KAPITOLA

8

Navrhujeme webovou databázi

V této části knihy budete pracovat s databázovým systémem MySQL. Než se jím začnete zabývat podrobněji (v příští kapitole), měli byste se dozvědět o:

- **Koncepce a terminologie relačních databází.**
- **Navrhování webové databáze.**
- **Architektura webové databáze.**

Když už znáte základy jazyka PHP, můžete se začít zabývat tím, jak začlenit databáze do svých skriptů. Možná si pamatujete, že kapitola 2 uváděla výhody databází oproti prostým textovým souborům. Patří k nim:

- Databáze poskytují rychlejší přístup k datům než prosté textové soubory.
- Databází se lze dotazovat na skupinu dat, která splňují určitá kritéria.
- Databáze mají vestavěné mechanizmy pro zacházení se současným přístupem více uživatelů a s konflikty z toho plynoucími, kterými se proto jako programátoři nemusíte zabývat.
- Databáze poskytují náhodný přístup k datům.
- Databáze mají vestavěný systém oprávnění.

Pomocí databází můžete například rychle odpovědět na otázky, odkud pocházejí vaši zákazníci, které produkty se prodávají nejlépe a jaký typ zákazníků nejvíce utrácí. Tyto informace vám můžou pomoct vylepšovat vaše webové stránky, abyste zaujali více návštěvníků. Z prostého textového souboru byste je dolovali jen stěží.

V této části knihy se naučíte následující:

- V kapitole 9, „Vytváříme webovou databázi“, se dozvíte o základní konfiguraci pro připojení databáze MySQL k webu. Naučíte se vytvářet uživatele, databáze, tabulky a indexy a dozvíte se o nejrůznějších typech úložišť systému MySQL.

- Kapitola 10, „Práce s databází MySQL“, vysvětluje, jak se dotazovat do databáze, jak přidávat, odstraňovat a upravovat záznamy – a to vše z příkazového řádku.
- V kapitole 11, „Přistupujeme do databáze z webu s jazykem PHP“, se dozvíte, jak propojit jazyk PHP se systémem MySQL, abyste mohli používat a spravovat svou databázi z webového rozhraní. Můžete používat dvě metody – nativní ovladač MySQL a databázově nezávislé rozšíření PDO.
- Kapitola 12, „Pokročilá správa systému MySQL“, se věnuje správě systému MySQL podrobněji. Obsahuje informace o systému oprávnění, zabezpečení a optimalizaci.
- Kapitola 13, „Pokročilé programování se systémem MySQL“, se zabývá typy databázových úložišť podrobněji, a to včetně transakcí, fulltextového vyhledávání a uložených procedur.

Koncepce relačních databází

Relační databáze jsou nejrozšířenějším typem databází. Svůj základ mají v relační algebře. Nemusíte rozumět relační teorii, abyste mohli používat relační databáze, ale musíte znát základní koncepce relačních databází.

Tabulky

Relační databáze se skládají z **tabulek**. Tabulky jsou přesně to, co naznačuje jejich název – tabulky dat. Pokud jste někdy pracovali s Excelem, už jste používali podobné tabulky.

Prohlédněte si ukázkovou tabulku na obrázku 8.1. Obsahuje jména a adresy zákazníků obchodu Book-O-Rama.

Customers

CustomerID	Name	Address	City
1	Jana Nováková	Dubová 25	Pelhřimov
2	Aleš Světlý	Kopřivová 47	Horní Bečva
3	Michal Starý	Severní 357	Pustiměř

Obrázek 8.1. Informace o zákaznících obchodu Book-O-Rama uložené v tabulce

Tabulka se jmenuje `Customers` (zákazníci), přičemž obsahuje několik sloupců s různými kusy data a řádky odpovídající jednotlivým zákazníkům.

Sloupce

Každý sloupec má jedinečný název a obsahuje různá data. Ke každému sloupci se váže určitý datový typ. V tabulce `Customers` na obrázku 8.1 můžete vidět, že sloupec `CustomerID` obsahuje celá čísla a v dalších třech sloupcích jsou textové řetězce. Sloupce se také občas nazývají pole nebo atributy.

