Univerzita Karlova v Praze

Filozofická fakulta

Ústav informačních studií a knihovnictví

Studijní program: informační studia a knihovnictví

Studijní obor: informační studia a knihovnictví

Rudolf Kreibich

Web Modeling Language

(přehledová studie)

Bakalářská práce

Praha 2007-08-03

Vedoucí bakalářské práce: PhDr. Helena Kučerová

Oponent bakalářské práce:

Datum obhajoby:

Hodnocení:

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité informační zdroje.

V Praze, 3. srpna 2007

… … … … … … … … … … ……

podpis studenta

Identifikační záznam

Kreibich, Rudolf. Web Modeling Language: přehledová studie [Web Modeling Language: Overview]. Praha, 2007-08-03. 45 s., V s. příl. Bakalářská práce. Univerzita Karlova v Praze, Filozofická fakulta, Ústav informačních studií a knihovnictví. Vedoucí bakalářské práce PhDr. Helena Kučerová.

Abstrakt

Bakalářská práce WebML: přehledová studie podává stručný přehled o metodologii WebML, zahrnující kompletní životní cyklus webové aplikace. Cílem práce je seznámit čtenáře s jazykem WebML a naučit jej modelovat hypertext, proto je tedy v textu uvedeno množství příkladů. Práce se nejprve věnuje úloze a významu modelování v lidském poznání a popisuje vztah mezi skutečností a jejím modelováním. Dále čtenáře seznamuje se základními charakteristikami a vývojem modelovacího jazyka WebML a s aplikací Webratio, která jazyk WebML implementuje. Stěžejní části práce jsou věnovány dvěma oblastem. První z nich je datový model, jeho struktura a vztahy mezi jeho jednotlivými prvky. Druhou oblastí je hypertextový model. V této části jsou detailně popsány elementární části hypertextu jako jsou jednotky, stránky, odkazy, webové pohledy či možnosti jeho organizace.

Klíčová slova

WebML, Web Modeling Language, návrh webové aplikace, hypertextový model, datový model, objektově orientovaný jazyk, životní cyklus webové aplikace, domain specific language. navigační model.

OBSAH

1. Předmluva

Za téma své práce jsem si zvolil jazyk WebML, který slouží k modelování různých aspektů životního cyklu vývoje webové aplikace. Podrobný přehled metodik WebML k dílčím problematikám při vývoji webové aplikace, by však byl nad rámec formátu, jež mám k disposici. Zkrácenější varianta by na druhou stranu byla až příliš obecná. Proto jsem zvolil alternativní přístup.

Za cíl práce jsem si stanovil, seznámení čtenáře tohoto textu s dvěma důležitými aspekty při vývoji webové aplikace - s datovým a hypertextovým modelem. Přesněji vyjádřeno, datový model v mém textu slouží jako podklad pro výklad hypertextového modelu, neboť jen tak může čtenář porozumět návaznostem mezi oběma modely.

Organizace informací v hypertextovém prostředí je a bude klíčovým aspektem webových aplikací. Důkazem může být postupné etablovaní postu informačního architekta v průběhu tohoto desetiletí. Komplexní webové aplikace institucí, států či podnikatelských útvarů musí nabízet svým uživatelům dobře navržené rozhraní pro uspokojení různých informačních potřeb. A právě naplnění informačních potřeb uživatelů, je v centru zájmu informačního architekta. Hypertextový model WebML je pro informačního architekta výbornou pomůckou, jak modelovat prostředí v němž se bude uživatel pohybovat. Navíc je tento model integrován v komplexní metodologii vývoje webové aplikace a tak může informační architekt hladce pracovat ve vývojářském týmu.

Vedle stanoveného cíle čtenáře také stručně seznámím s aspekty vývoje webových aplikací v metodologii WebML a představím mu CASE nástroj Webratio.

2. Úvod

Text této práce jsem koncipoval s určitou pedagogického motivací, jež se projevila v několika rovinách. Mimo jiné v četnosti uvádění praktických příkladů využití jazyka WebML. Práce obsahuje určité množství redundance, která je motivována snahou o snadnější pochopení látky čtenářem. Text jsem dále doplnil o diagramy, jež mají ilustrovat problematiku a pomoci ji lépe absorbovat. Vzhledem k určitým komplikacím s aplikací Webratio, jsem byl nucen převzít diagramy z publikace Designing Data-Intensive Web Application.

Při psaní práce jsem jako metodu využíval studium dokumentů a elektronických zdrojů. Prakticky jsem se vzdal citací a poznámky použil jako platformu pro krátké dodatky k textu, jež slouží k hladšímu přechodu mezi pojmoslovím této práce a původních primárně anglických zdrojů. Absenci přímých citací zmírním uvedením informačních zdrojů k jednotlivým kapitolám dále v tomto úvodu. Vzhledem k zaměření této práce na dva ústřední modely WebML navíc postrádá přímé citování funkční smysl, neboť specifikace WebML je zakotvena ve velmi omezeném množství zdrojů a veškerá další literatura slouží maximálně k lepšímu porozumění jazyka skrze příklady, spíše než k objevováním tajů zkoumaného jazyka.

Kapitol 3 se zabývá obecným úvodem do problematiky modelování. Stručně čtenáře seznámí s úlohou modelování v naší historii a se základním myšlenkovým rámcem pro pochopení procesu modelování. Obsah této části primárně vychází z knihy Object-Oriented Modeling And Design With UML.

Kapitola 4 popisuje jazyk WebML skrze stručný výklad o procesu vývoje webové aplikace. Dále čtenáře seznámí s možnostmi aplikace Webratio. Primárními zdroji pro tuto kapitolu byly webové stránky http://www.webml.org, http://www.webratio.com a kniha Designing Data-Intensive Web Application

Kapitola 5 popisuje datový model. Seznámí čtenáře s elementy konceptuálního návrhu datové struktury. Vychází z knih Designing Data-Intensive Web Application a Návrh databází.

Kapitola 6 představuje čtenáři modelování hypertextového modelu. Předvede mu vedle základní elementy modelu, jako jsou jednotky, stránky, odkazy atd. Obsah vychází primárně z knihy Designing Data-Intensive Web Application a seriálu o WebML na serveru http://www.interval.cz.

Kapitolu 7 tvoří závěr, jež shrne nejdůležitější prvky práce s WebML.

3. Modelování

3.1. O modelování v našem světě

Skutečnost je dynamická síť vztahů, jíž jsme součástí. Ať jsme vědci s tvrdou metodickou průpravou, či lidé s úplně jinými ambicemi, vytváříme si během našeho života názor na různé aspekty skutečnosti v nás a kolem nás. Žádné vyjádření o povaze reality, však zatím nevystihlo skutečnost jako celek. Jednotlivé aspekty skutečnosti byly na různých úrovních zkoumány. Poznatky se neustále vyvíjí, revidují se závěry a náš pohled na svět se zpřesňuje, avšak ani zdaleka není završen. Není totiž dokonalým popisem, jež by vysvětlil všechny procesy, jichž jsme součástí. Komplexní popis zahrnující celé spektrum specializovaného poznání je v nedohlednu. Opravdovou jistotu, že naše představa o světě přesně odpovídá skutečnosti, nemáme. Skutečnost se zdá jako jedna věc a naše názory na ni druhá.

Právě komplikovanost a dynamičnost systému, jež nazýváme skutečnost, se zdá být původcem nemožnosti, vyjádřit se o ní s naprostou přesností. Naše mysl je schopná pracovat jen s omezeným počtem informací, kdežto skutečnost je bezbřehá. Abychom ji zachytili, vybíráme z ní jen pro nás důležité aspekty, jež uvádíme do vztahu. A teprve z těchto vztahů si vytváříme představu o skutečnosti. Jinými slovy si skutečnost modelujeme a pokud mluvíme o našich názorech, je užitečné mluvit o nich jako o modelech poznání, neboť vychází z naší představy o skutečnosti.

3.2. Modelování jako návrhářská technika

Model je abstrakce, používáme jej, abych porozuměli dané věci dříve, než ji vytvoříme. Model vytváříme s určitým cílem, zkoumáme povahu problému, umožňuje nám zhodnotit, které aspekty jsou důležité a které nejsou rozhodující pro konečný výsledek. Je to způsob, jak se

vypořádat se složitostí, která panuje v našem světě. Inženýři, vědci, umělci a řemeslníci vytváří modely už tisíce let, aby vyzkoušeli své návrhy dříve, než je realizují. Můžeme vysledovat čtyři obecné důvody proč nejdříve modelovat a až poté realizovat.

a.) **Testování konečného výsledku před jeho realizací.** Stavitelé katedrál stavěli zmenšené modely, aby vyzkoušeli odolnost struktury. Dnes architekti vytváří zmenšené a virtuální modely, aby otestovali schopnost staveb odolat vlivům prostředí. Model je z hlediska zdrojů méně nákladný, pokud model v simulaci selže, nebo je nedostatečný, může být upraven či nahrazen.

b.) **Komunikace se zákazníky.** Model zahrady nebo bytového prostoru umožňuje přiblížit zákazníkovi koncepci návrháře. Nastoluje intuitivní komunikační prostor v němž může zákazník formulovat své požadavky a návrhář své zkušenosti.

c.) **Vizualizace.** Podobně jako v předchozím bodě vytváří vizualizace komunikační prostor avšak pro návrháře. V reklamním a filmovém průmyslu konstruuje tvůrčí tým storyboard, jenž umožňuje jednotlivým členům rychle vnášet nové nápady, návrhy či přímo strukturovat jak jednotlivé segmenty, tak celkovou podobu díla.

d.) **Zjednodušení komplikovaného.** Nejspíše nejdůležitější důvod k modelování, který zahrnuje předchozí body, je vypořádání se soustavami, které jsou příliš komplikované nato, aby bylo pochopeny přímo. Lidská mysl pracuje jen s omezeným počtem informací najednou. Modely redukují komplexnost a vymezují malý počet důležitých informací, které mysl může v daném čase zpracovat.

4. Web Modeling Language

4.1. Úvod

Vývoj webových aplikací je komplexní proces, na němž se podílí tým specialistů. Šíře znalostí, jimiž musí tým disponovat, je dána složitostí projektu. Při vývoji webových aplikací se potkávají profese z mnoha odvětví lidského vědění. Programátoři, systémoví administrátoři a databázoví specialisté jakožto zástupci technologicky orientovaných oborů, se střetávají s humanitněji orientovanými grafickými designéry, experty na uživatelská prostředí a spolu s marketingovými specialisty, analytiky podnikových procesů či copywritery vytvářejí směsici různých přístupů k problematice webové aplikace. Potřeba jednotného rámce, který rozdělí problematiku na dílčí úkoly, jež můžou specialisté odděleně řešit, je na bíle dni.

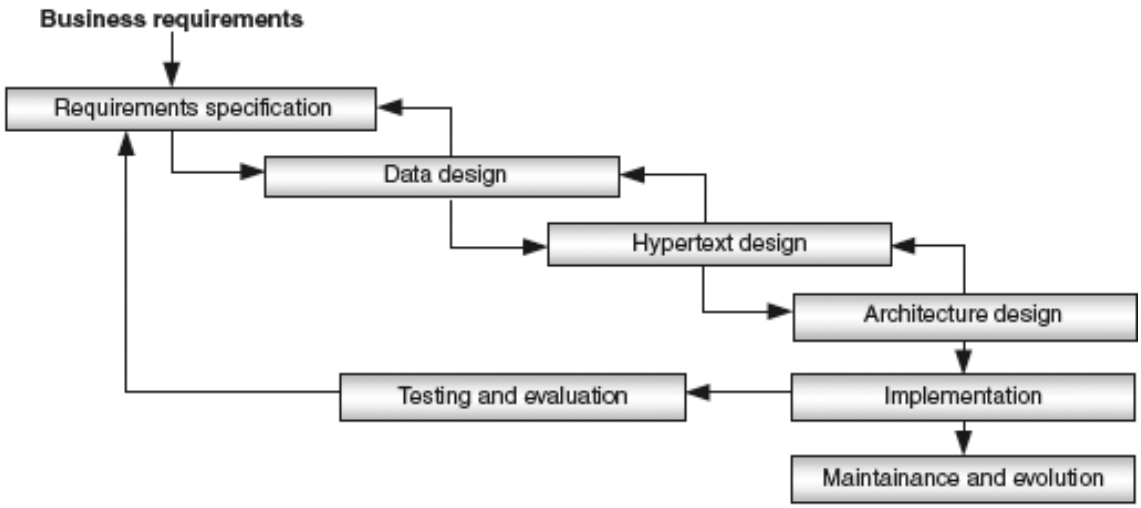
WebML vytváří rámec pro kompletní životní cyklus webové aplikace, od získávání informací o požadavcích zákazníka, přes návrh datových a hypertextových struktur k návrhu architektury na které bude aplikace běžet, až po její implementaci a následný provoz a rozvoj.

WebML umožňuje návrhářům internetových aplikací modelovat jejich chování a funkcionalitu, bez nutnosti zabývat se podrobnostmi konkrétní implementace. Jazyk disponuje bohatou grafickou notací popisující procesy ve všech částech návrhu. Je tak možné modelovat např. případové studie, uživatelské studie, business požadavky a ty následně mapovat na datový a hypertextový model. Jelikož má WebML definovanou i textovou notaci, lze podle sady pravidel vytvořit ze vzniklých textových popisů aplikační kód, určený pro implementaci do konkrétní architektury.

