II

MySQL

8

Projektovanje baze podataka za veb

Pošto smo opisali osnove programiranja na PHP-u, preći ćemo na integrisanje baze podataka u PHP skriptove. U poglavlju 2 naveli smo prednosti upotrebe baza podataka umesto običnih datoteka. Te prednosti su:

- Brži pristup podacima.
- Upotreba upita za jednostavno izdvajanje podataka koji ispunjavaju određene uslove.
- Ugrađeni su mehanizmi za rešavanje problema zbog istovremenog pristupa podacima tako da programer ne mora o tome da vodi računa.
- Obezbeđen nasumičan pristup podacima.
- Ugrađeni su sistemi ovlašćenja za pristup podacima.

Na primer, baza podataka brzo i jednostavno pruža odgovore na upite o tome odakle su kupci, koji proizvodi se najbolje prodaju i koji tipovi kupaca najviše troše. Informacije te vrste mogu vam pomoći da doterate veb lokaciju kako biste privukli što više korisnika, ali se veoma teško izdvajaju iz obične datoteke.

U ovom delu knjige opisaćemo bazu podataka MySQL. Pre nego što u sledećem poglavlju pređemo na specifičnosti MySQL-a, moramo razmotriti sledeće:

- Osnovne pojmove i terminologiju relacionih baza podataka
- Projektovanje baze podataka za veb
- Arhitekturu baze podataka za veb

U narednim poglavljima obrađujemo sledeće teme:

- Poglavlje 9, „Izrada baze podataka za veb“, govori o osnovnoj konfiguraciji potrebnoj za povezivanje MySQL-ove baze podataka na veb. Tu ćete naučiti da pravite korisničke naloge, baze podataka, tabele i indekse, a biće reči i o MySQL-ovim mašinama za baze podataka.
- Poglavlje 10, „Rad sa MySQL-ovom bazom podataka“, objašnjava kako se s komandne linije zadaju upiti, te dodaju, brišu i ažuriraju zapisi.

- Poglavlje 11, „Pristup MySQL bazi podataka s veba pomoću PHP-a“, objašnjava kako se PHP i MySQL povezuju tako da se preko veb interfejsa koristi i administrira baza podataka. Saznaćete da to možete raditi na dva načina: korišćenjem MySQL-ovog izvornog drajvera i korišćenjem interfejsa PDO (PHP Data Objects) nezavisnim od vrste baze podataka (engl. *database agnostic*).
- Poglavlje 12, „Naprednije tehnike administriranja MySQL servera“, detaljnije objašnjava kako se administrira MySQL; između ostalog, opisan je sistem prava i ovlašćenja, zaštita od neovlašćenog pristupa i optimizovanje MySQL-a.
- Poglavlje 13, „Naprednije tehnike programiranja u MySQL-u“, podrobnije opisuje mašine baze podataka – uključujući transakcije, tekstualno pretraživanje i uskladištene procedure.

Koncepti relacionih baza podataka

Relacione baze podataka – koje su najpopularnija vrsta baza podataka – zasnivaju se na teorijama relacione algebre. Da biste ih koristili, nije neophodno da poznajete pomenute teorije (mada ne može da vam škodi), ali treba da shvatite neke osnovne koncepte.

Tabele

Relacione baze podataka sastoje se od relacija (odnosa između podataka), koje se najčešće nazivaju tabelama. Tabela je upravo to što joj ime govori – tabela s podacima. Ako ste koristili program za tabelarna izračunavanja – koristili ste tabelu.

Slika 8.1 je primer tabele koja sadrži imena i adrese kupaca u knjižari Book-O-Rama.

KUPCI

ŠiraKupca	Ime	Adresa	Grad
1	Julija Savić	Banatska 25	Zrenjanin
2	Alen Vojnović	Bačka 47	Subotica
3	Milena Avramović	Raška 357	Kikinda

Slika 8.1 Podaci o kupcima knjižare Book-O-Rama čuvaju se u tabeli.

Tabela ima ime – Kupci (`Customers`), kolone u kojima su smeštene određene vrste podataka i redove koji opisuju pojedinačne kupce.