Řádky

Každý řádek v tabulce reprezentuje jiného zákazníka. Díky tabulkovému formátu mají všechny řádky stejné atributy. Řádkům se někdy rovněž říká záznamky.

Hodnoty

Jednotlivé řádky se skládají z různých hodnot pro jednotlivé sloupce. Každá hodnota musí mít datový typ, který odpovídá příslušnému sloupci.

Klíče

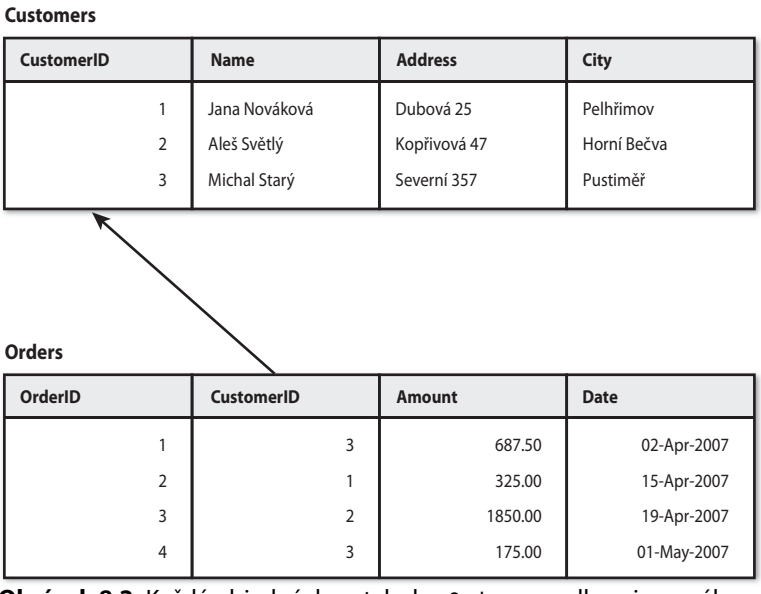
Musíte mít možnost identifikovat jednotlivé zákazníky. Jména nejsou vhodným údajem k identifikaci. Pokud máte rozšířené jméno, pravděpodobně chápete proč. Uvažte kupříkladu jméno Jana Nováková z tabulky `Customers`. Když si otevřete telefonní seznam, najdete v něm příliš mnoho výskytů tohoto jména.

Mohli byste rozlišovat Janu několika způsoby. O něco pravděpodobnější je, že Jana Nováková je jedinou nositelkou tohoto jména na své adrese. Identifikace „Jana Nováková, žijící na adrese Dubová 25, v Pelhřimově" je však poměrně krkolomná. Pravděpodobně by vyžadovala zapojení více sloupců tabulky.

V tomto příkladu jste udělali něco, co bude i skvělým řešením ve vašich aplikacích – přiřadili jste jednotlivým zákazníkům jedinečné číselné identifikátory `CustomerID`. Jedná se o stejný princip, proč například máte jedinečné číslo účtu v bance nebo jedinečné klubové číslo. Díky tomu bude ukládání osobních údajů do databáze jednodušší. U uměle vygenerovaného čísla lze snadno zaručit, aby bylo jedinečné. Jen málo typů skutečných informací má tuto vlastnost, a to i když jich spojíte více.

Identifikující sloupec tabulky se nazývá **klíč** nebo **primární klíč**. Klíče se můžou skládat z více sloupců. Kdybyste se opět vrátili k „Janě Novákové, Dubová 25, Pelhřimov", mohli byste složit klíč ze sloupců `Name` (jméno), `Address` (adresa) a `City` (město) a přitom byste stále nemohli zaručit jedinečnost.

