studentas mato, kaip vyksta procesai pagal parašytą programos kodą, turi galimybę programos kodo veikimo eigoje atsakinėti į dėstytojo sukurtus testinio tipo klausimus, tačiau neapima srities, kai studentas pats rašo programinį kodą. Dėl šios priežasties neužtenka vieno įrankio įvertinti visoms studento kompetencijoms, nes taip yra tobulinamos proceso suvokimo kompetencijos, tačiau nėra žinoma, ar studentas gebėtų pats realizuoti veikiantį programos kodą. Taip pat dėstytojai gali vertinti tik teorinio programinio kodo veikimo

proceso suvokimą, bet negali vertinti praktinių studento programavimo žinių. Nors platforma ir naudoja kelias skirtingas programavimo kalbas, SQL sintaksė ir SQL sakinių mokymasis nėra integruoti į šią programą, todėl, norint gilinti DBVS žinias, reikėtų rinktis kitą labiau specializuotą įrankį.

Dar viena mokymosi platforma, skirta mokytis programavimo kalbą, yra apžvelgiama Gordon Hunter, David Livingstone, Paule Neve ir Graham Alsop straipsnyje [4]. Apžvelgiama platforma *NoobLab* naudojasi tiek dėstytojai, tiek studentai. Studentai gali įgyti ne tik praktinių, bet ir teorinių įgūdžių. Programos dizainas yra paremtas tuo, kad vienoje lango dalyje studentas mato užduoties ar teorinės dalies aprašymą. Prie teorinės dalies yra pateikiamas ir kodo pavyzdys, kurį studentas pats gali kompiliuoti ir modifikuoti. Kitoje lango dalyje studentas mato programinį kodą, kurį gali keisti ir vykdyti bei gauti atsakymą. Ši platforma turi kelis privalumus – sąveikos, tokios kaip programinio kodo pakeitimas ir vykdymas, registravimas ir stebėjimas, vizualus progreso vaizdavimas, automatizuotas plagijatų nustatymas. Sistema registruoja kiekvieną studento programinio kodo vykdymą ir, naudodama Levenšteino algoritmą (angl. *Levenshtein algorithm*), nustato kiek daug įvyko programinio kodo pakeitimų tarp skirtingų jo vykdymų. Progreso vaizdavimas yra paremtas ne tik tuo, ar studentas užduotį atliko teisingai, ar neteisingai, ar kurias užduotis vykdė neįvesdamas atsakymo, tačiau tuo pačiu metu yra vizualiai pateikiama, kiek kartų atlikdamas užduotį studentas vykdė programinį kodą, prie kurios užduoties grįždavo atgal. Dar vienas svarbių dalykų, kad šioje platformoje naudojamas automatinis plagijatų nustatymas. Kiekvieną kartą kai programos failas yra išsaugomas, programa automatiškai pažymi antraštėje unikalų numerį. Studentas mato šį ženklinimą, tačiau jis yra užšifruojamas, o dar kartą įkėlus tą patį failą į sistemą, žymėjimas yra dešifruojamas programos ir ieškoma, ar šis failas tikrai buvo susietas su jį įkeliančiu naudotoju. Taip yra automatiškai įvykdomas autoriaus tikrinimas ir dėl to yra išvengiama failų dalinimosi problemos, nes keli studentai nebegali naudoti to pačio failo. Nors platforma yra gerai vizualiai pritaikyta, kad būtų patogi naudojimui, ir taip pat turi labai gerą pateikimą ir grįžtamojo ryšio suteikimą apie užduočių atlikimą, ji nėra pritaikyta mokytis ir gilinti SQL žinias. Šioje platformoje naudojamą funkcionalumą grįžtamojo ryšio suteikimui bei plagijatų išvengimui taip pat būtų galima pritaikyti ir kitose programavimo kalbų mokymosi platformose.

Vreda Pieterse straipsnyje [12] aprašo dar vieną platformą, skirtą automatiniam užduočių vertinimui. Kuriama programa turėtų būti nepriklausoma nuo programavimo kalbos, tačiau šiuo metu palaiko tik *C++* programavimo kalbą. Vertinimas yra kiek kitoks, nei didelėje dalyje platformų, nes vertinimas ir užduočių tikrinimas vyksta ne pagal sintaksinį tikrinimą ar konkretaus rezultato gavimą. Studentas į sistemą turi įdėti visus reikiamus failus, kad programa veiktų ir kompiliuotųsi. Studentui įvedus užduotį (visus reikiamus failus), sistema, naudodama testus, tikrina sukurtos programos funkcionalumą ir galimybes. Sistemoje taip pat yra naudojamas kokybės įvertinimas – efektyvumo ribos tikrinimas. Jeigu studento programos vykdymo laikas viršija numatytą maksimalų laiką, ji yra sustabdoma ir nevertinama. Ši funkcija taip pat yra naudinga rengiant programavimo olimpiadas ir konkursus.

1.2. DBVS mokymosi platformos

Kadangi DBVS mokymasis yra labiau specifinis, o rašomi SQL sakiniai yra kitokios struktūros, nei kitos išpopuliarėjusios programavimo kalbos, todėl didelėje dalyje edukacinių platformų, skirtų tobulinti programavimo žinias, nėra įtrauktas SQL mokymasis. Dėl šios priežasties SQL mokymuisi yra sukurta ne viena specializuota sistema.

Viena tokių specializuotų sistemų yra apžvelgiama Antonija Mitrovic straipsnyje [9]. Straips-

nyje aprašomas įrankis *SQL-Tutor* yra priskiriamas pažangiai apmokymo sistemai (angl. *Intelligent Tutoring System (ITL)*). Tai reiškia, kad sistema pritaikyta pagal studentų žinias ir mokymosi galimybes. Šis įrankis yra skirtas tik praktinių kompetencijų įgijimui ir kėlimui, todėl įrankis neatstoja pilno DBVS mokymosi. Straipsnyje yra pabrėžiama, kad studentai turi lygiagrečiai išklausyti ir teorinį kursą, o įrankis tėra skirtas kaip pagalba ir praktinių įgūdžių tobulinimo aplinka. Naudojantis *SQL-Tutor* yra tobulinamos tik SELECT tipo sakinio žinios, tačiau autorė neįvardina to kaip problemos, nes visų tipų sakiniai yra paremti ta pačia SELECT tipo sakinio struktūra.

Vienas iš platformos privalumų – klaidos pranešimai nėra sukoncentruoti tik į sintaksės problemas, tačiau taip pat gali generuoti pranešimus apie semantines klaidas. Pačią sistemą sudaro grafinė vartotojo sąsaja (angl. *interface*), pedagoginis modelis (angl. *pedagogical module*) ir studentų modeliatorius (angl. *student modeller*). Šiuo atveju vartotojo grafinės sąsajos nenagrinėsime. Studentų modeliatoriaus veikimas yra kiek įdomesnis – sistema modeliuoja studentus, atsižvelgdama į jų pateiktus sprendimus, lyginant juos su idealiais sprendimais. *SQL-Tutor* apribojimų bazę sudaro keli šimtai apribojimų, kurie yra įgyjami analizuojant srities žinias ir studento teisingus ir neteisingus atsakymus. Kiekvienas apribojimas turi unikalų numerį, kuriame yra aktualumo ir pasitenkinimo modeliai. Šie modeliai gali būti bet kokios loginės formulės, susidedančios iš bet kokio kiekio sąlygų. Kai kurios sąlygos atitinka tam tikras studento sprendimo dalis siekiant konkretaus modelio arba idealaus sprendimo. Studento modelis pateikia bendrą informaciją apie studentą, anksčiau išspręstų problemų istoriją ir informaciją apie apribojimų naudojimą. Kiekvienam apribojimui *SQL-Tutor* saugo informaciją apie tai, kiek kartų buvo nustatyta, kad ji yra tinkama idealiems sprendimams, kiek kartų ji iš tikrųjų buvo naudojama studento ir kiek kartų ji buvo teisingai naudojama. Ši informacija saugoma pagal tris rodiklius (susijusius, naudojamus ir teisingus), kad būtų pasirinktos naujos problemos ir atsinaujintų studentų modeliuotojas.

Pedagoginis modelis yra visos sistemos pagrindas. Būtent šis modelis ir atrenka problemas, kurios turi būti pateiktos studentui ir kuria mokymo veiksmus pagal studentų modelį. Taip pat šiame modelyje yra generuojamas ir grįžtamasis ryšys studentams. Dėl programoje plačiai išvystytos grįžtamojo ryšio sistemos buvo atliktas efektyvumo tyrimas, aprašomas Antonija Mitrovic ir Brent Martin straipsnyje [11]. Grįžtamasis ryšys yra skirstomas į penkis skirtingus lygius: teigiamas/neigiamas grįžtamasis ryšys (angl. *positive/negative feedback*), klaidos ženklas (angl. *error flag*), užuomina (angl. *hint*), visos klaidos (angl. *all errors*), dalinis sprendimas (angl. *partial solution*) ir visas sprendimas (angl. *complete solution*). Žemiausiame lygyje (teigiamas/neigiamas grįžtamasis ryšys) studentas yra paprasčiausiai informuojamas ar jo pateiktas sprendimas yra teisingas ar klaidingas, o esant klaidų pateikiamas ir jų kiekis. Užuominos lygyje studentui yra pateikiama, kurioje sąlygoje buvo padaryta klaida, o užuominos lygyje pranešimas papildomai nurodo klaidos tipą pagal bendrą principą. Visų klaidų lygyje studentui yra pateikiamos visos jo padarytos klaidos su užuominomis. Dalinio sprendimo lygyje studentui yra pateikiama teisinga sąlygos dalis, kurioje buvo padaryta klaida, o viso sprendimo lygyje studentui paprasčiausiai yra pateikiamas idealus užduoties sprendimas. Tyrimo metu iškelta hipotezė – žemo lygio grįžtamasis ryšys, kuriame pateikiama visa išsami informacija ir teisingi sprendimai, nėra produktyvus, o aukšto lygio grįžtamasis ryšys, susijęs su bendraisiais srities principais, yra labai veiksmingas.

Atliekant tyrimą studentai buvo paskirstyti atsitiktinimai į grupes pagal programos suteikiamą grįžtamąjį ryšį, visi studentai atlikinėjo tas pačias užduotis. Tyrimo metu buvo stengiamasi išsiaiškinti, ar tam tikras grįžtamasis ryšys padeda studentui greičiau mokytis. Po atlikto tyrimo paaiškėjo, kad pirmenybė turėtų būti teikiama aukštesnio lygio grįžtamajam ryšiui, kai studentui yra pateikiamas visas klaidų sąrašas ar užuomina. Abu šie pranešimų lygiai leidžia sugaišti mažiausiai laiko vienam bandymui, taip pat su šio lygio pranešimais reikia mažiausiai bandymų

problemos srendimui. Kaip galima matyti, programa turi tikrai gerą grįžtamojo ryšio suteikimą, tačiau ji kiek sunkiau gali būti naudojama atsiskaitymams paskaitų metu, o ne tik individualiam mokymuisi, nes atliekamos užduotys nėra vertinamos balais. Taip pat studentas negali pamatyti savo gaunamo atsakymo pagal parašytą SQL sakinį. Dėl šios priežasties studentas nemato klaidingai atrenkamų rezultatų ir tai gali daryti įtaką užduoties atlikimo laikui ir motyvacijai.

Dar vienas, beveik toks pats įrankis, skirtas kelti DBVS kompetencijas, aprašomas kitame Antonija Mitrovic straipsnyje [10]. Įrankis *SQLT-Web* paremtas prieš tai apžvelgtu įrankiu *SQL-Tutor*, tačiau yra nebe darbastalinė (angl. *desktop*), o internetinė versija. Žiniatinklinė aplikacija turi kelis privalumus, lyginant su darbastaline versija – mažinamos programinės įrangos platinimo vartotojams ir suderinamumo su aparatine ir programine įranga problemos. Taip pat žymiai greičiau yra pasiekiami programos atnaujinimai, o, svarbiausia, besimokantieji nebėra pririšti prie konkrečios mokymosi vietos kompiuterių klasėse. *SQLT-Web* funkcionalumas yra visiškai identiškas *SQL-Tutor* funkcionalumui. Kaip skirtumas, straipsnyje paminėta, kad yra saugoma istorija apie studentus, kuri apibendrina studento žinias ir saugo esamos ir ankstesnių sesijų istoriją. Kaip ir naudojant *SQL-Tutor*, taip ir naudojant *SQLT-Web* buvo atlikti keli tyrimai. Vienas iš tyrimų apima efektyvaus grįžtamojo ryšio suteikimo sritį, kitas tyrimas apima efektyvios versijos naudojimo sritį, o trečiojo tyrimo tikslas – išanalizuoti studentų metakognityvinius gebėjimus. Po pirmojo ir antrojo tyrimo dalyvavusiems studentams buvo įteikiama anketa, kurią studentas turėjo užpildyti. Klausimyno tikslas buvo įvertinti studentų suvokimą apie *SQLT-Web*. Didžioji dalis tyrime dalyvavusių studentų teigiamai įvertino naudojamą praktinę sistemą ir jos teikiamą naudą. Dėl šios priežasties galima teigti, kad ši edukacinė platforma, kaip ir *SQL-Tutor*, yra gerai ištobulinta ir atitinka daugelio studentų poreikius bei padeda kelti reikiamas DBVS kompetencijas.

Kita panaši SQL mokymosi sistema yra aprašoma Stijn Dekeyser, Michael de Raadt ir Tien Yu Lee straipsnyje [3]. *SQLify* sistema yra skirta vertinti studentų užklausų rašymo įgūdžius. Galima išskirti tris fazes, kaip studentai naudoja sistemą – tyrimas ir pateikimas, pastabų peržiūra, grįžtamojo ryšio ir įvertinimo gavimas. Studentai pateikia užduočių sprendimus. Pateikti sprendimai yra įvertinami bendraamžių, sistemos *SQLify* ir instruktoriaus. Studentai užpildo atsiliepimus apie kitus studentus, pagal skirtus įvertinimus. Galiausiai balai, gauti už pateikimą ir tikslumą, yra susumuojami ir studentas gauna galutinį įvertinimą. *SQLify* naudoja gana sudėtingą metodą priskirti galutiniams balams už užduotis – yra galimybė gauti platesnį ir tikslesnį vertinimą, nei tik teisingai arba klaidingai; naudojama duomenų bazių teorija, kad būtų gautas kompiuteriu pagrįstas vertinimas.

Kiek kitoks įrankis apžvelgiamas Shazia Sadiq, Maria Orlowska, Wasim Sadiq, Joe Lin straipsnyje [16]. Kaip teigia autoriai – pagrindinė įrankio *SQLator* funkcija yra vertinimas. Ši funkcija leidžia vartotojui įvertinti savo užklausos formuluotės teisingumą. Įvertinimo variklis pagrįstas sudėtingais heuristiniais algoritmais. Taip pat įrankis turi specifinį pranašumą – besimokantysis turi labiau sutelkti dėmesį į patį procesą, o ne į galutinį produktą. Tai yra įmanoma dėl to, kad *SQLator* mokymosi aplinka leidžia sutelkti studentui dėmesį į koncepcijos vystymą, o ne bandymus išgauti vertinimo rezultatą, kurio iš jo tikimasi. Ši mokymosi aplinka yra orientuota ne į teorinį, o į praktinį SQL mokymąsi. Vienas iš būdų siekti tokio mokymosi pobūdžio – spręsti tam tikrą aiškiai apibrėžtų užklausų rinkinį, apimantį įvairius kalbos aspektus, o tada duoti juos patikrinti ir pataisyti SQL vartotojams ekspertams. Būtent šiuo mokymosi metodu ir buvo sukurtas *SQLator*. Jis pakeičia SQL vartotoją ekspertą išmaniuoju varikliu, kuris vertina besimokančiųjų SQL užklausas ir pateikia naudingus atsiliepimus, didinančius besimokančiųjų patirtį.

Kita labai svarbi SQL mokymo ir mokymosi sritis – studentų SQL užklausų vertinimas. Kadangi yra ne vienas būdas, kaip galima teisingai užrašyti tą pačią užklausą, didelių užduočių kiekių

vertinimas gali būti labai sudėtingas. Tai reiškia, kad ne tik užduoties įvertinimas užtrunka labai daug laiko, tačiau tuo pačiu gali kilti ir nemažai grįžtamojo ryšio studentui problemų. *SQLator*, naudodamas savo vertinimo funkciją, labai supaprastina šią problemą. Svarbiausia įrankio dalis yra variklis *SQLator Equivalence Engine*, galintis spręsti, ar pasiūlytas SQL sprendimas yra teisingas. Variklis yra visiškai bendrinis, tai yra jis nenaudoja jokių įkeltų praktikos duomenų bazių ar užklausų. Šiam vertinimo varikliui pakenkti gali tik labai neįprastos užklausų formulės. Variklis susijęs su SQL SELECT tipo sakinio pagrindinėmis savybėmis. Be gerai ištobulinto vertinimo variklio, ši mokymosi platforma taip pat atlieka ir plagiatų nustatymą. Plagiatų aptikimą labai palengvina tai, kad visos besimokančiųjų patvirtintos užklausos yra išsaugomos duomenų bazėje. Dėl to galima palyginti įrašų atitikimus, naudojant teksto eilučių panašumo principus. Teksto eilučių panašumo palyginimas yra atliekamas pašalinus visus tarpus, tabuliacijos ir kitus skyrybos ženklus.

Visiškai kitokia SQL mokymosi technika ir įrankis yra nagrinėjami Mario Soflano, Thomas M. Connolly, Thomas Hainey straipsnyje [17]. Kaip žinia, kompiuterizuotos mokymosi sistemos vis populiarėja, o jų tyrimai rodo vien privalumus, keliant įvairias kompetencijas. Tačiau žaidimu grindžiamas mokymosi stilius, siekant tobulinti švietimo srities patirtis, dar nebuvo plačiai tirtas. Šių tyrimų tikslais buvo sukurtas trijų skirtingų režimų žaidimas: neprisitaikantis režimas, režimas, kuris pritaiko žaidimą pagal studento mokymosi stilių, nustatytą, naudojant mokymosi stiliaus klausimyną ir režimas, turintis žaidimo metu prisitaikančią sistemą, kuri dinamiškai ir nuolat derina turinį pagal studento veiksmus žaidime. Autoriai šiame straipsnyje aptaria prisitaikymo terminą žaidimu grįsto mokymosi kontekste ir pateikia eksperimento, kuriame nagrinėjami skirtingų žaidimo režimų mokymosi efektyvumo skirtumai, rezultatus, lyginant su tradiciniu mokymosi stiliumi.

Kaip teigiama, elektroninis mokymasis tiesiog kartoja tradicinę švietimo sistemą ir gali būti per daug orientuotas į pristatymo būdą, o ne mokymą iš tikrųjų stengiantis ir motyvuojant mokinius įsitraukti į mokymosi procesą. Priešingai nei tradicinis mokymosi stilius, žaidimai, ypač vaizdo, gali ilgą laiką motyvuoti žmones dėl natūralaus žmonių pomėgio žaidžiant vis kartoti žaidimą, kol pasieks aukščiausią rezultatą. Atliekant straipsnyje aprašomą tyrimą, buvo sukurta mokymosi programa, pagrįsta į žaidimą orientuoto mokymosi stiliumi. Žaidimas skirtas mokytis SQL pagrindus, o žaidime naudojamas mokymosi stiliaus modelis *Felder-Silverman*. Šiam tyrimui sukurto žaidimo žanras buvo vaidmenų žaidimas (angl. *role-playing game*), o naudojamas variklis – *NeverWinter Nights 2*. Žaidime integruoti visi trys aukščiau paminėti režimai: neprisitaikantis (angl. *non-adaptive*) režimas, kai visi besimokantieji yra vienodi ir nėra atsižvelgiama į studentui priimtiną mokymosi stilių; ne žaidime prisitaikantis (angl. *out-of-game adaptive*) režimas, kai studento mokymosi stiliaus charakteristikos nustatomos iš anksto, naudojant *Felder-Silverman* mokymosi stiliaus klausimyną, o žaidimo procesas pritaikomas pagal studento mokymosi stilių; žaidime prisitaikantis (angl. *in-game adaptive*) režimas, kai studento charakteristikos nustatomos jau žaidimo metu. Paskutinis režimas įdomus tuo, kad studentas mokymosi stilių gali keisti žaidimo eigoje, o žaidimas turi adaptuotą sistemą, kuri automatiškai pritaiko žaidimą realiuoju laiku, o dabartinis studento mokymosi stilius nustatomas analizuojant žaidėjo istoriją. Skirtumas tarp režimų yra prisitaikančiojo pobūdžio, o kiti žaidimo elementai, tokie kaip siužetas, žaidimo aplinka, yra identiški visuose režimuose.

Šio tyrimo metu studentai buvo mokomi: kaip parašyti SQL komandą, kaip iš duomenų bazės gauti informaciją, naudojant SELECT sakinį, kaip sukurti SQL sakinius, naudojant WHERE, ORDER BY sąlygas ir kitas bendrąsias SQL funkcijas, kaip naudoti raktinį žodį DISTINCT, norint pašalinti besidubliuojančius duomenis. Žaidime yra trys pagrindinės misijos ir kelios pagalbinės

misijos. Kiekvienoje misijoje besimokantysis mokosi rašyti skirtingas SELECT sakinio formas. Kad būtų lengviau atlikti užduotis, yra paruoštos instrukcijos, kurias studentas gali pasirinkti pagal savo mokymosi stilių – tekstas, paveikslėliai, diagramos. Kiekvienai užduočiai yra skirti šeši bandymai. Neteisingai atlikus visus šešis bandymus, besimokantysis nėra blokuojamas ir yra praleidžiamas žaidime toliau, kad nesustotų jo mokymosi procesas. Įvykdžius misijas, studentas gauna atsiliepimus iš pagrindinio žaidimo veikėjo.

Atlikus tyrimą buvo pastebėta, kad mokymosi turinio pritaikymas atsižvelgiant į studento mokymosi stilių turi teigiamą poveikį. Žaidimu grįstas mokymasis gali būti puikiai naudojamas mokytis SQL, nepriklausomai nuo besimokančiojo išsilavinimo, lyties, studijų programos ar mokymosi stiliaus. Taip pat tyrimas parodė, kad mokantis žaidime yra pasiekiami aukštesni rezultatai, nei mokantis tradiciniu stiliumi, pagal vadovėlį.

1.3. Automatinis užduočių tikrinimas ir vertinimas

1.3.1. Gautų atsakymų pagal įvestą SQL sakinį vertinimas

Dar vienas ne ką mažiau svarbus veiksnys keliant ir gilinant kompetencijas – vertinimas ir vertinimo būdas. Julia Coleman Prior ir Raymond Lister straipsnyje [13] kaip tik ir yra apžvelgiamas poveikis, kurį daro vertinimo strategijos mokiniams, kurie kelia kompetencijas duomenų bazių srityje. Taip pat apžvelgiama kaip suprojektuoti tokį vertinimo būdą, kuris skatintų studentą domėtis ir iš tikrųjų įgyti reikiamų įgūdžių. Vertinimo metodas turėtų paskatinti studentus gilinti žinias, o ne žvelgti į viską paviršutiniškai.

Straipsnyje yra iškelti trys tikslai SQL užklausų formulavimo įgūdžiams formuoti: vertinti studentus, tiksliai nustatant jų individualius SQL užklausų formulavimo įgūdžius, vertinti besimokančiuosius tokiu būdu, kuris parodys kaip jie panaudos savo SQL įgūdžius realaus pasaulio programinės įrangos kūrime, skatinti studentus praktikuotis ir tobulinti savo įgūdžius internete. Vienas iš vertinimo metodų – pateikti studentams tam tikrą užduočių rinkinį, pagal kurį studentai turi parašyti SQL užklausas. Tai gali būti kaip popierinė ar elektroninė užduotis. Toks metodas yra taikomas daugelyje mokymosi įstaigų, kurios organizuoja SQL mokymąsi. Tačiau, mokantis šiuo metodu, kyla problema. Nors studentai ir išklausė bei praėjo tokį kursą, tačiau tai nereiškia, kad jie turi pakankamai reikiamų SQL įgūdžių. Kadangi užduotis yra pateikiama kaip rašytas dokumentas, tai visiškai nemotyvuoja studento praktiškai įvykdyti užduotis bei patikrinti, kokį jos duos rezultatą. SQL užklausų kūrimas yra praktinis įgūdis ir negali būti įgyjamas be praktikos. Kai studentai nėra vertinami už gautą rezultatą, dauguma nebūna motyvuoti stengtis, ieškoti ir tikrinti rezultatus. Sukūrus užklausas, jas reikia kiekvieną kartą tikrinti vizualiai ir taip bus įgaunama žymiai daugiau reikiamų kompetencijų.

