VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

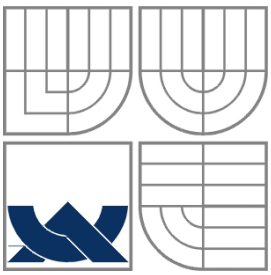
POROVNÁNÍ PŘÍSTUPŮ K UKLÁDÁNÍ DAT VE WEBOVÝCH APLIKACÍCH

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

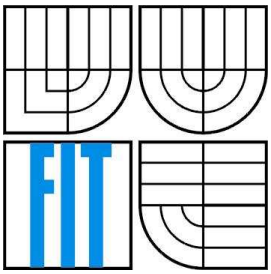
AUTOR PRÁCE
AUTHOR

JAKUB HRANÁČ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

POROVNÁNÍ PŘÍSTUPŮ K UKLÁDÁNÍ DAT VE WEBOVÝCH APLIKACÍCH

COMPARE DATA STORE METHODS IN WEB APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Jakub Hranáč
AUTHOR

VEDOUCÍ PRÁCE — Ing. Vrážel Dušan
SUPERVISOR

BRNO 2008

Abstrakt

Porovnání jednotlivých databází z pohledu vývojáře webové aplikace. Porovnání způsobů, kterými se k databázím přistupuje. Implementace jednoduchého projektu pro práci s různými databázovými servery.

Klíčová slova

MySQL, SQL Server, Oracle, PostgreSQL, bezpečnost, Transact-SQL, PL/SQL, porovnání, web, vývoj

Abstract

Comparing of different database servers from the web application developer’s point of view and different methods of accessing those servers. Implementation of simple project to demonstrate connection to some of the most used databases.

Keywords

MySQL, SQL Server, Oracle, PostgreSQL, security, Transact-SQL, PL/SQL, comparing, comparison, developer, developement, web-based appliacatin

Citace

Hranáč Jakub: Porovnání přístupů k ukládání dat ve webových aplikacích, **bakalářská práce, Brno, FIT VUT v Brně**, 2008

Porovnání přístupů k ukládání dat ve webových aplikacích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Dušana Vrážela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Rád bych poděkoval všem, kteří mi poskytnuli morální oporu při vypracování této práce. Dále vedoucímu, Dušanu Vráželovi, za vedení při práci.

Obsah

1 Úvod

Potřeba ukládat data byla pro aplikace téměř vždy nezbytná. Metoda pro jejich ukládání prošla mnoha změnami od prvních aplikací po moderní databázové stroje. S nástupem internetu a moderních informačních systémů bylo třeba umožnit víceuživatelský přístup k datům, jejich zabezpečení proti poškození i proti neoprávněnému přístupu. V současné době existuje několik databázových systémů, které jsou v široké míře používány. Samozřejmě, každý z těchto systémů je v jisté míře unikátní a od ostatních se odlišuje.

Cílem následujících kapitol je podchytit tyto rozdíly a porovnat možná užití jednotlivých typů databází. Jsou zaměřeny především na Microsoft SQL Server, Oracle, MySQL, XML, PostgreSQL. Jednotlivá řešení jsou porovnány z hlediska nabízených funkcí pro vývojáře, bezpečnosti, efektivnosti a v neposlední řadě ceny a uživatelské podpory.

Následně bude probrán návrh a implementace jednoduché aplikace, která podporuje univerzální rozhraní pro přístup k databázi.

2 Teorie

Při vývoji aplikace s propojením na databázový server se střetneme s několika problémy:

- aplikace má být schopna využít několik různých serverů

  Běžné pro „drobné“ aplikace, které nejsou vyráběny pro jednoho konkrétního zákazníka.
- využívá pouze jeden, jen je třeba zvolit který

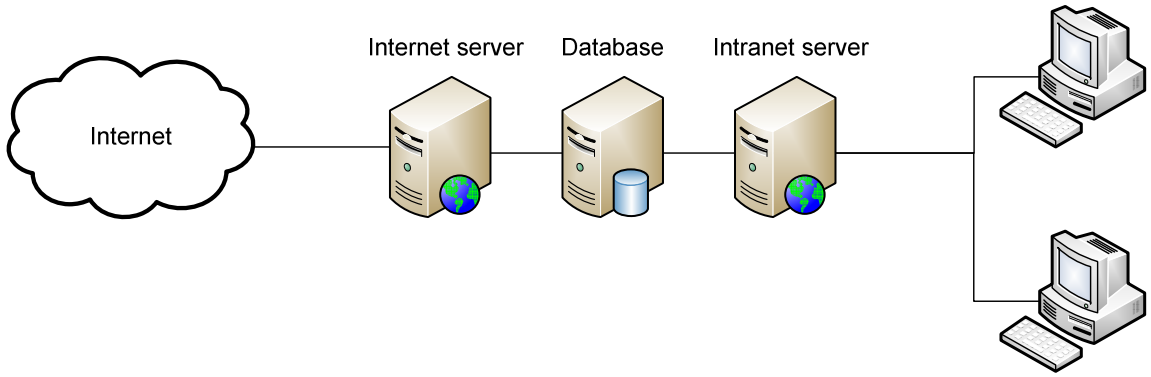
  S tímto problémem se potýkají velké projekty, kde se plánuje optimalizace aplikace pro potřeby databázového serveru.

Řešení prvního problému je relativně snadné. Nabízí se zde dvě hlavní možnosti. Použít univerzální napojení typu ODBC / JDBC / OleDB, nebo rozdělit aplikaci do modulů. Tyto moduly potom budou využívat funkce určené přímo pro komunikaci s danou databází.

Druhý problém je ovšem na řešení náročnější. Zde závisí na mnoha faktorech, mezi které patří:

- velikost databáze
- očekávané množství transakcí (vytížení databáze)
- znalosti firemních programátorů
- cena
- prostředí, v kterých bude aplikace vyvíjena a provozována
- firemní politika
- požadavky na funkce databázového systému (Reporty, Business Intelligence,…)

Ve výsledku se poté podílí na vybírání vhodné databázové platformy analytici, programátoři i manažeři.



Obr. 1: Možné využití databázového serveru jako úložiště dat ve firmě. Je využíván jak intranetovou aplikací (např. systém pro evidenci zboží, skladový systém, atd.), tak je skrz webový server vystaven internetu (např. elektronický obchod)

2.1 Problémy univerzálních připojení

SQL Server, Oracle i MySQL poskytují při užití specializovaných funkcí vcelku vysoký komfort pro vývojáře. Tyto funkce jsou již optimalizované, zpracování výsledků a předávání parametrů je snadnější. Při užití ODBC/OleDB dojde k určité degradaci spojení. V závislosti na programovacím jazyku aplikace je tato degradace rozdílná. Téměř vždy však znamená ztrátu výkonu a funkcionality. Při výrobě modulární aplikace zase je třeba vynaložit více úsilí na výrobu modulů.

2.1.1 Automatický unikátní identifikátor

Jako unikátní identifikátor v databázích lze užít jakýkoliv datový typ. V případě, že jej však chceme automaticky generovat na straně serveru, používá se prostý integer. Buď databáze umožňují atribut sloupce tzv. identity, při jehož vhodném nastavení, unikátní identifikátor není třeba složitě kalkulovat a je zapsán automaticky při operaci INSERT, nebo nabízí jiné řešení,jako např. Oracle, kde je v případě automatické tvorby identifikátoru třeba použít funkce sequenceru (objekt databáze, který podle stanovených pravidel počítá následující hodnotu. Nejčastěji řadu čísel od jedné nahoru). Problémy nastanou, pokud daná databáze podobnou vlastnost nemá (např. MySQL do verze 5.0). Poté je třeba identifikátor vypočítat ručně na straně klienta databáze (tedy aplikace, která databázi využívá, nikoliv uživatel té aplikace).

Kromě možnosti identity, některé servery nabízí další možnosti. Např. SQL Server zavádí datový typ Unique Identifier, díky němuž je možné dosáhnout unikátnosti ID v celé databázi, nikoliv jen v samotné tabulce (zde záleží na dalším nastavení).

Kvůli nejednoznačnosti normy ANSI/ISO SQL o unikátních identifikátorech, je tento atribut řešen různě pro různé databáze. Když je u konce vývoje aplikace generován skript pro tvorbu databáze, je nutné upravit jej pro každý daný databázový server.

2.1.2 Dialekty SQL

Při implementaci univerzálního napojení na databázi je třeba se vyhnout jakémukoliv „nadstandardnímu“ SQL skriptu. Při sestavování dotazů je lepší se vyvarovat některým komplexnějším typům JOINů, vnořeným SELECTům, atd. Určitou míru zátěže je tím pádem třeba přenést z databáze na úroveň aplikace.

3 Porovnání metod ukládání dat

3.1 XML

XML, nebo také Extensible Markup Language, vznikl v 90. letech za účelem sdílení rozdílně strukturovaných dat mezi informačními systémy. Vzniknul z formátu SGML. Dovoluje velmi volnou úpravu struktury při zachování přehlednosti a velmi snadné čitelnosti pro aplikaci. Neposkytuje sice jakékoliv databázové funkce, ovšem díky svým vlastnostem si našel tento formát své uplatnění.

3.1.1 Využití

Formát XML je používán v mnoha technologiích. Díky schopnosti zpracovat formátování Unicode se stal univerzálním prostředkem pro uložení dat v jakémkoliv světovém jazyce. Struktura tohoto souboru je tzv. „self documenting“, což znamená, že je obsažena přímo s daty a není třeba žádných dodatečných informací. Dokáže v sobě uchovat nejčastěji užívané struktury, jako jsou záznamy, seznamy nebo stromy. Díky nezávislosti na platformě a jasné, striktní specifikaci, si získal oblibu jako metoda pro sdílení dat. Právě kvůli striktní specifikaci je datový soubor XML rychle zpracovatelný a generovatelný.

Mezi jeho několika nevýhodami je však značná redundantnost, právě díky textové formě, která se nejvíce projevuje při ukládání různých seznamů. Jelikož jde o jediný datový soubor, je zde vyloučena možnost víceuživatelského přístupu jiná, než pouze pro čtení. To musí být implementováno na vyšší, než datové úrovni.

XML je vhodným prostředkem při komunikaci mezi rozdílnými informačními systémy. V praxi je XML používán převážně tam, kde není třeba jeho obsah často modifikovat (kromě případů kompletního přegenerování souboru). Vzniklo několik nástrojů (XPath, XQuery), díky nimž lze s XML snadno pracovat a dotazovat se na jeho obsah obdobně jako u SQL jazyků.

Často XML působí jako prostředník mezi aplikací a databázovým systémem. Většina moderních databází umožňuje export dat přímo do XML. Ty je poté možné pomocí jazyka XSLT zobrazit jako webové stránky s vhodným formátováním a uživatelskou interakcí. Často je používán jako sklad dokumentů v informačních systémech, kde je třeba zachovat nezávislost a národní abecedu.

