SPECIFIČNOSTI TROSLOJNOG MODELA KLIJENT - SERVER ARHITEKTURE IMPLEMENTIRANE U JAVA OKRUŽENJU

SPECIFIC FEATURES OF THREE-TIER CLIENT-SERVER ARCHITECTURE IMPLEMENTED IN JAVA ENVIRONMENT

Milan Zdravković, Miroslav Trajanović
Mašinski fakultet Univerziteta u Nišu

Sadržaj: Ovaj rad je nastao kao rezultat istraživanja čiji je cilj definisanje uslova koje treba da ispuni troslojna klijent-server arhitektura, u cilju obezbedjivanja potrebnih performansi, pouzdanosti i sigurnosti. Prikazani su principi njene realizacije u JAVA programskom jeziku, kao i specifičnosti nekih komercijalnih rešenja za rad sa distribuiranim bazama podataka.

Abstract: This paper is result of a research which goal is to define requirements are to be fulfilled by three-tier client-server architecture to provide it with strong performance, reliability and security. The principles of its realisation in JAVA programming language and specific features of few commercial solutions for development of distributed database system are shown.

1. UVOD

Stalna potreba za brzom, tačnom i potpunom informacijom je uslovila brzi razvoj distribuiranih baza podataka. Kao infrastruktura za distribuciju informacija, svojim prednostima se nametnula troslojna klijent-server arhitektura. Ustanovljenjem globalne računarske mreže - Interneta i razvojem web tehnologija, naročito pojavom JAVA programskog jezika, maja 1995. godine, programeri su dobili sredstvo za razvoj pouzdanih, sigurnih i fleksibilnih klijent-server sistema.

Integracijom JAVA Enterprise API-ja (Application Programming Interface) u JDK (JAVA Development Kit), u njegovoj 1.1 iteraciji, omogućena je složena komunikacija JAVA apleta ili aplikacija sa bazama podataka. Ovi alati omogućavaju jednostavan i brzi razvoj troslojne klijent-server arhitekture.

U ovom radu je dat uporedni prikaz tradicionalnih i savremenih načina za projektovanje troslojne klijent-server arhitekture, sa osvrtom na neka komercijalna rešenja. Cilj istraživanja je pokušaj da se definišu zahtevi koje treba da ispuni implementacija arhitekture u JAVA okruženju. Istraživanje je motivisano razvojem informacionog sistema biblioteke Mašinskog fakulteta u Nišu [1], [2].

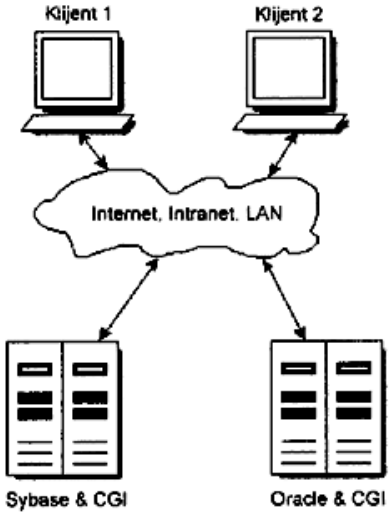
2. TRADICIONALNI NAČINI PROJEKTOVANJA TROSLOJNE KLIJENT-SERVER ARHITEKTURE

2.1 CGI (Common Gateway Interface)

Jedan od najeksploatisanijih puteva za projektovanje sistema za upravljanje distribuiranim bazama podataka je dosad bio korišćenje CGI (Common Gateway Interface) skriptova, odnosno, programa. Programi se izvršavaju na WEB serveru, odakle se, putem LAN mreže posreduje transakcijama sa back-end serverom baza podataka, pri čemu programi dinamički generišu HTML dokumente. U ovim dokumentima se nalaze rezultati transakcije, od značaja za korisnika na klijent strani arhitekture.

