6. Virtualios lentelės ir duomenų nepriklausomumo lygiai

6.1. Įvadas

Virtualia lentele (**vaizdu**) (angl. *view*) vadinama užklausa, kuriai suteikiamas vardas ir ji įsimenama duomenų bazėje. Vartotojui leidžiama peržiūrėti įsimintos užklausos rezultatą, kreipiantis į jį kaip į paprastą lentelę.

Antrame skyriuje mes susipažinome su laikinomis lentelėmis. Virtuali lentelė, kaip ir laikina lentelė, yra užklausos rezultatas, kuriam suteiktas vardas. Tačiau, laikina lentelė yra tik bendresnės užklausos tarpinis rezultatas, kuris nustoja egzistuoti įvykdžius užklausą. Sukūrus virtualią lentelę, ji išlieka tol, kol nebus sunaikinta. Virtualios lentelės turi laikinos lentelės savybes, bet jų vaidmuo ir pritaikymo sritis yra žymiai platesni.

Pagrindinės virtualių lentelių vartojimo priežastys yra šios:

- jomis galima sudaryti vartotojams savitą duomenų bazės sandaros įvaizdį;
- jomis galima paslėpti lentelių eilutes ir stulpelius;
- jos supaprastina vartotojo darbą, nes supaprastėja duomenų struktūra.

Paskutinė priežastis kartu yra ir laikinų lentelių naudingumo priežastis.

6.2. Virtualių lentelių sukūrimas

Virtuali lentelė neturi savo duomenų, ji remiasi kitų lentelių duomenimis. Apie virtualios lentelės egzistavimą sistema žino tik iš sisteminio katalogo, kuriame saugomi duomenys apie ją. Tikrai egzistuoja tik virtualios lentelės apibrėžimas, bet ne jos duomenys. Virtuali lentelė sukuriama (apibrėžiama) SQL duomenų apibrėžimo (DDL) sakiniu `CREATE VIEW`:

```
CREATE VIEW <virtualios lentelės vardas> [(<stulpelio vardas> {,<stulpelio vardas> })]
            AS <užklausa> [WITH CHECK OPTION].
```

Jei virtualios lentelės stulpelių vardai, nėra išvardyti už jos pavadinimo, tai jie paveldimi iš užklausos. Užklausoje išvardintas lenteles vadinsime pradinėmis vitualiai lentelei. Virtuali lentelė nustoja egzistavusi tik tuomet, kai ji sunaikinama sakiniu

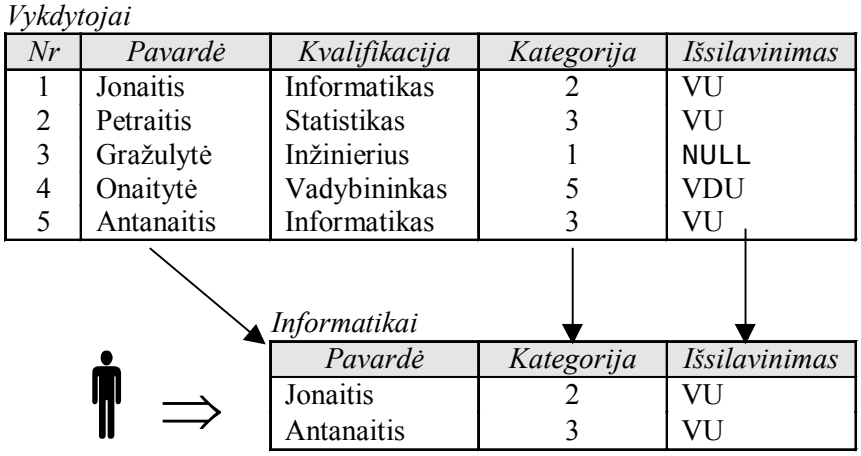
```
DROP VIEW <virtualios lentelės vardas>.
```

