1. Osnove in uvod

Podatkovna baza je zbirka medseboj logično povezanih podatkov, ki zadovoljujejo informacijske potrebe organizacije in njenih poslovnih procesov. PB je mehanizirana, večuporabniška, formalno definirana in centralno nadzorovana zbirka podatkov.

Aplikacija je program, ki v okviru IS delno ali v celoti podpira enega ali več poslovnih procesov in za shranjevanje podatkov uporablja PB.

Podatki so dejstva predstavljena z vrednostmi (številke, znaki, simboli), ki imajo pomen v določenem kontekstu.

Podatek je lahko:
- diskreten ; če se pri predstavitvi uporabljajo simboli npr. 25°C
- analogen ; če se za predstavitev uporablja kakšna fizikalna veličina npr. dolžina živosrebrovega stolpca

2. Relacijska algebra

Relacijska algebra vsebuje pet osnovnih operacij: selekcija, projekcija, kartezijski produkt, unija in razlika.

Izpeljane operacije: stik, presek, količnik.

Stične operacije: stik Theta, stik Equijoin, naravni stik in odprti stik

Vsak jezik, s katerim pridobimo relacije, ki jih je moč pridobiti z relacijskim računom je **relacijsko popoln**.

3. SQL in QBE

SQL je transformacijsko usmerjen jezik, ki ga sestavljata dve skupini ukazov (DML in DDL).

DDL (Data Definition Language) zajema stavke za manipulacijo s strukturo PB:
- omogoča definiranje podatkovnih struktur, podatkovnih tipov ter integritetnih omejitev
- vse specifikacije oz. definicije so shranjene v podatkovni bazi (podatkovni slovar ali sistemski katalog).
- kreiranje tabele pomeni poseg v sistemski katalog
- praviloma se v okviru SUPB uporablja isti DDL za vse tri sheme (ali vsaj za zunanjo in konceptualno)

DML (Data Manipulation Language) zajema: SELECT, INSERT, DELETE, UPDATE.
- omogoča manipuliranje s podatki (insert, update, delete)

Lastnosti SQL:
- enostaven
- nepostopkoven
- uporaben v okviru številnih nalog
- obstaja ISO standard za SQL
- SQL de-facto in tudi uradno standardni jezik za delo z relacijskimi PB
- SQL je do sedaj edini široko sprejet standardni jezik za upravljanje s podatki
- pojavile so se specializirane implementacije SQL npr. OLAP
- SQL stavki so sestavljeni iz rezerviranih in uporabniško definiranih besed

QBE je vizualen način dostopa do podatkov PB z uporabo vzorcev poizvedb.

QBE omogoča (poizvedba ki jo uporabnik napiše v QBE se v ozadju prevede v SQL):
- poizvedeti po podatkih ene ali več tabel
- določiti stolpce, ki jih želimo imeti v pogovoru
- določiti kriterije za izbiro vrstic

4. Nivoji abstrakcije, podatkovna neodvisnost in osnove SUPB

Zakaj več nivojev abstrakcije?
- vsi uporabniki morajo imeti možnost dostopa do istih podatkov, a vsak ob različnem pogledu na iste podatke
- spremembe pogleda na podatke enega uporabnika, ne smejo vplivati na poglede ostalih uporabnikov
- uporabnikom naj ne bo potrebno poznati podrobnosti fizičnega nivoja, niti podrobnosti o tem, kako in kje so shranjeni podatki ter kako se z njimi manipulira
- DBA mora imeti možnost spremeniti fizične parametre podatkovnih struktur ne da bi to kaj vplivalo na poglede uporabnikov

Nivoji abstrakcije – s tri-nivojsko abstrakcijo podatkov dosežemo podatkovno neodvisnost
- Zunanja shema – zunanji nivo (External level)
  - uporabnikov pogled na podatke
  - predstavlja tisti del PB, ki je relevanten za posameznega uporabnika
  - različni pogledi imajo lahko različne predstavitve istega podatka (EMŠO, datum rojstva)

- Konceptualna shema – konceptualni nivo (Conceptual level)
  - združen pogled na podatke
  - skrije podrobnosti o tem, kako so podatki dejansko shranjeni na disku, o strukturi datotek in o indeksih
  - sprememba sheme lahko vpliva na učinkovitost
  - opisuje kateri podatki so shranjeni v PB in razmerja med njimi
  - ne vsebuje podrobnosti o fizičnih parametrih (le atribut in podatkovni tip, brez fizičnih podrobnosti)
  - opisuje integritetne omejitve

- Fizična shema - fizični nivo (Internal level)
  - fizična predstavitev in organizacija podatkov v PB
  - opisuje, kako in kje (v fizičnem smislu) so podatki shranjeni v PB

- Vsaka podatkovna baza ima le eno konceptualno in fizično shemo

Podatkovna neodvisnost - zagotavlja, da višji nivo ostane nespremenjen kljub spremembi na nižjem nivoju
- Logična podatkovna neodvisnost
  - zagotavlja, da ostanejo ob spremembah v konceptualni shemi zunanje sheme nespremenjene
  - omogoča, da ostanejo aplikacije (njihova programska koda) kljub (morebitnim) spremembam v konceptualni shemi nespremenjene

- Fizična podatkovna neodvisnost
  - zagotavlja, da ob spremembi v fizični shemi ostane konceptualna shema nespremenjena (zamenjava podatkovnega strežnika)

Poizvedovanje v PB:
- enostavnost pridobivanja informacij iz PB je ključna prednost SUPB za uporabnike
- relacijske PB omogočajo uporabnikom postavljati enostavna vprašanja (poizvedbe), s katerimi pridobivajo podatke/informacije
- poizvedbe se podaja v jeziku, ki je prirejen za opisovanje poizvedb – poizvedovalni jezik
- relacijski model podpira zelo močne poizvedovalne jezike
- ena pomembnih nalog SUPB je optimizacija poizvedb, tako da se te čim hitreje izvedejo
- učinkovitost poizvedb je močno odvisna od načina, kako so podatki shranjeni v fizični obliki ter indeksirani
- SUPB omogoča uporabnikom izvajati poizvedbe ter kreirati in posodabljati vrednosti s pomočjo DML

Omejitve datotečnih sistemov:
- Ločevanje in izolacija podatkov
  - Vsaka aplikacija obvladuje svoje podatke
  - V okviru ene aplikacije lahko dostopamo le do podatkov te aplikacije (ne vemo za razpoložljive podatke v drugih aplikacijah)
- Podvajanje podatkov
  - Podvajanje vnosa – iste podatke vnašamo večkrat in na več mest
  - Odvečna poraba prostora na disku ter potencialna možnost za neskladnost
- Podatkovna odvisnost
  - Struktura podatkov je definirana v aplikaciji; sprememba v strukturi podatkov zahteva spremembo v aplikaciji
- Neskladnost med oblikami datotek
- Aplikacije, napisane v različnih programskih jezikih, ne morejo enostavno dostopati do datotek drugih sistemov

SUPB (Sistem za Upravljanje Podatkovnih Baz) je skupek programske opreme, ki omogoča kreiranje, vzdrževanje in nadzor nad dostopom do podatkov v podatkovni bazi:
- kreiranje podatkovnih struktur (DDL)
- vzdrževanje podatkov (DML)
- povpraševalni jeziki (query language).

