VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Report model

Ing. et Ing. Ladislav Ruttkay

26.8.2008

Anotácia

Predmetom tejto práce sú základy problematiky vytvárania modelov reportov, ktoré sú vytvorené pri vizualizácií OLAP databáz. Práca postupne preberá jednotlivé podstatné časť vytvárania modelu, a následne prakticky uvádza vzorový príklad. Práca nadväzuje na prácu *RUTTKAY, L., Vizualizácia dát, FIT VUT, 2007*.

Obsah

1. Úvod

Dáta všetkého druhu zohrávajú podstatnú rolu v mnohých odvetviach spoločnosti. Stávajú sa predmetom, obchodu, funkčnosti firiem, štatistík, vyhodnocovaní, určovania trendov a pod. Nie sú to však len dáta samotné ale predovšetkým informácie z nich získané. Na základe týchto informácií sa zodpovedné osoby dokážu efektívne rozhodovať. Je nutné tieto informácie zobrazovať vhodne pre človeka, čím sa práve stáva ich vizualizácia. Vyjadrenie pomocou grafov a iných postupov, je výraznejšie a zrozumiteľnejšie, ako jednoduchý text s množstvom hodnôt. O rôznych typoch vizualizácie bolo pojednávané v práci [1], teraz však budeme pojednávať o zdrojoch ktoré do tejto vizualizácie vchádzajú, a ich úpravách.

Nepochybným najčastejším zdrojom budú dáta, pochádzajúce z určitého dátového skladiska, databázy, XML dokumentov a pod. Je nutné, si ale uvedomiť, že zobrazované dáta by mali mať určitý zmysel v požadovanom kontexte. Ak by som napr. nad databázou reštaurácie vytváral tabuľku, ako stĺpce by vystupovali jednotlivé názvy surovín a ako riadky by boli položky majetku reštaurácie, je zrejme, že vyhodnotenie tejto tabuľky, by bolo nezmyslom. Tento príklad je jasný, a zrejme by nikoho pri práci s vizualizačným nástrojom, nenapadlo danú kombináciu vyhodnocovať. Podobná situácia ale nemusí byť často tak zrejmá a môže spôsobovať nedorozumenia a nepochopenie vyhodnotených dát. Človek, ktorý často práve tieto vyhodnotenie z databáze získava, nemusí byť práve znalec alebo tvorca databázy, aby automaticky ovládal dané dimenzie a ich podstatu.

Práve z týchto dôvodov navrhneme vytvorenie modelu reportu, ktorý bude predstavovať metadata vizualizácie. V tomto modelu budú dané pravidlá, ktorá dimenzia s ktorou sa môže vyhodnocovať a aký bude výsledok. V nasledujúcich kapitolách budeme hovoriť o možných pravidlách a vytváraní týchto metadát.

2. Report Model

Report model, reprezentuje abstraktnú vrstvu, skladajúcu sa z metadát, ktoré obsahujú informácie o fyzickej databáze. Model umožňuje vytvorenie reportov bez znalosti dotazovacieho jazyka a databázy. V modelu je možné určiť ktoré dáta a dimenzie budú poskytované vo výslednom zobrazení. Model sa skladá s entít, ktoré sú zložené z jednotlivých položiek. Model by mal umožňovať vytvárať skupiny z položiek a vytvárať agregátne hodnoty. Medzi entitami sú vzťahy, ktoré umožňujú ich vzájomné zobrazovanie vo výsledných reportoch. Taktiež je vhodné aby report umožňoval vytváranie rôznych kalkulácií a ďalších odvodených položiek. Týmito jednotlivými časťami sa budeme postupne v tejto kapitole zaoberať.

Model umožňuje vytvorenie reportu, ktorý bude statický, alebo dynamický. Dynamický report musí dokázať svoje dáta obnoviť podľa aktuálnosti databáze. Musíme brať do úvahy, že ak bol report vytvorený na základe určitého modelu, tak toto spojenie už nepretrváva, a existencia reportu je od modelu reportu nezávislá. Report si nesie informácie z modelu v svojich metadatach. Model je omnoho obsiahlejší ako konkrétny report a v podstate definuje všetky možnosti, aké môžu nastať pri ich kombinácií za účelom vytvorenia určitého reportu.