Apibrėžkime virtualią lentelę su duomenimis apie informatikus:

```
CREATE VIEW Informatikai (Pavardė, Kategorija, Išsilavinimas)
    AS SELECT Pavardė, Kategorija, Išsilavinimas
        FROM Vykdytojai WHERE Kvalifikacija = 'Informatikas'.
```

DB valdymo sistema, vykdydama šį sakinį, užklausos, esančios frazėje `AS`, nevykdo. Ši užklausa tik apibrėžia virtualios lentelės duomenis. Sakinio rezultatas – virtualios lentelės apibrėžimas duomenų bazės sisteminiame kataloge. Apibrėžus virtualią lentelę, vartotojas gali elgtis su ja kaip su paprasta lentele.

Virtuali lentelė yra dinamiška. Visi pasikeitimai pradinėje lentelėje *Vykdytojai* atsispindi virtualioje lentelėje (pradinės lentelės *Vykdytojai* vaizde) *Informatikai*, ir atvirkščiai, pakeitus duomenis virtualioje lentelėje, pakeitimai atsispindi ir pradinėje. Keičiant duomenis virtualioje lentelėje, iš tikrųjų, bus keičiami pradinės lentelės duomenys, nes virtualioji savų duomenų neturi. Virtuali lentelė yra lyg "langas" į pradinę lentelę. *Informatikai* yra "langas" į firmos darbuotojus (į lentelės *Vykdytojai* duomenis), per kurį matosi tik informatikai. Kitaip tariant, virtuali lentelė *Informatikai* – tai požiūris į darbuotojus, kai matomi tik informatikai ir nekreipiama dėmesio į jų numerius:



6.1 pav. Virtuali lentelė *Informatikai* – "langas" į lentelę *Vykdytojai*.

Virtualiai lentelei *Informatikai* priklauso (matoma per "langą") tik tos lentelės *Vykdytojai* eilutės ir stulpeliai, kurie atitinka paieškos sąlygą, nurodytą virtualios lentelės apibrėžime. Papildžius lentelę *Vykdytojai* naujais duomenimis, priklausomai nuo naujų darbuotojų kvalifikacijos, jie bus "matomi" per "langą" *Informatikai* (jei naujiems darbuotojams *Kvalifikacija* = 'Informatikas') arba "nematomi".

Užklausos virtualiai lentelei sudaromos kaip ir paprastai lentelei, pavyzdžiui, antros kategorijos informatikus sužinosime tokia užklausa:

SELECT * FROM *Informatikai* WHERE *Kategorija* = 2.

Kai DBVS aptinka SQL sakinyje kreipinį į virtualią lentelę, ji susiranda virtualios lentelės apibrėžimą sisteminiame kataloge. Pagal tai, DBVS perdaro užklausą virtualiai lentelei į užklausą pradinei lentelei ir vykdo ją. Taip DBVS sudaro virtualiai lentelei paprastos lentelės įvaizdį. Sistema, vietoje pateiktos užklausos, vykdys tokią užklausą:

SELECT *Pavardė*, *Kategorija*, *Išsilavinimas* FROM *Vykdytojai*
WHERE *Kvalifikacija* = 'Informatikas' AND *Kategorija* = 2.

Jei virtualios lentelės apibrėžimas toks nesudėtingas, kaip *Informatikai*, DBVS gali formuoti užklausos rezultatą iš karto, perrinkdama pradinės lentelės eilutes. Jei apibrėžianti užklausa yra sudėtinga, pvz. su duomenų grupavimu, sistemai tenka **materializuoti** virtualią lentelę. Pradžioje, DBVS įvykdo apibrėžiančiąją užklausą suformuodama ir įsimindama jos rezultatą. Tik po to vartotojo užklausa vykdoma pagal tarpinį rezultatą.

Apibrėžiančioje užklausoje neleidžiama naudoti duomenų rūšiavimo frazės ORDER BY. Virtualios lentelės duomenų sutvarkymą galima nurodyti užklausoje, pavyzdžiui,

SELECT * FROM *Informatikai* ORDER BY *Pavardė*.

