7 – Programska podrška za servere

S A D R Ž A J

7.1 Klijent server Data Base arhitektura

7.2 *Data Base Management System*

7.3 Alati za upite

7.4 SQL jezik

7.5 Vrste povezivanja sa DBMS

7.5.1 ODBC **7.5.2** OLE DB

7.5.3 ADO.Net **7.5.4** JDBC

7.6 Web/Data Base sistem

7.1 Arhitektura tri šeme

- Baze podataka i sistemi za njihovo upravljanje predstavljaju nezaobilazne komponente klijent server sistema.
- Smatra se da ne postoji ni jedna ozbiljnija klijent server aplikacija koja se ne oslanja na neku bazu podataka i njen mehanizam za upravljanje.
- Baza podataka se može definisati kao kolekcija međusobno povezanih podataka memorisanih na nekom zajedničkom medijumu.
- Pod podacima podrazumevamo činjenice o nekom segmentu realnog sistema koje imaju tačno određeno (implicitno) značenje za korisnike
- Podaci koji se čuvaju mogu se sastojati od svega nekoliko ulaznih podataka ili redova koji predstavljaju jednostavan adresar sa imenima, adresama i telefonskim brojevima.
- Baza podataka može da sadrži i milione zapisa koji opisuju katalog proizvoda, audio i video zapisa ili platni spisak neke velike kompanije.
- Kao i kod komponenti sistemskog softvera, gde se korisnici odvajaju od aplikacija, tako se i u upravljanju bazama podataka koriste neki modeli za odvajanje korisničkih aplikacija od fizičke baze podataka

7.1 Arhitektura tri šeme

Kod baze podataka taj model je predstavljen kroz ***arhitekturu tri šeme***:

1. ***Interni nivo*** koristi internu šemu pomoću koje se opisuje fizička struktura baze podataka. Interna šema opisuje sve detalje memorisanja podataka i pristupne mehanizme i puteve u bazi.
2. ***Konceptualni nivo*** koristi konceptualnu šemu pomoću koje se opisuje logička struktura baze podataka. U stvarnosti to je globalni opis baze podataka koji od korisnika krije nepotrebne i opterećujuće detalje fizičke arhitekture baze podataka. Konceptualni nivo opisom entiteta (objekata realnog mini sveta), tipom podataka i međusobnih odnosa između entiteta, omogućuje projektantima i korisnicima baze podataka da se potpuno posvete događajima u mini svetu, relacijama koje vladaju u njemu, atributima entiteta, i na taj način što tačnije definišu upotrebnu vrednost i efikasnost jedne konkretne baze podataka.
3. ***Eksterni nivo*** uključuje određeni broj eksternih šema, drugim rečima korisničkih pogleda kako na deo mini sveta koji ih interesuje, tako i na deo baze podataka koja prati taj mini svet. On treba da od korisničke grupe sakrije deo baze podataka koji nije interesantan za njih.

7.1 Client Server DataBase Arhitektura

- Kod klijent/server DataBase aplikacija, uglavnom se koriste **relacione baze podataka**, gde server predstavlja server baze podataka.
- Interakcija između klijenta i servera je u obliku **transakcija** u kojoj klijent nešto zahteva od baze podataka i dobija odgovor od nje.
- U arhitekturi takvog sistema, server je odgovoran za održavanje baze podataka, u tu svrhu je potreban veoma složen sistem za upravljanje.
- Postoje različite klijentske aplikacije koje omogućavaju pristup bazi
- Ono što je za njih zajedničko, tj. " lepak" koji povezuje klijent i server je softver koji omogućava klijentu da napravi zahtev za pristup bazi podataka servera, a to je **SQL** (***Structured Query Language***).
- Prema ovoj arhitekturi, sva logika aplikacija (softver koji se koristi za analizu podataka) nalazi se kod klijenta, dok se na serveru nalaze moduli koji se bave upravljanjem i održavanjem baze podataka.

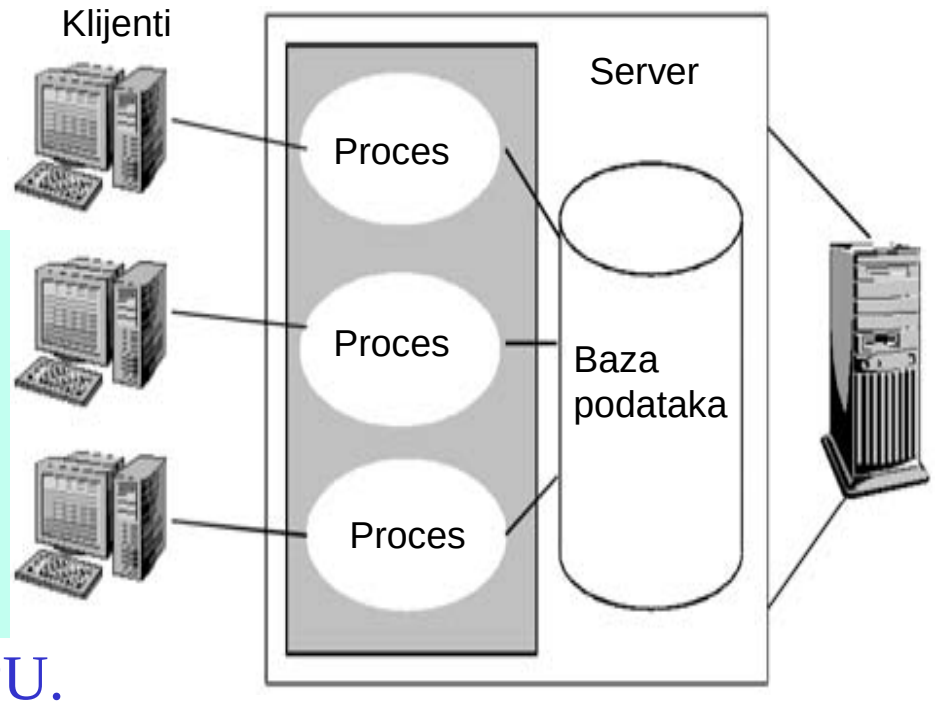
Prednosti ove arhitekture ogledaju se u sledećem primenama:

