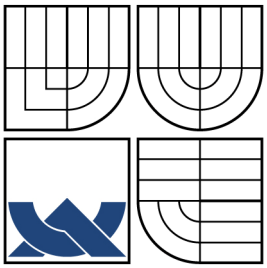
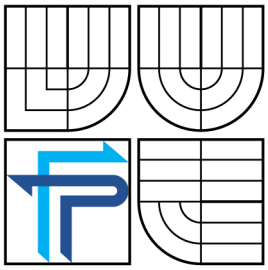


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY (UI)

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH FIREMNÍ DATABÁZE ZÁKAZNÍKŮ

DESIGN OF COMPANY CUSTOMERS DATABASE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — RADIM MĚŘIČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KŘÍŽ, Ph.D.
SUPERVISOR

BRNO 2007

Vysoká škola: Vysoké učení technické v Brně | Akademický rok: 2006/2007

Fakulta: podnikatelská | Ústav: informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Radim Měřička

6209R021 - Manažerská informatika

Ředitel ústavu v souladu se zákonem č. 111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů Vám zadává bakalářskou práci s názvem:

Návrh firemní databáze zákazníků

Design of Company Customers Database

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Analýza problému a současné situace
Teoretická východiska práce
Vlastní návrhy řešení, přínos (efektivnost) návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Rozsah grafických prací:	dle potřeby
Rozsah původní zprávy:	cca 40 stran

Seznam odborné literatury:

HOULETTE, Forrest. SQL Příručka programátora. Praha: SoftPress, 2001. 384 s. ISBN 80-86497-14-3.
ŠIMUNEK, Milan. SQL : Kompletní kapesní průvodce. 1. vyd. Praha: Grada Publishing, 1999. 248 s. ISBN 80-7169-692-7.
GILMORE, W. Jason. Velká kniha PHP5 a MySQL : Kompendium znalostí pro začátečníky i profesionály. Překlad RNDr. Jan Pokorný. 1. vyd. Brno : ZONER software s.r.o., 2005. 711 s. Encyklopedie webdesignera. ISBN 80-86815-20-X.
SCHLOSSNAGLE, George . Pokročilé programování v PHP 5. Překlad J. Gregor, J. Kuklínek, V. Šimek a M. Vokoun. 1. vyd. Brno: Zoner Press, 2004. 640 s. ISBN 80-86815-14-5.
DARIE, Cristian, et al. AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně. [s.1] : Zoner Press, 2006. 320 s. Vazba brožovaná. ISBN 80-86815-47-1.

Vedoucí bakalářské práce:	Ing. Jiří Kříž, Ph.D.
Datum zahájení bakalářské práce:	31. října 2006
Datum odevzdání bakalářské práce:	31. května 2007

Ing. Jiří Kříž. Ph.D.
Ředitel ústavu

Doc. Ing. Miloš Koch, CSc.
Děkan

V Brně dne: 16. února 2007

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Radim Měřička

Bytem: Palackého třída 2659/166, 612 00 Brno

Narozen/a (datum a místo): 21.března 1981 ve Slavičíně

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta podnikatelská

se sídlem Kolejní 2906/4, 612 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

Ing. Jiří Kříž, Ph.D., ředitel Ústavu informatiky

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☒ bakalářská práce
   - ☐ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	Návrh firemní databáze zákazníků
Vedoucí/ školitel VŠKP:	Ing. Jiří Kříž, Ph.D.
Ústav:	Ústav informatiky (UI)
Datum obhajoby VŠKP:	Červen 2007

VŠKP odevzdal autor nabyvateli v*:

- ☒ tištěné formě – počet exemplářů: 1ks
- ☒ elektronické formě – počet exemplářů: 1ks

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☒ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne:

…….. ……
Nabyvatel Autor

Abstrakt

Práce pojednává o problematice výběru vhodného databázového systému. Zabývá se historií jejich vývoje a vybrané systémy také porovnává. V díle je popsán návrh datového modelu pro firemní databázi zákazníků a nastíněno řešení jejího aplikačního rozhraní.

Klíčová slova

databáze, MySQL, FireBird, PostgreSQL, PHP, Microsoft Access

Abstract

This bachelor´s thesis deals with the issue of suitable database system selection. It content’s the historical development overview and it’s comparing. It content’s the historical development overview of the database systems and it’s comparing.

Keywords

database, MySQL, FireBird, PostgreSQL, PHP, Microsoft Access

Bibliografická citace práce

MĚŘIČKA, R. Návrh *firemní databáze zákazníků.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2007. 68 s. Vedoucí bakalářské práce Ing. Jiří Kříž, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorské právo (ve smyslu zákona č. 121/2000 Sb. O právu autorském a o právech souvisejících a právem autorským).

V Brně, dne 17. května 2007 ……

podpis

Poděkování

Rád bych na tomto místě poděkoval Ing. Jiřímu Křížovi, Ph.D. za podporu a cenné rady, které mi pomohly při zpracování bakalářské práce.

Dále děkuji firmě za možnost zpracovat pro ni tuto práci a za vstřícnou spolupráci.

Obsah

Úvod

Neexistuje podnikání, které by pro svou činnost nepotřebovalo evidenci dat. Vždyť už prosté shromažďování příjmových a výdajových dokladů je evidencí. Lidé však vymysleli počítače, pro které napsali statisíce programů a jednou z větví všemožných druhů se staly databáze.

Databázové systémy umožňují ukládání, archivaci a čtení různorodých druhů dat. Nejenže jsou schopny naleznout odpověď na správně formulovaný dotaz ve velmi krátkém časovém okamžiku, ale nabízejí efektivní a rychlou práci s miliony záznamů.

Význam elektronických databází je nepopiratelný. Vzpomeňme třeba na lístkové katalogy knih, dříve běžné ve všech knihovnách. Copak není rychlejší a pohodlnější zadat pár písmen do jednoho políčka na obrazovce, než se několik minut probírat nemalou hromádkou papírových lístků?

Počítačové databáze jsou skvělou náhradou papíru, ač by byl té nejlepší kvality.

1 Vymezení problému a cíle bakalářské práce

Od zavedení výpočetní techniky používá společnost ke správě a ukládání dat databázový program Microsoft Access. V průběhu let se však ukázal návrh datového modelu jako chybný. Souběžně se také projevily slabosti samotného programu. Mezi zásadní nedostatky stávající databáze patří nízká úroveň zabezpečení, nepřehlednost formulářů, programová závislost, těžkopádnost ovládání, rozmístění dat do několika různých souborů a jejich nenormalizované ukládání.

Tyto skutečnosti přiměly vedení firmy k vytvoření požadavku na vybudování plnohodnotné relační databáze.

Hlavním cílem mé bakalářské práce je **návrh a vytvoření datového modelu**, jehož stěžejním úkolem bude evidence a správa zákazníků firmy.

Sekundárním posláním práce **je návrh webové aplikace**, která bude zajišťovat komunikaci mezi databází a uživatelem.

Práci začnu krátkým pojednáním o důležitosti a významu databázových systémů. Dále pak představím „dotčený subjekt“ zanalyzuji současnou databázi a definuji požadavky, které musí být zohledněny při návrhu nové databáze. Teoretickou část zahájím povídáním o historii databázových systémů a posléze několik vybraných systémů podrobněji popíšu. V praktické části se budu zabývat výběrem nejvhodnějšího databázového systému a následně provedu samotný návrh datového modelu a aplikačního rozhraní. Závěrem svůj návrh řešení zhodnotím a letmo nastíním další možná vylepšení.

2 Analýza problému a současné situace

2.1 Analýza současné databáze

Databáze je vytvořena a spravována komerčním databázovým programem Microsoft Access. Program patří k produktům firmy Microsoft. Byl vyvinut s myšlenkou poskytnout běžnému uživateli prostředek, který od něj nebude vyžadovat speciální znalosti programovacích technik, ale umožní mu uspokojivě zpracovávat evidované údaje.

Prostředí tohoto programu poskytuje koncovým uživatelům nástroj na tvorbu jednoduchých, nenáročných aplikací na poměrně výkonném a velmi stabilním databázovém ovladači MSJet. Přestože je Microsoft Access vynikající produkt pro malé nasazení, může se ve větším projektu jevit spíše jako hřebík v botě.

Vybrané výhody programu Microsoft Access:

- Znatelná úspora času při tvorbě administrace.
- Snadná tvorba maker (automatizace práce).
- Rychlá odezva.
- Jednoduchost ovládání.
- Možnost integrace se stávajícími aplikacemi v MS Access.
- Široké možnosti pro import a export dat.

Vybrané nevýhody programu Microsoft Access:

- Funguje pouze pod Windows (případně MDBTools pod Linux).
- Zabezpečení je nutno řešit na straně databáze.
- Cena balíčku Microsoft Office.
- Nepodporuje UTF-8.

Jak již bylo zmíněno, tak je tento program vhodný spíše pro menší projekty. Databáze se rozrostla do takové míry, že je vhodné převést stávající data do jiné, plnohodnotné relační databáze, pro niž bude vytvořeno vhodné rozhraní.

Samotná data současné databáze jsou uložena v tabulkách o různé struktuře v sedmi souborech typu MDB. Každý soubor je zabezpečen heslem. Ačkoliv musím konstatovat, že pro všechny soubory heslem stejným, není to tak velký hřích. Důvodem obhajoby používání stejného hesla je fakt, že existují nástroje, které dokážou v řádech milisekund (maximálně sekund) heslo pro databázi MS Access zjistit! Jedním z nich je například program Password Recovery And Security od firmy LastBit Corp[1].

Základním účelem současné databáze je evidence skutečných a potencionálních zákazníků. Sekundárním posláním je evidence vystavených smluv a nabídek. Celá databáze se skládá z těchto devíti tabulek:

- Smlouvy,
- Nabídky,
- Zájemci,
- Plošný plán,
- Farní úřady,
- Obecní a městské úřady,
- Architekti,
- Projektanti,
- Sociální ústavy.

2.2 Současná databáze

S databází pracují ve firmě tři osoby a obvykle je využívána dvakrát až třikrát týdně. Běžným pracovním úkonem je vložení a čtení záznamů v tabulkách „Nabídky“, „Smlouvy“ a „Zájemci“. Mezi další obvyklé operace patří čtení dat pro tisk adres, přesun záznamů mezi databázemi potencionálních a skutečných zákazníků, vymazávání neexistujících osob, institucí a firem, apod.

Ačkoliv MS Access nabízí víceuživatelskou práci s databází, není tato schopnost ve firmě využívána, a tak prvním krokem před započetím práce s databází je nakopírování aktuálních souborů z centrálního úložiště. S tím souvisí nutnost jejich

[1] http://www.lastbit.com/

zpětné aktualizace po ukončení práce. Z toho vyplývá, že práce s databází musí být koordinovaná a nesmí nastat situace, při níž by se stejným souborem pracovali dva lidé.

Pracovní shon, lenost, zapomnětlivost nebo jiný faktor přispívá k situacím, kdy pověřená osoba zpětně nezkopíruje aktualizovaný soubor, díky čemuž dochází za několik málo dnů ke ztrátě předchozí práce přepsáním novějších souborů starými či pokračování práce se staršími daty.

Důsledkem toho se stává, že:

- v tabulkách jsou duplicitní záznamy,
- je neúplná evidence uskutečněných nabídek a smluv,
- selhává výmaz zákazníků, kteří si přáli být z databáze vymazáni,
- setrvává evidence adres, ze kterých se pošta pravidelně vrací.

Ve všech případech způsobuje nastanuvší situace určitou finanční ztrátu - např. zbytečné rozesílání korespondence, placení pracovní síly, opotřebení strojů. Jednak se jedná o zbytečně promrhané finance, ale především si firma poškozuje dobré jméno. Pro společnost je velmi nepříjemné obeslat všeobecnou aktuální nabídkou zákazníka, se kterým právě dojednává podmínky budoucí smlouvy.

Vedení firmy si je těchto neduhů vědomo, a je to jeden z aspektů, proč vybudovat plnohodnotnou relační databázi s vždy aktuálními výstupy a minimálním rizikem duplicity dat.

Mezi další patří již zmiňovaná nízká úroveň zabezpečení dat, omezené možnosti práce v lokální síti, problematické možnosti rozšíření a především pak nepřehlednost v datech rozložených do několika různorodých souborů.

Mimo vyřešení těchto negativních aspektů očekává společnost od nového systému sumární a statistické výstupy.

Momentálně jsou definovány dva požadavky:

- účinnost reklamních kampaní (příjmy vs. náklady),
- přehled uskutečněných zakázek v požadovaném období.