6.3. Virtualių lentelių rūšys

Virtualios lentelės dažnai taikomos vartotojų galimybėms (teisėms) valdyti. Pavyzdžiui, darbuotojams, dirbantiems konkrečiame projekte turėtų "nerūpėti" tai, kaip vykdomi kiti projektai. Lentelę *Vykdymas*, galima "padalyti" į tris virtualias lenteles, kurių kiekviena atitiktų vieną iš trijų projektų:

CREATE VIEW *Vykdymas1* AS SELECT * FROM *Vykdymas* WHERE *Projektas* = 1,
CREATE VIEW *Vykdymas2* AS SELECT * FROM *Vykdymas* WHERE *Projektas* = 2,
CREATE VIEW *Vykdymas3* AS SELECT * FROM *Vykdymas* WHERE *Projektas* = 3.

Virtualios lentelės, kurias sudaro pradinės lentelės dalis eilučių, vadinamos **horizontaliomis**. Lentelės dažnai skaidomos horizontaliai organizacijose, turinčiose keletą

padalinių. Tuomet kiekvieno padalinio darbuotojai dirba tik su savo padalinio duomenimis. Visų padalinių veiklos duomenys prieinami tik visos organizacijos valdybai.

Lentelės duomenis galima padalyti horizontaliai kiekvienam asmeniui atskirai. Sukurkime virtualią lentelę su duomenimis apie Jonaičio dalyvavimą projektuose:

```
CREATE VIEW Vykdo_Jonaitis AS
SELECT * FROM Vykdymas
WHERE Vykdytojas = (SELECT Nr FROM Vykdytojai WHERE Pavardė = 'Jonaitis').
```

Horizontaliomis virtualiomis lentelėmis paslepiamos pradinės lentelės eilutės. Nemažiau svarbu paslėpti duomenis, esančius stulpeliuose. Pavyzdžiui, paprastam firmos darbuotojui galima uždrausti žinoti, kiek uždirba jo kolegos. Uždrauskime firmos darbuotojams matyti kolegų kategorijas bei išsilavinimą, sukurdami virtualią lentelę, į kurią šie duomenys nepatenka:

```
CREATE VIEW Bendri_Duomenys SELECT Nr, Pavardė, Kvalifikacija FROM Vykdytojai.
```

Virtualios lentelės, kuriose yra visos pradinės lentelės eilutės, bet ne visi stulpeliai, yra vadinamos **vertikaliomis**.

Praktikoje dažnai vartojamos **mišrios** virtualios lentelės, kurios yra lentelės horizontalus ir vertikalus padalijimai. Jau apibrėžta virtuali lentelė *Informatikai* yra mišrios virtualios lentelės pavyzdys.

Jei virtualios lentelės apibrėžime duomenys grupuojami, tai tokia virtuali lentelė vadinama **grupine**. Grupinės virtualios lentelės vartojamos dažnoms ataskaitoms įsiminti. Sukurkime virtualią lentelę, surenkančią statistiką apie kiekvieno projekto vykdymą:

```
CREATE VIEW  Apie_Vykdymą(Projektas, Visos_Valandos, Vidurkis) AS
SELECT Projektas, SUM(Valandos), AVG(Valandos) FROM Vykdymas
GROUP  BY Projektas.
```

Tokia virtuali lentelė supaprastina daugelio statistikos uždavinių sprendimą. Pavyzdžiui, kiek vidutiniškai kiekvienam projektui skiriama valandų, sužinosime tokia nesudėtinga užklausa:

```
SELECT AVG(Visos_Valandos) FROM Apie_Vykdymą.
```

