VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

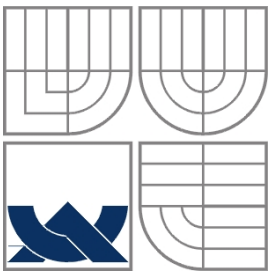
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIMODÁLNÍ DATABÁZE

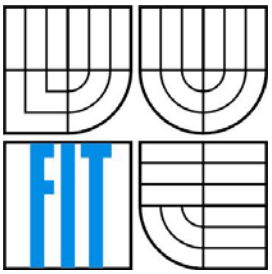
SEMESTRÁLNÍ PROJEKT
SEMESTRAL PROJECT

AUTOR PRÁCE Bc. JIŘÍ KUNCL
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIMODÁLNÍ DATABÁZE
MULTIMODAL DATABASE

SEMESTRÁLNÍ PROJEKT
SEMESTRAL PROJECT

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KUNCL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2007

Abstrakt

Tato práce vznikla v rámci semestrálního projektu jako obecný úvod k samotné diplomové práci věnované tématu multimediálních databází. Práce obsahuje celkový přehled o vývoji databázových systémů, výklad nejdůležitějších modelů pro ukládání dat a shrnutí základních poznatků o multimediálních datech.

Klíčová slova

multimediální, databáze, objektově orientovaný přístup, modely dat, vizuální formáty, zvukové formáty

Abstract

This work was created as general overview for master's thesis about multimodal databases. It contains general summary of history developement of database systems, overview of the most important data models and summary of the facts about multimedial formats.

Keywords

multimedial, database, object oriented, database model, visual format, audio format

Multimodální databáze

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Petra Chmelaře.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Kuncl
4.1.2008

Poděkování

Chtěl bych poděkovat vedoucímu mé práce Ing. Petru Chmelařovi za poskytnutí odborné pomoci při tvorbě této práce.

Obsah

1 Úvod

Databáze se v průběhu minulého století staly pro moderní lidskou společnost nenahraditelnými. S tím, jak výpočetní technika pronikala do dalších a dalších oborů lidské činnosti, získávaly i databáze na významu a důležitosti. Nyní se počítače používají téměř ve všech myslitelných odvětvích a pro nejrůznější účely. Přirozeným důsledkem tohoto procesu bylo neustále rostoucí množství dat, které je třeba uchovávat a spravovat. To mělo za následek prudký rozvoj systémů pro jejich ukládání – tzv. databází.

V dnešní době jsme na databázích zcela závislý, i když většina lidí si to vůbec neuvědomuje. S postupným zaváděním informačních technologií do bankovního sektoru a globalizací ekonomiky se na těchto systémech stal závislý například celý finanční sektor. Databáze obsahují záznamy o našich úsporách, půjčkách, důchodech a mnoho dalších – pro dnešní společnost klíčových – informací.

S postupným zaváděním databázových systémů v různých oborech rostly i požadavky na formát ukládaných dat. Prostý textový formát postupně přestal dostačovat a v dnešní době jsou na databázové systémy v tomto ohledu kladeny stále vyšší a vyšší nároky.

Výsledkem tohoto vývoje byl vznik tzv. multimodálních databází. Tyto databázové systémy jsou schopné uchovávat a dotazovat data nejrůznějších formátů. Od multimediálních dat, přes prostorová až po obecné objekty ve smyslu datových struktur.

Tato práce je pak věnována především databázím multimediálním – tedy systémům pracujícím nad vizuálními a zvukovými daty.

Druhá kapitola se věnuje historii databázových systémů obecně. Mapuje začátky relačního modelu, pronikaní objektového přístupu do databázových systémů a v závěru přibližuje vznik a dosavadní vývoj v oblasti multimediálních databázových systémů.

Třetí kapitola vysvětluje dva nejpoužívanější datové modely, se kterými pracují dnešní databázové systémy – relační model a objektový přístup. U relačního je zmíněn teoretický matematický základ, ze kterého tento model vychází a základní pojmy relačního vzoru. V případě objektového přístupu jsou vysvětleny nejdůležitější pojmy které se v tomto modelu vyskytují. Dále je nastíněn způsob realizace samotného databázového systému pracující s tímto přístupem (perzistence objektů apod.).

Čtvrtá kapitola je již výhradně věnovaná multimediálním databázím. Vysvětluje pojem multimediálních dat a představuje jednotlivé způsoby předávání informace. Dále obsahuje popis nejpoužívanějších formátů dat pro vizuální a zvuková data.

V závěru pak shrnuji dosavadní poznatky a nastiňuji další směřování práce v rámci diplomového projektu.

2 Historie databázových systémů

Za první systémy pro uchovávání většího objemu dat je možné považovány papírové kartotéky. Způsob správy a třídění informací byl v mnohém podobný dnešním databázím. Manipulaci s daty však zajišťovali lidé.

První stroj na zpracovávání dat se objevil v roce 1890 u příležitosti sčítání lidu ve Spojených státech amerických. Jeho tvůrcem se stal Herman Hollerith. Pracoval na bázi děrných štítků, které uchovávaly potřebné informace. Jejich ukládání a třídění probíhalo mechanickou cestou. Tento způsob uchovávání dat byl následně používán několik desítek let.

Další zlom přišel v roce 1935, kdy firma IBM sestrojila pro potřeby americké Správy sociálního zabezpečení první komerční digitální počítač UNIVAC I. Tento stroj však stále pracoval s děrnými štítky jako hlavním médiem pro udržování informací. Štítky však byly postupně nahrazeny magnetickými páskami. UNIVAC se stal poměrně úspěšný projektem a například americký Pentagon disponoval v roce 1959 již více než 200 těmito stroji [ZAK].

Postupně rostl tlak na sjednocení přístupu k uchovávání dat a vytvoření jednotného jazyka pro jejich dotazování a správu. V roce 1959 se konala konference zástupců firem, uživatelů a amerického ministerstva obrany s požadavkem na standardizaci databázového jazyka. Následně o rok později vzniklo seskupení CODASYL, které mělo za cíl standardizaci software aplikací. Výsledkem těchto snah byl jazyk COBOL, který se na dalších několik desítek let stal nejpoužívanějším jazykem pro hromadné zpracování dat.

V tomto období také začaly magnetické disky postupně nahrazovat dosud používané magnetické pásky. Tento krok byl důležitý pro odstranění sériového přístupu k datům, což následně otevřelo cestu k vytvoření použitelnějšího modelu pro databázové systémy.

V průběhu 60. let pak začaly vznikat první specifikace pro jazyky určené přímo pro práci s databázemi. V rámci seskupení CODASYL byla vytvořená samostatná skupina Database Task Group, která tento směr vývoje dále rozvíjela. V tomto období byly představeny i první produkty, které nesly znaky určitého DB managementu.

