5. Duomenų bazės sukūrimas ir užpildymas duomenimis

5.1. Įvadas

Jau aptarėme duomenų bazių loginės struktūros sudarymą. Sudarius DB gauname jos reliacinę schemą: visų atributų padalijimą į lenteles, lentelių pirminius bei išorinius raktus. Turint DB schemą, konkrečios DBVS terpėje galime fiziškai sukurti sudarytas lenteles ir kitus DB objektus.

Sukūrus lenteles, į jas galima įvesti duomenis. Duomenys įvedami duomenų įvedimo sakiniu. Tai - duomenų modifikavimo sakinys. Duomenų modifikavimo sakiniais duomenys taip pat keičiami bei šalinami. Priešingai užklausoms, kurios gali būti sudaromos kelioms lentelėms, duomenų modifikavimo sakiniais keičiama tik viena lentelė.

Esamų duomenų keitimo bei šalinimo sakiniai turi dvi formas: paieškos ir pozicinę. Šiame skyriuje aptarsime tik paieškos formos sakinius. Pozicinę sakinių formą aptarsime vėliau.

Prieš sukuriant lenteles, reikia sukurti pačią duomenų bazę.

5.2. Duomenų bazės sukūrimas

DB sistemose, funkcionuojančiose kompiuterių tinkluose, naujų DB kūrimu rūpinasi sistemos administratorius. Tam, kad būtų galima naudotis duomenų baze, esančia serveryje, vartotojo darbo vietoje reikia įdiegti DBVS vartotojo (kliento) posistemę. Tipinė DBVS vartotojo posistemė neleidžia darbo vietoje kurti savų DB. Tačiau, įdiegus reikiamus sistemos modulius, darbo vietos vartotojui – administratoriui leidžiama turėti savas DB.

Prieš kuriant DB, reikia įvertinti fizinės kompiuterio atminties poreikį būsimos DB duomenims saugoti ir parinkti tinkamą vietą. Priklausomai nuo DBVS ir operacijų sistemos ypatybių, fizinei DB galima skirti vietą kompiuterio diske, disko dalyje ar operacijų sistemos failuose. Visas tinkamas duomenų bazei fizines kompiuterio atminties vietas vadinsime **DB įrenginiu** (angl. *database device*).

SQL standarte nėra sakinio duomenų bazei kurti. Taip yra dėl DB įrenginio priklausomybės nuo konkrečios operacijų sistemos ir komercinėse DBVS esančių priemonių duomenų bazėms kurti įvairovės. Tačiau daugelyje DBVS yra komanda CREATE DATABASE ar atitinkama tarnybinė programa duomenų bazėms sukurti. DB valdymo sistemoje IBM DB2, duomenų bazę galima paprasčiausiai sukurti nurodžius DB vardą ir jos įrenginį. Pavyzdžiui,

CREATE DATABASE *Darbai* ON D:

Taip sukurtą DB *Darbai* galima sunaikinti sakiniu:

DROP DATABASE *Darbai* .

Reliacinėse DBVS svarbiausia duomenų bazės dalis yra lentelė. Savo ruožtu, lentelės pagrindinis struktūrinis elementas - stulpelis, kurio pagrindinė charakteristika yra jo duomenų tipas. SQL duomenų tipas yra reliacinės teorijos domeno atitikmuo. Prieš aptardami lentelės sukūrimą, susipažinsime su duomenų tipais.

5.3. Duomenų tipai

Stulpelio duomenų tipas apibrėžia stulpelio duomenų rūšį (simboliai, skaičiai, datos ir pan.). Reliacinės sistemos leidžia naudoti įvairius duomenų tipus. Labai svarbu stulpeliui parinkti tinkamą duomenų tipą, nes daugumoje DBVS yra gana sudėtinga stulpelio duomenų tipą pakeisti. Kita vertus, daugumoje DBVS yra gana daug funkcijų reikšmių tipui keisti. Pavyzdžiui, jei simbolių tipo stulpelyje yra tik skaičiai, pakeisti stulpelio tipą į skaičių tipą gali būti neįmanoma. Tačiau, specialiomis funkcijomis galima atlikti aritmetines operacijas su tokiais simboliniais duomenimis.

