

PREDMET

ADMINISTRACIJA BAZA PODATAKA

Izvođači nastave

Naziv predmeta: ADMINISTRACIJA BAZA PODATAKA

Semestar: VI

Broj časova: 45 časova predavanja i 45 časova računarskih vežbi

Nastavu na predmetu izvodi: Miroslav Bender; predavanja i vežbe

e-mail: bender@uns.ac.rs

Organizacija ispita

- Jeden domaca zadatka (seminarski) koji nosi 20 bodava. (**taj seminarski nije obavezan**)
- Usmeni (80 bodova)
- Završetak ispita i usmeni u junskom ispitnom roku 2023.

Cilj predmeta

- Razumevanje komponenti koje čine oblast administracije baza podataka
- Sticanje znanja i veština
 - Instaliranja SUBP
 - Upoznavanje s arhitekturom SUBP:
 - Njegovim modulima i softverskim komponentama
 - Načinom njihovog funkcionisanja
 - Konfigurisanja SUBP
 - Administriranja Sistema baza podataka

Literatura za predmet

- **Craig S. Mullins, Database Administration: The Complete Guide to DBA Practices and Procedures**, Addison-Wesley
 - Materijali sa predavanja
 - Materijali sa vežbi
 - Dokumentacija softverskih alata koji se koriste za vežbe

Platforma za održavanje nastave

- Microsoft SQL Server
- ORACLE
- PostgreSQL
- MySQL

Sadržaj kursa

- Zadaci administracije baza podataka
- Uvod u administraciju baze podataka
- Arhitektura SUBP-a – SUBP, priprema, instalacija, kreiranje
- Upravljanjeinstancama – Pokretanje baze, alati, instance
- Struktura baze podataka - Upravljanje skladištenjem u SUBP
- Sigurnost baze i podataka – Korsinci, privilegije, sigurnost
- Upravljanje objektima – Objasnjenje i korišćenje
- Backup i Oporavak – Korišćenje, alati
- Performanse – Praćenje i poboljšanje performansi, alati za praćenje

Koncepcija Baza podataka

- **Kurs je koncipiran tako da se podrazumeva poznavanje:**
 - Osnova Relacionih baza podataka i
 - Osnova SUBP-a
- **Dakle, šta je Bza podataka, a šta je Sistem za upravljanje bazom podataka (SUBP)?**
- **Koje su osnovne karakteristike SQL-a?**

Koncepcija Baza podataka - (nastavak)

- **Baza podataka**
 - je skup medjusobno povezanih podataka, uskladištenih s minimumom redudanse.
 - Može se reći i da je Baza podataka organizovano skladište podataka gde se podacima pristupa preko imenovanih elemenata (obeležja, tabela i datoteka).

Koncepcija Baza podataka - (nastavak)

- **Sistem za upravljanje Bazom podataka (SUBP)**
 - Je softver koji omogućava krajnjim korisnicima ili programerima aplikacija da dele podatke ili upravljaju njima.
 - Obezbeđuje sistematičan metod:
 - Kreiranja,
 - Ažuriranja,
 - Pretraživanja i
 - Smeštanja informacija u Bazi podataka

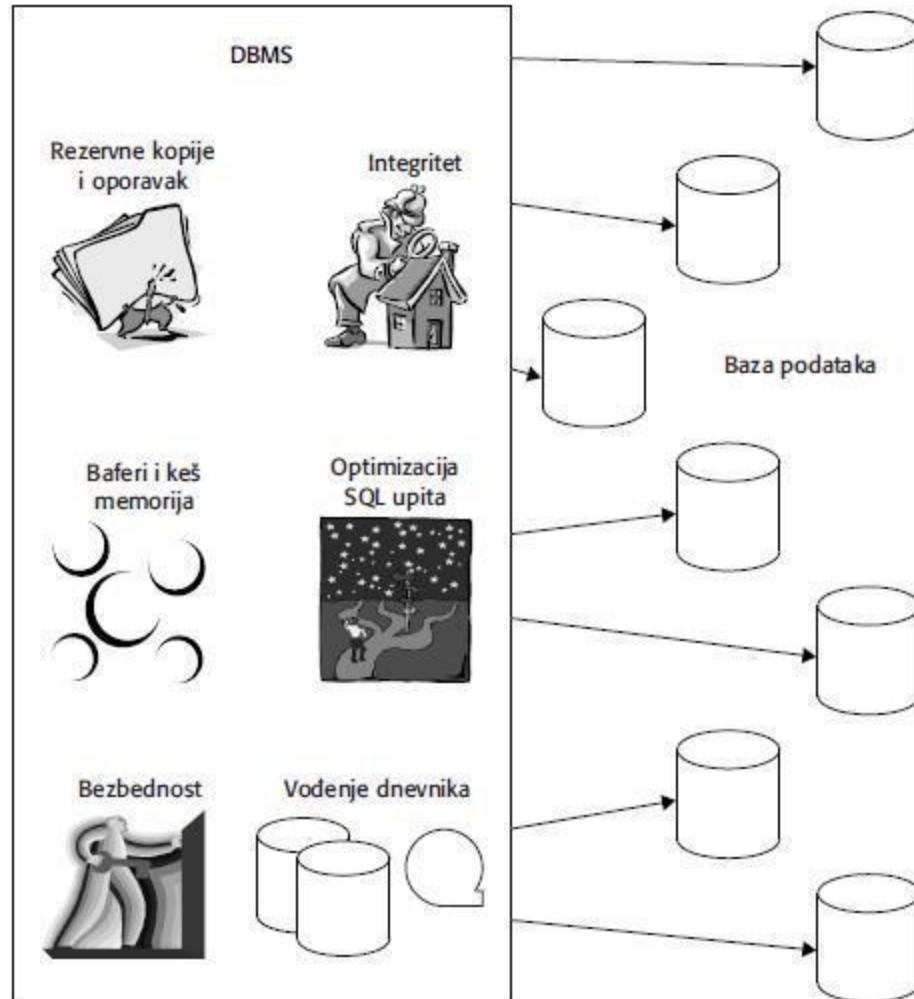
Koncepcija Baza podataka - (nastavak)

- Pored toga SUBP je generalno odgovoran za:
 - Integritet podataka,
 - Sigurnost podataka,
 - Kontrolu pristupa
 - Optimizaciju
 - Automatsko sažimanje
 - Restartovanje i oporavljanje

SQL - Neproceduralnost

- Šta znači da je SQL kao jezik neproceduralan?
 - Kakvi su to proceduralni programske jezike. Šta se njima definiše? Nabrojati neke?
 - Šta se definiše neproceduralnim programskim jezikom?
- Koje to posledice ima na arhitekturu relacionih SUBP?

Zavisnost između Baze podataka i SUBPa



Ko vodi računa o funkcionalnosti SUBPa

- ADMINISTRATOR BAZE PODATAKA
- Poslovi Administratora:
 - Učešće u izboru SUBP
 - Instalacija i konfiguracija SUBP
 - Ažuriranje SUBP
 - Uzimanje rezervnih kopija
 - Praćenje performansi i podešavanje
 - Oporavak
- Nijedna aplikacija ili baza podataka nije statična, pa

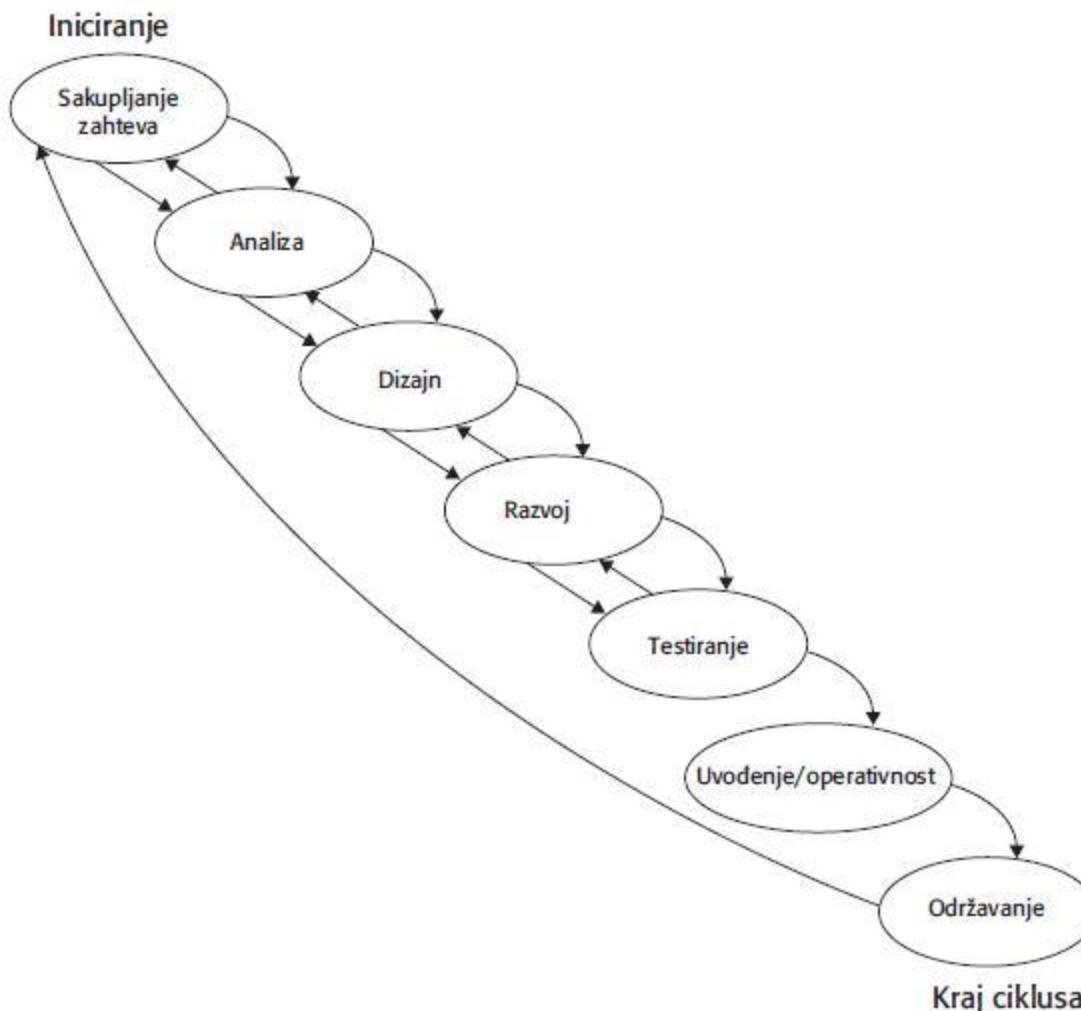
ADMINISTRATOR BAZE PODATAKA

- Zbog promene poslovnih potreba menjaju se i informacioni sistemi koji ih podržavaju
- Kada se zatraži izmena ili održavanje administrator ponovo biva angažovan u celokupnom procesu
- Administrator je odgovoran za upravljanje celokupnim okruženjem baze podataka
- To uključuje: procenu i izradu upita, postavljanje pravila i procedura da bi se upiti efikasno izvšili, kao i praćenje i optimizaciju upita

Šta je administrator Baze podataka

- Podaci su centar današnjih aplikacija
- Firme jednostavno ne mogu funkcionisati bez njih
- Što su bolji dizajn i upotrebljivost baze podataka, to će firma biti sposobnija na poslovnom polju
- Jedan od najvećih problema s kojim se susreću IT firme jeste obezbeđivanje kvalitetne administracije Baze podataka

Dobar administrator je deo celokupnog životnog ciklusa razvoja aplikacije



SLIKA 1.3 Životni ciklus razvoja aplikacija

Upravljanje administracijom baze podataka

- Na osnovu toga kako se reaguje na probleme možemo reći da postoje dva pristupa u administraciji:
 - **Reaktiv** (reaguje se kada se problemi pojave)
 - **Proaktiv** (predvideti probleme i reagovati pravovremeno)

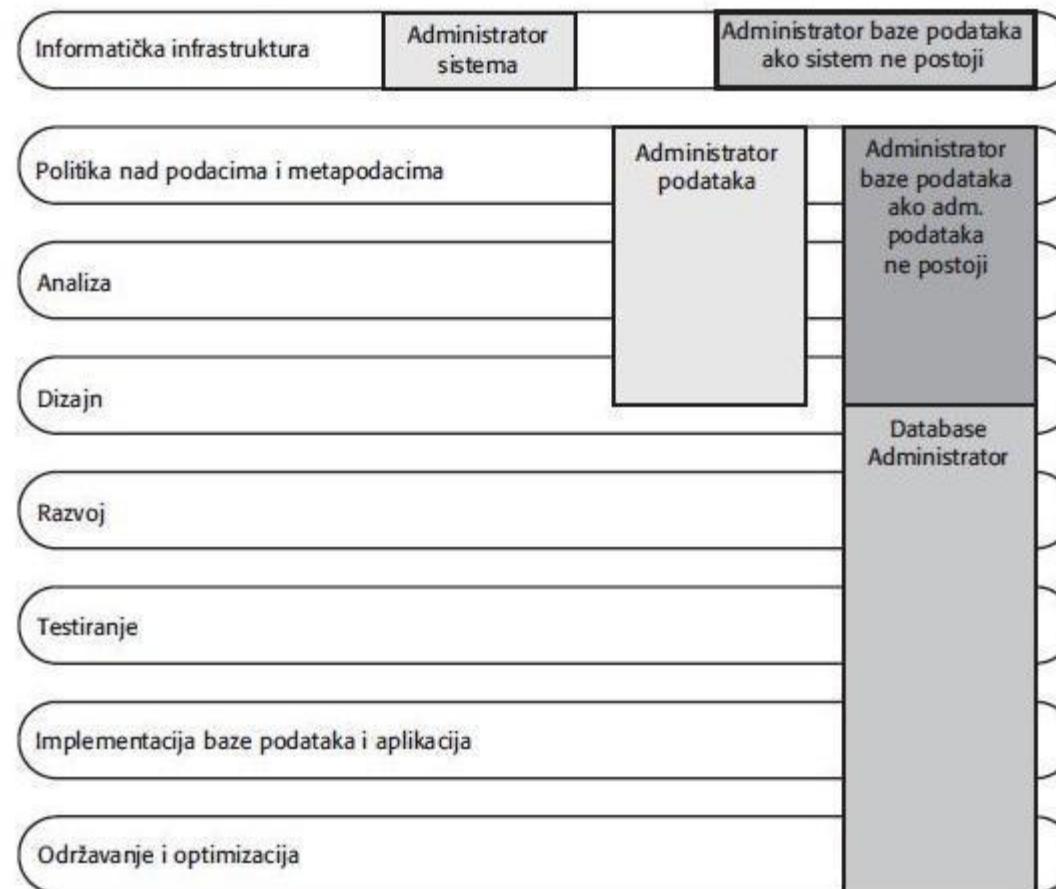
Baza podataka, podaci i administracija sistema

- Mnoge firme kombinuju ulogu administratora podataka sa administratorom baze podataka
 - Administracija podataka razdvaja poslovni aspekt upravljanja podacima od tehnologije koja se koristi da bi se podacima upravljalo
 - Administrator podataka je zadužen za razumevanje poslovnog rečnika i njegovo prevodenje u logički model podataka (prikljukane podatke, analiza, dizajn).
 - Administrator baze podataka angažovan u fazi dizajna, razvoja, testiranja, uvođenja i operativnog rada.

Baza podataka, podaci i administracija sistema

- Administracija podataka (odgovoran za konceptualni i logički model podataka)
- **Administracija baze podataka** – (transformacija logičkog modela baze u efikasan fizički model i efikasan operativan rad)
- Administracija sistema (obezbeđuje da je informatička infrastruktura operativna za razvoj baze podataka – nema direktnu odgovornost za dizajn baze i njenu podršku)
- Često je za sve tri vrste administracije odgovorna jedna osoba

Baza podataka, podaci i administracija sistema

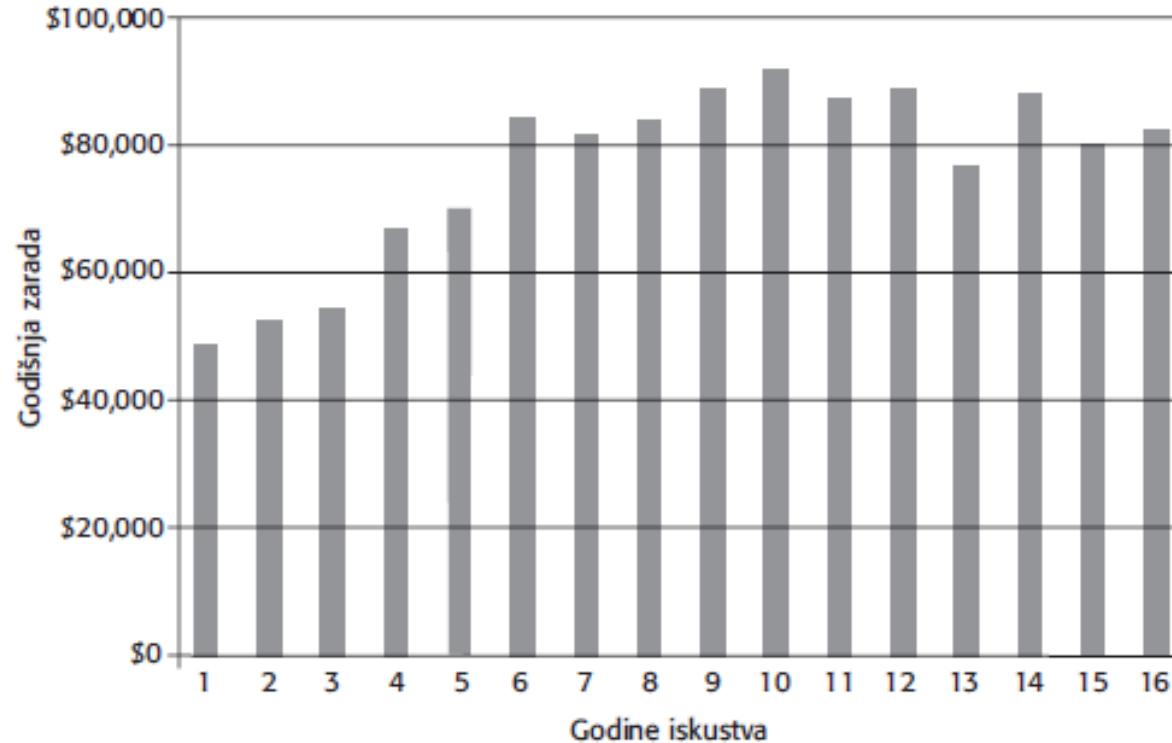


Odgovornost Administratora podataka, Administratora baze podataka i Administratora sistema

Poslovi administratora baze podataka

- Dizajn baze podataka
- Praćenje performansi i podešavanje
- Obezbeđenje raspoloživosti baze podataka
- Sigurnost baze podataka
- Uzimanje rezervnih kopija baze podataka (Backup) i oporavak baze podataka (Restore)
- Integritet podataka
- Migracija podataka na druge verzije SUBP-a
- Obezbeđuje proceduralnu funkcionalnost
 - Pogledi
 - Uskladištene procedure
 - Okidači (trigeri)
 - Korisnički definisane funkcije
- Administracija mobilnih uređaja (PDA)

Zarade Administratora baza podataka



Potrebna znanja DBA

- Poznavanje osnova fizičkog projektovanja baza podataka
- Poznavanje arhitekture Relacionih SUBP
- Poznavanje alata za administraciju konkretnog SUBP-a sa kojim se radi
 - MS SQL Server
 - Oracle
 - PostgreSQL
 - MySQL
 -

Fičko projektovanje Baze podataka

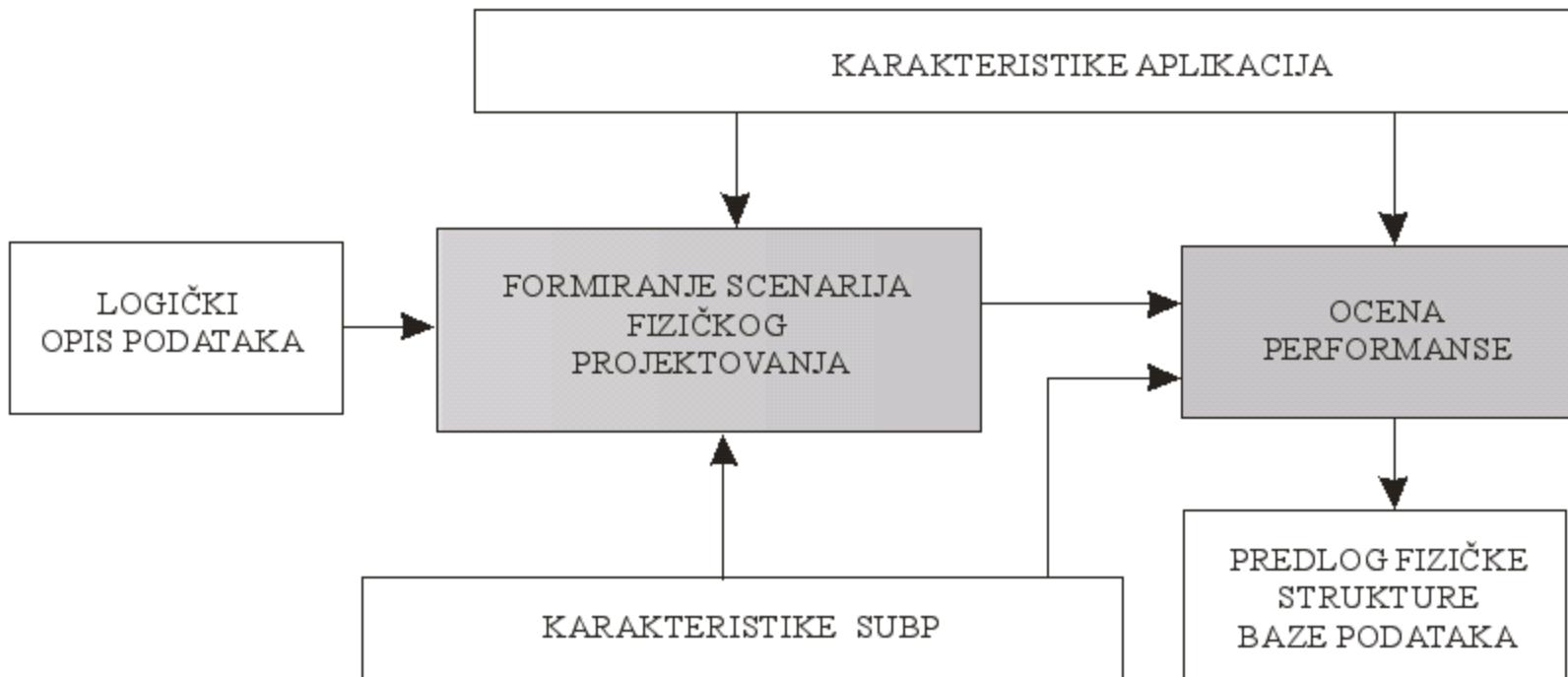
Osnovna karakteristika savremenih SUBP:

- Fizička nezavisnost podataka - razdvajanje logičkih tipova podataka i njima pridruženih operacija od fizičke reprezentacije

Posledica:

- Logički model podataka može se predstaviti sa više različitih fizičkih struktura podataka

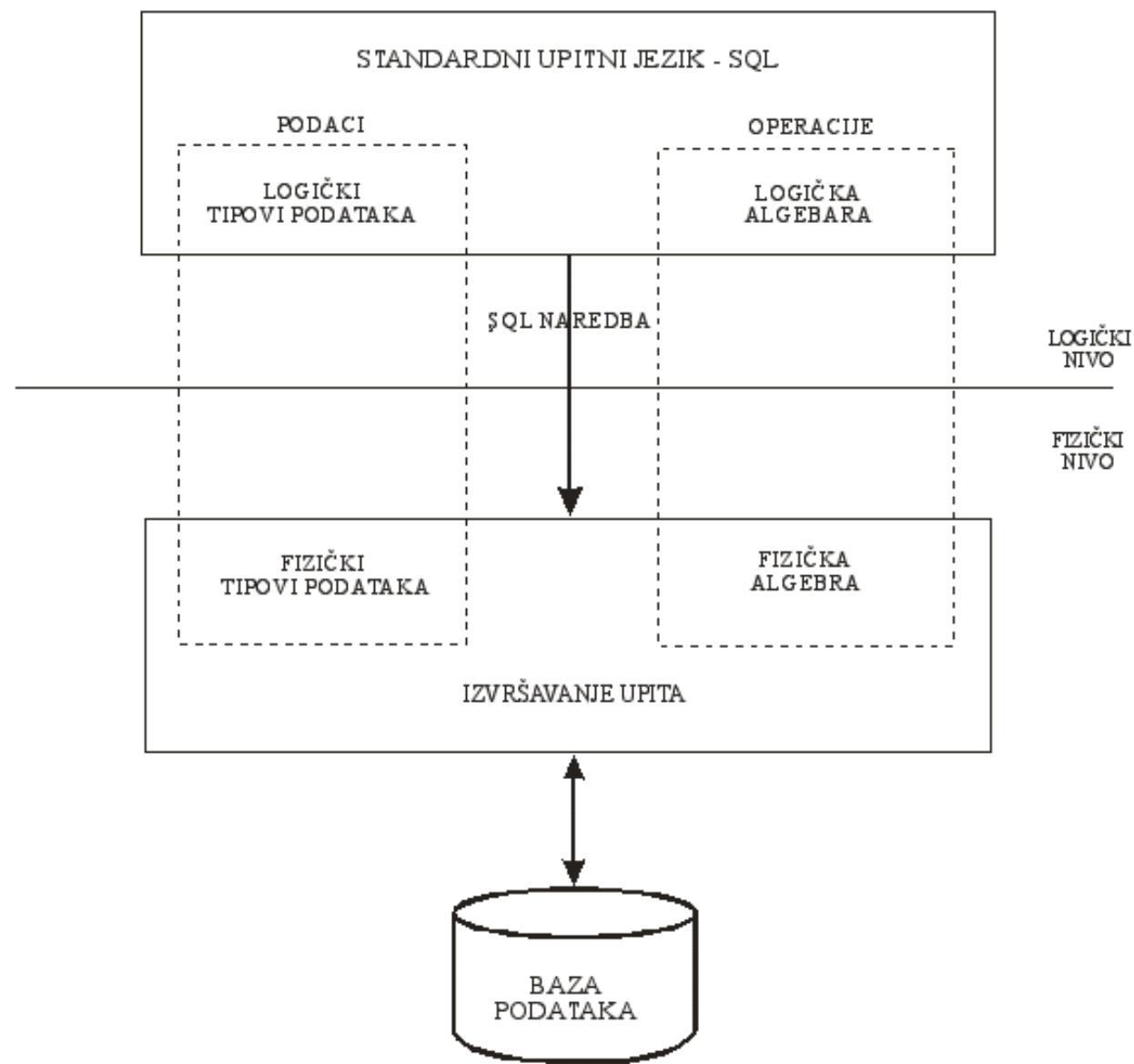
Blok šema Fizičkog projektovanja baza podataka



Karakteristike relacionih SUBP

- Neproceduralni upitni jezik
- SUBP obezbeđuje mehanizam da se dođe do traženih podataka
- Algoritmi obrade
 - implementacija relacionih operacija
 - implementacija funkcija (agregacija, grupisanja, eliminacija duplikata ...)
- Zadatak sistema je izbor optimalnog načina izvršavanja upita (za dati skup algoritama obrade i fizičku strukturu baze podataka)

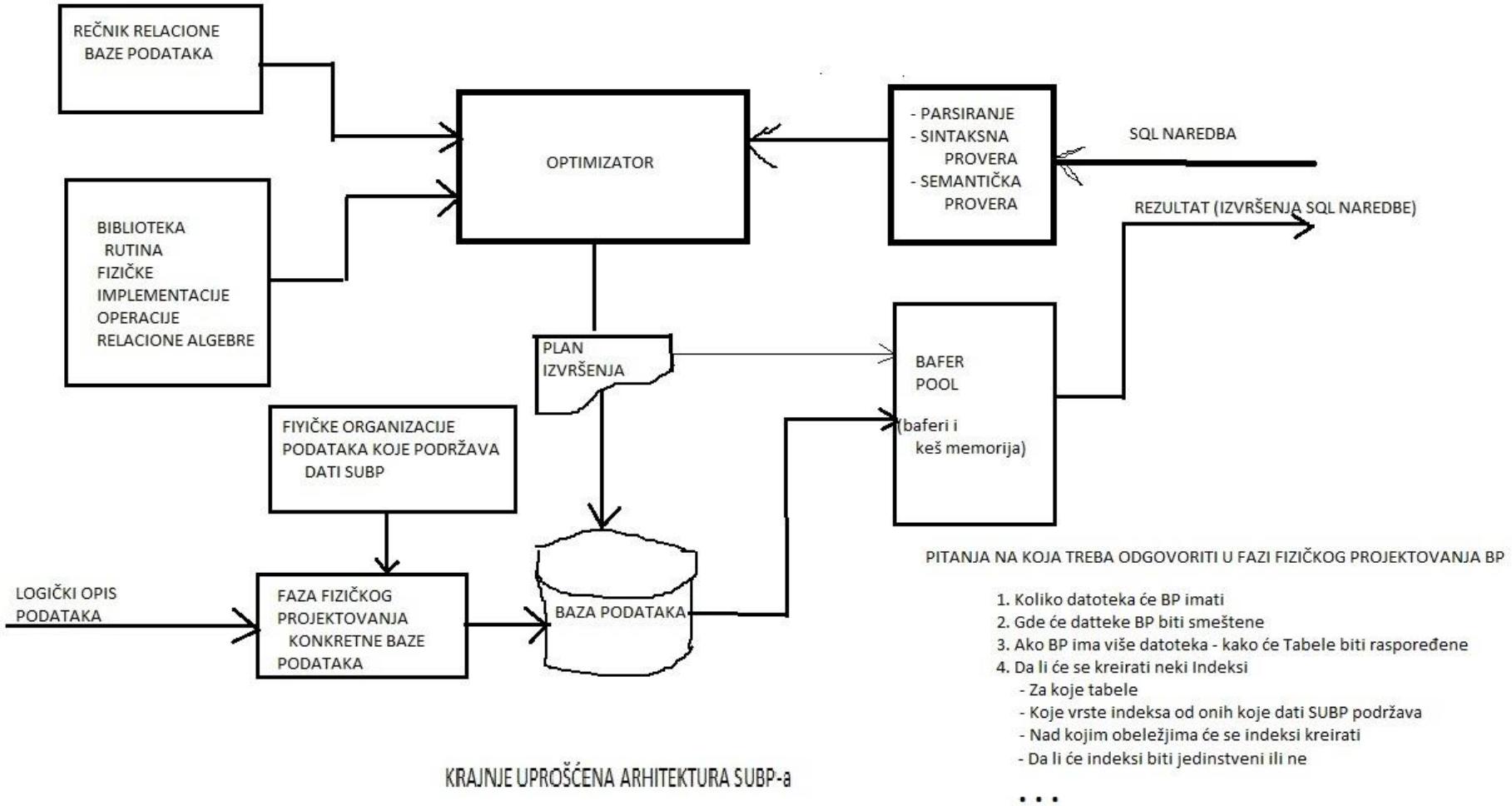
Šematski prikaz arhitekture RSUBP-a



Karakteristike relacionih SUBP bitne za fizičko projektovanje BP

- Fizičke strukture podataka koje konkretni SUBP podržava
- Način fizičke implementacije operacija relacione algebre:
 - Za svaku logičku operaciju relacione algebre u okviru SUBP ima više fizičkih algoritama procedura njihove fizičke realizacije.
- Posebno način implementacije operacije spoja kao najznačajnije operacije za relacioni SUBP
- Način optimizacije upita
- Ažuriranje (n -torki i indeksa)

UPROŠĆENA ARHITEKTURA SUBP-a



Tok izvršavanja SQL upita

1. Formiranje kanoničkog stabla izvršenja upita.
2. Perturbacija kanoničkog stabla izvršenja (radi unapređenja performase izvršenja upita).
3. Odabir fizičke rutina (algoritma) izvršavanja operacija relacione algebre.
4. Formiranje različitih *PLANOVA IZVRŠENJA*
5. Izbor *OPTIMALNOG PLANA IZVRŠENJA*
6. Izvršavanje *Plana izvršenja*
7. Formiranje skupa rezultata (odgovora na upit).

ADMINISTRACIJA BAZA PODATAKA(2)

Elementi fizičkog projektovanje baza podataka

Fizičko projektovanje Baze podataka

- Fizičko projektovanje je poslednja faza u procesu projektovanja baze podataka
- Određuje se način na koji se baza podataka konkretno implementira na izabranoj SUBP platformi
- Fizičko projektovanje je proces razvoja efikasno primenljive fizičke strukture baze podataka iz date logičke strukture u cilju zadovoljenja zahteva korisnika za informacijama
- Projektovanje fizičke strukture baze podataka obuhvata aktivnosti koje se tiču organizaciji prostora namenjenog bazi podataka

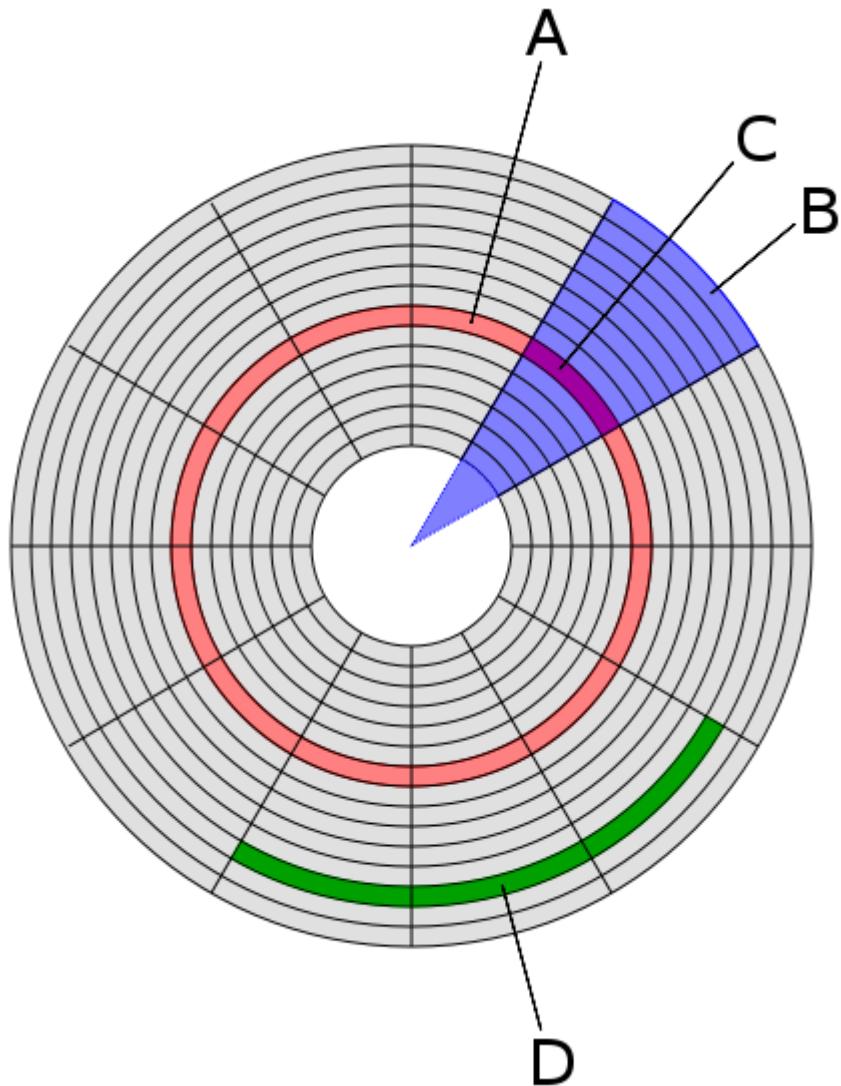
Projektovanje fizičke strukture baze podataka

- Organizacija datoteka sa podacima
- Particionisanje
- Indeksiranje
- Uvodjenje kontrolisane redundanse
- Definisanje pogleda
- Definisanje bezbednosnih i sigurnosnih mehanizama
- Druge aktivnosti kojima se preciziraju sve pojedinosti implementacije baze podataka

Elementi fizičkog projektovanja baza podataka

- Operativni sistem deli trajnu memoriju na blokove jednake veličine
- Veličina bloka je konstanta *Operativnog sistema*
- Blok može biti 512 bajtova, 2Kb, 4Kb
- Svaki blok jednoznačno je određen svojom adresom
- Osnovna operacija s trajnom memorijom je prenos bloka sa određenom adresom iz trajne memorije u operativnu memoriju
- Deo operativne memorije koji učestvuje u prenosu zove se **Buffer**
- Blok je najmanja količina podataka koja se može preneti (ako citamo samo 1Bajt prenosi se ceo blok)

Struktura Diska



- A) - Track - Staza
- B) – Sektor (geometrijski)
- C) - Track sector
- D) – Cluster (alocation unit)

Sektor – Blok Fajl sistema – Stranica BP

- Sektor je fizičko mesto na formatiranom disku koje sadrži podatke
- Sektori se definišu kada se formatira disk - koncentričnim prstenovima od centra ka spoljašnjosti površine diska formiraju se staze
- Svaka staza se deli na delove koji se nazivaju sektori – sadrži 512b (novi format 4096b)
- Osnovna jedinica I/O operacija sa diskom je **fizički blok** (jednak veličini podataka koje sadrži jedan sektor).
- **Logički blok** je blok podataka koji Fajl sistem OS koristi za čitanje i pisanje u i iz datoteka.

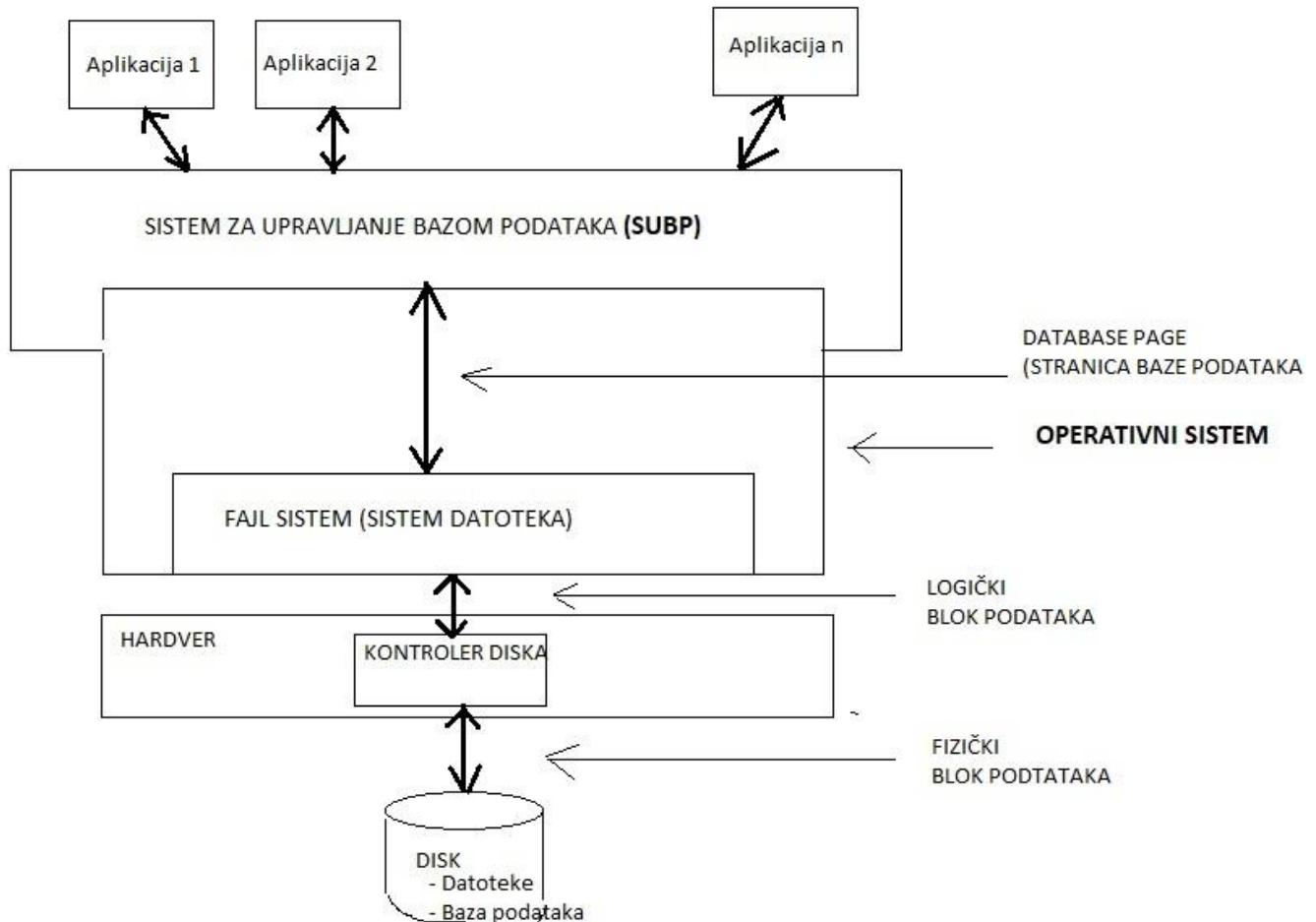
Logički – Fizički Blok

- Veličina Logičkog bloka se po pravilu razlikuje od veličine Fizičkog bloka
- Fizički blok je diktiran najmanjom količinom podataka koje kontroler diska može da pročita ili upiše na disk.
- Veličina logičkog bloka je određena Fajl sistemom operativnog sistema.
- Veličina logičkog bloka je konstanta za dati operativni sistem.
- Na primer, Windows NTFS ima veličinu bloka od 4KB

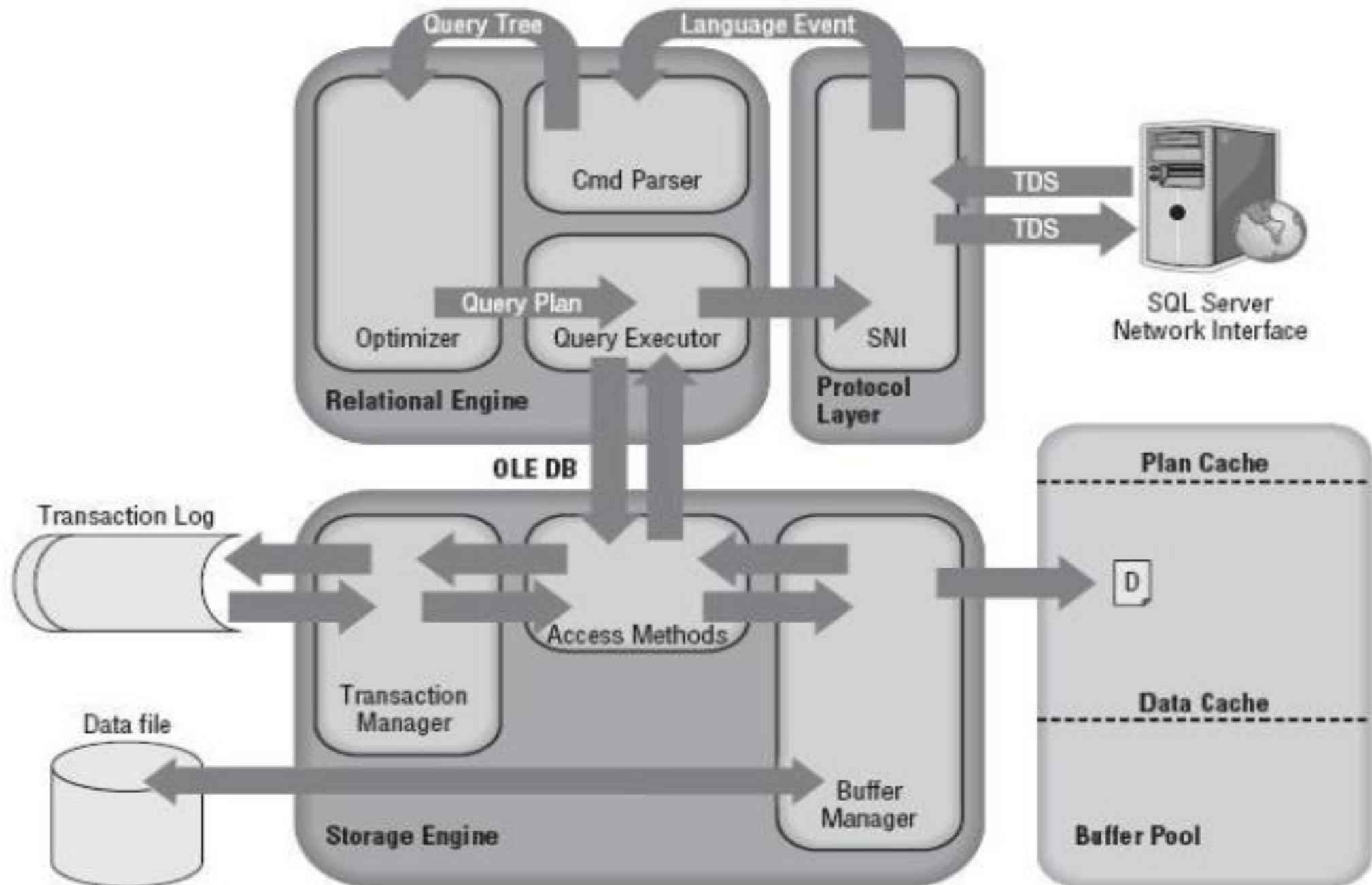
Elementi fizičkog projektovanja baza podataka

- Vreme potrebno za prenos bloka (milisekunde) nije konstantno već zavisi od trenutnog položaja glave diska.
- Vreme prenosa bloka neuporedivo je veće od vremena za bilo koju manipulaciju u operativnoj memoriji (nanosekunde)
- Brzina nekog algoritma za rad s trajnom memorijom određena brojem blokova koje algoritam mora preneti.
- Vreme obrade u operativnoj memoriji zanemarivo u odnosu na vreme potrebno za prenos blokova sa sekundarne u operativnu memoriju.

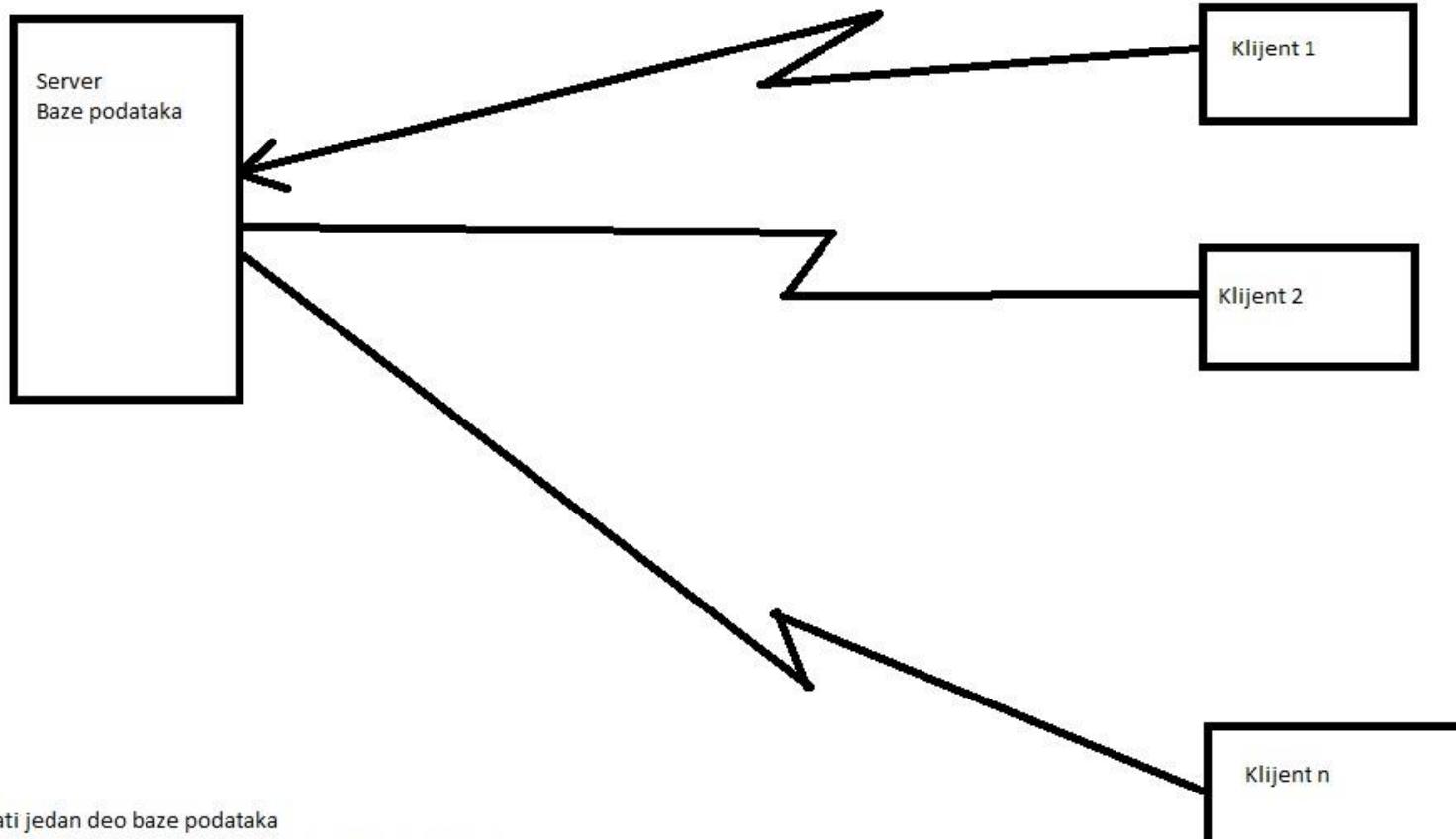
Odnos SUBP-a i modula Sstema datoteka OS



Arhitektura MS SQL Servera



Klijent-Server arhitektura



1. Treba

- Pročitati jedan deo baze podataka
- Izvršiti neka izračunavanja na pročitanom skupu n-torki
- Zatim ažurirati neki drugi deo BP

2. Štampa nekog izveštaja na osnovu sadržaja BP

1. - Prenos podataka (rezultujućih n-torki)kroz mrežu na klijenta
 - Obrada/Izračunavanje se vrši na strani klijenta
 - Vrši se prenos kroz mrežu prenos podataka za ažuriranje

Log Transakcija (Dnevnik Ažuriranja)

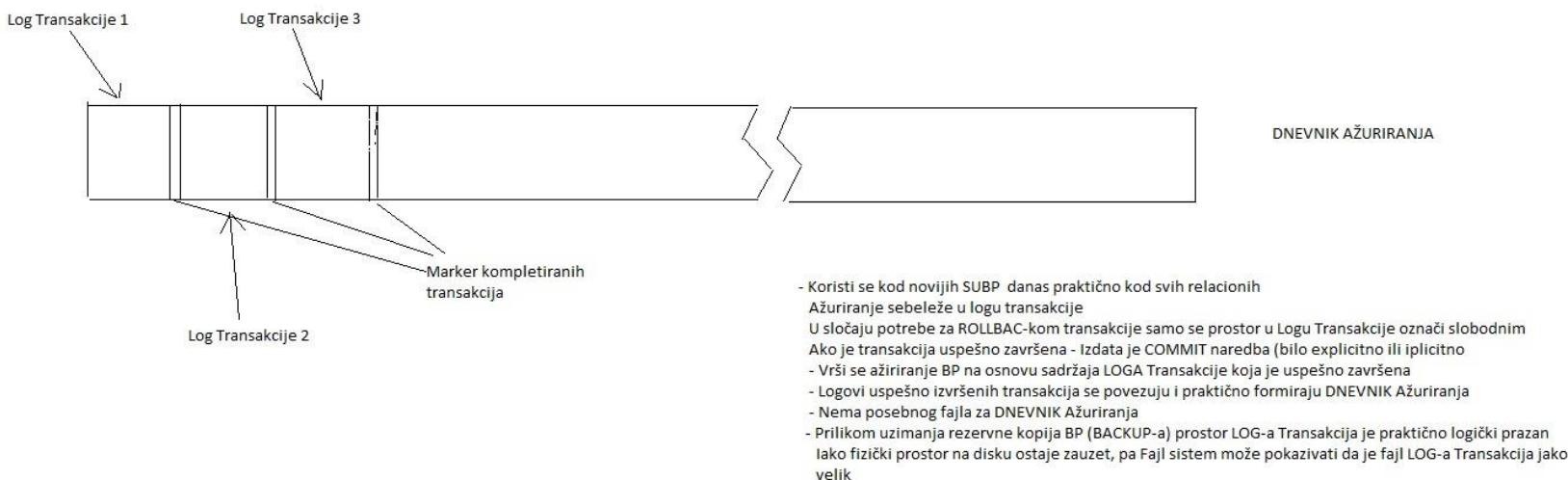
Vrste žurnalizacije

1. Before Image Žurnalizacija
2. After Image žurnalizacija

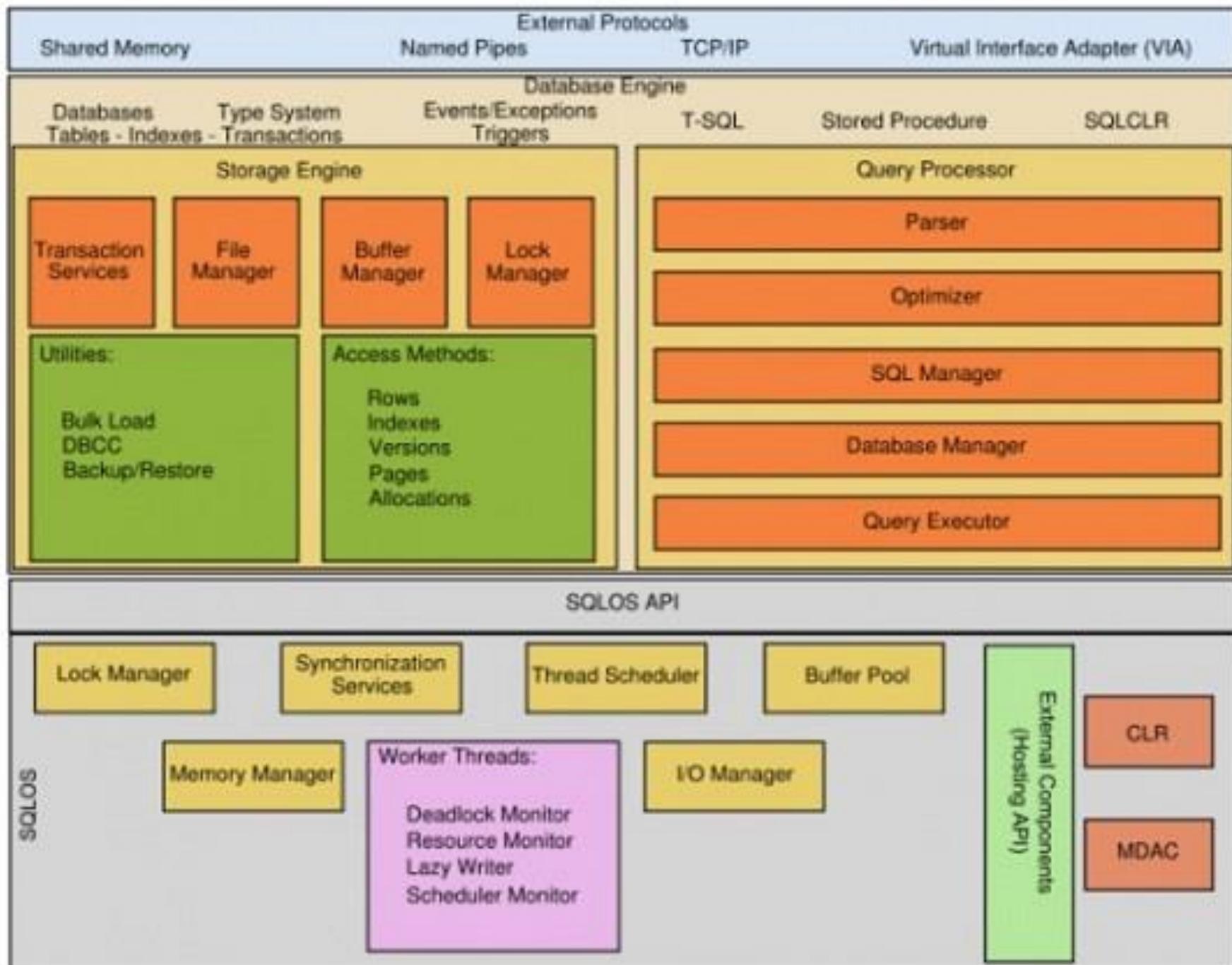
BEFORE IMAGE ŽURNALIZACIJA

- Korišćena kod starijih SUBP
Kod SUBP zasnovanih na mrežnom modelupodataka i pre toga
kod SUBP zasnovanih na hijerarhijskom modelu podataka
- U log transakcija se upisuju Stranice BP pre ažuriranja, a
- samo ažuriranje se vršilo u samoj BP
- Ukoliko dođe do potrebe za ROLLBACK-om stranice pre ažuriranja BP
su se vraćale u BP i na taj način se poništavalo ažuriranje
- Posebno se vodio DNEVNIK AŽURIRANJA u drugoj datoteci u odnosu na log transakcije

AFTER IMAGE ŽURNALIZACIJA



Arhitektura SQL Servera



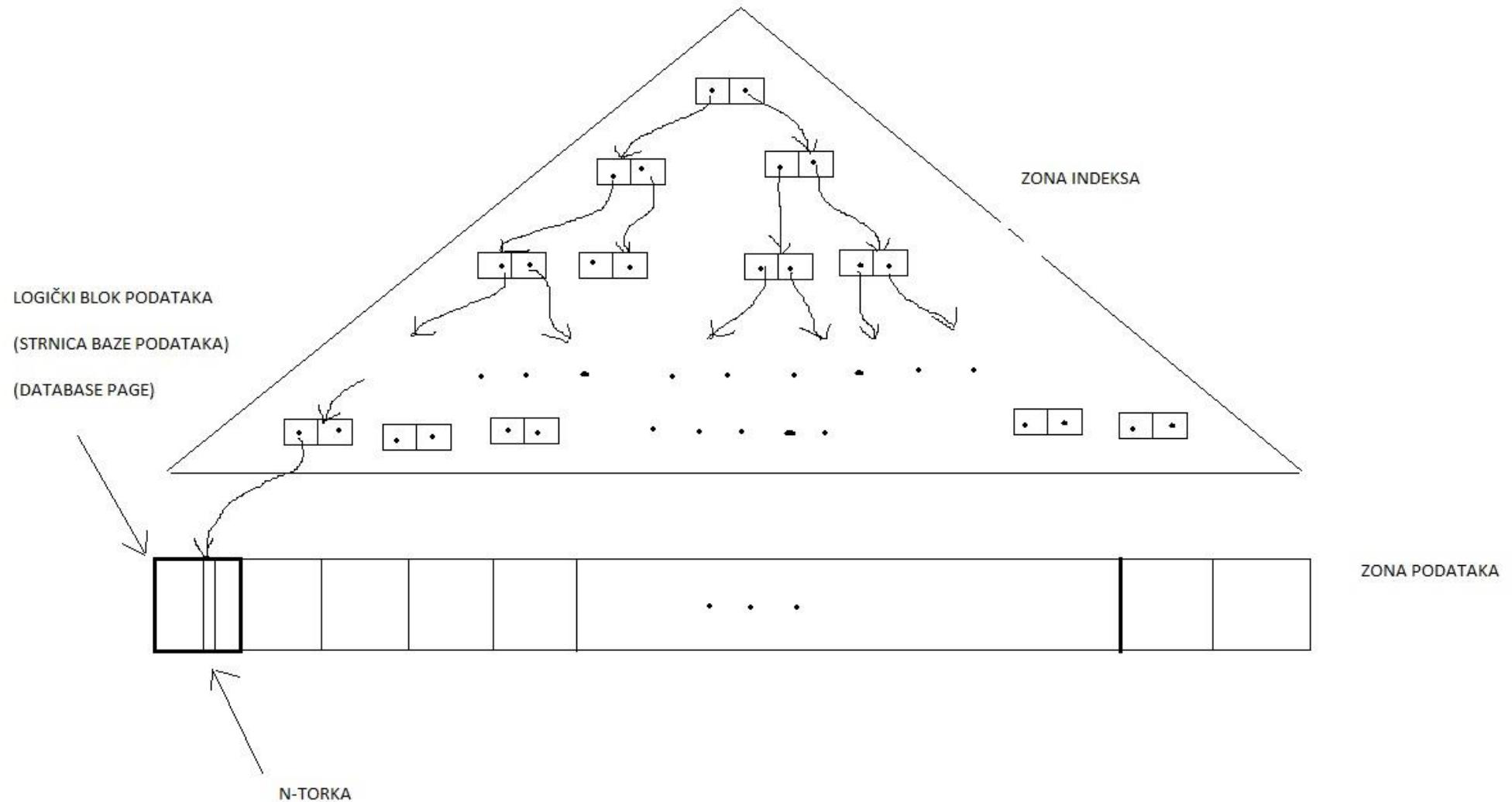
Elementi fizičkog projektovanja baza podataka

- Baza podataka je fizički organizovana u datoteke na sekundarnoj memoriji i objekti baze podataka se smeštaju u te datoteke baze podataka.
- Datoteka je konačan niz zapisa istog tipa uskladištenih u trajnoj memoriji.
- Tip zapisa se zadaje kao uređena n-torka osnovnih podataka (komponenti), gde je svaki osnovni podatak opisan svojim imenom i tipom.
- Sam zapis se sastoji od konkretnih vrednosti osnovnih podataka.

Elementi fizičkog projektovanja baza podataka

- Tipične operacije koje se obavljaju nad datotekama su:
 - Ubacivanje novog zapisa
 - Promena zapisa
 - Brisanje zapisa
 - Pronalaženje zapisa na osnovu vrednosti nekog podatka
- Složenost građe datoteka zavisi od toga koliko efikasno želimo da obavljamo pojedine operacije.
- Cela **Baza podataka** je građena kao skup datoteka.
- Zapisi u datotekama mogu biti međusobno povezani pokazivačima.
- Sve operacije nad Bazom podataka se svode na osnovne operacije nad datotekama.

Šematski prikaz prostora DATOTEKA I INDEKSA



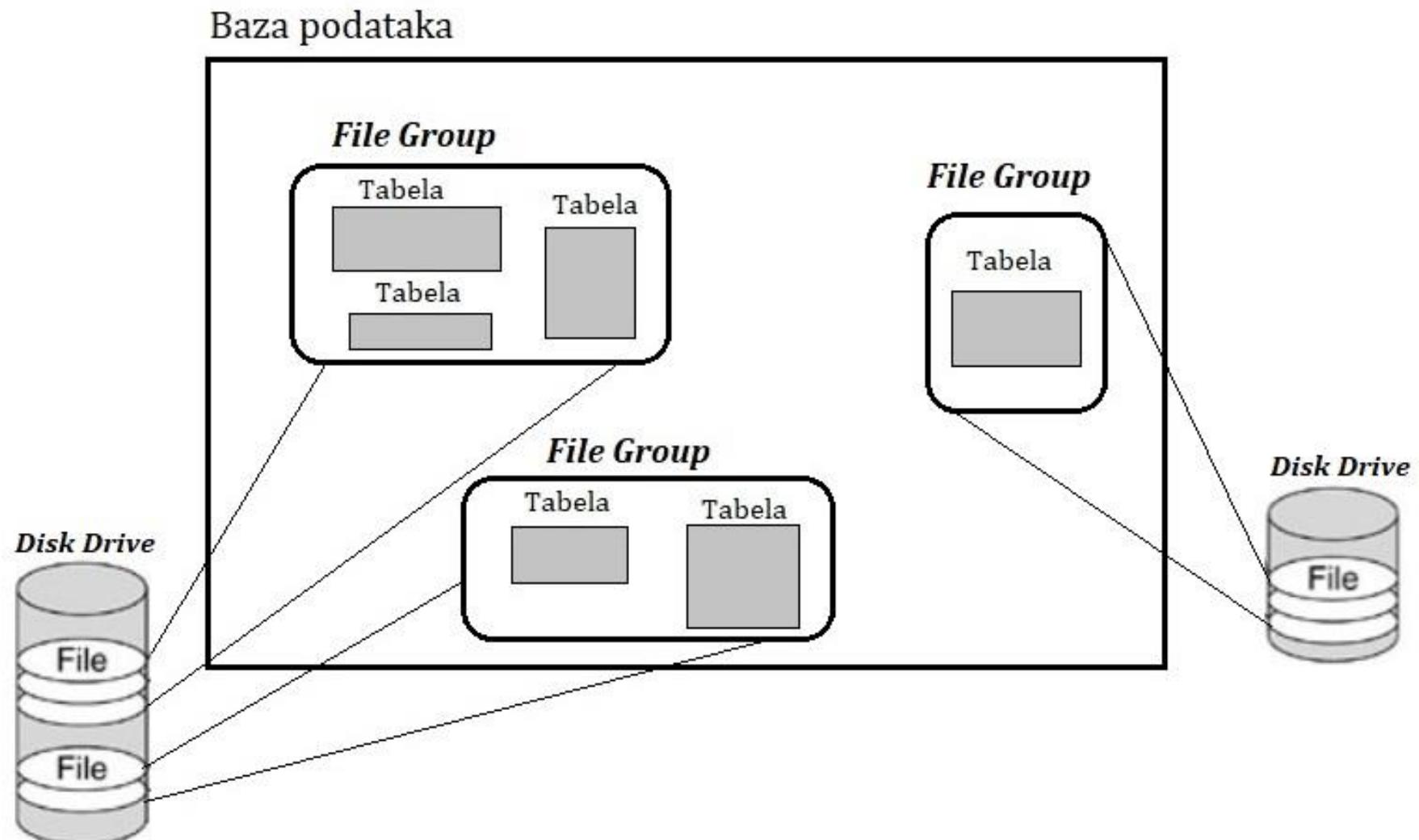
Prostor Baze podataka

- Podaci su korisnicima dostupni preko tabela, ali se iza tih tabela nalaze datoteke odnosno *Data setovi*.
- Tokom procesa Fizičkog projektovanja *ABP* mora mapirati svaku tabelu na odgovarajuću fizičku strukturu koja će skladištiti podatke – *table space*, *data space*, *file group*...
- Svaki *Tablespace (File Group)* je prostor na disku rezervisan za skladištenje podataka u formi datoteka.
- Fizička organizacija baze podataka se sastoji od jednog ili više ***Tablespace***-ova (*File Group-a*), a svaki ***Tablespace (File Group)*** sadrži podatke jedne ili više tabela.