Kolone

Svaka kolona u tabeli ima jedinstveno ime i sadrži odgovarajući tip podataka. Na primer, u tabeli sa slike 8.1 kolona `CustomerID` (šifra kupca) celobrojna je vrednost, dok ostale kolone sadrže znakovne nizove. Kolone tabele se ponekad nazivaju *polja* ili *atributi* tabele.

Redovi

Svaki red u tabeli predstavlja po jednog kupca. Zbog tabelarnog formata, svaki red ima iste atribute. Redovi tabela se nazivaju i *zapisi* (engl. *records*) ili *torke* (engl. *tuples*).

Vrednosti

Svaki red se sastoji od skupa pojedinačnih vrednosti koje se nalaze u kolonama. Svaka vrednost mora imati tip podataka određen kolonom u kojoj se nalazi.

Ključevi

Neophodno je da postoji način nedvosmislene identifikacije svakog kupca. Imena osoba obično nisu pogodna za tu namenu jer može postojati više kupaca sa istim imenom i/ili prezimenom.

Kupca Juliju Savić iz tabele sa slike 8.1 možete da razlikujete od ostalih na nekoliko načina. Ona je najverovatnije jedina Julija Savić na navedenoj adresi, ali navoditi uvek ime, prezime i punu adresu previše je nezgrapno i previše podseća na otuđeni jezik birokratije. Osim toga, taj oblik zahteva upotrebu više od jedne kolone u tabeli.

Zato smo u ovom primeru (a to ćete verovatno i vi činiti u svojim aplikacijama), dodelili svakom kupcu jedinstven identifikacioni broj ili šifru kupca (`CustomerID`), po istom principu po kojem u banci dobijate jedinstven broj računa. To omogućava da se podaci lakše razlikuju u bazi podataka. Može se garantovati da će veštački dodeljen identifikacioni broj biti jedinstven, što je znatno ređe za stvarne podatke, čak i ako se kombinuju.

Identifikaciona kolona u tabeli naziva se *ključ* (engl. *key*) ili *primarni ključ* (engl. *primary key*). Ključ ne mora da se sastoji samo od jedne kolone. Ako, na primer, odlučimo da Juliju zovemo „Julija Savić, Banatska 25, Zrenjanin“, ključ će se sastojati od kolona Ime, Adresa i Grad, bez garancije da će biti jedinstven.

Baze podataka se obično sastoje od više tabela za čije povezivanje se koriste ključevi. Na slici 8.2, dodali smo bazi podataka još jednu tabelu. Druga tabela – Porudžbine (`Orders`) – sadrži porudžbine kupaca. Svaki red u tabeli `Orders` predstavlja jednu porudžbinu jednog kupca. Zahvaljujući šifri kupca, `CustomerID`, znamo čija je svaka porudžbina. Na primer, porudžbina broj 2 pripada kupcu čija je šifra 1. Ako pogledate tabelu `Customers`, videćete da je to Julija Savić.

KUPCI

BrojKupca	Ime	Adresa	Grad
1	Julija Savić	Banatska 25	Zrenjanin
2	Alen Vojnović	Bačka 47	Subotica
3	Milena Avramović	Raška 357	Kikinda

PORUDŽBINE

BrojPorudžbine	BrojKupca	Iznos	Datum
1	3	27,50	02-Apr-2007
2	1	12,99	15-Apr-2007
3	2	74,00	19-Apr-2007
4	3	6,99	01-Maj-2007

Slika 8.2 Svaka porudžbina u tabeli Porudžbine (Orders) povezana je sa određenim kupcem u tabeli Kupci (Customers).

U relacionoj bazi podataka, za takvu vrstu relacija koristi se izraz *spoljni ključ* (engl. *foreign key*). Kolona `CustomerID` primarni je ključ u tabeli `Customers`, ali kada postoji i u nekoj drugoj tabeli, kao što je tabela `Orders`, u toj tabeli je spoljni ključ.

Možda se pitate zbog čega smo odlučili da radimo s dve tabele, umesto da adresu kupca Julije Savić smestimo u tabelu `Orders`. O tome ćemo detaljnije govoriti u sledećem odeljku.