Daugelyje reliacinių DBVS naudojami tokie duomenų tipai:

- **Simbolių duomenų tipai** (angl. *character datatypes*). Šių tipų reikšmės yra raidžių, skaitmenų ir specialųjų ženklų sekos – simbolių eilutės. Simbolių kiekis eilutėje vadinamas eilutės ilgiu. Kiekvienam simbolių duomenų stulpeliui būtina nurodyti didžiausią leistiną eilutės ilgį. Skiriamos fiksuoto ir kintamo ilgio simbolių eilutės. Fiksuoto ir kintamo ilgio simbolių eilutės skiriasi jų fiziniu vaizdavimu ir panaudojimo ypatybėmis.

  DB valdymo sistemoje IBM DB2 fiksuoto ilgio simbolių eilutės didžiausias ilgis gali svyruoti nuo 1 iki 254. Fiksuoto ilgio simbolių eilučių tipas žymimas `CHAR`(n), kur n yra didžiausias leistinas ilgis. Fiksuoto ilgio simbolių eilutei fizinėje kompiuterio atmintyje visuomet skiriama tiek baitų, kiek jų reikia didžiausio leistino ilgio simbolių eilutei, nepriklausomai nuo faktinio eilutės ilgio.

  Kintamo ilgio simbolių eilutės gali būti vieno iš trijų tipų: `VARCHAR`(n), `LONG VARCHAR`(n) ir `CLOB(`n[K|M|G]), kur n – didžiasias leistinas eilutės ilgis, kuris gali būti nurodytas K-kilo, M-mega ar G-giga baitais. Pagal nutylėjimą ilgis skaičiuojamas baitais. Šie trys tipai skiriasi didžiausiu eilutės ilgiu: `VARCHAR` (angl. *variable character*) tipo eilutė gali užimti atmintyje iki 4000 baitų, `LONG VARCHAR` – iki 32.700, o `CLOB` (angl. *character large object*) - net iki 2 gigabaitų (2GB).

  SQL riboja ilgų simbolių eilučių panaudojimą. Pvz., simbolių duomenų tipo stulpelių, kurių reikšmių didžiausias leistinas ilgis yra didesnis už 254, negalima naudoti `SELECT` sakinyje su fraze `DISTINCT.` Tokių stulpelių negalima naudoti ir `GROUP BY` bei `ORDER BY` frazėse. Panašių apribojimų yra ir daugiau. Dauguma šių apribojimų paaiškinami šių operacijų realizacijos sudėtingumu. Baitų kiekis, kurie išskiriami atmintyje kintamo ilgio simbolių eilutei, priklauso nuo faktinio jos ilgio.

  SQL sakiniuose simbolių duomenų tipų reikšmės (konstantos) yra išskiriamos apostrofais, pvz., ‘Matematikos ir informatikos fakultetas’, ‘Naugarduko 24, Vilnius’.

- **Skaičių duomenų tipai (**angl. *number datatypes***)**. Šių tipų reikšmės **– s**veiki ir trupmeniniai skaičiai. Šie tipai visuomet turi tikslumą - skaičiaus skaitmenų arba baitų kiekį.

  `SMALLINT` žymi “mažus” sveikus skaičius. Tai - skaičiai, kuriuos galima pavaizduoti dviejuose baituose: nuo -32768 iki 32767, pavyzdžiui, 23, +56, -789, 8965.

  `INTEGER` - “dideli” sveiki skaičiai, kurių vaizdui kompiuteryje reikia 4 baitų: nuo -2.147.483.648 iki 2.147.483.647. Pvz.: 589654, 23, -545545445.

  `REAL` - slankaus kablelio skaičiai, kuriems vaizduoti atmintyje skiriami 4 baitai. Pvz.: 1.02, -2E5, 5.555E-18, -.655645e8.

  `DOUBLE` arba `FLOAT` - dvigubo tikslumo skaičiai (skiriama 8 baitai) su slankiu kableliu. Tai – skaičiai, patenkantys į intervalus [-1.79769e308, -2.225e307] ir [2.225e-307, 1.79769e308]. Pvz.: 23E5, -2.555E-58, .655645e98.