1. Pretraživanje i sortiranje velikih baza podataka koje zahtevaju veliki memorijski prostor, velike brzine procesora i Input/Output sistema.
2. Smanjuju saobraćaj kroz mrežu jer se sve odrađuje na serveru

7.1 Client Server DataBase Arhitektura

Postoje razne vrste dostupnih klijent/server arhitektura baza podataka:

1. ***Process-per-client***
2. ***Multi-threaded***
3. ***Hybrid***

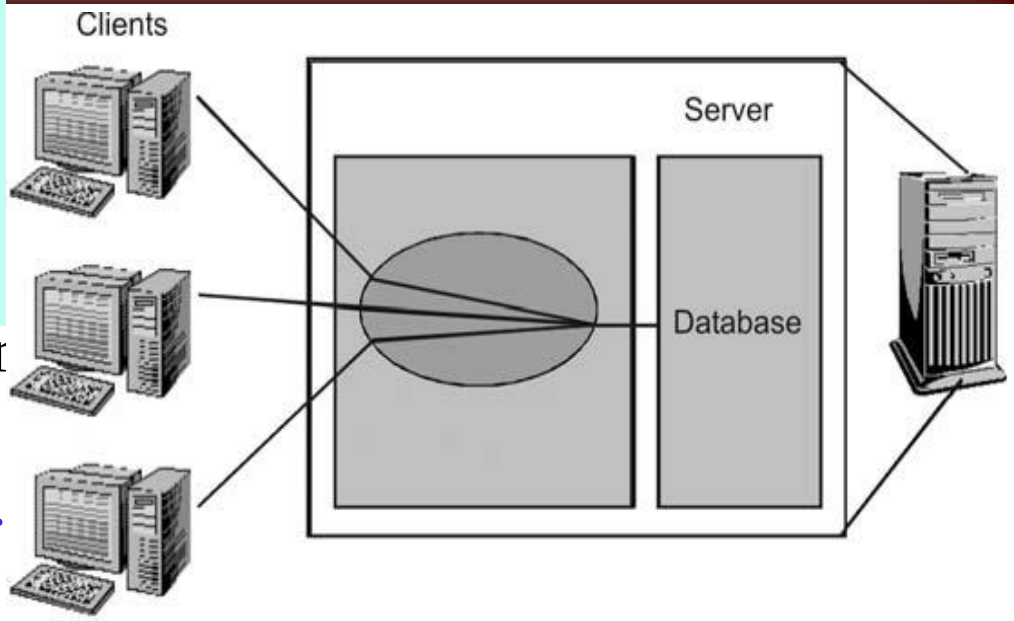


1. ***Process-per-client*** - Server proces smatra svaki klijent kao poseban proces i obezbeđuje poseban adresni prostor za svakog korisnika baze podataka.

- ✓ Svaki proces ima svoj zaseban CPU.
- ✓ Kao rezultat toga, troši se više memorije i CPU resursa.
- ✓ Zbog promene konteksta procesa i zbog interprocesne komunikacije ova arhtektura je znatno sporija od ostalih.
- ✓ Naročito slabi rezultati se postižu kada imamo veliki broj klijenata
- ✓ Sa druge strane ova arhitektura pruža najbolju zaštitu baze podataka.
- ✓ Primeri takve arhitekture su **DB2**, **Informix**, **Oracle6**.

7.1 Client Server DataBase Arhitektura

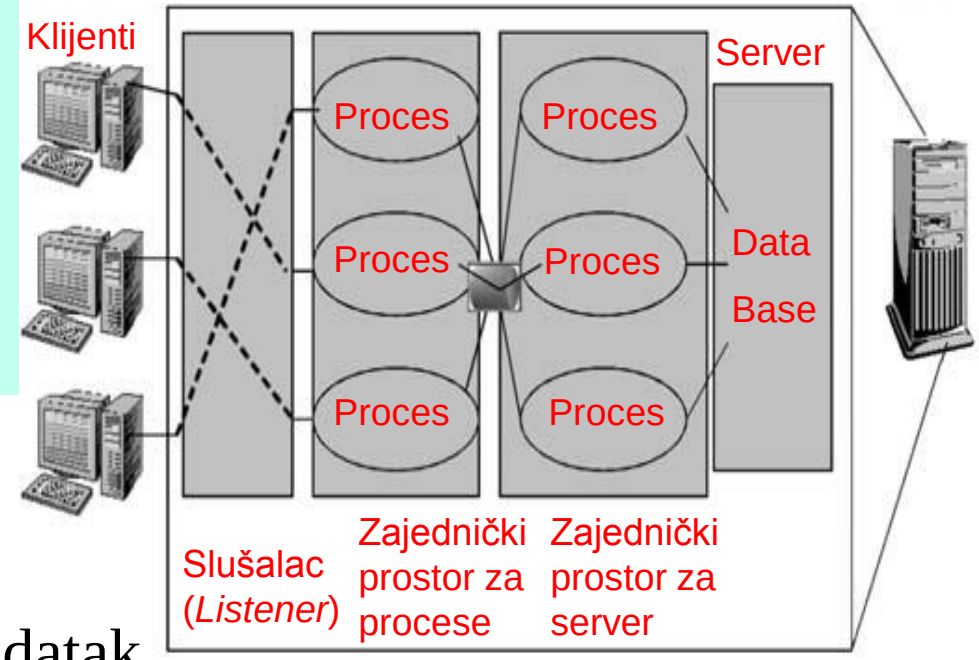
2. ***Multi-threaded*** arhitektura
   Podržava veliki broj klijenata koji zahtevaju jako kratke transakcije na bazi podataka.