Prostor Baze podataka

- *ABP*, na osnovu korišćenja podataka, odlučuje kako će mapirati tabele na *Tablespace-ove* (*File group-e*) zavisno od tipova *Tablespace-ova* (*File group-a*) koje SUBP podržava i specifičnosti samog SUBP-a.

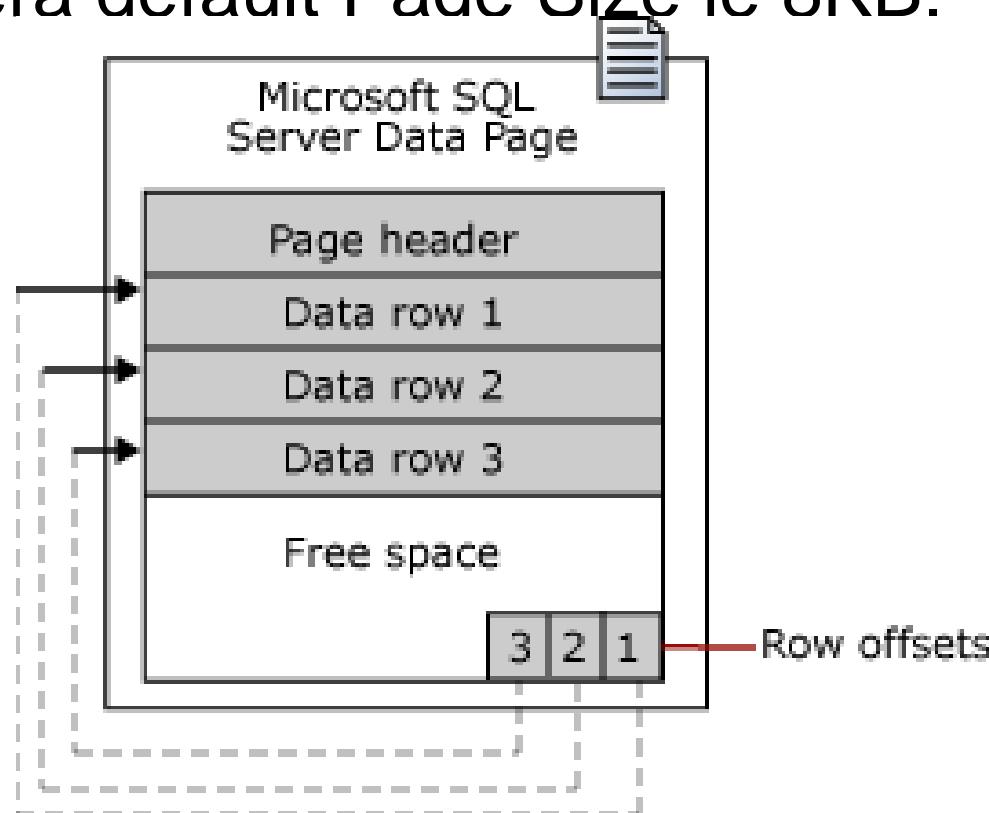
Prostor Baze podataka



Elementi fizičke organizacije skladištenja podataka

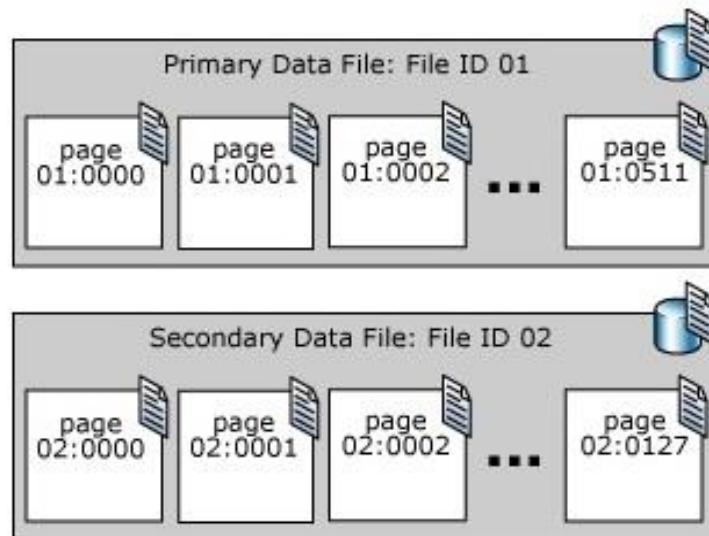
SQL Server data files

- U terminologiji datoteka baza podataka jedinica manipulacije podacima je stranica (Page).
- Važan parametar konfigurisanja fizičke strukture baze podataka je veličina stranice (Page Size).
- Kod SQL Servera default Page Size je 8KB.



Datoteke Baze podataka (SQL Server-a)

- Svaka BP SQL Server-a ima minimalno 2 (dve) datoteke u okviru Sistema datoteka OS (*FILE System-a*)
- Datoteku s podacima (sadrži objekte baze podataka)
- Datoteku loga transakcija (sadrži informacije za oporavak svih transakcija nad bazom podataka)



Slika 7. - Struktura datoteka BP sa prikazom stranica (DataBase pages)

Prostor Baze podataka - SQL Server

- U terminologiji SQL Server-a ekvivalent *Tablespace*-a je ***Filegroup***-a.
- Podrazumevana *Filegroup* se naziva ***Primary***, ali se mogu definisati i druge grupe fajlova za neku bazu podataka.
- Fajlovi imaju logički naziv unutar SUBP-a i fizičku adresu na nivou operativnog sistema.
- Specifikacija osobina fajlova se navodi pri kreiranju ili izmeni opisa Baze podataka.

Kreiranje Baze podataka

- Do sada smo to radili SQL naredbom:

CREATE DATABASE <Naziv_BazePodataka>

- Izvršimo kreiranje baze podataka pod sa nazivom *TestBazaPodataka*. Naredba je:

Create Database TestBazaPodataka

- Postavlja se pitanje:

- Gde su smeštene datoteke naše baze podataka i koliko ih ima?
 - Osim toga koliko prostora zauzima naša baza podataka i šta kada se taj rezervisani prostor popuni podacima.

Datoteke BP SQL Servera

Kod SQL Servera:

- Svaka baza podataka, uključujući i sistemske baze podataka, ima svoj skup datoteka i
- Te datoteke ne deli s ostalim bazama podataka.

Prostor Baze podataka -SQL Server

- U terminologiji SQL Server-a *ekvivalent Tablespace-a* je ***Filegroup***
- Skladištenje podataka
 - Primarna Zona podataka (tabele) zona indeksa
- Datoteke Baze podataka:
 - Primarne datoteke (*.mdf)
 - Sekundarne datoteke (*.ndf)
 - Datoteke Loga transakcija (*.ldf)
- Filegroups:
 - Logička kolekcija datoteka
 - Objekti se mogu kreirati nad Filegroup-om

Datoteke BP - DATA Folder SQL Server-a

This PC > Local Disk (C:) > Program Files > Microsoft SQL Server > MSSQL15.MSSQLSERVER > MSSQL > DATA

Name	Date modified	Type	Size
BazaZaVezbe	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
BazaZaVezbe_log	2/23/2023 3:37 PM	SQL Server Log	8,192 KB
master	2/23/2023 3:37 PM	SQL Server Database	4,544 KB
mastlog	2/23/2023 3:37 PM	SQL Server Log	2,048 KB
model	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
model_msdbdata	9/25/2019 12:09 AM	SQL Server Database	13,696 KB
model_msdblog	9/25/2019 12:09 AM	SQL Server Log	512 KB
model_replicatedmaster	9/25/2019 12:09 AM	SQL Server Database	512 KB
model_replicatedmaster	9/25/2019 12:09 AM	SQL Server Database	4,544 KB
modellog	2/23/2023 3:37 PM	SQL Server Log	8,192 KB
MSDBData	2/23/2023 3:37 PM	SQL Server Database	15,104 KB
MSDBLog	2/23/2023 3:37 PM	SQL Server Log	768 KB
Nastava	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
Nastava_log	2/23/2023 3:37 PM	SQL Server Log	8,192 KB
Salon	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
Salon_log	2/23/2023 3:37 PM	SQL Server Log	8,192 KB
Salon2	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
Salon2_log	2/23/2023 3:37 PM	SQL Server Log	73,728 KB
Salon3	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
Salon3_log	2/23/2023 3:37 PM	SQL Server Log	73,728 KB
SDKlub	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
SDKlub_log	2/23/2023 3:37 PM	SQL Server Log	8,192 KB
STUDENT	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
STUDENT_log	2/23/2023 3:37 PM	SQL Server Log	8,192 KB

Sistemске Baze SQL Servera

Posle instalacije SQL Servera na njemu se nalaze samo njegove sistemske baze podataka:

- master
- model
- msdb i
- tempdb

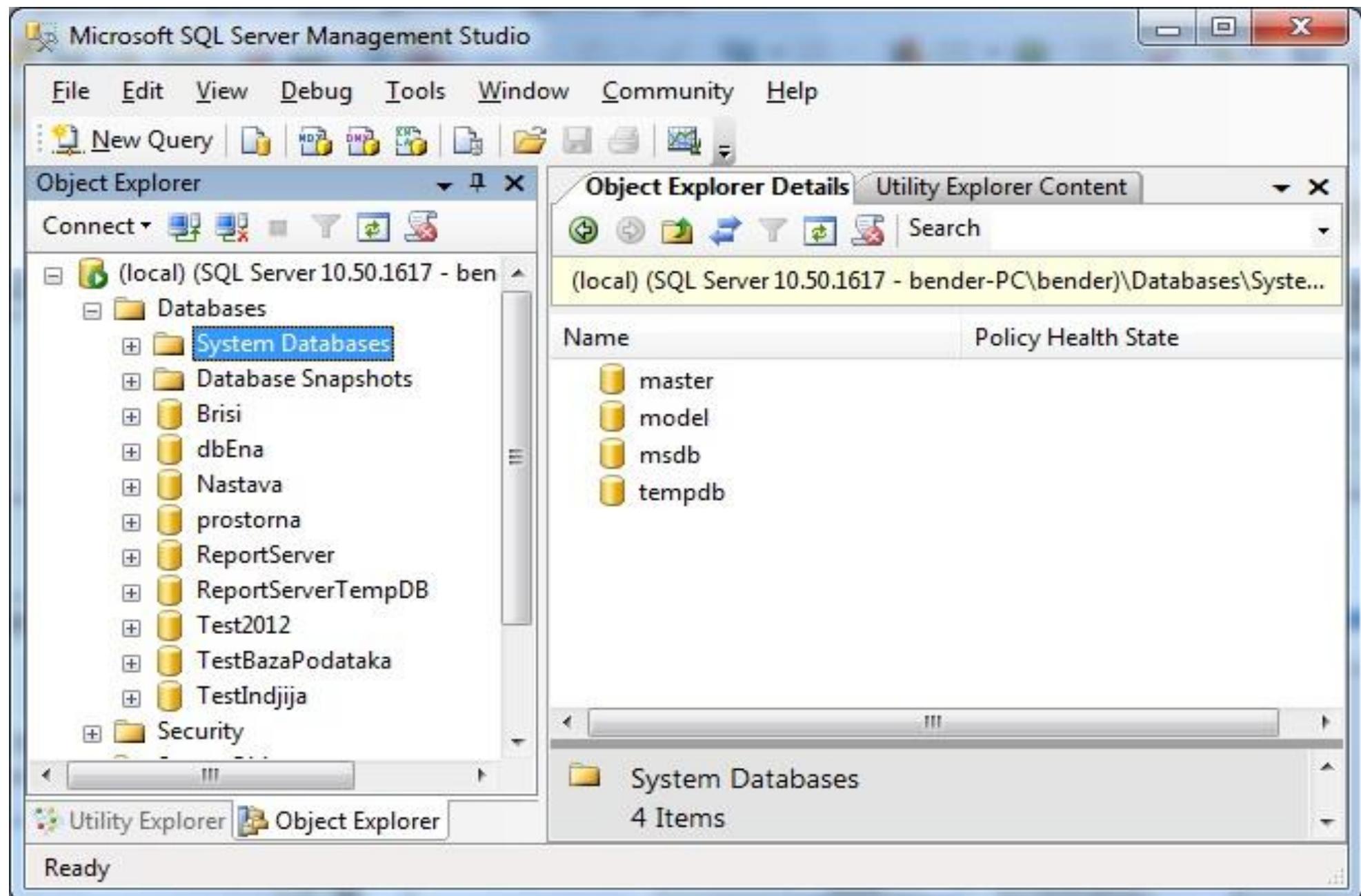
Praktično se rečnik podataka (system catalog) RSUBP MS SQL Servera satoji te četiri baze.

DBA – Koristi Alate za administraciju

Relacioni SUBP mogu se administrirati:

- preko komandnih linija
- Vizuelnim alatima za konfiguraciju i podešavanje sistema baza podataka
 - *MySQL Workbench*
 - *IBM DB2 Control Center*
 - *Oracle Enterprise Manager*
 - *SQL Server Management Studio*

SQL Server Management Studio



SQL Server – Sistemske baze podataka

Master baza podataka:

- Sadrži informacije potrebne samom SQL Servru.
- U njoj se memorišu korisnički nalozi, svi parametri konfiguracije sistema.
- U master bazi podataka se memorišu i podaci o svim korisničkim bazama podataka, uključujući lokaciju datoteka svake baze podataka.
- U master bazu podataka se upisuju informacije za inicijalizaciju SQL Servera.
- SQL Server se ne može pokrenuti ako nije raspoloživa master baza podataka (uzeti u obzir prilikom uzimanja rezervnih kopija BP).
- Zbog toga je uvek potrebno imati raspoloživu kopiju (backup) master baze podataka.

SQL Server – Sistemske baze podataka

Tempdb – sadrži:

- Privremene tabele i privremene uskladištene procedure
- Koristi se kada je potreban prostor za privremene tabele, međurezultate i radne tabele koje generiše sam SQL Server.
- Tempdb se ponovo kreira svaki put kad se startuje SQL Server, tako da se sistem startuje s praznom tempdb bazom podataka.
- Tokom rada se vrši auto-prosirenje tempdb baze podatka kad je potrebno više prostora u toku rada SQL Servera,
- ali se za razliku od ostalih baza podataka reinicijalizuje na početnu veličinu kad god se SQL Server ponovo startuje.

SQL Server – Sistemske baze podataka

Tempdb - nastavak

- Ako je inicijalno definisan prostor za tempdb mali, sistem će u toku rada deo vremena trošiti na povećanje prostora tempdb baze podataka koji je potreban za podršku obimu posla koji SQL Server izvršava.
- Da bi se izbeglo trošenje vremena Administrator Baze Podataka (ABP) može koristeći naredbu ALTER DATABASE povećati prostor tempdb sistemske baze podataka

SQL Server – Sistemske baze podataka

model sistema baza podataka se koristi za:

- Modele (template) za sve baze podataka koje se kreiraju na sistemu.
- Kada se izda CREATE DATABASE naredba, prvi deo nove baze podataka koja se kreira se kopiranjem sadržaja ***model*** sistema baza podataka.
- Ostatak nove baze podataka se popunjava praznim stranicama.
- Zbog toga što se ***tempdb*** sistema baza podataka ponovo kreira svaki put kad se startuje SQL Server, model sistema baza mora uvek postojati na SQL Serveru
- Svi korisnički definisani objekti u model bazi podataka se kopiraju u svaku novu bazu podataka kada se ona kreira.

SQL Server – Sistemske baze podataka

msdb sistema baza podataka se koristi:

- od strane Agenata SQL Servera za upravljanje upozorenjima, poslovima i za memorisanje (čuvanje) operatora
 - Šta su Agenti SQL Servera? – Njegove komponente koje se pokreću kao *Servisi operativnog sistema* (koji se startuju bilo ručno ili automatski prilikom startovanja).

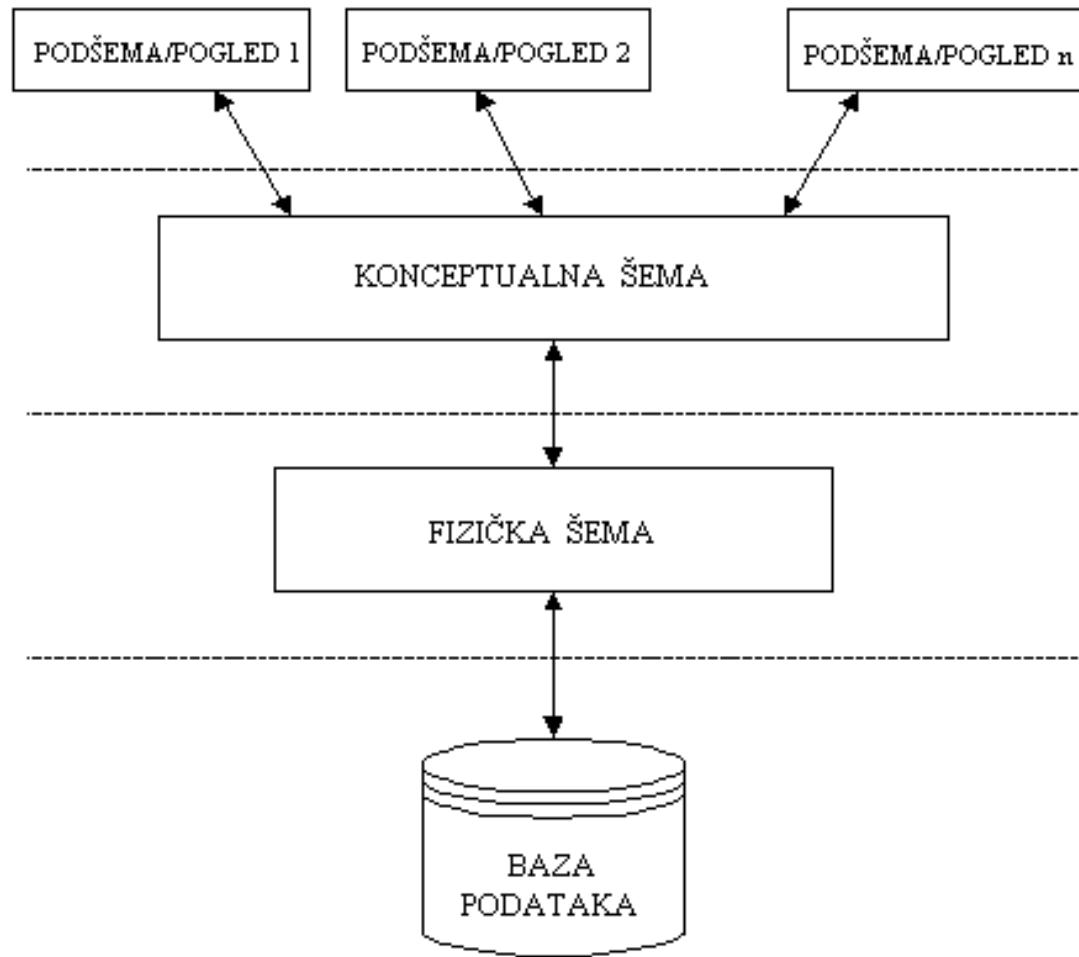
Administracija baza podataka (3)

Fizičko projektovanje baze podataka

Faze projektovanja BP

1. Prikupljanje i analiza zahteva
 2. Logičko projektovanje baze podataka
 3. Izbor sistema za upravljanje bazom podataka
 4. Prevođenje modela podataka
 5. **Fizičko projektovanje BP**
 6. Implementacija BP
-

Arhitektura tronivojskog SUBP

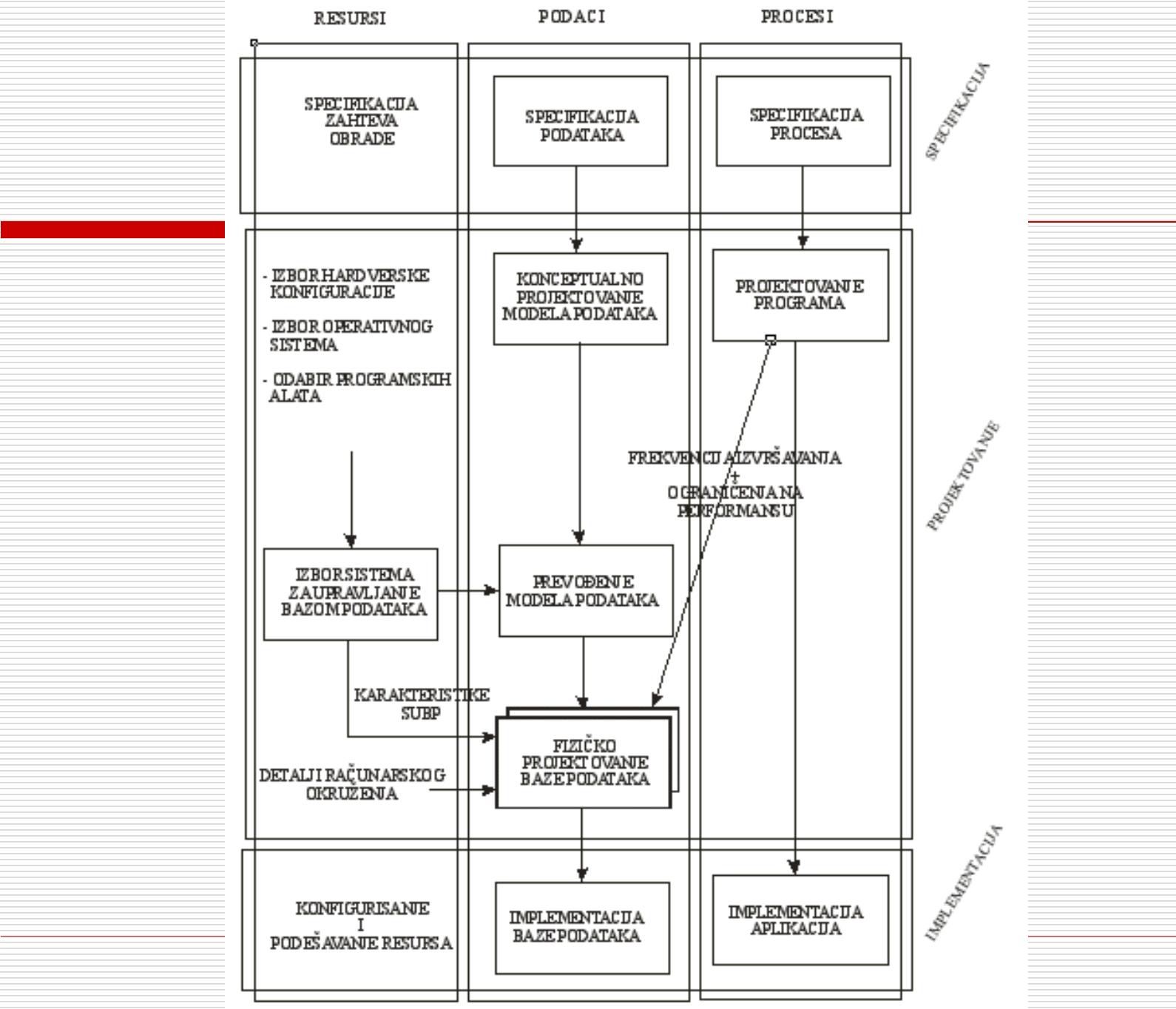


Osnovna karakteristika savremenih SUBP:

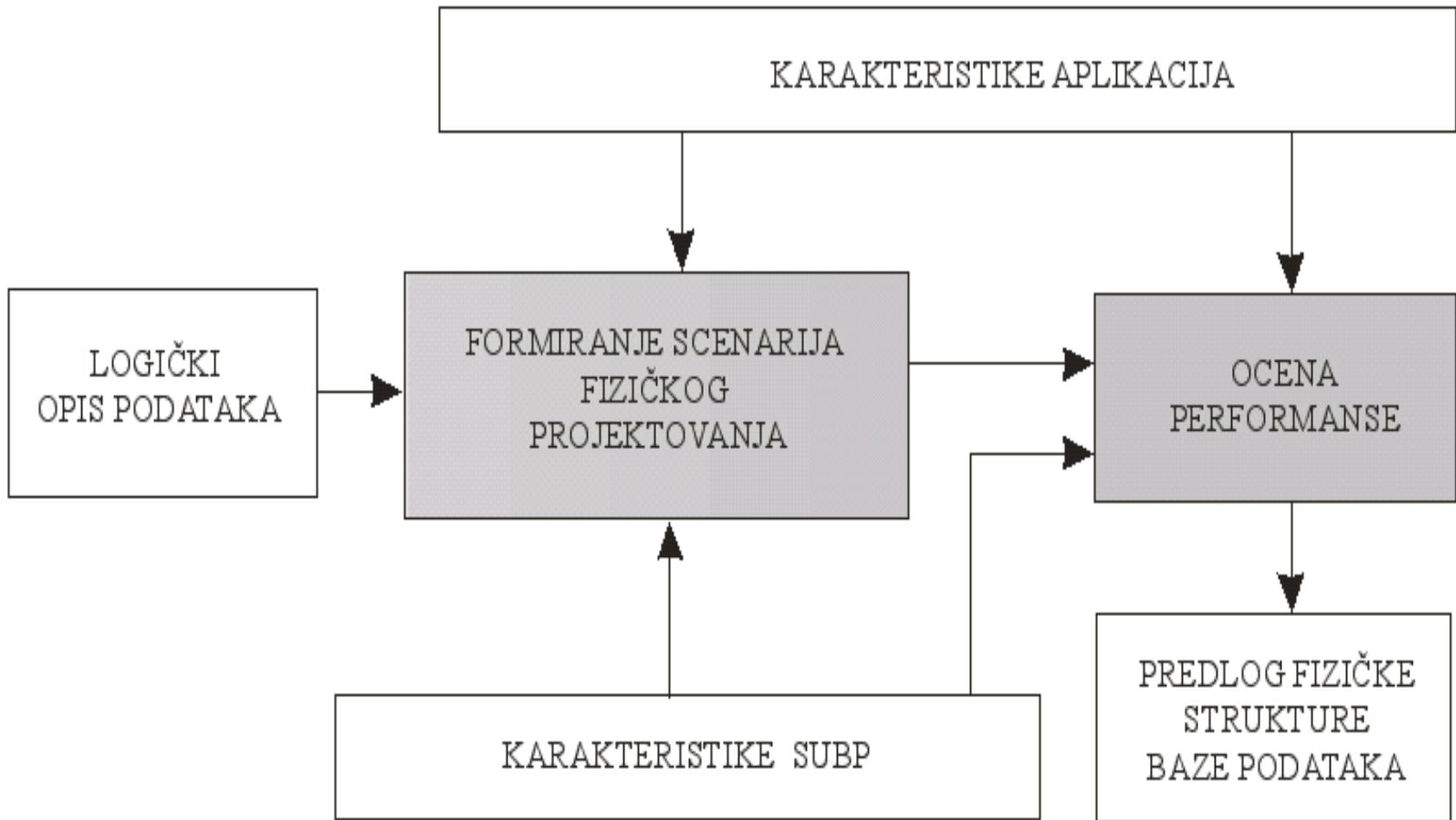
- Fizička nezavisnost podataka - razdvajanje logičkih tipova podataka i njima pridruženih operacija od fizičke reprezentacije

Posledica:

- Logički model podataka može se predstaviti sa više različitih fizičkih struktura podataka
-



Blok šema fizičkog projektovanja



Karakteristike relacionih SUBP

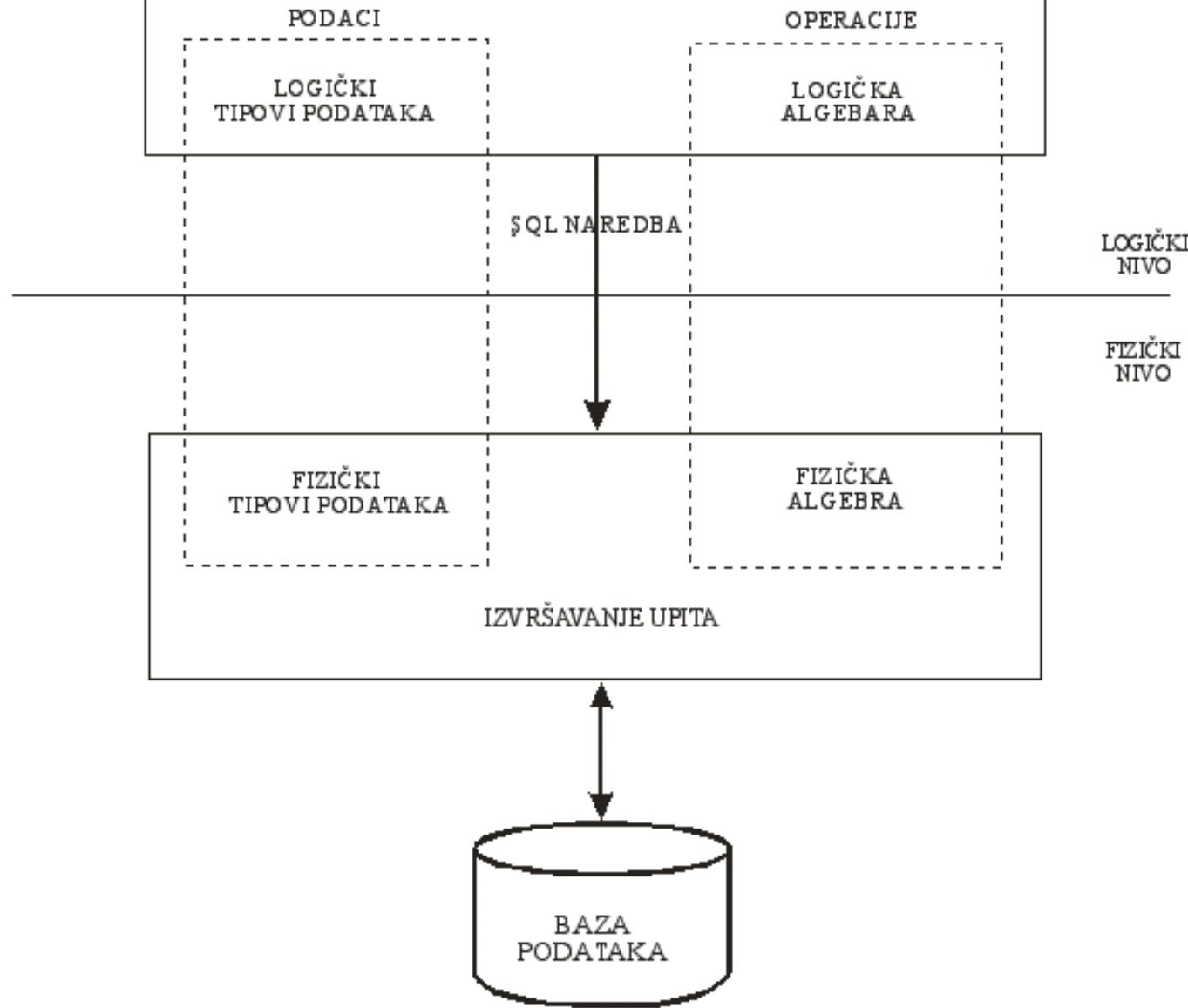
- **Neproceduralni upitni jezik**
 - **SUBP obezbeđuje mehanizam da se dođe do traženih podataka**
 - **Algoritmi obrade**
 - implementacija relacionih operacija
 - implementacija funkcija (agregacija, grupisanja, eliminacija duplikata ...)
-

Karakteristike relacionih SUBP

- Zadatak sistema je izbor optimalnog načina izvršavanja upita**

- Za dati skup algoritama obrade i**
 - Projektovanu fizičku strukturu baze podataka**
-

STANDARDNI UPITNI JEZIK - SQL



Karakteristike relacionih SUBP bitne za fizičko projektovanje

- Fizičke strukture podataka.
 - Implementacija osnovnih relacionih operacija
(Selekcije, Projekcije, Operacija nad skupovima)
 - Implementacija operacije Spoja
 - Način optimizacije upita
 - Ažuriranje (n -torki i indeksa)
-

Fizičke strukture podataka

- Kakvu fizičku organizaciju podataka podržava konkretni (odabrani) SUBP.
 - Kako je fizički organizovan Prostor baze podataka kod konkretnog (odabranog) SUBP.
 - Kakav koncept grupisanja podataka podržava (**File Group**-e, **Tablespace**-eve, *Data set*-ove....).
 - Koje vrste pristupnih struktura za brže pristup (pronalaženje) podataka u bazi podržava.
 - Koje vrste indeksa konkretni SUBP podržava i koju fizičke strukture podataka konkretni SUBP koristi.
 - Da li podržava heširanje - pozicija (fizička) podatka se određuje na osnovu njegove vrednosti

Implementacija operacije SELEKCIJE

- ❑ Za jedan uslov selekcije koristi se
 - pristupna putanja, ili
 - linearno pretraživanje
 - ❑ za složeni konjuktivan uslov (log. i) izbor putanje se vrši na osnovu
 - Faktora selektivnosti pojedinačnih uslova
 - ❑ za složeni disjunktivan uslov (log. ili) koristi se
 - linearno pretraživanje, ili
 - Pristup preko više putanja i formira se unija *IDN*-ova
-

Implementacija operacije PROJEKCIJE

- Jednostavna kad *<lista obeležja>* sadrži primarni ključ;
 - u suprotnom mora se vršiti eliminacija duplikata; vrlo često se koristi heširanje
-

Implementacija operacija nad skupovima

- Sortiranje relacija + linearni “prolaz” kroz relacije
 - Može se koristiti i heširanje ukoliko ga SUBP podržava
 - Pri implementaciji pristupnih rutina vrši se kombinacija operacija
-

Implementacija operacije SPOJ

Metod ugnježdenih petlji

- Najjednostavniji
 - Koristi se kad ne postoje pristupne strukture
 - Vreme izvršavanja proporcionalno ($n \times m$)
 - Gde su n i m brojevi n-torki u relacijama nad kojima se vrši operacija SPOJ-a
-

Implementacija operacije SPOJ

Sort-Merge metod

- Zahteva uređenost relacija po vrednosti obeležja spajanja
 - Kad su relacije unapred sortirane ima značajnu prednost u odnosu na metod ugnježdenih petlji
 - Troškovi proporcionalni ($n \times \log m$)
-

Implementacija operacije SPOJ

Heš metod

- Jedan od najefikasnijih načina fizičke implementacije operacije SPOJ-a.
 - Performansa izvršavanja zavisi od *heš funkcije*.
 - Troškovi proporcionalni $(n + m)$.
-

Implementacija operacije SPOJ

Korišćenjem posebnih struktura podataka

- Efikasne indeksne strukture
- Strukture za podršku samo fizičke implementacije operacije spoja:

Join indeksi	Indeksi koji se kreiraju i koriste samo za operaciju Spoj-a.
Kd – stablo	Višedimenzionalna samobalansirajuća B stabla
T – stablo	Vrsta binarnih stabala koja se koriste kada se indeksi i podaci skladište u operativnoj memoriji.

Zadatak optimizatora upita

- ❑ Određivanje alternativnih planova za izvršavanje upita
 - Radi smanjenja troškova optimizacije uglavnom se razmatra samo podskup svih mogućih planova
 - ❑ Estimacija troškova alternativnih planova i izbor plana sa najnižim troškovima
-

Osnovne tehnike optimizacije

Heuristička optimizacija -

Heuristička optimizacija transformiše stablo upita korišćenjem skupa pravila koji a tipično (ali ne u svim slučajevima) poboljšavaju performanse izvršenja.

Optimizacija estimacijom troškova izvršavanja upita -

Optimizacija zasnovana na troškovima je "skupa", SUBP-ovi mogu koristiti heurstiku optimizaciju da smanje broj izbora koji se moraju napraviti estimacijom troškovima.

Semantička optimizacija - Proces transformacije upita koji je izdao korisnik u drugi upit koji, zbog semantike daje isti odgovor.

Heuristička optimizacija

- Zasniva se na transformacionim pravilima;
 - Polazi se od kanoničkog stabla upita;
 - Vrši se transformacija polaznog, kanoničkog, stabla upita u konačno stablo upita koje ima bolju performansu;
 - Vrši se izbor pristupnih rutina i algoritama za operacije upita koje je moguće primeniti za raspoložive pristupne putanje baze podataka
 - Osnovno pravilo je unarne operacije izvršavati pre binarnih;
-

Optimizacija estimacijom troškova izvršavanja upita (1 / 2)

- Vrši se procena i poređenje troškova različitih planova izvršavanja upita;
 - Odabira se plan sa najnižim očekivanim troškovima;
 - Obično se ograničava broj planova koji se pri optimizaciji razmatra da bi se trošilo manje vremena;
 - Način optimizacije pogodan za kompilirane upite;
-

Optimizacija estimacijom troškova izvršavanja upita (2/2)

- Kod nekih SUBP se vrši “potpuna” optimizacija kompiliranih i “delimična” optimizacija interpretativnih upita, koja zahteva manje vremena;
 - Funkcija troškova je aproksimativna, pa je moguć i izbor plana izvršenja koji nije optimalan;
-

Ažuriranje relacionih baza podataka

- Neophodno je analizirati troškove pristupa podacima i troškove samog ažuriranja
 - Ograničavajući faktor na broj pristupnih struktura je odnos smanjenja troškova koje neki indeks donosi i troškova njegovog održavanja;
-

Ažuriranje relacionih baza podataka

- Troškovi ažuriranja n -torki i indeksa imaju važan uticaj na rezultat procesa selekcije indeksa i na odluke optimizatora;
 - U zavisnosti od toga da li se objektima BP pristupa istim redosledom kojim su smešteni u bazu podataka ili ne primenjuju se različite formule za računanje troškova;
-

Korisničko fizičko projektovanje BP

- Sledi posle projektovanja konceptualne šeme baze podataka;
 - Cilj je obezbititi dobru performansu često zahtevanih upita i operacija ažuriranjanad BP koja se projektuje;
-

Korisničko fizičko projektovanje BP

- Polazna osnova FP je logički opis podataka koji je rezultat *logičkog projektovanja*;
 - Svaki SUBP podržava određeni skup fizičkih struktura podataka, pa se FP može definisati kao proces stvaranja efikasne fizičke strukture BP za konkretni SUBP, na osnovu konceptualne šeme;
-

Korisničko fizičko projektovanje BP

- Rezultat fizičkog projektovanja je *fizička šema* BP za konkretni (odabrani) SUBP.
 - Pri FP nije cilj samo doći do fizičke strukture BP, već to uraditi na način koji garantuje dobru performansu Baze podataka i celog IS.
 - Pošto se korisnički zahtevi proširuju i menjaju tokom vremena, da bi se obezbedila dobra performanse uglavnom je neophodno *podešavanje* parametara BP pri uvođenju sistema u rad, a kasnije tokom eksploatacije.
-

Zahtevi pri fizičkom projektovanju

Da bi se minimizirali troškovi U/I operacija u sistemu baze podataka važno je:

- a) da strukture na sekundarnoj memoriji obezbeđuju pretraživanje relevantnih podataka preko efikasnih pristupnih putanja;
 - b) da su podaci organizovani i smešteni na sekundarnu memoriju na način koji minimizira U/I troškove pri pristupu podacima.
-

Opcije pri fizičkom projektovanju

- Najniži nivo FP baza podataka obuhvata:
 - Izbor formata datoteka; definiše se broj datoteka BP, veličina datoteka, veličina stranica datoteka baze podatka, način grupisanja stranica u jedinice U/I prenosa, veličina kontinualnih proširenja ...
 - Korišćenje pokazivača; podržavaju navigaciju kroz BP; unapređuju performansu pri uparivanju skupova podataka; implementiraju se kao identifikatori n -torki (*IDN*-ovi)
-

Opcije pri fizičkom projektovanju

- Vertikalnu podelu relacija;** rezultuje smanjenjem broja pristupa sekundarnoj memoriji; grupe obeležja iste relacije smeštaju se u različite datoteke BP.

 - Kompresiju podataka;** od interesa je iz dva razloga: smanjenje potrebnog prostora na sekundarnoj memoriji i poboljšanja performanse obrade;
-

Opcije pri fizičkom projektovanju

Asocijativno pretraživanje;

- Koristi se da bi se smanjio broj pristupa sekundarnoj memoriji;
 - Postoji u svim SUBP; što je razlog da bez obzira na odabrani SUBP ***izbor indeksa*** predstavlja aktivnost FP koja se uvek sprovodi bilo da se radi o projektovanju jednostavnih ili složenih baza podataka;
 - Najpoznatija i najčešće korišćena indeksna struktura u relacionim bazama podataka je B-stablo (različite varijante – zavisno od konkretnog SUBP-a).
-

Asocijativno pretraživanje (nastavak)

- Mogu se koristiti tako da redosled i organizacija indeksa određuje redosled n -torki u datoteci podataka; *grupišući (klaster) indeksi*;
 - Ostali indeksi se nazivaju *negrupišući*; moraju biti gusti, odnosno postoji isti broj ulaznih tačaka u indeksnoj strukturi koliko ima n -torki;
 - Najveći broj SUBP pri pretraživanju indeksa ne pristupa n -torkama, pa je u nekim slučajevima do podataka moguće doći samo pretraživanjem indeksa
-

Opcije pri fizičkom projektovanju

- Kontrolisano uvođenje redundantnih podataka
 - Replikacija - opcija FP kojom se definiše čuvanje istih delova baze podataka na više uređaja sekundarne memorije;
 - Izvedene informacije - odnosno materijalizovani relacioni pogledi, nastaju kao posledica smeštanja rezultata operacije spoja na sekundarnu memoriju;
-

Opcije pri fizičkom projektovanju

- Join indeksi** - U izvedene relacije se smeštaju samo neophodne informacije za efikasno izvršavanje operacije spoja; eliminisu problem ažuriranja;
 - Denormalizacija** - Replikacija pojedinačnih vrednosti obeležja radi bržeg pristupa; značajna opcija za sisteme koji ne podržavaju grupisanje po različitim tipovima podataka;
-

Opcije pri fizičkom projektovanju

- Način smeštanja i fizički redosled podataka utiču na performansu pristupa podacima:
 - **Grupisanje podataka** - Organizacija elemenata podataka na sekundarnoj memoriji vrši se na takav način da se maksimizira količina relevantnih podataka koji se čitaju jednom U/I operacijom i na taj način smanji broj potrebnih U/I operacija da bi se zadovoljio zahtev obrade BP;
 - **Razdvajanje relacija** - Kod nekih SUBP i stranica datoteka BP po više uređaja sekundarne memorije omogućava veći broj U/I operacija u jedinici vremena, a time i veću U/I propustnost.
-

Završna razmatranja opcija FP

- Većina analiziranih opcija egzistira nezavisno od modela podataka;
 - Zbog velikog broja opcija FP predstavlja kompleksan zadatak;
 - Izvor kompleksnosti FP je i to što je većina odluka međuzavisne;
 - U komercijalnim SUBP nisu sve opcije FP podjednako zastupljene i nisu sve podjednako važne;
-

Izbor indeksa za tebele relacione BP

Obuhvata:

- Izbor redosleda n -torki u relaciji.
- Izbor obeležja za definisanje skupa indeksa.

Zavisi od:

- Karakteristika podataka.
 - Karakterističnog skupa aplikacija.
-

Karakteristike podataka

❑ Karakteristike podataka obuhvataju:

- Broj relacija;
 - Očekivani broj n -torki svake relacije;
 - Broj obeležja svake relacije;
 - Za svako obeležje:
 - Tip obeležja,
 - Dužina u bajtima,
 - Očekivani broj mogućih vrednosti koje obeležje dobija u n -torkama relacije;
 - veličina svake relacije izražena u broju osnovnih jedinica U/I prenosa.
-

Opis karakterističnog skupa aplikacija

- Spisak upita i frekvencije njihovog izvršavanja;
 - Spisak ažuriranja i njihove frekvencije izvršavanja;
 - Ciljnu performansu za svaki tip upita i ažuriranja.
-

Opis upita u skupu aplikacija

- Kojim relacijama se pristupa;
 - Koja obeležja se zahtevaju (u SELECT klauzuli);
 - Koja obeležja učestvuju u izrazima za uslove selekcije i spajanja relacija (u WHERE klauzuli) i očekivanu selektivnost tih uslova.
-

Opis ažuriranja u skupu aplikacija

- Nad kojim obeležjima je definisan uslov selekcije ili spajanja (u WHERE klauzuli) i očekivanu selektivnost tih uslova;
 - Tip ažuriranja (INSERT, DELETE ili UPDATE), i relacije koje se ažuriraju;
 - Za naredbu UPDATE, obeležja čije vrednosti se modifikuju.
-

Obeležja za definisanje pristupnih putanja

- Konstantni indeksi
 - Pogodna obeležja
 - Obeležje nad kojim postoji predikat u naredbi, i ako sistem može koristiti (neki) indeks za obradu tog predikata (WHERE klauzula)
 - Obeležja koja se pojavljuju u okviru GROUP BY i ORDER BY klauzula
-

Preporuke za selekciju indeksa

- Postupak selekcije
 - Kriterijumi za selekciju
-

Kriterijumi za selekciju indeksa

- Kada indeksirati;
 - Izbor ključa pretraživanja;
 - Složeni ključ pretraživanja;
 - Grupisanje;
 - Heširanje ili stabla indeksa;
 - Uravnoteženje troškova održavanja indeksa;
-

Indeksi nad Relacijama (Tabelama) Baze podataka (1/ 3)

- Indeksi su pomoćne strukture koje se kreiraju da bi se urzao pristupa podacima u Relacijama (Tabelama) prilikom pretraživanja.
 - Kreira ih Administrator baze Podataka (ABP) preko SUBP-a.
 - SUBP ih automatski održava posle njihovog kreiranja.
 - Šta znači da se indeksna struktura (B-Stablo) drži balansiranim?
-

Indeksi nad Relacijama (Tabelama) Baze podataka (1/ 2)

- Sa aspekta grupisanja podataka postoji dve vrste indeksa:
 - Grupišući (*Cluster*) indeksi
 - Nad tabelom može biti samo jedan Grupišući (*Cluster*) indeks.
 - NeGrupišući (*Non Cluster*) indeksi
 - Nad tabelom može biti više Negrupišući (*NonCluster*) indeks-a.

Indeksi nad Relacijama (Tabelama) Baze podataka (3 / 3)

- Sa aspekta jedinstvenosti vrednosti indeksnog ključa postoji takođe dve vrste indeksa:
 - Jedinstveni indeksi
 - Nad jednom tabelom može biti definisano više jedinstvenih indeksa (nad različitim skupom obeležja).
 - Nejedinstveni indeksi
 - Nad tabelom takođe može biti definisano više nejedinstvenih indeksa.
- Šta je indeksni ključ?
 - Može biti Prost i Složen.

Administracija Baza podataka

Objekti Relacione baze podataka

Objekti Baze podataka

- Indeksi
 - Pogledi
 - Proceduralna proširenja SQL-a (Transact-SQL)
 - Trigeri
 - Korisničke funkcije (*User Defined Functions*)
 - Uskladištene procedure (*Stored procedures*)
-

Pogledi - VIEW

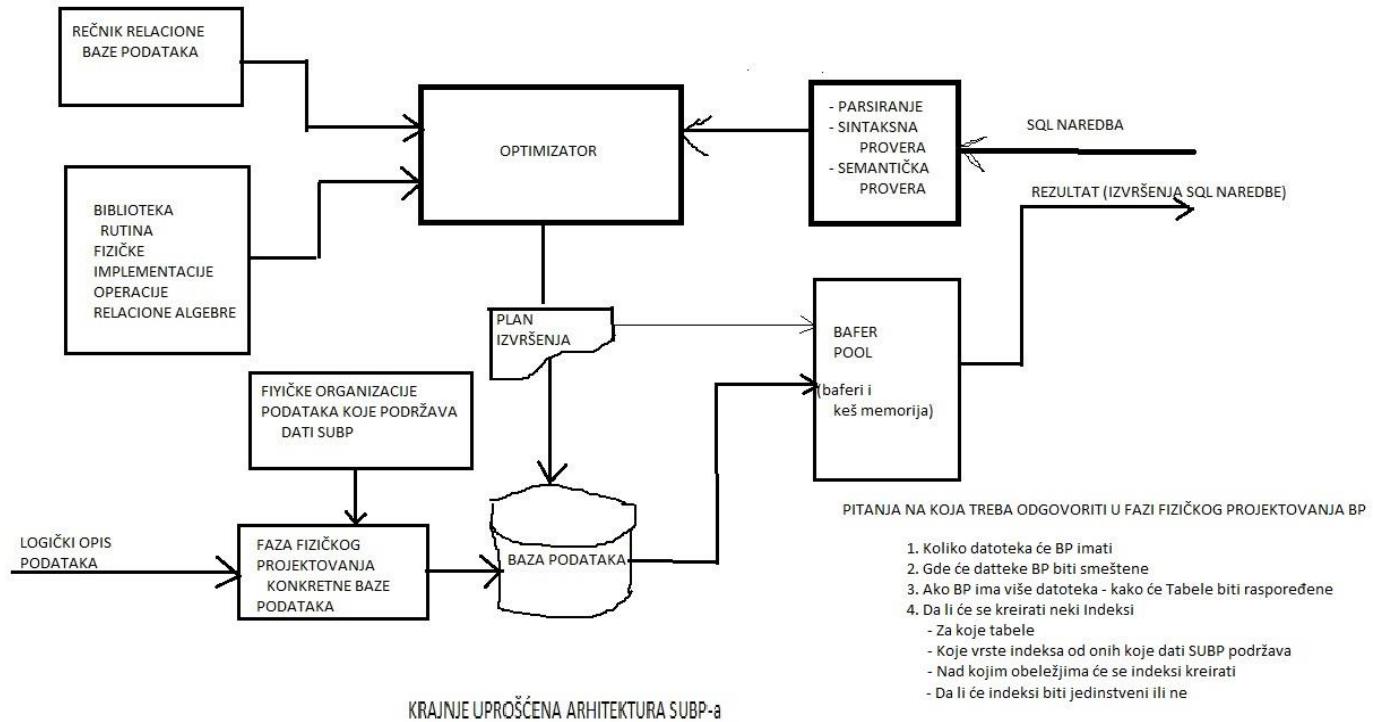
- Pogled je “prozor” kroz koji se vide podaci baze podataka
 - Pojednostavljuje koršćenje baze podataka
 - Može se korititi za zaštitu podataka od neovlašćenog pristupa
 - Sa aspekta performansi – pogledi se čuvaju u kompiliranom obliku
- Pogled se može posmatrati kao pdšema neke šema Relacione baze podataka.

Sintaksa naredbe za kreiranje pogleda

```
CREATE VIEW <naziv_pogleda>
[(naziv_obi1 [, naziv_obi2] , . . .)]
AS Podupit
```

- Podupit – Standardna Select SQL naredba (koju smo obradili iz predmeta Osnove baza podataka)

Pogled (VIEW) - šta je suštinski



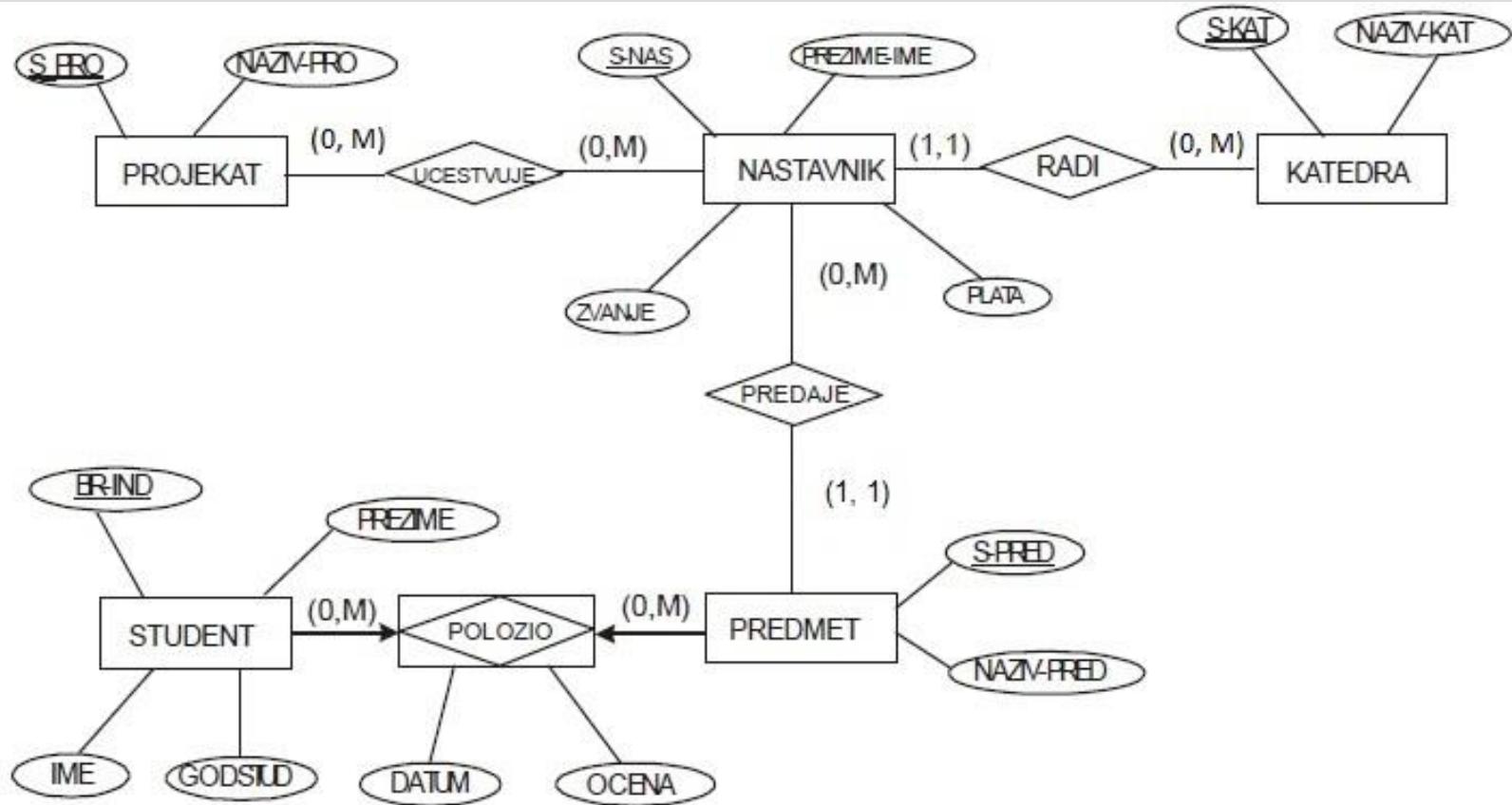
Pogledi (View) - primer

Kreirati pogled NastavnikPredmet kroz koji se "vide":

IdNas	Prezime	Ime	Naziv	FondCasova
-------	---------	-----	-------	------------

1. Formiranje i testiranje upita
2. Kreiranje pogleda
3. Korišćenje pogleda

Konceptualna šema Baze podataka Fakultet



Kreiranje pogleda - NastavnikPredmet

1. Kreiranje podupita

```
Select A.S_Nas IdNas, Prezime_Ime, Naziv_Pred,  
Casova As 'Fond casova'  
From Nastavnik A, Predmet B  
Where A.S_Nas = B.S_Nas
```

Kreiranje pogleda - NastavnikPredmet

Kreiranje pogleda

```
CREATE VIEW NastavnikPredmet
```

```
AS
```

```
Select A.S_Nas IdNas, Prezime_Ime,  
Naziv_Pred,
```

```
    Casova As 'Fond casova'
```

```
From Nastavnik A,Predmet B
```

```
Where A.S_Nas = B.S_Nas
```

Kreiranje pogleda

The screenshot shows the SSMS interface. The Object Explorer on the left lists the database structure, including databases like DESKTOP-4HPGDHI, tables such as Nastavnik and Predmet, and views like NastavnikPredmet. The central window displays a T-SQL script for creating a view:

```
CREATE VIEW NastavnikPredmet
AS
Select A.S_Nas IdNas, Prezime_Ime, Naziv_Pred,
Casova As 'Fond casova'
From Nastavnik A,Predmet B
Where A.S_Nas = B.S_Nas
```

Korišćenje pogleda

- Pogledi se koriste kao i bazne tabele.
- Korisnik ne zna da li koristi baznu tabelu ili pogled (odnosno izvedenu tabelu).

Na primer:

```
Select *
  From
    NastavnikPredmet
```

Uklanjanje pogleda

DROP VIEW <Naziv_Pogleda>

Ukloniti iz baze pogled pod nazivom
NastavnikPredmet.

Drop View NastavnikPredmet

Proceduralni mehanizmi za kontrolu integriteta

Proceduralni mehanizmi za kontrolu integriteta relacione baze podataka

- Proceduralni mehanizmi za proveru uslova integriteta se, u sistemima za upravljanje bazom podataka, najčešće realizuju korišćenjem
 - Okidača (*trigger*),
 - Procedura (*stored procedures*) i
 - Funkcija baze podataka (*user defined functions*)

Okidači (trigeri)

- Okidač je mehanizam SUBP koji se automatski pokreće i izvršava svaki put kada se izvrši odredjena operacija ažuriranja nad tabelom za koju je vezan triger.
- Okidači se najčešće koriste za realizaciju poslovnih pravila koja se tiču provere integriteta podataka i/ili obezbedjenja konzistentnog stanja baze podataka

Okidači (trigeri)

- Pri kreiranju okidača obavezno se navode predmet i vrsta ažuriranja koja bi trebala da aktivira okidač.
- Predmet ažuriranja je uvek neka tabela (ili pogled), a pod vrstom ažuriranja se podrazumevaju različite operacije ažuriranja, kao što su unos, modifikacija ili brisanje podataka.
- Uobičajeno je da se za svaku operaciju ažuriranja nad nekom tabelom definiše poseban okidač, mada je teoretski moguće koristiti isti okidač za sve tri operacije ažuriranja.
- Ono što ne može, jeste da se za istu operaciju ažuriranja nad jednom tabelom definiše više okidača.

Okidači (trigeri)

- Okidač obično služi za proveru ili izmenu podataka pomoću odgovarajućih SQL iskaza i ne bi trebao da vraćati neku vrednost(i) korisniku.
- DDL naredbe koji se koriste za rad sa okidačima su:
 - CREATE TRIGGER,
 - ALTER TRIGGER i
 - DROP TRIGGER

Okidači (trigeri)

```
CREATE TRIGGER trigger_name
ON { table | view }
{ {FOR | AFTER | INSTEAD OF } { [DELETE] [, ]
[INSERT] [, ] [UPDATE] }
AS
[ { IF UPDATE ( column )
[ { AND | OR } UPDATE ( column ) ]
[ ...n ] } ]
sql_statement [...n ]
}
```

Okidači (trigeri)

FOR | AFTER | INSTEAD OF

- Ključne reči koje određuju vreme aktiviranja okidača u odnosu na operaciju žuriranja. Navodi se jedna od tri opcije.

Okidači (trigeri)

FOR | AFTER

- FOR i AFTER imaju isti efekat.
- Ključna reč FOR se koristi zbog kompatibilnosti sa prethodnim verzijama SUBP.
- Danas se u osnovi razlikuju tri vrste okidača: AFTER, BEFORE i INSTEAD OF od kojih SQL - Server podržava samo 2 vrste (AFTER i INSTEAD OF) .
- AFTER okidač se aktivira tek nakon uspešnog izvršenja operacija ažuriranja. Ukoliko se pojavi neki problem pri izvršenju operacije ažuriranja okidač se neće ni pokrenuti.

Okidači (trigeri)

INSTEAD OF

- Govori da će se okidač izvršavati umesto okidajućeg SQL iskaza.
- Koristi se kada se izvršenje operacije ažuriranja uslovljava ispunjenošću nekih preduslova, pa se u telu okidača prvo proverava ispunjenost ovih uslova pa tek onda izvršava sama operacija ažuriranja.
- Ovo ažuriranje se vrši na osnovu sadržaja logičkih tabela koje se automatski formiraju pri aktiviranju okidača.

Okidači (trigeri)

[DELETE] [,] [INSERT] [,] [UPDATE]

- Ključne reči koje specificiraju koje operacije ažuriranja podataka nad tabelom ili pogledom aktiviraju okidač.
- Najmanje jedna opcija mora biti navedena.

AS

- Ključna reč koja označava početak tzv. tela okidača u kome se navode akcije koje okidač treba da preduzme pri sopstvenom aktiviranju.

Okidači (trigeri)

Telo okidača

sql_statement

- Okidači mogu sadržati proizvoljan broj i vrste Transact - SQL iskaza. Pored generičkih SQL iskaza (DDL i DML iskazi) u okidaču se mogu koristiti i naredbe kontrole toka kao i druge naredbe svojstvene struktuiranim programskim jezicima (naredbe dodeljivanja vrednosti (SET), ...).
- Rad sa okidačima je povezan sa korišćenjem specijalnih (implicitnih) tabela:
 - deleted i
 - inserted

Okidači (trigeri)

- deleted i
- inserted
- su logičke (konceptualne) tabele. One su strukturalno identične tabeli nad kojom je okidač definisan i sadrže stare ili nove vrednosti n-torki koje su obuhvaćene operacijom ažuriranja koja je aktivirala okidač.
- Na primer, da bi se pretražile sve vrednosti u deleted tabeli, treba koristiti sledeću naredbu:

Primer:

```
SELECT * FROM deleted
```

Okidači (trigeri)

IF UPDATE (column)

- Testira da li je izvršena INSERT ili UPDATE operacija nad specificiranim kolonom. Može se navesti više kolona. UPDATE(column) se može koristiti bilo gde unutar tela okidača.
- Ukoliko postoje ograničenja definisana nad trigerovanom tabelom, ona se proveravaju posle izvršenja INSTEAD OF okidača, odnosno pre izvršenja AFTER okidača.
- Ukoliko dodje do narušavanja ograničenja, akcije INSTEAD OF okidača se poništavaju (rollback), dok se AFTER okidač uopšte ne izvršava.

Proceduralna proširenja SQL-a

- ❑ Deklaracija promenljivih
- ❑ Naredbe za kontrolu toka programa
 - IF..... ELSE naredba
 - WHILE naredba
- ❑ Kursor i kursorski tip promenljive
 - Deklarisanje Kursora (DECLARE naredba)
 - Otvaranje kursora (OPEN naredba)
 - Čitače podataka iz kursora (FETCH naredba)
 - Zatvaranje kursora (CLOSE naredba)
 - Uklanjanje ukursora (DEALLOCATE naredba)

Deklaracija promenljivih

```
DECLARE {{ @local_variable data_type }
         | { @cursor_variable_name CURSOR }
         } [ ,...n]
```

- Promenljivama se vrednost dodeljuje pomoću SET ili SELECT iskaza.
- Kursorske varijable se mogu koristiti samo u iskazima koji se tiču rada sa kursorima.
- Nakon deklaracije sve varijable imaju NULL vrednost.

Naredbe za kontrolu toka

```
IF Boolean_expression
    { sql_statement | statement_block }
[ ELSE
    { sql_statement | statement_block } ]
```

Boolean_expression

- Je izraz koji vraća TRUE ili FALSE. Ukoliko sadrži SELECT iskaz isti se mora navesti u zagradama.

Naredbe za kontrolu toka

{sql_statement | statement_block}

- Ukoliko se IF ili ELSE grana odnosi samo na jedan SQL izkaz ovaj izkaz se može navesti odmah iza IF ili ELSE.
- Ukoliko se u nekoj grani mora izvršiti više od jedne naredbe ove naredbe treba grupisati kao blok izkaza.
- Blok izkaza se definiše pomoću ključnih reči BEGIN (za označavanje početka) i END (za označavanje završetka bloka).

