VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ KATEDRA

JavaScript objektų užtikrinimas duomenų bazių valdymo sistemose

Ensuring JavaScript Object Notation in database management systems

Bakalauro darbas

Atliko:	Viktor Ševaldin	(parašas)
Darbo vadovas:	prof. dr. Romas Baronas	(parašas)
Darbo recenzentas:	lekt. Viktoras Golubevas	(parašas)

Vilnius – 2016

SANTRAUKA

Šiame bakalauro darbe pristatoma duomenų bazių valdymo sistemų (DBVS) galimybių ir priemonių vertinimo metodika JavaScript objektų (JSON) užtikrinimo atžvilgiu. Aprašoma, kas yra JSON objektai, kur naudojami JSON objektai, kokiomis priemonėmis daugiausiai naudojamos DBVS užtikrina JSON, kokios galimybės yra suteikiamos.

Išskiriami DBVS priemonių vertinimo kriterijai, aprašoma, kokie yra suteikiamų galimybių ypatumai, į ką reikia atkreipti dėmesį vertinant DBVS vienu ar kitu aspektu. Remiantis metodika, galima įvertinti vieną ar keletą DBVS skirtingais aspektais. Pateikiamas praktinis sukurtos metodikos taikymas vertinant SQL Server 2016 CTP3 DBVS. Remiantis vertinimu, daroma išvada, kad SQL Server 2016 CTP3 suteikia didžiąją dalį galimybių, nurodytų vertinimo metodikoje ir sukurtą DBVS vertinimo JSON užtikrinimo atžvilgių metodiką galima taikyti vertinant DBVS JSON užtikrinimo priemones.

Raktiniai žodžiai: JSON, JavaScript Object Notation, DBVS, duomenų bazių valdymo sistema, vertinimo metodika.

SUMMARY

This bachelor's thesis presents database management systems (DBMS) assessment methodology on evaluating JavaScript Object Notation (JSON) assurance level. The thesis describes what JSON is, usage of JSON objects, the means most popular DBMSs provide ensuring JSON and options available.

The assessment criteria for DBMS are defined, the specifics of available features are presented: what you need to consider when evaluating the DBMS by one aspect or another. An evaluation of one or more DBMS by different criteria can be performed according to the methodology. The practical application of the methodology is presented assessing the SQL Server 2016 CTP3 DBMS. According to the assessment, the conclusion that SQL Server 2016 CTP3 DBMS provides the majority of features mentioned in the assessment methodology and that the presented methodology can be applied assessing DBMS JSON assurance level is made.

Keywords: JSON, JavaScript Object Notation, DBMS, database management system, assessment methodology.

TURINYS

ĮVADAS 6
1. JSON OBJEKTAI 8
1.1. JSON duomenų formatas 8
1.2. JSON taikymai 8
1.3. JSON užtikrinimo priemonių apžvalga kai kuriuose DBVS 9
1.4. JSON samprata 10
2. JSON UŽTIKRINIMAS DUOMENŲ BAZIŲ VALDYMO SISTEMOSE 11
2.1. Tipizavimas 11
2.2. Paieškos JSON objektuose galimybės 11
2.2.1. JSON kelio reiškiniai 12
2.2.2. Paieška pasinaudojant SQL užklausomis 12
2.2.3. Tekstinė paieška 12
2.3. Funkcijų darbui su JSON galimybės 12
2.3.1. JSON objektų sukūrimo funkcijos 13
2.3.2. JSON objektų apdorojimo funkcijos 13
2.3.3. Paieškos JSON objektuose funkcijos 13
2.4. Indeksavimo kriterijus 13
3. VERTINIMO METODIKA 15
3.1. Vertinimo kriterijai 15
3.1.1. Tipizavimo kriterijus 15
3.1.2. Paieškos JSON objektuose kriterijus 15
3.1.2.1. JSON kelio reiškinių vertinimo ypatumai 15
3.1.2.2. Paieška pasinaudojant SQL užklausomis ir tekstinė paieška 16
3.1.3. Funkcijų darbui su JSON kriterijus 16
3.1.4. Indeksavimo kriterijus 17
3.2. Vertinimo lentelė 17
3.3. Vertinimo eiga 19
4. METODIKOS TAIKYMAS 20
4.1. Tipizavimas 20
4.2. Paieškos JSON objektuose galimybės 21
4.3. Funkcijų darbui su JSON galimybės 22
4.4. Indeksavimas 23
REZULTATAI IR IŠVADOS 25
Rezultatai 25

Išvados 25
ŠALTINIAI 26

ĮVADAS

Duomenų perdavimas ir saugojimas informacinių technologijų pasaulyje yra viena iš pagrindinių užduočių, kuriai sukurtas ne vienas sprendimas. Duomenis tenka perduoti tarp įvairių programų sistemų, loginių programų sluoksnių. Konkretaus duomenų perdavimo būdo ir formato pasirinkimą lemia daug veiksnių: duomenų paskirtis, sudėtingumas, poreikiai apsaugoti duomenis juos šifruojant, išvengti klaidų, atsirandančių perdavimo metu, poreikis suspausti duomenis ir kiti. Vieni svarbiausių faktorių – duomenų formato užtikrinimas programavimo kalbose (palengvina programavimą), duomenų apdorojimo ir perdavimo greičiai (įtakoja greitaveiką), duomenų skaitomumas (palengvina programavimą), kompaktiškumas (reikalauja mažiau atminties ir laiko perduodant duomenis).

Yra keletas pasirinkimų, labiausiai paplitę: XML, JSON, Yaml ir kiti. Pastaruoju laiku, būtent JavaScript objektai (JSON) tampa vis labiau populiaresniu būdu, tiksliau – formatu, perduoti duomenis [Duv13]. Laikoma, kad šis formatas yra kompaktiškesnis, to pasėkoje greičiau perduodamas, užima mažiau vietos saugant, ir lengviau skaitomas žmogui nei XML [Ben13].

Kadangi duomenis tenka ne tik perduoti, bet ir saugoti, iškilo poreikis tokį (JSON) duomenų formatą užtikrinti ir duomenų bazių valdymo sistemose. Saugant ir operuojant duomenimis DBVS, reliacinių duomenų vientisumui ir korektiškumui užtikrinti naudojamos DBVS suteikiamos priemonės. Pavyzdžiui, reikalavimai duomenims (angl. *constraint*), trigeriai (angl. *trigger*), įvairus duomenų tipai. Paieškos patogumui naudojama struktūrizuota užklausų kalba (angl. *SQL*), greitesnei paieškai - indeksavimas. Tam, kad pilnaverčiai užtikrinti JSON objektų saugojimą ir apdorojimą, siekiama šiam duomenų formatui suteikti tokias pačias priemones DBVS, kas leistų ne tik saugoti, bet ir operuoti duomenimis.