Šema baze podataka

Skup struktura svih tabela u bazi naziva se *šema* (engl. *schema*) baze podataka. Ona predstavlja „nacrt" ili „plan" baze podataka. Šema se sastoji od opisa tabela i njihovih kolona, tipova podataka u tim kolonama, primarnog ključa svake tabele i eventualnih spoljnih ključeva. Šema ne sadrži podatke, ali određene podatke možete prikazati uz šemu kao primere, ako želite da bolje objasnite čemu ti podaci služe. Šema se može prikazati u obliku neformalnih dijagrama koje koristimo u ovoj knjizi, zatim u obliku *dijagrama relacija i entiteta* (koji nije objašnjen u ovoj knjizi) i u tekstualnom obliku, na sledeći način:

```
Customers(CustomerID, Name, Address, City)
Orders(OrderID, CustomerID, Amount, Date)
```

Primarni ključevi su podvučeni, a spoljni su istaknuti kurzivom.

Relacije

Spoljni ključevi predstavljaju relacije (veze) između podataka u dve tabele. Na primer, veza između tabela `Orders` i `Customers` predstavlja odnos ili povezanost između reda u tabeli `Orders` i reda u tabeli `Customers`.

U relacionoj bazi podataka postoje tri osnovne vrste relacija. One se klasifikuju prema broju povezanih redova na svakoj strani relacije. Tipovi relacija su: „jedan prema jedan", „jedan prema više" i „više prema više".

Relacija tipa „jedan prema jedan" povezuje po jedan red u obe tabele. Ako biste, na primer, adrese izdvojili iz tabele `Customers` i stavili ih u zasebnu tabelu, odnos između te dve tabele bio bi „jedan prema jedan". Spoljni ključ bi mogao da povezuje red iz tabele `Adresa` s redom u tabeli `Customers` ili obrnuto (nije obavezno da postoje oba).

U relaciji tipa „jedan prema više", jedan red u jednoj tabeli povezan je s više redova u drugoj tabeli. U ovom primeru, to znači da jedan kupac (iz tabele `Customers`) može da ima više porudžbina (u tabeli `Orders`). U toj vrsti relacije, tabela na strani „više" povezana je preko spoljnog ključa s tabelom na strani „jedan". Na slici 8.2, tu vezu smo prikazali tako što smo kolonu za šifru kupca (`CustomerID`) dodali u tabelu `Customers`.

U relaciji tipa „više prema više", više redova u jednoj tabeli povezano je sa više redova u drugoj. Na primer, ako imate dve tabele, `Knjige` i `Autori`, može se desiti da su neku knjigu napisala dva autora, od kojih je svaki napisao i neke druge knjige – samostalno ili kao koautor. Takav tip veza obično se predstavlja pomoću dodatne tabele, tako da bi postojale tabele `Naslovi`, `Autori` i `Autor_naslov`. Treća tabela, `Autor_naslov`, sadržala bi samo parove povezanih spoljnih ključeva iz prve dve i označavala bi koji su autori napisali koje knjige.

Projektovanje baze podataka za rad na vebu

Nije uvek očigledno kada je potrebna nova tabela i šta bi trebalo da bude njen ključ. Postoji obilje informacija o dijagramima relacija i entiteta kao i o normalizaciji baza podataka – koje nisu tema ove knjige – ali u većini slučajeva biće dovoljno nekoliko osnovnih principa koje ćemo predstaviti. Pogledajmo ih u kontekstu knjižare Book-O-Rama.

Razmotrite objekte iz stvarnog sveta čije modele pravite

Pravljenje baze podataka podrazumeva izradu modela objekata iz stvarnog sveta i njihovih relacija, te prikupljanje podataka o tim objektima i njihovim relacijama.

Obično je za svaku klasu objekata čiji model pravite potrebna po jedna tabela. Razmislite o sledećem problemu: želite da evidentirate iste podatke o svim kupcima. Ako imate skup podataka koji čini određeni „oblik“, lako možete napraviti tabelu koja odgovara tim podacima.

U našem primeru, u knjižari Book-O-Rama, želimo da evidentiramo podatke o kupcima, knjigama i porudžbinama. Svaki kupac ima ime i adresu. Svaka porudžbina ima datum, ukupan iznos i naslove naručenih knjiga. Svaka knjiga ima jedinstven ISBN broj, autora, naslov i cenu.

Na osnovu navedenih podataka može se zaključiti da baza podataka treba da ima tri tabele: `Kupci (Customers)`, `Porudžbine (Orders)` i `Knjige (Books)`. Početna šema prikazana je na slici 8.3.