  `DECIMAL` arba `NUMERIC` - dešimtainiai skaičiai, kurių tikslumas priklauso nuo jų apibrėžimo, bet ne didesnis nei 31 dešimtainis skaitmuo. Apibrėžiant tokio tipo lentelės stulpelį reikia nurodyti bendrą skaitmenų kiekį ir skaitmenų kiekį po dešimtainio kablelio (taško). Bendras dešimtainių skaičių tipo žymuo yra `DECIMAL(`n, m**)**, kur n - bendras skaitmenų kiekis ir m - skaitmenų po kablelio kiekis. Dešimtainių konstantų pavyzdžiai: 15, -125.5, 1234569012345.123.

- **Dvejetainių duomenų tipai** (angl. *binary datatypes*). Šių tipų stulpeliai skirti dvejetainiams kodams saugoti. Kaip ir simbolių eilutės, dvejetainiai kodai gali būti fiksuoto (`BIT`(n)) ir kintamo (`BIT VARYING`(n) ir `BLOB(`n[K|M|G])) ilgio, kur n – didžiausias leistinas dvejetainio kodo ilgis baitais. `BLOB` (angl. *binary large object*) tipo duomenys gali būti iki 2 gigabaitų (2GB). Dvejetainiais kodais saugomi grafiniai vaizdai, audio ar video įrašai ir pan.

- **Datos ir laiko duomenų tipai**. Daugelyje DBVS yra trys šios rūšies duomenų tipai: `DATE` – datos duomenų tipas, `TIME` – laiko duomenų tipas ir `TIMESTAMP` - tikslaus laiko duomenų tipas. Šių tipų duomenys kompiuterio atminyje vaizduojami vidiniu formatu,

kuris vartotojui neprieinamas. DATE tipo reikšmėms (datoms) kompiuterio atmintyje skiriama 4 baitai, TIME (laikui) - 3 baitai, o TIMESTAMP (data ir tikslus laikas) - 10 baitų. SQL sakiniuose data ir laikas vaizduojami ne vidiniame formate, bet vartotojui įprastai - simbolių eilutėmis. Datos ir laiko vaizdavimas priklauso nuo vartotojo terpės bei DB parametrų, pavyzdžiui, šalies kodo (angl. *country code*). Lietuvoje data paprastai vaizduojama 10 simbolių eilute, o laikas - 8 simboliais, pvz., '2001.01.01', '12:00:00'. Tikslaus laiko duomenų formatas nepriklauso nuo terpės. DBVS DB2 - tai 26 simbolių eilutė, pvz.,' 2001-01-01-12.15.55.330000'.

Visi SQL duomenų tipai turi ypatingą reikšmę NULL, skirtą pažymėti, kad stulpelis eilutėje neturi prasminės reikšmės.

Parinkus stulpeliui tinkamą duomenų tipą, reikia parinkti duomenų tipo ilgį ir tikslumą. Parenkant duomenų tipui ilgį reikia atsižvelgti į faktiškai išskiriamos atminties dydžio priklausomybę nuo reikšmės, t.y. ar reikšmei išskiriamas baitų kiekis priklauso nuo konkrečios reikšmės. Pavyzdžiui, visoms stulpelio, apibrėžto CHAR(40), reikšmėms kompiuterio atmintyje bus skiriama tiek baitų, kiek jų reikia keturiasdešimčiai simbolių kodų (40 baitų), nepriklausomai nuo to, kiek faktiškai simbolių įvedama. Ilgesnės įvestos reikšmės bus patrumpinamos, o trumpesnės – papildomos tarpais. Reikšmei 'Jonaitis' skiriama 40 baitų. Jei iš anksto žinoma, kad dauguma reikšmių bus trumpesnės už apibrėžime nurodytą ilgį, tai galimas didelis atminties pereikvojimas. Taip atsitinka parenkant tipą stulpeliui, skirtam pavardėms. Žinoma, kad lietuvišką pavardę sudaro iki 40 simbolių, nors daugumai pavardžių užrašyti užtenka 20 raidžių.

