# Propojení Oracle a PHP na Windows Serveru 2008

# Interconnecting Oracle and PHP on on Windows 2008 Server

Bc. Michal Babušík

Diplomová práce
2011

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Michal BABUŠÍK** |
| Osobní číslo: | **A07749** |
| Studijní program: | **N 3902 Inženýrská informatika** |
| Studijní obor: | **Informační technologie** |
| Téma práce: | **Propojení Oracle a PHP na Windows Serveru 2008** |

Zásady pro vypracování:

1. **Nastudujte problematiku propojení databáze Oracle s PHP na Windows Serveru 2008 a vypracujte literární rešerši na dané téma.**
2. **Porovnejte nalezené možnosti propojení a vyberte nejvhodnější řešení.**
3. **Zprovozněte a otestujte vybrané řešení na školním serveru s databází Oracle.**
4. **Popište výhody a úskalí Vámi vybraného řešení.**

Rozsah diplomové práce:

Rozsah přílohy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:


1. **STANEK, William. Microsoft Windows Server 2008 : Kapesní rádce administrátora. Ondřej Baše, Rostislav Cibulka, Petr Šetka, Pavel Vaida. 1. vyd. [s.l.] : Computer Press, 2008. 704 s. ISBN 978-80-251-1936-5.**
2. **LACKO, Luboslav. Oracle : Správa, programování a použití databázového systému. [s.l.] : Computer Press, 2007. 576 s. ISBN 978-80-251-1490-2.**
3. **LONEY, Kevin , BRYLA, Bob. Mistrovství v Oracle Database 11g. [s.l.] : Computer Press, 2009. 704 s. ISBN 978-80-251-2189-4.**
4. **CRAWFORD, Sharon, RUSSEL, Charlie. Microsoft Windows Server 2008 : Velký průvodce administrátora. [s.l.] : Computer Press, 2009. 1272 s. ISBN 978-80-251-2115-3.**
5. **MCLAUGHLIN, Michael, HARDMAN, Ron, URMAN, Scott. Oracle : Programování v PL/SQL. [s.l.] : Computer Press, 2008. 720 s. ISBN 9788025118702.**
6. **STANEK, William. Mistrovství v Microsoft Windows Server 2008. [s.l.] : [s.n.], 2009. 1368 s. ISBN 978-80-251-2158-0.**
7. **COREY, Michael, ABBEY, Michael, ABRAMSON, Ian. Oracle Database 11g. A Beginner s Guide. [s.l.] : [s.n.], 2008. 432 s.**


Vedoucí diplomové práce: **Ing. Kateřina Ježková**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **24. února 2011**

Termín odevzdání diplomové práce: **18. května 2011**

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.
*děkan*

L.S.

doc. Mgr. Roman Jašek, Ph.D.
*ředitel ústavu*

## ABSTRAKT

Cílem této diplomové práce je propojení databázového systému Oracle a skriptovacího jazyku PHP na operačním systému Windows 2008 Server v 64 bitové verzi. Práce popisuje instalaci, nastavení a vlastní propojení jednotlivých databázových a aplikačních prvků včetně využití webového serveru Apache.


Klíčová slova: Windows, Oracle, PHP, server, database, Apache, operační systém


## ABSTRACT

The main aim of this diploma thesis is to show interconnecting Oracle database system and PHP scripting language running on Windows 2008 Server x64 operating system. This thesis describe basic instalation, setup and main connection between database and application components with use Apache web server.


Keywords: Windows, Oracle, PHP, database, Apache, operating system

Na tomto místě bych rád poděkoval své vedoucí Ing. Kateřině Ježkové za její vedení, čas, odbornou pomoc a cenné informace potřebné k vypracování této práce.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

# OBSAH

# ÚVOD

V současné době je v mnoha odvětvích kladen velký důraz na evidenci. Může se jednat o zásoby, zaměstnanců a podobně. K ukládání a třídění takových dat se používá, různých databázových systému, do kterých lze tato data vkládat a pak samozřejmě také provádět různá vyhodnocení a statistiky.

Tato diplomová práce je zaměřena na teoretický a praktický popis propojení databázového systému Oracle se skriprovacím jazykem PHP na operačním systému Windows Server 2008 v 64 bitové verzi. V práci se jedná o instalaci jednotlivých součástí, které jsou potřebné k bezchybnému fungování a propojení databáze a PHP Výsledné řešení je aplikováno na školním serveru Univerzity Tomáše Bati ve Zlíně, který pak bude sloužit ke studijním účelům. Díky tomuto řešení získají studenti jak teoretické znalosti, tak i praktické dovednosti práce na několika významných a v širokém celosvětovém měřítku používaných softwarových platformách. Windows 2008 Server 64 bit je robustní operační systém, databázový systém Oracle patří zase k jednou z nejrozšířenějším komerčním databázím využívaných nejenom v soukromých společnostech a jazyk PHP je zase nejrozšířenějším programovacím jazykem, používaným při vývoji dynamických webových aplikací.

Zprovoznění serveru tak otevře cestu studentům, kteří se budou chtít ve své praxi věnovat databázovým systémům, jejich tvorbě a programování a v neposlední řadě také práci s daty a vyhodnocováním. Studenti budou mít možnost vyzkoušet si přímo práci se systémem Oracle s využitím nástrojů implementovaných přímo v databázovém stroji, jako jsou Oracle Enterprise Manager (OEM), nebo SQL Plus, tak i s použitím skriptovacího jazyka PHP.

# I. TEORETICKÁ ČÁST

# 1 OPERAČNÍ SYSTÉMY

Operační systém je základní programové vybavení počítače, které zprostředkovává komunikaci mezi uživatelem a hardwarem a řídí činnost jednotlivých částí počítače. V raných dobách výpočetní techniky operační systémy jako takové neexistovaly, počítače byly obsluhovány vědeckými pracovníky, kteří byli současně programátory i uživateli. S počítači komunikovali pomocí strojového kódu, přičemž museli znát přesnou konfiguraci počítače i jednotlivých připojených zařízení.

Absence operačního systému znamenala, že s počítačem mohl v jednom okamžiku pracovat pouze jeden uživatel a to bylo, vzhledem k astronomickým cenám počítačů, naprosto neefektivní. Prvním řešením byl rezervační systém, kdy byli jednotliví uživatelé počítačů přihlášeni k přiřazeným časovým úsekům, ale ani tento systém nebyl nijak zvlášť efektivní. Pokud uživatel dokončil svoji práci před vyčerpáním přiděleného strojového času, zůstal počítač po zbytek času v nečinnosti. Pokud naopak strojový čas vypršel před dokončením úkolu, musel práci přerušit, aby uvolnil prostor dalšímu uživateli. Navíc bylo nutné do rezervovaného strojového času započítat také činnosti, sloužící k načítání programů a vstupních dat. Proto došlo k rozdělení profesí. Zatímco programátoři připravili své práce na nějaké vstupní médium (děrné štítky, děrné nebo magnetické pásky), operátoři podle nastavených priorit vytěžovali počítač jednotlivými joby.

První programy přímo řídily všechny počítačové prostředky, včetně vstupních a výstupní zařízení. Každý program musel obsahovat kód pro kontrolu a provoz vstupu a výstupu. Programátoři, kteří psali programy, vytvářeli nejenom kód, nezbytný k provedení určité aplikace, ale také kód vstupně-výstupních instrukcí. Každý programátor psal své vlastní I/O rutiny. Sám si např. programově ošetřoval načtení vstupních dat z magnetické pásky nebo tiskový výstup na řádkovou tiskárnu. Aby nemusel každý programátor pokaždé znovu „objevovat Ameriku“, začali programátoři s vytvářením společných ovladačů. Ty pak byly organizovány do knihoven, odkud mohly být podle potřeby zařazovány do programů. Později začali výrobci počítačů dodávat vlastní standardní knihovny ovladačů a utilit. Konsolidace vstupně/výstupních rutin do společných knihoven, sdílených všemi programátory, vedla k oddělení I/O procesů od vlastního zpracování dat.

Většinu strojového času však stále zabíraly operace, spojené s nahráváním dat a s jejich výstupem, než čas, strávený jejich vlastním zpracováním. První systém, který vedl k automatizaci jednotlivých transakcí mezi joby, byl vyvinut na počátku roku 1956 výzkumnými laboratořemi firmy General Motors a byl určen pro sálový počítač IBM 701.

Dnes je považován za první dávkový operační systém a v podstatě za první operační systém vůbec. Operační systém načetl program i jeho data, pak program spustil a po ukončení nahrál další program. Takto pokračoval až do doby, dokud nezpracoval všechny joby. Úkoly tedy byly automatizovány, problém ovšem nastal, pokud se během zpracování programu vyskytla chyba. Pak musely další programy čekat třeba i celé hodiny a dny, než byl problém odstraněn a než mohly být zpracovány.

Obrázek 1 IBM 701

Další vývoj tedy směřoval ke sdílení strojového času. To umožnilo sdílet počítačové zdroje více uživatelům současně. Počítač alokoval každému programu pevně stanovenou dobu a po jejím uplynutí se věnoval běhu dalšího programu. Havarované programy tak již neblokovaly zpracování dalších jobů.[1][2]

Dalším významným mezníkem ve vývoji OS bylo vytvoření GUI - grafického uživatelského rozhraní, které má své kořeny v padesátých letech, ale rozvinulo se až letech v sedmdesátých, kdy skupina v Xerox Palo Alto Research Center (PARC) vyvinula první osobní počítač s grafickým uživatelským rozhraním ovládaným myší. Počítač se jmenoval Alto, měl černo-bílý monitor orientovaný na výšku a třítlačítkovou myš.

Skutečný průlom však představoval počítač Apple Lisa resp. Macintosh, který jako první GUI již v roce 1983, resp. 1984 zpřístupnil masám. Mac OS obsahoval mimo jiné drag-and-drop, přímé editování dokumentů, jména disků a aplikací, ovládací panely, plochu,

rozvírací menu, clipboard a další nové prvky. Postupně se vývojem počítačů, které by měly dostatečný výpočetní výkon pro provozování plnohodnotného grafického uživatelského rozhraní, začalo zabývat stále větší množství firem.[3]

## 1.1 Operační systém UNIX

Operační systém UNIX má za sebou dlouhou a fascinující historii. Poprvé byl představen Kenem Thompsonem a Dennisem Ritchiem v článku publikovaném v roce 1974 v "Communications of the ACM", ve skutečnosti však byl uveden na "trh" v roce 1969. A tak i přesto, že jeho počátky sahají až do 60. let minulého století, jeho vlastnosti byly natolik dobře vymyšlené a sofistikované, že s úspěchem mohou konkurovat operačním systémům, které vznikly mnohem později.

Historie UNIXu se začala psát už v roce 1964, kdy byl v Bell Telephone Laboratories (založeny r. 1925) ve spolupráci s MIT (Massachussets Institute of Technology) a General Electric Company zahájen projekt komplexního operačního systému MULTICS (Multiplexed Information and Computing Service), jehož hlavním cílem bylo vytvořit výpočetní systém, který by byl zobecněným prototypem interaktivního systému, v němž by bylo možné efektivní poskytování výpočetních prostředků pomocí vzdálených terminálů. Projekt byl od počátku pojat velmi velkoryse a moderně. Jeho koncepce byla v mnoha ohledech unikátní a řada technik v něm použitých byla přejata většinou pozdějších systémů (hierarchický souborový systém, stránkování na žádost, segmentace, ochrana a utajení informací). Výsledek však nesplnil očekávání. Systém byl velice robustní a nesnadno ovladatelný. Některé funkce se do něj nepodařilo implementovat. Přestože jedním z hlavních cílů byl multiuživatelský provoz, systém měl s tímto připojením problémy. Proto firma Bell Telephone Laboratories od projektu v roce 1969 upustila. Myšlenku jednoduchého, elegantního a snadno použitelného operačního systému však neopustili někteří programátoři projektu, mezi nimi i Ken Thompson, Dennis Ritchie a Brian Kernighan. Ti během roku vytvořili nový OS, který byl napsán v jazyce assembler a byl nazván UNICS ("Universal Information and Computing Service"). Jednalo se o slovní hříčku s názvem MULTICS. Název byl později pozměněn na UNIX. Kvůli přenositelnosti bylo však nutné OS přepsat do nějakého vyššího hardwarově nezávislého jazyka. Ale protože žádný takový jazyk neexistoval, začal K. Thompson předělávat jazyk Fortran upravený pro UNIX, nazval ho B. Na jeho práci navázal D. Ritchie, který nejprve vytvořil

beztypový jazyk BPCL, později po rozšíření o datové struktury vznikl jazyk C, který se ukázal pro účely systémového programování nejvhodnější, protože přes dosažení velmi dobré strojové nezávislosti dokázal generovat velice efektivní kód, jenž umožnil naprogramování tak náročného projektu, jakým bylo jádro operačního systému. To bylo přepsáno v roce 1973 a znamenalo zvrat v dějinách UNIXu. Velkou většinu systému bylo totiž možné přenést na jinou architekturu bez velkých zásahů.[4]

Vývoj Unixu se postupně rozštěpil do dvou větví. První vývojová větev se nacházela na univerzitě v Berkeley, kde vznikl pùvodní BSD Unix (BSD znamená Berkeley Software Distribution), jež byl časem uvolněn k volnému použití a dnes je z něj svobodný software. Je základem systémů jako FreeBSD, OpenBSD, NetBSD apod.[5]

Druhou větev vyvíjela skupina Unix System Laboratories (USL). Od ní jsou odvozeny dnešní systémy typu System V. Časem se začaly objevovat další klony Unixu (IBM AIX, HP-UX, SGI IRIX, Sun Solaris a další). Všechno to však byly operační systémy pro sálové počítače nebo minipočítače.[6]

Další mezník přišel v roce 1991, kdy student Helsinské univerzity Linus Torvalds začal tvořit jádro unixového systému pro počítače řady 386 a své zdrojové kódy dával postupně k dispozici na Internet. Díky tomu přilákal mnoho internetových nadšenců, kteří mu pomáhali vytvořit systém, který dnes zná snad každý - Linux.[7]

V roce 1989 byl na bázi UNIX BSD uveden nový, plně objektový, operační systém NeXTStep s kernelem Mach. Ten je úzce spjat se systémem Mac OS X, neboť když v roce 1997 došlo k odkoupení firmy NeXT firmou Apple, použila systém NeXTStep/OpenStep 4.2 k vývoji nového operačního systému pro počítače Apple Macintosh.[8]

## 1.2 Operační systémy Mac OS a Mac OS X

Historie operačního systému Mac OS sahá do roku 1977, kdy firma Apple Computers vyprodukovala komerčně úspěšný počítač Apple II. Skutečný Mac OS 1.0 však byl uveden na trh až v lednu roku 1984 spolu s prvním počítačem Macintosh. Mac OS byl velmi pokrokovým operačním systémem a kromě GUI obsahoval i další moderní prvky - ovládání myší, multitasking, multimedia, podporu práce v sítích, atd.

Mac OS byl do roku 2002 postupně uveden v 9 verzích, poté byl nahrazen zcela novým operačním systémem Mac OS X, jehož základem se stal objektově orientovaný operační systém NeXTSTEP postavený na bázi Unixu a vybavený vlastním grafickým rozhraním. Mac OS X je moderní objektově orientovaný systém založený na kvalitním a stabilním základu BSD Unix, vybavený vektorovým grafickým rozhraním Aqua GUI. Mac OS X nabízí kompletní implementaci systému X Window pro aplikace založené na X11, což umožňuje instalaci běžných Linuxových aplikací prostředí X11.

Základní myšlenkou systému je jednoduchost a intuitivnost uživatelského rozhraní. Ergonomie Mac OS X je ve srovnání s jinými OS na vysoké úrovni; výrazně jednodušší je mimo jiné i customizace OS. Jednou z největších výhod tohoto OS je rovněž prakticky nulové množství virů a spyware (jedinou známou výjimkou mohou být pouze macroviry v MS Office). [8]

## 1.3 Operační systémy Windows

Historie operačního systému Windows sahá do roku 1981, kdy firma IBM uvedla na trh první PC spolu se 16bitovým operačním systémem MS-DOS (Microsoft Disk Operating System). Pro další vývoj počítačového průmyslu a šíření OS od Microsoftu se ukázalo jako rozhodující, že firma IBM, na rozdíl od firem Atari, Commodore nebo Apple, umožnila výrobu klonů svých počítačů a ty se pak velmi rychle rozšířily do celého světa.

Ačkoliv MS-DOS v podstatě umožnil masivní rozvoj mikropočítačů, byl již v době svého vzniku nepohodlný a z hlediska návrhu nedostatečný a v podstatě zastaralý. MS-DOS podporoval pouze jednoho připojeného uživatele, který mohl v čase pracovat s pouze jedním programem (chybějící podpora multitaskingu). MS-DOS měl hardwarová omezení, nedokázal pracovat s pamětí větší než 640 kB nebo s disky většími než 30 MB. Většina nedostatků MS-DOSu, především uživatelská nepřívětivost a absence multitaskingu, byla postupně překonána pozdějšími verzemi systému Windows.

Významným mezníkem ve vývoji počítačů bylo uvedení počítače Apple Lisa s operačním systémem LisaDesk na bázi GUI v lednu 1983. U Microsoftu začal vznikat OS Windows jako GUI nadstavba MS-DOS. První verze OS s GUI od Microsoftu - Windows 1.0 byla uvedena na trh 20. listopadu 1985. Tato nadstavba OS DOS sice nebyla příliš použitelná, ale již např. nenutila uživatele ukončovat a znovu spouštět programy. Pokud chtěl s programy pracovat současně, mohl se mezi nimi přepínat, avšak okna se nemohla

překrývat. Tento nedostatek byl odstraněn ve verzi 2.0, která byla uvedena na trh v roce 1987. Zde již bylo možno okna překrývat jedno přes druhé, bez nutnosti je mozaikovitě skládat vedle sebe. Komerčního úspěchu a reálné použitelnosti se ale dočkal až OS MS Windows 3, uvedený na trh v roce 1990, který lze označovat za první reálně použitelný Windows s GUI. Windows verze 3.0 se velmi rychle rozšířily především díky velké hardwarové i softwarové podpoře významných nezávislých výrobců a předinstalováváním na nová PC. V roce 1991 byla vydána rozšířená verze Multimedia Extension pro práci Windows s multimédii. Na jádře pracujícím pod DOSem byly na trh později uvedeny další OS Windows:

Windows 3.1 z dubna 92 přinesla řadu vylepšení (jako podpora OLE, vylepšení správce souborů, vylepšení podpory tiskáren, nových ovladačů podporujících MS-DOS grafiku v okně, virtuální paměť lze měnit v ovládacím panelu, atd.) a odstranění některých chyb v uživatelském rozhraní. Následně byly vydány i Windows 3.11, kde byla přidána síťová podpora.