Jungtinė virtuali lentelė, apibrėžiama užklausa, kurioje jungiamos kelios lentelės. Sukurkime virtualią lentelę, kuri apimtų visus DB duomenis apie projektą Nr. 1:

```
CREATE VIEW Projektas1 AS
SELECT Projektai.*, Vykdytojas, Pavardė, Kvalifikacija, Kategorija, Išsilavinimas,
        Statusas, Valandos FROM Projektai, Vykdytojai, Vykdymas
WHERE Projektas = Projektai.Nr AND Vykdytojas = Vykdytojai.Nr  AND Projektas = 1.
```

Apibrėžiančioje virtualią lentelę užklausoje galima kreiptis ne tik į paprastą lentelę, bet ir į virtualią. Taigi, virtuali lentelė gali būti pradine kitai virtualiai lentelei. Apibrėžkime virtualią lentelę, kuri apimtų duomenis apie projektą Nr. 2. Tam galime pasinaudoti jau apibrėžta virtualia lentele *Vykdymas2*:

```
CREATE VIEW Projektas2
AS SELECT Projektai.*, Vykdytojas, Pavardė, Kvalifikacija, Kategorija, Išsilavinimas,
          Statusas, Valandos FROM Projektai, Vykdytojai, Vykdymas2
WHERE Projektas = Projektai.Nr AND Vykdytojas = Vykdytojai.Nr.
```

6.4. Virtualių lentelių atnaujinimas

Virtualios lentelės duomenų atnaujinimas pakeičiamas paprastos lentelės atnaujinimu. Padidinkime Jonaičiui kategoriją vienetu virtualioje lentelėje *Informatikai*:

UPDATE *Informatikai* SET *Kategorija* = *Kategorija* + 1 WHERE *Pavardė* = ‘Jonaitis’.

DBVS elgiasi panašiai, kaip vykdydama užklausą virtualiai lentelei: virtualios lentelės *Informatikai* duomenų atnaujinimas pakeičiamas pradinės lentelės *Vykdytojai* duomenų atnaujinimu. Taigi, iš tikrųjų vykdomas toks sakinys:

UPDATE *Vykdytojai* SET *Kategorija* = *Kategorija* + 1
WHERE *Pavardė* = ‘Jonaitis’ AND *Kvalifikacija* = ‘Informatikas’.

Informatikai yra mišri virtuali lentelė. Ji turi ir horizontalios, ir vertikalios virtualios lentelės savybes. Tiek horizontaliose, tiek ir vertikaliose virtualiose lentelėse reikšmes galima keisti, jei tai leidžiama pradinėje lentelėje.

Tarp grupinės virtualios lentelės ir jos pradinės lentelės eilučių nėra atitinkamybės “vienas su vienu”. Todėl grupinės virtualios lentelės duomenų negalima nei keisti, nei šalinti. Į grupinę virtualią lentelę negalima įterpti naujų eilučių. Tarp jungtinės virtualios lentelės ir pradinių lentelių eilučių kartais galima rasti vienareikšmišką atitinkamybę. Tačiau daugelyje komercinių DBVS, jungtinės virtualios lentelės duomenis keisti negalima, kadangi SQL sakiniu galima keisti tik vienos lentelės duomenis.

SQL1 standarte suformuluotos sąlygos virtualiai lentelei, kurios duomenis galima atnaujinti. Pagal standartą, virtualią lentelę galima atnaujinti, jei ją apibrėžianti užklausa tenkina šiuos reikalavimus:

- nėra frazės DISTINCT;
- frazėje FROM yra tik viena pradinė lentelė ir ją galima atnaujinti. Jei pradinė lentelė yra virtuali, tai ji turi tenkinti visus šiuos reikalavimus;
- kiekvienas stulpelis yra paprastas lentelės stulpelis, o ne reiškinys;
- sąlygoje nėra kitos užklausos;
- nėra duomenų grupavimo.

Išvardyti reikalavimai remiasi paprastu principu: virtualią lentelę galima atnaujinti tik tuomet, kai kiekvienai atnaujinamai virtualios lentelės eilutei DBVS gali vienareikšmiškai nustatyti vieną eilutę pradinėje lentelėje, ir kiekvienam stulpeliui - pradinės lentelės stulpelį. Jei virtuali lentelė tenkina išvardytus reikalavimus, tai ir joje, ir pradinėje lentelėje galima keisti reikšmes, įterpti naujas bei šalinti esamas eilutes.