Kintamo ilgio duomenys atmintyje užima tiek baitų, kiek jų reikia faktinei reikšmei užrašyti kompiuterio atmintyje. Pavyzdžiui, parinkus stulpeliui VARCHAR(40) tipą, ilgesnės reikšmės, kaip ir fiksuoto ilgio atveju, patrumpinamos. Tačiau trumpesnės reikšmės nepapildomos tarpais. Kintamo ilgio duomenų tipo atveju, reikšmei 'Jonaitis' skiriami 8 baitai.

Parenkant stulpeliui kintamo ilgio duomenų tipą, reikia prisiminti, kad kiekvienai reikšmei reikia saugoti ne tik ją pačią, bet ir faktišką jos ilgį arba jos pabaigos požymį. Todėl apibrėžimas VARCHAR(1) yra teisingas, bet neprasmingas. Be to, kintamo ilgio duomenų apdorojimas yra mažiau efektyvus negu fiksuoto ilgio.

Dauguma skaičių duomenų tipų yra fiksuoto ilgio ir tikslumo. Tik dešimtainių skaičių apibrėžime galima nurodyti tikslumą. Dešimtainiai skaičiai yra dažnai naudojami finansiniuose uždaviniuose. Pastebėsime, kad mūsų šalies biudžetą negalima išreikšti slankaus kablelio skaičiumi centų tikslumu, nes netgi dvigubo tikslumo reikšmėje neužtenka reikšminių skaitmenų. Dešimtainiuose skaičiuose gali būti net 31 reikšminis skaitmuo. DECIMAL(31,2) tipo reikšme galėsime tiksliai išreikšti net ir didelės šalies biudžetą.

5.4. Lentelių apibrėžimas

Lentelės (struktūros) apibrėžimas yra "aktyvus". SQL DDL sakiniu CREATE TABLE ne tik apibrėžiama duomenų struktūra, bet ir sukuriamas realus DB objektas. Lentelės apibrėžimas (kūrimas) yra vienkartinis veiksmas. DBVS neleidžia kurti lentelės su tokiu pat vardu kaip jau egzistuojančios. Kuriant (apibrėžiant) lentelę, būtinai nurodoma:

- lentelės vardas, kurį galima patikslinti schemos vardu;
- lentelės stulpelių vardai ir jų tipai.

Bendruoju atveju, sakinyje CREATE TABLE galima nurodyti daugiau lentelės savybių, pavyzdžiui, ar stulpelyje yra galimos NULL reikšmės. Kiekvienam stulpeliui galima nurodyti reikšmę pagal nutylėjimą (angl. *default value*), kuri priskiriama stulpeliui, kai įvedamoje eilutėje jam nenurodyta jokia reikšmė.

DB *Darbai* lenteles *Vykdytojai*, *Projektai* ir *Vykdymas* galima sukurti taip:

```
CREATE TABLE Vykdytojai (
    Nr          INTEGER     NOT NULL,
    Pavardė     CHAR(30)    NOT NULL,
```

```
    Kvalifikacija   CHAR(16)      DEFAULT 'Informatikas',
    Kategorija      SMALLINT,
    Išsilavinimas   CHAR(10)),

CREATE TABLE Projektai (
    Nr              INTEGER       NOT NULL,
    Pavadinimas     VARCHAR(254)  NOT NULL,
    Svarba          CHAR(10)      DEFAULT 'Vidutinė',
    Pradžia         DATE,
    Trukmė          SMALLINT),

CREATE TABLE Vykdymas (
    Projektas       INTEGER       NOT NULL,
    Vykdytojas      INTEGER       NOT NULL,
    Statusas        VARCHAR(32)   DEFAULT 'Programuotojas',
    Valandos        SMALLINT).
```

Šiuose trijuose SQL sakiniuose fraze `NOT NULL` pažymima, kad stulpelyje negali būti `NULL` reikšmės. Tuomet DBVS pasirūpins, kad jokiomis aplinkybėmis, jokioje eilutėje taip neatsitiktų. Stulpelyje galimos `NULL` reikšmės, jei frazė `NOT NULL` nenurodyta. Fraze `DEFAULT` apibrėžiama reikšmė pagal nutylėjimą.

