Obnavljanje podatkovne baze z dnevnikom in varnostno kopijo

UNDO · REDO · dnevnik · kontrolna točka · popolna, diferencialna in inkrementalna kopija

1. Zakaj potrebujemo obnavljanje?

Transakcija mora ohraniti **atomarnost** (vse spremembe ali nobena) in **trajnost** (potrjene spremembe preživijo odpoved). Podatki so lahko nekaj časa v medpomnilniku; podatkovne strani se na disk ne zapišejo nujno ob vsaki posamezni spremembi.

Dogodek	Kaj je ogroženo?	Način obnove
Odpoved procesa ali strežnika	Nepotrjene spremembe in potrjene spremembe, ki še niso na podatkovnih straneh.	**Dnevnik**: UNDO/REDO, po pravilih sistema.
Poškodba ali izguba diska	Podatkovne strani ali celotna baza niso več dostopne.	**Varnostna kopija + dnevnik** za kasnejše potrjene spremembe.



2. Trije različni mehanizmi

Mehanizem	Vloga
Varnostna kopija	Shrani stanje baze v določenem trenutku; nujna ob izgubi podatkovnih datotek.
Dnevnik (log)	Beleži spremembe in dogodke transakcij; osnova za UNDO/REDO.
Kontrolna točka (checkpoint)	Zmanjša količino dnevnika, ki jo je treba ob obnovi analizirati.

Glavna razlika: kopija vrne bazo na starejše shranjeno stanje, dnevnik pa omogoča naknadno obravnavo sprememb. Kopija ne nadomesti dnevnika; dnevnik brez nepoškodovane baze oziroma ustrezne kopije ne odpravi izgube celotnega diska.

Dnevnik, UNDO in REDO

3. Kako je videti zapis v dnevniku?

Poenostavljen fizični dnevnik vsebuje: transakcijo, vrsto operacije, podatkovno enoto, staro vrednost (before image) in novo vrednost (after image). V njem so tudi zapisi **START**, **COMMIT**, **ROLLBACK** in **CHECKPOINT**.

#	Dogodek	Podatek	Pred	Po
1	START T1	—	—	—
2	UPDATE T1	A	100	130
3	COMMIT T1	—	—	—
4	START T2	—	—	—
5	UPDATE T2	B	50	70
6	NESREČA	—	—	—

4. Razlaga istega dnevnika: primer korak za korakom

Pred začetkom sta vrednosti **A = 100** in **B = 50**. Predpostavimo, da sta neposredno pred odpovedjo na podatkovnem disku **A = 100** in **B = 70**: sprememba T1 še ni zapisana na podatkovno stran, sprememba T2 pa že je.

Transakcija	Status ob nesreči	Ukrep	Zakaj?
T1: A 100 → 130	Ima COMMIT	**REDO** A := 130	Sprememba je potrjena, a še ni na disku.
T2: B 50 → 70	Brez COMMIT	**UNDO** B := 50	Sprememba je lahko že na disku, a ni potrjena.

Stanje po obnovi: A = 130, B = 50. Potrjeno spremembo ohranimo, nepotrjeno razveljavimo.

5. Kaj pomenita UNDO in REDO?

• **UNDO:** razveljavitev sprememb **nepotrjene** transakcije; v poenostavljenem primeru uporabimo stare vrednosti, običajno v obratnem vrstnem redu zapisov.

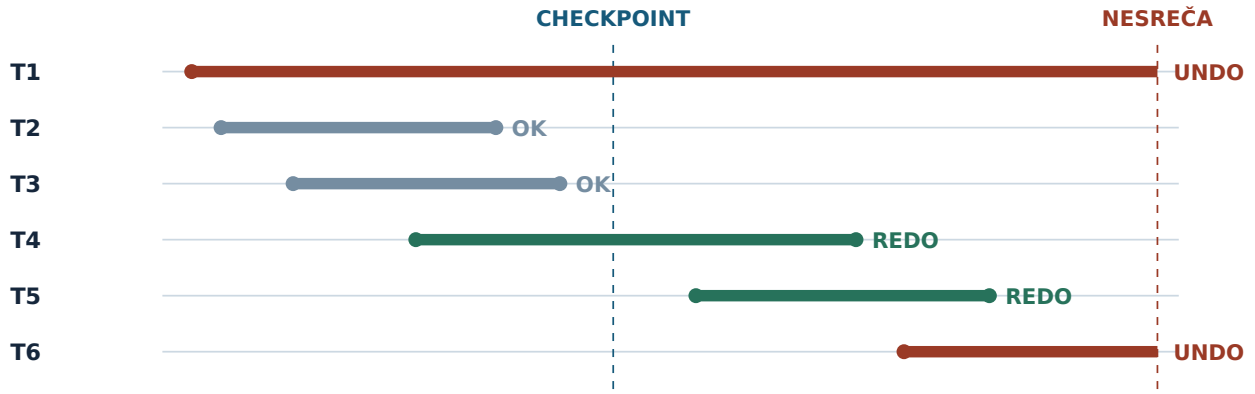
• **REDO:** ponovna uveljavitev sprememb **potrjene** transakcije; uporabimo nove vrednosti.