Kursor i kursorski tip promenljive

- Postoje situacije kada je posle izvršenog pretraživanja rezultat potrebno obradjivati red po red.
- Sistemi za upravljanje bazama podataka u svojim proceduralnim proširenjima SQL-a omogućavaju deklarisanje kursora ili kursorskih promenljivih koje obezbeđuju prihvatanje rezultata kada on sadrži više od jedne n-torke.
- Otvaranje kursora nad skupom rezultata omogućava obradu podataka red po red.

Deklarisanje kursora

```
DECLARE cursor_name CURSOR  
[ LOCAL | GLOBAL ]  
[ FORWARD_ONLY | SCROLL ]  
[ STATIC | KEYSET | DYNAMIC |  
FAST_FORWARD ]  
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC  
]  
[ TYPE_WARNING ]  
FOR select_statement  
[ FOR UPDATE [ OF column_name [ ,...n ] ] ]
```

Cursor

```
DECLARE cursor_name [ INSENSITIVE ] [  
    SCROLL ] CURSOR  
FOR select_statement
```

- Rezultat bilo koje SELECT naredbe predstavlja neki skup n-torki.
- Ukoliko rezultujući skup čini samo jedna n-torka, tada će vrednost promenljivih, za koje je u SELECT klauzuli specificirana dodela vrednosti, biti jednaka vrednosti odgovarajućih izraza u rezultujućoj n-torci.

Cursor

- Ako je rezultujući skup n-torki prazan skup, tada se promenljivima dodeljuje NULL vrednost.
- Međutim, ako se rezultujući skup sastoji od više n-torki, tada će vrednost promenljivih po izvršenju SELECT naredbe biti jednaka vrednosti odgovarajućih izraza u poslednjoj selektovanoj n-torci.
- Na ovaj način se gubi informacija o vrednostima izraza za sve ostale n-torke rezultujućeg skupa.
- Ukoliko je neophodno znati vrednosti za sve selektirane n-torke, a postoji mogućnost da rezultujući skup čini više n-torki, koriste se kurzori.

Cursor

Rad sa kursorima obično podrazumeva sledeće korake:

1. **DECLARE** - Deklaracija kursora pri kojoj se kursoru pridružuje neka SELECT naredba i definišu karakteristike kursora (npr. da li se n-torke u kursoru mogu modifikovati ili ne).
2. **OPEN** - Otvaranje kursora koje podrazumeva izvršavanje pridruženog SELECT iskaza i popunjavanje kursora n-torkama rezultujućeg skupa.
3. **FETCH** - Pregled n-torki u kursoru (pristup narednoj ili, eventualno, prethodnoj n-tortki se naziva fetch-om).
4. Opciono, modifikaciju tekuće n-torke
5. **CLOSE** - Zatvaranje kursora.

Cursor

INSENSITIVE

- Definiše kurzor koji pravi privremenu kopiju podataka koje koristi kurzor.
- Kurzor se nakon punjenja isključivo koristi podacima iz ove privremene tabele (koja se kreira u tempdb bazi), te se izmene koje su u međuvremenu učinjene nad baznim tabelama ne reflektuju na podatke koji se dobijaju pregledom kursora.
- Ovakav kurzor ne dozvoljava modifikaciju n-torki rezultujućeg skupa.
- Ukoliko se izostavi ključna reč INSENSITIVE, (commit-ovana) brisanja i modifikacije nad baznim tabelama će se reflektovati pri narednim fetch-evima.

Cursor

SCROLL

- Specificira da su sve fetch opcije (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE) na raspolaganju. Ukoliko nije naveden SCROLL jedina podržana fetch opcija je NEXT.

select_statement

- Je standardni SELECT izraz koji definiše rezultujući skup kursora.

READ ONLY

- Onemogućuje vršenje modifikacija na osnovu sadržaja kursora (koje je, inače, dozvoljeno po default-u).
- Kursor se ne može referencirati u WHERE CURRENT OF klauzuli UPDATE ili DELETE izraza.

Cursor

UPDATE [OF column_name[,...n]]

- Definiše kolone koje se mogu modifikovati unutar kursora.
- Ukoliko se koristi OF column_name [,...n], modifikacije je moguće vršiti samo nad navedenim kolonama. Ukoliko se navede samo FOR UPDATE bez navodjenja liste kolona, sve kolone se mogu modifikovati

Cursor

OPEN cursor_name

- Otvaranje kursora podrazumeva izvršavanje SQL naredbe definisane prilikom deklaracije kursora i podrazumeva popunjavanje kursora n-torkama rezultujućeg skupa izvršene SQL naredbe.
- Po otvaranju kursora aktuelna n-torka je prva n-torka rezultujućeg skupa.

Cursor

FETCH

```
[ [ NEXT | PRIOR | FIRST | LAST ]  
{ { [ GLOBAL ] cursor_name } | @cursor_variable_name }  
[ INTO @variable_name [ ,...n ] ]
```

FETCH omogućava pristup n-torkama koje se nalaze u kurzoru.

- Pomoću FETCH naredbe možemo biti sigurni da će svaka n-torka koja se nalazi u kurzoru proći odredjene modifikacije zadate u telu procedure ili trigera.

NEXT predstavlja defaultnu fetch opciju.

- Podrazumeva prisutnje svakoj n-torci respektivno. Prvom fetch-om pristupa se prvoj n-torci, zatim drugoj... Postupak se ponavlja sve dok se ne dodje do poslednje n-torke.

Cursor

PRIOR - podrazumeva prisutp, obradu, predhodne n-torke.

- Kursor se pozicionira tako da uvek pristupa predhodnoj n-torci i za nju daje rezultate. U slučaju da se radi o prvom fetch-u (pristup prvoj n-torci) kao rezultat promenljive će imati null vrednost i kursor će se pozicionirati na prvu n-torku. Tek pristupom drugoj n-torci, doći će do obrade prve n-torke.

FIRST – pozicioniranje na prvu n-torku u kursoru.

LAST – pozicioniranje na poslednju n-torku u kursoru

Cursor

CLOSE {cursor_name | cursor_variable_name}

- Zatvara otvoren kursor što podrazumeva pražnjenje kursora (oslobadja se memorijski prostor koji se koristio za čuvanje podataka kursora), ali njegova deklaracija ostaje i dalje aktuelna, te se zatvoren kursor uvek može ponovo otvoriti.

DEALLOCATE { cursor_name | @cursor_variable_name }

- Poništava deklaraciju kursora. Ukoliko se ovo ne bi uradilo prilikom sledećeg aktiviranja procedure ili okidača bi nastao problem usled nemogućnosti ponovne deklaracije kursora sa istim nazivom.

Cursor

@@FETCH_STATUS

- Predefinisana (sistemska) varijabla koja vraća status poslednjeg FETCH iskaza nad aktivnim kursorom.

Return vrednost	Opis
0	FETCH iskaz je bio uspešan.
-1	FETCH iskaz nije uspeo ili je n-torka izvan rezultujućeg skupa.
-2	Fetch-ovana n-torka ne postoji.

WHILE naredba

```
WHILE Boolean_expression
    { sql_statement | statement_block }
    [ BREAK ]
    { sql_statement | statement_block }
    [ CONTINUE ]
```

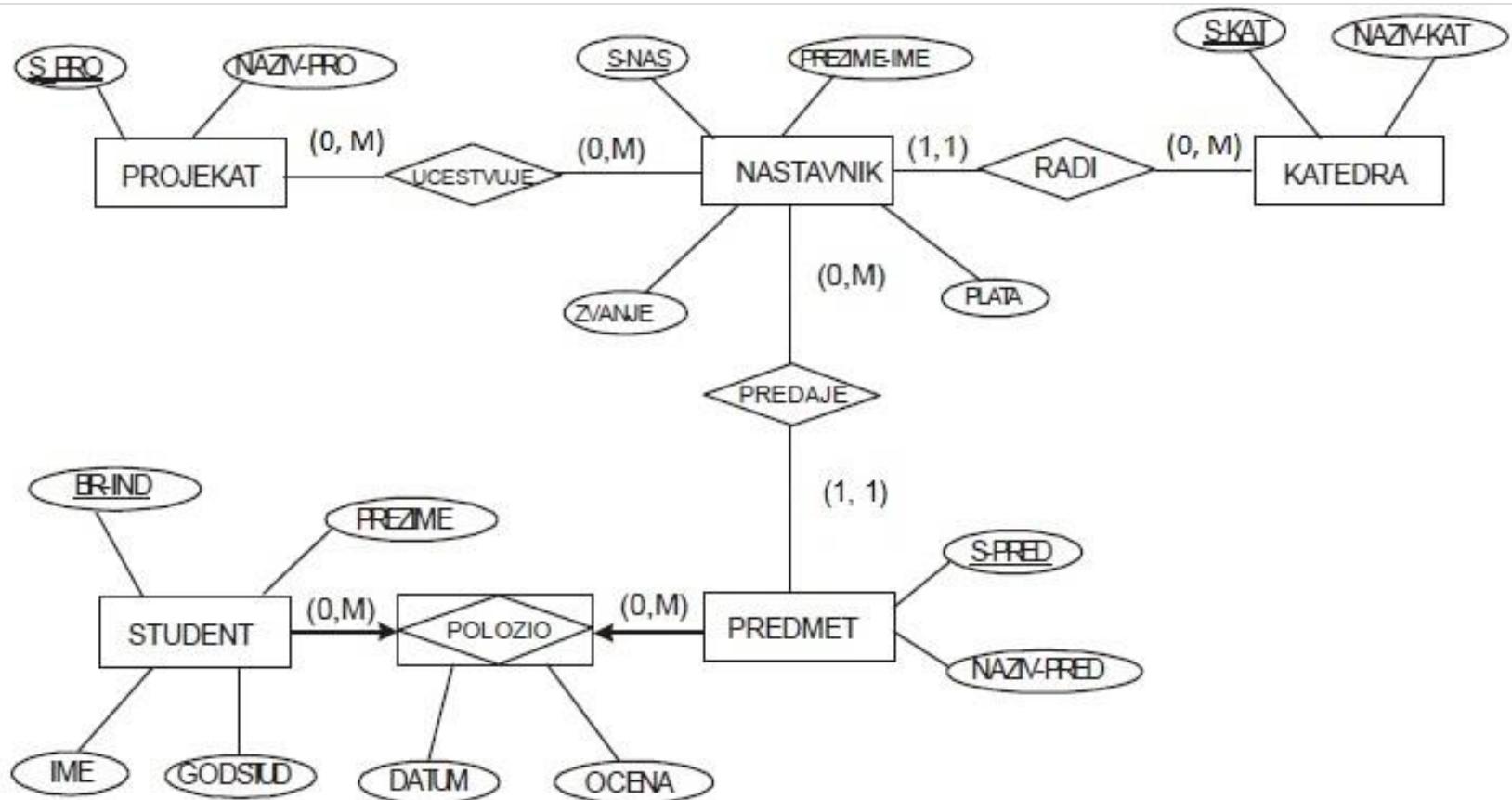
- Izvršenje bloka koda se ponavlja sve dok je uslov ispunjen.
- U nekim situacijama postoji potreba za postavljanjem dodatnih uslova unutar bloka koda koji mogu dovesti do nasilnog prekida izvršavanja WHILE ciklusa (BREAK).
- U slučaju navođenja iskaza CONTINUE nasilno se prekida dalje izvršenje ostatka bloka koda i inicira se ponovna provera ispunjenosti opšteg uslova izvršenja ciklusa

CURSOR - Zadatak

Zadatak: Koristeci kurzor napraviti proceduru kojom se iz baze podataka Fakultet prikazuje tabela sa sledecim zaglavljem:

Prezime	Ime	BrIsPita
---------	-----	----------

ER Model - BP Fakultet



CURSOR – Zadatak - Resenje

```
Declare @Prezime Varchar(20), @Ime Varchar(20), @BrIspita int;
```

```
DECLARE PolozeniIspiti_Cursor SCROLL CURSOR FOR
SELECT Prezime, Ime, Count(*) As BrIspita
FROM Student A, Polozio B
WHERE A.Br_Ind = B.Br_Ind
Group By Prezime, Ime
Order By BrIspita Desc
```

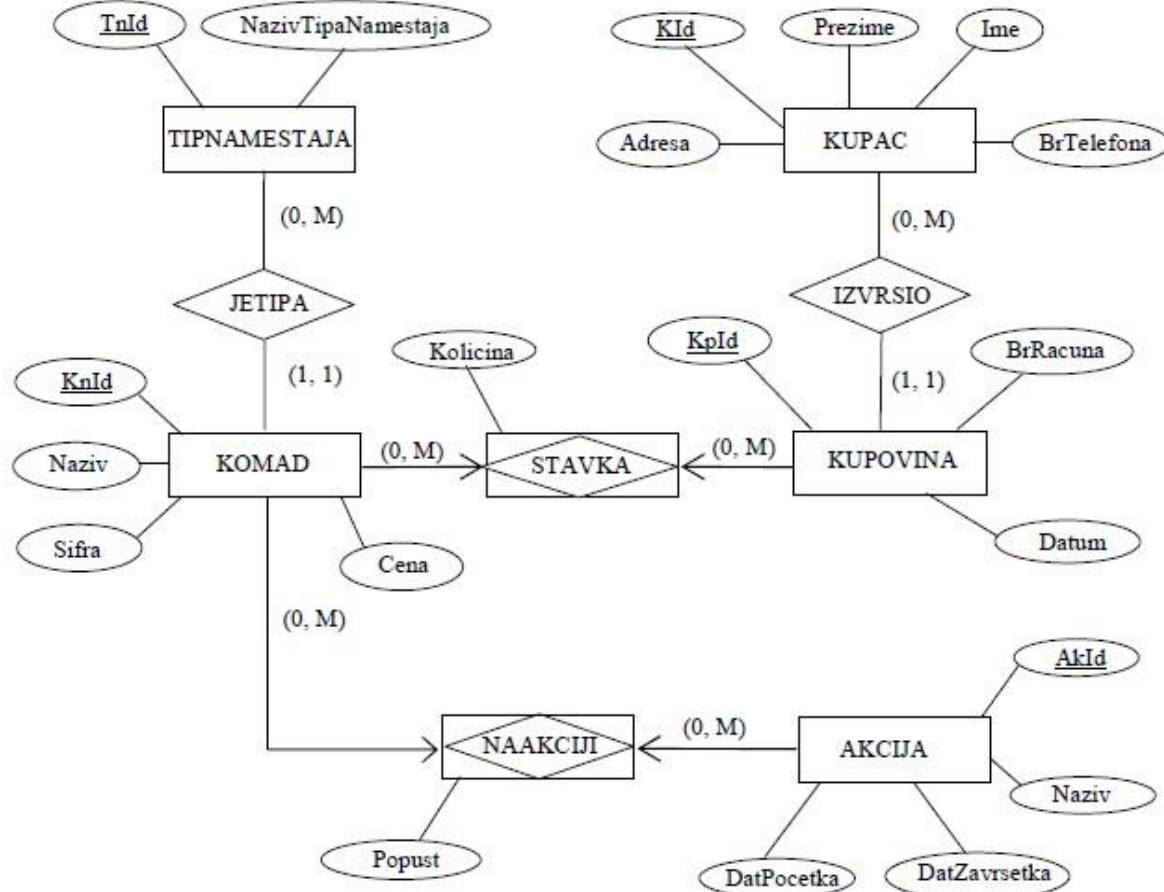
CURSOR – Zadatak - Resenje

```
OPEN PolozeniIspiti_Cursor;
Set @Prezime='Prezime
'
Set @Ime='Ime
'
Print @Prezime+' '+@Ime+' '+BrIspita'
FETCH PolozeniIspiti_Cursor INTO @Prezime, @Ime, @BrIspita;
Print '-----'
Print @Prezime+' '+@Ime+' '+Convert(Varchar,@BrIspita)
WHILE @@FETCH_STATUS = 0
BEGIN
    FETCH NEXT FROM PolozeniIspiti_Cursor
        INTO @Prezime, @Ime, @BrIspita
    Print @Prezime+' '+@Ime+' '+Convert(Varchar,@BrIspita)
END;
```

Trigger – primer

Zadatak 1: Kreirati triger Komad_Ins za proveru ispunjenosti referencijalnog uslova integriteta u odnosu na relaciju TipNamestaja, pri dodavanju n-torke u relaciju Komad.

Konceptualna šema BP - SALON



Trigger – primer – (Rešenje - Zadatak 1)

```
Create Trigger Komad_Ins
    On Komad After Insert
As
If (Select Count(*)
    From TipNamestaja A, Inserted
    Where A.TnId = Inserted.TnId) = 0
Begin
    Raiserror ('Vrednost TnId ne postoji u tabeli
TipNamestaja.', 16, 1)
    Rollback Transaction
    Return
End
Return
```

Triger - Primer

Zadatak 2: Kreirati triger TipNamestaja_Del za proveru ispunjenosti referencijalnog uslova integriteta u odnosu na relaciju Komad, pri brisanju n-torke u relaciju TipNamestaja.

Zadatak resiti tako da se, ukoliko se brise n-torka iz tabele TipNamestaja za koju postoje n-torke u tabeli Komad, "kaskadno" brisu i sve n-torke iz tabele Komad koje se referisu na n-torku Tabele TipNamestaja koja se brise.

Triger - Primer

```
Create Trigger TipNamestaja_Del  
On TipNamestaja After Delete  
As  
Delete From Komad  
Where TnId In (Select TnId From Deleted)
```

Triger – Primer 3

Zadatak 3: Kreirati triger nad tabelom Komad koji ce, pri brisanju n-torki iz tabele, prikazati sve obrisane n-torke.

Triger – Primer 3 - Resenje

```
Create Trigger PrikaziOrisano  
On Komad After Delete  
As  
Select *  
From Deleted
```

TRIGERI

```
CREATE TRIGGER trigger_name
ON { table | view }
{ {FOR | AFTER | INSTEAD OF} { [DELETE] [,]
    [INSERT][,][UPDATE] }
AS
[ { IF UPDATE ( column )
    [{ AND | OR } UPDATE ( column )]
    [ ...n ] } ]
sql_statement [...n ]
}
```

Trigger - primer

```
Create Trigger Komad_Ins  
On Komad After Insert  
As  
If (Select Count(*)  
    From TipNamestaja A, Inserted  
    Where A.TnId = Inserted.TnId) = 0  
Begin  
    Raiserror ('Vrednost TnId ne postoji u tabeli  
TipNamestaja.', 16, 1)  
    Rollback Transaction  
    Return  
End  
Return
```

Funkcje - Korisničke

- Kao i funkcije u programskim jezicima, SQL Server korisnički definisane funkcije su rutine koje prihvataju parametre, izvršavaju različite iskaze i vraćaju rezultate tih iskaza kao vrednosti.
- Povratna vrednost može biti jedna skalarna vrednost ili skup rezultata.
- Postoje sistemske i korisnički definisane funkcije baze podataka.
- Sistemske funkcije su uskladištene u master bazi i ne mogu se menjati.
- Kao primeri sistemskih funkcija mogu se navesti funkcije COUNT, SUM, ROUND, GETDATE,...

Funkcje

Bilo da se radi o sistemskim ili korisnički definisanim funkcijama, funkcije se mogu podeliti na:

a)

Skalarne - korisnički definisane skalarne funkcije vraćaju jednu vrednost podataka kao rezultat funkcije. Ukoliko telo funkcije sadrži više od jednog iskaza isti se moraju navesti kao blok iskaza. Povratna vrednost može biti bilo koji tip podataka, osim teksta, nteksta, slika, cursora i vremenskih tipova podataka.

b)

Table valued funkcije – predstavljaju korisnički definisane funkcije koje vraćaju tabele kao rezultat rada funkcije.

DDL naredbe za definisanje skalarne funkcije

```
CREATE FUNCTION NazivFunkcije
  ( [@parametar1 datatype1 [ = default ]
    [, @parametar2 datatype2 [ = default ]] [,..n] ]
    -- Ovde se dodaju ostali parametri funkcije
)
RETURNS Data_Type - Tip povratne vrednosti
AS
BEGIN
  -- Telo funkcije
  -- Ovde se dodaje kod tela funkcije
RETURN PovratnaVrednostFunkcije;
END;
```

Funkcje

Primer funkcije koja vraca skalarnu vrednost:

```
CREATE FUNCTION nVrednost (@Puta TinyInt)
    RETURNS Int
AS
BEGIN
    DECLARE @Umnozak Int
    SET @Umnozak = @Puta * 5
    RETURN (@Umnozak)
END
```

Funkcje

Primer poziva (koriscenja) funkcije koja vraca skalarnu vrednost:

```
Select dbo.nVrednost(2)
```

Funkcija vraca vrednost $5^2 = 10$

DDL naredbe za definisanje tabelarne funkcije

```
CREATE FUNCTION NazivFunkcije  
    (@Parametar1 TipPodatka, @Parametar2 tipPodatka)  
RETURNS TABLE  
AS  
RETURN  
(  
    SELECT Elm-Selekcije-1, Elm-Selekcije-2....  
        FROM NazivTabele  
        WHERE Uslov  
)
```

Funkcje – Primer Table valued funkcije

Primer funkcije koja kao povratnu vrednost vraca tabelu:

```
CREATE FUNCTION funIspitiStudenta (@Indeks char(6))
RETURNS TABLE
AS
RETURN (Select NazivPredmeta,
Convert(varchar, Datumispita, 104) As DatumIspita, Ocena
From StudentPolozio
Where BrInd = @indeks)
```

Funkcje – Primer poziva/koriscenja **Table valued funkcije**

Koristeći funkciju funIspitiStudenta prikazati položene ispite studenta s brojem indeksa 'E 7398'.

```
Select *
  From dbo.FunIspitiStudenta ('E 7398')
```

Funkcije - Zadaci

Zadatak 1:

Kreirati funkciju koja vraća broj položenih ispita za studenta sa zadatim brojem indeksa. Funkciju nazvati:

PolozenoIspita.

Funkcije - Zadaci

Create Function PolozenoIspita (@BrInd Char(6))
Returns Integer

AS

Begin

 Declare @BrIspita Int

 Select @BrIspita = COUNT(*)

 From Polozio

 Where Br_Ind = @BrInd

 Return @BrIspita

End

Procedure - Funkcije - Zadaci

Zadatak 2:

Kreirati funkciju ProsečnaOcena koja vraća prosečnu ocenu tokom studija za studenta sa zadatim brojem indeksa.

Funkcije - Zadaci

```
Create FUNCTION ProsecnaOcena (@BrInd Char(6))
RETURNS Decimal(4,2)
AS
BEGIN
    Declare @Prosecna Decimal(4,2)
    Select @Prosecna = AVG(Cast(Ocena As Decimal(4,2)))
        From Polozio
        Where Br_Ind = @BrInd
    Return @Prosecna
END
```

Funkcije - Zadaci

Zadatak 3:

Koristeci funkcije PolozenoIspita iProsecnaOcena kreirati uskladistenu proceduru SpisakStudenata koja vraca: Prezime, Ime studenta, Broj položenih ispita i prosecnu ocenu studija za svakog studenta koji je polozio najmanje jedan ispit.

Funkcije - Zadaci

```
Select Prezime, Ime, dbo.PolozenoIspita(Br_Ind)
    As Polozeno, dbo.ProsecnaOcena(Br_Ind)
    AsProsecnaOcena
From Student
Where dbo.PolozenoIspita(Br_Ind) > 0
Order By Polozeno Desc
```

Procedure - Funkcije - Zadaci

Zadatak: Koristeci kurzor napraviti Korisnicku uskladistenu proceduru pod nazivom Spisak kojom se iz baze podataka Student prikazuje tabela sa sledecim zaglavljem:

Prezime	Ime	BrIspita
---------	-----	----------

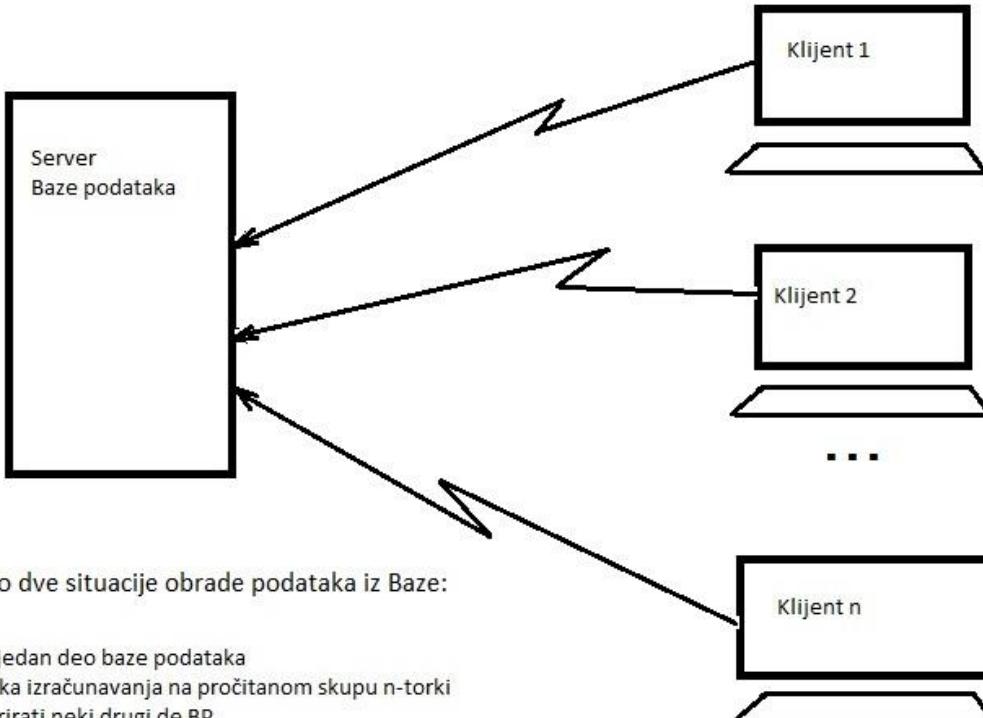
Uskladištene procedure

- Korisničke
- Sistemske

Uskladištene procedure

- Uskladištene procedure se mogu definisati kao kolekcija Transact - SQL izkaza koje mogu da vrate odredjene rezultate na osnovu prosledjenih parametara od strane korisnika.
- Uskladištene procedure se prevashodno koriste za neke obrade koje najčešće podrazumevaju čitnje podataka iz jednog dela BP, izračunavanja i ažuriranje sadržaja nekog drugog dela BP (obračun kamata na bankarske račune, obračun zarada,...).
- Ovi postupci obrade se mogu vršiti i pomoću programa pisanim u nekom od programskih jezika koji se izvršavaju na klijentima ali je prednost korišćenja uskladištenih procedura u tome što se one u nekim situacijama obrade (ali ne svim) mnogo brže izvršavaju.

Prednosti korišćenja Uskladištenih Procedure



Analizirajmo dve situacije obrade podataka iz Baze:

1. Treba

- Pročitati jedan deo baze podataka
- Izvršiti neka izračunavanja na pročitanom skupu n-torki
- Zatim ažurirati neki drugi deo BP

2. Štampa nekog izveštaja na osnovu sadržaja BP

Primer 2: Korišćenje Uskladištene procedure ne donosi prednosti

Primer 2: Opravdano korišćenje Uskladištene procedure jer se smanjuje mrežni saobraćaj u oba smera i od servera BP i ka serveru BP. Osim toga obrada se vrši u adresnom prostoru servera BP koji je po pravilu mnogo jača mašina.

- 1. - Prenos pocetaka (rezultujućih n-torki) kroz mrežu na klijenta
- Obrada/Izračunavanje se vrši na strani klijenta
- Vrši se prenos kroz mrežu pravcs podataka za ažuriranje

Uskladištene procedure

- Kod procedura baza podataka postoje dve vrste parametara:
 - Ulazni (input) i
 - Izlazni (output) parametri.
- Ulazni parametri su korisnički prosledjeni parametri koji služe za zadavanje uslova u samom telu procedure. Pomoću njih zadaju se odredjeni uslovi koji će izdvojiti n-torce za obradu.
- Izlazni parametri su parametri koji se vraćaju izvršavanjem tela procedure. Oni predstavljaju prikaz rezultata obrade podataka pomoću uskladištenih procedura.

Uskladištene procedure

Postoje dve osnovne vrste procedura

- Sistemske procedure i
- Korisnički definisane procedure baze podataka.

Uskladištene procedure

Sistemske procedure:

- U SQL Serveru, mnoge administrativne aktivnosti mogu se obavljati pomoću sistemskih procedura.
 - Svaki put kada se doda ili izmeni tabela, napravi rezervna kopija baze podataka (backup), kreira login ili korisnik, dodeljuju korisnička prava ili obavljaju druge administratorske aktivnosti , zapravo se pozivaju sistemske procedure specijalno napisane da izvrše željenu aktivnost.
 - Sistemske procedure imaju prefiks ***sp_*** i čine deo SQL Server instalacije. Čuvaju se u ***master*** bazi podataka i dostupne su za korišćenje u bilo kojoj bazi podataka konkretnе instance SQL Server-a.
-

Uskladištene procedure

Korisnički definisane procedure:

- Korisnički definisana procedura je svaki program napisan od strane Administratora baze podataka u okruženju SUBP-a uz korišćenje SQL naredbi i njegovih proceduralnih proširenja.
- Najčešći zadatak korisničkih uskladištenih procedura je ažuriranje sadržaja jednih tabela na osnovu obradjenog sadržaja drugih tabela baze podataka.
- Za razliku od sistemskih procedura, korisnički definisane procedure se čuvaju u lokalnoj bazi i samo su u njoj dostupne za korišćenje.

Uskladištene procedure – Osnovna sintaksa

```
CREATE PROC [ EDURE ] procedure_name [ ;number ]
    [ { @parameter data_type } [ OUTPUT ] ]
    [ ,...n ]
AS sql_statement [ ...n ]
```

procedure_name – definisanje naziva procedure koja se kreira

@parameter data_type – navodjenje parametara koji će se koristiti u telu procedure

OUTPUT – naglašavanje output parametra

AS - ključna reč koja služi za označavanje početka tela procedure

Uskladištene procedure

- Svi Transact - SQL izrazi koji se mogu navesti u telu okidača i funkcija stoje na raspolaganju i pri pisanju tela uskladištene procedure.
- Za razliku od okidača (*Trigger-a*) kod procedura ne postoje inserted i deleted logičke tabele, jer se procedure kao takve ne vezuju za konkretnu tabelu..

Primer - Uskladištene procedure

ZADATAK: Napisati uskladištenu proceduru ProcPolozeniIspiti koja za studenta sa zadatim brojem indeksa prikazuje položene ispite studenta, i to: NazivPredmeta, DatumPolaganja i dobijenu Ocenu.

```
Create Procedure ProcPolozeniIspiti @BrojIndeksa Char(10)
As
    Select NazivPredmeta, FORMAT(Datum, 'dd.MM.yyyy') As Datum, Ocena
        From PolozeniIspiti
            Where BrInd = @BrojIndeksa
```

Primer poziva uskladištene procedure

-- Primer> Poziv procedure koja ima ulazne parametre:

1. Nacin:

```
Exec ProcPolozeniIspiti @BrojIndeksa = 'E 7398'
```

2. Nacin:

```
Exec ProcPolozeniIspiti 'E 7398'
```

Ako ima vise ulaznih parametara pri pozivu procedure parametri se razdvajaju zarezom.

Uskladištene procedure - Primer -

ZADATAK: Koristeci cursor napraviti proceduru

Procedura _SpisakStudenata kojom se
iz baze podataka Fakultet prikazuje tabela sa sledećim
Zaglavljem.

Prezime

Ime

BrIspita

Uskladištene procedure - Primer -

```
Create Procedure Procedura_SpisakStudenata
AS
    Declare @Prezime Varchar(20), @Ime Varchar(20), @BrIspita int;
    DECLARE PolozeniIspiti_Cursor SCROLL CURSOR FOR
        SELECT Prezime, Ime, Count(*) As BrIspita
        FROM Student A, Polozio B
        WHERE A.Br_Ind = B.Br_Ind
        Group By Prezime, Ime
        Order By BrIspita Desc

    OPEN PolozeniIspiti_Cursor;
    Set @Prezime='Prezime
    Set @Ime='Ime
    Print @Prezime+' '+@Ime+' '+BrIspita'

    FETCH PolozeniIspiti_Cursor INTO @Prezime, @Ime, @BrIspita;
    Print '-----'
    Print @Prezime+' '+@Ime+' '+Convert(Varchar,@BrIspita)

    WHILE @@FETCH_STATUS = 0
        BEGIN
            FETCH NEXT FROM PolozeniIspiti_Cursor INTO @Prezime, @Ime, @BrIspita
            Print @Prezime+' '+@Ime+' '+Convert(Varchar,@BrIspita)
        END;
    Close PolozeniIspiti_Cursor;
    Deallocate PolozeniIspiti_Cursor
```

ADMINISTRACIJA BAZA PODATAKA

INFORMATION SCHEMA

(Skup sistemskih pogleda za pristup metapodacima
relacionih baza podataka)

Information Schema - Uvod

- INFORMATION_SCHEMA obezbeđuje mogućnost da se na standardizovan način dođe do metapodataka u različitim SUBP-ovima.
 - Metapodaci?
 - INFORMATION_SCHEMA se nalazi u većini modernih Relacionih SUBP. Omogućava korisnicima da pristupe metapodacima o strukturi i sadržaju baze podataka.
 - **INFORMATION_SCHEMA – Nije baza podataka**, već skup sistemskih pogleda. Omogućava korisnicima da pristupe metapodacima o strukturi i sadržaju baze podataka.
-

Šta je information schema?

- INFORMATION_SCHEMA se koristi za prikazivanje informacija o objektima baze podataka kao što su tabele, pogledi, ključevi, ograničenja, funkcije, indeksi itd.
 - Administrator baze podataka (i svi korisnici kojima je da administratorska prava u okviru instance SUBP-a) mogu koristiti ovaj sistemski skup pogleda za pregled podataka koje koristi sam SUBP .
 - INFORMATION_SCHEMA se može koristiti za standardizovan pristup rečniku različitih relacionih SUBP, jer je podržava većina modernih relacionih SUBP, kao što su: SQL Server, MySQL, PostgreSQL, itd.
-

Pristup REČNIKU preko Information Schema-e

- Rečnik, odnosno sistemska baza podataka različitih SUBP, sadrži slične ili gotovo iste podatke, međutim njihova organizacija je drugačija od jednog do drugog SUBP-a.
 - Uvođenjem koncepta Information Schema-e obezbeđuje se da, se skrivajući implementacione detalje konkretnog SUBP-a, rečniku pristupa na potpuno isti funkcionalan način korišćenjem pogleda Information Scheme.
 - Pristup rečniku preko pogleda Information Scheme je potpuno isti kod svih SUBP-ova koji je podržavaju - (to je smisao i cilj uvođenja koncepta Information Schema-e)
-

Kako se koristi Informatio schema?

- ❑ U SUBP-ovima koji je podržavaju postoji skup pogleda u okviru Information Schema-e.
 - ❑ Preko tih pogleda Information schema-e se vide metapodaci o samim bazama podataka i njihovim objektima.
 - ❑ Sistemske poglede iz Information schema-e moduće je koristiti za upite i preglede sistemskih (meta) podataka, ali se preko toh pogleda metapodaci ne mogu menjati (ažurirati).
-

Zašto je uveden?

- Bio je potreban standard za pregled metapodataka u SUBP-ovima
 - Pre information schema-e su se za pregled metapodataka koristile komande.
 - Metapodacima u svim komercijalnim SUBP (koji podržavaju koncept INFORMATION_SCHEMA) pristupa se na isti način bez obzira na različitu organizaciju sistemskih baza podataka svakog SUBP.
-

Standard

Database languages - SQL Standard

- ISO/IEC 9075 standard– je standard SQL upitnog jezika
- Ukupno je usvojeno 7 standarda do sada
- Standard se deli na 14 delova

Standard

Neki od SUBP-ova koji podržavaju *Information* :

- MS SQL Server
 - od verzije 7 pa naviše
 - MySQL
 - od verzije 5 pa naviše
 - PostgreSQL
 - od verzije 7.4 pa naviše
 - SQLite
 - od verzije 3.7 pa naviše
 - Oracle?
-

Pogledi – U okviru Information Schema-e

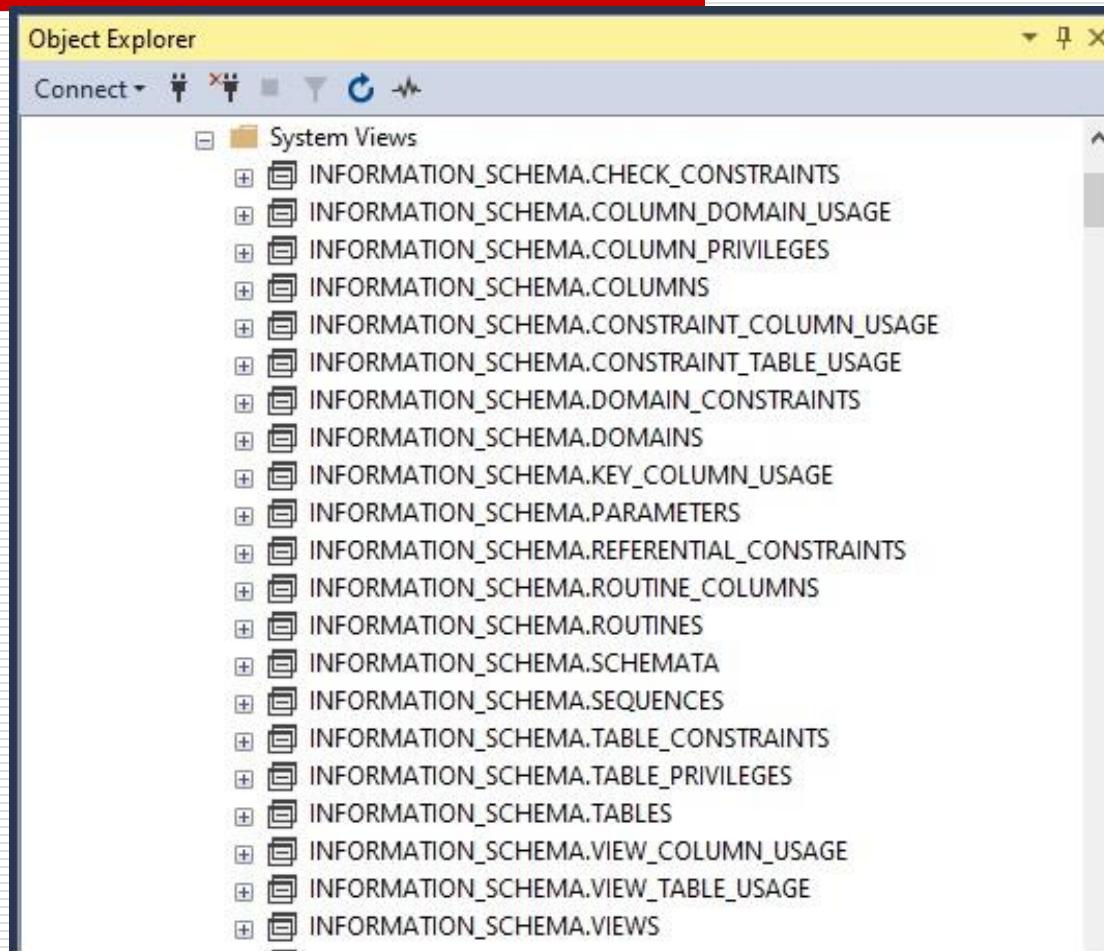
Nemaju svi SUBP-ovi isto tabela (pogleda) u Information_Schema-i

- ❑ MySQL ima vise tabela (pogleda) od ostalih SUBP-ova koji podržavaju Information Schema

Nek od pogleda su :

- *Tables , Columns , Key_Column_Usage , Views, Referential_Constraints...*
- ❑ Svi pogledi u okviru Information Schema imaju kolone kojima je opisana svaka baza podataka kojom rukuje konkretna instanca SUBP-a.
- ❑ Na primer, pogled **Tables** ima sledeće kolone:
 - *TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_TYPE*

INFORMATION SCHEMA – Skup pogleda



Primer 1 - Korišćenja Information Schema-e

Prikazati sva obeležja svih tabela baze podataka FAKULTET za koja je prilikom kreiranja šeme te baze podataka definisano da ne smeju poprimiti *NULL* vrednost.

	Column_Name	Table_Name	Is_Nullable	Data_Type
1	Br_Ind	Student	NO	char
2	Br_Ind	Polozio	NO	char
3	Budzet	Projekat	NO	money
4	Casova	Predmet	NO	int
5	Datum	Polozio	NO	datetime
6	GodStud	Student	NO	char
7	Ime	Student	NO	varchar
8	IznosHonorara	Ucestvuje	NO	money
9	Naziv_Kat	Katedra	NO	char
10	Naziv_Pred	Predmet	NO	varchar
11	Naziv_Pro	Projekat	NO	char
12	Ocena	Polozio	NO	int

Primer 1 - Korišćenja Information Schema-e

Uradićemo zadatak pomoću tabele **COLUMNS**, koja u sebi sadrži sva obeležja kolona svih tabela svih baza podataka. U sledećem primeru koristićemo kolone:

Column_name – Naziv kolone.

Table_name – Naziv tabele kojoj kolona pripada.

Is_nullable – Da li vrednost kolone sme biti null

Data_type – Tip podataka kolone.

Primer 1 - Korišćenja Information Schema-e

SQLI kod:

```
Select Column_Name, Table_Name, Is_Nullable, Data_Type  
From Information_Schema.Columns  
Where Is_Nullable = 'NO'  
Order by Column_Name
```

Primer 2 - Korišćenja Information Schema-e

Prikazati sve relacije (Tabele) u bazi podataka Hotel.

	TABLE_CATALOG	TABLE_NAME
1	Hotel	Cenovnik
2	Hotel	Gost
3	Hotel	Iznajmljivanje
4	Hotel	Soba
5	Hotel	TipIznajmljivanja
6	Hotel	TipSobe

Primer 2 - Korišćenja Information Schema-e

Udadićemo zadatak pomoću pogleda **TABLES** koja sadrži sve sve tabele aktuelne baze podataka. Iskoristićemo sledeće kolone iz pogleda:

TABLES:

Table_Catalog – Naziv baze podataka.

Table_Name – Naziv tabele u aktuelnoj bazi podataka.

Table_Type – Vrsta (tip) tabele (da li je bazna tabele ili izedena tabela, odnosno pogled)

Primer 2 - Korišćenja Information Schema-e

SQL kod:

```
Select TABLE_CATALOG, TABLE_NAME, TABLE_TYPE  
From INFORMATION_SCHEMA.Tables  
Order By TABLE_NAME
```

Zadataci

1. Koristeci INFORMATION_SCHEMA prikazati sve tabele u bazi podataka Fakultet.

 2. Koristeci INFORMATION_SCHEMA prikazati sva obelezja tabele Nastavnik iz baze podataka Fakultet.

 3. Koristeci INFORMATION_SCHEMA prikazati sve poglede u bazi podataka Fakultet.
-

Adminstracija baza podataka

Transakcije nad bazom podataka

Šta je Transakcija? (1/5)

- Transakcija je vremenski uređen niz nedeljivih radnji nad bazom podataka koje u celini ne remete uslove integrideta.
 - Transakcija je osnovna nedeljiva jedinica rada nad bazom podataka koja čini jednu logičku celinu posla.
 - Sa aspekta definisanih uslova integriteta transakcija transformiše jedno konzistentno u baze podataka u drugo takođe konzistentno stanje baze podataka.
 - Između dva konzistentna stanja dozvoljeno je da se baza podataka nađe i u nekonzistentnom stanju.
-

Šta je Transakcija? (2/5)

- Bitno je da transakcija bude tako oformljena da iz bilo kojih razloga (npr. otkaz hardvera ili softvera, višekorisnički rad ...) ne ostavi trajno bazu podataka u nekonzistentnom stanju.
 - Potrebno je obezbediti mehanizam da se transakcija u celini uspešno izvrši (obzirom da je nedeljiva), ili da se, ukoliko dođe do greške, ponište sva njena dejstva i baza podataka vrati u stanje pre početka izvršavanja transakcije.
 - Ukoliko dođe do greške u toku izvršavanja transakcije pre njenog kraja, a od početka transakcije je bilo ažuriranja, potrebno je poništiti sva ta ažuriranja i bazu podataka vratiti u stanje pre početka izvršavanja transakcije.
-

Šta je Transakcija? (3/5)

- Sa tačke gledišta uslova integriteta transakcija se ponaša ka jedinična radnja, što ne važi za pojedinačne radnje od kojih se transakcija sastoji.
 - SUBP koji podržavaju ***transakcionu obradu*** moraju da garantuju da, ukoliko se transakcijom vršilo ažuriranje baze podatakai iz bilo kojih razloga transakcija se nije normalno završila, ponište sva ažuriranja delimično izvršenih transakcija.
 - Na taj način transakcija se ili u potpunosti izvršava ili u potpunosti poništavaju njena dejstva.
 - Transakcionala obrada odvija se pod kontrolom ***administratora transakcija (transaction manager-a)*** kao dela (modula/komponente) SUBP-a čije naredbe COMMIT i ROLLBACK određuju način njegovog funkcionisanja
-

Šta je Transakcija? (4/5)

- Transakcija prevodi bazu podataka iz jednog konzistentnog stanja u drugo (ne nužno novo) takođe konzistentno stanje baze podataka.
 - Transakcije mogu biti jedinice ili sekvence rada izvršene logičkim redom, bilo ručno od strane korisnika ili automatski nekim programom koji upravlja bazom podataka.
 - Transakcija je propagiranje jedne ili više promena nad bazom podataka.
 - Na primer: Kada vršimo insert, update ili delete vrednosti iz baze, mi vršimo transakciju nad tabelom.
-

Šta je Transakcija? (5/5)

- Veoma je važno upravljati transakcijama kako bismo obezbedili integritet podataka i da bismo mogli da upravljamo greškama.
 - Praktično ćemo više SQL upita grupisati i izvršiti ih kao jednu logiču nedeljivu celinu, odnosno transakciju.
-

Nekoliko jednostavnih pitanja i odgovora u vezi transakcija

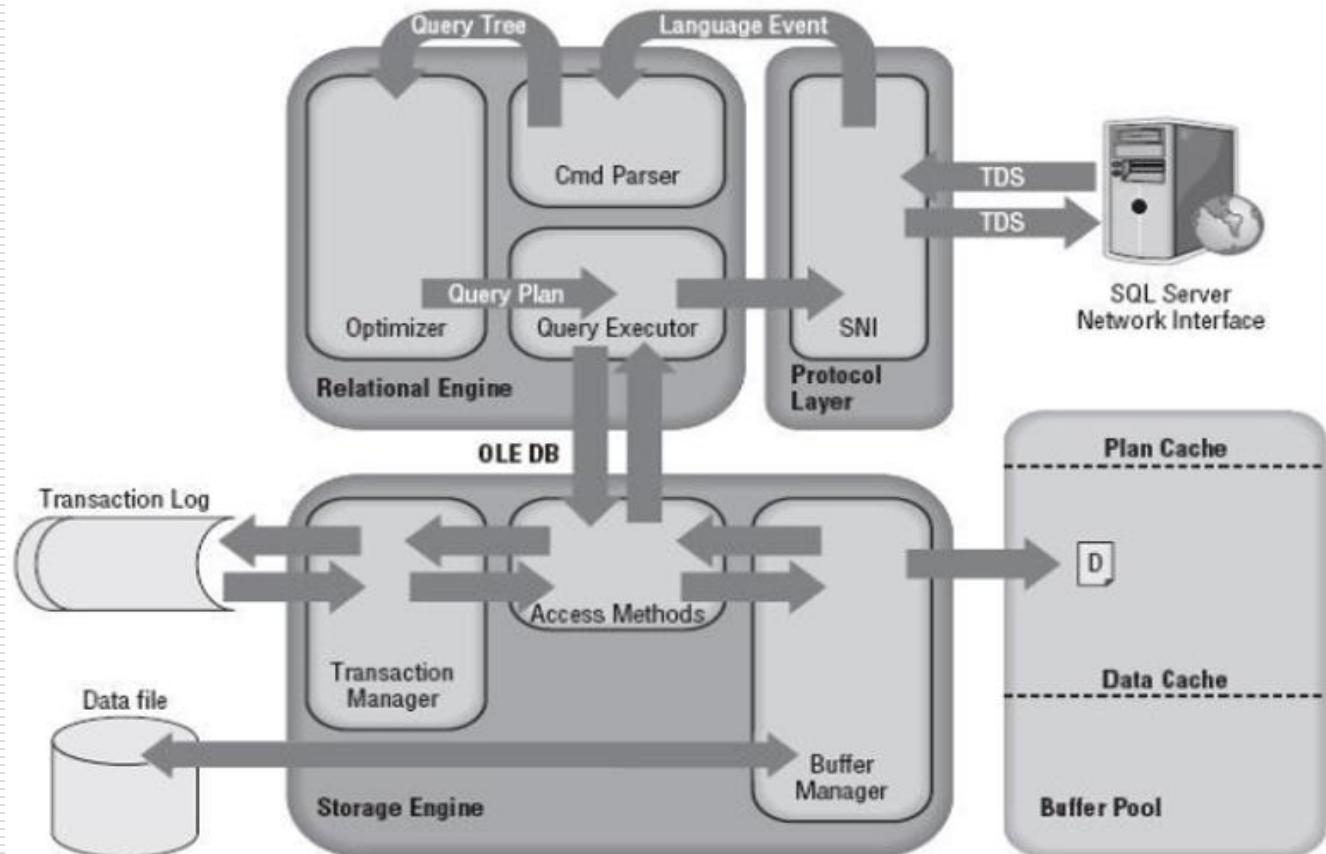
Ko formuluje i definiše transakciju?

- Transakciju nad bazom podataka formuluje i definiše korisnik koji želi da izvrši neku radnju nad bazom podataka.
- Korisnik može biti aplikacija, korisnički program ili osoba koja ima pristup bazi podataka.
- Aplikaciju osmišljava (dizajnira i piše njen programski kod) progamer, tako da on definiše i osmišljava i transakciju nad bazom podataka u okviru svog izvornog programa.
- Pri tome programer ili administrator baze podataka koristi mehanizme i naredbe koje podržava SUBP, odnosno njegova komponenta ***transaction manager***.

Šta je Administrator transakcija (*Transaction manager*)?

- Transaction Manager u SUBP-u je softverska komponenta koja upravlja transakcijama nad bazom podataka.
 - Ukratko, ***Transaction Manager*** u SUBP-u je ključna komponenta za održavanje integriteta baze podataka, obezbeđuje da se podaci sigurno i pouzdano čuvaju i obrađuju.
-

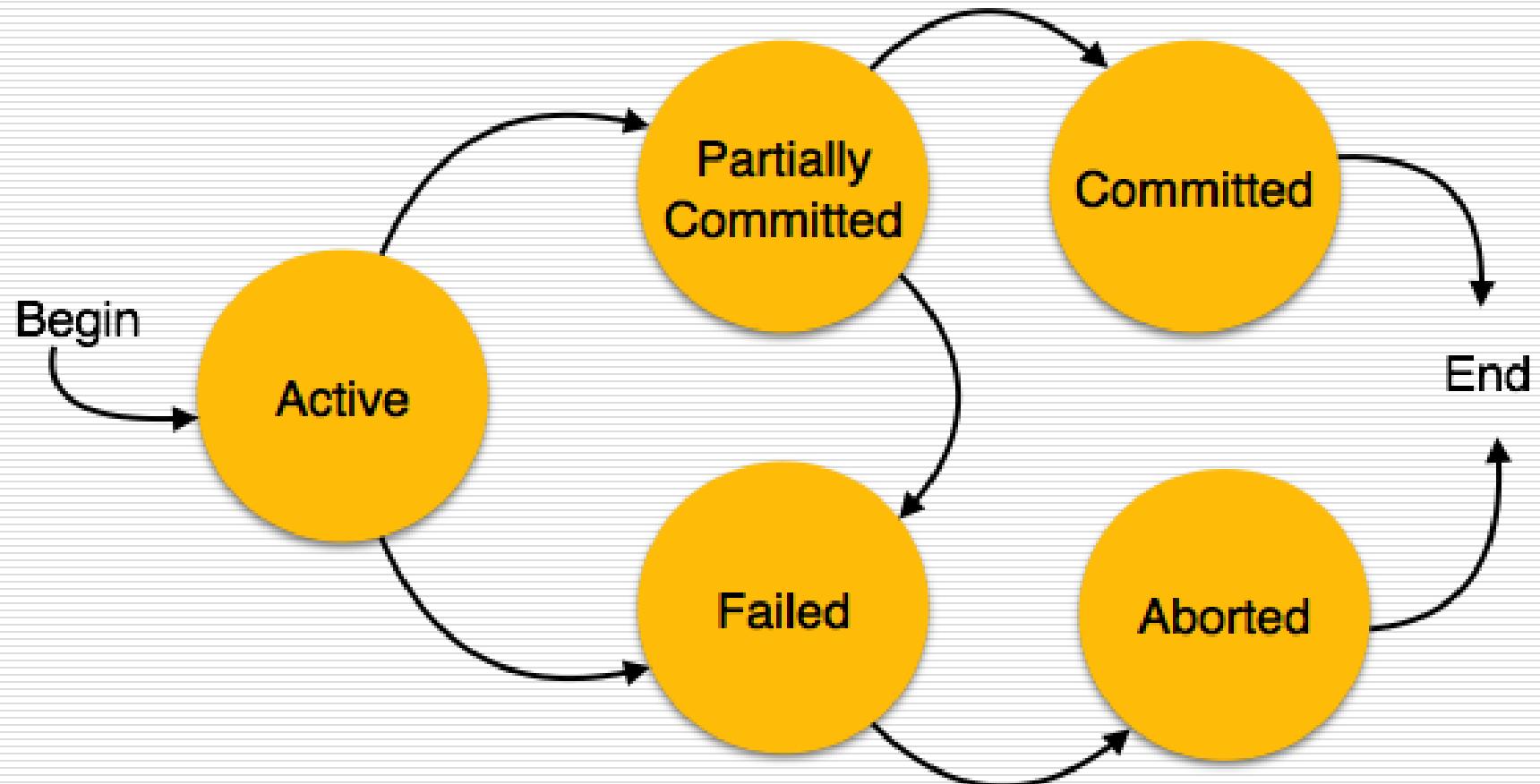
Mesto Administratora transakcija u okviru SUBP-a



Osobine transakcija: **ACID**

- Transakcije imaju 4 standardne osobine, obično opisane akronimom **ACID**.
 - **Atomicity** (atomarnost) – obezbeđuje nedjeljivost transakcije, što znači da se sve operacije u transakciji izvršavaju u potpunosti ili se ne izvršavaju uopšte.
 - Ako se bilo koja operacija u transakciji ne može izvršiti, sve promene se poništavaju i vraćaju u početno stanje pre početka transakcije (rollback operacija).
 - Što znači da će transakcija ili biti izvršena u potpunosti ili će se poništiti sva njena dejstva.
 - Ne sme postojati stanje baze u kome je transakcija samo parcijalno izvršena.
-

Osobine transakcija: ACID



Osobine transakcija: ACID

- **Consistency** (konzistentnost) – obezbeđuje da će baza pravilno promeniti stanje nakon uspešno izvršene transakcije.
 - Nijedna transakcija ne sme ostaviti negativan efekat na podatke koji se nalaze u bazi podataka.
 - Ako je baza bila u konzistentnom stanju pre izvršene transakcije, mora ostati u konzistentnom stanju i nakon izvršene transakcije.
-

Osobine transakcija: ACID

- ***Isolation*** (izolacija) – obezbeđuje da nijedna transakcija neće ugroziti izvršavanje neke druge transakcije.

 - U sistemu baza podataka, gde se jedna ili više transakcija izvršava redom, ili paralelno, izolacija omogućava da svaka transakcija bude izvršena kao da je jedina transakcija u sistemu.
-

Osobine transakcija: ACID

- ❑ **Durability** (trajnost) – obezbeđuje da će rezultat izvršene transakcije postojati i u slučaju otkaza sistema.
 - ❑ Baza podataka bi trebalo da bude trajna toliko da zadrži i poslednje napravljene izmene čak i u slučaju otkaza sistema ili ponovnog pokretanja.
 - ❑ Ako transakcija promeni deo podataka u bazi i ako je izvršena u potpunosti, promene nad bazom podataka će biti trajne čak id da neposredno posle toga dođe do "ispada" bilo kog dela sistema.
 - ❑ U slučaju da transakcija komituje, ali da baza otkaže pre nego što se podaci upišu na disk, podaci će biti ažurirani čim se sistem vrati u operativno stanje.
-

Kontrola transakcija

- ❑ Postoji nekoliko komandi za upravljanje transakcijama:
 - ❑ COMMIT – da sačuvamo promene.
 - ❑ ROLLBACK – da vratimo na prethodno stanje.
 - ❑ SAVEPOINT – tačka u transakciji na koju možemo da se vratimo bez potrebe za rollback-om cele transakcije
 - ❑ SET TRANSACTION – postavljamo ime transakciji.
-

Kontrola transakcija

- ❑ Komande za kontrolu transakcija se koriste isključivo za naredbe INSERT, UPDATE i DELETE.
 - ❑ Ne mogu se koristiti prilikom kreiranja ili brisanja tabele, zato što se ove operacije automatski komituju u bazu podataka.
-

Commit

- ❑ **COMMIT** – komanda koja se koristi da sačuvamo izmene koje je napravila transakcija.
 - ❑ Commit komanda čuva sve transakcije u bazi, od poslednje Commit ili Rollback naredbe.
 - ❑ Sintaksa za Commit naredbu:
 - ❑ **COMMIT;**
-

Commit

ID	Name	Age	Salary
1	Milan	25	3000
2	Goran	23	1450
3	Marko	25	2000
4	Petar	27	5000
5	Jovan	29	3150

ID	Name	Age	Salary
2	Goran	23	1450
4	Petar	27	5000
5	Jovan	29	3150

> DELETE FROM CUSTOMERS WHERE AGE = 25;
> COMMIT;

Rollback

- **ROLLBACK** – je komanda kojom poništavamo akcije transakcije koje još uvek nisu sačuvane u bazi podataka.
 - Rollback možemo koristiti da poništimo samo one transakcije koje su izvršene nakon poslednjeg commit-a ili rollback-a.
 - Sintaksa za Rollback naredbu:
 - **ROLLBACK;**
-

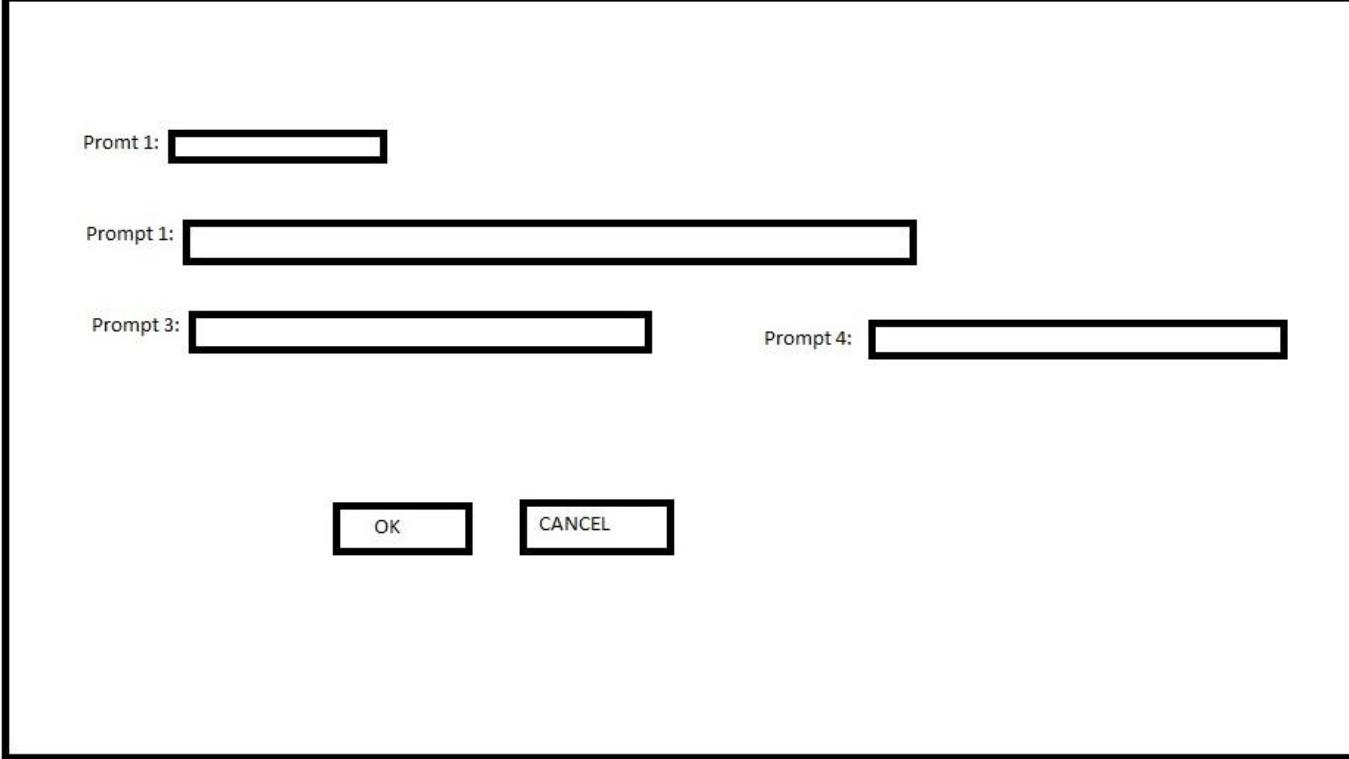
Rollback

ID	Name	Age	Salary
1	Milan	25	3000
2	Goran	23	1450
3	Marko	25	2000
4	Petar	27	5000
5	Jovan	29	3150

ID	Name	Age	Salary
1	Milan	25	3000
2	Goran	23	1450
3	Marko	25	2000
4	Petar	27	5000
5	Jovan	29	3150

> DELETE FROM CUSTOMERS WHERE AGE = 25;
> ROLLBACK;

Definisanje transakcije



Promt 1:

Promt 1:

Promt 3: Promt 4:

OK CANCEL

Kako se osmišljava transakcija?

ŠEMATSKI PRIKAZ PROGRAMIRANJA JEDNE TRANSAKCIJE NAD BAZOM PODATAKA

Begin Transaction

.....

.....

Insert 1

If ErrorStatus <> 0

ROLLBACK

.....

.....

Insert 2

If ErrorStatus <> 0

ROLLBACK

.....

.....

End Transaction

Savepoint

- **SAVEPOINT** – je tačka u transakciji, na koju možemo da se vratimo bez potrebe za rollback-om cele transakcije.
- Sintaksa za savepoint:

SAVEPOINT SAVEPOINT_NAME;

- Ova komanda služi samo za pravljenje „sigurne tačke“. Rollback-om se mogu poništiti ažuriranja transakcije do neke svapoint tačke.
- Sintaksa za rollback savepoint:

ROLLBACK TO SAVEPOINT_NAME;

Savepoint

ID	Name	Age	Salary
1	Milan	25	3000
2	Goran	23	1450
3	Marko	25	2000
4	Petar	27	5000
5	Jovan	29	3150

- SAVEPOINT SP1;
Savepoint created.
 - DELETE FROM CUSTOMERS WHERE ID = 1;
1 row deleted.
 - SAVEPOINT SP2;
Savepoint created.
 - DELETE FROM CUSTOMERS WHERE ID = 2;
1 row deleted.
 - SAVEPOINT SP3;
Savepoint created.
 - DELETE FROM CUSTOMERS WHERE ID = 3;
1 row deleted.
-

Savepoint

- ROLLBACK TO SP2;
Rollback complete.