Lastnosti in funkcije SUPB:
– podatkovna neodvisnost: Programi so neodvisni od predstavitve podatkov in načina shranjevanja podatkov
– zagotavlja abstrakcijo podatkov in ločuje programe od podrobnosti predstavitve podatkov.
– učinkovit dostop do podatkov
– varnost in integriteta podatkov
– administracija podatkov
– zagotavlja skladnost podatkov
– zagotavlja in nadzira sočasni dostop
– zagotavlja mehanizme za obnovo podatkov
– skrajša čas razvoja programov

Izjeme, ko uporaba SUPB ni primerna:
– za specializirane aplikacije (npr.: v realnem času) klasični SUPB niso primerni
– če SUPB ne podpira dela s podatki na način, ki ga zahteva program

Razmerje med SUPB in PB:
– en SUPB upravlja z eno ali več instanc PB na enem podatkovnem strežniku
– na enem podatkovnem strežniku je lahko nameščenih več SUPB

Komponente SUPB:
- Strojna oprema
  - podatkovni strežnik (mora imeti dovolj pomnilnika, hitri pomnilnik in diskovni prostor)
  - ostala strojna oprema
- Programska oprema
  - SUPB, operacijski sistem, omrežna programska oprema
  - različna sistemska oprema (razvojna orodja, orodja za dostop do podatkov, orodja za upravljanje s PB in SUPB)
  - specializirane (poslovne) aplikacije, datotečni sistemi in SUPB
- Podatki
- Postopki
  - Načini prijave
  - Uporaba posameznih orodij
  - Zagon (startup) in zaustavitev (shutdown) podatkovne baze
  - Izdelava varnostnih kopij
  - Obvladovanje nesreč/okvar
- Ljudje
  - Skrbnik podatkov
  - Skrbnik podatkovne baze (DBA – Database Administrator)
  - Analitik, načrtovalec PB
  - Razvijalci aplikacij
  - Končni uporabniki (izkušeni, neizkušeni)

Sistemski katalog (podatkovni slovar) - vsebuje opise podatkovnih struktur in omogoča fizično podatkovno neodvisnost:
– podatki o podatkih so shranjeni centralno
– centralno definiran pomen podatkov omogoča uporabnikom lažjo uporabo in interpretacijo podatkov
– redundanco je lažje odkriti, ker gre za centralizacijo
– spremembe, ki se izvajajo na strukturi podatkov so zabeležene
– lažja vzpostavitev sistema varnosti in zaupnosti
– omogoča zagotavljanje integritete

Struktura SUPB:

- Query processor
  - transformira poizvedbe v skupino ukazov nižje nivojskih jezikov in jih posreduje Database Managerju. Vsebuje tudi Query Optimizer.
- Database manager
  - prejema zahteve od Query processorja (poizvedbe uporabnikov in aplikacij) ter zahteve, ki so posledica ostalih ukazov
  - preverja zunanje in konceptualne sheme, da ugotovi, kateri zapisi konceptualnega nivoja "ustrezajo" zahtevi
  - preverja avtorizacijo
  - posreduje zahteve naprej File Managerju
- File manager
  - upravlja s podatkovnimi datotekami oz. fizičnimi enotami, kjer se nahajajo podatki. Učinkovitost njegovega delovanja povečajo sistemski vmesniki (System buffers)
- DML preprocessor
  - pretvarja DML ukaze bodisi v poizvedbe, bodisi v standardne funkcijske klice, ki se nahajajo prevedeni v Object code
- DDL compiler
  - na podlagi prejetega DDL ukaza izdela vse ukaze za poseg v eno ali več tabel v Data Dictionary in jih posreduje Dictionary Managerju
- Dictionary manager:
  - omogoča dostop do sistemskega kataloga
  - upravlja s sistemskim katalogom

Tipična zgradba SUPB - drug pogled:

- Stroj za evaluacijo poizvedb (Query Evaluation Engine)
  - Sintaktični analizator (Parser): Sintaktično analizira poizvedbo, ki jo SUPB-ju posreduje aplikacija
  - Optimizator (Optimizer): Na podlagi informacij o tem, kako so podatki shranjeni, izdela učinkovit plan za izvajanje poizvedbe. Plan izvajanja predstavlja načrt za izvedbo poizvedbe in je po navadi predstavljen kot drevo relacijskih operatorjev
  - Evaluator operatorjev (Operator Evaluator): Na osnovi plana izvajanja analizira poizvedbo in ugotovi, če lahko operatorje nadomesti z bolj učinkovitimi operatorji
  - Izvajalec plana (Plan Executor): Izvede poizvedbo po navodilih plana poizvedbe
- Datoteke in metode dostopa (Files and Access Methods): enota, ki omogoča delo z datotekami
- Upravljalec pomnilnika (Buffer Manager): Prenaša strani iz diska v pomnilnik glede na bralne potrebe
- Upravljalec prostora na disku (Disk Space Manager): Najnižji nivo SUPB je zadolžen za upravljanje z diskom. Vse operacije višjih plasti se tukaj prevedejo v nizko-nivojske ukaze za delo z diskom diskom
- Enota za nadzor sočasnosti (Concurrency Control):
  - Upravljalec transakcij (Transaction Manager): Zagotavlja zaseganje podatkov z uporabo določenih protokolov in skrbi za razporejanje izvajanja transakcij
  - Upravljalec zaklepanja (Lock Manager): Vzdržuje informacije o zahtevanih in odobrenih zaseženjih podatkov
- Upravljalec reševanja podatkov (Recovery Manager): Vzdržuje dnevnik in skrbi za obnavljanje sistema v zadnje skladno stanje pred nesrečo

Arhitekture (uporabe SUPB):