- ✓Ima znatno bolje performanse jer pokreće sve zahteve korisnika u jedinstvenom adresnom prostoru.
- ✓Ne ostvaruje dobre rezultate kod zahteva za većim brojem podataka.
- ✓Štedi memoriju i CPU resurse jer nema čestih promena konteksta
- ✓Povoljna je za prenošenje na različite platforme.
- ✓Javlja se problem jer pojedini procesi mogu da obore sve ostale procese koji se izvršavaju od različitih klijenata
- ✓Ukoliko zahtev nekog klijenta traje i suviše dugo, tj. zahteva više resursa, mogu se javiti kašnjenja kod ostalih klijenata
- ✓ Sa bezbedonosne tačke gledišta ovo nije dobar izbor arhitekture
- ✓Tipični predstavnici su: **Sybase i Microsoft SQL Server**.

7.1 Client Server DataBase Arhitektura

3. **Hibridni arhitektura**
   omogućava zaštićeno okruženje za izvršavanje korisničkih zahteva bez dodeljivanja stalnog procesa svakom korisniku.



✓ Pruža najbolji balans između servera i klijenta.

Sastoji se od tri komponente:

1. ***Multi-threaded*** slušalac:glavni zadatak je da dodeliti klijentsku vezu **planeru** (**dispečeru**).
2. **Dispečer procesi**: procesi su odgovorni za smeštanje klijent. zahteva u interni red kao i da vrate klijentu odgovor kada on dođe iz baze
3. **Višekratni, zajednički, radni procesi**: odgovorni za izabrani zahtev iz internog reda i njegovo izvršavanje kao i smeštanje dobijenog odgovora u izlazni red.

✓ Problem oko kašnjenja zbog smeštanja zahteva i odgovora u redove

✓ Neki od primera takvih arhitekturi su **Oracle7i** i **Oracle8i/9i**.

7.2 Data Base Managment System

- U centru svih klijent server aplikacija koje se bave bazama podataka nalazi se **DBMS** (***Data Base Managment System***).
- DBMS je skup komponenata za definisanje, izradu i korišćenje baze koji treba da omogući skladištenje,pretraživanje i upravljanje podacima
- Kada pomenemo sistem za upravljanje bazom podataka (DBMS), uglavnom mislimo na realcioni **DBMS** tj. **RDBMS**.
- Uslovi koje treba da zadovolji DBMS da bi klijent server radio:
  - ✓ Potreban je transparetni pristup podacima za različite klijente bez obzira na hardver, softver i mrežnu platformu koju koristi klijent.
  - ✓ Da se dozvoli da klijent može da šalje svoje zahteve prema serveru baze podataka (uključujući i SQL zahteve) kroz mrežu.
  - ✓ Svi zahtevi klijenta za podacima se obrađuju u lokalnom serveru.
  - ✓ Da se klijentu šalju samo podaci koji se dobijaju kao rezultat obrade njegovog zahteva.
- Uloga DBMS je da oslobodi klijenta od lokalne obrade podataka
- Redukuje se mrežni saobraćaj tj. ubrzava se rad sa traženim podacima.
- Klijentu se vraćaju samo oni podaci koji odgovaraju traženom zahtevu.

7.2 Data Base Managment System

- Klijent server sistem je potpuno promenio način na koji mi pristupamo i obrađujemo podatke u bazi podataka.
- Podaci mogu biti smešteni ne samo na jednoj lokaciji – serveru, već na više servera koji mogu biti na potpuno lokacijski različitim mestima.
- Ako su podaci smešteni na više mesta tada kažemo da se radi o distribuiranim klijent server bazama podataka.

Takve baze podataka karakteriše:

- ✓ Lokacija podataka je potpuno transparentna za korisnika - podaci mogu biti smešteni na lokalnom PC-ju, serveru odeljenja ili na velikim računarima širom zemlje. DBMS mora biti sposoban da upravlja distribucijom podataka između različitih čvorova.
- ✓ Podaci treba da budu pristupačni-njima se može manipulisati od strane korisnika u bilo koje vreme i više puta. Korisnik može manipulisati podacima na različite načine u zavisnosti od njegovih potreba. Moguće da iste podatke dobijamo u različitim grafičkim prikazima.
- ✓ Obrada podataka - (smeštanje, ispravnost, formatiranje, prezentacija) je distribuirana između različitih računara koje klijent vidi kao jedan.