4.2. O jazyku

Od roku 1996 začali výzkumníci Ústavu pro elektroniku a informace na Polytechnice v Miláně pracovat na vývojovém rozhraní pro tvorbu velkých webových aplikací. Vytvořili prototyp aplikace, schopné z modelu stránky vytvořit aplikační kód. V roce 1998 představili první mezinárodní příklad *The Autoweb system,* demonstrovaný na Evropské konferenci pokročilých databázových technologií. V roce 2000 je nejen definována ale i patentována první verze WebML a je sestaven balíček Java nástrojů pro vývoj velkých webových aplikací. V roce 2001 vzniká firma Web Models, jejímž cílem je ekonomické využití WebML a na trh se dostává produkt Webratio 3.0. Během sedmi týdenního vývoje je v tomto nástroji navržena webová aplikace firmy Acer. V roce 2003 vzniká kniha Designing Data-Intensive Web Applications, jež zevrubně popisuje technologii i metodologii vývoje webových aplikací pomocí WebML. Od té doby vývoj aplikace Webratio pokračuje a jazyk WebML je adaptován mnoha firmami.

WebML patří mezi tzv. Domain specific languages, neboli jazyky orientované na konkrétní oblast problémů. Popisuje celý životní cyklus webové aplikace od sbírání požadavků a zjišťování business modelu po implementaci a údržbu webové aplikace. Proces vývoje probíhá v cyklech, aby bylo možné dosáhnout výchozích požadavků. Takový přístup v podstatě vychází



Ilustrace 1: Fáze vývoje webové aplikace

z metodologie softwarového inženýrství. Hlavním specifikem však je, že WebML zahrnuje do vývoje aplikace datové a hypertextové modelování.[1]

4.3. Struktura jazyka

Jazyk WebML se zaměřuje na tři dominanty vývoje webových aplikací, tvorbu datového modelu, návrh hypertextu a prezentačního modelu. Všechny modely musí být schopny, vyjádřit a naplnit vztahy a potřeby definované požadavky.

Pro datové modelování WebML vychází z dobře zavedeného popisu datových struktur pomocí Entity-Relationship modelu ale i z některých prvků UML. Úhelným kamenem ER modelu jsou entity které se mezi sebou propojují vztahy. Určitým způsobem tak modelují vztahy mezi skutečnými předměty. Entita Osoba například zahrnuje všechny lidi, entita Pes zase všechny psy. Mezi entitami Osoba a Pes můžou existovat vztahy typu „Šmudla je můj pejsek“, ale z druhé strany, z pohledu psa, existuje vztah „Jaroslav je můj páníček“.

Hypertextový model je tvořen dvěma úzce souvisejícím pod-modely. Kompoziční model váže datové struktury (konkrétně entity) definované v datovém modelu na tzv. jednotky, které zobrazují jejich obsah. Jednotky pak sdružuje do stránek. Hypertextový model vytváří vztahy mezi jednotkami nebo stránkami pomocí odkazů.

Prezentační model transformuje předchozí dva modely do webové prezentace pomocí XSL.

4.4. Webratio a jiné implementace

Jak z historického nastínění zřejmé WebML je silně svázán s CASE[2] aplikací Webratio. Přesto se dá implementovat pomocí jiných nástrojů, pokud porozumíme jeho metodice. Na oficiálních stránkách WebML je ke stažení modul s grafickými značkami do aplikace Microsoft

1 Hypertext byl už modelován například podle OOHDM, a je modelován následovníkem SHDM s volně dostupným IDE HyperDE postaveném na Ruby on Rails.

2 Computer-aided software engineering je aplikace pomáhající při vývoji a údržbě softwaru.

Visio a metamodel pro MagicDraw, v nichž můžeme nakreslit modely naší aplikace a pomocí MagicDraw i namapovat model do konkrétního kódu (převaděč si ale musíme napsat), bez použití Webratio. Na stránkách je také k nalezení DTD pro WebML. Textové notace popisující vznikající model, mohou být zapsány v XML a pokud vytvoříme převaděč kódu, můžeme z modelu získat přímo aplikaci.

Ale vraťme se k Webratiu. Webratio je pohodlné prostředí, běžící na Java platformě, uzpůsobené k návrhu webových aplikací podle WebML. Je schopné z datových a hypertextových modelů vytvářet hotové aplikace. Tato funkce je samozřejmě možná i v průběhu vývoje aplikace a tak můžeme kontrolovat chování aplikace přímo ve Webratio.

Webratio integruje do svého prostředí trendy poslední doby. V aktuální verzi se pokouší o generování kódu s pomocí AJAX technologií. A slibně vypadá snaha implementovat sémantický web do generovaných stránek.

5. Datový model

5.1. Struktura

5.1.1. Entity, instance entit, atributy a primární klíče

Entita reprezentuje popis společných vlastností skupiny předmětů našeho světa. *Atribut* je vyjádřením společné vlastnosti skupiny předmětů. Samotné předměty v této skupině se chápou jako *instance* entit.[3] Za entitu může považovat například *osobu, auto* či *umělce.* Pokud je entitou osoba, za její atributy považujme *jméno, datum narození, pohlaví* atd. Instancí entity je pak jedinečný člověk, jež má své konkrétní jméno, datum narození a pohlavní příslušnost.

Je důležité vědět, že instance entit mohou nabývat pro některé atributy tzv. *null hodnot*[4]. Jestliže nemáme u konkrétní osoby vyplněné číslo řidičského průkazu, můžeme situaci interpretovat ze dvou úhlů. Buď daná osoba řidičský průkaz nemá, nebo jsme tuto informaci nezjistili. Důsledkem takové situace je nepřesnost informace. Přesto null hodnoty nejsou vždy na škodu, v textu se k nim dále dostaneme.

Aby bylo možné jednotlivé instance entity mezi sebou absolutně odlišit, zavádí se atribut zvaný primární klíč, který je u každé instance entity povinný a jedinečný. V našem příkladě by ke jménu, roku narození a pohlaví, přibylo i rodné číslo. V metodologii WebML se o primárním klíči mluví jako o *object identifier (OID),* budu jej dále v textu používat jako pojem pro primární klíč, i když je v českém kontextu méně obvyklý.

3 Ceri et al. 2003: 62

4 Null reprezentuje chybějící či neznámou hodnotu. Null není nulou ani řetězcem složeným z jedné nebo více mezer. (Hernandez, 2006, s. 68)

5.1.2. Typy atributů – doplnit o grafické znázornění

Pro implementaci datového modelu, je potřeba znát jaký typ dat bude atribut obsahovat. WebML popisuje tyto základní atributové typy:

Tabulka 1 Základní datové typy

Řetězec (String)	Krátká posloupnost znaků
Text (Text)	Dlouhá posloupnost znaků. Druhy textů mohou být upřesněny vyjádřením pomocí MIME (např. text/html)
Celé číslo (Integer)	Celočíselná forma čísla
Desetinné číslo (Float)	Desetinná forma čísla
Datum (Date)	Kalendářní datum
Čas (Time)	Číselné vyjádření času
Boolovský (Boolean)	Boolovská hodna ano či ne
Číselník (Enumeration)[5]	Uživatelem určená konečná množina hodnot
Binární (BLOB)	Binary Large OBject – velké binární datové typy, například obrázky, video atd. Mohou být upřesněny vyjádřením pomocí MIME (např. image/gif)
URL	Uniform Resouce Locator

Například atribut DatumNarození bude obsahovat datový typ *datum,* tedy např. hodnotu 7.6.1983. Atribut Pohlaví bude mít typ *číselník* o dvou předdefinovaných hodnotách mužské a ženské.

5.1.3. Dědičnost[6]

WebML podporuje základní typ dědičnosti tzv. *is-a hierarchii*, kdy jedna super-entita může mít jednu a více pod-entit. Pod-entity dědí atributy super-entity. Takže pokud vytvoříme super-entitu Osoba s atributy jméno a rok narození, budou pod-entity Umělec a Řemeslník dědit atributy jméno a rok narození a navíc si budou moci přiřadit atribut profesní specializace. Není podporována pokročilá dědičnost, kdy jedna pod-entita dědí atributy z více super-entit.

5 Číselník není v mnou zkoušené beta verzi webratio k disposici.

6 Pojmem dědičnost jsem nahradil původní termín Generalization Hierarchies, který není pro čtenáře na první pohled dostatečně jasný.

5.1.4. Vztahy

Entity mohou být sémanticky propojeny pomocí *vztahů*. Nejjednodušší vztah je binární, který propojuje dvě entity. Na binární vztah se pohlíží z perspektiv obou entit, každá entita proto vystupuje ve dvou rolích. Pokud vytvoříme vztah mezi entitami Autor a Dílo, můžeme vztah nazvat autorství. Z pohledu díla je autor v roli tvůrce díla a z pohledu autora je dílo v roli výtvoru autora. Vztah tedy spojuje zdroj s cílem a záleží na roli, která entita je chápána jako zdrojová a která cílová. Pro přehlednost se role vyjadřuje složeninou AutorToDílo, nebo DíloToAutor, kdy zdroj je vždy vlevo.[7]

Role mohou být označeny minimální a maximální kardinalitou, tedy číselným vymezením minima a maxima předmětů cílové entity ke kterému se může zdrojová entita vztahovat. Významné hodnoty jichž může minimální kardinalita nabývat jsou *nula* a *jedna*. Vztah je *nepovinný* pokud zdrojová entita nabývá minimální kardinality nula, jinak se jedná o vztah *povinný*. Povinné vztahy vytváří existenční závislost mezi entitami, neboť zdrojový předmět nemůže existovat bez alespoň jediné návaznosti na předmět cílové entity. Maximální kardinalita může nabývat hodnoty jedna nebo mnoho. Varianta mnoho se označuje jako „*N*“.

Na základě maximální kardinality se vztahy rozdělují na do tří skupin. Vztah *jedna k jedné* označuje vztahy dvou entit, jež v obou rolích nabývají kardinální hodnoty jedna. Vztah j*edna k mnoha* označuje, pokud jedna kardinalita nabývá hodnoty jedna a druhá hodnoty N. Vztah *mnoho k mnohému* označuje situaci, kde obě maximální kardinality nabývají hodnoty N.

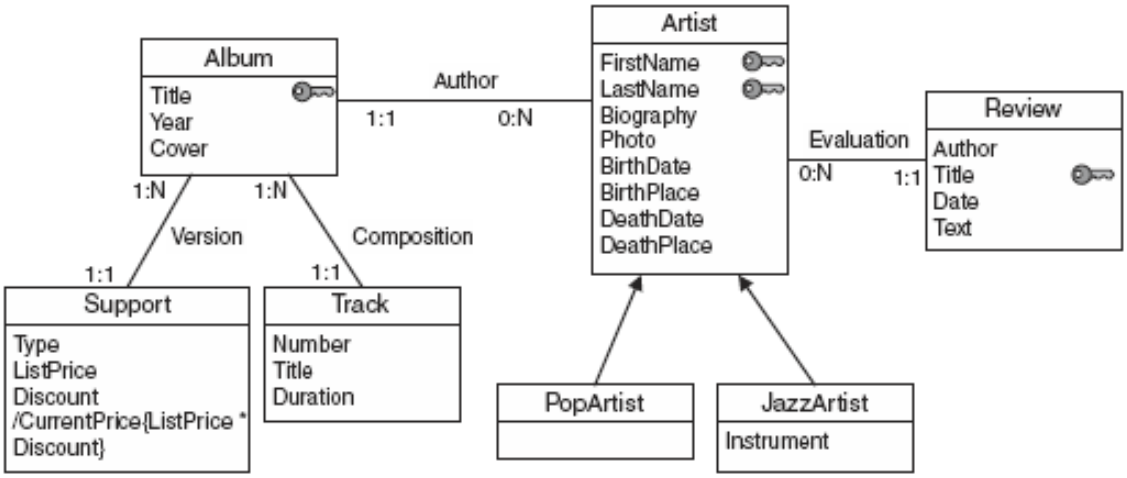
Jelikož datový model WebML vychází z ER schématu, podporuje i tzv. *N-arní vztahy* kde do vztahu vstupují více jak dvě entity a tzv. *vztahy s atributy*. Nicméně tyto komplikovanější vztahy je možné a navíc doporučené reprezentovat v podobě binárního vztahu za pomocí centrální spojovací entity. Centrální entita je pak napojena na entity pomocí dvou a více binární vztahů.

7 Při implementaci názvů entit se obecně vyhýbá diakritice, který se může v některých RDMBS chovat neočekávaně. V textu diakritiku pro větší gramatickou uhlazenost ponechávám. Zachovávám ovšem anglické „To“, přestože jej možné zastoupit českým k, či znakem „-“. Pomlčka nevyjadřuje dostatečně vztah mezi entitami a vzhledem k programátorskému zvyku udržovat text dobře čitelný, nevyužívám ani konstrukce pomocí „k“. Např. AutorKDílo či AUTORkDÍLO vs. AutorToDílo.

5.1.5. Odvozené informace

Při datovém modelování můžeme narazit na situaci, kdy hodnota atributu či vztahu není explicitně vyjádřena v schématu, ale je vypočtena z hodnot jiných elementů. Potřebujeme-li například znát celkový počet skladeb určitého interpreta, pomocí operace join[8] spojíme interpretem publikovaná alba a sečteme v nich obsažené stopy. Všechny takto vypočtené hodnoty atributů a vztahů jsou nazvány *odvozenými*.

Konstrukce derivovaného atributu či vztahu je přímočará. Je předznačen znakem lomítko („/“). Poté se atribut či vztah doplní o *výraz,* který vyjádří pravidlo pro výpočet hodnoty.[9] Jednoduchý příklad: entita Článek obsahuje dva obyčejné atributy Cena a Sleva, a derivovaný atribut /ZlevněnáCena. Ten jen vypočten výrazem (Cena*Sleva).



Ilustrace 2 Příklad datového schématu

Komplikovanější příklad s Entitou Interpreta který obsahuje mimo jiné derivovanou hodnotu /PočetAlb, se doplní o výraz `Count(Interpret.InterpretToAlbum)`. Takový výraz spočítá všechny alba asociované s interpretem podle vztahu InterpretToAlbum.