2.1 Základ modelu

Základom pre vytvorenie funkčného modelu je samotná štruktúra databázy. Predovšetkým je nutné určiť:

- Tabuľky (entity modelu), z ktorých bude model zostavený
- Atribúty (jednotlivé stĺpce), ktoré bude možné zobraziť
- Vzťahy medzi entitami

Vzťahy medzi entitami, sú na databázovej úrovni reprezentované cudzími kľúčmi medzi tabuľkami. Je možné použiť vzťahy ktoré existujú, prípadne by mohlo byť možné niektoré vzťahy zneplatniť, alebo vytvoriť vzťahy nové.

Tento vzťah v modeli umožňuje vytvorenie vizualizácie, ktorá môže používať iba entity, ktoré sú v priamom, alebo tranzitívnom vzťahu.

2.2 Filtre

V modeli zavedieme pojem filter. Za určitých okolností nechceme aby osoba poverená zostavovaním reportu mala možnosť zobrazenia všetkých dát, čo práve filter umožní.

Zavedieme dve možností:

- Umožni zobrazenie všetkých dát, **okrem** tých čo spĺňajú podmienku filtra
- Umožni zobrazenie **iba** tých dát, ktoré spĺňajú filter

Zadanie filtru bude nad danou entitou, alebo skupinou entít (JOIN) zadane pomocou SQL dotazu SELECT ... WHERE...

Je nutné ale uvažovať aj o možnosti, že dáta síce nebudú zobrazené, ale pri agregovaných funkciách, sa môže s nimi počítať. Túto možnosť je tiež u filtru nutné uviesť.

2.3 Premenovanie

Model vytvára abstraktnú vrstvu nad samotnou databázou a môžeme v ňom vykonávať akcie bez ovplyvnenia pôvodnej databázy. Medzi takéto akcie patrí aj možnosť mapovania mien entít a atribútov entít na nové, často práve zrozumiteľnejšie a výstižnejšie mená. Taktiež by model v tomto smere mohol poskytovať možnosti zobrazenia konečného reportu v rôznych jazykoch.

2.4 Fomátovanie

Formát dát, sa stáva často problémom ich zobrazenie. Ku nezrovnalostiam dochádza, napr. pri získaní dát z databázového servera, s lokalizáciou USA, kde je dátum vo formáte MM/DD/YYY (napr. 08/27/2008), ale pre potreby reportu je nutné tento formát previesť na formát DD.MM.YYY (napr. 27.08.2008). Podobné formátovania sa riešia, nie len pri dátumoch, ale aj zobrazovaní meny, času, a pod.

Do tejto kapitoly môžeme zaradiť aj prepočty jednotiek medzi formátmi, keď je nutné napr. míle prepočítať na km, libry na kg a pod. Z tohto dôvodu by bolo vhodné ak by existovali predpripravené formátovania a ich prepočty. Vhodné by bolo použitie a vytvorenie formátovacích zostáv, ktoré by obsahovali kompletnú skupinu preddefinovaného formátovania napr. pre Českú republiku, by boli dané formáty pre dátum DD.MM.YYYY, pre čas HH:MM:SS a pod.

2.5 Dynamické položky a prepočty

Ako už bolo spomenuté, report je od svojho modelu oddelený, teda môže vzniknúť až po určitom čase, po vytvorení samotného modelu. Prípadne môže to byť report dynamicky a teda môže vznikať pri každom svojom generovaní. V niektorých prípadoch potrebujeme, aby určité hodnoty boli prepočítané vzhľadom k času kedy je report generovaný. Jedná sa napr. o uvedenie samotného času generovania, alebo o zložitejšie hodnoty ako môže byť pre-

počet aktuálneho kurzu. V týchto prípadoch je vhodné do modelu zaviesť možnosť, kedy sa aktuálne môže report pripojiť prostredníctvom internetu na určitú službu a získať aktuálny menový kurz. Takto je možné zabezpečiť, aby nebolo nutné, napr. pri kurzovej zmene, meniť model, alebo report.

Vhodnú technológiu poskytujú napr. webové služby, ktoré pomocou špeciálneho protokolu založenom na XML, dokážu poskytnúť množstvo požadovaných informácií.