XML je také používán jako konfigurační soubor (například Web.config v ASP.NET). Díky své struktuře je konfigurace přehlednější. Při jeho použití se ušetří za prostředky, které by bylo třeba vynaložit na definici vlastního formátu.

```
<system.codedom>
 <compilers>
  <compiler language="c#;cs;csharp" extension=".cs"
            type="Microsoft.CSharp.CSharpCodeProvider,System,
            Version=2.0.0.0, Culture=neutral,
            PublicKeyToken=b77a5c561934e089" warningLevel="4"
  >
   <providerOption name="CompilerVersion" value="v3.5"/>
   <providerOption name="WarnAsError" value="false"/>
  </compiler>
  <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
            type="Microsoft.VisualBasic.VBCodeProvider, System,
            Version=2.0.0.0, Culture=neutral,
            PublicKeyToken=b77a5c561934e089" warningLevel="4"
  >
    <providerOption name="CompilerVersion" value="v3.5"/>
    <providerOption name="OptionInfer" value="true"/>
    <providerOption name="WarnAsError" value="false"/>
  </compiler>
 </compilers>
</system.codedom>
```

Obr. 2: Použití XML architektury jako konfigurace webové aplikace. Sekce nastavení překladače.

Jako jediná z posuzovaných metod, XML formát (nepočítaje nadstavby v podobě XQuery,…) nenabízí databázové funkce, jelikož se jedná o jediný soubor, nikoliv aplikaci. V praxi to znamená, že XML soubor neumožňuje žádná uživatelská oprávnění, kromě práv k celému souboru garantovaných operačním systémem. XML není schopen jakékoliv interakce sám o sobě v podobě triggerů, třídění, indexace, atd.

Existuje několik prostředků, jak XML data zobrazovat:

- XSL-FO

  Dovoluje reformátovat XML dokument do jiné výsledné formy. Například PostScript, nebo AdobePDF.

- XSLT

  XSLT procesor je schopen využít XSLT dokument ke konverzi dat z XML. Nejčastěji se používá k zobrazení XML dokumentu jako webové stránky.

- XQuery

  Je to jazyk, který slouží k dotazování se na obsah XML, jeho konstrukci a transformace.

- XPath

  Je využíváno třemi předchozími metodami jako prostředek pro získávání dat z XML.

3.1.1.1 AJAX

AJAX (Asynchronní Javascript a XML) je skupina technik na tvoření webových aplikací za účelem vyšší interaktivity. Toho dosahuje tím, že data se serverem jsou vyměňována na pozadí, takže není třeba načítat znovu celé stránky pokaždé, když je třeba získat odpověď serveru. Cílem je navýšení interakce, rychlosti, funkcionality.

3.2 Úvod k porovnání databázových serverů

Mezi tvůrci databázových aplikací existují skupiny programátorů a každá z těchto skupin upřednostňuje používání vybraných serverů a ostatním možnostem se pokud možno vyhýbají.

Snažit se zde porovnat jednotlivé servery po stránce výkonnosti, nebo možností T-SQL dialektu proti PL-SQL či jiným nářečím není na místě. Ve výsledku umožňují téměř totožné funkce i výkon ve vhodně konfigurovaném prostředí.

Jelikož je cílem porovnat ukládání dat právě na webu, je v každé části speciální kapitola pro .NET i PHP integraci daných databází (vzhledem k zastaralosti ASP, nebude tato technologie zmiňována).

3.2.1 Vysvětlení některých vlastností databázových serverů a souvisejících technologií

uložené procedury

Dovolují část SQL skriptu uložit v databázi a později zavolat.

uživatelské funkce

Funkce, kterou lze volat jako součást SQL skriptu. Uživatel si je může vytvořit.

triggery

Definují akce, které se provedou při událostech v databázi jako je vkládání, mazání, úpravy záznamů, atd.

index

Struktura umožňující rychlejší vyhledávání a řazení tabulek.

data-mining

Proces procházení dat a vybírání relevantních informací podle definovaných pravidel.

reporty

Formátované výstupy z databáze do html, pdf, excelu atd..

.NET Framework

Soubor knihoven od firmy Microsoft.

ASP.NET, PHP

Technologie zpracování serverových skriptů na webu .

ASPX

Soubor obsahující serverové skripty pro Internet Information Services webový server.

3.3 Microsoft SQL Server

Microsoft SQL server je často označován jako „nedospělá“ databáze v porovnání s Oracle. Avšak i když na trhu je ani ne polovinu času, který má Oracle, v poslední verzi (SQL Server 2005 – *Yukon*) dosáhnul porovnatelných kvalit. Letos očekávaná verze (SQL Server 2008 – *Katmai*) přináší opět mnoho zlepšení služeb a nárůst výkonu. Už ve standardní distribuci poskytuje širokou škálu nástrojů pro vývojáře, administrátory, ale i uživatele, které jsou ve zvyku Microsoftu velmi snadno ovladatelné a přehledné.

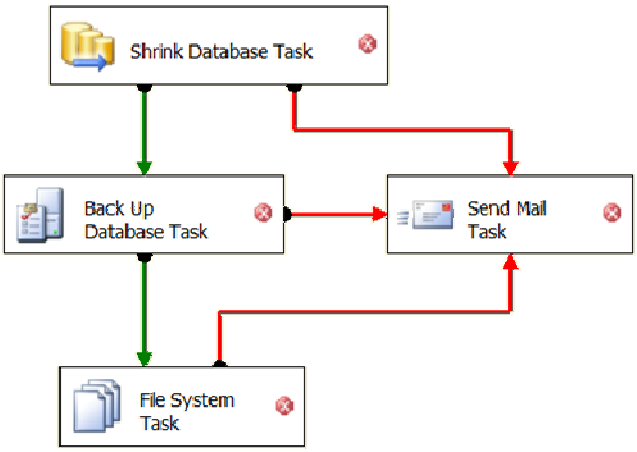
Právě těmito nástroji se SQL Server podstatně odlišuje od ostatních databázových řešení MySQL, nebo PostgrSQL. Díky silné integraci klienta do .NET Frameworku je nejlepší volbou právě pro vývojáře v .NET prostředí.

3.3.1 Vlastnosti z pohledu vývojáře

3.3.1.1 SQL

SQL Server 2005 přišel proti předešlé verzi s novinkou, BIDS (SQL Server Business Intelligence Developement Studio). Díky tomuto nástroji, který je plně integrován do Visual Studia 2005 (pro verzi *Katmai* se jedná o VS 2008) je možné plánovat a programovat akce nad daty prostým kliknutím myši. Business Intelligence obsahuje (viz [3]):

- SQL Server Analysis Service
  Na analýzu dat a data-mining.
- SQL Server Integration Services
  Umožňuje mnoho funkcí SQL Serveru od transformace dat po import/export, zálohování a další akce pro správu serveru.
- SQL Server Reporting Services
  Návrh a generování reportů do XML, HTML, Excelu nebo PDF.



Obr. 3: Nastavení zálohování databáze pomocí Integration Services. V případě selhání, zašle email.

Další vlastností SQL Serveru 2005 je tzv. CLR 2.0 (Common Language Runtime). SQL Server obsahuje přímo ve své architektuře značnou součást .NET Frameworku (alokace paměti, „threading“, správa zdrojů, ...). Díky tomu se nemusí spoléhat na operační systém při jejich využívání a umožňuje tvorbu vlastních uživatelských funkcí (User Defined Functions), uložených procedur (Stored Procedures) a triggerů s využitím libovolného jazyku Frameworku. Všechny zmíněné objekty jsou na rozdíl od Oracle uloženy přímo v databázi, díky čemuž je možno je spravovat stejně, jako každý jiný databázový objekt od přiřazení oprávnění po začlenění do zálohování nebo replikace. Díky obsažení Frameworku přímo v SQL Serveru jsou také rychlejší, než jejich klon od Oracle.

```
public class StoredProcedures
{
  [Microsoft.SqlServer.Server.SqlProcedure]
  public static void SalarySUM(out SqlMoney value)
  {
    using (SqlConnection connection = new
                  SqlConnection("context connection=true"))
    {
       value = 0;
       connection.Open();
       SqlCommand command = new SqlCommand("SELECT Salary FROM Employees",
                            connection);
       SqlDataReader reader = command.ExecuteReader();
       using (reader)
       {
         while (reader.Read())
         {  value += reader.GetSqlInt32(0); }
       }
    }
  }
}
```

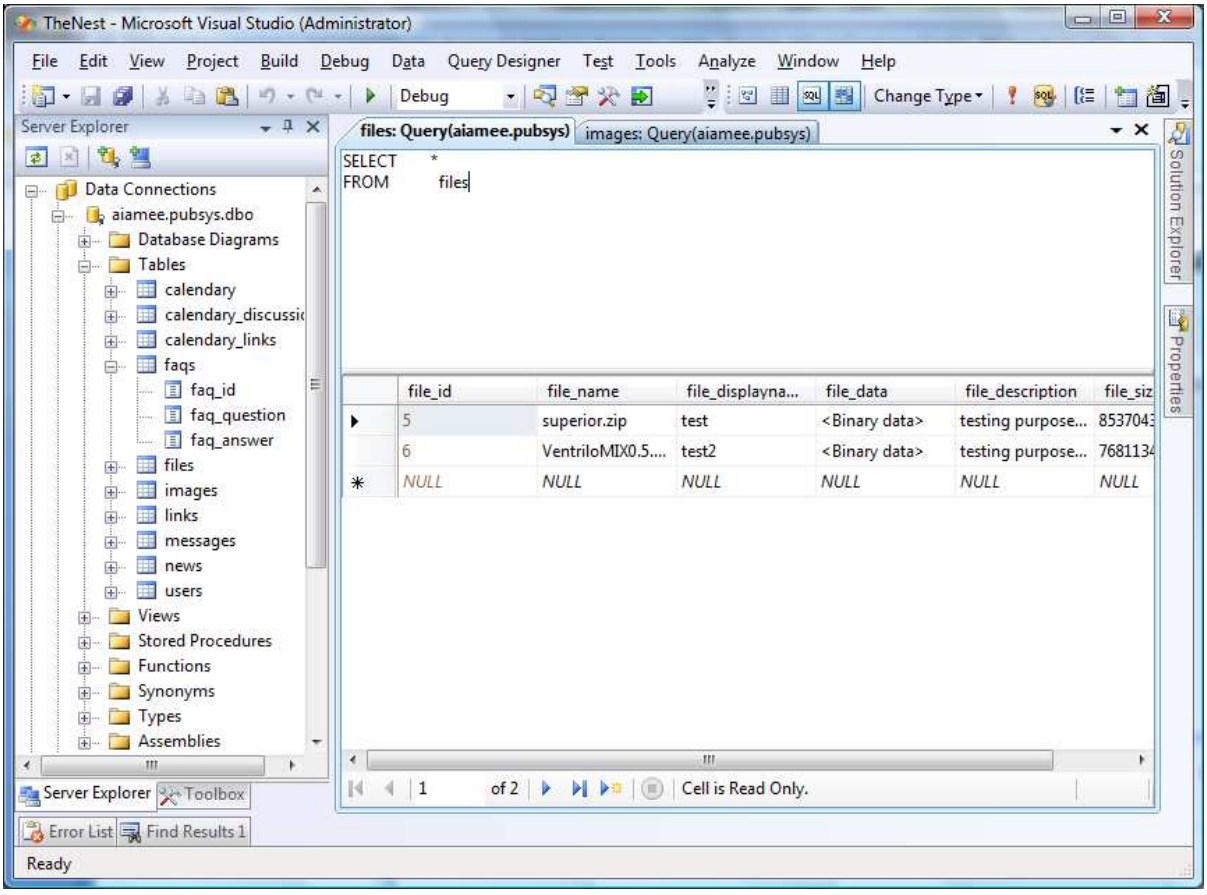
Obr. 4: Příklad C# uložené procedury vytvořené pomocí Visual Studia. Provede součet platů zaměstnanců a výsledek uloží do výstupního parametru value.