Tam, kad būtų sprendžiamos aukščiau minėtos problemos, buvo sukurtas internetinis įrankis *AsseSQL*, kuris ir yra apžvelgiamas straipsnyje. Internete yra nemažai ir kitų įrankių, kurie suteikia greitą grįžtamąjį ryšį studentui apie individualias užduotis, tačiau nesuteikia suminio rezultato. Vienas svarbiausių šio įrankio požymių – atliekant užduotį studentas mato atsakymą, kokį turi gauti įvykdęs savo parašytą SQL užklausą. Taip yra pašalinama pateikto klausimo dviprasmiškumo galimybė ir studentui tiksliau pateikiama kokio rezultato iš jo tikimasi. Sistema naudoja dvejetainį vertinimą – užduotis vertinama teisingai arba neteisingai. Dėl tokio vertinimo buvo išreikštas susidomėjimas, kad studentas, kuris visiškai nesigaudo ir studentas, kuris beveik pilnai viską supranta, gaus vienodą įvertinimą – neteisingai – vien dėl to, kad nėra dalinių vertinimų už užduotis. Tačiau, autorių teigimu, iš dalies teisingų atsakymų nėra, o dalinis vertinimas vertina tik studentų

pastangas, o ne atsakymo teisingumą. O po studento pateikto neteisingo atsakymo, sistema iškart informuoja, kur yra klaida, dėl to daliniai vertinimai tik skatintų studentus ne iki galo išsiaiškinti klaidas ir siekti tik dalinių rezultatų.

Gordon Russel ir Andrew Cumming savo straipsnyje [15] pabrėžia apie kompiuterinio mokymosi naudą, veiksmingumą ir efektyvumą, nors, dažniausiai, internetinio mokymosi idėja yra interpretuojama kaip internetiniai užrašai, viktorinos ir internetiniai forumai. Šiame straipsnyje yra nagrinėjamas mokymosi įrankis *SQLPlus*. Straipsnyje yra pateikiama mokymosi platformos apžvalga. SQL yra populiari kalba, norint manipuliuoti duomenimis duomenų bazėje. Taip pat SQL yra prieinamas daugeliui reliacinių duomenų bazių. Mokomieji darbai dažniausiai orientuojasi į duomenų ištraukimo komponentus. Nors pateikiama užduotis skamba ir paprastai, tačiau kiekvieną SQL sakinį galima užrašyti teisingai keliais skirtingais būdais. Dirbdamas su klausimais, studentas susiduria su dviem skirtingo tipo klaidomis – sintaksės klaidomis, kai SQL sakinys neteisingai struktūrizuotas ir nesikompiliuoja, ir loginėmis klaidomis, kai sakinys kompiliuosi, tačiau gaunamas ne tas atsakymas, kurio tikimasi. Dėl šios priežasties internetinė aplikacija turi priėjimą prie viso konteksto ir turimo gauti atsakymo, todėl gali vertinti ne tik sintaksės, bet ir logines klaidas. Naudojantis programa ir atliekant užduotis, studentai savo parašytas SQL užduotis gali vykdyti bet kuriuo metu ir keisti atsakymus, kol bus patenkinti gaunamu rezultatu. Kai yra vykdoma pateikta SQL užklausa, yra analizuojamos ne tik sintaksės klaidos, sistema analizuoja ir gautą atsakymą ir studentui pateikia, koks gautas tikslumas pagal teisingą atsakymą. Atsakymas yra vertinamas procentais, pagal tai, kiek stulpelių ir eilučių tarp studento ir teisingo gaunamų atsakymų sutampa.

Dar viena automatinio vertinimo sistema yra apžvelgiama Haifeng Ke, Gaoyan Zhang, Hui Yan straipsnyje [7]. Elektroninis egzaminavimas yra svarbi šiuolaikinės edukacinės plėtros dalis, kuri gali veiksmingai sumažinti dėstytojų darbo krūvį ir padidinti studentų mokymosi efektyvumą. Šiuo metu yra naudojama nemažai elektroninio egzaminavimo sistemų, tačiau daugelis tokių sistemų yra skirtos subjektyviems dalykams atsiskaityti, dažniausiai pritaikoma testinio pobūdžio atsiskaitymams ir nėra pritaikyta programavimo kalbų mokymuisi. Dėl šios priežasties Zheijang universitete, naudojant *C* programavimo kalbą, buvo sukurta ir išvystyta SQL mokymosi platforma, naudojanti automatinį vertinimą.

SQL programavimo klausimai yra pakankamai subjektyvūs. Norint išspręsti tam tikrą klausimą yra galimas ne vienas sprendimas. Todėl aprašomame įrankyje yra naudojamas automatinis SQL programinių klausimų vertinimas, kuris nėra pagrįstas SQL teiginių struktūros analize, bet yra lyginamas teisingas rezultatų rinkinys ir studento gautas rezultatų rinkinys. Prieš pradedant vertinimą yra atliekamas pirminis paruošimas. Šis paruošimas turi du privalumus – gaunamas didesnis sistemos veikimo efektyvumas ir gali būti pateikti išsamesni klaidų aprašymai. Pirminis paruošimas apima šiuos veiksmus: teisingumo patikrinimas, kuriam įvykus nesėkmingai yra pateikiamas klaidos pranešimas; objekto stulpelių skaičiaus tikrinimas, kurio metu pagal sakinį yra analizuojamas gaunamų stulpelių skaičius, ir esant kitokiam stulpelių kiekiui nei teisingas atsakymas, yra pateikiamas klaidos pranešimas; objektų stulpelių tvarkos tikrinimas, kurio metu pagal sakinį yra analizuojama stulpelių išdėstymo tvarka, kuriai neatitikus yra išvedamas klaidos pranešimas, rodantis netinkamus teiginius. Galutinis rezultatų rinkinio tikrinimas vyksta sukombinuojant stundento ir teisingus atsakymus, naudojant UNION sąlygą. Jeigu eilučių skaičius abiejuose rinkiniuose sutampa, sistema nusprendžia, kad studento gautas rezultatas yra toks pats, todėl užduotis yra atlikta sėkmingai.

Vienas iš sistemos privalumų, lyginant su kitomis anksčiau apžvelgtomis sistemomis – sistemoje yra vertinami ir tikrinami ne tik SELECT tipo sakiniai, bet apimami ir DELETE, UPDATE ir INSERT tipo sakiniai. Šio tipo sakiniams interpretuoti yra keli skirtingi būdai. Galima šiuos sa-

kinius vykdyti tiesiai duomenų bazėje ir palyginti likusius įrašus, tačiau tai gali sukelti problemų, nes patikrinti pavyks tik pirmąjį kartą dėl per didelio duomenų kiekio pasikeitimo. Kitas būdas – galima naudoti laikinąsias lenteles. Kiekvienam studentui suteikus laikinąsias lenteles ir vykdant veiksmus jose, galima vėliau palyginti duomenis su realiose lentelėse esančiais duomenimis ir taip nustatyti, ar užklausa buvo teisinga. Vis gi, šioje programoje pasitelktas dar kitoks metodas – visi sakiniai yra pertvarkomi į SELECT tipo sakinį ir interpretuojami taip pat kaip ir kiti SELECT tipo sakiniai.

1.3.2. Įvestų SQL sakinių vertinimas

Vertinant studento užduotis, svarbus yra ne tik gauto atsakymo įvertinimas ir patikrinimas, tačiau ne ką mažiau svarbu yra įvertinti ir studento sumodeliuotą SQL sakinį. Bikash Chandra, Mathew Joseph, Bharath Radhakrishnan, Shreevidhya Acharya, S. Sudarshan straipsnyje [2] apžvelgia sistemą *XData*, kuri yra automatizuota ir interaktyvi platforma, skirta SQL mokymuisi ir automatiniam SQL užklausų vertinimui.

Ši sistema pasižymi tuo, kad vertina ne tik pilną studento SQL sakinį, tačiau skiria ir dalinius vertinimus pagal pateiktą užklausą. *XData* sistema generuoja kelias specifines duomenų užklausas, pateiktas instruktoriaus, kad tokiu būdu būtų galima užfiksuoti įprastai padaromas klaidas. Tai yra svarbu, norint vykdyti dalinį užklausų vertinimą, kad būtų galima kuo tiksliau įvertinti, ar tikrai sakinys yra bent dalinai teisingas. Straipsnyje yra apibendrinama technika, kaip įvertinti SQL užklausą pagal tai, kaip arti tikimosi sakinio ji yra. Kadangi kiekvieną SQL sakinį galima užrašyti skirtingais būdais, pirmiausia studento ir dėstytojo sakiniai yra susiteminami pagal tam tikras taisykles, kad būtų galima tinkamai palyginti abu sakinius. Susisteminti sakiniai yra skaidomi pakomponenčiui, o suskaidyti komponentai yra lyginami tarpusavyje. Jeigu komponentas sutampa, tai yra vertinama teigiamai, jeigu nesutampa – neigiamai. Galiausiai yra suskaičiuojamas svertinis balas, kuris parodo, kaip arti instruktoriaus sakinio yra įvestas studento sakinys.

Sakinių susisteminimui, kad jie būtų palyginami, yra naudojamos tokios taisyklės: atitikmenų susikirtimas (jeigu prie atributo nėra nurodyta, iš kurios lentelės jis yra paimtas, prie jo yra prirašomas lentelės pavadinimas), WITH sąlygos pašalinimas, BETWEEN predikato pašalinimas (šis predikatas yra pakeičiamas lygiaverte sąlyga, naudojant santykinius operatorius mažiau („<“) ir daugiau („>“)), reliacinių predikatų normalizavimas (išrinkimo sąlygos, naudojančios NOT yra konvertuojamos, kad būtų pašalinamas šis operatorius, o sąlyginiai operatoriai atitinkamai sutvarkomi), sudėtinių sakinių normalizavimas (sudėtinė užklausa, naudojanti IN arba ANY sąlygas yra pakeičiama į sujungimą, naudojant EXISTS), sujungimo sąlygų konvertavimas (bet koks NATURAL INNER JOIN yra pakeičiamas paprastu INNER JOIN su lygiavertėmis sujungimo sąlygomis, naudojant ON jungimo sąlygą). Tvarkant ORDER BY sąlygą yra pašalinami visi atributai, kurie funkciškai nustatomi pagal kitus anksčiau pasirodžiusius atributus ORDER BY sąlygoje. GROUP BY sąlyga nėra pertvarkoma, o yra sutikrinama, ar visi studento ORDER BY sąlygų rinkiniai sutampa su dėstytojo pateiktas. DISTINCT sąlygos yra pašalinamos iš EXISTS/IN/ALL/ANY jungiamųjų užklausų, nes jos neturi dublikatų.

Gautos užklausos po apdorojimo yra palyginamos viena su kita. Prieš lyginimą jos yra suskirstomos į mažesnius komponentus, tokius kaip SELECT sąrašas, sąryšių sąlygos FROM, WHERE sąlygos predikatai ir pan. Kiekvienas atitinkamas studento sakinio komponentas yra lyginamas su dėstytojo atitinkamu komponentu. Už trūkstamus komponentus yra minusuojami balai, už papildomus nereikalingus komponentus taip pat yra minusuojami balai. Tokiu būdu yra įvertinamas kiekvienas komponentas. Kiekvieno komponento minimalus balas yra 0, todėl studento įvestas

sakinys negali būti per daug nubaustas. Kiekvienam instruktoriaus užklausos komponentui yra priskirtas tam tikras svoris W_c. Balai, priskiriami studento užklausai, yra skaičiuojami pagal formulę $\sum_{c \in components} W_c * M_c$, kur W_c ir M_c yra svoris ir balas, priskirtas komponentui. Studentų užklausoje gali būti daugiau komponentų nei dėstytojo užklausoje. Tokie pašaliniai komponentai yra baudžiami, pridedant neigimo ženklą kiekvienam tokiam komponentui.

Rankinis užduočių tikrinimas ir taisymas po egzamimo gali užimti labai daug laiko dėstytojui. Be to, vertinimas gali būti subjektyvus pagal dėstytojo nuotaiką bei gali būti neišvengta klaidų. Dėl to yra reikalingas automatizuotas užduočių tikrinimas. Vienas iš metodų – tekstų panašumo metrikos naudojimas vertinant SQL sakinius – yra pristatomas Ivan Stajduhas ir Goran Mausa straipsnyje [18]. Šis metodas naudoja kelias paprastas tekstų panašumo metrikas, kurių pagalba yra lyginami studento įvesti SQL sakiniai su dėstytojo pateiktais teisingais SQL sakiniais.

Straipsnio autoriai sukūrė SQL egzaminavimo sistemą, kuri leidžia vertinti studentų supratimą apie SQL labiau suprantamu ir sąžiningu būdu, nei atliekant egzaminą ant popieriaus. Viena iš tokios sistemos naudų yra ta, kad studentai įgyja galimybę patikrinti savo pažangą bei teisingai suprasti sintaksės klaidas, o tai yra pageidautina mokymosi procese. Sistemoje yra automatiškai palyginami studento sakinys ir teisingas užduoties atsakymas. Kad būtų įsitikinta metodikos tikslumu, buvo atliekamas testavimas. Semestro metu buvo surinkti 393 užduočių atsakymai ir studentų, besimokančių duomenų bazių kurse, visi sakiniai SELECT tipo. Kiekvienas atsakymas buvo įvertintas sistemos naudojamo algoritmo, taikant griežtą vertinimą, ir rankiniu būdu įvertinta dėstytojo, taikant palengvintą vertinimą. Jeigu studento įvestas atsakymas duodavo tokį patį vertinimo santykį kaip ir teisingu laikomas sakinys, užduotis iškart vertinama kaip visiškai teisinga. Ši lygybė buvo automatiškai palyginta ir pranešama sistemos, naudojant išraišką $(R_S \cup R_R) - (R_S \cap R_R)$, kur R_S reiškia santykį, kurį grąžina DBVS, praleidus studento įvestą sakinį, o R_R reiškia santykį, grąžintą DBVS, praleidus dėstytojo įvestą atsakymą. Jeigu išraiška grąžino tuščią ryšį, galima manyti, kad studento ir dėstytojo atsakymai sutampa. Vėliau rankiniu būdu buvo peržiūrėti atsakymų įvertinimai, siekiant sušvelninti vertinimą ir aptikti galimus plagiatus. Apdorojant ir paruošiant duomenis sakinių palyginimui, kai kurie iš šių konstruktorių buvo pašalinti iš sakinių, siekiant pasiekti didesnį panašumą. Kiekviename SQL sakinyje tokie simboliai, kaip „„“, „;“, „(“, „)“, „=“, „<“, „>“, „!“, „+“, „-“, „*“ yra praleidžiami, naujos eilutės simbolis pakeičiamas paprastu tarpu, visi pabaigos tarpai pašalinami. Taip pat, visi lentelių pavadinimai prieš stulpelio pavadinimą taip pat pašalinami.

Kuriant tikslų prognozavimo modelį yra reikalinga sukurti daugybę įvairių metrikų, skirtų įvertinti sakinių panašumui. Taikant straipsnyje aprašomą metodą buvo naudojamos tokios metrikos (S_S reiškia studento sakinį, S_R reiškia dėstytojo sakinį): normalizuotas absoliutaus ilgio skirtumas (ALD - absoliutus ilgio skirtumas tarp abiejų sakinių, padalintas iš dėstytojo sakinio ilgio $ALD = \frac{|length(S_R) - length(S_S)|}{length(S_R)}$); Normalizuotas Levenšteino atstumas simboliui (CLD - mažiausias simbolių redagavimo skaičius, norint pakeisti vieną tekstą kitu, padalintas iš dėstytojo sakinio ilgio $CLD = \frac{LevDist(S_R, S_S)}{length(S_R)}$); Normalizuotas Levenšteino atstumas žodžiui (WLD - mažiausias žodžių redagavimo skaičius, norint pakeisti vieną eilutę kita, padalintas iš dėstytojo sakinio žodžių kiekio $WLD = \frac{wordLevDist(S_R, S_S)}{noWords(S_R)}$); Euklidinis dažnis žodžių atstumui (EWFD - atstumas tarp žodžio dažnio vektoriaus taškų Euklidinėje erdvėje, atsižvelgiant į unikalius žodžius dėstytojo sakinyje $EWFD = \sqrt{\sum_i (f_i(S_R) - f_i(S_S))^2}$). Skaičiavimuose buvo naudojamos tik normalizuotos reikšmės pagal atitinkamą vertę, nes eksperimentiniai skaičiavimai parodė, kad normalizuotas reikšmes buvo lengviau pritaikyti nei nenormalizuotas.

Studento įvestų SQL sakinių automatiniam vertinimui taip pat yra sukurtas įrankis *aSQLg*, kuris yra apžvelgiamas Carsten Kleiner, Christopher Tebbe ir Felix Heine straipsnyje [8]. Šis įrankis gali būti naudojamas kaip mokymo priemonė studentų rezultatų kokybei gerinti ir taip pat gali būti naudojamas gerinti užduočių atlikimo efektyvumui. Vertinimo procesas vyksta tokiais etapais: visų SQL sakinių įkėlimas į sistemą, sakinio patikrinimas dėl draudžiamų elementų, patikrinimas ar sakinys yra lygus pamatiniam sprendimui, patikrinamas sintaksės teisingumas, patikrinama sakinio kaina, patikrinamas rezultatų teisingumas, patikrinamas sakinio stilius, po visų sakinių apdorojimo generuojamos ataskaitos, jeigu reikia, papildomai įvertinama rankiniu būdu. Visų elementų sakinyje gavimui yra naudojamas analizatorius *JSqlParser*, kurio pagalba iš elementų yra suformuojamas medis. Jeigu studento sakinys sutampa su teisingu sprendimu, studentui yra skiriamas maksimumas balų ir sakinys daugiau nėra analizuojamas. Jeigu sakiniai nesutampa pilnai, yra vykdomi tolimesni sakinio analizės veiksmai. Toliau yra tikrinamas sintaksės teisingumas. Jeigu sakinys yra sintaksiškai teisingas, tuomet studentui yra skiriami balai už sintaksės teisingumą. Po sintaksės patikrinimo yra tikrinamas efektyvumas, kuris priklauso nuo to, kaip arti yra studento sakinio ir teisingo sakinio kaina. Jeigu dėl tam tikrų priežasčių sistema negali automatiškai įvertinti sakinio, jis yra paliekamas vertinimui rankiniu būdu. Kiekvienas sakinys po automatinio įvertinimo taip pat gali būti įvertintas ir rankiniu būdu. Semantinė sakinių analizė įrankyje nėra naudojama, tačiau tai būtų naudinga, nes būtų galima skirti dalinius vertinimus beveik teisingiems SQL sakiniams.

2. Edukacinis modelis ir algoritmai

Viena svarbiausių funkcijų, norint palengvinti ir pagreitinti DBVS mokymosi procesą – automatinis SQL sakinių ir gautų atsakymų palyginimas ir vertinimas. Sukurti algoritmai skirti automatizuotam įvestų SQL sakinių, ar jų rinkinių, ir pagal šiuos sakinius gautų rezultatų palyginimui.

2.1. Skirtingų tipų sakinių struktūros

SQL sakiniams yra būdinga tam tikra sintaksė. Šiame darbe analizuojami SELECT, CREATE, INSERT, UPDATE, DELETE tipų sakiniai. Algoritmuose naudojama SELECT tipo sakinio funkcija yra pateikta 1 pav., CREATE tipo sakinio funkcija pateikta 2 pav., INSERT tipo sakinio funkcija 3 pav., UPDATE tipo sakinio funkcija 4 pav., DELETE tipo sakinio funkcija 5 pav.

Laužtiniuose skliaustuose („[“, „]“) pateiktos SQL sakinio dalys nėra privalomos, riestiniuose skliaustuose („{“, „}“) pateiktos dalys turi būti naudojamos privalomai, jeigu naudojamas atitinkamas pagrindinis komponentas, statmenu brūkšniu („|“) atskirtos sakinio dalys yra naudojamos pasirinktinai, tačiau ne visos kartu.

Skirtingų tipų sakiniai yra naudojami skirtingoms operacijoms atlikti. SELECT tipo sakinys skirtas jau egzistuojančių įrašų gavimui, filtravimui. CREATE tipo sakinys skirtas naujų lentelių duomenų bazėje kūrimui. INSERT tipo sakinys yra skirtas naujų įrašų įterpimui į jau egzistuojančias lenteles. UPDATE tipo sakinys skirtas jau egzistuojančių įrašų koregavimui. DELETE tipo sakinys skirtas jau egzistuojančių įrašų šalinimui. SELECT tipo sakinys taip pat gali būti naudojamas ir kai kurių kitų tipų sakiniuose. Šio tipo sakinį galima integruoti į INSERT, UPDATE ir DELETE tipo sakinius ir taip filtruoti reikiamus duomenis.

Kiekvieno tipo sakinys turi požymių (angl. *tokens*) aibę. Kiekvienas sakinio žodis, simbolis ar ženklas turi tam tikrus apibendrintus požymius, pavyzdžiu, žodžio SELECT požymis yra

TOKEN_SELECT, lentelės pavadinimo požymis yra TOKEN_ID ir kt.

```
1 SELECT [ TOP duomenu_kiekis ] renkamas_sarasas [ INTO nauja_lentele ]
2 [ FROM lenteles_pavadinimas
3 [ { LEFT | RIGHT | FULL } [ INNER | OUTER ] JOIN lenteles_pavadinimas ON
      jungimo_salyga] ]
4 [ WHERE ieskoma_salyga ]
5 [ GROUP BY grupavimo_issireiskimas [ HAVING ieskoma_salyga ] ]
6 [ ORDER BY rusiavimo_issireiskimas [ ASC | DESC ] ]
```

1 pav. SELECT tipo sakinio sintaksė

```
1 CREATE TABLE
2 [ duomenu_bazes_pavadinimas . [ schemos_pavadinimas ] . |
    schemos_pavadinimas . ] lenteles_pavadinimas
3 ( { <stulpeliu_aprasymai>
4 | [ <lenteles_apribojimai> ] [ ,... n ]
5 | [ <lenteles_indeksai> ] [ ,... n ] } )
```

2 pav. CREATE tipo sakinio sintaksė

```
1 INSERT INTO [ duomenu_bazes_pavadinimas . [ schemos_pavadinimas ] . |
    schemos_pavadinimas . ] lenteles pavadinimas
2 [ ( stulpelio_pavadinimas [ ,...n ] ) ]
3 {
4 VALUES ( { NULL | israiska } )
5 | SELECT <isrinkimo_kriterijai>
6 }
```

3 pav. INSERT tipo sakinio sintaksė

```
1 UPDATE [ duomenu_bazes_pavadinimas . [ shcemos_pavadinimas ] . |
    schemos_pavadinimas . ] lenteles_pavadinimas
2 SET { stuleplio_pavadinimas = { israiska | NULL } } [ ,...n ]
3 [ FROM salyga ]
4 [ WHERE <isrinkimo_kriterijai> ]
```

4 pav. UPDATE tipo sakinio sintaksė

Gauti rezultatai pagal SQL sakinį skiriasi pagal sakinio tipą. SELECT tipo sakinių rezultatai yra gaunami kaip virtuali lentelė. UPDATE, DELETE ir INSERT tipo sakinių rezultatai yra fiksuojami kaip lentelių duomenų pasikeitimai (eilučių pasikeitimai ar eilučių kiekio skirtumas). CREATE tipo sakinių rezultatai yra gaunami sisteminėse lentelėse, pagal kurias galima nustatyti, ar buvo sukurtos lentelės ir kokios jos buvo sukurtos.

```
1 DELETE FROM [duomenu_bazes_pavadinimas . [ schemos_pavadinimas ] . |
      schemos_pavadinimas. ] lenteles_pavadinimas
2 [ WHERE <paieskos_salygos> ]
```

5 pav. DELETE tipo sakinio sintaksė

Dėl šių priežasčių, vertinant SQL sakinius pagal sintaksę, galima atsižvelgti į pačius sakinius vizualiai pagal sakinio tipą ir struktūrą. Vertinant gautus rezultatus reikia atsižvelgti į sakinio tipą ir pagal tai vertinti skirtingų tipų rezultatus – gaunamą virtualią lentelę, pasikeitimus pagrindinėse duomenų bazės lentelėse arba pasikeitimus sisteminėse duomenų bazės lentelėse.

