SQL

Romas Baronas

Duomenų bazių valdymo sistemos

2. Informacijos išrinkimas

Turinys

Pratarmė 1
2.1. Duomenų bazė „Darbai“ 2
2.2. Sakinys SELECT 4
2.3. Paprasčiausios užklausos duomenims išrinkti 4
2.4. Užklausos rezultato rikiavimas 7
2.5. Reiškiniai 8
2.6. Lentelių jungimas 12
2.7. Struktūrinės užklausos 15
2.8. Laikinieji pavadinti užklausų rezultatai 16
2.9. Bendrumo ir egzistavimo kvantoriai užklausose 17
2.10. Duomenų grupavimas 19
2.11. Aibių operacijos 22
2.12. Lentelių reikšmių konstruktorius 23
2.13. Sąlyginiai reiškiniai 24
2.14. Rekursyviosios užklausos 26
2.15. Papildomos paieškos galimybės 28
2.16. Schemos 30
2.17. Sisteminis katalogas 31
2.18. Pratimai 33
Literatūra 34

Pratarmė

Ši mokomoji medžiaga (skyrius) yra leidykloje TEV išleistos knygos „Duomenų bazių valdymo sistemos“ (R. Baronas) pataisytas ir papildytas 2 skyrius [1]. Čia aptariamos SQL priemonės duomenų išrinkimui ir analizei. Pradedama paprasčiausiomis užklausomis duomenų, esančių vienoje ar keliose lentelėse, išrinkimu. Vėliau aptariamos pakankamai sudėtingos duomenų analizės užklausos. Šio skyriaus pabaigoje nagrinėjama duomenų paieška sisteminiame kataloge. Medžiaga parengta pagal plačiai žinomas mokomąsias knygas [2, 4, 6, 9, 14, 15, 18] bei internete laisvai prieinamą medžiagą. SQL sakinių sintaksė ir praktiniai pavyzdžiai yra pateikiami prisilaikant DBVS PostgreSQL reikalavimų [3, 5, 16,

17]. Pažymėtina, kad čia pateikiamos ne visos SQL sakinių ypatybės ir galimybės, bet tik esminės sakinių prasmei pažinti ir suvokti. Sistemos PostgreSQL specifinės SQL savybės yra trumpai aptariamos lyginant jas su kitomis DBVS ir tarptautiniu SQL standartu [7, 8, 12, 13]. Priminsime, kad originaliojoje knygoje [1] SQL sakinių sintaksė ir pavyzdžiai pateikti pagal DBVS IBM DB2 reikalavimus [2]. Gilesniam ir platesniam duomenų išrinkimo ir analizės SQL priemonėmis pažinimui ir įgūdžių įgijimui galima pasitelkti specifiškai tam skirtus šaltinius [10, 11].

2.1. Duomenų bazė „Darbai“

Su duomenų bazės duomenimis kur kas dažniau susipažįstama (jie tik peržiūrimi) nei jie atnaujinami ar įvedami. Dar rečiau ir tik nedaugeliui vartotojų (DB administratoriams ir taikomųjų sistemų kūrėjams) tenka kurti lenteles ir duomenų bazes. Vienas iš paprasčiausių būdų susipažinti su DB ir SQL kalba yra pradėti nuo duomenų bazėje jau esamų duomenų tyrinėjimo.

Tarkime, DB *Darbai* yra trys lentelės *Vykdytojai*, *Projektai* ir *Vykdymas*. DB objektų vardus paryškinsime kursyvu. 2.1 pav. parodyta, kaip lentelės užpildomos duomenimis

Vykdytojai

Nr	*Pavardė*	*Kvalifikacija*	*Kategorija*	*Išsilavinimas*
1	Jonaitis	Informatikas	2	VU
2	Petraitis	Statistikas	3	VU
3	Gražulytė	Inžinierius	1	NULL
4	Onaitytė	Vadybininkas	5	VDU
5	Antanaitis	Informatikas	3	VU

Projektai

Nr	*Pavadinimas*	*Svarba*	*Pradžia*	*Trukmė*
1	Studentų apskaita	Maža	2021-01-01	12
2	Buhalterinė apskaita	Vidutinė	2021-03-01	10
3	WWW svetainė	Didelė	2021-06-01	2

Vykdymas

Projektas	*Vykdytojas*	*Statusas*	*Valandos*
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100
2	1	Programuotojas	300
2	2	Analitikas	250
2	4	Vadovas	100
3	1	Programuotojas	250
3	2	Vadovas	400
3	3	Dizaineris	150

2.1 pav. Duomenų bazės *Darbai* lentelės su jose esančiais duomenimis

Duomenų bazėje *Darbai* sukaupta informacija apie įsivaizduojamoje informacijos technologijų įstaigoje vykdomus projektus (atliekamus darbus, užsakymus). Į lentelę *Vykdytojai*

įtraukti visi projektų vykdytojai (įstaigos darbuotojai). Šioje lentelėje vykdytojai surašyti eilės tvarka, nurodytos vykdytojų pavardės, turima kvalifikacija, darbuotojo kategorija, nusakanti jo kvalifikacijos lygį, taip pat išsilavinimą suteikusios institucijos pavadinimas. Kiekviena lentelės eilutė skirta vienam darbuotojui. Lentelėje *Projektai* sukaupta informacija apie įstaigos vykdomus projektus. Kiekviena lentelės eilutė atspindi konkretaus projekto numerį, pavadinimą, svarbos lygį ir vykdymo trukmę mėnesiais. Trečioje lentelėje *Vykdymas* surašyta informacija apie tai, kokius darbus kurie vykdytojai atlieka. Lentelės eilutėse nurodyti projekto ir darbuotojo eilės numeriai, dalyvaujančio projekte darbuotojo statusas ir kiek valandų vykdytojas skirs projektui. Kiekviena reikšmių pora (*Projektas*, *Vykdytojas*) lentelėje *Vykdymas* yra unikali, t. y. jei darbuotojas dalyvauja kuriame nors projekte, tai tam skirta tik viena eilutė.

Pastebėsime, kad DB *Darbai* yra tik mokomoji. Ji neatspindi realios situacijos. Pateiktoji DB yra pernelyg supaprastinta, kad atitiktų realių projektų vykdymą realioje įstaigoje. Šios DB pagrindinė paskirtis – išmokyti sudaryti užklausas.

Užklausose, kaip ir daugumoje kitų SQL sakinių, nenurodoma, kuriai DB skirtas sakinys. Duomenų bazės vardas nurodomas prieš veiksmus su duomenimis. DB, kuriai skiriami tolimesni SQL sakiniai, nurodoma sakiniu [7, 8, 12, 13]:

`CONNECT [TO]` <DB vardas> [`USER` <naudotojo vardas>].

Čia frazė „`CONNECT TO`“ yra bazinis SQL žodis (raktažodis), kuris gali būti trumpinamas praleidžiant dalelę `TO`. Laužtiniais sklaistais žymėsime nebūtinas SQL sakinių dalis. Eilutės gale padėtas taškas nėra SQL sakinio dalis – tai teksto sakinio pabaiga. SQL standartas nenumato sakinio pabaigos požymio, nors kai kuriose DBVS yra vartojamas sakinio pabaigos požymis, pavyzdžiui, kabliataškis. Pačiame SQL sakinyje gali būti vartojami įvairūs skyrybos ženklai. Raktažodžiai, kad išsiskirtų, rašomi kitokiu šriftu. SQL baziniai žodžiai gali būti rašomi didžiosiomis ir mažosiomis raidėmis. Tas pat taikoma ir DB objektų (lentelių, stulpelių ir kt.) pavadinimams – raidžių registras neturi įtakos. Šioje knygoje baziniai žodžiai rašomi didžiosiomis raidėmis.

Sakinys, kuriuo nurodoma duomenų bazė, su kuria bus toliau dirbama, yra „aktyvus“ – vykdant šį sakinį nustatomas fizinis ir loginis ryšys su DB serveriu ir pačia DB. Jei vykdant šį sakinį DBVS negalės palaikyti ryšio, pavyzdžiui, nutrūkus fiziniam ryšiui su DB serveriu, tai sakinys nebus įvykdytas. DBVS atitinkamu pranešimu informuoja vartotoją, ar sėkmingai įvykdytas SQL sakinys. Pradedant darbą su DB *Darbai*, reikia įvykdyti sakinį

`CONNECT TO` *Darbai*.

Prie DB serverio galima jungtis kitu, nei šiuo metu naudotojas yra prisijungęs prie kompiuterinių resursų. Vardas, kuriuo naudotojas prisistato DB serveriui nurodomas `CONNECT` sakinyje fraze `USER`, pvz.,

`CONNECT TO` *Darbai* `USER` *Stud*.

Baigus darbą, loginis ryšys su DB nutraukiamas sakiniu [7, 8, 12, 13]:

`DISCONNECT`.

Ne visos DBVS užtikrina šį sakinį. PostgreSQL, pvz. šio sakinio tiesiogiai neužtikrina, nors programavimo terpėse jį galima naudoti. Naudojantis interaktyviu SQL sakinių vykdytoju `psql`, duomenų bazių serveris ir duomenų bazės pavadinimas nurodoma `psql` parametruose [3, 5, 16, 17],

`psql [-d]`<DB vardas> `[-h`<DB serveris>`] [-U`<naudotojo vardas>`]`,

pvz.,

`psql -d` *Darbai* `-h` *pgsql.mif*,

`psql -d` *Darbai* `-h` *pgsql.mif* `-U` *Stud*.

Įvedus komandą `\q`, nutraukiamas ryšys su DB ir baigiamas darbas su programa `psql`.

Įsidėmėtina, kad nesant būtino reikalo, geriau nepalaikyti ryšio su DB, kadangi tai susiję su informacinių išteklių sąnaudomis vartotojo darbo vietoje ir serveryje. Geriausia užmegzti ryšį prieš pat pradedant darbą su DB ir nutraukti jį iš karto, kai tik naudotis ja nereikia (laikinai ar visiškai).

2.2. Sakinys SELECT

SQL sakinys `SELECT` yra pagrindinis sakinys duomenų bazėje turimiems duomenims peržvelgti. Šis sakinys turi daug įvairiausių variantų ir galimybių. Išsamiai sakinio sintaksei pateikti prireiktų keleto puslapių. Ne pačiu bendriausiu atveju jį galima užrašyti taip [7, 8, 12, 13]:

```
SELECT [DISTINCT] <stulpelių vardai>
       FROM       <lentelių vardai>
       [WHERE     <paieškos sąlyga>]
       [GROUP BY  <stulpelių vardai> [HAVING <paieškos sąlyga>]]
       [ORDER BY  <stulpelių vardai>].
```

Įvykdžius šį sakinį, DBVS pateikia vartotojui užklausos rezultatą – laikiną lentelę, kuri egzistuoja tik peržvelgiant užklausos rezultatą. Sistemos vartotojas rezultatą mato vaizduoklio ekrane arba gali apdoroti jį programiškai.

Paprasčiausia užklausa atrodo taip:

`SELECT` <stulpelių vardai> `FROM` <lentelės vardas>.

Tokios užklausos rezultatą sudaro nurodytieji pateiktos lentelės stulpeliai. Į rezultatą įtraukiamos visos lentelės eilutės. Sakinio dalį iki bazinio žodžio `FROM` vadinsime `SELECT` fraze. Ja nusakomi stulpeliai, kurie turi patekti į užklausos rezultatą.

Kadangi lentelėje gali būti labai daug eilučių, tai rezultatas, kurį sudaro lentelės visos eilutės, gali būti nepatogus peržvelgti. Sakinį papildžius fraze `WHERE` <paieškos sąlyga>, užklausos rezultatą sudarys tik tos eilutės, kurios tenkina nurodytą sąlygą. Tolimesniuose skyreliuose išsamiau aptarsime, kaip nustatomos paieškos sąlygos, taip pat kitos sakinio dalys.

2.3. Paprasčiausios užklausos duomenims išrinkti

SQL sakinį `SELECT`, kaip ir daugelį kitų sakinių, paaiškinsime nagrinėdami pavyzdžius. Tarkime, norime sužinoti visų informatikų – projektų vykdytojų – pavardes ir kategorijas. Nesunku pastebėti, kad visa ši informacija yra vienoje lentelėje *Vykdytojai*. Reikiamą informaciją galima sužinoti įvykdžius sakinį:

`SELECT` *Pavardė, Kategorija* `FROM` *Vykdytojai* `WHERE` *Kvalifikacija* = 'Informatikas'.

Šioje užklausoje pavartota frazė 'Informatikas' yra simbolių eilutė – SQL **konstanta** [7, 8, 12, 13]. Simbolių eilutė (tekstinių duomenų konstanta) SQL sakiniuose rašoma tarp apostrofų. Šios užklausos rezultatas bus laikina lentelė, kurią sudarys du stulpeliai, ir į ją įeis tos lentelės *Vykdytojai* eilutės, kuriose stulpelio *Kvalifikacija* reikšmė yra 'Informatikas':

Pavardė	*Kategorija*
Jonaitis	2
Antanaitis	3

Lentelės stulpelius užklausoje galima patikslinti nurodant lentelės vardą:

```
SELECT Vykdytojai.Pavardė, Vykdytojai.Kategorija FROM Vykdytojai
WHERE Vykdytojai.Kvalifikacija = 'Informatikas'.
```

Stulpelių vardų patikslinimai nėra būtini, jei SQL sakinyje vartojama tik viena lentelė, kaip pastarajame pavyzdyje. Vėliau pamatysime, kad stulpelių vardus gali tekti patikslinti, kai į sakinį patenka keletas lentelių. Patikslinant dažnai nėra patogu rašyti gana ilgus lentelių vardus. To galima išvengti naudojant lentelių vardų sinonimus, pvz.:

```
SELECT A.Pavardė, A.Kategorija FROM Vykdytojai AS A
WHERE A.Kvalifikacija = 'Informatikas'.
```

Paminėjus lentelės *Vykdytojai* vardą frazėje `FROM`, jam iš karto suteikiamas sinonimas *A* (raktažodis `AS` gali būti ir praleistas). Lentelei suteiktas sinonimas galioja tik tame viename SQL sakinyje.

