MigDB - rešerše

O projektu

Projekt migrace databáze (dále jen MigDB) si klade za cíl vyřešit problematiku převodu databázového schématu na nově definovaný při zachování dat.

Domovská stránka projektu je https://rabbit.felk.cvut.cz/trac/migdb.

Technologie

Během vývoje aplikace jsme se setkali s celou řadou softwarových technologií souvisejících s modelováním dat, které lze rozdělit do několika oblastí:

1. Eclipse Modeling Framework
2. jazyky popisující modely
3. transformační jazyky
4. třídy (API) pro manipulaci s modely

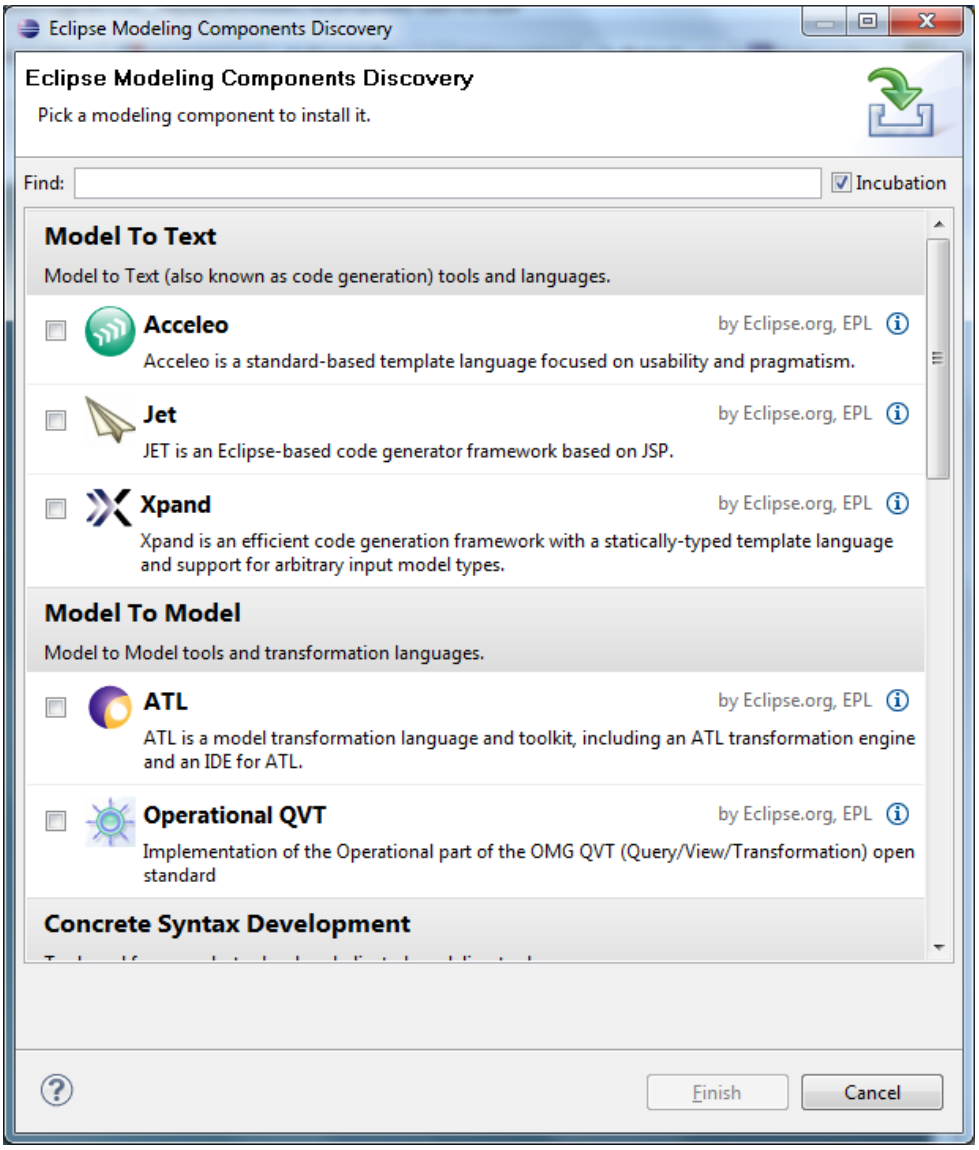
Následuje podrobnější popis jednotlivých oblastí.

