9.3. Bendro duomenų naudojimo problemos

DB yra bendrai naudojama sistema
– vienu metu vykdomos kelios transakcijos.

DBVS turi užtikrinti duomenų atkūrimą kiekvienam vartotojui (programai) atskirai.

Vartotojas turi jaustis lyg tik jis vienas dirbtų su DB.

Pakeistų duomenų praradimo problema

Transakcija A	Laikas	Transakcija B
–	|	–
FETCH R	t_1	–
–	|	–
–	t_2	**FETCH** R
–	|	–
UPDATE R	t_3	–
–	|	–
–	t_4	**UPDATE** R
–	↓	–

Momentu t_4 A **praranda** atnaujintus duomenis, kadangi jos pakeistus duomenis „pakeičia“ transakcija B.

Pakeistų duomenų praradimo problema susidaro kaskart, kai 2 transakcijos nuskaito tuos pačius duomenis, o po to jos abi keičia tuos duomenis. **Pvz.**:

- dvi transakcijos nuskaito tuos pačius duomenis,

  $x \leftarrow A$

- jos abi atnaujina anksčiau nuskaitytus duomenis (remdamosi anksčiau perskaityta reikšme),

  $x := x + 1,\ x \rightarrow A$

  čia x – programos kintamasis, A – stulpelis;

Jei pradžioje $A = 1$, **tai rezultate** A gali būti: 2 arba 3.

Priklausymo nuo neužfiksuoto pakeitimo problema

Transakcija A	Laikas	Transakcija B
–	|	–
–	t_1	**UPDATE** R
–	|	–
FETCH R	t_2	–
–	|	–
–	t_3	**ROLLBACK**
–	↓	–

Nuo t_2 A tolimesnio **darbo teisingumas priklauso** nuo to, kaip pasibaigs B, patvirtindama ar anuliuodama pakeitimą. Jei B atsisako pakeitimo (t_3), tai A naudoja neteisingus duomenis.

Dar blogiau, kai momentu t_2 transakcija A keičia duomenis:

Transakcija A	Laikas	Transakcija B
–	|	–
–	t_1	**UPDATE** R
–	|	–
UPDATE R	t_2	–
–	|	–
–	t_3	**ROLLBACK**
–	↓	–

Nuo momento t_2 A teisingumas priklauso nuo neužfiksuoto pakeitimo, o momentu t_3 A praranda ir pakeitimą.

Tai dar vienas **duomenų praradimo atvejis**.

Nesuderintų duomenų analizės problema.

Tarkime veikia dvi transakcijos:

A sumuoja valandas, kurios skiriamos projektui Nr. 3,

B vykdytojo Nr. 3 50 valandų paskiria vykdytojui Nr. 1.

Abi transakcijos naudoja tą pačią lentelę *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

Prieš: 250+400+150 = 800 val.
Po: 300+400+100 = 800 val.

Pažymėkime: R_i - *Vykdymas* eilutė su duomenimis apie vykdytojo Nr. i „indėlį“.

Transakcija A	Laikas	Transakcija B
–	|	–
FETCH R_1 *suma* = 250	t_1	–
–	|	–
FETCH R_2 *suma* = 650	t_2	–
–	|	–
–	t_3	**FETCH** R_3
–	|	–
–	t_4	**UPDATE** R_3 150 →100
–	|	–
–	t_5	**FETCH** R_1
–	|	–

–	t_6	**UPDATE** R_1 250 →300
–	|	–
–	t_7	**COMMIT**
–	|	–
FETCH R_3 *suma* = 750	t_8	–

A pabaigoje (laiko momentu t_8) gautas rezultatas (750) yra neteisingas.

A atliko nesuderintų duomenų analizę.

9.4. Duomenų blokavimas

Bendro duomenų naudojimo problemos sprendžiamos duomenų **blokavimu** (užrakinimu, angl. *locking*).

Blokavimo rezultate DB objektas yra izoliuojamas nuo kitų transakcijų.

