Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Porovnání objektových a relačních databázových systémů

Bakalářská práce

Jakub Geyer

Školitel: Mgr. Miloš Prokýšek.

České Budějovice 2012

Geyer, J., 2012: Porovnání objektových a relačních databázových systémů.
[Comparison of object and relational database systems. Bc. Thesis, in Czech.] – 37 p.,
Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Tato práce se snaží přispět k problematice vhodné volby databázových platforem. Klíčové vlastnosti objektových a relačních databázových systémů jsou zde vzájemně porovnávány a podrobeny testům na konkrétních zástupcích jednotlivých platforem.

Abstract

This thesis focuses on the issue of a convenient choice of database platforms. The key features of the object database systems and the relational database systems are mutually compared and tested on concrete representative samples of each individual platform.

Poděkování

Rád bych velice poděkoval školiteli mé bakalářské práce panu Mgr. Miloši Prokýškovi za cenné rady a čas strávený při konzultacích.

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

České Budějovice, 27. 4. 2012.

Jakub Geyer

Obsah

1. Úvod

Vývoj databázových systémů a způsobů ukládání dat obecně je stále rychleji se rozvíjející odvětví informatiky. Vznikají nové struktury a způsoby ukládání dat, jejichž cílem je co nejrychlejší a nejpřehlednější přístup k datům uložených v databázi. Jedním z hlavních trendů ve způsobu práce s daty, jakožto i v programovacích jazycích obecně, je přechod od strukturovaného programování k objektově orientovanému. Jako reakce na tento trend vznikly i objektově orientované, respektive objektové databáze.

I přes masivní rozvoj objektového programování však v současné době počtem nasazení stále převažují databáze relační, především díky jejich oblibě v oblasti webových stránek[1]. To je umožněno především jejich dostupností, poměrně snadnou implementací a jejich obvykle dostačujícími parametry pro konkrétní řešení.

Proti později vzniklým databázovým typům[2] (post-relační[3], objektové databáze) mluví fakt, že není přesně definováno, kdy se opravdu vyplatí tyto typy použít. Velká část vývojářů se tak spokojí s již zaběhlými relačními databázemi, které jsou vzhledem k době jejich používání již odladěny, existuje k nim velké množství materiálů a často hraje v jejich prospěch také fakt, že výhody nových databázových typů nemusí být na první pohled patrny.

1 *Most Widely Deployed SQL Database. SQLite [online]. 2006 [cit. 2012-03-29]. Dostupné z: http://www.sqlite.org/mostdeployed.html*

2 *Viz. Definice pojmů a zkratek - databázový typ.*

3 *INTERSYSTEMS CORPORATION. InterSystems Caché [online]. 1996-2012 [cit. 2012-03-29]. Dostupné z: http://www.intersystems.com/cache/*

1.1. Formulace problému

Ačkoliv jsou relační databáze s úspěchem využívány v řadě projektů, je možné se domnívat, že vzhledem k trendu přechodu ze strukturálního programování na objektové, by v některých případech bylo vhodnější využít databáze objektové.

Problémem však je, že velká část vývojářů zůstává i nadále u relačních databází z následujících důvodů[4, 5]:

- Velké rozšíření relačních databázových systémů. Díky tomuto rozšíření se část vývojářů uchýlí právě k relačním databázím. Z diskuzí vyplývá, že někteří o existenci objektových databází a jejich výhodách nevědí zcela.
- Dlouhodobé užívaní a odladěnost relačních databázových systémů je jeden z hlavních argumentů zastánců relačních databází. Vzhledem k počtu nasazení se lze domnívat, že chyby v distribucích (ve stabilních verzích) byly již odstraněny.
- Snadná dostupnost i nekomerčních verzí relačních databází. Naopak distribuce objektových databází existují především v komerčních verzích.
- Zákazníkům či vývojářům nejsou patrny výhody použití objektových databází.
- Nasazení v již existujících projektech by znamenalo nutnost převodu již existující databáze a případně i s ním spojený nárůst finančních prostředků.
- Nedostatečné množství materiálů věnovaných objektovým databázím.

Lze se domnívat, že z výše uvedených důvodů se vývojáři připravují o možné výhody, plynoucích z využití objektových databází. Jedná se například o přehlednější formu uložení dat (více odpovídající objektům v reálném světě), která je snadněji pochopitelná pro „neodborníky“ (např. osoby podílející se na vzniku projektu, kteří nejsou softwarový vývojáři, zadavatelé apod.), přímý přístup k datům (objektům) v paměti, potencionálně vyšší rychlost při práci s daty (při CRUD) nebo možnost využít vícenásobné dědičnosti a integrace funkcí (metod) přímo do objektů (resp. tříd).

4 Why don’t people simply use “Object Database”s?. In: Stackoverflow [online]. 2009-2010 [cit. 2012-03-29]. Dostupné z: http://stackoverflow.com/questions/1612169/why-dont-people-simply-use-object-databases

5 Object Oriented vs Relational Databases. In: Stackoverflow [online]. 2009-2011 [cit. 2012-03-29]. Dostupné z: http://stackoverflow.com/questions/800/object-oriented-vs-relational-databases

1.2. Výzkumné otázky

1. Za jakých okolností je nasazení objektové databáze vhodnější než nasazení databáze relační z pohledu výkonu databáze?
2. Jaké výhody může poskytnout objektová databáze vývojáři ve srovnání s relační?
3. Jaká je náročnost objektových databází na zdroje počítače (specificky zatížení procesoru, využití operační paměti a pevného disku) v porovnání s relačními databázemi?

1.3. Cíl práce

Hlavním cílem práce je identifikovat rozhodující ukazatele pro nasazení objektové či relační databáze. Tento cíl je dále možné rozdělit na následující dílčí cíle:

1) Určit klíčové vlastnosti databázových typů a formulovat hlavní rozdíly.
2) Definovat obecné charakteristiky situací, ve kterých je zvláště výhodné nasazení objektových databází.
3) Identifikovat a porovnat náročnost jednotlivých typů databází na zdroje počítače.

1.4. Úkoly pro dosažení cíle

Vzhledem ke značným odlišnostem v jednotlivých databázových typech je nejdříve nutno analyzovat základní principy práce s oběma typy, vyhledat případné společné rysy a určit možné měřitelné parametry pro srovnání. V kontextu k cílům lze poté definovat následující dílčí úkoly:

1. Analyzovat hlavní výhody databázových typů z hlediska:
    a. Metod programování.
    b. Struktury databáze.
2. Nalézt aktuálně dostupné produkty na trhu.
3. Popsat průběh převodu RDB => OODB.
4. Empiricky ověřit časovou náročnost dotazů na databázích sestavených podle modelu, který umožní otestovat odlišnosti obou databázových typů.

1.5. Použité nástroje a metody

Při plnění úkolů je nejdříve využito teoretických metod. Jedná se především o studium odborné literatury, odborných diskuzí a analýzy dostupných databázových produktů.

Pro ověření závěrů teoretické části bude využito metody experimentu. Během experimentu bude využit vlastní softwarový nástroj pro měření sledovaných ukazatelů a provádění operací nad databází. Výsledky experimentu budou zpracovány pomocí statistických nástrojů v programu Microsoft Excel 2010.

2. Teoretická část

Teoretická část práce se zaměřuje především na analýzu relačních a objektových databázových systémů, jejich historie, struktury, mechanizmů, odlišností a na jejich vzájemné porovnání.

2.1. Relační databáze

První návrh na implementaci RDM publikoval v roce 1970 pracovník firmy IBM matematik Edgar Frank „Ted“ Codd[6]. Ve své práci „A Relational Model of Data for Large Shared Data Banks“ navrhuje nahrazení hierarchické nebo síťové struktury jednoduchými tabulkami obsahující řádky a sloupce. Důraz je zde kladen na oddělení významových dat a dat představujících databázovou strukturu (relace), a rovněž na nutnost vhodně navrhovat databázové modely za účelem omezení redundance dat.[7]

Dalším krokem byla snaha o vytvoření jazyka pro práci s DB nezávislého na použitém vývojovém prostředí. Prvním takto nevrženým jazykem se stal Sequel (později SQL) představený v roce 1974, který byl použit v System-R od IBM.[8]

K prvnímu velkému rozšíření dochází v roce 1980, kdy firma Oracle představuje svou SQL databázi pro počítače VAX.