3.3.1.2 .NET

Jelikož nejpoužívanějším rozhraním pro vývoj .NET aplikací je Vistual Studio, integrace SQL Serveru do tohoto produktu je značná. Umožňuje tak rychlejší vývoj aplikací, než při kombinaci Visual Studia s např. MySQL. Díky této integraci je možno přímo z Visual Studia provádět akce jako:

- vytváření diagramů databáze
- návrh tabulek
- zobrazování dat
- tvorba a volání pohledů, uložených procedur a funkcí

Dále jde nastavit indexy a další vlastnosti, v případě, že k tomu má připojený uživatel oprávnění.



Obr. 5: Server Explorer integrovaný ve Visual Studiu se zobrazením tabulky „files“ s možností editace.

Při programování aplikace lze vytvořit objekt pro připojení k databázi a zavolání SQL dotazu snadným průvodcem během několika málo vteřin, což ušetří značné množství času. Další velkou výhodou je možnost ladění uložených procedur i funkcí.

Framework obsahuje několik metod, jak k SQL Serveru přistupovat:

- ODBC a OleDB.NET

  Jedná se o univerzální rozhraní, díky kterému je možno připojit se na množství jiných datových skladů. Dodáváno jako součást MDAC (Microsoft Data Access Components). Umožňuje podobné funkce jako native klient, ovšem má svá omezení (např. neschopnost předávat parametry jménem při vytváření SQLQuery viz Obr. 6).
- native klient - SQLConnection

  Nejlepší varianta, pokud neuvažujeme o univerzálnosti aplikace. Poskytuje optimalizovanou funkcionalitu pro SQL Server.

Při napojení ASP.NET na SQL Server pomocí OleDB providera, je problematické udržet názvy parametrů dotazů. Následující kód je ukázkou, jak může vypadat připojení přes OleDB:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:defaultConnector %>"
    ProviderName="<%$ ConnectionStrings:defaultConnector.ProviderName %>"
    SelectCommand="SELECT [nws_id], [nws_title], [nws_date], [nws_text],
                   [nws_author], [nws_hasimage] FROM [news]
                   WHERE
                   [nws_system]=? AND [nws_date]=?
                   ORDER BY [nws_date] DESC"
    >
          <SelectParameters>
            <asp:Parameter Name="sys_id" Type="Int16" DefaultValue="0" />
            <asp:Parameter Name="date_now" Type="DateTime"/>
          </SelectParameters>
</asp:SqlDataSource>
```

Obr. 6: Definování hodnot parametrů

Za zmínku stojí část **`[nws_system]=? AND [nws_date]=?`** . Zde ? značí místo vložení parametru. Bohužel, parametry musí být uváděny v totožném pořadí i níže, v sekci <SelectParameters>. Část Name potom u jednotlivých parametrů není třeba mít zachovanou, i když vhodně zvolené jméno vždy usnadňuje orientaci v kódu.

Při použití native SQL klienta, lze část `[nws_system]=? AND [nws_date]=?` nahradit za `[nws_system]=@sys_id AND [nws_date]=@date_now`. Tím vznikne možnost upravit si později pořadí parametrů, podle potřeb programátora. V případě komplikovanějších dotazů je přínos zřejmý.

3.3.1.3 PHP

Pro vývoj v PHP vydal Microsoft speciální balíček funkcí, které rozšiřují možnosti PHP aplikací o nativní připojení k MS SQL Serveru. Je však určen především pro PHP běžící na Windows. Lze však použít i univerzální metodu pomocí ODBC (nebo *iODBC* – open source projekt, který nabízí ODBC funkcionalitu do systémů jako je Linux, FreeBSD, Solaris, případne UnixODBC).

3.3.1.4 XML

MS SQL Server od verze 2005 podporuje datový typ XML (viz [3]). Díky tomu je možno XML soubor uložit přímo v tabulce. Samozřejmostí je automatická validace podle XML standardu a možnost dotazovat se na obsah pomocí XQuery případně XPath. Dále umožňuje podle obsahu XML souboru i indexovat (nikoliv na základě jeho textové interpretace jako Oracle, ale na základě dat v něm obsažených).

Další zásadní rozdíl od Oracle dosahuje SQL Server ve schopnosti uložit více XML dokumentů do jednoho sloupce záznamu. To je dosaženo vytvořením XML Schema Collection, která se skládá z několika XSD schémat.

SQL Server dovoluje definování constrains i pro XML soubor (viz Obr. 7). Zatímco první z INSERTů se provede, následující dva selžou.