Tarkime, mums reikia sužinoti visas aukštąsias mokyklas, kurių absolventai yra projektų vykdytojai. Šį uždavinį galime išspręsti tokia užklausa:

```
SELECT Išsilavinimas FROM Vykdytojai,
```

kurios rezultatas bus:

Išsilavinimas
VU
VU
NULL
VDU
VU

Kadangi užklausoje nėra nurodyta jokia sąlyga, tai rezultate yra tiek eilučių, kiek jų yra lentelėje *Vykdytojai*. Šiame rezultate tekstas 'VU' pasikartoja tris kartus, kas visiškai nebūtina. Vienodos eilutės galėtų netgi trukdyti, jei jų būtų daug. Vienodų eilučių galima išvengti naudojant bazinį žodį `DISTINCT` [7, 8, 12, 13]. Paskutinį sakinį perrašę taip:

```
SELECT DISTINCT Išsilavinimas FROM Vykdytojai,
```

gausime rezultatą, kurį sudarys tik trys eilutės. Taigi jei neįtraukiamas į užklausą bazinis žodis `DISTINCT`, tai užklausos rezultate galimos vienodos eilutės, tiksliau, vienodos eilutės nepanaikinamos.

Kai kurie stulpeliai užklausoje gali būti apskaičiuojami. Tarkime, lentelėje *Projektai* projektų trukmė nurodyta mėnesiais, o mums reikia žinoti trukmę dienomis. Paprastumo dėlei tarkime, kad kiekviename mėnesyje yra 30 dienų. Uždavinį išspręsime sakiniu:

SELECT *Pavadinimas, Trukmė* * 30 FROM *Projektai*.

Šiame sakinyje pavartotas skaičius 30 – tai skaitinių duomenų konstanta. Skaičiai SQL sakiniuose rašomi be jokių skyrybos ženklų ir tarpų. Šios užklausos rezultatas yra laikina lentelė, turinti du stulpelius. Pirmojo stulpelio reikšmės – tai lentelės *Projektai* stulpelio *Pavadinimas* reikšmės. Antrojo stulpelio reikšmės – tai tos pačios lentelės stulpelio *Trukmė* reikšmės, padaugintos iš 30. Antrasis stulpelis, skirtingai nuo pirmojo, nėra paprastas lentelės stulpelis, tai aritmetinis reiškinys, kuriame gali būti ir keli lentelės stulpeliai. Kai užklausos SELECT frazėje vartojamas reiškinys, rezultato stulpelio pavadinimas gali būti dviprasmiškas. Tokiais atvejais sistema, pateikdama užklausos rezultatą, stulpeliui suteikia tarnybinį pavadinimą. Skirtingos DBVS tarnybinį stulpelio pavadinimą parenga skirtingai. Pavadinimas dažnai atitinka stulpelio eilės numerį užklausoje (antrajam stulpeliui suteikiamas vardas „2“), bet suteiktas pavadinimas gali reikšti ir pavadinimo neapibrėžtumą, pvz. „?column?“ (PostgreSQL) [3, 5, 17, 17]. Tai ne visada priimtina. Vartotojas gali pats nurodyti norimą stulpelio pavadinimą, užrašęs bazinį žodį AS. Suteikiamas stulpeliui pavadinimas turi tenkinti tuos pačius reikalavimus, kaip ir duomenų bazės stulpelio pavadinimas. Norint stulpeliui suteikti pavadinimą, kuriame yra specialiųjų simbolių, būtina stulpelio pavadinimą rašyti tarp kabučių:

SELECT *Pavadinimas, Trukmė* * 30 AS "*Trukmė dienomis*" FROM *Projektai*.

Šioje užklausoje antrojo stulpelio pavadinimas yra tarp kabučių, nes jį sudaro du žodžiai (tiksliau, pavadinime yra tarpas). Užklausos rezultatas:

Pavadinimas	*Trukmė dienomis*
Studentų apskaita	360
Buhalterinė apskaita	300
WWW svetainė	60

Jau ir anksčiau vartojome stulpelių pavadinimus, kuriuos kai kuriose platinamose DBVS reikėtų rašyti tarp kabučių arba jie apskritai neleistini. Tai stulpelių pavadinimai, kuriuose pavartotos lietuviškos abėcėlės raidės, kurių nėra lotynų raidyne, pvz., *Pavardė*, *Pradžia*. Daugumoje DBVS duomenų bazės objektų (lentelių, stulpelių ir kt.) pavadinimuose (**SQL varduose**) leidžiama vartoti tik lotynų alfabeto raides. SQL kalboje raidėmis taip pat laikomi trys simboliai: $, # ir @. SQL vardai turi prasidėti raide. Sudarant SQL vardus galima vartoti raides, skaitmenis bei pabraukimo ženklą (_). Patogumo dėlei sutarsime, kad sudarant SQL vardus galima vartoti visas lietuvių kalbos raides. Tarp kabučių rašysime tik vardus, kuriuose yra specialiųjų simbolių.

Ankstesniajame pavyzdyje, SELECT frazėje, pavartojome aritmetinį reiškinį. Išskirtinis reiškinio atvejis yra konstanta. Pateiksime užklausą su konstanta SELECT frazėje:

```
SELECT Pavadinimas,
       'Trukmė dienomis:' AS Pastaba,
       Trukmė * 30 AS "Trukmė dienomis"
FROM   Projektai.
```

Užklausos antrojo stulpelio reikšmė yra konstanta. Visos šio stulpelio reikšmės yra vienodos ir lygios tekstinei konstantai 'Trukmė dienomis:':

Pavadinimas	*Pastaba*	*Trukmė dienomis*
Studentų apskaita	Trukmė dienomis:	360
Buhalterinė apskaita	Trukmė dienomis:	300
WWW svetainė	Trukmė dienomis:	60

Atkreipsime dėmesį į tai, kad užklausoje pavartoti ir apostrofai, ir kabutės. Priminsime, kad apostrofais žymimos tekstinės konstantos, o kabutėmis – lentelės stulpelių bei kitų duomenų bazės objektų pavadinimai.

Konstanta gali sudaryti ir visą užklausos rezultatą, pavyzdžiui, užklausos

`SELECT` 'Trukmė dienomis' `AS` *Pastaba* `FROM` *Projektai*

rezultatas bus sudarytas tik iš vienos konstantos:

Pastaba
Trukmė dienomis
Trukmė dienomis
Trukmė dienomis

Nors iš užklausos formuluotės aišku, kokia reikšmė sudarys užklausos rezultatą, DBVS vis tiek turi peržvelgti lentelę *Projektai*, nes rezultatą sudaro tiek eilučių, kiek jų yra lentelėje.

Norint užklausos rezultate gauti visų lentelės stulpelių reikšmes, užuot nurodžius juos `SELECT` frazėje, galima pavartoti specialųjį simbolį '*'. Visi duomenys, esantys lentelėje *Vykdytojai*, bus pateikti įvykdžius vieną iš šių užklausų:

`SELECT` * `FROM` *Vykdytojai*;
`SELECT` *Vykdytojai*.* `FROM` *Vykdytojai*;
`SELECT` *A*.* `FROM` *Vykdytojai A*.

2.4. Užklausos rezultato rikiavimas

Dažnai didelę duomenų aibę daug patogiau peržiūrėti, kai ši išdėstyta pagal kokį nors kriterijų. DBVS sutvarko užklausos rezultatą pagal tam tikrą kriterijų (arba keletą kriterijų), kai sakinyje `SELECT` yra tvarką apibrėžianti frazė `ORDER BY`. Šioje frazėje nurodomi stulpelių vardai arba stulpelių eilės numeriai, atskirti kableliais. Po kiekvieno stulpelio vardo (arba numerio) galima rašyti vieną iš bazinių žodžių: `ASC` (angl. *ascending*) arba `DESC` (angl. *descending*), kuriais nustatoma reikšmių **didėjimo** arba **mažėjimo tvarka**. Jei nepavartotas nė vienas iš šių žodžių, tai numatytoji tvarka yra `ASC`. SQL kalboje tarp tekstinių duomenų yra nustatyta abėcėlės tvarka [7, 8, 12, 13].

Pavyzdžiui, sistema, įvykdžiusi užklausą

`SELECT` *Pavardė* `FROM` *Vykdytojai* `ORDER BY` *Pavardė*,

pateiks visų vykdytojų pavardes, sutvarkytas abėcėlės tvarka:

Pavardė
Antanaitis
Gražulytė
Jonaitis
Onaitytė
Petraitis

Kai užklausoje nurodomas eilučių rikiavimas pagal kelis stulpelius, tai, visų pirma, rezultato eilutės rikiuojamos pagal pirmojo stulpelio reikšmes. Tik tuomet, kai keliose eilutėse pirmojo stulpelio reiškės sutampa, jos tarpusavyje rikiuojamos atsižvelgiant į antrojo stulpelio reikšmes. Bendruoju atveju, į *n*-ojo rikiavimo stulpelio, jei toks yra nurodytas, reikšmes atsižvelgiama tik tuomet, kai užklausos rezultate yra eilučių su vienodomis pirmų *n*-1 stulpelių reikšmėmis. Įvykdžiusi užklausą

```
SELECT Pavardė, Išsilavinimas, Kategorija FROM Vykdytojai
ORDER BY 2, Kategorija DESC,
```

sistema pateiks visų vykdytojų pavardes, jų išsilavinimą ir kategorijas, sutvarkytus pagal antrojo ir trečiojo stulpelių reikšmes. Rezultatas bus pateiktas abėcėlės tvarka pagal išsilavinimą. Esant vienodiems aukštųjų mokyklų pavadinimams, eilučių tarpusavio padėtis bus nustatoma pagal kategoriją, jų mažėjimo tvarka.

Pavardė	*Išsilavinimas*	*Kategorija*
Onaitytė	VDU	5
Petraitis	VU	3
Antanaitis	VU	3
Jonaitis	VU	2
Gražulytė	NULL	1

Gražulytės duomenys yra paskutinėje eilutėje todėl, kad ji dar neturi jokio išsilavinimo. `NULL` reikšmė laikoma didesne už bet kurią kitą reikšmę. Kadangi pavardės nėra tarp rikiavimo stulpelių, tai Petraičio ir Antanaičio eilučių tarpusavio padėtis praktiškai yra atsitiktinė.

2.5. Reiškiniai

Užklausose jau vartojome paprasčiausius reiškinius: konstantas, stulpelių pavadinimus ir aritmetinius reiškinius. Aptarsime juos išsamiau.

Vardinės konstantos (sisteminiai kintamieji) – tai reikšmės, kurias saugo DBVS ir kurias galima vartoti SQL sakiniuose. Paprastai tai išorinė DBVS atžvilgiu informacija, žyminti einamąją sisteminę datą (`CURRENT_DATE`), laiką (`CURRENT_TIME`), datą ir tikslų laiką (`CURRENT_TIMESTAMP`), sistemos vartotojo vardą (`CURRENT_USER`) ir pan. [7, 8, 12, 13]. Kai kuriose DBVS sisteminės reikšmės pasiekiamos ne per vardines konstantas, o pasitelkiant specialias funkcijas. Kita vertus, vardines konstantas taip pat galima vadinti funkcijomis.

Funkcija – tai operacija, nusakoma funkcijos vardu ir apskliaustais (lenktiniais skliaustais) argumentais, kurie vienas nuo kito atskiriami kableliais (kartais argumentų gali ir ne-

būti). Funkcija visuomet pateikia tam tikrą rezultatą (tai gali būti ir specialioji reikšmė NULL). Funkcijos skirstomos į jungtines (stulpelių) ir skaliarines [7, 8, 12, 13].

Jungtinės funkcijos – tai funkcijos, kurias galima pritaikyti eilučių aibėms: visoms eilutėms, tenkinančioms frazėje WHERE nurodytą sąlygą, arba eilučių grupei, apibrėžiamai fraze GROUP (eilučių grupavimą aptarsime vėliau) [7, 8, 12, 13]. Jungtinės funkcijos bet kokiam eilučių rinkiniui apskaičiuoja vieną reikšmę. Šios funkcijos dažnai vartojamos SELECT frazėje, nurodant stulpelį, pagal kurio reikšmes apskaičiuojamas funkcijos rezultatas. Jungtinės funkcijos vartojamos ir frazėje HAVING, kai funkcijos reikšmė skaičiuojama kiekvienai eilučių grupei. Štai keletas dažniau vartojamų jungtinių funkcijų:

Jungtinė funkcija	*Rezultatas*
SUM([DISTINCT] <reiškinys>)	(Skirtingų) ne NULL reikšmių suma
AVG([DISTINCT] <reiškinys>)	(Skirtingų) ne NULL reikšmių vidurkis
COUNT([DISTINCT] <reiškinys>)	(Skirtingų) ne NULL reikšmių skaičius
COUNT(*)	Eilučių skaičius aibėje
MAX(<reiškinys>)	Maksimali reikšmė
MIN(<reiškinys>)	Minimali reikšmė

Jei visos funkcijoje nurodyto stulpelio reikšmės yra NULL, arba stulpelis yra tuščias, tai funkcijų SUM, AVG, MIN, MAX rezultatas yra NULL, o funkcijos COUNT – nulis.

Pateiksime keletą užklausų su jungtinėmis funkcijomis. Pavyzdžiui, visų vykdytojų skaičių arba tiksliau, eilučių skaičių lentelėje *Vykdytojai* galima sužinoti įvykdžius sakinį:

```
SELECT COUNT(*) FROM Vykdytojai.
```

Visų vykdytojų, dalyvaujančių bent viename projekte, skaičių galima sužinoti įvykdžius sakinį:

```
SELECT COUNT(DISTINCT Vykdytojas) FROM Vykdymas.
```

Pirmosios (iš dviejų pastarųjų) užklausos rezultatas bus viena eilutė, kurioje pateiktas skaičius 5. Antrosios užklausos rezultatas – taip pat viena eilutė, tik joje bus skaičius 4 – tiek skirtingų vykdytojų numerių yra paminėta lentelės *Vykdymas* stulpelyje *Vykdytojas*.