2.6 Detaily

V oblasti vizualizácie je častým problémom množstvo zobrazovaných dát, ktoré je možné zobraziť. Často práve zobrazené tabuľky neposkytnú dostatočný priestor na obrazovke aby boli získané všetky požadované dáta, resp. všetky atribúty konkrétneho záznamu v tabuľke. Z tohto dôvodu, by bolo vhodné označiť niektoré atribúty entity ako „doplnkové“, ktoré budú prípadne zobrazené v dialógovom okne, alebo inou vhodnou formou.

2.7 Radenie

Vo výsledných reportoch, predovšetkým dynamických je vhodné poskytnúť možnosť zoradenia výsledkov podľa určitého atribútu.

V modeli bude možné zvoliť podľa ktorého atribútu bude výsledok zoradený, ako prednastavená možnosť. Prípadne bude tiež nastavenie podľa ktorých hodnôt je zoznam zoraditeľný. Je vhodné práve v modeli nastaviť nie len jeden atribút pre radenie, ale bude možné nastaviť aj sekundárny prípadne ternárny atribút, podľa ktorých sa bude postupovať v prípade zhody primárneho atribútu.

2.8 Vytváranie skupín

Vytváranie skupín bude v modeli v podstate určovať vhodných kandidátov z atribútov na operáciu GROUP BY. Je nutné aby model neposkytoval možnosť zoskupovania pre atribúty ako napr. rodné číslo, ale vhodné bude zoskupenie podľa napr. kategórie a pod. Samozrejmosťou je, že daný atribút môže byť len cudzím kľúčom. V tomto prípade by automaticky malo dôjsť k načítanou, vhodnej popisnej hodnoty z primárnej tabuľky.

2.9 Agregátne funkcie

Medzi agregátne funkcie patria operácie ako napr.: suma, priemer, min, max, a pod. Tieto operácie nie je vhodné vykonávať nad všetkými atribútmi, ale len nad vhodnými. Z tohto dôvodu budú agregátne funkcie vytvorené nad modelom. Taktiež by bolo vhodné pri určitých atribútoch určiť agregátne funkcie ako prednastavené, aby táto možnosť bola pri vizualizácií čo najlepšie dostupná.

2.10 Možnosti jednoduchých výberov

V databáze existujú často tabuľky ktoré obsahujú len malé množstvo hodnôt, ktoré predstavujú len určite, často statické možnosti, ako napr.: farba, kategória, kraj, a pod. V dynamických reportoch je často vhodné používať tieto možnosti ako filtrovacie atribúty. Ak zadávateľ pri generovaní reportu, ale hodnoty nepozná, tak nemôže presne napísať daný filter. Z tohto dôvodu by bolo vhodné v modely určiť atribúty ktoré budú vo výberoch zobrazované a rolovacie menu (Drop Down List). Obecne by bolo možné určiť hodnotu, ktorá bude hraničná pre toto vytváranie, a taktiež zvoliť samozrejme, že je možné toto vytváranie iba nad unikátnymi záznamami.

3. Uloženie modelu

Model, ako metadata reportu, je nutné určitou formou uložiť, aby mohol byť použitý pri vytváraní konkrétnych reportov. Vhodnú formu pre prehľadné uloženie poskytuje práve XML.