V roce 1986 byl pro tento jazyk přijat standard SQL-86, který však pro nedostatky v oblasti integrity databáze byl v roce 1992 nahrazen standardem SQL2 (SQL-92). Zatím posledního standardu se pak dočkal v roce 1999 jako SQL3 (SQL99), který reaguje na rozvoj objektových programovacích jazyků a přidává možnost vnořených objektů, abstraktních datových typů a použití metod.[9]

---

6 IBM Archives: Edgar F. Codd. IBM [online]. IBM Research News, 24.04.2003 [cit. 2012-04-04]. Dostupné z: http://www-03.ibm.com/ibm/history/exhibits/builders/builders_codd.html

*7 SELECT * FROM SQL History: Who was Edgar "Ted" Codd?. FairCom [online]. 2009 [cit. 2012-04-01]. Dostupné z: http://www.faircom.com/ace/enl_22_s12_t.php*

8 Root.cz. Root.cz: Historie relačních databází [online]. 19.10.2001 [cit. 2012-04-01]. Dostupné z: http://www.root.cz/clanky/historie-relacnich-databazi/

9 Oracle® Database SQL Reference. Oracle Documentation [online]. 1996-2003 [cit. 2012-04-07]. Dostupné z: http://docs.oracle.com/cd/B12037_01/server.101/b10759/toc.htm

2.1.1. Definice relačního modelu

Relační model dat má jediný konstrukt – databázovou relaci (R). Ta je vybavena pomocnou strukturou (schéma relace), která se skládá ze jména relace, jmen atributů a specifikace domén.

Domény (D_1, D_2, … D_n) jsou z databázového hlediska množiny hodnot, kterých může atribut nabývat. Prvky (x_1, x_2, … x_n) z jednotlivých domén tvoří uspořádané n-tice. Tyto prvky jsou atomické (dále nedělitelné). Toto omezení se nazývá 1. normální forma (1NF). Od matematické relace se ta databázová liší právě atomicitou jednotlivých prvků domén a vybaveností schématem relace.[10]

Schéma relace R je tvořeno atributy relace A_1:D_1, A_2:D_2, … A_n:D_n, přičemž A_c představují jména atributů a D_c domény. Schéma relace lze tedy zapsat jako R(A_1:D_1, … A_n:D_n). Relace R nad množinou A je libovolná podmnožina kartézského součinu domén D_1 x … x D_n. Doména náležící atributu C se označuje jako dom(C).

Primární klíč (K) je množinou atributů z A, jejichž hodnoty jednoznačně identifikují jedinou n-tici z relace R. Primární klíč musí proto být vždy unikátní pro každou n-tici, z čehož vyplývá integritní omezení.[11]

Dalším integritním omezením je referenční integrita, tedy existence vztahů[12] mezi dvěma relacemi. Tento vztah je definován pomocí cizího klíče – atributu, který je v nadřazené relaci klíčem primárním.

Vztah mezi relacemi lze podle násobnosti dále rozdělit na 1:1, 1:N a M:N. První dvě skupiny lze přitom jednoduše implementovat, v případě M:N by však mohlo dojít k expanzi atributů – vzniká tak další integritní omezení.[13]

10 CODD, E. F. A Relational Model of Data for Large Shared Data Banks. ACM [online]. 1970, Number 6 [cit. 2012-04-09]. Dostupné z: http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

11 POKORNÝ, Jaroslav. Databázové systémy. Praha: Karolinum, 1992. ISBN 80-7066-814-8.

12 Viz. Definice pojmů a zkratek - vztah.

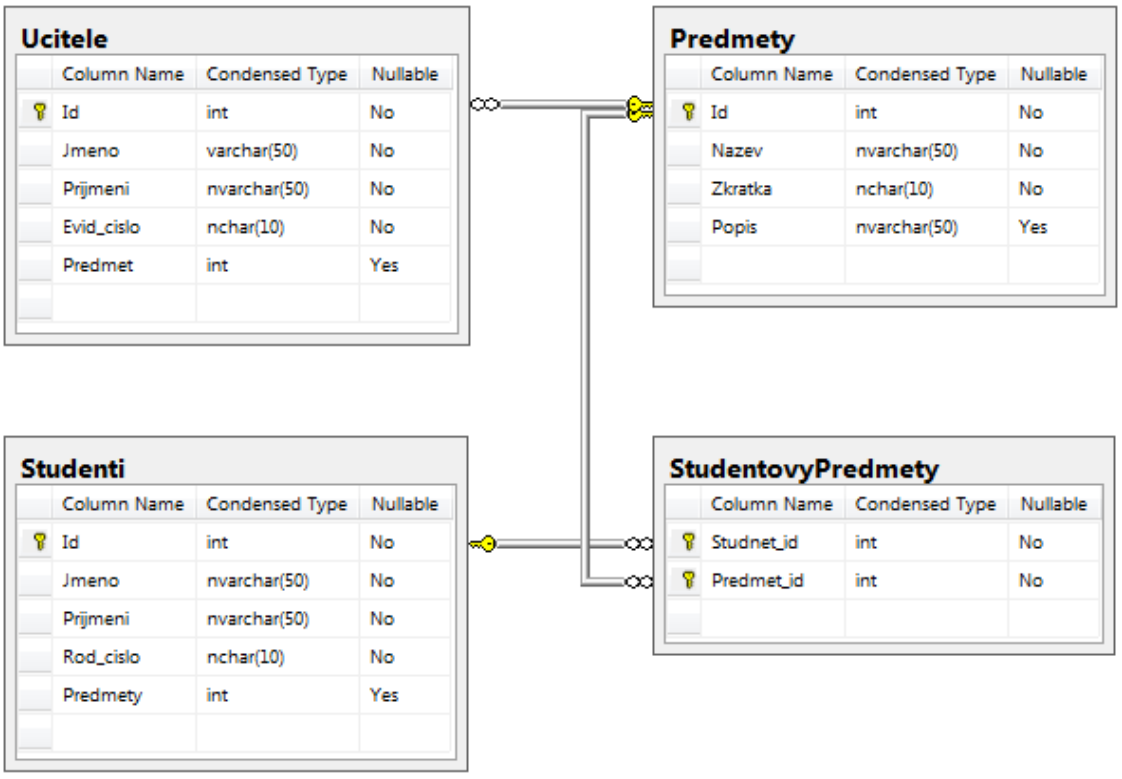
13 HERNANDEZ, Michael J. Nývrh databází. Praha: Grada, 2006. ISBN 80-247-0900-7.

Základní principy

- Nezávislost datové a aplikační vrstvy.
- Symetrický přístup k datům.
- Dvě metody práce s daty – relační kalkul a relační algebra.
- Omezení redundance dat.

2.1.2. Reprezentace relačního modelu

Jednotlivé relace jsou pro uživatele reprezentovány pomocí tabulek. N-tice pak představují řádky těchto tabulek a atributy její sloupce. Z výše uvedené definice tedy jasně plyne, že známe-li jméno relace (tabulky), primární klíč (identifikaci n-tice - řádku) a atribut (sloupec), jsme schopni číst data. Cizí klíče je možno chápat jako ukazatele z jedné tabulky do druhé, umožňující jejich logické provázání. Násobnosti vazby značí počet řádků jedné tabulky provázaných s počtem řádků tabulky druhé. Vazba M:N bývá realizována přidáním další tabulky (zvané asociační).