- Terminalski dostop
- Client-Server oz. Odjemalec-strežnik
  - odjemalec je zadolžen za izvajanje aplikacije: uporabniški vmesnik in poslovna logika
  - na strežniku se nahaja SUPB ter ena ali več (instanc) podatkovnih baz
  - problemi:
    - zahteva relativno zmogljive odjemalce
    - zahteva relativno velik napor za administracijo odjemalca
- Tri-nivojska arhitektura
  - zahteva manj zmogljive odjemalce
  - centralizirana administracija aplikacije
  - ustreza konceptu svetovnega spleta

6. Diski in datoteke

Hierarhija pomnilnika:
- dostop do primarnega pomnilnika je zelo hiter
- stroški so za določeno količino glavnega pomnilnika so približno 100 krat večji kot stroški za enako količino sekundarnega pomnilnika (disk)
- počasnejše enote sekundarnega pomnilnika igrajo pomembno vlogo, saj je podatkov običajno zelo veliko
- obstajajo tudi drugi razlogi za shranjevanje podatkov na sekundarnem pomnilniku:
  - 32 bitni naslovni prostor omogoča naslavljanje samo 4Gb podatkov....
  - podatki morajo biti obstojni, v primarnem pomnilniku podatki običajno niso obstojni.

Magnetni disk:
- omogoča neposreden dostop do želene lokacije na njem
- podatki na disku so shranjeni v enotah, ki se imenujejo BLOK-i
- BLOK predstavlja zaporedje nizov (byte) in je najmanjša enota, ki se jo lahko bere iz ali piše na disk
- BLOKI so organizirani v koncentrične kroge – SLEDI (track)
- SLEDI se nahajajo na eni ali obeh straneh magnetne plošče
- množica vseh SLEDI, ki so enako oddaljene od središča, se imenuje CILINDER
- vsaka SLED je razdeljena na ODSEKE (sektorje) .Velikost ODSEKA je določena z diskom in je ni mogoče spreminjati
- velikost BLOKA se določi pri formatiranju diska. Njegova velikost je mnogokratnik velikosti ODSEKA
- čas, ki je potreben za dostop do določene lokacije na disku (dostopni čas) je sestavljen iz več komponent:
  - iskalni čas (premik glave na ustrezno sled)
  - rotacijska zakasnitev (čakalni čas, da se ustrezen blok na sledi zavrti do glave). Povprečno znaša polovico časa rotacije in je manjši od iskalnega časa.
  - čas prenosa (dejanski prenos bloka – branje ali pisanje).

Vpliv strukture diska na performanse:
- podatki se morajo pred uporabo prenesti v glavni pomnilnik,
- najmanjša enota podatkov, ki se bere ali piše na disk je blok. Če se potrebuje samo en zapis iz bloka, se prenese celotni blok
- čas za pisanje ali branje podatkov variira in odvisno od položaja podatkov na disku (dostopni čas = iskalni čas + rotacijska zakasnitev + čas prenosa)
- čas, ki ga SUPB porabi za obdelavo podatkov je močno odvisen od lokacije podatkov na disku oz. razpršenosti podatkov po disku. Čas, potreben za premikanje blokov iz diska oz. na disk je tako po navadi večji kot čas potreben za obdelavo določenega podatka

RAID – Uvod:
- disk predstavlja potencialno ozko grlo za performanse in vpliva na zanesljivost delovanja sistema
- diski vsebujejo mehanske elemente, zaradi česar je verjetnost za napake večja kot pri notranjem pomnilniku
- če disk odpove, potem to v kontekstu podatkovnih baz pomeni izgubo podatkov zaradi podatkovne nesreče

REŠITEV: polje diskov (*disk array*), s katerim povečamo tako performanse kot zanesljivost delovanja

RAID (diskovna polja) – Razstavljanje podatkov:
- povečajo se performanse
- RAID se uporabniku kaže kot zelo velik disk
- podatki se razdelijo na enake particije (striping unit), ki se distribuirajo na več diskov
- enote se po diskih distribuirajo po “round robin” algoritmu. Dve logično sosedni particiji tako nista na istem disku
- če polje vključuje D diskov, se particija i zapiše na “i mod D” disk

RAID – Stopnje redundance:
- zanesljivost diskovnega polja se lahko poveča z redundantnimi podatki - povečamo MTTF (mean-time-to-failure)
- velik MTTF pomeni manjšo verjetnost za napako
- RAID 0: Nonredundant (data striping)
  - uporablja data striping za povečanje pasovne širine
  - ne vzdržuje nobene redundantne informacije
  - PROBLEM: MTTF pada linearno s številom diskov v polju
  - PREDNOSTI: najboljše performanse pisanja na disk
- RAID 1: Mirrored
  - najdražja rešitev za polje diskov, ker se vzdržujeta dve kopiji podatkov na dveh diskih

- vsako pisanje bloka na disk vključuje pisanje na dva diska
- pisanje se ne izvede hkrati, ampak eno za drugo (zaradi primera nesreče med pisanjem)
- branje lahko vključuje paralelno branje dveh različnih blokov iz dveh diskov. Branje se lahko dodeli na disk, ki ima najmanjši dostopni čas

Upravljanje prostora na disku:

- velikost strani je enaka velikosti bloka na disku, tako da se strani shranjujejo kot bloki na disku. Branje ali pisanje strani se tako lahko izvede v okviru ene I/O operacije
- Disk Space Manager
    - mora skriti podrobnosti strojne opreme (in tudi operacijskega sistema) in omogočiti višjim plastem programske opreme (SUPB), da obravnava podatke kot zbirko strani
    - mora vzdrževati stanje zasedenih in prostih blokov na disku (2 načina):
    - lahko uporablja datoteke operacijskega sistema. Posledično se celotna PB se nahaja v eni ali več datotekah - v tem primeru je zadolžen za upravljanje prostora v teh datotekah

Buffer Manager:

- je programska plast, ki skrbi za prenašanje ustreznih strani v pomnilnik
- upravlja z razpoložljivim pomnilnikom (buffer pool)
- zagotavlja višjim plastem SUPB-ja strani, ki jih te rabijo za svoje delo

Strategija zamenjevanja strani v buffer pool (buffer replacement policy). Najbolj uporabljena strategija je LRU – least recently used.

Buffer Manager : Predpomnilnik OS:

- obstaja podobnost med navideznim pomnilnikom operacijskega sistema in upravljanjem s pomnilnikom pri SUPB
- cilj obeh je zagotoviti dostop do več podatkov, kot pa jih lahko spravimo v pomnilnik

Zakaj ne uporabimo navideznega pomnilnika OS za potrebe SUPB?