Najveći nedostatak ovakve implementacije je činjenica da se prenos podataka u oba smera vrši preko HTTP sesija. Ove sesije traju samo onoliko dugo koliko traje prenos podataka. Za svaku transakciju se moraju otvoriti barem dve HTTP sesije: prva, kojom se vrši transfer instrukcije sa klijent strane ka serveru, i druga, kojom se rezultati transakcije prosledjuju klijentu. Jasno je da u uslovima višekorisničkog rada, često otvaranje i zatvaranje HTTP sesija, značajno usporava rad.

Sa klijent strane je nemoguće dobiti više informacija od onih koji se sadrže u sistemskim promenljivama, jer je na klijent strani obična HTML forma.



sl.2.1 CGI arhitektura

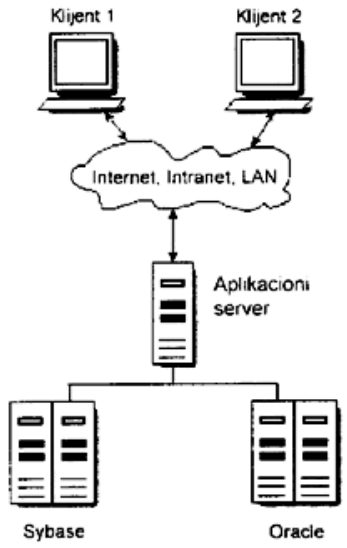
Iako se ovakva arhitektura uslovno može nazvati troslojnom, jer se u CGI program ili skript mogu implementirati poslovna pravila, kao i dodatni sigurnosni sistemi, navedene osobine ga čine skoro neupotrebljivim sredstvom za projektovanje složenog informacionog sistema.

2.2 Aplikacioni server

Primena aplikacionog servera kao srednjeg sloja u troslojnoj klijent-server arhitekturi je omogućila punu implementaciju principa na kojima se ona zasniva. Obezbedjena je mnogo veća sigurnost i integritet podataka. Performanse ovakvog sistema rastu kroz dva aspekta:

- ovakva arhitektura je u potpunosti skalabilna - efikasnom raspodelom resursa na aplikacionom serveru se omogućava višekorisnički rad;
- efikasno korišćenje mrežnih resursa - kao rezultat transakcije se klijentu prosledjuju samo podaci koji su od interesa za korisnika.

Jedna od najvažnijih prednosti arhitekture sa aplikacionim serverom je i mogućnost keširanja ograničene količine podataka. Ovakvi mehanizmi bi značajno smanjili opterećenje back-end servera baza podataka, što bi uslovilo bolje performanse celokupnog sistema. Jedna od vrlo važnih konsekvenci ovakvog rada je i eventualno smanjenje troškova za nabavku softvera na server strani, jer većina proizvodjača servera baza podataka naplaćuje usluge prema broju korisnika.



sl. 2.2. Tradicionalna troslojna klijent-server arhitektura

S druge strane, ovakav sistem karakterišu i neki ozbiljni nedostaci. Oni naročito dolaze do izražaja u radu u heterogenom okruženju, sa više različitih operativnih sistema, back-end servera baza podataka, kao i mrežnih protokola. Naime, u projektovanju ovakvog sistema se mora voditi računa o specifičnostima svakog njenog elementa, što neizbežno vodi ka njegovoj modularnosti, a ova osobina može otežati administraciju i održavanje aplikacionog servera. Za vezu sa svakim serverom se koriste odgovarajuće biblioteke klasa, odnosno, funkcija, a u nekim slučajevima se moraju koristiti i odgovarajući programski jezici.

3. OSOBINE TROSLOJNE KLIJENT-SERVER ARHITEKTURE U JAVA OKRUŽENJU

Izbor rešenja koje obezbedjuje optimalne performanse u ključnim aspektima je kritično za jednostavan razvoj i održavanje distribuiranih baza podataka. Razvoj troslojne klijent-server arhitekture je nametnuo niz zahteva, koji moraju biti ispunjeni:

- *Skalabilnost*

Ova osobina se zahteva u dva pravca: sistem mora da omogućava višekorisnički rad, pri čemu se mogu uzeti u obzir jedino hardverska ograničenja; arhitektura mora biti takva da se može koristiti kao baza za podsisteme različite složenosti, unutar sistema.