8 Join je operace popsaná v SQL a OQL, spojující tabulky. Realizuje se až v běžící aplikaci na úrovni RDBMS.

9 ER model nemá standardně žádný syntax pro výpočet odvozených informací. Odvozené informace je však možné definovat v každém modelovacím jazyce podporujícím *výrazy* (expressions) vytvořené z atributů a *výrazy cest* (path expressions). WebML využívá syntax jazyka Object Constraint Language (OCL) definovaného v UML.

Pro zápis výrazu pro derivovaný vztah použijeme úvodní příklad s výpočtem všech skladeb interpreta. Entita Interpret je spojena s entitou Stopa pomocí derivovaného vztahu /InterpretToStopa, který spojuje vztah interpreta a jeho alba se vztahem alba ke své stopě. Výraz je pak vyjádřen takto:

```
Interpret.PublikovanéStopy=Interpret.InterpretToAlbum.AlbumToStopa.
```

5.2.

6. Hypertextový model

6.1. Úvod

Cílem hypertextového modelování je specifikace uspořádání uživatelského rozhraní webové aplikace a vytvoření navigačního modelu.[10] Oba dva cíle lze modelovat pro různé skupiny uživatelů. Máme tedy možnost vytvořit model pro nepřihlášené uživatele, který může přistoupit jen k omezeným informacím a smí využívat nejzákladnějším funkce webové aplikace, jako je vyhledávání. Model pro přihlášeného uživatele pak definuje pokročilé funkce jako vkládání a úpravu obsahu webové aplikace, přístup k administrátorským nastavením aj.

Webová aplikace se v kontextu WebML chápe jako kolekce *stránek*. Každá stránka je složena z *jednotek* a *odkazů*. Tyto tři prvky jsou organizovány do vyšších struktur tzv. *oblastí* a *webových perspektiv*.[11] Skrze webové perspektivy se realizují webová rozhraní pro různé uživatelské role.

Jednotky jsou nejmenší části publikovatelného obsahu, jsou stavebním kamenem stránek. Stránka je tedy uživatelské rozhraní vybudované z různých typů jednotek (např. vstupní formuláře, textového bloku atd.) za účelem dosažení žádaného komunikačního efektu.[12] Stránky jsou propojeny odkazy a vytváří hypertextovou strukturu. Odkazy jsou úhelným kamenem hypertextového modelování, umožňují navigaci z jedné stánky na druhou, stejně jako správné zobrazení obsahu vytvořeného z jednotek.

10 Hypertextový model je složen ze dvou modelů. Kompoziční model definuje logické uspořádání uživatelského rozhraní a navigační model definuje provázanost stránek. Oba dva modely spolu úzce souvisí a tak je v textu pro jednoduchost nerozlišuji.

11 Originální znění jest: jednotky=units, stránky=pages, odkazy=links, oblasti=areas, webové perspektivy=views

12 Ceri et al. 2003: 103

6.1.1. Jednotky

Základními typy jednotek jsou *datové jednotky, vícečetné datové jednotky, indexové jednotky, rolovací jednotky a vstupní jednotky*.[13] První čtyři jednotky modelují publikované informace, pátá vyjadřuje možnost získává informací od uživatele webové aplikace. První dvě vybírají a zobrazují informační obsah z entit, další dvě slouží k výběru objektů. Jednotky berou obsah z datového schématu (kromě datového vstupu) – WebML používá dva koncepty původu obsahu – zdroj a selektor. Zdroj je jméno entity, z které jednotka extrahuje obsah. Selektory vybírají z atributu konkrétní položky podle zadaných podmínek.

6.1.1.1. Datové jednotky

Prezentují jednotlivou informaci z instance entity. O této informaci budu dále mluvit jako o *položce.* Datová jednotka se skládá ze *jména, zdroje, selektoru* (nepovinného) a *zahrnutých atributů.*

Jméno je návrhářem definované jméno pro datovou jednotku. Zdroj definuje entitu, která poskytne obsah datové jednotce. Selektor je predikátový výraz, který jednoznačně identifikuje konkrétní položku. Selektor je nepovinný, jen pokud zdrojová entita obsahuje jedinou položku. Zahrnuté atributy určují, které atributy zdrojové entity budou zobrazeny.

Pokud bych chtěl například zobrazit velmi zkrácenou informaci o nějakém interpretovi, v textové notaci ji vyjádřím takto:

```
DataUnit InterpretZkráceně
(source Interpret;
selector Jméno=“Amy“, Příjmení=“Winehouse“;
attributes Jméno, Příjmení, Foto)
```

Datovou jednotku jsem si pojmenoval InterpretZkáceně a v závorkách jsem definoval následující podmínky. Zdrojem z kterého budu čerpat obsah, bude entita Interpret. Selektor omezí můj výběr jen na interpreta, který splňuje podmínku, že obsahem atributu Jméno je Amy

13 Původní znění: datové jednotky=data units, vícečetné datové jednotky=multidata units, indexové jednotky=index units. rolovací jednotky=scroller units, vstupní jednotky=entry units.

a zároveň Příjmení je rovno hodnotě Winehouse. Zahrnuté atributy Jméno, Příjmení a Foto značí, že se do datové jednotky nahrají informace o jméně interpreta, doplněné fotkou.

Další predikátový výraz podporovaný u datové jednotky je *hodnotová disjunkce*, která se vyjadřuje znakem roury „|“ a koresponduje s operátorem „OR“. Zápis výrazu je ve formě [atribut operátor hodnota1 | hodnota2 | . . . | hodnotaN]. Případnou podmínkou je např. `MístoNarození contains „Jamajka“ | „Etiopie“.` Kde MístoNarození je atribut, contains je operátor a Jamajka s Etiopií jsou hodnoty jedna a dvě. Výraz tedy vyhodnotí podmínku za pravdivou, pokud pokud atribut MístoNarození bude obsahovat hodnotu Jamajka nebo Etiopie.

Atributová disjunkce zase umožňuje porovnávat jednu hodnotu ve více atributech. Konstrukce výrazu je v ve formě [atribut1 | atribut2 | . . . | atributN operator value]. Príklad je: `MístoNarození | Biografie contains „Jamajka“`. Výraz vyhodnotí podmínku za pravdivou, pokud atribut MístoNárození nebo Biografie bude obsahovat hodnotu Jamajka.

6.1.1.2. Vícečetné datové jednotky

Vícečetné datové jednotky prezentují několik položek entity najednou. Vícečetná datová jednotka se skládá ze *jména, zdroje, selektoru* (nepovinného), *zahrnutých atributů* a *řadícího výrazu* (nepovinné).[14] Na rozdíl od datové jednotky může vícečetná datová jednotka zobrazit více položek na jednou, takže pokud nespecifikujeme selektor, zobrazí se všechny existující položky v zadaných atributech. Řadící výraz navíc určí, podle kterých atributů se položky seřadí a zdali vzestupně (standardní nastavení pokud řadící výraz neuvedeme) nebo sestupně. [15]

Následující příklad zobrazí atributy Jméno, Příjmení a Foto ale díky řadícímu výrazu *orderby*, seřadí položky nejdříve podle příjmení a pokud jsou příjmení stejná, pořadí určí křestní jméno.

```
MultidataUnit IntepretVícekrát
(source Intepret;
```

14 Řadící výraz je v původním znění order clause.

15 Vzestupné řazení je vyjádřeno výrazem ascending, sestupné descending.

```
attributes Jméno, Příjmení, Foto;
orderby Příjmení, Jméno)
```

6.1.1.3. Indexové jednotky

Indexové jednotky zobrazí položky entity jako seznam. Na rozdíl od vícečetné datové jednotky zobrazuje indexová jednotka typicky jen jednu sadu položek. Lépe rozdíl pochopíme až budeme zkoumat odchozí odkazy v jedné z dalších kapitol. Indexová jednotka se skládá ze *jména, zdroje, selektoru* (nepovinného), *zahrnutých atributů* a *řadící podmínky* (nepovinné).

V následujícím jednoduchém příkladě zobrazíme pouze atribut Titul ze zdrojové entity Album.

```
IndexUnit AlbumIndex
(source Album;
attributes Titul;
orderby Titul)
```

Indexová jednotka připouští dva přístupy, jak zobrazit položky. V první variantě můžeme vytvořit seznam položek, kde každá položka je doplněna zaškrtávacím polem.[16] Takto je uživateli umožněno zaškrtnou si několik preferovaných položek, místo jediné. Tato varianta se nazývá *více-výběrová indexová jednotka* a definuje se v textové notaci následujícím způsobem: za deklarací typu jednotky a návrhářova pojmenování jednotky, následuje výraz multi-choice.

```
IndexUnit AlbumIndex multi-choice
(source Album;
attributes Titul;
orderby Titul)
```

Druhou variantou indexové jednotky je hierarchický *index*, v kterém jsou položky organizovány do vícevrstvého stromu. Hierarchie je reprezentována posloupností několika zdrojových entit spojených vztahem jedna k mnoha. První zdrojová entita vytváří nejvyšší stupeň hierarchie, následující zdrojová entita předznamenaná výrazem NEST další pod stupeň

16 Zaškrtávacím polem myslím klasický check-box.

hierarchie a tak dále. Každá vztahová role značí vztah rodič-potomek mezi po sobě jdoucími entitami.

Textová notace přidává ke každé další entitě v hierarchii vlastnosti zahrnuté v atributech a řadících podmínkách. Za deklaraci typu a popisu jednotky přidáme výraz hierarchical.

```
IndexUnit AlbumIndex hierarchical
(source Album;
attributes Titul;
orderby Titul;
        NEST Stopa
        selector AlbumToStopa;
        attributes Titul;
        orderby Titul)
```

Na každé úrovni hierarchie můžeme pro zdrojovou entitu zvolit selektor individuálně. Následující příklad ukazuje hierarchický index s alby vydanými v roce 2002, v seznamu se zobrazí jen stopy kratší jak dvě minuty.

```
IndexUnit AlbumIndex hierarchical
(source Album;
selector Rok=2002;
attributes Titul;
orderby Titul
        NEST Stopa
        selector AlbumToStopa, Trvani < 120;
        attributes Titul;
        orderby Titul)
```

Zvláštní případ hierarchického indexu dovoluje odkazovat entitu k sobě samé, pomocí sebe-referenčního vztahu definovaného v datovém modelu. Předpokládejme, že entita Součástka odkazuje k sobě pomocí vztahu SoučástkaToSoučástka (takový vztah vyjadřuje, že součástka je složena z menších součástek). V takovém případě má hierarchický index počet

úrovní odvíjející se od počtu položek v pod úrovni. Tato varianta hierarchického indexu se deklaruje předznamenáním výrazu NEST výrazem RECURSIVE.

```
IndexUnit SeznamSoučástí hierarchical
(source Součástka;
attributes JménoSoučástky;
orderby JménoSoučástky
        RECURSIVE NEST Součástka
        selector SoučástkaToSoučástka;
        atributes JménoSoučástky;
        orderby ČísloSoučástky)
```

6.1.1.4. Rolovací jednotky

Umožňují modelovat posouvání vetším počtem položek. Rolovací jednotka se skládá ze j*ména, zdroje, selektoru* (nepovinného), r*olovacího čísla* a *řadících podmínek.*[17] Kde selektor určuje položky kterými se posunuje. A rolovací číslo určuje kolik položek posouváme najednou. Řadící podmínky určují atributy podle kterých seřadíme položky a zda-li vzestupně (standardně) nebo sestupně.

V tomto příkladě vidíme, že rolovací jednotka je aplikována na entitu Album bez deklarace selektoru. Takto se můžeme pohybovat mezi všemi alby. Možné pohyby jsou vpřed, vzad a skok na poslední nebo první album.

```
ScrollerUnit AlbumRolování
(source Album;
blockFactor 1;
orederby Titul)
```

6.1.1.5. Vstupní jednotky

Podporují na formulářích založené datové vstupy. Vstupní jednotka je definována j*ménem a poli.*[18] Pole se chápou jako vstupní pole, do nichž se vkládají hodnoty. Typicky se využívají k

17 rolovací číslo=block factor

18 pole=fields

uživatelské interakci. Například při přihlašovaní do webové aplikace zadá uživatel vstupní jednotce své uživatelské jméno a heslo.

V tomto textovém příkladě má vstupní jednotka čtyři pole, jež slouží k vyhledávání interpreta:

```
EntryUnit InterpretVstup
(fields
JménoPole String;
PříjmeníPole String;
DatumNarozeníPole Date;
DatumSmrtiPole Date)
```

Vstupní pole se skládají ze *jména, typu, počáteční hodnoty* (nepovinné), *modifikovatelnosti, kontrolní operace*.[19] Typ deklaruje, že vstupní hodnota je řetězec, text, celé číslo, datum nebo jiný typ datového pole. Počáteční hodnota umožňuje zadat předvyplněnou hodnotu do pole, modifikovatelnost pak určuje, zda ji uživatele může či nesmí přepsat (bez deklarace se každé pole považuje standardně za modifikovatelné). Kontrolní operace umožňuje definovat způsob kontroly vstupní hodnoty zadané uživatelem. Operace se definuje jakýmkoliv logickým výrazem vytvořeným ze jména pole, specifického operátoru pro daný vstupní datový typ, a konstantního či proměnného termínu. Proměnným termínem může být jméno jiného pole, tak je umožněno porovnat hodnoty různých polí a například zajistit, že datum smrti interpreta je větší než datum jeho narození. Speciálním klíčovým slovem *notnull* zajistíme nutnost vyplnění daného pole.

```
EntryUnit InterpretVstup
(fields
JménoPole String, notnull;
PříjmeníPole String, notnull;
DatumNarozeníPole Date, notnull;
DatumSmrtiPole Date, DatumSmrtiPole > DatumNarozeníPole)
```

19 typ=type, počáteční hodnota=initial value, modifikovatelnost=modifiability, kontrolní operace=validality predicate

6.1.2. Stránky

Stránky jsou složené z několika spolu souvisejících jednotek, jež vytváří rozhraním s kterým komunikuje uživatel. Textový popis jednoduché stránky s dvěma indexovými jednotkami je následovný.

```
Page AlbumStránka
(units AlbumIndex, InterpretIndex)
```

6.1.3. Odkazy

Odkazy vytváří nejen spojení ze zdrojové stránky na cílovou, ale i mezi jednotkami. Při modelování vytváříme pomocí odkazů navigační model, který určuje dostupné cesty uživatele hypertextem a specifikujeme, jak bude moci uživatel ovlivnit zobrazení informací na stránce. Základními pojmy navigačního modelování jsou *odkaz, parametr odkazu a parametrický selektor.*[20] Odkaz je směrované spojení mezi dvěma jednotkami či stránkami. Parametrický odkaz je specifikace informace přenášené ze zdrojového odkazu k cílovému. Parametrický selektor vybírá jednotku, jež obsahuje referenci na parametr odkazu.