- SUPB lahko bolj natančno predvidi zaporedje, v katerem se bo dostopalo do strani, kot pa tipičen OS
- SUPB rabi več nadzora nad stranmi, ki se zapisujejo na disk, kot pa ga omogoča tipičen OS
- buffer manager uporablja strategijo “prefetching of pages”, ki omogoča predvidevanje več naslednjih zahtev in v skladu s tem se ustrezne strani prenesejo v pomnilnik, preden so dejansko zahtevane

7. Relacijski model

Model PB, ki temelji na relacijskem modelu, je predstavljena z množico relacij, kjer je vsaka relacija tabela z vrsticami in stolpci.

Terminologija pri relacijskem modelu:
– Relacija
– Atribut
– Domena
– n-terica
– Stopnja relacija
– Številnost relacije
– Relacijska shema
– Relacijska PB

Relacija:
– predstavljamo si jo kot dvodimenzionalno tabelo s stolpci in vrsticami
– velja za logično strukturo podatkovne baze in ne za fizično.
– vrstica relacije ima nek pomen in lahko predstavlja objekt, osebo, dogodek, pravilo
– pripada ji natanko ena relacijska shema

Atribut:
– je poimenovani stolpec relacije
– predstavlja lastnost tega (objekt, osebo, dogodek, pravilo), kar predstavlja relacijo
– določimo mu podatkovni tip in dolžino

Domena določa poleg podatkovnega tipa in dolžine tudi množico dovoljenih vrednosti enega ali več atributov, ki so vključeni v to domeno.

N-terica je ena vrstica v relaciji.
Številnost relacije je število n-teric relacije.
Stopnja relacije je število atributov v relaciji.

Relacijska shema:
– vsaki relaciji pripada relacijska shema.
– je sestavljena iz oznake sheme R ter liste oznak atributov Ai s pripadajočimi oznakami domen Di
– predstavlja semantiko ali pomen relacije.
– je del konceptualnih shem. Razlagajo pomen relacij.

Lastnosti relacij:
– ime relacije je enolično
– vsaka celica tabele (polje), ki predstavlja relacijo, vsebuje natančno eno atomarno vrednost
– vsak atribut relacije ima enolično ime
– vrednosti nekega atributa so vse iz iste domene
– vsaka n-terica relacije je enolična
– vrstni red atributov v relaciji je nepomemben
– vrstni red n-teric v relaciji je nepomemben

Poznamo več vrst odvisnosti:
– funkcionalne odvisnosti (functional dependency)
– večvrednostne odvisnosti (multivalued dependency)
– stične odvisnosti (join dependency)

Poznamo več vrst ključev:
– kandidat za ključ
je vsaka podmnožica atributov relacije, ki relacijo enolično določa
– primarni ključ (primary key)
je tisti kandidat za ključ, ki ga (med vsemi kandidati za ključi) izberemo za shranjevanje relacij v fizični podatkovni bazi
– tuji ključ (foreign key)
je množica atributov, v okviru ene relacije, ki je enaka primarnemu ključu neke druge ali iste relacije

Vrste omejitev:
- omejitve domene (Domain constraints)
- pravila oz. omejitve za celovitost podatkov (Integrity constraints):
  - omejitve entitet (Entity Integrity)
  - omejitve povezav (Referential Integrity)
- Splošne omejitve (General constraints)

Oznaka "Null":
- predstavlja vrednost atributa, ki je trenutno neznana ali irelevantna za n-terico
- gre za nepopolne podatke ali podatke pri izjemnih primerih
- predstavlja odsotnost podatka

Omejitev entitete:
- v osnovni relaciji ne sme biti noben atribut, ki je del ključa, enak Null
- primarni ključ kot integritetna omejitev
- ostale omejitve vezane na kombinacijo vrednosti posameznih atributov v n-terici

Omejitve povezav:
- če v relaciji obstajajo tuji ključi, potem morajo:
  - njihove vrednosti ustrezati tistim, so v obliki ključa zapisane v eni izmed n-teric neke druge ali iste relacije
  - ali pa mora biti tuji ključ v celoti enak Null.
- tuji ključ kot integritetna omejitev

Splošne omejitve:
- dodatna pravila, ki jih določi uporabnik ali skrbnik podatkovne baze, ki definirajo ali omejujejo nek vidik poslovne domene, za katero je narejena podatkovna baza

Pogledi (views):
- osnovna relacija (base relation)
  - poimenovana relacija, ki ustreza nekemu entitetnemu tipu v konceptualnem modelu, katere n-terice so fizično shranjene v podatkovni bazi.
  - predstavlja konceptualno shemo
- pogled (view)
  - rezultat ene ali več operacij nad osnovnimi relacijami z namenom pridobitve nove relacije
  - predstavlja zunanjo shemo
  - je navidezna relacija, ki ne obstaja v relacijski bazi, temveč se dinamično kreira takrat, ko nekdo po njej povprašuje
  - vsebina pogleda je definirana kot poizvedba nad eno ali več osnovnimi relacijami.
  - pogledi so dinamični (spremembe nad osnovnimi relacijami katerih atributi so zajeti tudi v pogledu, so v pogledu takoj vidne)

Namen uporabe pogledov:
- predstavljajo odličen mehanizem za zagotavljanje varnosti (skrivajo posamezne dele konceptualne sheme pred določenimi uporabniki)
- uporabnikom dajejo možnost, da do podatkov dostopajo na prilagojen način
- poenostavljajo kompleksne operacije nad osnovnimi relacijami
- omogočajo zunanjo shemo (nivoji abstrakcije)

Spreminjanje vsebine pogledov:
- vse spremembe nad osnovnimi relacijami morajo biti takoj vidne tudi v pogledih nad temi relacijami
- če spremenimo podatke v pogledu, se morajo spremembe poznati tudi v osnovnih relacijah, na katere se te spremembe nanašajo
- v pogledih niso možne vse spremembe. Veljajo naslednje omejitve:
  - nad pogledom so možne spremembe, če pogled zajema eno samo osnovno relacijo ter vključuje atribute, ki so kandidat za ključ relacije
  - če pogled zajema več relacij, spremembe niso možne (izjeme)
  - če je pogled pridobljen z agregacijo ali grupiranjem n-teric, spremembe niso možne

8. Organizacija datotek in indeksi

Stroškovni model omogoča oceniti stroške (v smislu izvajalnega časa) I/O operacij nad stranmi v PB.