ID	Name	Age	Salary
2	Goran	23	1450
3	Marko	25	2000
4	Petar	27	5000
5	Jovan	29	3150

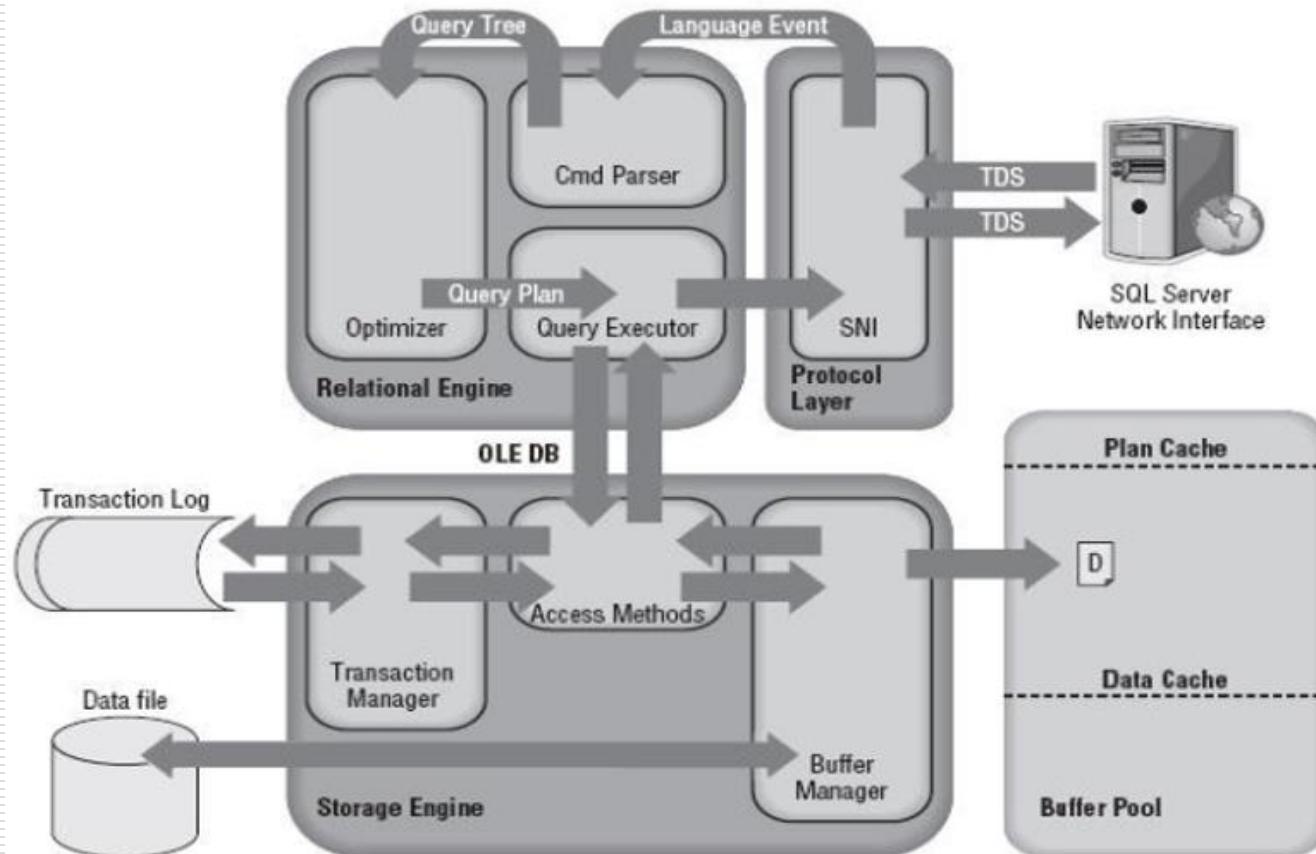
Release savepoint

- ❑ **RELEASE SAVEPOINT** – koristimo da obrišemo savepoint koji smo prethodno napravili.
 - ❑ Sintaksa za release savepoint:
 - ❑ **RELEASE SAVEPOINT SAVEPOINT_NAME;**
 - ❑ Onog trenutka kad obrišete savepoint, više ne možete da koristite rollback nad transakcijama izvršenim od tog savepoint-a.
-

Set transaction

- **SET TRANSACTION** – komanda kojom započinjemo transakciju nad bazom podataka.
 - Ovom komandom specifikujemo osobine određene transakcije.
 - Na primer, možemo da kažemo da je transakcija read-only, ili da je transakcija read-write
 - Sintaksa za set transaction:
 - **SET TRANSACTION [READ ONLY | READ WRITE];**
-

Mesto Administratora transakcija u okviru SUBP-a



Žurnalizacija transakcija

- Žurnalizacija transakcija nad bazom podataka i "transaction logging" je proces beleženja svih promena koje se događaju u bazi podataka kako bi se omogućio oporavak podataka u slučaju neočekivanog prekida rada sistema (SUBP-a, OS ...) ili kvara hardvera, na nivou transakcije i na nivou baze podataka
 - Koja je razlika između LOG-a transakcija i Žurnal fajla?
 - LOG transakcija i žurnala fajl su dva različita koncepta u kontekstu žurnalizacije transakcija u bazi podataka.
 - Log transakcija sadrži podatke svake transakcije koja se izvršava nad bazom podatakada bi se omogućilo vraćanje baze podataka u prethodno stanje u slučaju neočekivanih problema.
 - Žurnala fajl se koristi za smeštanje svih podataka o uspešno izvršenim transakcijama nad bazom podataka od trenutka uzete poslednje rezervne kopije baze podataka. Služi za oporavak (Restore) baze podataka.
-

Žurnalizacija transakcija (LOG fajl)

- ❑ *Before image* i *After image* su dva ključna koncepta u žurnalizaciji transakcija baze podataka.
 - ❑ Ove koncepcije se često koriste u tehnologijama za zaštitu podataka u slučaju prekida rada i oporavka podataka.
 - ❑ *Before image* žurnalizacija beleži stranice baze podataka pre ažuriranja, tako da se u slučaju greške u radu SUBP-a, operativnog sistema ili hardvera mogu oporaviti i obnoviti izmenjene stranice baze podataka iz žurnala.
 - ❑ *After image* žurnalizacij, obično se koristi u relacionim SUBP. Beleži promene u žurnal pre nego što se te promene upišu u stvarnu bazu podataka.
-

Žurnalizacija ažuriranja Baze podataka

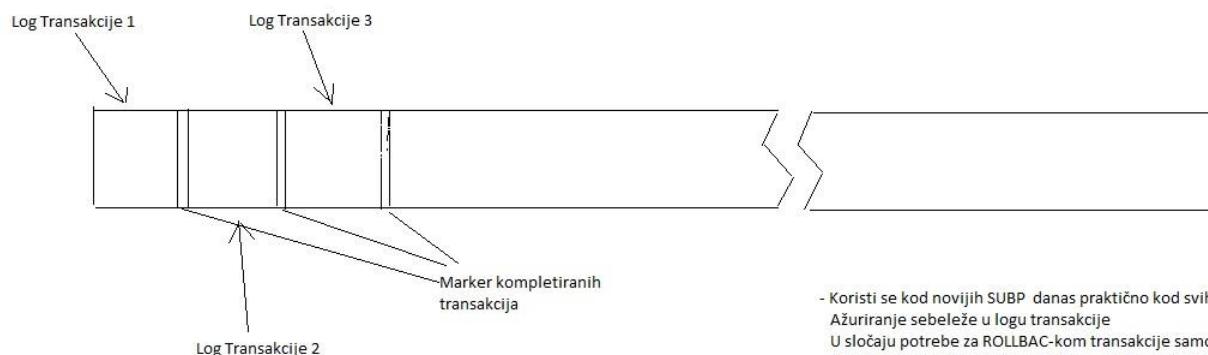
Vrste žurnalizacije

1. Before Image Žurnalizacija
2. After Image žurnalizacija

BEFORE IMAGE ŽURNALIZACIJA

- Korišćena kod starijih SUBP
- Kod SUBP zasnovanih na mrežnom modelupodataka i pre toga
- kod SUBP zasnovanih na hijerarhijskom modelu podataka
- U log transakcija se upisuju Stranice BP pre ažuriranja, a
- samo ažuriranje se vršilo u samoj BP
- Ukoliko dođe do potrebe za ROLLBACK-om stranice pre ažuriranja BP
- su se vraćale u BP i na taj način se poništavalo ažuriranje
- Posebno se vodio DNEVNIK AŽURIRANJA u drugoj datoteci u odnosu na log transakcije

AFTER IMAGE ŽURNALIZACIJA



- Koristi se kod novijih SUBP danas praktično kod svih relationalnih
- Ažuriranje se beleže u logu transakcije
- U slučaju potrebe za ROLLBACK-kom transakcije samo se prostor u Logu Transakcije označi slobodnim
- Ako je transakcija uspešno završena - Izdata je COMMIT naredba (bilo explicitno ili implicitno)
- Vrši se ažuriranje BP na osnovu sadržaja LOGA Transakcije koja je uspešno završena
- Logovi uspešno izvršenih transakcija se povezuju i praktično formiraju DNEVNIK Ažuriranja
- Nema posebnog fajla za DNEVNIK Ažuriranja
- Prilikom uzimanja rezervne kopije BP (BACKUP-a) prostor LOG-a Transakcija je praktično logički prazan
- Iako fizički prostor na disku ostaje zauzet, pa Fajl sistem može pokazivati da je fajl LOG-a Transakcija jako velik

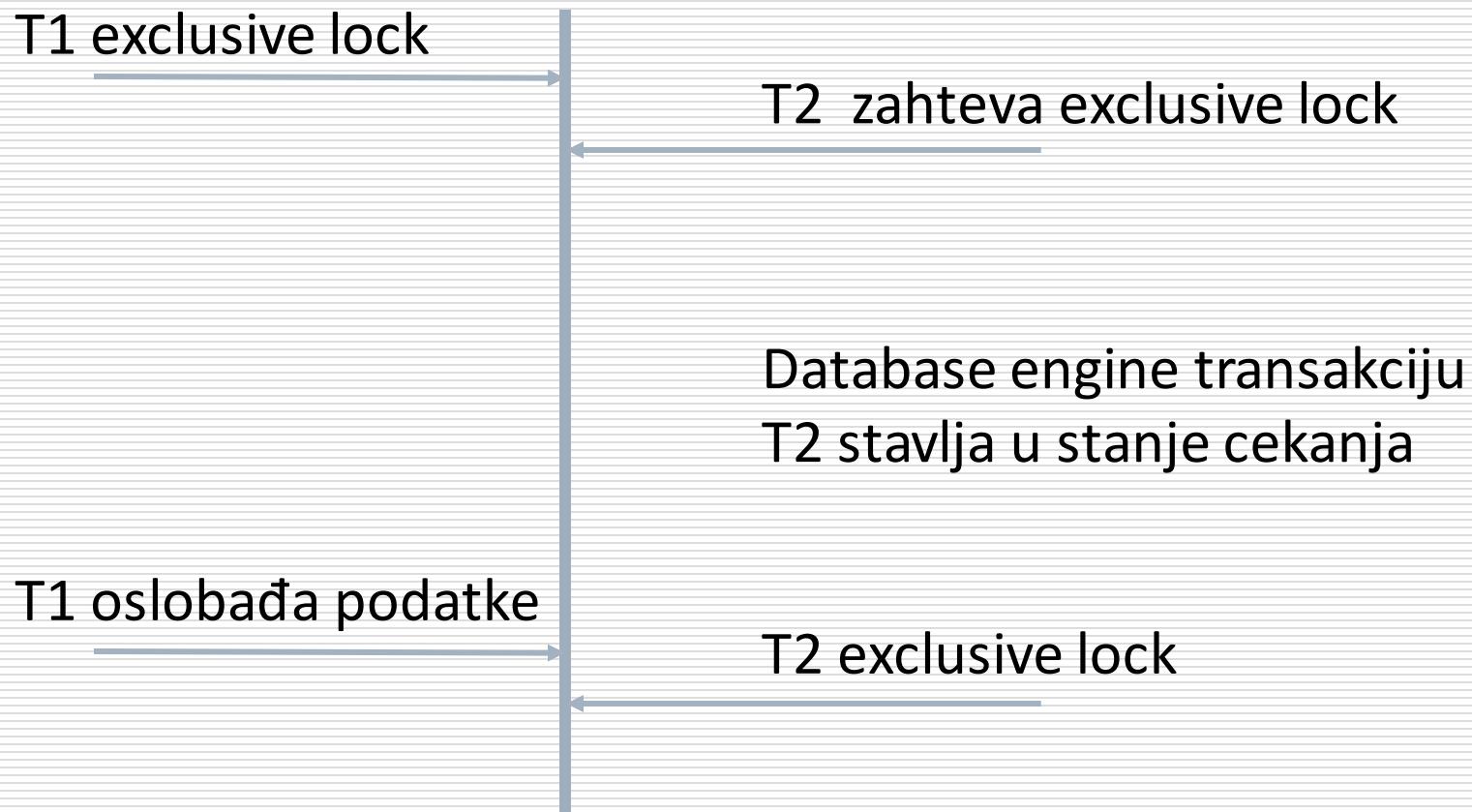
Zaključavanje baze podataka (locking in)

- **Zaključavanje** je mehanizam koji se koristi od strane SUBP-a da sinhronizuje pristup više korisnika istim podacima u isto vreme.
 - Pre nego što transakcija počne da menja stanje podataka, prilikom čitanja ili ažuriranja, mora da se zaštiti od efekata druge transakcije koja pokušava da menja podatke istovremeno.
 - Transakcija ovo realizuje slanjem zahteva za zaključavanjem tog dela podataka.
 - Zaključavanje ima različite režime, kao što su **shared** (deljen) ili **exclusive** (isključivo za sebe).
 - Zaključavanje definiše nivo zavisnosti koji transakcija ima nad određenim podacima.
-

Zaključavanje baze podataka

- Transakciji se ne može dodeliti pravo zaključavanja koje bi prouzrokovalo konflikt sa modom zaključavanja koji je druga transakcija već dobila.
 - Ako transakcija zahteva mod zaključavanja koji će prouzrokovati konflikt sa zaključavanjem koje je već dodeljeno nad istim podacima, SUBP će zahtevanu transakciju staviti u status cekanja sve dok se ne oslobole podaci.
 - Primer: T1 ima **exclusive** pravo nad podacima. T2 traži **exclusive** pravo nad istim podacima. Database engine transakciju T2 stavlja u stanje cekanja. T1 oslobađa podatke. Database engine dodeljuje **exclusive** pravo nad podacima za transakciju T2.
-

Zaključavanje baze podataka



Zaključavanje baze podataka

- Kada transakcija ažurira podatke, ona ih drži zaključanim štiteći ih od promena sve dok se transakcija ne zavrsi.
 - Vreme koliko transakcija drži podatke zaključanim zavisi od podešavanja kojima je definisan nivo izolacije transakcije.
 - Svako zaključavanje se oslobađa prilikom završetka transakcije (bilo u slučaju commit-a ili rollback-a).
-

Zaključavanje baze podataka

- ❑ Aplikacije obično ne zahtevaju zaključavanje podataka. Zaključavanjima se upravlja interno, komponentom Database Engine koji se zove **lock manager**.
 - ❑ Kada istanca Database Engine procesira SQL upit, Database Engine Query procesor određuje kojim reursima može da se pristupa.
-

Zaključavanje baze podataka

- Query procesor odlučuje koji tipovi zaključavanja su neophodni da bi se zaštitio svaki resurs.
 - Odlučivanje o tipu zaključavanja je zasnovano na tipu pristupa i nivou izolacije.
 - Query procesor dalje zahteva određeno zaključavanje od ***lock manager-a***.
 - Lock manager odobrava zaključavanje, samo ako ne postoji mogućnost da dodje do konflikta zbog neke druge transakcije.
-

Nivo izolacije transakcije

- *Isolation levels* su nivoi izolacije koji se koriste kako bi se regulisalo ponašanje transakcija u odnosu na ostale transakcije koje se izvršavaju u isto vreme.
 - Omogućavaju definisanje nivoa vidljivosti podataka između različitih transakcija i utiču na to kako će se podaci prikazati kada se izvrši upit, u zavisnosti od toga koje su druge transakcije bile aktivne u isto vreme.
 - Postoje četiri standardna nivoa izolacije transakcija, koji su definisani u ANSI/ISO SQL standardu. Oni su:
 1. Read Uncommitted (nivo izolacije čitanja nekomitovanih podataka)
 2. Read Committed (nivo izolacije čitanja komitovanih podataka)
 3. Repeatable Read (nivo izolacije ponovljivog čitanja)
 4. Serializable (nivo izolacije serijalizabilnosti)
-

Nivoi izolacije transakcije

1. Read Uncommitted (čitanje nekomitovanih podataka)

- Najmanje restriktivan nivo izolacije.
- Transakcija može da čita i prikaže podatke koji nisu još uvek komitovani u bazi podataka.
- Može da dovede do prikaza nekonzistentnih podataka, jer se druga transakcija može izvršiti između dva čitanja istih podataka.

2. Read Committed (čitanje komitovanih podataka)

- Najčešće koristi u bazama podataka.
- transakcija može da čita samo one podatke koji su već komitovani u bazi podataka.
- Obezbeđuje da će druga transakcija videti samo komitovane podatke.

Nivoi izolacije transakcije

3. Repeatable Read (ponovljivo čitanje)

- Transakcija može da čita podatke koji su bili prisutni u bazi podataka na početku transakcije.
- Bilo kakve promene koje se dešavaju u drugim transakcijama biti ignorisane,
- Transakcija ponašati kao da se ništa nije promenilo.

4. Serializable (serijalizabilnost)

- Najrestriktivniji nivo izolacije.
- Transakcije se izvršavaju jedna po jedna.
- Sve druge transakcije će biti blokirane dok se jedna transakcija ne završi. Ovo obezbeđuje najviši stepen konzistentnosti podataka
- Skup u smislu performansi

Repeatable Read (nivo izolacije ponovljivog čitanja)

- Obezbeđuje da se transakcija izvršava u okviru fiksne slike baze podataka, koja neće biti promenjena sve dok se transakcija ne završi.
 - Druga transakcija neće biti u mogućnosti da izmeni podatke koji su uključeni u transakciju koja je u toku, sve dok ona ne završi.
 - Ako je potrebno izvršiti nekoliko upita kako bi se izračunali rezultati transakcije, sve te upite treba izvršiti u okviru iste transakcije kako bi se osigurala konzistentnost podataka.
 - Podaci se mogu čitati više puta tokom izvršavanja transakcije, ali ukoliko se upit ponovi, vrednosti koje se vraćaju će biti iste.
 - Ovim nivoom izolacije, podaci se čitaju tako da se blokiraju ostale transakcije koje pokušavaju da izvrše ažuriranje tih istih podataka. na taj način se osigurava da se podaci neće promeniti tokom izvršavanja transakcije.
-

Granularnost i hijerarhija zaključavanja

- ❑ Database Engine ima multigranularno zaključavanje koje omogućava različitim tipovima resursa da budu zaključani određenom transakcijom.
 - ❑ Da bi smanjili troškove zaključavnaja, Database Engine zaključava resurse automatski na nivou koji je prikladan tom zadatku.
 - ❑ Zaključavajući manju granularnost resursa, kao što su redovi u tabeli, povećava se konkurentnost ali se troškovi povećavaju takođe jer više zaključavanja moramo obraditi istovremeno.
 - ❑ Zaključavanje većih granularnosti, kao što su tabele, smanjićemo opšte troškove ali ćemo zaključati celu bazu, odnosno niko neće moći da pristupi bilo kojem delu te baze, čime ćemo smanjiti konkurentnost.
-

Granularnost i hijerarhija zaključavanja

- Primenljivo na: SQL Server.
 - Database Engine mora da dobije multi level granularnosti da bi potpuno zaštitili podatke.
 - Ove grupe zaključavanja na multi levelu granularnosti se zovu **hijerarhija zaključavanja**.
 - Primer: Da bi potpuno zaštitili čitanje indeksa, Database Engine zahteva deljeno(shared) zaključavanje redova i (intent shared) deljeno zaključavanje stranica i tabele u bazi podataka.
-

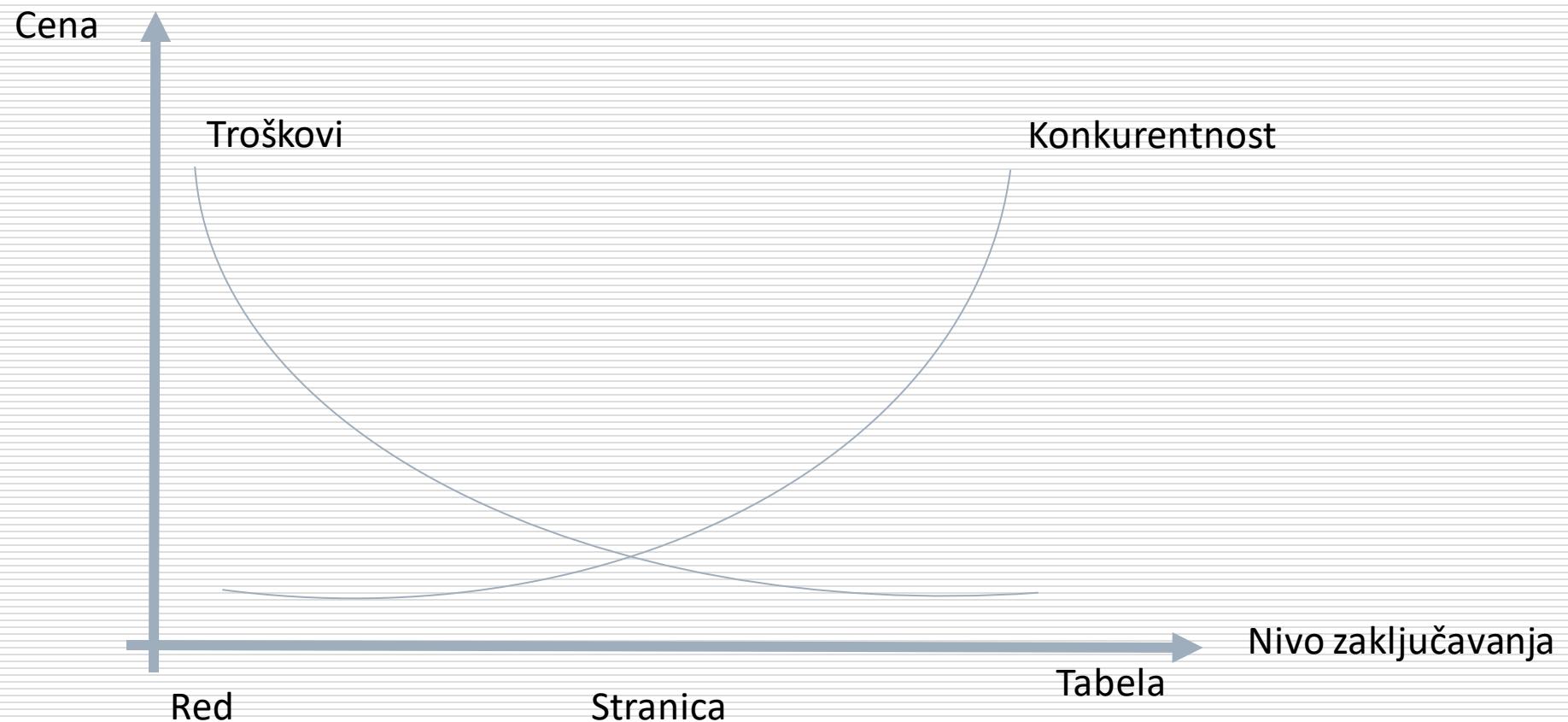
Granularnost i hijerarhija zaključavanja

- Nivoi zaključavanja:
 - RID (Row Identifier) – zaključavanje jednog reda.
 - KEY – zaključavanje reda unutar indeksa da bi zaštitili ključeve prilikom serijabilnih transakcija.
 - PAGE – 8KB stranica u bazi podataka.
 - EXTENT – grupa od 8 susednih stranica.
 - HoBT – a heap or B-tree. Zaključavanje štiti B-tree ili heap stranice u tabeli koje još uvek nemaju grupisane indekse.
-

Granularnost i hijerarhija zaključavanja

- TABLE – zaključavanje cele tabele, uključujući podatke i indekse.
 - FILE – fajl baze podataka.
 - APPLICATION – zaključavanje na nivou aplikacije.
 - METADATA – zaključavanje meta podataka.
 - ALLOCATION_UNIT – jedinica alokacije (veličina klastera)
 - DATABASE – zaključavanje cele baze podataka.
-

Granularnost i hijerarhija zaključavanja

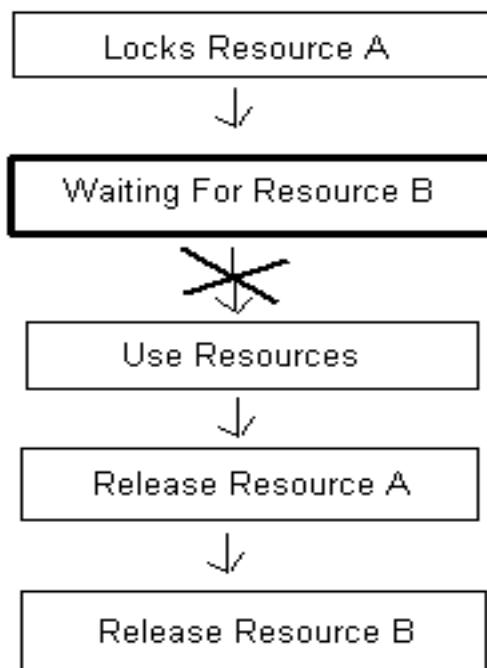


Modovi zaključavanja

- Microsoft SQL Server Database Engine zaključava resurse koristeći različite modove za zaključavanje kojima odlučuje na koji način se može pristupati podacima od strane konkurentnih transakcija.
 - Modovi zaključavanja koje Database Engine koristi:
 - **Shared (S)** – koristi se za operacije čitanja koje ne menjaju podatke, kao što je SELECT upit.
 - **Update (U)** – koristi se za resurse koje možemo promeniti. Štiti od uobičajene forme deadlock-a koji se dešava kada više sesija čita, zaključava i potencijalno menja podatke istovremeno.
-

Modovi zaključavanja - deadlock

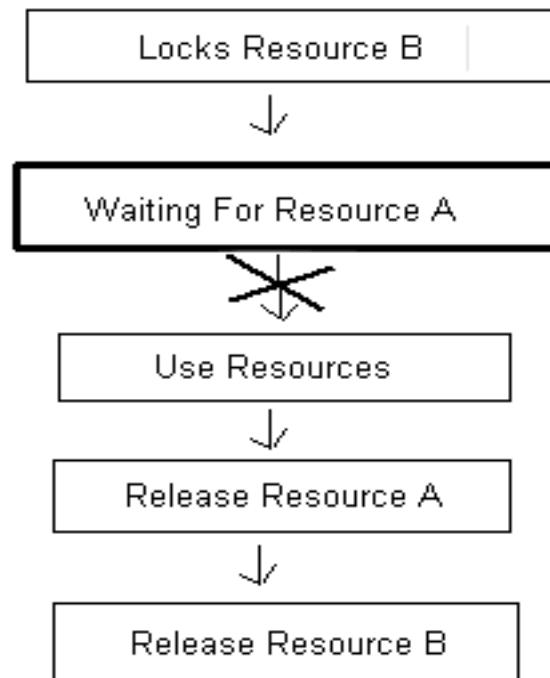
Transaction 1



Both transactions waiting
for a resource that the
other has just locked.

Result: Neither transaction
moving forward.

Transaction 2



Modovi zaključavanja

- **Exclusive (X)** - koristi se za operacije koje ažuriraju podatke kao što su INSERT, UPDATE ili DELETE. Obezbeđuje da ne možemo vršiti višestruko ažuriranje istih podataka istovremeno.
 - **Intent** – koristi se da uspostavi hijerarhiju zaključavanja. Tipovi intent zaključavanja su: intent shared (IS), intent exclusive (IX) i shared intent exclusive (SIX).
 - **Schema** – koriste se kada se operacije zavisne od šeme tabele izvršavaju. Tipovi schema zaključavanja su: schema modification (Sch-M) i schema stability (Sch-S).
-

Modovi zaključavanja

- **Bulk Update** (BU) – koristi se prilikom kopiranja podataka u tabelu i kada je TABLOCK hint naveden.
- **Key-Range** - štiti redove koji se čitaju od strane upita prilikom serijabilnih transakcija sa određenim nivoom izolacije. Obezbeđuje da druge transakcije ne mogu da dodaju redove koji će biti kvalifikovani za upit serijabilnih transakcija ako su upiti pokrenuti ponovo.

Modovi zaključavanja - kompatibilnost

Mode	NL	IS	IX	S	SIX	X
NL	Yes	Yes	Yes	Yes	Yes	Yes
IS	Yes	Yes	Yes	Yes	Yes	No
IX	Yes	Yes	Yes	No	No	No
S	Yes	Yes	No	Yes	No	No
SIX	Yes	Yes	No	No	No	No
X	Yes	No	No	No	No	No

Administracija baza podataka

Zaključavanje objekata baze podataka

Sadržaj

- Uvod
 - Nivoi izolacije transakcija
 - Granulacija i hijerarhija zaključavanja
 - Režimi zaključavanja
 - Kompatibilnost zaključavanja
 - Dinamičko zaključavanje
 - Prikazivanje informacija o zaključavanju
 - Deadlocking
-

Uvod

- Baza podataka je zajednički resurs za više programa istovremeno, što može izazvati probleme
 - Kao rešenje, koriste se *transakcije*
 - Pojam transakcije
 - Transakcija je jedno izvršenje neke “logičke jedinice posla”
 - Osobine transakcije: **Atomicity, Consistency, Isolation, Durability (ACID)**
-

Zaključavanje baze podataka (locking in)

- ❑ Zaključavanje je mehanizam koji se koristi od strane SUBP-a da sinhronizuje pristup više korisnika istim podacima u isto vreme.
 - ❑ Pre nego što transakcija počne da menja stanje podataka, prilikom čitanja ili ažuriranja, mora da se zaštiti od efekata druge transakcije koja pokušava da menja podatke istovremeno.
 - ❑ Transakcija ovo realizuje slanjem zahteva za zaključavanjem tog dela podataka.
 - ❑ Zaključavanje ima različite režime, kao što su **shared** (deljen) ili **exclusive** (isključivo za sebe).
 - ❑ Zaključavanje definiše nivo zavisnosti koji transakcija ima nad određenim podacima.
-

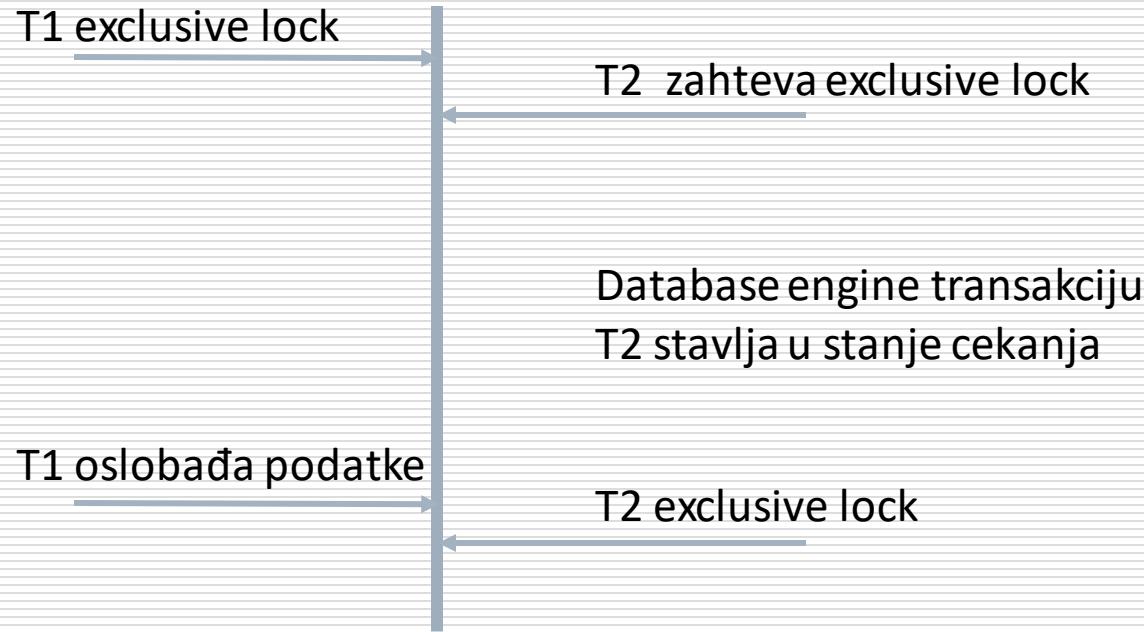
Zaključavanje baze podataka (locking in)

- ❑ **Zaključavanje** je mehanizam koji se koristi od strane SUBP-a da sinhronizuje pristup više korisnika istim podacima u isto vreme.
 - ❑ Pre nego što transakcija počne da menja stanje podataka, prilikom čitanja ili ažuriranja, mora da se zaštiti od efekata druge transakcije koja pokušava da menja podatke istovremeno.
 - ❑ Transakcija ovo realizuje slanjem zahteva za zaključavanjem tog dela podataka.
 - ❑ Zaključavanje ima različite režime, kao što su **shared** (deljen) ili **exclusive** (isključivo za sebe).
 - ❑ Zaključavanje definiše nivo zavisnosti koji transakcija ima nad određenim podacima.
-

Zaključavanje baze podataka

- Transakciji se ne može dodeliti pravo zaključavanja koje bi prouzrokovalo konflikt sa modom zaključavanja koji je druga transakcija već dobila.
 - Ako transakcija zahteva mod zaključavanja koji će prouzrokovati konflikt sa zaključavanjem koje je već dodeljeno nad istim podacima, SUBP će zahtevanu transakciju staviti u status cekanja sve dok se ne oslobode podaci.
 - Primer: T1 ima **exclusive** pravo nad podacima. T2 traži **exclusive** pravo nad istim podacima. Database engine transakciju T2 stavlja u stanje cekanja. T1 oslobođa podatke. Database engine dodeljuje **exclusive** pravo nad podacima za transakciju T2.
-

Zaključavanje baze podataka



Zaključavanje baze podataka

- Kada transakcija ažurira podatke, ona ih drži zaključanim štiteći ih od promena sve dok se transakcija ne zavrsi.
 - Vreme koliko transakcija drži podatke zaključanim zavisi od podešavanja kojima je definisan nivo izolacije transakcije.
 - Svako zaključavanje se oslobađa prilikom završetka transakcije (bilo u slučaju commit-a ili rollback-a).
-

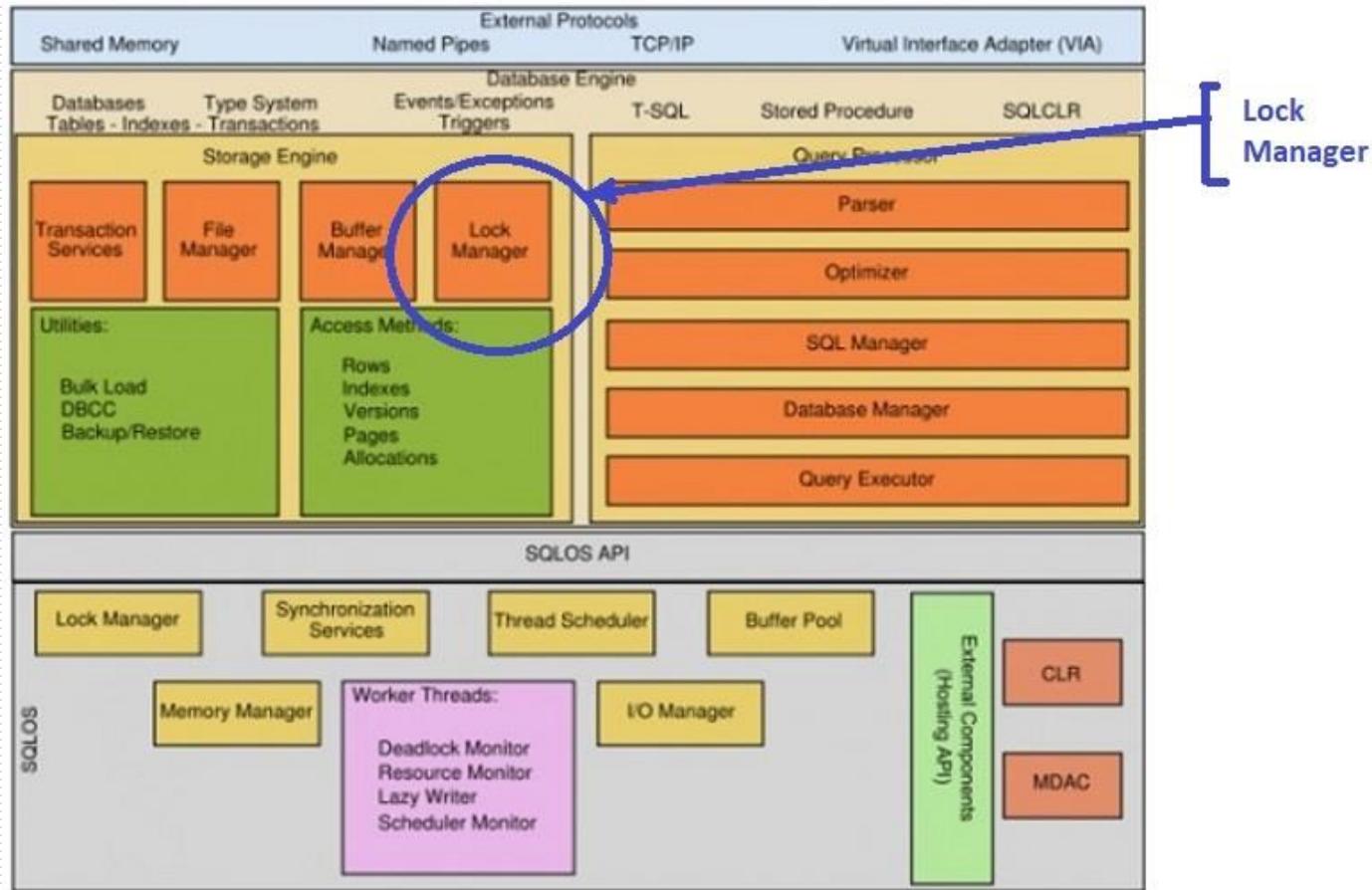
Zaključavanje baze podataka

- Aplikacije obično ne zahtevaju zaključavanje podataka. Zaključavanjima se upravlja interna komponentom Database Engine koji se zove **lock manager**.
 - Kada istanca Database Engine procesira SQL upit, Database Engine Query procesor određuje kojim reursima može da se pristupa.
-

Zaključavanje baze podataka

- Query procesor odlučuje koji tipovi zaključavanja su neophodni da bi se zaštitio svaki resurs.
 - Odlučivanje o tipu zaključavanja je zasnovano na tipu pristupa i nivou izolacije.
 - Query procesor dalje zahteva određeno zaključavanje od ***lock manager-a***.
 - Lock manager odobrava zaključavanje, samo ako ne postoji mogućnost da dodje do konflikta zbog neke druge transakcije.
-

Arhitektura SUBP – Lock Manager



Optimističko i pesimističko zaključavanja (1/2)

- Database locking, Optimistic and Pessimistic locking
 - **Locking** (zaključavanje, blokiranje) je mehanizam koji koristi SUBP da sinhronizuje pristup više korisnika istim podacima u istom vremenskom intervalu. Kada transakcija modifikuje deo podataka, ona postavlja “*katanac*” nad tim podacima, štiteći izmenu do kraja transakcije.
 - Savremeni SUBP sistemi uobičajeno obezbeđuju minimalno restriktivan nivo koji garantuje očuvanje konzistentnosti baze podataka u višekorisničkom režimu rada, tkzv. “**optimističko zaključavanje**”(engl. Optimistic locking).

Optimističko i pesimističko zaključavanja (2/2)

- Međutim, korisnik SUBP može sprovoditi eksplicitno zaključavanje resursa i pri tome samo može pooštriti restriktivnost zaključavanja koju nameće SUBP, tkzv. "**pesimističko zaključavanje**"(engl. Pessimistic locking), koje se postiže narednom LOCK TABLE.
-

Paralelnoizvršavanje transakcija

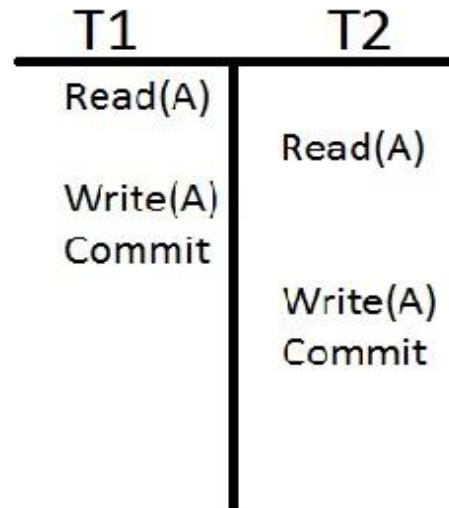
- SUBP treba da obezbedi adekvatno upravljanje paralelnim izvršavanjem transakcija kako one nepoželjno mešale svoja dejstva jedna na drugu.
 - U osnovi postoje tri slučaja kada je transakcija samostalno korektna, ali zbog mešanja transakcija može nastati pogrešan rezultat:
 1. Izgubljena ažuriranja
 2. Zavisnost od privremenih ažuriranja
 3. Narušavanje serijabilnosti
-

Problem izgubljenog ažuriranja

Konkurentno izvršavanje transakcija i problemi

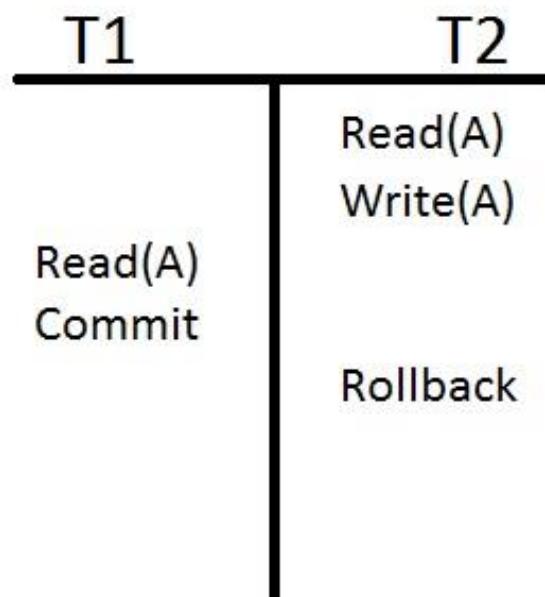
Konkurentan rad znači da više transakcija ima pristup istim podacima, istovremeno. Problemi:

■ **problem izgubljenih ažuriranja**



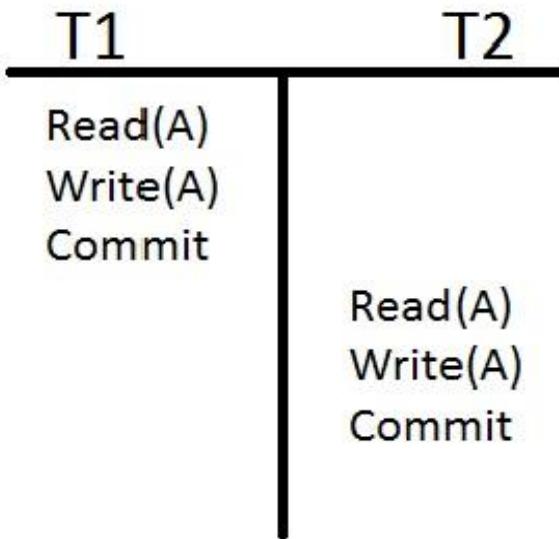
Zavisnost od privremenih ažuriranja

- Problem zavisnosti od poništenog ažuriranja

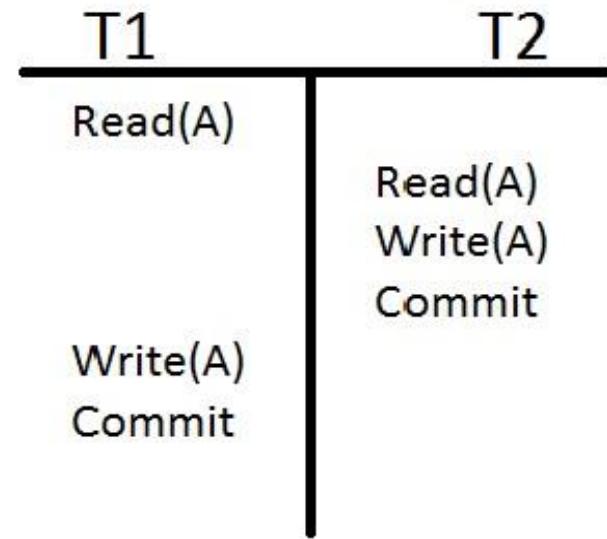


Narušavanje serijabilnosti

■ Problem nekonzistentne analize



Baza je konzistenta!



Baza nije konzistenta!

Ovi problemi se rešavaju sistemom zaključavanja (Database locking)

Nivoi izolacije transakcija

- Pomenuti konflikti se u okviru SUBP i njegove administracije kontrolišu tako što se definiše **Nivo izolacije transakcija** (*Isolation levels*)
 - Definiše se nivo vidljivosti promena koje su napravljene u jednom procesu drugim procesima.
 - Nivo izolacije (*Isolation levels*) u SUBP definiše koliko će promene koje se dešavaju u jednom procesu biti vidljive drugim procesima koji pristupaju istoj bazi podataka.
 - Postoji nekoliko nivoa izolacije u DBMS-u, a oni definišu kako se vrši kontrola konflikata u višekorisničkom okruženju.
-

Nivo izolacije transakcije

- *Isolation levels* su nivoi izolacije koji se koriste kako bi se regulisalo ponašanje transakcija u odnosu na ostale transakcije koje se izvršavaju u isto vreme.
 - Omogućavaju definisanje nivoa vidljivosti podataka između različitih transakcija i utiču na to kako će se podaci prikazati kada se izvrši upit, u zavisnosti od toga koje su druge transakcije bile aktivne u isto vreme.
 - Postoje četiri standardna nivoa izolacije transakcija, koji su definisani u ANSI/ISO SQL standardu. Oni su:
 1. Read Uncommitted (nivo izolacije čitanja nekomitovanih podataka)
 2. Read Committed (nivo izolacije čitanja komitovanih podataka)
 3. Repeatable Read (nivo izolacije ponovljivog čitanja)
 4. Serializable (nivo izolacije serijalizabilnosti)
-

Nivoi izolacije transakcije

1. Read Uncommitted (čitanje nekomitovanih podataka)

- Najmanje restriktivan nivo izolacije.
- Transakcija može da čita i prikaže podatke koji nisu još uvek komitovani u bazi podataka.
- Može da dovede do prikaza nekonzistentnih podataka, jer se druga transakcija može izvršiti između dva čitanja istih podataka.

2. Read Committed (čitanje komitovanih podataka)

- Najčešće koristi u bazama podataka.
 - transakcija može da čita samo one podatke koji su već komitovani u bazi podataka.
 - Obezbeđuje da će druga transakcija videti samo komitovane podatke.
-

Nivoi izolacije transakcije

3. Repeatable Read (ponovljivo čitanje)

- Transakcija može da čita podatke koji su bili prisutni u bazi podataka na početku transakcije.
- Bilo kakve promene koje se dešavaju u drugim transakcijama biti ignorisane,
- Transakcija ponašati kao da se ništa nije promenilo.

4. Serializable (serijalizabilnost)

- Najrestriktivniji nivo izolacije.
 - Transakcije se izvršavaju jedna po jedna.
 - Sve druge transakcije će biti blokirane dok se jedna transakcija ne završi. Ovo obezbeđuje najviši stepen konzistentnosti podataka
 - Skup u smislu performansi
-

Granularnost i hijerarhija zaključavanja

- ❑ Database Engine ima multigranularno zaključavanje koje omogućava različitim tipovima resursa da budu zaključani određenom transakcijom.
 - ❑ Da bi smanjili troškove zaključavnaja, Database Engine zaključava resurse automatski na nivou koji je prikladan tom zadatku.
 - ❑ Zaključavajući manju granularnost resursa, kao što su redovi u tabeli, povećava se konkurentnost ali se troškovi povećavaju takođe jer više zaključavanja moramo obraditi istovremeno.
 - ❑ Zaključavanje većih granularnosti, kao što su tabele, smanjićemo opšte troškove ali ćemo zaključati celu bazu, odnosno niko neće moći da pristupi bilo kojem delu te baze, čime ćemo smanjiti konkurentnost.
-

Nivo izolacije transakcije

- *Isolation levels* su nivoi izolacije koji se koriste kako bi se regulisalo ponašanje transakcija u odnosu na ostale transakcije koje se izvršavaju u isto vreme.
- Omogućavaju definisanje nivoa vidljivosti podataka između različitih transakcija i utiču na to kako će se podaci prikazati kada se izvrši upit, u zavisnosti od toga koje su druge transakcije bile aktivne u isto vreme.
- Postoje četiri standardna nivoa izolacije transakcija, koji su definisani u ANSI/ISO SQL standardu. Oni su:
 1. Read Uncommitted (nivo izolacije čitanja nekomitovanih podataka)
 2. Read Committed (nivo izolacije čitanja komitovanih podataka)
 3. Repeatable Read (nivo izolacije ponovljivog čitanja)
 4. Serializable (nivo izolacije serijalizabilnosti)

Nivoi izolacije transakcije

1. Read Uncommitted (čitanje nekomitovanih podataka)

- Najmanje restriktivan nivo izolacije.
- Transakcija može da čita i prikaže podatke koji nisu još uvek komitovani u bazi podataka.
- Može da dovede do prikaza nekonzistentnih podataka, jer se druga transakcija može izvršiti između dva čitanja istih podataka.

2. Read Committed (čitanje komitovanih podataka)

- Najčešće koristi u bazama podataka.
- transakcija može da čita samo one podatke koji su već komitovani u bazi podataka.
- Obezbeđuje da će druga transakcija videti samo komitovane podatke.

Nivoi izolacije transakcije

3. Repeatable Read (ponovljivo čitanje)

- Transakcija može da čita podatke koji su bili prisutni u bazi podataka na početku transakcije.
- Bilo kakve promene koje se dešavaju u drugim transakcijama biti ignorisane,
- Transakcija ponašati kao da se ništa nije promenilo.

4. Serializable (serijalizabilnost)

- Najrestriktivniji nivo izolacije.
 - Transakcije se izvršavaju jedna po jedna.
 - Sve druge transakcije će biti blokirane dok se jedna transakcija ne završi.
Ovo obezbeđuje najviši stepen konzistentnosti podataka
 - Skup u smislu performansi
-

Granularnost i hijerarhija zaključavanja

Šta je granulacija i hijerarhija zaključavanja?

- Zaključavanje se može zahtevati za različite vrste resursa, kao što su redovi, stranice, indeksi, tabele, ili baze podataka. Neke operacije zahtevaju postavljanje "katanaca" na više nivoa granularnosti, formirajući *hijerarhiju zaključavanja*.
- Na primer, da bi se potpuno zaštitilo čitanje nekog indeksa, instanca Database Engine-a mora da zaključa određene redove, i da zaključa određene stranice i tabele.
- Microsoft SQL Server Database Engine ima multigranularno zaključavanje što omogućava da transakcija zaključa različite vrste resursa.

Granularnost i hijerarhija zaključavanja

- Koje resurse SUBP može zaključati?
-

Granularnost i hijerarhija zaključavanja

Resurs	Opis
RID	Identifikator reda - zaključava samo jedan red
KEY	Katanac nad redom unutar indeksa – štiti ključne opsege
PAGE	8-kilobajtna strana - podaci ili indeksne stranice.
EXTENT	Susedna grupa od 8 stranica - podaci ili indeksne stranice.
HoBT	Katanac štiti B-tree (indeks) ili skup stranica podataka u tabeli
TABLE	Cela tabela uključujući sve podatke I indekse.
FILE	Fajl baze podataka
APPLICATION	Resurs određen od strane aplikacije
METADATA	Zaključavanje metapodataka
LOCATION_UN IT	Alokacija jedinica
DATABASE	Cela baza podataka

Režimi zaključavanja

Čemu služe režimi zaključavanja?

- Zaključavanje ima različite režime koji specificiraju koji nivo pristupa imaju druge transakcije zaključanim resursima. Microsoft SQL Server Engine zaključava resurse koristeći različite režime zaključavanja koji određuju kako resursima mogu pristupiti transakcije u istom vremenskom intervalu.
-

Koje režime zaključavanja koristi Database Engine?

Režim	Opis
Shared(S)	Koristi se za operacije čitanja koje ne menjaju i ne ažuriraju podatke, kao što je SELECT naredba.
Update(U)	Koristi se nad resursima prilikom ažuriranja. Sprečava čest oblika zastoja (engl. deadlock) do kojeg dolazi kada više sesija čita, zaključava, i potencijalno ažurira resurse.
Exclusive(X)	Služi za operacije modifikovanja podataka, kao što su INSERT, UPDATE ili DELETE. Obezbeđuje da se više ispravki ne mogu desiti u isto vreme nad istim resursom.
Intent	Koristi se za uspostavljanje hijerarhije zaključavanja. Vrste intentova-a su: Intent shared (IS), Intent exclusive(IX) i shared with intent exclusive(SIX).
Schema	Koristi se kada operacija zavisi od šeme tabele koja se izvršava. Vrste schema zaključavanja su: schema modification (Sch-M), i schema stability (Sch-S).
Bulk Update	Koristi se kada se najveći deo podataka kopira u tabelu a TABLELOCK hint (nagoveštaj) je specificiran.
Key Range	Štiti opseg redova pročitanih upitom. Osigurava da druge transakcije ne mogu umetnuti redove koji bi se mogli kvalifikovati za upit serijalizovane transakcije ukoliko bi se taj upit izvršio ponovo.

Kompatibilnost zaključavanja

- ❑ *Kompatibilnost zaključavanja* kontroliše da li više transakcija mogu steći “katance” nad istim podacima u isto vreme.

 - ❑ Ukoliko je resurs već zaključan od strane druge transakcije, zahtev za novim zaključavanjem se može odobriti samo ukoliko je zahtevano zaključavanje kompatibilno sa postojećim režimom.
-

Kompatibilnost zaključavanja

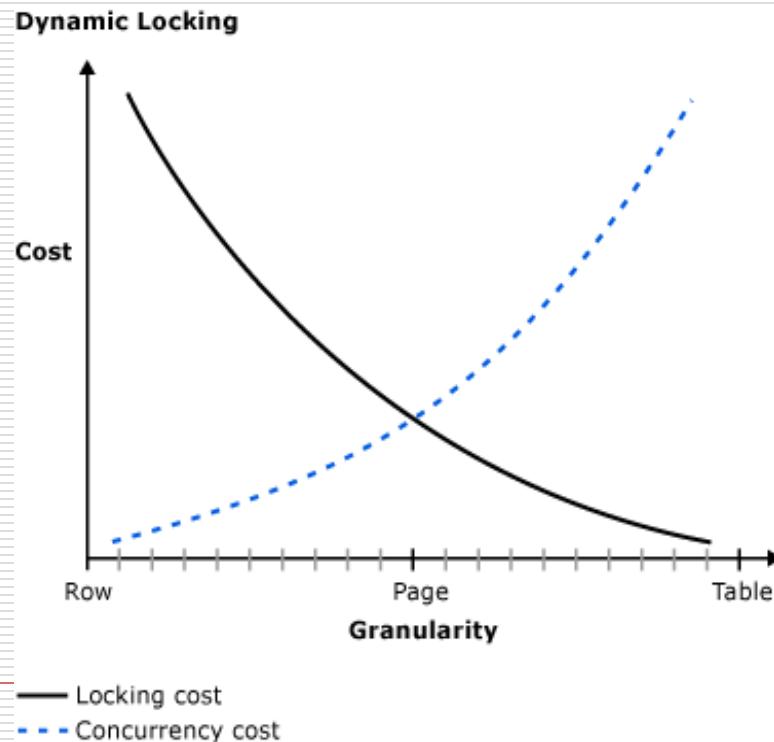
- Matrica kompatibilnosti zaključavanja

režim	Postojeći odobreni					
Zahtevani režim	IS	S	U	IX	SIX	X
Intent Shared (IS)	Yes	Yes	Yes	Yes	Yes	No
Shared (S)	Yes	Yes	Yes	No	No	No
Update (U)	Yes	Yes	No	No	No	No
Intent Exclusive (IX)	Yes	No	No	Yes	No	No
Shared with Intent Exclusive (SIX)	Yes	No	No	No	No	No
Exclusive (X)	No	No	No	No	No	No

Dinamičko zaključavanje

Šta je dinamičko zaključavanje?

- Microsoft SQL Server Database Engine koristi strategiju dinamičkog zaključavanja za utvrđivanje najisplativijih “katanaca”.



Dinamičko zaključavanje

Prednosti dinamičkog zaključavanja

- Pojednostavljena administracija baze podataka. Administratori bazi podataka ne moraju da prilagode "katance" eskalaciji pragova.
 - Povećane performanse. Database Engine minimizira sistemske troškove koristeći zaključavanje koje odgovara zadatku.
 - Developeri aplikacija se mogu koncentrisati na razvoj. Database Engine automatski prilagođava zaključavanje.
-

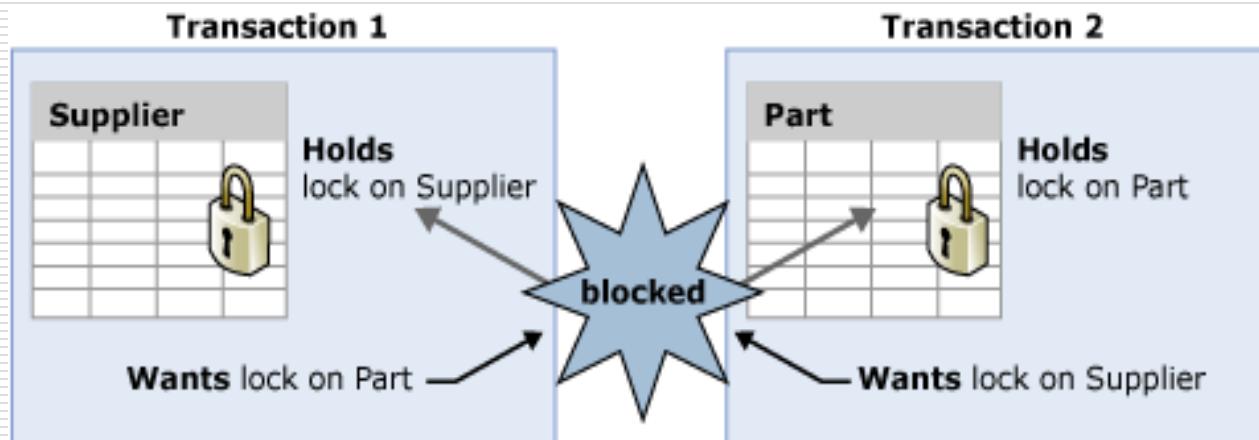
Prikaz informacija o zaključavanju

Tema	Opis
Lock Event Category	Koristeći SQL Server Profiler, možete odrediti kategoriju događaja da bi se prikupile informacije o događajima zaključavanja.
SQL Server, Lock Object	U System Monitor možete odrediti brojače iz objekta zaključavanja da prate nivo zaključavanja u instanci Database Engine-a.
sys.dm_tran_locks (Transact-SQL)	Možete upitom sys.dm_tran_locks dinamično upravljati pogledom da dobijete informacije o trenutnom stanju zaključavanja u instanci Database Engine-a.
EnumLocks	Aplikacija koja koristi SQL Server Management Objects API može dobiti listu aktivnih "katanaca" u instanci Database Engine-a koristeći EnumLocks metode na Server klasi.
EnumLocks	Aplikacija koja koristi SMO API može dobiti listu aktivnih "katanaca" u određenoj bazi podataka koristeći EnumLocks metodu klase Database.

Deadlock

Šta je deadlock?

- *Deadlock* nastaje kada dve ili više transakcija trajno blokiraju jedna drugu tako što jedna transakcija drži “katanac” nad resursom kojeg ostale transakcije pokušavaju da zaključaju.



Deadlock

- ❑ Microsoft SQL Server Database Engine deadlock kontrolor periodično proverava zadatke koji su u zastoju odnosno na mrtvoj tački.
 - ❑ Ukoliko kontrolor detektuje cikličnu zavisnost, on bira jedan od zadataka kao žrtvu, i prekida tu transakciju sa greškom.
 - ❑ Ovo omogućava da drugi zadatak završi svoju transakciju.
 - ❑ Aplikacija sa transakcijom koja se završila sa greškom može ponovo pokušati izvršiti transakciju, što se obično desi kada se druga transakcija na mrtvoj tački, završi.
-

Deadlock

Rukovanje sa deadlock-ovima

- Kada instanca Microsoft Sql Server Database Engine-a izabere transakciju kao žrtvu deadlock-a, ona uništava trenutne serije, izvršava rollback I vraća aplikaciji poruku o greški 1205.
 - Aplikacija treba da ima error handler koji hvata poruku o grešci 1205.
 - Implementacija error handler-a koji hvata poruku o grešci 1205 omogućava aplikaciji da upravlja sa deadlock situacijom, I da preuzme mere za otklanjanje (npr, automatski ponovo poslati upit koji je bio uključen u deadlock).
 - Aplikacija treba kratko pauzirati do ponovnog slanja upita.
-

Administracija Baza podataka

Sigurnost i zaštita baza podataka

Sigurnost - Pojam

- Zaštita baza podataka od neovlašćenog korišćenja.
 - Razlozi:
 - Zakonski propisi, društveni odnosi i etičke norme nalažu da se spreči neovlašćeno korišćenje podataka.
 - Podaci u kompanijama, državnim institucijama i drugim organizacijama moraju biti zaštićeni.
 - U okviru samog SUBP podaci, a i korisnički nalozi i lozinke moraju biti zaštićeni.
-

Sigurnost i zaštita Baza podataka

- ❑ Administracija sigurnosti je jedan od ključnih zadataka administracije baza podataka.
 - ❑ Razumevanje mogućnosti zaštite koje neki SUBP podržava je ključan za obezbeđivanje integriteta kompletног okruženja baze podataka.
 - ❑ Dobra implementacija i rukovanje modelom zaštite pomaže u smanjenju ranjivosti osetljivih podataka u bazi, a povećavaju ukupnu skalabilnost i pouzdanost kompletног okruženja baze podataka.
-

Koncepti sigurnosti baze podataka

- Zaštita baze podataka od neovlašćenog korišćenja odvija se na više nivoa:
 - Zaštita celog sistema;
 - Zaštita na nivou mreže;
 - Zaštita na nivou operativnog sistema;
 - Zaštita na nivou aplikacije;
 - Zaštita unutar same baze podataka.
-

Sigurnost celog sistema

- Sigurnost celog sistema podrazumeva sprečavanje pristupa neovlašćenoj osobi celokupnom sistemu baze podataka, obično se koriste korisnička imena i lozinke.

 - Ovaj koncept obuhvata i sprečavanje fizičkog pristupa mestu gde se nalazi baza podataka.
-

Sigurost baze podataka

- Neki od pristupa koji bazu podataka mogu učiniti sigurnijom:
 - Ograničiti pristup važnim resursima koji mogu biti pogrešno korišćeni – zlonamerno ili slučajno;
 - Onemogućiti nepotrebne komponente i servise sistema za upravljanje bazom podataka;
 - Ukloniti ili onemogućiti predefinisane korisičke naloge;
 - Izvršavati procese baze podataka pod namenskim neprivilegovanim nalozima.
 -

Načelo najmanjih ovlašćenja i razdvajanje

- ❑ Načelo najmanjih ovlašćenja (*least privilege*)
 - Korisnik ima minimalni skup dozvola koje su neophodne za obavljanje tekućeg zadatka
 - Pojedinac ima različite nivoe ovlašćenja u različito vreme zavisno od zadatka ili funkcije koju obavlja;
 - Sprečavanje obavljanja nepotrebnih i eventualno štetnih akcija.
- ❑ Razdvajanje dužnosti (*Separation Of Duty – SoD*)
 - Osetljive zadatke ne može u celini obavljati jedan korisnik;
 - Na taj način se smanjuje mogućnost zloupotrebe.

Mehanizmi zaštite na nivou SUBP

- Identifikacija i dokazivanje autentičnosti (predstavljanje onoga ko pristupa sistemu);
- Autorizacija i kontrola pristupa (ko nešto može da uradi);
- Enkripcija podataka (određivanje ko podatke može videti);
- Praćenje pristupa podacima i zapisivanje (prečenje ko je šta i kada uradio):
 - Upis izvornog teksta korisničkog zahteva;
 - Upis identifikatora klijanta/terminala sa kojeg je pristupljeno;
 - Upis datuma i vremena pristupa bazi;
 - Upis relacije, n-torke i atributa kojima je pristupano;
 - Upis starih i novih vrednosti podataka.

Model sigurnosti same baze podataka

- ❑ Osnovni model sigurnosti se može predstaviti preko skupa autorizacija, koje predstavljaju uredjene četvorke:

Autorizacija (Korisnik, Objekat, Operacija, Uslov)

- ❑ SUBP mora da omogući implementaciju i realizaciju opisanog koncepta autorizacije.
 - ❑ Deo SUBP koji obavlja ovaj zadatak naziva se podsistem sigurnosti ili podsistem autorizacije.
-

Sigurnost baze podataka

KONTROLA PRISTUPA

Kontrola pristupa

- Kontrola pristupa je deo sigurnosne politike koja je prisutna u gotovo svakom sistemu.
- Kad se spominje kontrola pristupa, uzimaju se u obzir četiri situacije:
 - Sprečavanje pristupa;
 - Ograničavanje pristupa;
 - Dozvola pristupa;
 - Oduzimanje prava pristupa

Kontrola pristupa

-  Korisnicima se dodeljuju ovlašćenja (eng. privileges) za povezivanje na bazu i rad sa njenim objektima.
 -  Ovlašćenja korisnicima može dodeljivati administrator baze, vlasnik objekata ili neki drugi autorizovani korisnik kome je dato to pravo.
 -  Ovlašćenja omogućavaju korisnicima da obavljaju određene akcije nad serverom baze i samom bazom podataka (sistemska ovlašćenja) ili objektima baze (objektni ovlašćenja).
-

Sistemska ovlašćenja

- 🔒 Sistemska ovlašćenja najčešće dodeljuje administrator baze podataka.
- 🔒 U ova ovlašćenja spadaju, npr. CREATE DATABASE, CREATE TABLE, CREATE VIEW i CREATE USER, koja dozvoljavaju korisniku da kreira novu bazu podataka, tabelu, pogled i novi korisnički nalog.