Šiuo metu, JSON užtikrinimo vystymas tapo viena iš DBVS tobulinimo krypčių: palaipsniui atsiranda naujos galimybės darbui su JSON DBVS lygyje. Tai galime matyti iš, pavyzdžiui, Oracle, PostgreSQL DBVS paskutinių versijų dokumentacijų [Ada14], [The15]. Taip pat, dokumentacijos analizė, atlikta kursinio darbo metu, parodė, kad kiekviena DBVS yra savotiškai skirtinga JSON užtikrinimo atžvilgiu. Žinoma, kad DBVS pasirinkimą įtakoja daug veiksnių, tačiau nėra visiškai aišku, kokios galimybės turi egzistuoti DBVS, maksimaliai geram JSON objektų užtikrinimui. Todėl būtų tikslinga sukurti DBVS galimybių vertinimo metodiką, nes remiantis ja galima būtų įvertinti keletą DBVS ir pasirinkti tinkamiausią.

Remiantis JSON samprata, paskirtimi ir savybėmis [Bra14] ir kelių vienų populiariausių [Sol15] (Oracle Database 12c, PostgreSQL 9.4, MySQL 5.7.7) duomenų bazių valdymo sistemų

JSON užtikrinimo priemonių analize, atlikta kursinio darbo metu, metodikai sukurti, tikslinga sudaryti galimybių rinkinį, išskirti kriterijus, pagal kuriuos bus vertinamos DBVS ir pateikti kelių DBVS pavyzdinį vertinimą. Kursiniame darbe panašus vertinimas buvo atliktas primityviai: lyginant DBVS galimybes tarpusavyje. Remiantis DVBS dokumentacijomis buvo aprašytos JSON užtikrinimo priemones, išskirtos priemonių bendros tendencijos.

Šio bakalauro darbo tikslas – parengti ir pritaikyti DBVS vertinimo metodiką JSON užtikrinimo atžvilgiu.

Darbo uždaviniai:

- parinkti JSON užtikrinimo vertinimo kriterijus;
- sukurti DBVS JSON užtikrinimo galimybių vertinimo sistemą remiantis kriterijais;
- sudaryti DBVS JSON užtikrinimo vertinimo remiantis sudarytais kriterijais metodiką;
- atlikti eksperimentinį sukurtos metodikos taikymą vertinant konkrečias DBVS.

1. JSON OBJEKTAI

JSON (JavaScript Object Notation) – tai yra tekstinis duomenų apsikeitimo formatas, apibrėžtas RFC7159 standartu (kuris patikslina ankstesnius ECMA-404 ir ECMA-262 standartus) [Bra14]. JSON atsirado 1999 – 2000 metais, esant realaus laiko tarp serverio ir naršyklės bendravimo poreikiui, be Flash ir Java įskiepių įsiterpimo. Šis duomenų formatas skirtas atvaizduoti duomenų struktūras ir masyvus.

JSON gali būti ir yra naudojamas kaip XML (eXtensible Markup Language) alternatyva, nes yra lengviau skaitomas žmogui, yra lakoniškesnis [Ben13]. Kaip ir XML, JSON gali būti aprašytas schema (angl. *JSON Schema*). Nors JSON buvo apibrėžtas JavaScript programavimo kalbos kontekste, jis nepriklauso nuo jokios programavimo kalbos ir yra palaikomas daugybėje programavimo kalbų. Pilnas sąrašas programavimo kalbų ir atitinkamai bibliotekų, skirtų darbui su JSON pateikiamas svetainėje www.json.org.

1.1. JSON duomenų formatas

JSON naudoja dvi duomenų struktūras:

1. Porų „raktas: reikšmė“ rinkinį (angl. *attribute – value*), programavimo kalbose apibrėžiamą objektu. Raktas apibrėžiamas eilute, tarp rakto ir reikšmės rašomas „:“ simbolis;
2. Reikšmių rinkinį, programavimo kalbose apibrėžiamą masyvu.

Reikšmės gali būti šių duomenų rūšių:

1. Skaičius, jeigu yra ir trupmeninė dalis, ji atskiriama tašku;
2. Eilutė;
3. Loginis tipas;
4. Masyvas - reikšmių aibė, apskliausta „[ ]“, tame pačiame masyve reikšmės gali būti skirtingų tipų, reikšmės atskiriamos kableliu;
5. Objektas – aibė porų, apskliaustu „{ }“, porų eiliškumas - nesvarbus;
6. Tušti reikšmė (null).

Kaip matome, struktūros gali būti įdedamos viena į kitą naudojant masyvus ir objektus. Pagal aukščiau minėtus standartus, JSON duomenų formato koduotė yra UTF8 [Bra14].

1.2. JSON taikymai

Dažniausiai JSON naudojamas duomenų persiuntimui tarp serverio ir naršyklės, naudojantis AJAX (asinchroninis JavaScript ir XML) technologija [Azi07]. Šios technologijos esmė –

dinaminis duomenų apsikeitimas, kurio dėka nereikia perkrauti internetinio puslapio, norint atnaujinti arba atvaizduoti papildomą informaciją.

Trumpas pavyzdinis veikimo aprašymas: JavaScript, veikiantis naršyklėje, generuoja AJAX užklausa ir išsiunčia ją serveriui. Serveris, savo ruožtu, suformuoja JSON eilutę ir išsiunčia ją naršyklei. JavaScript, gavęs atsakymą, atvaizduoja reikiamus duomenis puslapyje. Kadangi JSON yra JavaScript kalbos poaibis, JavaScript leidžia greitai JSON eilutę paversti į JavaScript kalbos objektą ir toliau naudoti programoje [Len14].

JSON yra plačiai naudojamas interneto-paslaugų srityje. JSON formatu perduodami duomenys, kuriuos gali apdoroti programa, interneto-paslaugos klientas. Vietoje JSON taip pat naudojamas ir XML, tačiau su XML perduoti tą patį kiekį informacijos reikalauja daugiau resursų (laiko, atminties), nes XML turi daugiau metaduomenų. Tai skatina naudotis JSON mobiliuosiuose įrenginiuose, kur būna ribojami mobilieji duomenis [Ben13].

JSON objektai yra tiesiogiai palaikomi ir, pavyzdžiui, *Objective-C* programavimo kalboje: JSON objektą galima be papildomo apdorojimo konvertuoti į šiuos kalbos duomenų tipus: *NSDictionary* ir *NSArray*. XML duomenų atveju tektų pasinaudoti papildomais kitų klasių metodais, kas ir vėl reikalautų daugiau resursų. Toks tiesioginis palaikymas leidžia sparčiau apdoroti duomenis [Ben13].