2.2. Algoritmų veikimo modelis

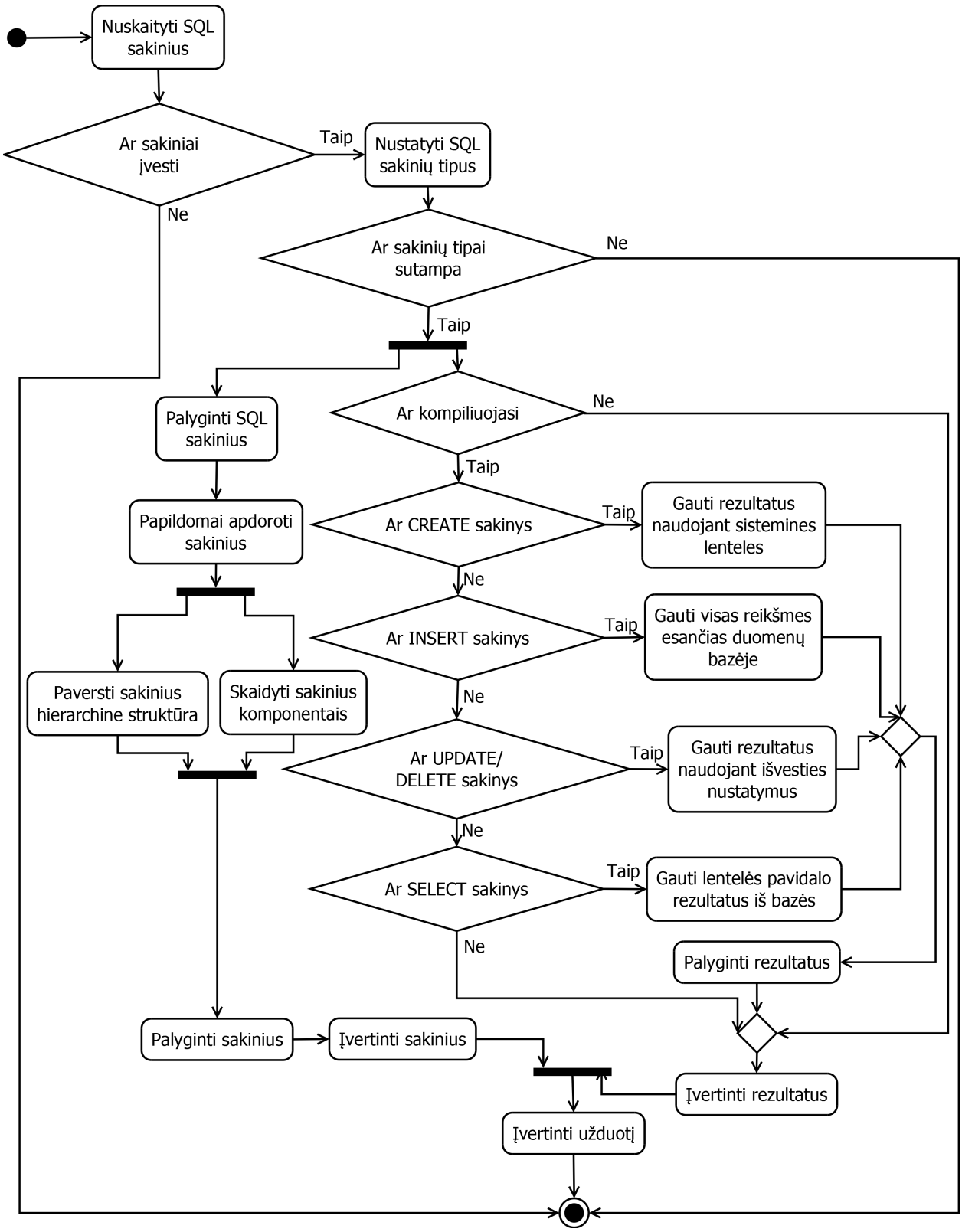
Bendra vertinimo algoritmų schema UML veiklos diagramos pavidalu pateikta 6 pav. Paveikslėlyje pavaizduotas bendras vertinimo procesas. Nuskaičius sakinius pirmiausia yra tikrinama, ar buvo įvesti SQL sakiniai. Jeigu bent vienas iš reikalingų sakinių buvo neįvestas, daugiau veiksmų nėra atliekama. Jeigu visi reikiami sakiniai yra įvesti, tuomet yra tikrinami vertinamų sakinių tipai. Jeigu tipai nesutampa – daugiau veiksmų nėra atliekama. Jeigu sakinių tipai sutampa, tuomet lygiagrečiai yra vykdomas SQL sakinių vertinimas ir gauto atsakymo pagal įvestą SQL sakinį vertinimas.

Lyginant SQL sakinius, pirmiausia sakiniai yra papildomai pertvarkomi, apdorojami ir kaip įmanoma labiau suvienodinami pagal iš anksto apibrėžtas taisykles, vaizduojamas 1 lentelėje. Suvienodinus sakinius, toliau lygiagrečiai vyksta du veiksmai – sakiniai yra verčiami į hierarchinę struktūrą ir sakiniai yra skaidomi mažesniais komponentais. Atlikus šiuos veiksmus, vykdomas SQL sakinių palyginimas. Atitinkamai yra lyginami sakiniai, paversti į hierarchinę struktūrą ir sakiniai, išskaidyti į mažesnius komponentus. Atlikus sakinių palyginimus, yra priskiriamas įvertinimas už įvestą sakinį.

Lyginant atsakymus pagal įvestą SQL sakinį, pirmiausia yra tikrinama, ar sakinys yra sintaksiškai teisingas ir kompiliuojasi. Jeigu sakinys nesikompiliuoja – už atsakymą iškart priskiriamas nulinis vertinimas. Jeigu sakinys kompiliuojasi, tuomet yra tikrinama, ar sakinys yra CREATE tipo. Jeigu sakinys yra CREATE tipo, tokiu atveju rezultatas yra gaunamas, naudojant sistemines lenteles. Jeigu sakinys nėra CREATE, yra tikrinama, ar tai INSERT tipo sakinys. Jeigu INSERT, tuomet yra gaunamos visos duomenų bazėje esančių lentelių reikšmės. Jeigu sakinys nėra INSERT, tikrinama, ar tai UPDATE arba DELETE tipo sakinys. Jeigu UPDATE arba DELETE, tuomet yra gaunamas rezultatas naudojant išvesties parametrus. Jeigu tai nėra nei UPDATE, nei DELETE tipo sakinys, tuomet yra tikrinama, ar tai SELECT tipo sakinys. Jeigu SELECT tipo, tuomet yra gaunamas paprastas lentelės pavidalo atsakymas iš duomenų bazės. Jeigu tai nėra SELECT, tuomet kaip galutinis įvertinimo rezultatas iškart priskiriamas nulinis rezultatas. Gauti rezultatai yra lyginami tarpusavyje tarp įvesto sakinio vertinimui ir įvesto tikimosi sakinio rezultatų. Palyginus rezultatus yra priskiriamas įvertinimas už gautą atsakymą pagal įvestą sakinį.

Galiausiai, suskaičiavus įvertinimus už įvestą SQL sakinį ir gautą pagal jį atsakymą yra, priskiriamas galutinis balas už užduotį. Galutinis balas gaunamas atsižvelgiant į abu vertinimus, skaičiuojamas abiejų prieš tai gautų vertinimų aritmetinis vidurkis.

Tolimesniuose šio skyriaus poskyriuose plačiau nagrinėsime įvertinimo algoritmus. Abu algoritmai ir su jais atliekami veiksmai bus nagrinėjami atskirai.



6 pav. Vertinimo algoritmų schema UML veiklos diagramos pavidalu

2.3. Algoritmai ir modelyje naudojami apibrėžimai

2.3.1. SQL sakinių palyginimas

SQL sakinių vertinimo veikimo principas – sakinys yra vertinamas dviem būdais. Vienas naudojamas būdas – naudojamas analizatorius, kuris paverčia sakinį į hierarchinę strukūrą, kitas nau-

dojamas būdas – SQL sakinio skaidymas mažesniais komponentais. SQL sakinių palyginimo algoritme yra naudojami tam tikri apibrėžimai ir kintamieji.

1 apibrėžimas. SQL sakinių tipai. SQL sakiniai priklauso aibei *Tipai*, kur kiekvienas tipas $t \in$ *Tipai*. Toliau darbe yra naudojami kintamieji *TikimasisSakinioTipas* ir *VertinamasSakinioTipas*. *TikimasisSakinioTipas* ir *VertinamasSakinioTipai* yra eilutės tipo kintamieji, kurie nusako kokio tipo *t* yra SQL sakiniai.

2 apibrėžimas. Sakiniai. SQL sakiniai yra apibrėžiami kaip *sakinys*, kuris priklauso tipui *t*. SQL sakiniai priklauso aibei *Sakiniai*. Toliau darbe kaip *sakinys* yra naudojami kintamieji *TikimasisSakinys* ir *VertinamasSakinys*. *TikimasisSakinys* yra eilutės tipo kintamasis, kuris yra naudojamas kaip teisingas sakinys, pagal kurį yra tikrinami kiti sakiniai. *VertinamasSakinys* yra eilutės tipo kintamasis, kuris yra naudojamas įvertinimui.

3 apibrėžimas. Įvertinimai. Algoritme yra naudojami įvertinimai, kurie apibrėžiami kaip *įvertinimas*. Toliau darbe yra naudojami kintamieji *MaxĮvertinimas*, *AutoAtsĮvertinimas*, *HSĮvertinimas*, *SakinioĮvertinimas*. *MaxĮvertinimas* yra skaičiaus su kableliu formato kintamasis, kurį į sistemą įveda užduoties kūrėjas. *MaxĮvertinimas* nusako, koks galimas didžiausias įvertinimas už užduotį. *AutoAtsĮvertinimas* yra skaičiaus su kableliu formato kintamasis, kuris nurodo, koks buvo skirtas įvertinimas už įvestą SQL sakinį. Šis skaičius svyruoja intervale *[0; MaxĮvertinimas]*. *HSĮvertinimas* yra skaičiaus su kableliu formato kintamasis, kuris yra skirtas SQL sakinio *sakinys* įvertinimui pagal hierarchinę struktūrą. *HSĮvertinimas* svyruoja intervale *[0; MaxĮvertinimas]*. *SakinioĮvertinimas* yra skaičiaus su kableliu formato kintamasis, kuris yra skirtas SQL sakinio *VertinamasSakinys* įvertinimui, išskirsčius jį mažesniais komponentais. *SakinioĮvertinimas* svyruoja intervale *[0; MaxĮvertinimas]*.

4 apibrėžimas. Hierarchinės struktūros. Hierarchinės struktūros yra pibrėžiamos kaip *HS*. Darbe yra naudojami kintamieji *TikimasisSakinysHS* ir *VertinamasSakinysHS*, kurie priklauso hierarchinėms struktūroms *HS*. Šios hierarchinės struktūros yra atitinkamai gaunamos pagal SQL sakinius *sakinys*. Taip pat darbe yra naudojama hierarchinė struktūra, apibrėžiama kintamuoju *AtsHS*, kuri yra gaunama palyginus dvi hierarchines struktūras *HS*.

5 apibrėžimas. Požymiai. Kiekvienas galimas sakinio dalies požymis *p* priklauso požymių aibei *Tokens*. Kiekvienas sakinio pagrindinių komponentų požymis *požymis* priklauso aibei *Pozymiai*. Požymių aibė *Tokens* yra generuojama analizatoriaus (angl. *parser*), o požymių aibė *Požymiai* yra generuojama rankiniu būdu pagal sakinio tipą *t*.

6 apibrėžimas. SQL sakinių komponentai. Kintamieji *TikimasisSakinysKomponentai* ir *VertinamasSakinysKomponentai* yra dvimačiai masyvai, kuriuose saugomi eilutės tipo duomenys. Masyvuose duomenys yra saugomi išskirstyti pagal mažesnius komponentus, kurie apibrėžiami pagal sakinio dalies požymius *p*. Kiekviena nauja masyvo eilutė prasideda pagrindiniu sakinio komponentu, kuris priklauso požymių aibei *Požymiai*.

7 apibrėžimas. Pertvarkymo taisyklės. Pertvarkymo taisyklės *ts* apibrėžiamos kaip poros *(before, after)* $\in X$, kur X = *{ "<>", "<=", ">=", "!=", "<", ">", "< 1 +", "> -1 +", "BETWEEN", "NOT", "IN" }*.

8 apibrėžimas. Funkcija gautiSakinioTipą(sakinys). Funkcija *gautiSakinioTipą(sakinys)* gauna sakinį *sakinys* iš sakinių aibės *Sakiniai*. Funkcijoje yra nustatoma, kokio SQL sakinio tipo *t* sakinys yra, ir grąžina sakinio tipą *t*.

9 apibrėžimas. Funkcija sakinioPertvarkymas(sakinys). Funkcija gauna sakinį *sakinys* iš sakinių aibės *Sakiniai*. Funkcija pagal iš anksto pateiktas taisykles *ts* pertvarko pateiktą SQL sakinį *sakinys*. Funkcija grąžina pagal nustatytas taisykles pertvarkytą sakinį.

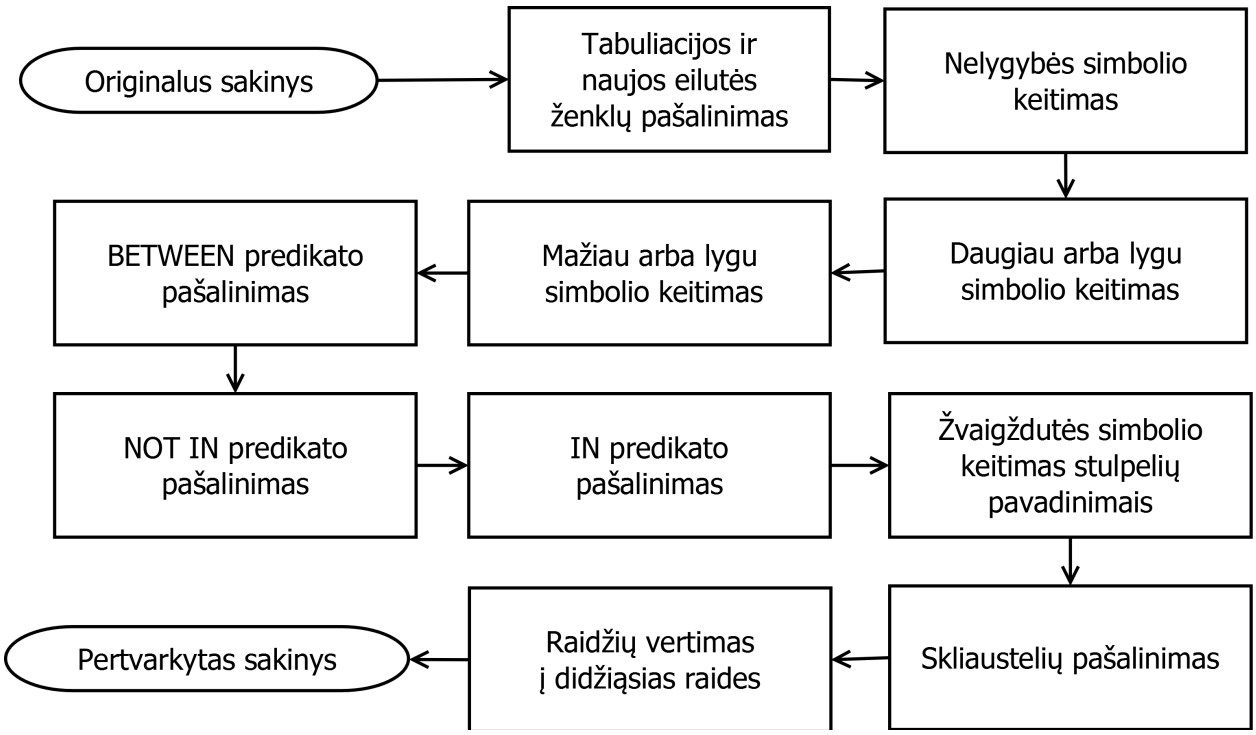
Prieš vertinant SQL sakinius, kiekvienas sakinys, tiek tikimasis, tiek vertinamas, yra apdorojamas pagal tam tikras taisykles, kad jie būtų kaip įmanoma panašesni. Sakinių pertvarkymas yra vykdomas algoritmo pradžioje, kaip pateikta Algoritmo 1 eilutėse 4 ir 5. Apdorojimo taisyklės yra pateiktos 1 lentelėje. Pirmiausia yra sutvarkomi santykiniai ir palyginimo operatoriai bei prie jų esančios reikšmės. Reikšmė „<>“ yra keičiama į reikšmę „!=“. Reikšmės „<=“ ir „>=“ atitinkamai yra keičiamos į „<“ ir „>“, o po šių operatorių atitinkamai reikšmei yra pridedamas vienetas arba atimamas. Tabuliacijos ir naujos eilutės ženklai yra pakeičiami paprastais tarpo simboliais. Sakinyje esantys predikatai BETWEEN yra pašalinami ir pakeičiami santykiniais operatoriais daugiau („>“) ir mažiau („<“). Taip pat yra pašalinami IN ir pakeičiami į lygybės operatorius („=“), atitinkamai NOT IN yra pakeičiami į nelygybės operatorių („!=“). SELECT tipo sakiniuose atrenkant visas lentelės reikšmes žvaigždutės („*“) simbolis yra pakeičiamas visų lentelės stulpelių pavadinimais. Prieš vertinant taip yra pašalinami laukų ir lentelių pervadinimai kartu su pervadinimo žodžiu AS. Taip pat yra pašalinami duomenų bazių ir schemų pavadinimai kartu su taško („.“) simboliu prieš stulpelio pavadinimus. Kai sakinys yra vertinamas naudojant skirstymą pakomponenčiui, taip pat yra pašalinami sakinyje esantys kablelio („,“) ir skliaustelių („(“, „)“) simboliai, nes jie nėra esminiai vertinant įvestą SQL sakinį.

Prieš apdorojimą	**Po apdorojimo**
x <> y	x != y
x <= y	x < 1 + y
x >= y	x > -1 + y
x BETWEEN y AND z	x > y AND x < z
x IN (y, z)	x = y OR x = z
x NOT IN (y, z)	x != y OR x != z

1 lentelė. Taisyklės sakinių pertvarkymui

Sakinių pertvarkymas atliekamas pagal taisykles paeiliui. Paeiliui einančios operacijos pateiktos 7 pav. Kaip matome pagal pateiktą paveikslėlį, pirmiausia SQL sakiniai yra sukeliami į vieną eilutę, po to yra apdorojami santykiniai operatoriai ir palyginimo operatoriai, sakinyje esantys predikatais. Pertvarkius operatorius, žvaigždutės simbolis yra pakeičiamas stulpelių pavadinimais, skliausteliai yra pašalinami ir sakinys yra paverčiamas į didžiąsias raides.

Sakinių pertvarkymo pavyzdys yra pateiktas 8 pav. ir 9 pav. Dėl patogesnio analizavimo ir vaizdavimo, pavyzdžiuose tabuliacijos ir naujos eilutės ženklai nėra pašalinti. 8 pav. yra pateiktas pradinis, neapdorotas SQL sakinys, o 9 pav. yra pateiktas sakinys po pirminio apdorojimo. Šiuo atveju iš aukščiau išvardintų taisyklių buvo pritaikytos kelios – duomenų bazių ir schemų pavadinimų pašalinimo pakeitimai, kuriuos galima matyti 1-oje, 2-oje, 3-oje, 5-oje ir 6-oje eilutėse, laukų ir lentelių pervadinimo pašalimo pakeitimai, kuriuos galima matyti 2-oje ir 5-oje eilutėse, žvaigždutės simbolio pakeitimas stulpelių pavadinimais pakeitimas, kuris matomas 4-oje eilutėje. Taip pat, kad būtų išvengta klaidų tolimesniame vertinime, visos sakinyje esančios raidės yra pakeičiamos į didžiąsias raides.



7 pav. Sakinių pertvarkymo etapai

```
1 select country.name, a.name, a.introDate
2 from country, country_currency as aa, currency as a
3 where country.cid = aa.countryid and aa.currencyid = a.cid and dateTo is
      NULL and a.name not like 'euro' and notexists (
4 select *
5 from currency as b, country_currency as c
6 where c.currencyid = b.cid and c.dateTo is NULL and b.introDate>a.
     introDate and b.name not like 'euro')
```

8 pav. Pradinis sakinys

```
1 SELECT NAME, NAME, INTRODATE
2 FROM COUNTRY, COUNTRY_CURRENCY, CURRENCY
3 WHERE CID = COUNTRYID AND CURRENCYID = CID AND DATETO IS NULL AND NAME
      NOT LIKE 'EURO' AND NOT EXISTS (
4 SELECT CID, NAME, CODE, INTRODATE
5 FROM CURRENCY, COUNTRY_CURRENCY
6 WHERE CURRENCYID = CID AND DATETO IS NULL AND INTRODATE > INTRODATE AND
      NAME NOT LIKE 'EURO')
```

9 pav. Sakinys po pirminio pertvarkymo

10 apibrėžimas. Funkcija gautiHS(sakinys). Tai yra funkcija, skirta SQL sakiniui *sakinys*, priklausančiam sakinių aibei *Sakiniai*, paversti į hierarchinę struktūrą. Funkcijos išvestis yra hierarchinio tipo kintamasis.

11 apibrėžimas. Funkcija HSSkirtumai(HS1, HS2). Tai yra funkcija, skirta nustatyti skirtu-

mams tarp dviejų hierarchinių stuktūrų *HS1* ir *HS2*. Funkcijai yra paduodamos dvi hierarchinės struktūros *HS1* ir *HS2*, kurios yra lyginamos tarpusavyje. Grąžinamas rezultatas yra hierarchinio tipo kintamasis, vaizduojantis skirtumus tarp paduotų funkcijai hierarchinių struktūrų.

12 apibrėžimas. Funkcijos gautiSutapimus(HS) ir gautiNesutapimus(HS). Funkcijos yra skirtos nustatyti sutapimų ir skirtumų kiekiui hierarchinėje struktūroje, kuri vaizduoja skirtumus tarp dviejų hierarchinių struktūrų. Funkcijos įvesties parametras *HS* yra hierarchinio tipo kintamasis, vaizduojantis skirtumus tarp dviejų hierarchinių struktūrų, o funkcijų išvestis yra atitinkamai sutapimų ir nesutapimų tarp SQL sakinių kiekis.

13 apibrėžimas. Funkcija gautiKomponentus(sakinys, požymiai<t>). Funkcija yra skirta SQL sakinio *sakinys* suskaidymui į dvimačius masyvus pagal sakinio dalių požymius *p* ir pagrindinius sakinio požymius *Požymiai*, kurie priklauso nuo sakinio tipo *t*. Funkcijos įvesties parametras *sakinys* nurodo, koks sakinys turi būti išskaidytas komponentais, parametras *požymiai<t>* nurodo, kokia bus naudojama požymių aibė *Požymiai* pagal sakinio tipą *t*.

14 apibrėžimas. Funkcija palygintiAtsakymus(KM1, KM2). Funkcija yra skirta dviejų dvimačių masyvų palyginimui ir sutampančių bei nesutampančių masyvų ląstelių santykių radimui. Įvesties parametrai *KM1* ir *KM2* yra masyvo tipo kintamieji. Funkcijos grąžinamas rezultatas yra santykinis sutampančių masyvų ląstelių kiekis, skaičiuojamas pagal didesnio masyvo bendrą ląstelių kiekį.

Sukurtame algoritme taip pat yra naudojami skaitliukai *sutapimai* ir *nesutapimai*, kurie yra sveikojo skaičiaus tipo kintamieji. Šie skaitliukai nurodo, kiek yra sutapimų ir skirtumų tarp hierarchinių struktūrų. Šie skaičiai yra gaunami, analizuojant hierachinę struktūrą *AtsHS*.