Databáze se obvykle skládají z více tabulek, přičemž klíče slouží jako odkazy z jedné tabulky do druhé. Obrázek 8.2 ukazuje druhou tabulku přidanou do dané databáze. Každý řádek tabulky `Orders` (objednávky) představuje objednávku od nějakého zákazníka. Abyste neztratili přehled, ke kterému zákazníkovi objednávka patří, budete si ukládat k objednávce identifikátor zákazníka. Když si prohlédnete tuto tabulku, zjistíte například, že objednávku `OrderID 2` vytvořil zákazník `CustomerID 1`. Pokud se potom podíváte do tabulky `Customers`, uvidíte, že zákazníkem `1` je Jana Nováková.



Obrázek 8.2. Každá objednávka v tabulce `Orders` se odkazuje na zákazníka z tabulky Customers

Relační databáze označuje tento typ vztahu jako **cizí klíč**. Sloupec `CustomerID` je primárním klíčem tabulky `Customers`, ale pokud se objeví v jiné tabulce, například v tabulce `Orders`, říká se mu cizí klíč.

Možná se divíte, proč byste měli mít dvě samostatné tabulky. Proč neuložit adresu Jany do tabulky `Orders`? O tomto problému pojednává následující část této kapitoly.

Schémata

Kompletní sada návrhů tabulek se nazývá **schéma** databáze. Jedná se o plán databáze. Schéma by mělo ukazovat tabulky s jejich sloupci, primární klíče a cizí klíče tabulek. Schéma neobsahuje data, ale můžete spolu s ním uvádět ukázková data, která pomáhají objasnit jednotlivé jeho komponenty. Schéma lze znázornit diagramem, kterému se říká **ERD diagram** (**entity relationship diagram**, jehož popis ale není součástí této knihy), nebo textově; například takto:

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

Podtržené sloupce jsou primární klíče tabulek, ve kterých jsou uvedené. Kurzíva označuje cizí klíče tabulek, v nichž jsou uvedené.

Vztahy

Cizí klíče reprezentují vztahy mezi daty dvou tabulek. Například cizí klíč z tabulky `Orders` odkazující do tabulky `Customers` představuje vztah mezi řádkem v tabulce `Orders` a řádkem v tabulce `Customers`.

V relačních databázích jsou k dispozici tři základní druhy vztahů. Rozdělují se podle počtu prvků na každé straně vztahu. Vztahy můžou být jeden-na-jednoho, jeden-na-mnoho a mnoho-na-mnoho.

Vztah jeden-na-jednoho znamená, že na každé straně se může nacházet jeden záznam. Když například oddělíte adresy a zákazníky do samostatných tabulek, můžete mezi nimi mít vztah jeden-na-jednoho. Cizí klíč byste mohli umístit buď do tabulky `Addresses` nebo do tabulky `Customers` (v obou být nemusí).

Ve vztahu jeden-na-mnoho může patřit k řádku z první tabulky libovolný počet řádků z druhé tabulky. Například jeden zákazník může odeslat více objednávek. V takovém případě tabulka, která má více řádků (`Orders`), bude obsahovat cizí klíč, který bude odkazovat na řádek druhé tabulky (`Customers`). Proto jste umístili cizí klíč `CustomerID` do tabulky `Customers`.

Ve vztahu mnoha-na-mnoho se více řádků z první tabulky hlásí k více řádkům z druhé tabulky. Předpokládejte kupříkladu, že máte dvě tabulky, `Books` (knihy) a `Authors` (autoři). Běžně se stává, že jednu knihu napíše více spoluautorů, případně že jeden autor napíše více knih, ať už sám, nebo jako spoluautor. Tento typ vztahu vyžaduje samostatnou tabulku, takže byste měli tabulky `Books`, `Authors` a `Books_Authors`. Tato třetí tabulka bude obsahovat jen klíče ze zbývajících dvou tabulek – tj. dvojice cizích klíčů, které říkají, jací autoři napsali jaké knihy.

Navrhování webové databáze

Poznat, že potřebujete novou třídu a jakou by měla mít klíč, je tak trochu umění. Můžete si přečíst hromadu materiálů o ERD diagramech a normalizaci databází. Tato témata jsou však nad rámec této knihy. Většinou si vystačíte s několika základními principy návrhu, o nichž se naučíte na příkladu obchodu Book-O-Rama.