Bendrą skaičių valandų, kurias vykdytojas Nr. 1 skiria visiems projektams, galima sužinoti įvykdžius užklausą:

```
SELECT SUM(Valandos) AS "Vykdytojo Nr. 1 valandos" FROM Vykdymas
WHERE Vykdytojas = 1.
```

Kadangi šis vykdytojas dalyvauja trijuose projektuose, t. y. lentelėje *Vykdymas* yra trys eilutės, tenkinančios nurodytą sąlygą (*Vykdytojas* = 1), ir 30 + 300 + 250 = 580, tai pastarosios užklausos rezultatas:

Vykdytojo Nr. 1 valandos
580

Skaliarinės funkcijos argumentas visuomet yra viena reikšmė. Funkcija gali turėti ir kelis argumentus, tačiau kiekvienas argumentas – tai viena reikšmė, o ne reikšmių aibė, kaip jungtinėje funkcijoje. Vartojant skaliarines funkcijas frazėje WHERE, funkcijos rezultatas ap-

skaičiuojamas tikrinant paieškos sąlygą kiekvienai lentelės eilutei atskirai. Tokios funkcijos dažnai vartojamos ir SELECT frazėje, kuomet rezultatas skaičiuojamas ne visoms eilutėms iš karto, kaip jungtinių funkcijų atveju, o kiekvienai eilutei atskirai. Paprastai DBVS leidžia vartoti gana daug skaliarinių funkcijų. Paminėsime keletą: EXTRACT(DAY FROM <data>) – diena (skaičius nuo 1 iki 31) datoje, EXTRACT(MONTH FROM <data>) – mėnuo (sakičius nuo 1 iki 12) datoje, EXTRACT(YEAR FROM <data>) – metai datoje, CHAR_LENGTH(<simbolių eilutė>) – simbolių eilutės ilgis, SUBSTRING(<simbolių eilutė> [FROM <pradžia>] [FOR <ilgis>]) – simbolių eilutės fragmentas ir pan. [3, 5, 7, 8, 12, 13].

Iš konstantų, kreipinių į funkcijas bei vardų, žyminčių DB objektus, atliekant operacijas galima sudaryti paprasčiausius reiškinius. SQL leidžia naudoti dvi vienvietes (unariąsias) operacijas: + (pliusas), – (minusas), bei keletą dviviečių (binariųjų): + (sudėtis), – (atimtis), * (daugyba), / (dalyba), || (sujungimas). Jei bent vienas iš pateiktų operacijų argumentų yra NULL, tai ir rezultatas yra NULL [3, 5, 7, 8].

Su tekstiniais duomenimis galima atlikti tik dviejų simbolių eilučių nuoseklų jungimą į vieną (||). Su skaičiais, kaip įprasta, galima atlikti visas aritmetines operacijas. Sudėtį ir atimtį galima atlikti ir su datos bei laiko duomenimis, tiksliau, turimą reikšmę galima padidinti, sumažinti arba rasti dviejų reikšmių skirtumą. Prie datos galima pridėti arba iš jos atimti tam tikrą sveikąjį skaičių dienų, mėnesių ar metų. Laiką galima pakeisti laiko vienetais. Atliekant tokias operacijas, vienas iš argumentų – datos ar laiko intervalas – turi turėti dimensiją – raktažodį, nurodantį vienetus. Dimensijų pavyzdžiai: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS. Reiškinio DATE '2020-12-31' + INTERVAL '3 DAYS' rezultatas yra DATE '2021-01-03', o reiškinys *Projektai.Pradžia* + INTERVAL '3 MONTHS' nusako datą, kuri bus praėjus 3 mėnesiams nuo *Projektai.Pradžia*. Atimant vieną iš kitos dvi datas ar du laikus, gaunamas laikotarpis. Galimas ir neigiamas laikotarpis. Tačiau daugelis DBVS turi savitą datos ir laiko aritmetiką.

Sudarant reiškinius, galima vartoti lenktinius skliaustus, taip keičiant operacijų atlikimo tvarką.

Reiškiniai gali būti predikatų argumentai (operandai). SQL kalboje **predikatas** – tai sąlyga lentelės eilutei ar eilučių grupei, kuri gali būti teisinga, neteisinga arba neapibrėžta. Visos predikate dalyvaujančios reikšmės turi būti tarpusavyje suderintos. Be to, simbolių eilutės, įeinančios į predikatus, dažniausiai turi būti ne ilgesnės už tam tikrą konkrečiai DBVS būdingą simbolių skaičių, pvz., 4000 [3, 5, 7, 8]. Paprasčiausi predikatai sudaromi naudojant palyginimo operacijas. Palyginime visuomet dalyvauja dvi reikšmės. SQL leidžiamos tokios palyginimo operacijos: =, <, <=, >, >=, <>. Jei palyginimo operacijose vienas iš operandų yra NULL, tai predikato rezultatas neapibrėžtas. Palyginimo operacijose operandai gali būti ne tik skaitiniai duomenys, bet ir tekstiniai bei datos ir laiko duomenys.

Prie paprasčiausių ir dažnai naudojamų predikatų priskiriama [3, 5, 7, 8]:

- *x* BETWEEN *y* AND *z* – rezultatas teisingas tik tuomet, kai reiškinio *x* reikšmė yra tarp *y* ir *z*, t. y. kai $x >= y$ ir $x <= z$;
- *x* NOT BETWEEN *y* AND *z* – rezultatas teisingas tik tuomet, kai *x* nėra tarp *y* ir *z*, t. y. kai $x < y$ arba $x > z$;
- *x* IN ($y_1, y_2,..., y_n$) – rezultatas teisingas tik tuomet, kai *x* reikšmė sutampa bent su viena iš reikšmių $y_1, y_2,..., y_n$;
- *x* NOT IN ($y_1, y_2,..., y_n$) – rezultatas teisingas tik tuomet, kai x nesutampa nė su viena iš reikšmių $y_1, y_n, ..., y_n$;

- *x* LIKE *y* – rezultatas teisingas tik tuomet, kai simbolių eilutė *x* yra „panaši“ į simbolių eilutę *y*. Simbolių eilutėje *y* galima naudoti formato simbolius: %, kuris atitinka bet kokį skaičių, bet kokių simbolių, įskaitant tuščią (nulinio ilgio) eilutę; _ (pabraukimo ženklas), kuris atitinka bet kurį vieną simbolį. Eilutėje *y* galimi ir bet kokie kiti simboliai, atitinkantys juos pačius;
- *x* NOT LIKE *y* – rezultatas teisingas tik tuomet, kai simbolių eilutė *x* nėra panaši į simbolių eilutę *y*. Eilutėje y gali būti pavartoti tie patys simboliai kaip ir predikate LIKE;
- *x* IS NULL – rezultatas teisingas tik tuomet, kai reiškinio *x* reikšmė yra NULL;
- *x* IS NOT NULL – rezultatas teisingas tik tuomet, kai reiškinio *x* reikšmė nėra NULL.

Iš predikatų, naudojant **logines operacijas**: AND (loginis „ir“), OR (loginis „arba“) arba NOT (loginis „ne“), galima sudaryti paieškos sąlygas. Paieškos sąlygos rezultatas gali būti: „tiesa“ (sąlyga patenkinta, teisinga), „netiesa“ (sąlyga nepatenkinta, neteisinga) arba neapibrėžtas [3, 5, 7, 8].

Loginių operacijų naudojimas SQL kalboje atitinka matematinėje logikoje priimtus susitarimus, tačiau reikia atsižvelgti į tai, kad argumentų reikšmės gali būti neapibrėžtos. Pateiksime loginių operacijų AND ir OR teisingumo lentelę [3, 5, 7, 8]:

X	*Y*	*X* AND *Y*	*X* OR *Y*
Tiesa	Tiesa	Tiesa	Tiesa
Tiesa	Netiesa	Netiesa	Tiesa
Tiesa	Neapibrėžta	Neapibrėžta	Tiesa
Netiesa	Tiesa	Netiesa	Tiesa
Netiesa	Netiesa	Netiesa	Netiesa
Netiesa	Neapibrėžta	Netiesa	Neapibrėžta
Neapibrėžta	Tiesa	Neapibrėžta	Tiesa
Neapibrėžta	Netiesa	Netiesa	Neapibrėžta
Neapibrėžta	Neapibrėžta	Neapibrėžta	Neapibrėžta

Loginė operacija NOT apibrėžiama taip: NOT (Tiesa) yra Netiesa, NOT (Netiesa) yra Tiesa ir NOT (Neapibrėžta) yra Neapibrėžta. Jei paieškos sąlygoje yra pavartoti skliausteliai, tai tarp jų esanti sąlyga įvertinama pirmiau.

Pavyzdžiui, projektų, kurių pavadinime yra frazė „apskaita“, jų svarba yra vidutinė ar didelė, ir kurie turėjo būti prasidėję iki šiandien, pavadinimus bei vykdymo pradžios datas galima sužinoti tokia užklausa:

```
SELECT Pavadinimas, Pradžia
FROM Projektai
WHERE Pavadinimas LIKE '%apskaita%' AND
      Svarba IN ('Vidutinė', 'Didelė') AND
      Pradžia < CURRENT_DATE.
```

Informacija apie vykdytojus informatikus arba vykdytojus, baigusius Vilniaus universitetą (nepriklausomai nuo kvalifikacijos) ir turinčius aukštesnę nei trečią kategoriją, bus pateikta įvykdžius užklausą:

```
SELECT * FROM Vykdytojai
WHERE Kvalifikacija = 'Informatikas' OR (Išsilavinimas = 'VU' AND Kategorija > 3).
```

2.6. Lentelių jungimas

Viena iš svarbiausių SELECT sakinio galimybių yra dviejų ar daugiau **lentelių jungimas** (angl. *join*) [2, 4, 6, 9, 15]. Tarkime, mums reikia sužinoti projekto Nr. 1 vykdytojų pavardes. Visa reikiama informacija yra dviejose lentelėse: *Vykdytojai* ir *Vykdymas*, todėl užklausoje turi dalyvauti ne viena, o dvi lentelės. Šios lentelės turi bendrą siejančią dalį – tai vykdytojo numeris. Pagal šį numerį lenteles galima logiškai sujungti. SQL kalba šį uždavinį galima užrašyti taip:

SELECT *Pavardė* FROM *Vykdytojai*, *Vykdymas*
WHERE *Vykdytojas* = *Nr* AND *Projektas* = 1.

Šioje užklausoje, sąlyga *Vykdytojas = Nr* yra nusakytas loginis ryšys tarp dviejų lentelių.

Jungiant lenteles, kiekviena vienos lentelės eilutė susiejama su kiekviena kitos lentelės eilute. Bendru atveju, jei užklausoje dviem lentelėms nėra jokios sąlygos ir vienoje iš lentelių yra *n* eilučių, o kitoje – *m* eilučių, tai rezultatą sudaro $m \times n$ eilučių. Tarkime, turime dvi lenteles:

LentelėA

A1	*B1*
a_1	b_1
a_2	b_1
a_2	b_2

LentelėB

A2	*B2*	*C2*
a_1	b_1	c_1
a_2	b_2	c_2

Tuomet užklausos

SELECT *A1*, *B1, A2, B2, C2* FROM *LentelėA, LentelėB*

rezultatas atrodys taip:

A1	*B1*	*A2*	*B2*	*C2*
a_1	b_1	a_1	b_1	c_1
a_2	b_1	a_1	b_1	c_1
a_2	b_2	a_1	b_1	c_1
a_1	b_1	a_2	b_2	c_2
a_2	b_1	a_2	b_2	c_2
a_2	b_2	a_2	b_2	c_2

Paprastai užklausoje dviem lentelėms yra paieškos sąlyga, kuri logiškai susieja vienos lentelės vieną ar kelis stulpelius su kitos lentelės atitinkamais stulpeliais. Papildę pastarąją užklausą paieškos sąlyga, kuri sujungtų abi lenteles, palyginant dviejų vienos lentelės stulpelių (*A1, B1*) reikšmes su kitos lentelės dviejų stulpelių (*A2, B2*) reikšmėmis:

SELECT *A1, B1, C2* FROM *LentelėA, LentelėB*
WHERE *A1= A2* AND *B1 = B2*,

gausime tik dvi eilutes:

A1	*B1*	*C2*
a_1	b_1	c_1
a_2	b_2	c_2

Suprantama, stulpelių porų (*A1, A2*) ir (*B1, B2*) reikšmės turi būti tarpusavyje palyginamos, t. y. jų tipai (reikšmių aibės) bent jau neturi konfliktuoti. Paskutinės užklausos SELECT frazėje vietoj anksčiau pavartotų penkių stulpelių turime tik tris, nes du kiti stulpeliai naujų reikšmių neduoda.

Užrašysime užklausą apie tai, kokie vykdytojai kokius projektus vykdo ir kiek kiekvienam projektui skiria valandų. Jei rezultate norime matyti vykdytojo pavardę ir projekto pavadinimą, o ne jų numerius, tai reikia formuluoti užklausą net trims lentelėms:

SELECT *Pavardė*, *Pavadinimas*, *Valandos*
FROM *Vykdytojai*, *Projektai*, *Vykdymas*
WHERE *Projektas* = *Projektai.Nr* AND *Vykdytojas* = *Vykdytojai.Nr*.

Šioje užklausoje dvi lentelės (*Vykdytojai* ir *Projektai*) turi po vieną stulpelį tuo pačiu pavadinimu *Nr*, todėl šio stulpelio pavadinimą būtina patikslinti lentelės pavadinimu.

Lentelę galima sujungti ir viduje. Sudarykime poras vienodos kvalifikacijos vykdytojų:

SELECT *A.Pavardė*, *B.Pavardė* FROM *Vykdytojai A*, *Vykdytojai B*
WHERE *A.Kvalifikacija* = *B.Kvalifikacija* AND *A.Nr* < *B.Nr*.

Jungdami lentelę viduje, galime tarti, kad turime du tos pačios lentelės egzempliorius (kopijas). Kad būtų galima kreiptis ir į abu vienos lentelės egzempliorius, būtina kiekvienam egzemplioriui suteikti pavadinimą (*A* ir *B*). Kadangi abiejų lentelių (tiksliau dviejų vienos lentelės egzempliorių) stulpelių vardai sutampa, tai užklausoje turėsime patikslinti stulpelių vardus lentelės egzemplioriaus pavadinimu. Predikatas *A.Kvalifikacija* = *B.Kvalifikacija* atlieka dviejų lentelių loginio susiejimo funkciją. Sąlyga *A.Nr* < *B.Nr* pavartota tik tam, kad būtų išvengta vykdytojo poros su savimi ir porų pasikartojimo. Jei rezultate turime porą (x, y), tai pora (y, x) nereikalinga, nes ji neteikia naujos informacijos. Nepadės čia ir bazinio žodžio DISTINCT pavartojimas, nes (x, y) ir (y, x) yra skirtingos eilutės, jei tik $x \neq y$.

Jau minėjome, kad, jungiant dvi lenteles, užklausos struktūra dažniausiai būna tokia:

SELECT <stulpeliai> FROM <lentelė 1>, <lentelė 2>
WHERE <jungimo sąlyga> [AND <paieškos sąlyga>].