7.2 Komponente klasičnog DBMS-a

1. ***Interfejs za aplikacije*** - predstavlja standardne biblioteke koje pomažu u komunikaciji sa DBMS-om. Većina DBMS-ova ima svoj jednostavni prevodilac u obliku komandne linije koji često koristi biblioteke za prenošenje zahteva unetih sa tastature do DBMS-a
2. ***Interpretator SQL koda*** - predstavlja sintaksni analizator koji proverava sintaksu prosleđenih upita i prevodi ih u interni prikaz.
3. ***Analizator upita*** - generiše razne planove izvršavanja upita na osnovu statistike baze podataka i njenih svojstava, vrši izbor jednog plana i prevodi ga u akcije koje izvršava na niskom nivou.
4. ***Pristup podacima*** - obuhvata module koji upravljaju pristupom podacima uskladištenim na disku. Tu spadaju:
   I. modul za upravljanje transakcijama,
   II. modul za upravljanje postupkom obnavljanja stanja,
   III. modul za upravljanjem baferom glavne memorije,
   IV. modul za zaštitu podataka
   V. modul za upravljanje pristupom datotekama.
5. ***Baza podataka*** podrazumeva same fizičke podatke koji su razvrstani u različitim tabelama kao i indeksne i statističke datoteke.

7.3 Alati za upite (Query Tools)

- Klijent-server arhitektura i alati vezani za takvu arhitekturu uticali su na razvoj korisnički orijentisanih alata za upite i izveštavanja
- Osiguravali su da svako može da napravi jednostavne upite i izveštaje
- Informatički specijalisti postaju nužni kako bi optimizirali performanse baze podataka, alocirali podatke i njihovu obradu između platformi, te kreirali i testirali složenu logiku obrade podataka.

Od savremenih alata za upite i izveštavanja očekuje se da osiguravaju:

- ✓ Pristup velikom broju redova baze podataka;
- ✓ Prenos velikog broja podataka iz baze na radne stanice korisnika;
- ✓ Složene SQL iskaze uključujući povezivanje više tabela i podupita;
- ✓ Složenu logiku izveštavanja kao grupnih postupaka koji uključuju veliki broj redova.

- Navedeni zahtevi utiču na to da se arhitektura naprednih alata za upite i izveštavanja sastoji od više slojeva:

1. Krajnji korisnici pokreću izveštavanja sa svojih radnih stanica.
2. Izveštavanja se smeštaju i pozivaju sa drugog sloja
3. Serveri baze podataka čine treći sloj.

7.3 Alati za upite (Query Tools)

- Arhitektura omogućava upitima i izveštajima da se efikasno izvršavaju nad velikim bazama podataka i da mogu vratiti velike količine podataka
- Činjenica je da alati za upite i izveštavanja postaju sve sofisticiraniji, kako se javlja potreba za pristupom srazmerno sve većim bazama
- Izbor alata za izveštavanje više nije tako jednostavan proces, pogotovo otkada performanse baze podataka nisu jedini kriterijum
- Potrebno je poznavati korisničke zahteve za pristupom podacima, strukture baze podataka, LAN/WAN komunikacijske mogućnosti i slično, kako bi se moglo odabrati odgovarajuće rešenje:

1. Alati za krajnje korisnike koji omogućavaju da baza podataka upravlja optimizacijom izvršenja upita su najbolji za izbor **malog broja redova** baziranog na jednostavnoj logici iz jedne ili dve ogromne tabele.
2. Prilagodljivi serveri za upite i izveštavanja su potrebni za odabir **srednjeg do ogromnog broja redova** bazirano na složenoj logici.
3. Server orijentisani proizvodni alati za upite i izveštavanja sa mogućnostima programiranja, pamćenja i optimiziranja upita, potrebni su za odabir **velikog broja redova** iz nekoliko različitih tabela

7.3 Alati za upite (Query Tools)

- Zahvaljujući metapodacima alati za upite i izveštavanja ukidaju potrebu da korisnici razumeju dizajn baze podataka i postavljaju relacije neophodne za povezivanje tabela kod upita
- Razvoj robusnijih alata za upite i izveštavanja napravio je značajan korak ka smanjenju, ali ne i eliminisanju, unapred definisanih izveštaja
- Ti alati su zamenili jedan način unapred definisanih upita, drugim.
- Tamo gde savremeni alati za upite i izveštavanja pomažu poslovnim korisnicima u formulisanju njihovih vlastitih upita, oni to mogu uraditi tako precizno zahvaljujući složenim SQL naredbama.
- Alati za upite i izveštavanja u suštini ne podržavaju stvarne *ad hoc* korisničke upite, tj. upite formirane "od nule" direktno na podacima
- Pored značajnog napretka u razvoju alata, oni su još uvek dostupni samo u manje složenim delovima sistema za podršku odlučivanju.
- Kako je za njih karakterističan nedostatak sofisticiranosti potrebne za složeno okruženje za potporu odlučivanju, u takvom okruženju oni se mogu upotrebljavati samo u kombinaciji sa drugim alatima koji omogućuju i daju podršku tom odlučivanju.

7.3 Alati za upite (Query Tools)

Osnovni nedostaci alata za upite i izveštavanja ogledaju se u sledećem:

1. Stalno učešće informatičara u procesu izrade i održavanja;
2. Poteškoće u ispunjavanju izazova vezanih za složena poslovna pitanja;
3. Nesposobnost podrške velikim upitima;
4. Nedostatak koncepta vremena, konsolidiranja i agregiranja;
5. Nemogućnost transparentne potpore i ujednačavanja različitih tehnika optimiziranja baza podataka;
6. Slaba povezanost između upita i izveštavanja;
7. Dizajn tipa "debeli" klijent.