Primerjava datotečnih organizacij:
- neurejeno datoteko (heap) - zapisi se vrstijo naključno
  - hitro skeniranje, vstavljanje in brisanje; počasno iskanje
  - je zelo primerna za skeniranje celotne datoteke
- urejeno datoteko (sorted) - vrstni red zapisov je določen glede na iskalni ključ
  - hitro iskanje območja; srednja hitrost pri iskanju; počasno vstavljanje in brisanje
  - je najprimernejša pri uporabi iskanja s pogojem ekvivalence
- razpršeno datoteko (hashed) - zapisi so grupirani v skupine na podlagi razpršilne funkcije (hash function)
  - hitro vstavljanje in brisanje zapisov; zelo hitro je iskanje na osnovi enakosti; nobene podpore izboru območja; skeniranje celotne datoteke je nekoliko počasnejše
  - je najprimernejša pri iskanju intervalov

Indeksi – uvod:
- INDEKS je zunanja podatkovna struktura, ki pomaga najti zapise, ki ustrezajo določenemu pogoju in lahko bistveno pospeši iskanje
- vsak indeks vsebuje iskalni ključ, ki je sestavljen iz enega ali več polj zapisov, ki se nahajajo v datoteki. Iskalni ključ je lahko katerikoli podmnožica polj (atributov) zapisov v datoteki
- datoteki, ki jo indeksiramo, pravimo tudi indeksirana datoteka
- INDEKS je datoteka, katere vsebina omogoča hiter dostop do zapisov v osnovni datoteki, ki je z indeksom indeksirana

Vrste indeksov:
- gosti indeks - na vsak zapis osnovne datoteke, kaže kazalec iz indeksa
- redki indeks - kazalci iz indeksa kažejo na skupine zapisov osnovne datoteke
- primarni indeks
  - indeksiranje osnovne datoteke je izvedeno po njenem primarnemu ključu (iskalni ključ in primarni ključ osnovne datoteke sta enaka)
  - ne vsebuje duplikatov (unique index)
- sekundarni indeks
  - indeksiranje osnovne datoteke je izvedeno po podatkovnem elementu (polju), ki nastopa v zapisu osnovne datoteke, vendar pa ni primarni ključ
  - lahko vsebuje duplikate

Za dva podatkovna vpisa pravimo, da sta **duplikata**, če vsebujeta enaki vrednosti za iskalni ključ.

Enonivojsko indeksiranje:
- indeksirana je osnovna datoteka
- v indeksu poiščemo kazalec na polje ali skupino polj v osnovni datoteki, kjer nato poiščemo iskani zapis

Večnivojsko indeksiranje:
- indeksirana je osnovna datoteka
- indeksiran je indeks na osnovno datoteko, indeksiran je indeks na indeks itd.

Statično indeksiranje:
- indeks ostaja nespremenjen
- ko postane iskanje neučinkovito, se izvede reorganizacija indeksa

Dinamično indeksiranje:
- indeks se ob dodajanju in brisanju zapisov osnovne datoteke prilagaja vsebini, tako da so iskalne poti do vseh zapisov v osnovni datoteki enako dolge
- indeksa ni potrebno nikoli reorganizirati.

Indeksi s kompozitnimi iskalnimi ključi:
- indeksni ključ lahko vsebuje več polj
- če imamo indeks z indeksnim ključem, lahko izvajamo poizvedbe, pri katerih iščemo zapise z vrednostmi polj, ki so enake ključu (to imenujemo equality query)
- druga vrsta poizvedbe pa nam omogoča da je samo, ena vrednost v ključu konstantna, druge pa ne. Tako lahko pri poizvedbi dobimo več zapisov (range query)

Drevesne strukture za indekse:

- ISAM (Indexed Sequential Access Method):
  - je statičen indeks,
  - učinkovit v primerih, ko osnovna datoteka ne spreminja pogosto
  - ni učinkovit pri datotekah, ki se hitro povečujejo ali krčijo
- B+ index:
  - je dinamična struktura, ki se zelo dobro prilagaja spremembam v osnovni datoteki podatkov,
  - B+ indeks je najbolj uporabljana vrsta indeksa saj omogoča hitro iskanje elementov, iskanje intervalov in se učinkovito prilagaja spremembam v osnovni datoteki

9. Obnavljanje PB

Podatkovne nesreče lahko povzročijo nekonsistentnost podatkov med seboj ali s stvarnostjo, ali pa onemogočijo nadaljnji dostop do PB.
Postopke, s katerimi se ohranja konsistentnost PB po nesrečah, imenujemo **obnavljanje (recovery)**.

Obnavljanje (povrnitev stanja) obsega:
- pripravo podatkov za obnovitev
- detekcijo podatkovne nesreče, ki je povzročila nekonsistentno stanje ali nedostopnost podatkov
- obnovitev PB v stanje pred nesrečo
- obnavljanje mora zagotoviti, da se obnovi v eno izmed veljavnih stanj

Transakcije:
- je logični vrstni red ažuriranj in povpraševanj, ki prevede PB iz enega v novo veljavno stanje. Transakcija ohranja konsistentnost podatkov v PB.
- če hočemo ohraniti konsistentno PB, se mora transakcija izvesti v celoti, ali pa sploh ne.
- je aktivna od ukaza "Začetek transakcije" do ukaza "Commit" (preide v stanje "uspešna transakcija")

Izvajanje transakcije:
1. začetek izvajanja transakcijskega programa
2. začetek transakcije (sprejem vhodnih podatkov)
3. ažuriranje
4. zaključek transakcije (pomni/pozabi)
5. začetek transakcije

Vrste podatkovnih nesreč:
- lokalnega značaja
  - transakcijske nesreče, ki jih odkrijejo uporabniški programi
  - transakcijske nesreče, ki jih odkrije SUPB
- širšimi posledicami
  - sistemske nesreče
    - za sistemsko nesrečo štejemo izgubo podatkov v notranjem pomnilniku, zaradi prekinitve delovanja le-tega
    - napaka je lahko povzročena s prekinitvijo napajanja ali napako pri branju ukaza ali podatka iz notranjega pomnilnika
    - sistemska nesreča povzroči prekinitev izvajanja trenutno aktivnih transakcij
    - pred nadaljnjo uporabo PB po sistemski nesreči, je potrebno obnoviti PB v zadnje veljavno stanje pred nesrečo in ponoviti izvajanje prekinjenih transakcij
  - diskovne nesreče
    - razlogi za okvaro:
      - okvara diskovne površine
      - okvara bralno pisalnih glav
      - okvara krmilnika diska itd.
    - tudi pri teh nesrečah so lahko nekatere transakcije med svojim izvajanjem prekinjene
    - ko PB spet začne delovati, je potrebno obnoviti PB v zadnje veljavno stanje pred nesrečo in ponoviti izvajanje prekinjenih transakcij

Vrste obnavljanja PB – Uvod:
- obnavljanje temelji na redundantnih podatkih
- redundantni podatki se generirajo občasno ali sočasno z izvajanjem operacij v okviru transakcij

Obnavljanja se razlikujejo glede na:
- hitrost obnovitve PB
- pogostost in količino beleženja redundantnih podatkov
- poslabšanja odzivnih časov, zaradi beleženja redundantnih podatkov

Dvojna podatkovna baza:
- najenostavnejši primer: diskovni krmilnik zapisuje sočasno podatke na 2 fizično ločena diska. Če se ena PB pokvari, se uporablja njena dvojnica
- za večjo zanesljivost: podvoji se tudi diskovni krmilnik ali kar cel računalniški sistem
- je relativno draga
- omogoča hitro obnavljanje PB predvsem po diskovnih in delno sistemskih podatkovnih nesrečah
- izvoz PB v datoteko
- periodično kopiranje PB na magnetni trak

Delna ali inkrementalna kopija se lahko izvaja tudi v času, ko je PB v uporabi.