Představujte si skutečné objekty, které se snažíte modelovat

Když vytvářítе databázi, obvykle modelujete skutečné objekty a vztahy, abyste o nich mohli ukládat informace.

Každá třída objektů, kterou modelujete, potřebuje svou vlastní tabulku. Můžete si to představit takto: chcete ukládat stejné informace o všech zákaznících. Pokud skupiny dat mají stejný „tvar“, můžete pro ně vytvořit odpovídající tabulku.

V obchodu Book-O-Rama budete chtít ukládat informace o zákaznících, o knihách, které prodáváte a o objednávkách. Všichni zákazníci se nějak jmenují a někde bydlí. Každá objednávka má datum, celkovou cenu a skupinu objednaných knih. Každá kniha má číslo ISBN, autora, název a cenu.

Potřebujete tudíž alespoň tři tabulky v této databázi – `Customers`, `Orders` a `Books`. Tento základ schématu si můžete prohlédnout na obrázku 8.3.

Customers

CustomerID	Name	Address	City
1	Jana Nováková	Dubová 25	Pelhřimov
2	Aleš Světlý	Kopřivová 47	Horní Bečva
3	Michal Starý	Severní 357	Pustiměř

Orders

OrderID	CustomerID	Amount	Date
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-May-2007

Books

ISBN	Author	Title	Price
9788025137505	Ondřej Baše	jQuery pro neprogramátory	424
9788025147375	Timothy Boronczyk	MySQL Okamžitě	212
9788025141960	Callum Hopkins	PHP Okamžitě	212

Obrázek 8.3. Základ schématu se skládá z tabulek Customers, Orders a Books

Ze současného modelu nemůžete určit, které knihy patří do dané objednávky. Tuto situaci vyřešíte za chvíli.

Neukládejte redundantní data

Dříve jsme si položili otázku: „Co nám brání, abychom uložili adresu Jany Novákové do tabulky `Orders`?“

Kdyby si Jana Nováková objednávala z obchodu Book-O-Rama častěji, ukládali bychom její adresu opakovaně. Tabulka `Orders` by nakonec mohla vypadat jako na obrázku 8.4.

OrderID	Amount	Date	CustomerID	Name	Address	City
12	4987.50	25-Apr-2007	1	Jana Nováková	Dubová 25	Pelhřimov
13	1075.00	29-Apr-2007	1	Jana Nováková	Dubová 25	Pelhřimov
14	399.99	30-Apr-2007	1	Jana Nováková	Dubová 25	Pelhřimov
15	594.00	01-May-2007	1	Jana Nováková	Dubová 25	Pelhřimov

Obrázek 8.4. Návrh databáze vede k ukládání redundantních dat, která zbytečně zabírají diskový prostor a můžou způsobovat různé problémy

Takový návrh s sebou nese dva problémy:

- Plýtvá diskovým prostorem. Proč ukládat údaje o Janě čtyřikrát, když je stačí uložit jednou?
- Může vést k aktualizačním anomáliím – to jsou situace, kdy změna v databázi skončí nekonzistentními daty. Jakmile dojde k narušení integrity dat, už nemůžete určit, která data jsou správná, a která nikoliv. To obvykle vede ke ztrátě dat.

Měli byste se vyhýbat třem aktualizačním anomáliím: anomáliím úprav, vkládání a mazání.

Kdyby se Jana přestěhovala v době, kdy bude mít nevyřízené objednávky, museli byste měnit její adresu na čtyřech místech současně – to je čtyřnásobné množství práce. Kdybyste ignorovali tuto skutečnost a změnili její adresu jen na jediném místě, měli byste nekonzistentní data v databázi (to je velmi špatná věc). Tyto problémy se označují anomálie úprav, protože nastávají při úpravách záznamů.

S tímto návrhem musíte vkládat Janiny osobní údaje, kdykoliv si něco objedná. Vždy musíte zajišťovat, aby její údaje zůstaly konzistentní s již uloženými daty. Kdybyste to neudělali, skončili byste s konfliktními informacemi o Janě. Například jeden řádek by tvrdil, že Jana žije v Pelhřimově, kdežto druhý řádek by tvrdil, že žije v Humpolci. Tuto situaci nazýváme anomálií vkládání, jelikož vzniká při vkládání záznamů.