Obrázek 1 Ukázka struktury relační databáze

Ucitele				
Id	Jmeno	Prijmeni	Evid_cislo	Předmět
1	Franta	Novák	123456	2
2	Jan	Duha	123789	1

Predmety			
Id	Nazev	Zkratka	Popis
1	Matematika	M	
2	Dějepis	D	

Obrázek 2 Ukázka grafické reprezentace dat v relační databázi

2.1.3. Definice plně relační databáze (podle E. F. Codda)

V roce 1985 prezentoval „Ted“ Codd 12 pravidel (resp. 13), jak by měla vypadat plně relační databáze[14, 15]:

- Základní pravidlo
  RDBMS musí spravovat svá uložená data pouze pomocí svých relačních schopností.
- Pravidlo reprezentace informací
  Všechny informace jsou reprezentovány jako proměnné v tabulce.
- Pravidlo garantovaného přístupu
  Všechna data v RDB jsou přístupná pomocí názvu tabulky, primárního klíče a názvu sloupce.
- Systematické řešení nulových hodnot
  Nulové hodnoty (jiné než prázdný řetězec typu char či string) jsou podporovány jako reprezentace chybějících údajů v databázi.
- Dynamický on-line katalog založený na relačním modelu
  Popis databáze je reprezentován na logické úrovni stejným způsobem jako běžná data, takže oprávnění uživatelé mohou použít stejný relační jazyk k dotazování jako při práci z běžnými daty.
- Pravidlo komplexního datového podjazyka
  Relační systém může podporovat několik jazyků a módů pro dotazování, vždy však musí existovat nejméně jeden jazyk s dobře definovanou syntaxí, který podporuje definice dat a pohledů, manipulaci s daty, integritu, autorizaci a transakce.

14 CODD, E. F. Is Your DBMS Really Relational?. ComputerWorld. 1985, 14. října.

15 CODD, E. F. Does Your DBMS Run By the Rules?. ComputerWorld. 1985, 21. října.

- Pravidlo tvoření pohledů
  Všechny pohledy, jejichž vytvoření je teoreticky možné, lze systémem vytvořit.
- Vkládání, úpravy a mazání na vysoké úrovni
  Schopnost zachovat relační pravidla u základních i odvozených relací platí nejen pro získávání dat, ale také pro jejich vkládání, úpravy a mazání.
- Fyzická datová nezávislost
  Aplikační vrstva je zcela oddělena od fyzické datové vrstvy.
- Logická datová nezávislost
  Aplikační vrstva je zcela nezávislá na logické datové struktuře.
- Integritní omezení
  Všechna integritní omezení musí být možné uložit v katalogu databáze, nikoliv v aplikaci.
- Distribuční nezávislost
  DBMS musí být schopen implementace na různých platformách.
- Jazyky nízké úrovně
  Pokud relační systém podporuje dotazovací jazyk nízké úrovně (jeden záznam najednou), pak tento nesmí být použit k porušení či obejití integritních pravidel a je nutno použít jazyk úrovně vyšší (více záznamů najednou).

V současné době neexistuje žádný databázový produkt, který by splňoval všechna tato pravidla. 11 z nich splňuje (některé alespoň částečně) Oracle Database 11g.[16]

16 ASNANI. Oracle Database 11g: Hands-On Sql & Pl/sql. New Delhi: PHI Learning Private Limited, 2010. ISBN 978-81-203-4020-6. Dostupné z: http://books.google.cz

2.1.4. Přehled zástupců

Existuje velké množství produktů z oblasti relačních databází, které pokrývají prakticky celé spektrum možných kombinací. Lze naleznout produkty pro všechny platformy a programovací jazyky (kde má využití databází smysl), jak komerční tak i volné verze (freeware i open source řešení).

Přehled nejpopulárnějších RDBMS[17]:

1. Oracle Database (Oracle Corp.)
2. SQL Server (Microsoft)
3. DB2 (IBM)
4. Sybase database (Sybase, an SAP company)
5. Ingress (Actian Corp.)
6. MS Access (Microsoft)
7. MySQL (Oracle Corp. - Sun Microsystems, Inc.)

2.2. Objektově orientované databáze

Myšlenka ukládat data v objektové podobě vznikla již s nástupem prvních objektových jazyků koncem 60. a v průběhu 70. let.

První návrhy objektových, respektive objektově orientovaných databází se objevují však až na počátku 80. let, kdy Won Kim ve firmě MCC (Microelectronics and Computer Technology Corporation) zakládá projekt Orion, jehož prototypem Orion-1 byly inspirovány jedny z prvních objektových databází ITASCA a Versant. [17]

Začátkem 90 let přichází první veřejně dostupné produkty od firem Gemstone a Versant. V roce 1991 je vytvořen první objektový standard ODMG 1.0, který však nebyl zcela podporován všemi (především nově vznikajícími) databázovými produkty. Poslední standard ODMG 3.0 z roku 2000 není rovněž podporován všemi výrobci objektově-orientovaných databází.[18]

17 Introduction to ODBMS: Short History. Object database management systems [online]. 2005-2012 [cit. 2012-04-03]. Dostupné z: http://www.odbms.org/Introduction/history.aspx

18 Ibidem

2.2.1. Definice objektového modelu[19, 20]

Objektově orientovaný model se více podobá objektům z reálného světa, místo řádků tabulky jsou zde tak ukládány přímo objekty, které jsou charakterizované pomocí tříd. Tyto objekty se také velmi podobají objektům známým z objektových programovacích jazyků.

Objekty jsou v databázi uloženy v kolekcích (List, Array, SortedCollection, Set, apod.), a jsou mezi nimi vazby, které představují logickou strukturu. Ke každému objektu přitom lze přistupovat jak samostatně, tak přes logickou vazbu v jiném objektu.

Pojem „třída“ představuje u objektově-orientovaných databází realizaci datového typu konkrétních objektů. Jednotlivé kolekce však nemusí obsahovat pouze objekty stejné třídy, ale díky polymorfizmu[21] (ať už vzniklého děděním, implementací rozhraní nebo dokonce jen společnými atributy) mohou obsahovat tříd více a provádět na nich operace, které jsou pro objekty společné (např. výběr či mazání).

Samotné objekty se skládají z datové složky (to mohou být jednoduché datové proměnné[22] ale i další objekty) a z metod (definovaných pomocí tříd), které objekt provádí. Tyto metody jsou jediný způsob přístupu k hodnotám v objektu (princip zapouzdření). Nejedná se však pouze o metody CRUD, ale i o složitější metody, které zpracovávají data z objektu a vytváří nová, která v objektu uložena nebyla (např. z data narození vypočte věk). Část výpočtů (nebo i všechny) lze tak přesunout z aplikační na databázovou úroveň.

19 KIM, Won. Introduction to Object-Oriented Databases. Cambridge, Massachusetts: The MIT Press, 1990. ISBN 978-0262111249.

20 ATKINSON, M., F. BANCILHON, D. DEWITT, K. DITTRICH, D. MAIER a S. ZDONIK. The Object-Oriented Database System Manifesto. Proceedings of the First International Conference on Deductive and Object-Oriented Databases [online]. 1989, December [cit. 2012-04-14]. Dostupné z: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/clamen/OODBMS/Manifesto/

21 Viz. Definice pojmů a zkratek - polymorfizmus.

22 Viz. Definice pojmů a zkratek – datové proměnné.

Každý objekt v OODB má své jednoznačné OID (object identifier), které je fyzickým ukazatelem na databázi a objekt v ní. Pří získávání objektu dochází k jedinému překladu, a to v rámci konkrétního databázového souboru z OID na adresu umístění objektu v daném souboru. OID (spravované pomocí OODBMS) je přiděleno každému nově vloženému objektu do databáze a zůstává neměnné po celou dobu existence objektu. Díky tomu může existovat i více obsahově zcela stejných objektů, které jsou však pro databázi pomocí OID rozlišeny. OID se nemění nikdy, tedy ani při změně atributů (i všech), a z pravidla nebývá ani po odstranění objektu přiděleno objektu jinému.

Základní principy

- Ukládání objektů (bez nutnosti transformace).
- Třídy obsahující datovou strukturu a metody.
- Zapouzdření objektů.
- Dědění tříd.
- Polymorfizmus.
- Co nejpřímější přístup k souborům (OID).

Použití s neobjektovými programovacími jazyky

Vzhledem ke způsobu ukládání dat není možné použít objektově-orientované databáze spolu s neobjektovými programovacími jazyky.