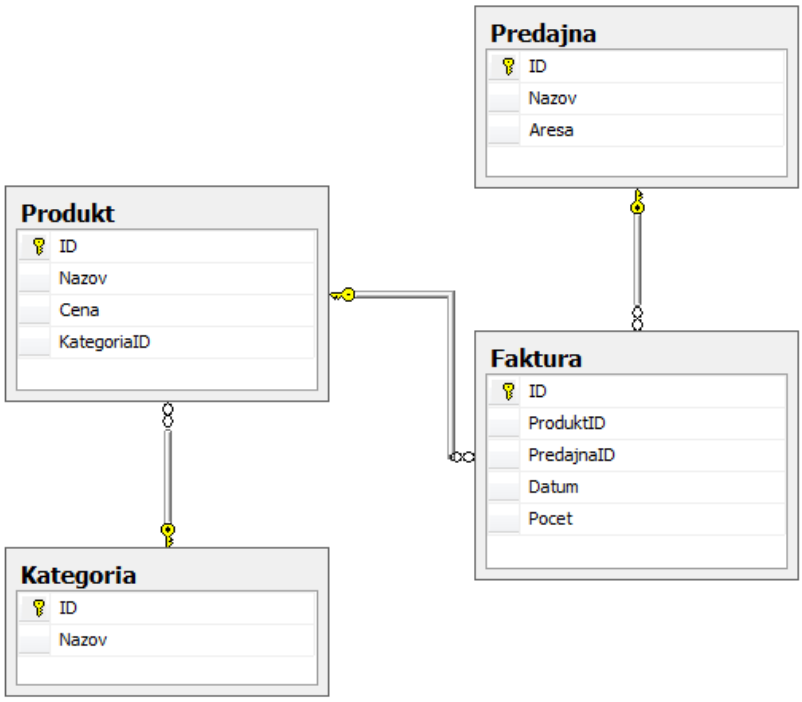
Základným elementom tohto XML by boli jednotlivé entity a pod elementy týchto entít by boli ich atribúty. Jednotlivé obmedzenia, formátovania, možnosti agregácií by boli uložené v ďalších elementoch etity, atribútu, alebo celého modelu.

Bližší formát bude uvedený a zrejmí na príklade.

4. Príklad

Pre názorné ukázanie niektorých, vyššie uvedených operácií a možností report modelu, uvedieme jednoduchý vzorový príklad, na ktorom budú dané postupy predvedené.

Jedna sa o databázu obchodnej siete predajní v svojom najjednoduchšom zmysle. Obsahuje 4 tabuľky: *Produkt*, *Predajna*, *Faktura*, *Kategoria*. V tabuľke *Predajna*, sú uvedené základné informácie, ako adresa a názov predajne. Tabuľka *Produkt* obsahuje popis jednotlivých predávaných produktov a jednotkovú cenu, taktiež každý produkt je zaradený jednej určitej kategórií. Tabuľka *Faktura*, obsahuje informáciu o tom, v akej predajni, aký produkt bol predaný a v akom čase, prípadne obsahuje aj informáciu o počte kusov ktoré boli naraz predané.



Obrázok 1 Schéma databázy

4.1 Vytvorenie základu modelu

Pokladajme schému databázy za korektnú, práve z toho dôvodu, že všetky vzťahy sú už v DB uvedené. V prípadne neúplných, alebo neplatných vzťahoch je v modelu možnosť túto skutočnosť zaznamenať. V podstate je model v svojej budúcej existencii na DB nezávislý, a z toho dôvodu aj vzťahy udržiavané v modeli musia byť od DB nezávislé. Z toho dôvodu

ich v modelu musíme vytvoriť a zaznamenať. V našom príklade prevezmeme všetky vzťahy z DB a vložíme ich do modelu.

Je nutné podotknúť, že v modeli musia byť uvedené tabuľky ktoré budú vystupovať ako entity vo vytvorených reportoch. Nie je nutné aby to boli všetky tabuľky z DB, ale vzťahy bude možné zaznamenať iba medzi tabuľkami, ktoré budú aj v modeli. V našom príklade to budú všetky tabuľky.

Poslednou podstatnou voľbou pri vytváraní základu modelu, je zaznamenanie všetkých platných atribútov pre entitu, ktoré budú v modeli zaznamenané. U každého atribútu bude taktiež zaznamenaný typ, pre svoje ďalšie použitie. Niektoré atribúty sú špeciálne, ako napr.: primárny kľúč, jedinečnosť, identity, tieto záznamy budú taktiež uvedené. Mapovanie základných typov je ukázané v nasledujúcej tabuľke:

DB server	Report Model
int	int
bit	bool
datetime	DateTime
decimal	decimal
float	float
char	char
nchar	char
varchar	string
nvarchar	string
real	float

Všetky spomenuté skutočnosti uvedieme vo vzorovom XML, kde bude hlavným elementom report, a podelementy budú jednotlivé tabuľky s atribútmi. Vzťahy budú zaznamenané u atribútov, ktoré predstavujú cudzí kľúč do určitej inej tabuľky v modeli.