V srpnu 1995 byly uvedeny Windows 95. Tento systém byl opatřen řadou vylepšení, například částečně 32bitovým jádrem (hybridní 16/32bitové), podporou dlouhých názvů souborů, lepší podporou sítí (byla integrována TCP/IP sada) a zcela novým grafickým rozhraním (GUI Windows 95 bylo tak úspěšné, že se jej Microsoft rozhodl využít i ve verzi Windows NT 4.0). Lze jej označit za povedenou, klíčovou a přelomovou verzi Windows, která bývá označována za první uživatelsky přívětivý operační systém od Microsoftu.

Windows 98, již 32bitový, byl uvedený v červnu 1998. Tento systém přinesl vylepšené grafické prostředí, integraci Webu do oken průzkumníka, plně funkční podporu USB, nový Microsoft Explorer 4. Systém však obsahoval mnoho chyb, kvůli kterým byl velmi nestabilní.

Windows Me (Millenium Edition) je posledním zástupcem této kategorie. Již neumožňoval restart do režimu čistého MS-DOSu, protože obsahoval nové grafické prostředí z Windows 2000 Profesional, Windows Media Player 7 a nové DirectX. Tento operační systém býval označován za nejvíc nestabilní operační systém Microsoftu. V polovině roku 1993 se na scéně objevily Windows NT, která byly zaměřené především na náročné uživatele a servery (zkratka NT znamená New Technology). Pátou verzi Windows NT uvedl Microsoft na trh 17. února 2000. Přesto, že je tento systém volným pokračováním

Windows NT 4.0, obsahoval tolik změn, že byl přejmenován na Windows 2000. Pro většinu domácích uživatelů byl však tento systém příliš robustní a náročný na hardware, proto býval nasazován především na výkonných serverech.

Teprve v roce 2001 Microsoft konečně uvedl na trh operační systém, určený běžným uživatelům, který byl postaven nikoliv na platformě DOSu, ale na přizpůsobené technologii NT. Byl nazván Windows XP (eXPerience). [8]

### 1.3.1 Windows Server2008 R2

Windows Server 2008 R2 je prvním operačním systémem Windows Server, který byl přesunut výhradně na 64bitovou architekturu. [9]

Nejedná se tedy o pouhý update verze Windows Server 2008, nýbrž o novou verzi operačního systému. Na rozdíl od verze 2008, která vychází z verze Vista, používá R2 kód obdobný Windows 7 (Windows NT 6.1). Z novinek v systému je nutné zmínit také .NET Framework 3.5 a IIS 7.5. Systém umožňuje spustit až 256 logických procesorů.

Minimální požadavky na system Windows 2008 Server R2:

| **Součást** | **Požadavek** |
|---|---|
| Procesor | Minimum: 1.4 GHz (x64) |
| | Poznámka: Systém Windows Server 2008 R2 pro počítače s procesorem Itanium požaduje procesor Intel Itanium 2. |
| Paměť | Minimum: 512 MB RAM |
| | Maximum: 8 GB (Foundation) nebo 32 GB (Standard) nebo 2 TB (Enterprise, Datacenter a systémy s procesorem Itanium) |
| Volné místo na pevném disku | Minimum: 16 GB nebo více |
| | Foundation: 10 GB |
| | Poznámka: Počítače s více než 16 GB paměti RAM budou potřebovat více volného místa na pevném disku pro stránkování, hibernaci a odkládací soubory. |
| Zobrazení | Monitor s rozlišením Super VGA (800 × 600) nebo vyšším |

Tabulka 1 Hardwarové požadavky Windows 2008 R2

Skutečné požadavky se budou lišit v závislosti na konfiguraci daného systému a aplikacích a funkcích, které budete instalovat. Výkon procesoru závisí nejen na taktovací frekvenci procesoru, ale také na počtu jader a kapacitě mezipaměti procesoru. Požadavky na volné místo na disku pro systémový oddíl jsou pouze přibližné. Odhady požadovaného místa na disku pro operační systémy pro počítače s procesory Itanium a x64 se budou lišit. Další volné místo na disku může být požadováno v případě instalace přes síť.

# 2 DATABÁZOVÉ SYSTÉMY

Díky počítačům žijeme v éře, v níž nejsou hlavním obchodním artiklem výrobní prostředky, ale informace. Informace utříděné i neutříděné. Utříděné informace bývají sdruženy v nejrůznějších typech databází, které nás obklopují doslova na každém kroku. Setkáváme se s nimi denně a jejich přítomnost si ani neuvědomujeme. Ale ať už se jedná o bankovní transakce, prohledávání webových stránek, výpisy hovorů, evidence zaměstnanců, podnikové agendy a množství jiných evidencí a transakcí, vždy jsou tyto údaje zaznamenávány právě do databází. Během okamžiku lze vyhledat požadovaná data nebo provést analýzu potřebných informací a na jejím základě kvalifikovaně rozhodnout o dalších krocích, postupech či trendech.

## 2.1 Historie a vývoj hromadného zpracování dat

Předchůdcem databází byly papírové kartotéky, které umožňovaly evidovat data a uspořádat je podle různých kritérií. Umožnily snadno přidávat nové položky a vyhledávat existující. Databáze jsou v podstatě elektronické kartotéky, které také umožňují přidávat a prohledávat položky, aktualizovat je nebo je analyzovat. Databáze prošly složitým vývojem od jednoduchých souborových systémů z padesátých a šedesátých let až po dnešní objektově orientované modely.

## 2.2 První velké strojové zpracování dat

Za první velké strojové zpracování dat lze považovat sčítání lidu ve Spojených státech. První americké sčítání lidu proběhlo roku 1790 a od té doby se konalo pravidelně každých deset let. S rostoucím počtem obyvatelstva však rostla i složitost celé práce. V roce 1880 se na práci, která trvala několik let, podílelo již 1500 úředníků. Bylo jasné, že způsob zpracování se musí zrychlit. Se zajímavou myšlenkou tehdy přišel mladý inženýr německého původu Herman Hollerith (1859-1929). V roce 1890 vyhrál konkurz Amerického statistického úřadu se svým rok starým návrhem děrnoštítkového stroje. Výsledky zpracování byly impozantní. Přestože proti předchozímu sčítání stouply náklady o 98%, doba se zkrátila ze sedmi let na pouhých šest týdnů. Genialita řešení spočívala v použití děrných štítků nikoli jako prostředků pro předpis programu, nýbrž jako nosičů dat Děrný štítek o velikosti jednodolarové bankovky navíc umožnil uchování dat pro pozdější použití. Tento způsob na dalších sto let určil charakter hromadného zpracování dat. K

zatím poslednímu slavnému použití děrných štítků došlo v roce 2000 při volbách prezidenta USA, kdy mezi sebou těsně soupeřili George W. Busch a AI Gore. [14]

## 2.3 IBM

V roce 1896 založil Herman Hollerith se svými společníky společnost Tabulating Machine Company. Tato firma se později spojila se dvěma dalšími podniky a dostala název Computing-Tabulating-Recording Corporation (CTR). V roce 1924 byla přejmenována na International Business Machines a to je ona společnost, kterou dnes známe pod zkratkou IBM. Kromě úspěšných elektrických strojů využívajících k třídění dat děrných štítků se IBM zabývalo také prodejem obchodních vah a součástek do domácích přístrojů. [5]

Když v roce 1935 vešel v USA v platnost zákon o sociálním zabezpečení (Sociál Security Act) zavádějící pojištění proti nezaměstnanosti, pojištění starobní a rozsáhlou péči chudinskou, získalo IBM obrovskou vládní zakázku, jejímž obsahem byla údržba evidence 26 milionů zaměstnanců. IBM se také podílelo na sestrojení reléového počítače MARK 1, který v roce 1943 uvedl do provozu Howard Aiken z harwardské univerzity. V roce 1949 spojil svůj život sIBM Edgar 'Teď Codd, pozdější tvůrce relačního datového modelu, dnes jednoznačně nejrozšířenějšího způsobu uložení dat v databázi. [14]

## 2.4 COBOL

Po mnoho let byl nejrozšířenějším jazykem hromadného zpracování dat programovací jazyk COBOL (COmmon Business Oriented Language), vytvořený za společné snahy firem Burroughs Corporation, IBM, Minneapolis-Honeywell (Honeywell Labs), RCA, Spěny.

Rand a Sylvania Electric Products a vládních organizací Spojených států. Jeho historie začíná v květnu roku 1959 na Pensylvánské univerzitě, kde se konal seminář zaměřený na programování a tvorbu software. V rámci tohoto semináře se konala schůzka významných počítačových firem a jejich uživatelů, která se zabývala možností vytvoření jednotného programovacího jazyka, který by byl přenositelný a umožnil by snadnou tvorbu programů. Organizace CODASYL (Conference On Data SYstem Languages), která vznikla jako výsledek konference, představovala dobrovolné sdružení uživatelů a výrobců počítačů a byla pověřena vytvořením jazyka, jež by specifikované požadavky splňoval. Zpočátku tvořilo pracovní skupinu nazvanou "Short Range Commitee" šest lidí, kteří pracovali tak

rychle, že již 17. prosince 1959 předložili výkonnému výboru první návrh jazyka COBOL. Jazyk vycházel z existujících jazyků FLOWMATIC, AIMACO, IBM Commercial Translator a jejich kompilátorů. Umožňoval práci se složitými datovými strukturami a jeho rozšíření bylo tak velké, že více než polovina fungujících programů do poloviny 80.let byla vytvořena právě v něm. První verze se dnes označuje jako COBOL-60. V letech 1968, 1974, 1985 prošel COBOL standardizací ANSI a v roce 2002 standardizaci ANSI/ISO. [14]

## 2.5 Cesta k relačním databázím

### 2.5.1 Síťový model Charlese Bachmana

Již v roce 1961 představil Charles Bachman z General Electric první integrovaný datový sklad. Byl postaven na síťovém modelu, pojmenován byl IDS (Integrated Data Storage) a obsahoval první náznak databázového managementu. Později v šedesátých letech Bachman a další výzkumníci založili v rámci organizace CODASYL pracovní skupinu Database Task Group (DBTG), která připravila návrh standardů síťového modelu. Ve zprávě „The DBTG Apríl 1971 Report" vydané v roce 1971 byla popsána celá architektura síťového databázového systému a objevily se zde pojmy jako schéma databáze, jazyk pro definici schématu apod.

### 2.5.2 Projekt Apollo a hierarchický model

Nezávisle na síťovém modelu byly v šedesátých letech vyvíjeny i hierarchické databáze.

V rámci projektu Apollo americké kosmické agentury NASA, jehož cílem bylo přistání na měsíci, vytvořila North American Aviation (NAA) v roce 1964 hierarchický souborový systém pojmenovaný Generalized Update Access Metod (GUAM). Později se k NAA přidala společnost IBM a ze systému GUAM vznikla první komerčně dostupná databáze s hierarchickým modelem. Pod názvem IMS (Information Management System) byla uvedena na trh v roce 1966

### 2.5.3 Relační model Edgara Franka Codda

V červnu 1970 zveřejnil výzkumný pracovník IBM, absolvent Oxfordu dr. Edgard Frank Codd v odborném časopise Communications of the ACM, the Journal of the Association for Computing Machineiy, Inc., v článku „A Relational Model of Data for Large Shared Data Banks" (Relační model dat pro velké sdílené databanky) návrh implementace nového

datového modelu, později nazvaného „relačním". Codd v něm načrtl možnost, jak použít relační kalkul a algebru pro manipulaci s daty. Už tato původní koncepce předpokládala ukládání dat do tabulek, nezávislost dat na použitém hardware a na způsobu jejich fyzického uložení. K datům se mělo přistupovat pomocí neprocedurálního jazyka vycházejícího z běžné angličtiny. Uživatel měl mít možnost specifikovat operaci nad jím definovanou množinou dat namísto pouhé manipulace s jedním záznamem. Podle relační teorie lze pomocí základních operací (sjednocení, kartézský součin, rozdíl, selekce, projekce a spojení) uskutečnit veškeré operace sdaty. Ostatní operace jsou již jen kombinacemi těchto pěti. Tento návrh byl z počátku vnímán jako intelektuální kuriozita a přestože byl E.F.Codd zaměstnancem IBM, nechtělo se IBM do implementace relačního modelu pustit. Už jenom proto, že by tím ve své podstatě zavrhlo svůj produkt IMS. Zveřejnění Coddova relačního modelu odstartovalo několikaleté období sporů mezi zastánci relačního a síťového modelu. Tyto spory vyvrcholily na konferenci zvláštní zájmové skupiny ACM SIGMOD (Speciál Interest Group on Management of Data), kde výhody relačního modelu obhajoval E.F.Codd a výhody síťového modelu Charles Bachman. Protože argumenty obou stran byly správné, byl výsledkem konference ještě větší zmatek než na jejím začátku. A tak teprve další vývoj databází a počítačových technologií nakonec přispěl k vítězství relačního modelu.

### 2.5.4 Relační databázový systém System R a jazyk SEQUEL

V polovině sedmdesátých let již byly práce na výzkumu a vývoji relačních databází v plném proudu. Mezi lety 1974 a 1978 vyvinul tým 15 pracovníků společnosti IBM prototyp relační databáze s názvem System R. Externí specifikace Systému R nezávisle implementoval také Larry Elison se společností, která později vešla ve známost jako Oracle. Ve firmě IBM se System R po částečném přepsání změnil na SQL/DS a poté na produkt DB2. Pro přístup k datům používal System R jazyk SEQUEL (Structured English Query Language). Výhody relačního přístupu si začaly uvědomovat i další firmy a tak vznikly databázové systémy Progress, Informix, SyBase. Také v těchto systémech se používaly různé verze jazyka SEQUEL, který byl později přejmenovaný na SQL[14]

### 2.5.5 Standardizace SQL

Vzrůstající význam relačních databází si vyžádal nutnost standardizace jazyka. Americký standardizační institut (ANSI) původně zamýšlel vytvořit standard na základě zcela nového jazyka RDL. V roce 1984 se však ukázalo, že se jazyk SQL stává de facto standardem, a

proto byl nový standard založen na tomto jazyku. Bývá označován jako SQL-86 podle roku 1986, ve kterém byl přijat. Protože obsahoval některé nedostatky, byl v roce 1992 přijat nový standard označovaný jako SQL-92 nebo pouze SQL2, který je v oblasti relačních databází standardem dodnes. Protože obecným problémem standardů přijímaných nadnárodními organizacemi je zpoždění, které vznikne mezi potřebou standardů a jejich přijetím, ponechávají jednotliví tvůrci databázových systémů z důvodů zpětné kompatibility ve svých systémech i prvky a konstrukce, které nejsou součástí standardu SQL. A tak i když existuje určitá základní množina příkazů SQL a jejich syntaxe, mohou se i zde vyskytovat odlišnosti, ztěžující přenositelnost mezi různými systém. [14]

### 2.5.6 Microsoft SQL Server

Společnost Sybase vytvořila úspěšný relační databázový systém pro uniové servery a při vývoji další generace Sybase, označované jako System 10, uzavřela dohodu se společností Microsoft na vytvoření databází pro servery Windows. Z důvodů, které nejsou veřejně příliš známé, se však v této spolupráci ještě před dokončením produktů objevily problémy, které vedly k jejímu přerušení. Po ukončení spolupráce si každá strana odnesla veškerou dokončenou práci. Microsoft pak uvedl svůj dokončený produkt pod názvem Microsoft SQL Server, zatímco Sybase uvedl databázi pod původním názvem Sybase System 10. Ale přestože se vývoj obou produktů rozešel, jsou kořeny Sybase v Microsoft SQL Serveru doposud zřetelné.

## 2.6 Databázové model

Databázový model popisuje charakter databázové platformy. Říká, jaké postupy pro zpracování dat jsou v databázové platformě k dispozici, jak je na data nahlíženo, jaká pravidla by měla data splňovat apod. Součástí modelu není, vyjma nejstaršího modelu správy souborů, definice způsobu fyzického uložení dat.

Vyvoj databázových modelů prošel těmito fázemi:

- Model správy souborů
- Hierarchický model
- Síťový model

- Relační model
- Objektově orientovaný model
- Objektově relační model

Starší z uvedených modelů se již samostatně nevyužívají ani nerozvíjejí. Přesto ale výrazně ovlivnily vývoj svých následníků a přinesly mnoho zajímavých teoretických i praktických vylepšení[14]

### 2.6.1 Model správy souborů

Databázový model využívající správy souborů předpokládá, že jednotlivá data jsou ukládána sekvenčně do jednoho velkého souboru umístěného na disku, respektive v dřívějších dobách především na magnetické a děrné pásky či děrné štítky. Toto ukládání sebou nese několik nevýhod. Mezi ty hlavní patří především nutná znalost fyzického uložení dat a sekvenční přístup k datům.

Tento databázový model se již nevyužívá.

