

Použití databází na Webu

4IZ228 – tvorba webových stránek a aplikací

Jirka Kosek

Poslední modifikace: \$Date: 2010/11/18 11:33:52 \$

Copyright © 2000-2010 Jiří Kosek

Obsah

Co nás čeká	3
Architektura webových databázových aplikací	4
K čemu se používají databázové aplikace na Webu	5
Databázové servery	6
Pár základních pojmů	7
Protokoly pro komunikaci s db serverem	8
Relační model dat	9
Základy	10
Vztahy	11
SQL	12
Úvod	13
SELECT (Výběr dat)	14
SELECT – Příklady	15
INSERT INTO (Přidání záznamů)	16
DELETE FROM (Mazání záznamů)	17
UPDATE (Modifikace záznamů)	18
CREATE TABLE (Vytvoření tabulky)	19
Využití databáze ve skriptových prostředích	20
Základní princip	21
Příklad práce s databází v PHP	22
Další možnosti přístupu k datům	23
Alternativní úložiště dat	24
„NoSQL“ databáze	25
XML databáze	26
Objektové databáze	27
RDF databáze	28
Problémy	29
Zvyšování výkonu webových databázových aplikací	30
Řešení simultánního přístupu k datům	31
Aplikační servery	32
Další zdroje informací	33
Další zdroje informací	34

Co nás čeká

- architektura webových databázových aplikací
- k čemu se používají databázové aplikace na Webu
- databázové servery
- relační model dat
- jazyk SQL
- spolupráce webové aplikace s databází
- problémy webových databázových aplikací

Architektura webových databázových aplikací

- typická třívrstvá architektura
- webový prohlížeč = velmi tenký klient
- webový server + webová aplikace = aplikační logika + generování prezentační vrstvy pro prohlížeč
- databázový server = databáze (někdy i část aplikační logiky)

K čemu se používají databázové aplikace na Webu

- skoro každá webová aplikace používá nějakou databázi – data je potřeba někam ukládat
- podnikové informační systémy – nízké TCO, ZAC
- vyhledávací služby, katalogy, knihovny
- i chat je databázová aplikace – jednotlivé zprávy je potřeba někam uložit
- nižší náklady na správu, centralizovaná údržba dat a aplikace je snazší a levnější

Databázové servery

Pár základních pojmů	7
Protokoly pro komunikaci s db serverem	8

Pár základních pojmů

- SŘBD (DBMS), databáze, databázový server, SQL server
- přístup datům řídí server (na rozdíl od souborových databází jako MS Access, Paradox, dBase apod.)
 - lze zajistit současnou práci více uživatelů
 - snese i velmi vysokou zátěž
 - komunikace se serverem většinou probíhá po síti (nejčastěji TCP/IP)
- Oracle, MS SQL Server, MySQL, PostgreSQL, Sybase, DB/2, PostgreSQL,
...

Protokoly pro komunikaci s db serverem

- nativní – každá aplikace má svůj protokol
 - lze plně využít všechny funkce databáze
 - přenos aplikace na jiný databázový server je komplikovaný
- standardizovaná rozhraní – ODBC, JDBC, ...
 - je přidána vrstva navíc, která odstiňuje nativní protokol
 - při změně databázového serveru stačí změnit ovladač
- pro předávání příkazů se používá jazyk SQL

Relační model dat

Základy	10
Vztahy	11

Základy

- data jsou ukládána do tabulek (relací)
- matematicky je model popsán relační algebrou
- pojmy
 - tabulka
 - položka/atribut/sloupec – má název a typ
 - záznam/řádek – je jednoznačně identifikován hodnotou primárního klíče
 - primární klíč – nejmenší množina atributů, které jednoznačně identifikují záznam

Vztahy

- druhy
 - 1:1
 - 1:N
 - M:N – musí se rozložit na dva vztahy 1:N
- v tabulkách se vztahy zaznamenávají pomocí primárních a cizích klíčů

SQL

Úvod	13
SELECT (Výběr dat)	14
SELECT – Příklady	15
INSERT INTO (Přidání záznamů)	16
DELETE FROM (Mazání záznamů)	17
UPDATE (Modifikace záznamů)	18
CREATE TABLE (Vytvoření tabulky)	19

Úvod

- SQL – Structured Query Language
- jednoduchý dotazovací jazyk
- většina databází implementuje standard plus nějaká rozšíření
- výběr dat, přidávání záznamů, mazání záznamů, modifikace záznamů, práce se strukturou databáze, s uživateli a právy, ...