Objektna ovlašćenja

- 🔒 Objektna ovlašćenja korisniku omogućavaju da izvrši operacije nad konkretnim objektima baze (kao što su određena baza, tabele, obeležja, pogledi...).
- 🔒 Ako korisnik treba da vidi podatke neke tabele, potrebno mu je dodeliti SELECT ovlašćenje nad tom tabelom (isto važi za INSERT, UPDATE, DELETE, ...).

Modeli kontrole pristupa

- Modeli kontrole pristupa u bazama podataka zasnovani su na generalnoj teoriji zaštite u računarskim sistemima.
- Modeli kontrole pristupa dele se u dve grupe:
 - 🔒 Modeli zasnovani na mogućnostima i
 - 🔒 Modeli zasnovani na listama kontrole pristupa (*ACL – Access Control List*), kojima se definiše lista dozvola nad nekim objektom.

Modeli kontrole pristupa

Kontrola pristupa deli se i prema metodama implementacije na:

- Diskrecioni model (*DAC – Discretionary Access Control*),
- Mandatorni model (*MAC – Mandatory Access Control*) i
- Model grupa i uloga (*RBAC – Role-based access control*)

Model zasnovani na mogućnostima

- 🔒 Na primer, neka postoji računarski sistem u kojem program, da bi pristupio određenom objektu, mora imati posebnu oznaku (eng. token). Oznaka (token) određuje objekat i pruža programu (subjektu) dovoljna ovlašćenja za izvođenje određenog skupa akcija, kao što su čitanje ili pisanje, nad tim objektom.
- 🔒 Mogućnosti se mogu delegirati i kopirati.

Liste kontrole pristupa (ACL)

- Lista kontrole pristupa je uređeni skup podataka o ovlašćenjima ili pravima pristupa svih subjekata (korisnika ili programa) nad bazom podataka.
 - Svaki korisnik ili grupa korisnika ima određena prava pristupa i ovlašćenja nad specifičnim objektom.
 - U sistemima koji imaju velik broj korisnika i gde se stalno menjaju korisnici, nije preporučljivo koristiti liste kontrole pristupa.
 - Diskrecioni i mandatorni modeli mogu koristiti liste kontrole pristupa u svojim implementacijama.
-

Diskrecioni model sigurnosti (DAC)

- ❑ *DAC* je sigurnosni model gde se prava pristupa daje na osnovu korisničkog identiteta.
- ❑ Svakom korisniku se daju specifična prava pristupa za različite objekte.
- ❑ Kreiranje autorizacije:

```
GRANT<lista privilegija> ON <objekat baze>
    TO <identifikator(i) korisnika>
        [WITH GRANT OPTION]
```

- ❑ Privilegije: SELECT, UPDATE, INSERT, DELETE, ALL PRIVILEGES...
 - ❑ Objekti: TABLE i DOMAIN
-

Diskrecioni model sigurnosti (DAC)

- 🔒 Opozivanje privilegije:

```
REVOKE [GRANT OPTION FOR] <lista privilegija>
ON <objekat baze> FROM <identifikator(i) korisnika>
[RESTRICT | CASCADE]
```

- 🔒 Opcija GRANT OPTION FOR se koristi kada se opoziva mogućnost prenosa privilegija.
- 🔒 RESTRICT – onemogućava opozivanje privilegija ukoliko su prenete drugim korisnicima.
- 🔒 CASCADE – opoziva pored navedenih privilegija i sve prenete privilegije.

Diskrecioni model sigurnosti (DAC)

- Napomena: Vlasnik šeme baze podataka (korisnik koje je kreirao šemu) ima sve privilegije u njoj.

Primer: *User1* je vlasnik šeme *Drzava*.

Drzava(DID, Naziv, Uredjenje)

Grad(DID, GID, Naziv)

1. GRANT SELECT, INSERT ON Drzava TO *User2*
WITH GRANT OPTION

2. GRANT SELECT, UPDATE ON Grad To *User3*

Mandatorni model (MAC)

- MAC je sigurnosni model gde korisnicima prava na resurse može da da samo administrator ili operativni sistem.
 - Subjektima i objektima je dodeljen skup sigurnosnih prava ili klasifikacija.
 - Pristup određenim resursima dozvoljen je samo subjektima s dodeljenom klasifikacijom.
 - Mandatorni model pristupa koristi se kod obrađivanja vrlo osetljivih podataka, kao što su poverljive informacije vladinih i vojnih organizacija.
-

Model zasnovan na ulogama (RBAC)

- *Role Based Access Control (RBAC)* – Način kontrole pristupa zasnovan na ulogama i pravima.
 - U ovom sigurnosnom modelu, pristup resursima je baziran na ulozi korisnika koju je dobio od administratora.
 - Administrator dodeljuje ulogu koja dolazi sa već određenim pravima i privilegijama, i korisnik može da vrši samo akcije dodeljene pravilima uloge.
 - *RBAC* je novija alternativa *DAC* i *MAC*.
 - *RBAC* model nudi mogućnost implementacije i *MAC* modela i *DAC* modela kontrole pristupa.
-

Princip minimalnih ovlašćenja

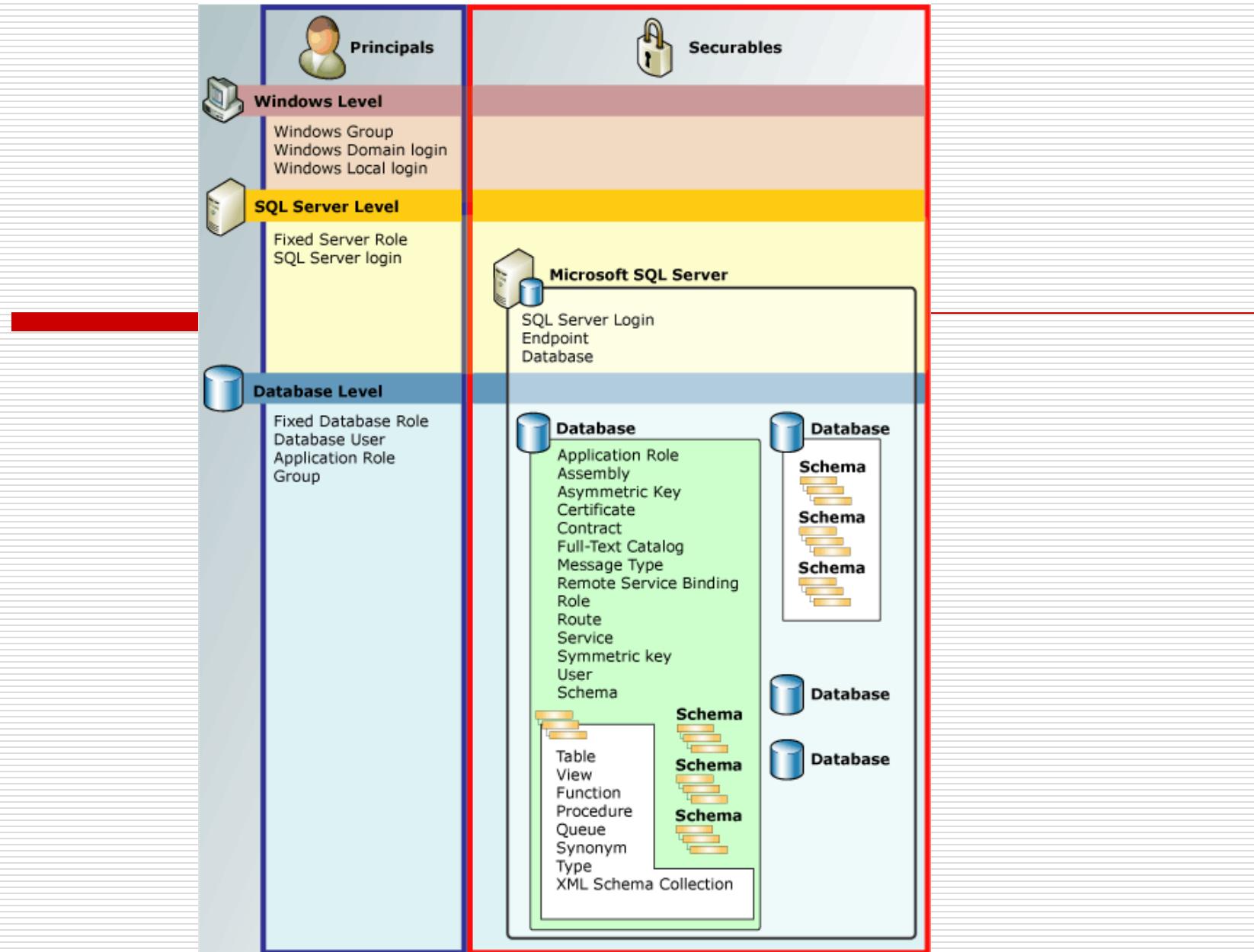
- Princip minimalnih ovlašćenja predviđa da korisnicima treba dodeliti samo minimum ovlašćenja potrebnih za obavljanje njihovih poslova nad bazom. To takođe predviđa:
 - Upotrebu uloga, koje sadrže grupe (skup) ovlašćenja i olakšavaju administraciju;
 - Upotrebu pogleda, koji ograničavaju pristup na definisane podskupove postojećih podataka;
 - Upotrebu uskladištenih procedura čijom se upotrebom može izbeći dodela konkretnih prava nad baznim tabelama korisnicima.
-

Sigurnost baze podataka

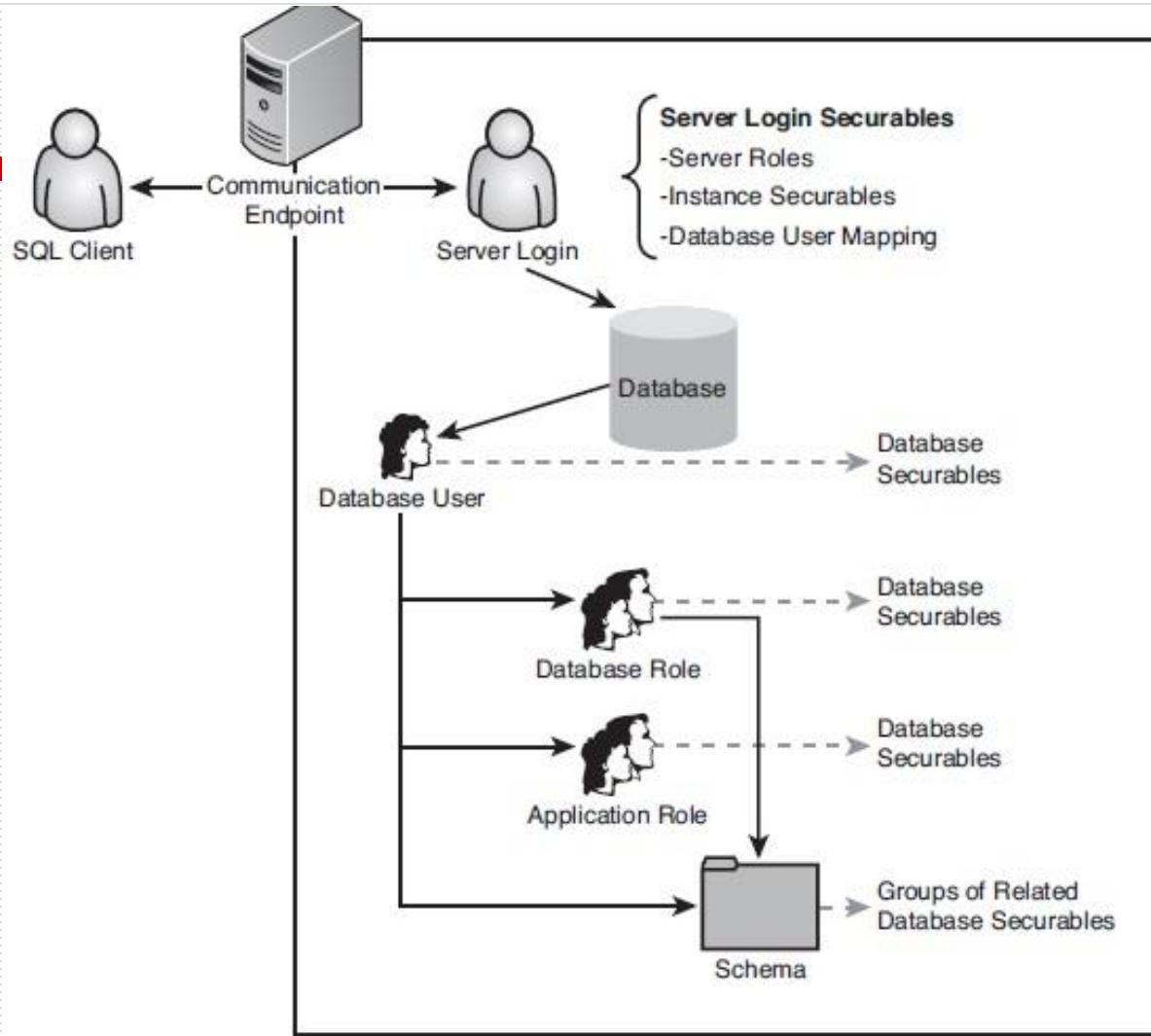
SIGURNOST kod MICROSOFT SQL SERVERA

SQL Server Security Overview

- ❑ Višenivojski model zaštite (*Layered Security Model*):
 - Windows Level
 - SQL Server Level
 - Database
 - Schemas (for database objects)
- ❑ Terminologija:
 - Principals – Objekti BP kojima se dodeljuju prava
 - Securables – Objekti baze podataka koji se štite
 - Permissions – Dozvole (prava pristupa)
 - Scopes and Inheritance



MS SQL Server - Security



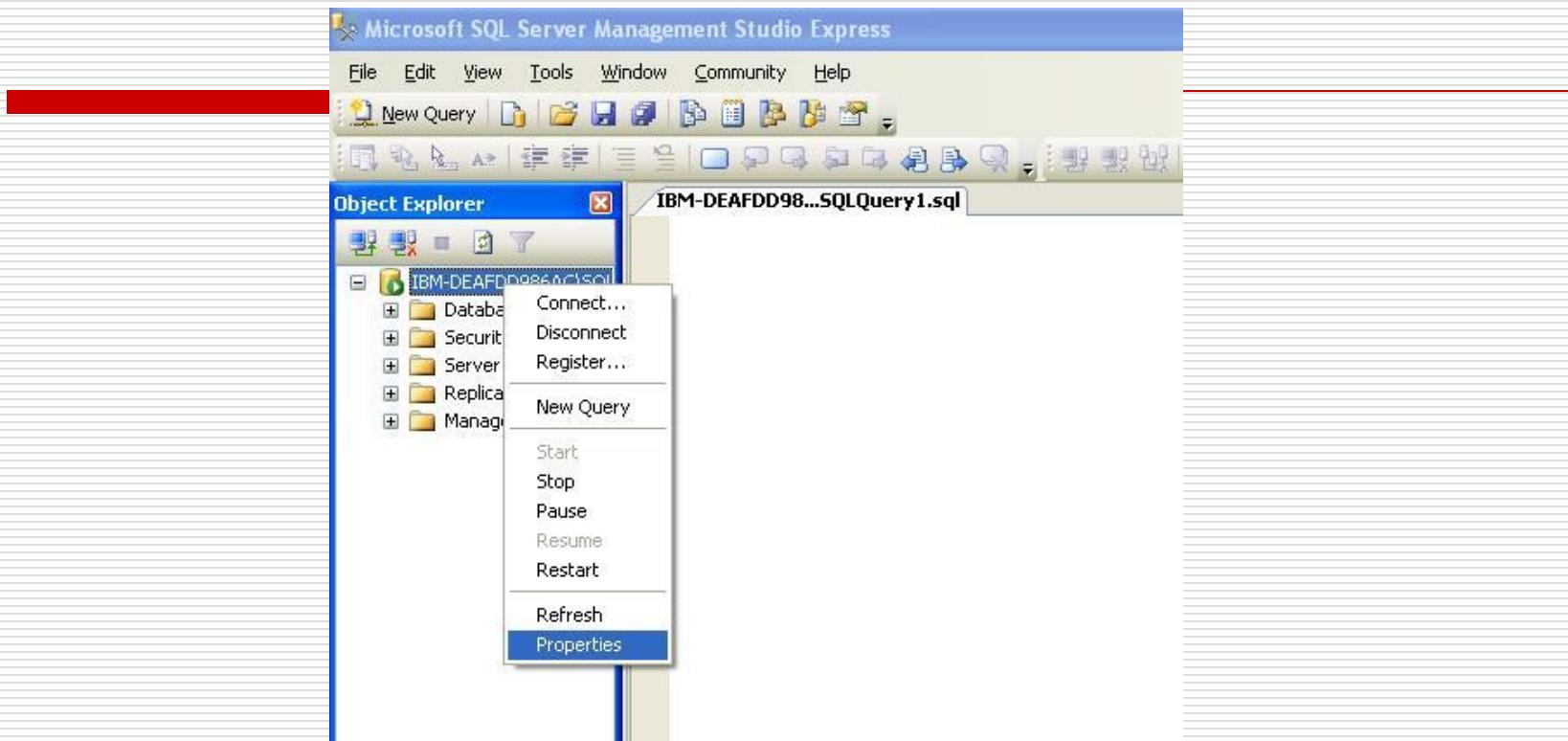
Preporuke - najbolja praksa -

- Administracija zaštite baze podataka treba da bude deo procesa standardne administracije baze podataka.
 - Koristiti princip najmanjih privilegija.
 - Implementirati dubinsku zaštitu na više nivoa.
 - Dozvoliti (*enejblovati*) samo potrebne servise i komponente.
 - Redovno pratiti podešavanja zaštite.
 - Obučavati korisnike o važnosti zaštite.
 - Definisati zaštitne uloge u sistemu na osnovu poslovnih pravila koja važe u sistemu.
-

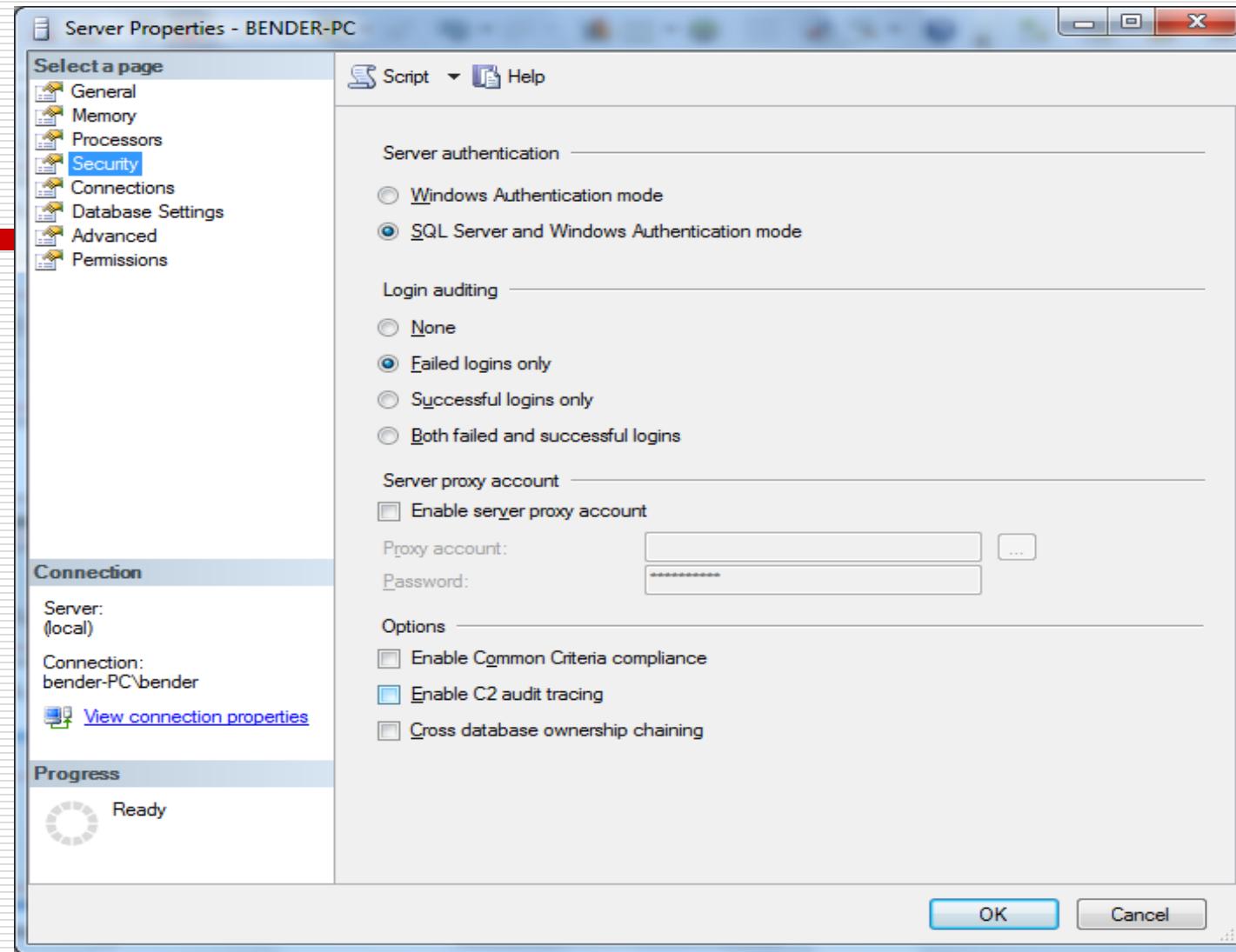
MS SQL Server - Režimi bezbednosti

- SQL Server poseduje dva načina na koje se korisnik (takođe i aplikacija ili Web sajt), može povezati.
 - SQL Server autentifikacija
 - Windows autentifikacija
 - Način autentifikacije se definiše pri instalaciji ali se može promeniti promenom parametara konfiguracije SQL Servera.
-

Konfigurisanje SQL Servera



Promena načina Autentifikacije



SQL Server autentifikacija

- Znači da je potrebno upisati korisničko ime i šifru za povezivanje.
 - Ovaj način povezivanja se koristi u slučaju peer to peer mreža, bez centralizovane lokacije na kojoj se čuvaju korisnički računi – aktivni direktorijum.
-

Windows autentifikacija

- ❑ Koristi se ime korisnika i šifra preko kojih se korisnik ulogovao na Windows.
 - ❑ Ideja iza ovoga je da jednom prijavljen korisnik na Windows mrežu, iste kredencijale koristi i za prijavu na SQL Server.
 - ❑ U predhodnom nastavku smo prilikom instalacije dodali korisnika Administrator, što znači da svako ko se na Windows prijavi kao administrator, automatski ima prava prijave i na SQL Server.
 - ❑ Ovo se može uraditi za bilo koji korisnički račun ili grupu na lokalnom računaru, kao i u domenskoj mreži.
-

Pristup MS SQL Serveru

- Bez obzira na način prijave koji je izabran, pre svega je neophodno uneti ime SQL Servera na koji želimo da se povežemo.
 - Standardno, ime SQL Servera je isto kao ime računara na koji je instaliran. Isto važi i za njegovu IP adresu.
 - Osnovni protokol koji koristi SQL Server je TCP/IP na portu 1433, što je značajan podatak u slučaju da postoji FireWall.
 - Umesto Imena SQL Servera na koji se vezujemo uvek se može uneti njegova IP adresa, kao i ime računara na kojem je instaliran (što se opet prevodi u njegovu IP adresu). Izuzetak od ovoga je slučaj kada se SQL Server nalazi na Internetu i tada se mora uneti njegova IP adresa.
-

Pristup MS SQL Serveru

- Postoje i dve skraćenice kada je u pitanju lokani SQL Server.
Može se upisati (**local**) ili jednostavno tačka(.)).
- Obe skraćenice se interno prevode na IP adresu **127.0.0.0** što označava lokalnu IP adresu lokalnog računara.
- SQL Server Express edition po inicijalnoj instalaciji predstavlja mali izuzetak od ovog pravila jer se njegovo ime formira po sistemu **ImeRačunara\SQLEXPRESS** kako bi se razlokovalo od "velikog" SQL Servera.

Logins and Users

- ❑ Kod SQL Servera postoje dva nivoa provere prava pristupa:
 1. Logins – Korisnički nalog na nivou cele instance SQL Servera.
 2. Users – Korisnički nalozi za posebnu bazu podataka kojom rukuje instanca SQL Servera.

Logins and Users

- Da bi se uspešno povezao sa SQL Serverom korisnik koji se povezuje na SQL Server mora imati oba naloga:
 - Login nalog za instancu SQL Servera i
 - User nalog za bazu podataka.
 - U protivnom neće moci da se konektuje na SQL Server.
 - Pored toga postoje i Windows korisnički nalozi kojima se rukuje na nivou domena mreže i za njih nije odgovoran DBA već Administrator sistema.
-

Dozvola pristupa servisima SQL Servera

- Dozvola pristupa servisima SQL Servera izvodi se u četiri koraka:
 1. Kreira se Login nalog.
 2. Kreira se User nalog – jedan za svaku bazu podataka kojoj se zahteva pristup.
 3. Pridruživanje Login naloga User nalogu.
 4. Definisanje prava pristupa i uloga (*Rols*) User nalogu.

System Administrator (sa) Login

- ❑ U procesu instalacije SQL Servera, ako se izabere *Mixed Mode authentication*, sistem zahteva da se postavi *Password* za **sa** predefinisani administrativni *Login* nalog.
- ❑ Potrebno je u procesu instalacije odmah dodeliti *Password* za **sa** *Login* nalog da bi se sprečio neautorizovani pristup instanci SQL Server-a korišćenjem **sa** *Login* naloga.
- ❑ Ukoliko se pri instalaciji izabere *Windows Authentication* mod potrebno je posle instalacije dodeliti *Password* za **sa** *Login* nalog da bi osigurali da **sa** *Login* nalog ima *Password* u slučaju da se kasnije promeni način autentifikacije u *Mixed Mode authentication*.
- ❑ Ne koristiti *blank password*.

System Administrator (sa) Login

- ❑ *System administrator (sa)* se koristi da bi se obezbedila kompadibilnost sa prethodnim verzijama softvera.
 - ❑ Inicialno *Login nalogu (sa)* se pridružuje **sysadmin fixed server role** i to se ne može menjati. Iako je *sa built-in administrator login* ne treba ga rutinski koristiti. Umesto toga potrebno je kreirati poseban administratorski login i pridružiti ga **sysadmin** serverskoj ulozi.
 - ❑ *sa built-in login* nalog koristiti samo kada nema drugog načina da se pristupi instanci SQL Server-a.
-

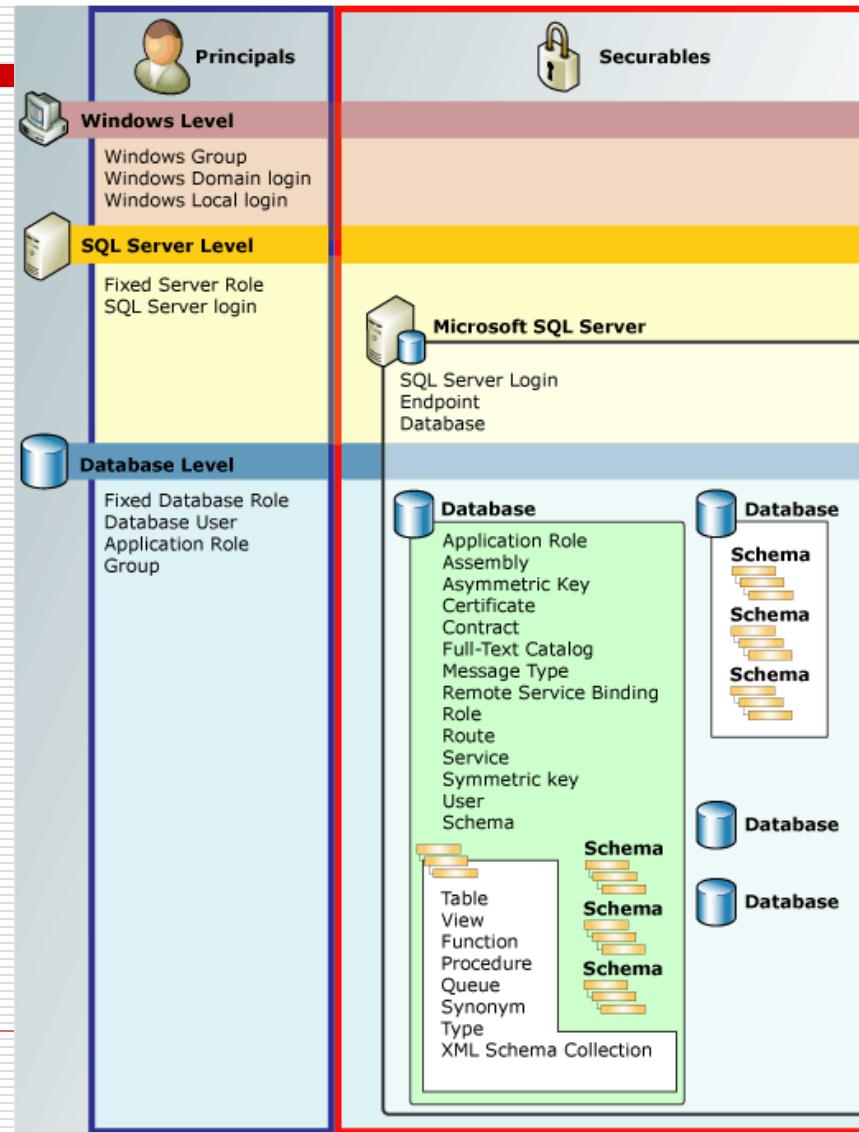
Identity and Access Control

- Kod SQL Server-a postoje različite metode i alati za konfiguriranje zaštite za korisnike, servise i druge naloge za pristup sistemu.
- Ovde su dati osnovni pojmovi koji se tiču rukovanja zaštitom kod SQL Server-a koje je potrebno poznavati
 - *Principals* (korisnici i korisnički nalozi)
 - *Roles* (groups of *Principals*);
 - Objekti zaštite (*Securables*)
 - Dozvole (*Permissions*)
- *Principals* - Opisuju entitete koji mogu zahtevati resurse SQL Servera
- *Server-Level Roles* – Opisuju uloge na nivou SQL Server-a.
- *Database-Level-Roles* – Opisuju uloge u SQL Serveru na nivou Baze podataka.

Identity and Access Control

- ❑ *Principals* – Subjekti kojima se dodeljuju prava pristupa.
 - ❑ *Securables* – Objekti čija zaštita se vrši.
 - ❑ *Permissions* – Dodeljena prava pristupa subjektima nad objektima zaštite.
-

Identity and Access Control



Principals (Database Engine)

- *Principals* – entiteti koji mogu zahtevati resurs SQL Servera.
 - Kao i kod drugih komponenti autorizacionog modela SQL Servera *Principals* mogu biti hijerarhijski organizovani.
 - Svaki Principal ima svoj *Security Identifier (SID)*.
-

Principals

- ❑ *Windows-level principals*
 - Windows Domain Login
 - Windows Local Login
 - ❑ *SQL Server-level principal*
 - SQL Server Login
 - ❑ *Database-level principals*
 - Database User
 - Database Role
 - Application Role
-

Database Users- Korisnici Baze podataka

- Korisnik (*User*) baze podataka je *Principal* na nivou baze podataka. Svakom korisniku baze podataka je dodeljena ***public role***.
- Pri kreiranju baze podataka po *default*-u se kreira korisnik ***guest***. Prava pristupa dodeljena ***guest*** user-u se dodeljuju korisnicima koji nemaju korisnički nalog u bazi podataka.
- ***guest*** user se ne može ukloniti (izbrisati) ali se mogu ukinuti prava konekcije.
- Dozvola konekcije se može ukinuti izvršavanjem REVOKE CONNECT FROM GUEST u bilo kojoj bazi podataka SQL Servera osim u ***master*** ili ***tempdb*** sistemskim bazama podataka.

Server-Level Roles - Uloge na nivou Servera

- ❑ Da bi se jednostavnije dodeljivala prava na nivou SQL Servera, SQL Server obezbeđuje nakoliko uloga (*roles*). Uloge su slične grupama kod Windows operativnog sistema.
 - ❑ Serverskim ulogama se dodeljuju prava pristupa na nivou instance SQL Servera.
-

Mogućnosti uloga na nivou servera(1)

- ❑ **sysadmin** – Članovi ove uloge mogu izvršavati bilo koju aktivnost na nivou servera.
 - ❑ **serveradmin** – Mogu menjati širok skup konfiguracionih parametara servera i vršiti zaustavljanje rada servera.
 - ❑ **securityadmin** – Mogu upravljati login nalozima i njihovim parametrima. Članovi ove uloge mogu izvršavati GRANT, DENY, i REVOKE naredbe za prava pristupa na nivou servera. Osim toga mogu resetovati password za login naloge SQL Server-a.
-

Mogućnosti uloga na nivou servera(2)

- ❑ **processadmin** – Članovi ove uloge mogu upravljati procesima koji se izvršavaju u okviru instance SQL Server-a.
 - ❑ **setupadmin** – Članovi ove uloge mogu dodavati i uklanjati povezane servera.
-

Mogućnosti uloga na nivou servera(3)

- ❑ **bulkadmin** – Članovi ove uloge mogu izvršavati BULK INSERT naredbe.
 - ❑ **diskadmin** – Ova serverska uloga se koristi za upravljanje datotekama baze podataka na disku.
 - ❑ **dbcreator** – Članovi ove uloge mogu vršiti kreiranje, izmenu, uklanjanje i vršiti *restore* bilo koje baze podataka.
 - ❑ **public** – Svaki login nalog SQL Servera pripada *public* serverskoj ulozi. Kada subjekt na nivou servera (*principal*) nema dodeljena i zabranjena neka posebna prava nad objektima zaštite, korisnik nasledjuje prava dodeljena *public* ulozi nad tim objektom.
-

Database-Level Roles

- ❑ Kod SQL Servera postoje dve grupe uloga na nivou baze podataka:
 - Fiksne uloge koje su unapred definisane u bazi podataka i
 - Promenljive uloge na nivou baze podataka koje se mogu kreirati od strane administratora.

Database-Level Roles

- ❑ Fiksne uloge na nivou baze podataka postoje u svakoj bazi podataka. Članovi ***db_owner*** i ***db_securityadmin*** mogu upravljati pridruživanjem fiksnih uloga baze podataka korisnicima baze podataka. Međutim, samo članovi uloge ***db_owner*** mogu pridruživati članove u fiksnu ulogu ***db_owner*** baze podataka. Osim toga, u *msdb* sistemskoj bazi podataka postoje neke uloge posebne namene.
 - ❑ ***db_owner*** – Članovi ove uloge mogu izvršavati sve konfiguracione aktivnosti nad bazom podataka, mogu izvršavati i *Drop Database* naredbu.
 - ❑ ***db_securityadmin*** – Članovi ove uloge baze podataka mogu modifikovati pripadnost korisnika baze podataka ulogama i upravljati pravima pristupa (privilegijama).
-

Database-Level Roles

- ***db_accessadmin*** – Članovi ove uloge mogu dodavati ili uklanjati pravo pristupa bazi podataka za Windows login naloge, Windows grupe i SQL Server login naloge
 - ***db_backupoperator*** – Članovi ove uloge mogu vršiti uzimanje rezervnih kopija baze podataka (*Backup*)
 - ***db_ddladmin*** – Članovi ove grupe imaju pravo izvršavanja bilo koje DDL naredbe nad bazom podataka.
-

Database-Level Roles

- ***db_datawriter*** – Članovi ove uloge mogu dodavati, brisati ili menjati podatke u svim korisničkim tabelama baze podataka.
- ***db_datareader*** – Članovi ove uloge imaju pravo čitanja iz svih korisničkih tabela baze podataka.

Prava nad objektima

- Three separate object permissions exist in SQL Server:
 1. **Grant** – can perform action
 2. **Deny** – cannot perform action (strong). This applies even if the user account is a member of a role which has been granted the permission.
 3. **Revoke** – cannot perform action (weak). This will be overridden by a grant to a role which the user account is a member of.
 - So Grant and Revoke are the same as in oracle. The additional Deny command is an extra strong form of Revoke.
-

Application Roles

- *Application role* je *principal* baze podataka koji omogućava aplikaciji da se aplikacijama dodeljuju dozvole (prava) kao korisnicima baze podataka
- Ova uloga omogućava pristup podacima samo onim korisnicima koji se na bazu podataka konektuju preko neke aplikacije.
- Za razliku od uloga baze podataka aplikacione uloge ne sadrže članove i nisu aktivne po default-u. Rade u oba režima autentifikacije SQL Servera.
- Aplikacione uloge se eneLBLUJU korišćenjem ***sp_setapprole***, koja zahteva *password*.

Application Roles

- Obzirom da je *application role* uloga na nivo baze podataka, preko nje se može pristupiti drugim bazama podataka samo na osnovu dozvola koje su u toj bazi podataka dodeljene **guest** korisniku.
 - Zbog toga, neće se moći pristupiti drugoj bazi podataka preko *application role* ako je u noj disejblovan **guest** korisnik.
-

Connecting with an Application Role

- The following steps make up the process by which an application role switches security contexts:
 - A user executes a client application.
 - The client application connects to an instance of SQL Server as the user.
 - The application then executes the **sp_setapprole** stored procedure with a password known only to the application.

Connecting with an Application Role

- If the application role name and password are valid, the application role is enabled.
- At this point the connection loses the permissions of the user and assumes the permissions of the application role.
- The permissions acquired through the application role remain in effect for the duration of the connection.

How to: Create a SQL Server Login

- Most Windows users need a SQL Server login to connect to SQL Server. This topic shows how to create a SQL Server login.
- To create a SQL Server login that uses Windows Authentication (SQL Server Management Studio)

How to: Create a SQL Server Login

1. In SQL Server Management Studio, open Object Explorer and expand the folder of the server instance in which to create the new login.
2. Right-click the Security folder, point to New, and then click Login.
3. On the General page, enter the name of a Windows user in the Login name box.
4. Select Windows Authentication.
5. Click OK.

How to: Create a SQL Server Login

- To **create a SQL Server login that uses SQL Server Authentication** (SQL Server Management Studio)

1. In SQL Server Management Studio, open Object Explorer and expand the folder of the server instance in which to create the new login.
2. Right-click the Security folder, point to New, and then click Login.
3. On the General page, enter a name for the new login in the Login name box.
4. Select SQL Server Authentication. Windows Authentication is the more secure option.
5. Enter a password for the login.
6. Select the password policy options that should be applied to the new login. In general, enforcing password policy is the more secure option.
7. Click OK.

How to: Create a Database User

- To create a database user using SQL Server Management Studio
 1. In SQL Server Management Studio, open Object Explorer and expand the Databases folder.
 2. Expand the database in which to create the new database user.
 3. Right-click the Security folder, point to New, and then click User.
 4. On the General page, enter a name for the new user in the User name box.
 5. In the Login name box, enter the name of a SQL Server login to map to the database user.
 6. Click OK.

How to: Create a Database User

- Kreiranje korisnika baze podataka korišćenjem Transact-SQL naredbe
 1. Potrebno je u *Query Editor*-u izvršiti konekciju na bazu podataka u kojoj želimo da kreiramo novog korisnika baze podataka izvršavanjem sledeće Transact-SQL komande:

USE <database name> GO

1. Kreiranje korisnika baze podataka može se izvršiti sledećom Transact-SQL naredbom:

CREATE USER <new user name>
FOR LOGIN <login name>

Kreiranje korisnika baze podataka(1)

- Administracija korisnika baze podataka može se vršiti korišćenjem:
 - Transact-SQL komande ili
 - SQL Server Management Studio

Kreiranje korisnika baze podataka(2)

- Korisnik je subjekt zaštite (Principal) na nivou baze podataka.
- Login nalozi se mapiraju korisnike baze podataka da bi se izvršila konekcija na bazu.
- Login nalozi mogu se mapirati na različite baze podataka preko različitih korisnika, ali se na svaku od baza mogu mapirati kao samo jedan korisnik u svakoj bazi podataka.

Kreiranje korisnika baze podataka(3)

- Ukoliko je u bazi podataka *enejblovan* korisnik **guest**, Login nalog koji nije mapiran na korisnika baze podataka može bazi podataka pristupiti kao **guest** korisnik .
- Napomena – **guest** korisnik je inicijalno *disejblovan* i ne treba ga *enejblovati* ukoliko to nije neophodno.
- Da bi se konektovalo na neko od baza podataka instance SQL Servera Login nalog mora biti mapiran na korisnika baze podataka.
- Dozvole, odnosno prava pristupa dodeljuju se korisnicima baze podataka a ne Login nalozima.

Using SQL Server Management Studio

- To create a database user
 1. In Object Explorer, expand the Databases folder.
 2. Expand the database in which to create the new database user.
 3. Right-click the Security folder, point to New, and select User....
 4. In the Database User – New dialog box, on the General page, select one of the following user types from the User type list: SQL user with login, SQL user without login, User mapped to a certificate, User mapped to an asymmetric key, or Windows user.

Using SQL Server Management Studio

5. In the User name box, enter a name for the new user. If you have chosen Windows user from the User type list, you can also click the ellipsis (...) to open the Select User or Group dialog box.
 6. In the Login name box, enter the login for the user. Alternately, click the ellipsis (...) to open the Select Login dialog box. Login name is available if you select either SQL user with login or Windows user from the User type list.
-

Using SQL Server Management Studio

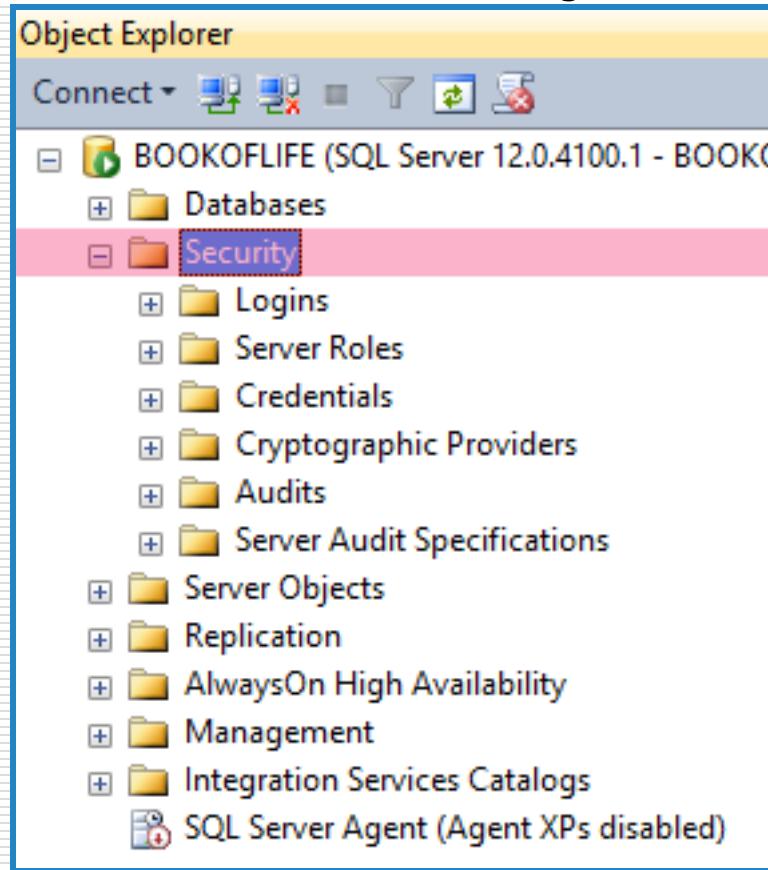
- In the Default schema box, specifies the schema that will own objects created by this user. Alternately, click the ellipsis (...) to open the Select Schema dialog box. Default schema is available if you select either SQL user with login, SQL user without login, or Windows user from the User type list.
 - In the Certificate name box, enter the certificate to be used for the database user. Alternately, click the ellipsis (...) to open the Select Certificate dialog box. Certificate name is available if you select User mapped to a certificate from the User type list.
-

Administracija sigurnosti kod SQL Servera

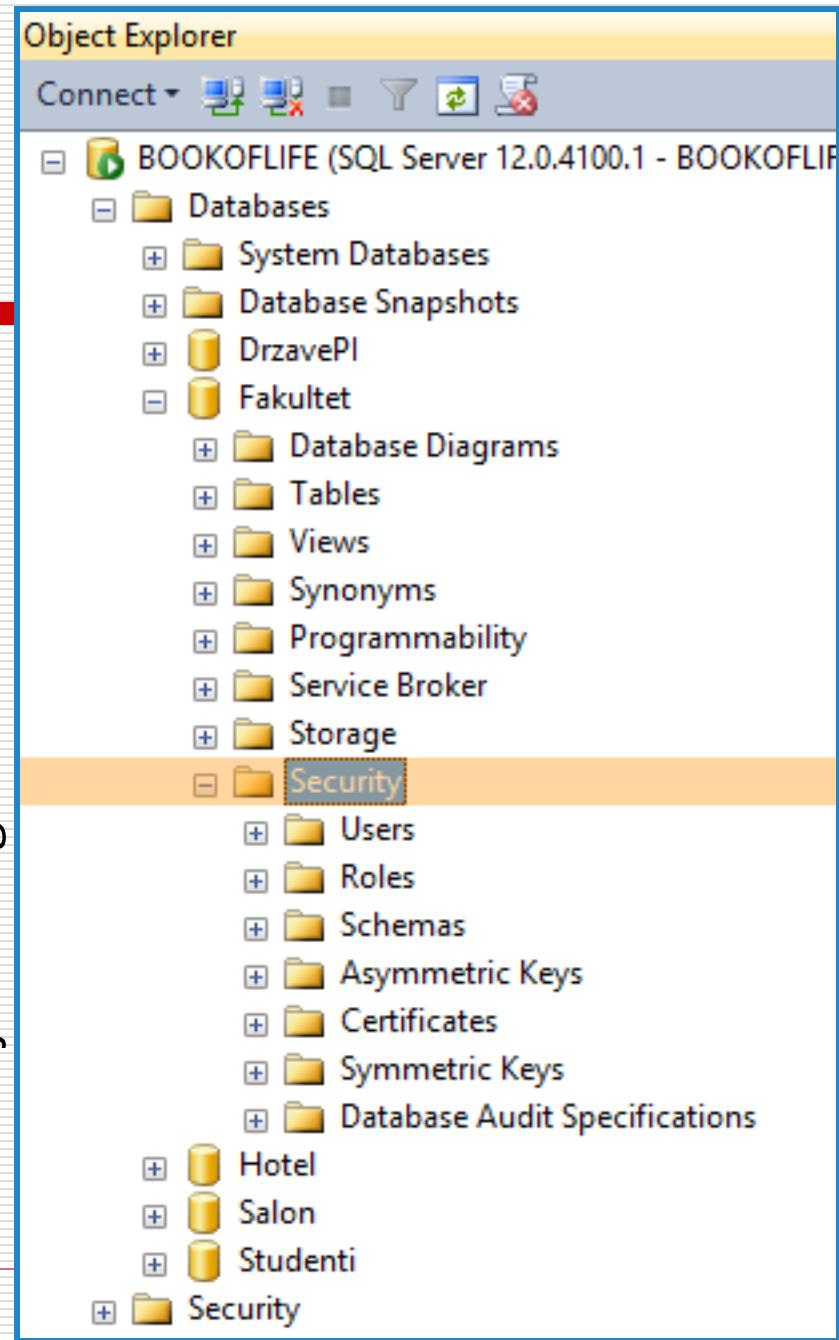
Korišćenje SQL Server Management Studio-a