Pravilo WAL (write-ahead logging): ustrezen zapis v dnevniku mora biti trajno shranjen **pred** zapisom ustrezne spremenjene podatkovne strani. Preden sistem potrdi COMMIT, mora zagotoviti trajnost potrebnih dnevniških zapisov. WAL ne pomeni, da se podatkovna stran ob COMMIT že nujno zapiše na disk.

Kontrolna točka (CHECKPOINT)

6. Primer šestih transakcij iz predavanj

Navpična črta **CHECKPOINT** pomeni kontrolno točko, črta **NESREČA** trenutek odpovedi. Spodnji diagram uporablja poenostavljen model kontrolne točke iz predavanj: potrjene spremembe pred kontrolno točko so že trajno zapisane v bazo.



Transakciji	Zakaj?	Ukrep
T2, T3	COMMIT pred kontrolno točko; spremembe že vključene v stanje na disku.	**Brez dodatnega REDO**
T4, T5	COMMIT po kontrolni točki, pred nesrečo.	**REDO**
T1, T6	Ob nesreči še nimata COMMIT.	**UNDO**

7. Zakaj checkpoint skrajša obnovo?

Kontrolna točka uskladi dnevnik in stanje podatkov, zato pri obnovi ni treba brez potrebe ponavljati vseh starih, že varno zapisanih sprememb. Če se je transakcija začela pred kontrolno točko in je ob nesreči še aktivna, jo je še vedno treba obravnavati.

CHECKPOINT	SAVEPOINT
Mehanizem SUPB za obnavljanje po odpovedi.	SQL-točka znotraj posamezne transakcije.
Namen: zmanjšati delo upravljalca obnovljivosti.	Namen: omogočiti delni ROLLBACK znotraj transakcije.

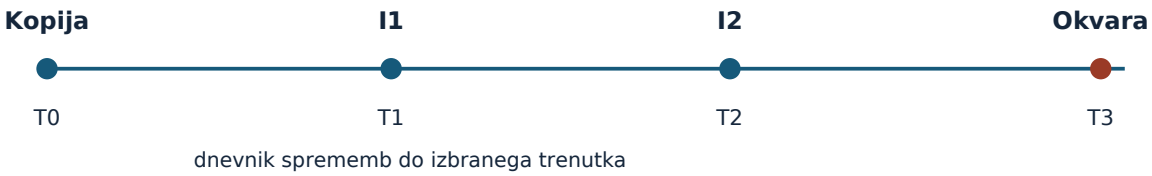
Pozor: dejanski SUPB lahko uporablja tudi delne (fuzzy) kontrolne točke; zato iz checkpointa v praksi ne smemo samodejno sklepati, da so vse podatkovne strani zapisane. Diagram je poenostavljen učni model, uporabljen v predavanjih.

Obnova z varnostno kopijo in dnevnikom

8. Kopija je nujna ob izgubi podatkovnega medija

Če odpove disk s podatkovno bazo, dnevnik sam ni dovolj, ker manjkajo podatkovne strani. Najprej obnovimo ustrezno kopijo, nato zaporedno uveljavimo potrebne potrjene spremembe iz ohranjenih dnevnikov do izbranega trenutka.

Vrsta kopije	Kaj vsebuje?	Kaj potrebujemo pri obnovi?
Popolna	Celotno bazo v času izdelave.	Zadnjo izbrano popolno kopijo + kasnejši dnevnik.
Inkrementalna	Spremembe od predhodne kopije (popolne ali inkrementalne).	Popolno kopijo + vse zaporedne inkrementalne kopije + potreben dnevnik.
Diferencialna	Vse spremembe od zadnje popolne kopije.	Popolno kopijo + zadnjo diferencialno kopijo + potreben dnevnik.



9. Primer poškodbe diska

V ponedeljek ob 00.00 izdelamo popolno kopijo, v torek ob 00.00 inkrementalno kopijo I1, v sredo ob 00.00 inkrementalno kopijo I2. Disk odpove v sredo ob 14.00. Dnevniki sprememb so shranjeni neodvisno in niso poškodovani.

Obnova: popolna kopija → I1 → I2 → ustrezni dnevniški zapisi od srede 00.00 do 14.00 (oziroma do zadnje razpoložljive potrjene transakcije). Nepotrjenih sprememb ne smemo uveljaviti.

10. Dve pomembni omejitvi

• **Obnovitev do trenutka (PITR)** uspe le, če imamo neprekinjeno ustrezno zgodovino dnevnika od izbrane kopije do cilja.

• **MariaDB:** binarni dnevnik, ki ga uporabljamo za ponovno uveljavitev potrjenih SQL-sprememb ob obnovi iz kopije, ni isto kot interni InnoDB redo/undo dnevnik, namenjen predvsem obnovi po sesutju.

V skupni fakultetni bazi **sandbox** ne izvajamo namernih okvar, brisanja sistemskih datotek ali celotne obnove strežnika. Naslednje naloge so zato pripravljene kot varna analiza dnevnikov in časovnic.