2.1 Vznik relačního modelu

Většina databázových systémů z tohoto období pracovala s tzv. síťový datový modele nebo hierarchický datový modele. Tyto se však dnes už kromě sálových počítačů takřka nepoužívají.

Byly nahrazeny tzv. relačním modelem, který v roce 1970 představil ve svém článku „A Relational Model of Data for Large Shared Data Banks“ zaměstnanec firmy IBM Ted Codd. Tento model se postupně stal nejrozšířenějším systémem pro uchovávání a dotazování dat. Používá ho velká většina dnešních databázových systémů.

Z počátku bylo však na tento model pohlíženo spíše jako na jakýsi teoretický model, který nebude v praxi použitelný. Dokonce ani firma IBM neočekávala, že bude relační model nasazený a nechystala žádnou jeho implementaci. Trvalo to dalších několik let diskuzí a ověřování teoretického základu, než se relační model prosadil a oba starší zatlačil do pozadí.

IBM začalo vyvíjet relační databázový systém v rámci projektu System-R. Druhým souběžně probíhajícím projektem byl projekt Ingres, který byl vyvíjen kalifornskou univerzitou v Berkeley.

Ke konci roku 1976 byla v časopise IBM Journal of R&D představena verze dotazovacího jazyka SEQUEL2, který vycházel z původního SEQUEL (Structured English Query Language). SEQUEL2 byl posléze přejmenován na SQL a stal se základem SQL tak, jak ho známe dnes. Konkurenční projekt Intres nepoužíval SQL, ale velmi podobný vlastní jazyk QUEL.

V roce 1980 byla dostupná první komerční verze systému Ingres. Lidé kolem projektu postupně zakládali vlastní společnosti a dále se věnovali vývoji databázové technologie v komerční sféře (např. Robert Epstein se stal spoluzakladatelem Sybase, Paula Hawthorn se podílel na založení Illustra Information Technologies Incorporated a další).

V roce 1980 pak byla uvedena první databáze založená na jazyku SQL. Stala se jí databáze od firmy Oracle, která byla z části inspirována projektem Systém-R od IBM. IBM následně přišla se svým vlastním komerčním produktem založeným na SQL a relačním modelu – systémem DB2.

V 80. létech pak pokračovalo další rozvoj SQL a databáze založené na relačním modelu se postupně stávají nejpoužívanějším systémem. Souběžně s tímto vývojem se však začaly objevovat i nové přístupy k ukládání dat v databázi. Např. databáze objektově orientované, prostorové, multimediální a další. Tyto souhrnně označujeme jako tzv. postrelační systémy.

2.2 Objektové databáze

Rozmach objektově orientovaného programování v posledních 20 letech měl za následek i vznik nového směru v databázových systémech. Vzrůstající potřeba persistentního uchovávání objektů inicializovala začátek vývoje databází schopné tyto datové typy uchovávat – objektové orientovaných databází.

Původní projekt Ingres, který byl založen na relačním modelu, byl v roce 1985 přeměněn na projekt Postgres, jehož cílem bylo vytvořit objektové-relační databázi. Byl opět vyvíjen na univerzitě v Berkeley. V roce 1994 byl projekt ukončen. V roce 1995 však na základě kódu z tohoto projektu vzniká pod názvem Postgres95 projekt nový a je pokračováno v jeho vývoji. O rok později je převeden do open-source podoby a opět přejmenován, tentokrát na PostgreSQL. Pod tímto názvem je vyvíjen dodnes. Systém však stále spadá do skupiny relačních, obsahuje však již některé objektové rysy.

Postupně však vzniklo několik databázových systémů, které už lze považovat za objektové (např. Caché od firmy InterSystems Corporation, systém O_2 aj.).

2.3 Multimediální databáze

První pokusy o uchovávání a správu většího množství multimediálních dat založené na počítačích byly zaznamenány už v průběhu druhé poloviny minulého století. Většinou se však nejednalo o pravé databázové systémy tak, jak je chápeme dnes. Data byla spravována i dotazována operačním systémem a nikoliv systémem řízení báze dat. Tyto systémy označujeme pojmem ad-hoc. Sloužily spíše jako datové sklady než plnohodnotné databáze .

Multimediální data nejsou z podstaty vhodná pro uchovávání v relačních databázích (tabulkách, viz níže). Vetší rozvoj proto zaznamenávají až s nástupem postrelačních systémů v devadesátých letech.

Toto období označujeme jako tzv. První vlnu. Zde již šlo o plnohodnotné a komplexní systémy. Poskytovaly podporu pro různé formáty dat a obsahovaly mechanismy pro dotazování, získávání, vkládání a aktualizaci uložených dat (např. Jasmíne, Media Way aj.).

Druhá vlna pak začíná v druhé polovině devadesátých let. Charakterovým rysem této skupiny systémů je masivní nasazení objektového přístupu. Komplexní objekty konečně umožňují pracovat s mnoha různými formáty dat. Objektově orientovaný přístup poskytuje prostředky pro definování vhodných datových struktur, které umožňují uchovávání nových mediálních formátů, jako jsou video soubory apod. Díky tomuto přístupu je nově možné definovat i operace určené přímo pro multimediální formáty. Z toho je vidět, jak spolu multimediální a objektové databáze úzce souvisí. Mezi typické zástupce této vlny patří například systém Informix, Oracle 10g, IBM DB2 a další.

Třetí vlnu pak představují právě běžící projekty nebo projekty nedávno dokončené. Hlavní důraz je v současnosti kladen na sémantický obsah a popis dat pomocí metadat. Většina dnešních projektů je založena na novém formátu pro popisování mediálních dat – MPEG-7 (tento formát bude představen v dalších kapitolách už v rámci diplomové práce). Za představitele třetí generace je považován například projekt MARS (Multimedia Analysis and Retrieval System).

3 Modely pro ukládání dat

Tato kapitola obsahuje stručný popis dvou dnes nejpoužívanějších modelů pro databázové systémy – relačního a objektového.

3.1 Relační model dat

Relační model dat se objevil v roce 1970 (viz kapitola 2.1). Jedním z hlavních důvodů vzniku tohoto modelu byla snaha o oddělení implementace logické struktury programu a dat, nad kterými program pracuje. Do té doby používaný síťový a hierarchický model toto neumožňovaly.

Multimediální databáze sice s relačním sch0matem přímo nepracují, avšak tento model je dnes v databázových technologiích stále široce uplatňovaný a i novější model objektový z něj do jisté míry vychází (některé dnešní systémy se dokonce označují relačně-objektové, protože vycházejí z obou modelů). Proto v následující kapitole zmíním základy této teorie a nastíním způsob ukládání dat v relačních databázích.