SQL sakinių palyginimo algoritmo veikimo principas – pirmiausia SQL sakiniai bus apdorojami ir kaip įmanoma sintaksiškai suvienodinami. Vienas iš naudojamų variantų tolimesniam vertinimui: sakinio dalys išskirstomos į mažesnius komponentus pagal standartinę sakinio struktūrą, o gauti mažesni komponentai lyginami tarpusavyje, priskiriant dalinius rezultatus. Taip pat į pagalbą yra pasitelkiamas analizatorius (angl. *parser*), verčiantis SQL sakinius į hierarchines struktūras, kurios yra lyginamos tarpusavyje. Algoritmo veikimas paremtas tuo, kad studentas jau turi bent minimalių SQL žinių ir žino, kokia tvarka turi būti išdėstyti SQL sakinio standartiniai komponentai.

Automatiniam SQL sakinių vertinimui yra naudojami du skirtingi metodai. Pirmasis metodas, nepriklausomai nuo to, ar sakinys kompiliuojasi ir yra sintaksiškai teisingas, ar nėra sintaksiškai teisingas, sakinių palyginimui ir vertinimui yra naudojamas analizatorius. Kitas naudojamas metodas – pagal atitinkamo tipo sakinio struktūrą sakinys yra padalinamas į mažesnius komponentus ir vertinimas bei sakinių lyginimas vyksta pakomponenčiui. Kiekvieną kartą įvertinimas yra paskaičiuojamas abiem metodais, ir, palyginus abu gautus rezultatus, kaip galutinis balas yra priskiriamas didesnis gautas skaičius. Taip visi sakiniai yra įvertinami studento naudai.

Algoritmas 1 vaizduoja pseudokodu pateiktą algoritmą, skirtą palyginti įvestiems SQL sakiniams ar sakinių rinkiniams.

Algoritmas 1 Automatinis įvesto sakinio palyginimas ir įvertinimas

---

Įvestis: *TikimasisSakinys, VertinamasSakinys, MaxĮvertinimas*

Išvestis: *AutoAtsĮvertinimas*

1: *TikimasisSakinioTipas* ← *gautiSakinioTipą(TikimasisSakinys)*;

2: *VertinamasSakinioTipas* ← *gautiSakinioTipą(VertinamasSakinys)*;

```
3: if TikimasisSakinioTipas = VertinamasSakinioTipas then
4:    TikimasisSakinys ← sakinioPertvarkymas(TikimasisSakinys);
5:    VertinamasSakinys ← sakinioPertvarkymas(VertinamasSakinys);
6:    TikimasisSakinysHS ← gautiHS(TikimasisSakinys);
7:    VertinamasSakinysHS ← gautiHS(VertinamasSakinys);
8:    AtsHS ← HSSkirtumai(TikimasisSakinysHS, VertinamasSakinysHS);
9:    if AtsHS = ∅ then
10:      HSĮvertinimas ← MaxĮvertinimas;
11:   else
12:      sutapimai ← GautiSutapimus(AtsHS)
13:      nesutapimai ← GautiNesutapimus(AtsHS);
14:      HSĮvertinimas ← (nesutapimai/(sutapimai + nesutapimai)) * MaxĮvertinimas;
15:   end if
16:   TikimasisSakinysKomponentai ← gautiKomponentus(TikimasisSakinys,
      Požymiai<TikimasisSakinioTipas>);
17:   VertinamasSakinysKomponentai ← gautiKomponentus(VertinamasSakinys,
      Požymiai<VertinamasSakinioTipas>);
18:   SakinioĮvertinimas ← palygintiAtsakymus(TikimasisSakinysKomponentai,
      VertinamasSakinysKomponentai, Tokens) * MaxĮvertinimas;
19: end if
20: if HSĮvertinimas > SakinioĮvertinimas then
21:   AutoAtsĮvertinimas ← HSĮvertinimas;
22: else
23:   AutoAtsĮvertinimas ← SakinioĮvertinimas;
24: end if
25: return AutoAtsĮvertinimas
```

Nepriklausomai nuo to, ar sakinys yra sintaksiškai teisingas, ar nėra, algoritmas išanalizuoja ir įvertina sakinį dviem būdais. Pirmiausia sakinys yra analizuojamas naudojant SQL analizatorių. Naudojantis analizatoriumi, SQL sakiniai yra paverčiami į hierarchinę struktūrą. Abi gautos hierarchinės struktūros tipo reikšmės yra lyginamos tarpusavyje, naudojant papildomą biblioteką. Palyginus struktūras yra išvedama galutinė struktūra, kurioje yra vaizduojami sutapimai ir neatitikimai tarp reikšmių.

Užduotis yra vertinama pagal gautą galutinę struktūrą, kurioje matomi sutapimai ir neatitikimai tarp SQL sakinių. Algoritmo pagalba yra suskaičiuojama, kiek buvo rasta sutapimų, kiek įterpimų, pakeitimų ir pašalinimų. Rasti sutapimai, įterpimai, pakeitimai ir pašalinimai yra susumuojami. Skaičiuojant galutinį rezultatą, nesutapimų (įterpimų, pakeitimų, pašalinimų) suma yra dalinama iš bendros visų reikšmių sumos. Algoritmo 1 eilutėse 6 ir 7 yra vykdomas sakinių vertimas į hierarchinę struktūrą, o eilutėse 8-15 yra vykdomas gautų hierarchinių struktūrų palyginimas ir vertinimas.

1 pavyzdys. Vaizduojami originalūs įvesti sakiniai, sakinys išskaidytas į hierarchinę struktūrą bei hierarchinė struktūra po sakinių palyginimo. 1 išeities kodas vaizduoja įvestą tikimąsi SQL sakinį, nepritaikius pirminio apdorojimo, 2 išeities kodas vaizduoja įvestą SQL sakinį vertinimui, taip pat nepritaikius pirminio sakinių apdorojimo. Abu šie sakiniai buvo paversti į hierarchines struktūras, o tos struktūros palygintos viena su kita. Vertinamas sakinys, paverstas į hierachinę

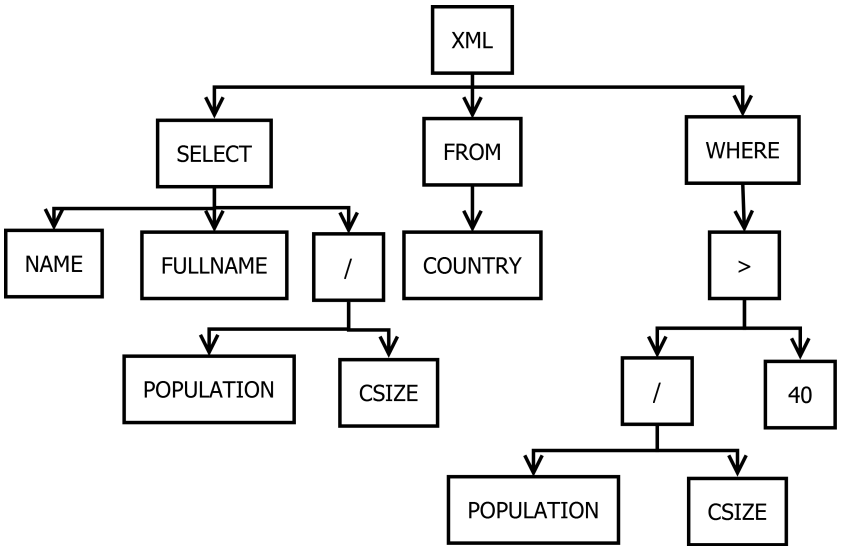
struktūrą, vaizduojamas 10 pav. Pagal pateikta pavyzdį galima matyti, kad sakinys yra skaidomas į pagrindines sakinio dalis, pagrindinės dalys į dar mažesnes ir taip yra gaunama reikalinga hierarchija. Hierarchijų palyginimo rezultatas pateiktas 11 pav. Pagal šį paveikslėlį matoma, kiek buvo sutapimų bei skirtumų tarp lyginamų struktūrų. Pagal gautą rezultatą yra skaičiuojama kiek buvo sutapimų (skaičiuojama skaičių suma, kuri yra šakose, turinčiose įrašą *match*) ir kiek buvo nesutapimų (skaičiuojama kiek yra šakų, turinčių įrašą *add* arba *remove*).

1 išeities kodas. Tikimasis SQL sakinys

```
1 select name, fullname, population/csize
2 from country
3 where population/csize>40 and fullname like '%Kingdom%'
```

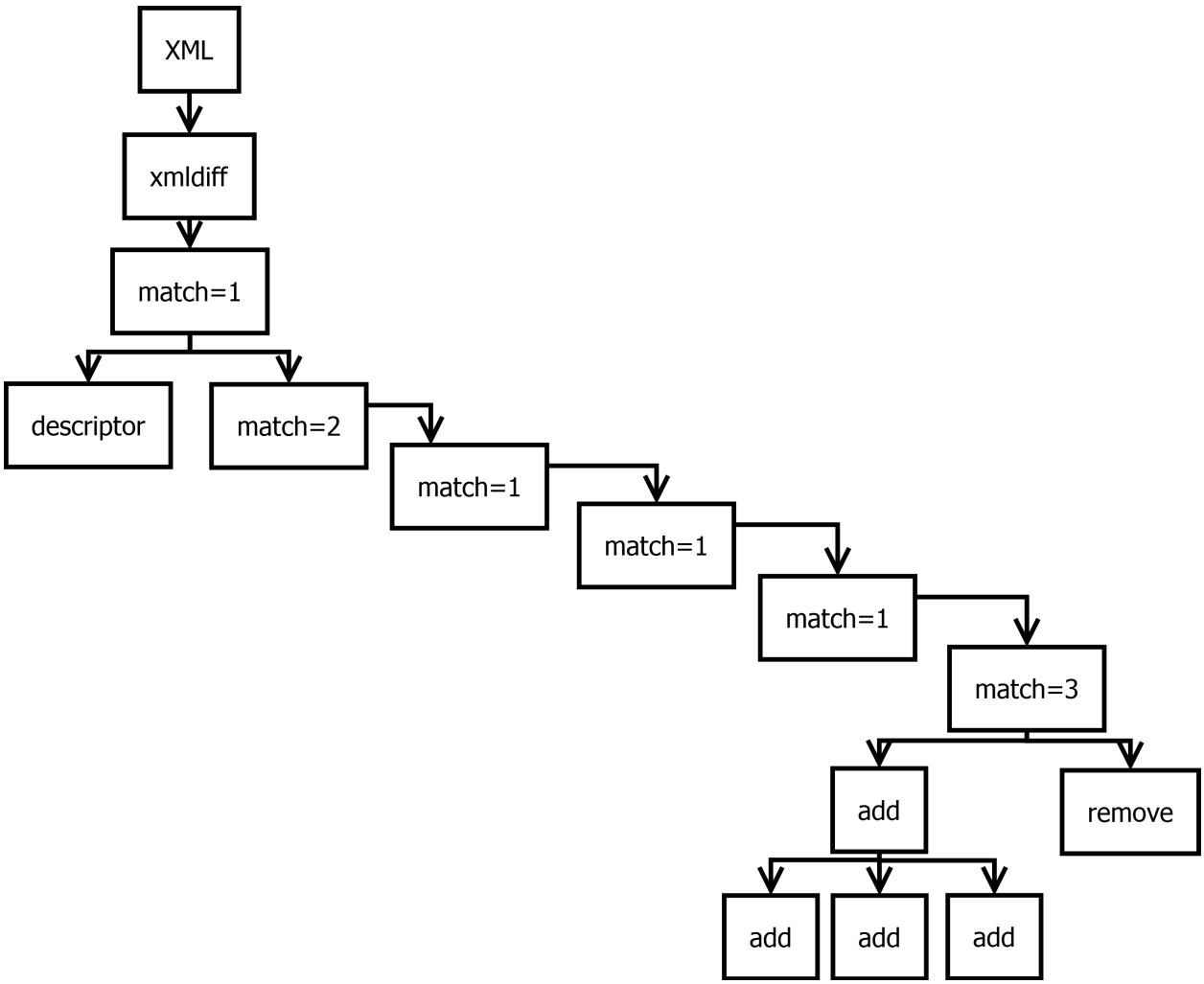
2 išeities kodas. Vertinimui skirtas SQL sakinys

```
1 select name, fullname, population/csize as density
2 from country
3 where  population/csize>40
```



10 pav. Hierarchinė sakinio struktūra

Kitas būdas, naudojamas įvertinti SQL sakinį – skaidymas komponentais. SELECT tipo sakinį, kaip galima matyti aprašytoje struktūroje 1 pav., galima išskirstyti į šešis skirtingus komponentus pagal skirtingas sakinio dalis, tokias kaip SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY. Sakinys yra skirstomas į komponentus, naudojant žymes (angl. *tokens*), kurios yra pateikiamos naudojamo analizatoriaus. Žymės aprašomos taip pat kaip ir komponentai, į kuriuos yra skirstomas sakinys: TOKEN_SELECT, TOKEN_FROM, TOKEN_WHERE, TOKEN_GROUP, TOKEN_HAVING, TOKEN_ORDER. Būtent pagal šias žymes sakinys yra išskirstomas mažesniais sakiniais ir patalpinamas į dvimatį masyvą. Jeigu sakinys yra sudėtinis, jis taip pat imamas ne kaip viena didelė dalis, o kaip mažesni komponentai. Taip pat yra pašalinami visi sakinyje esantys kableliai bei skliausteliai, nes jie nėra esminiai, vertinant sakinio teisingumą. Gauti masyvai yra lyginami pagal eilutę ir eilutėje yra tikrinama kiek yra sutampančių ląstelių, neatsižvelgiant į žymių išdėstymo tvarką.



11 pav. Hierarchinė struktūra po sakinių palyginimo

Algoritmo 1 eilutėse 16 ir 17 yra vykdomas sakinių vertimas į komponentus su žymėmims, o eilutėse 18-19 yra vykdomas gautų komponentų palyginimas ir vertinimas.

2 pavyzdys. Vaizduojami originalūs įvesti sakiniai, kaip įvesti sakiniai atrodo išskaidžius į mažesnius komponentus pagal raktinius žodžius bei kaip atrodo masyvas, palyginus abu masyvus su komponentų išskaidymu. 1 išeities kodas vaizduoja tikimąsi SQL sakinį, nepritaikius pirminio apdorojimo, 2 išeities kodas vaizduoja įvestą SQL sakinį, skirtą vertinimui, taip pat nepritaikius pirminio sakinių apdorojimo. Abu šie sakiniai buvo paversti į dvimačius masyvus, suskaidžius mažesniais komponentais pagal iš anksto aprašytas žymes. 2 lentelė vaizduoja dėstytojo sakinio išskaidymą komponentais, 3 lentelė vaizduoja studento sakinio išskaidymą komponentais, o 4 lentelė vaizduoja masyvą, gautą palyginus abu masyvus tarpusavyje. Abu sakiniai buvo išskaidyti pagal tris raktinius žodžius – SELECT, FROM ir WHERE. Dėl to abiejų sakinių masyvuose užpildyta po tris eilutes. Dėl patogumo WHERE komponentas pavaizduotas ne per vieną eilutę, o per dvi, tačiau tai yra ta pati eilutė. Pagal gautą galutinį masyvų palyginimą matome, kad vienas iš sakinių turi keturiais komponentais daugiau už kitą sakinį.

Taip pat kaip ir SELECT tipo sakinių vertinime, CREATE tipo sakinys taip pat vertinamas dviem būdais, nepriklausomai nuo to, ar jis yra sintaksiškai teisingas ar nėra. Vertinant CREATE tipo sakinius yra vertinama ne po vieną CREATE sakinį, o visas sakinių rinkinys iš karto.

Pirmiausia sakiniai yra analizuojami, naudojant SQL analizatorių. Naudojant jį, abu CREATE

1	TOKEN_ SELECT: SELECT	TOKEN_ ID: NAME	TOKEN_ ID: FULLNAME	TOKEN_ ID: POPULATION	47: /	TOKEN_ ID: CSIZE
2	TOKEN_ FROM: FROM	TOKEN_ ID: COUNTRY				
3	TOKEN_ WHERE: WHERE	TOKEN_ ID: POPULATION	47: /	TOKEN_ ID: COUNTRYID	TOKEN_ ID: CSIZE	62: >
		TOKEN_ INTEGER: 40	TOKEN_ AND: AND	TOKEN_ ID: FULLNAME	TOKEN_ LIKE: LIKE	TOKEN_ STRING: '%KING-DOM%'

2 lentelė. Tikimasis sakinys išskaidytas komponentais

1	TOKEN_ SELECT: SELECT	TOKEN_ ID: NAME	TOKEN_ ID: FULLNAME	TOKEN_ ID: POPULATION	47: /	TOKEN_ ID: CSIZE
2	TOKEN_ FROM: FROM	TOKEN_ ID: COUNTRY				
3	TOKEN_ WHERE: WHERE	TOKEN_ ID: POPULATION	47: /	TOKEN_ ID: COUNTRYID	TOKEN_ ID: CSIZE	62: >
						TOKEN_ INTEGER: 40

3 lentelė. Vertinamas sakinys išskaidytas komponentais

1										
2										
3								TOKEN_ AND: AND	TOKEN_ ID: FULLNAME	TOKEN_ LIKE: LIKE	TOKEN_ STRING: '%KINGDOM%'

4 lentelė. Išskaidytų sakinių palyginimas

sakinių rinkiniai yra verčiami į hierachines struktūras. Abi gautos struktūros yra lyginamos tarpusavyje ir taip yra gaunama nauja hierachinė struktūra, vaizduojanti skirtumus ir sutapimus tarp lyginamų reikšmių.

Kitas vertinimo būdas – sakinio skaidymas mažesniais komponentais pagal raktinius žodžius. CREATE tipo sakinio, kaip galima matyti aprašytoje struktūroje 2 pav., neskirstome į mažesnius komponentus, paliekame tik išskaidymą pagal raktinį žodį CREATE ir ALTER, jeigu jis buvo naudojamas. Sakinių rinkinys yra skirstomas į komponentus, naudojant žymes, kurios taip pat yra pateikiamos analizatoriaus. Žymės aprašomos taip pat kaip ir komponentai, į kuriuos skirstomas

sakinių rinkinys: TOKEN_CREATE, TOKEN_ALTER. Būtent pagal šias žymes sakinių rinkiniai yra išskirstomi mažesniais komponentais ir yra talpinami dvimačiame masyve. Gauti masyvai yra lyginami tarpusavyje ir yra ieškoma sutapimų, neatsižvelgiant į žymių komponento eilutėse išdėstymo tvarką.

INSERT tipo sakinys taip pat yra vertinamas dviem būdais, nepriklausomai nuo to, ar jis kompiliuojasi, ar nesikompiliuoja. Vertinant INSERT tipo sakinius, sakiniai yra vertinami ne po vieną sakinį, o vertinamas visas INSERT sakinių rinkinys.

Pirmiausia sakiniai yra analizuojami, naudojant SQL analizatorių. Naudojant analizatorių, abu INSERT sakinių rinkiniai yra verčiami į hierarchines struktūras, o tos struktūros yra lyginamos tarpusavyje. Kaip galutinis rezultatas yra gaunama hierarchinė struktūra, kuri vaizduoja sutapimus ir nesutapimus tarp dviejų hierarchinių struktūrų ir pagal tai yra apskaičiuojamas studentui skirtas balas už įvestą SQL sakinių rinkinį.

Kitas naudojamas būdas – sakinių rinkinių skirstymas mažesniais komponentais. INSERT tipo sakinio raktinius žodžius galima matyti 3 pav. aprašytoje struktūroje. Konkrečiais algoritme naudojamais atvejais yra naudojamas tik vienas komponentas (TOKEN_INSERT), tačiau pagal bendrą struktūrą pagal skirtingus atvejus reikėtų INSERT tipo sakinį skirstyti bent į du mažesnius komponentus, tokius kaip TOKEN_INSERT ir TOKEN_VALUES, o, taip pat, norint analizuoti sudėtingesnius INSERT tipo sakinius, papildomai reikėtų pridėti ir išskyrimą į SELECT tipo sakinio komponentus, kurie yra vaizduojami 1 pav. Pagal išskirtas žymes sakiniai yra lyginami tarpusavyje ir yra ieškoma sutapimų, neatsižvelgiant į sakinių išsidėstymo tvarką, nes yra vienu metu analizuojamas visas INSERT sakinių rinkinys.

Nepriklausomai nuo to, ar sakinys yra sintaksiškai teisingas, UPDATE tipo sakinys, kaip ir kiti sakiniai, yra vertinamas dviem būdais. Sakiniai yra vertinami ne blokais, kaip CREATE ir INSERT sakiniai, bet po vieną, nes vienas sakinys gali turėti įtaką daugiau nei vienai eilutei iš duomenų bazės lentelės.

Pirmiausia sakinys yra vertinamas naudojant SQL analizatorių. Naudojant analizatorių, abu sakiniai yra verčiami į hierarchines struktūras, o vėliau tos struktūros yra lyginamos tarpusavyje. Galiausiai yra gaunama galutinė hierarchinė struktūra, kuri vaizduoja skirtumus ir sutapimus tarp abiejų SQL sakinių. Skaičiuojant galutinį balą yra analizuojama galutinė hierarchinė struktūra ir pagal ją yra skaičiuojami sutapimai ir nesutapimai.

Kitas sakinio vertinimo būdas yra sakinio skirstymas mažesnėmis sakinio dalimis pagal UPDATE sakinio struktūrą. UPDATE tipo sakinio pagrindinė struktūra yra vaizduojama 4 pav. Algoritme yra naudojami trys UPDATE sakinio tipo komponentai – TOKEN_UPDATE, TOKEN_FROM ir TOKEN_WHERE. Vertinant ir analizuojant sudėtingesnius UPDATE tipo sakinius, taip pat reikėtų pridėti ir kitus galimus komponentus, kurie yra SELECT tipo sakinio struktūros, ir yra vaizduojami 1 pav. Pagal išskirtus komponentus sakiniai yra lyginami tarpusavyje ir taip yra vertinama kiek pakomponenčiui yra sutapimų tarp abiejų lyginamų sakinių.

DELETE tipo SQL sakinys taip pat nėra išskirtinis ir yra vertinamas taip pat kaip ir kiti prieš tai minėti sakiniai. Vertinimui yra naudojami du būdai – pritaikomas SQL analizatorius ir naudojamas sakinio skirstymas mažesniais sakinio komponentais.

Pirmiausia sakinys yra vertinamas pritaikant SQL analizatorių, kurio pagalba vietoj sakinių yra gaunamos hierarchinės struktūros. Abi struktūros yra lyginamos tarpusavyje ir taip yra gaunama nauja hierarchinė struktūra, kuri vaizduoja skirtumus ir sutapimus tarp lyginamų struktūrų. Apskaičiavus sutapimus ir nesutapimus yra skaičiuojamas galutinis balas už įvestą sakinį.

Tie patys sakiniai taip pat yra vertinami pirmiausia išskirsčius sakinį mažesniai komponentais ir lyginant tuos komponentus tarpusavyje. Pagrindinė DELETE tipo sakinio struktūra yra

vaizduojama 5 pav. Pagal šią struktūrą sakinys yra skaidomas į du komponentus pagal žymes – TOKEN_DELETE ir TOKEN_WHERE. Norint vertinti ir analizuoti sudėtingesnius DELETE tipo sakinius, sakinių skirstymui pakomponenčiui taip pat reikėtų naudoti ir SELECT tipo sakinio struktūrą, kuri yra vaizduojama 1 pav. Išskirsčius sakinius komponentais, gauti rezultatai yra lyginami tarpusavyje ir taip yra skaičiuojamas studento balas už įvestą SQL sakinį.

2.3.2. Atsakymų palyginimas

Gautų atsakymų pagal įvestą SQL sakinį veikimo principas – gauti rezultatai pagal įvestą sakinį ar sakinių rinkinį yra verčiami į lentelės pavidalą. Gautos lentelės lyginamos tarpusavyje, ieškant panašumų. Papildomai nuo SQL sakinio tipo priklauso, kaip bus ieškomas ir gaunamas rezultatas, kuris bus vertinamas. Gautų atsakymų pagal SQL sakinius algoritme yra naudojami tam tikri apibrėžimai ir kintamieji.

15 apibrėžimas. Lentelės tipo rezultatai. SQL sakinių gaunami rezultatai yra apibrėžiami kaip *rezultatas* ir jie yra dvimačio masyvo tipo. Toliau darbe kaip *rezultatas* yra naudojami kintamieji *TikimasisRezultatas* ir *VertinamasRezultatas*. Kintamuosiuose *TikimasisRezultatas* ir *VertinamasRezultatas* atitinkamai saugomi gaunami lentelės tipo rezultatai pagal SQL sakinius *TikimasisSakinys* ir *VertinamasSakinys*.

