# Rel-Miner
## Pohled pod kapotu

Tomáš Karban

KEG  28. dubna 2004

# Abstrakt

- Trocha teorie
  - Relační asociační pravidla
  - WARMR
- Praktické aspekty implementace relačního data miningu za pomoci bitových řetízků
  - Architektura systému
  - Návrh nové metabáze v XML
  - Některé algoritmy pro zpracování hypotéz
  - Výpočet celkového počtu hypotéz
  - Rozdělení prostoru hypotéz na části pro paralelní zpracování
  - Optimalizace výpočtů a datových přenosů
- Představení alfa-verze systému

# Relační hypotézy

- Zůstává jedna hlavní tabulka (řádek = jeden objekt)
- Další tabulky v relaci 1:N
  - cizí klíče, „master-detail“, schéma hvězdy
- Přirozené zobecnění – zavedení virtuálních atributů
  - virtuální atribut přináší informaci z podřízené tabulky
    - agregace
    - existenčně kvantifikovaná formule
    - asociační pravidlo, hypotéza
  - ve světě ILP se tento postup nazývá „propozicionalizace“

# Relační hypotézy - příklad

- „něco jako Barbora“:
  Clients: Birth, Gender, MaritalStatus, Children, LoanQuality
  Transactions: Date, TransactionAmount, SourceAccount, TargetAccount

- MaritalStatus(divorced) & Children(3) &
  SingleIncome(yes) & AvgIncome(< 1500) $\Rightarrow_{76\%}$ LoanQuality(bad)

- SingleIncome odvozený jako:
  TransactionAmount(> 500) $\Rightarrow_{93\%}$ SourceAccount(acc345) / Client(ABC)
  yes = síla hypotézy alespoň 90%

- AvgIncome odvozený jako:
  AVG(SELECT SUM(TransactionAmount)
  WHERE (TransactionAmount > 0) GROUP BY YearMonth)

# Metodologie

- CRISP-DM:
  ... předzpracování dat, modelování, vyhodnocení, ...

- Část předzpracování se přesouvá do fáze modelování
- Velikost úlohy (doba běhu) vynucuje spíše „skeptický" přístup k modelování
  - asi nelze zvládnout spočítat celou úlohu
  - běh úlohy musí postupovat smysluplným směrem, aby mohl být předčasně ukončen
  - možnost vyhodnocení dočasných výsledků za běhu, možnost malých modifikací zadání úlohy

# Relační data mining jinde ve světě...

- Vesměs práce založené na ILP
- Dva odlišné směry:
  - propozicionalizace
    (pak mohou následovat i metody mimo ILP)
  - obecně relační formule/hypotézy
    (není třeba mít jednu tabulku „hlavní“)

- Většina výzkumu se spíše orientuje na strojové učení, klasifikace/predikce
  - to je zřejmě dáno praktickou dostupností relačních dat určitého typu a s určitým cílem
  - především chemie, biochemie, genetika, bílkoviny, rakovina, ...

# Relační asociační pravidla ve světě...

- Jeden z možných přístupů je propozicionalizace
  - ... a následně běžné postupy (APRIORI, GUHA)
  - omezení na formule s existenčním kvantifikátorem
  - => omezení na určitý druh dat (doménu)
- Relační zobecnění algoritmu APRIORI → **WARMR**
  - Dehaspe & Toivonen, 1999
  - 2 fáze (stejně jako APRIORI)
    - hledání frekventovaných atomsetů
    - konstrukce asociačních pravidel
  - atomset je „konjunkce literálů" utvořených z libovolné tabulky, obsahuje proměnné, které se v konjunkci navzájem svazují

# WARMR – příklad

- Džeroski, Lavrač: Relational Data Mining, pp. 189-212

- ?- customer(X), parent(X, Y), buys(Y, cola) ... freq: 25%
- ?- customer(X), parent(X, Y) ... freq: 75%
- "?- customer(X), parent(X, Y)" ⇒
  "?- customer(X), parent(X, Y), buys(Y, cola)" ... conf: 33%

- v kontextu relačního miningu se tato forma patternu nazývá „query extension" (termín asociační pravidlo není příliš vhodný)
- zkrácený zápis (aby se na pravé straně neopakovala levá):
  ?- customer(X), parent(X, Y) ⇝ buys(Y, cola)

- pozn.: v APRIORI zápis ABC→D také znamená spíše ABC→ABCD

# WARMR – zadání úlohy

- Při zadání úlohy se specifikuje obecný způsob konstrukce atomsetu („declarative language bias")
  - určují se povolené vstupní/výstupní proměnné a konstanty