Reprezentácia základu modelu v XML:

```
<?xml version="1.0" encoding="utf-8"?>
<report created="28.8.2008">
  <tables>
    <table name="Predajna">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="Nazov" type="string"></atribute>
        <atribute name="Adresa" type="string"></atribute>
      </atributes>
    </table>

    <table name="Faktura">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="ProduktID" type="int" fk="Produkt.ID"></atribute>
        <atribute name="PredajnaID" type="int" fk="Predajna.ID"></atribute>
        <atribute name="Datum" type="DateTime"></atribute>
        <atribute name="Pocet" type="int"></atribute>
      </atributes>
    </table>

    <table name="Produkt">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="Nazov" type="string"></atribute>
        <atribute name="Cena" type="decimal"></atribute>
        <atribute name="KategoriaID" type="int" fk="Kategoria.ID"></atribute>
      </atributes>
    </table>

    <table name="Kategoria">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="Nazov" type="string"></atribute>
      </atributes>
    </table>

  </tables>

</report>
```

4.2 Definícia filtrov

Filtre budeme definovať nad konkrétnymi tabuľkami. Filter v podstate bude zodpovedať SQL dotazu WHERE. Ak tabuľka má vzťah s inou, teda obsahuje cudzí kľúč odkazujúci na primárny, tak filter môže vo svojom dotaze obsahovať aj podmienku cudzej tabuľky.

U filtru je nutné uviesť, či nezobrazované hodnoty budú, alebo nebudú započítavané do agregovaných funkcií.

V našom príklade uvedieme filter, ktorý zamedzí zobrazeniu položiek vo faktúre, ak atribút počet obsahuje číslo väčšie ako 10.

```
<filters>
        <filter name="MalyPocet" filters="WHERE Fakture.Pocet &lt; 10"
                type="allow" agregate="no" ></filter>
</filters>
```

Atribút *filters* obsahuje samotnú podmienku. V tomto reťazci sa nachádza výraz *&lt*, ktorý vyjadruje znamienko menší. Tieto možnosti uvedieme v nasledujúcom zozname[5]:

1. & - &
2. < - <
3. > - >
4. " - "
5. ' - '

Ďalším atribútom je *type*. Tento atribút nadobúda hodnoty *allow*, alebo *deny*. V podstate znamená, či dané hodnoty, obmedzené filtrom budú, alebo nebudú povolené v modeli. Atribút *agregate*, určuje či hodnoty zneplatnené filtrom budú, alebo nebudú vystupovať sa započítavať sa do agregátnych funkcií.

4.3 Premenovanie

K premenovaniu dochádza naplnením atribútu *alias* v elemente *table*, alebo elemente *atribute*. V našom prípade premenujeme tabuľku *Predajna*, na názov *Prevádzka*.

```
<table name="Predajna" alias="Prevádzka">
```

4.4 Formátovanie

K formátovaniu zvolíme atribút *format* v elemente atribútu tabuľky. Formátovanie bude zadané štandardným reťazcom, ktorý bude vyjadrovať požadovaný formát výstupu. Prehľad niektorých reťazcov je uvedený v dokumente [2].

V našom príklade určíme, že požadujeme formát položiek atribútu *Pocet* v tabuľke *Faktura*, v tvare 28.08.2008, teda reťazec DD.MM.YYYY.

Výsledné upravené XML bude vyzerať:

```
<atribute name="Datum" type="DateTime" format="DD.MM.YYYY"></atribute>
```

4.5 Dynamické položky

Dynamická položka predstavuje možnosť aktuálneho prepočtu podľa najnovších hodnôt. V podstate potrebujeme odkaz na túto položku, ktorou bude napr. aktuálny kurz a bude poskytnutý na internete. Tieto položky nemusia byť využité iba v jednom atribúte, alebo tabuľke, ale často budú použité na viacerých miestach reportu. Z tohto dôvodu zavedieme element *dynamics* a podelementy *dynmic*, ktoré budú odkazovať na spojenia na internet.

Použitie takejto položky môže byť v už existujúcom atribúte, a tým spôsobený jeho prepočet, alebo môžeme v reporte vytvoriť navrhovaný nový atribút, ktorý bude spočítaný s pôvodného a dynamickej položky.