3 Teoretická východiska řešení

3.1 Stručná historie vývoje databází a databázových systémů

Databázové systémy nejsou odvětvím informačních technologií, v nichž by byl vývoj takový, že by byla každý den uvolněna nová verze některého databázového systému, která by například přinášela nové vlastnosti či pouhé zvýšení výkonu. Problematiku ukládání strukturovaných dat řeší svět, potažmo počítačoví odborníci, od 90-tých let předminulého století. Ačkoliv to zní neuvěřitelně, tak již v roce 1890 vytvořil Herman Hollerith první automat na bázi děrných štítků. Zda lze tento rok pokládat za první rok databází jest otevřenou otázkou, nicméně v roce 1911 se firma Hermana Holleritha spojila s firmou jinou a vznikla firma International Business Machines, což je firma, kterou dnes známe pod zkratkou IBM.

Za jeden z největších impulsů v historii vývoje databází lze považovat rok 1935, v němž byla firma IBM pověřena vládou USA vytvořit zařízení, které by bylo schopno vést informace o 26-ti milionech zaměstnaných občanů USA. Projekt byl založen na státem podporovaném projektu Electronic Discrete Variable Automatic Computer (EDVAC).

V roce 1960 ustanovilo Ministerstvo obrany USA seskupení Data Systems Languages (CODASYL), které mělo standardizovat software aplikací, pracující s databázemi. Výsledkem činnosti bylo definování jazyka COBOL.

Mocnějšímu rozvoji databázových systémů do té doby bránila mnohá technická omezení, především pak sériové ukládání dat. Pokud chtěla aplikace přečíst nějaké data, musela projít celý záznamový nosič (typicky magnetickou pásku), což bylo samozřejmě zdlouhavé, omezující a nákladné. Nástup magnetických disků tak znamenal velký krok vpřed, protože k požadovaným datům mohlo být přistupováno rovnou.

První integrovaný datový sklad představil v roce 1961 Charles Bachman z firmy General Electric. Jeho návrh již obsahoval náznak moderního databázového managementu a jiné vlastnosti, které na příští desetiletí určily směr vývoje. Bachman spolu s dalšími odborníky své doby založili samostatnou skupinu Database Task Group (DBTG), co by odnož seskupení Codasyl. Skupina rozšiřovala specifikace

programovacích jazyků určených pro databáze, zvláště pak jazyka COBOL. Na jejich práci navazovaly mnohé společnosti, které vyvinuly řadu produktů (např. firmy Honeywell Incorporated, Siemens AG, Eckert-Mauchly Computer Corporation).

Svou cestou se však v šedesátých letech ubírala firma IBM. Rozdíl mezi jejími databázovými systémy a systémy CODASYL byl především ten, že IBM používala hierarchický model dat, zatímco většina ostatních systémů používala model síťový.

Velký třesk ve světě databázových systémů způsobil Edgar Frank Codd (zaměstnanec IBM), který v roce 1970 publikoval v časopise Communications of ACM článek „A Relational Model of Data for Large Shared Data Banks", což byl návrh na implementaci nového datového modelu, který byl záhy nazván "relačním". Teoreticky v něm popsal možnost použití relačního kalkulu a algebry, s jejichž pomocí by se data efektně ukládala i načítala. Codd nepoužil poznatky z teorie grafů, jak to učinili jeho předchůdci, nýbrž poznatky z teorie množin. Pomocí základních množinových operací (sjednocení, kartézský součin, rozdíl, selekce, projekce a spojení) lze totiž s daty provést téměř všechny operace a vše ostatní je jen kombinací základních operací. Přínosem byla jednoduchá forma relací a exaktně formulované operace s nimi.

Dále navrhl, aby byla data ukládána nezávisle na použitém hardware, na způsobu jejich fyzického uložení a také, aby byl k datům umožněn přístup pomocí neprocedurálního jazyka. Důležitá pro něj byla i možnost manipulace s vybranou množinou dat a ne pouze s jedním konkrétním záznamem.

Jeho návrhy byly zprvu opomíjeny a brány spíše jako kuriozita, nicméně po dvou letech byl stav takový, že na základě jeho úvah vznikly přinejmenším dva projekty pro vývoj relačních databází. Jedním byl projekt System-R firmy IBM a druhým se stal projekt Ingres (Interactive Graphics and Retrieval System) na University of California at Berkeley (UC Berkeley), který mimo jiné podporovala armáda Spojených států.

Systémy Ingres a System-R byly vyvíjeny pod různými operačními systémy. Společnost IBM zavedla pro System-R vlastní dotazovací jazyk, vydaný pod názvem SEQUEL (resp. SEQUEL2), což je zkratka slov Structured English QUEry Language. Z důvodů dřívější registrace názvu SEQUEL jinou firmou, byl tento jazyk přejmenován na SQL (Structured Query Language). Ingres měl podobný dotazovací jazyk QUEL.

Podobnost však není čistě náhodná – IBM i Ingres na vývoji spolupracovali a ačkoliv byl tým Ingres poloviční a členové jeho týmu chodili do IBM na školení, byl to právě jejich systém, o kterém lze říci, že se stal základem dnešních relačních databázových systémů. Velkou vinu na tom má totiž vedení společnosti IBM, které s přechodem na relační databáze dlouho váhalo.

Avšak první SQL databází na trhu byl systém ORACLE od stejnojmenné firmy. K prodeji byl uvolněn v roce 1977. Vycházel sice z návrhů projektu System-R, ale ve finále se jednalo o zcela jiný systém.

Tím pádem lze osmdesátá léta považovat za léta nástupu relačních databázových systémů. Některé se povedly více, jiné méně. Každopádně se díky poznatkům získaným při vývoji i běžném používání ještě více vyjasnily další požadavky na (relační) databázové systémy.

Projekt Ingres byl sice v roce 1982 ukončen, ale v roce 1985 se stal základem projektu Postgres, jehož cílem bylo vytvoření relačně-objektové databáze. Projekt probíhal na univerzitě v Berkeley devět let, když byl v roce 1994 ukončen. Nicméně se našli dva nadšení studenti, kteří jej v následujícím roce vzkřísili, začali na něm dále pracovat, a uvolnili jeho zdrojové kódy světu, co by databázový systém PostgreSQL, který je (samozřejmě v novějších verzích) používán dodnes.

V roce 1987 byl mezinárodní organizací pro normalizaci ISO (International Organization for Standardization) uznán standard SQL. O pět let později byl upraven a normalizován jako SQL2.

Dalším impulsem k rychlejšímu rozvoji databázových systémů se stal Internet. Celý svět rychle pochopil jeho velký potenciál a proto se vývojáři dali na vývoj databází, určených především pro toto prostředí. Orientačně lze zmínit rok 1997, protože v němž byla vydána třetí verze skriptovacího jazyka PHP, která způsobila velký poprask. Možnost publikovat libovolné textové informace přilákala mnoho nových lidí, potažmo vývojářů, kterým se otevírali nové možnosti.

Člověk obvykle není nikdy s ničím dlouho spokojen. Obecně to platí i o internetu, pro nějž lidé vyvíjejí stále složitější internetové aplikace. Pro ukládání strukturovaných dat však potřebují více než jednoduché textové soubory, v nichž byly data oddělována určitým separátorem. Takže netrvalo to dlouho a byly vyvinuty

aplikace, které byly schopny připojovat se k databázovým serverům, a nadneseně řečeno se jedná o internet, který známe dnes.

3.2 Databázové modely

Databázový model je nástroj popisující uspořádání (strukturu) a chování (funkcionalitu) databáze. Umožňuje definovat schéma databáze, pravidla pro organizaci dat, způsoby jejich ochrany a zajištění správnosti (integritní omezení) a také přípustné operace s daty. Na základě těchto modelů vznikaly konkrétní databázové systémy.

V současnosti jsou využívány především tyto modely:

- Hierarchický model,
- Síťový model,
- Relační model,
- Objektově-relační model,
- Objektový model.

První dva zmiňované modely už jen spíše dožívají a tak se lze prakticky setkat jen s relačními modely (tyto modely jsou nejrozšířenější), nastupujícími objektovými modely a objektově-relačními modely.

Teorie relačního modelu dnes tvoří základ databázové teorie vůbec. Přestože ve světě ještě existuje mnoho hierarchických i síťových databází, dlouhodobým trendem je výlučný přechod na relační databáze. Příliš mnoho na tom nemění ani dnešní tendence k objektové orientaci. Budu-li hovořit o databázové technologii, budu mít na mysli technologii relační (nebude-li uvedeno jinak).

3.2.1 Hierarchický model

V hierarchický datových modelech jsou data organizována do stromové struktury, která dovoluje dědit informace použitím mateřských a dětských vztahů: každý rodič může mít mnoho dětí, ale každé dítě má jen jednoho rodiče.

Výhodou databáze vybudované na tomto datovém modelu je velice rychlé vyhledávání, nevýhodou je to, že je vhodný pouze pro aplikace se stabilní strukturou,

tedy tam kde se primární vztahy mezi daty mění jen velice málo, což je omezení vyplývající právě z jeho výhod.

Hierarchické struktury jsou dnes používány v LDAP modelech, ale i zde se již uplatňují modernější modely. Nutno podotknout, že hierarchický model nemá standard.

3.2.2 Síťový model

Síťový model se od modelu hierarchického odlišuje především tou vlastností, že potomek může mít více rodičů, než jednoho, což je lepší. Za špatnou vlastnost je pak databázím postaveným na tomto modelu vyčítána přílišná složitost (vazby jsou velice komplikované), čímž se snadno ztrácí přehled nad databází, a pro udržení integrity databáze je nutný velký výkon.

Podobně jako hierarchický model, se ani tento model příliš neosvědčil.

3.2.3 Relační model

Základem každé relační databáze je tabulka (neboli relace – odtud název Relační modely a Relační databáze). Řádky v tabulce odpovídají záznamům, sloupce atributům. Databázi pak tvoří soubor tabulek, které jsou mezi sebou vzájemně provázány.

Relační databáze se vyznačují především svou jednoduchostí, snadnou pochopitelností a velkým rozšířením. Databázových systémů tohoto typu je velmi mnoho. Každý nabízí určitá vylepšení, ale v základu dodržují standard dotazovacího jazyka SQL.

Velkým přínosem relačního modelu je také fakt, že klade velký důraz na zachování integrity zpracovávaných dat a zavádí pojmy jako jsou referenční integrita, cizí klíče, primární klíče, kandidátní klíče, apod.

3.2.4 Objektový model

Vedle relačních databází se začal počátkem 90. let minulého století rozvíjet nový typ databázových systémů založený na principech objektově orientovaného programování.

Základem objektově orientované databáze není tabulka, nýbrž objekt. Každý objekt má atributy (zde je vidět analogie se sloupci v tabulce) a metody, které určitým způsobem manipulují s hodnotami atributů. Jednotlivé instance objektu s konkrétními hodnotami atributů tvoří záznamy (v relačních databázích 1 řádek). Pro jednoznačnou identifikaci objektu je používána objektová identita (OID). Lze využít všechny výhody dědičnosti (i vícenásobné), zapouzdření i polymorfismu. Také jsou podporovány abstraktní datové typy.

Tvorba objektově orientované databáze je mnohem složitější než vytvoření relační databáze, ale odměnou za vynaložené úsilí je mnohem snadnější tvorba dotazů. Dotazovacím jazykem již není SQL, ale k datům je umožněn přímý přístup pomocí objektově orientovaného jazyka.

3.2.5 Objektově-relační model

Současný vývoj směřuje ke sbližování objektových a relačních databází a vytváření tzv. objektově-relačních (někdy též nazývaných postrelačních) databázových platforem. Dá se tedy předpokládat, že v budoucnu nahradí nyní nejpoužívanější databáze založené na relačním modelu.

Všechny informace jsou stále v tabulkách, ale některé položky mohou mít bohatší datovou strukturu, nazývanou abstraktní datové typy (ADT). ADT je datový typ, který vznikne zkombinováním základních datových typů relační databáze.

Dotazovacím jazykem je zde rozšířená verze SQL – SQL3. Důvodem je podpora objektů. Přímou podporu objektových jazyků však využít již nelze.

3.3 Přehled vybraných relačních databázových systémů současnosti

Každý program má své silné i slabé stránky, které ho v porovnání s obdobnými produkty činí pro různé typy úloh lepším, srovnatelným či zcela nevhodným. Proto v této kapitole popíši a srovnám většinu významných databázových systémů, které jsou v současné době běžně k dispozici a to jak produkty komerční, tak i produkty distribuované zdarma pod různými svobodnými licencemi. Jedná se o výběr zhruba z osmi desítek databázových systémů.