Eclipse Modeling Framework

Veškeré, níže popsané věci, lze snadno používat v rozhraní Eclipse - podpora modelů, transformačních jazyků a prací s nimi v prostředí Eclipse je cílem projektu **Eclipse Modeling Framework** (EMF) - http://www.eclipse.org/modeling/emf/

Není-li zapotřebí konkrétních verzí jednotlivých komponent EMF, pak nejsnažší cestou ke zprovoznění všech potřebných věci je stáhnout z http://www.eclipse.org/downloads/ vydání **Eclipse Modeling Tools (includes Incubating components)**.

Toto sestavní již zahrnuje podporu Ecore a většinu ostatních komponent lze snadno přidat. Po spuštění je v menu pod položkou Help volba Install Modeling Components:



Obrázek 1 - rozšíření pro Eclipse

Pro náš projekt je nejduležitější **Operational QVT**, další co stojí za vyzkoušení je např. ATL a Acceleo.

Instalace požadovaných komponent probíhá snadno - v instalátoru stačí odsouhlasit výběr, později licenci, počkat až dojde ke stažení požadovaných pluginů a nakonec restartovat Eclipse.

Výjimkou je podpora Emfatic modelů - musíme zvolit Help > Install New Software..., tlačítkem Add přidat reposirotory http://scharf.gr/eclipse/emfatic/update (název např. Emfatic) a pod Uncategorized zvolit **Emfatic (Incubation)**. Zbytek instalace je identický s předchozím způsobem.

Jazyky popisující modely

Pro nás klíčovým formátem je **Ecore** (používá se přípona .ecore). Ecore v datové podobě odpovídá XMI (XML Metadata Interchange), díky XML je formát čitelný i v případě nedisponování vhodným software pro čtení tohoto souboru. Větší přednost spočívá především ve snadném strojovém zpracování (čtení/vytváření/modifikace). Za tímto účelem lze využít již hotových nástrojů,

které jsou uvedeny níže. Bohužel formát není úplně vhodný pro ruční psaní - režie (poměr obsah vs. forma) je poměrně vysoká.

Ukázka Ecore modelu v datové podobě:

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="Families">
  <eClassifiers xsi:type="ecore:EClass" name="Family">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="lastName" ordered="false"
        unique="false" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="father" ordered="false"
        lowerBound="1" eType="ecore:EClass test.ecore#/0/Member" containment="true"
        eOpposite="test.ecore#/0/Member/familyFather"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="mother" ordered="false"
        lowerBound="1" eType="ecore:EClass test.ecore#/0/Member" containment="true"
        eOpposite="test.ecore#/0/Member/familyMother"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="sons" ordered="false" upperBound="-1"
        eType="ecore:EClass test.ecore#/0/Member" containment="true" eOpposite="test.ecore#/0/Member/familySon"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="daughters" ordered="false"
        upperBound="-1" eType="ecore:EClass test.ecore#/0/Member" containment="true"
        eOpposite="test.ecore#/0/Member/familyDaughter"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Member">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="firstName" ordered="false"
        unique="false" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="familyFather" ordered="false"
        eType="ecore:EClass test.ecore#/0/Family" eOpposite="test.ecore#/0/Family/father"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="familyMother" ordered="false"
        eType="ecore:EClass test.ecore#/0/Family" eOpposite="test.ecore#/0/Family/mother"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="familySon" ordered="false"
        eType="ecore:EClass test.ecore#/0/Family" eOpposite="test.ecore#/0/Family/sons"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="familyDaughter" ordered="false"
        eType="ecore:EClass test.ecore#/0/Family" eOpposite="test.ecore#/0/Family/daughters"/>
  </eClassifiers>
</ecore:EPackage>
```

Obrázek 2 - Ecore model

Dalším formátem je **Emfatic** (.emf). Přednostní Emfaticu oproti Ecore je právě zaměření na jeho plain-textovou podobu:

```
        @OCL(inv= "self.owningTable = self.underlyingIndex.indexedTable")
class UniqueIndex extends TableConstraint {
        ref Index[1] underlyingIndex;
}
```

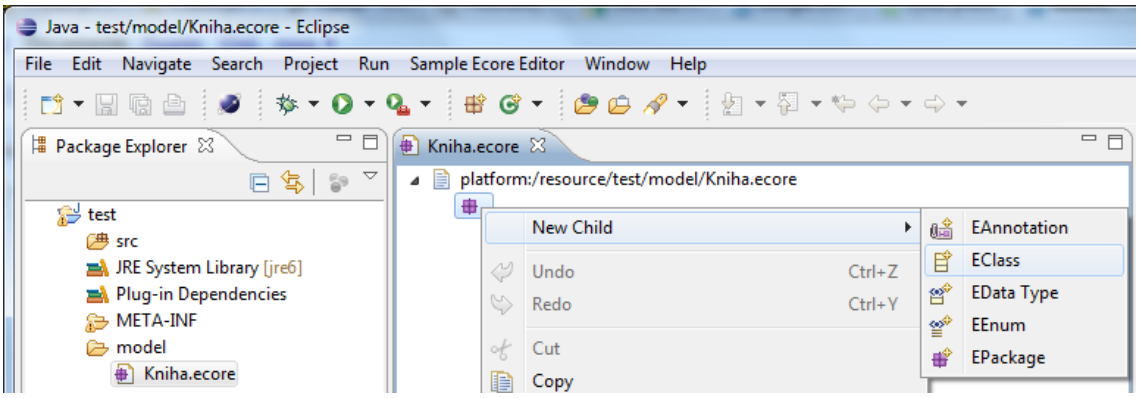
Struktura jazyka je evidentně snadněji ručně zapisovatelné, než u Ecore - zápis je blízký psaní kódu z vyšších programovacích jazyků - např. Java, podobnost s anotacemi, deklarací třídy, zápis dědičnosti, polí nelze přehlédnout. Emfatic je syntaxí blízký metamodelovacímu jazyku KM3 (**Kernel Meta Meta Model**), nejedná se však o ekvivalenty.

Podpora v Eclipse

Pro vyzkoušení vytvoříme nový projekt přes File > New > Project … > Model to Model Transformations > Operational QVT Project, po zvolení názvu projektu je projekt vytvořený.

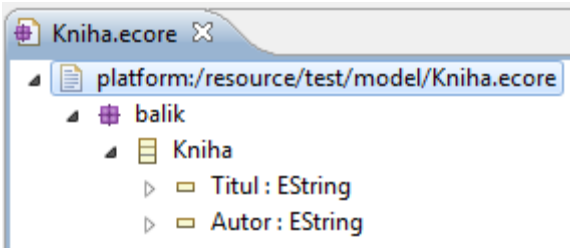
rada pro Eclipse nováčky - zavřete záložku Welcome …

Nyní vytvoříme nový Ecore model, např. na složce model v našem projektu klikneme pravým tlačítkem myši, zvolíme New > Other … > Eclipse Modeling Framework > Ecore Model. Zvolíme název (s koncovkou .ecore) - např. Kniha.ecore



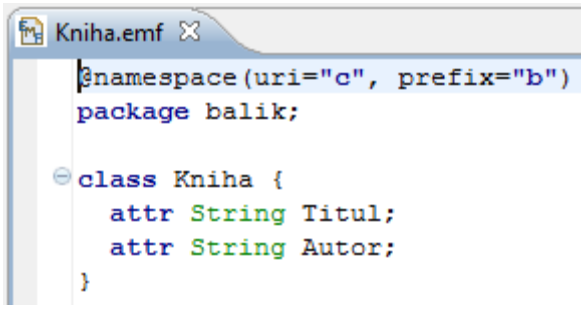
Obrázek 3 - Náhled na Ecore

V tomto vizuálním editoru můžeme snadno upravovat Ecore model. Pro demonstraci nejprve v Properties natavíme Name, Ns Prefix, Ns URI (libovolné názvy) a dále si vytvoříme EClass, v podokně Properties nastavíme Name na Kniha a na této EClass podobnými způsobem provedem New Child > EAttribute (2x). V jejich Properties nastavíme jednomu Name na Titul, druhému na Autor a oběma a EType na EString. Dostaneme tak takovýto model:



Obrázek 4 - Vyzualizované Ecore

Tento model můžem v případě potřeby převést na Emfatic - v podokně Package Explorer kliknout pravým tlačítkem myši na Kniha.ecore a vybrat **Generate Emfatic Source**:



Obrázek 5 - Zápis v Emfaticu

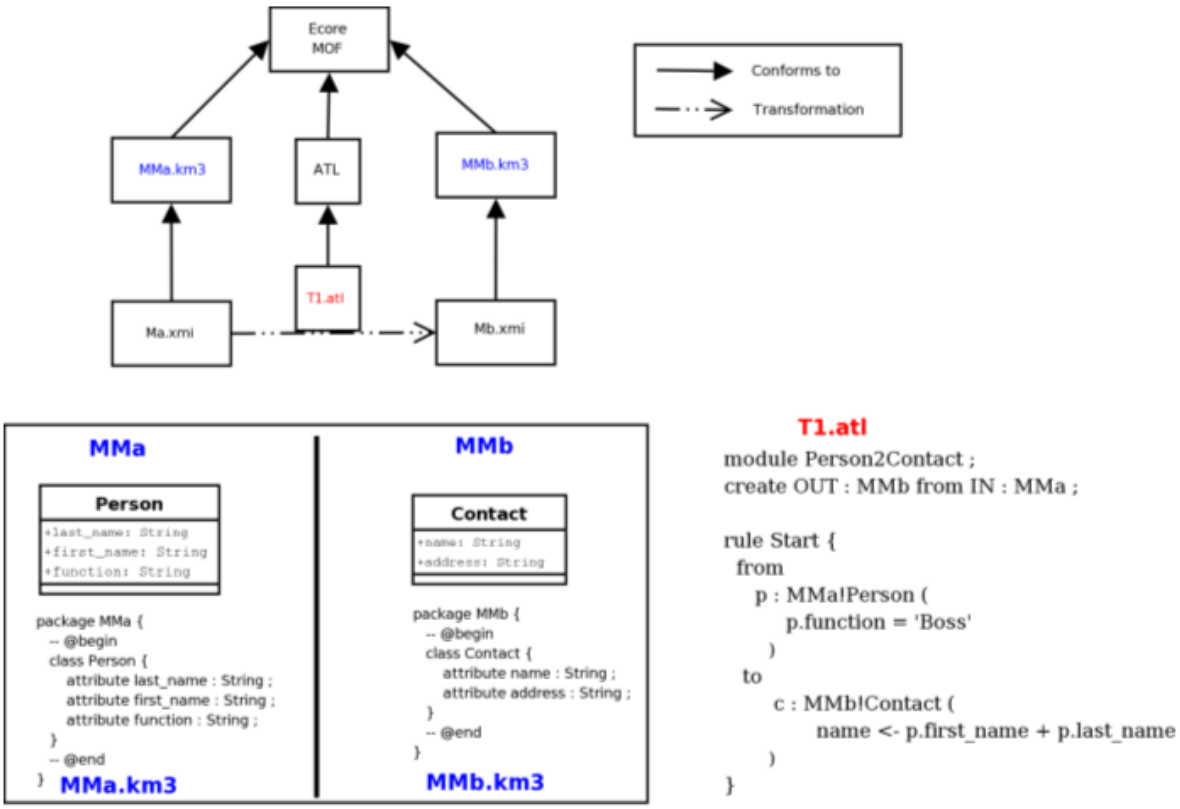
Konverze je možná i opačným směrem, na Emfatic modelu zvolit **Generate Ecore Model**.