SELECT

Výběr dat

- příkaz SELECT vybírá data z tabulek
- vrací zase tabulku

```
SELECT seznam výstupních položek  
FROM seznam tabulek  
WHERE podmínka  
GROUP BY seznam položek  
HAVING skupinová podmínka  
ORDER BY kritéria třídění
```

SELECT – Příklady

```
SELECT * FROM Zamestnanci;
```

```
SELECT Jmeno, Plat FROM Zamestnanci WHERE Plat > 10000;
```

```
SELECT * FROM Zamestnanci WHERE Jmeno LIKE 'Nov%';
```

```
SELECT * FROM Zamestnanci  
WHERE (Plat < 7000) AND NOT (Jmeno LIKE 'Novák %');
```

```
SELECT * FROM Zamestnanci  
ORDER BY Jmeno;
```

```
SELECT Nazev, Jmeno FROM Odberatele, Zamestnanci  
WHERE Odberatele.Zastupce = Zamestnanci.OsobniCislo
```

INSERT INTO

Přidání záznamů

```
INSERT INTO jméno tabulky  
(jméno položky, jméno položky, ...)  
VALUES (hodnota, hodnota, ...)
```

```
INSERT INTO jméno tabulky  
VALUES (hodnota, hodnota, ...)
```

```
INSERT INTO Zamestnanci  
VALUES (1023, 'Novák Jan', '561220/0235', 'Levá 13, Praha 4', 12000)
```

DELETE FROM

Mazání záznamů

```
DELETE FROM jméno tabulky WHERE podmínka
```

```
DELETE FROM jméno tabulky
```

```
DELETE FROM Zamestnanci WHERE OsobniCislo = 1023;
```

UPDATE

Modifikace záznamů

```
UPDATE jméno tabulky
SET jméno položky = hodnota položky,
    jméno položky = hodnota položky,
    ...
    jméno položky = hodnota položky
WHERE podmínka
```

```
UPDATE Zamestnanci
SET Jmeno = 'Procházková Alena'
WHERE OsobniCislo = 1168;
```

CREATE TABLE

Vytvoření tabulky

```
CREATE TABLE název tabulky (  
  název položky typ,  
  název položky typ,  
  název položky typ,  
  název položky typ,  
  ...)
```

```
CREATE TABLE Zamestnanci (  
  OsobniCislo int NOT NULL PRIMARY KEY,  
  Jmeno       varchar(40),  
  RC          char(11),  
  Adresa      varchar(60),  
  Plat        decimal(10,2))
```

```
CREATE TABLE Proj_Zam (  
  ID_Projektu char(6) NOT NULL,  
  OsobniCislo int NOT NULL,  
  PRIMARY KEY (ID_Projektu, OsobniCislo))
```

Využití databáze ve skriptových prostředích

Základní princip	21
Příklad práce s databází v PHP	22
Další možnosti přístupu k datům	23

Základní princip

- vytvoření připojení k databázi
- zaslání SQL příkazu k provedení
- zpracování výsledku
- odpojení od databáze

Příklad práce s databází v PHP

```
<?php

try
{
    // připojení k databázi
    $db = new PDO("mysql:host=localhost;dbname=test", "jméno", "heslo");

    // zaslání dotazu a čtení výsledku
    foreach ($db->query("SELECT * FROM Zamestnanci ORDER BY Jmeno") as ►
$radka)
    {
        // zpracování jednotlivých řádek výsledku
        echo $radka["OsobniCislo"], " ", $radka["Jmeno"], "<br>\n";
    }
}
catch (PDOException $e)
{
    // obsluha případné chyby při práci s databází
    echo "Při práci s databází došlo k chybě: " . $e->getMessage();
}

?>
```