Jelikož se všechny systémy neustále vyvíjejí, a rozdíl mezi jednotlivými verzemi bývá častokrát značný, budu popisovat tu verzi daného systému, kterou autoři k 31.3.2007 označili za finální. Do těchto verzí nespadají beta verze produktů, které v budoucnu nahradí současnou verzi konkrétního databázového systému.

Ty systémy, které jsou potencionálně využitelné pro vyřešení hlavního tématu této práce, analyzuji podrobněji. Jedná se o databázové systémy MySQL, PostgreSQL a Firebird. V následující kapitole vyberu z výše zmiňovaných systémů ten produkt, který budu považovat za nejvhodnější a při vlastním návrhu řešení jej použiji.

3.3.1 Berkeley

Databázový systém Oracle Berkeley DB pochází z rodiny open source databází. Původem se jedná o systém vyvinutý na kalifornské univerzitě v Berkeley, nicméně v současné době jej vlastní firma Oracle Corporation.

Těžko uvěřit, že verze 1.85 vydaná v roce 1994 se používá v nezměněné formě dodnes. Důvody jsou prosté: databáze je výkonná, chová se velmi stabilně, má jednoduché API a její kód je odladěný.

Systém Berekeley dovoluje vývojářům včlenit do jejich aplikací rychlý, škálovatelný, transakční databázový engine. Díky tomu mají zákazníci a koncoví uživatelé k dispozici aplikaci, která jednoduše pracuje, spolehlivě řídí data, zvládá nadměrná zatížení a také nevyžaduje žádnou pokročilou správu. Data jsou ukládána do jednotlivých souborů, přičemž každý soubor odpovídá jedné databázi.

O tomto systému se však ve střední Evropě takřka vůbec nemluví, takže snad i proto nemá příznivce, kteří by o ní publikovali. Nicméně ji některé české webhostingy nabízejí a pravděpodobně se tak Berkeley dočká zpopularizování. Zatím tomu brání i licenční politika, která umožňuje použít systém pouze v projektech open source.

3.3.2 dBase

Systém dBase byl první široce užívaný systém řízení báze dat určený pro mikropočítače. Později jej využívali i systémy Unix, Apple či DOS. Konvence formátu využitého v dBase se rychle stala průmyslovým standardem, který je používaný ještě dnes. Nutno podotknout, že nepodporuje transakční zpracování dat.

3.3.3 FireBird

„InterBase je mocný, kompaktní relační databázový systém na bázi klient/server, provozovaný na celé škále operačních systémů včetně MS Windows, Linuxu a řadě dalších UNIXových systémů, a nabízející vyšší úroveň podpory standardů jazyka SQL než většina dostupných databázových systémů,“[2] píše v úvodu knihy Interbase/Firebird Pavel Císař, považovaný za jednoho z největších odborníků na FireBird.

Relační databázový systém FireBird je založen na zdrojových textech systému InterBase, uvolněných společností Borland v létě roku 2000. Oba produkty se sice budou do budoucna ubírat odlišnou cestou, v současnosti si jsou však oba systémy velmi podobné a poměrně snadno lze vyměnit jeden systém za druhý.

Firebird je vyvíjen v rámci stejnojmenného open source projektu, který je financován neziskovou organizací FirebirdSQL Foundation a sdružením zainteresovaných firem. Jedná se totiž o produkt s velkým potenciálem, který by měl v budoucnu hrát ve světě databází velkou roli. Hodně populární je momentálně v Rusku. Důvodem oblíbenosti je účast ruských vývojářů a dnes již ukončený projekt Yaffil, který byl určen speciálně pro ruskou komunitu.

Vlastnostmi, které jsou nadstandardně propracované a posouvají FireBird nad hranici běžného databázového systému, jsou:

- **Transakční zpracování dat** (účelem je zabezpečit jejich konzistenci).
- **Spouště** (automaticky prováděné programy spojené s operacemi INSERT, UPDATE nebo DELETE nad konkrétní tabulkou).
- **Uložené procedury**, které lze dále rozčlenit na:
    - procedury **běžné**, které nevracejí žádné hodnoty,
    - procedury **dotazovatelné**, které hodnoty vracejí.
- **Uživatelské výjimky** (programátorem definované hlášení chybových stavů, které mohou nastat v uložených procedurách).
- **Zpracování chyb** (lze odchytávat chyby a postavit programy tak, aby na ně byly připraveny a mohly vykonat něco víc, než pouze skončit).

[2] CÍSAŘ, Pavel. *InterBase/FireBird : podrobná příručka : tvorba, programování a správa databází.* 1. vyd. Brno : Computer Press, 2003. 453 s. , 1 elektronický optický disk. ISBN 80-7226-946-1.

- **Rozesílání událostí klientům** (mechanismus umožňující rozesílat z procedur a spouští klientským aplikacím oznámení o událostech).
- **RDB$DB_KEY** (je známa logická adresa fyzického uložení dat).
- **Uživatelsky definované funkce** (rozšiřují možnosti SQL jazyka).

Naopak FireBird nepodporuje dočasné tabulky, nicméně je tato vlastnost očekávána ve verzi 2.1. Na druhou stranu ale nabízí excelentní propustnost při víceuživatelském zpracování. Funguje na Linuxu, Windows a řadě Unixových systémů.

S mírnou nadsázkou lze říci, že širšímu použití databázového systému Firebird brání už jen ostych programátorů. Čím více se však bude o FireBirdu psát, a čím kvalitnější bude dokumentace, tím častěji bude mu bude správa dat svěřována.

3.3.4 Ingres II

Ingres pravděpodobně po vzoru InterBase a SAP DB rozšíří řady dříve komerčních SQL databází nyní dostupných jako Open Source. Ingres byl původně vytvořen na UC Berkeley a je více než příbuzným PostgreSQL.

3.3.5 InterBase

Jedná se o silnou, výkonnou a víceplatformní databázi z dílny firmy Inprise/Borland známou pro svou nenáročnost na údržbu a velký výkon. Díky malým nárokům na hardware a údržbu je InterBase používána v celé řadě aplikací.

Zdrojové kódy InterBase se staly základem zrodu několika dalších databázových systémů, z nichž nejvýznamnějším je FireBird, který jej dnes v mnohém předčí. Po této zkušenosti přestala firma Borland kráčet cestou open source a vrátila se zpět ke komerčním produktům.

3.3.6 MS SQL

Microsoft SQL server je relační systém řízení báze dat a stejně jako firma Oracle je i Microsoft klíčovým hráčem na databázovém trhu. Přestože jeho databázové produkty nedosahují absolutních kvalit, je v hojné míře využíván. Ceněna je například snadná možnost provázání obsahu webových stránek s firemní databází nebo klasické

grafické rozhraní stylu Windows. MS SQL splňuje požadavky normy ANSI SQL - 92 a také ji rozšiřuje, ale méně než jiné systémy.

Ačkoliv je MS SQL Server poměrně vysoce stabilní, prochází si mnoha problémy spojených s bezpečností. S tím blízce souvisí nutnost častých aktualizací systému. Ty ale vyžadují restart systému, což je pro zákazníky, kteří požadují maximální dobu provozuschopnosti, nepřijatelné.

Cena za licenci pro SQL Server je poměrně vysoká. K ceně produktu se musí počítat i s náklady na operační systém, protože neexistuje jiná volba než Microsoft Windows. Existuje však i bezplatná verze serveru pod názvem Express edition. Ke stažení je k dispozici přímo na webu Microsoftu, na kterém lze naleznout i mnoho podpůrných informací.

3.3.7 mSQL

Vznikl v roce 1994, když zaplnil vakuum, které existovalo mezi databázemi typu Microsoft Access a výkonnými komerční databázemi jako Oracle či DB2. V následujících třech letech nabyl systém na popularitě a byl primární volbou výběru programátorů, kteří nechtěli utrácet za komerční produkty.

Přestože bylo možné databázi libovolně využívat, její zdrojové kódy uvolněny nebyly. V roce 1996 začal vývoj stagnovat a postupně jej začal nahrazovat MySQL, vycházející ze stejné architektury. Přestože je mSQL na ústupu, je stále v aktivním vývoji. Poslední verze 3.8 byla uvolněna 9. června 2006.

3.3.8 MySQL

MySQL je multiplatformní databázový systém, vytvořený švédskou firmou MySQL AB. První verze systému byla vydána v květnu 1995. Pro svůj vysoký výkon a především díky tomu, že se jedná o volně šiřitelný software (existuje však i komerční verze distribuce), má v oblasti databázových systémů vysoký podíl. Na oblibě mu přidává snadné propojení s populárním webovým serverem Apache a možnost komunikace se skripty hypertextového preprocesoru PHP. Jde o jednu z nejlepších voleb pro data poskytována přes Internet.

Kromě mnoha jiných vlastností se MySQL vyznačuje:

- Podporou multiprocesorových počítačů.
- Datové typy jsou schopny pokrýt prakticky všechny druhy dat.
- Propracované funkce pro matematické kalkulace a třídění dat.
- Snadné získání informací o stavu databáze.
- Až 32 indexů na jednu tabulku.
- Velmi dobře propracovaná dokumentace a velká podpora veřejností.

3.3.9 Oracle

Neoddiskutovatelně se jedná o nejlepší databázový systém na světě. Vyvíjen je od roku 1978 ve firmě Oracle Corporation, kterou založili přeběhlíci z univerzitního výzkumu v americkém Berkeley. V současné době firma zaměstnává přes 40.000 zaměstnanců ve 145 zemích světa a její přibližný roční obrat činí 10 miliard USD.

Systém Oracle je schopen běžet jak na obyčejném PC s operačním systémem Windows, tak i na řadě distribucí z rodiny UNIX a samozřejmě i na multiprocesorových počítačích. S přehledem dokáže zpracovávat až miliardy řádků dat[3] a je tak předurčen pro ty nejrozsáhlejší projekty v nejnáročnějších oblastech, jakými jsou například bankovní ústavy nebo státní správa.

Oracle Database je kompletní platforma pro ukládání, správu a analýzu dat. Unikátním způsobem je řešena bezpečnost, výkonnost a ukládání dat, což v kombinaci s mnoha dalšími vlastnostmi přináší nejspolehlivější databázi. Jedinečná je i technická podpora produktů. To vše je však vykoupeno vysokou cenou systému (samozřejmě existuje více variant), ale také náročnou instalací a správou.

Vyjmenovanými aspekty je zdůvodněno nasazování tohoto systému jen na opravdu velké projekty, které obvykle vyvíjí početný tým programátorů. Neznamená to však, že by nestačil na projekty malé. Šlo by ale o známé zabíjení komárů kladivem.

[3] Aplikace musí běžet na odpovídajícím hardware

3.3.10 PostgreSQL

Multiplatformní databázový systém PostgreSQL je následovníkem systému POSTGRES, který byl vytvořen v letech 1977-1985 týmem profesora Stonebrakera na univerzitě v Berkeley. Vývoj byl sice v jeden čas ukončen, ale zanedlouho byl naštěstí vzkříšen dvěma postgraduálními studenti.

Patří jim velký dík, protože systém PostgreSQL je v současnosti vychvalován mnoha tisíci spokojených uživatelů pro své skvělé vlastnosti, excelentně zpracovanou dokumentaci, vysokou spolehlivost a bezpečnost. Tým lidí okolo PostgreSQL odvádí skutečně pozoruhodný kus práce a jejich databáze umí hodně z toho, co lze od moderního systému očekávat. Nutno říct, že tak činí nezištně.

Kromě všech běžných schopností nabízí nejnovější verze i nadstandardní funkce, jakými jsou:

- Dvoufázové potvrzování dotazů.
- Podporu více procesorů.
- Částečné a funkcionální indexy.
- Informační schémata.

V současnosti je šířen pod BSD licencí, která je nejliberálnější ze všech open source licencí. Tato licence umožňuje neomezené používání, modifikaci a distribuci. PostgreSQL je možné šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně.

3.3.11 SQLite

SQLite je pouze malá knihovna napsaná v jazyce C, která v podstatě nedělá nic jiného, než implementuje jazyk SQL nad souborem dbm. Celá databáze je umístěna v jediném souboru na disku, přičemž tento soubor může být sdílen mezi různými počítači i platformami. SQLite se nechová jako server, takže pokud se s databází nepracuje, neběží v systému žádný proces. Projevuje se jednoduchostí, sympatickým výkonem, nízkou náročností na systémové zdroje a poměrně kompletní implementací jazyka SQL. Další zajímavou vlastností je velikost SQLite, která činí přibližně 350kB.

