**BP** je skup podataka koji su pohranjeni i organizirani tako da mogu zadovoljiti zahtjeve korisnika

**SUBP** je programski sustav koji upravlja pristupom bazi podataka od strane više korisnika/aplikacija uz osiguravanje sigurnosti i integriteta baze podataka

* trajna pohrana podataka
* osiguravanje programskog sučelja (DDL, DML)
* zaštita podataka (integritet, pristup podacima, potpora za upravljanje transakcijama)
* optimiranje metoda pristupa podacima

**SUBP kao komponenta složenijih sustava:**

**klijent – poslužitelj arhitektura**: Sloj korisničkog sučelja (Prezentacijski sloj + Sloj upravljanja dijalogom), Sloj obrade podataka (Sloj aplikacijske jezgre), Sloj upravljanja podacima (Sloj pristupa podacima + Sloj pohrane podataka)

**+**: Specijalizacija sklopovlja i programske potpore, povećana skalabilnost , povećana sigurnost, bolja zaštita integriteta podataka, povećana modularnost

**-** : povećanje konstrukcijske složenosti, lošija performansa radi fizičke razdvojenosti poslovne logike i pohrane podataka

**Komponente SUBP-a**

**Uspostava komunikacije s korisnikom** = **menadžer komunikacija** (uspostaviti i održati vezu prema klijentskoj aplikaciji, utvrditi sigurnosni status korisnika, prihvatiti zahtjeve, vratiti rezultate) + **menadžer procesa** (pokreće „server process“ (dretvu izvršavanja) koja će biti zadužena za obrađivanje zahtjeva vezanih za dotičnu korisničku sjednicu)

**Prevođenje i optimizacija** = **parser** + **modulu za pojednostavljenje i normalizaciju upita** (rezultat optimiranja upita je interni plan izvršavanja, kojim je određen optimalni redoslijed izvršavanja operacija) + **izvršitelj plana** (implementira operacije koristeći module niže razine) + **modul za autorizaciju**

**Izvršavanje** = **modul za izvršavanje** + **menadžer transakcije** (objedinjuje pojedine zahtjeve u transakcijena temelju transakcijskih naredbi) + **menadžer redoslijeda izvršavanja** (uz **menadžer zaključavanja** određuje redoslijed izvršavanja operacija kojim se uz istodobno izvršavanje transakcija nađe narušiti konzistentnost baze podataka) + **menadžer pristupa** (koristi prikladne algoritme i strukture podataka organiziranje i pristupanja podacima u sekundarnoj memoriji) +

**menadžer međuspremnika** (osigurava efikasnost prijenosa podataka između primarne i sekundarne memorije) + **menadžer obnove** (osigurava mogućnost obnove baze podataka u slučaju pogreške/kvara)

**Rezultati** = **modul za izvršavanje** dobivene rezultate smješta u međuspremnik menadžera komunikacija koji ih prosljeđuje klijentskoj aplikaciji

**Zajedničke komponente** = **menadžer procesa** (pokreće, zaustavlja i sinkronizira procese) + **menadžer memorije** (upravlja alokacijom i dealokacijom primarne memorije) + **menadžer kataloga** (osigurava meta-podatke koji se koriste za autorizaciju, parsiranje i optimizaciju upita) + niz pomodnih **modula** za nadgledanje i administraciju sustava

**Procesi u SUBP-u**

**User process** = proces klijentske aplikacije

**Server process** = zaprima zahtjeve klijentskih aplikacija i vraća rezultate + upravlja korisničkom sjednicom

**Lock manager process** = implementira funkcionalnost menadžera zaključavanja

**Database writer process** = 1 ili više, obavljaju prijenos podataka između primarne i sekundarne memorije

**Log writer process** = upravlja zapisivanjem dnevnika u stabilnu memoriju

**Checkpoint process** = periodički obavlja proces zapisivanja kontrolne točke

**Organizacija procesa u SUBP-u**

**OS process per dbms thread of execution** = koriste se mehanizmi operacijskog sustava + niska skalabilnost + relativno jednostavna implementacija