Sistemska i objektna sigurnost

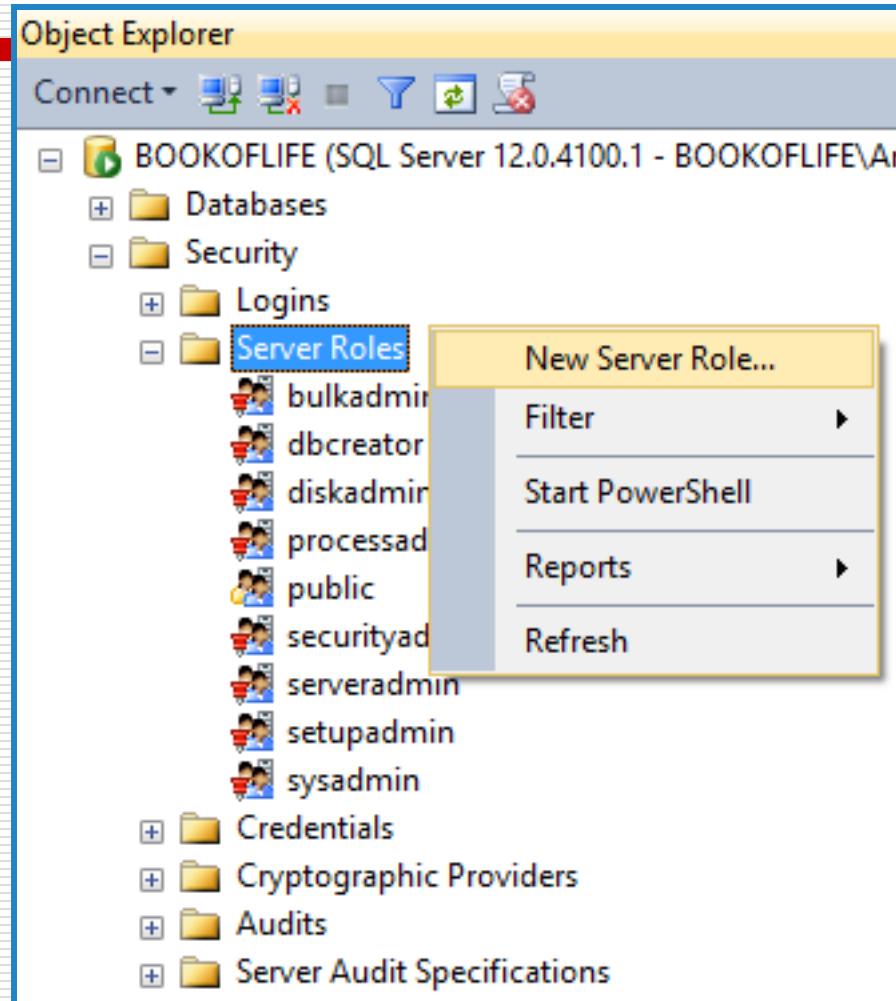
Slika 1: Sistemska sigurnost



Slika 2: Objektna sigurnost nad BP

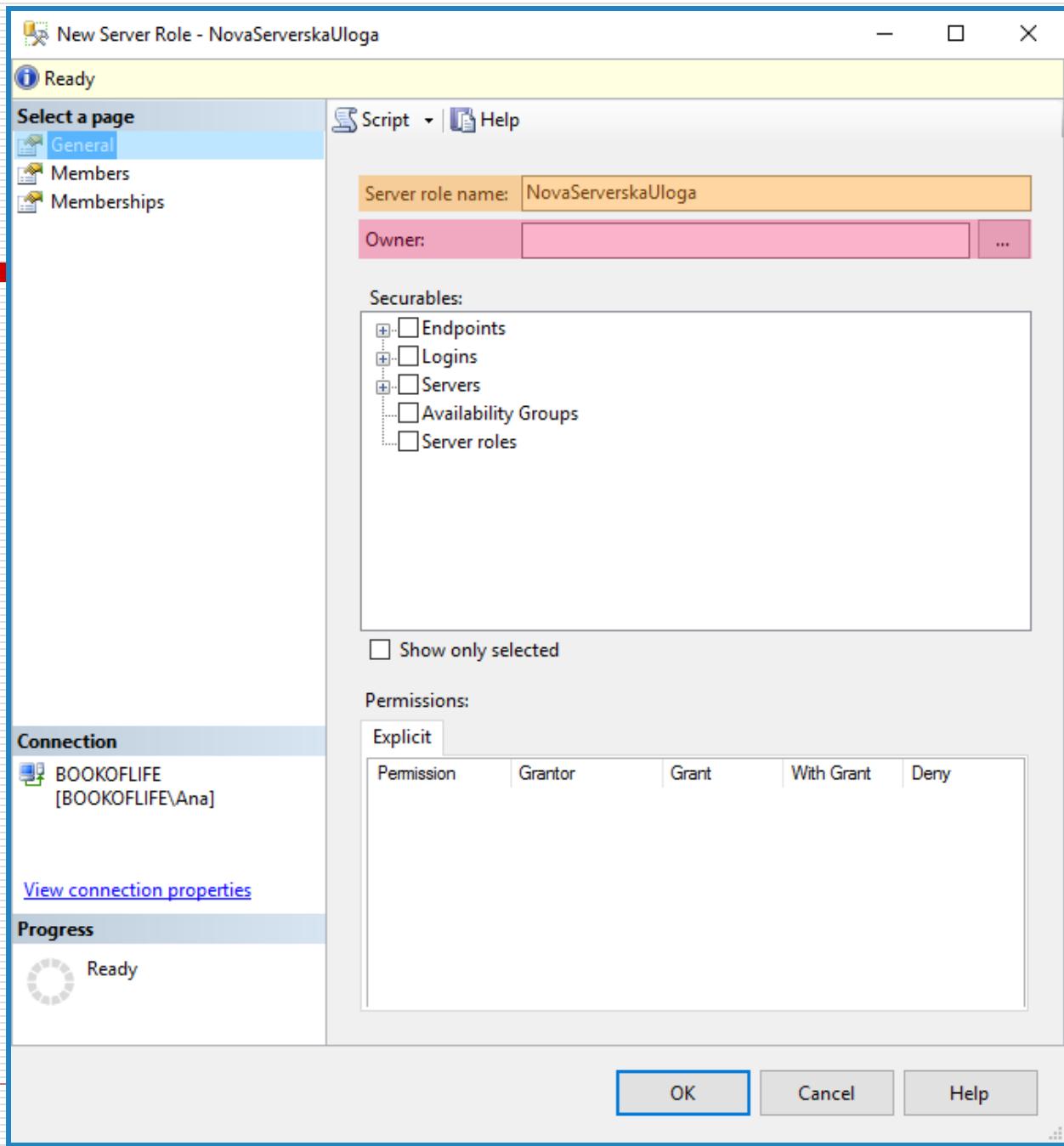


Sistemska sigurnost: kreiranje serverske uloge



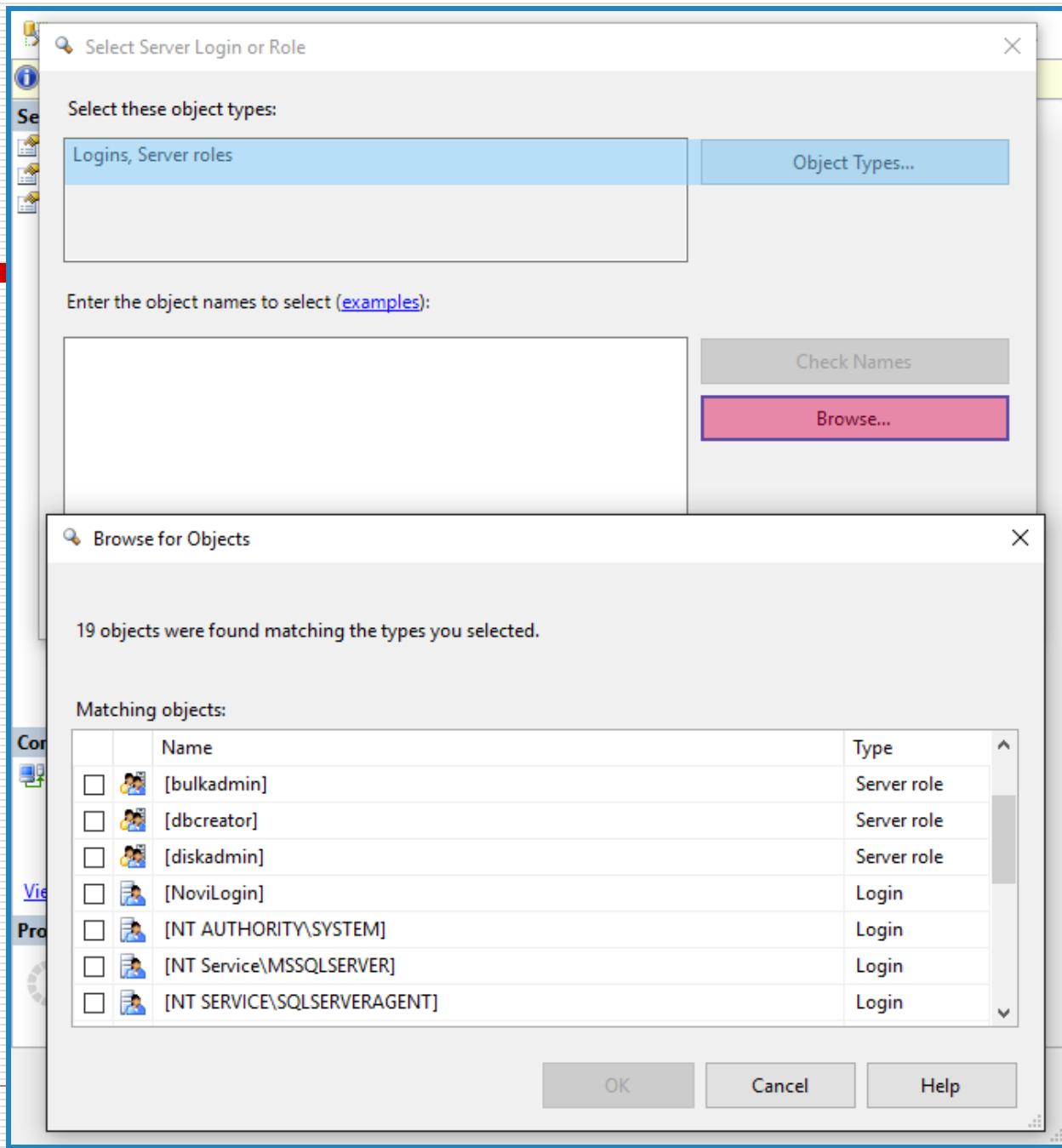
Slika 3: New Server Role

Sistemska sigurnost: kreiranje serverske uloge



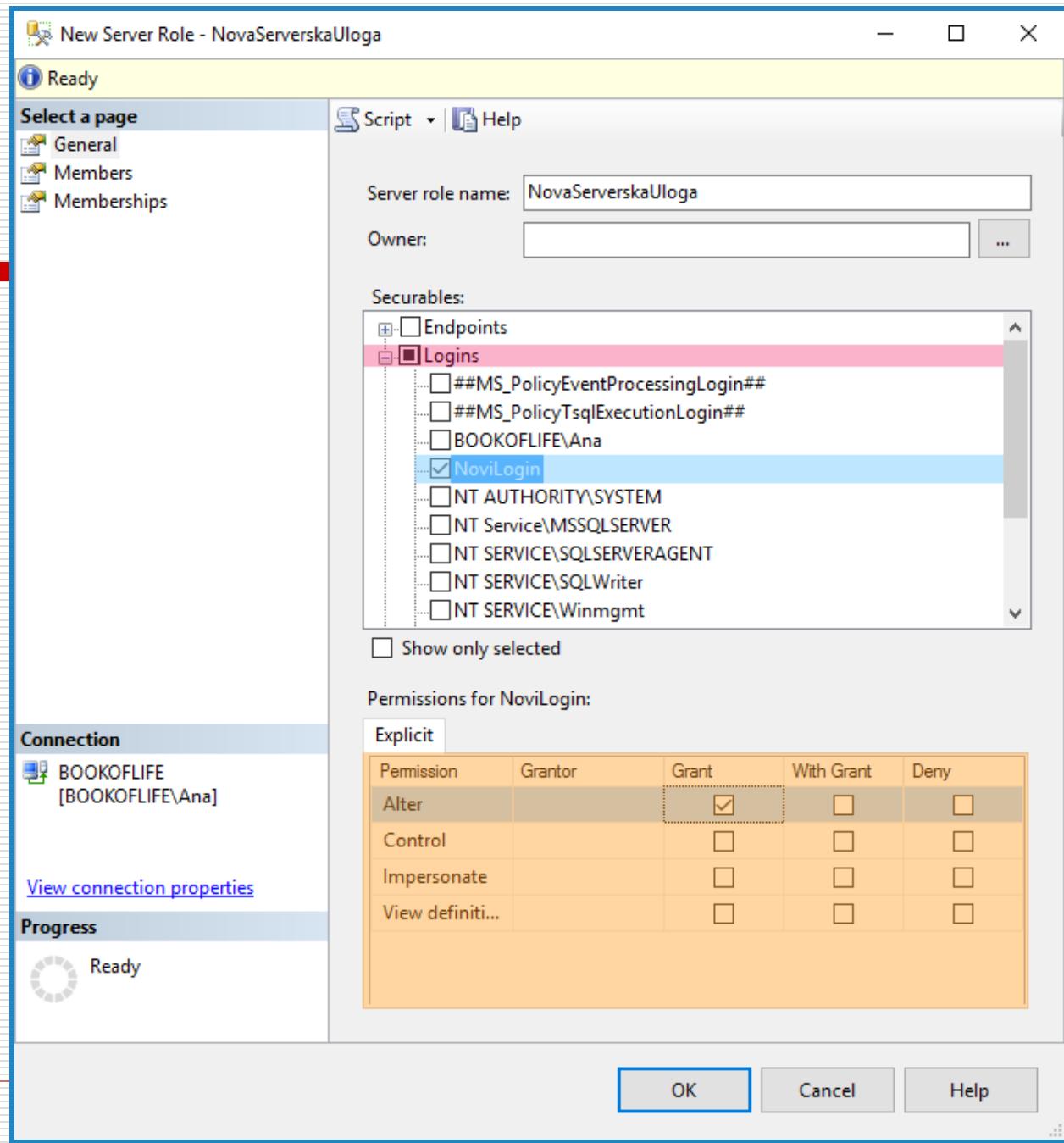
Slika 4: New Server Role

Sistemská sigurnost: kreiranje serveriske uloge



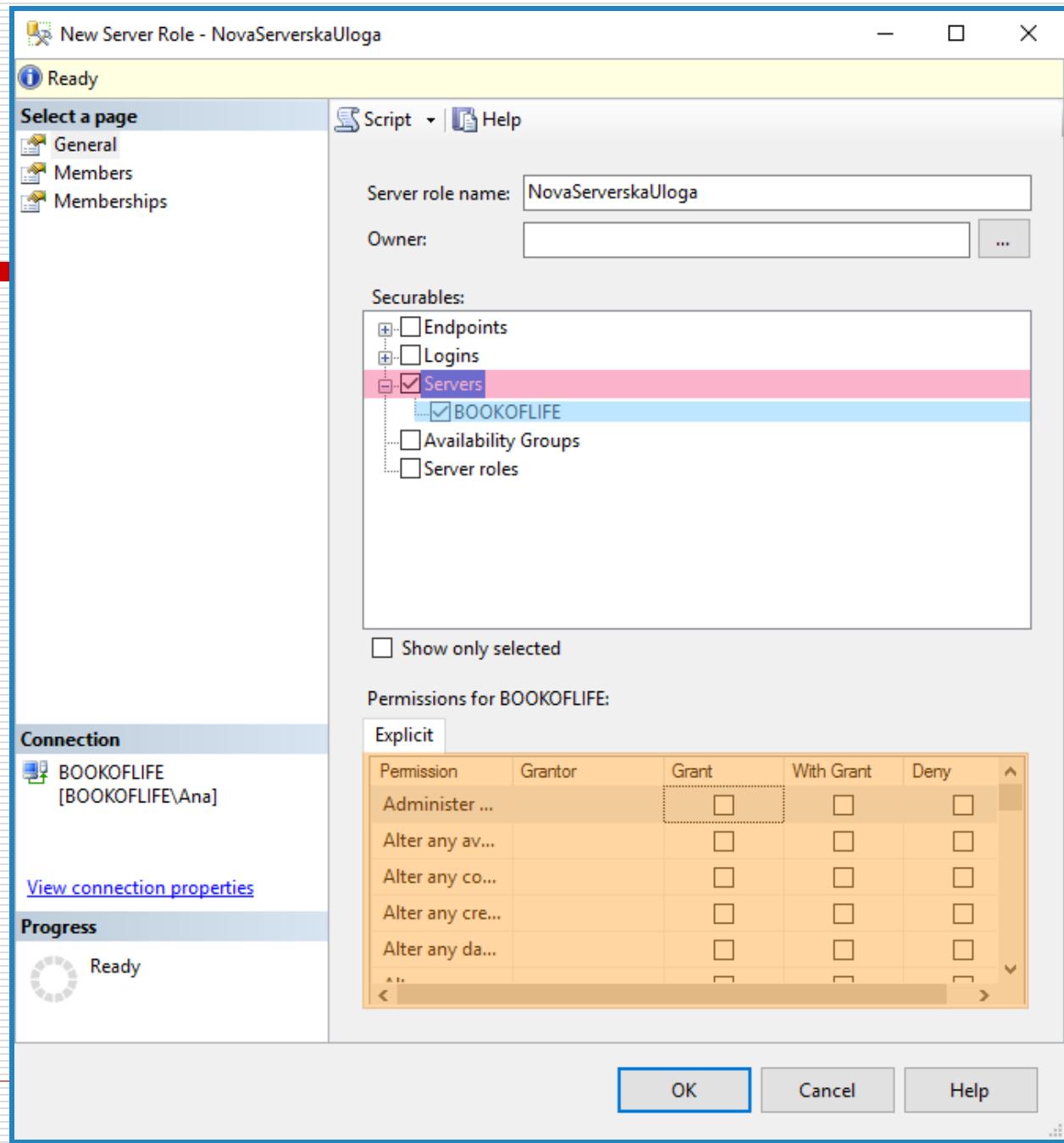
Slika 5: New Server Role

Sistemska sigurnost: kreiranje serverске uloge



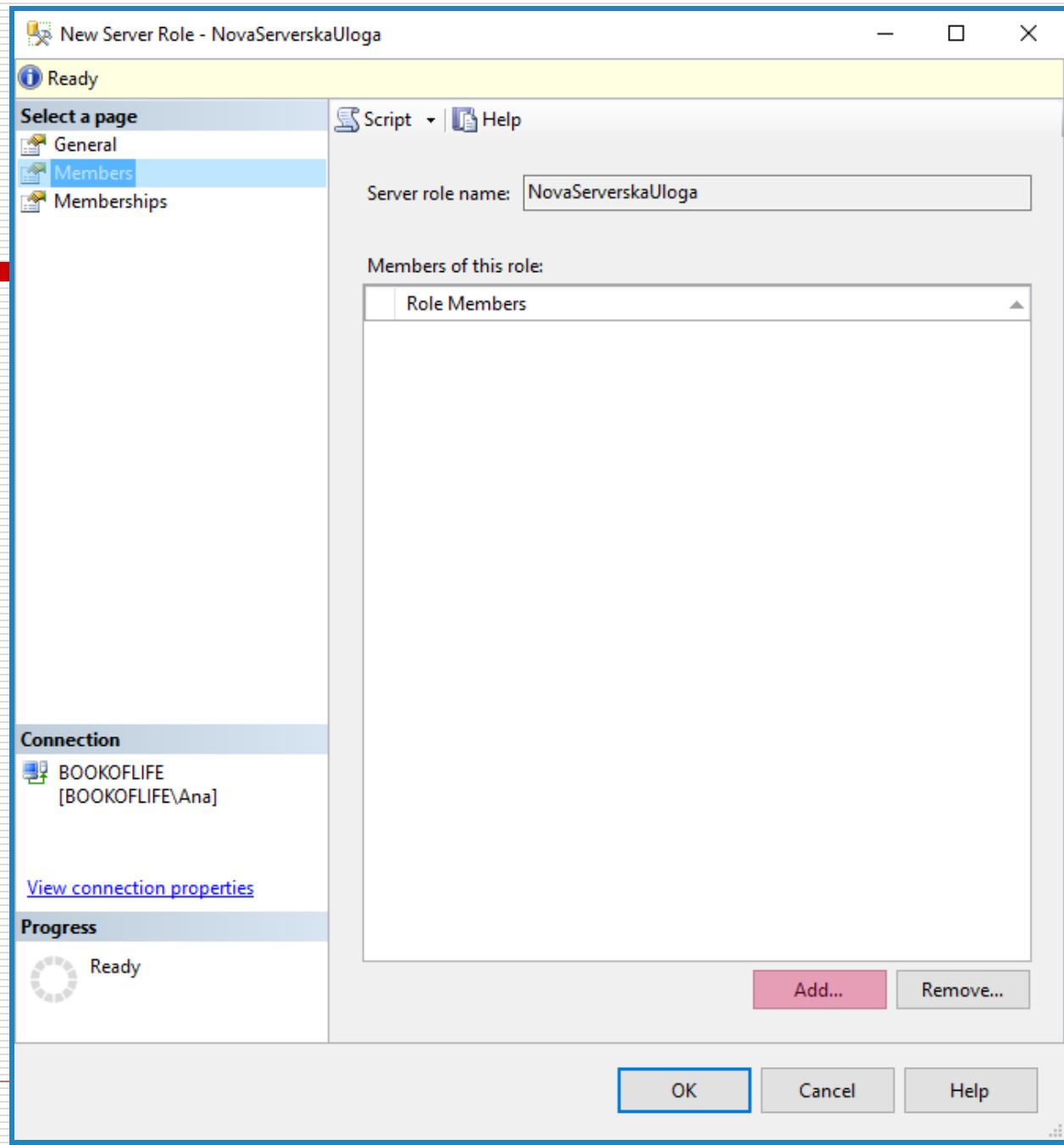
Slika 6: New Server Role

Sistemska sigurnost: kreiranje serverске uloge



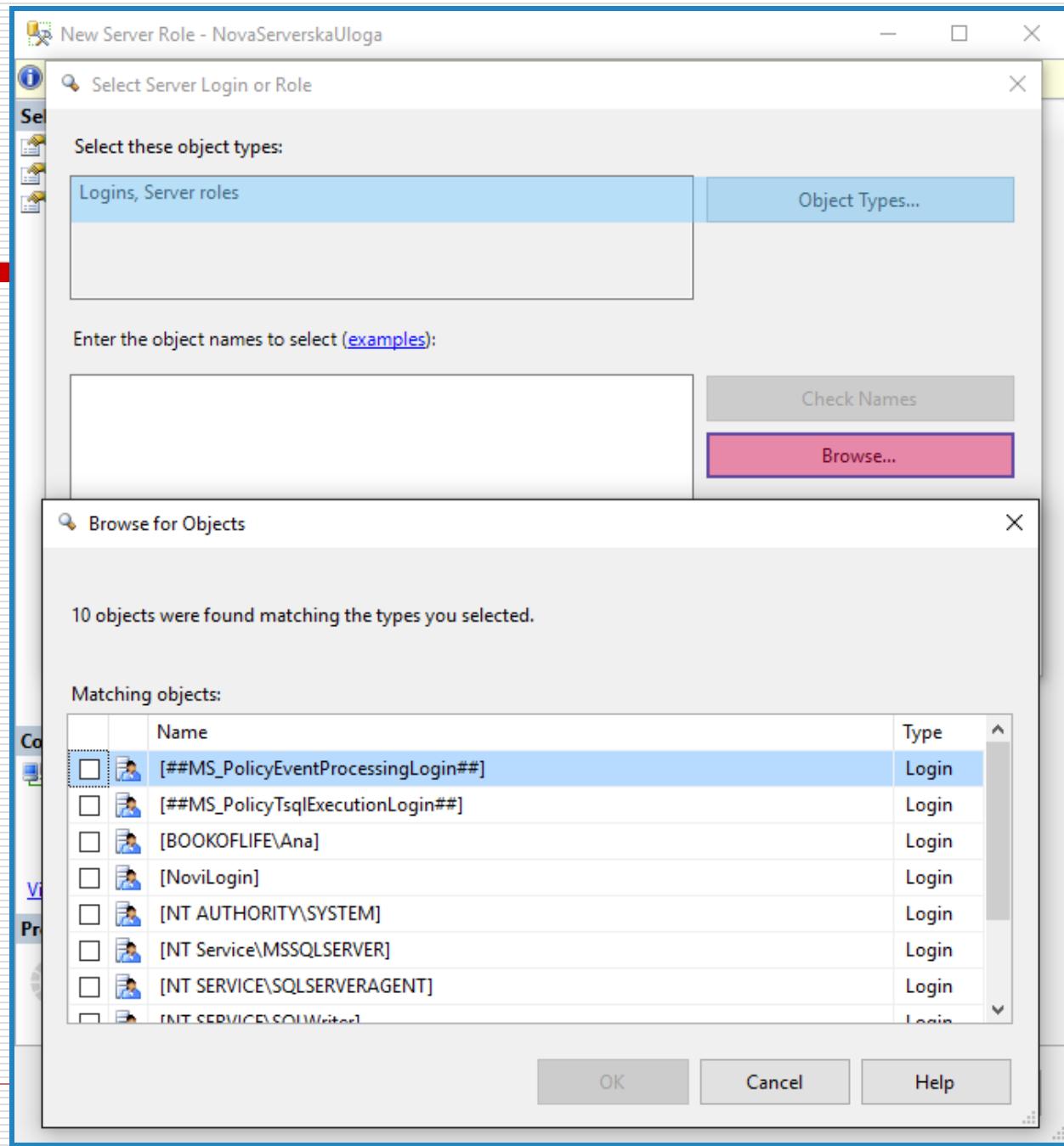
Slika 7: New Server Role

Sistemska sigurnost: kreiranje serverске uloge



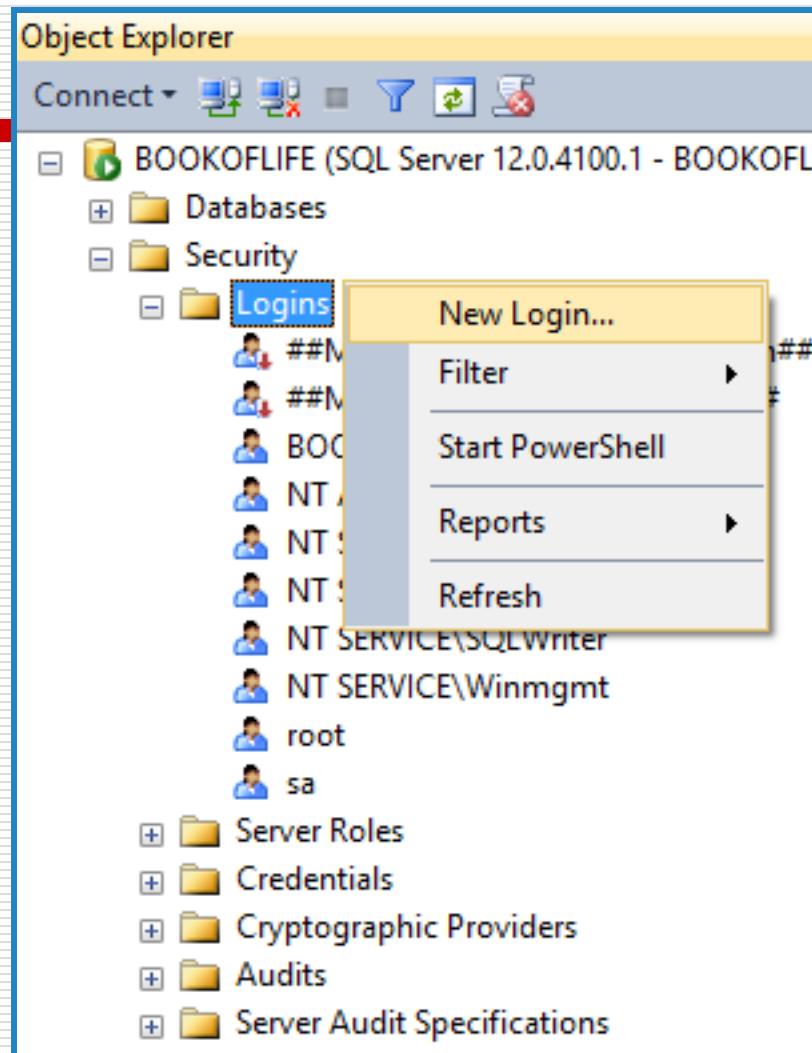
Slika 8: New Server Role

Sistemska sigurnost: kreiranje serverске uloge



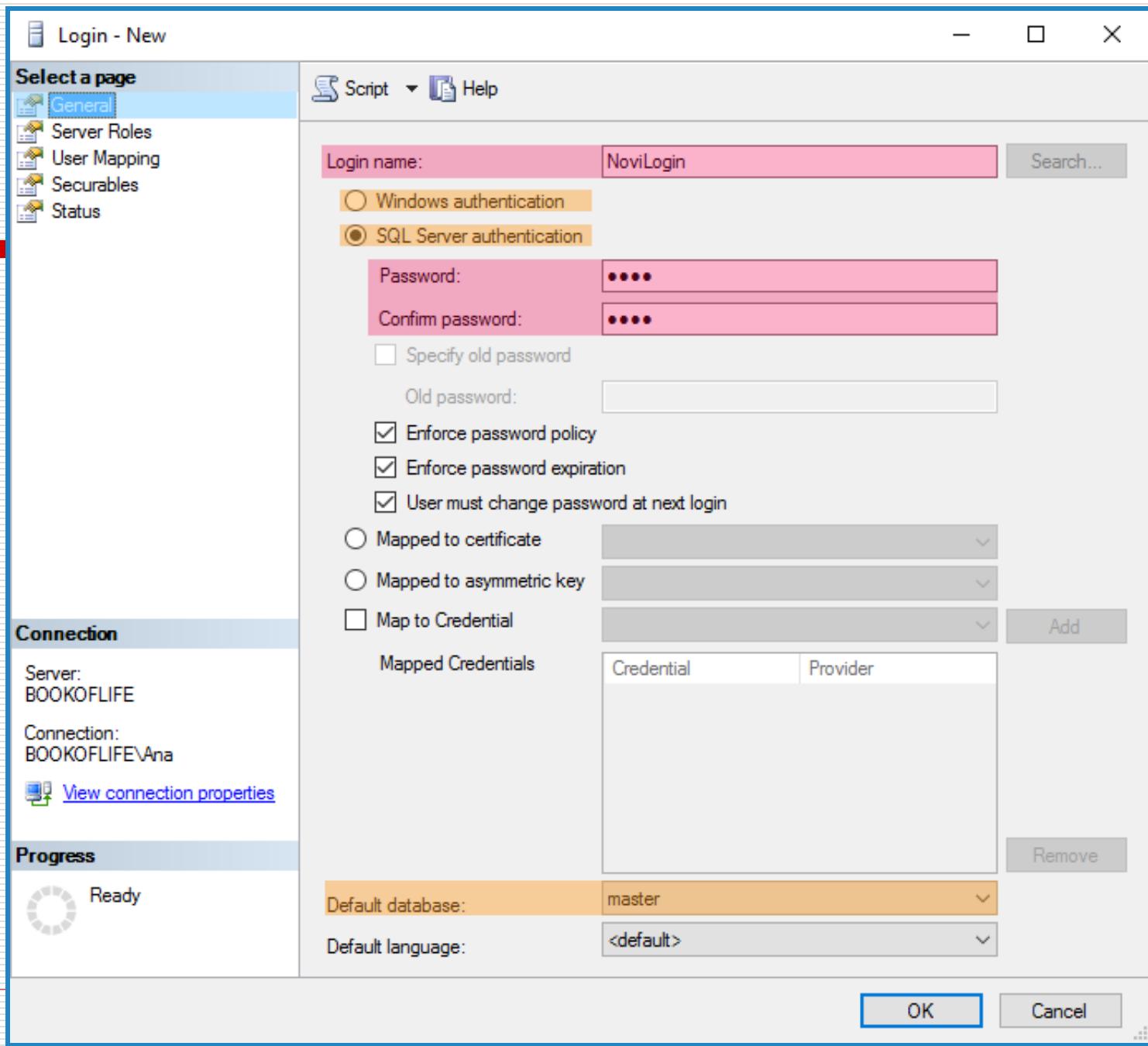
Slika 9: New Server Role

Sistemska sigurnost: kreiranje login-a

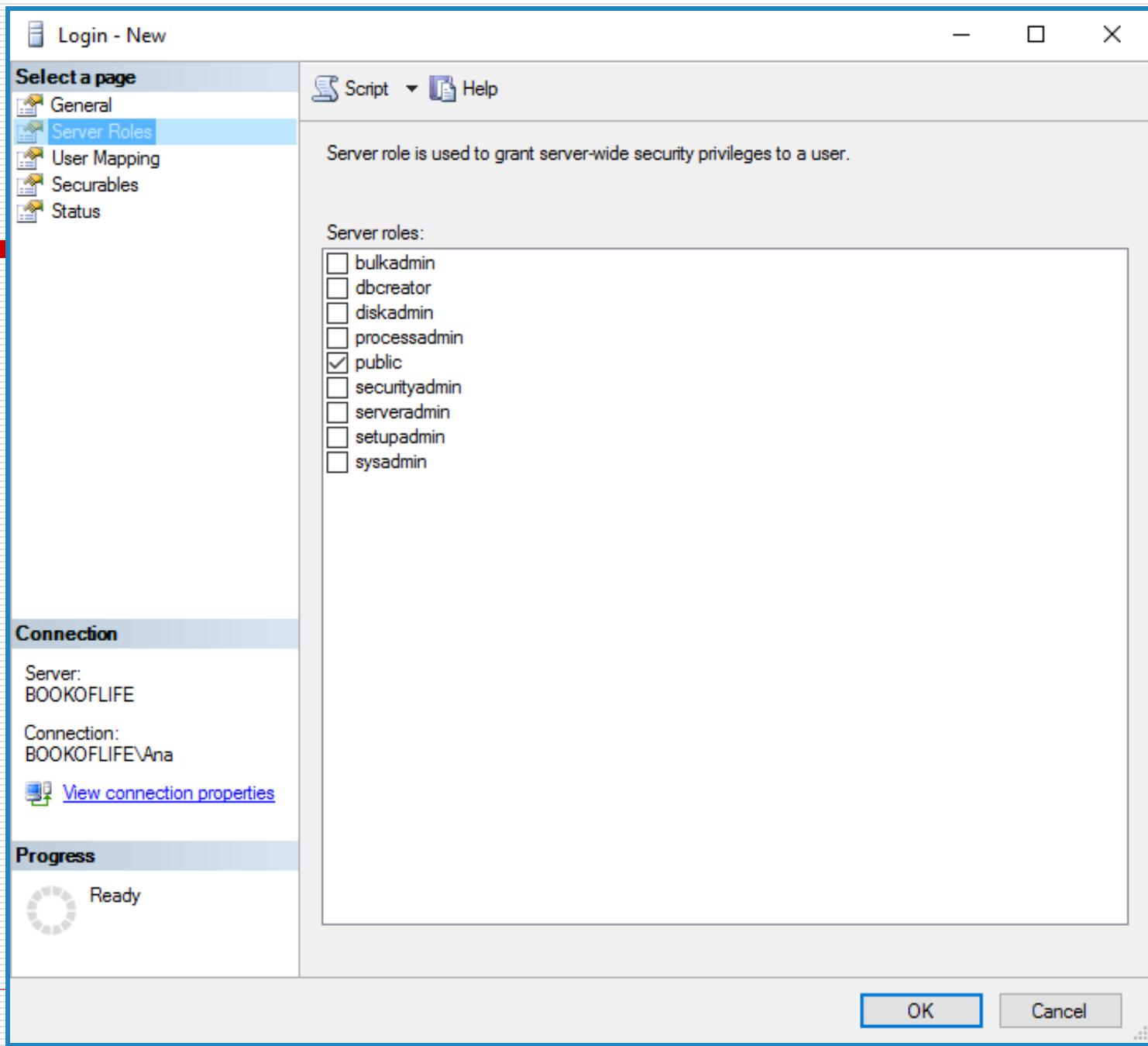


Slika 10: New Login

Sistemská sigurnost: kreiranje login-a

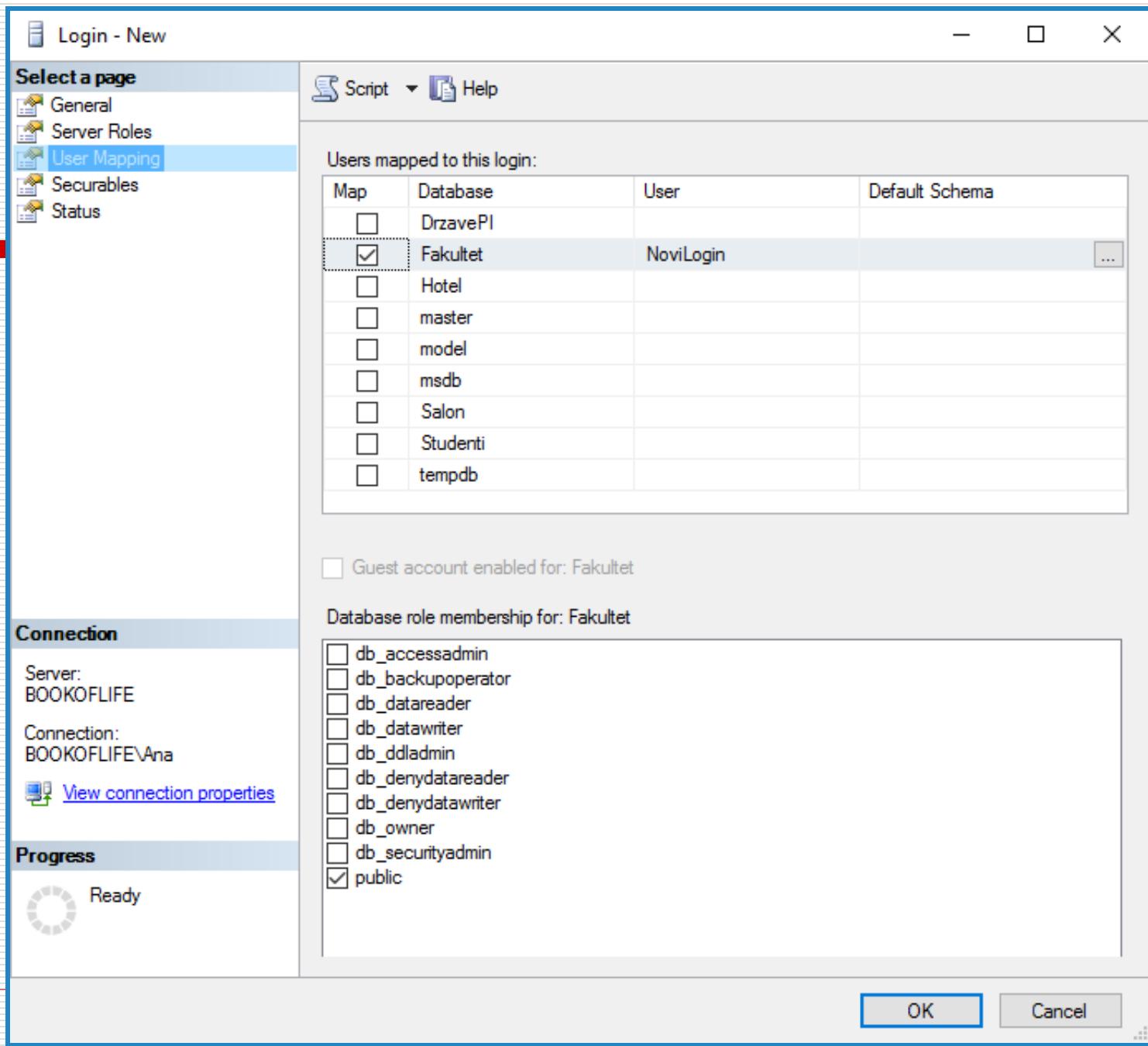


Sistemská sigurnost: kreiranje login-a



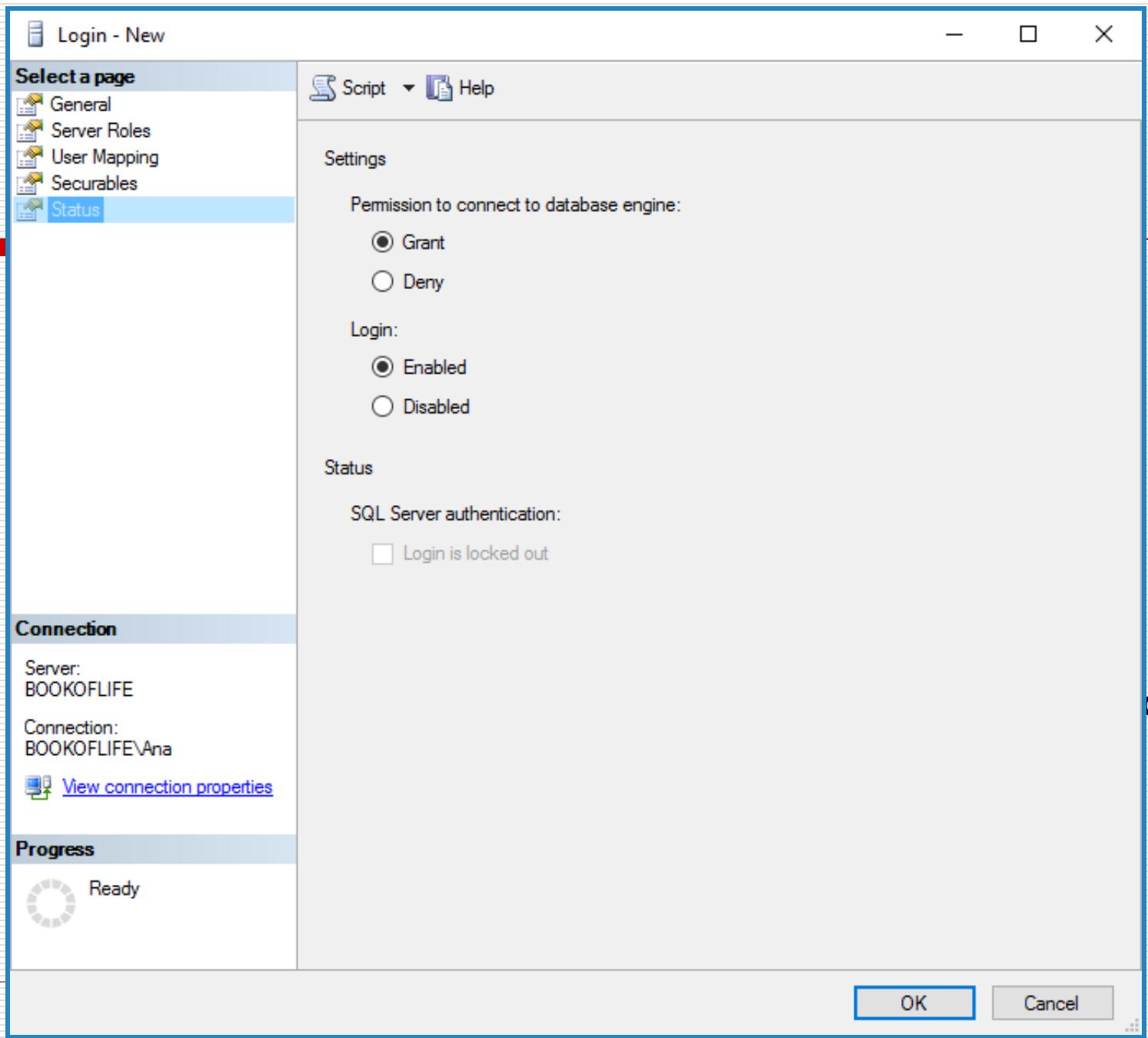
Slika 12: New Login

Sistemska sigurnost: kreiranje login-a

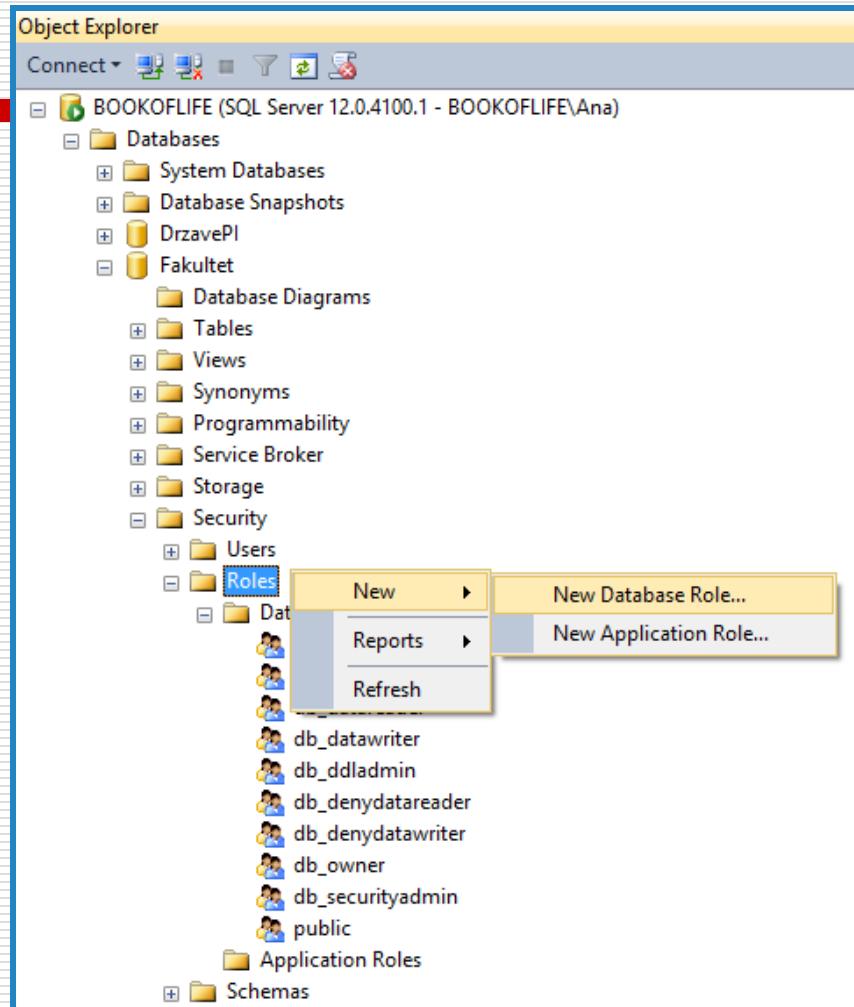


Slika 13: New Login

Sistemská sigurnost: kreiranje login-a

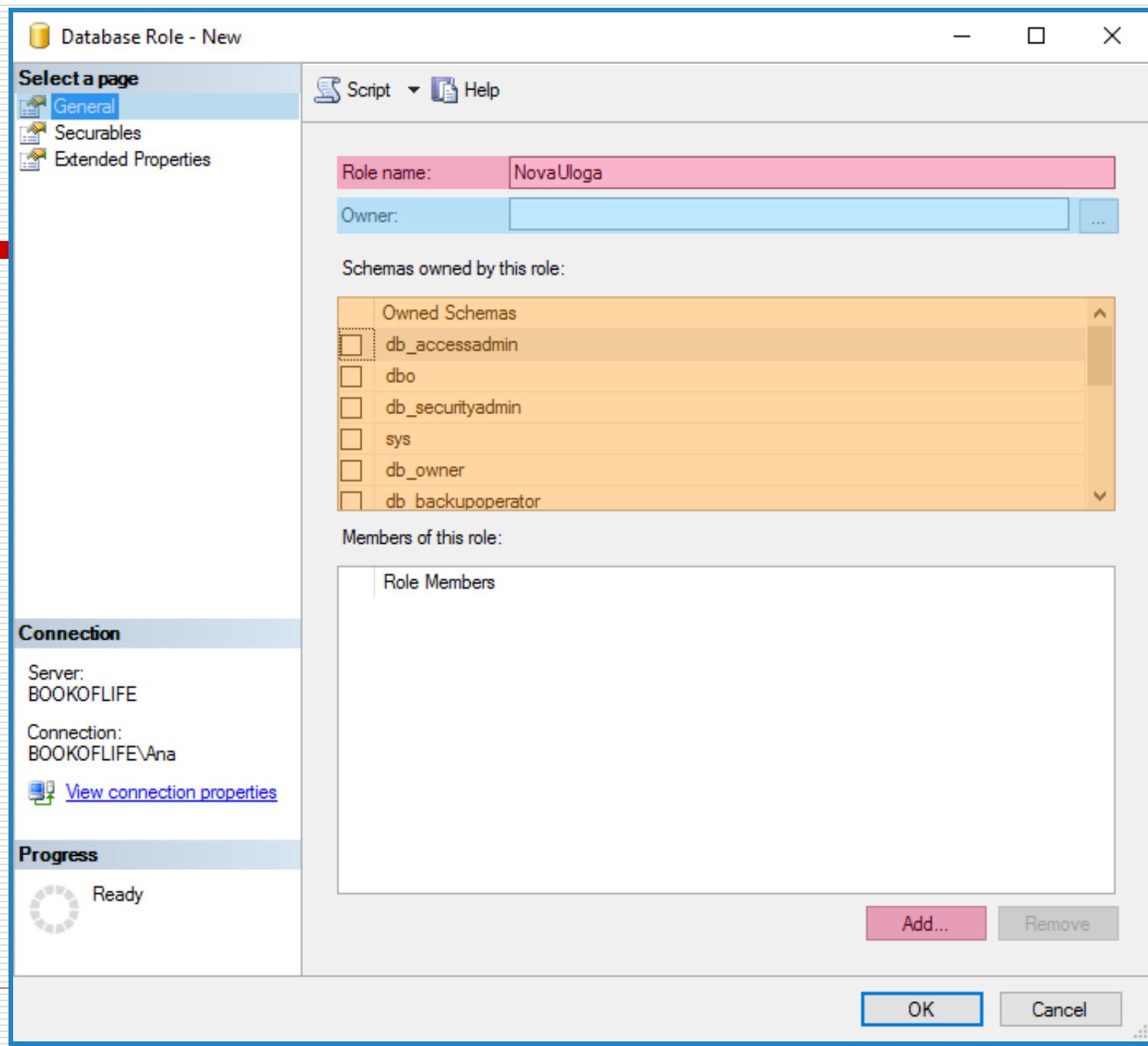


Objektna sigurnost: kreiranje uloge

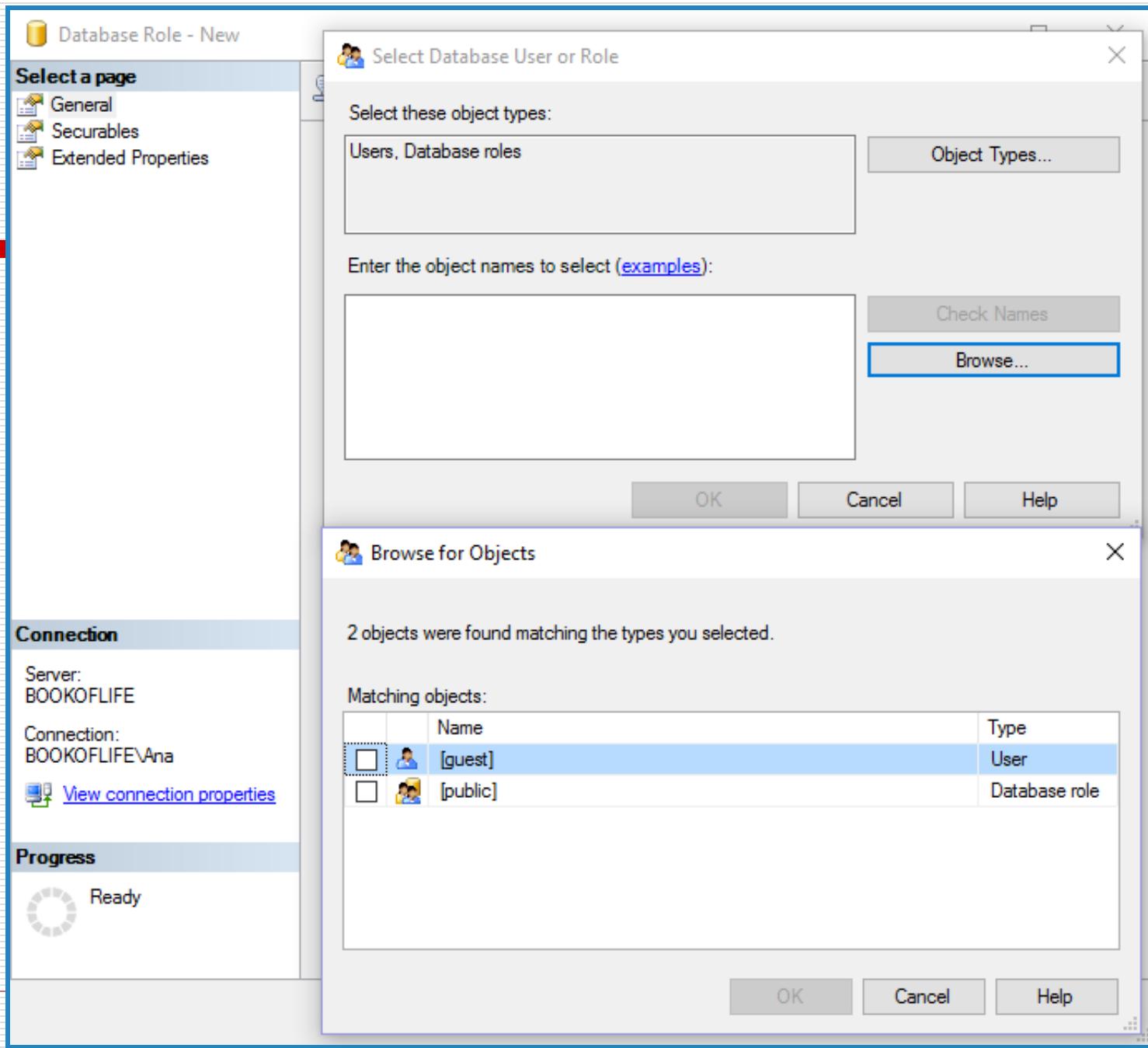


Slika 15: New Database Role

Objektna sigurnost: kreiranje uloge

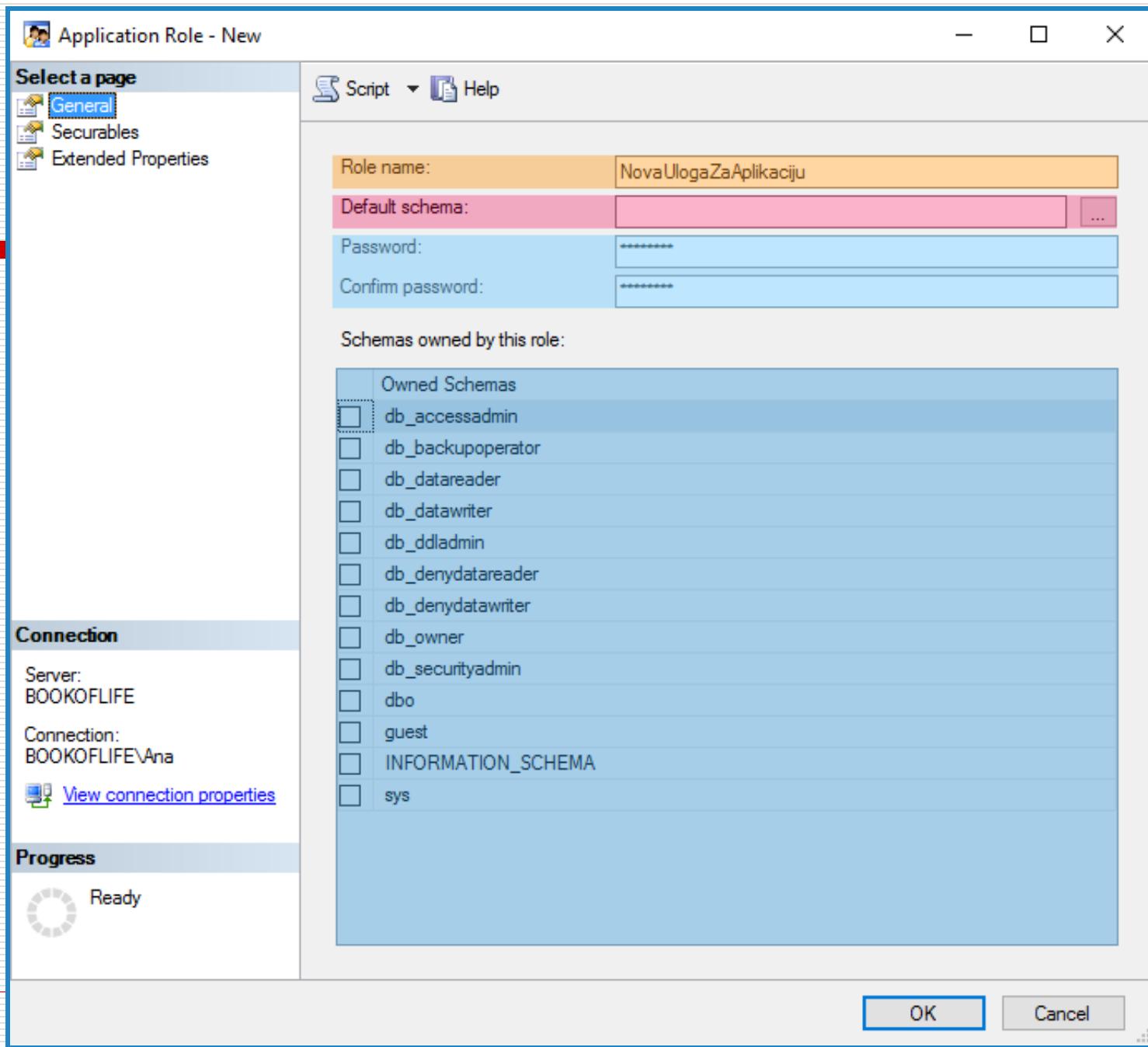


Objektna sigurnost: kreiranje uloge



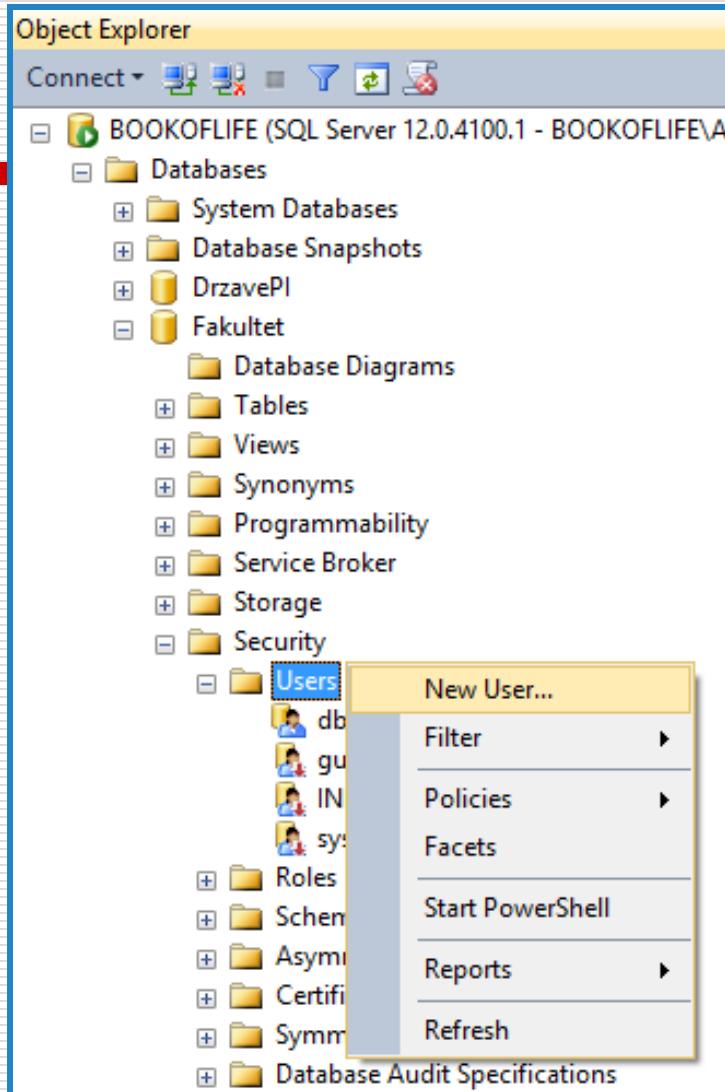
Slika 17: New Database Role

Objektua sigurnost: kreira je uloge



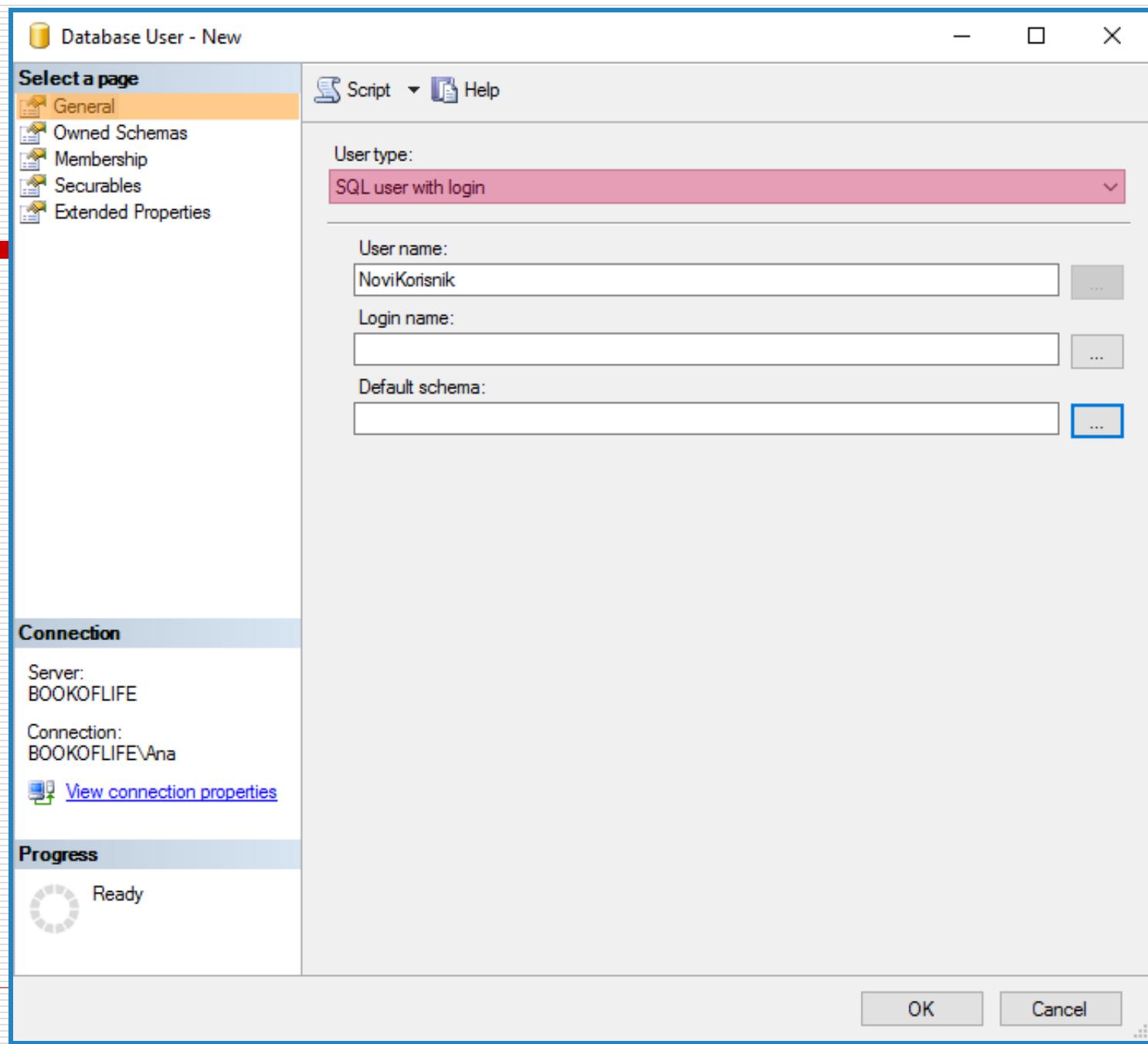
Slika 18: New Application Role

Objektna sigurnost: kreiranje korisnika

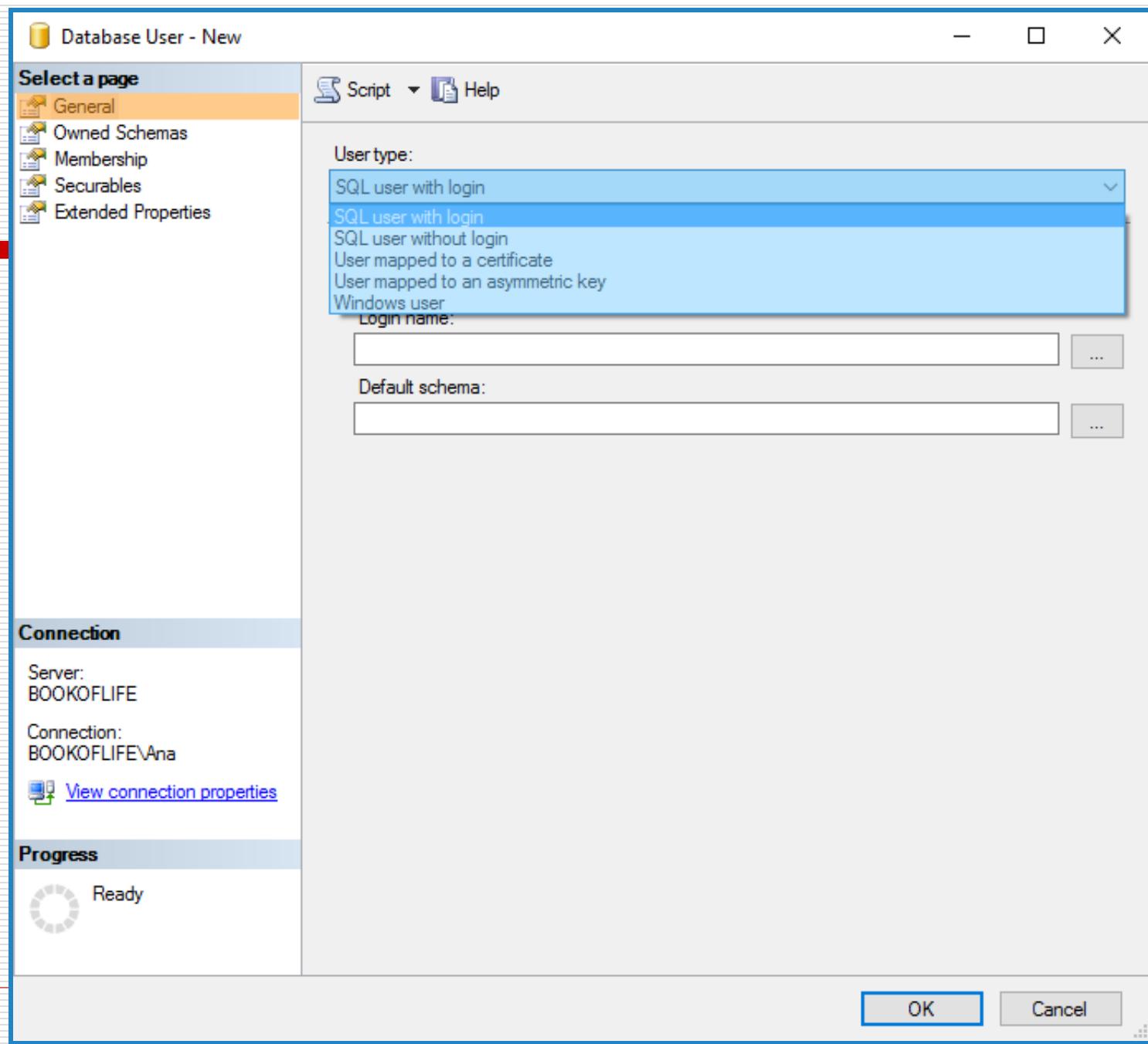


Slika 19: New User

Objektna sigurnost: kreiranje korisnika

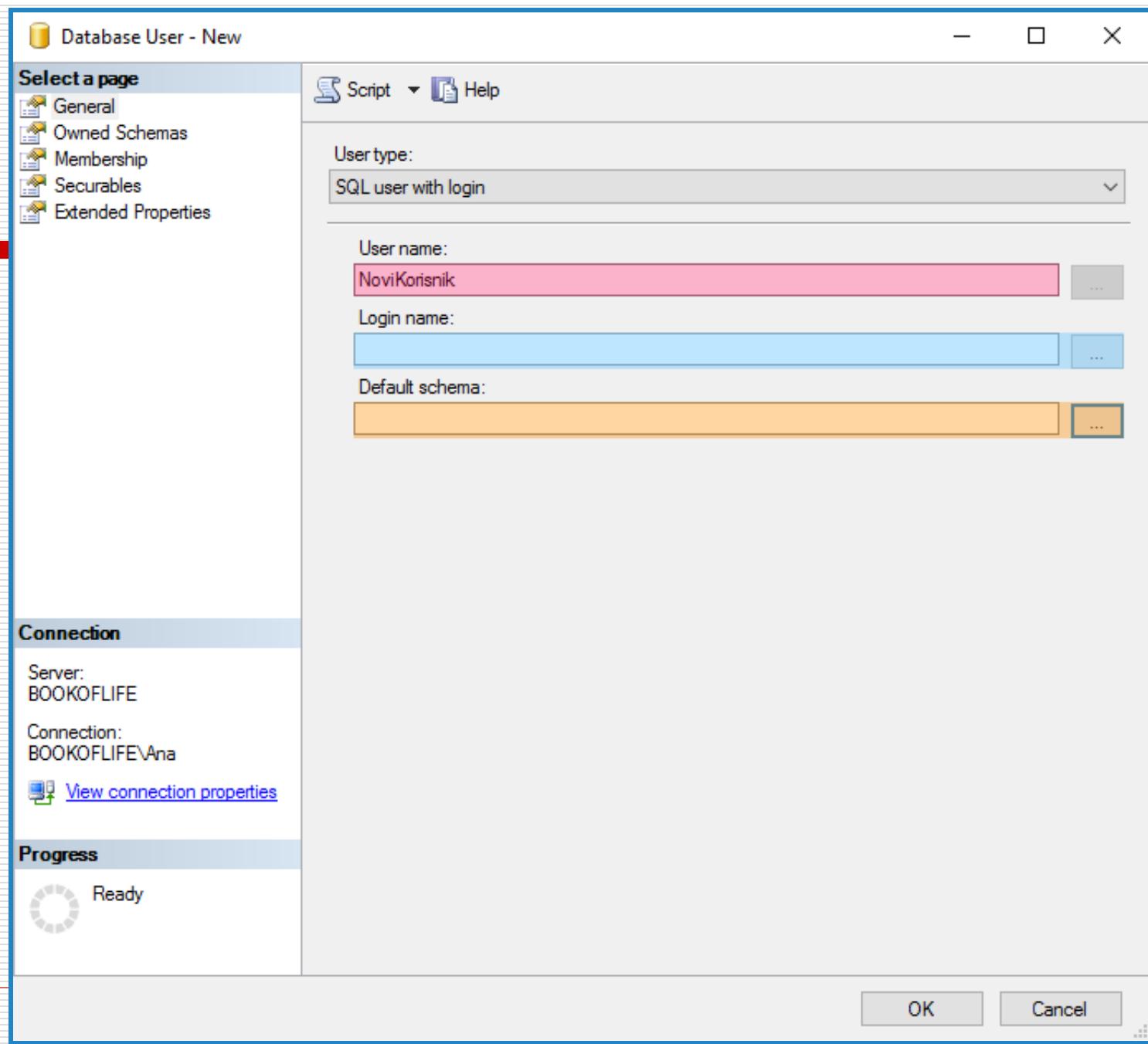


Objektna sigurnost: kreiranje korisnika

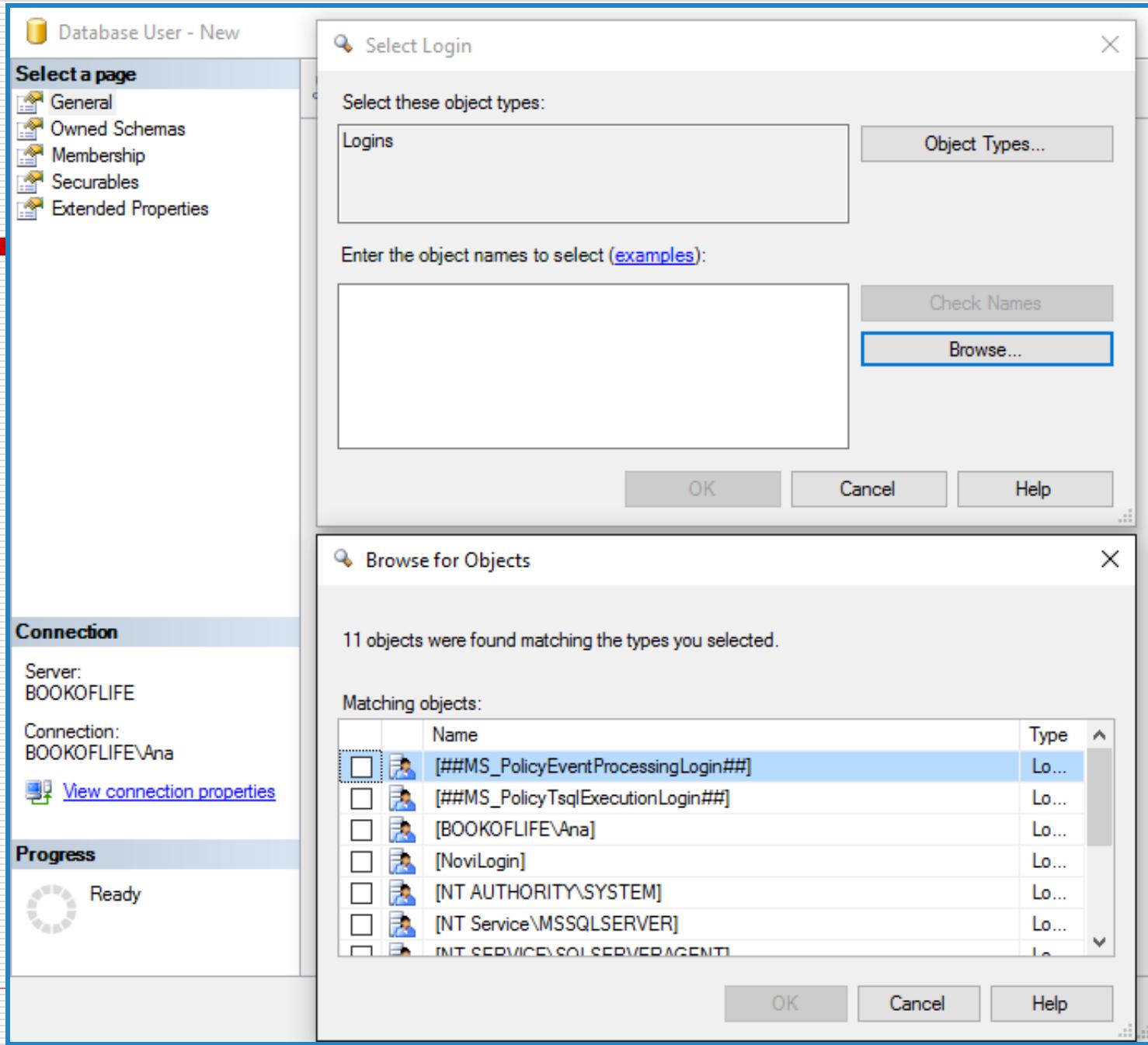


Slika 21: New User

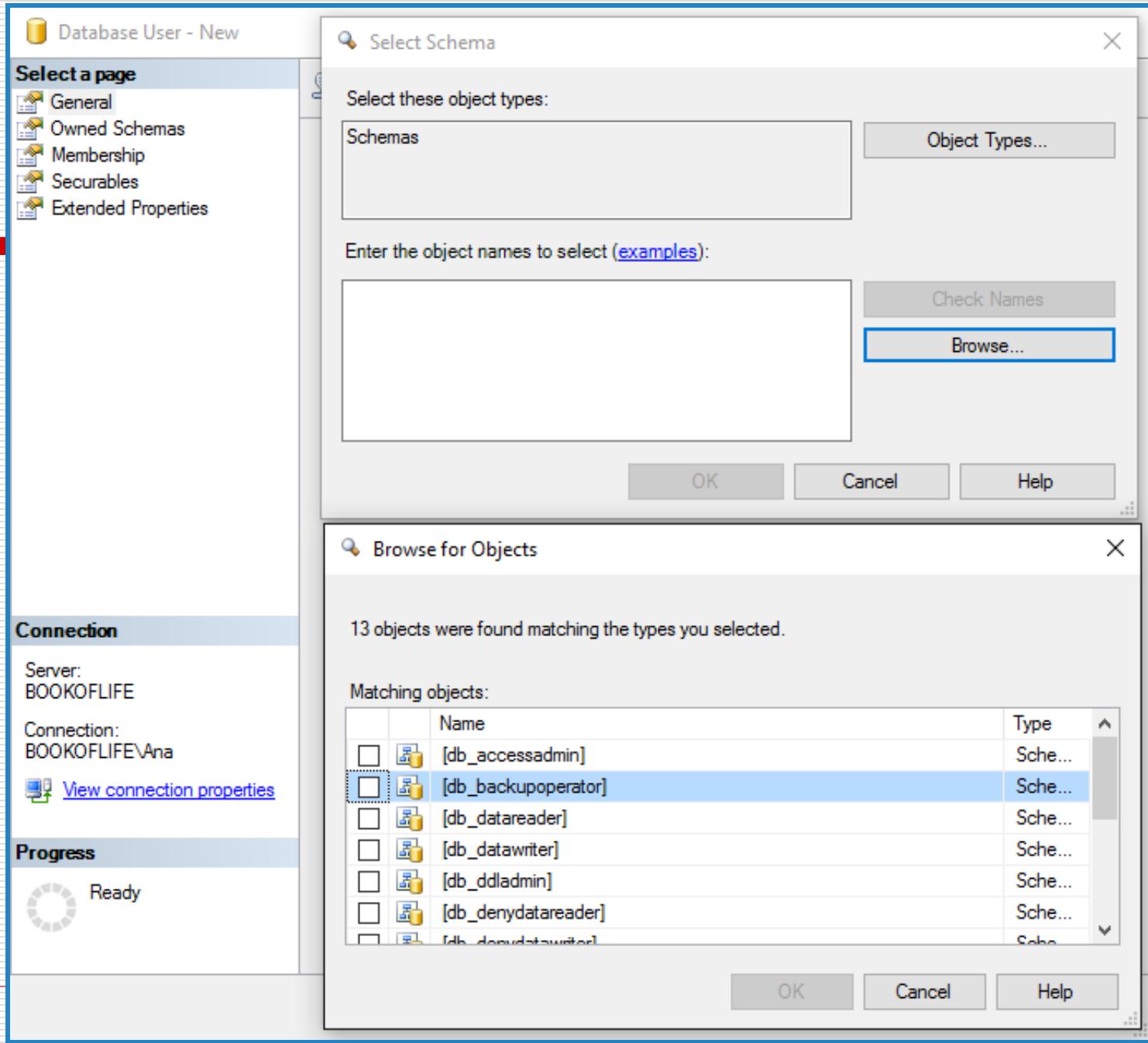
Objektna sigurnost: kreiranje korisnika



Objektna sigurnost: kreiranje korisnika

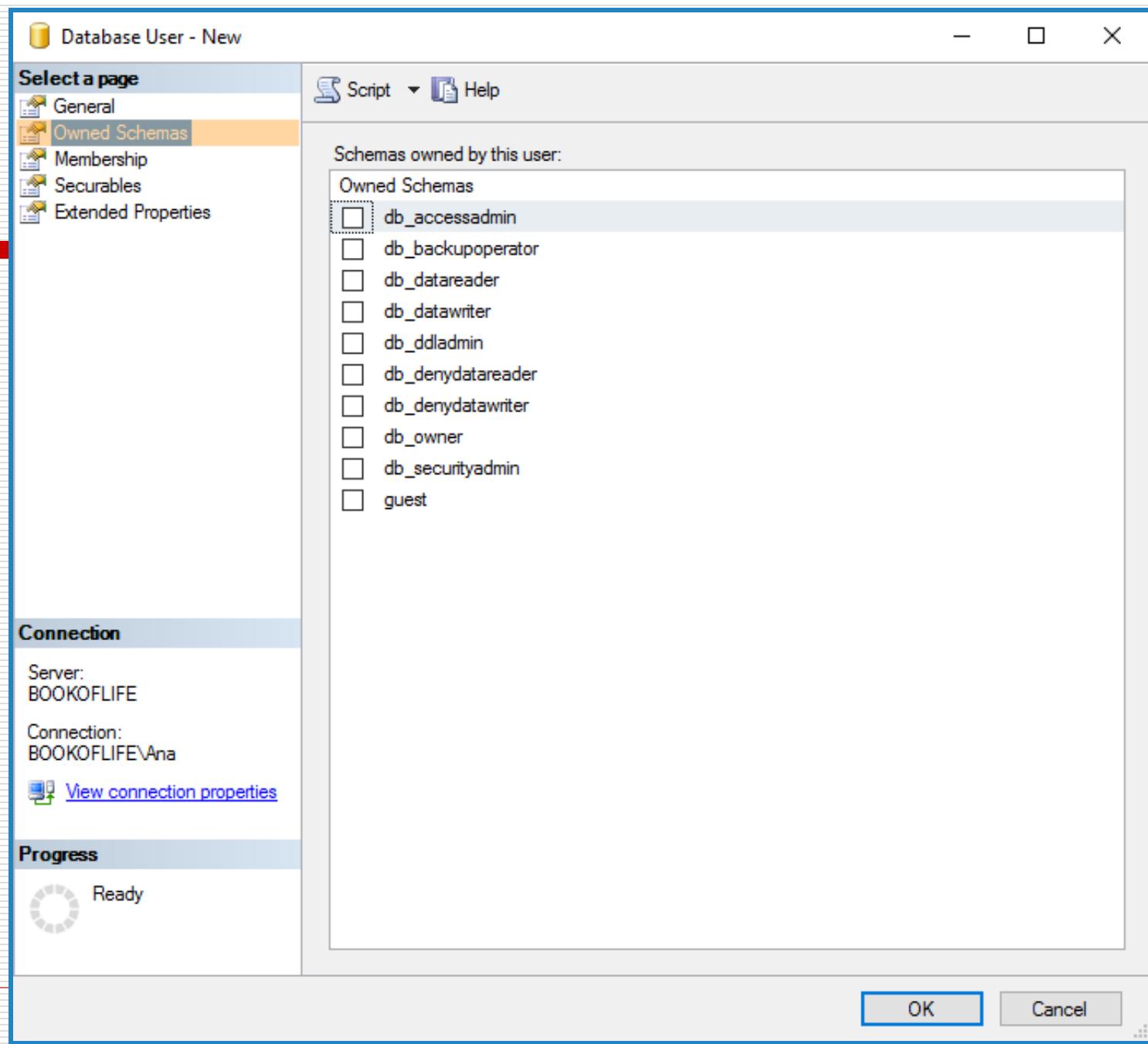


Objektna sigurnost: kreiranje korisnika

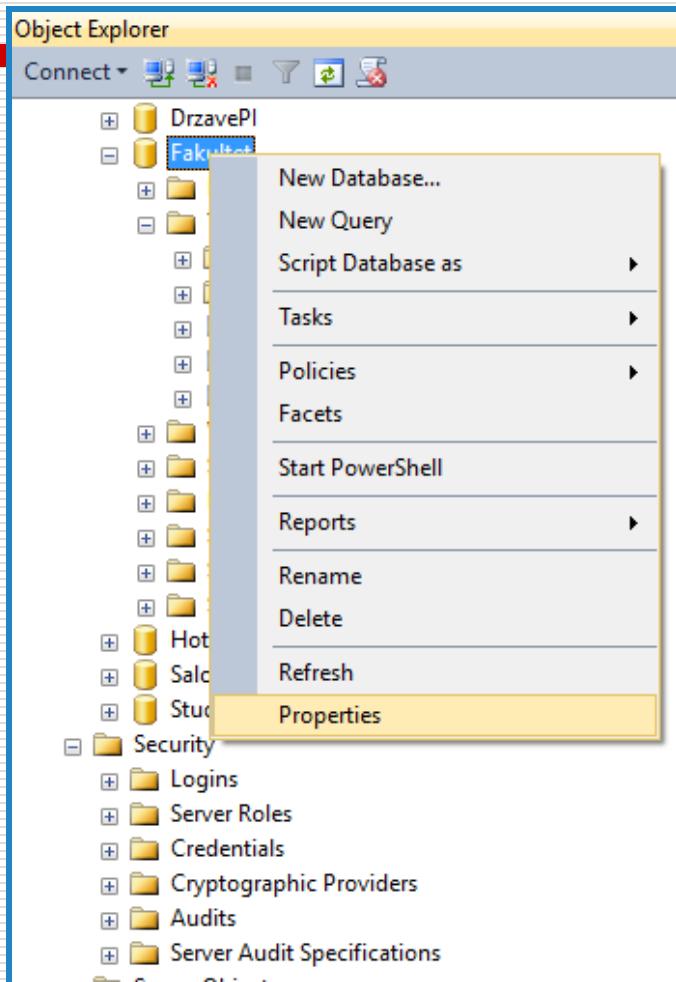


Slika 24: New User

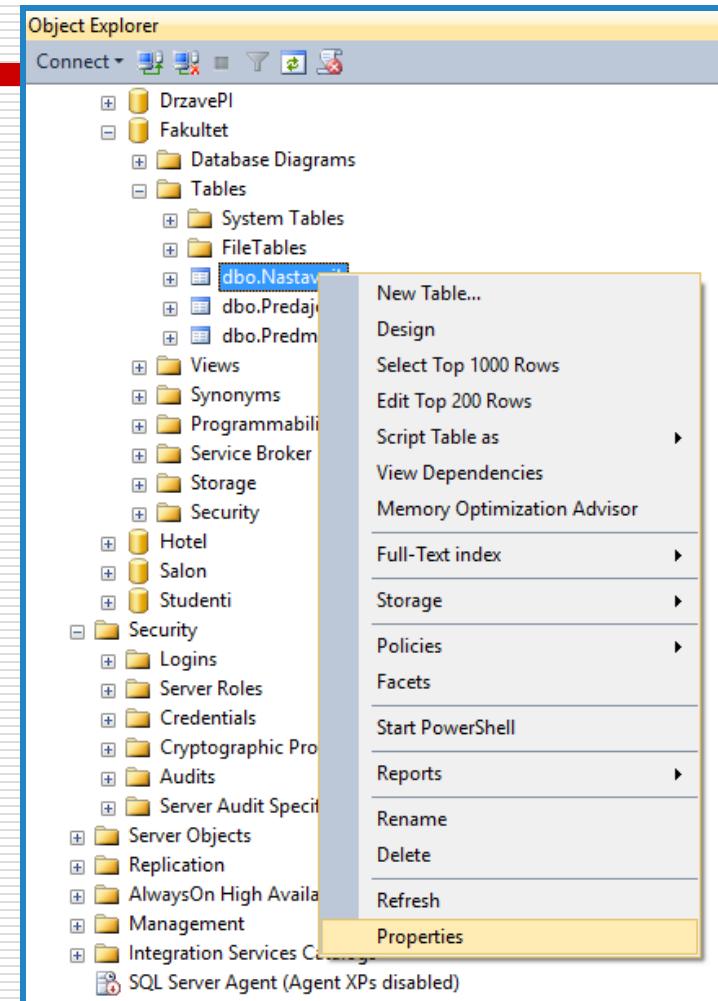
Objektna sigurnost: kreiranje korisnika



Objektna sigurnost: dozvole nad bazom ili tabelom



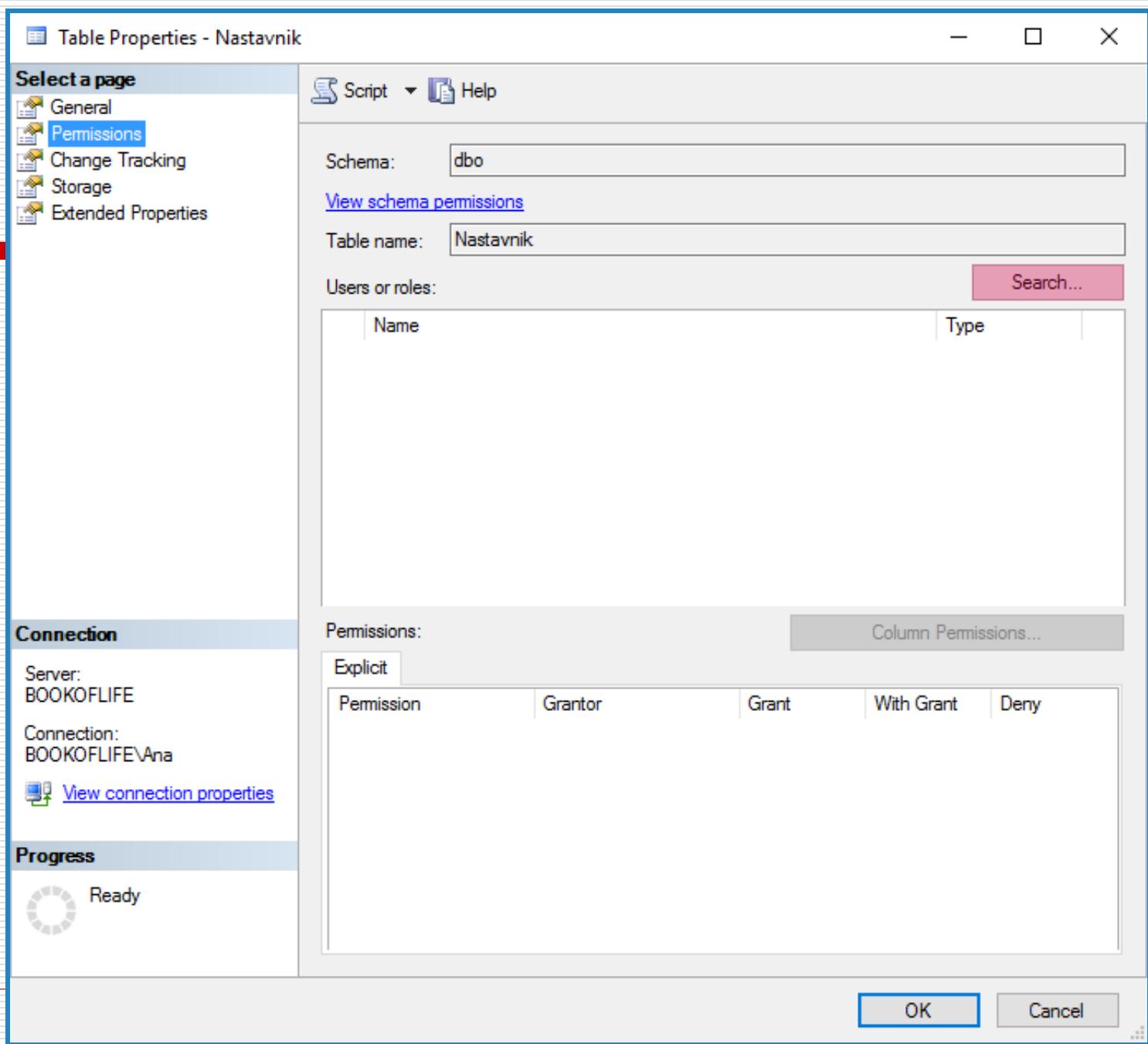
Slika 26: Premissions



Slika 27: Premissions

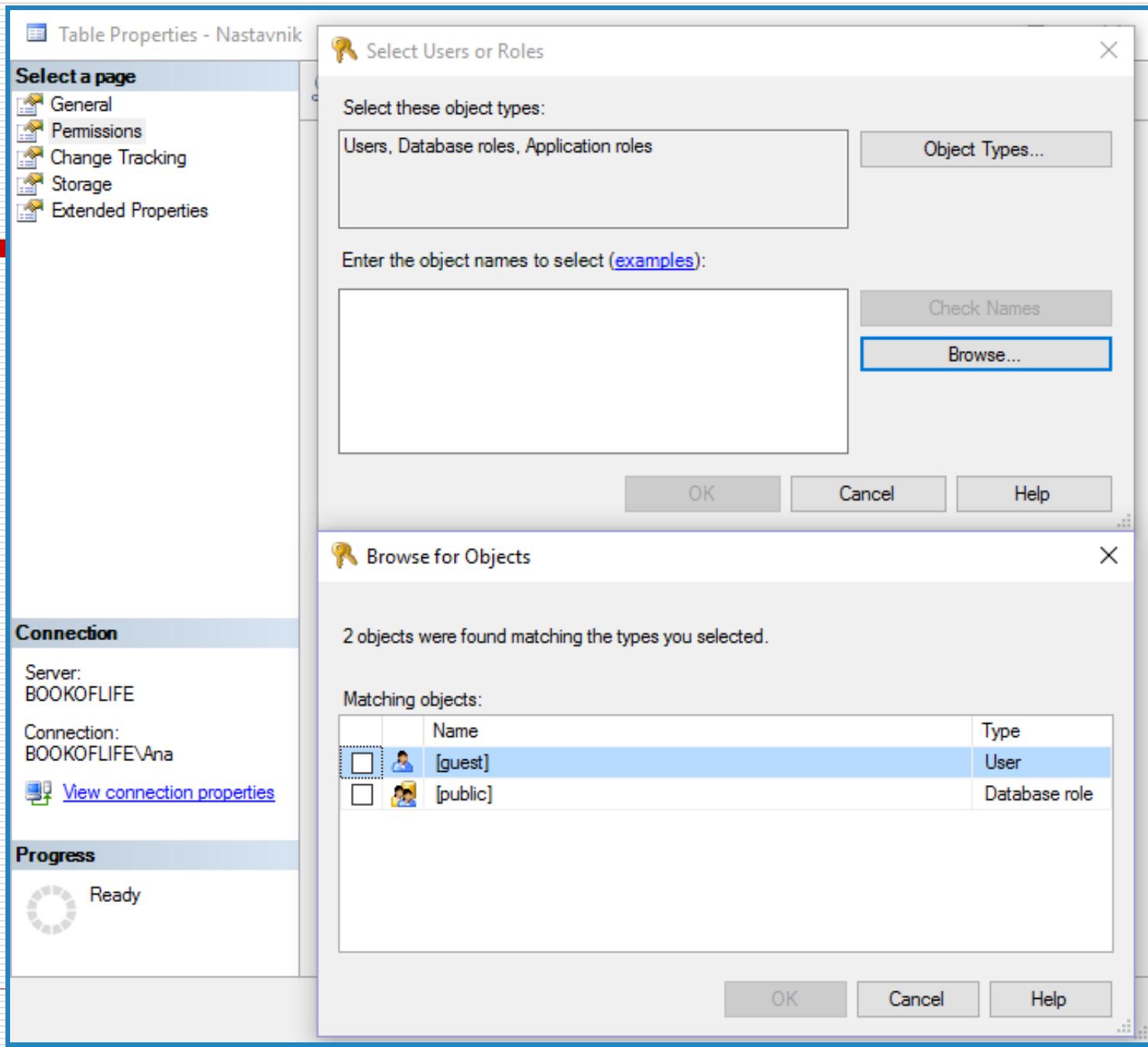
Slika 28: Permissions

Objekt na sigurnost: dozvole
nад базом или табелом



Slika 29: Permissions

Objekt na sigurnost: dozvole nad bazom ili tabelom



Slika 30: Permissions

Objekt na sigurnost: dozvole nad bazom ili tabelom

Table Properties - Nastavnik

Select a page: General, Permissions (selected), Change Tracking, Storage, Extended Properties

Script, Help

Schema: dbo

View schema permissions

Table name: Nastavnik

Users or roles:

Name	Type	...
guest	User	

Search...

Connection: Server: BOOKOFLIFE, Connection: BOOKOFLIFE\Ana, View connection properties

Progress: Ready

Permissions for guest:

Explicit	Effective	Column Permissions...			
Permission	Grantor	Grant	With Grant	Deny	
Alter		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Control		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delete		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Insert		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
References		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Select		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Take ownership		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

OK, Cancel

... (Redacted area)

Administracija baza podataka

Sigurnosne kopije i oporavak baze
podataka

Uvod – Sigurnosne kopije i oporavak BP

- ❑ Sigurnosna kopija (Backup) je kopiranje ili arhiviranje podataka kako bi se mogli oporaviti u slučaju gubitka podataka. Oporavak (Restore) je proces vraćanja podataka iz sigurnosne kopije.
 - ❑ U današnje digitalno doba, gubitak podataka može imati katastrofalne posledice - od gubitka ključnih poslovnih informacija do kršenja propisa o zaštiti podataka.
 - ❑ **Cilj predavanja:** Ovo predavanje ima za cilj da pruži osnovno razumijevanje o tome što su sigurnosne kopije i oporavak baza podataka, zašto su važni, različite metode i alate koji se koriste i najbolje prakse za implementaciju.
-

SIGURNOSNE KOPIJE BP

- ❑ Podaci predstavljaju jedan od najznačajnijih resursa u jednoj organizaciji;
- ❑ Postoji potreba za stalnim kreiranjem rezervnih kopija podataka da bi se sprečio njihov gubitak;
- ❑ Nije dovoljno samo posedovanje rezervnih kopija već je imperativ da one budu na drugoj lokaciji van opasnosti od uništenja.
- ❑ **Citat / Izjava:** "Ne postavlja se pitanje hoće li se gubitak podataka dogoditi, već kada će se dogoditi.
 - Sigurnosne kopije su osiguranje od gubitka podataka."

Osnovni pojmovi: Sigurnosna kopija i Oporavak

- **Sigurnosna kopija (Backup):** Sigurnosna kopija je proces formiranja kopija podataka koje se mogu koristiti za oporavak podataka usličaju njihovog gubitka, iz bilo kojih razloga.
 - **Oporavak (Restore):** Oporavak baze podataka je proces vraćanja podataka iz sigurnosne kopije. To se obično radi za oporavak podataka koji su izgubljeni, oštećeni ili izbrisani.
-

Zašto je rezervna kopija BP (Backup) važana?

- Prevencija gubitka podataka:** Bilo da se radi o tehničkim kvarovima, ljudskim greškama, krađi ili prirodnim katastrofama, formiranje rezervne kopije (backup-a) baze podataka je esencijalna mera zaštite.
 - Poslovni kontinuitet:** U slučaju katastrofe, pravovremeni backup može osigurati da poslovanje nastavi s minimalnim prekidima.
 - Regulatorni zahtevi i usklađenost:** Mnoge industrije zahtevaju redovne sigurnosne kopije kao deo usklađenosti s regulativama.
-

Zašto je oporavak BP (Restore) važan?

- Oporavak od gubitka podataka:** Restore je proces koji omogućava vraćanje podataka nakon gubitka, što obezbeđuje kontinuitet poslovanja.
 - Testiranje validnosti backupa:** Proces Restore-a takođe omogućuje testiranje validnosti backupa, što je ključno za obezbeđivanje njegove pouzdanosti.
 - Minimizacija vremena neposlovanja:** Brzi i efikasni procesi oporavka baze podataka mogu značajno smanjiti vreme neposlovanja prouzrokovano gubitkom podataka.
-

SIGURNOSNE KOPIJE BP

- Podaci predstavljaju jedan od najznačajnijih resursa u jednoj organizaciji;
 - Postoji potreba za stalnim kreiranjem rezervnih kopija podataka da bi se sprečio njihov gubitak;
 - Nije dovoljno samo posedovanje rezervnih kopija već je imperativ da one budu na drugoj lokaciji van opasnosti od uništenja.
-

POTREBA ZA REZERVNIM KOPIJAMA PODATAKA

- Rezervna kopija predstavlja glavnu komponentu plana kreiranja sigurnosnih kopija i oporavka baza podataka.
 - Ukoliko iz bilo kog razloga baza podataka postane nedostupna putem rezervne kopije se može izvršiti oporavak baze i tako je dovesti u stanje pre pada.
 - Jedan od najvažnijih zadataka administratora treba da bude redovno kreiranje kopija baze podataka.
 - To podrazumeva pravljenje konzistentnih kopija podataka obično u formi *image* fajlova.
-

Uzroci koji mogu dovesti do potrebe za oporavkom BP

Hardver

- Nekada je hardver bio glavni uzrok oporavka.
- Danas je hardver uglavnom pouzadan, ali opet može doći do kvara pojedinih komponenti.

Greške u samim aplikacijama

- U aplikacijama mogu postojati greške (tzv. *bug*).
- Mogu dovesti do neželjenih izmena nad podacima.

Padovi operativnog sistema

- Poput baze podataka operativni sistem takođe može da "padne".

Uzroci koji mogu dovesti do potrebe za oporavkom BP

Korisničke greške

- Pored grešaka u samim aplikacijama korisnici su najčešći uzrok potrebe za oporavkom.
- Nepažljivo ažuriranje podataka često dovodi do potrebe za oporavkom.

Sigurnosni propusti

- Danas sve više dobijaju na značenju.
 - Loše podešavanje *firewall-a*, o
 - Odsustvo bilo kakvog antivirusnog programa,
 - Neadekvatno dodeljene privilegija korisnicima
- Sve to može dovesti do neovlaštenog pristupa i izmene nad podacima.

Vanredne situacije

- Elementarne nepogode kao što su: poplave, požari i sl.

Vrste Sigurnosnih Kopija BP

- Potpune sigurnosne kopije,
 - Diferencijalne sigurnosne kopije,
 - Inkrementalne sigurnosne kopije,
 - Sigurnosne kopije Loga transakcija, itd.
-

Potpuna Sigurnosna Kopija (Full Backup)

- Potpuna sigurnosna kopija uključuje kopiranje svih podataka iz baze.
 - **Prednosti:** Oporavak baze podataka je jednostavan, jer sve potrebne informacije dolaze iz jedne sigurnosne kopije.
 - **Nedostaci:** Može zahtevati više vremena i prostora za formiranje rezervne kopije baze u poređenju s drugim vrstama sigurnosnih kopija (Backup-a).
-

Diferencijalna Sigurnosna Kopija (Differential Backup)

- Diferencijalna sigurnosna kopija uključuje kopiranje svih podataka koji su se promenili ili su dodati u bazu od formiranja poslednje potpune sigurnosne kopije baze podataka.
 - Prednosti:** Smanjuje vreme i prostor za formiranje rezervne kopije baze u poređenju s prostorom i vremenom formiranja ostalih vrsta rezervnih kopija baze podataka .
 - Nedostaci:** Oporavak baze može biti složeniji, jer zahteva poslednju potpunu sigurnosnu kopiju i poslednju diferencijalnu sigurnosnu kopiju baze podataka.
-

Inkrementalna Sigurnosna Kopija (Incremental Backup)

- ❑ Inkrementalna sigurnosna kopija uključuje kopiranje samo podataka koji su se promenili ili su bili dodati od trenutka uzimanja poslednje bilo kakve (potpune, diferencijalne ili prethodne inkrementalne) sigurnosne kopije baze podataka.
 - ❑ **Prednosti:** Zahteva najmanje vremena i prostora za formiranje rezervne kopije od svih tipova rezervnih kopija.
 - ❑ **Nedostaci:** Vraćanje podataka može biti najkomplikovanije, jer zahtijeva posljednju potpunu sigurnosnu kopiju i sve naknadne inkrementalne sigurnosne kopije.
-

Metode Sigurnosnih Kopija (Backup)

- Fizičke Sigurnosne Kopije**
 - Off-site Sigurnosne Kopije**
 - Cloud Sigurnosne Kopije**
-

Fizičke Sigurnosne Kopije

- ❑ Fizičke sigurnosne kopije podrazumijevaju smeštanje podataka na fizičke medijume, kao što su diskovi, NAS uređaji, magnetne trake ili optički diskovi (CD/DVD/Blu-ray).
- ❑ **Prednosti:** Brže performanse, pristup podacima čak i bez interneta, mogućnost skladištenja velikog volumena podataka.
- ❑ **Nedostaci:** Fizički medijumi mogu biti podložni oštećenju, krađi ili prirodnim katastrofama; potrebno je ručno upravljanje i organizacija.

Off-site Sigurnosne Kopije

- ❑ Off-site sigurnosne kopije podrazumijevaju smeštanje podataka na udaljenoj lokaciji, odvojenoj od primarnog poslovnog mesta.
 - ❑ **Prednosti:** Dodatna zaštita od lokalnih katastrofa, smanjenje rizika od gubitka podataka.
 - ❑ **Nedostaci:** Može biti sporije i skuplje od lokalnih kopija, potrebno je obezbediti sigurnost podataka na udaljenim lokacijama.
-

Cloud Sigurnosne Kopije

- ❑ Cloud sigurnosne kopije podrazumevaju smeštanje podataka na udaljenim serverima dostupnim preko interneta, često koristeći usluge poput Amazon S3, Google Cloud Storage ili Microsoft Azure.
- ❑ **Prednosti:** Jednostavno skaliranje, pristup podacima s bilo kojeg mesta preko interneta, automatsko upravljanje i ažuriranje.
- ❑ **Nedostaci:** Zavisnost od brzine interneta, potencijalni sigurnosni rizici, troškovi smeštanje podataka na osnovu potrošnje.

Oporavak baze podataka

- ❑ Oporavak podrazumeva vraćanje podataka iz sigurnosne kopije na izvornu ili novu lokaciju u cilju oporavka od gubitka podataka.
 - ❑ Oporavak je ključan deo strategije sigurnosnih kopija. Bez efikasnog i uspešnog procesa obnavljanja, sigurnosne kopije gube svoju svrhu.
 - ❑ Planiranje oporavka baze podataka podrazumeva razumevanje koje podatke treba obnoviti, u kojem redosledu, i koliko brzo.
-

Koraci Procesa Oporavka BP

- Izbor Sigurnosne Kopije:** Odabratи odgovarajućу sigurnosnu kopiju za oporavak na osnovu tipа gubitka podataka i vremenske tačke do koје se želi obnoviti BP.
 - Priprema za oporavak:** Proveriti da li je ciljna lokacija spremna za oporavak, uključujući dovoljan prostор за smeštanje i odgovarajuće dozvole.
 - Proces oporavka:** Inicirati proces oporavka korišćenjem alata za upravljanje sigurnosnim kopijama ili komandama baze podataka.
 - Provera:** Nakon oporavka, proveriti da li su podaci uspešno obnovljeni i da li sistem pravilno funkcioniše.
-

Najbolje prakse za Oporavka BP

- Testiranje Oporavka:** Redovno testirati proces obnavljanja kako bi se uverilo da su sigurnosne kopije ispravne i da se mogu uspešno obnoviti podaci.
 - Plan Oporavka:** Izraditi i održavati detaljan plan obnavljanja koji uključuje procedure i prioritete.
 - Edukacija:** Obezbediti da osoblje razume proces oporavka BP i da zna kako reagovati u slučaju gubitka podataka.
-

Najbolje Prakse: Regularnost i Planiranje

- ❑ **Regularnost Sigurnosnih Kopija:** Sigurnosne kopije treba praviti redovno, u skladu sa važnošću podataka i mogućnošću njihovog gubitka. Za kritične podatke, preporučuje se često pravljenje kopija.
 - ❑ **Planiranje:** Izrada plana za sigurnosne kopije i oporavak BP su ključni. Plan bi trebao sadržati: šta, kada i kako kopirati, kao i planove za oporavak i testiranje.
 - ❑ **Automatizacija:** Gde god je to moguće, automatizovati proces pravljenja sigurnosnih kopija kako bise smanjila mogućnost ljudske greške.
-

Najbolje Prakse: Testiranje i Verifikacija

- Testiranje Sigurnosnih Kopija:** Redovno testirajte sigurnosne kopije obnavljanjem podataka na testnoj lokaciji kako bise se uverilo da su kopije ispravne i upotrebljive.
 - Verifikacija Sigurnosnih Kopija:** Većina sistema za upravljanje bazama podataka nudi opciju za verifikaciju sigurnosnih kopija nakon što su napravljene. Ovo može pomoći u otkrivanju problema pre nego što dođe do potrebe za oporavkom BP.
 - Dokumentacija:** Zabeležiti sve relevantne informacije o sigurnosnim kopijama, uključujući vreme i datum, veličinu, lokaciju i eventualne greške koje su se pojavile tokom procesa.
-

Najbolje Prakse: Višestruke Kopije i Sigurnost

- ❑ **Višestruke Kopije:** Preporučuje se držanje više kopija podataka na različitim mestima i medijumima, uključujući i off-site i cloud kopije.
 - ❑ **Sigurnost:** Zaštititi sigurnosne kopije od neovlašćenog pristupa.
 - ❑ **Protokol za katastrofu:** Pripremiti plan za katastrofe koji uključuje korake za oporavak u slučaju ozbiljnih incidenata, kao što su prirodne katastrofe ili veliki sigurnosni incidenti.
-

Alati za Backup i Restore

- Alati za Backup i Restore su ključni za efikasno upravljanje sigurnosnim kopijama i oporavkom baza podataka.
- Postoji širok spektar alata, uključujući one koji dolaze sa SUBP-om, kao i nezavisne aplikacije.

Alat za Backup i Restore: MySQL Workbench

- MySQL Workbench je alat za Backup i Restore koji dolazi sa MySQL SUBP-om.
 - **Prednosti:** Integracija sa MySQL SUBP-om, podrška za potpune i inkrementalne kopije, jednostavan za upotrebu.
 - **Nedostaci:** Specifičan za MySQL, mogu biti ograničene mogućnosti za složenije scenarije.
-

Alat za Backup i Restore: MS SQL Server Backup

- MS SQL Server Backup je alat za Backup i Restore koji dolazi sa MS SQL Server SUBP.
 - **Prednosti:** Integracija sa MS SQL Server SUBP-om, podrška za potpune, diferencijalne i sigurnosne kopije Loga transakcija, mogućnost obnavljanja do tačke u vremenu (Point-in-Time Recovery).
 - **Nedostaci:** Specifičan za MS SQL Server, može biti kompleksan za nove korisnike.
-

Automatizacija Backupa u MS SQL Serveru

- **Značaj automatizacije:** Smanjuje mogućnost ljudske greške, obezbeđuje redovnost sigurnosnih kopija.
 - **Alati za automatizaciju:** Upoznavanje sa SQL Server Agentom i Maintenance Plans.
-

SQL Server Agent: Automatizacija Backupa

- **Šta je SQL Server Agent:** Servis za planiranje i automatizaciju zadataka u okviru MS SQL Server-a.