Ze standardu SQL92 toho umí poměrně mnoho, byť je to častokrát nějakým způsobem omezeno. SQLite například nezná datové typy a velikost jednoho záznamu

nesmí přesáhnout 1MB. Přesto je nasazování tohoto systému na nejmenší projekty velmi vhodné.

3.4 Srovnání databázových systémů

Pro možnost rychlého srovnání jsem vytvořil stručný přehled základních údajů výše popsaných systémů. Podrobnější srovnání je umístěno v přílohách této práce. Účelem srovnání není seřazení podle výkonnosti nebo podporovaných funkcí.

3.4.1 Přehled základních údajů

V následující tabulce bude použit výraz „Proprietární software“. Jde o software, u něhož jeho autor upravuje licenci (typicky EULA) či jiným způsobem upřesňuje možnosti používání. Obvykle spadá do kategorie komerčních a zpravidla nejsou jeho zdrojové kódy volně k dispozici, nelze v něm dělat úpravy a výsledné dílo distribuovat.

RDBMS	Vlastník / výrobce	První verze systému	Poslední stabilní verze	Licenční politika
Berkeley DB	Oracle	1991	3.2.23	proprietární SW
dBase	dataBased Intelligence	1978	2.63.1	?
FireBird	Firebird foundation	06/2000	2.0.1	proprietární SW
Ingres II	Ingres Corp.	1974	2006 (9.0.4)	GPL
InterBase	Borland	1985	7.5.1	proprietární SW
MS SQL	Microsoft	1989	9.00.3042	proprietární SW
mSQL	Hughes technologies	1994	3.8	OEM
MySQL	MySQL AB	11/1996	5.0.41	GPL
Oracle	Oracle Corporation	11/1979	10g release 2	proprietární SW
PostgreSQL	PostgreSQL Global Development Group	06/1989	8.2.4	BSD
SQLite	D. Richard Hipp	09/2000	3.3.7	public domain

Tabulka 1: Přehled základních údajů vybraných databázových systémů

3.4.2 Podpora operačních systémů

Při výběru databázového systému je postup zpravidla takový, že hledáme software pro vybraný operační systém a ne naopak. Je však dobré vědět, zda je zvolený systém podporován i na jiném OS. Ve výhodě jsou ty databáze, u nichž závisí dostupnost pouze na existenci Java Virtual Machine, nikoliv na operačním systému.

RDBMS	Windows	Mac OS	Linux	BSD	UNIX	z/OS
Berkeley DB	✓	✓	✓	✓	✓	✓
dBase	✓	×	×	×	×	×
FireBird	✓	✓	✓	✓	✓	?
Ingres II	✓	×	✓	✓	✓	?
InterBase	✓	×	✓	×	✓	×
MS SQL	✓	×	×	×	×	×
mSQL	✓	?	?	✓	✓	?
MySQL	✓	✓	✓	✓	✓	?
Oracle	✓	✓	✓	×	✓	✓
PostgreSQL	✓	✓	✓	✓	✓	×
SQLite	✓	✓	✓	✓	✓	?

Tabulka 2: Podpora operačních systémů vybranými databázovými systémy

4 Vlastní návrh řešení

4.1 Výběr vhodného databázového systému

Veřejností hýbe kontroverzní otázka: Dají se porovnávat různé databázové systémy? Ačkoliv jasná odpověď neexistuje, přesto se většina databázových odborníků přiklání k názoru, že to možné sice je, ale jen za zcela specifických podmínek. Těmito podmínkami se rozumí nastavení jednotlivých systémů tak, aby každý z nich poskytoval maximální výkon s ohledem na hardware, software a platformu, na němž by byly systémy testovány, přičemž by nastavení muselo být optimalizováno i pro typ a strukturu tabulek použitých v testech.

Pakliže budou dodrženy výše zmiňované podmínky a každý systém bude nastaven na své maximum, je možné hovořit o objektivních podmínkách pro testování. Ještě je však důležité zvolit vhodný počet tabulek a vazeb mezi nimi. S tím úzce souvisí i volba vhodných SQL dotazů, se kterými by se testy prováděly. Rozhodně by se mělo jednat o dotazy, jejichž výsledky budou seskládány z dat několika tabulek, přičemž počet řádků, které by měly systémy procházet, by byl v řádech statisíců.

Toto je jediný známý způsob, jak lze alespoň částečně porovnat databáze a získat tím objektivní přehled o jejich výkonu. Na internetu se sice objevují mnohé srovnávací testy, ale jen málo z nich dodrželo popsaná pravidla. Zkušení vývojáři totiž dost dobře vědí, že uskutečnění takovýchto testů není zhola snadná záležitost! Proto je lepší brát výsledky zveřejněných srovnávání s nadhledem a systém vybírat podle konkrétní situace.

Nesmí se však zapomenout na skutečnost, že každá databáze se hodí na něco jiného. Malé databázové platformy přesně odpovídají svému účelu, tedy provozu nenáročných evidencí a aplikací. To lze považovat za prvořadé kritérium, podle nějž by se měl databázový systém vybírat. V žádném případě tedy není dobré podlehnout falešné představě o ideálnosti vyspělých platforem pro libovolné nasazení.

V těsném závěsu za tímto kritériem je vlastní zkušenost programátora, neb platí to, že nerozhoduje programovací jazyk, nýbrž to, jak je programátor dobrý či špatný. Svou roli hraje i dostupný hardware, software a v neposlední řadě i cena produktu.

Mám-li vybrat mezi srovnatelnými databázovými systémy PostgreSQL, MySQL a Firebird, pak druhé kritérium rozhodne o vyřazení PostgreSQL. Ačkoliv je vysoce výkonný, podporuje mnoho věcí ze standardu ANSI-SQL, je dobře zdokumentovaný a v českém prostředí hojně používán, nemám s ní žádné osobní zkušenosti a tudíž je pro mě snadnější vybírat si mezi systémy MySQL a Firebird. Jedná se totiž o systémy, se kterými běžně pracuji. Nicméně nynější rozhodnutí vyřadit PostgreSQL z důvodu neznalosti neznamená, že jej v budoucnosti nepoužiji v jiném projektu.

Rozhodnutí o nepoužití PostgreSQL mi sice bylo poměrně dlouho dopředu jasné, ale tušil jsem také, že výběr mezi Firebirdem a MySQL bude těžký boj. O Firebirdu se hovoří jako o nově přicházejícím velkém hráči v oblasti open source databázových systémů. Vývojáři je vychvalována velmi sympatická svižnost, jednoduchá instalace a inteligentní zpracování transakcí. Na druhou stranu je systém stále ve vývoji a zpracování dokumentace je pro tým Firebird vedlejší záležitost. Sám jsem několikrát hledal odpověď na ne příliš složitou otázku jak ve Firebirdu něco správně formulovat nebo nastavit. Obvykle se mi to sice podařilo, ale musel jsem notnou chvíli hledat na internetu či se poradit se spolupracovníky. Nebýt tohoto velkého mínus, byl by Firebird jasnou volbou.

MySQL je nejrozšířenější a je znám jako velmi lehce zvládnutelný a famózně zdokumentovaný databázový systém, schopný dosahovat vysokých výkonů. Bohužel je tak jednoduchý a tak snadno dostupný, že je to první databáze, po které šáhnou začínající programátoři. Dopouštějí se však mnoha chyb a je zcela běžné, že jsou tyto chyby přeneseny z vývojového prostředí i na webhosting. Následkem toho je přetížení databázových serverů, což potažmo vyvolává u uživatelů dojem, že MySQL je databáze pomalá. Opak je však pravdou – pokud je databáze vhodně navržená, naindexovaná a SQL dotazy nejsou špatně zformulovány, jedná se o správnou volbu pro středně velké až velké projekty.

Faktorem, který by měl ovlivňovat můj výběr, je podpora databází u českých webhostingových společností. Protože však bude celá aplikace dostupná pouze na intranetu, není jejich nabídka rozhodující. Naopak mě zajímá snadnost instalace na lokálním počítači, který bude sloužit jako firemní server. Operačním systémem bude Windows XP. Instalace a nastavení MySQL je sice trochu složitější, ale obvykle je

databáze provozuschopná do 15-ti minut. S tím kontrastuje úžasně snadná instalace Firebirdu, trvající cca 20 vteřin.

Poslední věcí, která může do výběru systému výrazněji promluvit, je nabídka administračních nástrojů. Práce pomocí textové konzoly je totiž značně nepohodlná a nepřehledná. Pro obě databáze existují kvalitní grafické rozhraní, MySQL je však populárnější a tak pro ni lze naleznout více aplikací. Firebird zase těží z toho, že vychází z návrhu systému Interbase, čili programy pro správu Interbase jsou zpravidla vhodné i pro správu Firebird. V obou případech platí, že lepší produkty nejsou zdarma.

Velmi mocný nástroj pro správu MySQL je volně šiřitelný projekt phpMyAdmin. Je napsán v jazyce PHP, díky čemuž je nezávislý na platformě a je dostupný přes jakýkoliv prohlížeč kdekoliv na světě. Jeho nastavení je velmi snadné, ovládání intuitivní a vývojářům je tak poskytnut mocný nástroj pro vytváření a správu databáze. Pro Firebird a InterBase existuje podobný projekt ibWebAdmin, ale není zdaleka tak propracovaný, jako phpMyAdmin.

Obě databáze by sice šly ovládat pomocí ODBC ovladače, ale musela by se vytvořit další aplikace, takže tuto možnost jsem zavrhl a veškeré uživatelské operace budou prováděny pomocí vlastní aplikace.

Po shrnutí vlastností databázových systémů MySQL a Firebird, po porovnání programů vhodných pro správu databáze, zhodnocení obtížnosti instalace, zjištění rozsáhlosti dokumentace a na základě osobních zkušenosti jsem se rozhodl použít pro projekt „Návrh databáze firemních zákazníků“ databázový systém MySQL. Při výběru jsem samozřejmě zohlednil i požadavky na principy fungování, které by například mohly vyžadovat použití uložených procedur.

4.2 Návrh datového modelu

Cílem entito-relačního datového modelu je vytvořit popis modelované reality pomocí entit, atributů a relací mezi nimi. Entitou je každý významný, rozlišitelný a identifikovatelný objekt reality. Atributem je myšlena vlastnost či charakteristika entity a relace. Relací je chápáno vyjádření vztahu mezi entitami.

Při návrhu datového modelu jsem postupoval podle doporučovaných postupů.

4.2.1 Definice datových objektů, entit

Hlavními entitami, vyskytujícími se v tomto datovém modelu, jsou:

- Zákazník,
- Kampaň,
- Nabídka,
- Smlouva,
- Zaměstnanec.

A dále jsou využívány datové objekty Kontakt, Typ, Oslovení, PSČ, Kraj a Okres. S názvy entit a objektů korespondují názvy použitých tabulek.