**OS thread per dbms thread of execution** = jedan OS proces obavlja sve dretve + visoka skalabilnost + složena implementacija + otežana prenosivost SUBP-a na različite operacijske sustave

**Process pool** = više procesa (virtualni procesori), svaki izvršava više dretvi + virtualni procesori svrstani u razrede + broj virtualnih procesora može se dinamički mijenjati + virtualni procesori se po potrebi mogu vezati uz fizički prostor

**Mediji za pohranu podataka**

**Priručna memorija** = vrlo velike brzine pristupa podacima, vrlo skupa, vrlo ograničenog kapaciteta (mb)

**Glavna memorija** = velika brzina pristupa podacima, relativno skupa, kapacitet u gb + neprikladna za trajnu pohranu baze podataka jer: nedovoljan kapacitet + nepostojana memorija (sadržaj se gubi gubitkom napajanja) + previsoka cijena

**Sekundarna memorija** = sporija, vedeg kapaciteta u odnosu na primarnu, vedina današnjih baza podataka + flash memorija (postojana memorija, upotreba u ugrađenim računalima) + magnetski disk )najčešće korišteni medij za pohranu baze podataka)

**Tercijarna memorija** = izrada sigurnosnih kopija i vrlo velike baze podataka + optički mediji + magnetske trake + jukebox ili tape silo (za vrlo velike baze podataka)

**Karakteristike medija za pohranu baza podataka**

**Nepostojana memorija** = primarna memorija čiji se sadržaj gubi u slučaju pogreške sustava

**Postojana memorija** = sadržaj ostaje sačuvan u slučaju pogreške sustava + mogude su pogreške medija + sekundarna memorija

**Stabilna memorija** = postojana memorija čiji se sadržaj *nikad\** ne gubi + replikacija podataka na medijima s postojanom memorijom (RAID, CD/DVD, SAN)

**Organizacija podataka u sekundarnoj memoriji** je logički zapisi (n-torke) pohranjuju se u fizičke blokove unaprijed određene i nepromjenjive veličina, što znači da jedna n-torka može biti pohranjena u više blokova i više n-torki može biti pohranjeno u jednom bloku

**Pristup podacima u sekundarnoj memoriji** podaci se između primarne i sekundarne memorije prenose u blokovima + **dominiraju troškovi u/i operacija**, jer je vrijeme obavljanja U/I operacija mnogo veće od obavljanja operacija u primarnoj memoriji

**Organizacija podataka u primarnoj memoriji** – raspoloživa primarna memorija podijeljena u blokove jednake blokovima sekundarne + komponente SUBP-a koriste sekundarnu memoriju isključivo korištenjem menadžera međuspremnika

**Uloga menadžera međuspremnika** = proces koji želi čitati podatak šalje *read* zahtjev menadžeru međuspremnika, a menadžer međuspremnika u slučaju da se blok s traženim podatkom ne nalazi u međuspremniku dohvaća blok iz sekundarne memorije operacijom *input* + prilikom pisanja obavlja se slična procedura, samo s naredbama *write* i *output* + za izbacivanje blokova iz međuspremnika najčešće se koristi algoritam LRU + u slučaju više *server processa* nad istom bazom podataka dijeljena memorija omogućava sinkronizaciju podataka korištenjem dijeljenog međuspremnika

**Utjecaj međuspremnika operacijskog sustava** = OS koristi vlastite mehanizme za prijenos podataka između radnog spremnika i sekundarne memorije

**Double buffering** = redundantni i štetni + troši primarnu memoriju koja se mogla bolje iskoristiti + troši procesorsko vrijeme na nepotrebno kopiranje podataka (pisanje=međuspremnik SUBP>međuspremnik OS>sekundarna memorija)

**Cooked files** = korištenje funkcija OS-a za pristup podacima + lošije performanse zbog toga jer se logički kontinuirani podaci ne spremaju u fizički kontinuiran prostor

**Raw disk I/O** = SUBP potpuno upravlja procesima pristupa podataka zaobilazedi mehanizme OS-a + podacima se pristupa na razini bajta, a ne datotečnog sustava + bolje performanse zbog toga jer se logički kontinuirani podaci spremaju u fizički kontinuirani prostor - particije sekundarne memorije za SUBP moraju se unaprijed definirati