6.1.3.1. Specifikace odkazů

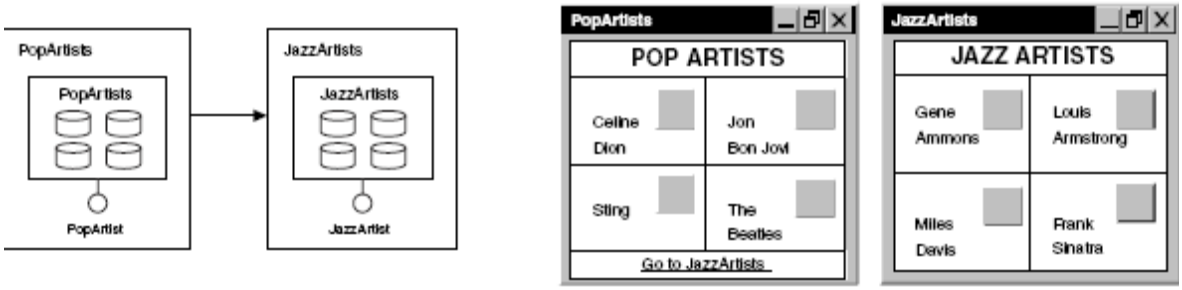
Odkazy slouží zaprvé k navigaci hypertextem tím, že umožňují uživateli přejít ze zdrojové stránky na cílovou. Zadruhé přenášejí informaci z jedné jednotky do jiné. Například odkazem přeneseme primární klíč vybrané položky z indexu do datové jednotky, která zobrazí žádanou položku.

Terminologie WebML rozlišuje několik druhů odkazů. Odkazy které směřují mimo hranice stránky se nazývají *mezi-stránkové odkazy.* Odkazy jež nepřekročí hranici zdrojové stránky jsou *vnitro-stránkové odkazy.* Odkazy přenášející informace jsou *kontextové odkazy*, *nekontextové odkazy* nepřenášejí žádnou informaci.

Graficky se odkazy reprezentují pomocí směrových šipek, které vedou od zdrojového elementu k cílovému. Ilustrace ukazuje mezi-stránkový nekontextový odkaz. Spojuje zdrojovou

20 odkaz=link, parametr odkazu=link parametr, parametrický selektor=parametric selector

stránku PopInterpreti s cílovou stránkou JazzIntepreti. Stránky PopInterpreti i JazzIntepretti obsahují vícečetnou datovou jednotku, jež zobrazuje interprety svého žánru. Obě stránky mají nezávislý obsah, proto není potřeba odkazem předávat jakoukoliv informaci.



Ilustrace 3 Nekontextový odkaz

Odkaz vyjádřený textově zahrnuje popisné jméno a určení zdroje a cíle.

```
link PopToJazz
(from PopIntepreti to JazzInterpreti)
```

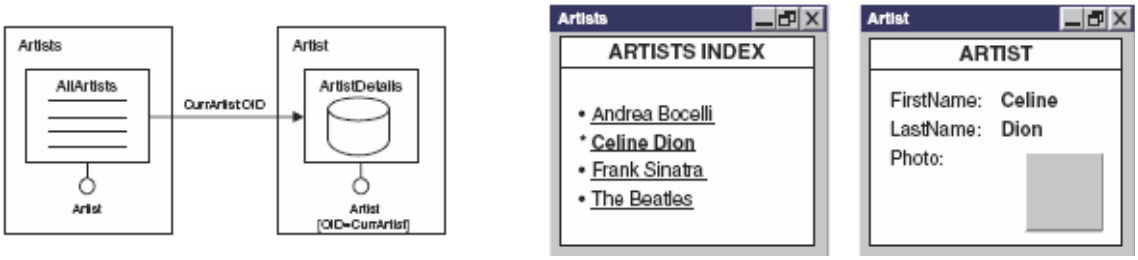
Mezi-stránkový kontextový odkaz je zobrazen níže. Stránka Interpreti obsahuje indexovou jednotku nazvanou Všichni Interpreti beroucí data z entity Interpret. Indexová jednotka je napojena na datovou jednotku InterpretDetaily, jež je umístěná na jiné stránce a bere data též z entity Intepret. Takto docílíme stránky, která zobrazí seznam všech interpretů a pokud uživatel klikne na jednoho z interpretů, dostane se na stránku s podrobnými informacemi o vybraném interpretovi. V tomto případě obsah cílové stránky závisí na informaci dodané ze zdrojové stránky.

6.1.3.2. Parametrické odkazy a selektory

Spojení mezi zdrojovou a cílovou jednotkou je reprezentováno parametrem odkazu, který se definuje v deklaraci odkazu. A parametrickým selektorem definovaném v cílové jednotce.

Parametr odkazu je hodnota spojená s odkazem. Tato hodnota se při kliknutí na odkaz pošle ze zdrojové jednotky do cílové. Odkaz může obsahovat tolik parametrů, kolik je požadováno cílovou jednotkou. Parametrický selektor je v podstatě jednotkový selektor, jehož podmínky deklarují jeden a více parametrů. Na ilustraci 4 vidíme, že odkaz obsahuje parametr (AkuálníInterpret) reprezentující primární klíč Interpreta vybraného ze seznamu a datová

jednotka má selektor [OID=AktuálníInterpret], který využívá parametr odkazu, aby získal a zobrazil informace o konkrétním interpretovi.



Ilustrace 4 **Mezi-stránkový kontextový odkaz**

Parametr odkazu má jméno a popisek oddělený středníkem. Jméno je návrhářem zvolený řetězec, který může odkazovat na parametrický selektor cílové jednotky. Popisek označuje obsah parametru odkazu, což je buď atribut nebo pole cílové jednotky. Jestliže se popisek vztahuje k atributu, zřetězíme entitu a jména atributů, kde entity a atributy mezi sebou oddělíme tečkou. Jméno entity může být vynecháno, pokud je z kontextu jasné, jako v našem příkladě, kdy popisek OID zastupuje entitu Interpret. Možné odchozí hodnoty zdrojových jednotek a popisků parametrických odkazů jsou shrnuty v tabulce 2.

Tabulka 2 Parametr dodaný odkazu ze zdrojové jednoty

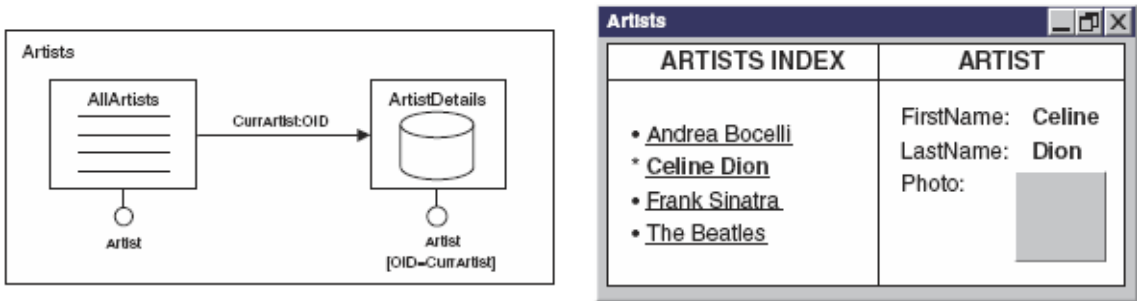
Zdrojová jednotka	Parametr/y předané odkazu	Popisky parametrických odkazů
Datová jednotka	Jakýkoliv atribut (včetně OID) položky zobrazené jednotkou.	ZdrojováEntita.JménoAtributu
Vícečetná datová jednotka	Skupina hodnot jakéhokoliv atributu (včetně OID) položek zobrazených jednotkou.	{ZdrojováEntita.JménoAtributu}
Indexová jednotka	Hodnota jakéhokoliv atributu (včetně OID) položky, vybrané kliknutím uživatele.	ZdrojováEntita.JménoAtributu
Hierarchická indexová jednotka	Hodnota jakéhokoliv atributu (včetně OID) položek vybraných klikáním uživatele. Mohou být vybrány všechny zobrazené položky, ze všech úrovní hierarchie.	ZdrojováEntita.JménoAtributu
Více-výběrová indexová	Skupina hodnot atributu (včetně OID) několika položek zvolených pomocí	{ZdrojováEntita.JménoAtributu}

jednotka	zaškrtávacích polí.	
Rolovací jednotka	Skupina hodnot jakéhokoliv atributu (včetně OID) skupiny položek	{ZdrojováEntita.JménoAtributu}
Vstupní jednotka	Hodnota vložená uživatelem do každého pole.	JménoPole

Textová specifikace parametru odkazu vypadá takto:

```
link ToIntepretiDetaily
(from VšichniInterpreti to InterpretiDetaily;
parametres AktuálníIntepret:OID)
```

Ilustrace 5 ukazuje vnitro-stránkový kontextový odkaz, který uživateli umožňuje vybrat si jednu položku, čímž aktualizuje obsah stránky požadovanou informací. Parametr odkazu i v tomto případě umožňuje přenos kontextové informace ze zdrojové indexové jednotky do cílové datové jednotky. Tato vazba mezi jednotkami je vyjádřena popiskem parametrického odkazu AktuálníInterpret a parametrickým selektorem datové jednotky (OID=AktuálníIntepret), která předaný parametr využije.

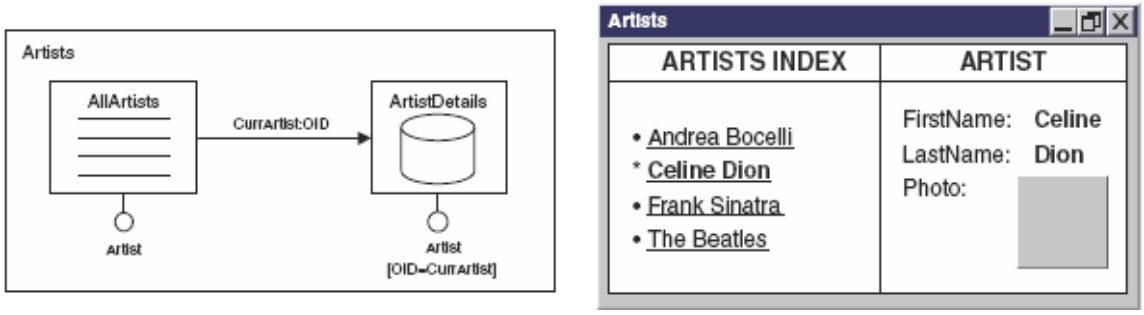


Ilustrace 5 Vnitro-stránkový kontextový odkaz s přidruženým parametrem

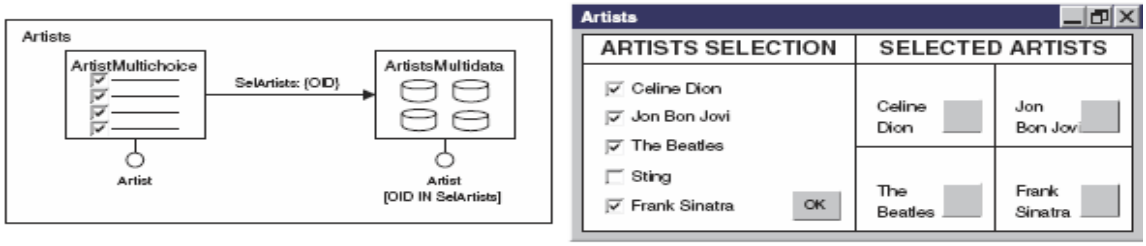
Vnitro-stránkové nekontextuální odkazy jsou sice méně časté, ale využívají se, jak si v jedné z dalších kapitol ukážeme, k propojení dvou různorodých vnořených pod-stran na jedné stránce.