1.3. JSON užtikrinimo priemonių apžvalga kai kuriuose DBVS

Analizuojant vienas iš populiariausių DBVS (Oracle Database 12c, PostgreSQL 9.4, MySQL 5.7.7) [Sol15] JSON užtikrinimo atžvelgiu, pastebėti kai kurie priemonių realizacijų panašumai, tendencijos. Taip, pavyzdžiui, siekiant užtikrinti JSON duomenų korektiškumą naudojami tokie sprendimai kaip reikalavimai stulpeliams (angl. *constraint*) arba specialiai JSON formato duomenims skirti duomenų tipai. Kai kurios DBVS, pavyzdžiui PostgreSQL, leidžia rinktis iš dvejų duomenų tipų: *json* ir *jsonb*, vienas iš jų skirtumų yra tai, kad *json* atveju duomenys saugomi kaip tiksli įvedimo kopija, o *jsonb* – konvertuojami į dvejetainį formatą, juos leidžiama indeksuoti ir apdoroti naudojant funkcijas ir operatorius [The15]. Suteikiamos galimybės naudotis kelio reiškiniais arba operatoriais, kas leidžia atlikti žingsnius pačiame JSON objekte, prieiti prie konkrečių JSON reikšmių DBVS pagalba. Įvairios funkcijos leidžia atlikti paieškos, reikšmių patikrinimo, reikšmių išrinkimo, JSON objekto konvertavimo į reliacinį formatą (tolimesnei paieškai arba kitiems veiksmams SQL pagalba), JSON objekto duomenų įterpimo, atnaujinimo, pašalinimo ir kitas operacijas. Skirtingos indeksavimo galimybės pagreitina paieškos, objektų palyginimo veikimą.

1.4. JSON samprata

Remiantis JSON paskirtimi, jo savybėmis [Bra14] šiame poskyriuje aprašysime, kokie veiksmai gali būti atliekami su JSON objektais. Į tai reikės atsižvelgti sudarant metodiką, nes turėti funkcionalumą, įprastą darbui su JSON yra patogu ir DBVS lygyje.

JSON objektą, sudarytą iš porų raktas – reikšmė, galime traktuoti kaip „duomenų saugyklą“, todėl natūralu, kad:

- į JSON galime įdėti naujas arba šalinti poras;
- pasiekti JSON porų reikšmes
- keisti porų reikšmes;

Jei poros reikšmė yra masyvas, jam turėtų būti taikytini veiksmai su masyvais: elementų pridėjimas į masyvą, šalinimas iš masyvo, elementų išrinkimas pagal indeksą. Elementų išrinkimas taip pat yra taikytinas ir pačiam JSON objektui: nurodant poros raktą, ar kelią iki poros.

Žiūrint į JSON, kaip į duomenų saugyklą, gali būti taikomi tokie veiksmai:

- patikrinti, ar objektas savyje turi porą su ieškomu pavadinimu (paprasta paieška);
- dviejų JSON objektų apjungimas.

Iš DBVS perspektyvos, su JSON patogu atlikinėti įprastus, kaip ir su duomenų bazėje saugomais duomenimis veiksmus, paversti JSON duomenis į reliacinį formatą ir atlikti paieškos arba duomenų atnaujinimo operacijas SQL pagalba. Tai yra leidžiama, pavyzdžiui, Oracle 12c [Ada14], PostgreSQL 9.5.2 [The16]. JSON pavertimu į reliacinį formatą ir atvirkščiai – lentelės įrašų transformacija į JSON galima naudotis ir paprastesnei DBVS integracijai su tinklo paslaugomis (angl. *web services*), kur dažnai naudojamas JSON duomenų apsikeitimo formatas.

Tolimesniuose skyriuose bus plačiau apžvelgiamos JSON užtikrinimo galimybes nagrinėtose DBVS.

2. JSON UŽTIKRINIMAS DUOMENŲ BAZIŲ VALDYMO SISTEMOSE

Detaliai susipažinus su galimybėmis darbui su JSON komercinėje (Oracle) ir nekomerciniuose (PostgreSQL ir MySQL) DBVS [Ada14, The15, Hil15], galime išskirti bendras JSON užtikrinimo sritis, tai yra bendra funkcionalumą, galimybes JSON atžvilgiu skirtingose DBVS:

- Tipizavimas;
- Paieškos JSON objektuose galimybės;
- Funkcijų darbui su JSON galimybės;
- Indeksavimas.

Atkreipkime dėmesį, kad išvardinti punktai pilnai apima funkcionalumą, skirtą JSON užtikrinimui DBVS.

2.1. Tipizavimas

Pagrindinis tipizavimo tikslas – užtikrinti tik atitinkančių JSON standartą duomenų saugojimą duomenų bazėje. Tai gali būti užtikrinama skirtingais būdais. Pavyzdžiui, DBVS gali turėti atskirą duomenų tipą arba tipus JSON objektams. Taip yra, pavyzdžiui, PostgreSQL ir MySQL DBVS [The15, Hil15]. Priklausomai nuo tipo pasirinkimo, DBVS gali papildomai apdoroti JSON duomenis spartesnei paieškai, arba leisti naudotis tam tikromis JSON apdorojimo funkcijomis. Kitas tipizavimo užtikrinimo būdas – reikalavimai stulpeliams, kuriuose saugomi tik JSON objektai. Reikalavimai stulpeliams (tiksliau – funkcija, kuria naudoja reikalavimas stulpeliui) tikrina JSON formato duomenys įrašymo į DB metu, o tipai leidžia duomenis indeksuoti, saugoti kitu formatu ar koduote greitesniam apdorojimui ar paieškai. Galima laikyti patogumu galimybę rinktis iš kelių duomenų tipų su skirtingomis savybėmis, kaip, pavyzdžiui, PostgreSQL DBVS [The15], nes tai padeda optimizuoti DBVS veikimą su tam tikros paskirties duomenimis: kartais gali prireikti duomenis indeksuoti, kartais - išsaugoti pradinę duomenų koduotę.

2.2. Paieškos JSON objektuose galimybės

Šį aspektą galima išskaidyti į tris dalis: paieška pasinaudojant JSON kelio reiškiniais, paieška pasinaudojant SQL užklausomis, tekstinė paieška. Funkcijos, skirtos paieškai, aprašomos 2.3 poskyriuje, kartu su kitomis funkcijomis. Toks skaidymas pagrindžiamas tuo, kad tai yra visa aibė priemonių, skirtų paieškai JSON objektuose DBVS, aptikta kursinio darbo metu, analizuojant populiariausias DBVS.

2.2.1. JSON kelio reiškiniai

JSON kelio reiškiniai gali būti naudojami skirtingais būdais, pavyzdžiui, pasinaudojant taško notacija SELECT sakiniuose arba kelio reiškiniu naudojant specialiomis funkcijomis. Naudojant taško notaciją, kas atrodo, pavyzdžiui, taip: SELECT json_stulpelis.json_laukas FROM json_lenta, reikia žinoti pilną kelią, tai yra kiekvieną poros raktą. Taško notacija atsirado JavaScript kalboje, ja įprasta naudotis operuojant JSON duomenimis. Kelio reiškiniuose kai kurios DBVS leidžia nerašyti kiekvienos poros rakto pavadinimo, o vietoje to pasinaudoti „*“ simboliu. Kelio reiškinių pagalba galima naviguoti per masyvo elementus, išrinkti keletą masyvo elementų vienu metu. Reikėtų atkreipti dėmesį ir į kitas papildomas galimybes kaip, pavyzdžiui, MySQL DBVS teikiamą galimybę surasti specifinius kelius pasinaudojant „**“ žingsniu nurodant tik pradinį ir galutinį poros raktą, tai yra, nereikia rašyti kelią iš visų raktų iki konkrečios poros reikšmės.