**Direct I/O** = korištenje izravnog pristupa memoriji u datotečnom sustavu

**Fizičke jedinice pohrane podataka u SUBP-u**

**Grumen (chunk)** = najveća jedinica fizičkog prostora koja se od strane OS-a dodjeljuje SUBP-u + particije diska koje se koriste kao *raw device* + datoteke kao *cooked file*

**Područje (extent)** = fizička jedinica prostora koja se kreira u trenutku stvaranja relacija ili kasnije tijekom njenog proširenja + CREATE TABLE … ( … ) EXTENT SIZE m NEXT SIZE n + osim ponovnim stvaranjem relacije, inicijalna veličina područja se ne može promijeniti, a zauzeta područja se ne mogu presložiti + dva područja iste relacije neposredno jedno iza drugog spajaju se u jedno područje

**Logičke jedinice podataka**

**Prostora baze podataka (database space)** = u jednom ili više dijelova jednog ili više grumena + može se proširivati novim grumenima + *root dbspace* sadrži sistemske podatke potrebne za rad poslužitelja + CREATE DATABASE dbName IN dbspace1 + CREATE TABLE tabname ( … ) IN dbspace1

**Prostor relacije (table space)** = skup svih područja (extent) koja sadrže podatke jedne relacije

**Fizička organizacija** = utječe na učinkovitost SUBP-a + pojam fizičke organizacije odnosi se na organizaciju datoteka i metode pristupa + važniji ciljevi fizičke organizacije: (minimizirati broj U/I operacija pri pohrani i dohvatu podataka, minimizirati utrošak prostora za pohranu + omogućiti različite metode pristupa koje se koriste za pronalaženje fizičke pozicije zapisa (ili fizičke pozicije bloka u kojem se taj zapis nalazi) na temelju vrijednosti ključa pretrage)

**Strukture podataka i metode pristupa podacima**

**Neporedana datoteka** = učinkovit upis, linearno (slijedno) pretraživanje + u kombinaciji s drugim organizacijama (npr. B-stablo)

**Poredana datoteka**

**Raspršena datoteka** = hash-funkcijom se izračuna adresa pretinca, a u pretincu piše fizička adresa bloka u kojem se nalazi zapis + najbrža metoda za pretraživanje prema ključu + efikasno dodavanje i brisanje + problemi: eventualni preljevi -> ulančavanje, pretraživanje samo prema ključu, ne i intervalu, zapisi nisu sortirani

**Indeksno-slijedna organizacija**

**B-stablo**

**Transakcija** je niz logički povezanih operacija koje se izvršavaju kao cjelina i prevode bazu podataka iz jednog u drugo konzistentno stanje

**ACID svojstva transakcije**

**Atomarnost (atomicity)** (osigiurava obnovu sustava)= ili su efekti svih operacija transakcije pravilno pohranjeni u bazu podataka ili transakcija nije utjecala na stanje baze podataka

**Konzistentnost (consistency)** = konzistentnost (zadovoljena SVA definirana ograničenja integriteta)+ ako transakcija započne u bazi podataka koja je u konzistentnom stanju te se obavlja u odsustvu drugih transakcija i pogrešaka sustava, tada de nakon završetka transakcije baza podataka biti u konzistentnom stanju (KOREKTNA TRANSAKCIJA) + provjera ograničenja integriteta: IMMEDIATE (nakon izvođenja svake DML naredbe, *statement-level rollback*) + DEFFERED (pri izvršavanju COMMIT WORK) + SET CONSTRAINTS constrname { IMMEDIATE | DEFFERED } + SET CONSTRAINTS ALL { IMMEDIATE | DEFFERED }

**Izolacija (isolation)** (osigurava sustav za kontrolu istodobnog pristupa)= učinak transakcija koje se obavljaju istodobno (paralelno) mora biti jednak učinku tih istih transakcija koje bi se obavljale jedna iza druge

**Izdržljivost(durability)** = izmjene u bazi podataka koje su rezultat potvrđenih transakcija ne smiju biti izgubljene neovisno o vrsti kvara sustava koji bi se mogao dogoditi nakon što je transakcija potvrđena