Parametry odkazů mohou mít jednu nebo více hodnot. Vícečetné jednotky typicky předávají skupinu položek, jako např. OID vybraných interpretů. Syntaxe je patrná z tabulky 2, pokud jednotka předá odkazu více položek, předávaný atribut se umístí mezi vlnité závorky. Grafickou reprezentaci tohoto případu ukazuje ilustrace 7. Textově se odkaz vyjádří takto:

```
link ToInterpretiDetaily
(from InterpretiVíceVýběr to InterpretiVíceData;
parameters VybraníInterpreti:{OID})
```



Ilustrace 6 Předání více položek parametru odkazu

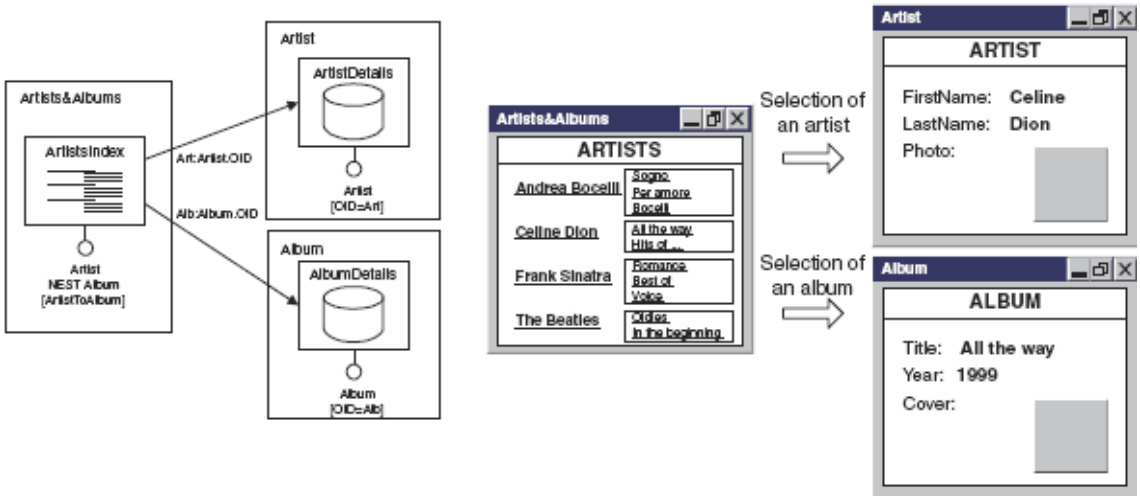


Ilustrace 7 Vice výběrový index a předávaný parametr

A popisuje situaci kdy si uživatel z více-výběrové jednotky se jménem InterpretiVíceVýběr vybere skupinu interpretů, jejichž podrobnosti se mu poté zobrazí pomocí vícečetné datové jednotky se jménem InterpretiVíceData. Jednotky si mezi sebou předávají OID interpretů. Přenos informací ze zdrojové do cílové jednotky, je umožněn jednak parametrem odkazu VybraníInterpreti:{OID}, který definuje jaký atribut ze zdrojové jednotky příjme. A zadruhé je umožněn parametrem selektoru [OID IN VybraníInterpreti] cílové vícečetné datové jednotky, který definuje jaký atribut příjme od odkazu.

Jednotky také mohou předávat parametry více odkazům. Ukážeme si to na příkladě hierarchické indexové jednotky pojmenované IndexInterpretů, kde nejvyšší úroveň hierarchie zobrazuje jednotlivé interprety a nižší jejich alba. Entity Interpret a Album jsou spojené vztahem InterpretToAlbum. Hierarchická jednotka je napojena dvěma odkazy k datovým jednotkám, umístěným na jiné stránce. Odkaz napojený na datovou jednotku InterpretDetaily má parametr Interpret.OID, kterým odkazuje na OID z nejvyšší úrovně hierarchie zdrojové jednotky, tedy k atributu OID entity Interpret. Odkaz napojený na datovou jednotku

AlbumDetaily má parametr Album.OID, kterým odkazuje k OID druhé úrovně hierarchie zdrojové jednotky, tedy k atributu OID entity Album. Ať si uživatel vybere album či interpreta, vždy se mu zobrazí žádaná stránka.



Ilustrace 8 Hierarchický index se dvěma odkazy s odlišnými parametry.

Aby diagramy vypadaly srozumitelněji, můžeme je zjednodušit. Pokud je z kontextu jasné, jaké má odkaz parametry, můžeme je prostě vynechat. Je to možné, neboť každá zdrojová jednotka má nastavené výchozí chování, jež určuje, jaký parametr předává odkazu. V tabulce 3 jsou popsány výchozí parametry, jež daný druh jednotky předává odkazu.

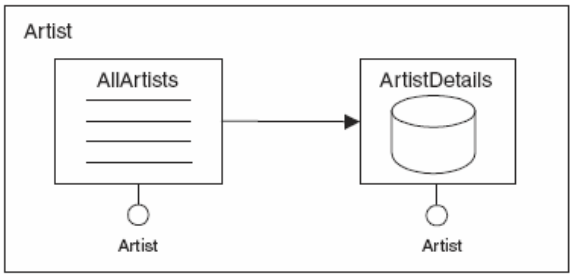
Tabulka 3: Výchozí parametry předávané odkazům

Zdrojová jednotka	Výchozí parametr jež se předává odkazu
Datová jednotka	OID zobrazené položky
Vícečetná datová jednotka	OID zobrazených položek
Indexová jednotka	OID uživatelem vybrané položky
Hierarchická datová jednotka	OID položky na nejvyšší úrovni vybrané uživatelem. Položky z nižších úrovní je nutné uvést
Více-výběrová indexová jednotka	OID uživatelem vybraných položek
Rolovací jednotka	OID vybraného úseku položek, nebo jediné OID pokud je velikost úseku nastavena na 1.

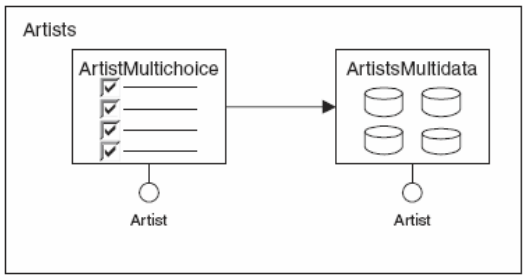
Stejně je i možné zjednodušit parametry selektoru. Tabulka 4 popisuje výchozí chování selektorů, pokud do cílové jednotky vede odkaz. Díky výchozím nastavením je diagram na ilustraci 6 shodný s diagramem na ilustraci 9 a digram 7 shodný s diagramem 10.

Tabulka 4: Výchozí chování jednotky bez selektoru, pokud do ní vede odkaz .

Cílová jednotka	Výchozí natavení selektoru pokud je
Datová jednotka	OID = <OID položky jež obdrží z odkazu>
Vícečetná datová jednotka, více-výběrová	OID IN <{OID} položek jež obdrží z odkazu>
indexová a rolovací jednotka	
Hierarchická indexová jednotka	OID IN <{OID} položek jež obdrží z odkazu> Výchozí selektor je definován jen pro v hierarchii nejvýše položenou entitu



Ilustrace 9 Zjednodušený zápis bez parametru.



Ilustrace 10 Zjednodušený zápis bez selektoru.

Předání hodnoty ze vstupní jednotky do cílové jednotky parametrem odkazu je stejné jako v předchozích případech. Ilustrace graficky zobrazuje tuto situaci. Vstupní jednotka KlíčovéSlovoVstup obsahuje jedno vstupní pole nazvané TitulKlíčovéslovo (což grafická reprezentace nezobrazuje), pomocí nějž může uživatel vyhledávat názvy Alb. Jakmile do pole naťuká uživatel svůj požadavek, odkaz jej předá jako popisek parametru selektoru cílové jednotky.[21] Tím se uživateli zobrazí alba podle jeho zadání. Textový popis je následovný:

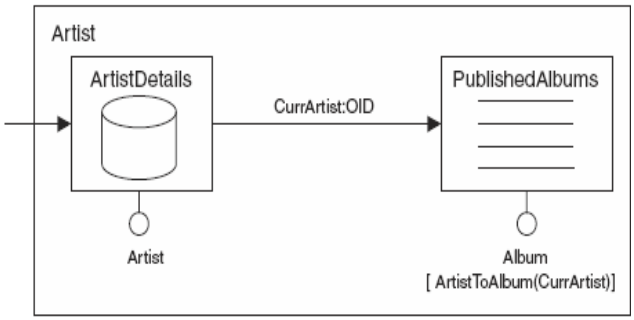
```
EntryUnit KlíčovéSlovoVstup
(fields TitulKlíčovéslovo String)

link VstupToAlbumyData
(from KlíčovéSlovoVstup to VstupToAlbumyData;
parameters Klíčovéslovo:TitulKlíčovéslovo)
```

21 Parametr jsem nazval Klíčovéslovo, jak bude patrné z textové notace.

Velice užitečné využití parametrických selektorů nastává, když jedna jednotka musí zobrazit všechny položky entity, které jsou ve vztahu k položkám jiné entity. V takovém případě může být selektor nastaven, aby získal položky zdrojové entity napojené určitým vztahem. Ilustrace 11 ukazuje příklad datové jednotky InterpretDetaily, která má za zdroj dat entitu Interpret a indexovou jednotku VydanáAlba, získávající data z entity Album. Smyslem je ukázat skutečnost, že se v indexové jednotce zobrazí jen alba vybraného intrepreta. Abychom toho dosáhli, indexové jednotce deklarujeme selektor [IntepretToAlbum(AktuálníInterpret)], tím odkazujeme na vztah mezi entitami Interpret a Album. Takto selektor omezí indexovou jednotku jen na alba, kde se OID interpreta shoduje s OID vyslaným z datové jednotky. Textová definice Vydaných Alb vypadá takto:

```
IndexUnit VydanáAlba
(source Album;
selector IntepretToAlbum(AktuálníInterpret);
attributes Titul;
orderby Titul)
```

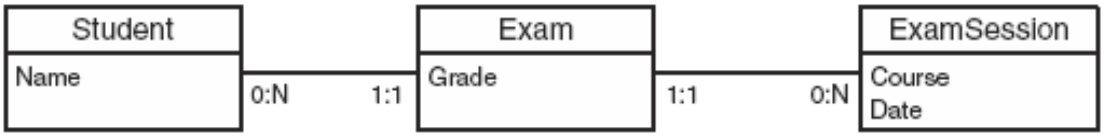


Ilustrace 11 Kontextový odkaz dodává kontext selektoru.

S parametry a selektory můžeme vyjádřit i komplexnější datové vztahy. Na datovém schématu v ilustraci vidíme entitu Hodnocení, kde každá položka atributu Známka je binárním vztahem napojena na jeden záznam z entity Student a na jeden záznam z entity Zkouška. Pro určený pár položek Student a Zkouška je možné vybrat přesně jedno hodnocení a zobrazit tak známku získanou studentem pří konkrétní zkoušce.

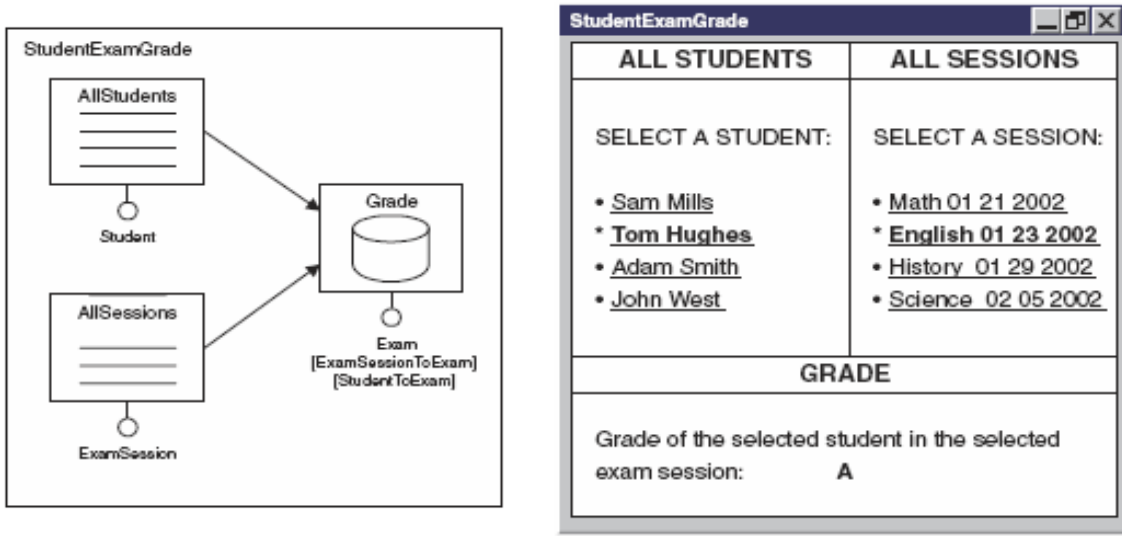
Na ilustraci máme dvě indexové jednotky VšichniStudenti a VšechnyZkoušky napojené na datovou jednotku Známka. OID vybraného studenta či zkoušky se parametry dvou odkazů

předá dvěma selektorům datové jednotky Známka. Selektory StudentToHodnocení ZkouškaToHodnocení umožňují získat známku vybraného studenta při vybrané zkoušce. Jakmile si uživatele z obou seznamů vybere položku, datová jednotka získá obě OID potřebné pro zobrazení studentovy známky v dané zkoušce.



Ilustrace 12 Datové schéma popisující zkoušky studentů

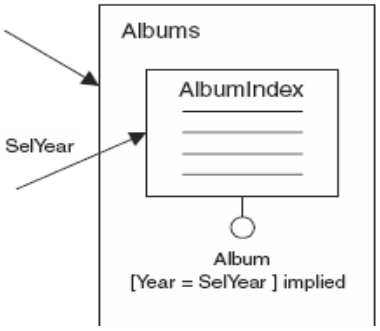
Selektor může být před značen výrazem *implied*, čímž vyjádří, že podmínka je nepovinná. V takovém případě pokud odkaz nepředá selektoru žádný parametr, selektor se chová jakoby žádnou podmínku neměl. Takový případ nastává ze dvou důvodů. Zaprvé odkaz nepřichází ze stránky, jež by požadovala zobrazení konkrétního obsahu, takže odkaz neobsahuje žádný



Ilustrace 13 Komplexní selektor zobrazující známky studena při určité zkoušce

parametr. Zadruhé odkaz nese parametr, který je určen pro jinou cílovou jednotku.

Graficky se nepovinný selektor zapisuje jak je vidět v ilustraci 14. V tomto příkladě když kontextový odkaz předá parametr VybranýRok selektoru indexové jednotky SeznamAlb, zobrazí se alba jen z požadovaného roku. Pokud dorazí nekontextový odkaz, který neobsahuje žádný parametr, jednotka zobrazí všechny alba.