6.5. Virtualių lentelių atnaujinimo valdymas

Jei virtualią lentelę apibrėžiančioje užklausoje yra paieškos sąlyga, tai virtualioje lentelėje bus “matomos” tik sąlygą tenkinančios eilutės. Kitos pradinės lentelės eilutės bus “nematomos”. Pavyzdžiui, taip apibrėžtoje virtualioje lentelėje:

CREATE VIEW *Gudručiai*
AS SELECT * FROM *Vykdytojai* WHERE *Kategorija* > 2,

“matysime” tik tų vykdytojų duomenis, kurių kategorija aukštesnė nei 2. Taip apibrėžta virtuali lentelė tenkina visus duomenų atnaujinimo reikalavimus. Į šią virtualią lentelę galima įterpti naujas eilutes, pavyzdžiui,

INSERT INTO *Gudručiai* VALUES (7, ‘Uždavinys’, ‘informatikas’, 3, ‘VU’).

DBVS įterps naują eilutę į pradinę lentelę *Vykdytojai*. Panagrinėkime, kas atsitiks įvykdžius tokį sakinį:

INSERT INTO *Gudručiai* VALUES (8, ‘Juozaitis’, ‘informatikas’, 2, ‘VDU’).

Šis duomenų įvedimo sakinys yra sintaksiškai teisingas, be to virtualią lentelę galima atnaujinti, todėl DBVS įterps naują eilutę į lentelę *Vykdytojai*. Tačiau įvykdžius šį sakinį ir pažiūrėjus į virtualią lentelę *Gudručiai*, įvestos eilutės nepamatysime. Sakinys

```
SELECT * FROM Gudručiai
```

pateiks tokį pat rezultatą, kaip ir prieš duomenų įvedimą. Taip atsitinka todėl, kad stulpelio *Kategorija* reikšmė 2 netenkina virtualią lentelę apibrėžiančios sąlygos: *Kategorija* > 2.

Įvykdžius sakinį:

```
UPDATE Gudručiai SET Kategorija = 2 WHERE Pavardė = 'Jonaitis'
```

bus dar keisčiau. Virtualios lentelės eilutėje pakeitus vieną reikšmę, ši eilutė išnyksta iš jos, lyg būtų įvykdytas `DELETE` sakinys, o ne `UPDATE`. Tiksliau, "matoma" eilutė pavirto "nematoma". Kad taip neatsitiktų, galima uždrausti įvesti "nematomas" eilutes į virtualią lentelę. Taip pat galima uždrausti atnaujinti duomenis, kai šie iš "matomų" tampa "nematomais". Tam reikia virtualios lentelės apibrėžimą papildyti fraze `WITH CHECK OPTION`. Jei virtualią lentelę *Gudručiai* sukursime taip:

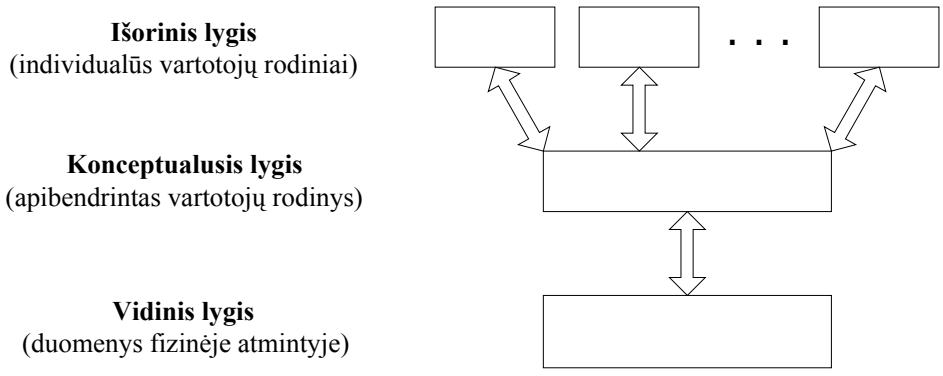
```
CREATE VIEW Gudručiai
AS SELECT * FROM Vykdytojai WHERE Kategorija > 2 WITH CHECK OPTION,
```