16 apibrėžimas. Funkcija gautiSelectAts(sakinys). Funkcija yra skirta gauti dvimačio masyvo tipo kintamajam pagal įvestą SELECT tipo sakinį *sakinys*. Funkcijoje yra gaunamas rezultatas lentelės tipo pavidalu ir jis konvertuojamas į dvimatį masyvą, kuris yra naudojamas kaip funkcijos išvesties parametras.

Algoritmas 2 vaizduoja pseudokodu pateiktą algoritmo dalį, skirtą palyginti gautus SELECT tipo sakinio atsakymus bei pagal juos gauti įvertinimą.

Algoritmas 2 SELECT tipo sakinio automatinis gauto rezultato palyginimas ir įvertinimas

Įvestis: *TikimasisSakinys, VertinamasSakinys, MaxĮvertinimas*
Išvestis: *AutoAtsĮvertinimas*

1: *TikimasisRezultatas* ← *gautiSelectAts(TikimasisSakinys);*
2: *VertinamasRezultatas* ← *gautiSelectAts(VertinamasSakinys);*
3: *AutoAtsĮvertinimas* ← *palygintiAtsakymus(TikimasisRezultatas, VertinamasRezultatas)* * *MaxĮvertinimas;*
4: **return** *AutoAtsĮvertinimas*

Jeigu įvestas SELECT tipo sakinys yra sintaksiškai teisingas ir kompiliuojamas be klaidų, pagal sakinį galima gauti lentelės pavidalo atsakymą. Vertinant atsakymą pagal įvestą SELECT tipo sakinį būtent ir yra vertinami gauti lentelės pavidalo atsakymai.

Algoritmo 2 1-3 eilutėse yra pateikta algoritmo dalis, kaip vyksta rezultato vertinimas, jeigu sakinio tipas yra SELECT. Detalus gautų atsakymų palyginimo algoritmo veikimas:

1. Tikimosi sakinio ir vertinamo sakinio gauti atsakymai yra perkeliami į skirtingus dvimačius masyvus;

2. Tarpusavyje lyginami du masyvai – atsakymo pagal tikimąsi sakinį ir atsakymo pagal vertinamą sakinį. Algoritme vienu metu yra sukami keli ciklai ir atliekami veiksmai:

   (a) sukamas ciklas atsakymo pagal tikimąsi sakinį dvimačiame masyve;

(b) sukamas ciklas atsakymo pagal vertinamą sakinį dvimačiame masyve;

(c) sukant ciklus masyvuose yra skaičiuojama, kiek jame esančių ląstelių sutapo.

3. Suskaičiuoti įrašų sutapimai toliau yra naudojami įvertinimui skaičiuoti.

3 pavyzdys. Vaizduojami gauti atsakymai pagal įvestus tikimąsi ir vertinamą SQL SELECT tipo sakinius ir rezultatų nesutapimų lentelė po rezultatų palyginimo. 5 lentelė vaizduoja gautą atsakymą pagal tikimąsi sakinį, 6 lentelė vaizduoja gautą atsakymą pagal vertinamą sakinį, o 7 lentelė vaizduoja rezultatą po atsakymų palyginimų. Pagal gautus rezultatus matome, kad gautame atsakyme, skirtame vertinimui, buvo daugiau eilučių nei gautame atsakyme, kurio tikimasi. Dėl šios priežasties gautas rezultatas nebus įvertintas galimu maksimaliu balu.

NAME	**FULLNAME**	**Column1**
Belgium	Kingdom of Belgium	367
United Kingdom	United Kingdom of Great Britain and Northern Ireland	187

5 lentelė. Gautas atsakymas pagal tikimąsi sakinį

NAME	**FULLNAME**	**Column1**
Lithuania	Republic of Lithuania	45
Belgium	Kingdom of Belgium	367
United Kingdom	United Kingdom of Great Britain and Northern Ireland	187
Ireland	Republic of Ireland	65

6 lentelė. Gautas atsakymas pagal vertinamą sakinį

NAME	**FULLNAME**	**Column1**
Lithuania	Republic of Lithuania	45
Ireland	Republic of Ireland	65

7 lentelė. Rezultatas po atsakymų palyginimo

17 apibrėžimas. Atributų vertinimas. Algoritmuose naudojamas kintamasis *AtributųVertinimas* yra skaičiaus su kableliu formato kintamasis, skirtas saugoti rezultatui už sukurtus atributus, naudojant CREATE tipo sakinius, duomenų bazėje.

18 apibrėžimas. Pirminių raktų vertinimas. Algoritmuose naudojamas kintamasis *PirminiųRaktųVertinimas* yra skaičiaus su kableliu formato kintamasis, skirtas saugoti rezultatus už sukurtus pirminius raktus, naudojant CREATE tipo sakinius, duomenų bazėje.

19 apibrėžimas. Išorinių raktų vertinimas. Algoritmuose naudojamas kintamasis *IšoriniųRaktųVertinimas* yra skaičiaus su kableliu formato kintamasis, skirtas saugoti rezultatus už sukurtus išorinius raktus, naudojant CREATE tipo sakinius, duomenų bazėje.

20 apibrėžimas. Patikrinimo sąlygų vertinimas. Algoritmuose naudojamas kintamasis *ChecksVertinimas* yra skaičiaus su kableliu formato kintamasis, skirtas saugoti rezultatus už sukurtas patikrinimo sąlygas, naudojant CREATE tipo sakinius, duomenų bazėje.

21 apibrėžimas. Apribojimų vertinimas. Algoritmuose naudojamas *ApribojimųVertinimas* yra skaičiaus su kableliu formato kintamasis, skirtas saugoti rezultatus už sukurtas apribojimo sąlygas, pagal CREATE tipo sakinius, duomenų bazėje.

22 apibrėžimas. Funkcija gautiAtributus(DB). Funkcija yra skirta gauti visiems atributams, atributų tipams, atributų pavadinimams, kurie yra duomenų bazėje pavadinimu *DB*.

23 apibrėžimas. Funkcija gautiPirminiusRaktus(DB). Funkcija yra skirta gauti visiems pirminiams raktams, kurie yra duomenų bazėje pavadinimu *DB*.

24 apibrėžimas. Funkcija gautiIšoriniusRaktus(DB). Funkcija yra skirta gauti visiems išoriniams raktams, kurie yra duomenų bazėje pavadinimu *DB*.

25 apibrėžimas. Funkcija gautiChecks(DB). Funkcija yra skirta gauti visoms patikrinimo sąlygoms, kurios yra duomenų bazėje *DB*.

26 apibrėžimas. Funkcija gautiApribojimus(DB). Funkcija yra skirta gauti visoms apribojimų sąlygoms, kurios yra duomenų bazėje *DB*.

Algoritmas 3 vaizduoja pseudokodu pateiktą algoritmo dalį, skirtą palyginti gautus CREATE tipo sakinio atsakymus bei pagal juos gauti įvertinimą.

Algoritmas 3 CREATE tipo sakinio automatinis gauto rezultato palyginimas ir įvertinimas

Įvestis: *TikimasisSakinys, VertinamasSakinys, MaxĮvertinimas*
Išvestis: *AutoAtsĮvertinimas*

1: *AtributųVertinimas ← palygintiRezultatus(gautiAtributus("DB1"), gautiAtributus("DB2"));*
2: *PirminiųRaktųVertinimas ← palygintiRezultatus(gautiPirminiusRaktus("PirmaDB"), gautiPirminiusRaktus("AntraDB"));*
3: *IšoriniųRaktųVertinimas ← palygintiRezultatus(gautiIšoriniusRaktus("DB1"), gautiIšoriniusRaktus("DB2"));*
4: *ChecksVertinimas ← palygintiRezultatus(gautiChecks("DB1"), gautiChecks("DB2"));*
5: *ApribojimųVertinimas ← palygintiAtsakymus(gautiApribojimus("DB1"), gautiApribojimus("DB2"));*
6: *AutoAtsĮvertinimas ← (AtributųVertinimas + PirminiųRaktųVertinimas + IšoriniųRaktųVertinimas + ChecksVertinimas + ApribojimųVertinimas) * MaxĮvertinimas;*
7: **return** *AutoAtsĮvertinimas*

CREATE tipo sakinių atsakymai vertinami kiek kitaip nei kitų tipų sakinių, nes kuriant lenteles realiai nėra gaunamas joks lentelės pavidalo rezultatas, kurį būtų galima iškart lyginti. Dėl šios priežasties, CREATE tipo sakiniuose kaip atsakymas yra vertinami atskirai atributai ir jų tipai, išoriniai raktai (angl. *foreign keys*), pirminiai raktai (angl. *primary keys*), patikrinimo sąlygos (angl. *check*), apribojimų sąlygos (angl. *constraints*) gaunamos iš sisteminių duomenų bazės lentelių.

Algoritmo 3 1-6 eilutėse yra pateikta algoritmo dalis, kaip vyksta rezultato vertinimas, jeigu sakinio tipas yra CREATE. Detalus gautų atsakymų palyginimo algoritmo veikimas:

1. Sukuriama nauja duomenų bazė, kurioje sukuriamos lentelės pagal įvestą tikimąsi CREATE sakinių rinkinį;

2. Sukuriama nauja duomenų bazė, kurioje sukuriamos lentelės pagal įvestą CREATE sakinių rinkinį, skirtą vertinimui;

3. Abiejose sukurtose bazėse yra ieškoma informacija, kuri yra perkeliama į dvimačius masyvus, o dvimačiai masyvai yra lyginami tarpusavyje. Bazėse ieškoma informacijos apie:

   (a) visų lentelių atributus ir jų duomenų tipus;

   (b) visų lentelių pirminius raktus;

   (c) visų lentelių išorinius raktus;

   (d) visų lentelių patikrinimo sąlygas;

   (e) visų lentelių apribojimų sąlygas;

4. Suskaičiuoti visi sutapimai yra toliau naudojami įvertinimui skaičiuoti.

4 pavyzdys. Vaizduojami gauti atsakymai pagal įvestus tikimąsi ir vertinamą SQL CREATE tipo sakinius. 8 lentelė vaizduoja gautą atsakymą pagal apribojimo sąlygas pagal tikimąsi sakinį, 9 lentelė vaizduoja gautą atsakymą pagal apribojimo sąlygas pagal vertinamą sakinį, o 10 lentelė vaizduoja rezultatą po atsakymų palyginimo. Pagal gautus rezultatus matome, kad gauti rezultatai nėra vienodi, skiriasi viena eilute. Dėl šios priežasties po rezultatų palyginimo bus gautas ne maksimalus vertinimas. Lygiai tokiu pačiu principu yra lyginami gauti rezultatai ir su kitomis tikrinimo sąlygomis.

TableName	**ColumnName**	**definition**
DAYMENU	price	((9.99))
RESTAURANT	mallid	(NULL)

8 lentelė. Gautas apribojimo sąlygų atsakymas pagal tikimąsi sakinį

TableName	**ColumnName**	**definition**
daymenu	price	((9.99))

9 lentelė. Gautas apribojimo sąlygų atsakymas pagal vertinamą sakinį

TableName	**ColumnName**	**definition**
RESTAURANT	mallid	(NULL)

10 lentelė. Rezultatas po atsakymų palyginimo

27 apibrėžimas. CREATE tipo sakiniai. Pagal įvestus INSERT tipo sakinius *TikimasisSakinys* ir *VertinamasSakinys* yra gaunami atitinkami CREATE sakiniai. Toliau darbe šie CREATE tipo sakiniai apibrėžiami kaip kintamieji *TikimasisCREATESakinys* ir *VertinamasCREATESakinys*.

28 apibrėžimas. Užduotis. Eilutės tipo kintamasis *Užduotis* yra sistemoje pasirinkta užduotis, kuri yra šiuo metu vertinama. *Užduotis* priklauso užduočių aibei *Užduotys*.

29 apibrėžimas. Funkcija gautiCREATESakinį(sakinys, užduotis). Funkcija yra skirta gauti atitinkamam CREATE tipo sakiniui pagal įvestą INSERT tipo sakinį *sakinys* ir pasirinktą vertinamą užduotį *užduotis*. Kintamasis *užduotis* priklauso užduočių aibei *Užduotys*.

30 apibrėžimas. Funkcija gautiInsertAts(sakinys). Funkcija yra skirta gauti visiems duomenims, esantiems duomenų bazėje. Funkcijoje po INSERT tipo sakinių rinkinio *sakinys* įvykdymo yra gaunami visi duomenys, esantys duomenų bazės lentelėse. Duomenys yra gaunami lentelės pavidalu, kurie yra konvertuojami į dvimačio masyvo formatą ir grąžinami kaip išvesties parametras.

Algoritmas 4 vaizduoja pseudokodu pateiktą algoritmo dalį, skirtą palyginti gautus INSERT tipo sakinio atsakymus bei pagal juos gauti įvertinimą.

Algoritmas 4 Automatinis gauto rezultato palyginimas ir įvertinimas

Įvestis: *TikimasisSakinys, VertinamasSakinys, MaxĮvertinimas*
Išvestis: *AutoAtsĮvertinimas*

1: *TikimasisCREATESakinys ← gautiCREATESakinį(TikimasisSakinys, Užduotis);*
2: *VertinamasCREATESakinys ← gautiCREATESakinį(VertinamasSakinys, Užduotis);*
3: *TikimasisRezultatas ← gautiInsertAts(TikimasisSakinys);*
4: *VertinamasRezultatas ← gautiInsertAts(VertinamasSakinys);*
5: *AutoAtsĮvertinimas ← palygintiAtsakymus(TikimasisRezultatas, VertinamasRezultatas) * MaxĮvertinimas;*
6: **return** *AutoAtsĮvertinimas*

INSERT tipo sakiniai sistemoje yra naudojami kiek kitaip nei SELECT ir CREATE tipo sakiniai. INSERT tipo sakiniai yra labai glaudžiai susiję su CREATE tipo sakiniais. Kadangi, naudojant INSERT tipo sakinius, nėra gaunamas joks atsakymas lentelės pavidalu, atsakymo vertinimui yra naudojami kiti kriterijai. Taip pat dažniausiai studentams pateikiamos užduotys yra susijusios kartu ir su CREATE tipo sakinio užduotimis, todėl INSERT ir CREATE tipo sakiniai yra neatsiejami.

Algoritmo 4 1-5 eilutėse yra pateikta algoritmo dalis, kaip vyksta rezultato vertinimas, jeigu sakinio tipas yra INSERT. Algoritmo dalis schematiškai pavaizduota 12 pav. Detalus atsakymų palyginimo algoritmas:

1. Pagal tikimąsi INSERT tipo sakinių rinkinį ir užduotį yra gaunamas susijęs CREATE užduoties atsakymas;

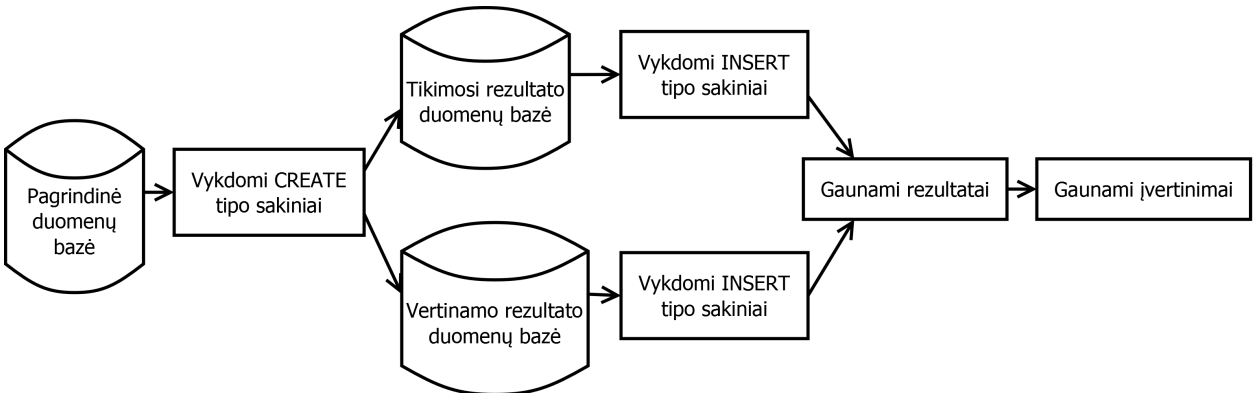
2. Pagal vertinamą INSERT tipo sakinių rinkinį ir užduotį yra gaunamas susijęs CREATE užduoties atsakymas;

3. Sukuriama nauja duomenų bazė, kurioje yra sukuriamos lentelės pagal įvestus susijusius CREATE tipo sakinių rinkinius pagal tikimąsi sakinių rinkinį;

4. Sukuriama nauja duomenų bazė, kurioje yra sukuriamos lentelės pagal įvestus susijusius CREATE tipo sakinių rinkinius pagal vertinamą sakinių rinkinį;

5. Sukurtose duomenų bazėse ir lentelėse įvykdomi atitinkami INSERT tipo sakiniai, kurie bus vertinami;

6. Gaunamas lentelės tipo atsakymas, kuris turi visų lentelių duomenis;



12 pav. INSERT sakinių rinkinio rezultatų gavimo schema

7. Pagal dvimačių masyvų panašumą yra įvertinamas atsakymų panašumas.

31 apibrėžimas. Funkcija pridėtiIšvestiesParametrą(sakinys). Funkcija yra skirta prie SQL sakinio *sakinys*, kurio tipas yra UPDATE arba DELETE, pridėti išvesties parametrą, kurio pagalba bus gaunamas lentelės tipo rezultatas. Funkcijos išvesties parametras yra UPDATE arba DELETE tipo sakinys su pridėtu išvesties parametru.

32 apibrėžimas. Funkcija gautiUpdateDeleteAts(sakinys). Funkcijoje po UPDATE arba DELETE tipo sakinio *sakinys* įvykdymo yra gaunami duomenys, kurie atitinkamai buvo atnaujinti arba pašalinti duomenų bazėje. Duomenys yra gaunami lentelės pavidalu, kurie yra konvertuojami į dvimačio masyvo formatą ir grąžinami kaip išvesties parametras.

Algoritmas 5 vaizduoja pseudokodu pateiktą algoritmo dalį, skirtą palyginti gautus UPDATE ir DELETE tipų sakinių atsakymus bei pagal juos gauti įvertinimą.

Algoritmas 5 UPDATE ir DELETE tipų sakinių automatinis gauto rezultato palyginimas ir įvertinimas

Įvestis: *TikimasisSakinys, VertinamasSakinys, MaxĮvertinimas*
Išvestis: *AutoAtsĮvertinimas*

1: *TikimasisSakinys* ← *pridėtiIšvestiesParametrą(TikimasisSakinys);*
2: *VertinamasSakinys* ← *pridėtiIšvestiesParametrą(VertinamasSakinys);*
3: *TikimasisRezultatas* ← *gautiUpdateDeleteAts(TikimasisSakinys);*
4: *VertinamasRezultatas* ← *gautiUpdateDeleteAts(VertinamasSakinys);*
5: *AutoAtsĮvertinimas* ← *palygintiAtsakymus(TikimasisRezultatas, VertinamasRezultatas)* * *MaxĮvertinimas;*
6: **return** *AutoAtsĮvertinimas*

UPDATE ir DELETE tipo sakiniai yra vertinami labai panašiai, todėl šias algoritmo dalis apžvelgsime kartu. Algoritmo 5 1-5 eilutėse yra pateikta algoritmo dalis, kaip vyksta rezultato vertinimas, jeigu sakinio tipas yra UPDATE arba DELETE tipo. Algoritme naudojamos funkcijos pačios nustato, kurio tipo tai yra sakinys ir pagal tai pritaiko atitinkamus nurodymus. Detalus atsakymų palyginimo algoritmas:

1. Pirmiausia tiek UPDATE, tiek DELETE tipo sakiniai yra papildomai apdorojami:

   (a) Sakinio pradžioje yra pridedama transakcijos pradžia;

(b) Sakinio pabaigoje yra pridedamas pakeitimų atmetimas. Tai yra daroma tam, kad kiekvieno studento sakiniai būtų vertinami, turint tokią pačią duomenų bazę ir joje esančius duomenis;

(c) Atitinkamoje sakinio vietoje yra įterpiamas išvesties parametras, kurio pagalba atitinkamai gaunamos atnaujintos eilutės arba pašalintos eilutės;

2. Apdoroti sakiniai yra įvykdomi ir pridėtų išvesties parametrų pagalba yra gaunamas lentelės tipo atsakymas;

3. Gauti atsakymai yra perkeliami į dvimačius masyvus, kurie yra lyginami tarpusavyje ir pagal sutapimą yra skaičiuojamas galutinis įvertinimas.

5 pavyzdys. Vaizduojamas UPDATE tipo sakinys ir apdorotas sakinys prieš gaunant galutinį rezultatą vertinimui. 3 išeities kodas vaizduoja originalų UPDATE tipo sakinį, o 4 išeities kodas vaizduoja apdorotą sakinį, kuris bus naudojamas rezultato gavimui. Sakinys yra naudojamas tranzakcijoje (1 ir 5 eilutės), o į sakinį pagal sintaksę yra įterptas išvesties parametras (3 eilutė), kurio pagalba yra gaunamas lentelės pavidalo rezultatas.

3 išeities kodas. UPDATE tipo sakinys

```
1 update daymenu set price = 13.45
2 where networkid = 3 AND restaurantinnetworkno = 1 AND courseid = 2
```

4 išeities kodas. Apdorotas UPDATE tipo sakinys

```
1 BEGIN TRANSACTION
2 UPDATE DAYMENU SET PRICE = 13.45
3 OUTPUT INSERTED.*
4 WHERE NETWORK = 3 AND RESTAURANTINNETWORKNO = 1 AND COURSE = 2
5 ROLLBACK TRANSACTION
```

2.4. SQL sakinių ir pagal juos gautų atsakymų vertinimas

Aukščiau aptartuose SQL sakinių ir gautų pagal juos atsakymų palyginimo algoritmuose taip pat yra naudojamas ir automatinis vertinimas. Vertinimas vyksta pagal įvestą SQL sakinį ir pagal gautą atsakymą atskirai, o galutiniame etape yra gaunamas vienas bendras įvertinimas.