Suprantama, paieškos sąlyga nėra būtina. Nors lentelių jungimo sąlyga yra bendrosios paieškos sąlygos dalis, tačiau, dėl jos svarbumo užklausose ir norint pabrėžti jos paskirtį, daugelyje DBVS yra užtikrinamas lentelių jungimas SELECT sakinio fraze JOIN. Tuomet užklausa su lentelių jungimu rašoma taip

SELECT <stulpeliai> FROM <lentelė 1> JOIN <lentelė 2> ON <jungimo sąlyga>
[WHERE <paieškos sąlyga>].

Užklausą visų projekto Nr. 1 vykdytojų pavardėms sužinoti galima suformuluoti taip:

SELECT *Pavardė* FROM *Vykdytojai* JOIN *Vykdymas* ON *Vykdytojas* = *Nr*
WHERE *Projektas* = 1.

Taip galima jungti ir daugiau lentelių, pvz., vykdytojų pavardes jų vykdomų projektų pavadinimus ir valandas galima sužinti taip:

```
SELECT Pavardė, Pavadinimas, Valandos
FROM (Vykdytojai JOIN Vykdymas ON  Vykdytojas = Vykdytojai.Nr)
                 JOIN  Projektai  ON  Projektas = Projektai.Nr.
```

Užklausą visų projekto Nr. 1 vykdytojų pavardėms sužinoti papildykime taip, kad rezultate būtų vykdytojų statusai ir valandos,

```
SELECT Pavardė, Statusas, Valandos
FROM Vykdytojai JOIN Vykdymas ON  Vykdytojas = Nr
WHERE Projektas = 1.
```

Vykdant šią užklausą, DBVS jungdama dvi lenteles, atrenka duomenis apie vykdytojus, vykdančius bet kuriuos projektus, o tuomet palieka tik tuos, kurie vykdo projektą Nr. 1. DBVS gali elgtis ir kita tvarka: pradžioje lentelėje *Vykdymas* atrinkti tik tuos vykdymus, kuriuose vykdomas projektas Nr. 1, o tuomet juos jungti su lentelėje *Vykdytojai* esančiais vykdytojais. Abiem atvejais rezultatas bus tas pat. Šiuo atveju, nekeičiant užklausos prasmės ir jos rezultato, paieškos sąlygą, nurodytą frazėje `WHERE`, galima perkelti į jungimo sąlygą,

```
SELECT Pavardė, Statusas, Valandos
FROM Vykdytojai JOIN Vykdymas ON (Vykdytojas = Nr  AND Projektas = 1).
```

Kaip tikėjomės, rezultate bus tik tų autorių pavardės, kurie dalyvauja projekte Nr. 1, t. y. vykdytojo Nr. 5 nebus. Tačiau noras šio uždavinio atsakyme matyti visus darbuotojus būtų visiškai suprantamas. Žinoma, darbuotojai, nedalyvaujantys projekte Nr. 1, turėtų būti atitinkamai pažymėti. Tokiems uždaviniams spręsti, SQL kalboje numatyta ypatingos jungimo operacijos savybės. Tai - **išorinis** (angl. *outer*) **jungimas**. Konkrečiai šiam uždaviniui tinka `LEFT OUTER JOIN` operacija, kai jungimo rezultatas yra papildomas kairiosios (pirmosios) lentelės nesujungiamomis eilutėmis. Dar yra galima `RIGHT OUTER JOIN` (papildoma dešiniosios lentelės eilutėmis, kurios nepatenka įprastos `JOIN` operacijos atveju) ir `FULL OUTER JOIN`, kai papildoma abiejų lentelių eilutėmis [7, 8, 10, 11, 12, 13].

Mūsų anksčiau taikytasis jungimas (`JOIN`) tiksliau yra vadinamas **vidiniu jungimu** (`INNER JOIN`), t. y. operacija `JOIN` yra tapati operacijai `INNER JOIN`.

Visų darbuotojų sąrašą su jų dalyvavimo ar nedalyvavimo projekte Nr. 1 informacija gausime įvykdę užklausą:

```
SELECT Pavardė, Statusas, Valandos
FROM  Vykdytojai LEFT OUTER JOIN Vykdymas
      ON (Vykdytojas = Nr AND Projektas = 1).
```

Užklausos rezultatas:

Pavardė	*Statusas*	*Valandos*
Jonaitis	Programuotojas	30
Petraitis	Dokumentuotojas	100
Gražulytė	Testuotojas	100
Onaitytė	Vadovas	100
Antanaitis	`NULL`	`NULL`

Tokį pat rezultatą gautume taikydami `RIGHT OUTER JOIN` operaciją ir sukeitę lenteles vietomis,

```
SELECT Pavardė, Statusas, Valandos
FROM  Vykdymas RIGHT OUTER JOIN Vykdytojai
      ON (Vykdytojas = Nr AND Projektas = 1).
```

Bendruoju atveju, jungimo sąlyga užklausose gali būti žymiai sudėtingesnė.

2.7. Struktūrinės užklausos

Vienoje užklausoje galima pavartoti ir kitą užklausą. Todėl žodis „struktūrinė“ įeina į kalbos pavadinimą. Keli SELECT sakiniai, pavartoti vienoje užklausoje, visuomet yra griežtai priklausomi. Ši priklausomybė yra hierarchinė – viena užklausa yra išorinė kitos atžvilgiu, o ši yra **vidinė** (dalinė, angl. *subquery*) išorinės atžvilgiu. Tačiau tai tik sintaksinė bei loginė priklausomybė; užklausų vykdymo tvarka nebūtinai yra tokia [2, 4, 6, 9, 15].

Projekte Nr. 1 dalyvaujančių vykdytojų pavardes apibrėšime pasitelkdami vidinę užklausą:

```
SELECT Pavardė FROM Vykdytojai
WHERE Nr IN (SELECT Vykdytojas FROM Vykdymas WHERE Projektas = 1).
```

Šioje užklausoje pavartotas bendresnis predikato IN variantas, kai antrasis parametras nurodytas vidine užklausa. Vykdant tokią užklausą, DBVS loginiu požiūriu (realiai gali būti ir kitaip) iš pradžių įvykdo vidinę (esančią sąlygoje) užklausą, ir gauna jos rezultatą – reikšmių aibę. Po to vykdoma išorinė užklausa, kiekvienai išorinės užklausos eilutei tikrinant, ar vykdytojo numeris priklauso aibei, gautai įvykdžius vidinę užklausą.

Pastarojoje užklausoje pavartoti du SELECT sakiniai nėra riba – jų gali būti ir daugiau. Pavyzdžiui, užklausą „pavardės vykdytojų, dalyvaujančių bent viename didelės svarbos projekte“ galima suformuluoti trimis SELECT sakiniais:

```
SELECT Pavardė FROM Vykdytojai
WHERE Nr IN
      (SELECT Vykdytojas FROM Vykdymas
       WHERE Projektas IN
            (SELECT Nr FROM Projektai WHERE Svarba = 'Didelė')).
```

Priminsime, kad SQL yra deklaratyvi kalba, t. y. formuluojant užklausas reikia apibrėžti, kas norima gauti užklausos rezultate, nesirūpinant, kaip tą rezultatą gauti. Pastaroji užklausa tėra formalus ir išsamus užrašas taip perfrazuotos užduoties: pavardės (*Pavardė*) vykdytojų (*Vykdytojai*), kurių numeris (*Nr*) yra tarp numerių vykdytojų (*Vykdytojas*), dalyvaujančių vykdant (*Vykdymas*) projektus, kurių numeris (*Projektas*) yra tarp numerių (*Nr*) projektų (*Projektai*), kurių svarba yra didelė (*Svarba* = 'Didelė'). Pastarojoje užklausoje visi SELECT sakiniai gali būti vykdomi nuosekliai, pradedant nuo pačios giliausios vidinės užklausos ir baigiant išorine.

Nors pastaroji užklausos formuluotė gana gerai atspindi žodinę užduoties formuluotę, didelis dalinių užklausų skaičius nereiškia gero užklausų sudarymo stiliaus. Dažnai jungiant keletą lentelių analogiška užklausa, užrašyta vienu sakiniu SELECT, atrodytų nesanti aiškesnė, tačiau ji trumpesnė, ir vis dėlto aiškesnė. Tačiau lentelėms sujungti reikia daugiau įgūdžių nei sudaryti dalinėms užklausoms. Pateiksime uždavinio apie vykdytojus, dalyvaujančius bent viename didelės svarbos projekte, sprendinį be dalinių užklausų:

```
SELECT DISTINCT Pavardė FROM Vykdytojai, Vykdymas, Projektai
WHERE Projektas = Projektai.Nr AND Vykdytojas = Vykdytojai.Nr AND
      Svarba = 'Didelė'.
```

Lentelių jungimas gali būti daug pranašesnis už dalines užklausas – tuo galima įsitikinti palyginus pastarąją užklausos formuluotę su ankstesnio skyrelio užklausa, kurioje jungiamos tos pačios trys lentelės. Pastebėsime, kad abiejų užklausų paieškos sąlygos yra labai panašios, skiriasi tik papildoma sąlyga, o lentelių jungimo sąlyga nesikeičia. Kartą išsiaiškinus, kaip lentelės logiškai siejamos tarpusavyje, vėliau daug užklausų galima parašyti tik papildant paieškos sąlygą.

Atkreipsime dėmesį, kad ta pati lentelė gali būti ir vidinėje, ir išorinėje užklausoje. Uždavinį „numeriai vykdytojų, kurie dalyvauja bent viename projekte, kuriame dalyvauja vykdytojas Nr. 1“ galime išreikšti ir taip:

```
SELECT DISTINCT Vykdytojas FROM Vykdymas
WHERE Vykdytojas <> 1 AND
      Projektas IN (SELECT Projektas FROM Vykdymas WHERE Vykdytojas = 1).
```

Nors šioje užklausoje ta pati lentelė pavartota du kartus, nebūtina patikslinti stulpelių vardus dėl panaudotų skliaustų ir galiojančios taisyklės: jei stulpelio vardas nėra tikslus, tai jis priklauso lentelei iš einamosios (vidinės ar išorinės) užklausos.

Anksčiau jau keletą kartų formuluotą uždavinį „pavardės vykdytojų, dalyvaujančių projekte Nr. 1“ galima suformuluoti ir taip:

```
SELECT Pavardė FROM Vykdytojai
WHERE 1 IN (SELECT Projektas FROM Vykdymas
            WHERE Vykdytojas = Vykdytojai.Nr).
```

Šią užklausą neformaliai galima perfrazuoti taip: projekte Nr.1 dalyvaujantys vykdytojai – tai tokie vykdytojai, kuriems tarp visų projektų, kuriuose jie dalyvauja, yra projektas Nr. 1. Šioje gana painioje užklausoje dalinė užklausa negali būti vykdoma iš anksto, prieš pradedant vykdyti išorinę. Dalinė užklausa vykdoma kiekvienai pagrindinės užklausos eilutei. Tai – priklausomos dalinės užklausos pavyzdys. **Priklausomoji užklausa** yra tokia, kurios vidinės užklausos rezultatas priklauso nuo išorinės užklausos rezultato. Priklausomoje užklausoje vidinė užklausa turi apibrėžtus parametrus, į ją įeina parametras (*Nr*), įgyjantis reikšmę išorinėje užklausoje. Vidinę užklausą tenka vykdyti kiekvienai to kintamojo reikšmei. Priklausomose užklausose negalima abiejų užklausų (vidinės ir išorinės) vykdyti nuosekliai. Priklausomos užklausos dažnai yra gana painios ir neefektyvios, todėl jų reikia vengti.

2.8. Laikinieji pavadinti užklausų rezultatai

Kadangi užklausos rezultatas turi lentelės (nors ir labai laikinos) savybes, tai užklausą galima vartoti ir sakinio SELECT frazėje FROM [7, 8, 12, 13]. Pavyzdžiui, anksčiau suformuluotą užklausą „numeriai vykdytojų, kurie dalyvauja bent viename projekte, kuriame dalyvauja vykdytojas Nr. 1“, galima užrašyti ir taip:

```
SELECT DISTINCT Vykdytojas
FROM Vykdymas,
        (SELECT Projektas FROM Vykdymas WHERE Vykdytojas = 1) AS Projektai1
WHERE Vykdymas.Vykdytojas <> 1 AND Projektai1.Projektas = Vykdymas.Projektas.
```

Šio SQL sakinio `FROM` frazėje pavartota užklausa, kurios rezultatui suteiktas vardas *Projektai1*. Taip apibrėžtas užklausos rezultatas egzistuoja tik kol vykdoma visa užklausa. Į tokį apibrėžimą galima žiūrėti kaip į vidinę užklausą, kurios rezultatui suteiktas vardas.

Vidinės užklausos naudojimas `FROM` frazėje, ypač kai tokių apibrėžimų yra keletas, gali padaryti užklausą sunkiai suprantamą. Kad taip neatsitiktų, galima pasinaudoti sudėtingoms užklausoms formuluoti skirta konstrukcija `WITH` (**`WITH` užklausa**, angl. `WITH` *query*) [7, 8, 12, 13]. Taip formuluojama užklausa dar vadinama **bendruoju lentelių reiškiniu** (angl. *CTE – Common Table Expression*), taip pat ir **laikinuoju pavadintu užklausos rezultatu** (angl. *temporary named result set*), o kartais ir laikinąja lentele (angl. *temporary table*) [2, 4, 6, 9, 15]. Į taip pažymėtą laikiną (tarpinį) užklausos rezultatą, kuriam suteiktas vardas, galima kreiptis tik užklausoje, kurioje jis yra apibrėžtas. Kadangi laikinajam (tarpiniam) užklausos rezultatui suteikiamas vardas, tai į jį galima žiūrėti, kaip į laikiną lentelę, kuri egzistuoja tik iki kol vykdoma visa užklausa.