Relační model má mezi ostatními systémy výjimečné postavení. Jako jediný stojí na formálním základu, který byl představen ještě před samotnou první implementací systému na bázi relací.

3.1.1 Formální definice relace

Následuje formální definice databázové relace. Definice byla převzata z [ZEN].

Nechť D_1,D_2, ... D_n jsou množiny atomických hodnot označované jako domény. Relace (databázová) na doménách D_1, D_2, ... D_n je dvojice $R = (R, R^*)$, kde $R = R(A_1 : D_1, A_2 : D_2, \ldots A_n : D_n)$ je schéma relace, kde $A_i (A_i \neq A_j \text{ pro } i \neq j)$ značí jméno atributu definovaného na doméně D_i a $R^* \subseteq D_1 \times D_2 \times \ldots \times D_n$ je tělo relace. Počet atributů n relace se označuje stupeň (řád) relace, kardinalita těla relace $m = |R^*|$ se označuje kardinalita relace.

3.1.2 Struktura databáze

Výše uvedená formální definice vymezuje pojem databázové relace. Tato relace je pak ve výsledné databázi reprezentována tabulkou. Sloupce, které tvoří tabulku se nazývají atributy. Množina všech možných hodnot, kterých může konkrétní atribut nabývat, se nazývá doména atributu. Trojice název relace (tabulky), počet a názvy jednotlivých atributů (sloupců) a jejich domény (množina přípustných hodnot) pak tvoří tak zvané schéma relace, které definuje strukturu tabulky.

Samotné tělo relace pak tvoří podmnožina kartézského součinu mezi množinami domén jednotlivých atributů. Tyto součiny označujeme jako n-tice a představují řádky dané tabulky (tělo

relace). Tyto n-tice (řádky) sdružují skupiny hodnot, které k sobě na základě nějakého vztahu patří (např. charakterizují zákazníka banky skupinou atributů rodné číslo, jméno, číslo účtu, adresa ... apod.).

Veškerá data v relačních databázích jsou na logické úrovni uchovávány v tabulkách. I výsledky dotazů nad těmito daty opět tvoří tabulky.

Hlavní výhodu relačního modelu představuje především jednoduchost. V podstatě omezuje strukturu veškerých dat v databází na kolekci tabulek, které tvoří pouze hodnoty z několika mála předem předdefinovaných množin. Tato vlastnost však zároveň znamená i jeden z největších problémů relačních databází. Tou je neschopnost efektivně pracovat se složitějšími datovými typy, jako jsou geografická nebo multimediální data.

S relačními databázemi úzce souvisí další pojmy, jako normalizace, primární a cizí klíče, integritní omezení a další. Vzhledem k tomu, že tato práce není věnována relačním databázím, nebudu se těmto termínům dále věnovat. Jejich definici a výklad lze nalézt v publikacích specializovaných na relační databáze.

3.2 Objektový model dat

Relační model klade na strukturu uchovávaných data přísná omezení. Ne všechna data se však dají transformovat na množinu vzájemně propojených tabulek definovaných nad několika málo základními typy. Z toho vyplývá nutnost určitých kompromisů, pokud po databázi požadujeme, aby pracovala se složitějšími datovým typy. Tyto problémy časem vyústily ve vznik databázových systémů založených na objektovém modelu dat.

Objektové orientované databáze pracující s objektovým modelem dat se začaly objevovat na přelomu 80. a 90. let minulého století (viz kapitola 2.2). Hlavním rysem tohoto modelu je snaha o zachování objektového pohledu na data a jejich následné perzistentního uložení v databázi.

Celý přistup tohoto modelu dat je založen na známých zásadách objektového návrhu a objektově orientovaného programování. V případě objektového pohledu pro databázové použití se však objevují některé specifické problémy, které databázové systémy používající tento model musí řešit. Jde především o techniky zaručení persistence pro uchovávaná data, transakční přístup, representace vztahů apod.

Objektový model dat nestojí - na rozdíl od relačního – na žádném formálním základě. Jde spíše o kolekci obecně přijatých pravidel, které vycházejí z podoby skutečného světa a nutnosti reprezentovat objekty z tohoto světa v softwarových aplikacích. Tyto pravidla pak tvoří jakousi koncepci objektového přístupu. Objektové orientované programování navíc představuje i určitý způsob uvažovaní, který je nutné aplikovat už ve fázi prvotního návrhu aplikace.

Následuje vysvětlení základních pojmů a technik používané v objektovém modelu dat.

3.2.1 Objekt

Už z názvu objektového modelu dat plyne, že základ tohoto modelu tvoří objekt. Jde o element, který vznikne dekompozicí informace z reálného světa (např. okno v grafické uživatelském rozhraní, tlačítko apod.) [DOS]. Každý objekt má svou jednoznačnou identifikaci. Tato identifikace je zaručena tzv. identifikátorem (objekt identifier – OID). Každý objekt disponuje tímto vlastním OID a je tedy rozlišitelný od ostatních objektů (třebaže datově úplně stejných). Tento identifikátor je většinou generován systémem a pro programátora i uživatele zůstává skrytý.

Objekt může obsahovat atributy, které popisují jeho aktuální stav. Může jít o různé jednoduché proměnné, ale i o jiné objekty apod. Tento stav si objekt pamatuje.

Objekty také mohou poskytovat svému okolí různé služby. Tyto služby nazýváme metody a pomocí nich objekt komunikuje s okolím. Základní metody představují tzv. konstruktor a destruktor. Ty jsou volány při vytvoření, resp. rušení objektu.

Objekty mohou využívat služeb jiných objektů. Důležité však je, že mají přístup pouze k rozhraní dané metody (služby). Samotné implementace metody zůstává pro volající objekt skryta. Tento způsob komunikace je pro objektově orientované programování typický a významně zvyšuje míru abstrakce. Objekt představuje jakousi „černou skříňku“, do které není vidět, a která s okolí komunikuje pouze pomocí rozhraní metod, které poskytuje.

3.2.2 Zasílání zpráv

Objekty spolu mohou vzájemně komunikovat pomocí zasílání zpráv. Tyto zprávy představují volání konkrétní metody cílového objektu objektem volajícím. Jak už bylo řečeno výše, volající objekt neví nic o implementaci dané metody. Zná pouze rozhraní metody (to znamená množinu zpráv, které může cílový objekt přijmout) a tzv. komunikační protokol objektu. Ten představuje jakousi sadu pravidel, podle kterých je nutné postupovat, chceme-li poslat cílovému objektu zprávu. Pokud tak učiníme, cílový objekt nám vrátí výstupní hodnotu volané metody. Ta je považována za poslání zprávy od volaného objektu.