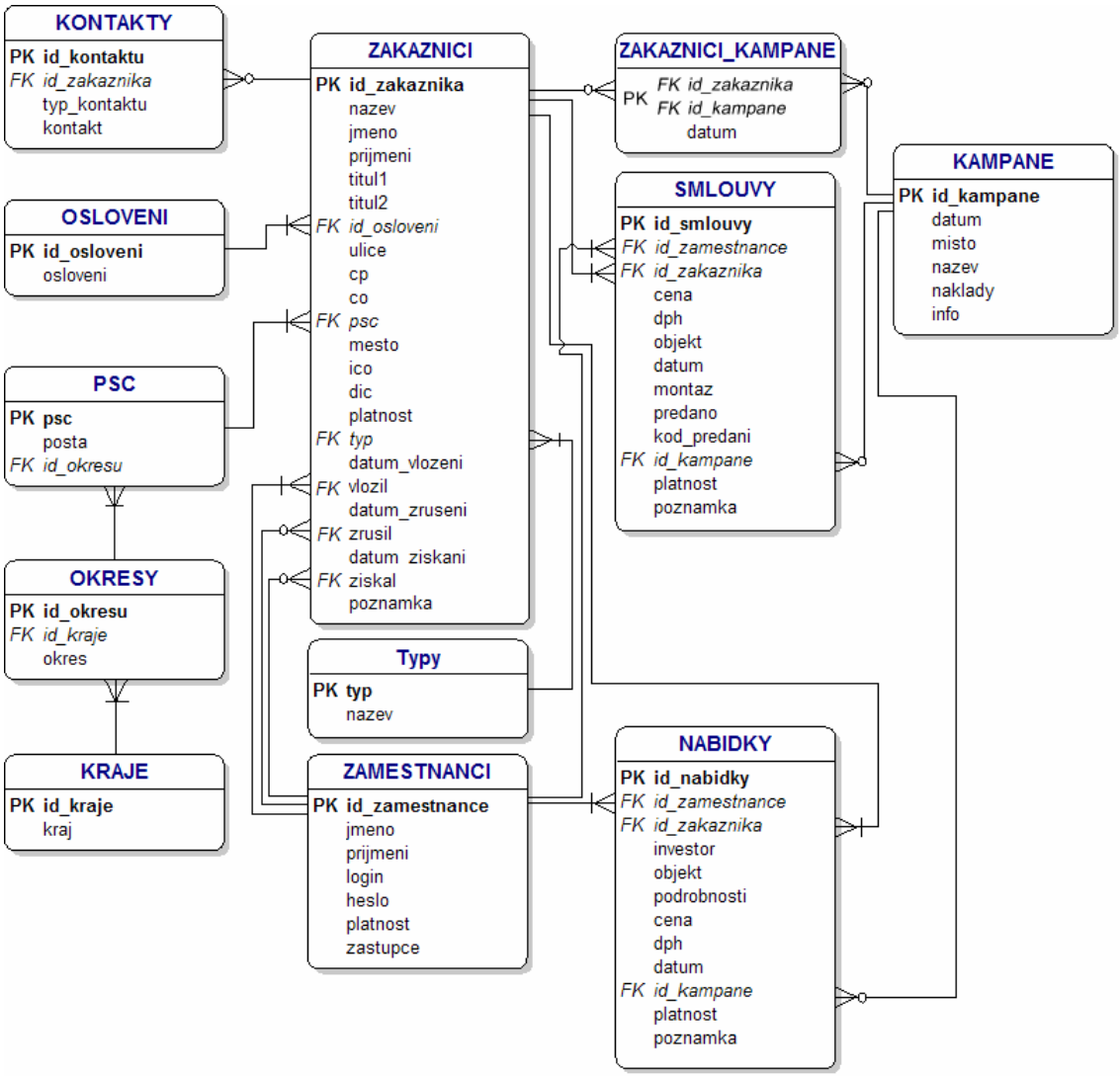
4.2.2 Diagram entito – relačního modelu

V entito-relačním diagramu je vazba N:M (u entit Kampaň a Zákazník) ošetřena dekompozicí. Diagram je zaznamenán způsobem nazývající se Zjednodušený styl.

Integritní omezení tohoto modelu:

- Zákazníka do databáze vložil právě jeden zaměstnanec.
- Zákazníka pro firmu získal právě jeden nebo žádný zaměstnanec.
- Záznam „zákazník“ může zneplatnit právě jeden zaměstnanec.
- Jeden zákazník má nula nebo více kontaktů (záznamů v tab. Kontakty).
- Zákazník má nastaven právě jeden druh oslovení.
- Ke každému zákazníkovi je přiřazeno právě jedno PSČ.
- PSČ je jedinečné a vyskytuje se právě v jednom okrese.
- Okres se nachází právě v jednom kraji.
- Zákazník spadá právě do jednoho typu zákazníků.
- Zákazníkovi je vytvořeno nula nebo více nabídek.
- Nejvýše jedna kampaň smí být přiřazena k jedné nabídce.
- Nabídku uzavřel právě jeden zaměstnanec.
- Se zákazníkem je vytvořeno nula nebo více smluv.
- Nejvýše jedna kampaň smí být přiřazena k jedné smlouvě.

- Smlouva není podmíněna záznamem v tabulce Nabídky.
- Smlouvu uzavřel právě jeden zaměstnanec.
- Jednou kampaní může být osloveno nula nebo více zákazníků.
- Jeden zákazník může být jednou kampaní osloven nejvýše jednou.



Obrázek 1: Diagram entito-relačního modelu

4.2.3 Definice primárních klíčů jednotlivých entit a datových objektů

U žádných jiných tabulek než PSČ, Kraj a Okres neexistuje přirozený kandidátní klíč, proto je u všech zbývajících tabulek zaveden primární klíč umělý. U dekompoziční tabulky „Zakaznici_kampane“ jsem primární klíč vytvořil složením primárních klíčů tabulek Zákazník a Kampaň.

Všechny klíče (vyjma PSČ) jsou celočíselného typu. Povoleny jsou nezáporná čísla (UNSIGNED), která jsou většinou zleva doplněna nulami na požadovanou délku (ZEROFILL). Příkaz AUTO_INCREMENT zajišťuje automatické navýšení hodnoty primárního klíče pro každou nově vkládanou položku.

U tabulek Okresy a Kraje jsou za primární klíče použity klasifikace CZ-NUTS. Jsou nedílnou částí evropské klasifikace NUTS a popisují územní jednotky České republiky podle pravidel daných Nařízením Evropského parlamentu a Rady (ES) č. 1059/2003.

entita / datový objekt	název primárního klíče
Zákazník	id_zakaznika
Kampaň	id_kampane
Zaměstnanec	id_zamestnance
Kontakt	id_kontaktu
Nabídka	id_nabidky
Smlouva	id_smlouvy
Oslovení	id_osloveni
PSČ	psc (poštovní směrovací číslo)
Kraj	id_kraje (identifikátor kraje NUTS)
Okres	id_okresu (identifikátor okresu NUTS)
Typy	typ

Tabulka 3: Stanovení primárních klíčů jednotlivých entit a datových objektů

4.2.4 Definice atributů

Pro popis atributů jsem se rozhodl použít jazyk SQL v podobě příkazů pro vytvoření tabulky. Je z nich patrná struktura tabulky i vlastnosti jednotlivých atributů. Všechny tabulky jsou normalizovány do 4. normální formy.

Zákazníci

Entita Zákazník je ukládána v tabulce Zákazníci. Ta se skládá z 23 atributů seřazených podle logiky vyplývajících z podstaty věci. Každá věta bude obsahovat jak základní informace (jméno, název firmy, adresa, datum vložení záznamu, …), tak i data nesoucí informace podstatné pro činnost firmy. Kromě jiného lze k libovolnému záznamu přidat poznámku nebo daného zákazníka zneplatnit nastavením

patřičného příznaku. Kontakty (telefon, mobil, fax, …) jsou zaneseny do stejnojmenné tabulky, která je s tabulkou Zákazníci v relačním vztahu N:1.

```
CREATE TABLE zakaznici (
  id_zakaznika mediumint(8) unsigned zerofill NOT NULL
auto_increment,
  nazev varchar(128) NOT NULL,
  jmeno varchar(18) NOT NULL,
  prijmeni varchar(24) NOT NULL,
  titul1 varchar(14) NOT NULL,
  titul2 varchar(14) NOT NULL,
  id_osloveni tinyint(3) unsigned default NULL,
  ulice varchar(64) NOT NULL,
  cp varchar(12) NOT NULL,
  co varchar(12) NOT NULL,
  psc varchar(5) default NULL,
  mesto varchar(36) NOT NULL,
  ico varchar(8) NOT NULL,
  dic varchar(10) NOT NULL,
  platnost enum('A','N') default 'A',
  typ tinyint(3) unsigned zerofill default NULL,
  datum_vlozeni date NOT NULL,
  vlozil tinyint(3) unsigned zerofill NOT NULL,
  datum_zruseni date default NULL,
  zrusil tinyint(3) unsigned zerofill default NULL,
  datum_ziskani date default NULL,
  ziskal tinyint(3) unsigned zerofill default NULL,
  poznamka text NOT NULL,
  PRIMARY KEY  (id_zakaznika)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Kontakty

Kontakty by mohly být sice součástí tabulky Zákazníci, ale protože zdroje jednotlivých dat nemají stejnou strukturu (seznam architektů neobsahuje email, seznam obcí neobsahuje telefonní ani faxová čísla, atp.), bylo by mnoho atributů nevyužitých, a naopak by se mohlo stát, že by pro některé kontakty atributy chyběly.

Struktura tabulky bude prostá. Bude obsahovat jednoznačný identifikátor záznamu, ID majitele záznamu, informaci o typu kontaktu a samotnou hodnotu kontaktu. Jak již bylo zmíněno, bude tato tabulka v relačním vztahu k tabulce Zákazníci N:1, tedy jeden zákazník má mnoho kontaktů, a jeden kontakt patří právě jednomu zákazníkovi. Ačkoliv je atribut „typ_kontaktu“ číselníkového typu, využil jsem pro něj datový typ ENUM, který je při malém a ustáleném množství prvků vhodnější, než vazba na další tabulku.

```
CREATE TABLE kontakty (
  id_kontaktu int(10) unsigned NOT NULL auto_increment,
  id_zakaznika mediumint(8) unsigned zerofill NOT NULL,
  typ_kontaktu enum('tel','mob','fax','email') default 'tel',
  kontakt varchar(48) NOT NULL,
  PRIMARY KEY  (id_kontaktu)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Zaměstnanci

Jde o jednoduchou evidenci zaměstnanců. Primárním cílem je získat seznam uživatelů, kteří mohou s databází pracovat a druhým cílem je mít seznam zaměstnanců, kteří mají za úkol získávat klienty a jejichž jména jsou nabídnuta ve výběrovém listu při zakládání věty do tabulky Nabídky či Smlouvy. Obě poslání se nevylučují.

```
CREATE TABLE zamestnanci (
  id_zamestnance tinyint(3) unsigned zerofill NOT NULL
auto_increment,
  jmeno varchar(12) NOT NULL,
  prijmeni varchar(16) NOT NULL,
  login varchar(12) NOT NULL,
  heslo varchar(32) NOT NULL,
  platnost enum('A','N') NOT NULL default 'A',
  zastupce enum('A','N') NOT NULL default 'N',
  PRIMARY KEY  (id_zamestnance)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Nabídky

Po vyslovení zájmu o firmou nabízené produkty je zákazník kontaktován, navštíven a je-li vysloven zájem o vysoušení objektů systémem, je vystavena nabídka. Skladba atributů tabulky Nabídky je ovlivněna hlavním významem této tabulky, a to formulací dopisu.

```
CREATE TABLE nabidky (
  id_nabidky smallint(5) unsigned zerofill NOT NULL
auto_increment,
  id_zamestnance tinyint(3) unsigned zerofill NOT NULL,
  id_zakaznika mediumint(8) unsigned zerofill NOT NULL,
  investor varchar(48) NOT NULL,
  objekt varchar(64) NOT NULL,
  podrobnosti text NOT NULL,
  cena mediumint(8) unsigned NOT NULL,
  dph tinyint(3) unsigned default NULL,
  datum date NOT NULL,
  id_kampane tinyint(3) unsigned zerofill NULL,
  platnost enum('A','N') NOT NULL default 'A',
  poznamka varchar(255) NOT NULL,
  PRIMARY KEY  (id_nabidky)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Smlouvy

Tato tabulka bude uchovávat data o uskutečněných zakázkách. Evidovány zde budou základní časové informace, údaje o vysoušeném objektu, objednavateli a ceně, za kterou byla zakázka uskutečněna. Jde o nejvýznamnější zdroj informací o účinnosti reklamních strategií. Agregačními funkcemi se budou počítat celkové příjmy a výdaje, a stejně tak příjmy a výdaje v přepočtu na jednoho zákazníka.

```
CREATE TABLE smlouvy (
  id_smlouvy smallint(5) unsigned zerofill NOT NULL
auto_increment,
  id_zamestnance tinyint(3) unsigned zerofill NOT NULL,
  id_zakaznika mediumint(8) unsigned zerofill NOT NULL,
  cena int(10) unsigned NOT NULL,
  dph tinyint(3) unsigned NOT NULL,
  objekt varchar(255) NOT NULL,
  datum date NOT NULL,
  montaz date NOT NULL,
  predano date NOT NULL,
  kod_predani varchar(4) NOT NULL,
  id_kampane tinyint(3) unsigned zerofill NULL,
  platnost enum('A','N') NOT NULL default 'A',
  poznamka varchar(255) NOT NULL,
  PRIMARY KEY  (id_smlouvy)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Kampaně

Struktura tabulky obsahuje atributy ukládající informace o datu kampaně, místu, názvu a nákladech. Součástí tabulky je i položka pro krátkou textovou informaci.