KUPCI

BrojKupca	Ime	Adresa	Grad
1	Julija Savić	Banatska 25	Zrenjanin
2	Alen Vojnović	Bačka 47	Subotica
3	Milena Avramović	Raška 357	Kikinda

PORUDŽBINE

BrojPorudžbine	BrojKupca	Iznos	Datum
1	3	27,50	02-Apr-2007
2	1	12,99	15-Apr-2007
3	2	74,00	19-Apr-2007
4	3	6,99	01-Maj-2007

KNJIGE

ISBN	Ime autora	Naslov	Cena
0-672-31697-8	Michael Morgan	Java 2 for Professional Developers	34,99
0-672-31745-1	Thomas Down	Installing GNU/Linux	24,99
0-672-31509-2	Pruitt.et al.	Teach Yourself GIMP in 24 Hours	24,99

Slika 8.3 Početna šema se sastoji od tabela Kupci (`Customers`), Porudžbine (`Orders`) i Knjige (`Books`).

Za sada se na osnovu modela ne može reći koje knjige pripadaju kojoj porudžbini. To ćemo rešiti kasnije.

Izbegavajte suvišne podatke

U ovom poglavlju smo se već jednom zapitali: zašto adresu Julije Savić ne bismo čuvali u tabeli `Orders`?

Ako Julija kupi knjige više puta (čemu se nadamo), svaki put bismo morali da ponovo upišemo njene podatke, pa bi tabela `Orders` mogla da izgleda kao na slici 8.4.

BrojKupca	Iznos	Datum	BrojKupca	Ime	Adresa	Grad
12	199,50	25-Apr-2007	1	Julija Savić	Banatska 25	Zrenjanin
13	43,00	29-Apr-2007	1	Julija Savić	Banatska 25	Zrenjanin
14	15,99	30-Apr-2007	1	Julija Savić	Banatska 25	Zrenjanin
15	23,75	01-Maj-2007	1	Julija Savić	Banatska 25	Zrenjanin

Slika 8.4 Baza podataka u kojoj se čuvaju suvišni podaci zauzima više prostora i može da bude uzrok neusklađenosti podataka.

Takvo rešenje prouzrokuje dva osnovna problema:

- Prostor se nepotrebno rasipa. Zašto nečije podatke upisivati triput ako je dovoljno samo jednom?
- Mogu se pojaviti *neusklađenosti pri ažuriranju* (engl. *update anomalies*) – izmene podataka koje mogu dovesti do njihove neusklađenosti. Narušava se integritet podataka i više se ne zna koji su podaci tačni a koji nisu. To obično dovodi do gubljenja podataka.

Treba izbegavati tri vrste neusklađenosti: pri izmenama, unošenju i brisanju podataka.

Ako se Julija preseli na novu adresu pre nego što joj dostavite knjige koje je kupila, njenu novu adresu treba uneti na tri mesta umesto na jedno, što znači triput više posla. Tu činjenicu lako možete prevideti i uneti njenu novu adresu na samo jedno mesto, usled čega podaci u bazi postaju neusklađeni (što je veoma loše). Ta vrsta problema zove se *neusklađenost izmena* (engl. *modification anomalies*) zato što nastaju pri pokušaju izmene baze podataka.

Pošto vas sadašnje rešenje prisiljava da podatke o Juliji ponovo unosite kad god dobijete novu porudžbinu od nje, svaki put morate proveriti da li su podaci koje ste uneli u skladu s postojećim redovima u tabeli. Ako to ne proverite, moglo bi se desiti da imate dva reda s različitim podacima o Juliji. Na primer, jedan red bi mogao da pokazuje da Julija živi u Zrenjaninu, a u drugom ime naselja može biti ZrenjaninMelenci. To se zove *neusklađenost pri unošenju* (engl. *insertion anomaly*) zato što nastaje pri unošenju podataka.

Treću vrstu čine *neusklađenosti pri brisanju* (engl. *deletion anomaly*) i nastaju prilikom brisanja redova u bazi podataka. Pretpostavimo da smo kupcu dostavili naručene knjige i da smo nakon toga tu porudžbinu obrisali iz baze podataka, tj. iz tabele `Orders`. To znači da više nemamo adresu kupca Julija i ne možemo da joj šaljemo obaveštenja o sniženjima ili specijalnim ponudama. Osim toga, kada ona sledeći put nešto naruči, moraćemo ponovo da unosimo njenu adresu i ostale podatke.