K datové složce cílového objektu většinou není povoleno přistupovat „z venku“ (tzn. z jiného objektu), ale pouze pomocí metod, které cílový objekt nabízí. Tento přístup skrývání implementace a dat se nazývá zapouzdření a je jednou z hlavních tezí objektově orientovaného programování.

3.2.3 Třída objektu

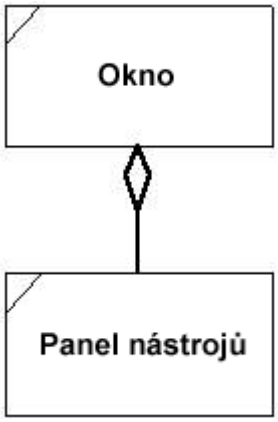
Jedna z definic třídy zni: „Třída objektů je abstrakce množiny podobných objektů“ [DOS]. Třída určuje datovou složku objektu a metody, které lze nad objektem provádět. Každý objekt je instancí nějaké třídy. Každá třída může mít teoreticky neomezený počet instancí.

3.2.4 Vztahy mezi třídami

Mezi jednotlivými třídami existují různé vztahy. Třída může být ve vztahu k několika dalším třídám, ale může existovat i samostatně. Tyto vztahy se většinou zobrazují tzv. diagramy tříd. V něm vystupují třídy jako uzly grafu a vztahy jako hrany, které tyto uzly spojují. Existují tři základní druhy vztahů:

3.2.4.1 Agregace

Vztah typu agregace vyjadřuje, že instance dané třídy v sobě obsahuje instanci třídy jiné. Například objekt `Okno`, představující okno v grafickém uživatelském rozhraní, obsahuje objekt `Panel`, který modeluje panel nástrojů patřící k tomuto oknu. Tento vztah přibližuje následující obrázek.



Obrázek 1 - vztah agregace

3.2.4.2 Asociace

Dva objekty se nachází ve vztahu asociace právě když o sobě navzájem vědí a mohou si posílat zprávy. To znamená, že znají rozhraní svých metod a příslušné komunikační protokoly k nim. Jde o poněkud obecnější vztah než v případě agregace protože objekty spolu nejsou tak těsně svázány. Jako příklad může posloužit vztah mezi instancí třídy `Knihovník` modelujícím správce seznamu dostupných knih v knihovně a třídy `Kniha` představujícím daný titul. Instance není vhodné navzájem spojit pomocí agregace, protože `Knihovník` daný titul přímo neobsahuje. Následující obrázek modeluje tento vztah mezi zmíněnými třídami.

Obrázek 2 - vztah asociace

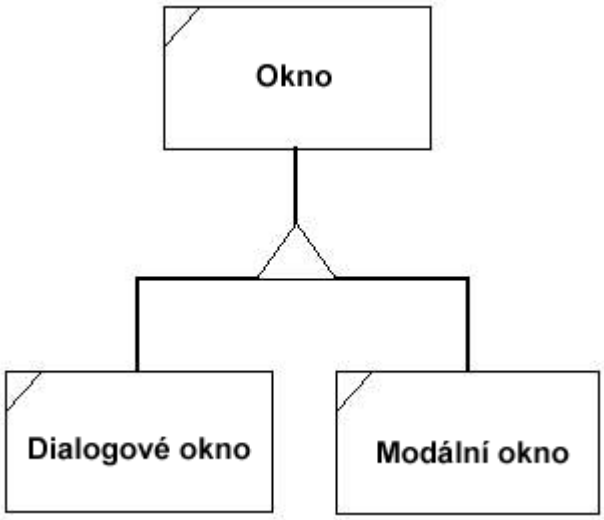
3.2.4.3 Dědičnost

Dvě třídy jsou ve vztahu dědičnosti (třída B dědí od třídy A), pokud děděná třída je podmnožinou třídy, ze které je děděno (třída B je podmnožinou třídy A). Tento vztah se někdy také označuje jako tzv. specializace (dědící třída B je speciálním případem původní zděděné třídy A).

Děděná třída přidá k původnímu schématu (popisu dat a metod, které originální třída nabízí) většinou některé nové specifické pouze pro ni. Může také upravit zděděné metody (viz kapitola 3.2.4.4).

Rozlišujeme dva typy dědičnosti – jednoduchou a vícenásobnou. V případě jednoduché je děděno pouze z jedné nadřazené třídy, naopak u vícenásobné je možnost použít nadřazených tříd více (u některých programovacích jazyků však vícenásobná dědičnost není povolena).

Následující obrázek modeluje třídy Modální okno a Dialogové okno, které dědí vlastnosti od nadřazené třídy Okno.



Obrázek 3 – dědičnost (specializace) – převzato z [DOS]

3.2.4.4 Polymorfismus

Polymorfismus (česky vícetvarost) představuje jednu z velmi důležitých vlastností v objektově orientovaném programování. V kombinaci se systémem dědičnosti poskytuje mocný nástroj, jak zvýšit abstrakci programu a v případě rozšíření už napsaného kódu značně zjednodušuje jeho úpravy a modifikace.

Polymorfismus je vlastnost týkající se metod objektů, které většinou vzájemně vystupují ve vztahu dědičnosti (to znamená specializace, resp. generalizace v opačném směru). Rozlišujeme dva základní případy polymorfismu.

V prvním případě se u jednoho objektu nacházejí dvě metody, které mají stejný název, ale jiný počet parametrů (případně jiný typ). V druhém případě se pak u dvou odlišných objektů různých tříd (z nichž jedna je většinou ve vztahu dědičnosti k druhé) nacházejí stejně pojmenované metody se stejným počtem i typem parametrů, ale implementace těchto metod se u obou tříd liší.

S polymorfismem pak úzce souvisí pojem pozdní vazba. Tímto termínem se označuje situace, kdy aplikace v době překladu neví, kterou z implementací polymorfní metody použije. Toto je rozhodnuto až při běhu programu na základě třídy objektu, nad kterým je metoda volána.

3.2.5 Perzistence dat v rámci objektového modelu

Zachování perzistence dat představuje jeden z nejdůležitějších úkolů jakéhokoliv databázového systémů napříč modely dat. Perzistentní data jsou taková data, jejichž životnost překračuje běh aplikačního programu i vypnutí počítače, na kterém je databázový systém provozován. Tyto data musí být i v případě nečekaného restartování systému dostupná a nepoškozená.

Běžné programovací jazyky většinou disponují širokými možnostmi jak spravovat data potřebná po dobu vykonávání programu (tyto data označujeme jako tzv. transientní). Nemají však žádné nástroje jak uchovávat data po ukončení běhu programu. Většinou spoléhají na služby operačního systému a data uchovávají ve formě běžného souboru uloženého na některém z dostupných paměťových médií. Tento přistup je však pro databázové systémy nevhodný.