Transakcija yra priversta laukti, jei nori pasiekti blokuotą objektą.

Komercinėse DBVS naudojami du blokavimo tipai (lygiai):

- **X** (angl. *eXclusive lock*) – visiška blokuotė;
- **S** (angl. *Shared lock*) – dalinė blokuotė.

X ir **S** blokuotės vadinamos **rašymo ir skaitymo blokuotėmis.**

Blokavimo taisyklės:

- **Jei** transakcija ***A*** **visiškai užblokuoja (X)** eilutę *R*, **tai** transakcija *B*, norinti perskaityti ar pakeisti tą eilutę, yra priversta laukti kol *A* atblokuos *R*.
- **Jei** transakcija ***A*** **dalinai užblokuoja (S)** eilutę *R*, **tai**:
  - **jei *B* nori visiškai (X)** užblokuoti eilutę *R*, **tai** *B* priverčiama lauki, kol *A* neatblokuos tą eilutę;
  - **jei *B* nori dalinai (S)** užblokuoti eilutę *R*, **tai** ir transakcijai *B* leidžiama užblokuoti *R* **S** blokuote.

Blokavimo lygių **suderinamumo matrica**

	X	**S**	–
X	Ne	Ne	Taip
S	Ne	Taip	Taip
–	Taip	Taip	Taip

Prašymai blokuoti lentelės eilutę yra netiesioginiai:

- kai transakcija nori skaityti eilutę (**SELECT**, **FETCH**), ji automatiškai „prašo“ leisti blokuoti tą eilutę lygiu **S**
- kai norima atnaujinti eilutę (**UPDATE**) **–** „prašoma“ blokuoti eilutę **X** lygiu.

Eilutė yra blokuojama automatiškai, kai tik transakcija „sulaukia“ DBVS leidimo.

Transakcija, blokavusi kurią nors eilutę paprastai **išlaiko blokavimą** iki transakcijos pabaigos.

Blokavimo išlaikymas priklauso ir nuo išorinių veiksnių.

9.5. Bendro duomenų naudojimo problemų sprendimas

Prarasto pakeitimo situacija

Transakcija *A*	Laikas	Transakcija *B*
–	|	–
FETCH *R* (prašo **S** ir blokuoja)	t_1	–
–	|	–
–	t_2	**FETCH** *R* (prašo **S** ir blokuoja)
–	|	–
UPDATE *R* (prašo užblokuoti **X**)	t_3	–
Laukimas	|	–

Laukimas	t_4	**UPDATE** *R* (prašo užblokuoti **X**)
Laukimas	↓	Laukimas

Prarasto pakeitimo situacija nesusidaro.

Tačiau, kilo kita problema: dvi transakcijos laukia viena kitos, kad pratęsti darbą.

Priklausymas nuo neužfiksuoto pakeitimo (1)

Transkacija *A*	Laikas	Transkacija *B*
–	↓	–
–	t_1	**UPDATE** *R* (prašoma blokuoti **X**)
–	↓	–
FETCH *R* (prašo blokuoti **S**)	t_2	–
Laukimas	↓	–
Laukimas	t_3	**ROLLBACK** (atšaukiamas **X**)
Perskaitoma *R* (blokuojama **S** lygiu)	t_4	–
–	↓	–

Priklausymas nuo neužfiksuoto pakeitimo (2)

Transkacija *A*	Laikas	Transkacija *B*
–	↓	–
–	t_1	**UPDATE** *R* (prašoma blokuoti **X**)
–	↓	–
UPDATE *R* (prašo blokuoti **X**)	t_2	–
Laukimas	↓	–
Laukimas	t_3	**ROLLBACK** (atšaukiamas **X**)
Atnaujinama *R* (blokuojama **X** lygiu)	t_4	–
–	↓	–

Nesuderintų duomenų analizės problema.