Zbog svega navedenog, bazu podataka treba projektovati tako da se pomenute neusklađenosti ne pojavljuju.

Kolone treba da sadrže proste (nedeljive) podatke

To znači da svaki atribut u svakom redu treba da sadrži samo podatak koji se ne može dalje razbiti na još prostije činioce (engl. *atomic column values*). Pretpostavimo da treba da utvrdite koje knjige pripadaju kojoj porudžbini. To možete saznati na nekoliko načina.

Jedno rešenje bi bilo da tabeli `Orders` dodate kolonu u kojoj ćete navesti sve naručene knjige, kao na slici 8.5.

PORUDŽBINE

BrojPorudžbine	BrojKupca	Iznos	Datum	Naručene knjige
1	3	27,50	02-Apr-2007	0-672-31697-8
2	1	12,99	15-Apr-2007	0-672-31745-1. 0-672-31509-2
3	2	74,00	19-Apr-2007	0-672-31697-8
4	3	6,99	01-Maj-2007	0-672-31745-1. 0-672-31509-2. 0-672-31697-8

Slika 8.5 U ovakvom idejnom rešenju, atribut `Books Ordered` (Naručene knjige) u svakom redu sadrži više vrednosti.

To rešenje nije dobro iz više razloga. Unutar jedne kolone ugnežđujemo zapravo celu jednu novu tabelu – onu koja povezuje porudžbine s knjigama. Ako kolonu definišete tako da sadrži kombinovane podatke, teško ćete naći odgovor na pitanje tipa „koliko je prodato primeraka knjige *Java 2 for Professional Developers*?“. Sistem više ne može da potraži odgovor tako što će prebrojati polja koja ispunjavaju uslov, već mora pojedinačno da analizira svaku vrednost atributa kako bi utvrdio da li sadrži traženi naslov.

Umesto da pravite tabelu unutar tabele, trebalo bi da napravite novu tabelu. Ta tabela se zove Stavke_porudžbine (`Order_Items`) i prikazana je na slici 8.6.

STAVKE_PORUDŽBINE

BrojPorudžbine	ISBN	Količina
1	0-672-31697-8	1
2	0-672-31745-1	2
2	0-672-31509-2	1
3	0-672-31697-8	1
4	0-672-31745-1	1
4	0-672-31509-2	2
4	0-672-31697-8	1

Slika 8.6 Ovakvo idejno rešenje olakšava utvrđivanje broja prodatih primeraka određenog naslova.

Prikazana tabela povezuje tabele `Orders` i `Books`. To je uobičajen tip tabele kada između dva objekta postoji relacija „više prema više“. U ovom slučaju, jedna porudžbina može da sadrži više knjiga, kao što i više osoba može da naruči istu knjigu.

Kada rešavate problem za koji su zaista potrebne složenije vrednosti u kolonama, trebalo bi da – umesto relacione baze podataka – koristite bazu koja je projektovana baš za takav tip podataka. Takve baze podataka nisu relacione, često se nazivaju NoSQL baze podataka ili *skladišta podataka* (engl. *datastores*) i nisu obrađene u ovoj knjizi.

Izaberite logične ključeve

Vodite računa o tome da ključevi koje ste izabrali budu jedinstveni. U ovom slučaju napravili smo poseban ključ za kupce (`CustomerID`) i za porudžbine (`OrderID`), zato što ti objekti u stvarnosti ne moraju imati identifikatore koji će garantovano biti jedinstveni. Za knjige ne morate praviti veštački jedinstven identifikator jer takav već postoji. To je ISBN broj. Tabeli `Order_Items` možete dodati još jedan ključ – ako želite – ali će kombinacija atributa `OrderID` i `ISBN` biti jedinstvena, pod uslovom da se svi primerci iste knjige u porudžbini upisuju u isti red. Zbog toga tabela `Order_Items` ima kolonu `Količina` (`Quantity`).