Kromě výše definovaného požadavku na zachování persistence dat musí databázový systém splňovat další dva základní předpoklady. (viz [SVE]). Perzistentní data i data v operační paměti počítače musí mít minimálně na logické úrovni stejnou strukturu. A dále libovolná data bez ohledu na typ musí být možné prohlásit za perzistentní i transientní.

Při splnění všech těchto požadavků databázovým systémem mluvíme o tak zvané ortogonální perzistenci dat. Tento model persistence splňuje většinu nároků, které jsou na objektově orientované databázové systémy kladeny (uchovávání dat ve stejném formátu jako v paměti, transparentní ukládání a otevírání, postupné načítání dat po menších částech (on-demand) atd. Z toho vyplývá, že tento systém je v databázích založených na objektovém modelu dat široce používán.

S otázkou perzistence úzce souvisí i způsoby ukládání aplikačního kódu dat. Obecně vzato existují čtyři základní přístupy k dané problematice. Prvním z nich je metoda používaná v klasický aplikacích, kdy jsou data i samotný kód programu uloženy nezávisle v systému souboru dat a spravovány příslušným operačním systémem.

Dále se nabízí přistup, který je většinou uplatňován v dnešních databázových systémem, kdy jsou perzistentní data spravována samotným databázovým systémem a aplikační program se nachází vně systému. Tento přístup je v současnosti široce uplatňován a tvoří základ pro vícevrstvé architektury (model klient / server)

Dalším, už méně aplikovaným přístupem, je model, kdy jsou perzistentní data i kód aplikačního programu spravována samotným databázovým systémem. V tom případě pak mluvíme o tzv. ortogonálně perzistentním jazyku.

Tento přístup lze dále rozšířit o modifikaci, kdy databázový systém perzistentně uchovává pouze třídy, které již byly v průběhu běhu programu instanciované některými objekty a dále pak samotný kód. Tento přístup asi nejlépe splňuje požadavky kladené na objektově orientované databázové systémy.

3.2.6 Volba ukládaných dat

V průběhu programu může vznikat velké množství pomocných objektů, které neslouží jako nositelé perzistentních dat, ale poskytují podporu pro běh celého systému (například provádějí některé výpočty, pomocné operace apod.). Z hlediska efektivnosti by bylo značně nevhodné uchovávat všechny tyto instance v perzistentní podobě přímo v databázovém systému. Je zde nutná určitá selekce, kdy systém (resp. programátor) určí, které objekty je nutné perzistentně uchovávat a které ne. Existuje několik přístupů k řešení této problematiky. Následující stručný přehled byl s drobnými úpravami převzatý z [SVE].

První metoda vychází z přímé definice persistence třídy už při jejím vytvoření. Pokud je tato nově definovaná třída označena jako perzistentní, všechny objekty, které jsou instancemi této třídy, pak současně také získávají perzistentní povahu a budou uloženy v databázi.

Další způsob funguje na podobném principu s rozdílem, že při vytvoření nové třídy zároveň vzniká i tzv. stínová třída (shadow class). Originální třída zůstává perzistentní a její stínová kopie naopak transientní. V případe, že je objekt instancí perzistentní verze, je uložen v databázi, pokud je naopak instanciován od stínové kopie, zůstává zachován pouze v operační paměti.

Další přístup využívá definici tzv. perzistentní kořenové třídy. Všechny třídy, které jsou v hierarchii dědičnosti (viz. kapitola 3.2.4.3) potomkem této kořenové třídy, zůstávají perzistentní (resp. objekty, které jsou instancemi těchto tříd).

V metodě tzv. perzistence specifikované při vytváření objektu není persistence definovaná na úrovni tříd, ale až na úrovni jednotlivých instancí (objektů). Při vytvoření nového objektu je určeno, zda bude persistentní či nikoliv (např. v operátoru `new`).

Další přístup vychází z klasických aplikací, kdy jsou data ukládána ve formě souborů. Objekt je určen jako persistentní explicitně některou ze svých metod. V tomto případě je tedy řízení ukládání objektů ponecháno plně na programátorovi, který sám označuje objekty k trvalému uchování.

Další mechanismus pro určení trvanlivosti objektu vychází z myšlenky tzv. persistencích kořenů. Tyto kořeny mají obvykle podobu kontejnerů a je možné do nich vkládat objekty. Ty jsou následně označeny jako persistentní. Mezi kontejnerem a objektem je tedy vytvořena jakási vazba, na základě které je následně rozhodnuto o uložení či neuložení daného objektu. Tato vazba může být i nepřímá.

Poslední technika pro určování perzistence objektů se nazývá pojmenované objekty jako perzistentní kořeny. Jde o podobný princip jako v předchozím případě, navíc však rozšířený o volitelnost perzistentních kořenů. Ty se považují za trvalé až v případě pojmenování objektů. Pojmenováním zde rozumíme přiřazení jména konkrétnímu objektu na úrovni schématu databáze. Za perzistentní se pak považují, podobně jako v předchozím případě, objekty, které mají vazbu na některý perzistentní kořen.

3.2.7 Rušení objektů

U jakkoliv velké kapacity úložného prostoru musí s přibývajícími daty dřív nebo později dojít k situaci, kdy už pro uložení nových informací není dostatek místa. Situaci lze řešit buď zvýšením úložného prostoru systému, nebo smazáním nepotřebných dat uložených v databázi. V případě objektové orientované databáze tedy vlastně odstranit již nepotřebné objekty. Otázkou zůstává, jak tyto nepotřebné objekty najít a označit. V případě odstranění objektu, který se stále ještě nachází ve vztahu k objektům jiným můžeme způsobit vážné problémy s konzistencí dat.

V podstatě existují dva základní přístupy, jak výše nastíněný problém řešit. Prvním z nich je přenesení zodpovědnosti na programátora, který je tak odpovědný za rušení již nepotřebných objektů. Tento způsob přináší výhodu v podobě přesného označení již nepotřebných dat. Na druhou stranu však vzniká velké riziko ztráty konzistence dat, pokud programátor označí stále potřebný objekt jako zbytečný a ten je následně smazán.

V případě, že je daný objekt vázán ve vztazích k jiným objektům, je třeba specifikovat, jak s těmito vazbami naložit. Je možné tyto vztahy zrušit a objekt následně odstranit, nebo odstranit objekt a zároveň všechny objekty, které jsou s ním svázány, popřípadě může být celá operace mazání prohlášena za sémanticky nepřípustnou.

Druhý způsob jak zajistit odstraňování nepotřebných dat představuje technika zvaná garbage collecting. V tomto případě je zodpovědnost přenesena z programátora na samotný systém, který sám označuje data určená ke zrušení. Pro učení nepotřebných objektů existuje několik základních technik.