Třetí typ anomálie se označuje jako anomálie mazání, protože nastává, když mažete záznamy z databáze. Představte si kupříkladu, že po odeslání objednávky zákazníkovi ji budete chtít vymazat. Jakmile byste vyřídili všechny Janiny objednávky, zmizely by z tabulky `Orders`. To by znamenalo, že už neznáte nadále Janinu adresu. Nemůžete ji posílat speciální nabídky, a až si bude chtít příště objednat, bude muset vyplňovat všechny osobní údaje znovu.

Databáze byste měli navrhovat tak, aby k těmto anomáliím nedocházelo.

Používejte atomické hodnoty sloupců

Používat atomické hodnoty sloupců znamená ukládat do každého sloupce každého řádku jen jedinou informaci. Kdybyste kupříkladu chtěli vědět, z jakých knih se objednávka skládá, mohli byste to udělat mnoha způsoby.

Jedním řešením by bylo přidat do tabulky `Objednavky` sloupec se všemi objednanými knihami, jak lze vidět na obrázku 8.5.

Orders

OrderID	CustomerID	Amount	Date	Books Ordered
1	3	687.50	02-Apr-2007	9788025137505
2	1	325.00	15-Apr-2007	9788025147375, 9788025141960
3	2	1850.00	19-Apr-2007	9788025137505
4	3	175.00	01-May-2007	9788025147375, 9788025141960, 9788025137505

Obrázek 8.5. S tímto návrhem by sloupec Books Ordered obsahoval více hodnot

Toto řešení není dobrým nápadem z mnoha důvodů. Ve skutečnost vkládáte celou tabulku do jednoho sloupce – tabulku, která by měla spojovat objednávky s knihami. Když vytváříte podobné sloupce, je složitější odpovídat na otázky typu: „Kolik kopií knihy *jQuery pro neprogramátory* si zákazníci objednali?“ Systém nemůže jednoduše spočítat odpovídající záznamy. Místo toho je nutné parsovat hodnoty ve sloupcích, abyste mohli porovnávat jejich obsah.

Protože ve skutečnosti vytváříte tabulku v tabulce, měli byste raději vytvořit samostatnou tabulku. Tuto novou tabulku můžete pojmenovat například `Order_Items` (viz obrázek 8.6).

Order_Items

OrderID	ISBN	Quantity
1	9788025137505	1
2	9788025147375	2
2	9788025141960	1
3	9788025137505	1
4	9788025147375	1
4	9788025141960	2
4	9788025137505	1

Obrázek 8.6. Tento návrh usnadňuje hledání objednaných knih

Tato tabulka vytváří vazby mezi tabulkami `Orders` a `Books`. Takový typ tabulky reprezentuje vztah mnoho-na-mnoho mezi dvěma objekty – jedna objednávka se může skládat z více knih a jedna kniha může být součástí více objednávek.

Jestliže máte řešit problém, který skutečně vyžaduje neatomické hodnoty sloupců, měli byste zvážit výběr databáze pro tento typ dat namísto relační databáze. Jedná se o tzv. nerelační databáze, databáze NoSQL nebo datová úložiště. (Databázemi NoSQL se však tato kniha nezabývá.)

Vybírejte smysluplné klíče

Ujistěte se, že vybíráte klíče, které jsou jedinečné. V tomto případě jste vytvořili speciální klíče pro zákazníky (`CustomerID`) a objednávky (`OrderID`), protože tyto objekty skutečného světa nemusí mít identifikátory, které jsou zaručeně jedinečné. Nemusíte ale vytvářet jedinečný identifikátor pro knihy, protože ten už existuje – jedná se o kód `ISBN`. Do tabulky `Order_Items` byste mohli doplnit další klíč, kdybyste chtěli, ale kombinace sloupců `OrderID` a `ISBN` je jedinečná, dokud pracujete s více kopiemi stejné knihy v objednávce jako s jedním záznamem. Tabulka `Order_Items` obsahuje sloupec `Quantity` právě z tohoto důvodu.