Ilustrace 14 Nepovinný selektor

6.1.3.3. Automatické a transportní odkazy

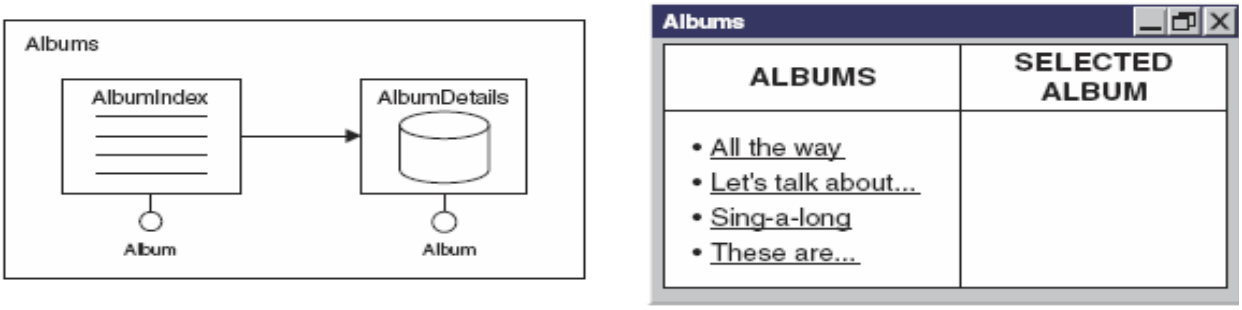
Doposud jsme vytvářeli modely, které vyžadovaly interakci ze strany uživatele. Když si uživatel zobrazí alba nějakého interpreta, spatří před sebou jejich seznam, ale o žádném albu nezíská podrobnější informace, dokud na něj neklikne. Někdy je ale užitečné, aby se již při načtení stránky zobrazily detailní informace o některé z položek. Kdy si uživatel nechá zobrazit seznam alb, bez jeho interakce se zobrazí podrobnosti například o první položce v seznamu. Takovéto chovaní aplikace se modeluje pomocí *automatických odkazů*, jež automaticky iniciují odkaz při načtení stránky, bez interakce ze strany uživatele.

Ilustrace 15 ukazuje příklad bez použití automatického odkazu a ilustrace 16 s ním. Jakmile je stránka s Alby navštívena, zobrazí nejen indexovou jednotku SeznamAlb, ale i datovou jednotku AlbumDetaily s podrobnostmi o albu. Které album stránka zobrazí záleží na okolnostech, jak je uspořádaný seznam alb klauzulí orderby, definovanou ve vlastnostech indexové jednotky SeznamAlb.

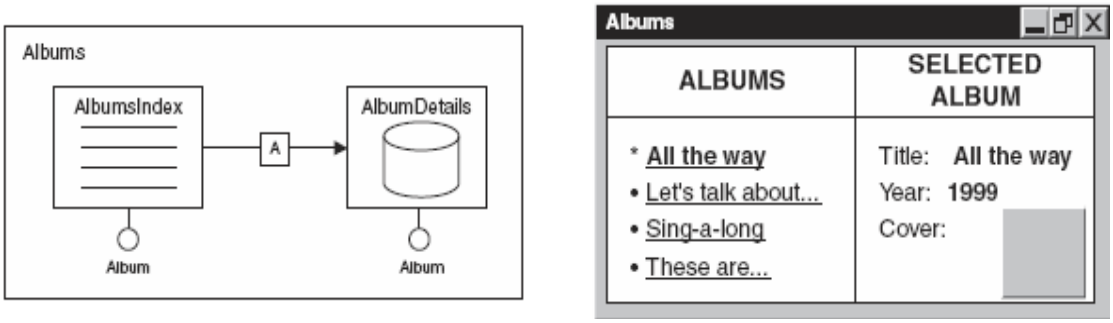
Grafické zobrazení automatických odkazů je zobrazeno na ilustraci 16. Odkaz je prostě doplněn velkým „A“. V textové notaci přidáme automatic do deklace odkazu:

```
link SeznamAlbToAlbumDetaily automatic
```

Doteď jsme hovořili jen o odkazech přístupných uživateli, tedy ty, na něž může kliknout. Jsou ale odkazy, které fungují pouze a jedině k předávání parametrů a které se proto uživateli nezobrazují. Nazývají se *transportní odkazy*, čímž dávají najevo, že neslouží k navigaci uživatele ale k přenosu parametrů.



Ilustrace 15 Normální odkaz

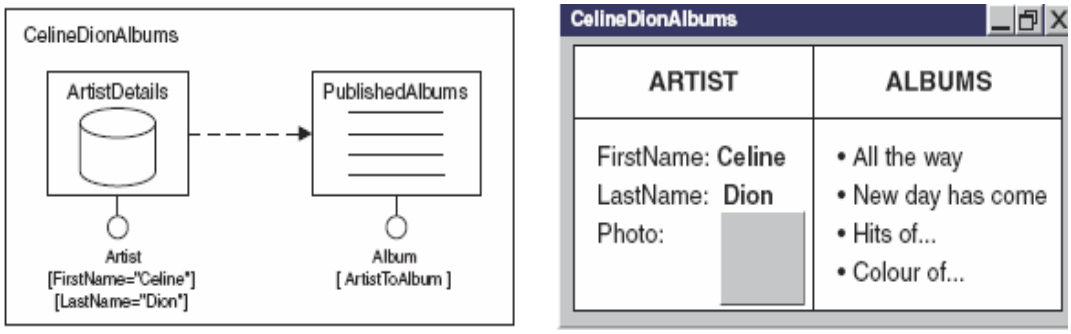


Ilustrace 16 Automatický odkaz

Ilustrace 17 představuje příklad transportního odkazu. Stránka CelineDionAlba obsahuje datovou jednotku InterpretDetaily, která zobrazuje podrobnosti o interpretovy zvaném Lou Reed a indexovou jednotku VydanáAlba, která zobrazuje alba jež vydal. Protože je odkaz definovaný jako transportní, tak jakmile uživatel vstoupí na stránku, datová i indexová jednotka načtou data, aniž by uživateli zobrazila odkaz.

Graficky je transportní odkaz znázorněn přerušovanou šipkou a v textové notaci se deklarace odkazu doplní o slovo transport.

```
link IntrepretDetailyToVydanáAlba
```



Ilustrace 17 Transportní odkaz

6.1.4. Globální parametry

V předchozích kapitolách se kontextové informace potřebné k zobrazení žádaných informací z jednotek přenášely pomocí odkazů, jež vedly z jednoho bodu hypertextu do jiného. Nicméně jsou situace, kdy je potřeba, aby kontextová informace nebyla přenášena z bodu do bodu, ale byla dostupná všem stránkám jako trvalý parametr. Nejtypičtějším příkladem jsou webové aplikace, jejichž obsah je dostupný v několika světových jazycích. Uživatel si vybere jazyk v jakém chce stránky prohlížet a zobrazovaný obsah bude pro něj nadále zobrazen v požadovaném jazyce.

WebML proto umožňuje deklarovat *globální parametr* pro uchovávání informací, dostupný všem stránkám. Globální parametr je informace v podobě buď OID nějaké položky, nebo hodnota dodané během pohybu hypertextem. V obou případech slouží k výpočtu zobrazení jednotek během další navigace. Globální parametr je vázaný na uživatelské sezení[22], pokud tedy dva uživatelé procházejí stejnou stránku umožňující zobrazit obsah ve více jazycích, každý jí uvidí ve své preferované jazykové mutaci. Neboť každý uživatel může mít nastaven vlastní globální parametr.

Deklarace globálního parametru začíná pojmenováním parametru návrhářem, určením typu informace, který bude globální parametr obsahovat a případnou výchozí hodnotou globálního parametru. Jako příklad poslouží globální parametr pro uchování informace o zemi, z níž uživatel přistupuje. Buď může aplikace globální parametr získat z hodnoty OID nějaké položky v entitě Státy, jak ukazuje první textový příklad. Nebo může jednoduše obsahovat řetězec jako v druhém příkladě. V takovém případě je navíc možné zadat výchozí hodnotu, například „Etiopie“

```
globalParameter CountryOID
(type OID;
entity Country)
```

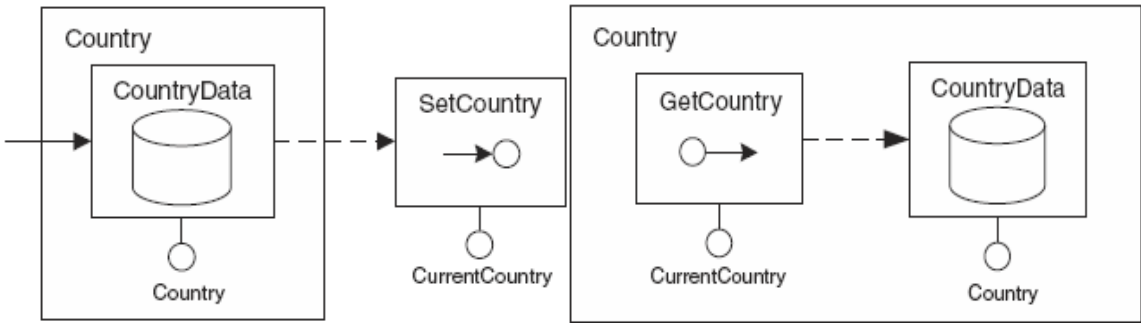
22 Neboli user session se z implementačního hlediska nejčastěji realizují pomocí http cookie či GET parametrů. Jedná se ve zkratce o technologie, jež umožňují http serveru rozlišit uživatele podle webového prohlížeče.

```
globalParameter CountryName
(type string;
initialValue „Etiopie“)
```

Hodnota se globálnímu parametru dodává pomocí pro tento účel vytvořené *set jednotky.*[23] Set jednotka přijímá parametr jen od jediného odkazu, který obsahuje hodnotu, jež se má předat globálními parametru. Jelikož má parametr v kontextu stránek globální působnost, zobrazuje se odkaz na každé stránce, a proto se i v grafické notaci set jednotka umisťuje mimo prostor stránky. Obyčejně se set jednotce dodává parametr transportním odkazem, aby nebylo třeba uživatelovy intervence. Ilustrace 18 představuje tento případ. Set jednotka ukládá do globálního parametru AktuálníStát hodnotu získanou transportním odkazem z datové jednotky. Jakmile je stránka stát navštívena, zobrazí se obsah datové jednotky StátData a OID země je oznámeno set jednotce transportním odkazem. Tak je uložena hodnota do globálního parametru AktuálníStát. Protože je odkaz k set jednotce transportní, není potřeba žádného klikatelného odkazu a globální parametr je nastavený bez zásahu uživatele.

```
link StátDataToSetStát transport
(from ZeměData to SetStát)

setUnit SetStát
(parameter AktuálníStát)
```



Ilustrace 18 set jednotka

Ilustrace 19 Nepovinný selektor

Pokud chceme do globálního parametru místo OID uložit jméno entity Stát, stačí v textovém popisu deklarovat odkazu parametr `Jméno:Země.Jméno.` Grafické vyjádření zůstává stejné.

23 Nenašel jsem pro set jednotky a za chvíli zmíněné get jednotky žádný použitelný český výraz.

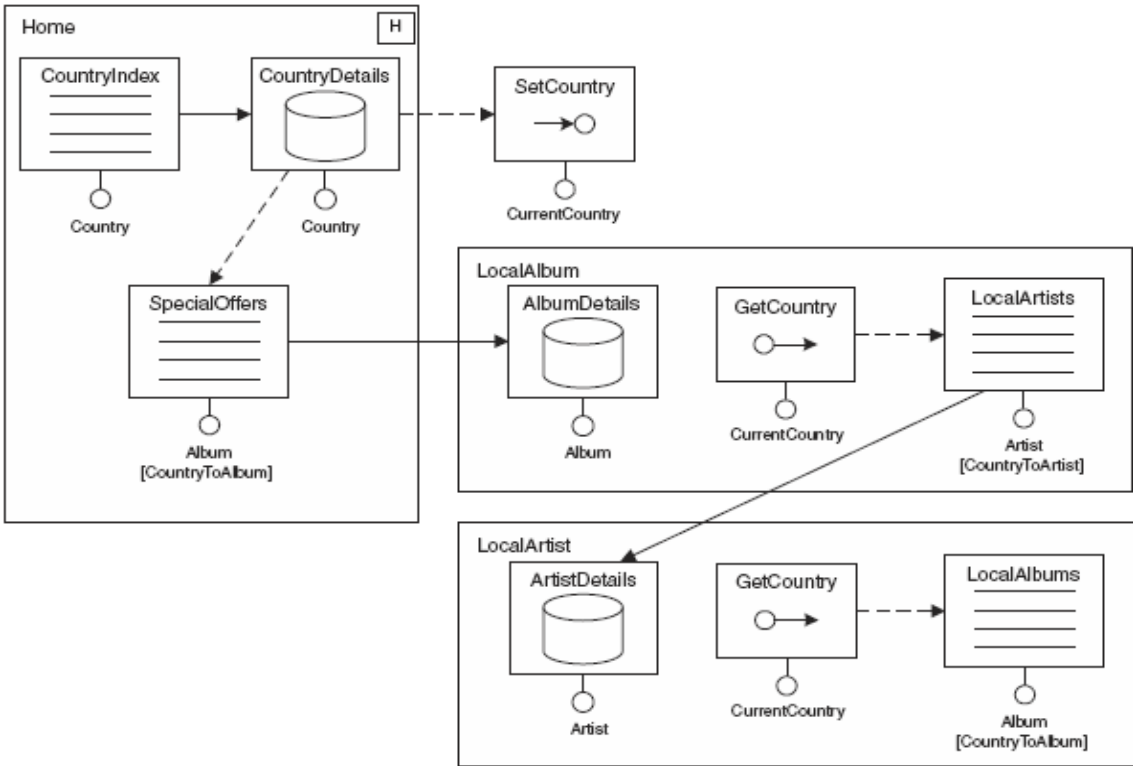
Globální parametr načteme pomocí *get jednotky*, jež přenáší informace jednosměrně pomocí odchozího odkazu. Get jednotka se zakresluje do stránek, kde je potřeba zobrazit obsah nějaké jednotky v závislosti na nastavení globálního parametru. Ilustrace 19 zobrazuje globální parametr AktuálníZemě obsahující OID země nastavené uživatelem. Datová jednotka ZeměData získá z příchozího automatického odkazu OID a díky výchozímu nastavení selektoru zobrazí informace o dané zemi. Textově se get jednotka s odkazem definuje takto:

```
getUnit GetZemě
(parametr AktuálníZemě)

link GetZeměToZeměData transport
(from GetZemě to ZeměData)
```

