Datové modelování, perzistence objektů, konceptuální a fyzický datový model.

Thursday, May 30, 2013 8:24 AM

Datové modelování

- jedna ze základních funkcí IS je ukládání a následné zpracování informací ve formě dat, proto se provádí při návrhu IS také datové modelování
- jeho úkolem je zvolit jaké objekty, jako nositele informace, potřebujeme ukládat do trvalé paměti a jaké jejich vlastnosti a vztahy mezi nimi chceme uchovávat
- při datovém modelování obvykle vytváříme konceptuální a fyzický datový model a také určujeme způsob perzistence objektů

- Cílem datového modelování je navrhnout „kvalitní“ datovou strukturu a databázový systém pro konkrétní IS.

Perzistence objektů

- zabývá se kam a jakým způsobem budou objekty trvale uloženy
- objekty se obvykle ukládají do relační databáze ve formě datových záznamů v tabulkách
- pro programový přístup do databáze se často používá standardizované softwarové API pro databáze jako ODBC nebo JDBC
- pro usnadnění programátorské práce při vytváření perzistentní vrstvy můžeme využít objektově-relační mapování, které nám zajistí automatickou transformaci ukládaných objektů do záznamů relační databáze - např. framework Hibernate pro Java aplikace

Konceptuální datový model

"Konceptuální modelování má své kořeny v počátcích datového modelování a úzce souvisí i například s teorií relačních dat. Jedná se o modelování reality prostřednictvím základních pojmů (konceptů) a jejich vzájemných souvislostí. Konceptuální modelování je dnes široce rozvinutý samostatný obor s propracovanými nástroji a metodami, jejichž principy se odráží i v pravidlech modelování tříd objektů pomocí UML."

- vyjadřuje jaké objekty a jejich atributy budeme ukládat a vztahy mezi nimi
- pro grafické vyjádření se používají ERA modely (diagramy)

Při datovém modelování vytváříme nejprve logický - konceptuální datový model. Konceptuální datový model představuje určité zobecnění oproti implementaci datové struktury v konkrétní relační databázi.

Konceptuální model – fáze návrhu, volby technologie, nezávislý na konkrétní databázi

Fyzický datový model

- odvozuje se z konceptuálního modelu a vyjadřuje navíc jak přesně budou data v konkrétní databázi uložena
- také se popisuje ERA modely, které obsahují i přesné databázové typy atributů tabulek

Zvolíme-li konkrétní databázi (např. Oracle) mluvíme o fyzickém datovém modelu, na který je konceptuální model převeden.

Fyzický model – fáze implementace pro konkrétní databázi, konkrétní implementace. Z fyzického modelu můžeme generovat SQL skripty, případně se napojit přímo na databázi. Na fyzické úrovni můžeme psát uložené procedury a triggery.
Dva přístupy:

- **Při tvorbě konceptuálního datového modelu vycházíme z diagramu analytických, resp. návrhových tříd s jasnou představou o požadavcích na perzistenci.Postup od objektové analýzy, přes návrhové třídy k implementaci.**

Primárně pracujeme s konceptuálním modelem, objektová nástavba je sekundární.