Mislite o tome šta hoćete da saznate iz baze podataka

Jedno od takvih pitanja moglo bi da bude: koje se knjige iz knjižare najbolje prodaju? Da bi mogla da odgovori na pitanja, baza podataka mora da sadrži sve neophodne podatke, a između tabela moraju da postoje odgovarajuće veze.

Izbegavajte strukture tabela s mnogo praznih atributa

Kada biste u bazi podataka želeli da evidentirate i prikaze knjiga, to biste mogli uraditi na barem dva načina (slika 8.7).

KNJIGE

ISBN	Ime autora	Naslov	Cena	Prikaz
0-672-31697-8	Michael Morgan	Java 2 for Professional Developers	34,99	
0-672-31745-1	Thomas Down	Installing GNU/Linux	24,99	
0-672-31509-2	Pruitt et al.	Teach Yourself GIMP in 24 Hours	24,99	

PRIKAZI_KNJIGA

ISBN	Prikaz

Slika 8.7 Prikaze knjiga možete uključiti u bazu tako što ćete dodati odgovarajuću kolonu tabeli Knjige (`Books`) ili dodati novu tabelu samo za tu namenu.

Prvo rešenje je da tabeli `Books` dodate kolonu `Prikaz` (`Review`). To podrazumeva da za svaku knjigu treba popuniti i polje `Review`. Ako u bazi postoji mnogo knjiga i ako recenzent ne planira da baš za svaku napiše prikaz, taj atribut neće imati vrednost u mnogim redovima tabele, tj. polje `Review` imaće vrednost *null*.

Nije dobro da baza podataka sadrži mnogo vrednosti Null jer se tako rasipa prostor na disku i nastaju problemi pri izračunavanju ukupnih iznosa i upotrebi drugih funkcija u numeričkim kolonama. Kada u tabeli vidi prazno polje, korisnik ne zna razlog – da li atribut nije relevantan, da li postoji greška u bazi ili podaci još nisu uneti.

Taj problem se najčešće izbegava tako što se koristi drugačija struktura baze podataka. U ovom slučaju, bolje je drugo rešenje sa slike 8.7. Tu su u tabeli Prikazi_knjiga (`Book_Reviews`) navedene samo knjige za koje postoje prikazi i tekstovi prikaza.

Imajte u vidu da to rešenje polazi od pretpostavke da postoji samo jedan recenzent koji je saradnik knjižare, tj. da između knjiga i prikaza postoji veza tipa „jedan prema jedan". Ako želite da više osoba piše prikaze za istu knjigu, to bi bila veza tipa „jedan prema više" i morali biste da se opredelite za drugo rešenje. Osim toga, ako za svaku knjigu može postojati samo po jedan prikaz, ISBN broj možete iskoristiti kao primarni ključ u tabeli `Book_Reviews`. Ukoliko dozvoljavate da postoji više prikaza po knjizi, moraćete da uvedete jedinstven identifikator svake od njih.

Kratak pregled vrsta tabela

Šema baze podataka najčešće se svodi na dve vrste tabela:

- Osnovne tabele, koje predstavljaju stvarne objekte. Te tabele mogu da sadrže i ključeve koji ih povezuju s drugim takvim tabelama, s kojima su u vezama tipa „jedan prema jedan" ili „jedan prema više". Na primer, jedan kupac može da ima više porudžbina, ali jedna porudžbina može da pripada samo jednom kupcu. Zato se u porudžbinu stavlja odrednica koja je povezuje sa odgovarajućim kupcem.
- Spojne tabele (engl. *linking tables*), pomoću kojih se uspostavljaju veze tipa „više prema više" između dve osnovne tabele – kao što je relacija između tabela `Orders` i `Books`. Te tabele često služe za obavljanje raznih vrsta transakcija iz stvarnog sveta.

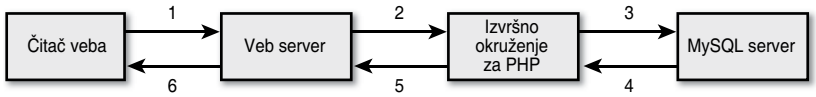
Arhitektura baze podataka za veb

Pošto smo objasnili unutrašnju strukturu baze podataka, preći ćemo na spoljnu arhitekturu sistema baze podataka za veb i reći nešto o metodologiji njegovog razvoja.