- **Kako koristiti SQL Server Agent:** Kreiranje Job-a za Backup, definisanje Step-a za izvršavanje T-SQL komande BACKUP DATABASE, postavljanje Schedule-a za Job.

Maintenance Plans: Automatizacija Backupa

- **Šta su Maintenance Plans:** Alat unutar SQL Server Management Studio-a (SSMS) za automatizaciju rutinskih zadataka.
- **Kako koristiti Maintenance Plans za Backup:** Kreiranje novog Maintenance Plan-a, dodavanje Backup Database Task-a, definisanje Schedule-a za Plan.

Najbolje Prakse za Automatizaciju Backupa

- Monitoring i upozorenja:** Konfiguracija SQL Server Agent Job-a ili Maintenance Plan-a da šalju obaveštenja u slučaju grešaka.
 - Testiranje:** Redovno testiranje procesa Backup-a kako bi se obezbedilo da automatizacija funkcioniše kako treba.
 - Dokumentacija:** Potrebno je detaljno dokumentovanje svih kreiranih Job-ova i Planova.
-

Backup naredba

Naredba za Backup baze podataka omogućuje:

- Backing up an entire database:
- Backing up specific files or filegroups:
- Backing up a transaction log-a:

Backup naredba - sintaksa

- Pogledati sintaksu u on-line dokumentaciji za MS SQL Server

Backup naredba - primer

Primar uzimanja sigurnosne kopije baze podataka Salon:

Backup DataBase Salon

To Disk = 'C:\BackupFail\BK141211.Bck'

Restore naredba

- Restauracija baze podatka na osnovu sigurnosne kopije omogućuje RESTORE naredba. Restore naredbom moguće je:
 - Restore an entire database:
 - Restore part of a database:
 - Restore specific files or filegroups:
 - Restore a transaction log-a:

Restore naredba - sintaksa

- Pogledati sintaksu u on-line dokumentaciji za MS SQL Server

Restore naredba - primer

- Primer restauracije baze podataka na osnovu sigurnosne kopije baze podataka

Restore Database Salon

From Disk = 'C:\BackupFail\BK141211.Bck'

Administracija baza podataka

Migracija baze podataka

Definicija migracije baze podataka

- Migracija baza podataka se odnosi na proces prenosa podataka iz jednog SUBP u drugi.
 - To je složen postupak koji zahteva pažljivo planiranje, implementaciju i testiranje kako bi se osigurala uspešna tranzicija.
 - Migracija može obuhvatiti prenos podataka, struktura baze podataka, logika i aplikacije koje koriste bazu podataka.
 - Cilj migracije je obezbediti kontinuitet poslovnih operacija i minimizirati prekid rada tokom tranzicije.
-

Razlozi za migraciju baze podataka

- Ažuriranje SUBP-a na noviju verziju
 - Prebacivanje sa jedne platforme na drugu (npr. sa Oracle na MS SQL Server)
 - Konsolidacija baza podataka radi smanjenja troškova i pojednostavljenja upravljanja.
 - Smanjenje troškova infrastrukture, održavanja i upravljanja podacima, pojednostavljenje administrativnih zadataka.
 - Spajanje ili deljenje baza podataka zbog promene poslovnih zahteva.
 - Prilagođavanje podataka novim poslovnim potrebama. Integracija podataka ili razdvajanje osetljivih informacija..
-

Migracija baze podataka

- Migracija podataka je složen proces, sastavljen od više aktivnosti, kojim se iz starog informacionog sistema podaci prenose u novi sistem.

 - U procesu migracije na podacima se vrše sve potrebne transformacije prema zahtevima novog sistema.
-

Razlozi za migraciju

- Razloga za zamenu postojećeg informacionog sistema novim, a samim tim i za migraciju podataka, ima više, medju kojima je najčešći modernizacija. Kompanije rastu, menjaju se i usvajaju nove biznis strategije.

 - Troškovi održavanja zastarelih informacionih sistema, kao i razvoj zastarelog aplikativnog softvera mogu da budu veoma značajna stavka u IT budžetu.
-

Rizici i teškoće

- Migracija šeme u novu bazom podataka je uvek rizik.
 - Potencijalni problem mogu doći iz mnogih izvora:
 - Oštećeni podaci koji su upisani od strane starog sistema i nisu popravljeni;
 - Implementirana zavisnost podataka za koju više нико не zna da postoji i šta znači;
 - Direktne izmene podataka bez korišćenja adekvatnih alata;
 - Greške u šemama i alatima za migraciju;
 - Greške u pretpostavkama kako bi se migracija trebala obavijati.
-

Rizici i teškoće

- Problem kvaliteta samih podataka;
 - Bez detaljnog razumevanja logike i poslovnih pravila koja se odnose na način skladištenja informacija u izvornom i ciljnom sistemu može doći do problema u nekoj od faza migracije.
 - Kompleksnost alata za Export, transformaciju i Import podataka.
 - Kako poslovni sistemi dodaju više funkcionalnosti i modula, migracija podataka postaje komplikovanija.
-

Metodologija

- Migracija podataka, shvaćena kao proces prenosa podataka iz jednog sistema u drugi, često noviji, **izgleda jednostavno.**
 - Ipak, i dalje zahteva dosta vremena da se pripremi kao i neko vreme da se završi sa pravilnom proverom migracije.
-

Koraci migracije BP

- Svaka od metodologija ukazuje da je potrebno razmotriti sledeće korake u migraciji podataka:
 1. Analiza poslovnog uticaja
 2. Prikupjanje informacija
 3. Mapiranje, dizajniranje
 4. Planiranje migracije
 5. Testiranje (plana) migracije
 6. Migracija
 7. Validacija
-

Tipovi migracije baze podataka

- ❑ Najjednostavnija situacija kod migracije baze podataka je da se baza podataka migrira sa jednog na drugi SUBP koji su iste verzije i istog proizvodjača.
 - ❑ Nešto komplikovanija migracija baze podataka je kada se migracija vrši izmedju SUBP-ova istog proizvodjača ali su verzije tih sistema različite.
 - Sto je veća vremenska razlika između prve pojave različitih verzija SUBP-ova moguće su i veće komplikacije pri migraciji baze podataka
 - ❑ Najkopljkovanija je migracija baze podataka s SUBP-a jednog proizvodjača na SUBP drugog proizvođača.
 - ❑ U svim navedenim tipovima migracije BP osim promene SUBP-a, sasvim je realna i promena strukture (i šeme) baze podataka.
-

Migracija na SUBP istog tipa

- Postoji mnogo softvera za prebacivanje podataka koje su ugrađeni u sisteme za upravljanje bazama podataka ili dolaze uz njih i razlikuju se od SUBP-a do SUBP-a.
 - Najjednostavniji način za prebacivanje podataka na drugu lokaciju istog SUBP-a, sastoji se u kreiranju **Backup-a** baze na lokalnom računaru, i izvršavanju akcije **Restore** baze na novoj lokaciji.
 - Ovo je situacija kada je šema baze podataka identična na obe lokacije.
 - Ako je potrebno migrirati podatke iz postojeće baze podataka u novu bazu (sa izmenjenom šemom), to nije moguće izvršiti korišćenjem **Backup-a** i **Restore-a**.
-

Migracija na SUBP istog tipa

- ❑ Ako je potrebno migrirati podatke iz postojeće baze podataka u novu bazu (sa izmenjenom šemom), to nije moguće izvršiti korišćenjem **Backup-a** i **Restore-a**.
 - U tom slučaju se mora vršiti export iz postojeće i Import podataka u novu bazu podataka.
 - Jednostavnija situacija je kada se mogu koristiti alati za Export i Import podataka koji se isporučuju kao deo samih SUBP-ova.
 - U nekim situacijama je potrebno pisati aplikacije (programe) za Export podataka i starog sistema u datoteke, pa Import podataka iz datoteka u bazu podataka novog sistema.
-

Migracija sa jednog SUBP na drugi SUBP

- Baza izgrađena sa jednim SUBP nije prenosiva na drugu SUBP (tj, drugi SUBP ne mogu je pokrenuti).
 - Postoje alati koji mogu da pomognu migraciju između specifičnih SUBP-a. Tipično proizvođač SUBP-a daje alate da pomognu Import baza podataka iz drugog popularnog SUBP-a.
 - Moderne aplikacije se uglavnom prave tako da migracija sa jednog SUBP na drugi zahteva samo ciklus testiranja da bi se uverilo da nije bilo negativnih uticaja na funkcionalne i ne-funkcionalne performanse.
 - Ovde, takođe, kao krajnje rešenje ostaje mogućnost pisanja aplikacije (programe) za Export podataka i starog sistema u datoteke, pa Import podataka iz datoteka u bazu podataka novog sistema.
-

Alati za migraciju BP

IMPORT/EXPORT

- Pojedinačnih tabela
 - U različitim formatima
 - Tekstualne datoteke
 - .XLS
 - Direktno preuzimanje tabela s drugog servera
- Dela baze podataka

Copy Baze podataka

Administracija baza podataka

Upravljanje performansom

UPRAVLJANJE PERFORMANCEM

- ❑ Štaje performansa Baze podataka?
 - Brzina kojom SUBP obezbeđuje informacije korisnicima.

 - ❑ Faktori koji utiču na performansu baze podataka:
 - Obim posla
 - Protok
 - Resursi
 - Optimizacija
 - Konkurentnost
-

PODEŠAVANJE PERFORMANCE

Obim posla je kombinacija:

- On-Line transakcija
- Batch poslova
- Ad hock upita
- Data Warehouse analiza i
- Sistemskih poslova

Koji se izvršavaju u datom trenutku.

- Menja se tokom vremena
 - Nekad je predvidiva, a nekad ne
-

PODEŠAVANJE PERFORMANCE

- Protok određuje ukupnu mogućnost računara da obradi podatke. Obuhvata:
 - Brzinu Ulaza/Izlaza
 - Brzinu procesora
 - Mogućnost paralelizma
 - Efikasnost Operativnog sistema i
 - Sistemskog softvera
-

PODEŠAVANJE PERFORMANCE

- Resursi** su hardverski i softverski alati koji su na raspolaganju sistemu. Neki od resursa su:
 - Kernel BP
 - Diskovi
 - RAM
 - Kontroleri

PODEŠAVANJE PERFORMANSE

- Optimizacija upita se kod relacione baze podataka obavlja u okviru SUBP-a.

 - Potrebno je optimizovati i druge faktore da bi optimizator Baze podataka kreirao najefikasnije pristupne putanje.
-

PODEŠAVANJE PERFORMANCE

- **Konkurenčija** nad resursima se javlja kad je potražnja za resursima velika.
 - Situacija kada dva ili više elemenata datog obima posla pokušavaju da koriste isti resurs u istom trenutku na konfliktan nacin
 - Sa porastom konkurenčnosti smanjuje se protok
-

PODEŠAVANJE PERFORMANCE

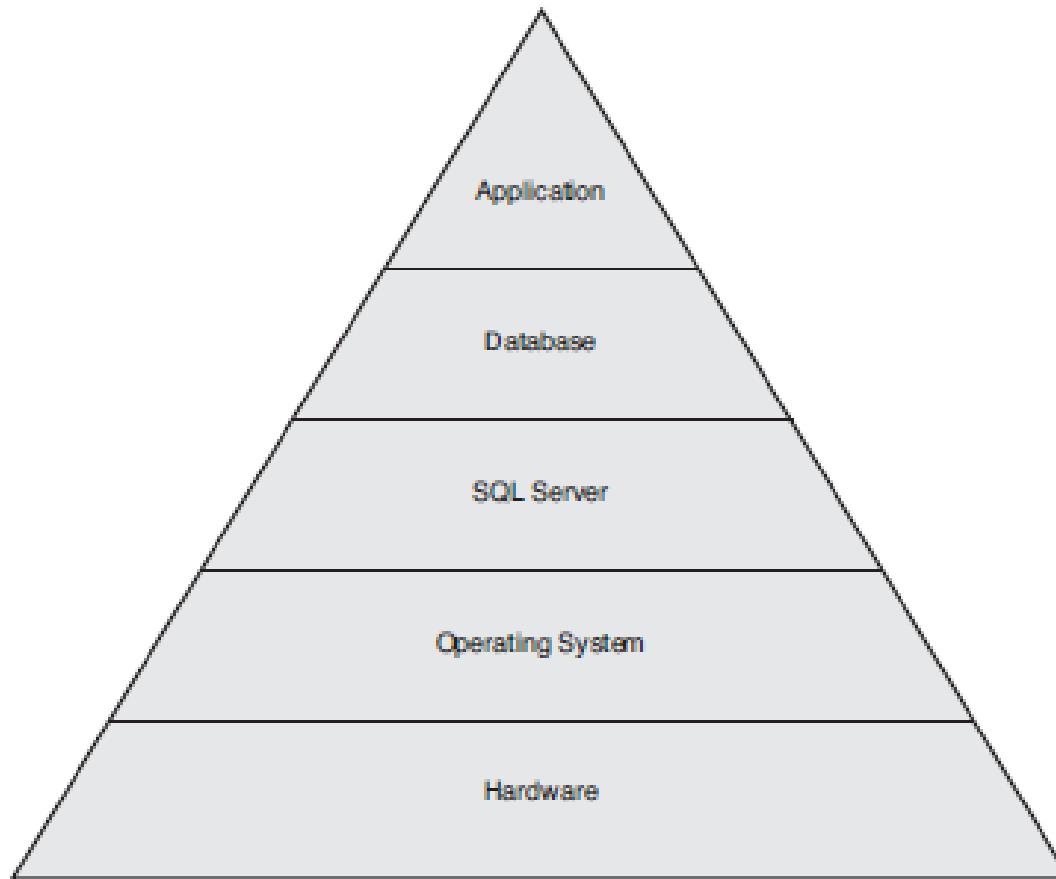
- Na osnovu prethodnih pet faktora performansi BP možemo definisati kao:
 - Optimizaciju upotrebe resursa kako bi se:
 - Povećao protok i
 - Minimizirala konkurencija
 - I time obezbedila obrada maksimalno mogućeg obima posla.
-

PODEŠAVANJE PERFORMANCE

Na performanse nekog aplikativnog sistema utiče više faktora u ‘lancu’:

- Hardver
 - Operativni sistem
 - SUBP
 - Implementacija Baze podataka
 - Aplikacija
-

PODEŠAVANJE PERFORMANCE



Plan upravljanja performansom

- Administrator Baze podataka (ABP) mora sastaviti osnovni plan kojim će osigurati:
 - Upravljanje i
 - Analizu performanse za sve aplikacije BP
 - Plan upravljanja BP uključuje:
 - Alate za praćenje performanse aplikacija i
 - Fino podešavanje BP i SQL koda
 - Pravilo 80/20
 - Prvi korak identifikacija problematičnih područja
-

Plan upravljanja performansom

- ❑ Najčešći krivac za većinu problema sa performansama aplikacija BP je:
 - Neefikasan SQL kod i
 - Programski kod aplikacija
 - ❑ 75% do 80% svih problema performanse BP
 - ❑ Do pogoršanja performanse može doći tokom vremena
-

Plan upravljanja performansom

Neki od razloga degradacije performansi:

- Rast Baze podataka
 - Nove pristupne putanje podacima
 - Dodatni korisnici
 - Promene u poslovanju
-

Plan upravljanja performansom

- SQL i aplikativni kod mogu i od početka biti loši.
 - Uzroci lošeg SQL koda mogu biti:
 - Nedostatak odgovarajućih indeksa
 - Neodgovarajući izbor indeksa
 - Ako se ne upotrebljavaju raspoloživi indeksi
 - Zastarele statističke informacije BP
 - Efikasni SQL kod unutar neefikasnog aplikativnog koda.
-

Plan upravljanja performansom

- ❑ Potrebno je pronaći SQL iskaze koji troše najviše resursa.
 - Nije jednostavan zadatak
 - ❑ Iteraktivni korisnici – Ad hoc upiti
 - Mogu značajno uticati na performasu
 - Mogu biti bilo gde u sistemu
 - ❑ Koristiti Monitore i druge administratorske alate
-

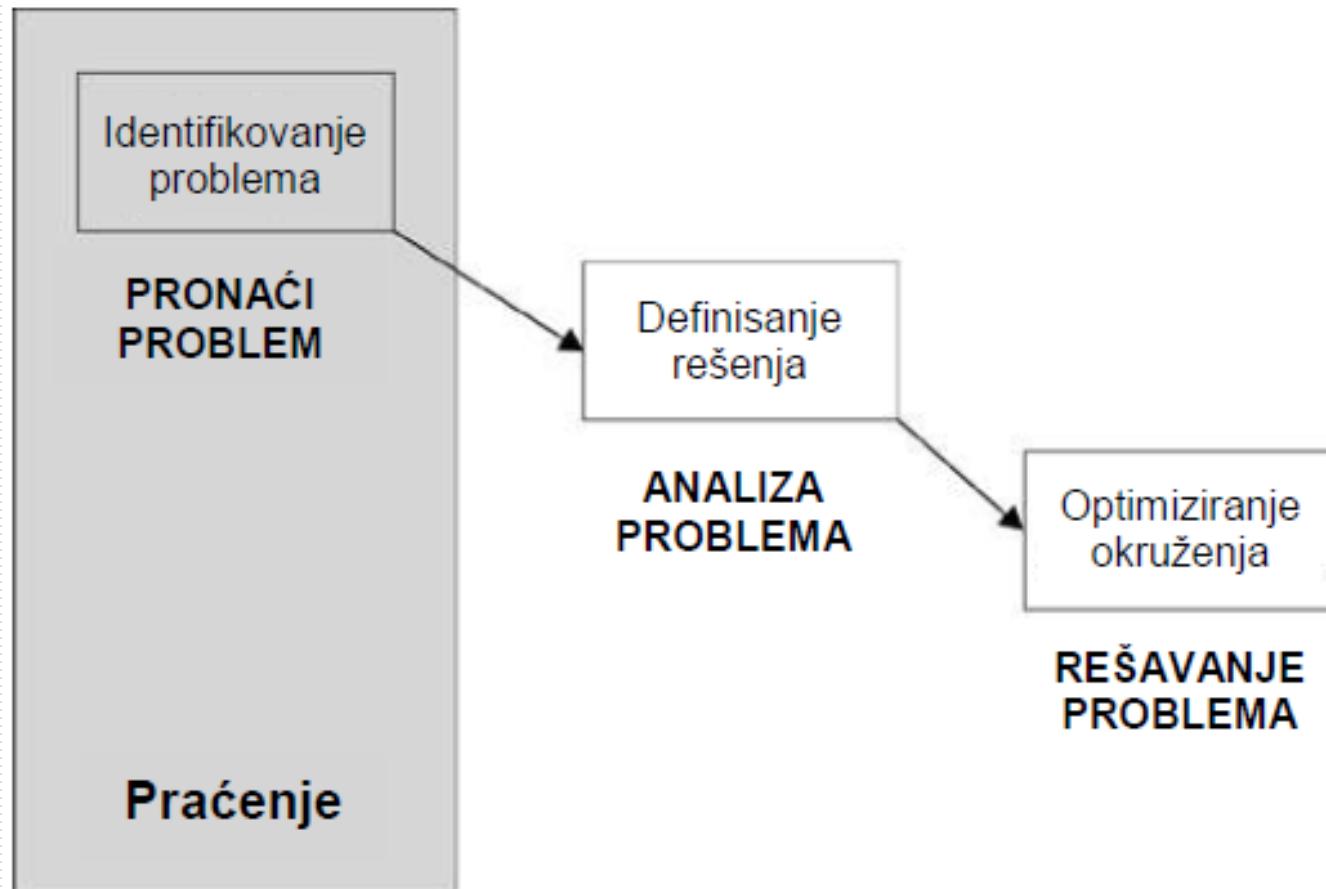
Plan upravljanja performansom

- Kada se identifikuju iskazi koji angažuju najviše resursa
 - Podešavanje se fokusira na najsuklje iskaze
 - Proveriti i druge faktore koji mogu negativno uticati na performansu:
 - Alokaciju memorije (buffer/cache za podatke)
 - I/O efikasnost (razdvajanje tabela I indeksa na disku, veličina BP...)
-

Plan upravljanja performansom

- Sastoji se iz tri koraka:
 - Prećenje
 - Analiza
 - Korekcija
-

Plan upravljanja performansom

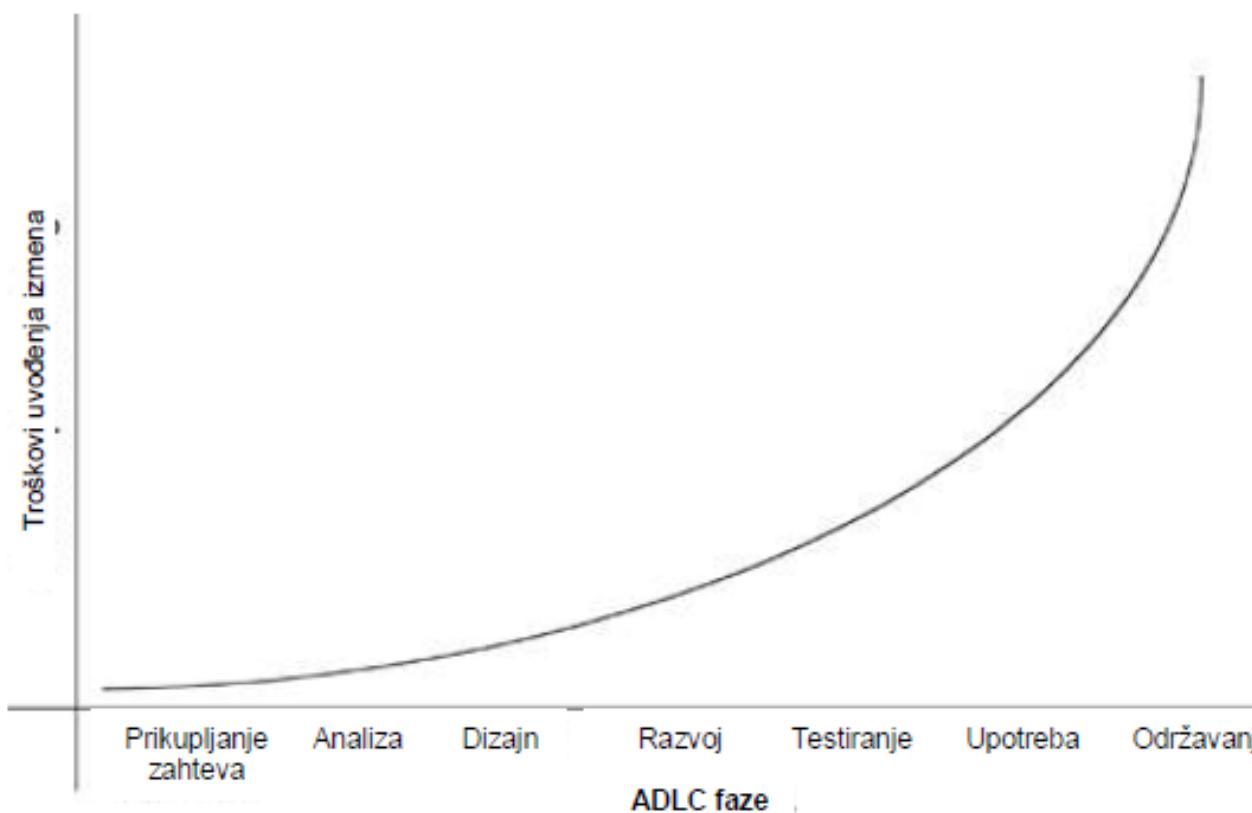


Plan upravljanja performansom

- Proaktivno
 - Mnogi problemi se mogu unapred otkriti i definisati rešenja
 - Reaktivno
 - Nemoguće je predvideti svaki tip problema
 - I sistem i aplikacija se menjaju tokom vremena
-

Plan upravljanja performansom

- Procena performanse pre razvoja aplikacije



Vrste podešavanja performanse

Podešavanje sistema

- Hardverski resursi (procesorska snaga, memorija, diskovi)
- Konfiguracija operativnog sistema (memorijski parametri, IO subsistem, raspoređivanje procesa)

Podešavanje BP

- Adekvatno fizičko projektovanje baze podataka, Izbor indeksa..
- Konfiguracija keš memorije (buffer cache)
- Konfiguracija parametara (broj istovremenih konekcija, veličina transakcione log datoteke)

Podešavanje aplikacije

- Efikasno korišćenje resursa (optimizacija koda, algoritmi, strukture podataka)
 - Izbor efikasnih SQL upita
-

Oblasti podešavanja

Sistem

Baza podataka

Aplikacija

Performansa Baze podataka

- Nikakvo podešavanje SQL koda i sistema ne može optimizovati performansu upita koji se izvršava nad:
 - Loše projektovanom ili
 - Neorganizovanom Bazom podataka
-

Alati za podešavanje performansi

- Monitori performanse
 - Alati za planiranje kapaciteta
 - Alati za SQL analizu i podešavanje
 - Alati konkretnog SUBP-a
 - Rade samo s jednim sistemom
 - Heterogeni alati
 - Rade s različitim BP i Operativnim sistemima
-

Alati za podešavanje performansi

Kod MS SQL Server-a

- Database Engine Tuning Advisor
 - System Stored Procedures
 - DBCC (Database Console Command) statements enable you to check performance statistics and the logical and physical consistency of a database
 - Built-in functions
-

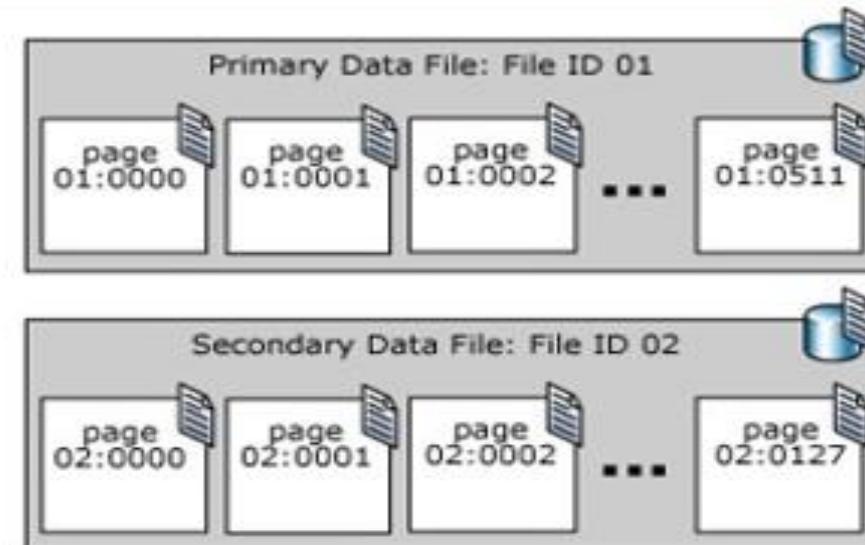
Administracija baza podataka (Vežbe)

Prostor baze podatka (1/6)

- Gde se smeštaju podaci baze podataka?
- Koji je odnos jedne BP i broja datoteka u koje se podaci baze smeštaju?
- Kako je prostor jedne (bilo koje) datoteke na disku logički organizovan?
- Šta je to Fizički blok podataka?
- Šta je to logički blok podataka?
- U kom su odnosu fizički i logički blokovi podataka?

Prostor baze podatka (2/6)

- Kako je organizovana datoteka baze podataka *SQL Server-a*?
- Šta je *DataBase Page* ili stranica baze podataka u terminologiji *SQL Server-a*?
- U terminologiji datoteka baza podataka jedinica manipulacije podacima je stranica baze podataka (***DataBase Page***).
- Fizički prostor datoteke baze podataka je podeljen na stranice baze podataka.



SLIKA 1. - Struktura datoteka BP sa prikazom stranica (***DataBase pages***)

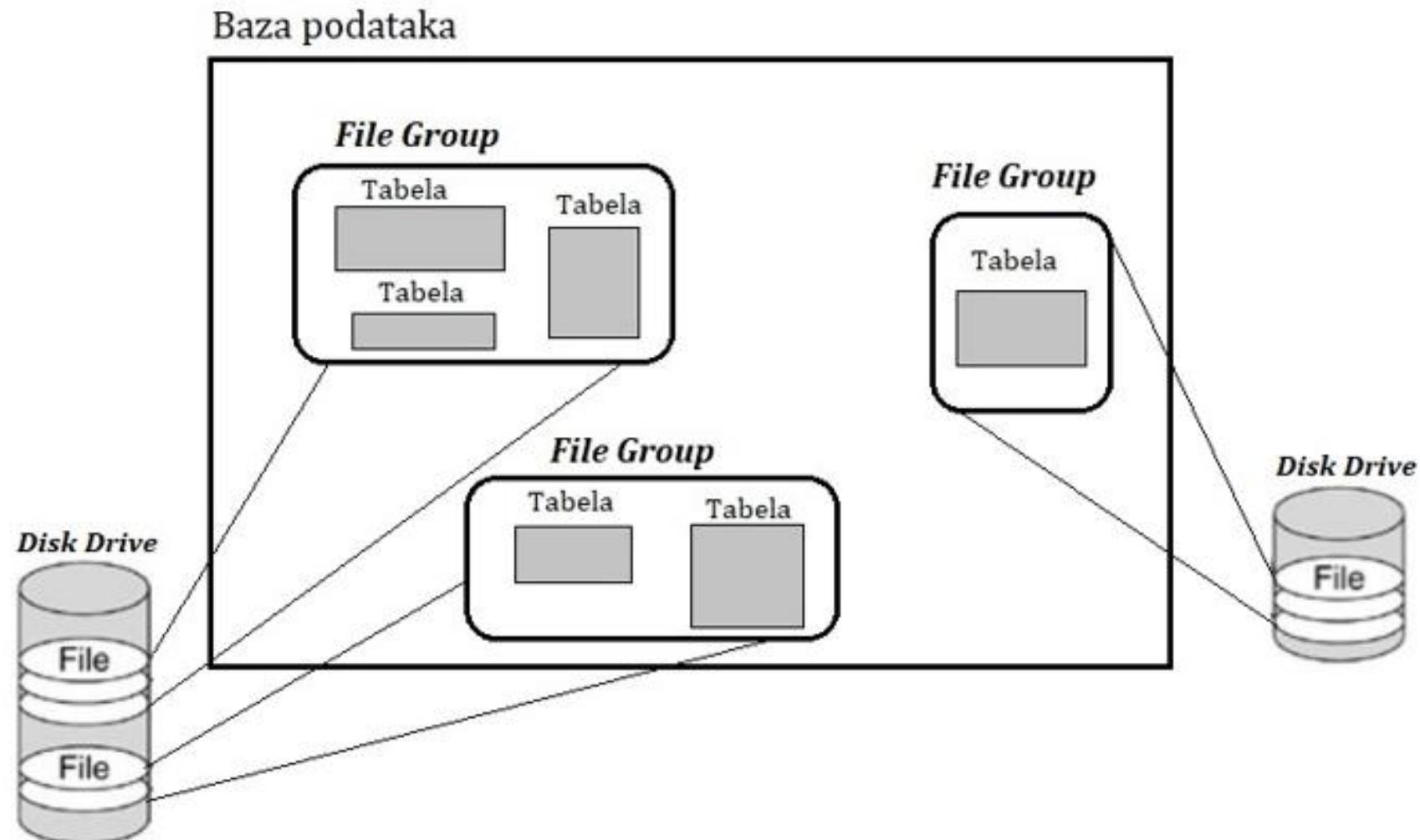
Prostor baze podatka (3/6)

- Cela ***Baza podataka*** je građena kao skup datoteka.
- Sve operacije nad Bazom podataka se svode na osnovne operacije nad datotekama.
- Podaci su korisnicima dostupni preko tabela, ali se iza tih tabela nalaze datoteke odnosno *File Group-e* (*u terminologiji SQL Server-a*).
- Tokom procesa Fizičkog projektovanja *ABP* mora mapirati svaku tabelu na odgovarajuću fizičku strukturu koja će skladištiti podatke – *File Group* (*ili u terminologiji Oracle SUBP-a - Tablespace*).

Prostor baze podatka (4/6)

- Svaka *File Group*-a je prostor na disku rezervisan za skladištenje podataka u formi datoteka.
- Fizička organizacija baze podataka se sastoji od jedne ili više *File Group*-a, a ***File Group***-a sadrži podatke jedne ili više tabela.
- *ABP*, na osnovu načina korišćenja podataka, odlučuje kako će mapirati tabele u *File Group*-e.

Prostor baze podatka (5/5)



Elementi fizičke organizacije skladištenja podataka

Prostor baze podatka (6/6)

- Podrazumevana *FileGroup*-a se naziva ***Primary***, ali se mogu definisati i druge (sekundarne) *FileGroup*-e za neku bazu podataka.
- Fajlovi imaju logički naziv unutar SUBP-a i fizičku adresu na nivou operativnog sistema.
- Specifikacija osobina fajlova se navodi pri kreiranju ili izmeni opisa Baze podataka.

Kreiranje Baze podataka

- Do sada smo to radili SQL naredbom:

CREATE DATABASE <Naziv_BazePodataka>

- Izvršimo kreiranje baze podataka pod sa nazivom *TestBazaPodataka*. Naredba je:

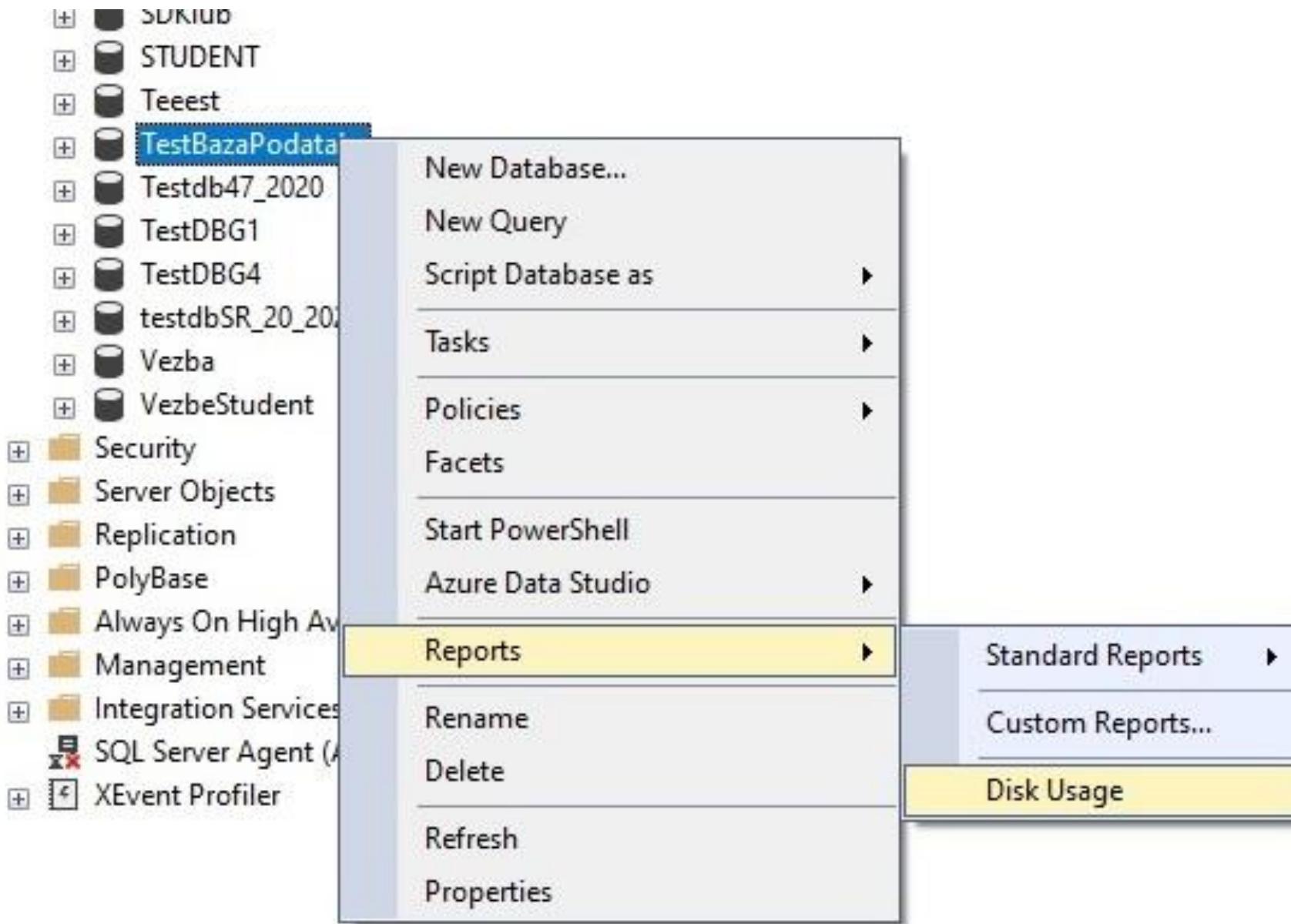
Create Database TestBazaPodataka

- Postavlja se pitanje:
 - Gde su smeštene datoteke naše baze podataka i koliko ih ima?
 - Osim toga koliko prostora zauzima naša baza podataka i šta kada se taj rezervisani prostor popuni podacima.