<u>ERA modely</u>

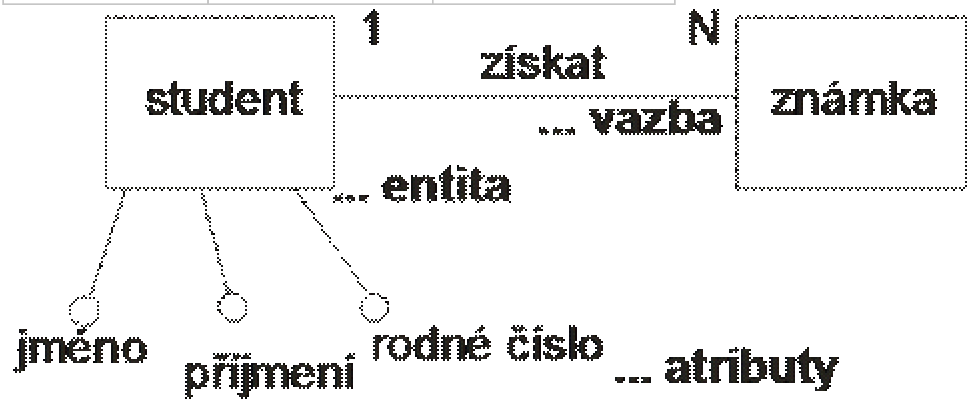
viz DB1 - vypsáno z DB1

Entita – model z reálného světa

Relation – podchycuje vztahy mezi entitami

ERA modely = modelová analýza

Datový model	Datová struktura	Souborový přístup
Entitní množina	Tabulka	Soubor
Entita	Řádka	Záznam
Atribut	Název sloupce	Položka



Vazby

1:N - vyjadřuje, že jedné entitě E1 může příslušet více entit z entitní množiny E2

jedné entitě z E2 přísluší jen jedna entita z E1

nejlépe 1:N(0)

př.: student – známka

1:1 - na vazbě se podílí jej jedna entita z E1 a jedna z E2

vždy se ptát, proč je to rozdělené, proč to není jedna entita

vazba je zajímavá, pokud alespoň jeden konec volný (nepovinný)

př.: student – známka (student nemusí mít známku)

N:N - jedné entitě z E1 přísluší více entit z E2

jedné entitě z E2 přísluší více entit z E1

př.: student – rozvrhová akce

ER modely

- primární – minimální množina atributů, která jednoznačně určuje entitu
- na vazbu se díváme jako na entitu – lze k ní přidat atributy (čtenář – výpůjčka – exemplář)
- vazba 1:N se při realizaci vytvoří tak, že do "podřízené" tabulky přenesu klíč (cizí klíč) z "nadřazené" tabulky
- vazba M:N nelze realizovat, nelze vyřešit pomocí cizích klíčů = musí se provést rozklad vazby
- mezi dvěma entitními množinami může existovat více vazeb
- vazba nemusí být binární, ale může být n-ární
- vazba může být i unární (sama na sebe)

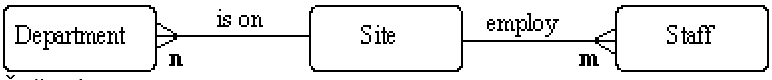
Některé datové modely rozlišují entitní množiny regulární a slabé

Slabá entitní množina je taková, u níž nelze určit, či nemůžeme zjistit nadřazenou množinu

FAN

Problém: 2 vazby 1:N se větví z jedné entity

datové položky ve dvou složkách nejsou v přímém vztahu, ale mají vazbu založenou na datových položkách ve třetí složce

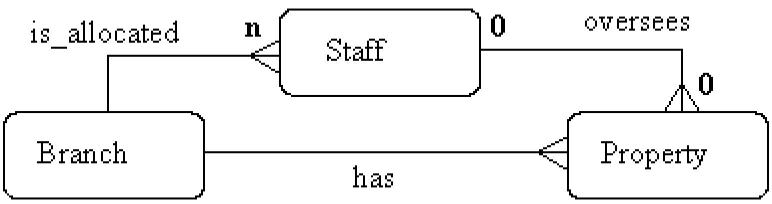


Řešení:

CHASM trap

Problém: Existuje vztah mezi entitami, ale chybí vazba.

Řešení:



Objektově relační mapování - Perzistence

Objektově relační mapování – O/R slouží k tomu, aby bylo možné snadno používat relační databáze v prostředí objektově orientovaných programovacích jazyků.
Vzhledem k tomu, že objektově orientovaný návrh dat není jednoznačně převoditelný na relační databáze a opačně, používají se různé formy mapování.
Mapování má za účel načítat data z relační databáze a naplnit jimi příslušné datové položky objektů včetně vazeb mezi objekty, případně naopak datové položky objektů ukládat do databáze.
Snahou ORM je co nejlepší využití obou zmíněných technologií

- objekty by měly reprezentovat objekty reálného světa, jak to požadují principy OOP
- na straně databáze bychom zase měli využít všech možností relačních databází – indexy, pohledy, primární klíče, triggery a uložené procedury.