2.2.2. Přehled zástupců

Tabulka 1 Přehled základních parametrů nejznámějších objektových databází

Název databáze (výrobce)	Popis (výtah z popisu výrobce)	Technické údaje (výtah, podrobnosti v odkazu)	Licence
DB4O (Versant corp.)	Open-source objektová databáze s API cílenými pro .NET a JAVU. Důraz je kladen na výkon, přehlednost, flexibilitu a snadné použití.	Aktuální verze: 8 Win / Linux / Mac OS API pro .NET 3,5 a 4.0 C#, Java JDK 5+ [23]	Open source (GPL) nebo komerční verze s plnou podporou (non-GPL).
Objectivity/DB (Objectivity, Inc.)	Objektová databáze s důrazem především na škálovatelnost a možnou velikost uložených dat (i více než yottabyte = 10^{24} bytů).	Aktuální verze: 10 Win / Linux / Mac OS API pro C++, C#, Java, Smalltalk, Python 32 i 64bit [24]	Trial 60dní / komerční.
ObjectStore (Progress Software)	Objektová databáze cílená pro aplikace s výpočty v reálném čase.	Aktuální verze: 7.3 Win / Linux API pro Java, C++ 32 i 64bit [25]	Demo / komerční verze.
ObjectDB (ObjectDB Software, Ltd)	Objektová databáze s důrazem na jednoduchost použití s API pouze pro JAVU (JPA 2 / JDO 2).	Aktuální verze: 2.3 Win / Linux / Mac OS API pro Java 32 bit [26]	Freeware (*s omezením*) nebo komerční verze s plnou podporou.
Caché (InterSystems) * jedná se o postrelační databázi	Umožňuje jak objektový tak relační přístup k datům (pomocí SQL dotazů).	Aktuální verze: 2012.1 Win / Linux / Mac OS API pro Java, C++, Basic 32 i 64 bit [27]	Freeware (*s omezením*) nebo komerční verze s plnou podporou.

23 Další specifikace dostupné na http://www.db4o.com/about/productinformation/datasheet/

24 Další specifikace dostupné na http://www.objectivity.com/pages/objectivity/features.asp

25 Další specifikace dostupné na http://www.progress.com/docs/datasheets/objectstore/objectstore_ds.pdf

26 Další specifikace dostupné na http://www.objectdb.com/object/db/database/features

27 Další specifikace dostupné na http://www.intersystems.com/cache/

2.3. Porovnání relačního a objektově orientovaného modelu

Z výše zpracované analýzy jednotlivých databázových typů plyne celá řada rozdílů, které mohou mít značný dopad na výhodnost použití jednoho či druhého typu.

Tabulka 2 Přímé porovnání z hlediska datového modelu

RDM	OODM
Relace (tabulka)	Kolekce (množina objektů – i z více tříd)
N-tice (řádek)	Objekt
Atribut	Datová složka a metody objektu
Primární klíč (na logické úrovni)	OID (na fyzické úrovni)

2.3.1. Porovnání z hlediska základních databázových operací (CRUD)

Podoba připojení do databáze se u obou databázových typů výrazně neliší. Oba vyžadují pro připojení identifikaci serveru, portu, cestu a ověření a oba využívají pro komunikaci TCP/IP protokol.

Vkládání dat do databáze se však u obou databázových typů zásadně liší. V případě relačních databází je nejdříve nutno vytvořit tabulky a nastavit typ jednotlivých atributů, primárních a cizích klíčů, autoincrementací apod., teprve potom je možné vkládat řádky. V případě objektových databází je ukládání objektů mnohem jednodušší. Objekt zde lze uložit přímo bez dalších nezbytných příprav, obvykle pomocí metod data provideru, kde se jako parametr použije celý objekt, jehož uložení je požadováno. Není potřeba specifikovat strukturu objektů – ta je již vytvořena v aplikaci a je proto do databáze převzata včetně metod.

Při čtení dat závisí konkrétní podoba dat získaných z databáze především na použitém dotazovacím jazyku. Z principů relačního modelu vyplývá, že je schopen vracet uspořádané hodnoty pouze v podobě jednoduchých datových typů (string, integer, boolean, apod.). S těmi lze posléze pracovat přímo, nebo je lze mapovat na objekty, což umožňují některé dotazovací jazyky (např. Linq To SQL[28]). Naproti tomu návratovou hodnotou objektově-orientované je objekt, mapování nebo další úpravy tedy není nutné provádět.

28 LINQ to SQL: .NET Framework 4. Microsoft development network [online]. 2010-2012 [cit. 2012-04-16]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb386976.aspx

Na druhou stranu může být zapotřebí dalších operací (jako např. vytvoření prototypu – dotazovací jazyk QBE), které je nutné provést před samotným dotazem.

U úprav dat existuje zásadní rozdíl v oblasti, na které se úpravy provádějí. Zatímco u relačních databází je možné editovat konkrétní položku v n-tici („buňku tabulky“), v objektově orientovaných se čte i ukládá vždy celý objekt. To je dáno možností atributů i jiných než primitivních typů. Při drobné úpravě v objektu v databázi se tak ukládá znovu celý objekt (přepsání), zatímco v relačních lze ukládá jen část.

Mazání n-tic či objektů v obou databázových typech je obdobné, výhodou objektových databází však může být OID jako přímý odkaz do paměti.

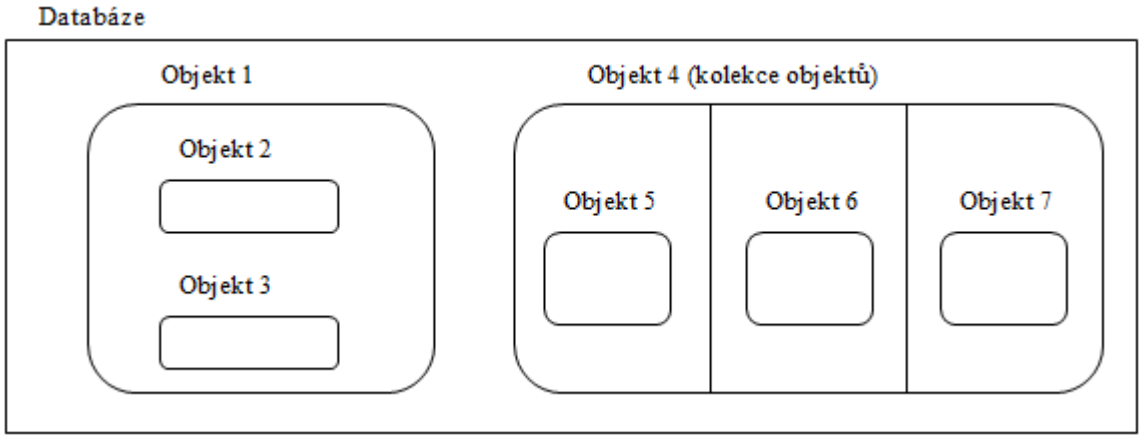
2.3.2. Porovnání způsobu provázání dat

Z analýzy obou databázových typů plyne, že mají zcela odlišný způsob k vytváření logické struktury ukládaných dat. Relační databáze se snaží logickou strukturu striktně oddělit od fyzické vrstvy, naopak objektové databáze mají za cíl co nejpřímější přístup, což může mít zásadní vliv na výkon databáze.

RDB řeší logickou provázanost dat pomocí vztahů mezi n-ticemi (za pomoci vazby primární - cizí klíč), což obvykle také vede k vytváření pohledů, autoincrementací a jejich sekvencím a spouštím. Získávání konkrétních dat pak spočívá v procházení logické struktury a až následným čtením fyzických dat. Nevýhodou tohoto principu je nutnost vyhledat odpovídající dvojici primárního a cizího klíče při každém logickém provázání 2 tabulek. Problém rychlosti vyhledávání odpovídajících klíčů řeší RDB pomocí možnosti indexování sloupců tvořících konkrétní relaci.