- Sva ova navedena ograničenja, kao i ukupan razvoj alata za upite i izveštavanja, koji je primarno bio vezan za transakcijske a ne za sisteme za podršku odlučivanju, dovela su do toga da su ti alati ostali marginalni sa aspekta podrške odlučivanju
- Zato su za potrebe podrške odlučivanju razvijali mnogo sofisticiraniji alati bazirani isključivo na potrebama tih sistema.

7.4 SQL jezik

- Skoro svi sistemi relacionih baza podržavaju SQL kao alat za izradu, zaštitu i pretraživanje baze podataka, kao i za upravljanje njome.
- SQL predstavlja mnogo više od običnog jezika za upite; to je u potpunosti usavršena alatka za sve aspekte održavanje jedne baze

SQL se sastoji iz četiri glavna dela i to:

1. ***Jezik za definisanje podataka*** ***(Data Definition Language)*** predstavlja skup SQL komandi koje formiraju i brišu bazu podataka, dodaju ili uklanjaju tabele, formiraju indekse i unose izmene.
2. ***Jezik za manipulisanjem podacima*** ***(Data Management Language)*** je skup komandi koje rade sa DBMS-om i bazom podataka. To su komande za pretraživanje, umetanje i brisanje podataka u tabelama
3. ***Upravljanje transakcijama*** podrazumeva da SQL sadrži komande za označavanje grupe komandi kao celine ili transakcije.
4. ***Napredne mogućnosti*** DML i DDL omogućavaju da se SQL naredbe mogu ubaciti i u programske jezike opšte namene i definisanje prikaza postojećih podataka za specijalne namene, dodeljivanje i oduzimanje prava pristupa DBMS-u i bazi podataka.

7.5 Vrste povezivanja sa DBMS

➢ **Veze između klijenata i servera** koje nam omogućuju da možemo da pristupamo različitim bazama podataka pod različitim DBMS-ovima:

- **ODBC** (*Open DataBase Connectivity*) predstavlja najstariji Microsoft-ov standard za povezivanje aplikacionog softvera sa različitim relacionim bazama podataka koji je razvijen 1992 od SQL Access Group. Stariji COM interfejsi kao što su **DAO** (*Data Access Objects*) i **RDO** (*Remote Data Objects*) zasnivali su se upravo na modelu ODBC preko koga su pristupali određenoj bazi podataka.
- **OLE DB** Microsoft standard koji definiše jedinstveno podržavanje aplikacija koje pristupaju relacionim i nerelacionim bazama podataka. Postoje različiti OLE DB interfejsi za svaki različiti izvor podataka.
- **ADO** (*ActiveX Data Objects*) predstavlja Microsftov COM bazirani objektni modul koji omogućava da upravljamo OLE DB izvorima podataka
- **JDBC** (*Java Database Connectivity*) je biblioteka klasa koja se koristi za povezivanje na različite relacione baze iz Java koda.

7.5.1 ODBC(*Open DataBase Connectivity*)

- Pristupa mu se preko ***ODBC** Data Source* administratora.
- On dodaje **ODBC drajvere** i konfiguriše da rade sa određenom bazom
- Sloj koji apstrahuje pristup bazama podataka
- Nezavisan je od **programskog jezika**, **baze podataka** i **OS**
- Omogućuje razvoj aplikacija na programskom jeziku C
- Pristup odgovarajućoj bazi podataka se ostvaruje instaliranjem odgovarajućeg ODBC drajvera
- Aplikacija sa bazom komunicira isključivo kroz interfejs ODBC-a
- ODBC nudi interfejs pristupa bazama podataka preko SQL upita
- ODBC "preuređuje" upite pre nego što ih izda na izvršavanje konkretnom drajveru baze
- ODBC je predstavljao niski nivo API-a i zahtevao je od programera da vode računa o memorijskoj raspodeli i dodeli.
- To je otežavalo rad jer je optimizacija baze podataka bila otežana.
- Postoji preko **100** različitih ODBC drajvera za različite soft. platforme
- **Neke mane** ODBC-a su **stabilnost drajvera** i njegove **performanse**

7.5.2 OLE DB (Object Linking and Embedding)

- Postoji potreba za čuvanje pojedinih podataka izvan baze podataka
- Ne postoji jedinstven način za pristup ovakvim podacima tj. nije moguće vršiti SQL upite nad ovakvim izvorima podataka
- Potrebno je omogućiti jedinstven način za pristup svim podacima
- OLE DB je deo grupe Microsoft tehnologija koje se nazivaju ***Microsoft Data Access Components*** (**MDAC**)
- Uvedena je podrška za pristup nerelacionim izvorima podataka koji, u opštem slučaju, ne podržavaju SQL upite.
- Baziran je na COM modelu pa predstavlja skup interfejsa koji su implementirani korišćenjem COM tehnologije
- Predstavlja zamenu i naslednika ODBC-a koji je znatno unapredio
- Proširio je skup funkcionalnosti za potrebe rada sa objektnim bazama i izvorima podataka koji ne podržavaju SQL jezik.
- U OLE DB arhitekturi aplikacija koja pristupa podacima zove se **korisnik podataka** (na primer Excel), a program koji omogućava osnovni pristup podacima zove se **dobavljač baze podataka** (na primer Microsoft OLE DB dobavljač za SQL Server).