Taigi galime išvardinti, kuo gali išsiskirti JSON kelio reiškinių funkcionalumas:

- Taško notacija SELECT sakiniuose;
- Nebūtinas pilno kelio (išvardinant visus porų raktus) nurodymas;
- Specifinių kelių paieška;
- Galimybė išrinkti keletą masyvo elementų žingsnyje.

2.2.2. Paieška pasinaudojant SQL užklausomis

DBVS būna realizuota galimybė atlikti paiešką JSON objekte jį konvertuojant į reliacinį formatą pasinaudojant funkcijomis. Skirtumai tarp šio funkcionalumo realizacijos tarp įvairių DBVS yra tame, kiek veiksmų reikia atlikti konvertuojant duomenis. Pavyzdžiui, PostgreSQL atveju papildomai tenka nurodyti lentelės tipus, Oracle DBVS atveju užtenka iškviesti funkcija su JSON objekto argumentu.

2.2.3. Tekstinė paieška

Patogi yra galimybė atlikti tekstinę paiešką JSON dokumente: galima tuo pasinaudoti kaip sąlyga atrenkant reikalingus duomenis SQL užklausa. Papildomai gali būti galimybė sužinoti kelio reiškinį arba reiškinius, nurodantį, kur randasi ieškoma tekstinė reikšmė JSON objekte.

2.3. Funkcijų darbui su JSON galimybės

DBVS funkcijos darbui su JSON suteikia naudotojams galimybę atlikti reikiamus veiksmus su JSON objektais DBVS lygyje. Funkcijas galima išskirti į keletą tipų: JSON objektų sukūrimo, JSON objektų apdorojimo, paieškos JSON objektuose funkcijos.

2.3.1. JSON objektų sukūrimo funkcijos

Sukūrimo funkcijos yra labiau būdingos DBVS, kur JSON duomenų saugojimui naudojami specialus tipai. Dažniausiai, jų paskirtis – įvairių tipų duomenys paversti į JSON skirtam tipui duomenis. Tokios funkcijos skiriasi JSON duomenų struktūros atžvilgiu, tai yra: funkcijų rinkinys gali leisti kurti ne tik paprastus JSON objektus, bet ir masyvus, daugiamačius masyvus. Kai kurios DBVS, pavyzdžiui, PostgreSQL 9.5.2 [The16] turi galimybę kurti JSON objektus iš tekstinių duomenų (masyvų) lentų įrašų.

2.3.2. JSON objektų apdorojimo funkcijos

Apdorojimo funkcijų paskirtis – atlikti duomenų pašalinimo, įrašymo, atnaujinimo operacijas JSON objektuose. Reikia atkreipti dėmesį į tai, ar yra suteikiamos funkcijų pagalba arba kitais būdais priemonės redaguoti duomenis, tai yra, įrašyti naują reikšmę ar porą į JSON objektą ar masyvą, ištrinti porą ar reikšmę, atnaujinti poros ar masyvo elemento reikšmę. Gali būti galimybę vienu funkcijos iškvietimu redaguoti keletą JSON objekto porų ar reikšmių, pavyzdžiui, nurodant keletą kelio reiškinių.

Prie apdorojimo funkcijų galima priskirti ir validacijos funkcijas, patikrinančias, ar argumentas yra taisyklingas JSON objektas.

2.3.3. Paieškos JSON objektuose funkcijos

Paieška JSON objektuose gali būti atliekama įvairiai. Vienas iš būdų – paieškos funkcijos. Jos gali patikrinti, ar JSON objektas turi savyje nurodyta JSON objektą, gali būti ir galimybė nurodyti, kokiu keliu ieškoti. Gali būti galimybė sužinoti reikšmės nurodytu keliu tipą, ištraukti JSON objekto arba nurodytu keliu esančio elemento raktus. Taip pat čia priskirkime įvairias pagalbines funkcijas kaip, pavyzdžiui, funkciją, grąžinančią masyvo ilgį arba reikšmės elementų skaičių.

2.4. Indeksavimo kriterijus

Pagrindinė indeksavimo paskirtis – pagreitinti duomenų paiešką ir išrinkimą vykdant SQL užklausas. Taip pat indeksavimas gali būti naudojamas duomenų stulpelyje unikalumui užtikrinti.

JSON atžvilgiu indeksavimas gali būti taikomas efektyvesniam funkcijų veikimui - funkcijos paremtas indeksavimas (angl. *function-based indexing*)) – tokiu būdu leidžiama indeksuoti tam tikrą kelio reiškinį (tiksliau, reikšmes, esančias tuo keliu), kuris naudojamas kviečiant pačią funkciją. Toks indeksavimas naudingas, jei funkcija, naudojanti JSON kelio reiškinį yra dažnai kviečiama.

Tekstinei duomenų paieškai gali būti taikomas ir įprastas *bitmap* indeksavimas, jei indeksuojamų reikšmių aibė nėra didelė (visos reikšmės nėra unikalios) arba B-medžių indeksavimas (angl. *B-tree*). Tekstiniai paieškai taip pat gali būti taikomas ir specialus JSON paieškos indeksavimas (angl. *JSON search index*): jo tikslas – leisti tekstine duomenų paieška su tam tikra funkcija, kuriai nurodomas JSON objektas, kelio reiškinys ir ieškoma eilutė. Pastarasis indeksavimas yra realizuotas, pavyzdžiui, Oracle 12c DBVS.

Naudingas gali būti ir kelių reikšmių indeksavimas. Jis gali būti naudojamas tada, kai WHERE dalyje SELECT sakinio ieškoma pagal keletą suindeksuotų stulpelių.

3. VERTINIMO METODIKA

3.1. Vertinimo kriterijai

Antrame skyriuje pateiktos JSON užtikrinimo sritys padengia visas galimybes darbo su JSON, aptiktas nagrinėtose DBVS. Todėl vertinimo kriterijai gali susidaryti pagal tokį skirstymą. Remiantis kriterijais galėsime įvertinti DBVS suteikiamas galimybes. Atsižvelgiant į kiekvieno kriterijaus specifiką, šiame poskyriuje aprašysime, į ką reikėtų atkreipti dėmesį analizuojant DBVS galimybes.

3.1.1. Tipizavimo kriterijus

Kadangi tipizavimas pagrinde skirtas tik atitinkančių JSON standartą duomenų saugojimui užtikrinti, būtų pakankama, kad bent koks nors tipizavimo būdas JSON duomenims DBVS būtų suteikiamas. Turėdami bent vieną tipizavimo užtikrinimo būdą, galime būti tikri, kad saugosime tik korektiškus duomenis, kas leis naudotis visomis suteikiamomis priemonėmis darbui su jais be klaidų dėl duomenų formato neatitikimo. Todėl skirtingas tipizavimą užtikrinančias galimybes vertinsime vienodai.