OODB řeší logickou provázanost zcela odlišně. Logická struktura je tvořena vnořováním objektů (či dokonce kolekcí objektů) do jiného objektu. Na fyzické úrovni je toto vnoření reprezentováno pomocí OID, které lze přeložit přímo na fyzický odkaz na objekt. To umožňuje co nejpřímější přístup bez ohledu na umístění objektu v databázi. Při získávání dat není tedy třeba nic vyhledávat, stačí identifikovat „vrchní objekt“ a všechny objekty vněm vnořeny jsou načteny společně s ním. Ke každému objektu lze přitom přistupovat jak pomocí jeho umístění (vnoření) v jiných objektech, tak i samostatně.

Nevýhodou je, že stačí-li k uspokojení konkrétního dotazu data z jednoho objektu, ale tento objekt obsahuje i další vnořené objekty (a ty mohou obsahovat další objekty), je pomocí dotazu získána celá tato struktura, i když není potřeba. Tento problém řeší OODB možností nastavením „hloubky“ dotazu, kterou lze nastavit jak při čtení, tak v případě kaskády při úpravách nebo mazání objektů.



Obrázek 3 Ukázka struktury objektové databáze

2.3.3. Porovnání funkčního zpracování dat na databázové úrovni

Oba databázové typy umožňují provádění určitých funkcí na databázové vrstvě a umožňují tak získávat z databáze již zpracovaná data (tedy jiná než jen ty původně uložená). Principem se však řešení velmi liší.

Objektové databáze ukládají metody (definované podle jeho třídy, nebo i zděděné z třídy nadřazené) jako součást uložených objektů. Tyto metody lze tak provádět přímo na objektech bez nutnosti dalších referencí.

Relační databáze umožňují ukládat do pouze základní datové typy. Pro zpracování uložených dat zde slouží tzv. procedury. Ty mají jako klasické metody vstup a výstup, kde data pro zpracování lze však získávat i pomocí interního dotazu do databáze. Z pohledu aplikace tak nemusí být vůbec zřejmé, s kterými tabulkami a řádky procedura pracuje.

2.4. Převod RDB na OODB

Z webových diskuzí vyplývá, že značná část vývojářů se obává převodu již existujících relačních databází na objektové.

Z analýzy principů objektových databází vyplývá, že převod dat z jedné struktury do druhé je možný, za předpokladu vytvoření odpovídajícího objektového modelu. Objektový model umožňuje realizovat všechny typy vazeb užívaných v relačních databázích (1:1, 1:N, N:M). Jednoduché vazby 1:1 jsou přitom realizovány pomocí vnořování objektů a vazby 1:N a N:M pomocí vnořování kolekcí objektů.

Vazby typu M:N lze přitom realizovat přímo bez nutnosti vytváření obdoby asociačních tabulek, neboť v objektových databázích, kde jsou objekty identifikovány pouze pomocí OID mohou existovat i vzájemné vazby.

Na převod mezi některými konkrétními databázovými již produkty existují specializované nástorje, např. DataWander[29], kterým lze provádět migraci z databáze Oracle 10g na DB4O 8. Největším problémem převodu databáze se tak často může stát úprava aplikací, které databázi využívají.

[29] *http://community.versant.com/Blogs/Db4o/tabid/197/entryid/883/Default.aspx*

3. Praktická část

Tato kapitola je věnována především popisu experimentálního šetření provedeného za účelem ověření některých zjištění vyplívajících z teoretické části této práce. Experiment byl navržen především s cílem porovnání rychlosti zpracování dotazů na RDB a OODB vzhledem k rozdílným přístupům k logické struktuře dat.

3.1. Porovnání rychlosti zpracování dotazu

Při porovnávání jednotlivých databázových typů bylo zjištěno velké množství odlišností, kde jedním z nejzásadnějších je způsob logického provázání dat v databázi. Obě řešení mají své výhody a nevýhody a nelze proto říci, že jeden je obecně výhodnější. Následující experiment by měl proto otestovat vhodnost jejich použití v závislosti na množství provázanosti uložených dat.

3.1.1. Cíl experimentu

Cílem tohoto experimentu je porovnat rychlost zpracování dotazu v databázi Oracle 11g a DB4O vzhledem ke hloubce dotazovaných dat.

Tabulka 3 Hardware použitý při experimentu

Základní deska	ASUS P8Z68-M PRO, Intel Z68, LGA1155, DDR3, mATX
Procesor	Intel Core i7 2700K (3,5GHz, 4 jádra, 8 vláken, 8MB L3)
Paměti	Corsair 4x4GB DDR3 1600MHz CL9 (CML16GX3M4A1600C9B)
Pevný disk	SSD OCZ Agility 3, SATA 6GB/s (525/500MB/s čtení/zápis)

Tabulka 4 Software použitý při experimentu

Relační databáze	Oracle Database 11.2.0.1.0 Standard Edition (64bit)
Objektová databáze	DB4O 8.0 for .NET 4.0 (64bit support)
Operační systém	MS Windows 7 Professional SP1 (64bit), ver 6.1.7601
Vývojové prostředí	MS Visual Studio 2010 Ultimate SP1, ver 10.0.40219.1 .NET Framework 4, ver 4.0.30319 programovací jazyk C#

Pro experiment byly zvoleny databáze Oracle database, jako jedena z nejrozšířenějších a nejdéle existujících RDB a DB4O od společnosti Versant, jako jedna z nejrozšířenějších čistě objektových databází. Přestože Oracle DB umožňuje částečně i možnost objektového přístupu, je zde použito pouze metod odpovídajících relačním databázím. Ačkoliv existuje i verze Oracle Database Express Edition, která je zdarma, je zde použita komerční verze Standart, vzhledem k omezení Express Edition na maximální využití 1GB RAM, což by mohlo vézt ke zkreslení výsledků měření.

3.1.2. Průběh experimentu

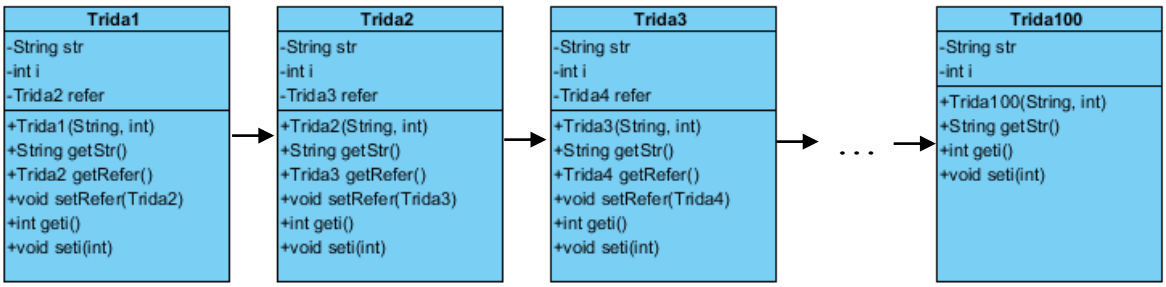
Instalace obou databázových produktů proběhla zcela bez problémů v řádu několika málo minut, vyjma nastavení složky pro cíl instalace byla všude použita defaultní nastavení.

Vytvoření databáze je v DB4O velmi jednoduché a rychlé, stačí specifikovat název a umístění databázového souboru (jediný soubor). Pro vzdálený přístup je pro větší množství databází pod jedním OS používaná jedna služba, naproti tomu pro každou Oracle databázi se vytvoří služba samostatná. K vytvoření Oracle databáze byl použit integrovaný Database Configuration Assistant, který nabízí značné množství konfigurace a doplňků. Velikost paměti RAM pro databázi zde byla nastavena na 7,5 GB a archivace byla vypnuta, jinak opět defaultní nastavení. Nutno podotknout, že následné vytváření a registrace databáze trvalo 19 minut a vytížilo 2 jádra procesoru na 100%.