7.5.3 ADO (ActiveX DataAccess Objects)

- Skup COM objekata za pristup izvorima podataka koji mogu da budu baze podataka, tekstualni fajlovi, excel dokumenti, XML fajlovi
- To je sloj koji je iznad OLE DB i koji čine **kolekcije i objekti**
- Predstavlja skup klasa za rad sa podacima koje su značajno unapredili rad sa bazama podataka pa se smatra i potpuno novim proizvodom.
- Objektno orijentisan skup biblioteka koje omogućavaju interakciju sa različitim izvorima podataka (*data sources*)
- Osnovni snabdevači su SQL Server i OLE DB snabdevači podataka.
- Postoje dva osnovna načina rada: **konektovani** i **diskonektovani**.
- Kod konektovanog scenarija resursi se uzimaju sa servera sve dok se konekcija ne zatvori i korisnik je konstantno povezan na izvor podataka
- U diskonektovanom scenariju podskup podataka iz baze podataka se kopira na lokalni računar gde se radi dalja obrada tih podataka.
- Dok se korisnik nalazi u diskonektovanom radu ostali korisnici mogu da koriste konekciju.
- Diskonektovani rad povećava skalabilnost aplikacije ali podaci nisu stalno ažurni za razliku od konektovanog gde su podaci stalno ažurni.

7.5.3 ADO Net arhitektura

- Pristup podacima se oslanja na dve komponente:
  1. ***DataSet***: lokalna kopija podataka iz baze podataka, u memoriji ne zahteva direktnu povezanost za bazom podataka, omogućava i perzistenciju podataka
  2. ***DataProvider***: upravlja vezama ka bazi podataka (*connections*), izvršava SQL upite, za bilo koji izvor podataka moguće je implementirati odgovarajući DataProvider:

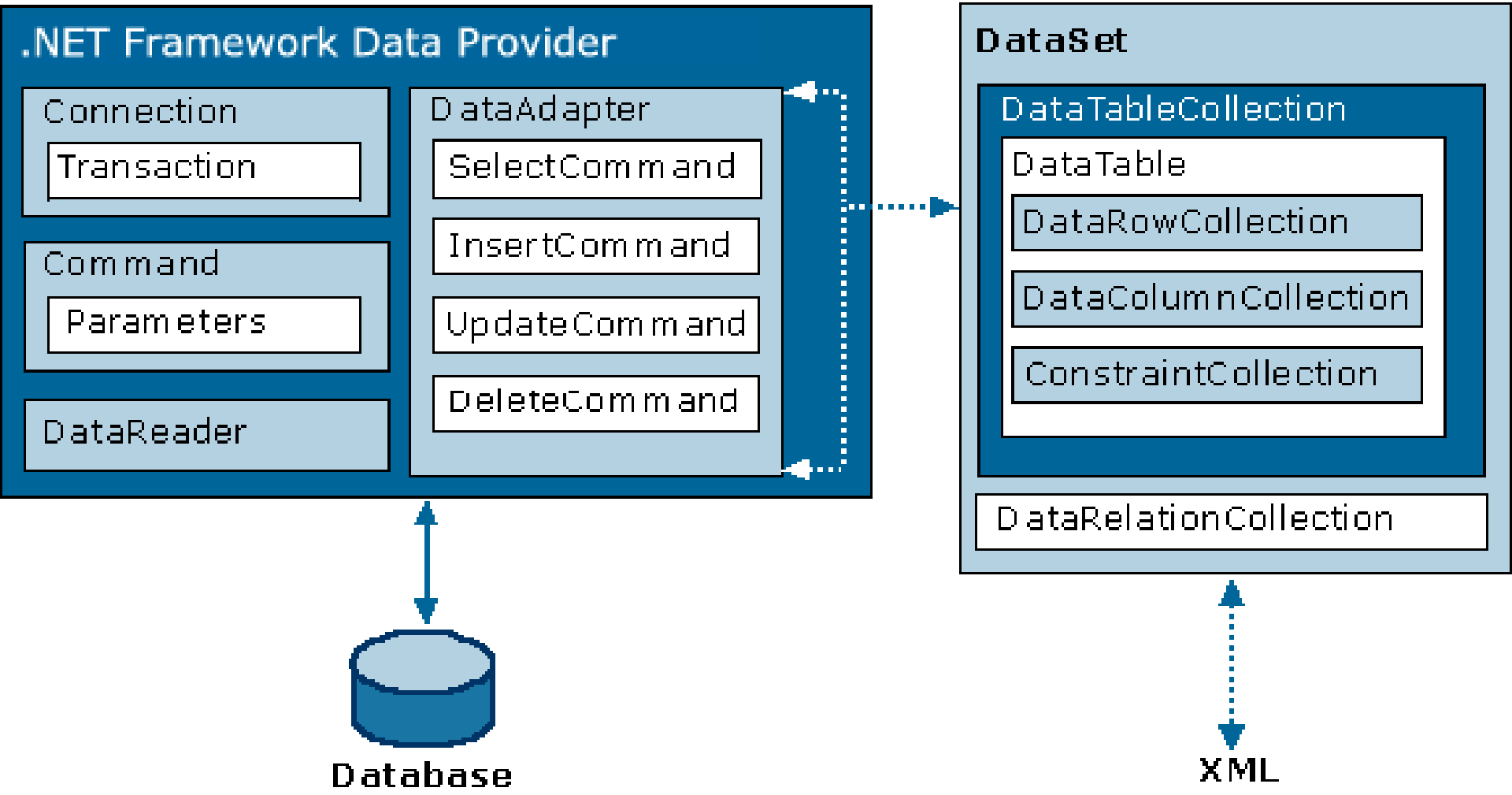
     I. Bridge provajderi – služe za pristup OLE DB i ODBC izvorima podataka kao i omogućavaju korišćenje biblioteka dizajniranih za ranije tehnologije pristupa podacima