- *Podrška za glavne proizvodjače servera baza podataka*

Zahteva se da aplikacija podržava pristup podacima koji se mogu nalaziti u različitim bazama podataka. Obavezna je podrška za Oracle, Sybase, Informix, Watcom i Microsoft SQL Server.

- *Efikasno korišćenje mrežnih resursa*

Mrežni resursi su u najvećem broju slučajeva kritično mesto na kojem opadaju performanse ovakvih sistema. Pravilno izabrana arhitektura ovaj problem može rešiti bez ulaganja u mrežnu infrastrukturu. Potrebno je, takodje, pronaći način da do krajnjeg korisnika dodju samo oni podaci koji su mu stvarno potrebni.

- *Jednostavno održavanje*

Kod tradicionalnog pristupa projektovanja troslojne klijent-server arhitekture, od administratora se zahtevalo da instalira i održava softver na klijent računarima. Očigledno je da, primenom JAVA tehnologija, odnosno korišćenjem apleta kao klijenta u arhitekturi, ovaj problem može biti prevazidjen.

- *Funkcionalnost u Intranet i Internet okruženju*

Uz korišćenje JAVA apleta, za server baze podataka, lokacija klijenta može biti potpuno transparentna.

- *Sigurnost podataka - ograničeni pristup podacima*

U srednjem prstenu, odnosno, server aplikaciji se lako može implementirati dodeljivanje i provera prava pristupa podacima. Uz korišćenje firewall

tehnologija, server se može smatrati skoro potpuno sigurnim od neovlašćene manipulacije podacima.

- *Manji broj konekcija na back-end serveru baza podataka*

Raznim mehanizmima keširanja podataka, implementiranim u srednjem sloju arhitekture, može se smanjiti broj konekcija na serveru baza podataka, čime se smanjuju mrežni saobraćaj i troškovi kod licenciranja servera.

- *Klijent je programiran u JAVI*

Svi, široko rasprostranjeni WEB browseri poseduju JAVA virtuelnu mašinu koja može izvršavati aplete. Ovi browseri su portovani na svim popularnim hardverskim platformama i operativnim sistemima. Uz korišćenje JAR arhiva za transfer database-specific klasa, koje se ne sadrže u JDK, upotreba apleta na klijent strani je u svakom pogledu najbolje rešenje, naročito za primenu u heterogenim sistemima.

- *Implementacija JDBC API-ja*

Korišćenjem JDBC API-ja kao jezgra troslojne arhitekture, omogućava se nivo kompatibilnosti sa odgovarajućim sistemima, kao i potpuna nezavisnost od proizvodjača servera baza podataka, s obzirom na to da su na raspolaganju drajveri za skoro sve poznate servere.

4. SAVREMENA REŠENJA TROSLOJNE KLIJENT-SERVER ARHITEKTURE U JAVA OKRUŽENJU

4.1 Savremena koncepcija troslojne klijent-server arhitekture

Na slici 4.1 je prikazana savremena koncepcija troslojne klijent-server arhitekture. Njene osnovne karakteristike su:

✓ *Thin klijent*

Jedini uslov koji treba da omogući potpunu funkcionalnost klijentu je TCP/IP konfiguracija mrežnog okruženja. Klijent je JAVA aplet koji se izvršava na virtuelnoj mašini WEB pretraživača, koji podržava JDK 1.1 standard. Ovaj aplet mora imati specifikaciju 100% Pure JAVA, iz razloga kompatibilnosti sa svim hardverskim platformama i operativnim sistemima. Klase koje su potrebne za rad funkcija, specifičnih za baze podataka (npr. skrolovanje) se dinamički download-uju sa WEB servera.