Ilustrace 20 ukazuje komplexní příklad použití globálních parametrů. Domovská stránka se skládá z indexové jednotky, umožňující výběr země (SeznamZemí); datové jednotky zobrazující podrobnosti o vybrané zemi (ZeměPodrobně) a ze seznamu alb dané provenience (SpeciálníNabídky), získaných selektorem ze vztahu ZeměToAlbum, který spojuje každý stát s lokální nabídkou alb. Jakmile si uživatel na domovské stránce vybere jednu zemi, předá se OID země nejen jednotce ZeměPodrobně ale v důsledku transportních odkazů vedených z jednotky ZeměPodrobě, je OID postoupeno globálnímu parametru (přes jednotku SetZemě)a indexové jednotce SpeciálníNabídky, jež zobrazí speciální nabídky pro vybranou zem. Uživatel si vybereme album ze speciální nabídky pro určitou zem a kontextovým odkazem je přenesen na stránku MístníAlbum. Kde se nachází datová jednotky AlbumPodrobně. Která, jak jsme zvyklí, zobrazí podrobnosti o vybraném albu, podle OID dodaného z indexové jednotky SpeciálníNabídky. Na stránce se ale také nachází indexová jednotka MístníInterpreti, kde zobrazený obsah je určen pomocí globálního parametru AktuálníZemě, jež uživatel nastavil na domovské stránce aplikace. Globální parametr dodaný z jednotky GetZemě předá indexové jednotce MístníAlba hodnotu OID, jíž selektor využije aby získal položky spojené vztahem ZeměToInterpret. Tak se uživateli zobrazí nabídka interpretů pro danou zemi. Výběrem interpreta se uživatel kontextovým odkazem přenese na stránku MístníIntepret, kde se o něm zobrazí informace v datové jednotce InterpretPodrobně, opět pomocí implicitně předaného OID mezi jednotkami. Stránka obsahuje také obdobu indexové jednotky z předchozí stránky

nazvanou MístníAlba, jež seznam alb pro danou zemi vytvoří podle hodnoty nastavené v globálním parametru, tím že ji předá selektoru jednotky MístníAlba, který získá položky spojené vztahem ZeměToAlbum odpovídající nastavení země na domovské stránce.



Ilustrace 20 Komplexní schéma s globálními paramety

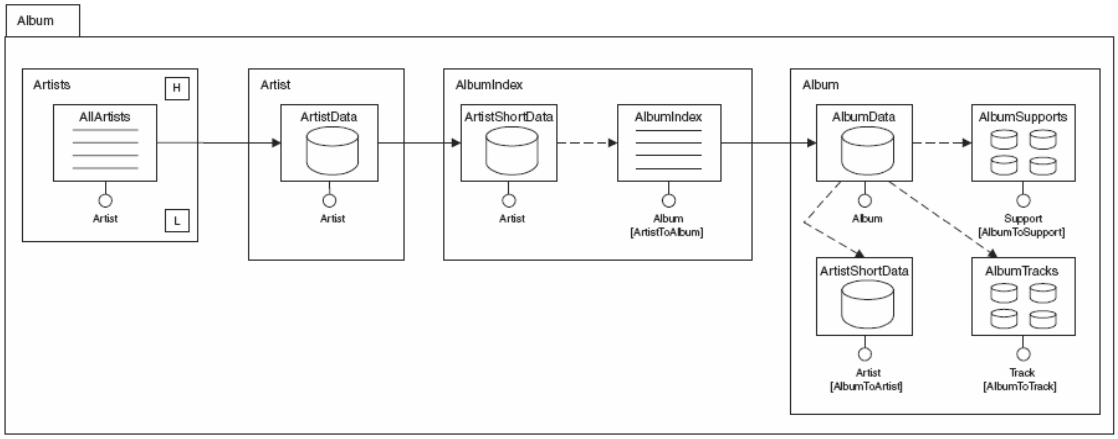
Jak předvedl tento příklad, globální parametry představují navigační paměť, jež ovlivňuje zobrazované informace v kontextu celé aplikace.

6.1.5. Organizace hypertextu

Ze stránek a jednotek můžeme vytvořit model jen velmi nekomplikované webové aplikace. Při navrhování rozsáhlé a složité hypertextové struktury je potřeba stránky sdružovat do větších celků, jež nám umožňují lépe modelovat a organizovat chování aplikace. Mezi tyto celky patří např. webový pohled, oblast či vnořené stránky. Tyto celky jsou mezi v hierarchické postavení, kdy nejvyšší místo zastává webová perspektiva.

6.1.6. Webové pohledy

Webový pohled je nejvýše postavený celek webové aplikace a zahrnuje všechny nižší struktury. Ilustrace 21 ukazuje grafické zobrazení webového pohledu. Pohled je definován jménem a obsahuje skupiny stránek nebo oblastí (stránky a oblasti se nevylučují). V našem příkladu pohled Album obsahuje stránky Interpreti, Interpret, SeznamAlb a Album. Uživatel si z první stránky vybere interpreta, jehož podrobné informace se zobrazí na stránce Interpret.



Ilustrace 21: Webový pohled

Uživatel si nechá zobrazit alba, čímž se objeví na stránce SeznamAlb, kde vedle diskografie vidí navíc zkrácenou variantu podrobností o Interpretovi. Jakmile si uživatel z diskografie vybere konkrétní album, ocitne se na stránce Album, která zobrazuje informace o konkrétním albu, stopách na něm, ostatních hudebnících a zkrácené informace o interpretovi. Textová notace je následovná:

```
siteview Album
        (pages Interpreti, SeznamAlb, Album, Intepret)
```

6.1.7. Oblasti, výchozí stránky a home pages

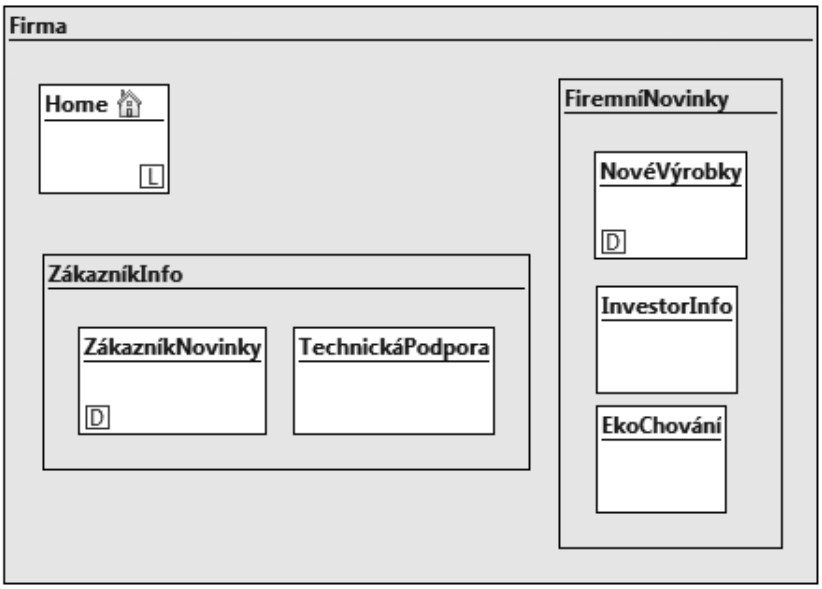
Mnoho skutečných webových aplikací je uspořádáno hierarchicky, stránky jsou shluknuty do sekcí, jež sdružují obdobný obsah. WebML nabízí struktury pro zlepšení organizace pohledů a stránek: *oblasti, výchozí stránky* a *home pages.*

Oblasti jsou schránky na stránky nebo další oblasti, zvané podoblasti. Většina webových aplikací je rozdělena do oblastí (například produkty, podpora, kontakt atd.), kde každá oblast sdružuje stránky na své téma. Ilustrace 22 zobrazuje obvyklý příklad strukturování pohledu do dvouvrstvé hierarchie. Pohled obsahuje několik oblastí, kde každá oblast sdružuje stránky na určité téma - oblast ZákazníkInfo tak druží stránky Kontakt a TechnickáPodpora.

Odkazy můžeme mezi jednotkami a stránkami zakreslovat bez ohledů na oblasti. Odkaz prostě značí že přecházíme z jedné oblasti do druhé.

Jestliže pohled sdružuje jak stránky tak oblasti, tak v textovém zápisu popíšeme stránky a oblasti z nejvyšší hierarchie a poté rozepíšeme nižší úroveň. Textový zápis Ilustrace 22 je tedy:

```
siteview Firma
(areas FiremníNovinky, ZákazníkInfo;
        pages Home)
area FirmaNovinky
        (page NovéVýrobky, InvestorInfo, EkoChování)
area ZákazníkInfo
        (page ZákazníkNovinky, TechnickáPodpora)
```



Ilustrace 22 Rozčlenění pohledu do dvou vrstvé hierarchie.

Stránky a oblasti mohou být charakterizovány několika vlastnostmi, které upozorňují na jejich důležitost. Přesněji mohou nabývat tří vlastností.

Home page neboli domovská stránka určuje výchozí stránku aplikace, nebo první stránku jež se uživateli zobrazí po přihlášení. Domovská stránka musí být jedinečná pro webový pohled. Graficky se taková stránka znázorňuje pomocí „H“, v aplikaci Webratio ikonkou domečku. Textově se deklaruje přidáním slova home na konec deklarace stránky.

Výchozí stránka[24] je první stránka, kterou uvidí uživatel, pokud vstoupí do určité oblasti. V rámci oblasti je jedinečná. Graficky se znázorňuje „D“ uvnitř stránky a textově se definuje přidáním slova default na konci deklarace stránky.

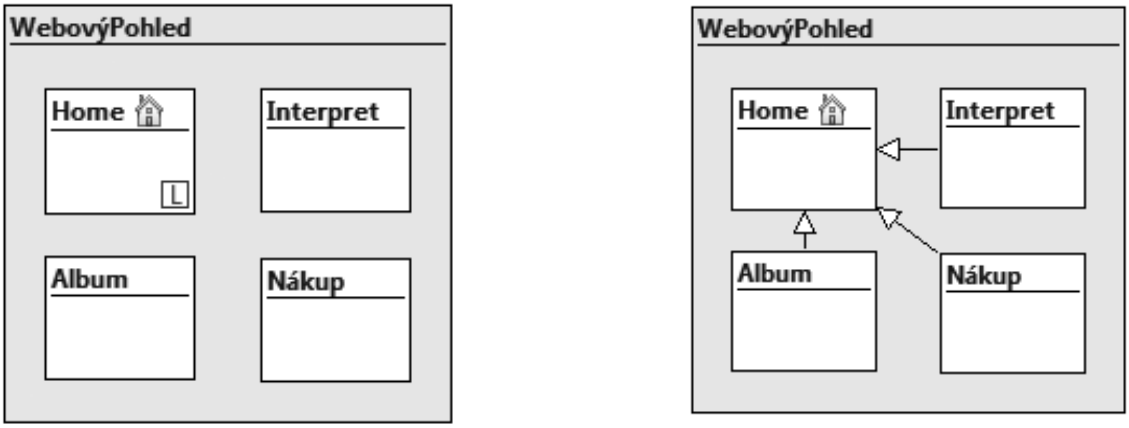
Orientační stránka[25] je stránka přístupná ze všech stránek nebo oblastí, které jsou součástí jí nadřazené struktury (pohled, či nadoblast). Graficky se vyjadřují pomocí „L“ vloženého do stránky a textově doplněním deklarace stránky o slovo landmark.

Také oblasti mohou být orientační a výchozí. Graficky i textově se deklarují stejně jako u stránek. Výchozí oblast je ta, do které se uživatel dostane, pokud vstoupí do jí nadřazené oblasti. Pokud uživatel klikne na odkaz, jež vede k oblasti nadřazené té výchozí, je přesměrován na výchozí stránku obsaženou ve výchozí oblasti.

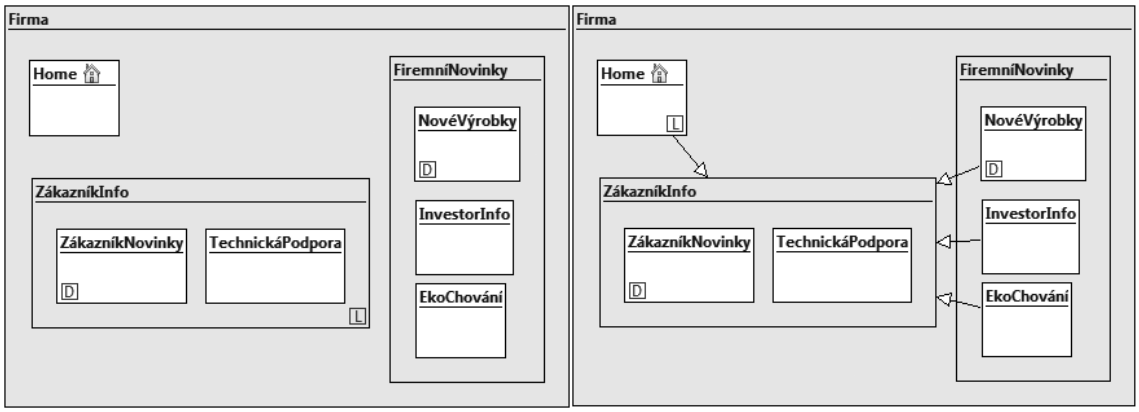
24 Původní pojem: Default page

25 Původní pojem: landmark

Orientační oblast je dostupná všem stránkám a oblastem, které jsou součástí jí nadřazené oblasti. Užitečnost orientační vlastnosti si zobrazíme na ilustraci 23 se dvěma identickými diagramy. Pokud deklarujeme orientační vlastnost tak jako na levém diagramu, nemusíme explicitně vyjadřovat odkazy směřující z ostatních stránek, jako na diagramu vpravo. Obdobný příklad nabízí ilustrace 24, kde levý diagram s nastavenou orientační vlastností pro oblast nemusí explicitně vyjadřovat odkazy z okolních stránek.