### 2.6.2 Hierarchický model

Základem tohoto modelu je stromová struktura, jejímž výchozím prvkem je uzel. V každém systému existuje právě jeden kořenový uzel, ostatní uzly jsou pak umístěny na větvích systému. Jednotlivé uzly jsou propojeny pomocí ukazatelů a mohou být propojeny nejen v klasickém hierarchickém vztahu rodič-syn, ale také v rámci jedné úrovně. Toto propojení umožňuje rychlejší vyhledávání, protože na rozdíl od souborového systému již není nutné prohledávat celý objem dat. Vazby jsou však omezeny na propojení 1:N (jeden potomek má jen jednoho rodiče). Realizace vazeb N:M je sice řešitelná, ale pro svoji komplikovanost pomocí j-edundantních přístupů a cyklických vztahů je velmi obtížná. Dalším omezením hierarchického modelu je nutnost přepracování celé struktury databáze v případě změny požadavků. Naopak jednou z hlavních výhod tohoto modelu je vedle zefektivnění vyhledávání také odklonění se od fyzického modelu. [14]

### 2.6.3 Síťový model

Podstatou síťového modelu je použití ukazatelů vyjadřujících vztahy mezi jednotlivými databázovými položkami. Vzhledem k tomu, že tyto ukazatele mohou být lineární i

cyklické a svým způsobem mohou vyjadřovat skutečné vztahy mezi objekty v databázi (tedy objekty reprezentujícími reálné objekty skutečného světa), lze poměrně snadno pomocí síťového modelu realizovat vztahy N:M, Ve složitějších případech to ale znamená vysokou režii. Vyhledávám konkrétního údaje není v rámci síťového modelu nijak složité, stejně jako vyhledávání odvozených hodnot. Na druhou stranu i síťový model trpí podobnou nevýhodou jako model hierarchický. Tímto handicapem jsou omezení ve změnách struktury databáze. V mnoha případech musí autor aplikace znovu vytvořit celý databázový model. Oproti hierarchickému modelu je však toto omezení méně restriktivní. Přestože se síťový databázový model dočkal celé řady rozšíření a standardů, neexistuje dnes téměř žádná větší komerčně dostupná implementace. Důvod je prostý, velkému zájmu se od poloviny sedmdesátých let začal těšit relační databázový model. [14]

### 2.6.4 Relační model

Relační model je v současné době jednoznačně nejrozšířenějším způsobem uložení dat v databázi. Tento model přináší celou řadu výhod, zejména přirozenou reprezentaci zpracovávaných dat, možnost snadného definování dat a zpracování vazeb. Velkým přínosem relačního modelu je fakt, že klade důraz na zachování integrity zpracovávaných dat a zavádí pojmy jako jsou referenční integrita, cizí klíče, primární klíče apod. Výhodou je, že data lze svázat až podle potřeby, nikoli podle předem definovaných vazeb. Relační model je navíc postaven na tom, že dotazy pracují vždy s určitou množinou dat, nikoli s jednotlivými záznamy jako je tomu u síťového a hierarchického modelu. [14]

### 2.6.5 Objektově orientovaný model

Filozofie objektově orientovaných databází byla přebrána z objektově orientovaných jazyků, proto v objektově orientovaném modelu rozumíme pod pojmem objekt logické seskupení příbuzných dat a datové logiky, které společně reprezentuje nějakou věc či osobu z reálného světa. V rámci objektu se jednotlivé datové položky nazývají proměnné. Funkce, které pracují nad určitými objekty se nazývají metody. K proměnným je možné přistupovat pouze prostřednictvím metod. Této vlastnosti se říká zapouzdření objektů.

Objekty jsou uspořádány do hierarchie tříd. Společné metody a proměnné je proto možné nadefinovat pouze na jednom místě a z něho je pak všichni členové stejné třídy zdědí. Přestože se očekávalo, že tyto databáze vytlačí relační systémy, předpoklady se nenaplnily. Vznikla kompromisní objektově-relační technologie[14]

### 2.6.6 Objektově relační model

Objektově orientovaný model minimalizuje díky zapouzdření dopady modifikací systému, ale na druhou stranu neumožňuje vytváření jednorázových dotazů. Proto někteří výrobci relačních databází doplnily do svých produktů objektově orientované funkce a vznikly tzv. objektově relační databázové systémy. Za ně lze dnes označit databáze Oracle, Informix a DB2. [14]

## 2.7 Relační databáze

Relační databázové systémy vykazují následující charakteristické vlastnosti:

- Veškerá data se dají reprezentovat v pravidelně uspořádaných strukturách s řádky a sloupci, kterým se říká relace
- Všechny hodnoty v databázi jsou skalární. To znamená že v každé konkrétní pozici řádku a sloupce
- Operace v databázi se provádějí vždy nad celou relací a jejich výsledkem je opět jiná relace

### 2.7.1 Základní pojmy

Nejabstraktnější částí návrhu databáze je datový model. Vyjadřuje se pomocí entit, atributů, domén a vztahů.

- *Entity* jsou osoby, místa, věci nebo události o nichž se v systému shromažďují nějaká data (např. Zaměstnanci).
- *Atributy* jsou jednotky faktů a skutečností nějakých způsobem charakterizující a popisující entity (např. číslo zaměstnance).
- *Domény* jsou množiny hodnot stejného významového typu (např. Věk)
- *Relace* popisují vzájemné vztahy mezi entitami. Mohou být typu 1 : 1 (každé instanci jedné entity může být přiřazena nejvýše jedna instance druhé entity), 1 : N (libovolná instance první entity může být přiřazena jedné nebo více instancím druhé entity), N : M (libovolná instance první entity může být sdružena s žádnou, jednou nebo více instancemi druhé entity).
- *Tabulky* jsou v relačním modelu primární jednotky k ukládání dat. Je to dvojrozměrná struktura složená z řádků a sloupců. Každý řádek reprezentuje jeden výskyt entity, kterou tabulka modeluje a každý sloupec reprezentuje jeden atribut této entity.

- *Primární klíč* je sloupec nebo množina sloupců, které dohromady jednoznačně identifikují každý řádek tabulky. Omezení primárního klíče zaručí, že žádné dva řádky tabulky nebudou mít ve sloupci nebo sloupcích duplicitní hodnoty. Pokud se primární klíč skládá z několika sloupců pak jednotlivé sloupce mohou obsahovat duplicitní hodnoty, ale kombinace všech sloupců nikoli.
- *Cizí klíč* je sloupec nebo množina sloupců, která přiřazuje příslušný řádek jedné tabulky s odpovídajícím řádkem druhé tabulky.

### 2.7.2 Operace s relacemi

- *Selekce (Selection)* provádí horizontální výběr z relace. Z relace je možné vybrat různé entity a vytvořit relaci novou.
- *Projekce (Projection)* realizuje vertikální výběr z relace. Vybírá hodnoty jednoho nebo více atributů relace.
- *Sjednocení (Union)* je operace, do které vstupují dvě relace a vzniká relace třetí. Aby bylo možné relace sjednotit, musí být relace stejného stupně (obě relace musí mít stejný počet atributů), atributy v hlavičce obou relací musí být definovány nad stejnými doménami a atributy definované nad stejnými doménami musí být v hlavičkách obou relací uvedeny ve stejném pořadí.
- *Průnik (Intersection)* lze provést pouze nad dvěmi relacemi, které jsou kompatibilní z hlediska sjednocení. Průnik dvou relací obsahuje pouze entity, které jsou přítomné v obou relacích.
- *Rozdíl (Difference)* lze provést nad libovolnými dvěmi relacemi, které jsou kompatibilní z hlediska sjednocení. Rozdíl dvou relací obsahuje všechny prvky první relace, které nejsou obsaženy ve druhé relaci.
- *Spojení (Join)* spojí entity jedné relace s jednou či více entitami druhé relace přes množinu společných záznamů.

### 2.7.3 Přístupová práva

Pracuje-li s databází více uživatelů, není žádoucí, aby všichni mohli provádět v databázi změny nebo aby měli přístup ke všem informacím v databázi. Přístupová práva se týkají vždy jednoho databázového objektu (tabulky, pohledu apod.) a konkrétního uživatele. Existují dvě skupiny přístupových práv.

- *Právo čtení* - uživatel ví o existenci objektu a může přečíst data v něm uložená.

- *Právo zápisu* - uživatel může provádět změnu databázového objektu a dat v něm uložených

Uživatel, který objekt vytvořil má automaticky přidělena všechna práva. Zároveň má právo přidělovat práva dalším uživatelům a to včetně práva na další přidělování vlastněných práv. Práva se přidělují pomocí příkazu *GRANT* a odebírají pomocí příkazu *REVO KE.*

### 2.7.4 **Architektura databázových systémů**

Databázový systém se skládá ze tří základních částí. Z databáze, databázového stroje a aplikace. Databázi tvoří data, které jsou uložené v tabulkách, pohledy na data, dotazy, uložené procedury apod. Fyzickou manipulaci s daty (načítám dat z databáze a ukládám na disk) zabezpečuje databázový stroj. Aplikace se skládá z formulářů a sestav s nimiž uživatel pracuje.

### 2.7.5 Databázové nástroje

Databázové nástroje slouží k výstavbě relačních databází. Na nejnižší úrovni se nacházejí databázové stroje, které provádějí vlastní fyzickou manipulaci s daty. Nad nimi se nacházejí objektové modely pro přístup k datům. Jedná se o jakési „lepidlo", které k sobě pojí programové prostředí a databázový stroj. Strukturu databáze je možné definovat programově nebo pomocí vestavěných nástrojů. Další nástroje slouží k tvorbě uživatelského rozhraní.

Pro transparentní přístup k datům uloženým v různých databázích je zapotřebí standardní způsob spojení se všemi databázemi a standardní jazyk pro práci s daty. Standardní způsob spojení s databázemi poskytuje ODBC (Open Database Conectivity). K datům se přistupuje pomocí jazyka SQL (Structured Query Language). [14]

# 3 APACHE SERVER

Apache HTTP Server je oblíbený softwarový multiplatformní webový server (aplikace určená k poskytování webových služeb, provozovaná v rámci nějakého fyzického serveru), poskytovaný s otevřeným kódem a určený k provozování na systémech GNU/Linux, BSD, Solaris, Mac OS X, Microsoft Windows a dalších platformách.

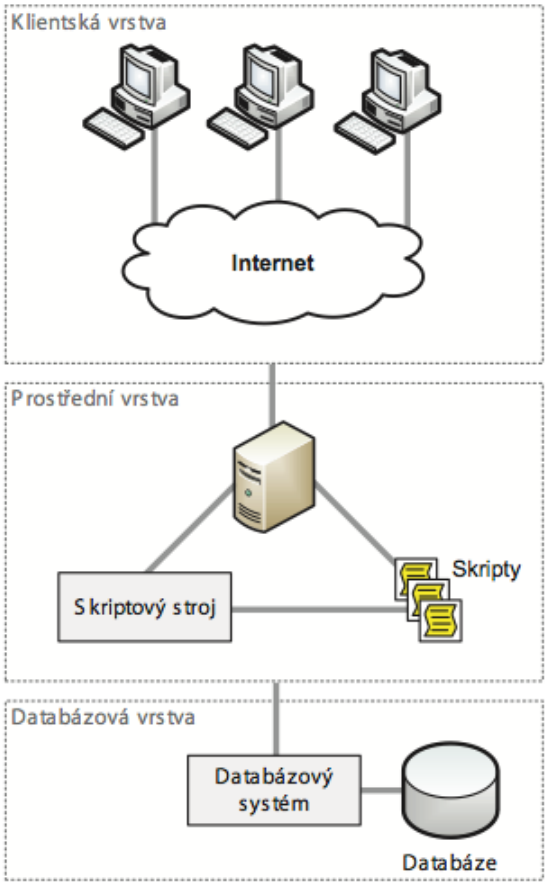
Vývoj Apache začal v roce 1993 v NCSA (National Center for Supercomputing Aplications) na Illinoiské univerzitě. Původní jméno projektu bylo NCSA HTTPd. V dalším roce však vývojářský tým opustil hlavní programátor Rob McCool, čímž došlo ke zpomalení vývoje a poté, v roce 1998, k jeho úplnému zastavení.

NCSA HTTPd však mezitím už používali správci webových serverů a dodávali k němu vlastní úpravy – patche (patch = záplata). Hlavní úlohu v dalším vývoji sehráli Brian Behlendorf a Cliff Skolnick, kteří založili e-mailovou konferenci a začali sběr úprav a jejich distribuci koordinovat. První veřejná verze s označením 0.6.2 byla vydána v dubnu 1995. Pak následovalo kompletní přepsání kódu a tak Apache2 už neobsahuje nic z původního NCSA HTTPd, nakonec došlo k založení Apache Group, která je dnes základem vývojářského týmu.

Od dubna 1996 byl Apache nejpopulárnější server na internetu. V květnu 1999 běžel na 57 % všech serverů a v listopadu 2005, podle výsledků měření Netcraft, jeho používanost dosáhla 69 % .

Název vznikl buď z úcty a obdivu k domorodému kmenu nativních Američanů – Apačů anebo z anglického slovního spojení „A patchy server“ (patchovaný server), protože kdysi byl Apache pouze sada patchů pro jiný web server. Jako indiánský symbol je ve znaku ptačí pero.[10]

Konfgurace serveru se provádí pomocí souboru httpd.conf.




Obrázek 2 Třívrstvý model aplikace

# 4 PHP

PHP (původně Personal Home Page) je serverový skriptovací programovací jazyk, určený především k tvorbě dynamických webových stránek, který však lze s úspěchem použít také k tvorbě konzolových a desktopových aplikací. Při použití PHP jsou skripty prováděny na straně serveru, k uživateli je přenášen pouze výsledek jejich činnosti. To znamená, že nic z toho co PHP provádí, neprobíhá na straně klienta (jako je tomu např. u Java scriptu), ale interpretuje se na straně serveru a generuje výstup (např. ve formě HTML), který pak dává k dispozici uživateli. PHP je Open Source, tedy volně šiřitelná technologie, není závislé na žádné konkrétní platformě a není vázané s žádným konkrétním serverem. Rozdíly v operačních systémech se vzhled k PHP omezují na několik OS-závislých funkcí, takže skripty lze většinou portovat bez jakýchkoli úprav.

Počátky vývoje PHP sahají do roku 1994, kdy dánský student Rasmus Lerdorf vyvtvořil jednoduchý systém evidence přístupů ke svému webu. Ten byl napsán nejdříve v jazyce Perl, poté v jazyce C a rozšířil se také mezi další uživatele, kteří přicházeli s požadavky na vylepšení. Vznikl tak systém Personal Home Page Tools, později Personal Home Page Construction Kit. Erasmus Lerdorf později vytvořil i nástroj, umožňující začleňování SQL dotazů a tím zpřístupnil databází na serveru – Forms Interpreter (FI). V roce 1997 bylo PHP/FI 2.0 oficiálně uvolněno, brzy potom vznikla verze PHP 3.0. Tu vytvořil Andi Gutmans a Zeev Suraski a původní zkratka dostala nový význam – PHP: Hypertext Preprocessor. V roce 2002 bylo oficiálně uvolněné PHP4, které bylo výkonější verzí než PHP 3, přidalo podporu pro mnoho WWW serverů, HTTP sessions, buffering výstupu, bezpečnější způsoby zpracování vstupů uživatele a nové jazykové konstrukty. V červnu 2003 byla oficálně uvolněna betaverze PHP 5, u níž byla největší změna v objektovém modelu, PHP se tak přiblížilo ostatním jazykům podporujícím OOP.[12]

PHP podporuje pro různé účely mnoho knihoven - např. zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (např. MySQL, ODBC, Oracle, PostgradeSQL a další) a také podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP).

Veškeré nastavení PHP skriptů se provádí v souboru php.ini

## 4.1 Vlastnosti jazyka PHP

- Jazyk PHP je dynamicky typový, tzn. že datový typ proměnné se určí v okamžiku přiřazení hodnoty.
- Kvůli tomu má PHP dva operátory porovnání: == a === . Při použití prvního dochází před porovnáním ke konverzi, při použití druhého je výraz pravdivý, jen když jsou oba dva operandy stejného datového typu a jejich obsah má stejnou hodnotu.
- Pole jsou heterogenní (stejné pole může obsahovat prvky různých typů), řídká a počet dimenzí není omezen. Dají se definovat výčtem hodnot (v tom případě se indexují celými čísly, počínaje od 0), nebo mohou fungovat jako hashovací tabulka (kdy může být indexem pole libovolné číslo, řetězec nebo logická hodnota). Stejné pole může obsahovat oba typy indexů
- Řetězce lze uzavírat do uvozovek (při vyhodnocení se provede nahrazení proměnných uvnitř) nebo do apostrofů (nahrazuje se jen escape sekvence \').
- Chyby se vypisují na standardní výstup a mají několik úrovní (poznámka, varování, fatální, …). PHP nabízí možnosti chybové hlášky nastylovat, zakázat výpis té které úrovně, nebo si pro ně vytvořit callback funkci.
- Kromě proměnných, které lze vytvářet i rušit, lze definovat konstanty. Proměnné mají své úrovně viditelnosti a pravidla pro jejich perzistenci (např. proměnná vytvořená ve funkci nebo metodě je po jejím vykonání automaticky zrušena, naproti tomu, proměnná vytvořená např. v cyklu nebo vloženém souboru bude viditelná do skončení programu). Konstanty jsou viditelné ze všech úrovní a po jejich definování je nelze zrušit.
- PHP podporuje reference, pomocí kterých lze do proměnných ukládat odkazy na libovolnou jinou proměnnou, nebo i prvek jejího pole. Jako reference lze volat i parametry funkce. U každé proměnné PHP eviduje, kolik na ni směřuje referencí, a podle toho se rozhoduje, kdy může kterou proměnnou zrušit.
- PHP do verze 4.2.0 ve výchozím nastavení automaticky přejímalo veškeré proměnné poslané jakoukoliv metodou (HTTP POST, HTTP GET, HTTP cookie, ale i ze zabudovaného mechanismu sessions) a umožňovalo s nimi dále pracovat jako s globálními – tato možnost ale představovala bezpečnostní riziko.
- Od verze 4.2.0 lze hodnotu získat z tzv. superglobálních proměnných s garancí původu informace – tedy že data byla odeslána požadovanou metodou. Používání

globálních proměnných je stále možné pomocí konfigurační direktivy register_globals povolit, ale z bezpečnostních důvodů je to silně nedoporučováno.