První z nich je označována jako tak zvaný reference counting (počítání referencí). Tento systém pro každý objekt v databázi udržuje záznam o počtu odkazů, který se na tento objekt vyskytují. V případě, že u některého z objektů klesne počet referencí na nulu, tak je objekt odstraněn.

Výhoda tohoto systému spočívá zejména v tom, že objekty jsou označeny jako zbytečné prakticky okamžitě po té, co dojde ke ztrátě poslední reference na objekt. Další výhodou je také vcelku jednoduchá implementace tohoto algoritmu.

Na druhé straně však v sobě počítání referencí skrývá i některé nevýhody. První z nich představuje množství systémových prostředků, které tento systém vyžaduje. Zejména aktualizace referencí v průběhu vykonávání je systémově neúměrně náročná. Další z nevýhod pak spočívá v neschopnosti odhalit uzavřené smyčky objektů, které udržují reference sami na sebe, ale v podstatě už nejsou dosažitelné. Základní algoritmus (tzv. naivní přístup) musí být tedy rozšířen o detekci těchto smyček.

Počítání referencí byla první implementace garbage collectoru vůbec. V dnešních systémech se však už příliš nepoužívá a místo ní jsou uplatňovány přístupy modernější, pracující na odlišném principu.

Jeden z nich představuje skupina algoritmů označovaných jako sledovací (tracing). Ty jsou dnes velice oblíbené. Pracují na principu procházení všech dostupných objektů z kořenové množiny programu. V průběhu provádění je program pozastaven, sledovací algoritmus začne procházet všechny objekty – začíná u kořenových a postupně se po referencích přesouvá na další a označuje je jako navštívené. Po skončení této fáze následuje rušení všech objektů, které nebyly označeny jako navštívené a program je opět spuštěn. Jedním z nejznámějších představitelů této skupiny algoritmů je Mark & Sweep, který je používán například platformou .NET.

Kromě zmíněných dvou přístupů existují ještě další způsoby, jak řešit problem garbage collectingu (např. generační algoritmus). Jejich výklad však už přesahuje rámec této práce. Informace k nim je možné dohledat například na [WIKI2].

4 Multimediální databáze

Obdobím prudkého rozvojem multimédií v oblasti výpočetní techniky, které zaznamenáváme od 90. let minulého století mělo za následek prvních databázových systémů, které byl přímo určený pro uchovávání, správu a dotazování multimediálních dat (viz. kapitola 2.3).

V posledních letech však zažíváme překotný vývoj v oblasti zpracování multimediálních dat pomocí výpočetní techniky. Tento trend byl umožněn zejména stále se zvyšujícím výkonem počítačů na jedné straně a jejich klesající cenou na straně druhé. I relativně velmi levné počítače tak nyní mají schopnost v reálném čase zpracovávat velmi složitá data, kterými multimediální formáty bezesporu jsou. Mnoho nejrůznějších odvětví a oborů lidské činnosti (medicína, obchodní služby, průmysl a další) proto začalo využívat výpočetní techniku pro práci s těmito, dnes především audiovizuálními daty. Tento vývoj měl přirozeně za následek velmi rychlý růst množství digitalizovaných multimediálních dat, které bylo potřeba uchovávat a navíc s nimi dále pracovat.

Zpočátku byly multimediální soubory ukládány vně databáze a samotný systém řízení báze dat nedisponoval žádnými prostředky, jak s těmito daty přímo manipulovat nebo nad nimi provádět dotazy. S rostoucím množství takto uložených informací však postupně vznikaly stále větší problémy s efektivitou a rychlostí při práci s těmito daty.

Tento vývoj měl za následek vznik databázových systému, které byly nově obohaceny o schopnost přímo pracovat s multimediálními daty. Toto rozšíření je většinou realizováno ve formě doplňku k původnímu systému. Multimediální databáze musí obsahovat podobné nástroje a prostředky pro práci s daty, jako v případě tradičních systémů. Těmito nástroji rozumíme především možnosti pro definici multimediálních formátů dat, jejich modelování a pak všechny schopnosti pro manipulaci s daty, které známe z tradičních databázových systémů (přidávání, mazání, řízení přístupu apod.). Velmi vysoké požadavky jsou kladeny také na vyhledávání podle obsahu, které je v případě multimediálních dat mnohem náročnější než u klasických, strukturovaných, většinou textových informací.

4.1 Multimediální data

Pojem multimediální vznikl spojením latinských slov multum a medium. Jeho význam lze intuitivně chápat jako „médium, které využívá více významových forem“ [WIKI3]. Český ekvivalent by mohl znít například mnohazpůsobový. Termín se pak často používá ke zdůraznění více informačních kanálů, které médium používá. Typicky jde o kombinaci textových, zvukových, obrazových a jiných dat. Toto dělení vychází ze způsobu, kterým je konkrétní informace předávána. Jednotlivé informační kanály lze mezi sebou kombinovat a pak mluvíme o tzv. audiovizuálních datech. Tyto hlavní způsoby předávání informace lze ještě dále dělit podle významu konkrétních dat, které jsou médiem

předávána. U obrazových dat můžeme například použít dělení na 2D (plošné), 3D (prostorové) a 4D (s časovým rozměrem, typicky video) informace.

V případě multimédií ve výpočetní technice se postupem času objevilo velké množství formátů, které slouží k uchovávání multimediálních dat. Typicky každý slouží pro jeden typ informace. Některé formáty však umožňují kombinaci i více forem dohromady.

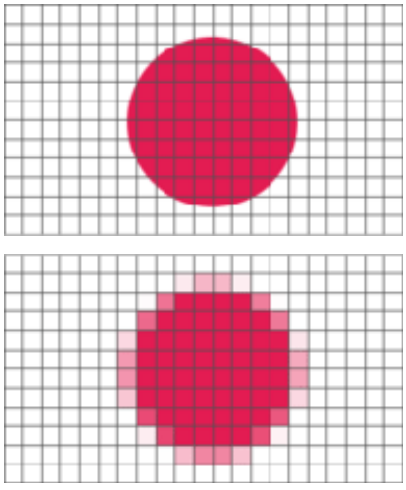
Vzhledem k velice silné redundanci (opakování) v multimediálních datech a jejich prostorové náročnosti také rychle začaly vznikat algoritmy pro odstranění tohoto problému s cílem úspory místa nutného pro uložení konkrétní multimediální informace. V tomto významu pak mluvíme o tzv. kompresi dat.

Většinou jde o kompresi ztrátovou – tedy část informací z původního média je v průběhu převodu ztraceno. Vzhledem ke zmíněné vysoké redundanci informací v případě multimediálních dat to však většinou nepředstavuje vážný problém a při volbě rozumného způsobu komprimace je původní informace srozumitelná i po komprimaci. Velká část formátů pro uložení multimediálních dat používaná v dnešní době s nějakou formou komprese pracuje.