Taip formuluojant užklausas elgiamasi panašiai, kaip ir apibrėžiant vidinę užklausą frazėje `FROM`. Iš pradžių apibrėžiamas laikinasis užklausos rezultatas (gaunamas tarpinės užklausos rezultatas) suteikiant jam vardą, o paskui rašomas `SELECT` sakinys, kuriame kreipiamasi į aukščiau apibrėžtą tarpinį rezultatą. Užklausa formuluojama nuosekliai: laikinasis rezultatas yra apibrėžiamas (užklausos tekste) prieš tai, kai jis panaudojamas. Ankstesniąją užklausą galima suformuluoti taip:

```
WITH Projektai1 AS (SELECT Projektas FROM Vykdymas
                    WHERE Vykdytojas = 1)
SELECT DISTINCT Vykdytojas
FROM Vykdymas, Projektai1
WHERE Vykdymas.Vykdytojas <> 1 AND
      Projektai1.Projektas = Vykdymas.Projektas.
```

Taip apibrėžtos užklausos tarpiniame rezultate (laikinoje lentelėje) pavadintame *Projektai1* yra vienas stulpelis vardu *Projektas*, nes toks stulpelis yra nurodytas apibrėžiančiojoje (vidinėje) užklausoje. Apibrėžiant tarpinį rezultatą (laikiną lentelę), tuoj pat po jo pavadinimo, tarp skliaustų, galima išvardyti stulpelių pavadinimus, kurie bus suteikti tarpiniam rezultatui, kaip laikinai lentelei.

Viena fraze `WITH` galima apibrėžti kelis laikinuosius rezultatus, atskiriant vieną nuo kito kableliais. Formuluojant sudėtingas užklausas, ši konstrukcija leidžia uždavinio sprendimui panaudoti skaidymą. Suformulavus užklausą tarpiniams rezultatams gauti, jai suteikiamas vardas ir toliau formuluojama aukštesnio lygio užklausa.

2.9. Bendrumo ir egzistavimo kvantoriai užklausose

Pateiksime ankstesnio uždavinio „vykdytojų, dalyvaujančių projekte Nr. 1, pavardės“ dar vieną sprendimą. Tam taikysime predikatą, atitinkantį matematinės logikoje plačiai taikomą egzistavimo kvantorių:

```
SELECT Pavardė FROM Vykdytojai
WHERE  EXISTS ( SELECT * FROM Vykdymas
                    WHERE Vykdytojas = Nr AND Projektas = 1 ).
```

Šiame sakinyje išorinės užklausos paieškos sąlygoje yra pavartotas predikatas, kurio sintaksė yra EXISTS (SELECT * FROM…) [7, 8, 12, 13]. Šio predikato reikšmė yra „tiesa“ tuomet ir tik tuomet, kai vidinės užklausos rezultatas yra netuščioji aibė. Predikatas EXISTS yra egzistavimo kantoriaus atitikmuo.

Pastaroji užklausa yra priklausomoji. Vidinės, predikate pavartotos užklausos rezultatas priklauso nuo parametro *Nr*, kuris įgyja reikšmę išorinėje užklausoje. Šioje užklausoje projekto Nr. 1 vykdytoju yra laikomas tas vykdytojas, kuriam egzistuoja bent vienas dalyvavimas projekte.

Paieškos sąlyga

```
EXISTS (SELECT * FROM…)
```

yra tapati sąlygai

```
(SELECT COUNT(*) FROM…) > 0.
```

Šiame reiškinyje vienas palyginimo operacijos operandas yra užklausa. Kadangi užklausos, kurioje nėra duomenų grupavimo ir SELECT frazėje yra tik jungtinė funkcija, rezultatas visuomet yra vienintelė reikšmė, tai tokią užklausą galima naudoti palyginimo operacijoje.

Predikatų logikos **kvantorių** atitikmenys naudojami ir tokiuose predikatuose [7, 8, 12, 13]:

<reiškinys> <palyginimo operacija> <ALL | ANY | SOME> (<užklausa>)

Apskaičiuojant predikato ALL reikšmę lyginamojo reiškinio reikšmė yra lyginama su visomis užklausos rezultatą sudarančiomis reikšmėmis. Predikato ALL reikšmė yra „tiesa“ tuomet ir tik tuomet, kai užklausos rezultatas yra tuščioji aibė arba palyginimo reikšmė yra „tiesa“ visoms užklausos rezultato reikšmėms. Tai – **bendrumo kvantoriaus** atitikmuo.

Predikato ANY reikšmė yra „tiesa“ tuomet ir tik tuomet, kai palyginimo reikšmė yra „tiesa“ bent vienai užklausos rezultato reikšmei. Predikatas SOME yra predikato ANY sinonimas.

Darbuotojų kvalifikacijas, kuriose visi darbuotojai yra ne žemesnės negu antros kategorijos, galima sužinoti užklausa

```
SELECT DISTINCT A.Kvalifikacija FROM  Vykdytojai A
WHERE 2 <= ALL (SELECT B.Kategorija FROM Vykdytojai B
                    WHERE  A.Kvalifikacija = B.Kvalifikacija).
```

Predikatas ALL labai palengvina tokio uždavinio sprendimą. Kad užklausos rezultatą sudarytų tik skirtingos kvalifikacijos, panaudojome frazę DISTINCT. Pagrindinės (išorinės) užklausos sąlyga yra tikrinama kiekvienai lentelės *Vykdytojai* eilutei. Kai šioje lentelėje yra labai daug eilučių užklausos vykdymui gali prireikti nemažai laiko, nes vidinė užklausa yra priklausoma nuo parametro *A.Kvalifikacija*. Tarus, kad skirtingų kvalifikacijų yra žymiai mažiau negu darbuotojų, galima sudaryti efektyvesnį uždavinio sprendinį. Užklausoje galima įvardinti tarpinį laikiną rezultatą, kad vidinė užklausa būtų vykdoma tik skirtingoms kvalifikacijoms,

```
SELECT A.Kvalifikacija
FROM  (SELECT DISTINCT Kvalifikacija FROM  Vykdytojai) AS A
WHERE 2 <= ALL (SELECT B.Kategorija FROM Vykdytojai B
                    WHERE  A.Kvalifikacija = B.Kvalifikacija ).
```

Šią užklausą užrašius su fraze WITH, sakinys tampa šiek tiek aiškesniu,

```
WITH  Kvalifikacijos  AS (SELECT DISTINCT Kvalifikacija FROM  Vykdytojai)
SELECT A.Kvalifikacija
FROM  Kvalifikacijos AS A
WHERE 2 <= ALL (SELECT B.Kategorija FROM Vykdytojai B
                    WHERE  A.Kvalifikacija = B.Kvalifikacija).
```

Jau žinomas vykdytojų, dalyvaujančius projekte Nr. 1, pavardes galima sužinoti dar ir taip:

```
SELECT Pavardė FROM Vykdytojai
WHERE Nr = ANY(SELECT Vykdytojas FROM Vykdymas WHERE Projektas = 1).
```

Nesunku pastebėti, kad ši užklausa struktūriškai yra labai panaši į užklausą, pateiktą ankstesniajame skyrelyje, panaudojant predikatą `IN`. Predikatą `IN` galima išreikšti per predikatą `ANY`, o predikatą `NOT IN` per predikatą `ALL` [7, 8, 12, 13]. Užrašas

<reiškinys> `IN` (<užklausa>)

yra ekvivalentiškas užrašui

<reiškinys> `= ANY` (<užklausa>),

o užrašas

<reiškinys> `NOT IN` (<užklausa>)

yra ekvivalentus užrašui

<reiškinys> `<> ALL` (<užklausa>).

Darbuotojų kvalifikacijas, kuriose visi darbuotojai yra ne žemesnės negu antros kategorijos, taip pat galima išreikšti be predikato `ALL`

```
SELECT DISTINCT A.Kvalifikacija FROM  Vykdytojai A
WHERE  (SELECT COUNT (*) FROM Vykdytojai B
          WHERE  A.Kvalifikacija = B.Kvalifikacija  AND  A.Kategorija < 2) = 0.
```

Šioje užklausoje kvalifikacijos, kuriose visi darbuotojai yra ne žemesnės negu antros kategorijos, yra apibrėžiamos kaip kvalifikacijos, kuriose nėra žemesnės negu antros kategorijos darbuotojų.

2.10. Duomenų grupavimas

Tarkime, kiekvienam projektui reikia apskaičiuoti, koks visų vykdytojų jam skiriamas laikas iš viso. Šiam uždaviniui spręsti iki šiol nagrinėtų sintaksės priemonių nepakanka. Nesunku apskaičiuoti bendrą valandų skaičių kuriam nors vienam projektui, pavyzdžiui, Nr. 1:

```
SELECT  SUM(Valandos) FROM Vykdymas WHERE Projektas = 1.
```

Šiame SQL sakinyje paieškos sąlygoje yra apibrėžtas vienas projektas. Visoms sąlygą tenkinančioms eilutėms pritaikyta funkcija valandoms sumuoti. Pradedant spręsti uždavinį, šią stulpelių funkciją reikia pritaikyti kiekvienai eilučių grupei, atitinkančiai visus skirtingus projektus. **Eilučių grupavimą** užklausose realizuoja konstrukcija `GROUP BY` [7, 8, 12, 13], kurią pavartojus uždavinio sprendinys būtų toks:

```
SELECT Projektas, SUM(Valandos) AS Valandos
FROM Vykdymas
GROUP BY Projektas.
```

Visos lentelės *Vykdymas* eilutės grupuojamos taip, kad į kiekvieną grupę patektų eilutės su vienodomis stulpelio *Projektas* reikšmėmis. Po grupavimo `SELECT` frazė taikoma kiekvienai grupei (o ne eilutei), sudarant vieną užklausos rezultato eilutę. Vykdant pateiktą užklausą, DBVS kiekvienam projektui (kiekvienai grupei, kurią sudaro duomenys apie vieno projekto vykdymą) įtraukia į užklausos rezultatą projekto numerį ir sumą valandų, kurias skiria visi vykdytojai projektui vykdyti,

Projektas	*Valandos*
1	330
2	650
3	800

Pastebėsime, kad šiek tiek pakeitus pastarąją užklausą:

```
SELECT Projektas, Vykdytojas, SUM(Valandos) AS Valandos
FROM Vykdymas
GROUP BY Projektas
```

gaunama **neteisinga** užklausa. Sugrupavus lentelės *Vykdymas* eilutes taip, kad į vieną grupę patektų visos eilutės su ta pačia stulpelio *Projektas* reikšme, grupėje gali atsirasti kelios eilutės su skirtingomis stulpelio *Vykdytojas* reikšmėmis. Tuomet neaišku, kuri *Vykdytojo* reikšmė turėtų „atstovauti“ grupei užklausos rezultate.

Kadangi pritaikius `SELECT` frazę grupei eilučių gaunama tik viena rezultato eilutė, visi šios frazės reiškiniai turi būti vienareikšmiškai apskaičiuojami, t. y. turi įgyti vieną reikšmę visoje grupėje. Užklausos su grupavimu `SELECT` frazėje, su nedidelėmis išimtimis, tegali būti [7, 8, 12, 13]:

- stulpelis, paminėtas frazėje `GROUP BY` (grupavimo stulpelis);
- konstanta;
- reiškinys pagal konstantas ir grupavimo stulpelius;
- jungtinė (stulpelių) funkcija (`SUM`, `MIN`, `MAX`, `COUNT`, `AVG` ir kt.), kuri eilučių grupei pateikia vieną reikšmę.

Prieš `GROUP BY` frazę galima vartoti paieškos sąlygą, kuri tikrinama prieš eilučių grupavimą. Jei norime paieškos sąlygą pritaikyti ne kiekvienai lentelės eilutei, o kiekvienai grupei, tai frazė `GROUP BY` vartojama kartu su fraze `HAVING`. Sąlyga grupėms tikrinama po grupavimo. Grupės „atstovas“ į užklausos rezultatą įtraukiamas tik tuomet, kai grupė tenkina paieškos sąlygą, nurodytą frazėje `HAVING` [7, 8, 12, 13].

Taigi jei užklausoje su grupavimu yra pavartota frazė `WHERE`, tai iš pradžių kiekvienai eilutei tikrinama ši sąlyga, ir visos šią sąlygą tenkinančios eilutės yra sugrupuojamos. Jei užklausoje pavartota ir frazė `HAVING`, tai pastarojoje frazėje nurodyta sąlyga yra tikrinama

kiekvienai grupei. Pavyzdžiui, numeriai projektų, kuriuos vykdo daugiau nei vienas vykdytojas, gali būti sužinomi tokia užklausa:

```
SELECT Projektas FROM Vykdymas
GROUP BY Vykdytojas HAVING COUNT(*) > 1.
```

Galima grupuoti ir pagal kelis stulpelius. Tuomet sakoma, kad grupuojama grupės viduje. Tarkime, reikia sužinoti, kiek yra vykdytojų pagal tokius konkrečius požymius: baigta aukštoji mokykla ir turima kategorija. Užklausai sudaryti reikia apibrėžti grupavimą pagal du lentelės *Vykdytojai* stulpelius: *Išsilavinimas* ir *Kategorija*:

```
SELECT Išsilavinimas, Kategorija, COUNT(*) AS "Vykdytojų skaičius"
FROM Vykdytojai
WHERE Išsilavinimas IS NOT NULL
GROUP BY Išsilavinimas, Kategorija
ORDER BY Išsilavinimas.
```

Šios užklausos rezultatas, surikiuotas pagal išsilavinimą abėcėlės tvarka, atrodo taip:

Išsilavinimas	*Kategorija*	*Vykdytojų skaičius*
VDU	5	1
VU	2	1
VU	3	2

Tarkime, reikia sužinoti, kiek kiekvienas vykdytojas skiria valandų visiems projektams vykdyti. Taip pat mus domina tik tie vykdytojai, kurie visiems projektams vykdyti skiria daugiau nei 300 valandų. Jei mums pakaktų vykdytojo eilės numerio, tai uždavinį galėtume spręsti panašiai kaip uždavinį, kuriame valandos buvo skaičiuojamos kiekvienam projektui. Jei mus domina vykdytojų pavardės, užklausą reikia formuluoti dviem lentelėms:

```
SELECT Pavardė, SUM(Valandos) AS "Visos valandos"
FROM Vykdytojai, Vykdymas
WHERE Nr = Vykdytojas
GROUP BY Pavardė HAVING SUM(Valandos) > 300
ORDER BY "Visos valandos".
```

Sąlygoje grupei (frazėje `HAVING`) negalima vartoti `SELECT` frazėje apibrėžto stulpelio pavadinimo „*Visos valandos*“. Tokio stulpelio pavadinimas grupėse negalioja. Jo tiesiog grupėse nėra. Stulpelio pavadinimas prasmingas tik užklausos rezultato kontekste, todėl jį galima vartoti apibrėžiant eilučių tvarką.