Ilustrace 23 Orientační stránka (vlevo) vs. explicitně vyjádřené odkazy (vpravo)



Ilustrace 24: Orientační oblast (vlevo) vs. explicitně vyjádřené odkazy (vpravo)

6.1.8. Vnořené stránky

Umožňují členit jednotky v rámci stránky, můžeme tak jednotky strukturovat do hierarchie a vytvářet podstránky. Vnořené[26] podstránky se dělí na pojící a vylučovací.[27]

Pojící vnořené stránky (či AND podstránky) se zobrazí vedle sebe, tak umožní rozdělit stránku do úseků, kde jedna je stálá a ostatní zobrazují různé informace, dle chování uživatele.

26 Původní pojen: nested pages

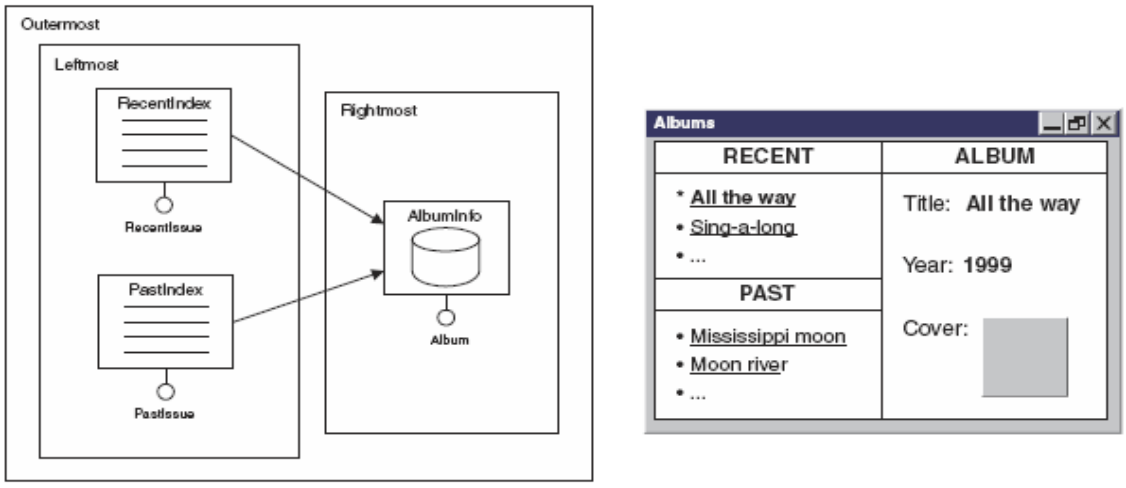
27 Původní pojmy: conjunctive and disjunctive pages

Ilustrace 25 obsahuje dvě AND podstránky pojmenované LevýOkraj, která zobrazuje dva seznamy s aktuálními a starými alby. A podstránku PravýOkraj zobrazující informace o vybraném albu, která se změní pokaždé, když si uživatel vybere album z levé podstránky. Textová reprezentace celé stránky pak vypadá takto:

```
page Nejzevnější
(and-pages LevýOkraj, PravýOkraj)

page LevýOkraj
(units NedávnéSeznam, MinuléSeznam)

page PravýOkraj
(units AlbumInfo)
```

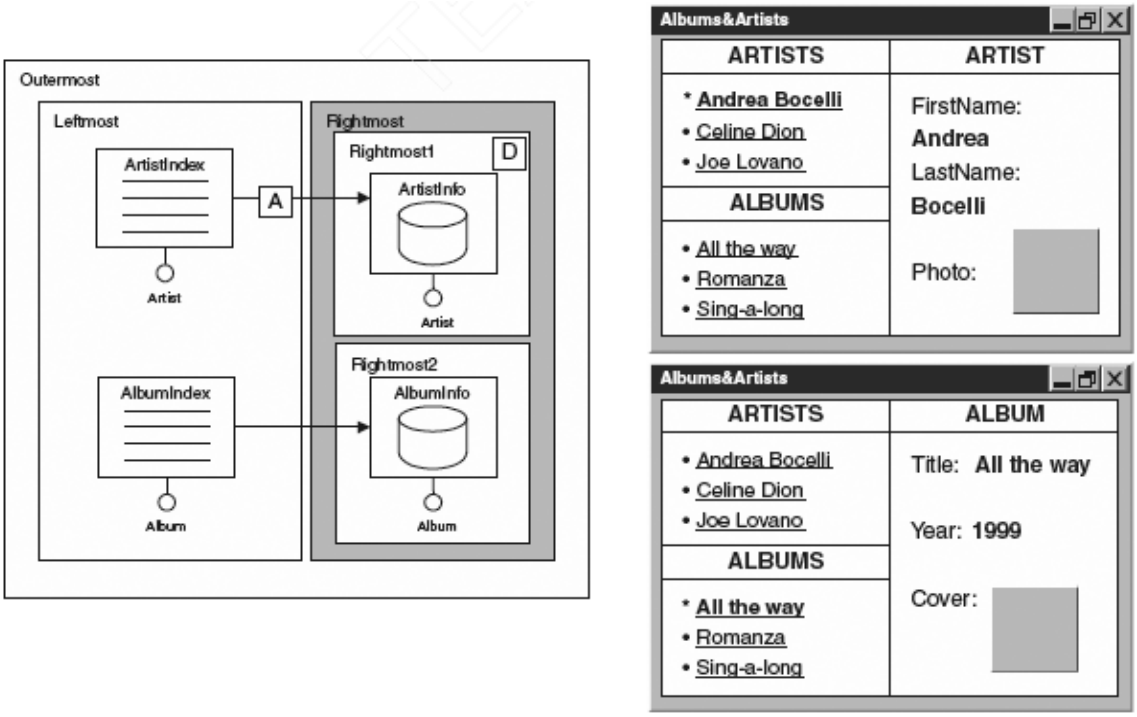


Ilustrace 25 Vnořené podstránky

Vylučovací vnořené stránky (či OR podstránky) vyjadřují, že část stránky může obsahovat různě nastavené jednotky, kdy se zobrazí buď jedna nebo druhá, podle uživatelovského výběru. Ilustrace 26 představuje stránku se seznamem alb a interpretů spolu s informací buď o konkrétním albu či interpretovi. Jakmile si uživatel vybere interpreta či album, změní se celá stránka a zobrazí v pravém okraji buď podrobnosti o interpretovi či albu. Jedna z OR podstránek může být označena jako *defualt OR sub-page,* čímž se při první načtení stránky, zobrazí právě takto označená podstránka. Na ilustraci je také vidět, že stránky obsahující OR podstránky mají odlišnou barevnou výplň. Textově se stránka PravýOkraj zobrazí takto:

```
page PravýOkraj
(or-pages PravýOkraj1 default, PravýOkraj2)

page PravýOkraj1(units IntepretPodrobnosti)
page PravýOkraj2(units AlbumPodrobnosti)
```



Ilustrace 26 Vnořené OR podstránky

WebML nabízí ještě pokročilejší modelovací techniky pro návrh redakčních systému a modelování práce s daty. Ale vzhledem k vymezenému prostoru se jim, již nemůžeme v tomto textu věnovat. Jsou však naznačeny na oficiálních stránkách WebML a podrobně rozebrány v knize Designing Data-Intesive Web Applications.

7. Závěr

Cílem práce bylo přinést čtenáři náhled na problematiku aplikace WebML v procesu vývoje datového a hypertextového modelu. Seznámil jsem čtenáře se základním rámcem pro pochopení úlohy modelování v našem světě. Stručně jsem uvedl historický původ WebML a první úspěšné projekty. Dále jsem se v textu naznačil problematiku vývoje webových aplikací a vymezil jsem metodologii WebML aplikovanou na různé aspekty vývoje aplikace. Zmínil jsem možnosti implementace WebML mimo CASE nástroj Webratio. Strukturu WebML jsem rozčlenil do tří základních modelů. Na datový, hypertextový a prezenční model. Prezenčnímu modelu jsem se věnoval jen okrajově. Datový model jsem uvedl na příkladech a největší prostor jsem věnoval, v intencích cílů své práce, hypertextovému modelu. Strukturoval jsem jej na nejmenší celky a ty postupně a na příkladech spojoval.

Do práce se nevešla problematika složitějšího hypertextového modelování, vzhledem k omezenému rozsahu. Ovšem stručné seznámení s metodikou návrhu hypertextového modelu je dobrým úvodem pro další studium pokročilých metod modelování ve WebML.

8. Seznam použité literatury

ALHIR, Sinan Si. *UML (Unified Modeling Language) in a nutshell: a desktop quick reference.* Sebastopol : O'Reilly & Associates, 1998. 273 s. ISBN 1-56592-448-7.

ANDRADE, A.R. De; MUNSON, E.V.; PIMENTEL, M.G.C. Engineering Web applications with XML and XSLT. In *WebMedia & LA-Web 2004 proceedings : 12-15 October 2004, Ribeirao Preto- SP, Brazil.* Los Alamitos : IEEE Computer Society, 2004, p. 86-93. Komerčně dostupný z WWW: <http://80.csdl.computer.org.dialog.cvut.cz/dl/proceedings/lawebmedia/2004/2237/00/22370086.pdf> ISBN: 0-76952-237-8

BLAHA, Michael, RUMBAUGH, James. Object-oriented modeling and design with UML. Upper Saddle River : Prentice Hall, 2005. 477 s. ISBN 0-13-196859-9.

BONGIO, O.; CERI, S.;FRATERNALI, P. aj. Modelling Data Entry and Operations in WebML In The World Wide Web and Databases: Third International Workshop WebDB 2000, Dallas, TX, *USA, May 2000. Selected Paper.* Berlin : Springer, 2005, s. 201-214. Dostupný komerčně z WWW: <http://www.springerlink.com/content/82rbf8hjefw8whn5/fulltext.pdf> ISSN 0302-9743.

BONIFATI, A.; CERI, S.; FRATERNALI P. aj. Building Multi-device, Content-CentricApplications Using WebML and the W3I3 Tool Suite. In *Conceptual Modeling for E-Business and* the Web: ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The World *Wide Web and Conceptual Modeling, Salt Lake City, Utah, USA, October 2000. Proceedings.* Berlin : Springer, 2000, s. 64-75. Dostupný komerčně z WWW: <http://www.springerlink.com/content/1m6dh3qad8ry1pxw/fulltext.pdf> ISBN 3-54041-073-2

BRAMBILLA, M.; CELINO, I.; CERI, S. aj. A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In *The Semantic Web - ISWC 2006: 5th* International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006. *Proceedings.* Berlin : Springer, 2006, s. 172-186. Dostupný komerčně z WWW: <http://www.springerlink.com/content/k0ptv6038u227710/fulltext.pdf> ISBN 3-54049-029-9

BRAMBILLA, M.; COMAI, S.; TZIVISKOU, C. Exception Management Within Web Applications Implementing Business Processes. *Advanced Topics in Exception Handling Techniques.* 2006, vol. 4119, s. 101-120. Dostupný komerčně z WWW: <http://www.springerlink.com/content/k416583k420521qx/fulltext.pdf> ISSN 0302-9743

CAMAI, S.; MATERA, M.; MAURINO A. A Model and an XSL Framework for Analysing the Quality of WebML Conceptual Schemas In *ER 2002 Workshops, ECDM, MobIMod, IWCMQ, and eCOMO, Tampere, Finland, October 7-11, 2002, Revised Papers.* Berlin : Springer, 2003, s. 339- 350. Dostupný komerčně z WWW: <http://www.springerlink.com/content/98a8tdpq3hn2bwkl/fulltext.pdf> ISBN 3-5402-025-

CERI, S. aj. Designing data-intensive Web applications. Amsterdam : Morgan Kaufmann, 2003. 561 s. ISBN 1-55860-843-5.

HERNANDEZ, Michael J. *Návrh databází*. Praha : Grada, 2006. 408 s. ISBN 80-247-0900-7.

MOLHANEC, Martin. WebML - objektově orientovaná metodika pro tvorbu webových sídel. [online]. 2003. 12 s. [cit. 2007-07-14]. Dostupný z WWW: <http://martin.feld.cvut.cz/~molhanec/VaV/files/publik/2003/WebML-CO.pdf>.

MORVILLE, P. *Ambient Findability: What We Find Changes Who We Become*. Sebastopol : O'Reilly, 2005. 204 s. ISBN 0-596-00765-5

ROSENFELD, L.; MORVILLE, P. *Information architecture for the World Wide Web*. 3nd ed. Sebastopol : O'Reilly, 2006. 461 s. ISBN 0-596-52734-9.

TIDWELL, *Jennifer. Designing Interfaces: Patterns for Effective Interaction Design*. Sebastopol : O'Reilly, 2005. 352 s. ISBN 0-596-00803-1

Unified Modeling Language : Infrastructure. Version 2.0, formal/05-07-05. S.l. : Object Management Group, March 2006. 206 s. Dostupné z WWW: <http://www.omg.org/>.

Unified Modeling Language : Superstructure. Version 2.0, formal/05-07-04. S.l. : Object Management Group, August 2005. 694 s. Dostupné z WWW: <http://www.omg.org/>.

ZELENKA, Petr. WebML - projektování webových aplikací. Interval [online]. 2003 [cit. 2007-07-12]. Dostupný z WWW: <http://interval.cz/clanky/webml-projektovani-webovych-aplikaci/>.

Přílohy

Přehled hypertextových prvků WebML, zdroj:

http://www.webml.org/webml/upload/ent17/1/webml_elements.pdf