```
CREATE TABLE kampane (
  id_kampane tinyint(3) unsigned zerofill NOT NULL
auto_increment,
  datum date NOT NULL,
  misto varchar(64) NOT NULL,
  nazev varchar(128) NOT NULL,
  naklady int(10) unsigned NULL,
  info varchar(255) NOT NULL,
  PRIMARY KEY  (id_kampane)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Zákazníci - Kampaně

Jelikož může být jeden klient osloven více kampaněmi a jedna kampaň osloví více zákazníků, bylo nutné tento vztah upravit dekompoziční tabulkou, jedinou v tomto datovém návrhu. Primárním klíčem je klíč složený z primárních klíčů tabulek Kampaně a Zákazníci. V datových větách této tabulky jsou uloženy informace o tom, kterými

kampaněmi byl klient osloven a naopak. Tabulka neslouží k uchovávání informací, zda základě této kampaně byla na zakázka uskutečněna či nikoliv.

```
CREATE TABLE zakaznici_kampane (
  id_kampane tinyint(3) unsigned zerofill NOT NULL,
  id_zakaznika mediumint(8) unsigned zerofill NOT NULL,
  datum date NOT NULL,
  PRIMARY KEY  (id_kampane,id_zakaznika)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

PSČ

Jedná se o běžný seznam českých pošt aktualizovaný k 31.3.2006. Primárním klíčem je PSČ. Tabulka je doplněna o cizí klíč „id_okresu“, díky němuž je možné lokalizovat dané město (poštu) na úrovni okresu[4].

```
CREATE TABLE psc (
  psc char(5) NOT NULL,
  posta varchar(39) NOT NULL,
  id_okresu char(6) NOT NULL,
  PRIMARY KEY  (psc)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Okresy

Byť jsou okresní jednotky již málo využívaným členěním státu, stále jde o velmi užitečnou informaci. Primárním klíčem je zvolen atribut s názvem CZNUTS. Popisuje územní jednotky České republiky podle pravidel daných Nařízením Evropského společenství a Rada (ES) č. 1059/2003[5].

```
CREATE TABLE okresy (
  id_okresu char(6) NOT NULL,
  id_kraje char(5) NOT NULL,
  okres varchar(21) NOT NULL,
  PRIMARY KEY  (id_okresu)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Kraje

Jedná se o tabulku obdobnou tabulce Okresy, avšak místo seznamu okresů jsou evidovány kraje. Je tedy zřejmé, že se jedná o tabulku se 14-ti záznamy a její obsah by neměl být měněn, stejně jako u tabulky Okresy.

```
CREATE TABLE kraje (
  id_kraje char(5) NOT NULL,
  kraj varchar(23) NOT NULL,
```

[4] Lokalizace je možná pouze v rámci České republiky (firma nepůsobí mimo ČR)
[5] Aktualizace klasifikace CZ-NUTS podle EU byla oznámena ve Sdělení č. 228/2004 Sb.

```
  PRIMARY KEY  (id_kraje)
) ENGINE=InnoDB DEFAULT CHARSET=cp1250;
```

Typy

Číselník obsahuje odkaz na typ zákazníka. Typickými typy zákazníka jsou: občan, firma, obecní úřad, architekt, farnost, …

```
CREATE TABLE `typy` (
`typ` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`nazev` VARCHAR( 14 ) NOT NULL
) ENGINE =InnoDB DEFAULT CHARSET=cp1250;
```

4.3 Import dat

Předchozí data byla uložena v souborech typu MDB[6], které byly vyexportovány do jednoduchého textového souboru s daty rozdělenými středníkem a ohraničenými apostrofy. Tím se jednoznačně označily jednotlivé atributy, a díky tomu je bylo možné importovat do nové databáze. Jelikož ale byla data o zákaznících rozložena do tabulek s rozlišnou strukturou, musel být každý z vyexportovaných souborů upraven tak, aby odpovídal struktuře nové tabulky. V první řadě se jednalo o odstranění nadbytečných atributů, což jsem učinil pomocí tabulkového programu Microsoft Excel, v řadě druhé o rozložení složených atributů. Na tyto operace jsem již byl nucen použít skriptovací jazyk PHP a jeho regulární výrazy. Pomocí nich jsem také dokázal automaticky určit vhodný typ oslovení. Kontaktní údaje jsem z jednotlivých záznamů vyčlenil a do tabulky Kontakty ukládal během provádění PHP skriptu, který záznamy importoval. Nebýt omezení v podobě kontaktů patřících do jiné tabulky, stačilo by pro import dat použít příkaz MySQL – LOAD DATA LOCAL INFILE.

Je možné, že tato funkce bude použita při pozdějším hromadném vkládání dat. Není však možné připravit databázi (aplikaci) tak, aby univerzálně zpracovala jakýkoliv soubor. Tudíž hromadný import dat je operace, která si zpravidla vyžádá účast programátora. To lze sice považovat za slabinu systému, ale má svůj důvod: není nikterak zajištěno, že nově příchozí data budou mít sourodou strukturu, a nemá tedy smysl systém na tuto situaci připravovat.

Import nabídek a smluv byl jednorázovou operací, využívající stejné postupy, jako import zákazníků. Naplnění tabulky PSČ jsem provedl exportem dat z datového

[6] Jedná se o datové soubory, typicky využívané programem Microsoft Access

souboru získaného na stránkách České pošty, tabulky Okresy a Kraje jsem naplnil manuálně, přičemž zdrojem dat mi byla Klasifikace územních statistických jednotek publikovaná na webových stránkách Českého statistického úřadu.

Problém, avšak netechnického charakteru, nastal až při snaze naplnit daty tabulku Kampaně. Firma má totiž velmi špatně zdokumentované jednotlivé akce, takže se podařilo získat jen nekompletní seznam propagačních a reklamních akcí.

4.4 Aplikační rozhraní

Zavedení nového databázového systému by nemělo smysl, kdyby nad databází neexistovala aplikace, která by s daty pracovala. Nabízí se několik možností, jak data spravovat. Mezi tyto varianty patří:

- Propojení databáze s programem MS Access pomocí ODBC ovladače.
- Vytvoření speciální binární aplikace pracující s MySQL.
- Využití webového serveru, skriptovacího jazyka a internetového prohlížeče.

Nejschůdnější variantou bylo naprogramování sady PHP skriptů a jejich umístění na webový server (byť se nachází v intranetové síti). Obecný princip tohoto modelu je takový, že z webového prohlížeče přicházejí požadavky na webový server. Ten je předává PHP, to se - v předem připravených skriptech - připojuje do MySQL, čímž jsou s databází vykonávány požadované operace, a jako výstup je sestavován XHTML soubor, který je vracen uživateli do webového prohlížeče.

4.4.1 Výběr potřebných programů

U všech programů lze vybírat mezi několika produkty, a to jak mezi produkty nabízenými zdarma, tak i mezi produkty komerčními. Neplatí však, že by programy nabízené za poplatek byly lepší, než ty, které jsou vyvíjeny dobrovolníky (obvykle širokou komunitou vývojářů).

Web-server

Web-server nebylo zapotřebí dlouze vybírat. Zvolil jsem osvědčený Apache (používán je na 56% webových serverů[7]) ve verzi 2.0.54 pro platformu WIN32. Umístili jsme jej na starší počítač s operačním souborem Windows XP (ve firmě byly nadbytečné licence, tudíž se nejednalo o investici). Jelikož jsou PHP skripty i databáze platformově nezávislé, je možné vyměnit operační systém za systém UNIXového typu.

FTP server

Je velmi pravděpodobné, že v budoucnu bude nutné spravovat programové skripty, editovat konfigurační soubory a provádět zálohy databáze. Jedna z možností, jak toto provádět na dálku, je instalace FTP serveru. Vybral jsem mnou vyzkoušený program CesarFTP 0.99g. Byť je pět let starý, je velmi stabilní a jednoduše nastavitelný.

MySQL

Databázový systém byl nainstalován ve verzi 5.0.27. Za výchozí znakovou sadu, používanou při porovnávání a transformaci textových řetězců, jsem zvolil cp1250_czech_cs. Díky použití této sady jsou při vyhledávání a seřazování řetězců rozlišována velká a malá písmena. Jako typ datového úložiště jsem vybral InnoDB.

PHP

Mnou použitá verze PHP je 5.2.0, který běží jako modul web-serveru.

4.4.2 Popis aplikace

Aplikace bude dostupná jen na počítačích připojených do lokální sítě. Pokud by bylo nutné pracovat s aplikací vzdáleně (není tím myšlen upgrade PHP skriptů), dalo by se to řešit pomocí programů pro vzdálenou správu počítače nebo pomocí VPN.

Přístup je podmíněn HTTP autentizací. Přihlašovací jméno a heslo jsou ověřovány se záznamy v tabulce Zaměstnanci. Výhodou HTTP autentizace je evidence přihlášení již na úrovní serveru a ukládání této informace do log souboru. Jde tak dohledat informace o tom, kdo a kdy si vyžádal určitou stránku.

[7] Podle průzkumu anglické společnosti NETCRAFT LTD provedeného 1. května 2007 – viz. http://news.netcraft.com/archives/2007/05/01/may_2007_web_server_survey.html

Úspěšně přihlášeným uživatelům je poskytnut přístup k aplikaci. Má jednoduché grafické rozhraní navržené pro rozlišení monitoru 1024x768 pixelů a více. Web je rozčleněn na pět základních částí (přičemž každá je rozdělena na několik dalších):

- Zákazníci,
- Smlouvy,
- Nabídky,
- Kampaně,
- Zaměstnanci.

Struktura PHP skriptů je koncipována tak, že do záhlaví každého souboru je vložen soubor, který obsahuje vlastní knihovny s PHP funkcemi. Dále je v záhlaví souboru umístěn příkaz pro započetí „kreování“ obsahu stránky do paměti PHP, což přináší jisté výhody. Na konci souboru je „bufferování“ ukončeno, obsah cache předán funkci, která sestaví kompletní XHTML stránku.

4.5 Příklady výběrových a statistických dotazů

Posláním uložených procedur je zjednodušit práci s databází. V jakékoli databázi by však měly jen pramalý význam, kdyby se jim nedaly posílat parametry. Vybral jsem tři uložené procedury, které jsou v navržené databázi použity. Zda je vykonávání dotazů v uložených procedurách rychlejší oproti jejich rozepsání ve skriptech aplikace nelze jednoduše stanovit. Umístění SQL dotazů mimo aplikaci ji činí přehlednější. Z hlediska bezpečnosti jsou uložené procedury nesrovnatelně lepší.

4.5.1 Výběr zákazníků z určitého kraje

Procedura vrací seznam zákazníků, kteří se nacházejí v daném kraji. Jelikož jde vždy o tisíce záznamů, je výsledek tohoto dotazu vracen v množství 30-ti položek, přičemž parametrem „od“ určuji pozici, od které má databáze odpovídající záznamy vracet. Jako druhý parametr (v pořadí jako první) je do uložené procedury předáván NUTS identifikátor kraje (uživatel jej nepřímo zná díky jeho výběru ze seznamu krajů).

Procedura je pojmenována jako „vyber_podle_kraje“. Postavena je tak, že ze seznamu okresů vybere ty, jež přísluší danému kraji. Dále pak v jednotlivých

okresech vybere všechny poštovní směrovací čísla, která jsou následně porovnávána s obsahem tabulky Zákazníci. Jelikož se jedná o dotaz s dvojím subselectem, může být jeho zpracování výkonnostně náročné.

```
CREATE PROCEDURE vyber_podle_kraje (kraj varchar(5), od tynint)
BEGIN
     SELECT id_zakaznika FROM zakaznici WHERE psc IN(
          SELECT psc FROM psc WHERE id_okresu IN (
             SELECT id_okresu FROM okresy WHERE id_kraje=kraj
          )
     )
     WHERE platnost='A'
     ORDER BY id_zakaznika DESC
     LIMIT od,30;
END;
```

Příklad volání procedury:

```
CALL vyber_podle_kraje('CZ072', 0);
```

4.5.2 Příjmy získané na základě reklamních kampaní

Konfrontace nákladů a příjmů reklamních akcí je pro management podniku informace značně důležitá. Na jejím základě se mohou rozhodovat, ve kterém typu reklamních kampaní je dobré nadále pokračovat, a které jsou naopak ztrátové.

Předávaným parametrem je ID zkoumané kampaně.

```
CREATE PROCEDURE statistika_kampane (kampan tynint)
BEGIN
     SELECT
          SUM(cena) as 'příjmy',
          COUNT(id_smlouvy) as 'počet zákazníků'
     FROM smlouvy JOIN kampane USING(id_kampane)
     WHERE platnost='A' AND kampane.id_kampane=kampan;
END;
```

4.5.3 Přehled příjmů v jednotlivých kvartálech

Procedura vrací seznam příjmů získaných v jednotlivých ročních kvartálech. Sestavuje se z dat tabulky Smlouvy. Výsledek je seřazen sestupně podle let a kvartálu.