tai DBVS neleis įvesti į ją naujų eilučių, netenkinančių nurodytos paieškos sąlygos. Kiekvieną kartą atnaujinant duomenis taip apibrėžtoje virtualioje lentelėje, DBVS tikrina, ar neatsiras "nematomų" eilučių. Duomenis atnaujinti neleidžiama, jei tokios eilutės gali atsirasti.

Atnaujinimo tikrinimu negalima piknaudžiauti. Šis režimas neigiamai atsiliepia SQL sakinių `UPDATE` ir `INSERT` vykdymo efektyvumui. Kiekvieną kartą atliekant šiuos sakinius, DB valdymo sistemai reikia papildomai tikrinti "matomumo" sąlygą.

6.6. Duomenų nepriklausomumo lygiai

Viena iš pagrindinių virtualių lentelių paskirčių yra loginio duomenų nepriklausomumo užtikrinimas. Sistema užtikrina **loginį duomenų nepriklausomumą**, jei vartotojas ir jo vykdomos programos nepriklauso nuo loginės duomenų bazės struktūros (sandaros) pasikeitimų. Atitinkamai **fizinis duomenų nepriklausomumas** reiškia vartotojų ir jų programų nepriklausomumą nuo fizinės DB ypatybių. Abiems duomenų nepriklausomumams užtikrinti ANSI (American National Standards Institute) XX a. aštuntojo dešimtmečio viduryje pasiūlė trijų sluoksnių duomenų bazių valdymo sistemų architektūrą. Ši architektūra yra daugelio komercinių DBVS pagrindas. Architektūroje yra išskiriami vidinis, išorinis ir konceptualus lygiai (sluoksniai):



6.2 pav. Trys DBVS architektūros lygiai

Bendrais bruožais šiuos tris lygius galima apibūdinti taip:

- **Vidinis** lygis yra artimiausias fiziniam duomenų saugojimui. Jis susijęs su duomenų saugojimo fiziniuose nešėjuose būdais.

- **Išorinis** yra artimiausias vartotojui lygis, kuris susijęs su duomenų vaizdavimu atskiriems vartotojams. Tai vartotojų individualus DB loginės struktūros įvaizdis.
- **Konceptualusis** – tai tarpinis lygis, kuriame atsispindi visų duomenų, saugomų fizinėje atmintyje, loginė struktūra. Tai apibendrintas dalykinės srities modelis.

Loginis duomenų nepriklausomumas – tai nepriklausomumas tarp išorinio ir konceptualaus lygių, o fizinis nepriklausomumas – tarp konceptualaus ir vidinio lygių. Fizinis nepriklausomumas numato saugomų duomenų pernešimą iš vienų nešėjų į kitus, nekeičiant taikomųjų programų, dirbančių su DB. Loginis duomenų nepriklausomumas leidžia išlikti vartotojų programoms veikiančiomis, netgi pakeitus konceptualų duomenų modelį.

Fizinis nepriklausomumas pasiekiamas, daugiausiai, operacijų sistemos, kurioje funkcionuoja DBVS, priemonėmis. Plačiau aptarsime loginį nepriklausomumą.

6.7. Loginio duomenų nepriklausomumo užtikrinimas

Su loginiu duomenų nepriklausomumu yra susiję du pagrindiniai aspektai – DB **struktūros augimas** ir **keitimas**.

DB **struktūros augimas** galimas dviem būdais:

- papildant lentelę nauju stulpeliu;
- sukuriant naują lentelę.