```
CREATE TABLE T (
    Col1 int primary key,
    Col2 xml check(dbo.my_udf(Col2)>10))
GO
INSERT INTO T values(1,'<ProductDescription Prize="15" />')
INSERT INTO T values(1,'<ProductDescription Prize="9" />')
INSERT INTO T values(1,'<Product />')
```

Obr. 7: Příklad constrains na sloupec s datovým typem XML.

3.3.2 Bezpečnost

Zabezpečení SQL Serveru i Oracle je velmi podobné, co se nabízených funkcí týče. Rozdílem je, že SQL Server je dodáván se stejným zabezpečením ve všech verzích, zatímco u Oracle je třeba si tyto komponenty dokoupit (viz [22]).

SQL Server obsahuje následující zabezpečení:

- Network Packet Encryption

  SQL Server dovoluje kódovat přenášená data pomocí IP Sec případně SSL certifikátem. SSL je podporováno buď integrací s Internet Information Service (IIS) nebo za podpory certifikačního serveru, který je součástí SQL Serveru.
- Data Encryption

  SQL Server obsahuje několik funkcí, které dovoluji kódování a dekódování dat na základě certifikátu, hesla, nebo asymetrického klíče.
- Public Key Infrastructure
- Kerberos Support

  SQL Server dovoluje autentizovat uživatele proti doméně, nebo systému, na kterém instance běží.
- Schemas, Database and Server Roles

  Databázové objekty jsou přiděleny do tzv. schémat. Je možno je chápat jako uživatelské skupiny, které obsahují namísto uživatelů tabulky, pohledy, procedury a jiné objekty databáze. Uživatelé jsou zase přiděleni do rolí, což jsou ve své podstatě uživatelské skupiny. Práva jsou potom definována pro jednotlivé role na schéma. Dále jsou na serveru

definovány tzv. Server roles. Uživatelé zařazeni do nich získávají práva na provádění záloh, tvorbu databází a jiné úkony, které se týkají celého serveru a ne jednotlivých databází.

- Auditing
- Profiles/Policies
- Certificate Services
- Execution with Least Privileges

  SQL Server umožňuje spouštět skripty jako jiný uživatel.

Zabezpečení proti zneužití dat není jediným typem zabezpečení, který je třeba zhodnotit. Schopnost zabezpečit data proti zničení je podstatnou částí databáze. SQL Server nabízí možnosti automatického zálohování, merge replication a log shipping (viz [22]).

- Log shipping

  Jednou z možností replikace dat na MS SQL Serveru je tzv. log shipping. Je zhotovena replika databáze obnovením zálohy na „příjemci“ a „vysílač“ odesílá informace o změnách, které jsou poté aplikovány i na příjemci. V případě pádu vysílače, převezme jeho funkci přijímač a spustí se event (možno nastavit odeslání varování, nebo jakýkoliv jiný naplánovaný job).
- Merge replication

  Databáze je přenesena na druhou instanci SQL Serveru. Přístup uživatelů je poté dělen mezi tyto dva servery a v pravidelných intervalech jsou data sjednocena. Pokud dojde ke konfliktu (byl změněn stejný záznam na obou serverech), je třeba jej řešit. A to buď manuálně, nebo pomocí předdefinovaných pravidel.

3.4 Oracle

Oracle, který je na trhu již téměř třicet let, má pověst gigantu s vysokými nároky na hardware ale také špičkovým výkonem na poli databázových serverů. Právě díky své robustnosti (a z ní vyplývající ceně), nebývá nasazován jako řešení pro malé a středně velké databáze.

Podobně jako u SQL Serveru, jsou k dispozici přehledně vypracované nástroje na administraci serveru a vývoj aplikací.

3.4.1 Vlastnosti z pohledu vývojáře

Pro vývoj aplikací vydala firma Oracle nástroj srovnatelný s možnostmi Visual Studia firmy Microsoft pod jménem Developer Suite (viz [14]). Produkt je určen převážně pro vývoj Java aplikací. Oracle dále dodává ve své nabídce několik produktů, které umožňují relativně snadnou práci s databází a tzv. Business Intelligence . Díky své nezávislosti na platformě je nejpoužívanější databází na většině velkých projektů, kde z jakéhokoliv důvodu nechtějí použít Windows.

3.4.1.1 SQL

Existuje řada nástrojů, které podporují snadnou práci s Business Intelligence (viz [12]):

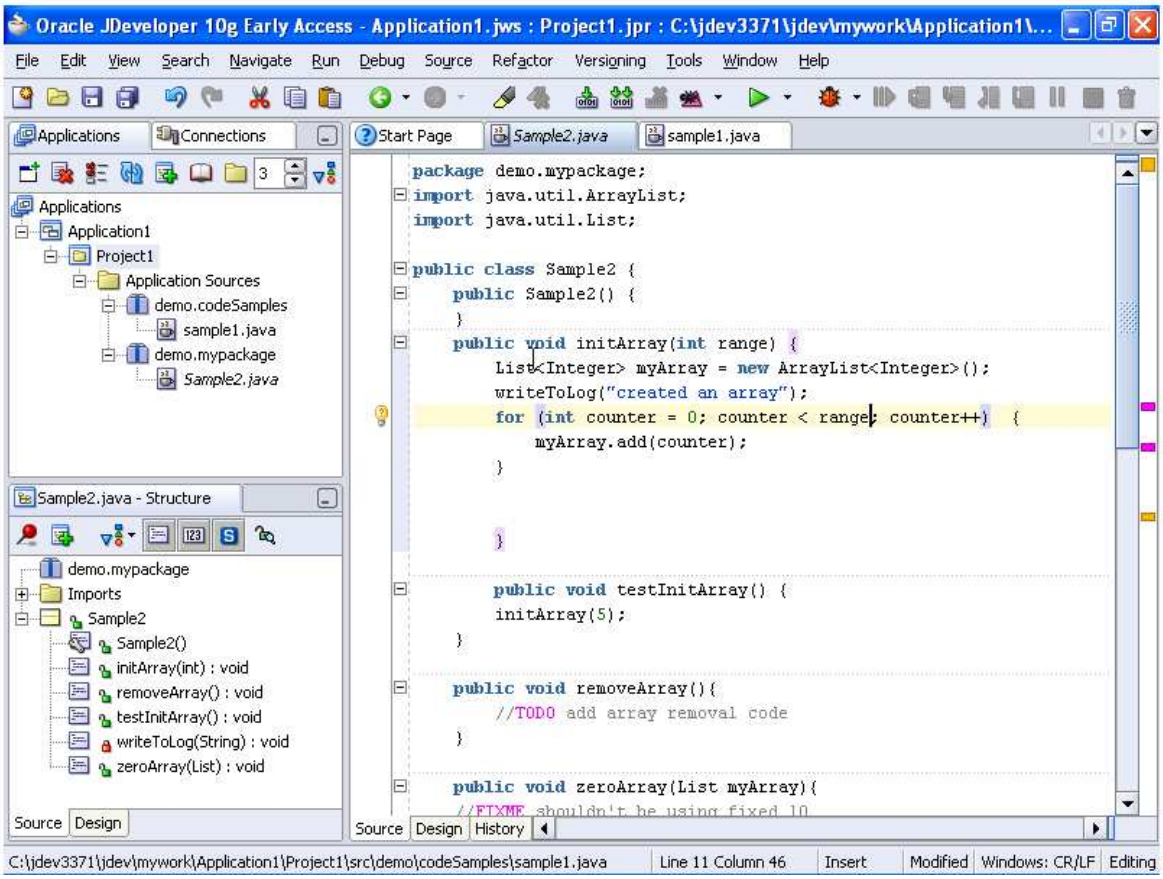
- Oracle OLAP Option na analýzu dat.
- Oracle Data Mining Option na data-mining.
- Oracle Reports návrh a generování reportů.
- Oracle Warehouse Builder na ETL (Extract, Transform, Load).
- Oracle BI Beans na vývoj OLAP aplikací.

Většina těchto nástrojů je komponentou Application Serveru.

Oracle, podobně jako MS SQL Server, umožňuje tvořit uložené procedury v jazyce jiném, než PL/SQL (T-SQL v případě SQL Serveru). Oracle užívá jazyk Java. Díky tomu je zajištěna určitá míra nezávislosti na platformě. Další zajímavou vlastností je schopnost Oracle překládat PL/SQL kód do tzv. native-code. To urychluje podstatně jeho provádění (řádově v desítkách procent), podle složitosti zdrojového skriptu.

3.4.1.2 Java

Stejně jako je SQL Server založen na integraci s .NET Frameworkem, Oracle je schopen spolupracovat s Javou. Právě díky Developer Suite je k dispozici intuitivní grafické rozhraní ne odlišné od varianty Microsoftu. Nabízí standardní možností editace kódu, modifikace databáze, atd. až pokročilé funkce UML, vývoje Web Services, či dokonce možnosti týmového projektu.



Obr. 8: Prostředí Oracle JDeveloper.

3.4.1.3 .NET

Oracle povoluje využití .NET Frameworku (CLR 1.1) místo Javy pro implementaci uložených procedur, avšak nikoliv v takové kvalitě, jako SQL Server. Kód procedur není uložen v databázi, ale v dodatečných souborech, které je třeba v případě RAC distribuovat společně s databází.

Integrace do Visual Studia je podobná jako u SQL Serveru. Server Explorer je nahrazen Oracle Explorerem, který poskytuje stejné schopnosti pro úpravu databáze. Jediným nedostatkem je neschopnost ladit uložené procedury. K tomu je třeba JDeveloper (součást Developer Studia).

Oracle vyvinuli doplněk .NET Frameworku, který umožňuje přímé napojení na databázi bez nutnosti používat obecné ODBC / OleDB spojení (to je samozřejmě možné použít, pokud aplikace má být univerzální).

3.4.1.4 PHP

Podobně jako u SQL Serveru, PHP nemá native podporu Oracle. Je třeba si obstarat Oracle Instant Client pro daný operační systém, který možnosti PHP doplní o schopnost využít Oracle API. A stejně tak Oracle nenabízí žádné nátroje pro vývoj PHP aplikací (viz [13]).

3.4.1.5 XML

Podpora XML v Oracle o málo zaostává proti SQL Serveru. Oracle umožňuje ukládat data v podobě XML do záznamů databáze, avšak jejich zpracování je horší. Indexovat podle nich zvládá pouze podle textové formy a v nutnosti jejich extrakce je třeba využít Javu.

3.4.2 Bezpečnost

Oracle poskytuje téměř totožné funkce pro zabezpečení, jako SQL Server, co se ochrany soukromí týká. Rozdílem je však to, že tyto vlastnosti jsou k dipozici až u verze Enterprise s dokoupeným balíčkem Advanced Security Option. Kódování dat a přenosu, podpora autentizace pomocí Kerberos, nebo certifikační služba jsou komponenty, které je třeba dokoupit, a ve standardní verzi není možno je provozovat.

Na rozdíl od SQL Serveru, Oracle implementuje některé služby jinak (viz [22]):

- Network Encryption

  Pro kódování přenášených dat je možno použít RSA nebo DES algoritmy, podobné těm, které užívá SQL Server pro šifrování dat uložených. Dále je však možnost přenášet kromě dat i MD5 kontrolní součet, díky němuž je možno zjistit, zda nedošlo k modifikaci v průběhu přenosu. Další zajímavostí je, že v Oracle je kódování standartně vypnuto, zatímco v SQL Serveru je po instalaci zapnuto.

Jak bylo zmíněno, ochrana dat proti zneužití není jedinou potřebnou vlastností databázového stroje. Oracle poskytuje kromě standardních funkcí zálohování dat také možnost clusteringu pomocí technologie RAC (Oracle Real Application Clusters). Díky tomu je umožněno dvěma a více databázovým serverům běžícím na různých fyzických počítačích přistupovat ke stejným datům stejné databáze (viz [15]).

3.5 MS SQL Server vs. Oracle

Neexistuje jednoduchá metoda, jak tyto dva servery porovnat. Možnosti obou produktů jsou velmi bohaté. Právě díky jejich rozsáhlosti není příliš snadné vybrat vhodné parametry pro porovnání. Oba produkty jsou určeny pro trochu odlišnou skupinu aplikací a oba nabízí něco, co druhá strana neumí. V určitých místech se střetávají a tam nastávají neshody, co je výhodnější. Na to neexistuje obecná odpověď a výsledek je vždy závislý na mnoha faktorech.

Z hlediska nástrojů pro vývoj aplikací jsou si velmi blízké.

- nástroje pro vývoj a správu

  SQL Server i Oracle poskytují téměř shodnou sadu nástrojů pro vývoj databází a aplikací včetně nástrojů pro Business Intelligence a správu databáze.

  Zatímco Oracle upřednostňuje správu fyzického úložiště databáze administrátorem a poskytuje k tomu široké možnosti, SQL Server se o data stará sám, bez možnosti zásahu administrátorem. Jedinou možností je definování tzv. cluster indexu, podle kterého jsou potom data ukládána na disku za sebou.
- bezpečnost

  Obě databáze si z pohledu zabezpečení vedou dobře, ovšem nevýhodou je nedostupnost zabezpečení pro standardní verzi Oracle. Je třeba vlastnit licenci na Enterprise a dokoupit bezpečnostní balíčky. Zde je Microsoft napřed a poskytuje stejné zabezpečení pro Enterprise, Developer i Free verzi SQL Serveru.
- platformová nezávislost

  Nezávislost na platformě pro SQL Server neexistuje. Funguje pouze na Windows (32bit / 64bit). Za zmínku stojí, že právě na Windows si Oracle vede výkonově nejhůř.
- další rozdíly

  Je potřeba si dát pozor na rozdíly v PL/SQL a Transact-SQL. Nedostupnost BEFORE triggeru v SQL Serveru, stejně jako neznalost automatické inkrementace pro Oracle jsou jen jedním z mála rozdílů. Dalším nedostatkem Oracle je nízká úroveň zpracování XML souboru.

3.5.1 Která z databází je bezpečnější

Bezpečnost je něco, co v posledních letech Oracle částečně zanedbává. V posledních letech bylo zaznamenáno více průniků do serverů Oracle. Oracle taktéž reaguje pomaleji na bezpečnostní rizika a pomaleji opravuje mezery v zabezpečení (viz. [22] SQL 2005 and Oracle 10g Security).

3.6 MySQL a PostgreSQL

MySQL a stejně tak PostgreSQL jsou dvojicí volně dostupných databázových řešení. Zatímco PostgreSQL je čistě Open Source, MySQL je vlastnictvím Sun Microsystems a většina zdrojových kódů podléhá autorským právům (je však k dispozici pod podmínkami GNU GPL a několika dodatky). Hlavně ve světě Linuxu jsou obě databáze velmi oblíbeny a to především díky své dostupnosti a snad i proto, že jedinou alternativou na této platformě je Oracle.

3.6.1 MySQL

MySQL přináší několik zajímavých vlastností, které u jiných databázových serverů nejsou běžné. Například možnost zvolit si engine (rozhraní, které se stará o rozhraní mezi fyzickými daty a databázovým serverem) pro ukládání dat podle potřeb aplikace. Engine je třeba zvolit podle potřeb aplikací, jelikož ne všechny umožňují stejnou funkcionalitu.

	MyISAM	**InnoDB**	**MEMORY**	**NDB**
Multi-statement transactions, ROLLBACK	-	X	-	X
Foreign key constraints	-	X	-	-
Uzamykání záznamů	tabulka	záznam	Tabulka	záznam
BTREE index	X	X	-	X
FULLTEXT index	X	-	-	-
HASH vyhledávání	-	X	X	X
Jiné „in-memory tree-based“ indexy	-	-	4.1.0	-
GIS, RTREE indexy	4.1.0	-	-	-
Unicode	4.1.0	4.1.2	-	-
Merge (union views)	X	-	-	-
Komprese dat (pouze pro čtení)	X	-	-	-
Relativní využití disku	Nízké	Vysoké	-	Nízké
Relativní využití paměti	Nízké	Vysoké	Nízké	Vysoké

Obr. 9: Porovnání rozdílů mezi způsoby ukládání dat v MySQL

3.6.1.1 .NET

Přístup k MySQL z prostředí .NET je možný několika způsoby. Buď univerzální ODBC / OleDB, nebo využití MySQLDirect (produkt firmy CoreLab). Jakákoliv jiná integrace do Frameworku ani Visual Studia v současné době neexistuje.

3.6.1.2 PHP

PHP je ideální variantou front-end rozhraní, pokud jde o přístup k MySQL z internetu. V PHP je zahrnuto značné množství funkcí pro práci s touto databází. Linuxová komunita navíc poskytuje

spoustu různých doplňků, které je možno použít pro správu databáze po internetu (např. phpMyAdmin).

3.6.1.3 XML

Podpora XML v MySQL neexistuje. Pokud je třeba uschovat XML soubor v databázi, musí se použít buď textové, nebo binární pole. Naštěstí existuje řada doplňků k existujícím programovacím a skriptovacím jazykům, které umožňují výsledek SQL dotazu uložit jako XML, případně užít XML jako vstup(viz [21]).

Pro porovnání PHP kód s manuální generací XML:

```
use strict;
use DBI;

my $dbh = DBI->connect ("DBI:mysql:test","testuser", "testpass",
                        { RaiseError => 1, PrintError => 0});
my $sth = $dbh->prepare ("SELECT name, category FROM animal");
$sth->execute ();
print "<?xml version=\"1.0\"?>\n";
print "<dataset>\n";
while (my ($name, $category) = $sth->fetchrow_array ())
{
    print " <row>\n";
    print "  <name>$name</name>\n";
    print "  <category>$category</category>\n";
    print " </row>\n";
}
$dbh->disconnect ();
print "</dataset>\n";
```

Obr. 10: manuální generování XML souboru.

A s automatickou generací XML souboru:

```
use strict;
use DBI;
use XML::Generator::DBI;
use XML::Handler::YAWriter;

my $dbh = DBI->connect ("DBI:mysql:test","testuser", "testpass",
                        { RaiseError => 1, PrintError => 0});
my $out = XML::Handler::YAWriter->new (AsFile => "-");
my $gen = XML::Generator::DBI->new (
                                Handler => $out,
                                dbh => $dbh
                            );
$gen->execute ("SELECT name, category FROM animal");
$dbh->disconnect ();
```

Obr. 11: využití XML::Generator pro automatizaci tvorby XML.

Širší podpora XML je až ve verzi MySQL 5.1 a 6.0. Zde je již možno importovat i exportovat data mezi XML a databází přímo na úrovni MySQL, bez nutnosti využití jiného nástroje.

3.6.2 PostgreSQL

PostgreSQL se od ostatních databázových serverů mírně odlišuje. Jako jediný jej lze označit jako object-relational database management system (ORDBMS). Na rozdíl od klasických DBMS, ORD nabízí možnost definovat vlastní typy objektů. Nejčastěji datové typy a funkce, které se k nim vztahují. Zatímco klasická databáze se snaží o rychlou práci na limitovaném rozsahu datových typů a jejich efektivní uložení na disku (v paměti), ORD má většinou velmi komplexní data. PostgreSQL dovoluje definovat nejenom typy, ale i přetěžování operátorů, úpravy způsobů přetypování, atd.

Dále PostgreSQL definuje dědičnost. Díky tomu je možno nastavit vztah mezi tabulkami tak, že při jakékoliv změně vlastností rodiče, dojde ke stejné změně potomka (toto se samozřejmě nevztahuje na data v tabulkách uložená, nýbrž na vlastnosti celé tebulky). Např. při přidání sloupce do parent tabulky, dojde k jeho přidání i do dědice.

Jelikož PostgreSQL je opensource projekt, existuje ředa doplňků, které umožňují práci s geografickými informacemi, shortest-path-algoritmy, fulltext vyhledávání a další. Podobně jako Oracle, umožňuje PostgreSQL uložené procedury programovat v jazyku Java. Existuje řada dalších neoficiálních rozšíření, která podporují Perl, Python a další.

K PostgreSQL je několik možných administračních rozhraní:

- psql
  Textové rozhraní. Umožňuje volat přímo SQL, nebo velké množství meta-příkazů.
- pgAdmin
  Grafické rozhraní, podporující většinu používaných platforem.
- phpPgAdmin
  Klon phpMyAdminu na použití s PostgreSQL.

3.6.2.1 .NET a PHP

Podobně jako pro MySQL, společnost CoreLab vyvinula pro .NET Framework i rozhraní pro PostgreSQL (PostgreSQLDirect .NET). Stejně tak existuje řada doplňků k PHP na usnadnění připojení a práci s PostgreSQL.

3.6.2.2 XML

PostgreSQL poskytuje podobnou interakci s XML jako SQL Server. Vestavěný datový typ XML umožňuje uložit a validovat dokument XML, dotazování se na jeho vlastnosti a obsah, atd.

3.7 Oracle/SQLServer proti MySQL/PostgreSQL

Při porovnání komerční databáze (Oracle a SQL Server) s volnými (MySQL a PostgreSQL) je rozdíl podstatně zřejmější, než při porovnání Oracle a SQL Serveru. Je hlavně v kvalitě služeb a dostupnosti nástrojů. Zatímco komerční databáze nabízejí oficiální podporu a opravy v mezerách v zabezpečení po zakoupení produktu, MySQL jako jediná z volných databází za doplatek poskytuje podporu od firmy Sun Microsystems. Oprava a vylepšování bezpečnosti je pro MySQL relativně pomalé v porovnání s ostatními.

PostgreSQL nabízí množství rozšíření, které vyplývají z faktu, že je opensource projektem a spadá do rodiny ORDBMS.

Největší nevýhodou MySQL i PostgreSQL je absence vývojových prostředí, která by bylo možno použít pro vývoj aplikací. Navíc neexistují volně šiřitelné nadstavby pro Framework, takže programátor .NET si je buď musí zakoupit, nebo obejít pomocí OleDB / ODBC a tím ztrácí efektivní spolupráci databáze – aplikace. Z tohoto pohledu nejsou MySQL ani PostgreSQL pro komerční databáze konkurencí. Na poli výkonu je boj téměř srovnatelné do určité rozsáhlosti aplikací. Poté je třeba vynaložit další úsilí na optimalizaci nastavení volných databází, aby se udržely v kvalitě, kterou poskytuje Oracle nebo SQL Server.

Správa MySQL i PostgreSQL je prováděna pomocí rozhraní od třetí strany. Navíc chybí port webové administrace do ASPX, případně Javy, aby bylo možno ji zprovoznit na IIS bez nutnosti instalovat plug-in pro PHP.

3.8 Ostatní databáze

Za zmínku stojí i další databázové stroje. Jedním z nich je DB2 (odIBM), Informix (od roku 2001 IBM), Teradata (Teradata Corporation), FirebirdSQL (originálně od Inprise, nyní pod licencí IDPL), Sybase (od Sybase Inc.) a MaxDB (od SAP AG).

Tyto databáze nejsou rozšířeny tolik, jako hodnocené, avšak mají svůj podíl na trhu. Například Teradata je používáno jako řešení pro extrémně velké databáze u firem Wal-Mart, AT&T, Bank of America, Best Buy, Continental Airlines, Coca-cola, nebo FedEx.

4 Aplikace

4.1 Analýza problému

4.1.1 Zadání

Cílem je vytvoření internetové aplikace s propojením na databázi. Musí umožňovat zadávání a zobrazování novinek, odkazů a často kladených otázek, přidávání a stahování obrázků a souborů. Další součástí je kalendář, který umožňuje vypisování událostí s možností přihlášení se na ni (zde je třeba několik stavů: přihlášen, přijat, zamítnut, nemůže se zúčastnit). Ke každé události musí existovat diskuze, kde je možno zasílat komentáře. Uživatelé musí mít možnost se registrovat a přihlásit. Je třeba definovat minimálně 5 úrovní oprávnění.

Aplikace bude provozována na volných, nebo levných portálech. Přístup do databáze má jediný, předem určený uživatel. Typ a umístění databáze není zatím jisté (je třeba dodržet univerzálnost řešení).

4.1.2 Analýza požadavků

4.1.2.1 Funkcionální požadavky

- Vytváření a správa novinek, obrázků, souborů, odkazů, FAQ.
- Kalendář událostí:
  - možnost vypisovat a mazat události
  - přihlášení se k události, případně oznámení o neúčasti
  - moderování přihlášení (potvrzení přihlášky, zamítnutí)
- Registrace uživatele, přihlášení uživatele a nastavení oprávnění:
  - prohlížení webu
  - přihlášení k události
  - vypsání události
  - správa webu
  - správa uživatelů

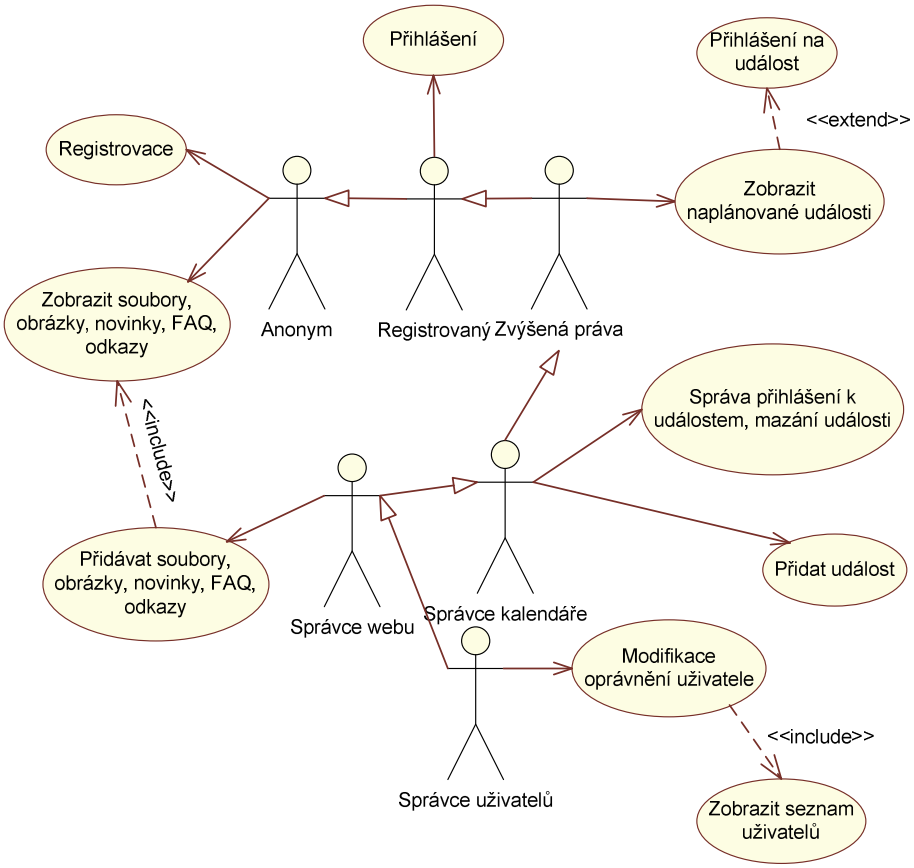
4.1.2.2 Požadavky na provoz a výsledný systém

Aplikace musí umožnit přístup více uživatelům najednou, neustálý provoz a dostupnost přes internet. Data musí výt persistentní.

4.1.2.3 Požadavky na rozhraní

Webové rozhraní s použitím HTML/XHTML a serverových skriptů. Kompatibilita s nejčastěji používanými prohlížeči (Internet Explorer, Mozilla, FireFox, Opera).

4.1.3 Use-case diagram



Obr. 12: Use-case diagram.

Existuje 6 úrovní oprávnění, kterých může uživatel dosáhnout:

- Anonym

  Má povoleno zobrazit vše až na kalendář a seznam uživatelů.
- Registrovaný

  Smí se přihlásit. Je přiřazeno automaticky při registraci
- Zvýšená práva

  Jedná se o uživatele s právem přihlásit se na události v kalendáři.
- Správce kalendáře

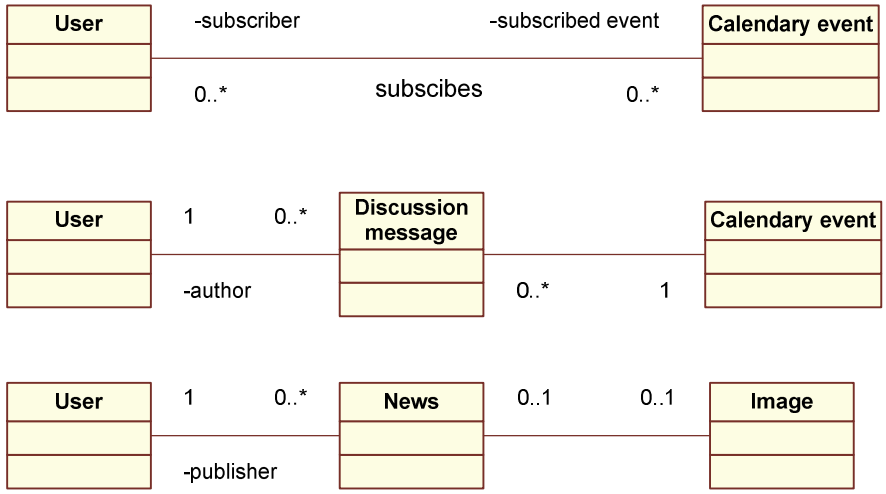
  Smí vypisovat události a potvrzovat / zamítat účast.
- Správce webu

  Má povoleno přidávat novinky, soubory, odkazy, obrázky a často kladené otázky.
- Správce uživatelů

  Smí modifikovat uživatelská oprávnění.

4.2 Návrh a implementace databáze

4.2.1 ERD



Obr. 13: Vazby mezi objekty

4.2.2 Implementace databáze

Při tvorbě byly k dispozici dvě možnosti.

- použít uložené procedury a triggery
- veškerou „inteligenci“ implementovat na straně aplikace

V případě užití triggerů a uložených procedur, by bylo nemožné aplikaci provozovat jinde, než na databázovém serveru (nebylo by možno databázi provozovat na např. MS Access) a pro různé servery by bylo třeba tvořit mírně rozdílná volání procedur. Při druhém způsobu zase není aplikace tolik efektivní. Po zvážení požadavků na univerzálnost je nakonec databáze použita jen jako datové úložiště.

Název tabulky	Stručný popis
Calendary	Vypsané události. U každé události je uvedeno datum a čas počátku a konce, popis a kdo ji vypsal.
Calendary_discussion	Diskuze k událostem. Obsahuje identifikátor události, autora zprávy a text.
Calendary_links	Zařazení uživatelů k událostem. Obsahuje stav zařazení (přijat, přihlášen, zamítnut, nemůže se účastnit), plánovaný počátek a konec (pro případ zpoždění, nebo předčasného odchodu), procentuální pravděpodobnost a komentář, identifikátor události a uživatele.
Faqs	Obsahuje často kladené otázky a odpovědi.
Files	Seznam souborů ke stažení. Obsahuje název souboru, zobrazované jméno, popis, binární verzi souboru a jeho velikost.
Images	Seznam obrázků. Osahuje název obrázku, zobrazované jméno, popis, binární verzi obrázku a náhledu, velikost obrázku a náhledu v bytech. Dále obsahuje identifikátor novinek
Links	Odkazy. Záznam se skládá z názvu, popisu a hypertextového odkazu.
News	Seznam novinek. Titulek, datum vypsání, text novinky, autor a příznak, zda novinka má přiřazený obrázek.
Users	Seznam uživatelů. Obsahuje přihlašovací jméno a heslo, zobrazované jméno a identifikátor skupiny oprávnění.

Obr. 14: Přehled tabulek v databázi

4.2.3 Primární a cizí klíče, auto-inkrementační hodnoty

Každá tabulka podle normy musí vlastnit právě jeden primární klíč. Nejčastěji unikátní identifikátor, který je automaticky počítán. Dále je možné definovat tzv. cizí klíče, které určují jakousi vazbu mezi tabulkami. Při datovém modelování je dobré cizí klíče kvůli zpřehlednění schémat uvádět, avšak při konečné tvorbě databáze mohou být v některých případech vynechány (například při potřebě odkazovat se do jedné z několika tabulek).

Při tvorbě primárních klíčů bylo třeba generovat tři různé skripty pro torbu databáze (jeden pro každou platformu). V případě MySQL a MSSQL jsou skripty téměř identické, až na zaměnění parametru IDENTITY za AUTO_INCREMENT. Systém Oracle však neumožňuje ve vlastnostech sloupce definovat takovouto hodnotu. Je třeba si pomoci triggerem, který je aktivován při vkládání nového záznamu. Zavolá tzv. sequencer a hodnotu ID nahradí následující hodnotou ze sequenceru. Nelze jednoznačně říci, který přístup je lepší, zda automatické generování, nebo „poloautomatické“ v Oracle. Zatímco trigger vyžaduje další SQL dotaz a tím zpomalení serveru, umožňuje definovat číselné řady pro ID stylem, který v ostatních databázích není příliš běžný. Díky němu lze mít

souvislou číselnou řadu pro různé skupiny tabulek (použijeme pro ně stejný sequencer) bez duplicity. v MySQL je podobná funkcionalita nedosažitelná. MS SQL Server nabízí alternativu, tou je použití UNIQUEIDENTIFIER, který je unikátní celosvětově. Jeho nevýhodou je velikost (16bajtů).

Porovnání tvorby unikátního identifikátoru pomocí auto-inkrementačního integeru:

SQL Server (Transact-SQL):