**Prekid i poništavanje transakcije** = prekid transakcije izazvan programom ili zahtjevom korisnika + prekid transakcije izazvan od strane SUBP-a + SQL naredba ROLLBACK WORK predstavlja zahtjev za prekidom transakcije (*abort*), nakon kojeg SUBP odmah obavlja i *rollback* + ako se SQL naredba ne može obaviti u cijelosti, obavlja se poništavanje na razini SQL naredbe (*statement-level rollback*)

**Stanja transakcija**

**Aktivna** = operacije se obavljaju bez pogreške ili se događaju samo pogreške kojima transakcija eksplicitno upravlja i za koje postoji kompenzacijska akcija nakon koje se transakcija može nastaviti (a ako kompenzacijska akcija ne postoji aplikacija upućuje eksplicitni zahtjev za poništavanjem transakcije) + prijelazi: djelomično završena (end\_t), neispravna (abort)

**Djelomično završena** = sve naredbe osim naredbe za potvrđivanje transakcije su obavljene + prijelazi: neispravna (abort), potvrđena (commit)

**Neispravna** = ustanovljena je pogreška zbog koje daljnje izvršavanja transakcije nije moguće + prijelazi: poništena (rollback)

**Potvrđena** = svi efekti transakcije su trajno zabilježeni u bazi podataka

**Poništena** = efekti operacija koje su obavljene u okviru transakcije su poništeni

**Završena** = stanje potvrđena + stanje poništena

**Klasifikacija transakcija**

**On-line transakcije** = relativno kratko vrijeme izvođenja, pristup malom broju elemenata baze podataka

**Batch transakcije** = trajanje u min, h ili d, pristup velikom broju elemenata baze podataka

**Konverzacijske (interaktivne) transakcije** = za vrijeme obavljanja transakcije postoji interakcija s korisnikom

**Pojmovi i oznake** = transakcije se označavaju oznakama **t1, t2, …, ti, tj, …** + čitanje nad elementom ***x*** baze podataka u transakciji **i** se označava **ri[x]**, a pisanje **wi[x]** + operacija *abort* označava se s **ai**, a *commit* s **ci**

**Model transakcije** = transakcija se modelira kao parcijalni poredak, pri čemu vrijedi: dopuštene su operacije **ri[x]**, **wi[x]**, **ai**, **ci** + u transakciji se nalazi jedna i samo jedna od operacija *abort* ili *commit* + operacija *abort* ili *commit* mora biti posljednja operacija svake transakcije + ako transakcija obavlja operaciju čitanja i pisanja nad istim elementom ***x***, za te dvije operacije mora biti definiran odnos, jedna mora prethoditi drugoj

**Zadada sustava za obnovu** = u slučaju pogreške dovesti SUBP u stanje ekvivalentno zadnjem konzistentnom stanju prije pogreške + ostvariti čim veću sposobnost oporavka SUBP-a u odnosu na različite vrste pogrešaka + ostvariti čim veću raspoloživost SUBP-a

**Dijelovi supb-a koji sudjeluju u obnovi** = menadžer transakcija +menadžer redoslijeda izvršavanja + menadžer podataka = menadžer obnove + menadžer međuspremnika

**Postupci obnove** = postupci koji se provode tijekom redovitog funkcioniranja sustava i osiguravaju prikupljanje dovoljnog broja podataka za obnovu baze podataka nakon pogreške + postupci pomoću kojih se u slučaju pogreške sustav na pouzdan način vraća u konzistentno stanje + koristi se **redundancija** – svaki podatak mora se modi rekonstruirati iz redundantnog podatka pohranjenog u nekom drugom dijelu sustava

**Vrste pogrešaka**

**Pogreške transakcija** = pogreška uzrokovana ulaskom transakcije u stanje koje onemogućava daljnje ispravno obavljanje operacija transakcije + SUBP implicitno prekida i poništava dotičnu transakciju + ne dolazi do gubitka sadržaja niti postojane niti nepostojane memorije + ne dolazi u zastoju funkcioniranja SUBP-a + ako transakcija eksplicitno rukuje pogreškom, onda se ta pogreška ne smatra pogreškom transakcije