Alternativou k O/R mapování je použití objektové databáze, která je navržená přímo pro ukládání objektů. Použití takové databáze eliminuje potřebu převádět data z objektové podoby do relační. Data jsou uložena přímo ve své objektové reprezentaci.
Objektové databáze zatím nejsou příliš rozšířené.
V současnosti je nejpoužívanějším nástrojem pro ORM produkt od firmy JBOSS Hibernate.
Hibernate je O/R mapovací nástroj pro jazyk Java. Jde o volně šiřitelný open source software.
Nabízí prostředí pro mapování objektového modelu na tradiční relační schéma.

Perzistentní třídy musí splňovat jisté vlastnosti, obsahovat

- konstruktor bez parametrů
- getter a setter metody pro perzistentní položky

Konstruktor bez parametrů Hibernate používá při načítání objektu z databáze. Jeho zavoláním v paměti vytvoří prázdný objekt, jehož položky následně nastaví podle hodnot uložených v databázi pomocí setter metod. Getter metody Hibernate používá při čtení položek při ukládání objektu do databáze.

Základní mapování - mapování tříd na tabulky

Perzistentní (entitní, bussines) třídy, resp. jejich instance - objekty odpovídají entitám konceptuálního datového modelu, resp. řádkům tabulek fyzického modelu. Atributy třídy se stanou sloupci tabulek.

Mapovací soubory pro Hibernate se píší v jazyce XML nebo se využívá tzv. anotací Javy (zápis metadat přímo do kódu). Každá třída se mapuje na jednu tabulku. Každá tabulka musí obsahovat primární klíč.
Mapovací soubor obsahuje výčtem elementů <property>, které reprezentují položky, které mají být ukládány a jak mají být ukládány.

Způsob uložení položek lze ovlivnit velkým množstvím atributů. Několik nejpoužívanějších popisuje následující výčet.

- column="column_name" - určuje název sloupce. Implicitně se používá název proměnné.
- type="typename" - určuje typ konverze mezi Java typem a SQL typem.
- not-null="true|false" určuje zda je možné do daného sloupce uložit NULL hodnotu.

Mapování atributů musí odpovídat konverzi datových typů, norma SQL – 92 definuje standardy datových typů (opakem jsou transientní třídy).

Mapování vztahu
Mapováním vztahů zajišťujeme, že se může mezi objektovým modelem a databází současně „přenášet" síť vzájemně provázaných – asociovaných objektů. Základní výhoda ORM
Při použití Hibernatu jako ORM vrstvy se vazby dělí ještě na jednosměrné a obousměrné. Rozdíl je v tom, že u jednosměrné vazby má referenci jen jedna entita. Druhá tedy žádnou referenci nemá, kdežto u obousměrné vazby mají reference obě entity.
Vztah se v mapovacím souboru mapuje pomocí jednoho z elementů

- <one-to-one>
- <one-to-many>
- <many-to-one>
- <many-to-many>

podle kardinality daného vztahu. Jediným povinným atributem je name, ostatní atributy jsou nepovinné. Nepovinné atributy jsou podobné jako u elementu <property>.

Mapování dědičnosti
Protože cílový fyzický datový model nepřipouští dědičnost tabulek, viz norma SQL 92, existují tři možnosti:

- mapování 1:1 – každá třída se mapuje do samostatné tabulky, jedna instance objektu je rozložena po více tabulkách, řada nevýhod.
- zahrnutí do nadtřídy – atributy podtříd jsou zahrnuty do nadtřídy, z třídy a jejich podtříd vnikne jedna tabulka. Vhodné v případě malého počtu podtříd.
- rozpuštění do podtříd – všechny atributy nadtřídy jsou přeneseny do tabulek pro všechny podtřídy. Počet tabulek odpovídá počtu podtříd. Vhodné pro velký počet podtříd.
  Viz CASE

From <https://d.docs.live.net/e3534876709763a3/Dokumenty/ZCU/Statnice/Statnice.docx>