Galimybė reikalavimuose stulpeliams nurodyti kelią JSON objekte galime vertinti papildomai. Jo pagalba galima būtų tikrinti norimą sąlygą. Jo panaudojimas, pavyzdžiui, gali būti toks: neleisti saugoti JSON objektų, kurių pora su tam tikrų pavadinimu (esančia tame tikrame kelyje) turi tuščią reikšmę. Tai pasiekti galėtumėme ir kitu būdu: JSON duomenų įrašymui arba atnaujinimui turėdami funkciją, kuri prieš įrašant duomenis juos tikrintų. Tačiau tai reikalauja papildomo laiko kūrimui ir gali apsunkinti duomenų bazės palaikymą.

3.1.2. Paieškos JSON objektuose kriterijus

Vertinant paieškos JSON objektuose galimybes reikėtų atkreipti dėmesį į tai, kad neužtenka apsiriboti vien tokiais kriterijais, kaip JSON kelio reiškiniai, paieška pasinaudojant SQL užklausomis ir tekstinė paieška. Jeigu DBVS suteikia skirtingas priemones paieškai JSON objektuose, tai turėtų būti įvertinta atsižvelgiant į naudotojo poreikius.

3.1.2.1. JSON kelio reiškinių vertinimo ypatumai

Skirtingos DBVS suteikia skirtingas galimybes darbui su JSON kelio reiškiniais. Kai kurios papildomos galimybės turi būti įvertintos atskirai, pavyzdžiui:

- Privalumu laikysime galimybė pasinaudoti „*“ simboliu JSON kelyje vietoje kiekvienos poros pavadinimo vardinimo.

- Navigacijos per masyvo elementus, kelių masyvo elementų išrinkimą vienu metu irgi laikysime privalumu lyginant su galimybe išrinkti tik vieną masyvo elementą su viena užklausa, nes tai sumažina užklausų kiekį į DBVS, kas teigiamai įtakoja duomenų apdorojimo greitį, palengviną programavimą.
- Galimybė pasinaudoti „**“ žingsniu nurodant tik pradinį ir galutinį poros raktą (nereikia rašyti kelio iš visų raktų), vertinsime teigiamai, kadangi kuo daugiau skirtingų galimybių paieškai yra įgyvendintą DBVS lygyje, tuo efektyviau (greičiau) jie veiks lyginant su kitais architektūriniais programų sistemos sluoksniais.

Be abejo, skirtingose DBVS tos pačios galimybės gali būti realizuotos skirtingai. Pavyzdžiui, lyginant Oracle ir PostgreSQL JSON kelio reiškinius, pastebime, kad pirmu atveju kelio reiškinys atrodo: „$.JsonRaktas1.raktas2“, o antru – „JsonObjektas#>>'{JsonRaktas1,raktas2}'“, bet abiejų reiškinių rezultatas yra tas pats, taikant juos tam pačiam JSON objektui. Taigi kelio reiškinių, nesant keleto masyvo elementų išrinkimo, galimybės vienodos ir vertinamos vienodai, o jau išrinkti keleto masyvo elementų su PostgreSQL kelio reiškiniais nepavyks. Be to, PostgreSQL JSON kelio reiškiniai reikalauja po kiekvieno paieškos žingsnio JSON objekte turėti tik vieną reikšmę, kas apriboja naudotojo paieškos galimybes.

3.1.2.2. Paieška pasinaudojant SQL užklausomis ir tekstinė paieška

Teigiamai vertinama paieškos SQL užklausomis, konvertuojant JSON duomenis į reliacinį formatą, galimybė. Vertinimas turi priklausyti tik nuo to, kiek veiksmų reikia atlikti konvertuojant duomenis. Kuo mažiau papildomų veiksmų, tuo vertinimas didesnis.

Tekstinės paieškos galimybė irgi vertinama teigiamai, papildomai vertinama galimybė sužinoti kelio reiškinį iki poros, kurioje randamas ieškomas tekstas, arba pasinaudoti sudėtingesnėmis (pavyzdžiui, reguliariosios išraiškos) priemonėmis tekstiniai paieškai.

3.1.3. Funkcijų darbui su JSON kriterijus

Akivaizdu, kad kuo daugiau yra galimybių apdoroti duomenis DBVS priemonėmis, tuo geriau. Didžioji dalis galimų šio kriterijaus funkcionalumų aprašyta 2.3 skyrelyje.

JSON objektų sukūrimo funkcijas vertinsime JSON duomenų struktūros atžvilgiu: jei funkcijų rinkinys leidžia kurti ne tik paprastus JSON objektus, bet ir masyvus, arba net daugiamačius masyvus, tokios DBVS šio kriterijaus vertinimas turi būti didesnis.

Apdorojimo funkcijos arba kitos JSON apdorojimo priemones turi leisti redaguoti JSON duomenis. Privalumu laikysime galimybę vienu funkcijos iškvietimu redaguoti keletą JSON objekto porų ar reikšmių, pavyzdžiui, nurodant keletą kelio reiškinių.

Prie paieškos funkcijų galima priskirti ir kitas panašaus pobūdžio, 2.3.3 poskyriuje neišvardintas funkcijas, ir atsižvelgti į tai vertinant DBVS. Kiekviena tokio pobūdžio papildoma ar pagalbinė funkcija bus laikoma privalumu, tik nereikėtų pamiršti apie tai, kad jeigu DBVS leidžia tą patį rezultatą pasiekti kitais būdais, reikalaujančiais apytiksliai tiek pat resursų, pavyzdžiui, išreikšti kokią nors funkciją kitomis, DBVS vertinime turi būti skiriama tiek pat arba šiek tiek mažiau taškų.

3.1.4. Indeksavimo kriterijus

Vertinant indeksavimo galimybes JSON atžvilgiu, reikėtų atkreipti dėmesį į tai, kad kartais duomenų indeksavimas galimas tik su papildomomis sąlygomis, pavyzdžiui indeksavimas taikomas tik virtualiems stulpeliams arba reikalaujama tam tikro stulpelio tipo. Tais atvejais, kai to paties tipo indeksavimas vienoje DBVS suteikiamas su mažesniu kiekiu papildomų sąlygų nei kitoje, pirmoji DBVS turi būti įvertinta geriau.

3.2. Vertinimo lentelė

Vertinimo lentelę sudarys kiekvieno kriterijaus punktai ir atitinkamai taškų kiekis, skiriamas už punkto realizaciją vertinamojoje DBVS. Visų pirmą nurodysime taškų sumą už kiekvieną kriterijų, o ją išdalinsime tarp skirtingų galimybių. Taip pat, nurodysime papildomus taškus nenumatytoms metodikoje galimybėms įvertinti.

Kiekvienam kriterijui įvertinti, galime skirti taškus. Kuo daugiau taškų surenkama už kiekvieną kriterijų, tuo geresniu galime laikyti JSON užtikrinimą to kriterijaus atžvilgiu.

Tipizavimo kriterijų įvertinkime 15 taškų, nes jo galimybių nėra tiek daug, kiek, pavyzdžiui, paieškos arba funkcijų kriterijuje. Daugiausia taškų skirkime paieškos ir funkcijų kriterijams: atitinkamai 40 ir 30 taškų, kadangi tai yra pagrindiniai manipuliavimo JSON duomenimis įrankiai. Indeksavimui – 20 taškų.