```
CREATE TABLE t_test(
     id INTEGER IDENTITY
     PRIMARY KEY,
     name VARCHAR(10)
 );
```

MySQL:

```
CREATE TABLE t_test(
     id INTEGER AUTO_INCREMENT
     PRIMARY KEY,
     name VARCHAR(10)
 );
```

Oracle (PL/SQL):

```
CREATE SEQUENCE sq;
CREATE TABLE t_test(
      id INTEGER PRIMARY KEY,
      name VARCHAR(10)
);
```

```
CREATE TRIGGER increment_trgr
BEFORE INSERT IN t_test
 FOR EACH ROW
 BEGIN
 SELECT sq.nextval
      INTO :new.id
 FROM dual;
 END
;
```

Obr. 15: Tvorby automatického identifikátoru v Oracle, MySQL a SQL Serveru

4.3 Implementace aplikace

4.3.1 Klientská strana aplikace

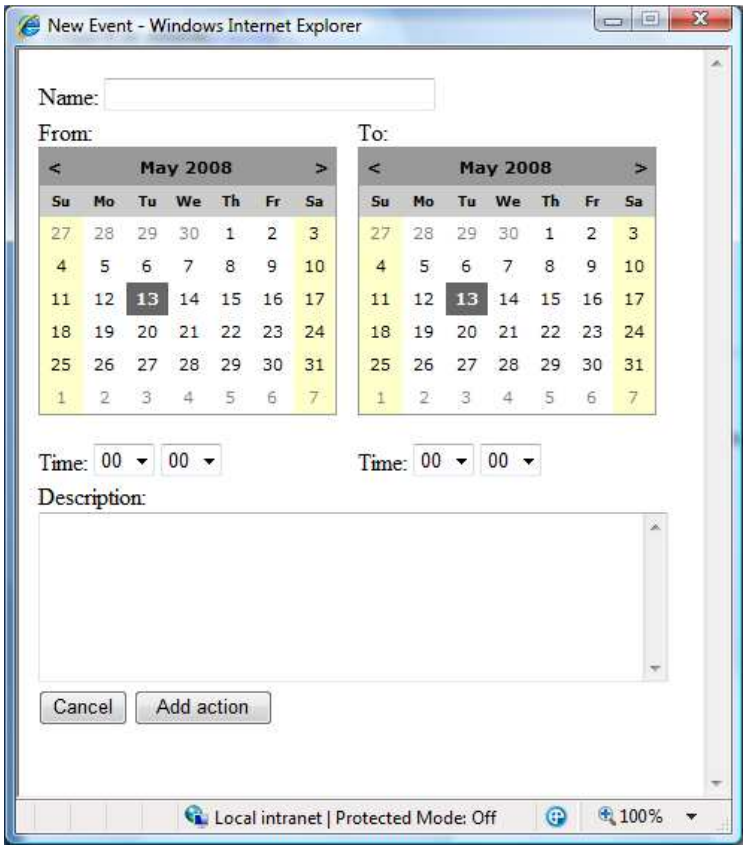
Uživatelské rozhraní je vytvořeno pomocí ASP.NET pro verzi Frameworku 3.5 (aktuální poslední release verze). Pro jeho spuštění je třeba mít nainstalovaný IIS (Internet Information Services), povoleny „server redirects“ (od verze 7.0) a mít .NET Framework 3.5.

Díky využití ASP.NET je možné formátovat výstupy z databáze pomocí předem připravených komponent. Jejich předchůdce, ASP, a stejně tak PHP musí vytvořit dotaz, zaslat jej na databázový server, výsledky uložit do tzv. recordsetu a odtud jednotlivé záznamy ručně zpracovat jeden po druhém. Takto bylo možno v cyklech generovat tabulky a seznamy.

ASP.NET nabízí alternativní řešení, GridView. Tato komponenta provede většinu operací za programátora a dovoluje editovat přímo strukturu a vzhled jednotlivých záznamů bez nutnosti práce s cykly. Je plně stylovatelná, implementuje stránkování a další užitečné funkce.

GridView v praxi je často používáno v profesionálních aplikacích jako rodič odvozených tříd. Ty poté umožňují generaci výsledného HTML skriptu způsobem, který více vyhovuje tvůrcům.

Komponenta GridView umožňuje editace a mazání záznamů při použití vhodného nastavení. V projektu je použita pouze jako komponenta pro zobrazení dat a její plný potenciál není využit. Bylo výhodnější použít vlastní formuláře pro editaci dat a tvorbu nových záznamů. Ty se při zavolání zobrazují jako pop-up okna a při zavření zasílají zprávu do okna rodiče, by obnovilo obsah. Tím je zajištěna aktuálnost obsahu.



Obr. 16: Pop-up dialog pro tvorbu nového záznamu do kalendáře (cal_add.aspx)

GridView umožňuje definování několika typů sloupců. Nejčastěji používaným v projektu je TemplateField. Dovoluje vkládat libovolný obsah ve formátu HTML, který je poté zobrazen jako jeden záznam v seznamu / tabulce. V projektu je často pro generaci obsahu použita uživatelská funkce (např. *rights(int,int)* v případu manage.aspx, viz Obr. 17.), která vysází tagy pro obrázky, odkazy a další prvky v závislosti na oprávněních uživatele nebo obsahu záznamu, který je aktuálně zpracováván.

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataSourceID="SqlDataSource1" AllowSorting="True"
            EmptyDataText="No users registered yet.."
>
<Columns>
  <asp:BoundField DataField="usr_displayname" HeaderText="Display name"
                  SortExpression="usr_displayname" ReadOnly="true" />
  <asp:BoundField DataField="usr_group" HeaderText="usr_group"
                  SortExpression="usr_group" Visible="False" />
  <asp:TemplateField HeaderText="Rights">
    <ItemTemplate>
      <%# rights(Convert.ToInt32(Eval("usr_group")),
          Convert.ToInt32(Eval("usr_id")))%>
    </ItemTemplate>
  </asp:TemplateField>
</Columns>
</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:defaultConnector %>"
    SelectCommand="SELECT [usr_id], [usr_displayname], [usr_group]
                   FROM [users] WHERE [usr_group]<10"
    ProviderName="<%$ ConnectionStrings:defaultConnector.ProviderName %>">
</asp:SqlDataSource>
```

Obr. 17: Volání ASP.NET komponenty GridView. Převzato z manage.aspx (viz zdrojové kódy na přiloženém CD)

4.3.1.1 Strukturované uspořádání uživatelského rozhraní a jeho zabezpečení

Projekt obsahuje MasterPage, kterou využívají všechny další ASPX stránky, kromě těch, které se zobrazují pomocí pop-up oken a plní úlohu dialogů.