V našom príklade zvolíme odkaz na internete na webovú službu, ktorá poskytne aktuálny kurz českej koruny k euru. Pomocou tento dynamickej položky vytvoríme nový atribút v entite *Faktura*.

```
<dynamics>
      <dynamic name="toEuro" type="decimal"
      ws="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl"
      method="GetRate(&quot;czech &quot;, &quot;euro &quot;)"></dynamic>
</dynamics>
```

Atribút *ws*, označuje webovú službu (Web Service), *method*, označuje metódu, ktorú je nutné použiť na získanie konverzného kurzu. Dynamická položka obsahuje informáciu taktiež o svojom type, ktorý v našom prípade je *decimal*.

Následne uplatníme túto položku v novom atribúte.

```
<atribute name="CenaEU" type="decimal" mode="alternate"
          function="toEuro * Produkt.Cena" ></atribute>
```

Atribút *mode* určuje, že daný atribút je doplnený a nie je povodne súčasťou DB. Atribút *function* označuje postup akým dôjde ku vypočítanou daného nového alternatívneho atribútu.

4.6 Výber datailov

Pri výbere detailov určujeme, ktoré atribúty entity sú podstatné a majú byť zobrazené v detailovom okne, ktoré môže byť na strane tabuľky, alebo zobrazené ako dialóg a podobne. Takýto atribút označíme položkou *detailed="yes"*. V našom príklade takto označíme atribút *Adresa* tabuľky *Predajna*.

```
<atribute name="Adresa" type="string" detailed="yes"></atribute>
```

4.7 Radenie

Pri radení, určíme v každej tabuľke podľa ktorého atribútu je možné radiť záznamy. V prevažnej časti prípadov, ale chceme povoliť radene v reporte pre akýkoľvek zobrazený stĺpec, preto tento atribút budeme pokladať prednastavenie na kladnú hodnotu. Z toho vyplýva, že budeme radenie explicitne len zakazovať. To bude atribútom *order* nastaveným na hodnotu *no*.

Druhou podstatnou častou u radenia, je vhodná voľba prednastaveného radenia. To nastavíme nad tabuľkou pomocou atribútu *orderBy*. Hodnoty budú jednotlivé stĺpce tabuľky, ale stĺpce cudzej tabuľky získanej dostupným vzťahom. Je nutné uvažovať aj o primárnom a sekundárnom radení. Toho bude dosiahnuté cez zložený atribút z viacerých riadiacich prvkov. Prednastavený smer radenia budeme predpokladať za vzostupný (ASC), ak by bolo nutné ho zmeniť, tak k tomu bude slúžiť atribút *orderDirection* s hodnotou DESC.

V našom príklade vytvoríme prednastavené radenie podľa názvu produktu a sekundárne podľa ceny.

Výsledná hlavička tabuľky s prednastaveným radením bude vyzerať nasledovne:

```
<table name="Produkt" orderBy="Produkt.Nazov,Produkt.Cena">
```

4.8 Vytváranie skupín

V modeli budeme určovať nad ktorými atribútmi tabuľky je možné vytvoriť skupinu. V jednej tabuľke bude možné vytvoriť vo výslednom report skupinu nad rôznymi atribútmi. Túto skutočnosť zaznamenáme v novom atribúte elementu tabuľky. Určenie skupín bude explicitné, teda bez tohto určenia v modeli nebude možné v reporte vytvoriť zoskupenie nad zvoleným atribútom.

V našom prípade povolíme vytváranie skupín nad atribútmi *Nazov* a *KategoriaID* v tabuľke *Produkt*. Výsledné XML potom bude mať tvar:

```
<table name="Produkt" orderBy="Produkt.Nazov,Produkt.Cena"
 group="Produkt.Nazov,Produkt.KategoriaID">
```

4.9 Agregátne funkcie

Agregátne funkcie sú s prevažnej časti práve záležitosťou konkrétneho reportu, nad atribútom, ktorý umožňuje túto možnosť. Prevažne sa samozrejme jedná o číselné atribúty. Z tohto dôvodu budú agregátne funkcie nad stĺpcami číselných typov implicitne povolené.

U agregátnych funkcií v modelu, ale bude vhodné určiť odporúčanú agregátu funkciu nad určitou tabuľkou. To vykonáme pridaním potrebných elementov pre funkciu.