4.1.1 Obrazová data

V případě statických vizuálních dat rozlišujeme dva základní způsoby uložení obrázku – uložení pomocí rastru a uložení pomocí vektoru.

V prvním případě je celý obrázek rozdělen na pixely pomocí mřížky, tzv. rastru. Každý pixel je poté popsán pomocí své barvy (například modelem RGB) a jasové složky. Rastrové uložení obrázků způsobuje velmi silnou redundanci dat. Je proto vhodné použít některý z kompresních algoritmů. Hlavní výhodou tohoto způsobu ukládání vizuálních dat spočívá v jednoduchém vytvoření cílového obrázku. Naopak velkou nevýhodu představuje citelná ztráta kvality při přibližování (zoom). Následující obrázek ukazuje postup převodu geometrického obrazce do rastru.



Obrázek 4 - převod na rastr - převzato z [WIKI4]

Dalším způsobem ukládání vizuálních dat je tzv. vektorová grafika. V tomto případě je obraz složen z množiny geometrických objektů (body, přímky, křivky, polygony apod.). Tyto objekty jsou v souboru uloženy pomocí své matematické reprezentace. Výhodou vektorového uložení je možnost měnit měřítko obrázku bez ztráty kvality. Naopak nevýhodou je obtížnější pořízení obrázku, resp. jeho převod do vektorovém vyjádření.

4.1.2 Zvuková data

Z fyzikálního hlediska je zvuk vlnění definované pomocí frekvence (udává výšku tónu) a amplitudy (udává sílu zvuku). Pro převod zvuku do digitální podoby je třeba původní spojitý signál reprezentovat diskrétními vzorky. Tento proces se nazývá vzorkování. S určitou frekvencí probíhá snímání původní amplitudy, jejíž hodnoty jsou ukládány. Tímto vznikne digitální obdoba původního signálu, která aproximuje originální vlnění.

Takto digitalizovaný signál obsahuje podobně jako v případě rastrové grafiky velké množství redundantních dat. V případě bezeztrátové komprese je možné dosáhnout kompresního poměru maximálně 1:2. Proto se většinou v případě delších záznamů používá některý ze způsobů ztrátové komprese.

4.1.3 Video data

Video data chápeme jako sekvenci obrázků, kde každý obrázek má definován svůj čas, ve kterém se má objevit. Existuje zde tedy určitý temporální charakter těchto informací [CHME]. V případě videa se často spojuje vizuální obsah se zvukovým. Někdy jsou navíc přiloženy další dodatečné informace ve formě například textových dat (titulky) apod. Proto někdy o formátech schopných uchovávat různé data v několika odlišných formách hovoříme jako o tzv. multimediálních kontejnerech [CHME].

Podobně jako zvuková, i video data obsahují obrovské množství redundantních informací, které je možné odstranit bez většího dopadu na kvalitu výsledného záznamu. Tyto kompresní algoritmy většinou pracují s jednotlivými snímky (framy) video souboru. Přehled nejpoužívanějších formátu pro uchování videa se bude nacházet v závěrečné diplomové práci.

4.1.4 Meta data

Meta data představují nástroj pro popis již existujících dat. Tyto speciální informace slouží pro snadnější správu, katalogizaci a vyhledávání samotných požadovaných dat. Obecně lze meta data rozdělit do dvou skupin – ty které popisují samotné médium (např. kdy byl obrázek pořízen, jeho velikost apod.) a na ty, které popisují obsah zkoumaných dat (např. na obrázku je startující letadlo).

V případě tradičních médií byly postupem času přijaty určité standardy, které jsou obecně uznávány. Například kniha má svého autora, byla vydána nějakým vydavatelstvím a je označena

unikátním číslem (ISBN), pod kterým je vedena v různých databázích. Pomocí těchto údajů je pak snadné najít konkrétní požadovaný titul nebo v databázi knih klasifikovat uložené údaje.

V případě digitálních multimediálních dat však žádný obecně přijatý a široce používaný standard zatím neexistuje. Místo toho jsou používány způsoby popisu vyvinuté speciálně pro konkrétní formát (např. standard Exif sloužící pro popis obrázků pořízeny digitálními fotoaparáty ve formátu `TIFF` nebo `JPEG`, nebo technologie ID3 tagů používaná pro charakteristiku souborů ve formátu `MP3`).

V oblasti multimediálních databázích však v poslední době probíhá v problematice způsobů popisu dat značný rozvoj. Třetí generace databází pracující s multimediálním obsahem je již z velké části založena na meta datech (viz kapitola 2.3).

4.1.4.1 Formát MPEG-7

Formát MPEG-7 (celým názvem Multimedia Content Description Interface) byl vyvinut speciálně k popisu multimediálních dat a jejich obsahu. Jde o otevřený standard, které je možné použít pro různé formáty dat. Jedním z hlavních cílů při vývoji tohoto rozhraní byla snaha o konečnou standardizaci při popisu multimediálních dat. Byl vyvinut skupinou Motion Pictures Expert Group, která mimo jiné stojí za rodinou populárních komprimačních algoritmů MPEG.

Tento formát je založen na množině tzv. deskriptorů, které slouží pro popis samotných dat. Ty definují strukturu samotného popisu konkrétního média. Liší se podle formátu dat . MPEG-7 nabízí skupinu standardních deskriptorů určených pro nejběžnější formáty - obrázky, video, zvuk, řeč atd..

Samotný popis média nijak nesouvisí s jeho uložením. Může jít o tištěný obrázek, stream dat nebo klasický analogový film.

Deskriptory jsou uchovávány ve formátu založeném na populárním značkovacím standardu XML. Tento textový formát lze převézt i do efektivnější binární podoby (BiM).