     Primeri: OLE DB.NET Data Provider i ODBC.NET Data Provider

     II Native provajderi - jedan nivo apstrakcije manje, donose poboljšanja u pogledu performansi i specijalno su napisani za konkretnu bazu podataka

     Primeri: SQL Server.NET Data Provider i Oracle.NET Data Provider

7.5.3 ADO Net arhitektura



7.5.4 JDBC API

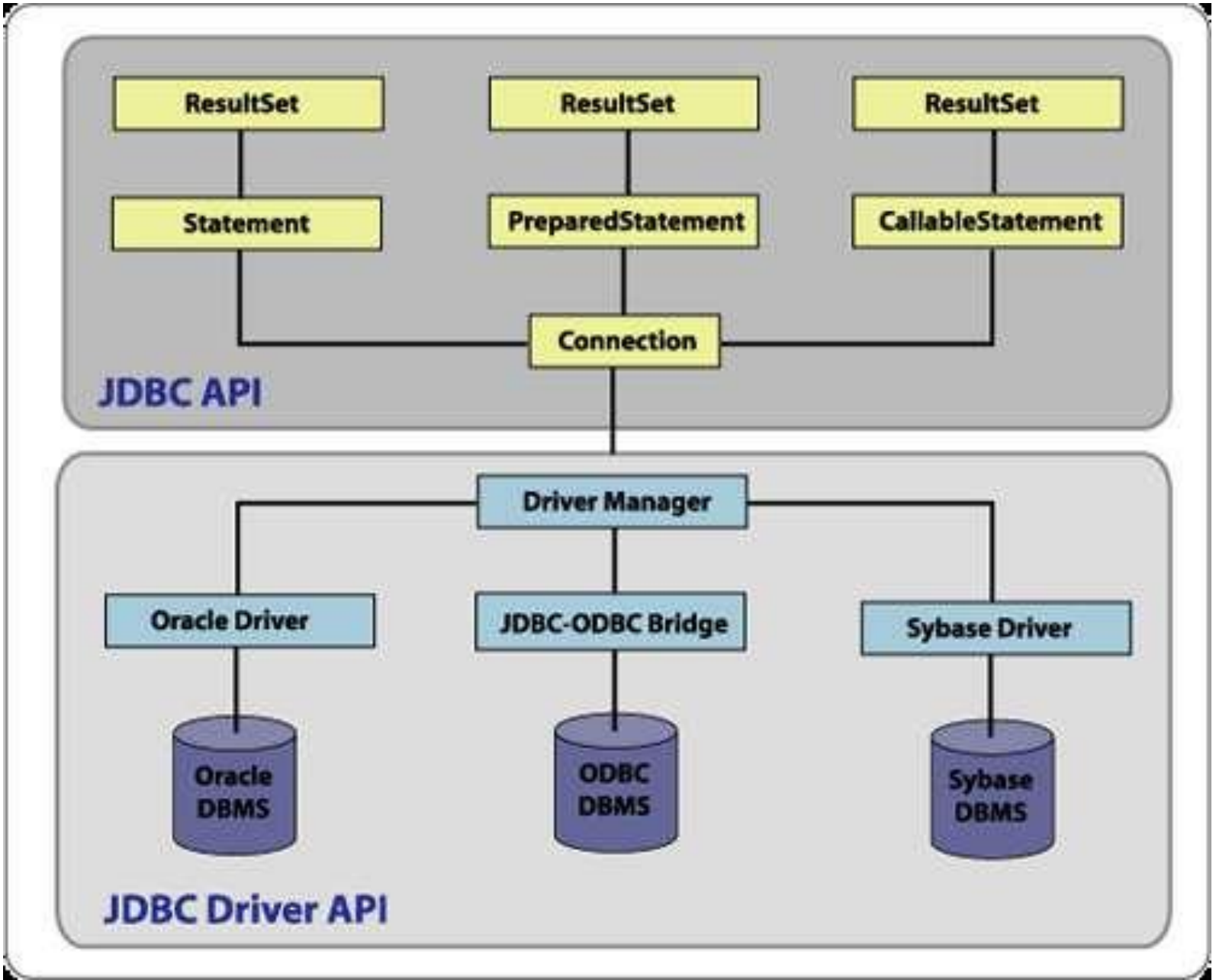
- Uloga Interneta u svakodnevnom životu postaje sve značajnija
- Java programski jezik koji je nezavistan od platforme, omogućuje lako pravljenje Java aplikacija za rukovanje bazama podataka na Internetu
- ***Java Database Connectivity*** (***JDBC***) je biblioteka klasa koja se koristi za povezivanje na različite relacione baze iz Java koda i obuhvata skup funkcija (API) za pristup bilo kojoj bazi podataka iz Java programa
- Aplikacije koje rade sa bazama su obično pisane kao tzv. klijent-server aplikacije, koje povezuju korisnika sa onim ko te informacije pruža.
- Glavni problem u radu sa bazama je veliki broj formata koji se koriste, pri čemu svaki od njih ima svoj način čuvanja i pristupanja podacima.
- Postojanje standardnog jezika za postavljanje upita nad podacima (*SQL*) znatno olakšava rad sa relacionim bazama.
- ***JDBC*** podržava ***SQL***, što omogućava da koristite veliki broj različitih formata baza bez potrebe da znate strukturu same baze.
- ***JDBC*** biblioteka sadrži klase za sve uobičajene operacije nad bazama kao što su uspostavljanje veze sa bazom, kreiranje i izvršavanje ***SQL*** naredbe i obrada rezultata.