Transkacija *A*	*Laikas*	Transkacija *B*
–	↓	–
FETCH R_1 (prašoma ir blokuojama **S**) *suma* = 250	t_1	–
–	↓	–
FETCH R_2 (prašoma ir blokuojama **S**) *suma* = 650	t_2	–
–	↓	–
–	t_3	**FETCH** R_3 (prašoma ir blokuojama **S**)
–	↓	–

–	t_4	**UPDATE** R_3 (prašoma ir blokuojama **X**) 150 →100
–	↓	–
–	t_5	**FETCH** R_1 (prašoma ir blokuojama **S**)
–	↓	–
–	t_6	**UPDATE** R_1 (prašoma blokuoti **X**) Laukiama
–	↓	Laukiama
FETCH R_3 (prašoma blokuoti **S**) Laukiama	t_7	Laukiama

Blokavimo mechanizmas užkerta kelią klaidingoms situacijoms kai vienu metu su tais pačiais duomenimis dirba kelios transakcijos.

Tačiau tai gali susidaryti būsena, kai dvi transakcijos laukia viena kitos, kol kita nutrauks blokavimą.

9.6. Aklavietės ir jų likvidavimas

Aklavietė - tai būsena, kai dvi ar daugiau transakcijų vienu metu **laukia viena kitos** kol atšauks blokavimą tam, kad pratęsti darbą.

Apibendrintą aklavietės susidarymo situaciją tarp dviejų transakcijų *A* ir *B* galima aprašyti taip:

Transkacija *A*	Laikas	Transkacija *B*
–	↓	–
R_1 blokuojama **S** arba **X**	t_1	–
–	↓	–
–	t_2	R_2 blokuojama **S** arba **X**
–	↓	–
Prašoma blokuoti R_2 **X**	t_3	–
Laukimas	↓	–
Laukimas	t_4	Prašoma blokuoti R_1 **X**
Laukimas	↓	Laukimas

Praktiškai nesutinkamos aklavietės, apimančios > 2 transakcijas.

DBVS veda laukiančių **transakcijų žurnalą**.

Aklavietė - ciklas **transakcijų būsenų diagramoje**.

Aptikusi aklavietę, DBVS vieną iš transakcijų parenka **auka** ir:

- išrinktą transakciją nutraukia, atstatydama jos pradinius duomenis;
- nutraukia SQL sakinį grąžindama klaidos kodą.

9.7. Duomenų blokavimo tvarkymas

Blokavimo tvarkymo priemonės:

- **lentelės blokavimas**;
- **transakcijos blokavimo** (izoliavimo) **lygio** (angl. *isolation level*) priskyrimas transakcijai.

Visos lentelės užblokavimo **privalumai**:

- nėra sąnaudų, susijusių su kiekvienos eilutės blokavimu;
- pradėjus duomenų apdorojimą, veiksmų atlikimas neužtruks dėl eilučių, užblokuotų kitos transakcijos;
- apdorojus dalį eilučių nesusidarys aklavietė.

Visos lentelės užblokavimo **trūkumas**: visos kitos transakcijos bus priverstos laukti, kol lentelės užblokavimas bus atšauktas.

Lentelės blokavimas turėtų būti vartojamas tik programose.

```
LOCK TABLE <lentelės vardas>
IN [SHARE | EXCLUSIVE] MODE
```

`LOCK TABLE` *Vykdymas* `IN SHARE MODE`

`LOCK TABLE` *Vykdytojai* `IN EXCLUSIVE MODE`

Lentelė išlieka užblokuota iki transakcijos pabaigos.

9.8. Transakcijų apsauga

DBVS gali blokavimą valdyti efektyviau, jei ji žino, kaip transakcijoje dirbama su duomenimis.

Transakcijos apsaugos (izoliavimo) lygis apibūdina lygiagrečiai vykdomų transakcijų kišimosi viena į kitos darbą laipsnį. **Tai**:

- **laipsnis**, kuriuo lentelių eilutės, perskaitytos ar pakeistos transakcijoje, yra pasiekiamos kitoms transakcijoms;
- **laipsnis**, kuriuo kitų transakcijų perskaitytos ar pakeistos eilutės yra pasiekiamos einamojoje transakcijoje.