1 lentelė. JSON užtikrinimo galimybių vertinimo lentelė

Suteikiama galimybė	Taškai
Tipizavimas (viso 15 taškų):	
• Tipizavimo užtikrinimas reikalavimu stulpeliams	5
o Galimybė reikalavimuose stulpeliams nurodyti kelią JSON objekte	3

Suteikiama galimybė	Taškai
• Tipizavimo užtikrinimas duomenų tipu	5
• Galimybė rinktis iš keleto duomenų ar reikalavimų stulpeliams tipų	2
Funkcijos (viso 30 taškų):	
• JSON objekto sukūrimas iš kokio nors tipo duomenų	2
o iš duomenų su masyvais	1
o iš duomenų su daugiamačiais masyvais	2
o iš lentos įrašų	2
• JSON objekto naujų porų pridėjimas	2
o Porų šalinimas	2
o Poros reikšmės keitimas	2
• Elementų pridėjimas į JSON objekto masyvą	2
• Galimybė redaguoti keletą JSON objekto porų ar reikšmių vienu metu	2
• JSON objekto validacija	2
• Galimybė patikrinti, ar JSON objektas turi savyje kitą JSON objektą	2
• Galimybė sužinoti reikšmės nurodytu keliu tipą	2
• Galimybė ištraukti JSON objekto arba nurodytu keliu esančio elemento raktus	2
• Galimybė suskaičiuoti masyvo elementus	1
• Kitos aukščiau nenumatytos funkcijos	Iki 4
Paieška (iki 40 taškų):	
• JSON kelio reiškinių palaikymas	7
o Galimybė žingsnyje nenurodyti poros pavadinimo	3
o Galimybė nurodyti tik pradinį ir galutinį porų pavadinimus	3
o Galimybė išrinkti keletą masyvo elementų žingsnyje	4
• Taško notacijos palaikymas SELECT sakiniuose	5
• Galimybė JSON formato duomenys paversti į reliacinį formatą	6
• Galimybė atlikti tekstinę paiešką JSON objekte	5
o Galimybė sužinoti kelią, kur buvo rasta tekstinė reikšmė	3
• Kitos aukščiau nenurodytos galimybės	Iki 4
Indeksavimas (iki 20 taškų):	
• Funkcijos paremto indeksavimo palaikymas	4
• *B-tree* indeksavimo palaikymas	3

Suteikiama galimybė	Taškai
• *Bitmap* indeksavimo palaikymas	3
• Kelių reikšmių indeksavimas	4
○ Nereikalauja papildomų sąlygų išpildymo	2
• Kiti indeksavimo būdai	Iki 4

Įvertinus vieną DBVS, matysime kiek gerai yra užtikrinamas vienas arba kitas aspektas, remiantis surinktų taškų kiekiu lyginant su maksimaliu kriterijaus kiekiu. Įvertinus keletą DBVS galėtumėme lyginti kaip taškų kiekius pagal kriterijus, taip ir bendras sumas.

3.3. Vertinimo eiga

Atlikta Oracle 12c, PortgreSQL 9.5.4, MySQL 5.7 DBVS dokumentacijų [Ada14, The16, Hil15] analizė parodė, kad skirtingose DBVS tos pačios galimybės užtikrinamos šiek tiek skirtingais būdais. Pavyzdžiui, naudojami skirtingi operatoriai navigacijai JSON objektuose, skiriasi panašaus pobūdžio funkcijų pavadinimai ir jų argumentai, skiriasi indeksavimo būdai ir reikalavimai jiems. Taigi, neišeina kitaip apibendrinti (pavyzdžiui, sudaryti "universalias" SQL užklausas, kurios tiktų bet kokiai DBVS patikrinti) kriterijų tikrinimą, nei dokumentacijos nagrinėjimas ir paieška išvardintų kriterijų užtikrinančių priemonių, nes kiekviena DBVS yra savotiškai skirtinga JSON užtikrinimo atžvilgiu. Aišku, nereikia analizuoti visos dokumentacijos, dažniausiai JSON užtikrinimo priemonės aprašomos atskiruose skyriuose.

4. METODIKOS TAIKYMAS

Šiame skyriuje įvertinsime SQL Server 2016 CTP (Community Technology Preview) 3, remiantis šiuo metu pateikiama dokumentacija [Mic16a, Mic16b, Mic16c, Mic16d] ir kitais šaltiniais [Pop16a, Pop16b, Pop16c, Pop16d]. Vertinimą atliksime remiantis metodiką, aprašysime kiekvieno aspekto vertinimą iš eilės.

4.1. Tipizavimas

SQL Server 2016 DBVS neturi atskiro duomenų tipo JSON objektams, duomenis saugomi tekstinio tipo NVARCHAR stulpeliuose. Tipizavimo kriterijų užtikrina reikalavimas stulpeliams, kuriame galima pasinaudoti ISJSON funkcija. Pavyzdys:

```
CREATE TABLE Duomenys (
 Id int IDENTITY PRIMARY KEY NONCLUSTERED,
 Pavadinimas nvarchar(100),
 Json nvarchar(max)
 );
```

Sukuriama lentelė, kurios stulpelis „Json“ yra tekstinio tipo. Šiam stulpeliui pritaikysime reikalavimą, kad jame būtų saugomi tik JSON tipo duomenys:

```
ALTER TABLE Duomenys
ADD CONSTRAINT [JSON tikrinimas] CHECK (ISJSON(Json)>0);
```

Funkcija ISJSON gražina 1, jei jos argumentas yra JSON objektas, priešingu atveju - 0. Jei argumentas yra NULL reikšmė, funkcija grąžina NULL. Taigi, DBVS užtikrina JSON duomenų tipizavimą reikalavimais stulpeliais. Metodikoje dar nurodėme, kad papildomai vertinsime galimybę patikrinti duomenis nurodytu kelio reiškiniu, ką galime atlikti nagrinėjamoje DBVS, pavyzdžiui, taip:

```
ALTER TABLE Duomenys
ADD CONSTRAINT CHECK [JSON poros tikrinimas] (ISJSON(Json)>0 AND
JSON_VALUE(Json, '$.Flag') = 1);
```

Čia pasinaudojome funkcija JSON_VALUE, kuri grąžina JSON objekto poros reikšmė nurodytu kelio reiškiniu. Kelio reiškiniai leidžia išrinkti vieną masyvo elementą žingsnyje, žingsniai atliekami rašant „.“, pradžioje nurodomas „$“, reiškiantis šakninį elementą. Kiti tipizavimo būdai neaprašyti dokumentacijoje. Kitų papildomų galimybių, aprašytų metodikoje, kelio reiškiniai neturi. Į juos atsižvelgsime vertindami atitinkamai paieškos kriterijų.

Už tipizavimo suteikiamas galimybes DBVS skiriami 5+3=8 taškai.