✓ *Veza izmedju klijenta i middleware-a baze podataka se ostvaruje preko otvorenog socket-a*

Komunikacija se ostvaruje putem postojeće mrežne infrastrukture, korišćenjem java.net klasa.

✓ *Podrška za pristup većem broju servera baza podataka*

Uslov skalabilnosti sa klijent strane je ispunjen samom koncepcijom klijent-server arhitekture. Middleware mora da poseduje funkcije koje ispunjavaju ovaj uslov i sa strane servera baza podataka. Rad sa više servera treba da bude podržan bibliotekama native funkcija za pristup serverima koje se dobijaju od proizvodjača, ili preko JDBC:ODBC mosta.

✓ *Aplikaciona logika može biti integrisana u middleware-u baza podataka*

U slučaju JDBC:ODBC konekcije sa lokalnim datotekama baza podataka (dBase V, Access), aplikaciona logika (business rules) mora biti implementirana unutar middleware-a. Na ovaj način se štiti integritet podaka u lokalnim datotekama.

✓ *Podržana ANSI SQL92 Entry Level specifikacija*

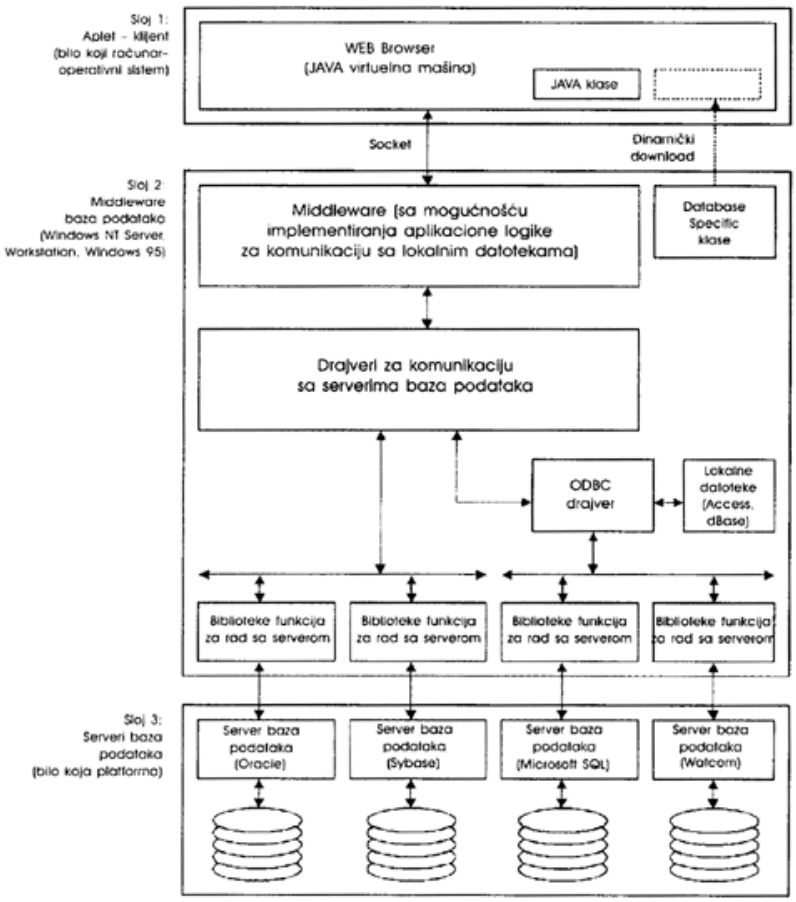
Drajveri moraju podržavati standardni set komandi SQL-a.

✓ *Middleware treba da omogući keširanje podataka*

Raznim mehanizmima keširanja podataka, može se ubrzati rad klijent-server arhitekture. Oni se mogu implementirati na nivou klijenta, čime se štedi i na mrežnim resursima, kao i unutar srednjeg sloja, što smanjuje mrežni saobraćaj ka serverima baza podataka i donosi uštedu u licencnim troškovima.