Přemýšlejte, na co se chcete dotazovat databáze

Popřemýšlejte, na jaké otázky by měla vaše databáze odpovídat. Například, které knihy se v obchodě Book-O-Rama prodávají nejlépe? Nezapomeňte, že vaše databáze by měla obsahovat všechna nezbytná data a že k získání odpovědí můžete používat i vztahy mezi tabulkami.

Vyhýbejte se návrhům s mnoha prázdnými sloupci

Kdybyste chtěli přidat recenze knihy do vaší databáze, mohli byste to udělat přinejmenším dvěma způsoby. Oba můžete vidět na obrázku 8.7.

Books

ISBN	Author	Title	Price	Review
9788025137505	Ondřej Baše	jQuery pro neprogramátory	424	
9788025147375	Timothy Boronczyk	MySQL Okamžitě	212	
9788025141960	Callum Hopkins	PHP Okamžitě	212	

Books_Reviews

ISBN	Review

Obrázek 8.7. Recenze knih je možné doplnit přidáním sloupce Review do tabulky Books nebo přidáním samostatné tabulky pro recenze

První řešení spočívá v přidání sloupce `Review` do tabulky `Books`. Tímto způsobem získají všechny knihy sloupec `Review`. Pokud máte v databázi spoustu knih a recenzent je neplánuje recenzovat všechny, spousta řádků nebude mít hodnotu v tomto poli. Ve skutečnosti ale bude mít hodnotu `NULL`.

Mít spoustu hodnot `NULL` v databázi není rozumné. Plýtváte tak diskovým prostorem a způsobuje to problémy se součty a jinými funkcemi pro číselné sloupce. Když uživatel uvidí hodnotu `NULL` v tabulce, neví, jestli je tento sloupec irelevantní, databáze obsahuje chybu nebo zatím nikdo tato data nezadal.

Problémům s mnoha hodnotami `NULL` se můžete vyhnout, když navrhnete svou databázi jinak. V tomto případě byste mohli zvolit druhý návrh z obrázku 8.7. V tabulce `Book_Reviews` jsou uvedené pouze knihy s recenzemi.

Zapamatujte si, že první návrh předpokládal, že máte recenzenta, který nebude psát více recenzí pro jedinou knihu – tj. mezi knihami a recenzemi existuje vztah jeden-na-jednoho. Druhý návrh umožňuje psát více recenzí ke stejné knize – tj. mezi knihami a recenzemi je vztah jeden-na-mnoho. Pokud si vystačíte s jednou recenzí na knihu, můžete u druhého návrhu udělat primární klíč ze sloupce `ISBN` v tabulce `Book_Reviews`. Kdybyste chtěli mít více recenzí jedné knihy, měli byste zavést ještě nový jedinečný identifikátor do této tabulky.

Přehled typů tabulek

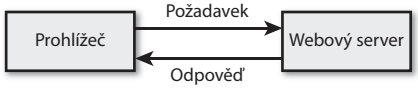
Často zjistíte, že váš návrh tabulek dopadne tak, že bude obsahovat dva typy tabulek:

- Jednoduché tabulky popisující reálné objekty. Mohou také obsahovat cizí klíče odkazující na další jednoduché objekty, s nimiž jsou ve vztahu jeden-na-jednoho nebo jeden-na-mnoho. Například – zákazník může mít více objednávek, ale objednávku může vytvořit jediný zákazník. Proto vkládáte odkaz na zákazníka do objednávky.
- Spojovací tabulky, které popisují vztahy mnoho-na-mnoho mezi dvěma reálnými objekty, jako jsou kupříkladu vztahy mezi objednávkami a knihami. Tyto tabulky většinou odpovídají nějakému typu transakcí skutečného světa.