4.2. Paieškos JSON objektuose galimybės

Nagrinėjama DBVS palaiko JSON kelio reiškinius, bet jokių iš papildomai išvardintų galimybių nėra [Mic16b]. Kelio reiškinius galima nurodyti funkcijose JSON_VALUE ir JSON_QUERY. Pirmoji skirta skaliarinėms reikšmėms ištraukti, o antroji – JSON objektams arba masyvams. Jų argumentai – JSON objektas ir kelio reiškinys. Nėra palaikoma taško notacija SELECT sakiniuose, kaip, pavyzdžiui, tai leidžia daryti Oracle 12c DBVS: `SELECT json.flag FROM Duomenys json;`.

SQL Server 2016 CTP3 turi galimybę paversti JSON duomenis į reliacinį formatą [Mic16c]. Tai užtikrinama OPENJSON funkcija. Jos pagalba galima paversti JSON formato duomenis į reliacinį formatą tolimesniam apdorojimui SQL užklausa arba įrašymui į lentą. Pavyzdžiui, apibrėžkime JSON objektą taip:

```
declare @json nvarchar(max) = '{
"id" : 2,
"vardas": "Viktor",
"pavarde": "Ševaldin",
"dirba": true,
"amzius": 22,
"gimimoData": "1994-03-15T15:45:00"
}';
```

Turėdami lentą „Zmogus“:

```
CREATE TABLE Zmogus (
Id int,
vardas nvarchar(100),
pavarde nvarchar(100),
dirba bit,
amzius int,
gimimoData datetime2);
```

Tokiu atveju, norėdami įrašyti duomenis į lentą „Zmogus“, turime įvykdyti sekantį sakinį:

```
INSERT INTO Zmogus
SELECT *
FROM OPENJSON(@json)
WITH (id int,
vardas nvarchar(50), pavarde nvarchar(50),
dirba bit, amzius int,
gimimoData datetime2);
```

JSON objekto porų reikšmės įsirašo į atitinkamus lentos stulpelius. Čia pasinaudojome OPENJSON funkcija, leidžiančia JSON duomenis konvertuoti į reliacinį formatą ne tik įrašymui į lentas, bet ir paieškai su SQL užklausa.

Tarp dokumentacijoje išvardintų funkcijų nėra atskiros funkcijos paieškai JSON objekte, bet, kadangi JSON objektai saugomi tekstiniu formatu, jose galima atlikti tekstinę paieška naudojant, pavyzdžiui, DBVS funkcijas CONTAINS arba FREETEXT. Kadangi atskirų funkcijų paieškai JSON nėra, nėra ir galimybės sužinoti kelią JSON objekte, kur rasta ieškoma reikšmė.

Vertindami paieškos kriterijų, suteiksime taškus už JSON kelio reiškinių palaikymą, galimybę JSON formato duomenys paversti į reliacinį formatą ir galimybė atlikti tekstinę paiešką JSON objekte: 7+6+5=18 taškų.

4.3. Funkcijų darbui su JSON galimybės

Kadangi nagrinėjama DBVS neturi atskiro tipo JSON duomenims, JSON objektų sukūrimas realizuotas tik iš lentelės įrašų. Tai daroma su FOR JSON PATH reiškiniu. Papildomai galime nurodyti keletą parametrų pasirinktinai: ROOT leidžia nurodyti išorinio JSON objekto rakto pavadinimą, WITHOUT_ARRAY_WRAPPER leidžia nuimti išorinio masyvo skliaustus („[]“), INCLUDE_NULL_VALUES leidžia rezultate gauti ir tuščias reikšmes (jei eilutės stulpelio reikšmė DB yra null), kurios JSON objekte pasižymės kaip „null“. Pavyzdžiui, turėdami sekančius duomenys lentelėje „Zmogus“:

	Id	vardas	pavarde	dirba	amzius	gimimodata
1	1	Petras	Petraitis	1	25	1991-05-01 00:00:00.0000000
2	2	Viktor	Ševaldin	1	22	1994-03-15 15:45:00.0000000
3	3	Studentas	Mifietis	0	20	1996-04-22 00:00:00.0000000

1 pav. Lentelės "Zmogus" duomenys

Įvykdžius `SELECT * FROM Zmogus FOR JSON PATH, ROOT ('Zmones');` sakinį, gausime sekantį JSON obektą (pateikiamas nesuformatuotas):

{ "Zmones":[{ "Id":1, "vardas":"Petras", "pavarde":"Petraitis", "dirba":true, "amzius":25, "gimimodata":"1991-05-01T00:00:00" }, { "Id":2, "vardas":"Viktor", "pavarde":"Ševaldin", "dirba":true, "amzius":22, "gimimodata":"1994-03-15T15:45:00" }, { "Id":3, "vardas":"Studentas", "pavarde":"Mifietis", "dirba":false, "amzius":20, "gimimodata":"1996-04-22T00:00:00" }] }

Funkcija JSON_MODIFY leidžia modifikuoti JSON objektą: keisti porų reikšmes, pridėti naujas poras, šalinti poras, pridėti elementus į masyvą. Kadangi JSON_MODIFY funkcijos rezultatas yra atnaujintas JSON objektas, kviesdami ją iš savęs rekursiškai gauname galimybę keisti keletą JSON objekto porų vienu metu. Pavyzdžiui, turėdami JSON objektą:

```
DECLARE @json NVARCHAR(100) = '{"vardas":"Studentas",
"dalykai":["DBVS","Java"]}'
```

Atnaujinama poros su raktu „vardas“ reikšmė:

```
SET @json = JSON_MODIFY(@json, '$.vardas', 'Viktor')
```

Pridedama nauja pora:

```
SET @json = JSON_MODIFY(@json, '$.pavarde', 'Ševaldin')
```

Ištrinama pora:

```
SET @json = JSON_MODIFY(@json, '$.vardas', NULL)
```

Pridedama reikšmė į masyvą „dalykai“:

```
SET @json = JSON_MODIFY(@json, 'append $.dalykai', 'Pascal')
```

Vertindami funkcijų darbui su JSON galimybes, suteiksime taškus už JSON sukūrimą iš lentos įrašų, JSON objekto porų pridėjimą, šalinimą, reikšmių keitimą, elementų pridėjimą į masyvą, galimybę keisti keletą JSON objekto porų vienu metu, galimybę patikrinti, ar JSON objektas yra validus (validuoti JSON objektą leidžia aukščiau aprašyta funkcija ISJSON): 2+2+2+2+2+2+2=14 taškų.

4.4. Indeksavimas

Kadangi SQL Server 2016 CTP3 neturi JSON duomenų tipo, jokių specifinių indeksavimo galimybių JSON objektams nėra. Duomenų paieškai pagal JSON objekto poros reikšmę pagreitinti galima suindeksavus virtualų stulpelį, kurio reikšmės būtų JSON objekto poros reikšmės. Virtualaus stulpelio reikšmės nurodomos JSON kelio reiškiniu, pavyzdžiui:

```
ALTER TABLE Duomenys ADD vPavarde AS JSON_VALUE(Json,
'$.pavarde');
```

Po virtualaus stulpelio sukūrimo jam pritaikomas indeksas:

```
CREATE INDEX PavardesIndeksas ON Duomenys(vPavarde);
```

Reikėtų atkreipti dėmesį, kad naujai sukurtas stulpelis iš tikrųjų nėra saugomas (neužima papildomos vietos lentelėje), o išskaičiuojamas tik kai indeksas turi būti atnaujintas. Kad indeksas būtų naudojamas, svarbu pačioje užklausoje nurodyti tokį patį JSON kelio reiškinį (šiuo atveju $.pavarde).