Apibrėžiant lentelę, papildomai galima nurodyti pirminį raktą bei išorinius raktus. Šias ir kitas galimybes aptarsime vėliau. Daugumą lentelės savybių galima apibrėžti vėliau, jau sukurtai lentelei. Tai atliekama sakiniu `ALTER TABLE`. Juo galima papildyti lentelę, pavyzdžiui, nauju stulpeliu. Tačiau keičiant lentelės struktūrą, susiduriama su apribojimais. Dažnai DB valdymo sistemos neleidžia naujiems stulpeliams nurodyti savybės `NOT NULL`. Papildykime lentelę *Vykdytojai* gimimo datos stulpeliu:

```
ALTER TABLE Vykdytojai ADD Gimtadienis DATE.
```

SQL standarte numatyta galimybė sakiniu `ALTER TABLE` pašalinti lentelės stulpelį, pavyzdžiui,

```
ALTER TABLE Vykdytojai DROP Gimtadienis.
```

Tačiau konkrečios DBVS neleidžia šalinti stulpelių. Sakiniu `ALTER TABLE` taip pat galima keisti stulpelio savybes, pavyzdžiui, reikšmę pagal nutylėjimą:

```
ALTER TABLE Projektai ALTER Svarba DROP DEFAULT;
ALTER TABLE Projektai ALTER Svarba SET DEFAULT 'Didelė'.
```

5.5. Naujų duomenų įvedimas

Nauji duomenys į DB bazę įvedami sakiniu `INSERT`, įterpiant eilutes į pasirinktą lentelę. Šis sakinys yra dviejų formų. Pirmosios formos sakiniu įterpiama viena nauja eilutė. Stulpelių reikšmės nurodomos pačiame sakinyje:

```
INSERT INTO <lentelės vardas> [(<stulpelio vardas>{, <stulpelio vardas>})]
VALUES     (<reikšmė> {,<reikšmė>})
<reikšmė> ::= <reiškinys> | NULL | DEFAULT .
```

Šiuo sakiniu į lentelę įterpiama viena nauja eilutė. Išvardintiems stulpeliams priskiriamos reikšmes, esančios už bazinio žodžio `VALUES`. Tarp stulpelių ir jų reikšmių nustatoma atitinkamybė "iš kairės į dešinę". Jei stulpelių vardai praleisti, laikoma, kad stulpeliai išvardinti ta tvarka, kuria jie buvo išvardinti sukuriant lentelę. Jei kuriam nors lentelės stulpeliui nenurodyta reikšmė, tai jis įgyja `NULL` reikšmę arba reikšmę pagal nutylėjimą, jei tokia reikšmė stulpeliui buvo priskirta. Sakinio vykdymas baigiasi nesėkme, jei stulpeliui su

savybe NOT NULL, sakinyje nenurodyta jokia prasminė reikšmė ir jam neapibrėžta reikšmė pagal nutylėjimą arba tiesiog nurodyta reikšmė NULL.

Užregistruokime naują besimokantį bendradarbį pavarde Baltakis. Suteikime jam informatiko kvalifikaciją ir antrą kategoriją. Jo duomenis galima įvesti vienu iš šių sakinių:

```
INSERT INTO Vykdytojai (Nr, Pavardė, Kvalifikacija, Kategorija, Išsilavinimas)
            VALUES (6, 'Baltakis', 'Informatikas', 2, NULL),

INSERT INTO Vykdytojai VALUES (6, 'Baltakis', 'Informatikas', 2, NULL),

INSERT INTO Vykdytojai (Nr, Pavardė, Kategorija) (6, 'Baltakis', 2),
```

Antrosios formos INSERT sakiniu galima įterpti į lentelę užklausos rezultatą:

```
INSERT INTO <lentelės vardas> [(<stulpelio vardas>{, <stulpelio vardas>})]
            <užklausa> .
```

Šiuo sakiniu visos užklausos rezultato eilutės įtraukiamos į pasirinktą lentelę. Atitinkamybė tarp lentelės stulpelių ir užklausos rezultato stulpelių nustatoma taip pat, kaip pirmosios formos sakinyje.