Jei norime užklausos rezultate pamatyti ne tik vykdytojų pavardes, bet ir jų numerius, tai, atsižvelgiant į jau pateiktus apribojimus `SELECT` frazei, tenka grupuoti pagal du stulpelius:

```
SELECT Pavardė, Nr, SUM(Valandos) AS "Visos valandos"
FROM Vykdytojai, Vykdymas
WHERE Nr = Vykdytojas
GROUP BY Pavardė, Nr
HAVING SUM(Valandos) > 300
ORDER BY Pavardė.
```

Tai, kad kiekvienam vykdytojui (jo pavardei) gausime ne daugiau kaip po vieną eilutę, lems savybė (žinoma, jei tokia savybė tenkinama), kad nėra dviejų vykdytojų vienodomis pavardėmis. Jei keli vykdytojai galėtų turėti vienodas pavardes, bet skirtingus numerius, tai pastaroji užklausa išliktų teisinga, tuo tarpu ankstesnė, kurioje grupuojama tik pagal pavardes, logiškai būtų klaidinga (sintaksiškai išliktų teisinga), nes visi bendrapavardžiai būtų laikomi tuo pačiu asmeniu. Taigi ir ankstesnėje užklausoje būtų buvę teisingiau (užklausos teisingumas nepriklausytų nuo apribojimų duomenims) pavartoti grupavimą pagal abu vykdytojo tapatumo atributus: numerį ir pavardę.

2.11. Aibių operacijos

Kadangi užklausos rezultatas yra eilučių aibė, tai prasminga dviejų užklausų rezultatams taikyti aibių operacijas: sąjungą, sankirtą, skirtumą. Kiekvienai iš šių trijų aibių teorijos operacijų SQL kalboje yra po dvi savas operacijas. Taip yra todėl, kad užklausų rezultatuose, skirtingai negu aibių teorijoje, galimos vienodos eilutės. SQL kalboje yra šešios aibių operacijos: `UNION`, `UNION ALL`, `INTERSECT`, `INTERSECT ALL`, `EXCEPT` ir `EXCEPT ALL` [7, 8, 12, 13]. Apibrėšime šių operacijų rezultatus.

`UNION` ir `UNION ALL` – rezultatas yra dviejų užklausų rezultatų (*R1* ir *R2*) eilučių sąjunga. Operacijos rezultatas sudaromas iš visų *R1* ir *R2* eilučių, tada, jei nepavartotas bazinis žodis `ALL`, panaikinamos pasikartojančios eilutės, paliekant po vieną kiekvienos eilutės egzempliorių.

`INTERSECT` ir `INTERSECT ALL` – rezultatas yra dviejų užklausų rezultatų (*R1* ir *R2*) eilučių sankirta. Rezultatas sudaromas iš eilučių, kurios įeina ir į *R1*, ir į *R2*, po to, jei nepavartotas bazinis žodis `ALL`, panaikinamos pasikartojančios eilutės, paliekant po vieną kiekvienos eilutės egzempliorių.

`EXCEPT` ir `EXCEPT ALL` – rezultatas yra dviejų užklausų rezultatų (*R1* ir *R2*) eilučių skirtumas. Rezultatas sudaromas iš *R1* eilučių, kurių nėra rezultate *R2*, atliekant tai kiekvienam eilutės egzemplioriui atskirai, bet prieš tai, jei nepavartotas bazinis žodis `ALL`, iš *R1* ir iš *R2* yra pašalinamos pasikartojančios eilutės, paliekant po vieną kiekvienos eilutės egzempliorių.

Kad būtų galima atlikti aibių operacijas su užklausų rezultatais, stulpelių skaičius operacijų argumentuose (užklausų rezultatuose *R1* ir *R2*) būtinai turi sutapti, o atitinkamų stulpelių tipai neturi būti prieštaringi. Jeigu, pavyzdžiui, vienos užklausos rezultate būtų vienas stulpelis, o kitos – du, tai būtų neaišku, kiek stulpelių turėtų būti tokių užklausų rezultatų sankirtoje. Todėl panašios situacijos SQL sintaksė neleidžia.

Paaiškinsime visas SQL aibių operacijas. Tarkime, turime du vienodos struktūros užklausų rezultatus (lenteles) *R1* ir *R2*, ir iš viso yra šešios skirtingos eilutės, kurias pažymėsime e_1,..., e_6. Tarkime, pirmosios užklausos rezultate *R1* yra dešimt eilučių: tris kartus pasikartoja eilutė, pažymėta e_1, du kartus – e_2, vieną kartą – e_3, tris kartus – e_4 ir vieną kartą – e_5, o *R2* sudaro septynios eilutės: dukart pasikartojanti eilutė e_1, tris kartus – e_3, vieną kartą – e_4 ir viena eilutė, pažymėta e_6. Tuomet visų šešių aibių operacijų rezultatus galima pavaizduoti tokia lentele [7, 8, 12, 13]:

R1	*R2*	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTERSECT ALL	INTERSECT
e_1	e_1	e_1	e_1	e_1	e_2	e_1	e_1
e_1	e_1	e_1	e_2	e_2	e_5	e_1	e_3
e_1	e_3	e_1	e_3	e_2		e_3	e_4
e_2	e_3	e_1	e_4	e_4		e_4	
e_2	e_3	e_1	e_5	e_4			
e_3	e_4	e_2	e_6	e_5			
e_4	e_6	e_2					
e_4		e_3					
e_4		e_3					
e_5		e_3					
		e_3					
		e_4					
		e_4					
		e_4					
		e_4					
		e_5					
		e_6					

Atlikus aibių operacijas, gautą užklausos rezultatą galima rikiuoti. Tada eilučių tvarką apibrėžianti frazė rašoma po antrojo aibių operacijos operando. Pavyzdžiui, visus vykdytojus (jų numerius), kurie nedalyvauja nė viename projekte, galima sužinoti įvykdžius užklausą su aibių skirtumo operacija EXCEPT:

```
SELECT Nr FROM Vykdytojai
EXCEPT
SELECT Vykdytojas FROM Vykdymas
ORDER BY 1.
```

Kadangi operacijoje EXCEPT dalyvaujančių užklausų rezultatuose stulpelių pavadinimai (*Nr ir Vykdytojas*) yra skirtingi, tai rikiavimo stulpelis šioje užklausoje nurodytas eilės numeriu (1). Pastebėsime, kad šioje užklausoje operaciją EXCEPT pakeitę operacija EXCEPT ALL, gausime tą patį rezultatą. Taip yra todėl, kad lentelėje *Vykdytojai* nėra vienodų stulpelio *Nr* reikšmių.

2.12. Lentelių reikšmių konstruktorius

Standarte SQL2 yra numatyta galimybė apibrėžti lentelę betarpiškai SQL sakinyje. Tam naudojamas **lentelių reikšmių konstruktorius** VALUES [7, 8, 12, 13]. Konstruojant lentelę visos jos eilutės išvardijamos tiesiogiai, atskiriant jas vieną nuo kitos kableliais. Kiekvienos eilutės stulpelių reikšmės yra rašomos tarp riestinių skliaustų, atskiriant jas kableliais. Stulpelių reikšmės pateikiamos reiškiniais, paprasčiausiu atveju – konstantomis.

Sakinys VALUES taip pat yra ypatinga užklausa duomenų išrinkimui. Sakinio rezultatas yra lentelė, kurios turinys yra nurodytas pačiame sakinyje. Pavyzdžiui, sistemos datą galima sužinoti tokia užklausa

```
VALUES (CURRENT_DATE).
```

Lentelė (užklausa), sudaryta iš vienos eilutės, kurioje yra 3 datos: vakar, šiandien ir rytoj, sudaroma sakiniu:

```
VALUES  (CURRENT_DATE – INTERVAL '1 DAY',
         CURRENT_DATE,
         CURRENT_DATE + INTERVAL '1 DAY').
```

Lentelę su tais pačiais duomenimis, tačiau išdėstytais vieno stulpelio trijose eilutėse, galima išreikšti labai panašiai

```
VALUES  (CURRENT_DATE – INTERVAL '1 DAY'),
        (CURRENT_DATE),
        (CURRENT_DATE + INTERVAL '1 DAY').
```

Jei sakinyje VALUES yra išvardyta *n* eilučių: e_1, e_2, ..., e_n, kur $n > 1$, t. y.

VALUES (*e*1), (*e*2),..., (*en*),

tai šis sakinys yra tapatus sakiniui

VALUES(*e*1) UNION ALL VALUES(*e*2) ... UNION ALL VALUES(*e*n).

Visos sakinyje VALUES išvardytos eilutės turi būti tarpusavyje suderintos – sudarytos iš vienodo skaičiaus reikšmių (stulpelių) ir turi būti neprieštaringi atitinkamų reikšmių tipai.

Sakiniu VALUES sudarytą lentelę, kaip kiekvienos užklausos rezultatą, galima rikiuoti ir grupuoti, pvz.,

```
VALUES  (CURRENT_DATE – INTERVAL '1 DAY'),
        (CURRENT_DATE),
        (CURRENT_DATE + INTERVAL '1 DAY') ORDER BY 1 DESC.
```

Sakinyje VALUES negalima suteikti stulpeliams pavadinimų. Tačiau, tai galima padaryti įvardinant laikiną (tarpinį) rezultatą, pvz.,

```
SELECT Vakar, Šiandien, Rytoj
FROM  (VALUES (CURRENT_DATE – INTERVAL '1 DAY',
               CURRENT_DATE,
               CURRENT_DATE + INTERVAL '1 DAY'))
      AS  Dienos(Vakar, Šiandien, Rytoj).
```

Šioje užklausoje sukonstruotoji lentelė pažymima kaip įvardintas laikinasis rezultatas. Sakinys VALUES dažniausiai naudojamas vardinių konstantų (sistemos laiko, datos, vartotojo vardo ir kt.) reikšmėms sužinoti.

2.13. Sąlyginiai reiškiniai

Sudarykime projektų sąrašą, kuriame šalia projekto pavadinimo būtų pažymėta, ar jis trumpalaikis, ar ilgalaikis. Projektą laikysime trumpalaikiu, jei jo vykdymo trukmė ne ilgesnė kaip 6 mėn., o ilgesnius projektus vadinsime ilgalaikiais. Tokį sąrašą galima gauti užklausa:

```
SELECT Pavadinimas, 'Trumpalaikis'  FROM Projektai WHERE Trukmė <= 6
UNION
SELECT Pavadinimas, 'Ilgalaikis'    FROM Projektai WHERE Trukmė > 6.
```

Šioje užklausoje, priklausomai nuo sąlygos teisingumo, stulpelio reikšmė (bendruoju atveju – reiškinys) pakeičiama viena ar kita reikšme (kitu reiškiniu). Jei norėtume projektų vykdymo trukmę skirstyti į daugiau intervalų, tai šitaip sprendžiant uždavinį, kiekvienam papildomam trukmės intervalui reikėtų pavartoti naują aibių sąjungos operaciją. Tokio uždavinio sprendimą galima supaprastinti pavartojus konstrukciją `CASE`, įgalinančią užrašyti **sąlyginį reiškinį** [7, 8, 12, 13]. Sąlyginio reiškinio sintaksę galima užrašyti taip:

```
CASE WHEN <paieškos sąlyga> THEN <reiškinys>
     {WHEN <paieškos sąlyga> THEN <reiškinys>}
     [ELSE <reiškinys>] END.
```

Sąlyginio reiškinio reikšmė priklauso nuo paieškos sąlygų teisingumo. Vykdant užklausą, kiekvienai eilutei peržiūrimos visos paieškos sąlygos, paeiliui tikrinant jų teisingumą, kol randama teisinga sąlyga. Tuomet viso reiškinio reikšme tampa reikšmė, esanti teisingos sąlygos dešinėje, už bazinio žodžio `THEN`. Jei tokios sąlygos nėra, tai viso sąlyginio reiškinio reikšmė apskaičiuojama pagal reiškinį, esantį dešinėje raktažodžio `ELSE`, jei tik toks yra, kitaip viso reiškinio reikšmė yra `NULL`.

Pastarąją užklausą galima užrašyti be aibių operacijos, vienu sakiniu `SELECT`:

```
SELECT Pavadinimas,
       CASE WHEN Trukmė <= 6 THEN 'Trumpalaikis' ELSE 'Ilgalaikis' END
FROM Projektai WHERE Trukmė IS NOT NULL.
```

Pateiksime dar vieną sąlyginio reiškinio vartojimo pavyzdį. Tarkime, kiekvienam vykdomam projektui turime pateikti statistinius duomenis: bendrą visų projekto vykdytojų skaičių ir jų skiriamą laiką, bei tokius pat duomenis apie informatikų bei statistikų dalyvavimą projekte atskirai. Visą šią informaciją galima sužinoti trimis labai panašiomis užklausomis:

```
SELECT Pavadinimas, COUNT(DISTINCT Vykdytojas) AS "Visi vykdytojai",
       SUM(Valandos) AS "Visų valandos"
FROM Projektai, Vykdymas WHERE Nr = Projektas GROUP BY Pavadinimas;

SELECT Pavadinimas, COUNT(DISTINCT Vykdytojas) AS Informatikai,
       SUM(Valandos) AS "Informatikų valandos"
FROM Projektai, Vykdymas, Vykdytojai
WHERE Projektai.Nr = Projektas AND Vykdytojai.Nr = Vykdytojas AND
   Kvalifikacija = 'Informatikas'
GROUP BY Pavadinimas;

SELECT Pavadinimas, COUNT(DISTINCT Vykdytojas) AS Statistikai,
       SUM(Valandos) AS "Statistikų valandos"
FROM Projektai, Vykdymas, Vykdytojai
WHERE Projektai.Nr = Projektas AND Vykdytojai.Nr = Vykdytojas AND
      Kvalifikacija = 'Statistikas'
GROUP BY Pavadinimas.
```