## 4.2 Výhody PHP

- PHP je specializovaný program na webové stránky.
- Rozsáhlý soubor funkcí - v základní knihovně PHP jich je přes pět a půl tisíce, další funkce v PECL.
- Nativní podpora mnoha databázových systémů.
- Multiplatformost (zejména Linux a Microsoft Windows).
- Možnost využití nativních funkcí operačního systému (možná nekompatibilita s jiným OS).
- Strmá křivka učení.
- Obrovská podpora na hostingových službách – PHP je fakticky standardem.
- Obrovské množství projektů a kódů, které lze zdarma využít (WordPress, phpBB a další).
- Dobrá dokumentace.
- Svobodná licence, která (v protikladu k např. GPL) neobsahuje copyleft.

## 4.3 Nevýhody PHP

- Jazyk PHP není nikde kompletně definován, je popsán pouze jeho implementací.
- Mírně nekonzistentní vývoj v minulosti, který si s sebou PHP nese dosud.
- Nekonzistentní pojmenování funkcí, např.: např. strpos(), strchr(), ale str_replace(), str_pad() atd.
- Nejednotné názvosloví skupin funkcí: mysql_XXXX, imap_XXXX, json_XXXX (s podtržítkem) versus imageXXXX, bcXXXX, gzXXXX (bez podtržítka).
- Nejednotné pořadí parametrů.
- Ač jazyk podporuje výjimky, jeho knihovna je používá jen zřídka.
- Slabší podpora Unicode, pouze přes PHP knihovnu.
- Ve standardní distribuci chybí ladící (debugovací) nástroj.
- Po zpracování požadavku neudržuje kontext aplikace, vytváří jej vždy znovu čímž oslabuje výkon. Tuto nevýhodu lze potlačit vytvářením kódu s vysokou znovupoužitelností nebo použitím jednoho z mnoha PHP frameworků.

- Některá bezpečnostní opatření jako safe_mode, open_basedir jsou příliš elegantně implementovány.[11]

# II. PRAKTICKÁ ČÁST

# 5 INSTALACE WINDOWS 2008 SERVER R2 X64

Windows Server 2008 R2 byl uvolněn 22. července 2009 jako nástupce Windows Server 2003. Windows Server 2008 R2 obsahuje integrované technologie pro web a virtualizaci. Nástroje systému nabízejí kontrolu nad servery a optimalizují konfiguraci a správu.

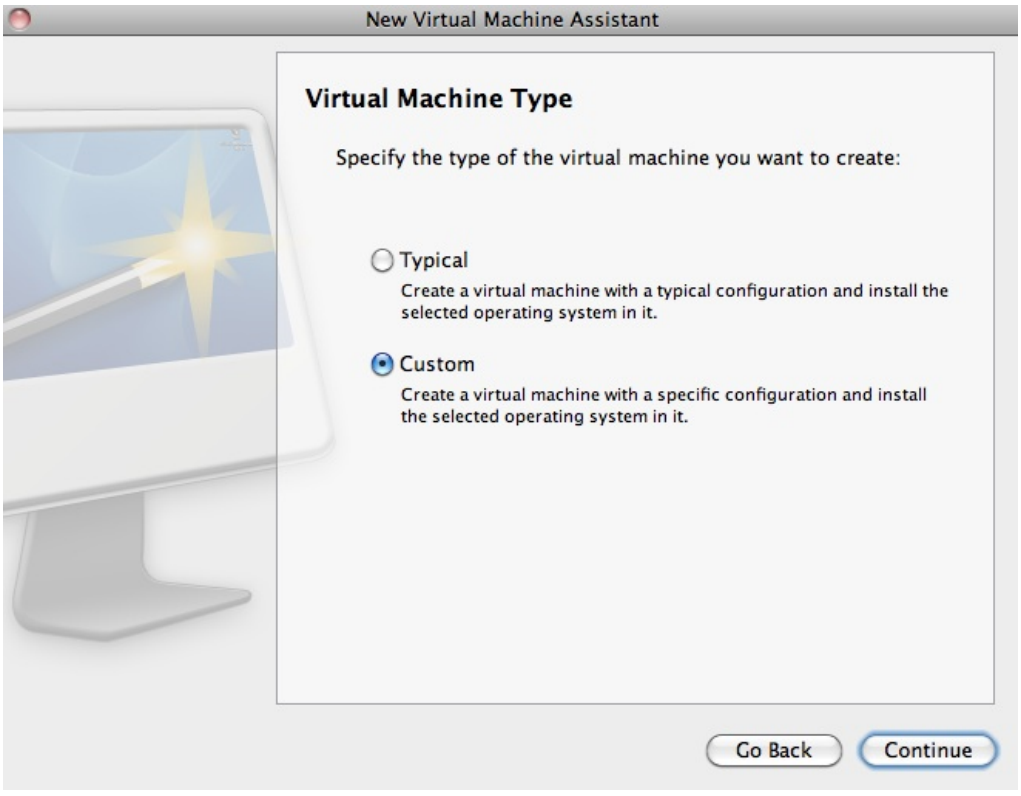
Pro instalaci systému je potřeba mít hardware o minimální konfiguraci procesor: 1.4 GHz (x64), paměť: 512 MB RAM, volné místo na pevném disku: 16 GB a zobrazení s rozlišením Super VGA (800 × 600).

## 5.1 Použitý hardware

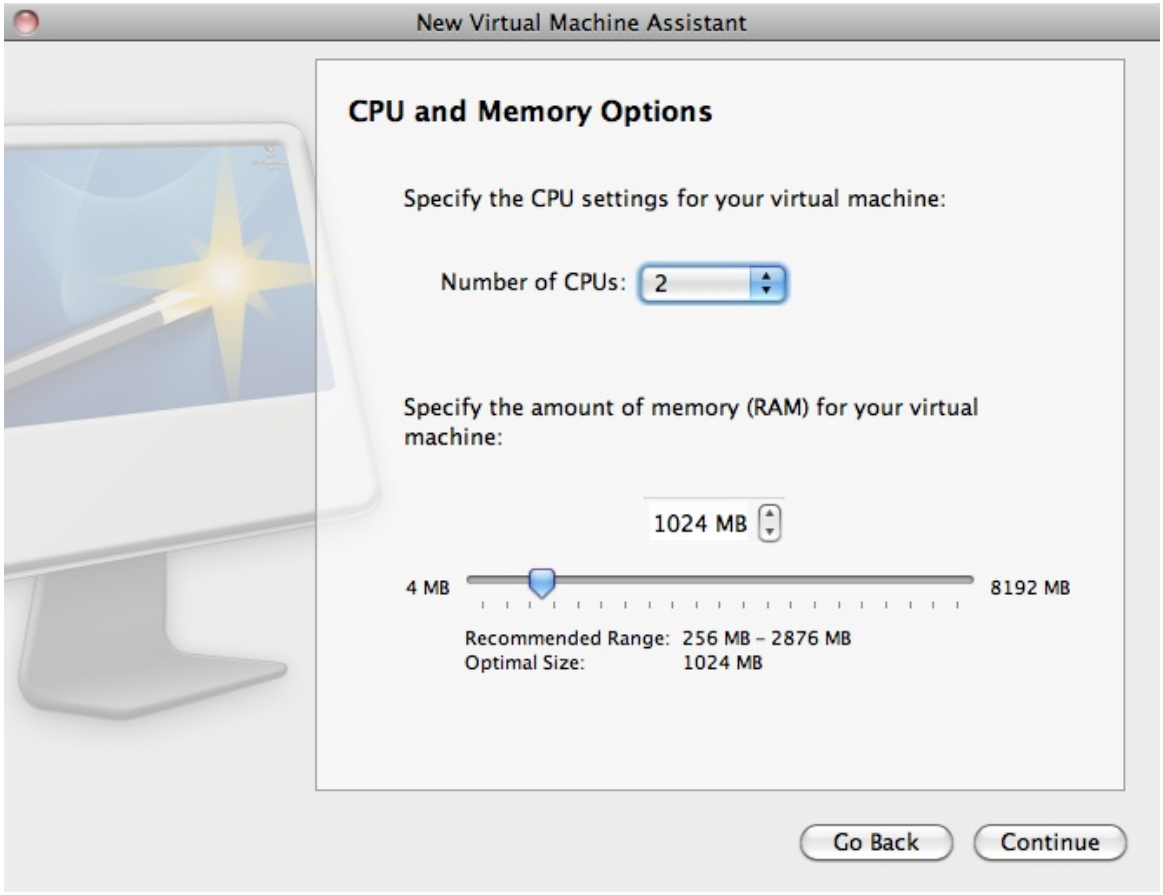
Windows Server 2008 R2 byl nainstalován jako virtuální operační systsém na notebooku MacBook Pro s konfigurací CPU Core2Duo 2,2 Ghz, operační pamětí 4GB a nainstalovaným operačním systémem OS X verze 10.6.7.

## 5.2 Postup Instalace

Instalace operačního systému byla spuštěna pomoci průvodce instalací nového virtuálního systému v programu Parallels Desktop.




Obrázek 3 Volba instalace




Obrázek 4 Konfigurace virtuálního systému

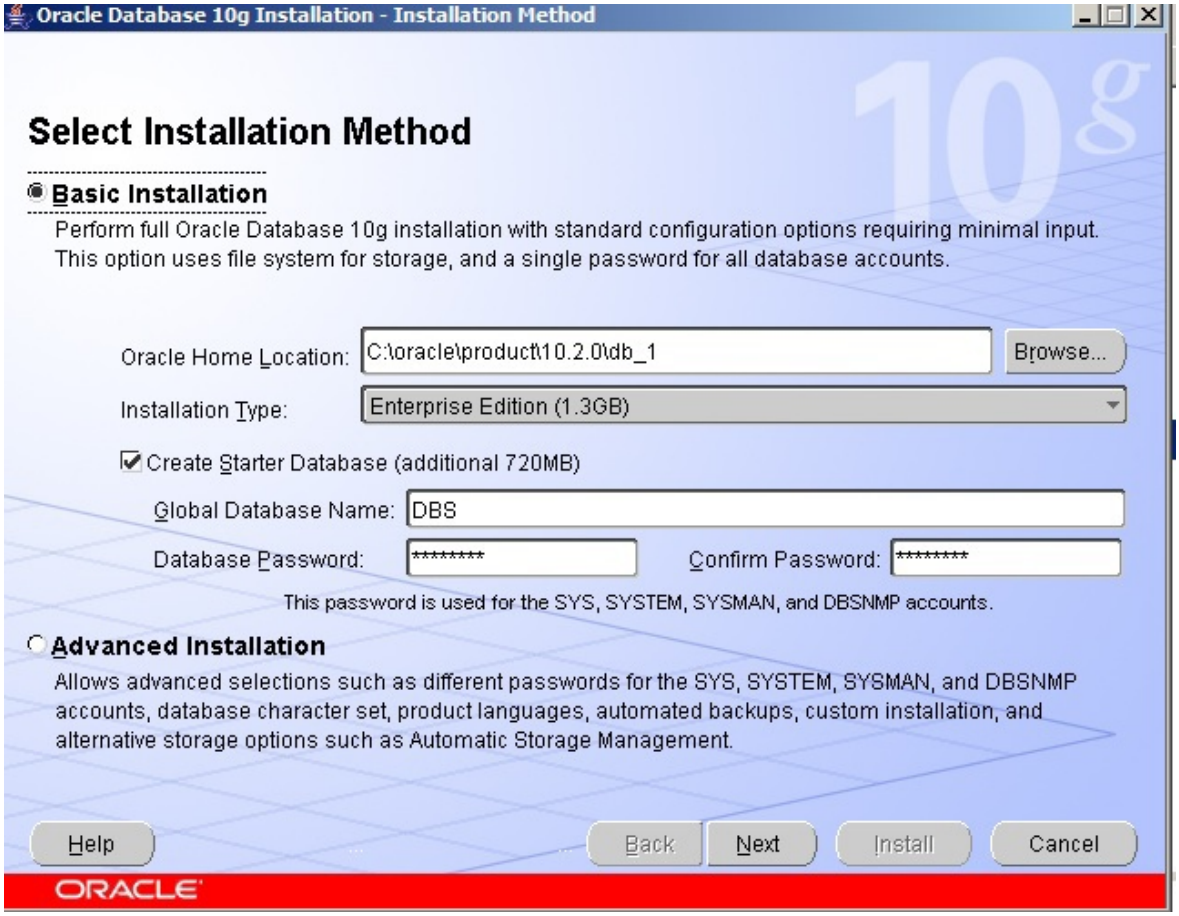
Typ instalace byl zvolen Custom (vlastní), z důvodu nastavení hardwarových prostředků, které bude virtuální operační systém využívat. Virtuálnímu OS bylo přiřazeno 1024 MB operační paměti, využití dvou procesorů a 16 GB prostoru pevného disku. Po nastavení parametrů byl spušten grafický instalátor systému Windows 2008 R2 Server, ve kterém byl zvolen jazyk instalace a typické nastavení systému. Při prvním spuštění musí být vytvořeno nové administrátorské heslo, které musí obsahovat jak malá i velká písmena, tak i číslice.

# 6 INSTALACE ORACLE

Po kontrole vhodnosti operačního systému se zobrazí hlavní obrazovka instalačního programu pro nastavení parametrů. Tady se můžeme rozhodnout pro základní anebo rozšířenou instalaci. Pokud chceme vytvořit implicitní databázi, musíme zadat heslo pro její administraci. Při rozšířené instalaci si můžeme zvolit nejenom domovský adresář, ale i typ instalace a specifikovat účel použití databáze. K nabídnutým typům instalace instalátor zobrazuje předběžný odhad diskového prostoru, který příslušná instalace zabere.

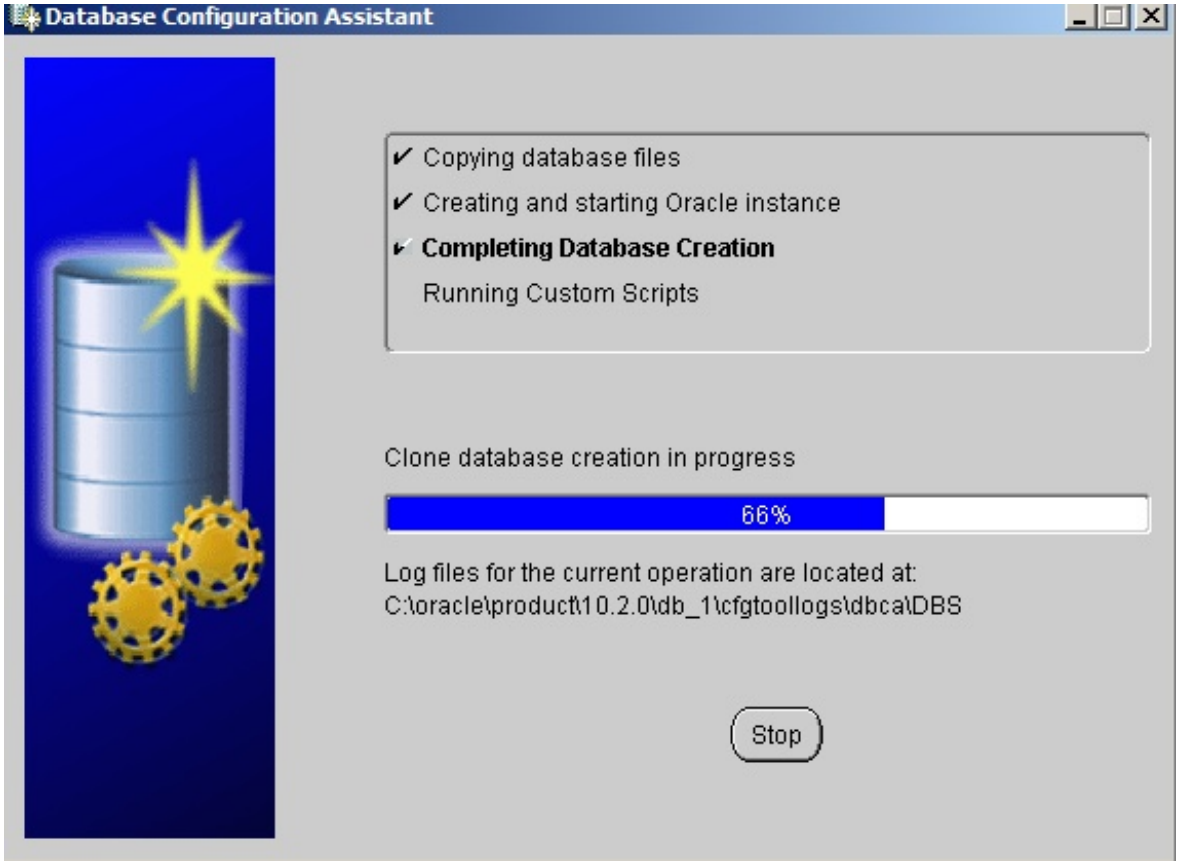
Databázový server si můžeme představit jako službu v pozadí, která spravuje údaje v databázi a reaguje na požadavky klientů. Aby jsme mohli pracovat s databázovým serverem, potřebujeme k tomu dva druhy nástrojů. Jednak nástroj pro správu databáze Oracle Enterprise Manager a jednak konzolovou aplikaci pro zadávání SQL příkazů.

K samotné instalaci byla použita verze produktu Oracle Database 10g. Po spuštění instalace byla vybrána základní metoda instalace a byla nastavena počáteční velikost, jméno databáze, její umístění a heslo pro administraci.




Obrázek 5 Výběr a nastavení instalace Oracle

Po nastavení všech potřebných parametrů pro instalaci a vytvoření databáze instalační program ověřil jestli jsou dostupné potřebné prostředky (volné místo na disku, dostatek operační paměti a podobně) a následně byla spuštěna samotná instalace při které proběhlo nainstalování programových aplikací, vytvoření databáze a nakonfigurování, zaregistrování a spuštění systémových služeb potřebných pro spuštění a provoz databázového serveru.