2.4.1. SELECT tipo sakinio ir gauto atsakymo vertinimas

Galutinio įvertinimo skaičiavimo etapai:

1. Galimą maksimalų užduoties įvertinimą į sistemą įveda užduoties kūrėjas. Galutinio rezultato skaičiavimas vyksta pagal galimą gauti maksimalų įvertinimą;

2. Įvesto SQL SELECT tipo sakinio vertinimas vyksta keliais etapais:

   (a) Sakinio vertinimas pagal sakinių skirstymą komponentais:

i. Po sakinių pertvarkymo ir suskirstymo komponentais, suskaičiuojama kiek komponentų dvimačiame masyve pagal tikimąsi sakinį užpildyta ląstelių (angl. *cell*) ($lasteles_D = (stulpeliai_{DM} * eilutes_{DM}) - tuscios\_lasteles_{DM}$, kur $lasteles_D$ yra bendras masyvo ląstelių skaičius, $stulpeliai_{DM}$ yra masyvo stulpelių kiekis, $eilutes_{DM}$ yra masyvo eilučių kiekis, o $tuscios\_lasteles_{DM}$ yra dvimačio masyvo neužpildytų ląstelių kiekis);

ii. Po sakinių pertvarkymo ir suskirstymo komponentais, suskaičiuojama kiek komponentų dvimačiame masyve pagal vertinamą sakinį užpildyta ląstelių (angl. *cell*) ($lasteles_S = (stulpeliai_{SM} * eilutes_{SM}) - tuscios\_lasteles_{SM}$, kur $lasteles_S$ yra bendras masyvo ląstelių skaičius, $stulpeliai_{SM}$ yra masyvo stulpelių kiekis, $eilutes_{SM}$ yra masyvo eilučių kiekis, o $tuscios\_lasteles_{SM}$ yra dvimačio masyvo neužpildytų ląstelių kiekis);

iii. Apskaičiavus masyvų užpildytų ląstelių kiekius, toliau yra sukamas ciklas ir skaičiuojama pagal eilutę kiek ląstelių tarp abiejų masyvų sutampa;

iv. Suskaičiavus sutampančių ląstelių kiekį yra skaičiuojamas balas už įvestą sakinį ($balas_k = max\_ivertinimas * (sutampa/lasteles_{DSM})$, kur $balas_k$ yra įvertinimas už įvestą sakinį pagal sakinių skirstymą komponentais, $max\_ivertinimas$ yra užduoties kūrėjo įvestas galimas maksimalus įvertinimas už užduotį, $sutampa$ yra sutampančių ląstelių tarp dviejų masyvų kiekis, o $lasteles_{DSM}$ yra masyvo užpildytų ląstelių kiekis, priklausomai nuo to, kuris skaičius yra didesnis);

(b) Sakinio vertinimas pagal sakinių vertimą į hierarchinę struktūrą:

i. Po sakinių pavertimo į hierarchinę struktūrą ir po hierarchinių struktūrų palyginimo yra analizuojamas gauta galutinė hierarchinė struktūra. Pirmiausia yra suskaičiuojama kiek buvo sutapimų tarp abiejų struktūrų, pagal šakas atitinkamu pavadinimu;

ii. Suskaičiavus sutapimus yra skaičiuojama kiek buvo nesutapimų, skirtumų tarp struktūrų pagal šakas atitinkamu pavadinimu;

iii. Suskaičiavus sutapimus ir nesutapimus yra skaičiuojamas balas už įvestą sakinį ($balas_x = sutampa/(sutampa + skiriasi)$, kur $balas_x$ yra įvertinimas už įvestą sakinį pagal sakinių vertimą į hierarchinę struktūrą, $sutampa$ yra sutapimų skaičius tarp dviejų hierarchinių struktūrų, o $skiriasi$ yra skirtumų kiekis tarp dviejų hierarchinių struktūrų);

(c) Įvertinus sakinį abiem būdais yra lyginami abu gauti rezultatai ir kaip galutinis įvertinimas už įvestą sakinį yra naudojamas gautas didesnis balas;

3. Gauto atsakymo pagal įvestą SQL sakinį vertinimas vyksta keliais etapais:

(a) Suskaičiuojama, kiek pagal tikimąsi sakinį gauto atsakymo masyvą iš viso gaunama ląstelių ($lasteles_{DMA} = stulpeliai_{DMA} * (eilutes_{DMA} - 1)$, kur $lasteles_{DMA}$ yra bendras atsakymo masyvo ląstelių kiekis, $stulpeliai_{DMA}$ yra gautos lentelės stulpelių kiekis, o $eilutes_{DMA}$ yra gautos lentelės eilučių kiekis);

(b) Suskaičiuojama, kiek pagal vertinamą sakinį gauto atsakymo masyvą iš viso gaunama ląstelių ($lasteles_{SMA} = stulpeliai_{SMA} * (eilutes_{SMA} - 1)$, kur $lasteles_{SMA}$ yra bendras atsakymo masyvo ląstelių kiekis, $stulpeliai_{SMA}$ yra gautos lentelės stulpelių kiekis, o $eilutes_{SMA}$ yra gautos lentelės eilučių kiekis);

(c) Įvykdžius gautų atsakymų palyginimo algoritmą, gaunamas skaičius, kiek ląstelių tarp masyvų sutapo;

(d) Atlikus visus nurodytus veiksmus su atsakymų palyginimu yra skaičiuojamas galutinis balas už gautą atsakymą ($balas_a = max\_ivertinimas * (sutampa/lasteles_{DSMA})$, kur $balas_a$ yra galutinis įvertinimas už gautą atsakymą, $max\_ivertinimas$ yra užduoties kūrėjo įvestas galimas maksimalus įvertinimas už užduotį, $sutampa$ yra atsakymų sutampančių ląstelių kiekis, o $lasteles_{DSMA}$ yra masyvo ląstelių kiekis, priklausomai nuo to, kuris skaičius yra didesnis);

4. Suskaičiavus vertinimą už įvestą SQL sakinį ir pagal jį gautą atsakymą, abu balai yra sudedami ir dalinami pusiau (išvedamas aritmetinis vidurkis), kad gautume galutinį įvertinimą ($balas = (balas_s + balas_a)/2$, kur $balas$ yra galutinis užduoties įvertinimas, $balas_s$ yra anksčiau suskaičiuotas įvertinimas už įvestą SQL sakinį, o $balas_a$ yra anksčiau suskaičiuotas įvertinimas už gautą atsakymą).

6 pavyzdys. Vertinamas SELECT tipo SQL sakinys pagal sakinio sintaksę. Vertinamas SQL sakinys, vaizduojamas kaip 2 išeities kodas. Pirmiausia vertinimas vyksta naudojant analizatorių. Hierarchinių struktūrų skirtumas pavaizduotas 11 pav. Pirmiausia yra suskaičiuojami sutapimai tarp SQL sakinių. Sutapimai vaizduojami medžio šakose, kurios vadinamos *match*. Yra sudedami skaičiai ($sutampa = 1 + 2 + 1 + 1 + 1 + 3 = 9$). Suskaičiavus sutapimus yra skaičiuojami skirtumai. Skirtumus vaizduoja šakos, pavadinimais *add*, *remove*. Suskaičiuojama, kiek iš viso tokių šakų yra. Pateiktu atveju šakos yra penkios ($skiriasi = 5$). Tokiu atveju, pritaikius formulę, gaunamas rezultatas pagal hierarchinę struktūrą $balas_x = 9/(9 + 5) = 0.64$. Apskaičiavus balą pagal hierarchinę struktūrą, tie patys sakiniai yra skaidomi į komponentus. Komponentų skaidymas pavaizduotas 2 pav. ir 3 pav. Skirtumai vaizduojami 4 pav. Pagal šį gautą masyvą ir bus skaičiuojamas galutinis įvertinimas už įvestą SQL sakinį. Pirmiausia yra suskaičiuojamas tikimosi sakinio masyvo ląstelių kiekis ($lasteles_D = 19$), suskaičiuojamas vertinamo algoritmo ląstelių kiekis ($lasteles_S = 15$). Suskaičiavus ląstelių kiekius yra skaičiuojama, kiek buvo sutapimų tarp abiejų masyvų. Šiuo atveju sutapimų buvo 15 ($sutapimai = 15$). Galutinis rezultatas $balas_k = 15/19 = 0.79$. Galutinis balas už įvestą SQL sakinį bus $balas_s = 0.79$, nes $0.79 > 0.64$.

7 pavyzdys. Vertinamas gautas atsakymas pagal SELECT tipo sakinį. Gautas atsakymas pagal tikimąsi sakinį vaizduojamas 5 lentelėje, gautas rezultatas pagal vertinamą sakinį vaizduojamas 6 lentelėje, o skirtumai tarp šių rezultatų vaizduojami 7 lentelėje. Pirmiausia yra suskaičiuojama, kiek pagal tikimąsi sakinį gautame rezultato masyve yra ląstelių ($lasteles_{DMA} = 6$), suskaičiuojama, kiek gautame masyve pagal vertinamą sakinį yra ląstelių ($lasteles_{SMA} = 12$). Suskaičiavus ląstelių kiekius yra skaičiuojama, kiek buvo sutapimų tarp abiejų masyvų. Šiuo atveju sutapimų buvo 6 ($sutapimai = 6$). Galutinis rezultatas $balas_a = 6/12 = 0.5$.

2.4.2. CREATE tipo sakinio ir gauto rezultato vertinimas

Galutinio įvertinimo skaičiavimo etapai:

1. Galimą maksimalų užduoties įvertinimą į sistemą įveda užduoties kūrėjas. Galutinio rezultato skaičiavimas vyksta pagal galimą gauti maksimalų įvertinimą;

2. Įvesto SQL CREATE tipo sakinio vertinimas vyksta lygiai taip pat kaip SQL SELECT tipo sakinio vertinimas, todėl papildomai aprašymas nebus pateiktas;

3. Gautų rezultatų pagal įvestą SQL CREATE tipo sakinį vertinimas vyksta keliais etapais:

   (a) Atrenkami visi atributai ir jų tipai pagal lentelės pavadinimą iš abiejų sukurtų duomenų bazių ir atsakymai perkeliami į dvimačius masyvus:

      i. Suskaičiuojama, kiek pagal tikimąsi sakinį gauto rezultato masyvą iš viso gaunama ląstelių ($lasteles_{DA} = stulpeliai_{DA} * eilutes_{DA}$, kur $lasteles_{DA}$ yra bendras ląstelių skaičius pagal rezultatą, $stulpeliai_{DA}$ yra gautos lentelės stulpelių kiekis, o $eilutes_{DA}$ yra gautos lentelės eilučių kiekis);

      ii. Suskaičiuojama, kiek pagal vertinamą sakinį gauto rezultato masyvą iš viso gaunama ląstelių ($lasteles_{SA} = stulpeliai_{SA} * eilutes_{SA}$, kur $lasteles_{SA}$ yra bendras ląstelių skaičius pagal rezultatą, $stulpeliai_{SA}$ yra gautos lentelės stulpelių kiekis, o $eilutes_{SA}$ yra gautos lentelės eilučių kiekis);

      iii. Įvykdžius gautų masyvų palyginimo algoritmą, gaunamas skaičius, kiek ląstelių tarp masyvų sutapo;

      iv. Atlikus visus prieš tai veiksmus yra skaičiuojamas galutinis balas už gautą rezultatą pagal atributus ($balas_{atr} = sutampa/lasteles_{DSA}$, kur $balas_{atr}$ yra įvertinimas už gautą rezultatą pagal atributus, $sutampa$ yra atsakymų sutampančių ląstelių kiekis, o $lasteles_{DSA}$ yra masyvo ląstelių kiekis, priklausomai nuo to, kuris skaičius yra didesnis);

   (b) Atrenkami visi visų lentelių, esančių duomenų bazėje, pirminiai raktai pagal lentelės pavadinimą ir atsakymai perkeliami į dvimačius masyvus. Toliau su masyvais atliekami tokie patys veiksmai, aprašyti lyginant masyvus pagal atributus;

   (c) Atrenkami visi visų lentelių, esančių duomenų bazėje, išoriniai raktai pagal lentelės pavadinimą ir lentelių, į kurias rodo raktai, pavadinimus ir atsakymai perkeliami į dvimačius masyvus. Toliau su masyvais atliekami tokie patys veiksmai, aprašyti lyginant masyvus pagal atributus;

   (d) Atrenkamos visos visų lentelių, esančių duomenų bazėje, patikrinimo sąlygos pagal lentelės ir stulpelio pavadinimą ir atsakymai perkeliami į dvimačius masyvus. Toliau su masyvais atliekami tokie patys veiksmai, aprašyti lyginant masyvus pagal atributus;

   (e) Atrenkamos visos visų lentelių, esančių duomenų bazėje, apribojimų sąlygos pagal lentelės ir stulpelio pavadinimą ir atsakymai perkeliami į dvimačius masyvus. Toliau su masyvais atliekami tokie patys veiksmai, aprašyti lyginant masyvus pagal atributus;

   (f) Suskaičiavus vertinimus pagal gautus rezultatus yra skaičiuojamas bendras balas už gautą atsakymą. Skaičiuojant galutinį balą taip pat yra naudojami papildomi koeficientai pagal gaunamų rezultatų tipą ($balas_a = max_i vertinimas * (koef_{atr} * balas_{atr} + koef_{PK} * balas_{PK} + koef_{FK} * balas_{FK} + koef_C * balas_C + koef_{CO} * balas_{CO})$, kur $balas_a$ yra galutinis įvertinimas už gautą rezultatą, $max_i vertinimas$ yra dėstytojo įvestas galimas maksimalus įvertinimas už užduotį, $koef_{atr}$ yra koeficientas už atributų dalies vertinimą, $balas_{atr}$ yra įvertinimas už atributų dalies rezultatus, $koef_{PK}$ yra koeficientas už pirminių raktų dalies vertinimą, $balas_{PK}$ yra įvertinimas už pirminių raktų dalies rezultatus, $koef_{FK}$ yra koeficientas už išorinių raktų dalies vertinimą, $balas_{FK}$ yra įvertinimas už išorinių raktų dalies rezultatus, $koef_C$ yra koeficientas už patikrinimo sąlygų dalies vertinimą, $balas_C$ yra įvertinimas už patikrinimo sąlygų dalies rezultatus, $koef_{CO}$ yra koeficientas už apribojimo sąlygų dalies vertinimą, $balas_{CO}$ yra įvertinimas už apribojimo sąlygų dalies rezultatus);

4. Suskaičiavus vertinimą už įvestą SQL sakinį ir pagal jį gautą atsakymą, abu balai yra sudedami ir dalinami pusiau (išvedamas aritmetinis vidurkis), kad gautume galutinį įvertinimą ($balas = (balas_s + balas_a)/2$, kur $balas$ yra galutinis užduoties įvertinimas, $balas_s$ yra anksčiau suskaičiuotas įvertinimas už įvestą SQL sakinį, o $balas_a$ yra anksčiau suskaičiuotas įvertinimas už gautą atsakymą).

2.4.3. INSERT tipo sakinio ir gauto rezultato vertinimas

Galutinio įvertinimo skaičiavimo etapai:

1. Galimą maksimalų užduoties įvertinimą į sistemą įveda užduoties kūrėjas. Galutinio rezultato skaičiavimas vyksta pagal galimą maksimalų įvertinimą;

2. Įvesto SQL INSERT tipo sakinio vertinimas vyksta taip pat, kaip ir CREATE tipo sakinio vertinimas, todėl papildomas aprašymas nebus pateiktas;

3. Gautų rezultatų pagal įvestą SQL INSERT tipo sakinį vertinimas vyksta keliais etapais:

   (a) Kadangi šiame modelyje INSERT sakiniai yra naudojami iškart po CREATE sakinių panaudojimo, atsakymas yra gaunamas lentelės pavidalu, ištraukiant visiškai visas reikšmes, esančias duomenų bazėje (šis veiksmas atliekamas tiek su tikimosi sakinio, tiek su vertinamo sakinio gautais rezultatais po INSERT sakinių panaudojimo);

   (b) Suskaičiuojama, kiek pagal tikimąsi sakinį gauto rezultato masyvą iš viso gaunama ląstelių, kurių reikšmė nėra tuščia ($lasteles_{DA} = (stulpeliai_{DA} * eilutes_{DA}) - lasteles_{NULL}$, kur $lasteles_{DA}$ yra bendras ląstelių skaičius pagal rezultatą, $stulpeliai_{DA}$ yra gautos lentelės stulpelių skaičius, $eilutes_{DA}$ yra gautos lentelės eilučių skaičius, o $lasteles_{NULL}$ yra ląstelių kiekis, kurių reikšmė yra tuščia);

   (c) Suskaičiuojama, kiek pagal vertinamą sakinį gauto rezultato masyvą iš viso gaunama ląstelių, kurių reikšmė nėra tuščia ($lasteles_{SA} = (stulpeliai_{SA} * eilutes_{SA}) - lasteles_{NULL}$, kur $lasteles_{SA}$ yra bendras ląstelių skaičius pagal rezultatą, $stulpeliai_{SA}$ yra gautos lentelės stulpelių skaičius, $eilutes_{SA}$ yra gautos lentelės eilučių skaičius, o $lasteles_{NULL}$ yra ląstelių kiekis, kurių reikšmė yra tuščia);

   (d) Įvykdžius dviejų masyvų palyginimo algoritmą, gaunamas skaičius, kiek masyvų ląstelių, kurios nėra tuščios, sutapo;

   (e) Atlikus visus prieš tai aprašytus veiksmus yra skaičiuojamas galutinis balas už gautą atsakymą ($balas_a = max\_ivertinimas * (sutampa/lasteles_{DSA})$, kur $balas_a$ yra galutinis balas, gautas už atsakymą, $max_ivertinimas$ yra užduoties kūrėjo įvestas galimas maksimalus įvertinimas už užduotį, $sutampa$ yra sutampančių ląstelių kiekis tarp masyvų, $lasteles_{DSA}$ yra masyvo ląstelių kiekis, priklausomai nuo to, kuris skaičius yra didesnis);

4. Suskaičiavus abu įvertinimus, už įvestą sakinių rinkinį ir už gautą atsakymą, yra skaičiuojamas galutinis balas už užduotį. Abu gauti balai yra sudedami ir iš jų yra išvedamas aritmetinis vidurkis, kuris ir reiškia galutinį balą už užduotį ($balas = (balas_s + balas_a)/2$, kur $balas$ yra galutinis balas už užduotį, $balas_s$ yra gautas įvertinimas už įvestą SQL sakinį, $balas_a$ yra gautas įvertinimas už rezultatą).

2.4.4. UPDATE ir DELETE tipų sakinių ir gautų atsakymų vertinimas

Galutinio įvertinimo skaičiavimo etapai:

1. Galimą maksimalų užduoties įvertinimą į sistemą įveda užduoties kūrėjas. Galutinio rezultato skaičiavimas vyksta pagal galimą maksimalų balą;

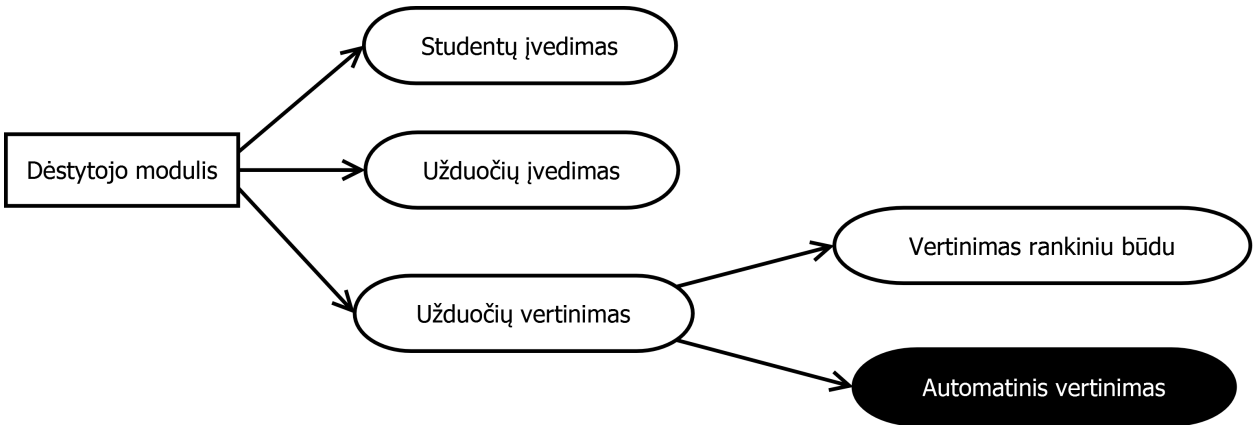
2. Įvestų UPDATE ir DELETE sakinių vertinimas vyksta taip pat kaip ir kitų tipų sakinių vertinimas, todėl papildomas aprašymas nebus pateikiamas;

3. Kadangi UPDATE ir DELETE sakinių atsakymams gauti yra naudojami išvesties parametrai, todėl gautų atsakymų vertinimas yra analogiškas kaip ir vertinant SELECT tipo sakinius, todėl papildomas aprašymas taip pat nebus pateikiamas;

4. Suskaičiavus vertinimą už įvestą SQL sakinį ir pagal jį gautą atsakymą, abu balai yra sudedami ir iš jų yra vedamas aritmetinis vidurkis ($balas = (balas_s + balas_a)/2$, kur $balas$ yra galutinis užduoties įvertinimas, $balas_s$ yra suskaičiuotas įvertinimas už SQL sakinį, o $balas_a$ yra suskaičiuotas įvertinimas už gautą atsakymą) ir taip yra gaunamas galutinis užduoties įvertinimas.

3. Algoritmų įgyvendinimas

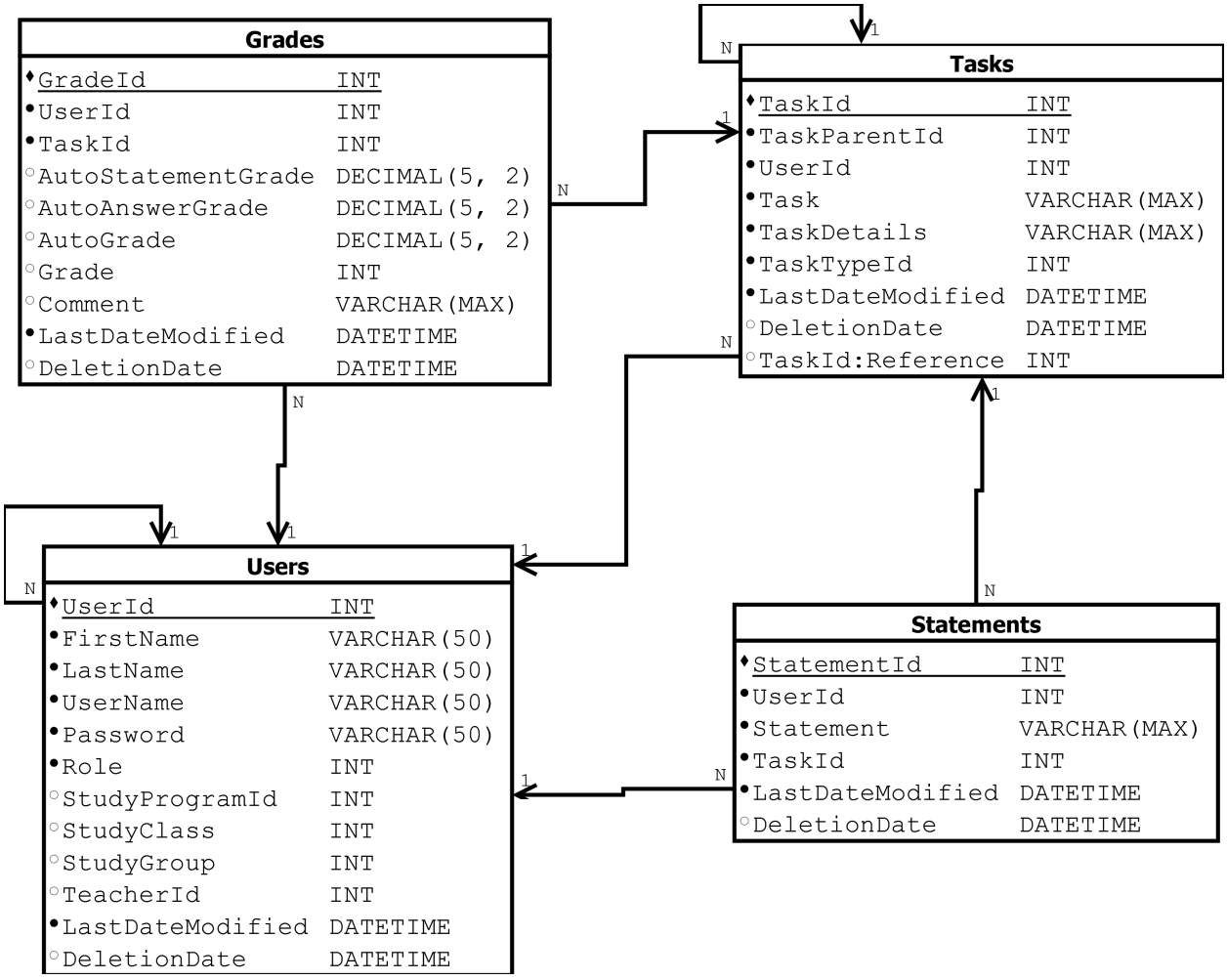
Darbe aprašytas edukacinis modelis ir jame naudojami algoritmai yra integruoti į edukacinę platformą *SQLintojas* [1], kuri buvo sukurta bakalaurinio darbo metu. 13 pav. vaizduojamas sistemos *SQLintojas* dėstytojo modulis, kuris turi tam tikras funkcijas. Dėstytojo modulyje yra galimybė įvesti studentus, įvesti užduotis bei tas užduotis įvertinti. Užduočių vertinimas gali vykti rankiniu būdu, kai dėstytojas pats įveda įvertinimą į sistemą ir taip pat užduočių vertinimas gali vykti automatiškai. Šis modulis buvo papildytas automatinio vertinimo algoritmais.



13 pav. Dėstytojo modulio funkcijos

Visi duomenys apie užduotis, užduočių atsakymus, įvertinimus yra saugomi duomenų bazėje. Duomenų bazės schemos dalis pateikta 14 pav. Pateiktos pagrindinės reikalingos lentelės, naudojamos automatiniam vertinimui. Lentelėje *Grades* yra saugomi įvertinimui už užduotis, lentelėje *Tasks* yra saugomos užduotys ir jų formuluotės, lentelėje *Users* yra saugomi visi sistemos vartotojų duomenys, o lentelėje *Statements* yra saugomi užduočių atsakymai (SQL sakiniai). Visos lentelės turi pirminį raktą ir bent vieną išorinį raktą. Taip pat kiekviena duomenų bazės lentelė turi laukus

LastDateModified ir *DeletionDate*. Laukas *LastDateModified* yra skirtas saugoti datai, kuri nurodo kada paskutinį kartą įrašas buvo koreguotas. Laukas *DeletionDate* yra skirtas saugoti datai, kuri nurodo, kada įrašas buvo pašalintas. Įrašai realiai nėra šalinami iš duomenų bazės, o yra tiesiog užpildomas laukas *DeletionDate*. Tai yra daroma saugumo sumetimais, kad nebūtų per klaidą ištrinti reikalingi įrašai.