Obnavljanje z dnevnikom in kopijo:
- temelji na izdelavi kopije PB in izdelavi dnevnika:
  - s kopijo lahko PB obnovimo v veljavno stanje, ko se je izdelala kopija
  - v dnevnik se zapisujejo podatki, s katerimi je možno s kopijo obnovljeno PB obnoviti v zadnje veljavno stanje tik pred nesrečo (vsebuje tudi podatke, s katerimi je možno ponovno izvesti tudi transakcije, ki so bile zaradi nesreče prekinjene)

2 vrsti obnavljanja PB z dnevnikom glede na čas izvedbe ažuriranja PB:
- odloženo ažuriranje
  - vsa ažuriranja v okviru transakcije se najprej shranijo v dnevnik. Pri uspešnem zaključku transakcije se izvede dejansko ažuriranje PB
  - je učinkovitejše, če se v povprečju izvede več neuspešnih transakcij
  - v dnevnik se beležijo:
    - zapisi, ki vsebujejo podatke, potrebne za izvedbo odloženega ažuriranja PB in
    - podatki za njeno morebitno obnavljanje
  -
- sprotno ažuriranje
  - vsa ažuriranja se sproti izvajajo v PB. V dnevnik se vpisujejo le podatki, ki so potrebni za morebitno obnavljanje PB
  - je učinkovitejše, če se v povprečju izvede več uspešnih transakcij

Dnevnik je zaporedna datoteka, v katerem so zapisi urejeni po času njihovega nastanka. Zapisi se v dnevnik vedno dodajajo na konec datoteke.

Obnavljanje z dnevnikom in kopijo - Razveljavitev transakcije:
- pri sprotnem ažuriranju se razveljavitev izvede tako, da se iz dnevnika bere stare zapise od zadnjega proti prvemu po padajočem času
- v primeru sistemske nesreče je potrebno razveljaviti ažuriranja večih transakcij
- v dnevniku so njihovi zapisi med seboj pomešani, vendar pa so urejeni po času, kar omogoča izvesti razveljavitev

Obnavljanje z dnevnikom in kopijo - Ponovitev ažuriranj transakcije:
- ažuriranja, ki jih izvede uspešna transakcija, je v primeru sistemskih in diskovnih podatkovnih nesreč treba ponoviti
- vzroki:
  - ker so se ažuriranja izgubila ali pa
  - ni zanesljivo, da so se vsi datotečni vmesniki z ažuriranimi zapisi tudi izpisali na disk (sistemske nesreče)
- po izvedeni ponovitvi se v PB zanesljivo nahajajo vsa ažuriranja, ki so jih izvedle uspešne transakcije

Obnavljanje z dnevnikom in kopijo – Obnavljanje po sistemskih in diskovnih nesrečah:
- po ponovnem zagonu je PB treba obnoviti, tako da najprej razdelimo transakcije na:
  - prekinjene transakcije: v dnevniku obstajajo njihovi zapisi
  - uspešne transakcije: tiste, za katere obstaja v dnevniku zapis z oznako “Pomni” in
  - neuspešne transakcije: tiste, za katere obstaja v dnevniku zapis z oznako “Pozabi”

Obnavljanje uspešnih transakcij:
- pri njih nismo prepričani, ali so se strani, ki so jih transakcije ažurirale, zanesljivo zapisale na disk
- zaradi tega je potrebno njihova ažuriranja ponoviti, tako pri uporabi sprotnega, kot odloženega ažuriranja

Obnavljanje neuspešnih transakcij:
- pri teh transakcijah pri sprotnem ažuriranju nismo prepričani, ali so se njihova ažuriranja tudi že zares razveljavila v PB
- take transakcije je zato potrebno ponovno razveljaviti

Obnavljanje prekinjenih transakcij:
- najprej izvedemo njihovo razveljavitev, nato jih vrnemo transakcijskim programom v ponovno izvajanje

Obnavljanje z dnevnikom in kopijo - Kontrolna točka:
- čas obnavljanja skrajšamo z zahtevo po izpisu vseh datotečnih vmesnikov na disk. Tako smo prepričani da so bile transakcije ki so bile zaključene pred izpisom vmesnikov, zanesljivo uveljavljene ali razveljavljene v PB na disku
- kontrolna točka se izvede v odvisnosti od števila transakcij na časovno enoto
- ob sistemski nesreči je potrebno razveljaviti oz. ponoviti le transakcije, ki so bile aktivne v času izdelave kontrolne točke, ali so se pričele izvajati kasneje

Pri kontrolni točki se izvedejo naslednje operacije:
1. prekine se izvajanje novih ukazov transakcijskih program., dokončajo se vse razveljavitve in uveljavitve transakcij,
2. izvede se izsiljeni izpis vseh datotečnih vmesnikov dnevnika na disk,
3. izvede se izsiljeni izpis vseh datotečnih vmesnikov PB na disk,
4. v dnevnik se doda zapis “Kontrolna točka” in izvede izsiljen izpis datotečnega vmesnika dnevnika na disk,
5. v startno datoteko PB se zapiše naslov zapisa “Kontrolna točka” v dnevniku,
6. nadaljuje se izvajanje ukazov transakcijskih programov.

Obnavljanje z dnevnikom in kopijo – Dnevnik:
- dnevnik se uporablja tudi za tekoče razveljavljanje (sprotno ažuriranje) oz. uveljavljanje (odloženo ažuriranje). To mora biti izvedeno v čim krajšem času
- najpogostejša varianta realizacije dnevnika:
  - dve dnevniški datoteki na disku in ena na magnetnem traku – arhivska
  - ob izvajanju transakcij se dnevniški zapisi vpisujejo v eno izmed datotek na disku (npr.: v dnevnik A)
  - ko se dnevnik A napolni približno 90%, se izvrši preklop vpisovanja
  - vsi dnevniški vpisi, ki se pričnejo po preklopu, se vpisujejo v dnevnik B, zapisi transakcij, ki so se začele pred preklopom, pa se še naprej zapisujejo v dnevnik A.