Instalace nástroje ObjectManager Enterprise pro DB4O do Visual Studia byla pomocí instalátoru bezproblémová. Při instalace doplňku ODT (Oracle Developer Tools) došlo hned k několika problémům. Prvním byl nefunkční odkaz na stažení aktuální verze ODT 11.2.0.3 na stránkách Oracle. V návodu na použití Visual Studia[30] je navíc uveden odkaz na základní ODT (bez klasického instalátoru a bez komponent pro VS2010), přestože dále v návodu jsou využívány součásti, který tento balík neobsahuje. O existenci ODAC with ODT (Oracle Data Access with ODT) zde bohužel nebyla ani zmínka. Instalace ODAC je již možná pomocí integrovaného instalátoru Oracle Universal Installer, nicméně pro .NET 4 existuje v tuto chvíli pouze 32bitová verze.

[30] *http://www.oracle.com/technetwork/articles/dotnet/vs2010-oracle-dev-410461.html*

Po úspěšné instalaci a integraci do VS2010 bylo možné začít pracovat na samotné testovací aplikaci. V první části bylo nutné obě databáze naplnit shodnými daty. Bylo vytvořeno 100 jednoduchých testovacích tříd podle následujícího schématu.



Obrázek 4 Schéma testovacích tříd

Klíčovou je zde vlastnost „refer“, která ukazuje vždy na objekt z následující třídy. Od každé této třídy bylo vytvořeno 100 objektů (v aplikaci nazvaných „testovací set“) s náhodně zvoleným objektem přiřazeným odpovídající třídy přiřazeným do refer, které byly uloženy do objektové databáze a po vytvoření odpovídajících tabulek i do relační databáze. Z programátorského hlediska zde stojí za zmínku, že v Oracle DB 11g není možné v jednom příkazu poslat více SQL příkazů oddělených středníky, což je často řešeno v diskuzích[31].

Po naplnění daty byly vytvořeny testovací metody pro obě databáze. Princip testu spočívá dotazování na hodnoty s různým množstvím propojení („referencí“) v dotazu, tedy v počtu spojení (INNER JOIN) v RDB a hloubkou navracených objektů spolu s vrchním objektem z ODB. Pomocí knihovny Stopwatch[32] je měřena čistá doba provedení dotazu (tedy bez přípravy dotazu a inicializace propojení do databáze).

3.1.3. Naměřená data

Databáze byly opakovaně dotazovány při zvyšujícím se počtu propojení (po pěti). Každý dotaz byl několikrát opakován na různých počátečních objektech/řádcích. Následně byla do databáze přidána další testovací sada. Testované rozmezí je 100 – 10 000 objektů jedné třídy / řádků jedné tabulky), tedy 10 000 až 1 milion objektů / řádků celkem.

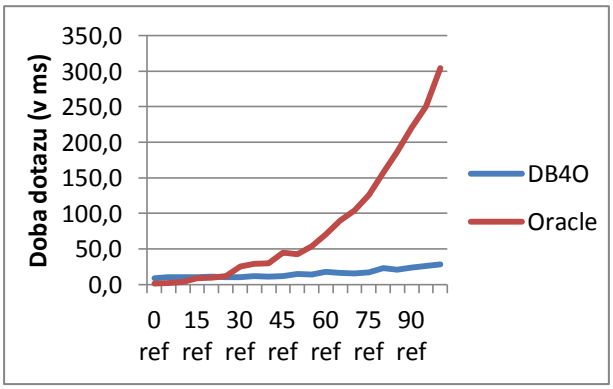
31 Např. http://stackoverflow.com/questions/685850/run-multiple-commands-in-one-executescalar-in-oracle

32 http://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch.aspx

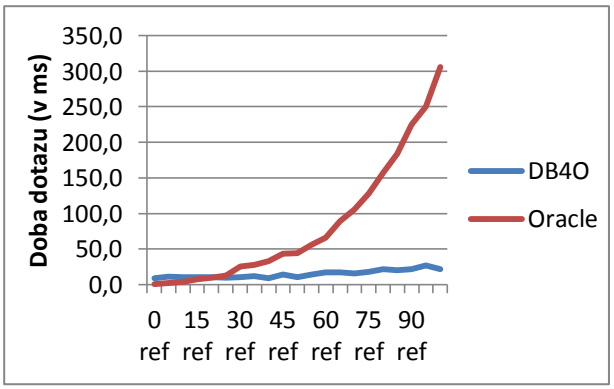
Všechny testy byly provedeny dvakrát, jednou bez indexace (resp. pouze s indexací primárního klíče v databázi Oracle) a podruhé s indexací (primárních a cizích klíčů v tabulkách databáze Oracle a vlastnosti refer v DB4O). Veškeré naměřené hodnoty jsou k dispozici v elektronické příloze práce.

3.1.4. Výsledky

Z naměřených dat vyplývá, že při malé hloubce dotazu jsou výsledná data rychleji získávaná u databáze Oracle. S přibývajícím hloubkou však roste doba vykonání dotazu na této databázi exponenciální řadou. To je dáno složitým procházením logické struktury před samotným čtením fyzických dat. Oproti tomu objektová databáze DB4O díky překladu OID na fyzickou adresu získává data ihned při procházení logické struktury. Výsledkem je pouze lineární závislost na množství objektů.



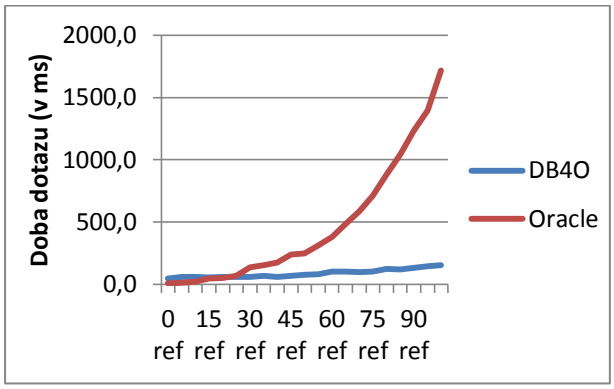
Obrázek 5 Graf závislosti doby vykonání dotazu na jeho hloubce při 1 uložených sad bez indexace.



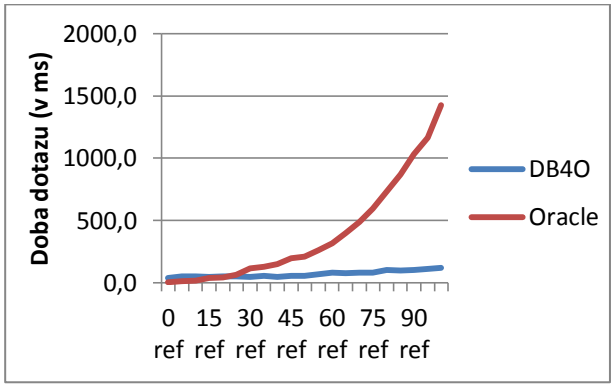
Obrázek 6 Graf závislosti doby vykonání dotazu na jeho hloubce při 1 uložené sadě s indexací.

Jak je vidět z grafů na obrázcích 5 a 6, při menším množství dat v jednotlivých řádcích v relační databázi a menším počtu objektů různých tříd v databázi objektové, jsou naměřené hodnoty obdobné bez ohledu na indexaci.

Naopak při větším množství dat se již indexace projevuje především o relační databáze, jak je vidět z grafů na obrázcích 7 a 8.