Galimybė įvesti duomenis, kurie jau yra kurioje nors vienoje ar keliose lentelėse, gali pasirodyti gana keista. Ji naudinga duomenims administruoti. Kartais lentelėse susikaupia labai daug duomenų. Dėl to darbas su lentele gali pastebimai sulėtėti. Tarkime, kad seni duomenys naudojami daug rečiau nei nauji. Tuomet senuosius duomenis galima perkelti į archyvo lentelę. Susikaupus labai daug duomenų, paieška archyve sulėtės, tačiau duomenų paieška pagrindinėje lentelėje, kuri naudojama dažniau, išliks greita. Be to, po tam tikro laiko gali tapti netikslinga saugoti didelį kiekį senų duomenų. Dažnai jau po kelerių ar net vienerių metų išsamūs duomenys tampa nereikalingais, nes pakanka tik mėnesinių ataskaitų. Todėl patogu turėti atskirą lentelę ataskaitoms kaupti, į kurią sugrupuoti pagrindinės lentelės duomenys yra periodiškai perkeliami.

Tarkime, lentelėje *Projektai* saugomi duomenys tik apie projektus, kurie vykdomi dabar ar pasibaigė ne seniau nei prieš metus. Duomenys apie senesnius projektus kaupiami kitoje lentelėje *Seni_Projektai*, kurios struktūra yra tokia pat kaip lentelės *Projektai*. Įtraukime į lentelę *Seni_Projektai* duomenis apie prieš metus ar anksčiau pasibaigusius projektus:

```
INSERT INTO Seni_Projektai
SELECT * FROM Projektai
WHERE Pradžia + Trukmė MONTHS < CURRENT DATE – 1 YEAR.
```

Taip lentelės *Projektai* duomenys kopijuojami į lentelę *Seni_Projektai*. Seni duomenys išlieka ir lentelėje *Projektai*. Kaip pašalinti duomenis iš lentelės sužinosime vėliau.

Duomenis į lentelę galima įvesti ir specialiomis programomis. Duomenų, esančių faile, įterpimas į lentelę vadinamas **duomenų importu**. Įvairiose DBVS vartojami skirtingi duomenų failų formatai. Greta duomenų importo programų yra ir duomenų eksporto programos. **Duomenų eksportas** leidžia užklausos rezultatą išsaugoti faile pasirinktame formate. Vėliau duomenis galima importuoti iš failo į pasirinktą DB lentelę. Kai kurie duomenų formatai, pavyzdžiui, ASCII, yra bendrai naudojami daugelyje komercinių DBVS. Taip galima apsikeisti duomenimis tarp DB, funkcionuojančių skirtingose DBVS.

5.6. Duomenų šalinimas

Kai realaus pasaulio objektas "išnyksta", prireikia pašalinti eilutę iš lentelės. Pavyzdžiui darbuotojui išėjus iš darbo, saugoti jo duomenis bazėje tampa netikslinga. Objektui išnykus, pašalinami ir jį atitinkantys duomenys, kad DB išliktų tiksliu realaus pasaulio modeliu. Panašiai yra su projektais. Užbaigus vykdyti projektą, tikslinga atitinkamas eilutes pašalinti iš lentelių *Projektai* ir *Vykdymas*. Tačiau nepašalinus šių eilučių nereiškia, kad DB tampa prieštaringa. Tiesiog tuomet laikoma, kad bazėje saugomi duomenys ne tik apie šiuo metu vykdomus projektus, bet ir apie anksčiau vykdytus.

Lentelės eilutės šalinamos SQL sakiniu:

DELETE FROM <lentelės vardas> [WHERE <paieškos sąlyga>]

Frazėje FROM nurodoma lentelė, iš kurios eilutes reikia šalinti. Fraze WHERE nurodoma šalintinų eilučių atrinkimo sąlyga. Atliekant sakinį visos (!) tenkinančios paieškos sąlygą eilutės pašalinamos iš lentelės. Iš lentelės pašalinamos visos eilutės, jei frazės WHERE nėra. Sakiniu

DELETE FROM *Vykdymas*

pašalinamos visos lentelės *Vykdymas* eilutės. Lentelė (jos apibrėžimas), žinoma, išlieka. Lentelė tampa tuščia. Todėl, vartojant duomenų šalinimo sakinį, reikia būti ypač atidiems nurodant paieškos sąlygą, kad nepašalinti perdaug ar ne tas eilutes.