**Pogreške sustava** = pogreška sklopovlja, programske podrške, nestanak napajanja + uzrokuje trenutačni prekid rada SUBP-a i gubitak sadržaja međuspremnika + gubitak sadržaja nepostojane memorije, prekid rada, postojana memorija očuvana

**Pogreške medija** = djelomični ili potpuni gubitak podataka u postojanoj memoriji + katastrofe su specifičan oblik

**Adresni prostor SUBP-a** = adresni prostor transakcije + adresni prostor međuspremnika + adresni prostor bloka na diskovima

**Temeljne operacije transakcije**

**input (x)** = iz adresnog prostora diska kopiraj blok koji sadrži element *x* u međuspremnik

**read (x, p)** = ako blok u kojem je *x* nije u međuspremniku, obavi *input(x)* + kopiraj sadržaj *x* iz međuspremnika u lokalnu varijablu transakcije *p*

**write(x, p)** = ako blok u kojem je *x* nije u međuspremniku, obavi *input(x)* + kopiraj sadržaj lokalne varijable transakcije *p* u međuspremnik

**output (x)** = kopiraj blok međuspremnika u kojem je *x* u odgovarajudi blok na disku

**Dnevnik** = aktivnosti transakcija bilježe se u dnevniku + dnevni obuhvaća niz zapisa dnevnika u obliku datoteke u koju se zapisi dodaju isključivo na začelje + u isti dnevnik bilježe se aktivnosti svih transakcija u tijeku + zapisi dnevnika stvaraju se u nepostojanoj memoriji, a menadžer međuspremnika upravlja njihovim zapisivanjem u stabilnu memoriju

**Dnevnik izmjena** = zapis koji opisuje početak transakcije **< start ti >** + zapis koji opisuje operaciju pisanja **< ti**, **xj**, **vi**, **v2 >** (*write*)+ zapis koji opisuje potvrđivanje transakcije **< commit ti >** + zapis koji opisuje prekid transakcije **< abort t >**

**Završena transakcija** = u dnevniku je zabilježen ishod transakcije + nije bitno jesu li podaci pohranjeni u postojanu memoriju

**Nekompletna transakcija** = u dnevniku ne postoji ni zapis *abort* ni *commit* + nije bitno jesu li podaci zapisani u postojanu memoriju

**Točka potvrđivanja** = transakcija ti je dostigla točku potvrđivanja u trenutku kada je u stabilnu memoriju upisan zapis dnevnika < commit ti >

**Tehnike obnove**

**Undo (undo/no-redo)** = tijekom obnove neke se transakcije poništavaju

**Redo (no-undo/redo)** = tijekom obnove neke se transakcije ponovno obavljaju

**Undo/redo** = tijekom obnove neke se transakcije poništavaju, a neke ponovno obavljaju

**No-undo/no-redo** = bez dnevnika, koriste se kopije blokova s podacima

**Undo tehnika obnove** = zapis koji opisuje operaciju pisanja sadrži samo staru vrijednost elementa **< ti** , **xj** , **v >**

**Pravila vođenja dnevnika:** ako transakcija *t* obavlja operaciju *write(x, v)*, operacija *flush log* koja de upisati zapis dnevnika *< t, x, v >* mora biti obavljena prije operacije *output(x)* + operacija *flush log* kojom se zapis dnevnika *< commit t >* upisuje u stabilnu memoriju obavlja se isključivo nakon što su obavljene sve *output(x)* operacije transakcije *t*

**Postupak obnove:** čitati dnevnik od kraja prema početku teza svaki zapis iz dnevnika + ako pročitaš zapis *< commit ti >*, dodaj *ti* u listu potvrđenih transakcija + ako pročitaš zapis *< ti , x, v >* (ako je *ti* u listi potvrđenih transakcija ne činiti ništa +inače ako *ti* nije u listi potvrđenih transakcija, obavi operaciju ***undo(x, v)***)

**Nedostaci:** = potreba obavljanja većeg broja u/i operacija prije potvrđivanja transakcije + ograničena mogućnost obnove baze podataka pomoću arhivske kopije