```
CREATE PROCEDURE kvartalni_prijmy ()
     SELECT
          DATE_FORMAT(datum,"%Y")as 'Rok',
          QUARTER(`datum`) as 'Kvartál',
          SUM(cena) as 'Příjmy'
     FROM `smlouvy`
     WHERE platnost='A'
     GROUP BY CONCAT(DATE_FORMAT(datum,"%Y"),QUARTER(`datum`))
     ORDER BY Rok DESC, Kvartál DESC;
END;
```

5 Přínosy návrhu řešení

Navrhované řešení odpovídá současnému trendu řešení obdobných aplikací. Kombinace programů Apache + PHP + MySQL je osvědčenou volbou, která je prověřena statisíci instalací. Základní podmínkou pro stabilní běh databáze je správná architektura aplikace, což obnáší řadu pravidel, které by neměly být porušovány.

V navržené databázi se však s největší pravděpodobností nestane, že její běh zkolabuje. Mé tvrzení je opřeno o předprojektovou analýzu, v níž jsem se spolu s vedením firmy zabýval otázkou, kolik uživatelů bude databázi využívat. Odpověď byla taková, že v jeden okamžik bude s databází pracovat jeden či maximálně dva uživatelé. To je tak zanedbatelné množství, že by odpovídající výkon poskytlo mnoho jiných databázových systémů. Tím obhajuji kritérium použité při výběru nejvhodnějšího systému – vlastní zkušenost s konkrétními produkty.

Ačkoliv se jedná o databázi s malým počtem tabulek, celkový počet záznamů koketuje s hranicí padesáti tisíci položek. To vypovídá o nutnosti použít systém, který dokáže hlídat relační vztahy mezi jednotlivými entitami, čemuž MySQL také vyhovuje (ale v režimu ukládání dat InnoDB).

Obecně vzato má vývoj systému z finančního hlediska odpovídat míře důležitosti dat, potažmo míře zodpovědnosti dodavatele při selhání systému. Mé vztahy s firmou AQUPAPOL nejsou pouze na čistě obchodní úrovni a to se také promítlo na honoráři, který byl za kompletní projekt vyplacen. Částka 10 000 Kč by v reálném světě činila třetinu až pětinu ceny, kterou by za stejný produkt mohla požadovat jakákoliv firma, zabývající se vývojem webových aplikací. Je totiž obecně známo, že se průměrná měsíční mzda programátora se znalostmi jazyků PHP, SQL, XHTML a JavaScript, pohybuje nad hranicí 20 000 Kč. Samozřejmě jde o tzv. částku „čistou“, takže reálné náklady jsou ještě vyšší. Připočteme-li potřebu vytvořit firmě zisk a fakt, že jednomu programátorovi zabere práce tři až čtyři týdny, lze částku 10 000 Kč považovat za velmi přátelskou.

Investice do hardware a software byly takřka nulové. Jako webový server nyní slouží nevyužitý desktopový počítač, na němž je nainstalována obdobně nadbytečná licence operačního systému Windows XP. Ten samozřejmě není nejvhodnějším

systémem na kterém by měl webový server běžet, ale vybrali jsme jej záměrně, a to proto, že v našem okolí nebyl nikdo, kdo by disponoval dostatečnými znalostmi pro nasazení některého z operačních systémů UNIXového typu. Nic však nebrání tomu, aby byl v budoucnu vyměněn stávající systém za jiný.

Nejmarkantněji se jednotlivé přínosy ukázaly po srovnání předchozí aplikace se současnou:

- Těžkopádné prostředí formulářů nahradilo příjemné a intuitivní grafické rozhraní.
- Jednoduché nalezení zákazníka podle strohých informací.
- Vyhledávání podle více parametrů.
- Přehledná evidence o tom, který zákazník byl jakou kampaní osloven.
- Generování smluv a nabídek ve formátu PDF s možností automatického odeslání na zákazníkův email.
- Možnost vzdálené práce s aktuální verzí databáze pomocí internetu.
- A další …

Za největším z přínosů považuji vytvoření nástrojů, které vyhodnocují reklamní kampaně. Management firmy tím získá dostatečný přehled o tom, jak se jim jednotlivé akce vyplácí.

Závěr

Na otázku, zda navrhované řešení splnilo představy majitelů firmy, není zatím přesná odpověď. Vývoj aplikace byl samozřejmě průběžně konzultován, takže k výraznějšímu odklonu od požadovaných představ nedošlo. Je však otázkou času, až bude definitivně zodpovězena otázka, zda vytvořená aplikace pohltila veškeré nedostatky, kvůli kterým se nový systém vyvíjel.

Nejpravděpodobnější odpovědí bude však „Ne“. Je takřka jisté, že jsme něco opomněli, že něco musí být jinak a také to, že něco budeme chtít mít jinak. Lépe řečeno jsme na něco zapomněli, něco udělali špatně a něco jiného jsme chybně vyhodnotili. Plurál jsem použil schválně, neboť v tento okamžik nejsou mnou, ani firmou žádné chyby identifikovány a **systém je považován za správně fungující celek**. Tím pádem nelze s jistotou říci, zda nově objevená chyba bude mít původ v nepřesném zadání či v rukou programátora.

Největší slabinou této aplikace je zálohování. Prozatím je řešeno hromadným exportem databáze do SQL dotazů ukládaných do komprimovaného textového souboru. Jde o řešení neefektivní a to už z těch důvodů, že není automatizované, databáze je rozsáhlá a data jsou stále fyzicky uložena na témže pevném disku! Jde tedy o první bod zapsaný do seznamu následujících oprav systému.

Nutno podotknout, a je tomu nastíněno již v předchozí kapitole, že projekty podobného rozsahu nejsou obvykle prací, kterou by vykonával jediný člověk. Aplikace by si zasloužila projít odbornýma rukama grafika, databázového specialisty, odborníka na CSS layout i kvalitního PHP programátora. Rozložením práce i zodpovědnosti by každý z nich mohl nasadit do projektu méně sil, což by bylo rozhodně jen k užitku.

Nicméně se podařilo i přes absenci týmu odborníků vytvořit aplikaci, která se umí k datům o zákaznících firmy chovat tak, jak si její majitelé přáli:

- Přehledná evidence všech zákazníků.
- Snadné vyhodnocování úspěšnosti reklamních akcí.
- Vyšší bezpečnost uložených dat.

Jako nejobtížnější dílčí úkol považuji import stávajících dat. Do té doby používané tabulky byly totiž získány v průběhu několika let, což je důvod toho, že tabulky neměly jednotnou strukturu, několik atributů bylo duplicitních či nadbytečných, mnoho dalších nemělo zcela jasný význam a spoustě jiným chyběla atomičnost. Tím se mi opětovně potvrdila všeobecně známá teorie o správně normalizovaném datovém modelu. Je totiž velmi obtížné pracovat s databází, jejíž model navrhnul člověk, který o normalizaci snad ani nikdy neslyšel.

O to víc si teď vážím odborníků na databázovou problematiku. Jak jsem se v teoretické části přesvědčil, jsou databázové technologie tak rozsáhlou oblastí, že rozhodně není možné tuto problematiku vstřebat v krátkém časovém období, jakým je například jeden rok.

Seznam použité literatury

Písemné zdroje publikované

Knihy

(1) CÍSAŘ, Pavel. *InterBase/FireBird : podrobná příručka : tvorba, programování a správa databází.* 1. vyd. Brno : Computer Press, 2003. 453 s., 1 elektronický optický disk. ISBN 80-7226-946-1.

(2) DARIE, Cristian. *AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně.* Zoner Press, 2006. 320 s. ISBN 80-86815-47-1.

(3) DELISLE, Marc. *PhpMyAdmin : efektivní správa MySQL.* Přeložil Jan Pokorný. 1. vyd. Brno : Zoner Press, 2004. 264 s. ISBN 80-86815-09-9.

(4) GILMORE, W. Jason. *Velká kniha PHP5 a MySQL : Kompendium znalostí pro začátečníky i profesionály*. Překlad RNDr. Jan Pokorný. 1. vyd. Brno : ZONER software s.r.o., 2005. 711 s. Encyklopedie webdesignera. ISBN 80-86815-20-X.

(5) HOULETTE, Forrest. *SQL Příručka programátora*. Praha: SoftPress, 2001. 384 s. ISBN 80-86497-14-3.

(6) KOCH, M. *Datové a funkční modelování.* 1.vyd. Brno: Vysoké učení technické v Brně - Fakulta podnikatelská, 2004. s. (108 s.) ISBN: 80-214-2724-8.

(7) SCHLOSSNAGLE, George . *Pokročilé programování v PHP 5*. Překlad j. Gregor, J. Kuklínek, V. Šimek a M. Vokoun. 1. vyd. Brno: Zoner Press, 2004. 640 s. ISBN 80-86815-14-5.

(8) STANÍČEK, Petr. *CSS Kaskádové styly : Kompletní průvodce*. 2. vyd. Brno: Computer Press, 2003. 178 s. ISBN 80-7226-872-4.

(9) ŠIMŮNEK, Milan. *SQL : Kompletní kapesní průvodce.* 1. vyd. Praha : Grada Publishing, 1999. 248 s. ISBN 80-7169-692-7.

(10) YARGER, Randy Jay, REESE, George, KING, Tim. *MySQL and mSQL*. [s.l.]: O'Reilly Media, 1999. 496 s. ISBN 1-56592-434-7.

Časopisy

(11) LOUDA, Pavel. *Informační systémy ve finančním sektoru*. COMPUTERWORLD. 6.10.2006, roč. XVII, č. 32, s. 17-19.

(12) VOKŮRKOVÁ, Lenka. *Skončit na půli cesty se nevyplácí*. COMPUTERWORLD. 6.10.2006, roč. XVII, č. 32, s. 10.

Zákony a vyhlášky

(13) Zákon č. 480/2004 Sb. *o některých službách informační společnosti* platný k 7.9.2004

Sborníky a jiné neperiodické publikace

(14) STANÍČEK, Petr. *Manifest Dogma W4*. [online]. 2002, poslední revize 26.2.2003 [cit. 2006-10-30]. Dostupné z: <http://pixy.cz/dogma/dogmaw41/cs>

Internetové zdroje

(15) CÍSAŘ, Pavel. *PostgreSQL 8.0 - útok na podnikovou sfér* [online]. 16.6.2005 [cit. 2007-05-17]. Dostupný z WWW: <http://www.linuxexpres.cz/praxe/postgresql-8-0-utok-na-podnikovou-sferu>. ISSN 1801-3996.

(16) CÍSAŘ, Pavel. *Relační databázový systém Firebird – výhled do budoucna* [online]. 25.3.2005 [cit. 2007-05-17]. Dostupný z WWW: <http://www.linuxexpres.cz/praxe/relacni-databazovy-system-firebird-vyhled-do-budoucna-1>. ISSN 1801-3996.

(17) *Comparison of relational database management systems* [online]. Wikimedia Foundation, Inc., 2007 , 14.5.2007 [cit. 2007-05-17]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems>.

(18) ČINČURA, Jiří. *Jaká je ta správná databáze?* Databázový svět [online]. 2006 [cit. 2006-10-30]. Dostupný z WWW: <http://www.dbsvet.cz/view.php?cisloclanku=2006030601>.

(19) FALTÝNEK, Lukáš. *Structured Query Language* [online]. 25.1.2007 [cit. 2007-05-17]. Dostupný z WWW: <http://www.linuxexpres.cz/praxe/structured-query-language>. ISSN 1801-3996.

(20) *Integritní omezení a transakce v MySQL* [online]. c2005-2006 , 21.11.2006 [cit. 2007-05-17]. Dostupný z WWW: <http://www.manualy.net/article.php?articleID=17>.