Jau pateikėme sakinį lentelės *Projektai* eilutėms, atitinkančioms prieš metus pabaigtus projektus, įvesti į tokios pat struktūros lentelę *Seni_Projektai*. Užrašykime sakinį perkeltoms į "archyvą" eilutėms pašalinti iš pradinės lentelės:

DELETE FROM *Projektai*
WHERE *Pradžia* + *Trukmė* MONTHS < CURRENT DATE – 1 YEAR.

Tarkime, Baltakis išėjo iš darbo. Pašalinkime eilutę, atitinkančią šį žmogų, iš lentelės *Vykdytojai*, be to pašalinkime visas eilutes, susijusias su jo dalyvavimu vykdant projektus:

DELETE FROM *Vykdymas*
WHERE *Vykdytojas* = (SELECT *Nr* FROM *Vykdytojai* WHERE *Pavardė* = 'Baltakis'),

DELETE FROM *Vykdytojai* WHERE *Pavardė* = 'Baltakis'.

Pastebėsime, kad šiuos du sakinius reikia vykdomi būtent tokia tvarka. Vėliau sužinosime, kad papildžius lentelių *Vykdymas* ir *Vykdytojai* apibrėžimus, pirminiais ir išoriniais raktais, pirmasis šalinimo sakinys iš pastarųjų dviejų taps nereikalingu. Dėl išorinio rakto, pašalinus eilutę iš lentelės *Vykdytojai*, lentelės *Vykdymas* reikiamos eilutės bus pašalintos automatiškai.

Pateiktas sakinys lentelės *Vykdymas* eilutėms šalinti nėra vienintelis. Užrašysime dar kelis sakinius, lentelės *Vykdymas* eilutėms, susijusioms su Baltakio dalyvavimu projektuose, šalinti:

DELETE FROM *Vykdymas*
WHERE 'Baltakis' = (SELECT *Pavardė* FROM *Nr* = *Vykdytojas*),

DELETE FROM *Vykdymas*
WHERE EXISTS (SELECT * FROM *Vykdytojai*
WHERE *Pavardė* = 'Baltakis' AND *Nr* = *Vykdytojas*).

Abiejuose šiuose sakiniuose pavartotos užklausos yra priklausomos (žr. skyrelį 2.6), kadangi turi parametrą (*Vykdytojas*), kuris įgyja reikšmę užklausos išorėje. Todėl, pastarieji du sakiniai rodo kaip nereiktų daryti!

5.7. Esamų duomenų keitimas

Duomenų bazėje saugomus duomenis reikia atnaujinti, kai įvyksta pasikeitimai atitinkamuose realaus pasaulio objektuose. Tarkime, firmos darbuotoja Gražulytė baigė Vilniaus universitetą ir dėl to jai buvo pakelta kategorija vienetu. Tam, kad DB išliktų tiksliu realaus pasaulio modeliu, reikia pakeisti lentelės *Vykdytojai* eilutę.

Pasirinktos lentelės duomenys keičiami tokiu sakiniu:

UPDATE <lentelės vardas> SET <stulpelio vardas> = <reiškinys>
{,<stulpelio vardas> = <reiškinys>}
[WHERE <paieškos sąlyga>]

Fraze WHERE nurodoma keistinų eilučių atrinkimo sąlyga. Atliekant sakinį atnaujinamos visos (!) atitinkančios paieškos sąlygą lentelės eilutės. Fraze SET nurodomi stulpeliai, kurių reikšmes norime keisti, bei naujosios reikšmės.

Minėti Gražulytės gyvenimo pasikeitimai atsispindės duomenų bazėje įvykdžius sakinį

UPDATE *Vykdytojai* SET *Išsilavinimas* = 'VU', *Kategorija* = *Kategorija* + 1
WHERE *Pavardė* = 'Gražulytė' .