**Redo tehnika obnove** = zapis koji opisuje operaciju pisanja sadrži samo novu vrijednost elementa **< ti , xj , v >**

**Pravila vođenja dnevnika:** = svaka operacija *output(x)* transakcije *t* treba se obaviti tek nakon što su svi zapisi dnevnika *< t, x, v >* i zapis *< commit t >* transakcije upisani u stabilnu memoriju

**Postupak obnove:** čitati dnevnik od početka prema kraju + ako za transakciju *ti* u dnevniku postoji zapis *< commit ti >* tada obaviti operaciju ***redo (xj , v)*** za svaki zapis *< ti , xj, v >* iz dnevnika + inače ne činiti ništa

**Nedostaci** (u odnosu na *undo*): zahtjev za velikim međuspremnikom podataka

**Prednosti** (u odnosu na *undo*): manja ograničenja redoslijeda obavljanja operacija +veća mogućnost obnove baze podataka iz arhivske kopije

**Undo/redo tehnika obnove** = zapis koji opisuje operaciju pisanja **< ti , xj , v1 , v2 >**

**Pravila vođenja dnevnika:** ako transakcija *t* obavlja operaciju *write(x, t2)* operacija *flush log* kojom je zapis dnevnika *< t, x, v1, v2 >* upisan u stabilnu memoriju mora biti obavljena prije operacije *output(x)*

**Postupak obnove:** čitati dnevnik od početka prema kraju:ako za transakciju *ti* u dnevniku postoji zapis *< commit ti >* tada obaviti operaciju ***redo(xj , v2)*** za svaki zapis *< ti , xj , v1 , v2 >* iz dnevnika + čitati dnevnik od kraja prema početku:ako za transakciju *ti* u dnevniku ne postoji zapis *< commit ti >* tada obaviti operaciju ***undo(xj , v1)*** za svaki zapis *< ti , xj , v1 , v2 >* iz dnevnika

**Prednost** (u odnosu na *redo*): nema potrebe za velikim međuspremnikom podataka

**Nedostatak** (u odnosu na *undo* i *redo*): povećani utrošak prostora za pohranu dnevnika

**Fizički dnevnik** = zapisi dnevnika sadrže stare i/ili nove vrijednosti cijelih blokova + nepotreban utrošak prostora za dnevnik + problem poništavanja transakcija (npr. Dva elementa koja se mijenjaju u istom bloku)

**Logički dnevnik** = zapisi dnevnika sadrže podatke o izmjenama unutar bloka + „na nižoj logičkoj razini“ koriste se u slučaju jednostavnih operacija + „na višoj logičkoj razini“ koriste se za pri obavljanju operacija nad b-stablom

**Sadržaj dnevnika – omogudavanje poništavanja transakcija** = pokazivač prema prethodnom zapisu iste transakcije u dnevniku = omogućava poništavanje transakcije zbog pogreške transakcije ili eksplicitnog zahtjeva za poništavanjem transakcije

**Kontrolna točka** = periodički sustav određuje kontrolnu točku: SUBP prestaje obavljati operacije koje zahtijevaju pisanje u međuspremnik podataka ili međuspremnik dnevnika + sadržaj međuspremnika dnevnika zapisuje se u stabilnu memoriju + sadržaj međuspremnika podataka zapisuje se u postojanu memoriju + u dnevnik se upisuje zapis **< chkpt l >** (l je lista identifikatora transakcija aktivnih u trenutku kontrolne točke) + SUBP nastavlja obavljati operacije koje zahtijevaju pisanje u međuspremnik podataka ili dnevnika

**Postupak obnove:** = formiraj dvije prazne liste: redo-list i undo-list + čitaj dnevnik unatrag od kraja dnevnika do prvog *< chkpt l >* zapisa (za svaki zapis oblika *< commit ti >* dodaj *ti* u redo-list +za svaki zapis oblika *< start ti >* ako se *ti* ne nalazi u redo-list dodaj *ti* u undo-list) + svaku transakciju *ti* iz *l* koja nije u redo-list dodaj u undo-list + čitaj dnevnik unatrag od kraja (za svaki zapis *< ti , x , v1 , v2 >* ako je *ti* u undo-listi, obavi ***undo(x, v1)*** +čitanje dnevnika unatrag prestaje kad se pronađu zapisi *< start ti >* za svaku transakciju *ti* iz undo-liste) + čitaj dnevnik od zadnje kontrolne točke do kraja (za svaki zapis *< ti , x , v1 , v2 >* ako je *ti* u redo-listi, obavi ***redo(x, v2)***)