Netgi visas šias užklausas sujungus į vieną, naudojantis aibių sąjungos operacija rezultatą peržiūrėti nepatogu, nes informacija apie vieną projektą bus trijose eilutėse. Informaciją apie projektą būtų patogiau peržiūrėti, jei visa ji būtų vienoje eilutėje. Tokį rezultatą gausime pavartoję sąlyginius reiškinius:

```
SELECT Pavadinimas,
   COUNT(DISTINCT Vykdytojas) AS "Visi vykdytojai",
   SUM(Valandos) AS "Visų valandos",
   COUNT(DISTINCT CASE WHEN Kvalifikacija = 'Informatikas'
         THEN Vykdytojas END) AS "Visi informatikai",
   SUM(CASE WHEN Kvalifikacija = 'Informatikas' THEN Valandos END)
         AS "Informatikų valandos"
   COUNT(DISTINCT CASE WHEN Kvalifikacija = 'Statistikas'
         THEN Vykdytojas  END) AS "Visi statistikai",
   SUM(CASE WHEN Kvalifikacija = 'Statistikas' THEN Valandos END)
         AS "Statistikų valandos"
FROM Projektai, Vykdymas, Vykdytojai
WHERE Projektai.Nr = Projektas AND Vykdytojai.Nr = Vykdytojas
GROUP BY Pavadinimas.
```

Dažnai vartojamiems sąlyginiams reiškiniams užrašyti yra numatyti sutrumpinimai – skaliarinės funkcijos: `NULLIF` ir `COALESCE` [7, 8, 12, 13]. Šios funkcijos apibrėžiamos taip (čia r_1, r_2,..., r_N žymi reiškinius):

Sąlyginis reiškinys	*Ekvivalenti funkcija*
CASE WHEN $r_1 = r_2$ THEN NULL ELSE e_1 END	NULLIF(r_1, r_2)
CASE WHEN r_1 IS NOT NULL THEN r_1 ELSE r_2 END	COALESCE(r_1, r_2)
CASE WHEN r_1 IS NOT NULL THEN r_1 ELSE COALESCE(r_2, ..., r_N) END	COALESCE(r_1, r_2,..., r_N)

Tarkime, mums reikia sužinoti, kokios būtų vykdytojų kategorijos, jei visiems vykdytojams jas padidintume vienetu, o tiems vykdytojams, kuriems iki šiol kategorija nebuvo suteikta (tarkime, tokių gali būti), priskirtume pirmą kategoriją. Šiam uždaviniui spręsti pavartosime sutrumpintą sąlyginį reiškinį:

```
SELECT Pavardė, Kategorija AS "Esama kategorija",
         COALESCE(Kategorija, 0) + 1  AS  "Naujoji kategorija"
FROM Vykdytojai.
```

2.14. Rekursyviosios užklausos

`WITH` užklausos gali tapti rekursyviomis (angl. *recursive query*), jei tik greta raktinio žodžio `WITH` rašomas papildomas raktinis žodis `RECURSIVE`. Tokiuose SQL sakiniuose, tarpinį rezultatą apibrėžiančioje užklausoje galima rekursyviai kreiptis į apibrėžiamą laikinąją lentelę. Taip formuluojamos užklausos dar vadinamos rekursyviais **bendraisiais lentelių reiškiniais** (angl. *recursive common table expressions* – CTE) [8, 10, 11, 12]. Šios užklausos standartizuotos SQL3.

Rekursijai išreikšti pasitelkiamos aibių operacijos su užklausų rezultatais, dažniausiai, jų sąjunga: `UNION ALL` ar `UNION`. Pirmoji sąjungos užklausa (nerekursyvioji dalis) paprastai nusako paprasčiausiąjį (pirmąjį) atvejį, o kita (rekursyvioji dalis) – bendrąjį atvejį, naudojant apibrėžiamąją lentelę. Rekursyviojoje dalyje rekursyviai kreipiamasi į apibrėžiamąją lentelę [8, 10, 11, 12]:

```
WITH RECURSIVE Lentelė AS (
      <užklausa> -- nerekursyvioji dalis
   UNION [ALL]
      <užklausa> -- rekursyvioji dalis
)
SELECT * FROM Lentelė.
```

Rekursyvioji užklausa vykdoma nuosekliai [8, 10, 11, 12]:

1) įvykdoma nerekursyvioji dalis ir suformuojamas rezultato pagrindas R_0 – dažniausiai viena eilutė;
2) įvykdoma rekursyvioji dalis anksčiau gautiems duomenims R_i ir suformuojamas rezultatas R_{i+1};
3) žingsnis (2) kartojamas tol, kol rekursyviosios dalies rezultatas tampa tuščiu;
4) galutinis laikinosios lentelės turinys gaunamas taikant `UNION` ar `UNION ALL` tarpiniams rezultatams R_0, R_1, ..., R_n.

Plačiai žinomas rekursyviosios užklausos pavyzdys – sveikų skaičių nuo 1 iki 100 suma:

```
WITH RECURSIVE Skaičiai(N) AS (
      VALUES (1)
   UNION ALL
      SELECT N + 1 FROM Skaičiai WHERE N < 100
)
SELECT SUM(N) FROM Skaičiai.
```

Apibrėžiančioje užklausoje `UNION ALL` pakeitus į `UNION`, rezultatas nepakistų, bet paieškos sąlyga `WHERE` $N < 100$ yra labai svarbi, be jos, rekursija, o tuo pačiu ir visa užklausa nesibaigtų sėkmingai. Kartu tai rodo, kad rekursyviųjų užklausų mechanizmas, savo prigimtimi, yra artimesnis iteracijai, nei rekursijai, bet SQL3 tai vadinama rekursija.

Rekursyvios užklausos esminiai pasitelkiamos apdorojant hierarchinius ar grafo struktūros duomenis. Tarkime, lentelėje *Kvalifikacijos* yra apibrėžta hierarchinė darbuotojų kvalifikacijų sąranga:

Kvalifikacijos

Pokvalifikacis	*Kvalifikacija*
Medicinos statistikas	Statistikas
DB specialistas	Informatikas
DB projektuotojas	DB specialistas
DB administratorius	DB specialistas
PostgreSQL administratorius	DB administratorius
MySQL administratorius	DB administratorius
Programuotojas	Informatikas

Visų kvalifikacijų, kurios tiesiogiai ar netiesiogiai yra priskirtos informatiko kvalifikacijai, pavadinimus galima sužinoti pasitelkus rekursyvią kvalifikacijų hierarchijos peržiūrą:

```
WITH RECURSIVE Informatikai(Pavadinimas, Lygmuo) AS (
      SELECT Pokvalifikacis, 1 FROM  Kvalifikacijos
      WHERE  Kvalifikacija = 'Informatikas'
   UNION ALL
      SELECT A.Pokvalifikacis, B.Lygmuo + 1
      FROM  Kvalifikacijos A, Informatikai B
      WHERE  A. Kvalifikacija = B.Pavadinimas
)
SELECT  Pavadinimas, Lygmuo FROM Informatikai ORDER BY Lygmuo, Pavadinimas.
```

Šio sakinio rezultatu bus kvalifikacijų, kurios laikomos informatikos kvalifikacijos specializacijomis (specifiniais atvejais), pavadinimai ir lygmuo, nusakantis specializacijos gylį (lygmenį) informatikos atžvilgiu:

Pavadinimas	*Lygmuo*
DB specialistas	1
Programuotojas	1
DB administratorius	2
DB projektuotojas	2
MySQL administratorius	3
PostgreSQL administratorius	3

2.15. Papildomos paieškos galimybės

Jau sprendėme, kaip kiekvienam projektui apskaičiuoti bendrą visų vykdytojų jam skiriamą laiką. Šį uždavinį išsprendėme lentelės *Vykdymas* eilutėms pritaikę grupavimą pagal stulpelį *Projektas*. Gautąjį rezultatą, kuriame yra tiek eilučių, kiek tuo metu yra vykdomų projektų, galima papildyti sumine eilute, kurioje būtų visiems projektams visų vykdytojų skiriamo laiko suma. Tai galima atlikti naudojant **duomenų sumavimą grupuojant** [8, 10, 11, 12],

```
SELECT Projektas, SUM(Valandos) AS Valandos
FROM Vykdymas
GROUP  BY CUBE (Projektas),
```

Projektas	*Valandos*
1	330
2	650
3	800
NULL	1780

Šios užklausos rezultato paskutinėje eilutėje yra visiems projektams visų vykdytojų skiriamos valandos. `NULL` reikšmė paskutinėje eilutėje žymi, kad šioje eilutėje duomenys ne apie konkretų projektą, bet apie visus juos. Rezultato pirmosios trys tokios, kaip atitinkamos užklausos be sumavimo (be `CUBE`), o paskutinėje eilutėje esantis skaičius atitinka užklausos be grupavimo rezultatą,

```
SELECT SUM(Valandos) FROM Vykdymas.
```

Kai grupuojama pagal kelis stulpelius, raktinis žodis CUBE grupavimo rezultatą papildo suminėmis eilutėmis pagal visas grupavimo stulpelių kombinacijas. Jei sumavimą pritaikytume skaičiuojant vykdytojus kiekvienai jų baigtai mokyklai (išsilavinimui) ir turimai kategorijai 2.10 skyrelyje raštą užklausą reikėtų papildyti raktiniu žodžiu CUBE:

```
SELECT Išsilavinimas, Kategorija, COUNT(*) AS "Vykdytojų skaičius"
FROM Vykdytojai
WHERE Išsilavinimas IS NOT NULL
GROUP BY CUBE(Išsilavinimas, Kategorija),
```

Išsilavinimas	*Kategorija*	*Vykdytojų skaičius*
VDU	5	1
VDU	NULL	1
VU	2	1
VU	3	2
VU	NULL	3
NULL	2	1
NULL	3	2
NULL	5	1
NULL	NULL	4

Šio užklausos rezultato pirma, trečia ir ketvirta eilutės sudaro visą atitinkamos be sumavimo užklausos rezultatą, pateiktą 2.10 skyrelyje. Antroje eilutėje yra pateiktas visų vykdytojų, baigusių VDU, skaičius, penktoje – VU (3 = 1+2). Šeštoje eilutėje yra visų antros kategorijos vykdytojų skaičius, o septintoje ir aštuntoje – trečios ir penktos kategorijų. Galiausiai – visų įgijusių išsilavinimą vydytojų skaičius (4).

Sudarinėjant ataskaitas, neretai nėra labai svarbu sumuoti pagal visas visas grupuojamų stupelių reikšmių kombinacijas, pakanka hierarchinio sumavimo kai sumos kaupiamos tik kairiau esantiems grupavimo stulpeliams. **Hierarchinis sumavimas** žymimas ROLLUP raktažodžiu.

Pateiktoje užklausoje CUBE raktažodį pakeitus ROLLUP raktažodžiu, rezultate nebelieka sumavimo vien tik kategorijoms (6-8 eilučių) [8, 10, 11, 12],

```
SELECT Išsilavinimas, Kategorija, COUNT(*) AS "Vykdytojų skaičius"
FROM Vykdytojai
WHERE Išsilavinimas IS NOT NULL
GROUP BY ROLLUP(Išsilavinimas, Kategorija).
```

Šios užklausos rezultatas, išdėstytas pagal išsilavinimą abėcėlės tvarka, atrodo taip:

Išsilavinimas	*Kategorija*	*Vykdytojų skaičius*
VDU	5	1
VDU	NULL	1
VU	2	1
VU	3	2
VU	NULL	3
NULL	NULL	4

Sumavimas grupuojant buvo standartizuotas SQL3, o PostgreSQL šią galimybę užtikrina nuo ver. 9.5. Čia pateiktos sumavimo grupuojant būdai nėra vieninteliai numatyti SQL3 standarte. SQL duomenų analizės galimybės nuolant plečiamos.

Praktikoje, neretai tenka vykdyti užklausas kai lentelėse yra labai daug eilučių. Tuomet, testuojant užklausas, patogu pasinaudoti **rezultato eilučių ribojimo** galimybe. Sakinio `SELECT` gale fraze `FETCH FIRST` <eilučių skaičius> `ROWS ONLY` apibrėžiama eilučių skaičiaus rezultate riba, t. y. rezultate bus pateikiama tik pirmosios eilutės, neviršijant nurodytą eilučių skaičiaus ribą. Fraze `OFFSET` <eilučių skaičius> galima nurodyti, kiek pirmųjų eilučių praleisti. Pavyzdžiui, vykdant užklausą

```
SELECT Nr, Pavardė
FROM Vykdytojai
ORDER BY Nr DESC
FETCH FIRST 2 ROWS ONLY OFFSET 1,
```

sistema lentelės *Vykdytojai* eilutes peržiūrės *Nr* mažėjimo tvarka, praleis pirmąją didžiausiu *Nr* eilutę (dėl `OFFSET` 1) ir rezultate pavaizduos dvi kitas eilutes:

Nr	*Pavardė*
4	Onaitytė
3	Gražulytė

`SELECT` frazėje, panaudojus funkciją `ROW_NUMBER()` `OVER` (<rikiavimo tvarka>), užklausos **rezultato eilutes sunumeruosime**, t. y. turėsime stulpelį, kuriame bus užklausos rezultato eilutės numeris [10, 11]. Rezultato eilutės numeruojamos frazėje `OVER` nurodyto stulpelio reikšmių didėjimo ar mažėjimo tvarka, pvz.,

```
SELECT ROW_NUMBER() OVER (ORDER BY Pavadinimas ASC) AS Eilė,
       Nr, Pavadinimas
FROM Projektai
ORDER BY Pavadinimas ASC,
```

Eilė	*Nr*	*Pavadinimas*
1	2	Buhalterinė apskaita
2	1	Studentų apskaita
3	3	WWW svetainė

Rezutato eilutės stulpelyje *Eilė* yra tos eilutės stulpelyje *Pavadinimas* esančios reikšmės eilės numeris tarp visų stulpelio *Pavadinimas* reikšmių, skaičiuojant reikšmes alfabeto tvarka.

2.16. Schemos

Paprastai lentelės duomenų bazėje priklauso kuriai nors schemai. **Schema** (angl. *schema*) – tai struktūrinis DB vienetas, kuriame būna lentelės ir kiti DB objektai ir kuriame kitų schemų jau negali būti [7, 8, 12, 13]. Schema nusakoma jos pavadinimu – tai schemos tapatumo požymis.

Panašiai, kaip kompiuterio failinėje sistemoje gali būti keli failai tuo pačiu pavadinimu, bet skirtinguose aplankuose (angl. *folder*), taip DB-je gali būti kelios lentelės vienodu pavadinimu, bet skirtingose schemose. Panašiai, kaip failų pavadinimai gali būti tikslinami

aplanko pavadinimu, kuriame failas yra, taip ir lentelių bei kitų DB objektų pavadinimai gali būti tikslinami schemos pavadinimu, rašant šį prieš objekto pavadinimą ir tarp jų dedant tašką: <schemos vardas>.<DB objekto vardas>. Pavyzdžiui, *Stud.Vykdytojai* ir *Mano.Vykdytojai* žymi dvi skirtingas leneteles vienodu pavadinimu *Vykdytojai*, bet esančias skirtingose schemose (vietose) *Stud* ir *Mano*. Kai lentelės ar kito DB objekto vardas rašomas be schemos, suprantama, kad jis yra numatytoje schemoje, kurios vardas neretai sutampa su DB naudotojo vardu.