Datoteka Baze podataka

- Svaka baza podataka, uključujući i sistemske baze podataka, ima svoj skup datoteka.
- Nijedna baza podataka ne deli svoje datoteke s ostalim bazama podataka.
- Gde su smeštene datoteke svake baze podataka?
- Koliko datoteke baze podataka inicijalno (posle kreiranja) zauzimaju prostora?

Koliko prostora BP zauzima nakon kreiranja



Koliko prostora zauzimaju i gde se nalaze datoteke BP

Disk Usage

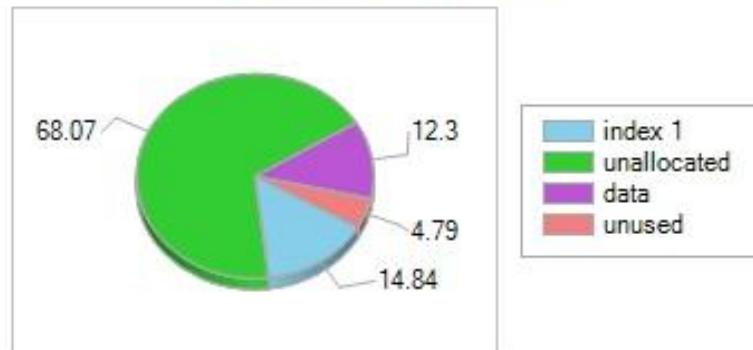
[TestBazaPodataka]

on DESKTOP-4HPGDHI at 3/18/2023 4:15:11 PM

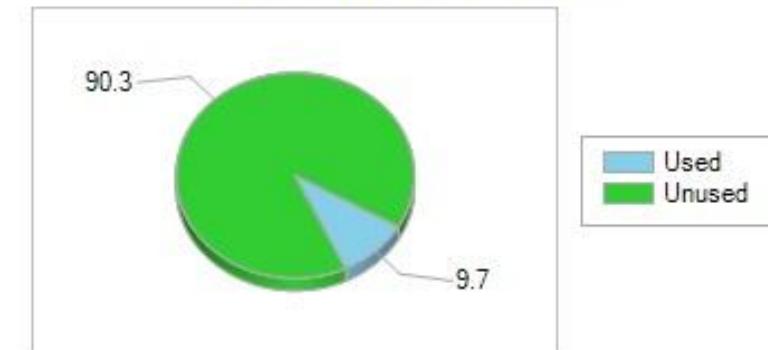
This report provides overview of the utilization of disk space within the Database.

Total Space Reserved	16.00 MB
Data Files Space Reserved	8.00 MB
Transaction Log Space Reserved	8.00 MB

Data Files Space Usage (%)



Transaction Log Space Usage (%)



No entry found for autogrow/autoshrink event for TestBazaPodataka database in the trace log.

Disk Space Used by Data Files

Filegroup Name	Logical File Name	Physical File Name	Space Reserved	Space Used
PRIMARY	TestBazaPodataka	C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\TestBazaPodataka.mdf	8.00 MB	2.63 MB

Sekundarne File Group-e

- Datoteke Baze podataka:
 - Primarne datoteke (*.mdf)
 - Sekundarne datoteke (*.ndf)
 - Datoteke Loga transakcija (*.ldf)
- Filegroups:
 - Logička kolekcija datoteka
 - Objekti se mogu kreirati nad Filegroup-om

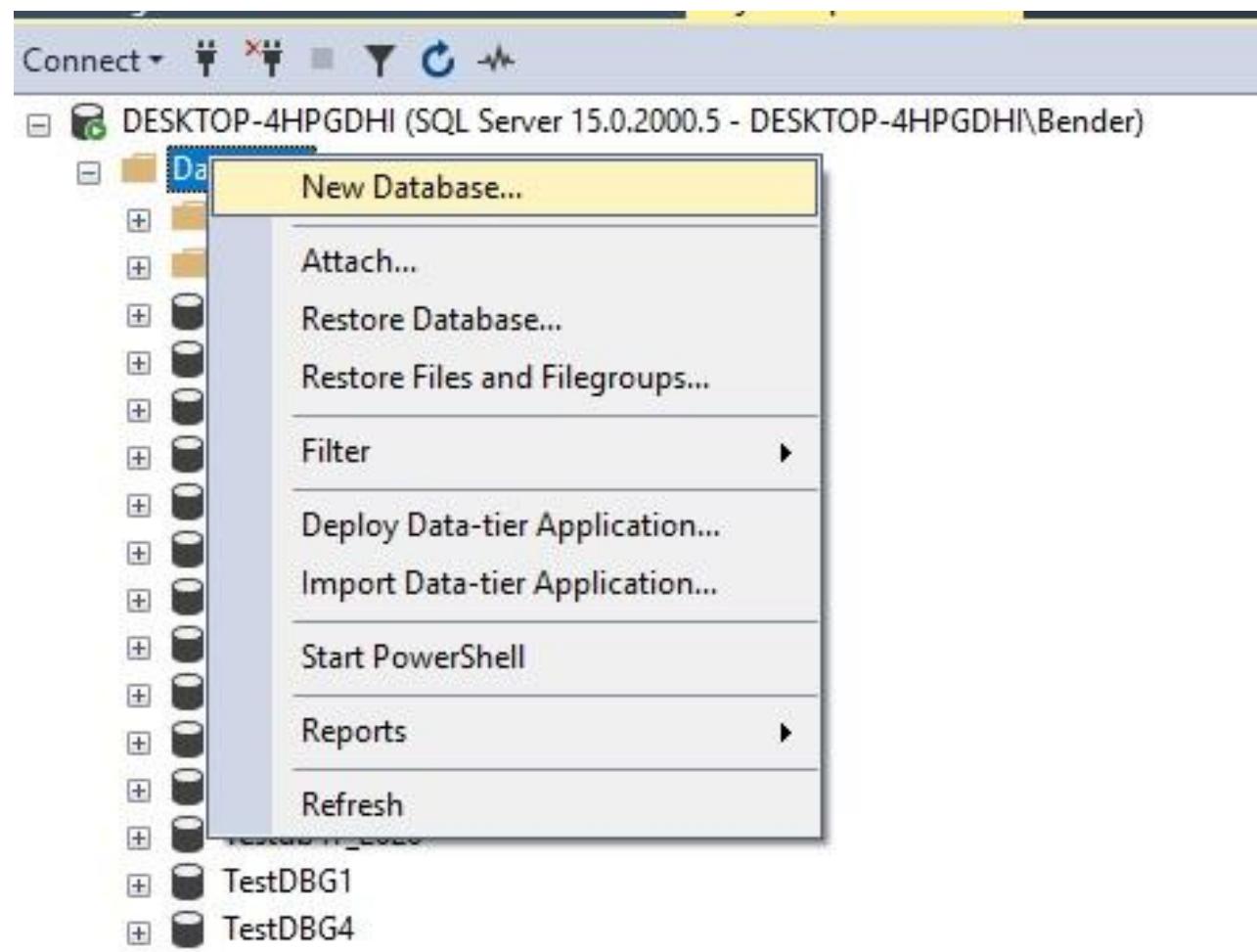
Kako se objekti BP rasporedjuju po datotekama

- Kako se definiše u koju **File Group-u** se smešta svaka Tabela (a generalno svaki objekat BP)?
- U šemi relacije se za svaku tabelu (relaciju) definiše u koju File Group će se smeštati.

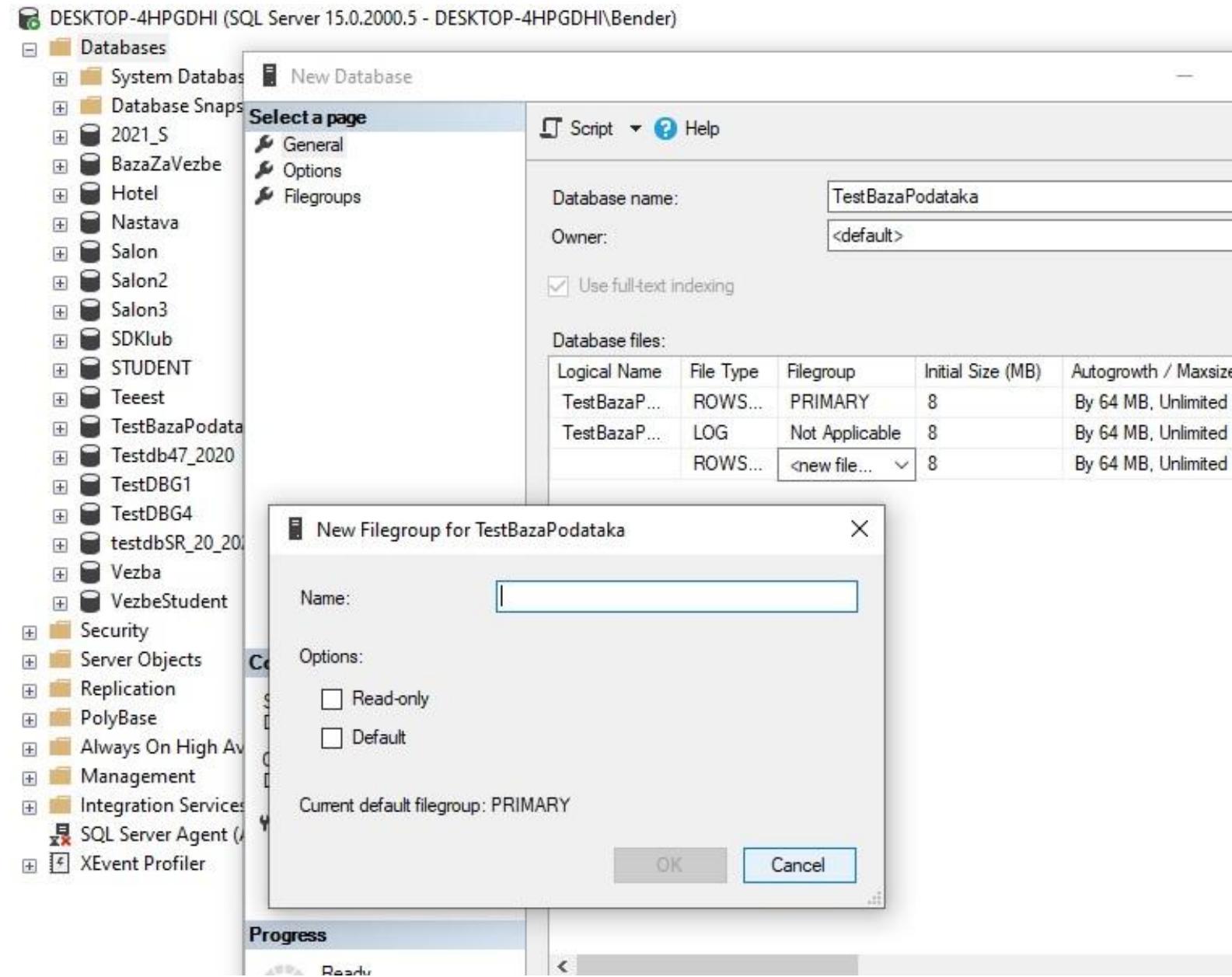
```
CREATE TABLE
  { database_name.schema_name.table_name | schema_name.table_name | table_name }
  [ AS FileTable ]
  ( { <column_definition>
    | <computed_column_definition>
    | <column_set_definition>
    | [ <table_constraint> ] [ ,... n ]
    | [ <table_index> ] }
    [ ,... n ]
    [ PERIOD FOR SYSTEM_TIME ( system_start_time_column_name
      , system_end_time_column_name ) ]
  )
  [ ON { partition_scheme_name ( partition_column_name )
    | filegroup
    | "default" } ]
  [ TEXTIMAGE_ON { filegroup | "default" } ]
  [ FILESTREAM_ON { partition_scheme_name
    | filegroup
    | "default" } ]
  [ WITH ( <table_option> [ ,... n ] ) ]
  [ ; ]
```

Definisanje u koju **File Group-u** se smešta Tabela.

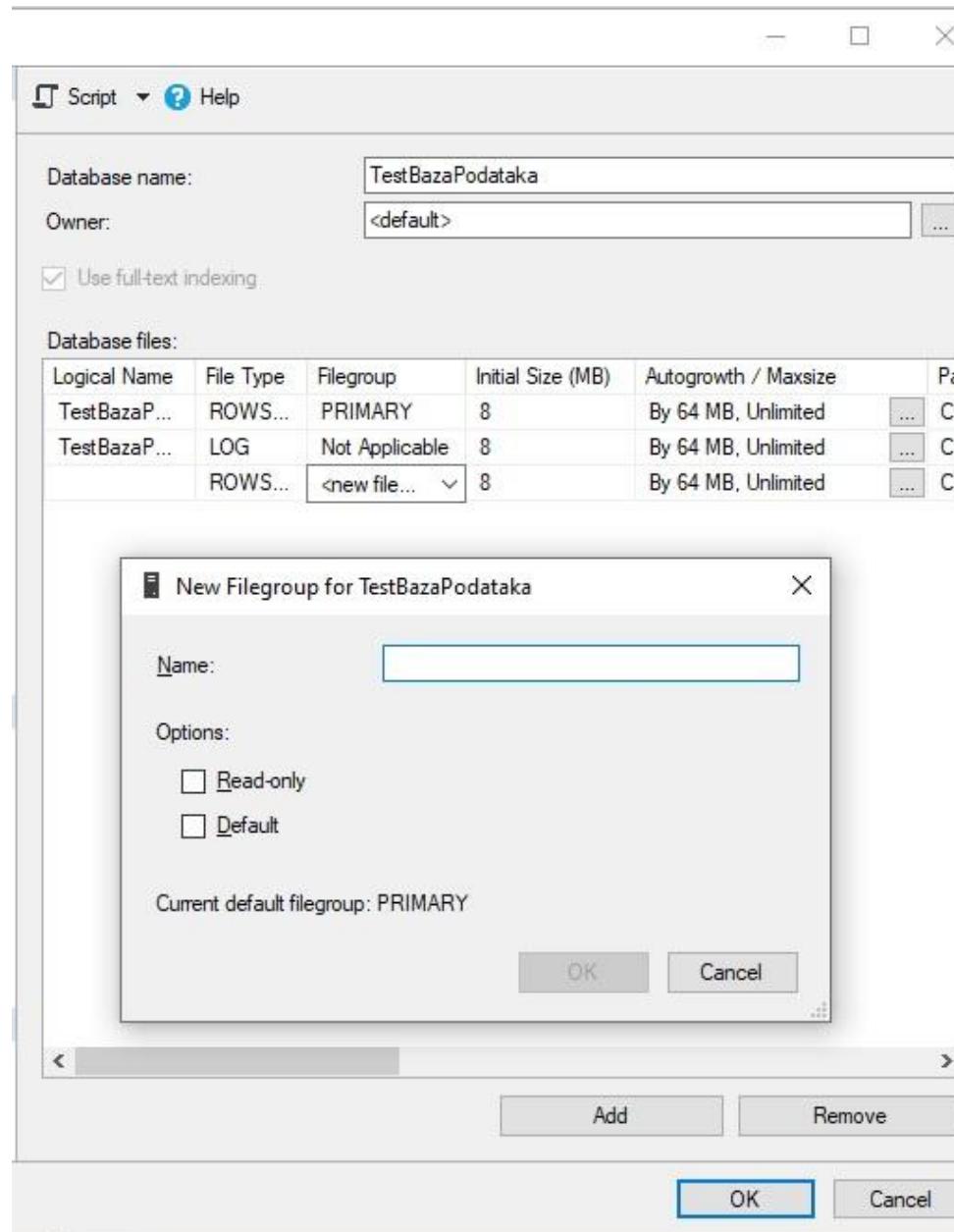
Kreiranje *File Group*-a



Kreiranje *File Group-a*



Kreiranje *File Group-a*



Zadatak 1 – Fizička organizacija BP

Kareirati Bazu podataka Hotel koja ima 4 **FileGroup**-e i tabele rasporediti na sledeći način:

<u>FileGroup</u>	<u>Tabela</u>
Primary	Gost
Sobe	TipSobe i Soba
Cene	TipIznajmljivanja, Cenovnik
Posete	Iznajmljivanje

Datoteke baze podataka smestiti na folder C:\Documents\ABP\Hotel

Faze projektovanja (svake) Baze podataka

1. Prikupljanje i analiza zahteva
2. Logičko projektovanje baze podataka
3. Izbor sistema za upravljanje bazom podataka
4. Prevođenje modela podataka
- 5. Fizičko projektovanje BP**
6. Implementacija BP

ZADATAK 1 – Napraviti logički model BP za Hotel

Treba da omogući administraciju:

1. Hotelskih soba.

Za svaku sobu evidentira se broj sobe, tip sobe (broj kreveta u sobi), da li ima TV, da li ima Mini bar.

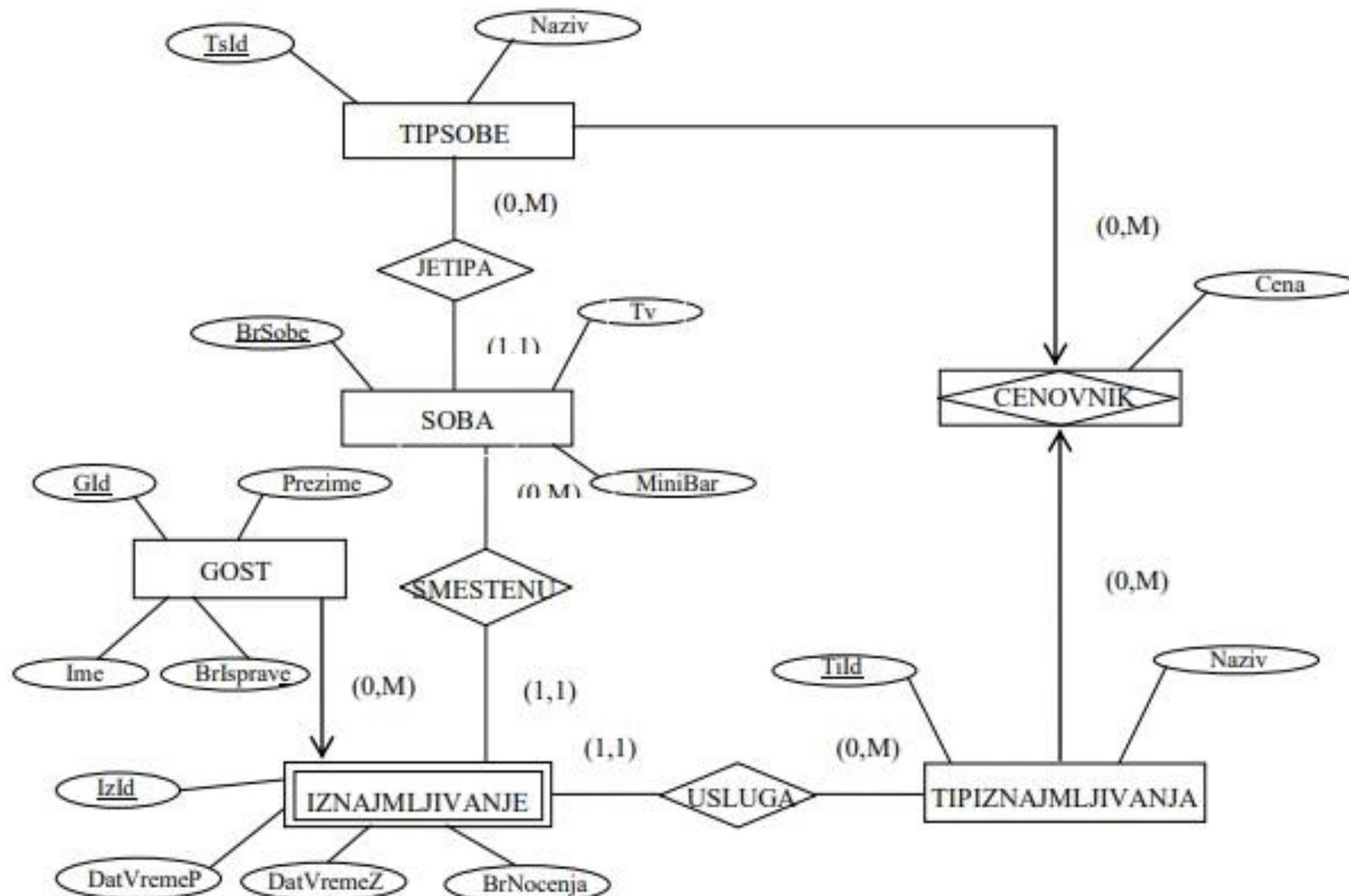
2. Iznajmljivanje hotelskih soba. Gosti hotela koriste sobe, pri čemu se evidentira:

- tip iznajmljivanja (noćenje ili dnevni boravak),
- ime, prezime i broj lične isprave svakog od gostiju koji koriste sobu,
- datum i vreme početka iznajmljivanja,
- datum i vreme završetka iznajmljivanja,
- ukupna cena iznajmljivanja (računa se po završetku boravka).

3. Cenovnika hotela.

- Cenovnik hotela za svaki tip sobe definiše cenu nocenja i cenu dnevnog boravka

ZADATAK 1 – Izrada konceptualne šeme u MOV (ER Model)



Zadatak 1 - Prevodenje MOV u Relacioni Model

LOGIČKA ŠEMA HOTEL:

TIPSOBE(TsId, Naziv)

SOBA(BrSobe, Tv, MiniBar, TsId)

SOBA[TsId] \subseteq TIPSOBE[TsId]

TIPIZNAJMLjIVANJA(Tild, Naziv)

CENOVNIK(TsId, Tild, Cena)

CENOVNIK[TsId] \subseteq TIPSOBE[TsId]

CENOVNIK[Tild] \subseteq TIPIZNAJMLjIVANJA[Tild]

GOST(Gid, Ime, Prezime, BrIsprave)

IZNAJMLjIVANjE(Gid, IzId, DatVremeP, DatVremeZ, BrNocenja, BrSobe, TiID)

IZNAJMLjIVANJE[Gid] \subseteq GOST[Gid]

IZNAJMLjIVANJE[BrSobe] \subseteq SOBA[BrSobe]

IZNAjMLjIVANjE[Tild] \subseteq TIPIZNAJMLjIVANJA[Tild]

Zadatak 1 - Prevodenje u Šemu BP razumljivu SUBP

Create Table TipSobe

```
(TsId          TinyInt Primary Key,  
 BrKreveta    TinyInt,  
 NazivTipaSobe VarChar(20)
```

) On Sobe

Create Table Soba

```
(BrSobe        Int Primary Key,  
 MiniBar       TinyInt, /* 1 - Ima Mini bar; 0 - Nema mini bar */  
 Tv            TinyInt, /* 1 - Ima Tv; 0 - Nema Tv */  
 TsId          TinyInt Foreign Key  
             References TipSobe(TsId)
```

) On Sobe

```
Create Table TipIznajmljivanja  
    (TiId  TinyInt Primary Key,  
     NazivTipaIznajmljivanja VarChar(20)  
) On Cene
```

```
Create Table Cenovnik  
    (TsId  TinyInt Foreign Key  
        References TipSobe(TsId),  
     TiId  TinyInt Foreign Key  
        References TipIznajmljivanja(TiId),  
     Cena    Money,  
     Primary Key (TsId, TiId)  
) On Cene
```

```
Create Table Gost
```

```
(Gid          TinyInt Primary Key,  
 Prezime      VarChar(18),  
 Ime          VarChar(18),  
 BrojLk       VarChar(12)
```

```
)
```

```
Create Table Iznajmljivanje
```

```
(Gid          TinyInt Foreign Key  
          References Gost(Gid),  
 IzId         TinyInt,  
 BrSobe       Int Foreign Key  
          References Soba(BrSobe),  
 TiId         TinyInt Foreign Key  
          References TipIznajmljivanja(TiId),  
 DatVremeP    SmallDateTime,  
 DatVremeZ    SmallDateTime,  
 BrNocenja    TinyInt,  
 Primary Key (Gid, IzId)
```

```
) On Posete
```

Fizične datoteke BP HOTEL

This PC > Local Disk (C:) > DATA

Name	Date modified	Type	Size
Cene	11/12/2021 6:53 PM	SQL Server Database	8,192 KB
Hotel	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
Hotel_log	2/23/2023 3:37 PM	SQL Server Database	8,192 KB
posete	12/23/2021 9:06 PM	SQL Server Database	8,192 KB
Sobe	11/20/2021 2:58 PM	SQL Server Database	8,192 KB

ZADATAK 2 – Napraviti logički model BP za SALON nameštaja

Potrebno je za SALON namestaja omogućiti administraciju:

1. Nameštaja

Za svaki komad nameštaja evidentira se naziv, Sifra, Jedinična cena, Količina (u magacinu) i Tip nameštaja (npr. Kreveti, Stolovi, Kuhinjski nameštaj,...)

2. Prodaje nameštaja

Svaka prodaja sadrži određeni broj komada nameštaja. Evidentira se datum prodaje, Broj računa i Kupac (Ime, Prezime, Adresa)

3. Akcijska prodaja

Svaka akcija ima Naziv, Datum početka i završetka . U toku trajanja akcije određeni Komadi nameštaja su na popustu. Popust se određuje za svaki komad nameštaja pojedinačno.

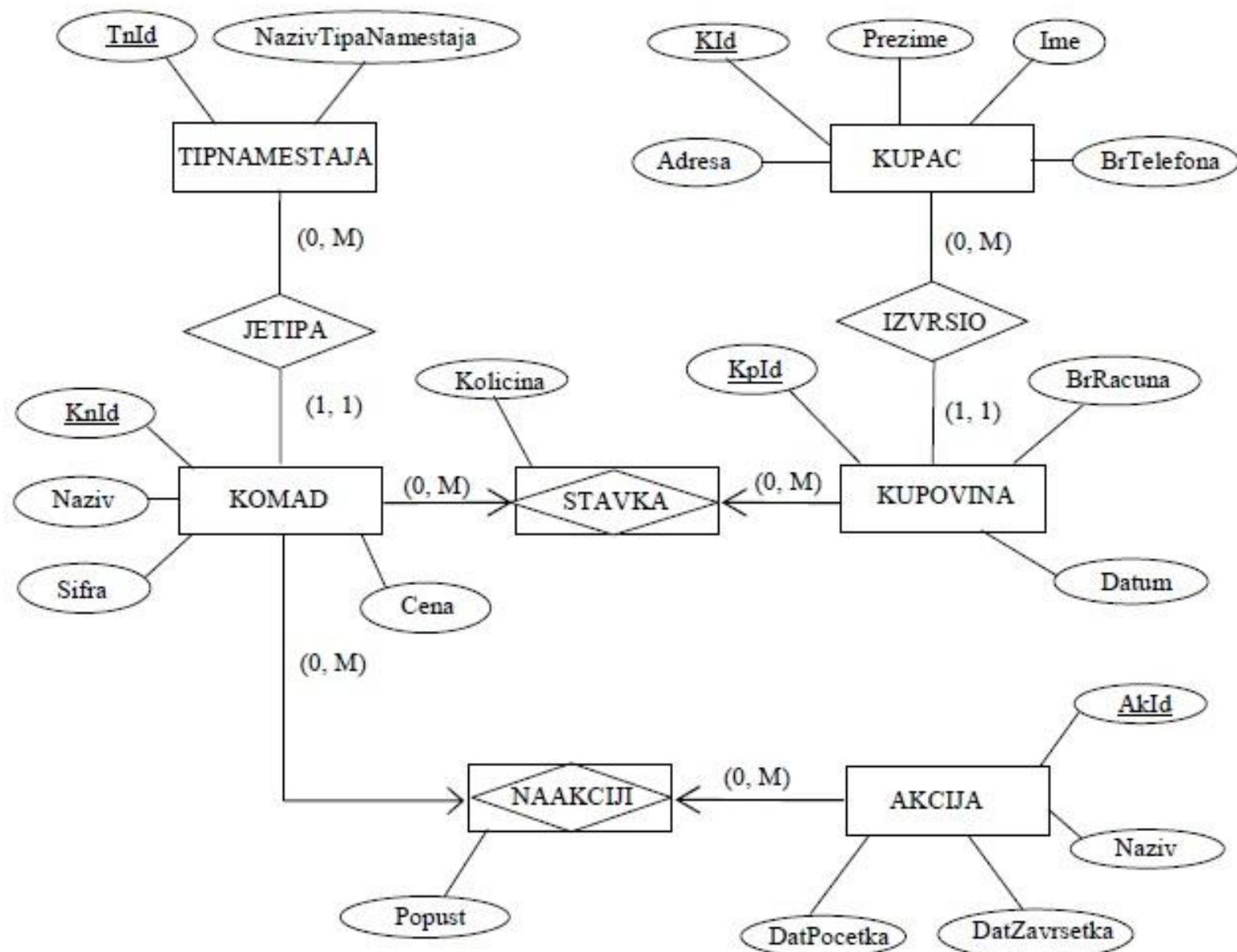
Zadatak 2 – Fizička organizacija BP

Kreirati Bazu podataka Salon koja ima 4 **FileGroup**-e i tabele rasporediti na sledeći način:

<u>FileGroup</u>	<u>Tabela</u>
Primary	KUPAC
Namestaj	TIPNAMSESTAJA, KOMAD
Akcije	AKCIJA, NAAKCIJI
Prodaja	KUPOVINA, STAVKE

Obezbediti da se podaci mogu upisivati i ćiriličnim i latiničnim pismom.
Datoteke baze podataka smestiti na folder C:\Documents\ABP\Salon

ZADATAK 2 – Izrada konceptualne šeme u MOV (ER Model)



Zadatak 2 - Prevodenje MOV u Relacioni Model

LOGIČKA ŠEMA SALON:

TIPNAMESTAJA(TnId, NazivTipaNamestaja)

KOMAD(KnId, Naziv, Sifra, Cena, TnId)

KOMAD [TnId] \subseteq TIPNAMESTAJA [TnId]

AKCIJA(AkId, DatPocetka, DatZavrsetka, Naziv)

NAAKCIJI(AkId, KnId, Popust)

NAAKCIJI [AkId] \subseteq AKCIJA [AkId]

NAAKCIJI [KnId] \subseteq KOMAD [KnId]

KUPAC(KId, Prezime, Ime, Adresa, BrTelefona)

KUPOVINA(KpId, BrRacuna, Datum, KId)

KUPOVINA [Kid] \subseteq KUPAC [Kid]

STAVKE(KpId, KnId, Kolicina)

STAVKE [KpId] \subseteq KUPOVINA [KpId]

STAVKE [KnId] \subseteq KOMAD [KnId]

Zadatak 2 - Prevodenje u Šemu BP razumljivu SUBP

```
Create Table Kupac
(KId      TinyInt Primary Key,
 Prezime  VarChar(18) Not Null,
 Ime      VarChar(18) Not Null
)

Create Table TipNamestaja
(TnId      TinyInt Primary Key,
 NazivTipaNamestaja VarChar(30)
)

Create Table Komad
(KnId      TinyInt Primary Key,
 Naziv     Varchar(25),
 Sifra    Varchar(10),
 Cena     Money,
 TnId     TinyInt
)
```

```
Create Table Akcija  
    (AkId          TinyInt Primary Key,  
     DatPocetka   SmallDatetime,  
     DatZavrsetka SmallDateTime,  
     NazivAkcije  VarChar(25)  
)  
Create Table NaAkciji  
    (AkId          TinyInt Foreign Key  
     References Akcija(AkId),  
     KnId          TinyInt Foreign Key  
     References Komad(knId),  
     Popust        Decimal(6,2),  
     Primary Key  (AkId, KnId)  
)
```

```
Create Table Kupovina  
    (KpId      TinyInt Primary Key,  
     BrRacuna  VarChar(10),  
     Datum     SmallDateTime,  
     KId       TinyInt Foreign Key  
             References Kupac(KId)  
)
```

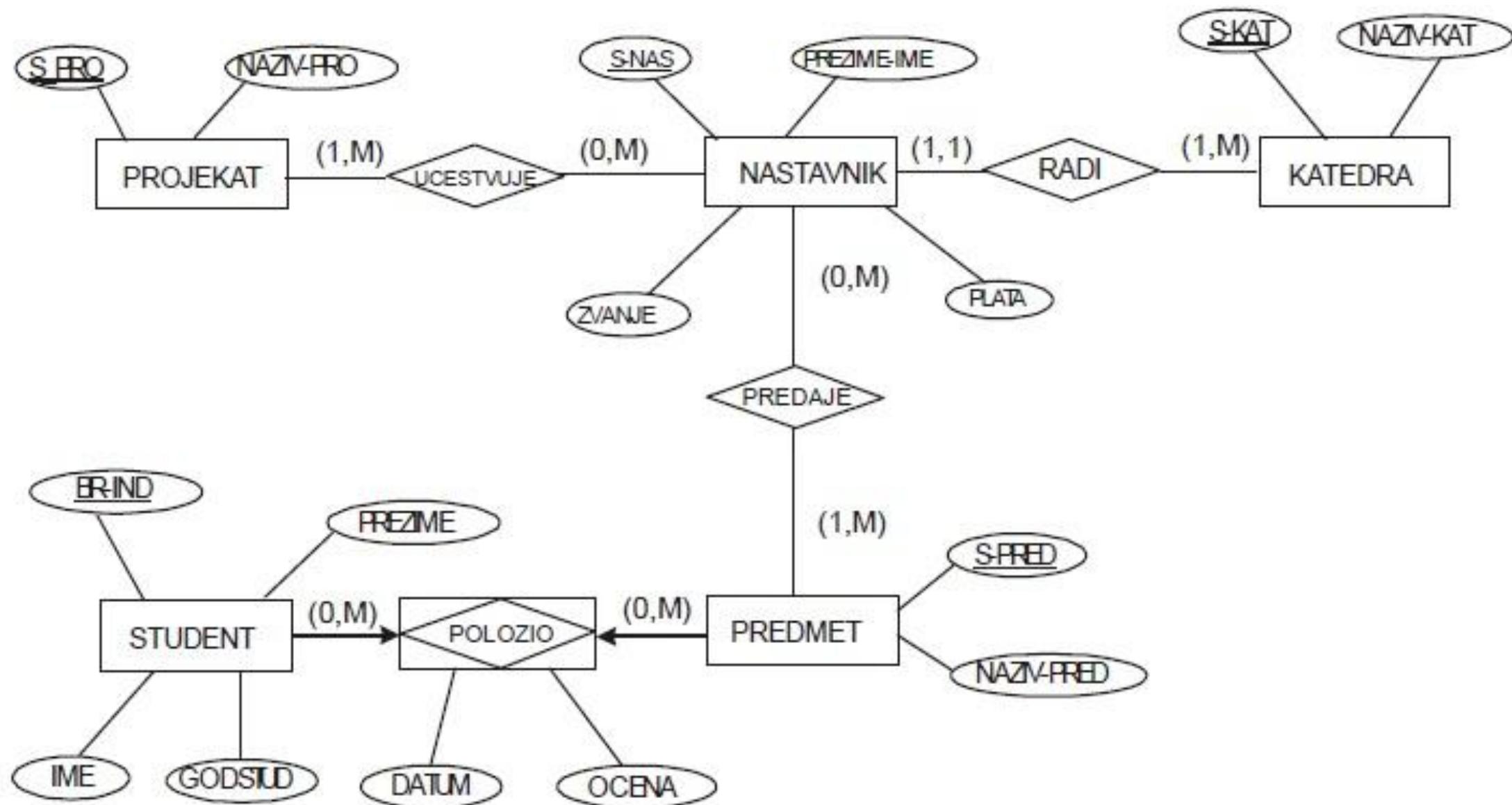
```
Create Table Stavke  
    (KpId      TinyInt Foreign Key  
             References Kupovina(KpId),  
     KnId      TinyInt Foreign Key  
             References Komad(KnId),  
     Kolicina  TinyInt,  
     Primary Key (KpId, KnId)  
)
```

ZADATAK 3 – Napraviti logički model BP za NASTAVA

Potreбно је омогући администрацију:

- Наставника (За сваког наставника evidentirati Sifru, Prezime _Ime, Zvanje i platu. На којој кatedri је наставник зaposlen)
- Проекта (За сваки проектат evidentirati: Sifru, Naziv пројекта и budzet)
- Предмете (са сифром и називом)
- Студенте (са Бројем индекса, Ime, Prezime и годину студија)
- Пратити који Наставник који предмет предaje, при чеми сваки предмет може да предaje само један наставник; и
- Полозене испите студента са: Datumom polaganja испита и Ocenom
- Настavnici осим наставе учествују и на пројектима па је за сваког наставника потребно pratiti на којем пројекту учествује, са колико сати ангажovanja и колики је износ honorара
- Осим тога потребно је pratiti недељни фонд часова predavanja неког наставника на предмету који предaje.

Konceptualna šema baze podataka - NASTAVA



Logička šema baze podataka - NASTAVA

PROJEKAT(S_Pro, NazivProjekta)

KATEDRA(S_Kat, NazivKatedre)

NASTAVNIK(S_Nas, Prezime, Ime, Zvanje, Plata, S_Kat)

NASTAVNIK [S_Kat] \subseteq KATEDRA [S_Kat]

STUDENT(BrInd, Prezime, Ime, GodStud)

PREDMET(S_Pred, Naziv_Pred)

UCESTVUJE(S_Pro, S_Nas)

UCESTVUJE [S_Pro] \subseteq PROJEKAT [S_Pro]

UCESTVUJE[S_Nas] \subseteq NASTAVNIK [S_Nas]

PREDAJE(S_Nas, S_Pred)

PREDAJE [S_Nas] \subseteq NASTAVNIK [S_Nas]

PREDAJE[S_Pred] \subseteq PREDMET [S_Pred]

POLOZIO(BrInd, S_Pred, Datum, Ocena)

POLOZIO [BrInd] \subseteq STUDENT [BrInd]

POLOZIO [S_Pred] \subseteq PREDMET [S_Pred]

Zadatak 3 – Fizička organizacija BP

Kareirati Bazu podataka Nastava koja ima samo Primary *FileGroup*-u.

- Obezbediti da se podaci mogu upisivati i čiriličnim i latiničnim pismom.
- Datoteke baze podataka smestiti na folder C:\Documents\ABP\Nastava

Administracija baza podataka

Pogledi - VIEW

(Kao objekti relacione baze podataka)

Pogledi - VIEW

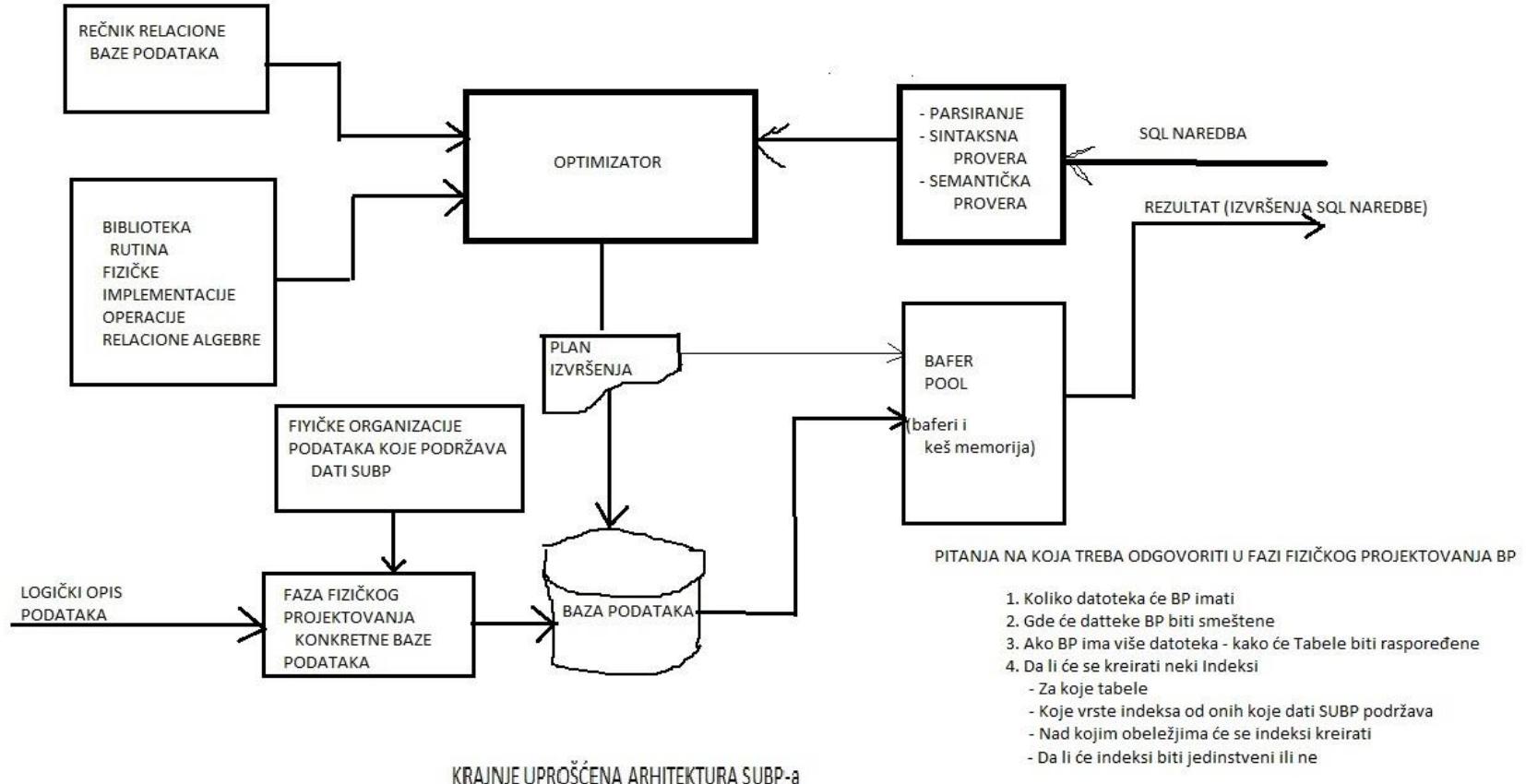
- Pogled se funkcionalno "ponaša" kao "prozor" kroz koji se vide podaci baze podataka
 - Pojednostavljuje koršćenje baze podataka
 - Može se korititi za zaštitu podataka od neovlašćenog pristupa
 - Sa aspekta performansi – pogledi se čuvaju u kompiliranom obliku
- Pogled se može posmatrati kao pdšema neke šema Relacione baze podataka.
- Kreirani pogled se u okviru SUBP-a (njegovog rečnika) čuva kao plan izvršenja čijim se izvršavanjem dobijaju podaci koji se "vide" kroz/preko pogleda.

Sintaksa naredbe za kreiranje pogleda

```
CREATE VIEW <naziv_pogleda>
[(naziv_obi1 [, naziv_obi2] , . . .)]
AS Podupit
```

- Podupit – Standardna Select SQL naredba (koju smo obradili iz predmeta Osnove baza podataka)

Pogled (VIEW) - šta je suštinski



Pogledi (View) - primer

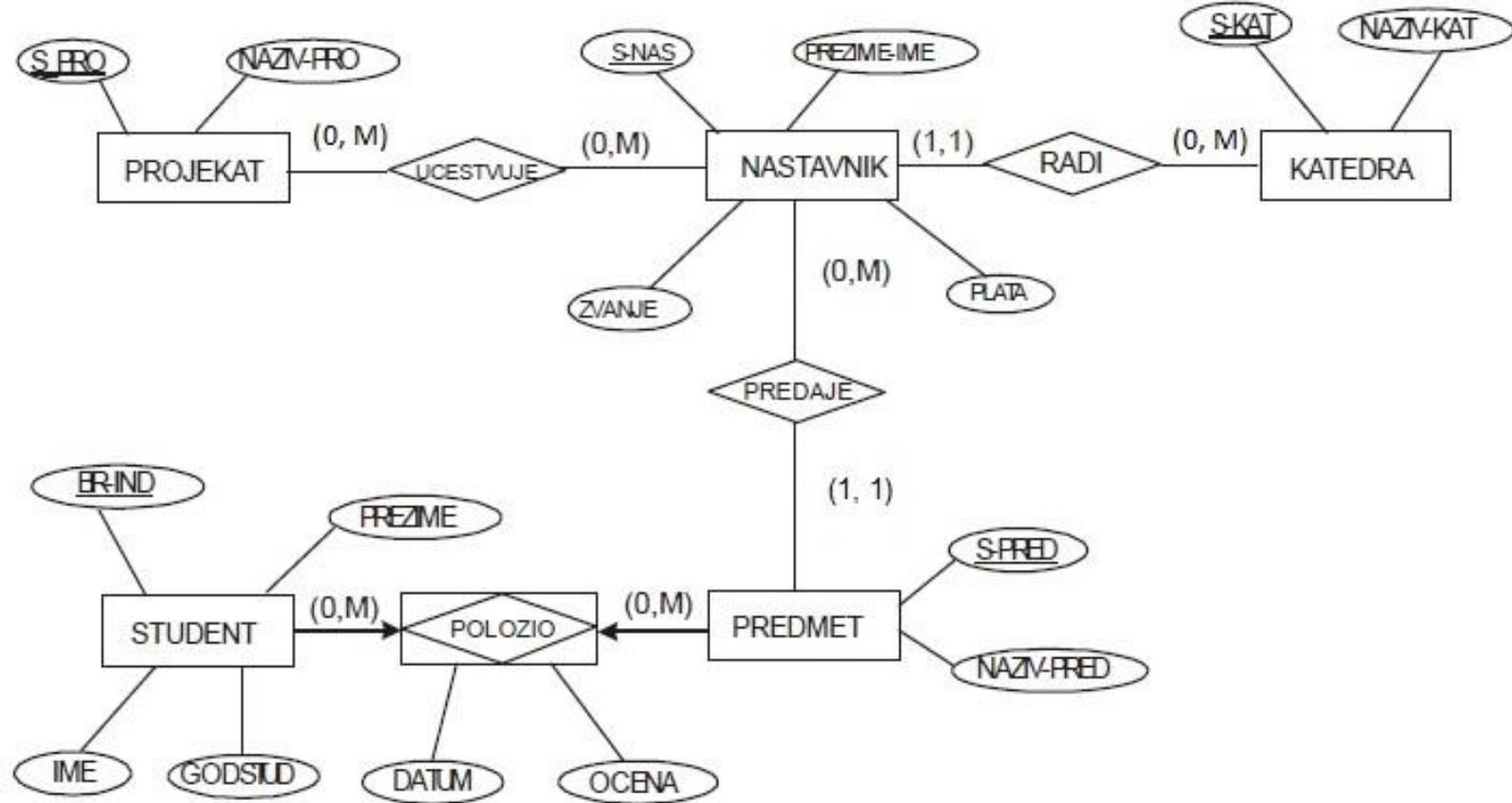
Kreirati pogled NastavnikPredmet kroz koji se "vide":

IdNas Prezime_Ime Naziv FondCasova

U radu s pogledima uočavamo sledeća 3 koraka:

1. Formiranje i testiranje upita
2. Kreiranje pogleda
3. Korišćenje pogleda

Konceptualna šema Baze podataka - Fakultet



Kreiranje pogleda - NastavnikPredmet

1. Kreiranje podupita

```
Select A.S_Nas IdNas, Prezime_Ime, Naziv_Pred, Casova As 'Fond casova'  
From Nastavnik A, Predmet B  
Where A.S_Nas = B.S_Nas
```

Kreiranje pogleda - NastavnikPredmet

2. Kreiranje pogleda

```
CREATE VIEW NastavnikPredmet  
AS  
Select A.S_Nas IdNas, Prezime_Ime, Naziv_Pred,  
Casova As 'Fond casova'  
From Nastavnik A, Predmet B  
Where A.S_Nas = B.S_Nas
```

Kreiranje pogleda - Izvršena DDL naredba kreiranja pogleda

The screenshot shows the SSMS interface. On the left is the Object Explorer pane, which displays the database structure of 'DESKTOP-4HPGDHI (SQL Server 15.0.2014.5)'. Under the 'Fakultet' node, the 'Views' folder is expanded, showing a 'NastavnikPredmet' view. The main window on the right contains the following SQL code:

```
CREATE VIEW NastavnikPredmet
AS
Select A.S_Nas IdNas, Prezime_Ime, Naziv_Pred,
Casova As 'Fond casova'
From Nastavnik A,Predmet B
Where A.S_Nas = B.S_Nas
```

Korišćenje pogleda

3. Korišćenje pogleda

- Pogledi se koriste kao i bazne tabele.
- Korisnik ne zna da li koristi baznu tabelu ili pogled (odnosno izvedenu tabelu).

Na primer:

```
Select *
From
    NastavnikPredmet
```

Uklanjanje pogleda

```
DROP VIEW <Naziv_Pogleda>
```

Ukloniti iz baze pogled pod nazivom NastavnikPredmet.

```
Drop View NastavnikPredmet
```

Pogledi – Zadaci za vezbu

Z2: Kreirati pogled PolozeniIspiti kroz koji se vidi:

BrInd	Prezime	Ime	NazivPredmeta	Datum	Ocena
-----	-----	-----	-----	-----	-----

1. Koristeći pogled PolozeniIspiti prikazati sve ispite koje je polozio student s Brojem Indeksa 'E 7398'
2. Prikazati Broj indeksa i Prezime ime studenata i ocenu kojom je ispit polozio, za studente koji su polozili ispit i predmeta 'Osnove baza podataka'.

Pogledi – Zadaci za vezbu

```
Create View PolozeniIspiti (BrInd, Prezime, Ime,  
NazivPredmeta, DatumIspita, Ocena) As  
Select B.Br_Ind, Prezime, Ime, Naziv, Datum, Ocena  
From Student A, Polozio B, Predmet C  
Where A.Br_Ind = B.Br_Ind And  
B.S_Pred = C.S_Pred
```

Pogledi – Zadaci za vezbu

z2.1:

```
Select NazivPredmeta, DatumIspita, Ocena  
From StudentPolozio  
Where BrInd = 'E 7398'
```

Pogledi – Zadaci za vezbu

Z2 .2:

```
Select BrInd, Prezime, Ime, Ocena  
From PolozeniIspiti  
Where NazivPredmeta = 'Osnove baza podataka'
```

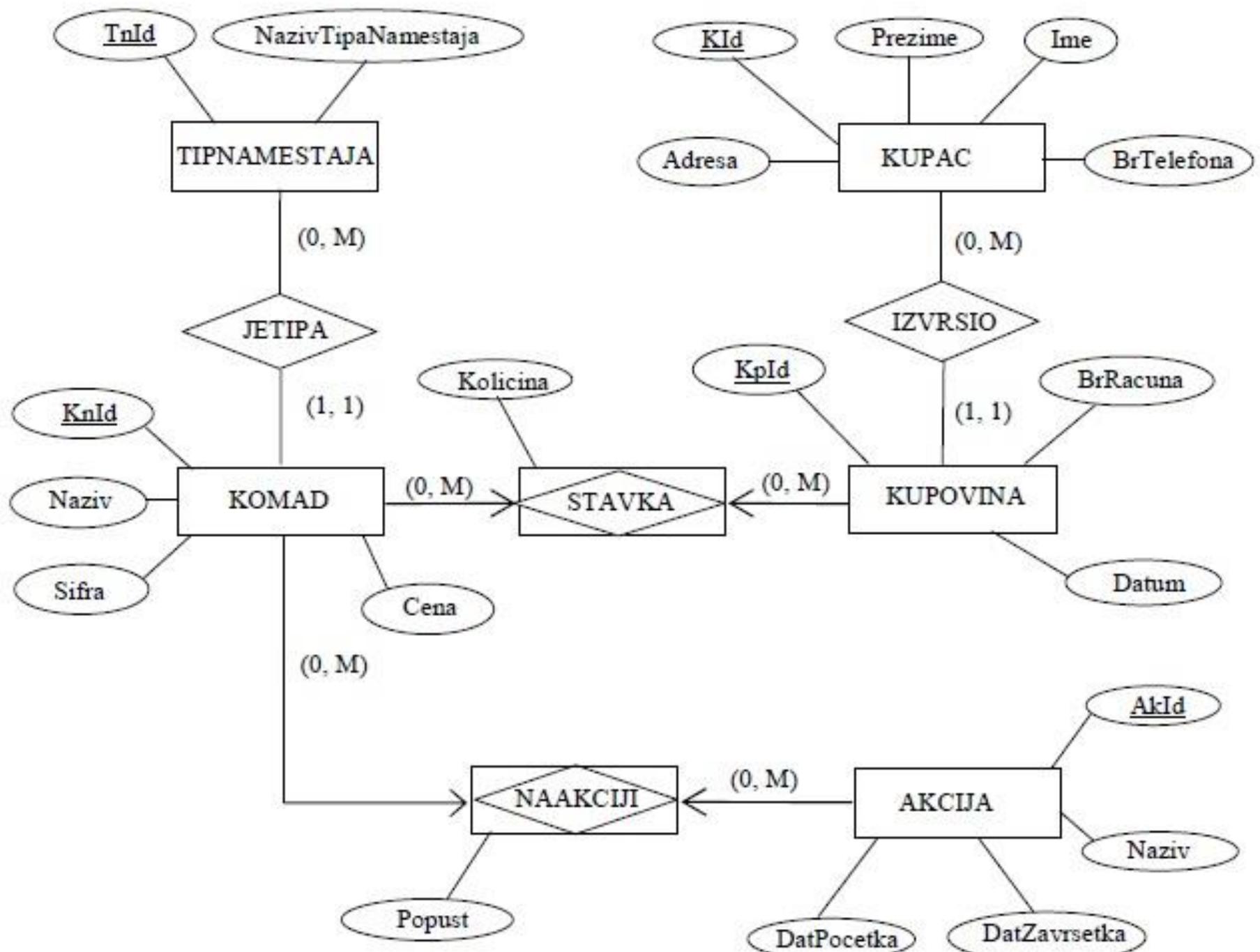
Pogledi – Zadaci za vezbu

Z3. Kreirati pogled Cene kroz koji se vidi:

NazivTipaSobe (VrstaSobe), NazivTipaIznajmljivanja (Najam) i Cena

Z4. Kreirati pogled Posete kroz koji se vidi:

Gid, Prezime, Ime gosta, BrSobe, TsId, NazivTipa Sobe, TiId, NazivNajma, BrNocenja i Datum dolaska u hotel



Pogledi – Zadaci za vezbu

Z3. Kreirati pogled Cene.

Create View Cene

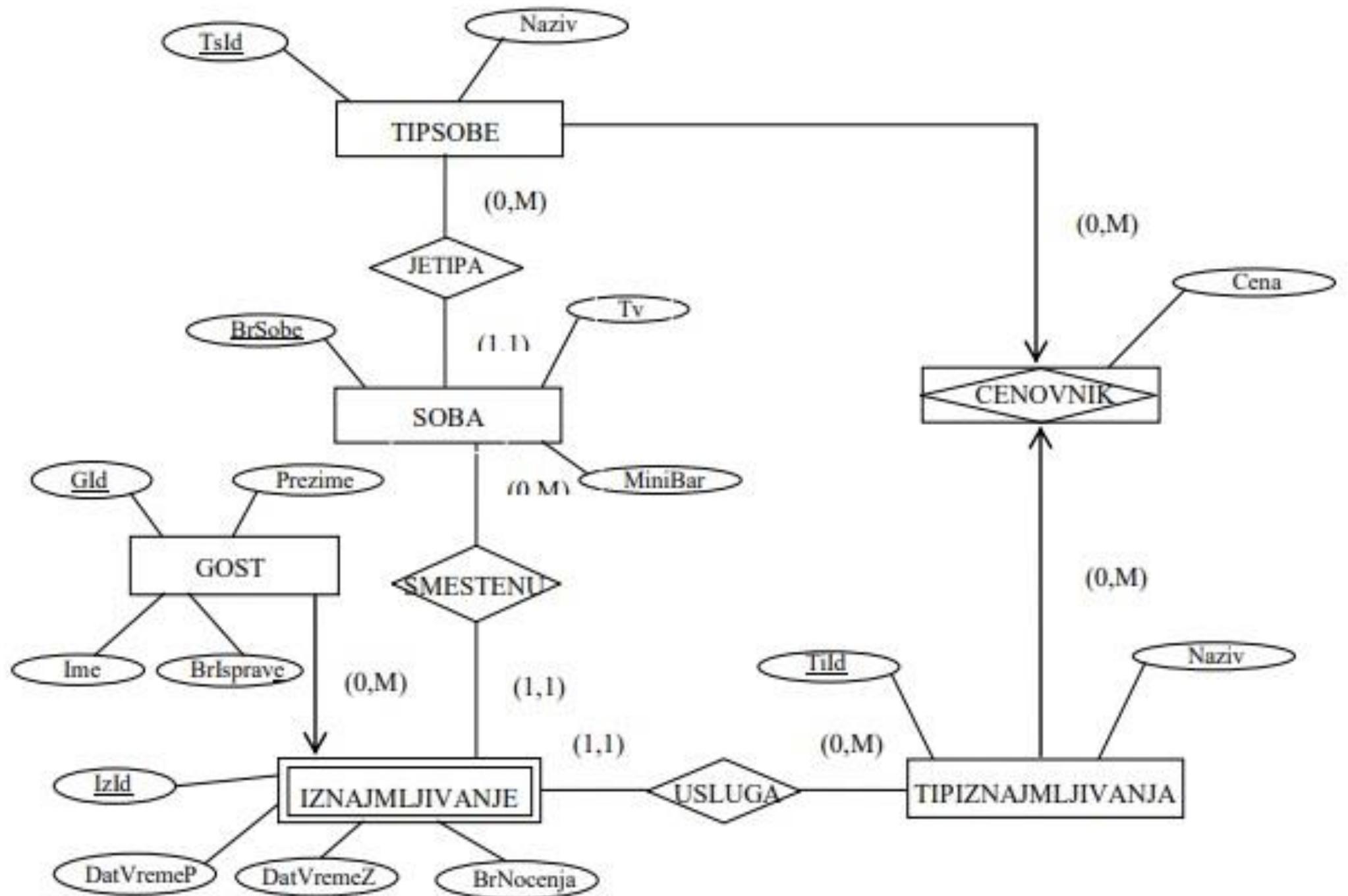
(VrstaSobe, Najam, Cena)

As

Select NazivTipaSobe, NazivTipaIznajmljivanja, Cena

From TipSobe A, Cenovnik B, TipIznajmljivanja C

Where A.TsId=B.TsId And B.TiId=C.TiId



Pogledi – Zadaci za vezbu - (Kreiranje pogleda POSETE)

Z4. Kreiranje pogled Posete

Create View Posete

(Gid, Prezime, Ime, BrSobe, TsId, TipSobe, TiId, TipNajma,
BrNocenja, Datum)

As

Select A.Gid, Prezime, Ime, E.BrSobe, B.TsId, NazivTipaSobe,
D.TiId, NazivTipaIznajmljivanja, BrNocenja, DatVremeP
From Gost A, TipSobe B, Soba C, TipIznajmljivanja
D, Iznajmljivanje E

Where A.Gid = E.Gid And C.BrSobe=E.BrSobe And C.TsID =
B.TsID And D.TiId = E.TiID

Zadatak – Koriscenje pogleda POSETE

ZADATAK 5: Koristeci pogled POSETE za Goste koji su imali vise od dve posete hotelu prikazati Prezime, Ime, Ukupan broj poseta hotlu (BrPoseta), Ukupan Broj nocenja (BrNocenja) i UkupanIznos novca placenog za boravak u Hotelu (za sve posete). Rezultate urediti po opadajucem ukupnom broju nocenja.

Korišćenje pogleda POSETE

ZADATAK 5 - Rešenje:

```
Select Prezime, Ime, Count(*) As BrPoseta, Sum(BrNocenja) As  
UkupnoNocenja, Sum(BrNocenja*Cena) As UkupanIznos  
From Cenovnik C, Posete P  
Where C.TsId = P.TsId And C.TiId = P.TiId  
Group By Prezime, Ime Having Count(*) > 2  
Order By Sum(BrNocenja) Desc
```

Pogledi - Zadaci

ZADATAK 6:

Kreirati pogled (VIEW) pod nazivom KomadiNaAkciji kroz koji se za sve akcije vide:
NazivAkcije, BrojKomada - nameštaja na akciji i prosecan procenat (ProsecanPopust)
- popusta zaokruzen na ceo broj.

Pogledi - Zadaci

ZADATAK 6 - Rešenje:

```
Create View KomadiNaAkciji  
As  
Select NazivAkcije, Count (*) As BrKomada,  
      Round(Avg(Popust), 0) ProsecanPopust  
From Akcija A, NaAkciji B  
Where A.AkId = B.AkId  
Group By NazivAkcije
```

Pogledi - Zadaci

ZADATAK 7: Kreirati pogled (VIEW) pod nazivom ProdataKolicina kroz koji se vide: Identifikator komada namestaja (KnId), Naziv (Komada namestaja) i UkupnaProdataKolicina - svakog Komada, za komade namestaja koji su prodati u kolicini vecoj od jednog komada.

Pogledi - Zadaci

ZADATAK 7 - Rešenje:

Create View

```
ProdataKolicina (S_Komada, NazivKomada, ProdataKolicina)
AS
Select A.KnId, Naziv, Sum(Kolicina)
From Komad A, Stavke B
Where A.KnId = B.KnId
Group By A.KnId, Naziv Having Sum(Kolicina) > 1
```

Pogledi - Zadaci

ZADATAK 8:

Kreirati pogled (VIEW) pod nazivom KupovineKupca kroz koji se vide: Prezime, Ime, Ukupan broj kupovina i Ukupan iznos novca koji je potrosen na sve njihove kupovine.

Pogledi - Zadaci

ZADATAK 8 – Rešenje:

Create View KupovineKupca

(KupacId, PrezimeKupca, ImeKupca, BrojKupovina, PotroseniIznos)

AS

Select A.Kid, Prezime, Ime, Count(Distinct B.KpId), Sum(Cena *Kolicina)

From Kupac A, Kupovina B, Stavke C, Komad D

Where A.Kid = B.Kid And B.KpId = C.KpId And C.KnId = D.KnId

Group By A.Kid, Prezime, Ime

Pogledi - Zadaci

ZADATAK 9:

Kreirati pogled (VIEW) pod nazivom ProdatiKomadi kroz koji se vide: Identifikator komada namestaja (KnId), Naziv (Komada namestaja) , Cena, Ukupna prodata kolicina po svakom komadu i Ukupan iznos novca dobijen od prodaje svakog komada namestaja.

Pogledi - Zadaci

ZADATAK 9 - Rešenje:

```
Create View ProdatiKomadi  
    (KomadId, NazivKomada, CenaKomada, ProdataKolicina, IznosProdaje)  
AS  
Select A.KnId, Naziv, Cena, Sum(Kolicina), Sum(Kolicina*Cena)  
From Komad A, Stavke B  
Where A.KnId = B.Knid  
Group By A.KnId, Naziv, Cena
```

Information Schema (1/1)

- Šta je Information Schema?
 - Za čega se koristi Information Schema?
 - Zbog čega je uveden koncept INFORMATION SCHEMA?
-

Information Schema (2/2)

- ❑ **INFORMATION_SCHEMA** - je skup sistemskih pogleda.
- ❑ Omogućava korisnicima da pristupe metapodacima o strukturi i sadržaju baze podataka.
- ❑ Nalazi se u većini modernih Relacionih SUBP. Ona omogućava korisnicima da pristupe metapodacima o strukturi i sadržaju baze podataka.
- ❑ Koristi za prikazivanje informacija o samoj bazi i objektima baze podataka kao što su tabele, pogledi, ograničenja, funkcije, indeksi itd.
- ❑ Obezbeđuje se da se, skrivajući implementacione detalje same Sistemske baze konkretnog SUBP-a, rečniku pristupa na potpuno isti funkcionalan način.
- ❑ Pristup sistemskim podacima preko pogleda Information Schema je potpuno isti kod svih SUBP-ova koji je podržavaju.

Information Schema – Zadaci (1/3)

ZADATAK: Prikazati sve relacije (Tabele) u bazi podataka Hotel.

```
Select TABLE_CATALOG, TABLE_NAME, TABLE_TYPE  
From INFORMATION_SCHEMA.Tables  
Order By TABLE_NAME
```

Information Schema - Zadaci (2/3)

- **ZADATAK:** Prikazati sva obeležja svih tabela baze podataka FAKULTET za koja je prilikom kreiranja šeme te baze podataka definisano da ne smeju poprimiti *NULL* vrednost.

```
Select Column_Name, Table_Name, Is_Nullable, Data_Type  
From Information_Schema.Columns  
Where Is_Nullable = 'NO'  
Order by Column_Name
```

Information Schema - Zadaci (3/3)

- ZADATAK 1:** Prikazati sva obeležja relacije (tabele) IZNAJMLJIVANJE u bazi podataka Hotel.
 - ZADATAK 2:** Koristeci INFORMATION_SCHEMA prikazati sve tabele u bazi podataka Fakultet.
 - ZADATAK 3:** Koristeci INFORMATION_SCHEMA prikazati sva obeležja tabele Nastavnik iz baze podataka Fakultet.
 - ZADATAK 4:** Koristeci INFORMATION_SCHEMA prikazati sve poglede u bazi podataka Fakultet.
-