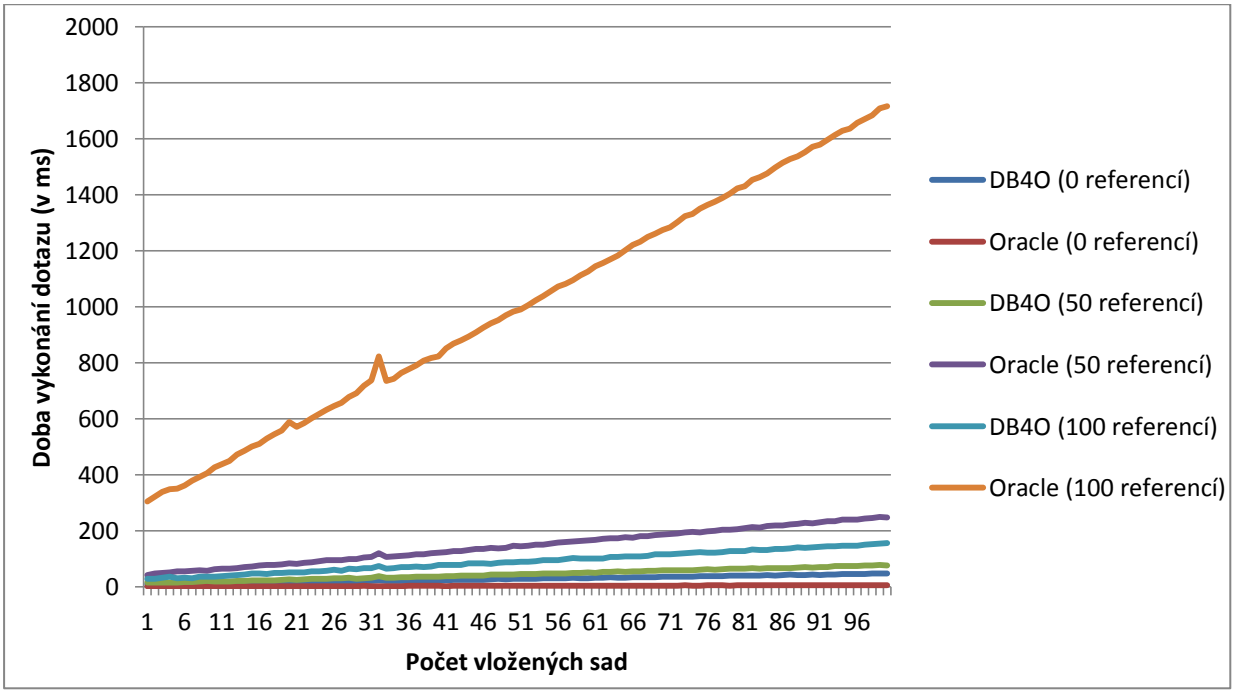


Obrázek 8 Graf závislosti doby vykonání dotazu na jeho hloubce při 100 uložených sadách bez indexace.



Obrázek 7 Graf závislosti doby vykonání dotazu na jeho hloubce při 100 uložených sadách s indexací.

Konkrétní závislost délky dotazu je zobrazena v grafu na obrázku 9, kde je patrna lineární závislost vzhledem k počtu dat u všech databází.

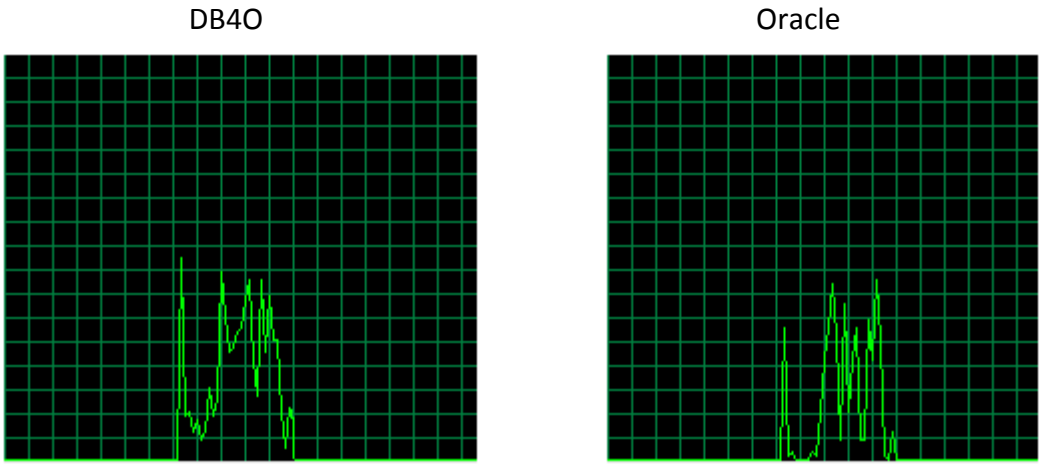


Obrázek 9 Graf závislosti doby vykonání dotazu na množství dat v databázi.

3.2. Náročnost na zdroje počítače

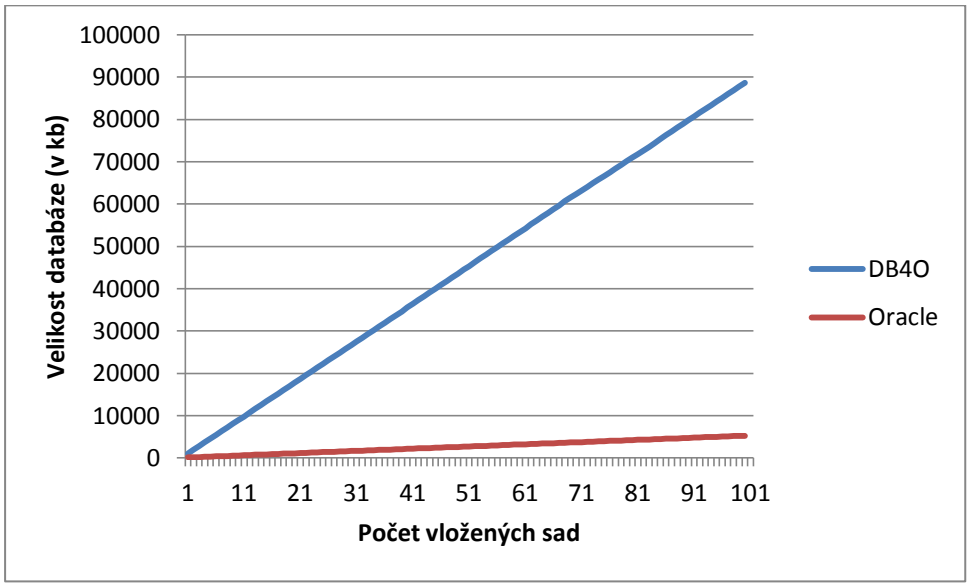
Součástí experimentu bylo i monitorování využití hardwaru (procesoru, RAM, využitého místa na disku). Vzhledem k faktu, že velikost RAM paměti se pro Oracle databázi volí při jejím vytváření, není možné z tohoto hlediska zvolené produkty porovnat, protože zatímco DB4O pracuje s pamětí dynamicky, Oracle si drží stále nastavenou hodnotu.

Využití procesoru bylo monitorováno programem Microsoft Process Explorer. Využití procesoru (resp. jednoho jádra) konkrétním procesem mělo vždy stejný průběh, a to nárůst na téměř polovinu při každém dotazu. Přesné srovnání však rovněž nelze provést, protože využívaných procesů má databáze Oracle hned několik (jeden pro každou databázi a 3 další, z nichž 2 jsou při provádění dotazu na databázi využity).



Obrázek 10 Ukázka zatížení jádra procesoru databázovými procesy.
(5 dotazů při hloubce 100 a při 50 000 uložených objektech)

Data získaná měřením velikosti databází po každém vložení sady ukazují, že obě databáze rostou lineárně vůči množství dat, nicméně DB4O rostla v tomto konkrétním případě průměrně 17 krát rychleji než databáze Oracle. To je dáno především ukládáním struktury objektů a jejich metod.



Obrázek 11 Velikost databází v průběhu experimentu.

4. Závěr

Tato práce se snaží přispět k problematice vhodné volby databázových platforem, kde v současné době v počtu nasazení výrazně převažují relační databáze nad objektovými i přes stále větší popularitu objektového přístupu k programovacím jazykům.

V praktické části práce je zkoumán jeden z nejvýraznějších faktorů, tedy ovlivnění výkonu na základě zvyšujícího se množství zanoření dotazů do databáze. Z naměřených dat lze vyvodit závěr, že od určité „hloubky“ dotazu lze z hlediska výkonu považovat za vhodnější nasazení objektové databáze. Přestože tyto výsledky lze do jisté míry zobecnit, jsou však tyto závěry platné pouze pro v této práci uvedené konkrétní situace a zástupce jednotlivých databázových typů.