✓ *Jednostavna administracija middleware-a baza podataka*

Prikazana arhitektura omogućava centralizovano i jednostavno održavanje na nivou sva tri sloja.



slika 4.1 Savremena koncepcija troslojne klijent-server arhitekture

U daljem tekstu je predstavljen kratak prikaz nekih komercijalnih okruženja za razvoj troslojne klijent-server arhitekture. Pri izboru je uzeto obzir prisustvo na tržištu kroz podršku najpoznatijim RAD (Rapid Application Development) alatima za razvoj klijent-server sistema (Symantec Visual Cafe Database Developer Edition i Borland JBuilder Client-Server Edition), kao i neke principijelne specifičnosti.

4.2 Symantec dbAnywhere server

dbAnywhere server je middleware RDBMS firme Symantec. Distribuira se kao samostalan proizvod i integrisan u vizuelno okruženje za razvoj distribuiranih baza podataka Visual Cafe Database Developer Edition.

Ovaj proizvod u potpunosti nasledjuje sve osobine savremene koncepcije troslojne klijent-server arhitekture.

Korišćenjem DataView komponenti klijenti mogu "deliti" rezultate jednog upita, čime se značajno ubrzava rad.

Middleware se integriše u okruženje operativnog sistema Windows NT, tako da se podešavanje, monitoring i planiranje kapaciteta može obavljati i kroz NT-ove administrativne alate (Service Manager i Performance Monitor). Oporavak baza u slučaju narušenja integriteta baze je jednostavan jer se sve transakcije čuvaju u Event Application Log datoteci Windowsa NT.

4.3 InterBase InterServer - InterClient

Ovaj paket se distribuira i kao deo Borland JBuilder Client-Server Edition RAD alata, koji je bez premca medju vizuelnim JAVA alatima. Kao SQL engine se koristi InterBase server, a InterServer je middleware koji kontroliše tok informacija izmedju klijenta i InterBase servera, odnosno servera drugih proizvodjača.

U odnosu na savremenu koncepciju troslojne klijent-server arhitekture, karakteriše se odredjenim specifičnostima:

- Uz klijent se distribuira i InterClient - skup database-specific funkcija koje se samostalno konfigurišu na klijent-računaru, a potom i uklanjaju.
- InterServer je proces koji se izvršava na istom računaru kao i InterBase server.
- Mogu se koristiti klasični SQL iskazi (Statement), parametrizovani SQL iskazi (Prepared Statement) i procedure,

uskladištene na serveru baza podataka (Stored Procedures).

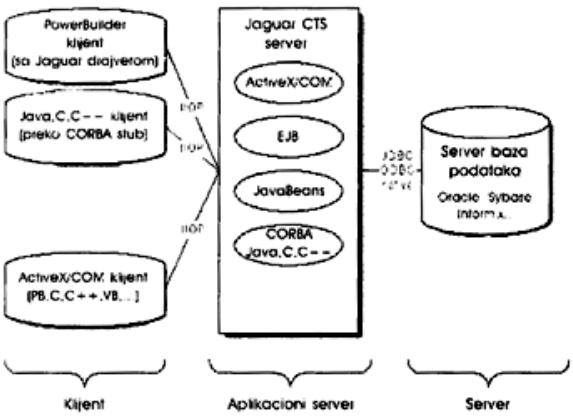
Korišćenjem DataExpress komponenti JBuilder paketa, realizacija klijent-server arhitekture je izuzetno brza, uz podršku za grupni rad, CORBA (Common Object Request Broker Architecture) i RMI (Remote Method Invocation).

4.4 Sybase Enterprise Application Server

Sybase Enterprise Application Server je predstavnik nove generacije komercijalnih aplikacionih servera koji se oslanjaju na CORBA tehnologiju kao snažan objektno-orijentisani temelj za razvoj distribuiranih baza podataka. Sastoji se od PowerDynamo HTTP servera i JaguarCTS servera za distribuiranje aplikacione logike, bazirane na komponentnom modelu.