Obrázek 6 Průběh instalace Oracle

Po instalaci softwaru Oracle se na serveru vytvoří základní struktura, která je na všech serverech stejná, s tím rozdílem, že si zvolíme jen hlavní tzv. base adresář. Jeho proměnná je definovaná jako ORACLE_BASE. ORACLE_BASE je domovský adresář pro software Oracle. Ostatní podadresáře jsou pojmenovány a uloženy dle standardů OFA (oracle flexibile architecture). OFA nepopisuje jen podadresáře, ale pojednává se v ní i o jmenných konvencích pro soubory, přípojné body, přípony souborů nebo pojmenování tabulkových prostorů.

Databázi se instaluje do příslušného adresáře ORACLE_HOME, kam se dokopírují potřebné soubory, a hlavně vytvoří datové soubory dané databáze. Dle OFA se umístí do

adresáře /u01/oradata. Jestliže je databáze vytvořena pomocí asistenta dbca, pak se automaticky spustí.

Po nainstalování softwaru a databáze pomocí dbca se vytvořil i základní init soubor, jehož jmenná konvence je vždy stejná. Jedná-li se o dynamický soubor spfile, pak to je spfile.ora nebo textový soubor init.ora. Textový soubor je možné libovolně editovat, ale změny v databázi se promítnou až po jejím restartování. Naopak dynamický spfile nemůžeme měnit přímo, ale parametry můžeme měnit za běhu databáze. V init souboru jsou uloženy hodnoty všech zásadních parametrů. Jako nejdůležitější z nich lze označit parametr db_name, kde se za rovnítko napíše jméno databáze a parametr control_files, kam se zadá cesta ke kontrolnímu souboru nebo souborům. Samozřejmě se zde nastavují různé parametry týkající se paměti, redo souborů, logovacích adresářů, chování CBO, přístupy k diskům a mnoho dalších parametrů.

## 6.1 Oracle Enterprise Manager

Aplikace Oracle Enterprise Manager (OEM) nám poskytne najkomplexnejší možnosti pro správu databázového serveru Oracle 10g a podrobné informaci o stavu v jakém se databázový server momenátálně nachází, případně jsou signalizované potenciální problémy. K OEM se musíme prihlásit ze správcovského účtu, například SYS.

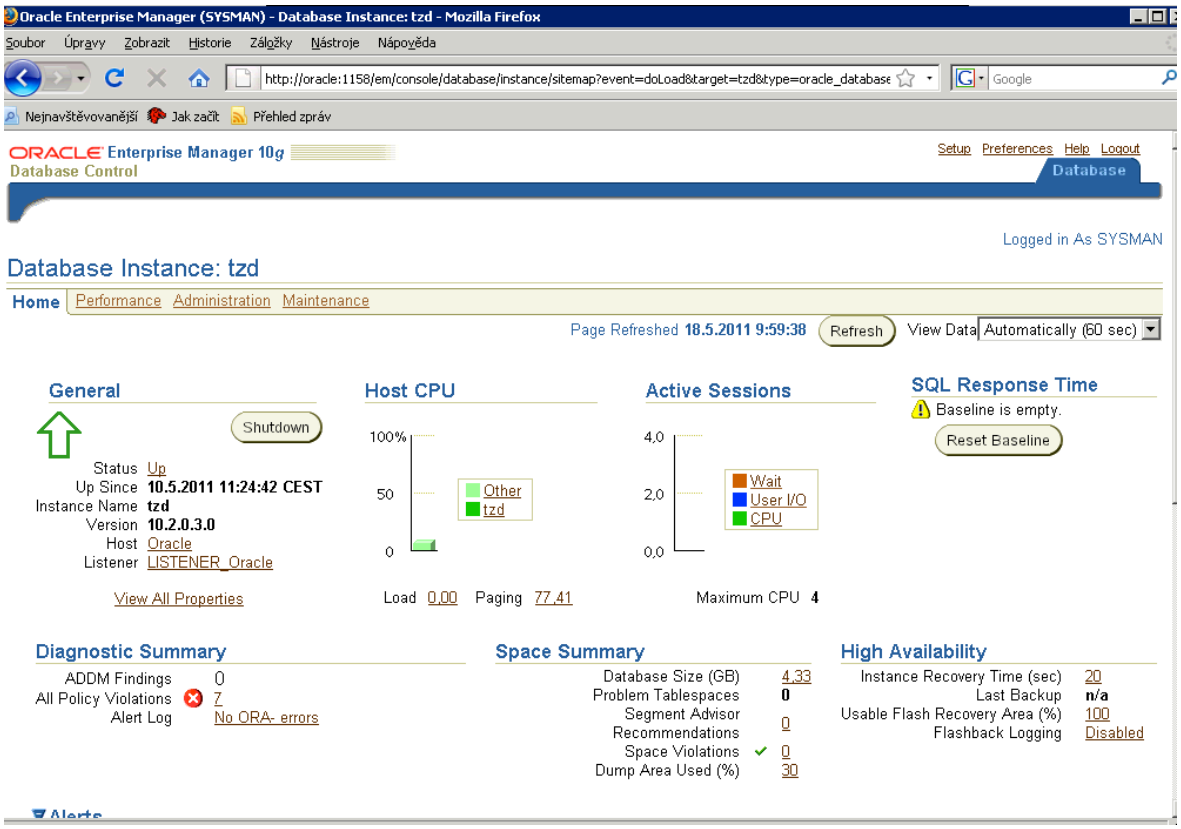
Po úspěšném přihlášení se nacházíme na hlavní stránce OEM. Táto stránka zahrnuje čtyři záložky:

- Home
- Performance
- Administration
- Maintenance (údržba)

Po přihlášení sa nacházíme v záložce Home.

### 6.1.1 Záložka Home

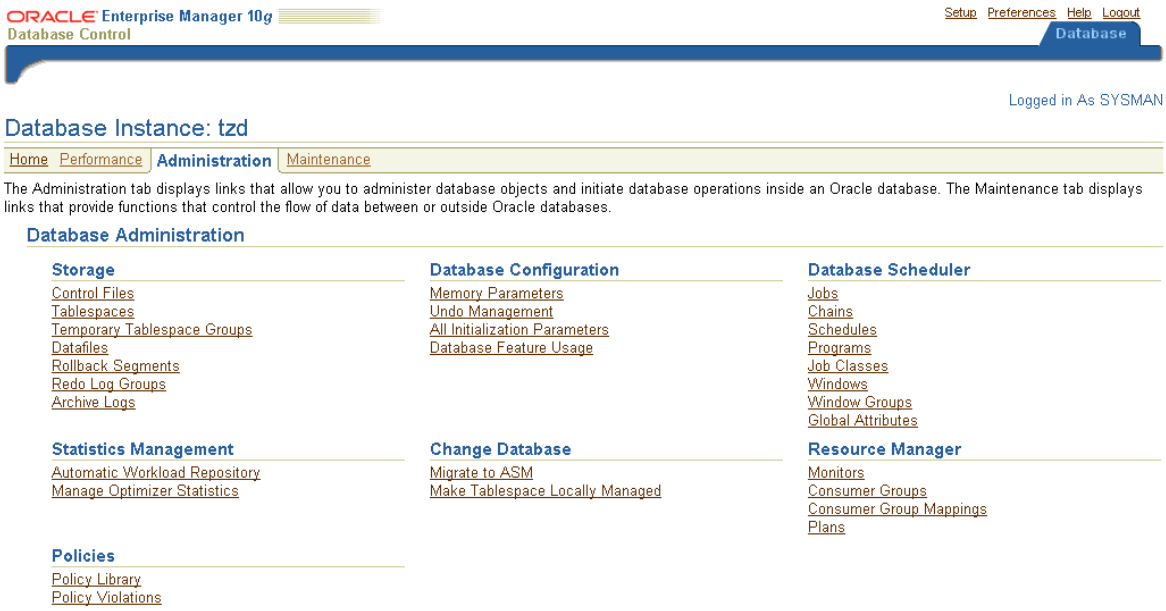
„Domovská“ stránka Oracle Enterprise Manageru umožňuje jednak zjistit základní parametry databáze a sledovat záťěž CPU a stav aktivní session. Jednotlivé informace jsou seskupené do bloků.




Obrázek 7 OEM záložka home

### 6.1.2 Záložka Administration

Na této stránce je možné nastavovat a upravovat konfiguraci databáze.

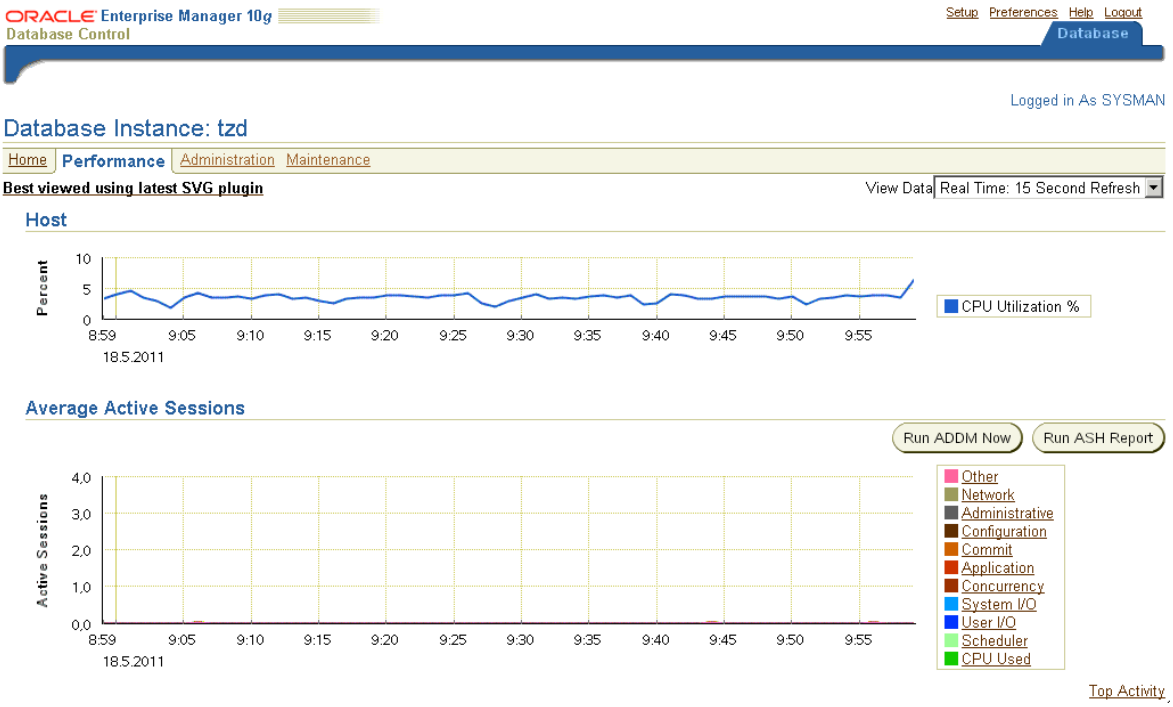



Obrázek 8 záložka administration

### 6.1.3 Záložka Performance

Na stránce Preformance jsou zobrazené tři hlavní oblasti pro přehledné grafické zobrazení časového průběhu záťěže:

- Host
- Sessions
- Instance Throughput




Obrázek 9 záložka performance

### 6.1.4 Záložka Maintenance

Funkce, které je možné aktivovat ze stránky Maintenence slouži pro export a import údajů, například mezi databází a soubory pro zobrazení, statistiky a podobně. Funkce jsou sdružené do tří bloků:

- Utilities
- Backup/Recovery
- Deployments.

V bloku Utilities jsou združené funkce:

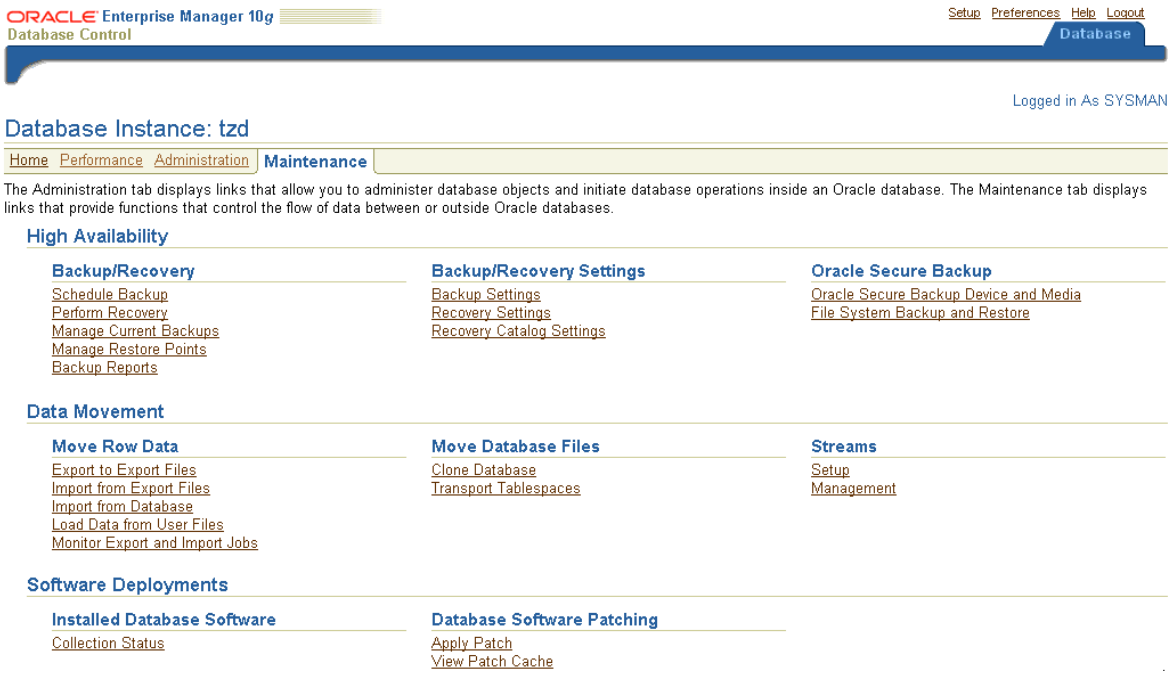
- Export to File(s)
- Import from File(s)
- Import from Database

- Load Data from File
- Gather Statistics
- Clone
- Reorganize Objects
- Make Tablespace Locally Managed

Blok Backup/Recovery obsahuje funkce:

- Schedule
- Perform Recovery
- Configure Backup
- Configure Recovery
- Manage Current
- Configure Recovery Catalog

Složka Deployments slouží pro Software Management.