(21) JAKUBČÍK, Ondřej. *Srovnání databázových serverů* [online]. 6.3.2007 [cit. 2007-05-17]. Dostupný z WWW: <http://www.linuxexpres.cz/software/srovnani-databazovych-serveru>. ISSN 1801-3996.

(22) JUN, Adam. *MySQL manuál : Český manuál k relačnímu databázovému systému MySQL. Popis nejužitečnějších funkcí včetně příkladů.* [online]. 1995-2005, 3. března 2005 [cit. 2005-12-07]. Dostupný z WWW: <http://mm.gene.cz>

(23) *Klasifikace CZ-NUTS* [online]. Český statistický úřad, 2007 , 17.3. 2005 [cit. 2007-05-17]. Dostupný z WWW: <http://www.czso.cz/csu/klasifik.nsf/i/3_klasifikace_cz_nuts>.

(24) KOCAN, Marek. *Malý neznamená bezmocný* [online]. AVRE Publishing, spol. s r.o., 2004 , 13.7.2004 [cit. 2007-05-17]. Dostupný z WWW: <http://www.dbsvet.cz/view.php?cisloclanku=2004071301>.

(25) KOLÁŘ, Radim. *Embedded databáze : Rodina databází DBM* [online]. Internet Info, s.r.o., c1998-2007, 8.9.2004 [cit. 2007-05-17]. Dostupný z WWW: <http://www.root.cz/clanky/rodina-databazi-dbm/>. ISSN 1212-8309.

(26) LAZECKÝ, Petr. *Jak na to - Zabezpečení MS Access databáze* [online]. Devmasters s.r.o., c2002-2007 , 21.6.2005 [cit. 2007-05-17]. Dostupný z WWW: <http://blog.vyvojar.cz/lazo/archive/2005/06/21/6193.aspx>.

(27) LEBEDA, Martin. *SQLite - ultra lehké sql* [online]. Internet Info, s.r.o., c1998-2007, 4.8.2003 [cit. 2007-05-17]. Dostupný z WWW: <http://www.root.cz/clanky/sqlite-ultra-lehke-sql/>. ISSN 1212-8309.

(28) *List of relational database management systems* [online]. Wikimedia Foundation, Inc., 2007 , 9.5.2007 [cit. 2007-05-17]. Dostupný z WWW: <http://wikipedia.org/wiki/List_of_relational_database_management_systems>.

(29) MELICHAR, Jan. *Stručná historie PostgreSQL* [online]. AVRE Publishing, spol., 2004 , 14.4.2004 [cit. 2007-05-17]. Dostupný z WWW: <http://www.dbsvet.cz/view.php?cisloclanku=2004041401>.

(30) OLSON, Michael, BOSTIC, Keith, SELTZER, Margo. *Berkeley DB. In Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference*. [s.l.] : [s.n.], 1999. s. 10. Dostupný z WWW: <http://www.usenix.org/events/usenix99/full_papers/olson/olson.pdf>.

(31) POLÁŠEK, Marek. *Databáze z hlediska podnikových informačních systémů*. IT System 7-8/2003 : Data Warehousing a Business Intelligence [online]. 2003 [cit. 2006-10-30]. Dostupný z WWW: <http://office.ccb.cz/site/data-warehousing/7lofelmann.htm>.

(32) *Porovnání způsobů vysoušení vlhkého zdiva* [online]. EStav.cz, c2000-2007 [cit. 2007-05-17]. Dostupný z WWW: <http://estav.cz/zpravy/povoden_16.asp>

(33) SKŘIVAN, Jaromír. *Databáze a jazyk SQL* [online]. 4.8.2000 [cit. 2007-02-20]. Dostupný z WWW: <http://interval.cz/clanky/databaze-a-jazyk-sql/>

(34) STĚHULE, Pavel. *Letmý pohled do uložených procedur (první část)* [online]. Internet Info, s.r.o., c1998-2007, 27.7.2006 [cit. 2007-05-17]. Dostupný z WWW: <http://www.root.cz/clanky/letmy-uvod-do-ulozenych-procedur-mysql-prvni-cast/>. ISSN 1212-8309.

(35) ZAJÍC, Petr. *MySQL (1) - pestrý svět databází* [online]. c2003-2007 , 1.3.2005 [cit. 2007-05-17]. Dostupný z WWW: <http://www.linuxsoft.cz/article.php?id_article=731>. ISSN 1801-3805.

(36) ZAJÍC, Petr. *MySQL (4) – něco terminologie* [online]. c2003-2007 , 11.3.2005 [cit. 2007-05-17]. Dostupný z WWW: <http://www.linuxsoft.cz/article.php?id_article=744>. ISSN 1801-3805.

(37) ŽÁK, Karel. *Historie relačních databází* [online]. Internet Info, s.r.o., c1998-2007, 19.10.2001 [cit. 2007-05-17]. Dostupný z WWW: <http://www.root.cz/clanky/historie-relacnich-databazi/>. ISSN 1212-8309.

(38) ŽÁK, Karel. *Modelování databází* [online]. Internet Info, s.r.o., c1998-2007, 3.4.2002 [cit. 2007-05-17]. Dostupný z WWW: <http://www.root.cz/clanky/modelovani-databazi/>. ISSN 1212-8309.

Seznam použitých zkratek

ADT	Abstraktní Datový Typ
ANSI	American National Standards Institute
API	Application Programming Interface
BSD	Berkeley Software Distribution
Codasyl	Conference on Data Systems Languages
CSS	Cascading Style Sheets
ČSÚ	Český statistický úřad
dbm	Database manager
EULA	End User License Agreement
FK	Foreign Key
GmBH	Gesellschaft mit beschränkter Haftung
GPL	General Public License
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
ISO	International Organization for Standardization
LDAP	Lightweight Directory Access Protocol
Mac OS	Macintosh Operating System
MS	Microsoft
NUTS	Nomenclature of Units for Territorial Statistics
ODBC	Open Database Connectivity
OEM	Object Exchange Model
OID	Object identifier
OS	Operační Systém
PK	Primární klíč (Primary Key)
RDBMS	Relational Database Management System
s.r.o.	Společnost s ručením omezeným
SEQUEL	Structured English Query Language
SQL	Structured Query Language
SW	Software
UC	University of California
UTF	Unicode Transformation Format
VPN	Virtual Private Network
XHTML	Extensible hypertext markup language

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1: Seznam známých relačních databázových systémů

Komerční software

* 4th Dimension
* Alpha Five
* Greenplum
* CA-Datacom
* Dataphor
* Daffodil database
* EnterpriseDB
* eXtremeDB
* DB2
* FastPLUS
* FileMaker
* Greenplum
* Helix database
* Informix
* InterBase
* Kognitio, WX2
* Linter
* Matisse
* Microsoft Jet Database Engine
* Microsoft SQL Server
* Microsoft Visual FoxPro
* Mimer SQL
* mSQL
* Netezza
* NonStop SQL
* Openbase
* Oracle
* Oracle Rdb for OpenVMS
* OpenLink Virtuoso Universal Server
* Pervasive
* Pyrrho DBMS
* Progress 4GL
* Sand Analytic Server (dříve Nucleus)
* SIR
* GUPTA SQLBase
* Sybase Adaptive Server Anywhere
* Sybase Adaptive Server Enterprise
* Sybase Adaptive Server IQ
* Teradata
* ThinkSQL
* TimesTen
* Valentina (Database)
* Vertica
* VistaDB
* VMDS
* Whitecross Systems
* WinBase602
* Oddity Databases

Software poskytovaný zdarma nebo jako open source

* Derby
* CSQL
* Firebird
* FrontBase
* Gladius DB
* H2
* HSQLDB
* Ingres
* Java DB
* MaxDB
* MonetDB
* MySQL
* PostgreSQL
* SmallSQL
* SQLite
* tdbengine
* OpenLink Virtuoso
* txtSQL
* Mckoi SQL Database
* Rebol sql-protocol

Historické systémy

* Britton-Lee IDM
* Oracle Rdb
* Paradox
* PRTV
* QBE
* SQL/DS
* Sybase SQL Server
* Micro DBMS

Příloha 2: Srovnání vybraných relačních databázových systémů

Databázový systém	Verze	Dodavatel	APP1	APP2	APP3	APP4	Cena jednouživatelská	Cena 50 uživatelská
Adabas	7.4.3	Software AG	1	4	4	3	jednotky tisíc	desítky/stovky tisíc
Advantage CA-IDMS/DB Database	2.6	Computer Associates	1	3	5	5	jednotky tisíc	desítky/stovky tisíc
Caché	5.0	InterSystems	2	4	5	4	jednotky tisíc	desítky/stovky tisíc
DB2 Universal Database Version	8.2	IBM	2	4	5	5	zdarma	desítky/stovky tisíc
Firebird	1.5	nezávislí vývojáři	1	3	4	3	jednotky tisíc	zdarma
Informix Dynamic Server	9.4	IBM	1	4	5	5	jednotky tisíc	desítky/stovky tisíc
Ingres	r3	Computer Associates	1	4	3	3	zdarma	zdarma
InterBase	7.1	Borland	1	3	4	3	jednotky tisíc	desítky/stovky tisíc
JDataStore	7	Borland	1	4	3	1	jednotky tisíc	jednotky/desítky tisíc
MaxDB	7.5	MySQL AB	1	4	3	1	zdarma	zdarma
MS Access	2003	Microsoft	5	3	1	1	jednotky tisíc	nemá smysl
MS SQL Server	2000	Microsoft	2	3	5	4	jednotky tisíc	desítky/stovky tisíc
MySQL	4.1	MySQL AB	1	4	3	2	zdarma	zdarma
Oracle Database	10g	Oracle	1	3	5	5	jednotky tisíc	desítky/stovky tisíc
PostgreSQL	7.4.6	nezávislí vývojáři	1	4	3	2	zdarma	zdarma
Progress RDBMS	10	Progress Software	1	3	5	4	jednotky tisíc	desítky/stovky tisíc
Souborový formát DBF[1]	4	--	4	1	1	1	zdarma	zdarma
Sybase Adaptive Server Enterprise	12.5	Sybase	1	2	5	5	jednotky tisíc	desítky/stovky tisíc
Sybase SQL Anywhere Studio	9	Sybase	4	5	3	1	jednotky tisíc	nemá větší smysl
Tamino XML Server	4.2	Software AG	1	5	4	2	jednotky tisíc	desítky/stovky tisíc

Zdroj: http://www.dbsvet.cz/storage/dbs.pdf

[1] s adekvátním vývojovým nástrojem

Příloha 3: Základní rysy vybraných relačních databázových systémů

Databázový systém	Základní rysy				Tabulky a pohledy		Ostatní charakteristika						Partition			
	ACID	Referenční integrita	Transakce	Unicode	Dočasné tabulky	Pohledy	Domain	Kurzory	Trigger	Function	Procedure	External routine	Range	Hash	Composite (Range + Hash)	List
Adaptive Server Anywhere	ano	ano	ano	ano	ano	>10.0	ano	ano	ano	ano	ano	ano	?	?	?	?
Adaptive Server Enterprise	ano	ano	ano	ano	ano	ne	ano	ano	ano	ano	ano	ano	ano	ano	ne	ano
Apache Derby	ano	ano	ano	ano	ano	ne	ne	ano	ano	ano	ano	ano	ne	ne	ne	ne
DB2	ano	ano	ano	ano	ano	ano	ne	ano	ano	ano	ano	ano	ano	ano	ano	ano
Firebird	ano	ano	ano	ano	>2.1	ne	ano	ano	ano	ano	ano	ano	ne	ne	ne	ne
HSQLDB	ano	ano	ano	ano	ano	ne	?	ne	ano	ano	ano	ano	ano	ano	?	?
H2	ano	ano	ano	ano	ano	ne	ano	ne	ano	ano	ano	ano	ano	ano	ano	ano
Informix	ano	ano	ano	ano	ano	ano	?	ano	ano	ano	ano	ano	ne	ne	ne	ne
Ingres	ano	ano	ano	ano	ano	Ing r4	ano	ano	ano	ano	ano	ano	ne	ne	ne	ne
InterBase	ano	ano	ano	ano	ano	ne	ano	ano	ano	ano	ano	ano	ano	ne	ne	ne
MaxDB	ano	ano	ano	ano	ano	ne	ano	ano	ano	ano	ano	?	ano	ano	ano	ne
MS SQL Server	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano
MonetDB	ano	ano	ano	ano [a]	ano	ne	ne	ne	ano	ano	ano	ano	ano	ano	ano	ano
MySQL	ano	ano	ano	ano [a]	ano	ne	ne	ano	ano	ano	ano	ano	ano	ne	ne	ne
Oracle	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano
OpenEdge	ano	ne	ano	ano	ano	ne	?	?	?	?	?	?	ne	ne	ne	ne
OpenLink Virtuoso	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano	ne	ne	ne	ne
PostgreSQL	ano	ano	ano	ano [a]	ano	ne	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano
Pyrrho DBMS	ano	ano	ano	ano	ne	ne	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano
SQLite	ano	ne	Basc4	ano	ano	ne	ne	ne	ano	ne	ne	ano	?	?	?	?
Teradata	ano	ano	ano	ano	ano	ano	ne	ano	ano	ano	ano	ano	ano	ano	ne	ano

Zdroj: http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

[a] Částečně / UTF-8