Toks indeksas atitinka metodikoje nurodytą „funkcijos paremto indeksavimo palaikymas“ galimybę. Už tai DBVS suteikiami 4 taškai. Kitų indeksavimo būdų SQL Server 2016 CTP3 neturi.

Taigi, gavome tokį DBVS vertinimą remiantis metodiką:

- Tipizavimo kriterijus: 8 iš 15 taškų;
- Funkcijų darbui su JSON galimybės: 14 iš 30 taškų;
- Paieškos JSON objektuose galimybės: 18 iš 40 taškų;
- Indeksavimas: 4 iš 20 taškų.

Nors daugelių atvejų DBVS nesurinko daugiau nei pusės vertinimo, galime daryti išvadą, kad suteikiamų galimybių pakanka nesudėtingiems veiksmams su JSON objektais atlikti. Taip pat verta paminėti, kad taikant metodiką DB valdymo sistemai, kurios JSON užtikrinimo galimybėmis nebuvo remtasi kuriant metodiką, visos vertinamos DBVS galimybės buvo padengtos sukurta metodika, kas leidžia daryti išvadą, kad sukurtą metodiką galima taikyti praktikoje.

REZULTATAI IR IŠVADOS

Rezultatai

Šio bakalauro darbo rezultatai yra:

- parinktų JSON užtikrinimo DBVS galimybių vertinimo sistema;
- DBVS vertinimo JSON užtikrinimo atžvilgių metodika;
- SQL Server 2016 CTP3 DBVS JSON užtikrinimo vertinimas remiantis metodika.

Išvados

Remiantis darbo rezultatais, galima daryti tokias išvadas:

- taikant metodiką SQL Server 2016 CTP3 DBVS, kurios JSON užtikrinimo galimybėmis nebuvo remtasi kuriant pačią metodiką, visos vertinamos DBVS galimybės buvo padengtos išskirtais kriterijais;
- SQL Server 2016 CTP3 suteikia didžiąją dalį galimybių, nurodytų vertinimo metodikoje;
- sukurtą DBVS vertinimo JSON užtikrinimo atžvilgių metodiką galima taikyti vertinant DBVS JSON užtikrinimo priemones.

ŠALTINIAI

[Ada14] Drew Adams. Oracle XML DB Developer‘s Guide, 12c Release 1 (12.1). [Žiūrėta 2015-02-20] Prieiga per internetą: <http://docs.oracle.com/database/121/ADXDB/json.htm>

[Azi07] Atif Aziz, Scott Mitchell, An Introduction to JavaScript Object Notation (JSON) in JavaScript and .NET, [Žiūrėta 2016-04-27] Prieiga per internetą: <https://msdn.microsoft.com/en-us/library/bb299886.aspx>

[Ben13] Geoff Bender. XML vs JSON Based Web Services: Which is the Best Choice? [Žiūrėta 2015-03-27] Prieiga per internetą: <http://www.seguetech.com/blog/2013/04/01/xml-vs-json-web-services-best-choice>

[Bra14] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. [Žiūrėta 2015-02-26] Prieiga per internetą: <http://www.rfc-editor.org/rfc/rfc7159.txt.>

[Duv13] Adam DuVander. JSON’s Eight Year Convergence With XML. [Žiūrėta 2015-04-28] Prieiga per internetą: <http://www.programmableweb.com/news/jsons-eight-year-convergence-xml/2013/12/26>

[Hil15] Rick Hillegas, Evgeny Potemkin, Dag Wanvik, Knut Anders Hatlen, Alfredo Kojima. Server-side JSON Functions (Functional Spec). [Žiūrėta 2015-06-13] Prieiga per internetą: <http://dev.mysql.com/worklog/task/?id=7909 >

[Len14] Jason Lengstorf. JSON: What It Is, How It Works, & How to Use It. [Žiūrėta 2015-02-27] Prieiga per internetą: <http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/>

[Mic16a] Microsoft Developer Network (MSDN), SQL Server 2016 Technical Documentation [Žiūrėta 2016-05-01] Prieiga per internetą: <https://msdn.microsoft.com/en-us/library/dn921897.aspx>

[Mic16b] Microsoft Developer Network (MSDN), SQL Server 2016 Technical Documentation, JSON Path Expressions (SQL Server) [Žiūrėta 2016-05-01] Prieiga per internetą: <https://msdn.microsoft.com/en-us/library/mt577087.aspx>

[Mic16c] Microsoft Developer Network (MSDN), SQL Server 2016 Technical Documentation, Convert JSON Data to Rows and Columns with OPENJSON (SQL Server) [Žiūrėta 2016-05-01] Prieiga per internetą: <https://msdn.microsoft.com/en-us/library/dn921879.aspx>

[Mic16d] Microsoft Developer Network (MSDN), SQL Server 2016 Technical Documentation, JSON_MODIFY (Transact-SQL) [Žiūrėta 2016-05-01] Prieigą per internetą: <https://msdn.microsoft.com/en-us/library/dn921892.aspx>

[Pop16a] Jovan Popovic, JSON in SQL Server 2016: Part 1 of 4, [Žiūrėta 2016-05-02] Prieiga per internetą: <https://blogs.technet.microsoft.com/dataplatforminsider/2016/01/05/json-in-sql-server-2016-part-1-of-4/>

[Pop16b] Jovan Popovic, JSON in SQL Server 2016: Part 2 of 4, [Žiūrėta 2016-05-02] Prieiga per internetą: <https://blogs.technet.microsoft.com/dataplatforminsider/2016/01/06/json-in-sql-server-2016-part-2-of-4/>

[Pop16c] Jovan Popovic, JSON in SQL Server 2016: Part 3 of 4, [Žiūrėta 2016-05-02] Prieiga per internetą: <https://blogs.technet.microsoft.com/dataplatforminsider/2016/01/07/json-in-sql-server-2016-part-3-of-4/>

[Pop16d] Jovan Popovic, JSON in SQL Server 2016: Part 4 of 4, [Žiūrėta 2016-05-02] Prieiga per internetą: <https://blogs.technet.microsoft.com/dataplatforminsider/2016/01/08/json-in-sql-server-2016-part-4-of-4/>

[Sol15] solid IT gmbh. DB-Engines Ranking. [Žiūrėta 2015-06-10] Prieiga per internetą: <http://db-engines.com/en/ranking>

[The15] The PostgreSQL Global Development Group. PostgreSQL 9.4.0 Documentation. [Žiūrėta 2015-04-27] Prieiga per internetą: <http://www.postgresql.org/docs/9.4/static/index.html>

[The16] The PostgreSQL Global Development Group. PostgreSQL 9.5.2 Documentation. [Žiūrėta 2016-04-27] Prieiga per internetą: <http://www.postgresql.org/docs/9.5/static/index.html>