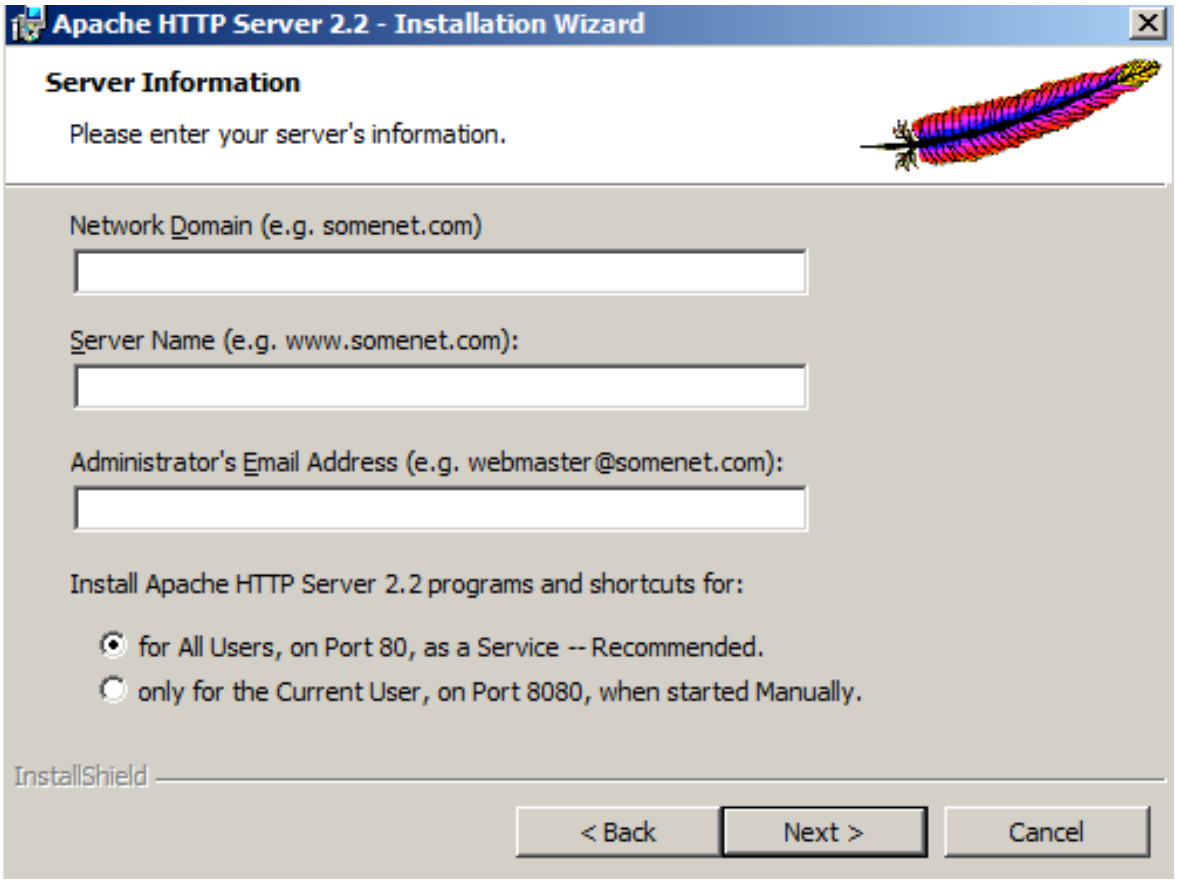



Obrázek 10 záložka maintenance

# 7 APACHE SERVER

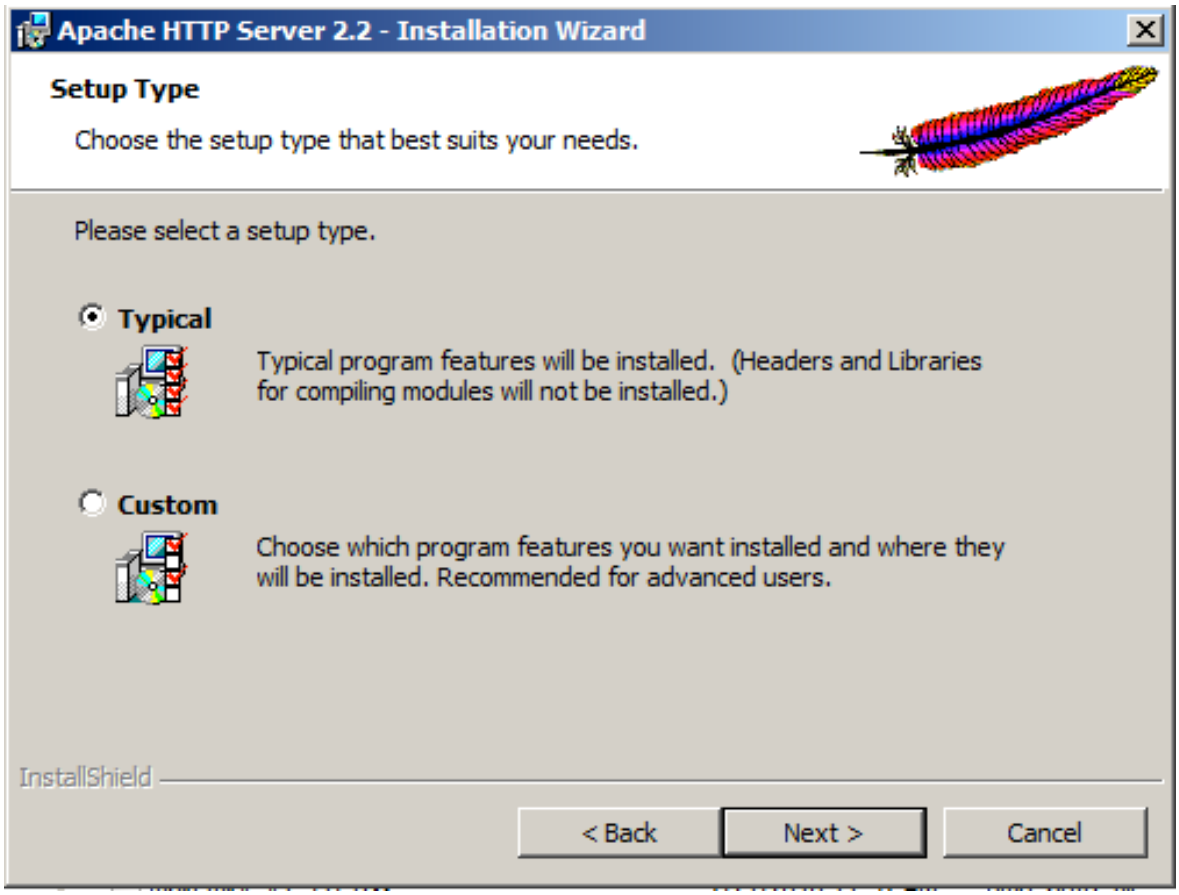
## 7.1 Instalace web Serveru Apache

Instalace Apache serveru probíhá pomoci instalátoru. Při instalaci je potřeba zadat jméno serveru a e-mailovou adresu administratora.




Obrázek 11 Nastavení instalace Apache

Při vlastní (custom) instalaci je pak možné zvolit, které součásti mají být nainstalovány. V typické instalaci se zvolí pouze umístění, kam má být Apache nainstalován.




Obrázek 12 Výběr instalace Apache

Všechna nastavení serveru Apache se dělají úpravou souboru **httpd.conf. S**oubor se obvykle nachází ve složce c:\apache\conf\. Co která direktiva, umístěná v konfiguračním souboru, znamená, se dá pochopit z komentářů a nápověd. Pomoci Apache lze vytvářet virtuální servery, přesměrování, logování a podobně.

## 7.2 Configurační soubor httpd.conf

Konfgurace serveru se provádí pomocí souboru httpd.conf. Sekce globálního nastavení (Global Environment) ovlivňuje základní vlastnosti serveru, jako jsou režim běhu, umístění konfiguračních souborů či počet souběžných dotazů, kterým server zároveň vyhoví. Při zápisu cest se i ve Windows používají klasická lomítka.

- ServerType standalone - typ serveru rozlišuje režim standalone (samostatný) nebo inetd (dostupné jen na Unix platformách).
- ServerRoot "c:/Apache" - adresář, ve kterém je Apache nainstalován. Od tohoto adresáře se odvozují všechny další relativní cesty. Uvědomte si, že nejde o místo, kde je nainstalován spustitelný soubor Apache, ale o adresář obsahující nastavení,

byť v případě Win platformy lze s úspěchem obojí držet v jednom (na Unix platformách obvykle /etc/httpd).

- PidFile "logs/httpd.pid" - soubor, do kterého si při startu server ukládá identifikační číslo svého procesu. Praktické využití je asi pouze na Unix systémech, kde se díky tomuto souboru zjišťuje číslo procesu pro restart, zaslání signálu k ukončení běhu serveru apod.
- ScoreBoardFile "logs/apache_status" - soubor, do kterého si Apache ukládá interní informace. Není to vždy zapotřebí, ale pokud se vám tento soubor vytvoří při běhu, musíte zajistit, aby se v případě vícenásobného spuštění serveru nesdílel stejný soubor. Stručně řečeno, každá instance serveru potřebuje svůj vlastní ScoreBoardFile, na což si musíte dát pozor při psaní jednotlivých konfiguračních souborů.
- Timeout 300 - počet vteřin, po jejichž uplynutí se předpokládá přerušení spojení.
- KeepAlive On - aktivuje tzv. persistentní spojení, kdy může prohlížeč při jednom spojení žádat více dotazů. Velmi to urychlí přenos, neboť klient nemusí zbytečně otevírat spojení a v jednom sledu může stáhnout jak stránku, tak i např. obrázky do stránky vložené.
- MaxKeepAliveRequests 100 - maximální počet dotazů povolených během jednoho persistentního spojení. Čím více, tím větší výkon serveru. Pokud zadáte nulu, povolíte neomezený počet dotazů na jedno spojení, ovšem otevíráte cestu k zahlcení serveru, pokud by více klientů tahalo "nekonečné" množství dotazů.
- KeepAliveTimeout 15 - jak dlouho se udržuje otevřené persistentní spojení, než dojde další dotaz. V praxi je to doba, do které musí dojít od stejného klienta na jednom spojení další dotaz, aby se vyřídil v rámci jednoho persistentního spojení.
- MinSpareServers 8 a MaxSpareServers 32 - nastavení týkající se většinou pouze Unix platformy, které určuje, kolik instancí serverů bude vyčkávat v době nečinnosti. Výsledkem je zrychlení odezvy, než kdyby se proces vytvářel teprve až na základě dotazu. Drobnou nevýhodou je zvýšení zatížení systému a odčerpání části systémových zdrojů.
- StartServers 16 - počet serverů, které se vytvoří ihned po spuštění. Důvody a vlastnosti jsou stejné, jako u předchozích direktiv MinSpareServers a MaxSpareServers.

- MaxClients 150 - maximální počet současně běžících procesů serveru, čímž se vlastně limituje i maximální počet současných spojení. Není vhodné nastavovat příliš nízké číslo, jinak při zatížení mnoho klientů buď nepříjemně "zatuhne" při čekání na spojení a nebo rovnou nahlásí chybu připojení. Na druhou stranu tato direktiva nastavuje omezení počtu procesů, po jehož překročení by se server začal přespříliš zpomalovat, takže by nakonec nedokázal např. v limitu (viz timeout) vyřizovat přicházející dotazy. Konkrétní nastavení tedy záleží na výkonu počítače, na kterém server běží.
- MaxRequestsPerChild 0 - maximum dotazů vyřízených potomkem Apache serveru. Obvykle nastaveno na nulu, což znamená bez omezení, ale u platforem, kde dochází k únikům paměti, je to cesta, jak po určité době proces zrušit, uvolnit jím okupovanou paměť a založit nový proces.
- ThreadsPerChild 50 - omezení maximálního počtu programových "vláken", která vyřizují dotazy. Čím více povolíte, tím více současných požadavků se vyřizuje. O to více se ale zatíží systémové zdroje a vlastní zpracování se vzhledem k multitaskingu odpovídajícím způsobem zpomalí.
- Listen 300 nebo Listen 127.0.0.1:8080 - pro začátek ne moc důležitá direktiva, která určuje, na kterých adresách či portech bude server naslouchat přicházejícím dotazům. První varianta určuje pouze port, druhá přímo IP adresu včetně portu. Využití najde spíše na systémech s více síťovými kartami či adresami. Názorným příkladem může být aktivace podpory SSL, Na okraj - tohle je direktiva, kterou se aktivuje v případě podpory SSL, aby server naslouchal i na portu pro https://, což dosáhnete přidáním direktiv <IfDefine SSL> Listen 443 </IfDefine> - tuto část je nutné zadat až za sekci nahrávající rozšiřující moduly (DSO). Navíc je zde vidět i jedna z direktiv "podmíněných", která umožňuje zpracování konfiguračního souboru, podle zjištěného prostředí. Další podobnou direktivou je například <IfModule>.
- ExtendedStatus Off - definuje, zda má server generovat úplný přehled svého stavu, pokud se použije "handler" *server-status* nebo *server-info*.
- BindAddress * nebo BindAddress 192.168.0.1 nebo BindAddress apache.server.net - určuje, na kterých adresách Apache server naslouchá. Tato direktiva je důležitá, pokud chcete provozovat více virtuálních serverů na jednom počítači. Uvedením hvězdičky určujete, že se přijímají požadavky ze všech dostupných IP, které jsou na

příslušném počítači nakonfigurovány. Jinak lze specifikovat vybranou IP adresu a nebo plnohodnotné jméno serveru (přeložitelné přes DNS server na IP adresu).

- Dynamic Shared Objects (DSO) - prostor, ve kterém lze k vlastnímu serveru nahrát přídavné moduly, které zajišťují rozšíření funkcí.[12]

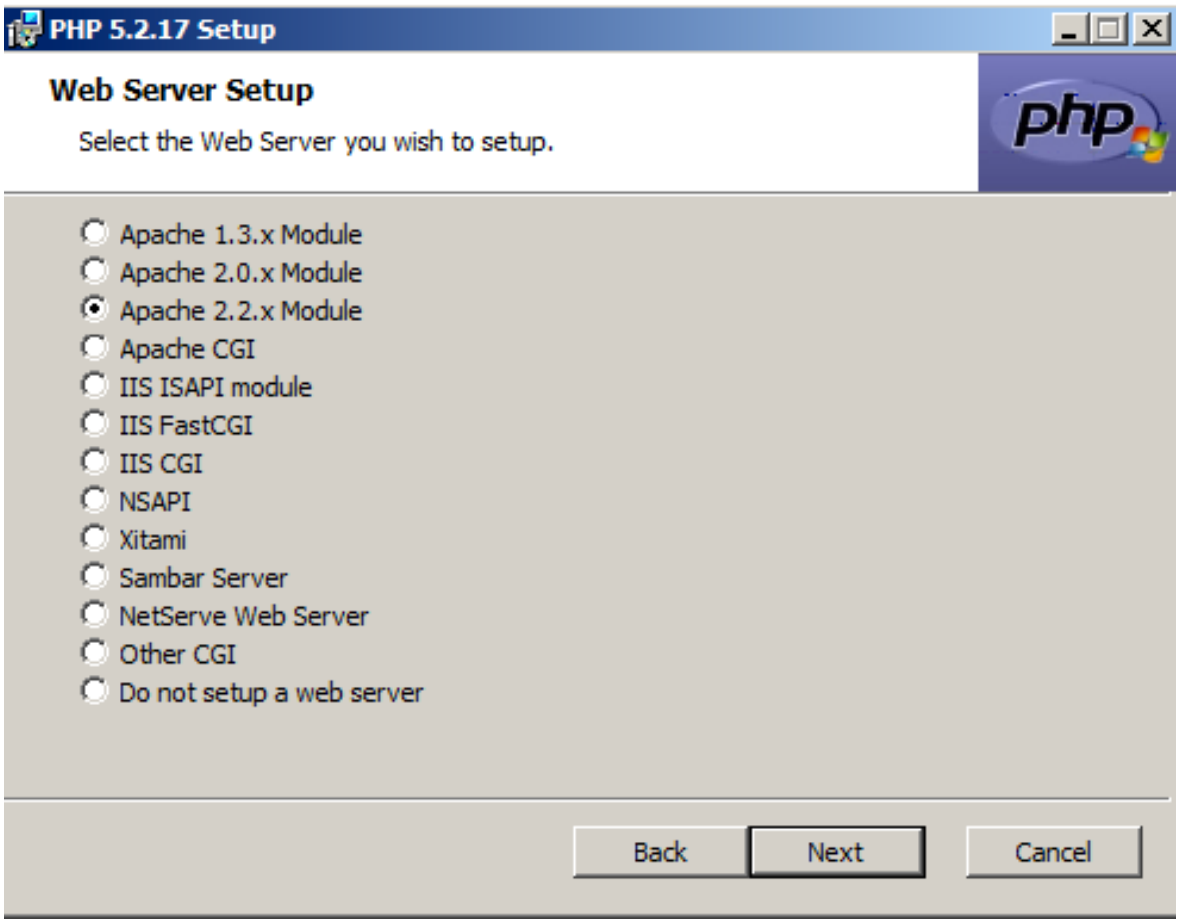
Konfigurační soubor je umístěn v C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\httpd.conf a musí obsahovat následující řádky a nastavení:

- PHPIniDir "C:/PHP"
- LoadModule php5_module "c:/PHP/php5apache2_2.dll"
- DirectoryIndex index.php index.html

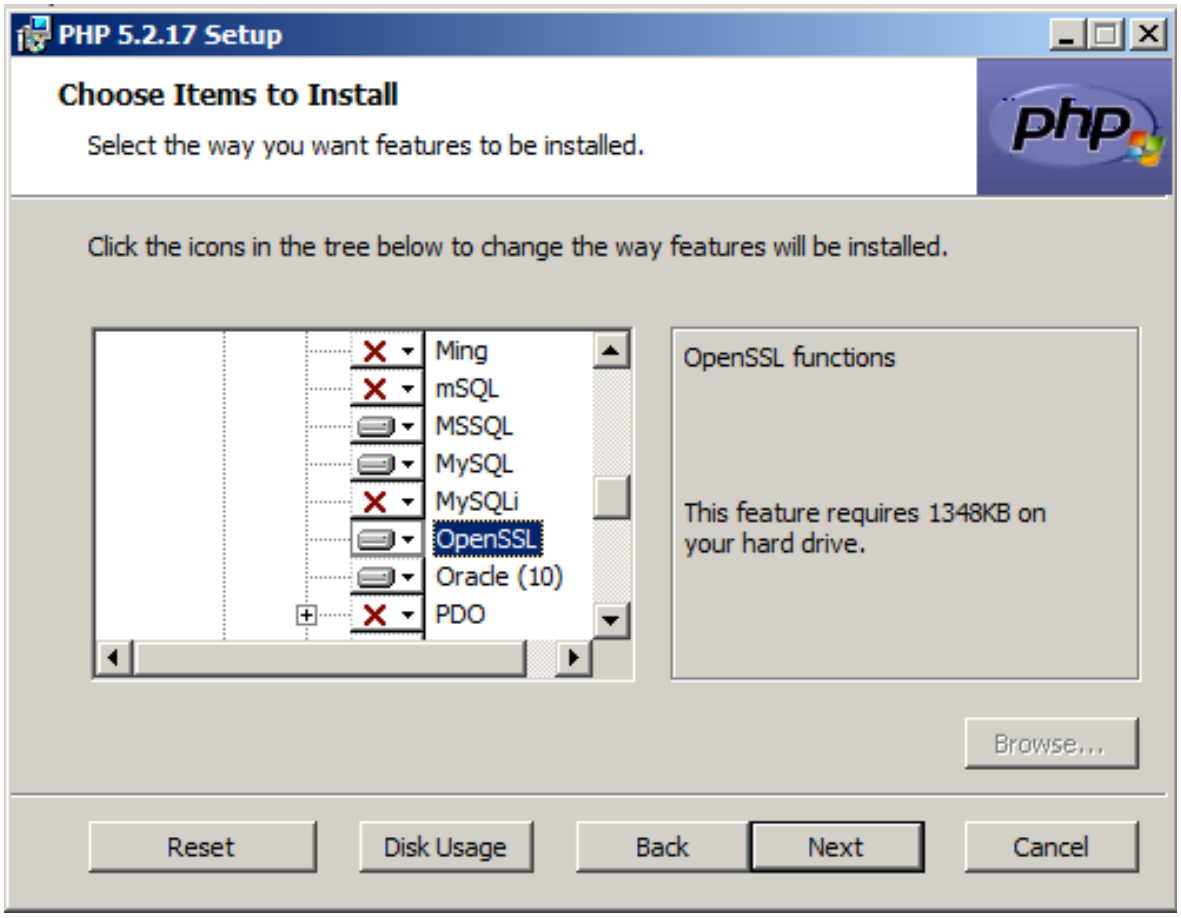
# 8 PHP

## 8.1 Instalace PHP

Při instalaci PHP je potřeba zvolit web server, pro který je instalován a následné moduly, které mají být nainstalovány.