Následuje přehled a popis nejdůležitějších komponent MPEG-7, které definuje norma tohoto formátu:

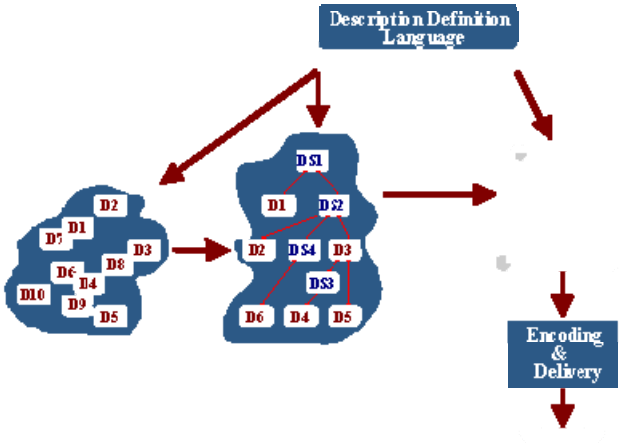
- Deskriptory (Descriptors – D) – deskriptory popisují vlastnosti zkoumaného multimediálního obsahu, jsou založeny na katalozích (např. informace jako název, autor, popis). Jejich další součástí jsou popisy sémantiky (např. informace o objektech a událostech), syntaxe média (například barva obrazu) a samotné technologii přenosového média (formát, rozlišení, vzorkovací frekvence aj.).
- Schémata popisu (Description Schemes – DS) – určují strukturu a sémantiku vztahů mezi komponentami (jak mezi deskriptory, tak mezi schématy popisu). Může jít například o původ média, jeho typ, vlastnosti, historii použití apod..
- Datové typy (Datatypes – D) – nástroje pro definici nových datových typů. Původní schéma XML není určené pro popis multimedií. Proto norma MPEGu-7 mimo jiné

zavádí některá rozšíření v oblasti datových typu (např. jednoduché pole, matice). Kromě toho umožňuje definovat uživateli vlastní odvozené datové typy.

- Jazyk pro definici deskriptorů (Description Definition Language – DDL) – tento jazyk umožňuje vytvoření nových schémat popisu nebo úpravu těch stávajících, definici nových deskriptorů, datových typů apod..
- Systémové nástroje (Systém tools) – přináší podporu pro binární formu popisů, jejich synchronizaci, správu, ochranu autorských práv apod.

Samotnou normu MPEGu-7 je možné podle tématu, kterému se věnuje, rozdělit na 8 hlavních částí (převzato z [CHME]):

- Systém: poskytuje nástroje pro přípravu deskriptorů k přenosu, ukládání, kompresi a pro synchronizaci s obsahem.
- DDL: určuje jazyk pro definování standardní množiny popisových nástrojů (DS, D, DT) a nových nástrojů, založeno na XML.
- Vizuální: obsahuje nástroje pro popis vizuální složky, specifikuje základní kategorie – barva, textura, tvar, pohyb, pozice, rozpoznávání obličeje.
- Audio: definuje nástroje pro popis zvukové složky jako spektrální, časové, dynamické vlastnosti (low-level) nebo rozpoznání hlasu, barva nástroje, melodie (high-level).
- Schémata popisující multimédia: nástroje pro popis multimediální části popis obsahu – audio i vizuální (video), použití obsahu, organizace, navigace, interaktivity.
- Referenční software: implementace standardu se stále vyvíjí, protože norma popisuje způsob uložení popisu, nikoli způsob jeho získání.
- Testování shody: poskytuje vodítka pro testování shody implementace deskriptorů.
- Extrakce a použití MPEG-7 schémat. Pouze informativní, není součástí normy, například implementace experimentálního modelu.



Obrázek 5 - hlavní elementy systému MPEG-7 (převzato z [MP7O])

5 Závěr

Práce osahuje obecný úvod k tématu multimediálních databází. Byla v ní nastíněna stručná historie databázových technologií, jejich dnešní stav, principy relačního modelu dat a objektově orientovaného přístupu a základní pojmy z oboru multimediálních dat a databází. Byl vysvětlen termín meta data a s ním související formát MPEG-7.

V dalších kapitolách, už v rámci závěrečné diplomové práce, se budu věnovat nejdůležitějším formátům ukládání multimediálních dat. Dále pak vyložím základní poznatky z oboru teorie vyhledávání v multimediálních datech podle obsahu. Budu se věnovat i extrakci rysů získávání popisu obsahu. Nakonec se zaměřím na implementaci multimediální databáze s důrazem na vizuální data.

Literatura

[ZEN] ZENDULKA, Jaroslav. *Databázové systémy*., 2005. 217 s. Fakulta informačních technologií, VUT Brno. Studijní materiál.

[SVE] ŠVEC, Martin. *Objektové databáze*., 2003. 20 s. Fakulta informačních technologií, VUT Brno. Studijní materiál.

[CHME] CHMELAŘ, Petr. *Multimediální databáze*., 2006. 59 s. Fakulta informačních technologií, VUT Brno. Studijní materiál.

[ZAK] ŽÁK, Karel. Historie relačních databází. *Root.cz* [online]. 2001 [cit. 2007-12-26]. Dostupný z WWW: <http://www.root.cz/clanky/historie-relacnich-databazi/>.

[DOS] DOSTÁL, Radim. Objektově orientované programování v C++ : Základní pojmy objektově orientovaného programování. *Builder* [online]. 2000 [cit. 2007-12-27]. Dostupný z WWW: <http://www.builder.cz/art/cpp/cpp_zaklad_obp.html>.

[WIKI1] Reference counting. *Wikipedia* [online]. 2007 [cit. 2007-12-28]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Reference_counting>.

[WIKI2] Garbage collection (computer science) . *Wikipedia* [online]. 2007 [cit. 2007-12-28]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Tracing_garbage_collection>.

[WIKI3] Multimedia. *Wikipedia* [online]. 2007 [cit. 2007-12-28]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Multimedia>.

[WIKI4] Rastrová grafika. *Wikipedia* [online]. 2007 [cit. 2007-12-29]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Rastrov%C3%A1_grafika>.

[WIKI5] Metadata. *Wikipedia* [online]. 2007 [cit. 2008-01-03]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Metadata>.

[ZAV] ZÁVODNÝ, Tomáš. *Videoformáty, videokodeky*. [s.l.], 2002. 30 s. Fakulta informatiky, Masarykova Univerzita. Bakalářská práce. Dostupný z WWW: <http://www.fi.muni.cz/~xpavlov/xml/examples/bc3/bc3.html>.

[HOF] HOFMAN, Jiří. *Formáty multimediálních soubor*ů [online]. 2006. [cit. 2006-06-19]. Dostupný z WWW: <http://www.aldebaran.cz/studium/formaty.html>.

[KODO] KOSCH, Harald, DÖLLER, Mario. Multimedia Database Systems : Where are we now?. [online]. 2006 [cit. 2007-12-20]. Institute of Information Technology, University Klagenfurt. Dostupný z WWW: <http://www-itec.uni-klu.ac.at/~harald/MMDBoverview.pdf>.

[MP7O] ISO/IEC JTC1/SC29/WG11. *MPEG-7 Overview* [online]. Martínez, José M. Palma de Mallorca : 2004 [cit. 2005-10-22]. Dostupný z WWW: <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm>.

[HRU] HRUŠENSKÝ, Michal. *MPEG-7*. [online]. 2007 [cit. 2008-01-03]. Dostupný z WWW: <http://miska.borec.cz/download/mpeg7-c.pdf>.