Instance namodelovaných dat můžeme podržet v XMI souborech. Takovýto XMI soubor snadno vytvoříme přes Ecore model - na vybrané EClass v Ecore editoru v menu vyvolaném přes

pravé tlačítko zvolíme **Create Dynamic Instance...** - tímto způsobem můžeme XMI upravovat podobně jako Ecore model s tím rozdílem, že data v XMI odpovídají našemu modelu.

Transformační jazyky

Pro ty, kteří s transformačními jazyky nemají žádnou zkušenost je lze poměrně snadno přiblížit tímto schémetem:



Obrázek 6 - Znázornění transformace

Diagram zobrazuje Ecore MOF (Meta-Object Facility), v tomto případě můžeme uvažovat za výkonnou část, dále pak metamodel A (zdrojový) a metamodel B (cílový). Metamodely jsou zapsány ve formátu KM3. Transformace je zapsaná v jazyce ATLAS.

Cílem této transformace je vzít data popsané metamodelem A (osoba má atributy křestní jméno, příjmení a funkci) do formátu popsaného metamodelem B (kontakt má jméno a adresu). Má-li osoba funkci “Boss”, pak se vytvoří kontakt, kde jméno vznikne zřetězením křestního jména a příjmení.

Transformační jazyk tedy popisuje způsob převodu dat, které jsou popsány metamodely.

Existuje široká škála transformačních jazyků: ATL, Beanbag, GReAT, Kermeta, M2M, Mia-TL, MOF, MOLA, MT, QVT, SiTra, Stratego/XT, Tefkat, VIATRA. Rozděleny mohou být do 2 hlavních kategorií - imperativní a deklarativní.

Imperativní jazyky

- program v imperativním jazyku je popis algoritmu
- jednotlivé kroky programu (algoritmu) na sebe navazují, je důležité jejich pořadí
- používají cykly for, while, do-while
- používají větvení (switch, if, goto)
- používají přiřazení
- vyšší chybovost
- vyšší optimalita

Deklarativní programování

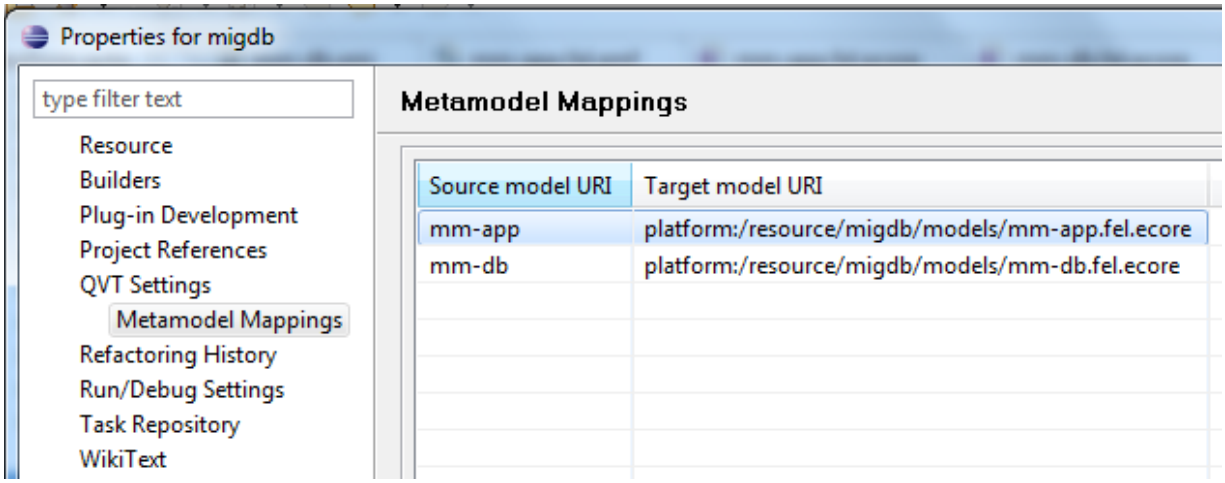
- nestará se o algoritmus, specifikuje požadovaný cíl
- závisí na interpretu jazyka, který provádí algoritmickou část
- definuje se množina funkčních závislostí nebo pravidel
- nebývá důležité pořadí jednotlívých pravidel
- střídmé využívání proměnných
- cykly řešeny rekurzí

QVT operational transformace (QVTo) v Eclipse

Na začátku QVTo transformace se nachází definice modelů:

```
modeltype APP uses 'mm-app';
modeltype RDB uses 'mm-db';
```

aby Eclipse vědělo, o které modely se jedná, je zapotřebí toto nastavit v Project > Properties a v otevřeném okně zvolit QVT Settings a Metamodel Mappings nastavit tak, aby mm-app ukazoval na Ecore soubor:



Obrázek 7 - Nastavení metamodelů

Transformace se v našem projektu spustí přes soubor run.launch > (pravoklik) > Run As > 1 run

Třídy (API) pro manipulaci s modely

Doposud popsané věci jsou užitečné do doby, kdy modelů využíváte např. k dokumentaci projektu či např. transformaci dat. V momentě, kdy potřebuje samotný model zpracovávat v aplikaci, je výhodné využít hotového aplikačního rozhraní (API).

Kód projektu Eclipse Modeling Framework je šířen pod svobodnou licencí s otevřeným zdrojovým kódem – Java

V našem projektu potřebuje pracovat s Ecore modelem, k tomu stačí využít balíčků:

- `org.eclipse.emf.common`
- `org.eclipse.emf.ecore.xmi`
- `org.eclipse.emf.ecore`

Práce nad modelem pak připomíná práci s Document Object Model (DOM) nad XML dokumentem - pomocí gettrů získávat jména tříd, atributů a datové typy. Pomocí setterů je měnit a do kolekcí přidávat nové třídy, atributy atd.

Zdroje a odkazy

http://eclipse.org/emf/
http://wiki.eclipse.org/Emfatic
http://en.wikipedia.org/wiki/ATLAS_Transformation_Language