Architektura webové databáze

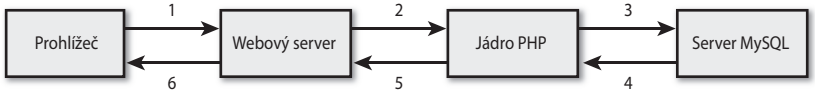
Když už známe interní architekturu databáze, můžeme se zaměřit na externí architekturu webového databázového systému a ukázat si, jak vyvíjet takový systém.

Obrázek 8.8 popisuje základní operace webového serveru. Celý systém se skládá ze dvou objektů – webového prohlížeče a webového serveru. Mezi nimi je nezbytná komunikační linka. Webový prohlížeč odesílá požadavek serveru. Server posílá zpět odpověď. Tato architektura vyhovuje skvěle serveru, který prezentuje statické stránky. Architektura s webovými stránkami postavenými na databázi je o něco složitější.



Obrázek 8.8. Vztah klient-server mezi webovým prohlížečem a webovým serverem vyžaduje komunikační spojení

Webové aplikace s databází, s nimiž se setkáte v této knize, se řídí běžnou strukturou webové databáze, kterou můžete vidět na obrázku 8.9. Převážnou část této struktury byste měli už znát.



Obrázek 8.9. Vztah klient-server mezi webovým prohlížečem a webovým serverem vyžaduje komunikační spojení

Typická transakce webové databáze má níže uvedené fáze, které jsou očíslované na obrázku 8.9. Tyto fáze si popíšeme na příkladu obchodu Book-O-Rama:

- Webový prohlížeč uživatele odesílá požadavek HTTP pro určitou webovou stránku. Například uživatel si může skrze webový formulář vyžádat vyhledání všech knih, které napsala Laura Thomson. Stránka s výsledky hledání se jmenuje *results.php*.
- Webový server přijímá požadavek na stránku *results.php*, načítá tento soubor a předává ho jádru PHP ke zpracování.
- Jádro PHP začíná parsovat tento skript. Uvnitř tohoto skriptu se nachází příkazy pro připojení k databázi a pro provedení dotazu (vyhledávajícího knihy). Interpret PHP otevírá spojení se serverem MySQL a posílá mu tento dotaz.
- Server MySQL přijímá tento dotaz do databáze, zpracovává ho a odesílá výsledky (seznam knih) zpět jádru PHP.
- Jádro PHP dokončí provádění skriptu, který většinou formátuje výsledky dotazu do kódu HTML. Potom vrací výsledný kód jazyka HTML webovému serveru.
- Webový server posílá tento kód HTML zpět webovému prohlížeči, v němž si uživatel může prohlédnout seznam vyhledaných knih.

Tento proces zůstává stejný, ať používáte jakékoliv skriptovací jádro nebo databázový server. Někdy běží webový server, jádro PHP a databázový server na jediném stroji. Není ale ani neobvyklé, aby databázový server běžel na jiném stroji. Může tomu tak být kvůli vyšší bezpečnosti, vyšší kapacitě nebo rozdělování zátěže. Z pohledu vývoje se ale tento proces nemění, mění se jen jeho efektivita.

Jak bude vaše aplikace narůstat, začnete ji rozdělovat do vrstev – obvykle se jedná o databázovou vrstvu pro komunikaci s databází, vrstvu řídicí logiky, která představuje jádro aplikace,

a prezentační vrstvu, která se stará o generování výstupu. Základní architektura z obrázku 8.9 zůstává i v tomto případě zachována – pouze lépe strukturujete kód jazyka PHP.

Další zdroje

V této kapitole jste se seznámili s návrhem relačních databází. Pokud chcete více proniknout do teorie relačních databází, zkuste si přečíst knihy od nějakého relačního guru, například od C.J. Data. Měli byste však vědět, že jeho knihy můžou být značně teoretické a jejich praktický přínos pro webového vývojáře nemusí být přímočarý. Běžná webová databáze není příliš složitá.

Co bude dál

V příští kapitole vytvoříte databázi MySQL. Nejprve se naučíte, jak ji nakonfigurovat pro web a jak se do ní dotazovat, a potom také to, jak se dotazovat z jazyka PHP.