Ko se zaključijo vse transakcije, ki se vpisujejo v dnevnik A, se izvede kontrolna točka. S tem so se vse transakcije v A zaključile, pred kontrolno točko.

Arhivski dnevnik vsebuje le še tiste zapise, ki bi jih utegnili rabiti za obnavljanje, kar dosti zmanjša majhna velikost arhivskega dnevnika.

10. Sočasni dostop

Transakcije, ki jih izvajajo uporabniški programi se izvajajo prepletajoče, zato tudi SUPB izvaja ukaze prepletajoče
Prepletajoče izvajanje transakcij pa skriva dve pasti:
– podatkovna baza lahko zaide v nekonsistentno stanje
– rezultati povpraševanj v podatkovni bazi so lahko napačni

Izgubljeno ažuriranje:
Na osnovi istega prebranega zapisa se izvede ažuriranje istega zapisa s strani dveh transakcij, kar pomeni, da obvelja samo zadnje ažuriranje.

Branje "neobstoječega zapisa":
Predpostavimo, da neka transakcija prebere zapis, ga je ravnokar ažurirala druga transakcija, ki se je takoj zatem ponesrečila in zato razveljavila. Prva transakcija je torej prebrala zapis (oz. njegovo stanje), ki ni nikoli veljavno obstajal v PB.

Naloga nadzora nad sočasno uporabo PB (concurrency control) je:
- ohraniti podatkovno bazo v konsistentnem stanju,
- dopustiti čimvečjo sočasnost izvajanja transakcij
- v ta namen se uporabljata dve tehniki:
  - zaseganje zapisov (locking),
  - časovno označevanje (timestamping)

Sočasno izvajanje transakcij:
– zaporedje izvajanja ukazov v okviru (več) transakcij se imenuje razpored (**schedule**)
– zaporeden razpored: pri sočasnem izvajanju transakcij se najprej izvedejo vsi ukazi ene transakcije, nato pa vsi ukazi druge transakcije
– transakcije so med seboj neodvisne, če se lahko izvajajo v poljubnem vrstnem redu: zaporedno ali prepletajoče, pri čemer se kakšne izmed transakcij lahko zaključijo tudi neuspešno

Velja naslednje:
Če vsaka izmed transakcij ohranja konsistentnost podatkovne baze, potem jo ohranja tudi vsak njihov zaporedni razpored.

Zaporedniški razpored:
Učinkuje na PB enako kot kak izmed zaporednih razporedov.

Zaseganje zapisov:
• Možna rešitev predstavljenih problemov: možnost, da si transakcijski program pridobi izključno pravico dostopa do zapisov, do katerih transakcija dostopa
• S tem se prepreči vmešavanje sočasnih transakcij v njen postopek ažuriranja

Piščevo pravilo:
To je pravilo po katerem se mora ravnati SUPB, da zaščiti konsistentnost PB. Ko se v okviru transakcij izvaja ažuriranje dela PB, mora biti celotno zaporedje operacij ažuriranja zaščiteno pred vmešavanjem s strani transakcij, ki žele sočasno ažurirati isti del PB.

Bralčevo pravilo:
To je pravilo po katerem se mora ravnati SUPB, da zaščiti konsistentnost rezultatov povpraševanja. Ko se v okviru transakcije izvaja le povpraševanje v PB, potem je lahko (ali pa tudi ne) celotno zaporedje operacij povpraševanja zaščiteno pred mešavanjem s strani transakcij, ki bi žele ažurirati isti del PB.

2 pravili glede zaseganja podatkov:
– ekskluzivno zaseženi podatek se ne more še dodatno zaseči, niti ekskluzivno, niti delno
– deljeno zaseženi podatek se lahko dodatno deljeno zaseže, ne more pa se zaseči ekskluzivno

Zaseganje zapisov - PXC, PSC:
- protokol zagotavlja, da se prepletajoče izvajanje transakcij izvaja po enem izmed zaporedniških razporedov
- poznamo 2 protokola:
  - PXC (Protocol eXclusive Commit): temelji na ekskluzivnem zaseganju podatkov
  - PSC (Protocol Shared Commit): temelji na ekskluzivnem in deljenem zaseganju

Pravila, ki jih uvaja PXC protokol:
– transakcija, ki želi podatek ažurirati, ga mora najprej ekskluzivno zaseči
– če zahteva po zaseženju ne more biti takoj odobrena, preide transakcija v stanje čakanja, njeno izvajanje se nadaljuje po odobritvi zaseženja
– vsa zaseženja se smejo sprostiti šele po zaključku transakcije (uspešnem ali neuspešnem), transakcija ki želi le prebrati podatek in ji ni mar za sočasno
– transakcija, ažuriranje tega podatka s strani kake druge transakcije, ga sme prebrati ne glede na to, ali je podatek zasežen ali ne

Dodatno vsebuje protokol PSC še naslednje pravilo:
– transakcija, ki želi ekskluzivno zaseči podatek, mora imeti pred tem odobreno njegovo deljeno zaseženje

Razlike med protokoloma so v obsegu dopuščenega sočasnega izvajanja transakcij:
– PSC dopušča sočasno izvajanje dveh zgolj povpraševalnih transak., ki zahtevata zaščito pred ažurir.; PXC pa ne
– PSC veliko bolj dopušča nastop mrtve zanke (medsebojno blokiranje transakcij)

Da se protokol za zaseganje podatkov lahko izvaja, mora SUPB za podatke voditi evidenco o tem:
– ali podatek je ali ni zasežen
– kako je zasežen
– kdo vse ga je zasegel
– kdo vse ga želi zaseči in na kakšen način

Zato se za vsak podatek vzdržujeta 2 listi:
– lista odobrenih zaseženj ki vsebuje pare: (Oznaka transakcije Vrsta zaseženj, transakcije, Odobrenega Zaseženja)
– lista zahtevanih zaseženj, ki vsebuje pare: (Oznaka transakcije, Vrsta Zahtevanega Zaseženja)

Do sprememb na listah prihaja:
– pri sprejetju zahteve po zaseženju
– pri odobritvi zaseženja
– ob zaključku transkacije

Zaseganje zapisov – Objekti zaseženja:
- Objekti zaseženja so lahko različni:
  - logični
    - relacije
    - n-terice v relacijah,...
  - fizični
    - celotna fizična podatkovna baza
    - tabele
    - fizični bloki oz. strani
    - fizični zapisi v tabeli

Granulacija objektov zaseženja vpliva na:
– obseg sočasnoti pri izvajanju transakcij
– obseg podatkov o odobrenih in zahtevanih zaseženjih
– stopnjo dodatne obremenitve SUPB z izvajanjem nadzora nad zaseženji

Mrtva zanka – Uvod:

Livelock:
Posamezno transakcijo lahko pri odobritvi zaseženja določenega podatka prehitevajo vse ostale transakcije in se s tem njeno čakanje raztegne na nedoločen čas (problem je rešljiv z algoritmi za dodeljevanje zaseženj).