7.5.4 Arhitektura JDBC sistema

1. **Dvoslojni model** : aplikacija komunicira direktno sa izvorom podataka koji može biti na istom ili na drugom računaru
2. Troslojni model: aplikacija komunicira indirektno sa izvorom podataka jer postoji **međusloj** koji nudi servise kontrole pristupa, filtriranja sadržaja, pojednostavljuje instalaciju i poboljšava performanse aplikac.

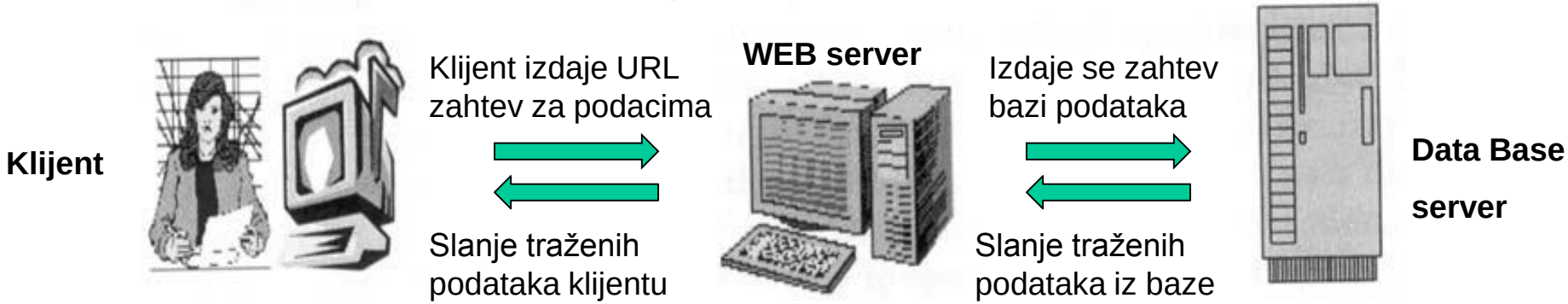
- Java programi koji koriste ***JDBC*** ne moraju da vode računa o formatu baze kojoj pristupaju, kao ni o platformi na kojoj je ta baza kreirana.
- Ovu nezavisnost od baze i platforme omogućuju **drajver menadžeri**.
- Oni vode računa o drajverima neophodnim za pristupanje slogovima baze, i sve klase ***JDBC***-a se oslanjaju na njih.
- Za svaki format baze potrebno je koristiti i odgovarajući **drajver**.
- ***JDBC*** drajveri baze mogu biti napisani u Javi ili implementirani koristeći neke druge metode kako bi se povezala Java aplikacija.
- ***JDBC*** takođe uključuje i drajver koji premošćava ***JDBC*** i ***ODBC***.
- Pristup različitim bazama podataka je potpuno transparentim za programera i aplikaciju jer su aplikacije i programi nezavsni od platforme (Java) i nezavisni od baze podataka (JDBC API)

7.5.4 Vrste povezivanja sa DBMS



7.6 WEB/DataBase sistem kod klijent servera

- Danas, gotovo sve informacije dobijamo *online*, putem Interneta
- Na ovaj način, pronalaženje informacija postaje brzo i lakše.
- Očigledno je da veliki deo toga dobijamo putem mnogih Web stranica
- Linkovi na početnoj stranici obezbeđuju tkz. **Korporativni Intranet**, koji omogućuje veliki broj različitih informacija iz organizacije
- Da bi se obezbedio pristup velikim bazama podataka potrebno je da se Web servisi, putem sopstvenog DBMS, povežu sa bazom podataka



- Klijent koji radi kao Web pretraživač izdaje zahtev za dobijanje informacija u obliku URL (*Uniform Resource Locator*) referencu.
- Ova referenca pokreće program na Web serveru, koji sada izdaje ispravnu komandu bazi podataka i upućuje je ka serveru baze podataka.
- Podaci se vraćaju Web serveru, konvertuju se u HTML i šalju klijentu

7.6 Prednosti WEB/DataBase sistema

- ✓ **Administracija**: jedina veza sa bazom podataka je Web server. Svako dodavanje novog tipa servera baze podataka ne zahteva konfiguraciju novih drajvera i interfejsa za svakog klijenta već samo na Web serveru.
- ✓ **Razvoj**: Web pretraživači su dostupni skoro na svim platformama. Oni oslobađaju programere aplikacija da implementiraju grafički korisnički interfejs za različite računare i OS. Većina Web pretraživača podržava nove Web servise što oslobađa klijente od instalacije i sinhronizacije.
- ✓ **Brzina**: Veliki deo normalnog razvojnog ciklusa, kao što su razvoj i dizajn klijenta, ne odnose se na projekte vezane za Web aplikacije.
- ✓ **Prezentacija**: Multimedijalna osobina Web servisa omogućuje prikaz podataka u velikom broju različitih formata koji su najbolji za podatke

Nedostaci *Web/DataBase* sistema u odnosu na tradicionalni pristup:

- **Funkcionalnost**: U poređenju sa funkcionalnošću klasičnog pristupa u grafičkom prikazu podataka tipičan Web pretraživač je ipak ograničen.
- **Operacije**: priroda HTTP je takva da svaka interakcija između pretraživača i servera predstavlja posebnu transakciju koja je nezavisna od prethodnog ili narednog zahteva.Web server ne prati stanja korisnika

Hvala na pažnji !!!

Pitanja

? ? ?