14 pav. Duomenų bazės schema

Svarbiausia lentelė, naudojama automatiniam vertinimui, yra *Grades*. Automatinio algoritmo įvertinimai yra saugomi laukuose *AutoStatementGrade*, *AutoAnswerGrade* ir *AutoGrade*. Lauke *AutoStatementGrade* yra saugomas vertinimo algoritmo skirtas įvertinimas už įvestą vertinamą SQL sakinį. Lauke *AutoAnswerGrade* yra saugomas vertinimo algoritmo skirtas įvertinimas už gautą rezultatą pagal įvestą SQL sakinį. Lauke *AutoGrade* yra saugomas galutinis automatinio vertinimo algoritmo skirtas įvertinimas.

4. Algoritmų testavimas

Algoritmai buvo testuojami naudojant realius duomenis. Naudojamos Vilniaus universiteto Matematikos ir informatikos fakulteto Informacinių technologijų studijų programos 2 kurso studentų Duomenų bazių valdymo sistemų dalyko užduotys ir studentų atsakymai. Algoritmai testuoti su SELECT, CREATE, INSERT, UPDATE, DELETE tipo sakiniais.

4.1. SELECT tipo sakinių testavimo rezultatai

SELECT tipo sakinių analizei buvo naudoti 11-os studentų sakiniai. Kadangi sakiniai buvo rašomi, naudojant *PostgreSQL* duomenų bazę, o algoritme yra naudojamas *Microsoft SQL Server*, prieš vertinimą sakiniai buvo apdoroti rankiniu būdu, pritaikant sintaksę iš vienos duomenų bazės standarto į kitos duomenų bazės standartą.

11 lentelė vaizduoja gautus rezultatų skirtumus tarp dėstytojo įvertinimo ir algoritmo įvertinimo. Jeigu skaičius yra teigiamas, reiškia dėstytojo įvertinimas buvo didesnis, o jeigu skaičius yra neigiamas – algoritmo įvertinimas buvo didesnis. Paskutinėje lentelės eilutėje pateikiama užduoties rezultatų skirtumų suma. Pilni dėstytojo ir algoritmo vertinimo rezultatai pateikti A priede.

Stud. nr.	**I.**	**II.**	**III.**	**IV.**	**V.**
1	0.06	0.02	0.09	-0.46	-0.17
2	0	0.21	0.09	0	0
3	-0.02	0.33	0.02	-0.38	-0.12
4	0.06	0.14	0.27	0.08	0
5	0.03	0.55	0.29	0.1	0.08
6	0.09	0.61	0.26	0.09	0.09
7	0	-0.01	0.6	0.18	0.07
8	0.08	0.59	0.26	-0.1	0
9	0	0.18	0.34	0.23	0
10	0	0.21	0.43	0	-0.01
11	0.06	0.14	0.26	0.15	0.1
$\sum$	**0.36**	**2.99**	**2.94**	**-0.07**	**0.05**

11 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje skaičius nurodo, kuri tai yra užduotis.

Kadangi gauti rezultatai labai skirtingi ir įvairūs – paanalizuokime gautus rezultatus pagal atskiras užduotis:

1. I užduotis. Kaip matome pagal rezultatus, dalis užduočių vertinimų pilnai sutapo tarp dėstytojo ir studento vertinimo. Kiti vertinimo nuokrypiai yra nežymūs. Nuokrypiai galėjo atsirasti dėl to, kad studentai naudojo papildomas, nebūtinai reikalingas, sąlygas gauti teisingam rezultatui;

2. II užduotis. Antrosios užduoties rezultatų nuokrypis žymiai didesnis nei pirmosios užduoties. Dėstytojo skirti įvertinimai studentams buvo žymiai didesni, nei įvertinimai, skirti algoritmo. Taip įvyko dėl to, kad užduotyje buvo naudojami sudėtiniai sakiniai, kurie visi buvo visiškai skirtingi ir sintaksiškai mažai panašūs tarpusavyje;

3. III užduotis. Nuokrypis taip pat labai didelis kaip ir antrosios užduoties. Beveik visų studentų užduotis dėstytojas įvertino žymiai didesniais balai nei algoritmas, nes sakiniai buvo sudėtiniai ir mažai sintaksiškai panašūs, nors rezultatas pagal sakinius ir gaunamas vienodas;

4. IV užduotis. Bendras vertinimo nuokrypis labai nežymus. Kaip matoma lentelėje, kai kurias užduotis algoritmas įvertino žymiai geriau nei dėstytojas. Taip nutiko dėl to, kad dėstytojas

skyrė nulinius vertinimus, nes sakiniai neturėjo prasmės, tačiau algoritmas vis tiek įvertino įvestus sakinius pagal sakinių panašumą;

5. V užduotis. Šios užduoties vertinimų nuokrypis visiškai nežymus. Kai kurios užduotys nebuvo pateiktos, todėl skirtumo tarp nulinių vertinimų nėra. Kiti vertinimai, užduočių, kurios buvo įvestos, tarp dėstytojo skirto įvertinimo ir algoritmo skirto įvertinimo yra labai panašūs ir nuokrypis labai nedidelis.

Taip pat kai kurie žymesni nuokrypiai nuo dėstytojo įvertinimo yra todėl, kad gaunamas atsakymas visiškai neatitinka reikiamo atsakymo, nors SQL sakinys yra beveik teisingas. Pavyzdžiui, jeigu studentas nurodo palyginimo ženklą („<“ arba „>“) ne į tą pusę, SQL sakinys bus laikomas beveik teisingu ir bus įvertintas beveik maksimaliu balu, tačiau gautas atsakymas greičiausiai bus visiškai neteisingas ir bus įvertintas nuliniu balu. Nors, jeigu būtų vertinama rankiniu būdu, studentas greičiausiai būtų įvertintas aukštesniu balu, nei vertinant automatiniais algoritmais, kai vertinama ne tik pagal įvestą sakinį, bet ir pagal gautą atsakymą vienodais koeficientais.

Papildomai galima paanalizuoti paskutinę lentelės eilutę, kurioje yra pateikta užduoties įvertinimų suma. Kaip galima matyti, mažiausi nuokrypiai tar dėstytojo ir algoritmo įvertinimų yra IV-osios ir V-osios užduoties. I-osios užduoties bendras vertinimo nuokrypis taip pat nėra labai žymus, tačiau II-osios ir III-oios užduočių vertinimai tarp dėstytojo ir algoritmo žymiai skiriasi. Pagal šiuos rezultatus galima vertinti, kad II-oji ir III-oji užduotys galėjo būti pakankamai sudėtingos ir didelis nuokrypis gaunamas dėl daug sudėtinių sakinių naudojimo, arba užduotys galėjo būti ne visiškai suprantamos, todėl įvestas sakinys arba gautas atsakymas neatitiko norimo rezultato, arba yra galimybė daryti labai daug interpretacijų rašant sakinius, todėl daugelis studentų pasirinko visiškai kitokį būdą užklausos rašymui, nei tikėjosi dėstytojas.

4.2. CREATE tipo sakinių testavimo rezultatai

CREATE tipo sakinių analizei buvo naudoti 17-os studentų sakinių rinkiniai. Kadangi sakiniai buvo rašomi naudojant *PostgreSQL* duomenų bazę, o algoritme naudojamas *Microsoft SQL Server*, todėl prieš vertinimą sakiniai buvo apdoroti rankiniu būdu, pritaikant sintaksę iš vienos duomenų bazės standarto į kitos duomenų bazės standartą.

12 lentelė vaizduoja gautus rezultatų po dėstytojo vertinimo ir po algoritmo, vertinančio gautą rezultatą, vertinimo skirtumus. Lentelėje pateikti vertinimai atskirai pagal penkias vertinamas sritis – atributai ir jų tipai (stulpelis Atr.), pirminiai raktai (stulpelis PK.), išoriniai raktai (stulpelis FK.), patikrinimo sąlygos (stulpelis Ck.), apribojimų sąlygos (stulpelis CO.). Paskutinėje eilutėje pateikiama stulpelio vertinimų suma, pagal kurią galima matyti vertinimų tarp dėstytojo ir algoritmo skirtumą. Jeigu reikšmė yra neigiama, vadinasi algoritmo vertinimas buvo didesnis nei dėstytojo, o jeigu teigiamas – dėstytojo vertinimas buvo didesnis. Pilni dėstytojo ir algoritmo vertinimo rezultatai pateikti B priede.

Kadangi rezultatai yra pakankamai skirtingi ir vertinimas vyksta už skirtingas sritis – paanalizuokime gautus rezultatus pagal atskirą vertinamą sritį:

1. Atributai ir jų duomenų tipai. Pagal gautus rezultatus stulpelyje „Atr.“ galima matyti, kad pilnai vertinimai tarp dėstytojo ir algoritmo sutapo tik tris kartus iš 17. Dalis vertinimų yra labai arti dėstytojo pateikto vertinimo. Viena iš priežasčių, kodėl algoritmas galėjo įvertinti užduotis kitaip nei dėstytojas – duomenų tipų nesutapimas. Nemažai studentų vengia naudoti *integer* duomenų tipą ir dažniau viskam renkasi tekstinius duomenų tipus. Kita priežastis

Stud. nr.	Atr.	PK.	FK.	Ck.	CO.
1	0.07	0.11	0	0.8	0
2	0	0	0	0.73	0
3	0.06	0.17	0.15	0.12	0.5
4	0.47	0.78	0.75	0	1
5	0	0	0.2	0	0
6	0.04	0.11	0.3	0	0
7	0	0	0	0	0
8	-0.19	0	0	0.43	0
9	-0.06	0	0	0	0
10	0.02	0.11	0.1	0	0
11	-0.25	-0.06	-0.2	0.73	0.5
12	0.12	0.44	0.35	0	0.5
13	0.19	0	0	0.73	0
14	-0.18	0	0.15	0	0
15	0.11	0	0	0	0
16	0.04	0.11	-0.05	0	0.5
17	0.33	0.44	0.35	0	1
$\sum$	**0.8**	**2.21**	**2.1**	**3.54**	**4**

12 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis trupinys nurodo, kuri sritis yra vertinama.

dėl rezultatų nesutapimų – atributų pavadinimai, kuriuos studentas įveda ne pagal užduotyje pateiktus reikalavimus, įveda su rašybos klaidomis.

2. Pirminiai raktai. Pagal gautus rezultatus stulpelyje „PK.“ galima matyti, kad beveik pusė gautų rezultatų naudojant algoritmą sutapo su dėstytojo vertinimais. Kita dalis vertinimų buvo arti dėstytojo skirtų rezultatų ir tik labai maža dalis žymiai skyrėsi nuo dėstytojo vertinimų.

3. Išoriniai raktai. Pagal gautus rezultatus stulpelyje „FK.“ galima matyti, kad beveik visi skirti daliniai vertinimai tarp dėstytojo vertinimo ir algoritmo neatitiko, tačiau nuokrypis tarp vertinimo skirtumų nėra labai žymus. Tokius skirtumus galėjo lemti tai, kad ne visi studentai naudoja išorinius raktus apibrėžti keliems atributams iškart ir dėstytojas už tokias klaidas minusuoja mažiau balų nei tai daro naudojamas algoritmas.

4. Patikrinimo sąlygos. Pagal gautus rezultatus stulpelyje „Ck.“ galima matyti, kad nė vienas vertinimas, išskyrus nulinius vertinimus, tarp dėstytojo ir algoritmo vertinimų neatitiko. Tai galėjo lemti priežastis, kad dėstytojas ir studentai tas pačias patikrinimo sąlygas galėjo apibrėžti visiškai skirtingai, kai reikšmė reiškia lygiai tą patį, pavyzdžiui, studento įvestas patikrinimas *name is not null* ir dėstytojo įvestas patikrinimas *name != null* yra skaitomi kaip vienodi vertinant rankiniu būdu, tačiau yra vertinama kaip skirtingi, kai vertina algoritmas.

5. Apribojimų sąlygos. Pagal gautus rezultatus stulpelyje „CO.“ galima matyti, kad daugelis gautų rezultatų sutampa. Nesutampa kiek mažiau nei trečdalis rezultatų. Tai galėjo nutikti

dėl tos priežasties, kad studento įvesta apribojimo sąlyga neatitiko standartų ir dėl to nebuvo įtraukta į duomenų bazę.

Toliau buvo skaičiuojami galutiniais rezultatai už atliktą užduotį. Maksimalus balas, kurį galėjo gauti studentas – 7. 13 lentelė vaizduoja galutinių dėstytojo vertinimų ir algoritmo vertinimo už sakinius skirtumą, galutinius algoritmo vertinimo už gautą rezultatą (priskyrus atitinkamus koeficientus kiekvienai vertinamai sričiai) ir dėstytojo galutinio vertinimo skirtumus, galutinius įvertinimų skirtumus už užduotį tarp algoritmo bei dėstytojo skirtų įvertinimų. Stulpelio pavadinimas nurodo, kuris algoritmas buvo naudojamas vertinimui. Jeigu reikšmė yra neigiama, vadinasi algoritmo vertinimas buvo didesnis nei dėstytojo, o jeigu teigiamas – dėstytojo vertinimas buvo didesnis.

Stud nr.	**Sakinys.**	**Atsakymas.**	**Galutinis.**
1	1.64	1.78	1.71
2	-0.11	1.46	0.67
3	-0.59	1.27	0.34
4	0.54	3.75	2.14
5	-0.68	0.4	-0.14
6	0.21	0.75	0.48
7	-3.11	0	-1.56
8	0.94	0.67	0.8
9	-2.74	-0.06	-1.4
10	1.55	0.33	0.94
11	-0.38	1.25	0.43
12	-1.57	1.76	0.09
13	1.15	1.65	1.4
14	-0.09	0.12	0.01
15	-1.29	0.11	-0.59
16	-0.74	0.55	-0.1
17	-2.13	1.97	-0.08
$\sum$	**-6.9**	**18.26**	**5.68**

13 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje žodis nurodo, kuris algoritmas buvo naudojamas vertinimui.

Gautus galutinius rezultatus paanalizuokime atskirai, lygindami galutinį dėstytojo rezultatą su atskirais algoritmų vertinimais bei galutiniu algoritmų vertinimu:

1. Algoritmo vertinimas už įvestą sakinį ir galutinis dėstytojo vertinimas. Pagal gautus rezultatus stulpelyje „Sakinys.“ galima matyti, kad vertinimas vien pagal įvestą SQL sakinį yra gerokai didesnis nei galutinis dėstytojo užfiksuotas rezultatas. Taip yra dėl to, kad vertinant įvestą sakinį yra atsižvelgiama į sakinių sintaksinį panašumą ir nėra atskirai sakiniuose analizuojami naudojami pirminiai raktai, išoriniai raktai, atributai bei kiti papildomi aspektai.

2. Algoritmo vertinimas pagal gautą rezultatą ir galutinis dėstytojo vertinimas. Pagal gautus rezultatus stulpelyje „Atsakymas.“ galima matyti, kad vertinimas taip pat žymiai skiriasi, kaip ir vertinant tik pagal įvestą SQL CREATE tipo sakinį. Vertinant pagal gautą atsakymą,

algoritmo vertinimai yra žemesni nei dėstytojo skiriami vertinimai. Detalesnis nagrinėjimas apžvelgiamas 12 lentelės analizėje.

3. Algoritmo vertinimas gaunant gautinį rezultatą ir galutinis dėstytojo vertinimas. Pagal gautus rezultatus stulpelyje „Galutinis.“ galima matyti, kad rezultatai yra panašiausi, nei lyginant atskirai pagal gautą atsakymą ar gautą rezultatą.

Taigi, pagal gautus galutinius rezultatus galima teigti, kad pasiteisina studentų užduočių vertinimas ne tik pagal įvestą sakinį arba tik gautą rezultatą. Kombinuojant abu šiuos vertinimus yra pasiekiamas tiksliausias galutinis vertinimas už užduotį. Nors algoritmo vertinimai ir nėra visiškai atitinkantys dėstytojo vertinimų, tačiau galutinis rezultatus yra pakankamai geras. Atliekant tolimesnius algoritmo patobulinimus taip pat reikėtų atsižvelgti ir į tai, kaip griežtai turi būti vertinamos užduotys.

4.3. INSERT tipo sakinių testavimo rezultatai

INSERT tipo sakinių analizei buvo naudoti 17-os studentų sakinių rinkiniai. Kadangi sakiniai buvo rašomi naudojant *PostgreSQL* duomenų bazę, o algoritme naudojamas *Microsoft SQL Server*, todėl prieš vertinimą sakiniai buvo apdoroti rankiniu būdu, pritaikant sintaksę iš vienos duomenų bazės standarto į kitos duomenų bazės standartą.

14 lentelė vaizduoja gautus rezultatų skirtumus po dėstytojo vertinimo ir po algoritmo vertinimo. Stulpelio pavadinimas nurodo, koks naudojamas vertinimo būdas (Sakinys – vertinimas už INSERT tipo sakinių rinkinius ir galutinio dėstytojo įvertinimo skirtumas, Atsakymas – vertinimas už INSERT tipo sakinių gautus atsakymus ir galutinio dėstytojo įvertinimo skirtumas, Galutinis – gautas galutinis studento įvertinimas naudojant automatinio vertinimo algoritmą ir dėstytojo galutinio įvertinimo skirtumas). Paskutinėje lentelės eilutėje yra pateikiama atitinkamo stulpelio vertinimų suma, kad būtų galima palyginti nuokrypius tarp skirtingų vertinimo būdų ir galutinių rezultatų. Jeigu reikšmė yra teigiama, vadinasi dėstytojas įvertino užduotį didesniu balu nei algoritmas, o jeigu neigiamas – algoritmo vertinimas buvo didesnis. Pilni dėstytojo ir algoritmo vertinimo rezultatai pateikti C priede.

Gautus rezultatus paanalizuokime atskirai, lygindami galutinį dėstytojo rezultatą su atskirais algoritmų vertinimais bei galutiniu algoritmų vertinimu:

1. Algoritmo vertinimas už įvestą sakinį ir galutinis dėstytojo vertinimas. Pagal gautus rezultatus stulpelyje „Sakinys.“ galima matyti, kad vertinimas vien pagal įvestą SQL INSERT tipo sakinių rinkinį yra gerokai mažesnis nei galutinis dėstytojo užfiksuotas rezultatas. Kadangi INSERT sakiniai yra pritaikyti atitinkamai duomenų bazės struktūrai pagal CREATE sakinių rinkinius, gali būti neatitikimai tarp dėstytojo ir studento sakinių dėl naudojamų skirtingų duomenų tipų skirtingose duomenų bazėse. Taip pat daugelyje lentelių yra naudojami įvairūs pirminiai raktai, kurie irgi gali būti naudojami skirtingi dėstytojų ir studentų, jeigu užduotyje nėra konkrečiai apibrėžta, kokie jie turi būti.

2. Algoritmo vertinimas pagal gautą rezultatą ir galutinis dėstytojo vertinimas. Pagal gautus rezultatus stulpelyje „Atsakymas.“ galima matyti, kad vertinimas naudojant algoritmą taip pat yra žemesnis nei dėstytojo skirtas vertinimas. Vertinant atsakymus nėra atsižvelgiama į duomenų bazės lentelėse naudojamus duomenų tipus, tačiau neatitikimas taip pat yra dėl naudojamų pirminių raktų ir jų reikšmių, jeigu užduotyje nėra apibrėžtos konkrečios reikšmės, kurios turi būti naudojamos.

Stud. nr.	Sakinys.	Atsakymas.	Galutinis.
1	0.29	0.17	0.23
2	0.31	0.17	0.24
3	0.33	0.21	0.27
4	0.58	-0.18	0.2
5	0.41	0.09	0.25
6	0.05	0.54	0.29
7	0.6	0	0.3
8	0.17	0.01	0.09
9	-0.05	-0.07	-0.06
10	0.28	0.17	0.22
11	0.58	0.07	0.32
12	-0.19	-0.05	-0.12
13	0.41	0.18	0.29
14	0.25	0.18	0.21
15	0.06	0.21	0.13
16	0.82	0.24	0.53
17	0.17	0.18	0.17
$\sum$	**5.05**	**2.1**	**3.54**

14 lentelė. Gautų vertinimo rezultatų palyginimas. Stulpelio pavadinimas nurodo, kuris algoritmas buvo naudojamas vertinimui.

3. Galutinis rezultatas pagal algoritmą ir dėstytojo įvertinimą. Pagal gautus rezultatus stulpelyje „Galutinis.“ galima matyti, kad rezultatai yra taip pat skirtingi, ir algoritmas vertina prasčiau nei dėstytojas. Tai yra dėl tų pačių priežasčių, kurios yra įvardintos lyginant vertinimą pagal įvestą sakinį ir gautą atsakymą.

Pagal gautus galutinius rezultatus matoma, kad algoritmus dar reikia tobulinti ir atsižvelgti į lentelių naudojamus pirminius raktus, kad būtų galima gauti kaip įmanoma tikslesnį ir geresnį vertinimą. Šiuo metu nuokrypis tarp dėstytojo ir algoritmo galutinių rezultatų vertinimo yra pakankamai didelis.

4.4. UPDATE tipo sakinių testavimo rezultatai

UPDATE tipo sakinių analizei buvo naudoti 11-os studentų sakiniai. Kadangi sakiniai buvo rašomi naudojant *PostgreSQL* duomenų bazę, o algoritmo realizavime naudojamas *Microsoft SQL Server*, prieš vertinimą sakiniai buvo apdoroti rankiniu būdu, pritaikant sintaksę reikiamam duomenų bazės serveriui.

15 lentelė vaizduoja gautus rezultatų skirtumus po dėstytojo ir algoritmo vertinimo. Lentelės antraštėje skaičius nurodo, kelinta tai užduotis. Paskutinėje lentelės eilutėje pateikiama užduoties rezultatų suma. Jeigu rezultatas yra neigiamas, vadinasi algoritmo vertinimas buvo didesnis nei dėstytojo, o jeigu rezultatas yra teigiamas – dėstytojo vertinimas buvo didesnis. Pilni dėstytojo ir algoritmo vertinimo rezultatai pateikti D priede.

Kadangi gauti rezultatai labai skirtingi ir įvairūs – paanalizuokime gautus rezultatus pagal atskiras užduotis:

Stud. nr.	**I.**	**II.**	**III.**	**IV.**	**V.**
1	-0.22	0	-0.02	0.17	0
2	0.25	0	-0.14	0	0
3	-0.22	0.18	-0.52	-0.35	-0.85
4	-0.22	0.1	0	0.07	0.01
5	0.17	-0.02	-0.34	-0.45	0
6	-0.22	0	0.06	0.07	-0.21
7	0	0	-0.1	0.33	-0.19
8	0.22	0.18	-0.14	0.23	0
9	0.08	0.08	0.05	-0.13	0
10	-0.21	0.18	-0.14	-0.13	0
11	-0.28	0	-0.14	0.15	-0.19
$\sum$	**-0.95**	**0.7**	**-1.43**	**-0.14**	**-1.33**

15 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje skaičius nurodo, kuri užduotis yra vertinama.

1. I užduotis. Pagal dėstytojo vertinimus, studentai buvo įvertinti tik maksimaliu balu arba puse balo. Algoritmo vertinimai yra kiek įvairesni ir dauguma algoritmo vertinimų yra didesni nei dėstytojo vertinimų. Dėl to bendras užduoties vertinimas pagal algoritmą yra didesnis nei pagal dėstytoją. Nuokrypis vertinant užduotį gali būti didesnis, nes dėstytojas tikriausiai vertino užduotį tik pagal įvestą sakinį, tačiau nebuvo atsižvelgiama į gaunamą rezultatą.

2. II užduotis. Kaip matome, beveik pusė vertinimų tarp dėstytojo ir algoritmo vertinimų sutapo. Kiti sakiniai buvo panašūs ir įvertinti beveik vienodai, todėl nuokrypis tarp vertinimų nėra didelis.