Mrtva zanka (deadlock):
Ko dve (ali več transakcij) zaseže vsaka svoj podatek, vsaka od njiju pa želi še podatek, ki ga je že zasegla tekmica. Zaradi tega transakciji čakata druga na drugo, in ker se po protokolu sprostijo zaseženi podatki le ob zaključku transakcije, ni čakanja nikoli konec.

Problem mrtve zanke se razreši na dva načina:
– preprečimo, da se mrtva zanka sploh lahko pojavi, kar rešujemo z enim ali več od naslednjih prijemov
  – urnikom izvajanja transakcij
  – vnaprejšnja zahteva po zaseženjih zapisov
  – ureditev objektov zaseganja, odloča transakcijski program in prekinitev in ponovno izvajanje transakcij
– odpravimo mrtvo zanko potem, ko je ta že nastopila

Preprečevanje nastopa mrtve zanke:

- urnik izvajanja transakcij
  - z izvajanjem transakcij po urniku ne dopustimo sočasnega izvajanja takih transakcij oziroma transakcijskih programov, ki bi utegnili imeti konfliktne podatkovne zahteve
  - v tem primeru tudi ni potrebno izvajati zaseganja podatkov. Pravimo, da se na ta način poveča "propustnost" podatkovne baze
  - ker praviloma ni vnaprej znano katere podatke bo transakcija ažurirala, se možna sočasnost (prepletajoči način) izvajanja transakcij zelo omeji
  - v najneugodnejšem primeru je urnik transakcij ekvivalenten zaseženju celotne podatkovne baze s strani posamezne transakcije
- vnaprejšnja zahteva po zaseženju podatkov
  - transakcija postavi pred svojim prvim ažuriranjem zahtevo po zaseženjih vseh podatkov, ki jih namerava ažurirati:
    - če je možno vsa zahtevana zaseženja takoj odobriti, jih tudi SUPB odobri in transakcija se lahko nemoteno izvaja vse do svojega zaključka
    - če kakšnega izmed zaseženj ni možno odobriti, se ji ne odobri nobenega zaseženja in transakcija preide v stanje čakanja
  - pri taki rešitvi ne sme biti izbor podatka, ki ga bo transakcija ažurirala, pogojen z uspehom ali neuspehom kakšnega predhodnega ažuriranja, zato ker v takem primeru transakcija:
    - bodisi ne more vedeti, katere podatke naj zaseže,
    - bodisi zaseže preventivno bistveno večje število podatkov, kot bi bilo potrebno, zato se zmanjšuje možnost sočasnega izvajanja drugih transakcij. (primer na naslednji strani)

Ureditev objektov zaseganja:

- objekte zaseganja, ki so lahko zapisi, strani zapisov, n-terice, tabele, se sme zaseči samo po določenem vrstnem redu, ki mora biti znan vnaprej vsem transakcijskim programom
- če je objektov zaseganja veliko (npr. zapisi), je določanje njihovega vrstnega reda in tudi nadzor nad pravilnim vrstnim redom zaseganja dokaj zamudno opravilo
- poleg tega pa je potrebno vnaprej poznati podatke, ki se bodo v okviru transakcije ažurirali

Odloča transakcijski program:

- če zahteva po odobritvi zaseženja podatka v okviru transakcije ni takoj odobrena, odloči transakcijski program, kako naprej:
  - izvajanje transakcije lahko program takoj prekine in ko jo SUPB razveljavi, prične z njenim ponovnim izvajanjem; lahko pa jo uvrsti na začetek čakalne vrste in prične izvajati transakcijo, ki je naslednja na vrsti;
  - v presledkih lahko izvede nekaj poskusov zaseganja podatka in če ne uspe, prekine izvajanje transakcije

Prekinitev in ponovno izvajanje transakcij:

- Uporaba protokolov:
  - čakaj ali izdihni (Wait Die) in
  - rani ali čakaj (Wound-Wait)

Odpravljanje mrtve zanke:

- mrtvo zanko je potrebno najprej odkriti, nato pa pristopimo k njenemu razreševanju
- v ta namen lahko uporabimo čakalni graf
  - čakalni graf je usmerjni graf G = < V, P >
    - “V” predstavlja množico vozlišč, v kateri vsak predstavlja po eno aktivno transakcijo Ti
    - “P” predstavlja množico usmerjenih povezav (Ti, Tj), ki predstavljajo čakanje transakcije Ti na odobritev zaseženja podatka, ki ga ima zaseženega transakcija Tj
    - mrtva zanka se v grafu kaže kot cikel. V njem se lahko nahajata dve ali več transakcij, druga drugo čakajo na sprostitev zaseženj.
- odpravi jo tako, da eno izmed transakcij v mrtvi zanki izbere za žrtev in prekine njeno izvajanje

SUPB izbere žrtev preko enega od naslednjih kriterijev:

- čas izvajanja transakcij ("starost") do nastopa mrtve zanke
- predvideni čas izvajanja transakcij do njihovega zaključka
- število odobrenih zaseženj
- število izvedenih ažuriranj

Ne glede na izbrani kriterij je potrebno poskrbeti, da ne bo vedno znova razveljavljena ena in ista transakcija.

Časovno označevanje:

- postopek za nadzor nad sočasnim izvajanjem transakcij, ki ne temelji na zaseganju, zato predstavlja manjšo dodatno obremenitev SUPB
- postopek je uporaben tudi pri porazdeljenih PB, kjer ne obstaja centraliziranega nadzora nad izvajanjem transakcij
- podatki se ne zasegajo, zato tudi ne more priti do mrtve zanke
- transakcije se izvajajo tako, da je njihov zaporedniški razpored ekvivalenten takemu zaporednemu razporedu, po katerem se najprej izvede starejša transakcija in nato mlajša
- konfliktne oziroma potencialno konfliktne zahteve po branju ali ažuriranju podatkov se razrešujejo s prekinitvijo in ponovnim izvajanjem transakcije