Transakcijų apsaugos lygiai – tai kompromisas tarp visiško izoliavimo ir efektyvaus darbo.

SQL2 yra apibrėžti 4 transakcijos apsaugos lygiai.

- **Visiško blokavimo** (`SERIALIZABLE`) lygis:
  - bet kuri transakcijoje perskaityta eilutė nebus pakeista jokios kitos transakcijos kol duotoji transakcija nesibaigs;
  - jokia eilutė, pakeista kitoje transakcijoje nebus perskaityta duotojoje, kol pakeitusi eilutę transakcija nesibaigs.

Užtikrinama, kad vykdant pakartotinį duomenų išrinkimą, kiekvieną kartą bus perskaityti tie patys duomenys.

- **Pakartotinio duomenų skaitymo** (`REPEATABLE READ`).
  - Kitos transakcijos gali įterpti naujas eilutes, kurios patenka tarp duotosios transakcijos perskaitytų duomenų.
  - Pakartotinai išrenkant duomenis gali būti perskaitytos papildomos eilutės – „vaiduoklės“ (angl. *phantom*).

- **Užfiksuotų duomenų skaitymo** (`READ COMMITTED`).
  - Pakeista ir neužfiksuota kitoje transakcijoje eilutė, nebus prieinama.
  - Užfiksuoti kitų transakcijų rezultatai yra matomi. Pakartotinai skaitant galima aptikti pasikeitimus.
  - Perskaitytai eilutei `S` išlaikomas tik kol ši išlieka einamąja.

Šis lygis **vartotinas, kai**:

- nėra reikalinga tas pačias eilutes peržiūrėti keletą kartų
- nėra kaupiama suminė užklausos rezultato informacija.

- **Neužfiksuotų duomenų skaitymo** (**READ UNCOMMITTED**)

Perskaitytą eilutę, leidžiama pakeisti kitoje transakcijoje.

Pakeista kitoje transakcijoje eilutė, gali būti perskaityta net jei tas pakeitimas nebuvo užfiksuotas.

Užtikrinama, kad nebus prarastų pakeitimų.

Kitų transakcijų neužfiksuotus duomenis galima tik perskaityti bet ne keisti.

Duomenų atnaujinimo požiūriu šis apsaugos lygis veikia kaip **READ COMMITTED**.

Transakcijos apsaugos lygių suvestinė

Apsaugos lygis	Neužfiksuotų duomenų perskaitymas	Pasikeitimai iš naujo perskaičius	Eilutės-„vaiduoklės“
SERIALIZABLE	Negalimas	Negalimi	Negalimos
REPEATABLE READ	Negalimas	Negalimi	Galimos
READ COMMITTED	Negalimas	Galimi	Galimos
READ UNCOMMITTED	Galimas	Galimi	Galimos

SQL2 standarto sakinys apsaugos lygiui nustatyti:
SET TRANSACTION ISOLATION LEVEL <apsaugos lygis> [<apdorojimo rūšis>]
Transakcijoje atliekamų operacijų rūšis:
- duomenų išrinkimo operacijos (**READ ONLY**)
- duomenų keitimo (**READ WRITE**)

Operacijų tipas padeda optimizuoti DBVS veiksmus.
Pagal nutylėjimą - **SERIALIZABLE** lygis.
Jei nurodyta **READ UNCOMMITTED**, tai, pagal nutylėjimą, priimama **READ ONLY** ir keisti duomenų neleidžiama.
Kitais atvejais, pagal nutylėjimą, priimama **READ WRITE**.
Sakinys turi būti pirmasis transakcijos sakinys.

Euristikos apsaugos lygiui parinkti

Duomenų apdorojimo rūšis	Reikalingas didelis duomenų stabilumas	Nebūtinas didelis duomenų stabilumas
Skaitymo ir rašymo transakcijos	**REPEATABLE READ**	**READ COMMITTED**
Tik skaitymo transakcijos	**SERIALIZABLE**	**READ UNCOMMITTED**