3. III užduotis. Kaip ir pirmoje užduotyje taip ir šioje, algoritmas užduotis įvertino didesniais balais nei dėstytojas. Taip galėjo įvykti dėl to, kad sakiniai buvo pakankamai sintaksiškai panašūs, o gaunami rezultatai pagal įvestą sakinį taip pat teisingi arba beveik teisingi.

4. IV užduotis. Maksimaliai dėstytojo įvertintos užduotys nebuvo įvertintos maksimaliai algoritmo, nes užduotyje buvo naudojami sudėtingesni sudėtiniai sakiniai, kurie gali būti užrašomi daug skirtingų būtų. Nors atsakymai pagal studentus buvo visiškai teisingi, vertinamas sakinys nebuvo visiškai toks pats kaip dėstytojo. Kai kurie sakiniai, kurie buvo įvertinti dėstytojo nuliniu balu, algoritmo įvertinti ne nuliniu balu. Tai nutiko dėl to, kad nors ir sakinys nėra teisingas, kokio reikalauja užduotis, tačiau jame buvo paminėtų žodžių, kurie sutapo, taip pat galėjo būti gautas rezultatas, kurio dalis taip pat sutapo.

5. V užduotis. Bendras užduoties įvertinimas tarp studentų yra gerokai didesnis vertinant algoritmo nei vertinant dėtytojo. Kaip matome, tai yra dėl to, kad viena užduotis dėstytojo buvo įvertinta nuliniu balu, o algoritmas įvertino labai dideliu balu. Taip nutiko dėl to, kad, nors sakinys nebuvo teisingas, net nebuvo toks, kokio yra tikimasi, tačiau pagal jį buvo gautas kažkoks rezultatas, kurio dalis sutapo su patikrinimo rezultatu, o taip pat sakinyje buvo žodžių, kurie sutapo su dėstytojo sakinyje esančiais ir dėl to studentas gavo didesnį nei nulinį balą.

Kaip matome pagal gautus galutinius rezultatus, algoritmą dar reikėtų tobulinti. Algoritmas turėtų labiau atsižvelgti į sakinius, kurie gali neturėti jokios prasmės pagal atliekamą užduotį, kad būtų išvengiama didelių vertinimo neatitikimų. Šiuo metu nuokrypis tarp dėstytojo ir algoritmo vertinimų nėra labai didelis, tačiau daugeliu atveju algoritmo vertinimas yra didesnis nei dėstytojo, todėl tobulinant algoritmą reikėtų atsižvelgti į tai, kaip griežtai vertinimus atliko dėstytojas ir kaip griežtai tai turėtų daryti algoritmas.

4.5. DELETE tipo sakinių testavimo rezultatai

DELETE tipo sakinių analizei buvo naudoti 11-os studentų sakiniai. Kadangi sakiniai buvo rašomi naudojant *PostgreSQL* duomenų bazę, o algoritmo realizavime naudojamas *Microsoft SQL Server*, prieš vertinimą sakiniai buvo apdoroti rankiniu būdu, pritaikant sintaksę naudojamam duomenų bazės serveriui.

16 lentelė vaizduoja gautus rezultatų skirtumus po dėstytojo ir algoritmo vertinimo. Lentelės antraštėje skaičius nurodo, kelinta tai užduotis. Jeigu gaunamas rezultatas yra neigiamas, vadinasi algoritmo įvertinimas buvo didesnis nei dėstytojo, jeigu teigiamas – dėstytojo vertinimas buvo didesnis. Pilni dėstytojo ir algoritmo vertinimo rezultatai pateikti E priede.

Stud. nr.	**I.**	**II.**	**III.**	**IV.**
1	0	-0.03	0.24	0
2	0	-0.04	0	0
3	0.2	-0.47	-0.29	-0.84
4	0.11	0	0.08	0.03
5	-0.02	-0.3	-0.41	0
6	0	-0.04	0.08	-0.19
7	0	-0.05	0.37	-0.17
8	0.22	-0.1	0.15	0
9	0.08	0.06	-0.09	0
10	0.22	-0.1	-0.09	0
11	0	-0.1	0.19	-0.17
$\sum$	**0.83**	**-1.21**	**0.23**	**-1.24**

16 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje skaičius nurodo, kuri užduotis yra vertinama.

Kadangi gauti rezultatai yra pakankamai skirtingi, o atliekamos užduotys taip pat skirtingos, paanalizuokime gautus rezultatus pagal atskiras užduotis:

1. I užduotis. Kaip matome lentelėje, beveik pusė vertinimų sutapo. Nesutapę beveik visi vertinimai dėstytojo buvo įvertinti aukštesniais balais.

2. II užduotis. Šią užduotį algoritmas įvertino didesniais balais nei dėstytojas. Taip galėjo nutikti dėl to, kad dėstytojas vertino užduotis griežčiau nei algoritmas, dėl to algoritmo visi vertinimai yra didesni, išskyrus vieną.

3. III užduotis. Maksimaliai dėstytojo įvertintos užduotys nebuvo maksimaliai įvertintos algoritmo, nes sakiniai tarp dėstytojo ir stundentų nebuvo identiški, nors pasiektas atsakymas buvo identiškas. Kai kurie sakiniai, kurie buvo įvertinti dėstytojo nuliniais balais, algoritmo

buvo įvertinti ne nuliniu balu. Nors sakiniai ir nebuvo teisingi arba panašūs į reikalaujamus pagal užduotį, tačiau jie buvo pateikti ir buvo bent iš dalies panašūs į reikalingus, todėl algoritmas įvertino juos kažkokiu balu.

4. IV užduotis. Bendras užduoties vertinimas pagal algoritmą yra didesnis nei pagal dėstytojo vertinimus. Šis didelis nuokrypis gavosi dėl to, kad viena užduotis, kuri buvo įvertinta dėstytojo nuliniu balu, buvo įvertinta algoritmo arti maksimalaus balo. Taip galėjo nutikti dėl tos pačios priežasties kaip ir III užduotyje įvertintos užduotys, kurios algoritmo vertinime gavo didesnį nei nulinį balą, nors dėstytojo įvertinimas yra nulinis.

Kaip matome pagal galutinius gautus rezultatus, algoritmas yra pakankamai tikslus, imant bendrą visų studentų užduoties vertinimą. Tačiau algoritmą reikėtų šiek tiek patobulinti, atsižvelgiant į individualius variantus bei sudėtinius sakinius ir sąlygas, dėl kurių gaunasi nuokrypis, kurio neturėtų būti.

Išvados ir rekomendacijos

Šiame darbe aprašyti automatinio palyginimo ir vertinimo algoritmai turėtų palengvinti studentams ir dėstytojams DBVS mokymosi procesą, nes besimokantis gali greičiau gauti grįžtamąjį ryšį. Tai nebetrukdys mokymosi procesui, kai vertinimo yra laukiama labai ilgai ir dėl tos priežasties besimokantis nebesigilina į padarytas klaidas, nepriklausomai nuo to, kokį įvertinimą gavo. Tai reiškia, kad, vykstant automatizuotiems procesams, mokytojas galės daugiau laiko skirti besimokančiųjų individualioms konsultacijoms arba didesniam užduočių kiekio paruošimui.

Ištestavus algoritmus su realiais duomenimis, galima daryti išvadą, kad kai kurių tipų sakiniams algoritmas vis dar yra tobulintinas, nes tikslumas yra gaunamas per mažas, kad būtų galima aklai pasitikėti algoritmais. SELECT tipo sakiniai yra vertinami pakankamai tiksliai, tačiau, esant sudėtingesniems sakiniams, reikėtų didinti tikslumą, suteikiant galimybę įvesti kelis skirtingus teisingus atsakymus. CREATE sakinių tikslumas taip pat yra pakankamai didelis, tačiau taip pat tobulintinas. Vertinant CREATE tipo sakinius būtų naudinga įtraukti galimybę užduotis vertinti griežtai ir mažiau griežtai. INSERT tipo sakinių vertinimo tikslumas yra per mažas, nes sakiniai yra susiję su atitinkamais CREATE sakiniais, todėl, vertinant INSERT tipo sakinius, reikėtų labiau atsižvelgti į naudojamas duomenų bazės lenteles pagal CREATE tipo sakinius.

Vertinant gautą atsakymą pagal SELECT tipo sakinį yra gaunamas tam tikras nuokrypis, jeigu dėstytojo nurodytas tik vienas teisingas atsakymas, nors gali būti keli skirtingi atsakymai. Jeigu teisingas atsakymas yra, pavyzdžiui, pilno pavadinimo arba sutrumpinto pavadinimo gavimas, šiuo atveju kaip teisingas bus užskaitytas tik vienas atsakymas, kuris bus gaunamas kaip dėstytojo viename įvestame SQL sakinyje. Dėl to studentams taip pat mažėja įvertinimas, o tai neskatina studentų labiau stengtis ir gilintis, nes už teisingą atsakymą jie vis tiek negauna maksimalaus įvertinimo. Taip pat reikėtų patobulinti vertinimą pagal gautą atsakymą, kad būtų galima pateikti kelis skirtingus atsakymų variantus. CREATE tipo atsakymų vertinimas turi nedidelį nuokrypį, tačiau, kaip ir vertinant įvestus sakinius, būtų naudinga įtraukti galimybę vertinti griežtai arba mažiau griežtai. INSERT tipo atsakymų vertinimas nėra labai tikslus, nes labai daug įtakos daro įvestos reikšmės, kurios nėra konkrečiai apibrėžtos užduotyje. Dėl šios priežasties reikėtų užduotyje arba apibrėžti visas reikšmes, įskaitant ir pirminius raktus, arba vertinant gautą atsakymą, reikėtų nevertinti pirminių ir išorinių raktų reikšmių.

Ateities tyrimų planas

Ateityje algoritmai galėtų būti tobulinami keliomis kryptimis. Reikėtų išanalizuoti daugiau uždavinių ir pagal juos gaunamų atsakymų, kad būtų galima algoritmui pateikti konkretesnes, tikslesnes ir dažniau naudojamas taisykles, siekiant kaip įmanoma tikslesnio įvertinimo. Taip pat naudingas algoritmo papildymas būtų įtraukiant galimybę įvertinti studento įgyjamas kompetencijas.

Dar vienas naudingas patobulinimas – algoritmo pritaikymas skirtingoms duomenų bazėms. Šiuo metu algoritmai yra naudojami kartu su *Microsoft SQL Server*, tačiau gana dažnai mokymosi procese būna naudojama *PostgreSQL* ar *Oracle* duomenų bazė. Dėl šios priežasties, norint algoritmus naudoti universaliai, ne tik vienoje platformoje, reikėtų juos pritaikyti ir kitoms duomenų bazių struktūroms. Norint algoritmus pritaikyti skirtingoms duomenų bazėms, reikėtų išanalizuoti kitų duomenų bazių naudojamas sistemines lenteles ir pagal jas pritaikyti algoritmo dalį, kurioje yra naudojamas sisteminių lentelių tikrinimas.

Kadangi įvesti SQL sakiniai yra saugomi duomenų bazėje, į sistemą taip pat būtų galima įtraukti ir algoritmus, skirtus plagiatų nustatymui. Taip galėtų būti užstikrintas mokymosi skaidrumas ir galimybė įvertinti individualų pačio studento darbą.

Kitas naudingas algoritmos patobulinimas – automatinis vieno įvesto sakinio perdarymas į kitus panašius sakinius, kad būtų galima gauti kaip įmanoma daugiau skirtingų variantų ir kaip įmanoma tiksliau įvertinti studento įvestus sakinius. Pavyzdžiui, tai būtų galima pritaikyti INSERT tipo sakiniams. Algoritmas galėtų sugeneruoti iš vieno sakinio kelis panašius sintaksiškai teisingus sakinius.

Vertinant atsakymus arba sakinius taip pat būtų naudinga įtraukti ne tokį griežtą vertinamą ir vertinti ne pagal idealiai atitinkančius žodžius, o pagal žodžių panašumą. Pavyzdžiui, jeigu studentas įvedė žodį lotyniškomis raidėmis, o dėstytojas tą patį žodį pateikia naudodamas lietuviškas raides, studento žodis turėtų būti vertinamas kaip teisingas, jeigu vertinimas nėra griežtas ir reikalauja bent panašaus žodžio, bet nebūtinai idealiai tokio pačio.

Literatūros šaltiniai

[1] Agnė Čeponkutė. Aplinka DBVS praktinių gebėjimų ugdymui. Bakalauro baigiamasis darbas, 2016.

[2] Bikash Chandra, Mathew Joseph, Bharath Radhakrishnan, Shreevidhya Acharya, and S. Sudarshan. Partial Marking for Automated Grading of SQL Queries. *PVLDB*, 9(13):1541–1544, 2016.

[3] Stijn Dekeyser, Michael de Raadt, and Tien Yu Lee. Computer Assisted Assessment of SQL Query Skills. In *Database Technologies 2007. Proceedings of the Eighteenth Australasian Database Conference, ADC*, pages 53–62, 2007.

[4] Gordon Hunter, David Livingstone, Paul Neve, and Graham Alsop. Learn Programming++: The Design, Implementation and Deployment of an Intelligent Environment for the Teaching and Learning of Computer Programming. In *The 9th International Conference on Intelligent Environments*, pages 129–136, 2013.

[5] INFOBALT. INFOBALT naujienos, 2017. https://www.infobalt.lt/lt/naujienos/i/940, žiūrėta 2017-09-30.

[6] INFOBALT Lietuva. Lietuvos IRT specialistų pasiūlos ir paklausos analizė, 2017. https://www.infobalt.lt/lt/naujienos/i/940, žiūrėta 2017-09-30.

[7] Haifeng Ke, Gaoyan Zhang, and Hui Yan. Automatic Grading System on SQL Programming. In *International Conference on Scalable Computing and Communications / Eighth International Conference on Embedded Computing, ScalCom-EmbeddedCom*, pages 537–540, 2009.

[8] Carsten Kleiner, Christopher Tebbe, and Felix Heine. Automated grading and tutoring of SQL statements to improve student learning. In *13th Koli Calling International Conference on Computing Education Research*, pages 161–168, 2013.

[9] Antonija Mitrovic. A Knowledge-Based Teaching System for SQL. In ED-MEDIA 98, 1998.

[10] Antonija Mitrovic. An Intelligent SQL Tutor on the Web. *I. J. Artificial Intelligence in Education*, 13(2-4):173–197, 2003.

[11] Antonija Mitrovic and Brent Martin. Evaluating Effectiveness of Feedback in SQL-Tutor. Advanced Learning Technologies. IWALT, 2000.

[12] Vreda Pieterse. Automated Assessment of Programming Assignments. In *Proceedings of the 3rd Computer Science Education Research Conference, CSERC*, pages 45–56, 2013.

[13] Julia Coleman Prior and Raymond Lister. The backwash effect on SQL skills grading. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pages 32–36, 2004.

[14] Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, and Tapio Salakoski. VILLE – A Language-Independent Program Visualization Tool. Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), 2007.

[15] Gordon Russell and Andrew Cumming. Improving the Student Learning Experience for SQL Using Automatic Marking. In *Cognition and Exploratory Learning in Digital Age (CELDA'04), Proceedings of the IADIS International Conference*, pages 281–288, 2004.

[16] Shazia Wasim Sadiq, Maria E. Orlowska, Wasim Sadiq, and Joe Y.-C. Lin. SQLator: an online SQL learning workbench. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pages 223–227, 2004.

[17] Mario Soflano, Thomas M. Connolly, and Thomas Hainey. An application of adaptive games-based learning based on learning style to teach SQL. *Computers & Education*, 86:192–211, 2015.

[18] Ivan Stajduhar and Goran Mausa. Using string similarity metrics for automated grading of SQL statements. In *38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO*, pages 1250–1255, 2015.

[19] Elektroninių studijų ir egzaminavimo centras. Vilniaus universiteto virtuali mokymosi aplinka, 2017.
http://moodle.esec.vu.lt/.

Priedai

Dokumentą sudaro penki priedai: A priede pateikti SELECT tipo sakinių vertinimo rezultatai, B priede pateikti CREATE tipo sakinių vertinimo rezultatai, C priede pateikti INSERT tipo sakinių vertinimo rezultatai, D priede pateikti UPDATE tipo sakinių vertinimo rezultatai, E priede pateikti DELETE tipo sakinių vertinimo rezultatai.

A. SELECT tipo sakinio vertinimo rezultatai

Stud. nr.	**I.D.**	**I.A.**	**II.D.**	**II.A**	**III.D.**	**III.A.**	**IV.D.**	**IV.A.**	**V.D.**	**V.A.**
1	0.7	0.64	0.3	0.28	0.3	0.21	0	0.46	0	0.17
2	1	1	0.7	0.49	0.8	0.71	0	0	0	0
3	0.7	0.72	0.5	0.17	0.2	0.18	0	0.38	0	0.12
4	0.7	0.64	0.7	0.56	0.7	0.43	0.5	0.42	0	0
5	1	0.97	0.9	0.35	1	0.71	1	0.9	0.7	0.62
6	1	0.91	1	0.39	0.7	0.44	1	0.91	0.9	0.81
7	1	1	0.7	0.71	0.9	0.3	0.8	0.62	0.7	0.63
8	0.8	0.72	1	0.41	0.7	0.44	0.3	0.4	0	0
9	1	1	0.6	0.42	0.7	0.36	0.6	0.37	0	0
10	1	1	0.7	0.49	0.7	0.27	0.6	0.6	0.7	0.71
11	0.7	0.64	0.7	0.56	1	0.74	1	0.85	0.1	0
$\sum$	**9.6**	**9.24**	**7.8**	**4.81**	**7.7**	**4.76**	**5.8**	**5.87**	**3.1**	**3.05**

17 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis skaičius nurodo, kuri tai yra užduotis, o raidės nurodo, kas vertino užduotį. D – vertino dėstytojas, A – algoritmas.

B. CREATE tipo sakinio vertinimo rezultatai

Nr.	Atr.D.	Atr.A.	PK.D.	PK.A.	FK.D.	FK.A.	Ck.D.	Ck.A.	CO.D.	CO.A.
1	1	0.93	1	0.89	0	0	1	0.2	1	1
2	1	1	1	1	0	0	1	0.27	1	1
3	1	0.94	0.5	0.33	0.75	0.6	0.25	0.13	1	0.5
4	0.75	0.28	1	0.22	0.75	0	0	0	1	0
5	1	1	1	1	1	0.8	0	0	1	1
6	1	0.96	0.5	0.39	0.5	0.2	0	0	1	1
7	1	1	1	1	0	0	0	0	0	0
8	0.75	0.94	1	1	1	1	1	0.57	1	1
9	0.75	0.81	0	0	0	0	0	0	1	1
10	1	0.98	1	0.89	0.75	0.65	0	0	0.5	0.5
11	0.75	1	0.5	0.56	0.75	0.95	1	0.27	1	0.5
12	0.75	0.63	1	0.56	0.75	0.4	0	0	0.5	0
13	1	0.81	1	1	0	0	1	0.27	1	1
14	0.75	0.93	1	1	0.75	0.6	0	0	1	1
15	1	0.89	1	1	0.75	0.75	0	0	1	1
16	1	0.96	0.5	0.39	0.75	0.8	0	0	1	0.5
17	1	0.67	1	0.56	0.75	0.4	0	0	1	0
$\sum$	**15.5**	**14.7**	**14**	**11.79**	**9.25**	**7.15**	**5.25**	**1.71**	**15**	**11**

18 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis trumpinys nurodo, kuri sritis yra vertinama, o raidės nurodo, kas vertino užduotį. D – vertino dėstytojas, A – algoritmas.

Nr.	Sakinys.A.	Atsakymas.A.	Galutinis.A.	Galutinis.D.
1	3.36	3.22	3.29	5
2	5.11	3.54	4.33	5
3	5.09	3.23	4.16	4.5
4	3.71	0.5	2.11	4.25
5	5.68	4.6	5.14	5
6	3.29	2.75	3.02	3.5
7	5.11	2	3.56	2
8	5.81	6.08	5.95	6.75
9	3.99	1.31	2.65	1.25
10	2.95	4.17	3.56	4.5
11	6.13	4.5	5.32	5.75
12	5.32	1.99	3.66	3.75
13	3.85	3.35	3.6	5
14	4.34	4.13	4.24	4.25
15	5.79	4.39	5.09	4.5
16	4.74	3.45	4.10	4
17	6.13	2.03	4.08	4
$\sum$	**80.4**	**55.24**	**67.82**	**73.5**

19 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis žodis nurodo, kuris algoritmas buvo naudojamas vertinimui, o raidės nurodo, kas vertino užduotį. D – vertino dėstytojas, A – algoritmas.

C. INSERT tipo sakinio vertinimo rezultatai

Nr.	Sakinys.A.	Atsakymas.A.	Galutinis.A.	Galutinis.D.
1	0.71	0.83	0.77	1
2	0.69	0.83	0.76	1
3	0.67	0.79	0.73	1
4	0.17	0.93	0.55	0.75
5	0.59	0.91	0.75	1
6	0.7	0.21	0.46	0.75
7	0.4	1	0.7	1
8	0.83	0.99	0.91	1
9	0.25	0.27	0.26	0.2
10	0.72	0.83	0.78	1
11	0.17	0.68	0.43	0.75
12	0.69	0.55	0.62	0.5
13	0.59	0.82	0.71	1
14	0.75	0.82	0.79	1
15	0.94	0.79	0.87	1
16	0.18	0.76	0.47	1
17	0.83	0.82	0.83	1
$\sum$	**9.88**	**12.83**	**11.39**	**14.93**

20 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis žodis nurodo, kuris algoritmas buvo naudojamas vertinimui, o raidės nurodo, kas vertino užduotį. D – vertino dėstytojas, A – algoritmas.

D. UPDATE tipo sakinio vertinimo rezultatai

Nr.	I.A.	I.D.	II.A.	II.D.	III.A.	III.D.	IV.A.	IV.D.	V.A.	V.D.
1	0.72	0.5	1	1	0.62	0.6	0.43	0.6	0	0
2	0.25	0.5	1	1	0.84	0.7	0	0	0	0
3	0.72	0.5	0.52	0.7	0.82	0.3	0.35	0	0.85	0
4	0.72	0.5	0.9	1	1	1	0.93	1	0.89	0.9
5	0.67	0.5	0.72	0.7	0.84	0.5	0.45	0	0	0
6	0.72	0.5	1	1	0.84	0.9	0.93	1	0.91	0.7
7	1	1	1	1	0.8	0.7	0.47	0.8	0.89	0.7
8	0.28	0.5	0.52	0.7	0.84	0.7	0.87	1	0	0
9	0.92	1	0.72	0.8	0.95	1	0.43	0.3	0	0
10	0.71	0.5	0.52	0.7	0.84	0.7	0.63	0.5	0	0
11	0.78	0.5	1	1	0.84	0.7	0.45	0.6	0.89	0.7
$\sum$	**7.45**	**6.5**	**8.9**	**9.6**	**9.23**	**7.8**	**5.94**	**5.8**	**4.43**	**3.1**

21 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis skaičius nurodo, kuri užduotis yra vertinama, o raidės nurodo, kas vertino užduotį. D – vertino dėstytojas, A – algoritmas.

E. DELETE tipo sakinio vertinimo rezultatai

Nr.	I.A.	I.D.	II.A.	II.D.	III.A.	III.D.	IV.A.	IV.D.
1	1	1	0.62	0.59	0.36	0.6	0	0
2	1	1	0.84	0.8	0	0	0	0
3	0.48	0.7	0.77	0.3	0.29	0	0.84	0
4	0.89	1	1	1	0.92	1	0.87	0.9
5	0.72	0.7	0.8	0.5	0.41	0	0	0
6	1	1	0.84	0.8	0.92	1	0.89	0.7
7	1	1	0.8	0.75	0.43	0.8	0.87	0.7
8	0.48	0.7	0.8	0.7	0.85	1	0	0
9	0.72	0.8	0.94	1	0.39	0.3	0	0
10	0.48	0.7	0.8	0.7	0.59	0.5	0	0
11	1	1	0.8	0.7	0.41	0.6	0.87	0.7
$\sum$	**8.77**	**9.6**	**9.01**	**7.8**	**5.57**	**5.8**	**4.34**	**3.1**

22 lentelė. Gautų vertinimo rezultatų palyginimas. Lentelės antraštėje pirmasis skaičius nurodo, kuri užduotis yra vertinama, o raidės nurodo, kas vertino užduotį. D – vertino dėstytojas, A – algoritmas.