Další možnosti přístupu k datům

- ORM (Object-Relational Mapping)
 - aplikace nepracuje přímo s databází, ale používá se mezivrstva, která zajišťuje transparentní mapování a perzistenci objektů v paměti na data v databázi
 - programátor pracuje s objekty, nepíše přímo SQL kód
 - jednodušší vývoj, menší riziko chyby a překlepu v SQL kódu
 - v mezních případech může automatické mapování generovat pomalé dotazy a je nutný ruční zásah

Alternativní úložiště dat

„NoSQL“ databáze	25
XML databáze	26
Objektové databáze	27
RDF databáze	28

„NoSQL“ databáze

- typicky jednoduché úložiště klíč/hodnota
- hodnota může obsahovat cokoliv, například strukturovaný datový záznam zapsaný pomocí JSON
- dotazy je potřeba „dělat ručně“, protože NoSQL nepodporuje dotazovací jazyk, spojení tabulek, ...
- díky jednoduchosti oproti SQL-databázím teoreticky umožňuje lepší škálovatelnost
- kromě samostatných produktů tento přístup často používají cloudová úložiště (Amazon S3, Google Storage, ...)

XML databáze

- relační datový model je umělý, datový model XML je v mnoha případech mnohem bližší modelované realitě
- do databáze se ukládají jednotlivé XML dokumenty – např. stránky v CMS, faktury, objednávky, ...
- k dispozici jsou speciální dotazovací jazyky pro XML – XQuery, XPath, ...
- hodí se pro aplikace, které pracují se silně strukturovanými daty, pro které se nehodí relační datový model

Objektové databáze

- databáze rovnou ukládá objekty, se kterými pracuje aplikace
- technologie se nikdy příliš nerozšířila

RDF databáze

- sémantický web – databáze ukládá přímo logické výroky
- existují speciální dotazovací jazyky – SPARQL
- pro produkční nasazení nevyspělá technologie

Problémy

Zvyšování výkonu webových databázových aplikací	30
Řešení simultánního přístupu k datům	31
Aplikační servery	32

Zvyšování výkonu webových databázových aplikací

- perzistentní spojení (connection pooling)
 - pozor, může dojít k překročení limitu spojení do databáze
- použití databází optimalizovaných na čtení – portály
 - v případě potřeby předgenerovat co se dá do souborů nebo sdílené paměti
- v žádném případě nepoužívat MS Access a podobné „rádoby“ databáze

Řešení simultánního přístupu k datům

- nelze použít klasické zamykání záznamů, protože prohlížeč (klient) se po každém požadavku odpojí
- problém se řeší jinak
 - vítězí, kdo přišel první
 - zamknutí záznamu s identifikací uživatele a časovým limitem
 - vhodné pro systémy, kde je možnost kolize vysoká
 - vítězí, kdo první provede změnu
 - kontrola změny dat před jejich konečnou modifikací
 - jednodušší na implementaci
 - vhodné pouze pro případy, kdy je pravděpodobnost kolizí malá

Aplikační servery

- označení poměrně široké skupiny produktů
- nabízejí infrastrukturu potřebnou pro tvorbu rozsáhlejších aplikací
 - sdílení přístupu k databázi (connection pooling)
 - podpora uživatelských relací (session state management)
 - transakční zpracování
 - vyrovnávací paměti
 - řízení přístupu k aplikaci
- příklady: implementace J2EE, ASP.NET, Zope, ...

Další zdroje informací

Další zdroje informací	34
------------------------------	----

Další zdroje informací

- DiBi – Knihovna pro transparentní práci s různými databázemi v PHP¹
- NotORM – další zajímavý nástroj pro jednoduchou práci s databází v PHP²
- Seriál o ORM frameworku Doctrine 2 pro PHP³
- Série článků o různých nerelačních databázích⁴

¹ <http://dibiphp.com/>

² <http://www.notorm.com/>

³ <http://zdrojak.root.cz/serialy/doctrine-2/>

⁴ <http://zdrojak.root.cz/serialy/nerelacni-databaze/>