Obrázek 13 Instalace PHP




Obrázek 14 Váběr modulů k instalaci

## 8.2 Configurační soubor php.ini

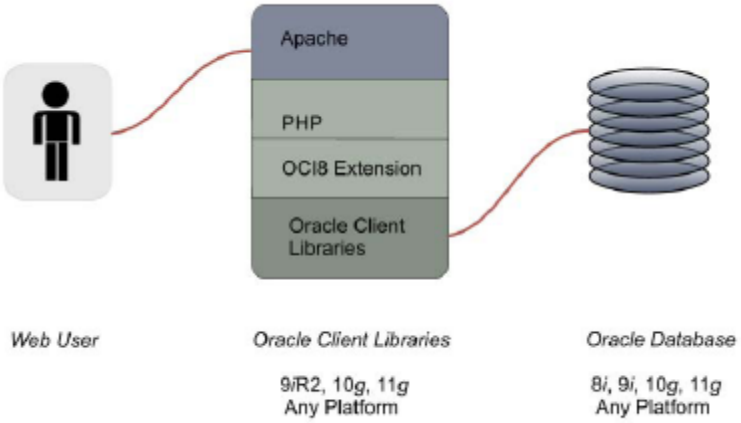
Konfigurační soubor je standardně umístěn v C:\PHP\php.ini

V konfiguračním souboru jsem upravil následující:

- display_errors = On – při spuštění skriptu s chybou se nezobrazí prázdná stránka
- short_open_tag = On – pokud je nastaveno Off, tak pokud je ve skriptu <? ignoruje a funguje pokud je uvedeno <?php
- Register_globals = on - nastavení na off, znemožní přebírání proměnných z adresy ve tvaru $proměnná

# 9 ORACLE CALL ITERFACE

Oracle Call Interface ( OCI) je aplikační rozhraní (API), pomocí něhož lze programovat externí moduly pro Oracle v jazycích C, C++, Java, Rubby, PHP, Python, Perl, .Net, VB a dalších. Funkce OCI pokrývají téměř veškerou funkčnost poskytovanou databázovým serverem Oracle a to včetně zpracování SQL příkazů a manipulaci s objekty.




Obrázek 15 PHP a Oracle

Přístup k databázovému serveru Oracle pomocí OCI nabízí celou řadu výhod, mezi které patří např.:

- Možnost detailnějšího řízení všech stránek návrhu aplikace, která je možná díky existenci mnoha vestavěných funkcí, jež umožňují rozdělit jediný krok v jiných prostředích do řady na sebe navazujících volání funkcí OCI.
- Vysoký stupeň kontroly provádění programu, což je dáno tím, že primárním prostředím je např. C/C++, které svými prostředky pro řízení toku programu a řízení využití zdrojů vytváří kostru, do níž jsou zasazena jednotlivá volání funkcí OCI.
- Vysoká efektivita práce programátora, který při práci používá standardní vývojové prostředí, na něž je zvyklý.
- Snadný přístup k metadatům Oracle serveru
- Dostupnost dynamického SQL.
- Asynchronní zpracování příkazů. V tomto případě se po vyvolání OCI funkce vrací aplikaci řízení zpět ještě před dokončením příkazu (tzv. non-blocking mode,) a funkce pak vracejí informaci, zda je příkaz ještě prováděn. Toto může být výhodné

např. při provádění dlouhotrvajících operací, kdy aplikace místo "vytuhnutí" průběžně informuje uživatele o průběhu zpracování.

- A další ....

Funkcionalitu OCI lze rozdělit do pěti hlavních skupin:

- API pro vytváření vícevláknových aplikací
- Funkce pro zpracování SQL příkazů a manipulaci se získanými objekty
- Mapování datových typů a funkcí pro manipulaci s atributy Oracle datových typů
- Funkce pro zpřístupnění dat z databáze bez použití SQL
- funkce externích procedur, které zapisují zpětná volání (callbacky) z PL/SQL

Funkce OCI, určené pro komunikaci databázového stroje s PHP, jsou dostupné v modulu (rozšíření) nesoucí název **php_oci8**. Toto rozšíření poskytuje celou řadu funkcí pro práci s databází. Mezi nejdůležitější funkce patří:

*oci_connect()*

poskytuje připojení do databáze konkrétnímu uživateli.

*oci_parse()*

umožňuje připravit příkaz jazyka SQL, který bude v databázi spuštěn. Tento příkaz může při spuštění požadovat parametry.

*oci_execute()*

umožňuje vykonat připravený příkaz jazyka SQL v databázi.

oci_close()

ukončení spojení s databází

## 9.1 OCI pro spojení s databází

Otevření spojení na databázi Oracle

```
$c=oci_connect($username,$password, $dbname, $charset);
```

Uzavření spojení

```
oci_close($c);
```

Změna hesla

```
oci_password_change($c, $username, $oldPassword, $newPassword);
```

## 9.2 Vykonání dotazu select

Parse - rozebrání (parsování) příkazu pro vykonání

Bind –přiřazení dat do dotazu pro lepší výkon a bezpečnost

Execute –vykonání dotazu

Fetch – načtení a vrácení výsledku dotazu z databáze

```
$c = oci_connect("hr", "hrpwd", "localhost/XE");
$s = oci_parse($c, 'select city, postal_codefrom location');
oci_execute($s);
print '<table border="1">';
while ($row = oci_fetch_array($s, OCI_NUM+OCI_RETURN_NULLS))
{
print '<tr>';
foreach($row as $item)
print '<td>'.htmlentities($item).'</td>';
print '</tr>‘;
}
print '</table>';
oci_free_statement($s);
```

## 9.3 Vrácení výsledku dotazu

oci_fetch_all()–vrátí výsledek najednou

oci_fetch_array() –vrátí výsledek v poli

oci_fetch_assoc()–vrátí výsledek v asociativním poli

oci_fetch_object()–vrátí výsledek jako objekt

oci_fetch_row() –vrátí výsledek v indexovaném poli

```
$rowarray= oci_fetch_array($statement, $mode);
```

## 9.4 Příkazy Insert, Update, Delete, Create a Drop

Pro vykonání Data Definition Language (DDL) and Data Manipulation Language (DML) dotazů, jako například CREATE, INSERT,atd. je nejjednodušší použít příkaz pro parsování SQL dotazu (oci_parse) a jeho následné vykonání (oci_execute).

```
$s = oci_parse($conn, "create table i1test (col1 number)");
oci_execute($s);
```

## 9.5 Transakce

Transakce je logická jednotka zpracování dat, která se skládá z jednoho nebo více SQL příkazů provedených jedním uživatelem.

Transakce končí buď promítnutím změn do databáze (commit) nebo vrácením databáze do stavu, v jakém se nacházela před transakcí (rollback).

```
$c = oci_connect('hr', 'hrpwd', 'localhost/XE');
$s = oci_parse($c, "insert into mytablevalues ('abc')");
oci_execute($s, OCI_DEFAULT); // bez provedení commitu
$s = oci_parse($c, "begin updatelog('INSERT attempted');end;");
oci_execute($s, OCI_DEFAULT); // bez provedení commitu
oci_rollback($c); // vrácení databáze do původního stavu
```

## 9.6 Obsluha chyb

Obsluha chyb je v knihovně OCI řešena funkcí oci_error(). Funkce vyžaduje různé argumenty závisející na kontextu předešlého volání.Návratová hodnota funkce je pole.

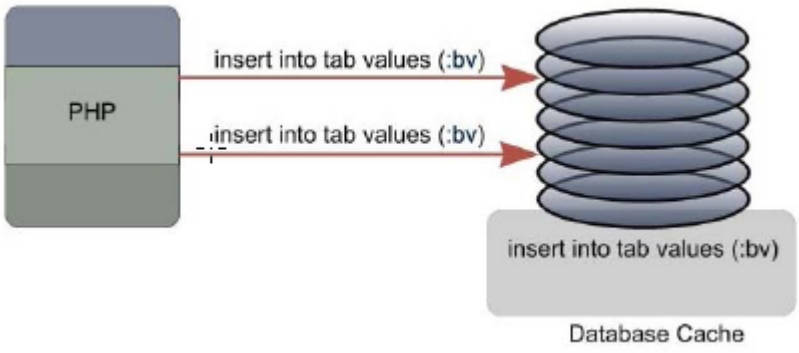
Prvky pole vrácen funkcí oci_error():

```
$e["code"] –číslo chyby
$e["message"] –chybová zpráva
```

```
$e["offset"] –pozice chyby v SQL dotazu
$e["sqltext"] –SQL dotaz
$c = oci_connect("hr", "not_hrpwd", "localhost/XE");
if (!$c)
{
$e = oci_error();
var_dump($e);
}
$s = oci_parse($c, "select city from locations");
$rc= oci_execute($s);
if (!$rc)
{
$e = oci_error($s);
var_dump($e);
}
$rc= oci_fetch_all($s, $results);
if (!$rc)
{
$e = oci_error($s);
var_dump($e);
 }
```

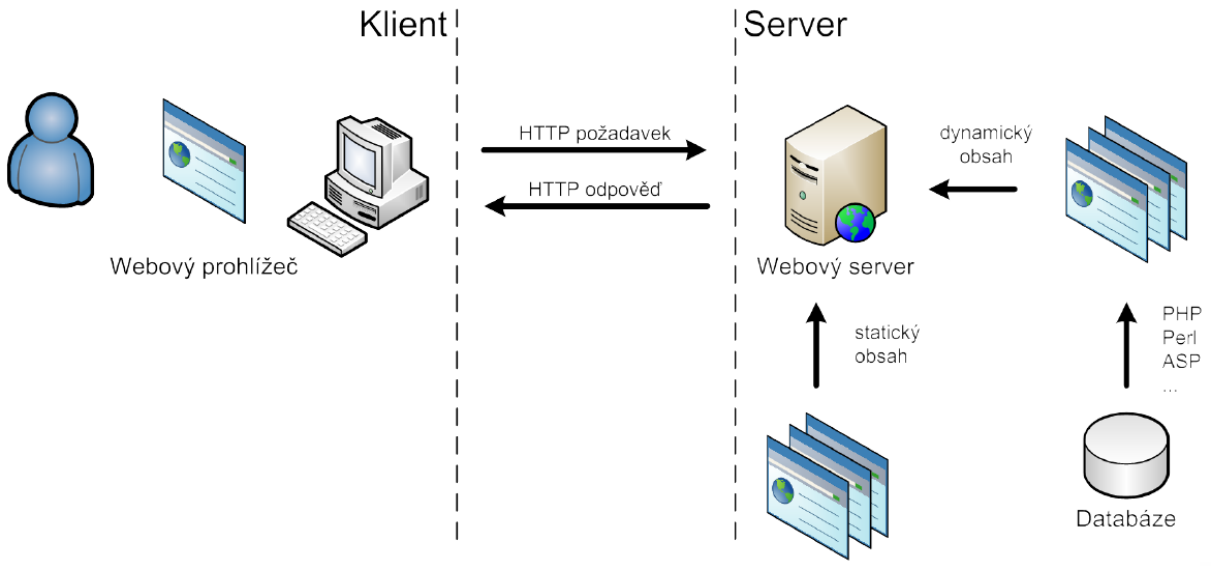
## 9.7 Přiřazení dat

Binding je doporučovaným prostředkem pro cachování dotazu na databázovém serveru a znovupoužití exekučních plánů a také způsob ochrany dotazů proti SQL injection.




Obrázek 16 PHP – Databáze [15]

```
$s = oci_parse($c, "select last_namefrom employees where employee_id= :eidbv");
$myeid= 101;
oci_bind_by_name($s, ":eidbv", $myeid);
oci_execute($s);
$row = oci_fetch_array($s, OCI_ASSOC);
echo "Last name is: ". $row['LAST_NAME'] ."<br>\n";
```



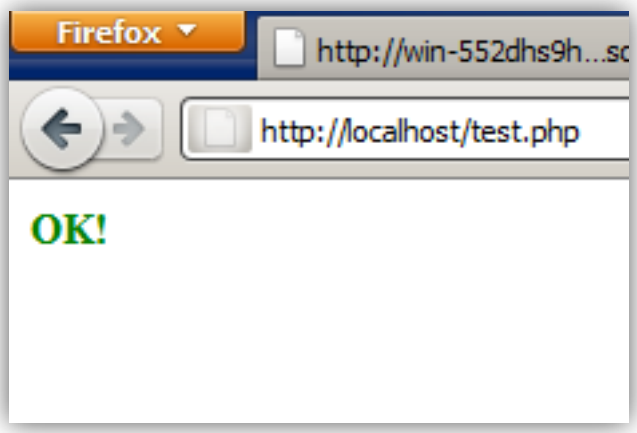

Obrázek 17 Komunikační schéma webové aplikace [15]

# 10 PŘIPOJENÍ K DATABÁZI

Přístup do Oracle databáze z PHP je zprostředkován prostřednictvím extenzí PHP, které komunikují s Oracle client libraries (OCI).

Po nainstalování a nastavení všech aplikací jsem otestoval spojení s databází pomoci skriptu:

```
<?php
$conn = oci_connect('system', 'admin', 'localhost/database');
if (!$conn) {
trigger_error("Could not connect to database", E_USER_ERROR);
}
else {
   echo '<b><font color="green">OK!</font></b>';
   oci_close($conn);
}
?>
```

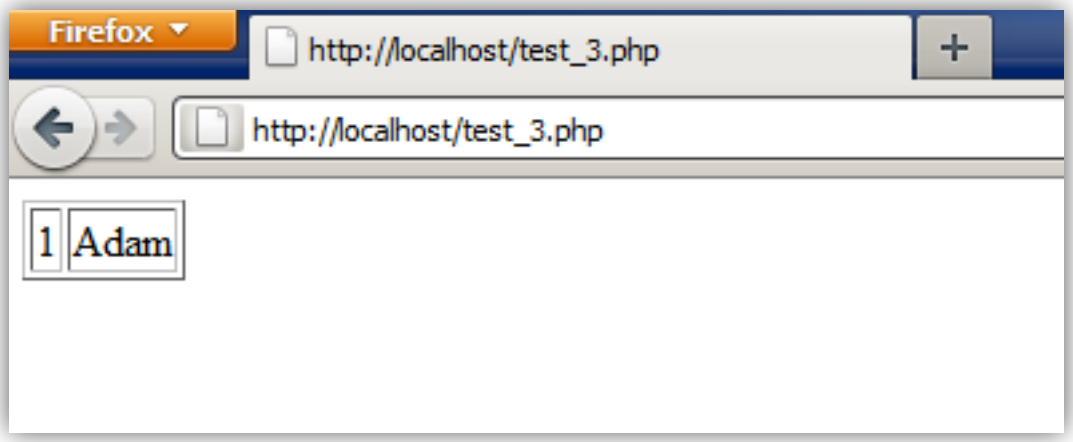



Pro otestování vkládání a čtení dat z databáze jsem použil skript:

```
<?php
$conn = oci_connect("uzivatel", "heslo", "localhost/nazev_databaze");
if (!$conn) {
$m = oci_error();
```

```
echo $m["message"];
exit;
}
// Vlozeni do tabulky
$stid = oci_parse($conn, "insert into tabulka values (:i_b, :v_b)");
$id = 1;
$val = "Adam";
oci_bind_by_name($stid, ":i_b", $id);
oci_bind_by_name($stid, ":v_b", $val);
oci_execute($stid);

$stid = oci_parse($conn, "select * from tabulka");
oci_execute($stid);

// vypis tabulky
echo "<table border='1'>";
while ($row = oci_fetch_array($stid, OCI_ASSOC+OCI_RETURN_NULLS)) {
echo "<tr>";
foreach ($row as $item) {
echo "<td>".($item!==null?htmlentities($item):" ")."</td>";
}
echo "</tr>".PHP_EOL;
}
echo "</table>";

// Smazani radku
$stid = oci_parse($conn, "delete from tabulka where id = :i_b");
$id = 1;
oci_bind_by_name($stid, ":i_b", $id);
oci_execute($stid);

oci_free_statement($stid);
```