Práce rovněž v rámci experimentu přináší údaje o náročnosti zvolených zástupců na velikost databáze, tedy využité místo na disku. Porovnat další nároky na hardware (procesor, operační paměť) se vzhledem k vlastnostem databáze Oracle nepodařilo.

Z teoretické části vyplývají i další rozdíly, které mohou ovlivňovat výkon či práci s databází, jako například počet atributů v jedné n-tici/objektu, zpracování dat na databázové úrovni (procedury RDB / metody objektů), a další, která však nebylo možné v rámci rozsahu této práce hlouběji prozkoumat.

5. Definice pojmů a zkratek

DB	Databáze, pro potřeby této práce ve smyslu počítačové databáze. Dále zle dělit na relační (RDB), objektově orientované (OODB) a případně postrelační databáze.
Databázový typ	Označení typu databáze - tedy relační, objektové, případně hybridní (relačně-objektové) databáze. [33]
DBMS	DataBase Management Systém[34] (systém řízení báze dat) je software zajišťující práci s databází. Dále lze dělit na RDBMS (Relational DataBase Management System) pro RDB a na OODBMS (Object-Oriented Database Management System) pro OODB.
CRUD	Create, Read, Update, Delete (vytváření, čtení, úpravy, mazání) jsou základní databázové operace prováděné pomocí RBMS.
Vztah	Anglicky relationship je z hlediska relačních databází propojení relací pomocí primárního a cizího klíče.
Polymorfizmus	Polymorfními operace je taková operace, kterou lze provádět na objektech z různých tříd.
Datové proměnné	Jednoduché datové typy (string, int, char, bool, apod.).

33 Web Databases: Introduction to Relational & Object-Oriented Databases. BRADLEY, Janette B. AXSWAVE SOFTWARE, Inc.™. AxsWave Software: Information & World Wide Web Consulting Services [online]. 1996 [cit. 2012-03-28]. Dostupné z: http://www.axswave.com/weblibry/relobjdb.htm

34 Database Management System. CHAPPLE, Mike. ABOUT.COM, a part of The New York Times Company. About.com [online]. 2009 [cit. 2012-03-29]. Dostupné z: http://databases.about.com/od/administration/g/dbms.htm

6. Seznam obrázků

7. Seznam tabulek

8. Použité zdroje

LINQ to SQL: .NET Framework 4. Microsoft development network [online]. 2010-2012 [cit. 2012-04-16]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb386976.aspx

Root.cz. Root.cz: Historie relačních databází [online]. 19.10.2001 [cit. 2012-04-01]. Dostupné z: http://www.root.cz/clanky/historie-relacnich-databazi/

FairCom [online]. 2009 [cit. 2012-04-01].

SELECT * FROM SQL History: Who was Edgar "Ted" Codd?. FairCom [online]. 2009 [cit. 2012-04-01]. Dostupné z: http://www.faircom.com/ace/enl_22_s12_t.php

Proceedings of the First International Conference on Deductive and Object-Oriented Databases [online]. 1989, December [cit. 2012-04-14].

Object database management systems [online]. 2005-2012 [cit. 2012-04-03].

Introduction to ODBMS: Short History. Object database management systems [online]. 2005-2012 [cit. 2012-04-03]. Dostupné z: http://www.odbms.org/Introduction/history.aspx

Database Journal: The knowledge center for database professionals [online]. 24.06.2002 [cit. 2012-04-01].

Oracle Documentation [online]. 1996-2003 [cit. 2012-04-07].

Oracle® Database SQL Reference. Oracle Documentation [online]. 1996-2003 [cit. 2012-04-07]. Dostupné z: http://docs.oracle.com/cd/B12037_01/server.101/b10759/toc.htm

ACM [online]. 1970, Number 6 [cit. 2012-04-09].

ComputerWorld. 1985, 21. října.

Root.cz: Historie relačních databází [online]. 19.10.2001 [cit. 2012-04-01].

IBM Archives: Edgar F. Codd. IBM [online]. IBM Research News, 24.04.2003 [cit. 2012-04-04]. Dostupné z: http://www-03.ibm.com/ibm/history/exhibits/builders/builders_codd.html

Web Databases: Introduction to Relational & Object-Oriented Databases. BRADLEY, Janette B. AXSWAVE SOFTWARE, Inc.™. AxsWave Software: Information & World Wide Web Consulting Services [online]. 1996 [cit. 2012-03-29]. Dostupné z: http://www.axswave.com/weblibry/relobjdb.htm

Microsoft development network [online]. 2010-2012 [cit. 2012-04-16].

Database Management System. CHAPPLE, Mike. ABOUT.COM, a part of The New York Times Company. About.com [online]. 2009 [cit. 2012-03-29]. Dostupné z: http://databases.about.com/od/administration/g/dbms.htm

SQLite [online]. 2006 [cit. 2012-03-29].

Most Widely Deployed SQL Database. SQLite [online]. 2006 [cit. 2012-03-29]. Dostupné z: http://www.sqlite.org/mostdeployed.html

INTERSYSTEMS CORPORATION. InterSystems Caché [online]. 1996-2012 [cit. 2012-03-29]. Dostupné z: http://www.intersystems.com/cache/

Stackoverflow [online]. 2009-2010 [cit. 2012-03-29].

Why dont people simply use “Object Database”s?. In: Stackoverflow [online]. 2009-2010 [cit. 2012-03-29]. Dostupné z: http://stackoverflow.com/questions/1612169/why-dont-people-simply-use-object-databases

Stackoverflow [online]. 2009-2011 [cit. 2012-03-29].

Object Oriented vs Relational Databases. In: Stackoverflow [online]. 2009-2011 [cit. 2012-03-29]. Dostupné z: http://stackoverflow.com/questions/800/object-oriented-vs-relational-databases

IBM [online]. IBM Research News, 24.04.2003 [cit. 2012-04-04].

ComputerWorld. 1985, 14. října.

ASNANI. Oracle Database 11g: Hands-On Sql & Pl/sql. New Delhi: PHI Learning Private Limited, 2010. ISBN 978-81-203-4020-6. Dostupné z: http://books.google.cz

ATKINSON, M., F. BANCILHON, D. DEWITT, K. DITTRICH, D. MAIER a S. ZDONIK. The Object-Oriented Database System Manifesto. Proceedings of the First International Conference on Deductive and Object-Oriented Databases [online]. 1989, December [cit. 2012-04-14]. Dostupné z: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/clamen/OODBMS/Manifesto/

BRADLEY, Janette B. AXSWAVE SOFTWARE, Inc.™. AxsWave Software: Information & World Wide Web Consulting Services [online]. 1996 [cit. 2012-03-29].

CODD, E. F. Is Your DBMS Really Relational?. ComputerWorld. 1985, 14. října.

CODD, E. F. A Relational Model of Data for Large Shared Data Banks. ACM [online]. 1970, Number 6 [cit. 2012-04-09]. Dostupné z: http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

CODD, E. F. Does Your DBMS Run By the Rules?. ComputerWorld. 1985, 21. října.

GILFILLAN, Ian. Introduction to Relational Databases. Database Journal: The knowledge center for database professionals [online]. 24.06.2002 [cit. 2012-04-01]. Dostupné z: http://www.databasejournal.com/sqletc/article.php/1469521/Introduction-to-Relational-Databases.htm

HERNANDEZ, Michael J. Nývrh databází. Praha: Grada, 2006. ISBN 80-247-0900-7.

CHAPPLE, Mike. ABOUT.COM, a part of The New York Times Company. About.com [online]. 2009 [cit. 2012-03-29].

KIM, Won. Introduction to Object-Oriented Databases. Cambridge, Massachusetts: The MIT Press, 1990. ISBN 978-0262111249.

POKORNÝ, Jaroslav. Databázové systémy. Praha: Karolinum, 1992. ISBN 80-7066-814-8.

9. Přílohy

1. Aplikace pro měření doby dotazu v závislosti na jeho hloubce. (pouze v elektronické podobě)

2. Hodnoty naměřené při experimentu. (pouze v elektronické podobě)