Naujos lentelės sukūrimas neturi įtakos nei patiems vartotojams, nei jų programoms. Papildomos lentelės atsiradimas duomenų bazėje negali pakeisti naudojamų užklausų ir duomenų atnaujinimo.

Naujas stulpelis lentelėje gali daryti įtaką tik užklausoms, kurių `SELECT` frazėje yra nuoroda į visus lentelės stulpelius. Iš anksto neperspėjus vartotoją apie lentelės papildymą nauju stulpeliu, jis, eilinį kartą įvykdęs užklausą `SELECT * FROM`…, gali nemaloniai nustebinti, pamatęs nelauktą stulpelį. Nepakankamai kvalifikuotai sudaryta programa, kurioje tikimasi konkrečių užklausos rezultato stulpelių, gali nustoti veikti gavusi daugiau stulpelių.

Ankstesniame skyrelyje pateikėme duomenų įterpimo sakinį, kuriame duomenys apibrėžiami užklausa:

`INSERT INTO` *Seni_Projektai* `SELECT * FROM` *Projektai.*

Papildžius lentelę *Projektai* nauju stulpeliu ir nepadarius to paties su lentele *Seni_Projektai*, šio sakinio negalėsime vykdyti. Todėl, sudarant užklausas, ypač, jei jos vykdomos daugelį kartų, reikia vengti neišreikštinio stulpelių nurodymo.

Jei nelaukto stulpelio problema iškyla, tai naująjį stulpelį galima "paslėpti" virtualia lentele. Tarkime, lentelę $L(R)$ reikia papildyti naujais stulpeliais A. Pradžioje, sukuriame naują lentelę $L_1(R, A)$, į kurią perkeliame visus lentelės L duomenis, po to lentelę L sunaikiname ir sukuriame virtualią lentelę, atitinkančią pradinę lentelę L. Išreiškime tai SQL sakiniais:

`CREATE TABLE` $L_1(R, A)$;
`INSERT INTO` $L_1(R)$ `SELECT` R `FROM` L;
`DROP TABLE` L;
`CREATE VIEW` $L(R)$ `AS SELECT` R `FROM` L_1.

Su sukurta virtualia lentele L bus galima dirbti taip pat, kaip su ankstesne lentele L. Sukurdami virtualią lentelę mes išvengėme nepatogumų, kilusių dėl būtinumo papildyti lentelę nauju stulpeliu. Sprendžiant naujus uždavinius, kuriuose naujieji stulpeliai yra prasmingi, dirbsime su lentele L_1.

Kartais tenka keisti DB struktūrą taip, kad bendras duomenų kiekis išlieka toks pat, bet jų išdėstymas pasikeičia. Tuomet lentelių stulpeliai (atributai) yra pertvarkomi. Taip atsitinka kai pasikeičia atributų tarpusavio priklausomybės ir DB struktūra pradeda neatitikti konkrečios normalinės formos reikalavimų. Tarkime, mums reikia (priežastis nesvarbi) lentelę $L(A, B, C)$ pakeisti dviem lentelėmis $L_1(A, B)$ ir $L_2(A, C)$. Tuomet pradinės lentėlės L įvaizdį atstatysime sukurdami virtualią lentelę tuo pačiu vardu kaip pradinė lentelė:

```
CREATE VIEW L(A, B, C) AS
SELECT L1.A, L1.B, L2.C FROM L1 , L2 WHERE L1.A = L2.A .
```

Nuo šiol bet kuri programa ir interaktyvi operacija, kurioje vartojamas senasis lentelės vardas *L*, dabar kreipsis į virtualią lentelę tuo pačiu vardu *L*. Tačiau virtualioje lentelėje, kuri jungia dvi lenteles, negalima atnaujinti duomenų. Todėl sukūrę virtualią lentelę, mes pasiekėme tik užklausų loginį nepriklausomumą. Visiško loginio nepriklausomumo pasiekti nepavyksta.