Korišćenje CORBA tehnologija omogućava interoperativnost svih komponenti u sistemu bez obzira na to u kom su programskom jeziku napisane. Na slici 4.2, predstavljena je šema troslojne klijent-server arhitekture, realizovane uz pomoć CORBA tehnologije. CORBA omogućava integraciju svih elemenata sistema, bez obzira na tehnologiju u kojoj su realizovane, korišćenjem komponentnog modela, koji predstavlja generalizaciju JavaBeans tehnologije na CORBA nivou.

Uz klijent proxy podršku za sve komponentne modele (AciveX/COM, Enterprise JavaBeans, JavaBeans, CORBA), kao sastavni deo Sybase-ovog Enterprise Application Studio paketa, čiji je sastavni deo i EAServer, nalazi se i PowerBuilder, RAD alat za projektovanje klijent aplikacija za rad sa EAServerom, kao i PowerJ - vizuelno okruženje za projektovanje Java apleta ili aplikacija.



sl.4.2 Šema troslojne klijent-server arhitekture realizovane CORBA tehnologijama

Jedan od najvažnijih aspekata, u kome se CORBA razlikuje od već navedenih rešenja je i povećana sigurnost transakcija. Komunikacija klijenata sa aplikacionim serverom se vrši posredstvom IIOP (Internet Inter-ORB Protocol) protokola koji je šifrovan preko SSL (Secure Socket Layer), sa podrškom za autentifikaciju preko X.509 digitalnog sertifikata.

4.5 Servleti

CGI pristup je pre svega karakterističan za dvoslojnu klijent-server arhitekturu. Razlog za to je činjenica da integrisanje aplikacione logike u CGI program ili skript nije ekonomično sa stanovišta performansi. JAVA razvojni tim je ponudio jedno rešenje ovog problema. Počev od 1.1 verzije JDK, kao njen integralni deo je uključen i Servlet API, koji sadrži klase za rukovanje HTTP zahtevima za pokretanje eksternih aplikacija. Za razliku od tipičnih CGI aplikacija, servlet pokreće JAVA virtuelna mašina, i to samo jednom, kada se prvi put pozove. Servlet ostaje rezidentan u memoriji i, zapravo se ponaša kao integralni deo HTTP servera; može jednako efikasno rukovati informacijama statičke i dinamičke prirode i to uz višekorisnički rad, što se omogućava korišćenjem JAVA multithreading klasa.

S druge strane, servleti mogu da rade samo kao ekstenzije HTTP servera koji ih podržavaju. Iako se danas na tržištu može naći puno takvih servera (JAVA Web Server, firme Javasoft; JRun, firme Live Software, itd.) , odnosno, dodataka za poznate WEB servere, ovo se smatra lošom osobinom, jer su za razvoj klijent-server arhitekture potrebne dodatne investicije.

5. ZAKLJUČAK

Pojava JAVA programskog jezika je prouzrokovala trend ubrzanog razvoja tehnologija za projektovanje distribuiranih baza podataka. Uvodjenjem komponentnog modela (JavaBeans), stekli su se uslovi za razvoj upotrebljivih RAD alata. Integrisanjem CORBA tehnologija, omogućena je koegzistencija i interoperativnost u svetu različitih programskih jezika.

S obzirom na to da u ovom trenutku, razvojni timovi raspolažu velikim dijapazonom sredstava za brzu i efikasnu realizaciju troslojne klijent-server arhitekture, napori treba da se preusmere na pažljivo snimanje potreba i analizu postojećih hardverskih i softverskih resursa na čijoj bazi treba izgraditi hipotetički sistem.

Kao rezultat ovog istraživanja treba da proistekne sistem koji će omogućiti da do klijenta, sigurnim putem, u najkraćem roku stigne što valjanija informacija, bilo da se radi o rezultatu upita ili izvršene transakcije.