Naloge

Naloge so razporejene od prepoznavanja zapisov do obnove iz kopije. Rešitve so zbrane na koncu dokumenta.

N1. Prepoznaj UNDO in REDO

Dnevnik pred nesrečo: `START T1; UPDATE T1, A, 10→20; COMMIT T1; START T2; UPDATE T2, B, 8→5;` NESREČA. Odloči, kaj je treba ponoviti, kaj razveljaviti in kolikšni sta končni vrednosti A in B.

N2. Preveri rezultat obnove

Začetno stanje je X = 40, Y = 60. Dnevnik: `START T1; UPDATE T1, X, 40→55; UPDATE T1, Y, 60→45; COMMIT T1; START T2; UPDATE T2, X, 55→80; NESREČA`. Kaj se zgodi z obema transakcijama in kakšno stanje mora biti po obnovi?

N3. Kontrolna točka

Uporabi časovnico šestih transakcij na strani 2. Razvrsti T1–T6 v tri skupine: UNDO, REDO in brez ponovitve. Pojasni, zakaj se T4 obravnava, čeprav se je začela pred checkpointom.

N4. Izbira kopij

V ponedeljek je nastala popolna kopija F. V torek in sredo sta nastali inkrementalni kopiji I1 in I2. V četrtek odpove disk. Katere kopije potrebuješ? Kaj pa, če bi v sredo namesto I2 nastala diferencialna kopija D2?

N5. Pravilo WAL

Ažuriranje podatkovne strani X je že pripravljeno v medpomnilniku, ustrezni dnevniški zapis pa še ni trajno shranjen. Ali sme SUPB podatkovno stran zapisati na disk? Zakaj?

N6. Odloči, kaj je treba uporabiti

Primer	Odgovori: dnevnik, kopija + dnevnik ali SAVEPOINT?
a) Program se prekine pred COMMIT, podatkovni disk ostane delujoč.	
b) Disk s podatkovnimi stranmi je uničen.	
c) V tekoči transakciji želiš razveljaviti samo drugi INSERT.	

N7. Dopolni dnevnik (dodatna naloga)

A = 20, B = 30. T1 najprej zapiše A := 25, T2 nato B := 50. T2 izvede COMMIT, T1 pa se prekine brez COMMIT. Zapiši možne zapise START, UPDATE, COMMIT in ROLLBACK/nesreča ter določi, katera transakcija potrebuje UNDO in katera REDO.

Rešitve nalog

Najprej primerjaj status transakcije z zapisi COMMIT; nato določi obnovitveni ukrep.

R1. Rešitev N1

T1 → REDO (potrjena sprememba A := 20). **T2 → UNDO** (nepotrjena sprememba B := 8). Končni stanji sta **A = 20**, **B = 8**. Če je bila sprememba T1 že trajno zapisana, je njeno ponovno izvajanje lahko nepotrebno; za učni primer jo vseeno uvrstimo med REDO.

R2. Rešitev N2

T1 je potrjena, zato ohranimo obe spremembi: X := 55 in Y := 45. T2 ni potrjena, zato njeno spremembo X := 80 razveljavimo. Končni rezultat: **X = 55, Y = 45**. UNDO T2 mora uporabiti staro vrednost iz njegovega dnevniškega zapisa (55), ne začetne vrednosti 40.

R3. Rešitev N3

UNDO	REDO	Brez dodatne ponovitve
T1, T6	T4, T5	T2, T3

T4 se je začela pred kontrolno točko, a je COMMIT izvedla **po** njej. Zato ob nesreči njeni zapisi morda še niso v celoti na podatkovnih straneh; po učnem modelu izvedemo REDO.

R4. Rešitev N4

Za inkrementalne kopije potrebujemo **F → I1 → I2**, nato dnevnik od konca zadnje kopije naprej. Če v sredo nastane diferencialna kopija D2, potrebujemo **F → D2** in ustrezen kasnejši dnevnik; I1 pri tej poti ni potrebna.

R5. Rešitev N5

Ne. Po pravilu WAL mora biti dnevniški zapis trajno shranjen pred zapisom spremenjene podatkovne strani. Sicer lahko ob nesreči izgubimo podatke, potrebne za UNDO oziroma pravilno obnovitev.

R6. Rešitev N6

a)	b)	c)
Dnevnik: UNDO/obnova po odpovedi.	**Kopija + dnevnik**: fizična obnova in nadaljnje spremembe.	**SAVEPOINT**: delna razveljavitev znotraj transakcije.

R7. Rešitev N7

Primer pravilnega dnevnika: `START T1; UPDATE T1, A, 20→25; START T2; UPDATE T2, B, 30→50; COMMIT T2; NESREČA`. T1 je nepotrjena in potrebuje **UNDO** (A := 20), T2 je potrjena in je kandidat za **REDO** (B := 50). Rezultat po obnovi: **A = 20, B = 50**.

Ključno pravilo: **COMMIT določa, katere spremembe morajo ostati**. UNDO odstrani učinke nepotrjenih transakcij, REDO zagotovi učinke potrjenih. Pri poškodbi diska začnemo z ustrezno varnostno kopijo.