Schemos kuriamos SQL sakiniu [7, 8, 12, 13]:

`CREATE SCHEMA` <schemos vardas> [<schemos parametrai>].

Tarp schemos parametrų gali būti nurodytas schemos savininkas, išvardinti DB objektų, kurie galės būti kuriami schemoje, rūšys ir kiti parametrai.

Iki šiol SQL sakiniuose rašant lentelių pavadinimus nebuvo naudotas schemos pavadinimas, laikant kad lentelės yra numatytoje schemoje.

Tuščią schemą, kurioje nėra jokių lentelių ir kt. DB objektų galima sunaikinti SQL sakiniu:

`DROP SCHEMA` <schemos vardas> .

2.17. Sisteminis katalogas

Kad galėtų sėkmingai vykdyti vartotojo užklausas ir valdyti duomenis, DBVS turi „įsiminti“ gana didelį kiekį informacijos, susijusios su DB struktūra. Reliacinėje DB tokia informacija saugoma **sisteminiame kataloge** – sistemos **informacinėse lentelėse**, kurias DBVS naudoja savo vidinėms reikmėms [7, 8, 12, 13]. Sisteminiame kataloge saugomas ir DB struktūros (lentelių, stulpelių ir kt.) aprašas.

Nors sisteminis katalogas, visų pirma, skirtas sistemos vidinėms reikmėms, jame esanti informacija prieinama ir DBVS vartotojams. Reliacinėje DB yra gana detalus jos pačios aprašas, kurį vartotojas gali sužinoti užklausomis.

Sisteminio katalogo lentelės automatiškai sudaromos kuriant DB. Vartotojui nereikia rūpintis jų turiniu. Sistema pati rūpinasi, kad sisteminis katalogas atspindėtų einamąją DB būseną. Vartotojui draudžiama tiesiogiai keisti sisteminio katalogo turinį – tai išskirtinė DBVS privilegija. Vartotojas gali tik užklausti sisteminio katalogo duomenų.

Vykdydama SQL sakinius DBVS nuolat kreipiasi į sisteminį katalogą. Pavyzdžiui, kad įvykdytų užklausą dviem lentelėms, DBVS turi:

- patikrinti, ar egzistuoja tos dvi lentelės;
- patikrinti, ar einamasis vartotojas turi teisę kreiptis į abi lenteles;
- patikrinti, ar lentelėse yra stulpeliai, nurodyti užklausoje;
- nustatyti, kuriai lentelei priklauso stulpeliai, kurių vardai užklausoje nurodyti be lentelės pavadinimo;
- kiekvienam stulpeliui nustatyti duomenų tipą.

Visa ši informacija yra iš anksto apibrėžtose sisteminio katalogo lentelėse. Todėl DBVS informacijos paieškai sisteminiame kataloge gali naudoti ypač efektyvius metodus ir algoritmus.

Paprastai, visos informacinės lentelės yra tam skirtoje DB schemoje, dažniausiai schemoje *Information_schema* [7, 8, 12, 13, 16]. Praktiškai visose platinamose RDBVS tarp kitų sisteminio katalogo lentelių yra lentelės, aprašančios visas DB lenteles ir visų lentelių

visus stulpelius, pvz. *Information_schema.Tables* ir *Information_schema.Columns*. Šios dvi lentelės, kaip ir visos kitos sisteminio katalogo lentelės, taip pat yra aprašytos jose pačiose. Todėl užklausa:

```
SELECT * FROM Information_schema.Columns
```

galima sužinoti išsamią informaciją apie visų DB lentelių (tarp jų ir lentelės *Information_schema.Columns*) stulpelius. Detalesne užklausa:

```
SELECT Column_name FROM Information_schema.Columns
WHERE Table_schema = 'information_schema' AND Table_name = 'tables'
```

sužinosime sisteminio katalogo lentelės *Information_schema.Tables*, kurioje yra DB visų lentelių aprašai, stulpelių pavadinimus. Schemos pavadinimą *Information_schema* šioje užklausoje rašėme ir iš didžiosios, ir iš mažosiomis raidės. SQL sakinyje, kaip įprasta, tiek schemos, tiek lentelės, tiek ir kitų DB objektų pavadinimus galima rašyti įvairiai – tai neturi įtakos. Tačiau paieškos sąlygoje, tarp apostrofų pavartotas schemos pavadinimas 'information_schema' ir lentelės pavadinimas 'tables' būtinai turi būti užrašyti tiksliai taip, kaip jie įrašyti DB-je, nes tai tekstinės konstantos, šiuolaikinėse DBVS, dažniausiai - mažosiomis raidėmis. Šioje užklausoje, *Table_name* žymi sisteminio katalogo lentelės *Information_schema.Columns* stulpelio pavadinimą, o 'tables' – tos lentelės stulpelio *Table_name* reikšmę.

Sisteminio katalogo lentelių ir jų stulpelių pavadinimai yra pakankamai informatyvūs – iš pavadinimo galima suprasti jo paskirtį. Tarus, kad visos sisteminio katalogo lentelės yra schemoje *Information_schema*, visų sisteminio katalogo lentelių pavadinimus, išdėstytus abėcėlės tvarka, sužinosime įvykdžius užklausą:

```
SELECT Table_name FROM Information_schema.Tables
WHERE Table_schema = 'information_schema'
ORDER BY 1.
```

Toks pat lentelių pavadinimų sąrašas bus gautas, tik ne taip sparčiai, nes teks peržiūrėti žymiai daugiau duomenų turinčią lentelę, įvykdžius ir tokią užklausą:

```
SELECT DISTINCT Table_name FROM Information_schema.Columns
WHERE Table_schema = 'information_schema'
ORDER BY 1.
```

Kadangi lentelėje *Information_schema.Columns* kiekvienai duomenų bazės lentelei gali būti po keletą eilučių (tiek, kiek lentelėje yra stulpelių), tai raktinis žodis `DISTINCT` užklausoje užtikrina, kad rezultate bus pateikti tik skirtingi lentelių pavadinimai. Taip gaunamuose sisteminio katalogo lentelių pavadinimų sąrašuose nėra lentelių schemų pavadinimų, nes konkreti lentelių schema yra nurodyta paieškos sąlygoje. Tarp šių užklausų rezultate gautų reikšmių bus ir 'columns' - lentelės *Information_schema.Columns* pavadinimas be schemos.

Jau minėjome, kad duomenų bazėje gali būti keletas lentelių tuo pačiu pavadinimu, bet skirtingose schemose. Visą lentelės pavadinimą sudaro schemos ir lentelės pavadinimai kartu. Išsamius visų duomenų bazės lentelių pavadinimus galima sužinoti užklausa:

```
SELECT Table_schema, Table_name FROM Information_schema.Tables
ORDER BY 1, 2.
```

Taip pat minėjome, kad, sukuriant lentelę, kiekvienam stulpeliui yra priskiriamas duomenų tipas. Detaliau tipus aptarsime vėliau. Dabar, kiekvienam stulpelio tipui apskaičiuokime, kiek yra lentelių, turinčių bent vieną tokio tipo stulpelį. Stulpelių tipų vardai yra užrašyti sisteminės lentelės *Information_schema*.*Columns* stulpelyje *Data_Type*. Siekiamą rezultatą gausime, įvykdę užklausą:

```
SELECT Data_Type, COUNT(DISTINCT Table_schema || Table_name)
FROM Information_schema.Columns
GROUP BY Data_Type.
```

Šioje užklausoje, norėdami suskaičiuoti tik skirtingas lenteles, skaičiuojame skirtingas lentelių schemų ir jų vardų poras. Tam funkcijos `COUNT` argumente taikome simbolių eilučių jungimo operaciją. Atkreipsime dėmesį į tai, kad reiškinys `COUNT(DISTINCT` *Table_schema*, *Table_name*) yra neteisingas. Funkcija `COUNT` yra vienvietė (vieno argumento). Todėl, norint skaičiuoti skirtingus kelių stulpelių reikšmių rinkinius, reikalingas reiškinys, apjungiantis dvi reikšmes (schemos ir lentelės pavadinimus) į vieną. Dar teisingiau būtų skaičiuojama, jei tarp lentelės schemos ir vardo panaudotume kokį nors skirtuką, pvz.,

`COUNT(DISTINCT` *Table_schema* || *'.'* || *Table_name*).

Paliekame skaitytojui pagalvoti, kodėl skirtukas užtikrintų teisingesnį atsakymą. Antra vertus, daugelis DBVS funkcijos `COUNT` argumentą, sudarytą iš kelių stulpelių, apskliaustų riestiniais skliausteliais, interpretuoja, kaip sudėtinę reikšmę ir leidžia rašyti taip:

```
SELECT Data_type, COUNT(DISTINCT (Table_schema, Table_name))
FROM Information_schema.Columns
GROUP BY Data_type.
```

Detalesnį sisteminio katalogo aprašą galima rasti konkrečios DBVS dokumentacijoje.

2.18. Pratimai

1. Sudarykite užklausas šiems duomenims DB *Darbai* surasti:
   a) Visų projektų pavadinimai ir jų pradžios. Rezultatą pateikti, sutvarkytą pagal projektų pradžios datą.
   b) Visų užregistruotų projektų skaičius.
   c) Visų didelės svarbos projektų skaičius.
   d) Pavardės visų vykdytojų, kurie vykdo mažos svarbos projektus.
   e) Pavardės vykdytojų, dalyvaujančių bent dviejuose projektuose.
   f) Pavadinimai visų projektų, kurių pabaigos data jau praėjo.
   g) Kiekvieno projekto pavadinimas, jo pradžios data ir dienų, praėjusių nuo jo vykdymo pradžios iki dabar, skaičius.
   h) Pavadinimai projektų, kuriuos nevykdo nė vienas vykdytojas.
   i) Pavardės vykdytojų, kurie vadovauja bent vienam projektui.
   j) Pavardės vykdytojų, kuriems bendras projektams skiriamų valandų skaičius yra didesnis už visų vykdytojų bendrųjų valandų vidurkį.
   k) Visos lentelės, esančios schemoje *STUD*.
2. Apibūdinkite ir apskaičiuokite šių užklausų rezultatus:
   a) `SELECT COUNT(`*Išsilavinimas*`) FROM` *Vykdytojai*

b) SELECT COUNT(DISTINCT *Išsilavinimas*) FROM *Vykdytojai*
c) SELECT COUNT(DISTINCT *Projektas*) FROM *Vykdymas*
d) SELECT COUNT(*Projektas*) FROM *Vykdymas*
e) SELECT *Pavardė, Statusas, Valandos*
FROM *Vykdytojai* LEFT OUTER JOIN *Vykdymas* ON *Vykdytojas* = *Nr*
WHERE *Projektas* = 1
f) SELECT *Pavardė, Statusas, Valandos*
FROM *Vykdytojai* LEFT OUTER JOIN *Vykdymas* ON *Vykdytojas* = *Nr*
WHERE *Projektas* = 1 OR *Projektas* IS NULL

3. Tarkime, turime lentelę *L(A,B,C)*, kuri jau yra užpildyta duomenimis:

A	*B*	*C*
2	1	3
1	2	3
2	2	3

Užrašykite šių užklausų rezultatus:
a) SELECT DISTINCT *A* FROM *L* ORDER BY 1 DESC
b) SELECT *A* FROM *L* WHERE *A* IN (SELECT *B* FROM *L*)
c) SELECT *A,* MAX(*B*) , SUM(*C*) FROM *L* GROUP BY *A*
d) SELECT COUNT(*) FROM *L* AS *L1, L* AS *L2*
e) (SELECT *A* FROM *L*) INTERSECT (SELECT *B* FROM *L*)
f) (SELECT *A* FROM *L*) UNION ALL (SELECT *B* FROM *L*)
g) SELECT 1 FROM *L*
h) SELECT 1, 1 FROM *L*

Literatūra

1. R. Baronas. *Duomenų bazių valdymo sistemos*. TEV, 2005.
2. C. J. Date. *An Introduction to Database Systems*, 8th ed. Pearson, 2004.
3. J. D. Drake, J. C. Worsley. *Practical PostgreSQL*. O'Reilly Media, 2002.
4. R. Elmasri, S. Navathe. *Fundamentals of Database Systems*, 7th ed. Person, 2015.
5. L. Ferari, E. Pirozzi. *Learn PostgreSQL: Build and manage high-performance database solutions using PostgreSQL 12 and 13*, 4th ed. Packt Publishing, 2020.
6. H. Garcia-Molina, J. Ulman, J. Widom. *Database Systems: The Complete Book*, 2nd ed. Pearson, 2008.
7. J. R. Groff, P. N. Weinberg, A. Oppel. *SQL: The Complete Reference*, 3rd ed. McGraw-Hill, 2009.
8. P. Gulutzan, T. Pelzer. *SQL-99 Complete, Really*. Cmp Books, 1999.
9. G. W. Hansen, J. V. Hansen. *Database Management and Design*, 2nd ed. Prentice Hall, 1995.
10. U. Malik, M. Goldwasser, B. Johnston. *SQL for Data Analytics: Perform fast and efficient data analysis with the power of SQL*. Packt Publishing, 2019.
11. A. Molinaro. *SQL Cookbook: Query Solutions and Techniques for Database Developers*. O'Reilly Media, 2006.
12. A. Kriegel, B. M. Trukhnov. *SQL Bible*, 2nd ed., Willey, 2008.
13. P. J. Pratt, M. Z. Last. *A Guide to SQL*, 9th ed., Cengage Learning, 2014.

14. R. Ramakrishnan, J. Gehrke. *Database Management Systems*, 3rd ed. McGraw-Hill, 2002.
15. A. Silberschatz, H. F. Korth, S. Sudarshan. *Database System Concepts*, 7th ed. McGraw-Hill, 2019.
16. H.-J. Schonig. *Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13*, 4th ed. Packt Publishing, 2020.
17. The PostgreSQL Global Development Group. *PostgreSQL*, https://www.postgresql.org/
18. J. Ulman, J. Widom. *A First Course in Database Systems*, 3rd ed. Pearson, 2007.