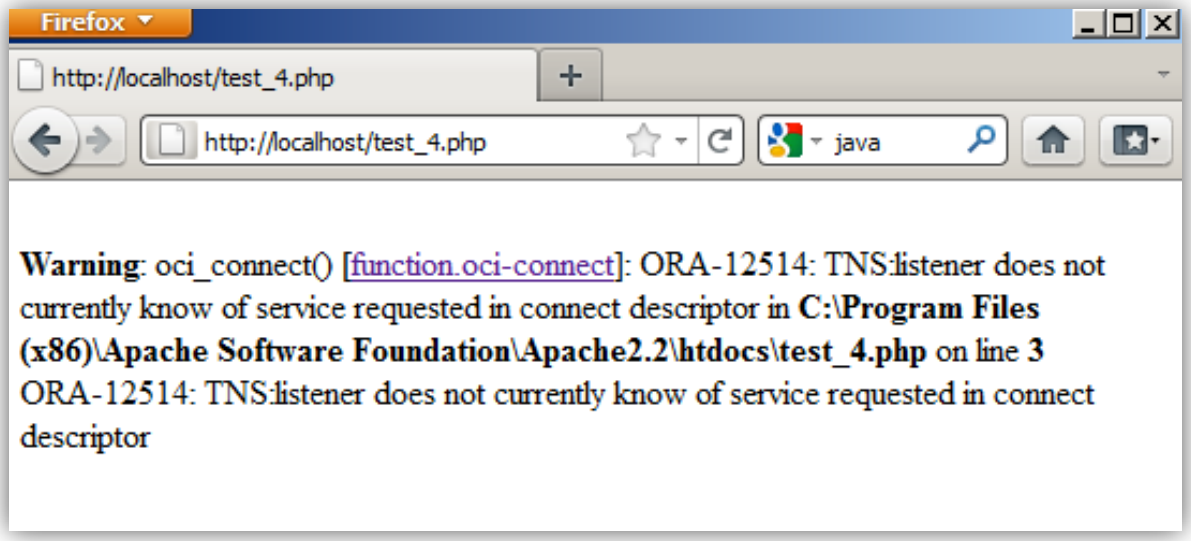
```
oci_close($conn);
?>
```




Obrázek 18 Test zápisu do databáze

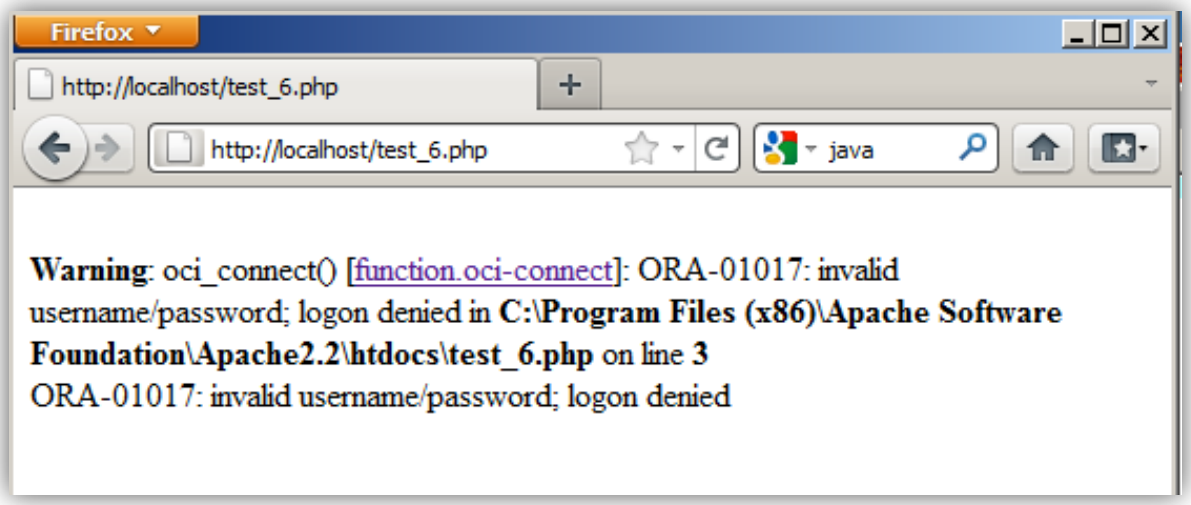
Na následujících obrázcích je ukázáno, jak by vypadal výstup v prohlížeči pokud by se:

- Nezdařilo připojení k databázi (bylo špatně zadáno jméno databáze)



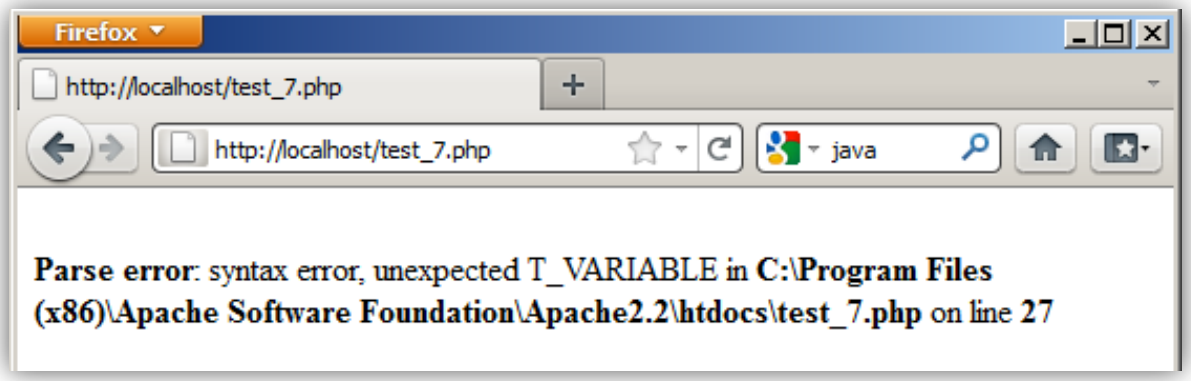

Obrázek 19 Chybové hlášení špatně zadaného jména databáze

- Spojit s databází se podařilo, ale bylo odmítnuto jméno nebo heslo




Obrázek 20 Chybové hlášení špatně zadaného hesla

- Při vykonáváni skriptu objevila chyba v syntaxi




Obrázek 21 Špatná syntaxe

## 10.1 SQL PLUS

Oracle SQL Plus je nástroj pro příkazový řádek, který umožňuje uživateli zadávat příkazy SQL přímo do databáze Oracle.

SQL Plus umožňuje uživateli:

- Zadávat, upravovat, ukládat, načíst a spustit SQL příkazy
- Zobrazit seznam sloupců a definic pro libovolné tabulky
- Formátovat, ukládat a tisknout výsledky dotazu ve formě zpráv
- Přístup a kopírování dat mezi databázemi SQL

Základní příkazy pro SQL Plus jsou tyto:

- ALTER - Změna existující tabulky, zobrazení nebo definice indexu
- AUDIT - Sledování změn provedených v tabulce
- COMMENT - Přidat komentář ke stolu nebo sloupec v tabulce
- COMMIT - Proveď všechny nedávné změny trvale
- CREATE - Vytvořit nové databázové objekty, jako jsou třeba tabulky
- DELETE - Odstranění řádků z databázové tabulky
- GRANT - Povolit jinému uživateli přístup do databáze objektů
- INSERT - Vložení nových dat do databázové tabulky
- No AUDIT - Vypnutí funkce kontroly
- REVOKE - Zakázat přístup uživatelů k databázi a objektům
- ROLLBACK – Vrátit poslední změny do databáze
- SELECT - Načtení dat z tabulky databaze
- UPDATE - změna hodnoty některé datové položky v tabulce databaze

# ZÁVĚR

Cílem mé této diplomové práce bylo propojení databáze Oracle s scriptovacím jazykem PHP na Windows 2008 Server v 64 bitové verzi.

Teoretickým i praktickým způsobem jsem se snažil popsat fungování jak databázového systému Oracle na operačním systému Windows 2008, tak i skriptovacího jazyka PHP. Detailně jsou rozebrány a popsány nástroje a principy, které jsou nutné ke komunikaci mezi jednotlivými aplikacemi. K dispozici jsou následující možnosti: použití instalačního balíku XAMPP (balíček předkonfiguravaných instalací PHP a dalšího souvisejícího softwaru), použití IIS (Internet Information Server) a samostatné instalace webového serveru Apache a PHP. Vzhledem k tomu, že instalační balíček XAMPP obsahuje, také instalaci MySQL, SQLLite, FileZilla FTP server a další aplikace, které by nebyly použity a IIS je platformě závislá aplikace zvolil jsem z nabízených řešení samostatnou instalaci Apache, neboť se jedná o rozšířený, velmi populární a platformě nezávislý webový server s otevřeným kódem.

V teoretické části práce jsem se snažil nastínit fungování a vývoj databází a operačních systémů od jejich počátku až po současnost. Dále se zde zabývám charakteristikou aplikací použitých v praktické částí této práce.

V rámci praktické části jsem nainstaloval virtuální server, na který jsem nainstaloval všechny aplikace, které jsou potřebné k úspěšnému spuštění databázového serveru a jeho propojení se skriptovacím jazykem PHP. V postupu nastavování byl detailně popsán a zdůvodněn každý jednotlivý krok konfigurace a nastavení parametrů a odstranění chyb, které mohou v průběhu vzniknout. Po důkladném otestování propojení mezi databází a PHP na virtuálním serveru a po odzkoušení základních skriptů pro čtení a zápis do databáze jsem řešení včetně všech zvolených nastavení aplikoval na školním serveru, kde toto řešení bude sloužit jako pomůcka k výukovým účelům pro studenty pro vytváření PHP skriptů a pro seznámení se se základním funkcí databáze Oracle a webového serveru Apache.

# CONCLUSION

The main aim of this diploma thesis was connection Oracle database with PHP script language on Windows 2008 Server in 64 bit version.

Not only I tried to describe operation Oracle database system on Windows 2008 operation system but PHP script language as well, both theoretically and practically. Instruments and principals have been dissected and described which are important communication among various applications. Available are following options: use of installation package XAMPP (package of preconfigured installations PHP and other connected software). Use of IIS (Internet Information System) and independent installations of web server Apache and PHP. Whereas XAMPP installation package contains also installations of MySQL, SQLLite, FileZilla FTP server and other applications which would not be used and IIS is platform depending application, I chose from offered solutions independent installation Apache, because it is widely extended, popular and platform independent web server with open source code.

In theoretical part I tried outline operation and development of databases and operation systems from their beginning till the present. Further I occupied with characteristics of applications used in practical part of this thesis.

Within the practical part I installed virtual server, on which I installed all applications, which were necessary for successful initiation of database server and its connection with script language PHP. In the process of installation was detailed described and substantiated each step of configuration and set-up of parameters and failures elimination, which could have occurred. After careful testing connection between database and PHP on virtual server and after testing basic scripts for reading and record to database I applied solutions including all selected set-ups on university server where this solution will be used as an instrument for educational purposes for creation PHP scripts and for orientation and familiarization with basic functions of Oracle database and Apache web server.

# SEZNAM POUŽITÉ LITERATURY

[1] *Http://www.osdata.com/kind/history.htm* [online]. 2001 [cit. 2011-05-17]. Dostupné z WWW: http://www.osdata.com

[2] *Http://www.computinghistorymuseum.org* [online]. 2001 [cit. 2011-05-17]. Http://www.computinghistorymuseum.org/teaching/papers/research/history_of_operating_system_Moumina.pdf. Dostupné z WWW: http://www.computinghistorymuseum.org

[3] *Http://www.mujmac.cz* [online]. 2006 [cit. 2011-05-17]. Http://www.mujmac.cz/art/polemiky/historie-operacnich-systemu-win-unix-macosx.html. Dostupné z WWW: <http://www.mujmac.cz>.

[4] *Http://www.fi.muni.cz* [online]. 1999 [cit. 2011-05-17]. Http://www.fi.muni.cz/usr/jkucera/pv109/xklika-history.html. Dostupné z WWW: http://www.fi.muni.cz

[5] *Http://www.abclinuxu.cz* [online]. 2009 [cit. 2011-05-17]. Http://www.abclinuxu.cz/ucebnice/historie/historie-unixu). Dostupné z WWW: http://www.abclinuxu.cz

[6] *Http://airborn.webz.cz* [online]. 2007 [cit. 2011-05-17]. Http://airborn.webz.cz/histos.html. Dostupné z WWW: http://airborn.webz.cz

[7] *Http://martin.hinner.info* [online]. 2001 [cit. 2011-05-17]. Http://martin.hinner.info/articles/unix.pdf. Dostupné z WWW: http://martin.hinner.info

[8] *Http://www.mujmac.cz* [online]. 2006 [cit. 2011-05-17]. Http://www.mujmac.cz/art/polemiky/historie-operacnich-systemu-win-unix-macosx.html. Dostupné z WWW: http://www.mujmac.cz

[9] Windows Server 2008 R2. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-05-17]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Windows_Server_2008_R2

[10] Apache HTTP Server. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-05-18]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Apache_HTTP_Server

[11] *ttp://www.webtvorba.cz* [online]. 2003 [cit. 2011-05-18]. Http://www.webtvorba.cz/php/uvod-do-php.html. Dostupné z WWW: http://www.webtvorba.cz

[12] PHP. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-05-18]. Dostupné z WWW: http://cs.wikipedia.org/wiki/PHP

[13] *Http://milanc.chvalsiny.net* [online]. 2008 [cit. 2011-05-18]. Http://milanc.chvalsiny.net/apache-server-konfigurace/. Dostupné z WWW: http://milanc.chvalsiny.net

[14] POKORNÁ, Dagmar. *Řešení integrace databázových systémů* [online]. Zlín : Bakalářská práce zveřejněná UTB, 2007. 71 s. Bakalářská práce. UTB. Dostupné z WWW: <https://dspace.knihovna.utb.cz/handle/10563/3428>.

[15] Http://jls.webz.cz [online]. 2007 [cit. 2011-05-17]. Http://jls.webz.cz/mff/oracle/oci.html. Dostupné z WWW: http://jls.webz.cz

[16] STANEK, William. Microsoft Windows Server 2008 : Kapesni rSdce administrStora. Ondiej Ba5e, Rostistav Cibulka, Petr Setka, PavelVaida.l.vyd.[s.l.l : Computer Press, 2008. 7A4 s.lSBN 978-80-251-1935-5.

[17] LACKO, Luboslav. Oracle : SprSva, programovSni a pouZiti datab5zov6ho syst6mu. [s.l.l : Computer Press, 2007.575 s.|SBN 978-80-251'1490'2.

[18] LONEY, Kevin , BRYLA, Bob. Mistrovstvi v Oracle Database 1lg. Is.lJ : Computer Press, 2009. 7O4 s.|SBN 978-80'751'2189-4.

[19] CRAWFORD, Sharon, RUSSEL, Charlie. Microsoft Windows Server 2008 : Velkf pr0vodce administrStora. [s.l.l : Computer Press, 2009. 1272 s. ISBN 978-80-251-2fl5-3.

[20] MCLAUGHLIN, Michael, HARDMAN, Ron, URMAN, Scott. Oracle: Programov5ni v PL/SQL. [s.l.l: Computer Press,2008.72O s.lSBN 9788025118702.

[21] STANEK, William. Mistrovstvi v Microsoft Windows Server 2008. [s.l.l : [s.n.l, 2009. 1358 s. ISBN 978-80-251-2158-0.

[22] COREY, Michael, ABBEY, Michael, ABRAMSON,Ian. Oracle Database 1lg, A Beginners Guide. [s.l.l : [s.n.l, 2008. 432 s.

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# SEZNAM PŘÍLOH

# PŘÍLOHA P I: SEZNAM FUNKCÍ OCI8

- oci_bind_array_by_name — Binds a PHP array to an Oracle PL/SQL array parameter
- oci_bind_by_name — Binds a PHP variable to an Oracle placeholder
- oci_cancel — Cancels reading from cursor
- oci_close — Closes an Oracle connection
- OCI-Collection->append — Appends element to the collection
- OCI-Collection->assign — Assigns a value to the collection from another existing collection
- OCI-Collection->assignElem — Assigns a value to the element of the collection
- OCI-Collection->free — Frees the resources associated with the collection object
- OCI-Collection->getElem — Returns value of the element
- OCI-Collection->max — Returns the maximum number of elements in the collection
- OCI-Collection->size — Returns size of the collection
- OCI-Collection->trim — Trims elements from the end of the collection
- oci_commit — Commits the outstanding database transaction
- oci_connect — Connect to an Oracle database
- oci_define_by_name — Associates a PHP variable with a column for query fetches
- oci_error — Returns the last error found
- oci_execute — Executes a statement
- oci_fetch_all — Fetches multiple rows from a query into a two-dimensional array
- oci_fetch_array — Returns the next row from a query as an associative or numeric array
- oci_fetch_assoc — Returns the next row from a query as an associative array
- oci_fetch_object — Returns the next row from a query as an object
- oci_fetch_row — Returns the next row from a query as a numeric array
- oci_fetch — Fetches the next row from a query into internal buffers
- oci_field_is_null — Checks if the field is NULL
- oci_field_name — Returns the name of a field from the statement
- oci_field_precision — Tell the precision of a field
- oci_field_scale — Tell the scale of the field
- oci_field_size — Returns field's size

- oci_field_type_raw — Tell the raw Oracle data type of the field
- oci_field_type — Returns field's data type
- oci_free_statement — Frees all resources associated with statement or cursor
- oci_internal_debug — Enables or disables internal debug output
- OCI-Lob->append — Appends data from the large object to another large object
- OCI-Lob->close — Closes LOB descriptor
- oci_lob_copy — Copies large object
- OCI-Lob->eof — Tests for end-of-file on a large object's descriptor
- OCI-Lob->erase — Erases a specified portion of the internal LOB data
- OCI-Lob->export — Exports LOB's contents to a file
- OCI-Lob->flush — Flushes/writes buffer of the LOB to the server
- OCI-Lob->free — Frees resources associated with the LOB descriptor
- OCI-Lob->getBuffering — Returns current state of buffering for the large object
- OCI-Lob->import — Imports file data to the LOB
- oci_lob_is_equal — Compares two LOB/FILE locators for equality
- OCI-Lob->load — Returns large object's contents
- OCI-Lob->read — Reads part of the large object
- OCI-Lob->rewind — Moves the internal pointer to the beginning of the large object
- OCI-Lob->save — Saves data to the large object
- OCI-Lob->saveFile — Alias of oci_lob_import
- OCI-Lob->seek — Sets the internal pointer of the large object
- OCI-Lob->setBuffering — Changes current state of buffering for the large object
- OCI-Lob->size — Returns size of large object
- OCI-Lob->tell — Returns the current position of internal pointer of large object
- OCI-Lob->truncate — Truncates large object
- OCI-Lob->write — Writes data to the large object
- OCI-Lob->writeTemporary — Writes a temporary large object
- OCI-Lob->writeToFile — Alias of oci_lob_export
- oci_new_collection — Allocates new collection object
- oci_new_connect — Connect to the Oracle server using a unique connection
- oci_new_cursor — Allocates and returns a new cursor (statement handle)
- oci_new_descriptor — Initializes a new empty LOB or FILE descriptor

- oci_num_fields — Returns the number of result columns in a statement
- oci_num_rows — Returns number of rows affected during statement execution
- oci_parse — Prepares an Oracle statement for execution
- oci_password_change — Changes password of Oracle's user
- oci_pconnect — Connect to an Oracle database using a persistent connection
- oci_result — Returns field's value from the fetched row
- oci_rollback — Rolls back the outstanding database transaction
- oci_server_version — Returns server version
- oci_set_action — Sets the action name
- oci_set_client_identifier — Sets the client identifier
- oci_set_client_info — Sets the client information
- oci_set_edition — Sets the database edition
- oci_set_module_name — Sets the module name
- oci_set_prefetch — Sets number of rows to be prefetched by queries
- oci_statement_type — Returns the type of a statement