Nurodant atnaujinamų duomenų paieškos sąlygą, reikia būti nemažiau atidiems, negu šalinant duomenis. Jei pastarąjį sakinį, pavyzdžiui, įvykdytume, nenurodę jokios paieškos sąlygos, tai kategorija būtų padidinta bei išsilavinimas taptų vienodu visiems vykdytojams.

Dar kartą pašalinti jau pašalintus duomenis neįmanoma. Pakartotinas duomenų šalinimo sakinio įvykdymas lentelės turinio nepakeis. Keičiant duomenis, lentelės turinys gali keistis po kiekvieno sakinio įvykdymo. Pavyzdžiui, įvykdžius pastarąjį UPDATE sakinį du kartus, kategorija padidės ne vienetu, bet dviem.

Tarkime, anksčiau minėtas Baltakis išėjo iš darbo ir dėlto, projektų, kuriuose jis dalyvavo, vykdymas sulėtėja. Realiai, darbuotojo išėjimas iš darbo negali būti priežastimi pratęsti projektų vykdymo terminą. Visgi, pratęskime visų projektų, kuriuose dalyvavo Baltakis, vykdymo terminą dešimtadaliu:

UPDATE *Projektai* SET *Terminas* = *Terminas* * 1.1
WHERE *Projektai.Nr* IN (SELECT *Projektas* FROM *Vykdymas*, *Vykdytojai*
WHERE*Vykdytojas* = *Vykdytojai.Nr* AND *Pavardė* = 'Baltakis').

Paieškos sąlygoje yra pavartota užklausa, kurios rezultatas yra projektų, kuriuose dalyvavo Baltakis, numeriai. Šiuo sakiniu duomenys keičiami tik vienoje lentelėje *Projektai*, tačiau apibrėžiant atnaujinamas šios lentelės eilutes kreipiamasi į kitas dvi lenteles.

Padidinkime kategoriją visiems darbuotojams, kurie dalyvauja bent dviejuose projektuose:

UPDATE *Vykdytojai* SET *Kategorija* = *Kategorija* + 1
WHERE (SELECT COUNT(*) FROM *Vykdymas* WHERE *Vykdytojas* = *Nr*) >= 2.

Užklausa, esanti paieškos sąlygoje, yra priklausoma. Tai pavyzdys, kad užklausoje kartais yra sunku išvengti parametro. Net ir šiame uždavinyje išvengsime priklausomos užklausos, jei pasinaudosime duomenų grupavimu:

UPDATE *Vykdytojai* SET *Kategorija* = *Kategorija* + 1
WHERE *Nr* IN (SELECT *Vykdytojas* FROM *Vykdymas*
GROUP BY *Vykdytojas* HAVING COUNT(*) >= 2).

Šio sakinio vidinės užklausos rezultatas yra visų vykdytojų, kurie dalyvauja bent dviejuose projektuose, numeriai. Kadangi tokia užklausa neturi parametro, tai jos rezultatą galima sudaryti prieš duomenų keitimą.

5.8. Lentelių ir DB šalinimas

Paprastai, lentelė duomenų bazėje sukuriama vieną kartą ir išlieka joje ilgą laiką. Lentelės egzistavimo metu į ją įvedamos naujos, keičiamos bei šalinamos anksčiau įvestos eilutės. Ypatingais atvejais, pavyzdžiui keičiant DB struktūrą (reliacinę schemą), prireikia pašalinti lentelę. Lentelės pašalinimas reiškia, ne tik visų jos eilučių pašalinimą, bet ir lentelės apibrėžimo pašalinimą. Tai atliekama SQL sakiniu:

DROP TABLE <lentelės vardas>.

Įvykdžius tokį sakinį, lentelė nustoja egzistavusi. Lentelę *Vykdymas* galima sunaikinti taip:

DROP TABLE *Vykdymas*.

Praktikoje prireikia pašalinti ir visą DB. SQL standarte nėra sakinio, kuriuo tai būtų galima atlikti. Panašiai kaip ir DB kūrimo atveju, dauguma DBVS turi komandą `DROP DATABASE`, kuri leidžia tai padaryti. Pavyzdžiui, sakiniu

```
DROP DATABASE Darbai
```

DB *Darbai* pašalinsime iš kompiuterio atminties. Dirbti su šia DB negalėsime.