Kita vertus, keičiant DB struktūrą, loginis duomenų nepriklausomumas turi prasmę tik tuomet, kai DB prieš pakeitimą ir DB po pakeitimo yra tapačios saugomais duomenimis. Pavyzdžiui, negalime kalbėti apie loginio duomenų nepriklausomumo užtikrinimą, kai pašalinami lentelės stulpeliai ar visos lentelės, t.y. kai duomenų bazės duomenys prarandami. Todėl, DB struktūros keitimo sąvoka neapima stulpelių ir lentelių šalinimo.

Loginio nepriklausomumo nuo struktūros keitimo užtikrinimas yra tik vienas virtualių lentelių panaudojimas. Susisteminsime virtualių lentelių privalumus.

6.8. Virtualių lentelių trūkumai ir privalumai

Pagrindiniai virtualių lentelių **privalumai** yra šie:

- **Saugumas**. Kiekvienam vartotojui galima sudaryti ir pateikti nedaug virtualių lentelių, kurios apimtų tik jam skirtus duomenis. Taip galima apriboti duomenų naudojimą. Sukuriant virtualias lenteles galima "paslėpti" realių lentelių stulpelius bei eilutes. Tuomet vartotojas "mato" tik tuos duomenis, kurie yra "matomi" per jam skirtą "langą". Tai nevisiškai išsprendžia duomenų apsaugos problemą, nes vartotojui neuždraudžiama peržiūrėti pradines lenteles. Virtualios lentelės tik papildo specialias duomenų apsaugos priemones, kurias aptarsime vėliau.
- **Užklausų paprastumas.** Dažnai vartojamoms sudėtingoms užklausoms galima apibrėžti virtualias lenteles. DBVS tokias užklausas "įsimina" sisteminiame kataloge. Tuomet vietoje sudėtingos užklausos pradinėms lentelėms, galima sudaryti paprastesnes užklausas virtualiai lentelei. Virtualios lentelės, kurios kuriamos naudoti užklausose, dažnai vadinamos **makrosais**. Šiame taikyme, virtualios lentelės apibrėžimas yra vardo suteikimas užklausai. Vėlesnėse užklausose tas vardas naudojamas užklausai sutrumpinti.
- **Struktūros paprastumas.** Virtualios lentelės sukuria kiekvienam vartotojui "savitą" DB struktūrą. Jungiant kelias lenteles į vieną virtualią, esama DB struktūra labai supaprastėja.
- **Loginis nepriklausomumas**. Duomenų bazės struktūra (reliacinė schema) gali keistis, pavyzdžiui, atsirasti naujų stulpelių. Atributų savybės bei jų tarpusavio ryšiai taip pat gali keistis. Tuomet lenteles prireikia skaidyti. Keičiantis DB struktūrai, galima sukurti virtualias lenteles, kurios "paslepia" daugelį pasikeitimų. Taip virtualios lentelės dalinai užtikrina loginį nepriklausomumą.

Virtualios lentelės keletą **trūkumų**. Štai jie:

- **Našumas**. Virtualios lentelės sudaro lentelės egzistavimo iliuziją. DB valdymo sistema užklausas virtualiai lentelei pakeičia užklausomis pradinėms (paprastoms) lentelėms. Jei virtuali lentelė jungia daug lentelių, tai paprasta užklausa virtualiai lentelei gali tapti sudėtingu lentelių jungimu, kuriam atlikti reikės daug kompiuterio resursų.
- **Ribotas atnaujinimas**. Duomenų atnaujinimas yra galimas tik gana paprastose virtualiose lentelėse. Virtualiose lentelėse, kurios jungia kelias lenteles, yra galima tik duomenų paieška.

Išvardyti trūkumai reiškia, kad negalima piktnaudžiauti virtualiomis lentelėmis. Kiekvienu konkrečiu atveju, kuriant ar vartojant virtualią lentelę, reikia atsižvelgti tiek į privalumas, tiek ir į trūkumus.