**Predslika bloka** = originalna vrijednost bloka podataka, kakva je bila u trenutku posljednje kontrolne točke + koristi se za brzu obnovu

**Pravila vođenja fizičkog dnevnika** = fizički dnevnik se počinje puniti predslikama nakon kontrolne točke + prije prve izmjene bloka međuspremnika podataka, predslika bloka upisuje se u međuspremnik fizičkog dnevnika (blok se može mijenjati više puta nakon kontrolne točke, ali se pohranjuje 1 predslika + ne pohranjuju se predslike blokova koji se nisu promijenili) + prije nego se izmijenjeni blok podataka pohrani u postojanoj memoriji, njegova preslika se obavezno pohranjuje u fizički dnevnik u stabilnoj memoriji + u kontrolnoj točki se svi blokovi iz međuspremnika podataka zapisuju u postojanu memoriju, a fizički dnevnik se prazni

**Postupak brze obnove** = ako u trenutku ponovnog pokretanja sustava fizički dnevnik nije prazan -> bila je pogreška i menadžer obnove provodi postupak brze obnove: sve predslike iz fizičkog dnevnika prebacuju se iz fizičkog dnevnika u međuspremnike podataka i zapisuju u blokove podataka u postojanoj memoriji -> stanje u trenutku posljednje kontrolne točke + obavlja se *redo* za sve zapise dnevnika nastale nakon posljednje kontrolne točke + obavlja se *undo* za zapise dnevnika onih transakcija koje nisu bile potvrđene do trenutka u kojem je nastala pogreška sustava

**Upravljanje dnevnicima** = zapisi dnevnika pohranjuju se u stabilnu memoriju + prostor za pohranu dnevnika podijeljen je u datoteke dnevnika, koje se ciklički izmjenjuju + čim se popuni, datoteka dnevnika može se arhivirati + u dostupnoj stabilnoj memoriji moraju se čuvati: datoteka dnevnika koja sadrži posljednju kontrolnu točku i sve datoteke dnevnika koje slijede iza nje + datoteke dnevnika koje sadrže zapise transakcija koje su bile aktivne u trenutku posljednje kontrolne točke + problem dugih transakcija uzrokuje nedostatak slobodnih datoteka dnevnika

**Obnova nakon pogreške medija**

**problem**: izgubljen sadržaj postojane (ili čak stabilne) memorije (baza podataka + dnevnici)

**rješenje**: korištenje redundantnih kopija (arhiva) sadržaja postojane i stabilne memorije + kopija se mora čuvati na mediju s nezavisnim modalitetom kvarova

**Arhiviranje baze podataka** = periodičko kopiranje cijelog sadržaja baze podataka na uređaj s nezavisnim modalitetom kvarova + u slučaju kvara medija baza podataka vraća se u stanje u trenutku posljednje arhive + zaustavljanje sustava za vrijeme izrade arhive + izgubljen rezultat transakcija provedenim nakon izrade posljednje arhive + rješava se arhiviranjem svih datoteka dnevnika nakon izrade posljednje arhive + arhiviranje cijele baze je dugotrajan postupak i zahtijeva puno prostora

**Kreiranje baze podataka u ovisnosti o dnevnicima**

**Create databse dbname** = ne kreiraju se zapisi logičkog dnevnika + nije mogude poništavanje transakcija (ni na razini naredbe) + inicijalno punjenje baze podataka

**Create database dbname with buffered log** = zapisi logičkog dnevnika ne zapisuju se u stabilnu memoriju tijekompotvrđivanja transakcije + koristiti kada se može dopustiti gubitak određenog dijela podataka

**Create database dbnane with log** = zapisi logičkog dnevnika koji pripadaju nekoj transakciji zapisuju se u stabilnu memoriju najkasnije u trenutku potvrđivanja te transakcije