V našom príklade predvolíme, ako odporúčané agregátne funkcie, sumu a priemer ceny nad tabuľkou *Faktura*. Medzi faktúrou a produktom je platný vzťah a na jeho základe bude možné danú agregátnu funkciu zostaviť.

Výsledný element v elemente *table* bude vyzerať nasledovne:

```
<agregates>
   <agregate name="Celková tržba" function="SUM(Produkt.Cena)"> </agregate>
   <agregate name="Priemerná tržba" function="AVG(Produkt.Cena)"> </agregate>
</agregates>
```

4.10 Možnosti jednoduchých výberov

Predstavuje možnosť rolovacieho menu, pre položky tabuľky, ktorých je obecne malé množstvo. Tento atribút bude zvolený automaticky pre určitú, užívateľom definovanú, hodnotu počtu položiek v tabuľke. Ďalšou podmienkou bude aby daný stĺpec bol definovaný ako jedinečný (*unique*) v databáze.

Explicitne môžeme umožniť nastavenie tohto atribútu ako *eselect="yes",* v možnosti atribútov konkrétneho stĺpca v modeli.

4.11 Výsledné kompletné XML

```
<?xml version="1.0" encoding="utf-8"?>
<report created="28.8.2008" name="Model1">
  <tables>
    <table name="Predajna" alias="Prevádzka">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="Nazov" type="string"></atribute>
        <atribute name="Adresa" type="string" detailed="yes"></atribute>
      </atributes>
    </table>

    <table name="Faktura">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="ProduktID" type="int" fk="Produkt.ID"></atribute>
        <atribute name="PredajnaID" type="int" fk="Predajna.ID"></atribute>
        <atribute name="Datum" type="DateTime" format="DD.MM.YYYY"></atribute>
        <atribute name="Pocet" type="int"></atribute>
      </atributes>
      <filters>
        <filter name="MalyPocet" filters="WHERE Fakture.Pocet &lt; 10"
                type="allow" agregate="no" ></filter>
      </filters>
    </table>

    <table name="Produkt" orderBy="Produkt.Nazov,Produkt.Cena"
           group="Produkt.Nazov,Produkt.KategoriaID">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="Nazov" type="string"></atribute>
        <atribute name="Cena" type="decimal"></atribute>
        <atribute name="KategoriaID" type="int" fk="Kategoria.ID"></atribute>
        <atribute name="CenaEU" type="decimal" mode="alternate"
                  function="toEuro * Produkt.Cena" ></atribute>
      </atributes>

      <agregates>
        <agregate name="Celková tržba" function="SUM(Produkt.Cena)"> </agregate>
        <agregate name="Priemerná tržba" function="AVG(Produkt.Cena)"> </agregate>
      </agregates>

    </table>

    <table name="Kategoria">
      <atributes>
        <atribute name="ID" type="int" primary="yes" identity="yes"></atribute>
        <atribute name="Nazov" type="string"></atribute>
      </atributes>
    </table>

  </tables>

  <dynamics>
    <dynamic name="toEuro" type="decimal"
             ws="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl"
             method="GetRate(&quot;czech &quot;, &quot;euro &quot;)"></dynamic>
  </dynamics>

</report>
```

5. Záver

Vytváranie metadát pre samotný výsledný report zatiaľ nepatrí k úplne bežnej praxi. V tomto texte bol uvedený jednoduchý a základný postup, ako by daný model mohol byť vytvorený. Pre reálne nasadenie je však nutné doriešiť detaily, pri hlbšej a podrobnejšej analýze. Vytvorenie daných modelov môže poskytnúť vhodnú oporu pre návrh vizualizačného rozhrania databáz, za účelom poskytnutia lepšej ovládateľnosti, pre konečných užívateľov.

6. Literatúra

[1] RUTTKAY, L., Vizualizácia dát, FIT VUT, 2007

[2] http://john-sheehan.com/blog/wp-content/uploads/msnet-formatting-strings.pdf 28.8.2008

[3] Pearson, W.: Create a Report Model, http://www.databasejournal.com/features/mssql/article.php/10894_3598931_2 28.8.2008

[4] http://www.devhood.com/Tutorials/tutorial_details.aspx?tutorial_id=454 28.8.2008

[5] Special characters and XML strings, http://www.devx.com/tips/Tip/14068 28.8.2008