Dále je definována jedna komponenta, loginbox. ASP.NET poskytují sice i tuto komponentu, ovšem pro potřeby projektu je až příliš složitá. Loginbox vyhodnocuje při každém volání MasterPage, zda je uživatel přihlášen. Informace o aktuálně přihlášeném uživateli jsou uloženy v Session proměnných (proměnné uloženy na serveru, které se vztahují na současné připojení. Po zavření okna exploreru dojde ke změně SessionID a po deseti minutách neaktivity i k odstranění Session proměnných na straně IIS). Kromě uživatelského ID je zde také jeho oprávnění. Pokud Loginbox zjistí, že uživatel přihlášen není, zobrazí se rámeček pro přihlášení. Jinak je zobrazena v horní části okna informace o uživateli s možností manuálního odhlášení.

Když je bezpečnost řešena na straně aplikace a ne databázového serveru, je třeba zabezpečit všechny stránky tak, aby na ně mohl přistupovat pouze někdo, kdo k tomu má oprávnění.

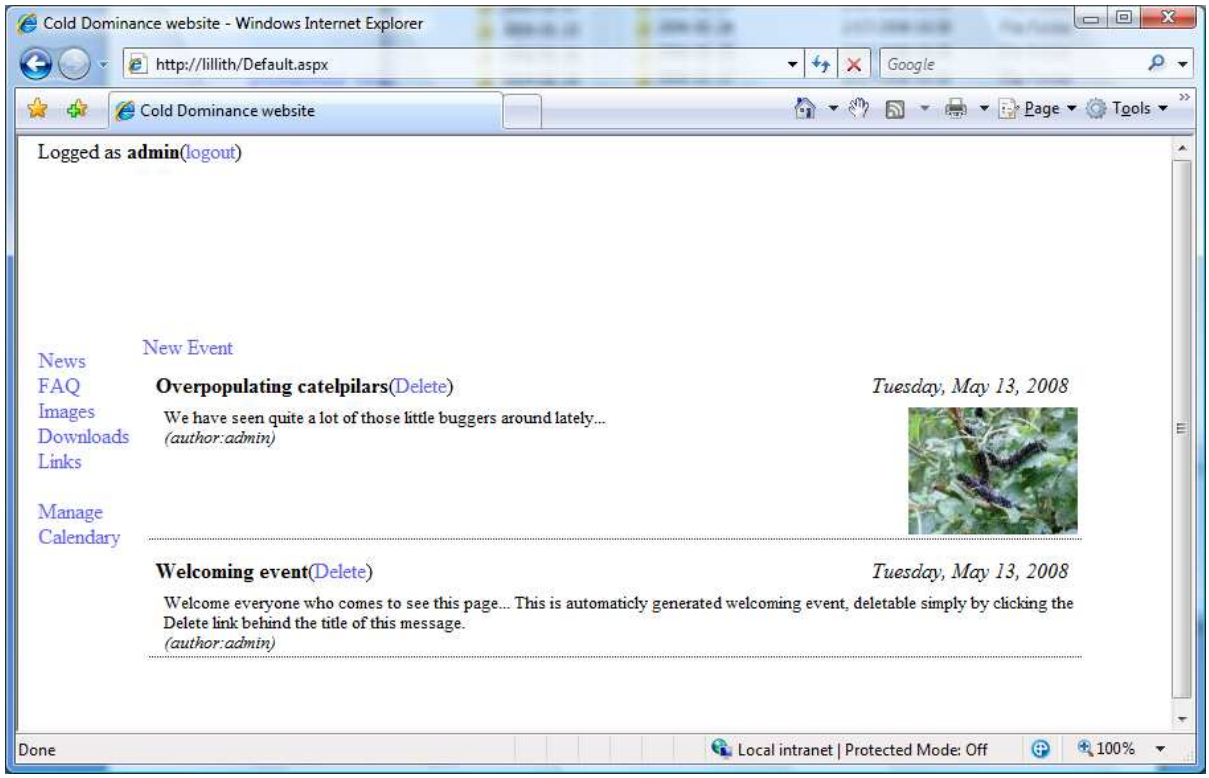
Jednotlivé části webu po přihlášení poskytují rozšířenou funkcionalitu v závislosti na úrovni oprávnění, kterou uživatel má. Pro neautorizovaného uživatele jsou rozšířené volby skryty (nikoliv neaktivní) a při jakémkoliv pokusu o přístup pomocí změny adresy v prohlížeči je dříve, než je zaslána jakákoliv informace o cílové stránce automaticky přesměrován na error.aspx s informací, že pro provedení požadované operace nemá dostatečná oprávnění.

```
protected void Page_Load(object sender, EventArgs e)
{
  if (Convert.ToInt32(Session["group"]) < 9)
       Response.Redirect("error.aspx");
}
```

Obr. 18: Příklad přesměrování v případě o pokus k přístupu ke správě uživatelů. Proměnná group v Session nese informaci o úrovni oprávnění uživatele.

4.3.2 Uživatelské rozhraní

Aplikace nabízí požadovanou funkcionalitu dostupnou pomocí jednoduchého menu na levé straně okna prohlížeče viz Obr. 19.



Obr. 19: Titulní strana – default.aspx po přihlášení s právy administrátora webu.

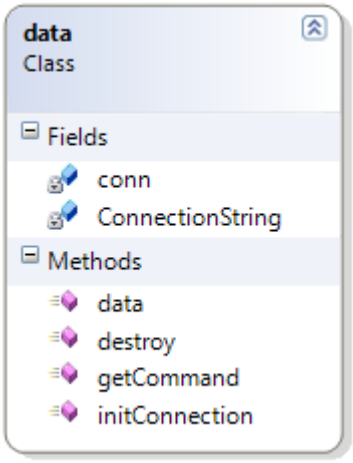
Ovládání je velmi intuitivní. Nástroje pro editaci jsou zobrazeny jen při dostačující úrovni oprávnění. Je třeba povolit pop-up okna, aby bylo možné využít dialogů pro nové události, obrázky, atd. (viz Obr. 16)

4.3.3 Propojení na databázi

Konstrukce OleDB spojení v .NET vyžaduje zisk tzv. ConnectionStringu. To je textový řetězec obsahující jméno „providera“ (název ovladače ODBC/OleDB pro práci s danou databází), adresu serveru a autentizaci. Projekt využívá možnosti Web.Config souboru a ukládá tento ConnectionString zde. Jeho zisk je potom otázkou dvou příkazů a při změně databáze stačí tento řetězec upravit tak, aby místo původního SQL Serveru adresoval např. Oracle.

```
<connectionStrings>
    <add name="defaultConnector"
         connectionString="Provider=SQLOLEDB; Data Source=Aiamee;Initial
            Catalog=pubsys;Persist Security Info=True;
            User ID=pubsys;Password=pubsys"
         providerName="System.Data.OleDb"/>
</connectionStrings>
```

Obr. 20: Příklad ConnectionStringu ze souboru Web.Config pro připojení na MS SQL Server prostřednictvím OleDB. Jméno SQL Serveru je Aiamee, login i heslo „pubsys“. Jako počáteční databázi vybereme pubsys.



Pro další zjednodušení volání SQL dotazů je zavedena nová třída *data*. Ta implementuje načtení ConnectionStringu, tvorbu připojení, inicializaci SQL dotazu a následné zničení spojení.

Obr. 21: architektura třídy data.

Díky této malé třídě, je možno zavolat SQL dotaz velmi snadno třemi řádky kódu.

```
data dat = new data();
OleDbCommand comm = dat.getCommand("SELECT COUNT(*) FROM messages");
int rcount = Convert.ToInt32(comm.ExecuteScalar());
dat.destroy();
```

Obr. 22: Příklad volání SQL dotazu pomocí třídy data. Není třeba se zabývat konstrukcí spojení. Vše je obstaráno transparentně.

Třída *data* je však vhodná k použití pouze pro dotazy, kde neočekáváme návratovou hodnotu v podobě indexu naposledy vloženého záznamu. K tomu účelu jsou v databázích různé systémové proměnné a funkce (např. *@@IDENTITY* v MS SQL, *last_identity*() pro MySQL, atd.). Jelikož je aplikace univerzální, tyto funkce nelze použít. Pokud bychom je plánovali využít, bylo by třeba volat různé SQL skripty v závislosti na typu aktuálně připojené databáze. Proto je pro operace INSERT použita složitější konstrukce, bez využití třídy data.

```
OleDbConnection dbConn1 = new OleDbConnection(ConnectionString);
OleDbDataAdapter dbAdapt1 = new OleDbDataAdapter("SELECT * FROM news", dbConn1);
dbAdapt1.MissingSchemaAction = MissingSchemaAction.AddWithKey;
OleDbCommandBuilder dbCB1 = new OleDbCommandBuilder(dbAdapt1);
dbConn1.Open();
DataSet dbSet1 = new DataSet();  dbAdapt1.Fill(dbSet1, "news");
DataTable dbTable1 = dbSet1.Tables["news"];
DataRow dbRow1 = dbTable1.NewRow();

dbRow1["nws_title"]  = ...; dbRow1["nws_date"]   = ...;
dbRow1["nws_text"]   = ...; dbRow1["nws_system"] = ...;
dbRow1["nws_author"] = ...; dbRow1["nws_hasimage"] = ...;

dbTable1.Rows.Add(dbRow1);  dbAdapt1.Update(dbSet1, "news");

if (!dbRow1.IsNull("nws_id"))    NWS_ID = (int)dbRow1["nws_id"];

dbConn1.Close();
```

Obr. 23: Příklad volání SQL dotazu INSERT na tabulku „news“. Hodnota auto-inkrementačního primárního klíče je následně uložena do proměnné NWS_ID.