Vrlo sažet opis principa rada veb servera prikazan je na slici 8.8. Sistem se sastoji od dva objekta: čitača veba i veb servera, između kojih mora da postoji komunikaciona veza. Čitač veba šalje zahtev serveru, koji potom vraća odgovor. Takva arhitektura sasvim odgovara za server koji isporučuje isključivo statičke stranice. Međutim, arhitektura veb lokacije čiji se sadržaj učitava iz baze podataka, donekle je složenija.

Slika 8.8 Klijentsko-serverski odnos između čitača veba i veb servera zahteva komunikaciju između njih.

Aplikacije baza podataka za veb koje ćemo praviti u ovoj knjizi slede opštu strukturu prikazanu na slici 8.9. Trebalo bi da vam ta struktura bude uglavnom poznata.



Slika 8.9 Osnovna arhitektura baze podataka za veb sastoji se od čitača veba, veb servera, mašine za izvršavanje skriptova i servera baze podataka.

Tipična transakcija u bazi podataka na vebu sastoji se od koraka numerisanih na slici 8.9. Objasnićemo ih na primeru knjižare Book-O-Rama.

1. Korisnikov čitač veba šalje HTTP zahtev za određenu veb stranicu. Na primer, korisnik je HTML obrascem zatražio da vidi sve knjige određenog autora koje se prodaju u knjižari. Stranica koja prikazuje rezultate pretraživanja zove se `results.php`.
2. Veb server prima zahtev za prikazivanje stranice `results.php`, učitava datoteku u kojoj se ona nalazi i prosleđuje sadržaj te datoteke PHP-ovom izvršnom okruženju na obradu.
3. PHP-ovo izvršno okruženje počinje sintaksnu analizu skripta. Skript sadrži naredbe za uspostavljanje veze s bazom podataka i izvršenje upita koji traži sve knjige određenog autora. PHP otvara vezu sa MySQL serverom i šalje mu odgovarajući upit.
4. MySQL server prima upit, obrađuje ga i zatim rezultat – listu traženih knjiga – vraća PHP-ovom izvršnom okruženju.
5. PHP-ovo izvršno okruženje za PHP završava obradu skripta, što obično podrazumeva i formatiranje rezultata upita u HTML kôd. Zatim skript prosleđuje veb serveru.
6. Dobijeni HTML kôd veb server prosleđuje čitaču veba koji je poslao zahtev, što korisniku omogućava da vidi spisak traženih knjiga.

Postupak je u suštini isti, bez obzira na to koju mašinu za izvršavanje skriptova ili server baze podataka koristite. Ponekad, softver za veb server, PHP-ovo izvršno okruženje i server baze podataka rade na istom računaru. Nije neuobičajeno ni da server baze podataka radi na nekom drugom računaru zbog bezbednosnih razloga, većeg kapaciteta i boljeg raspoređivanja opterećenosti. Iz razvojne perspektive, druga opcija se gotovo ne razlikuje od prve, ali može da pruži značajno bolje performanse.

Ako aplikacija počne da raste i da postaje sve složenija, moraćete je raspodeliti na više *slojeva*; to će najčešće biti sloj baze podataka koji direktno komunicira s MySQL-om, zatim sloj poslovne logike koji čini srž aplikacije i sloj za prezentaciju podataka koji se stara o generisanju izlaznog HTML koda. Međutim, osnovna arhitektura, prikazana na slici 8.9, i dalje važi; jedina razlika je to što PHP komponenta postaje složenija.

Dodatna literatura

U ovom poglavlju naveli smo neke smernice za projektovanje relacionih baza podataka. Ako vas zanima teorija na kojoj se zasnivaju relacione baze podataka, više ćete saznati iz knjiga neprikosnovenih stručnjaka iz te oblasti, kao što je C. J. Date. Imajte, međutim, u vidu da taj materijal može biti isuviše teorijski i – kao takav – ne mnogo značajan za komercijalnog veb programera. Prosečne baze podataka za veb obično nisu toliko složene.

Šta sledi

U sledećem poglavlju počinjemo izradu MySQL baze podataka. Prvo ćete naučiti kako da napravite takvu bazu podataka, kako da joj postavljate upite i kako da joj upite postavljate iz PHP-a.