Je zřejmé, že tato metoda postrádá určitou eleganci. Ovšem to je cena za univerzálnost aplikace. Další možností by bylo vytvoření uložených procedur pro každý INSERT, které by vracely potřebnou hodnotu. Aplikace by potom mohla tyto procedury volat (postup je stejný pro všechny databázové servery). V případě potřeby komplikovanějších SQL dotazů by to byla výhodnější cesta. Navíc tímto neztrácí aplikace nezávislost na databázi. V případě, že by cílový uživatel chtěl jako databázi použít např. MS Access, nebylo by to s uloženými procedurami možno.

4.3.3.1 Přístup k souborům a obrázkům

Při ukládání souborů pro potřeby webové aplikace jsou dvě možnosti. Využít jako úložiště disk a do databáze pouze zaznamenat jméno souboru, nebo celý soubor uložit v binární podobě v databázi.

V projektu je použito řešení druhé, jelikož je bezpečnější proti průniku a poskytuje snadnější možnost zálohování databáze (soubory jsou automaticky zálohovány, jelikož jsou pouze dalším záznamem v tabulce. Pokud by byly na disku, bylo by třeba zálohovat i tuto adresářovou strukturu).

Dalším důvodem je, že projekt je provozován na portálu, kde proces IIS nemá právo zápisu do adresářové struktury. Pokud by aplikace měla ukládat soubor na disku, potřebuje k danému adresáři oprávnění.

Aby bylo možno obrázek uložit do databáze, je potřeba jej nejdříve převést zpět na pole typu *byte*. Následně vytvořit spojení k databázi a zapsat jej do sloupce typu *varbinary*. Pro jeho zpětné načtení je potřeba znát jeho formát a velikost v bytech.

```
byte[] image;

MemoryStream ms = new MemoryStream();
img.Save(ms, img.RawFormat);
image = ms.ToArray();
 [...]
dbRow["img_name"] = FileUpload1.FileName;
dbRow["img_data"] = image;
dbRow["img_thumb"] = thumbnail;
dbRow["img_datalen"] = image.Length;
dbRow["img_thumblen"] = thumbnail.Length;

dbTable.Rows.Add(dbRow);
dbAdapt.Update(dbSet, "images");
```

Obr. 24: Zapsání obrázku a náhledu do databáze.

Po úspěšném uložení souboru je zde již jen otázka jeho opětovného získání. V případě souboru na disku by stačilo uvést v tagu IMG k němu cestu. V tomto případě je potřeba jako zdroj obrázku uvést skript, který jej vygeneruje.

```
protected string download(int id)
{
  string str;
  str = "<a target=_new href=\"get_image.aspx?id=" + id.ToString() + "\">
        <img src=\"get_thumb.aspx?id=" + id.ToString() + "\" alt=\"\"/>
        </a>";
  if (Convert.ToInt32(Session["group"]) >= 7)
     str += "<br /><a href=\"delete_image.aspx?id=" + id.ToString() +
            "\">Delete</a>";
  return str;
}
```

Obr. 25: Funkce z images.aspx, která obstarává generování odkazu a tagu IMG na obrázek. V případě dostatečných práv přidá možnost soubor smazat.

Jak je zřejmé z Obr. 25, místo obrázku je v parametru *src* u tagu IMG uváděn další skript. Tento skript potom musí získat obrázek z databáze a přesměrovat na svůj výstup. Dále je třeba doplnit hlavičky o informacích o souboru.

Informace, které skript (viz Obr. 26) zapisuje do hlavičky, jsou jeho nejdůležitější částí. Tím dá prohlížeči informace o tom, jaký typ dat očekávat a ten nepozná rozdíl mezi daty získanými touto cestou a klasicky adresovaným obrázkem. Stejným způsobem jsou zpracovávány i skripty get_image.aspx a get_file.aspx pro získání nezmenšené verze obrázku a souboru z databáze.

```
protected void Page_Load(object sender, EventArgs e)
{
  string SQL = "SELECT img_thumb, img_name, img_thumblen FROM images WHERE
               img_id = " + Request.QueryString["id"];
  string ConnectionString = [...];

  OleDbConnection dbConn = new OleDbConnection(ConnectionString);
  OleDbCommand dbComm = new OleDbCommand(SQL, dbConn);
  dbConn.Open();
  OleDbDataReader dbRead = dbComm.ExecuteReader();
  dbRead.Read();
  Response.Clear();
  string name = (string)dbRead["img_name"];
  string ext = name.Substring(name.LastIndexOf('.'));
  Page.Response.AddHeader("content-disposition", "filename=" + name);
  Response.ContentType = ext;
  Response.OutputStream.Write((byte[])dbRead["img_thumb"], 0,
                              (int)dbRead["img_thumblen"]);
  dbConn.Close();
  Response.End();
}
```

Obr. 26: Tělo funkce Page_Load ze skriptu get_thumb.aspx

5 Shrnutí

	SQL Server	Oracle	MySQL	PostgreSQL
Nezávislost na platformě	Ne	Ano	Ano	Ano
Vhodné řešení pro webový Frontend	ASP.NET	Java	PHP	PHP
Cena	$24,999 (Enterprise) $5,999(Standard) $49(Developer) zdarma(Express)	Enterprise: $40,000 / CPU $800 / uživatel RAC: $20,000 / CPU $400 / uživatel Security Pack : $10,000 / CPU	Zdarma €479 za podporu	Zdarma
Business Intelligence	Ano	Ano	Ne	Ne
Bezpečnost	Vynikající	Dobrá (pomalé opravy a více průniků)	Špatná (více mezer a pomalé opravy)	Vynikající
Podpora XML	Uložení, indexace, validace, kontrola obsahu	Uložení	žádná	Uložení, indexace, validace
Administrace	Snadná, přehledná	Nutno hlouběji znát problematiku databází	Nástroje třetí strany pro správu přes PHP	Nástroje pro správu přes PHP nebo klientskou aplikací
Vhodná velikost projektu (informativní)	Střední a velké projekty pro Enterprise edici, malé a střední pro Standard a Express	Velké projekty	Malé a střední	Střední a velké

6 Závěr

Bylo celkem obtížné porovnávat některé databázové stroje. Lze je porovnávat z pohledu výkonu, z pohledu stability, bezpečnosti, nebo z pohledu vývojáře. Nejmarkantnější rozdíly jsou právě v poslední kategorii. Není možné říci, že MySQL se nehodí na velké projekty, nebo že Oracle je používán na všech velkých projektech, jelikož to není pravda. Při vodné úpravě každé z hodnocených databází, dosahují dobrých výsledků. Právě proto jsem se pokusil zaměřit více na přínosy pro správce a vývojáře aplikací, které budou databáze využívat. V tomto ohledu je zřejmé, že Microsoft i Oracle jsou roky před konkurencí. Jednoduchost, s jakou je možno databázi do aplikace začlenit a potom hlavně možnosti integrovaného prostředí Visual Studia / Developer Suite jsou něco, co zvyšuje produktivitu u .NET / Java vývojářů vysoko nad ostatní.

Měl jsem možnost již dříve pracovat s Developer Suite a Oraclem. Některé vlastnosti jsou zde proti VS lepší (editace kódu – např. automatické vkládání use namespace při kopírování kódu). Ovšem preferuji ASP.NET s SQL Serverem hlavně díky nižším nárokům na hardware a nízké ceně Developer verze ($49), díky čemuž není třeba vynakládat další prostředky na databázi.

Další vývoj tohoto projektu vidím v jediné alternativě a to rozdělit aplikaci tak, aby část z .NET pracovala s databázovými servery pomocí komponent, které nejsou univerzální (tj. se SQL Serverem pomocí SQLConnection, s Oracle pomocí OracleConnection). Dále udělat klon do PHP, který by pracoval taktéž za pomoci funkcí určených přímo k danému serveru a taktéž klon do Javy, kde by se projevil vliv začlenění Javy do Oracle. Mohlo by být zajímavé porovnat si možnosti jednotlivých alternativ. Bohužel však komponenty pro přístup k MySQL a PostgreSQL z .NET jsou licencované a podléhají autorským právům.

Literatura

[1] MSSQL2005_ORACLE10g_compare.pdf
(http://www.wisdomforce.com/dweb/resources/docs/MSSQL2005_ORACLE10g_compare.pdf)

[2] OW_30820_JAVA_STORED_PROC_paper.pdf
(http://www.oracle.com/technology/tech/java/java_db/pdf/OW_30820_JAVA_STORED_PROC_paper.PDF)

[3] Comparing SQL Server 2005 and Oracle
(http://www.microsoft.com/sql/prodinfo/compare/oracle/default.mspx)

[4] Ajax (programming) - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/AJAX)

[5] Connection Strings
(http://www.carlprothman.net/Default.aspx?tabid=81)

[6] Extract, transform, load - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/Extract,_transform,_load)

[7] Microsoft SQL Server - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/Microsoft_SQL_Server)

[8] Microsoft SQL Server Comparing SQL Server 2005 and Oracle
(http://www.microsoft.com/sql/prodinfo/compare/oracle/default.mspx)

[9] Migrating from Oracle When SQL Server™ 2005 Makes Sense
(http://www.devx.com/MicrosoftISV/Article/34970/0/page/1)

[10] MySQL - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/MySQL)

[11] MySQL - MySQL vs MS SQL Server
(http://swik.net/MySQL/MySQL+vs+MS+SQL+Server)

[12] Oracle Database - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/Oracle_Database)

[13] Oracle PHP Developer Center
(http://www.oracle.com/technology/tech/php/index.html)

[14] Oracle Developer Suite 10g Product Center
(http://www.oracle.com/technology/products/ids/index.html)

[15] Oracle RAC - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/Oracle_RAC)

[16] Oracle vs. SQL Server Why Oracle wins
(http://searchoracle.techtarget.com/tip/1,289483,sid41_gci910543,00.html?ShortReg=1&mboxConv=searchOracle_RegActivate_Submit&)

[17] OracleJVM and Java Stored Procedures
(http://www.oracle.com/technology/tech/java/jsp/index.html)

[18] PostgreSQL - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/PostgreSQL)

[19] Road test Four databases tested - Architect - Database - Builder AU
(http://www.builderau.com.au/architect/database/soa/Road-test-Four-databases-tested/0,339024547,339224962-7,00.htm)

[20] Teradata - Wikipedia, the free encyclopedia
(http://en.wikipedia.org/wiki/Teradata)

[21] Using XML with MySQL
(http://www.kitebird.com/articles/mysql-xml.html)

[22] SQL 2005 and Oracle 10g Security
(http://www.microsoft.com/sql/prodinfo/compare/oracle/ss2005oracle10gsecuritycompare.msp)

Seznam příloh

Příloha 1. CD/DVD : obsahuje aplikaci ve zdrojové podobě, projektovou dokumentaci a elektronickou formu této práce.