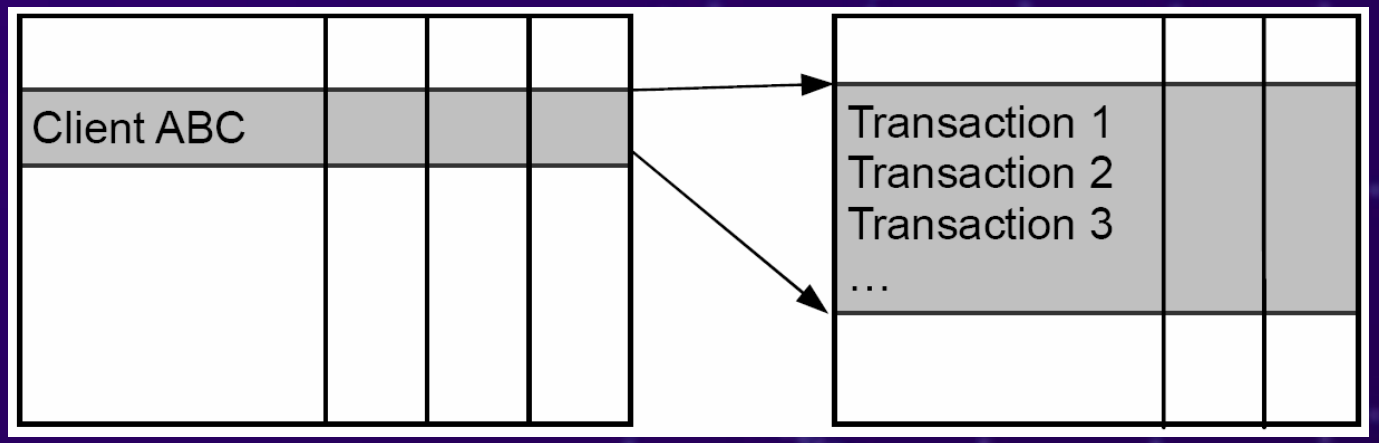
| definice | korektní atomset | nekorektní a. |
|---|---|---|
| {p(-,-), q(-)} | ?- p(X, Y), q(Z) | ?- p(X,Y), q(Y) |
| {p(+,-), q(-)} | ?- q(X), p(X,Y), p(Y,Z) | ?- p(X,Y) |
| {p(+,±), q(-)} | ?- q(X), p(X,X), p(X,Y) | ?- q(X), p(Y,X) |
| {p(±,a), q(±)} | ?- p(X,a), q(X), q(Z), p(Z,a) | ?- p(X, Y) |

# WARMR – srovnání

- Založený na existenciálně kvalifikovaných formulích
  - vhodné pro „strukturálně těžké" databáze s „jednoduchými" daty
- Naproti tomu Rel-Miner:
  - umí navíc agregace, což je přirozené pro jednoduché hvězdicové schéma databáze s „těžkými" daty
  - umí navíc virtuální atributy ve formě asociačních pravidel (hypotéz) na podřízených datech
  - existenciální atributy lze simulovat s použitím fundované implikace bez antecedentu a s omezením $p = 0\%$
    - nelze jít do hloubky, nelze rekurze
    - není příliš efektivní
  - asociační pravidla (z důvodu interpretovatelnosti) kladou důraz na hlavní datovou tabulku, virtuální atributy jsou spíše „doplňkové"

# Prostor všech hypotéz (1)

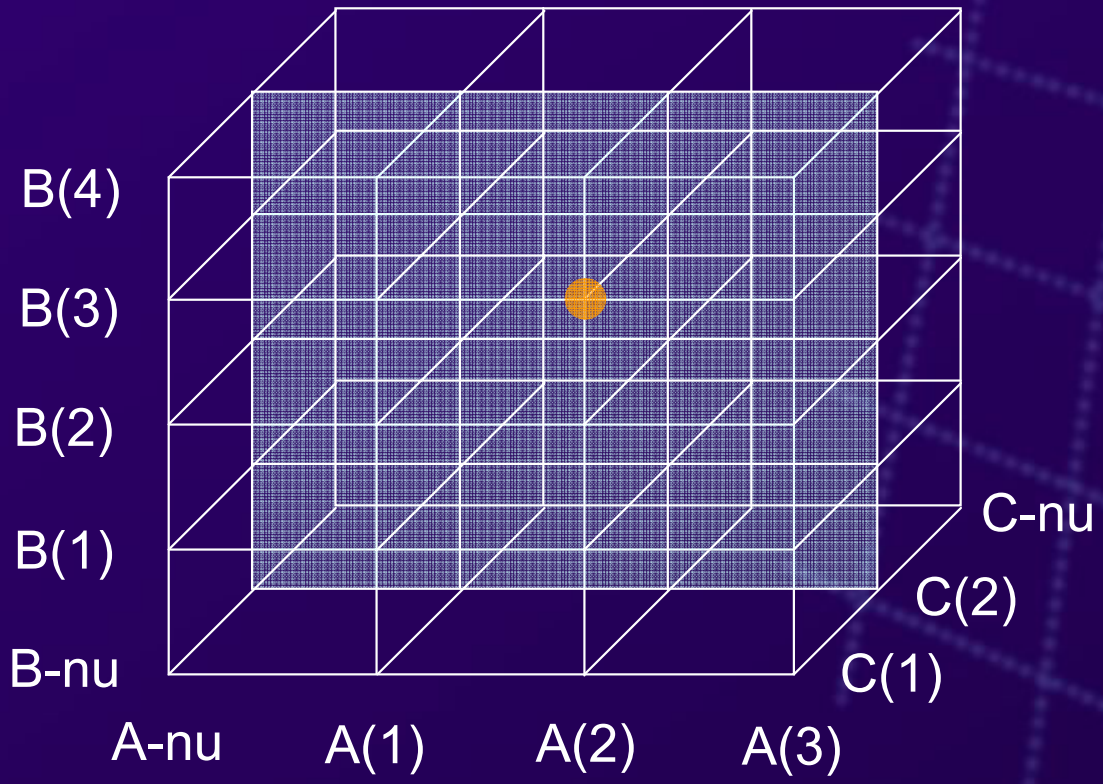
- ... roste exponenciálně v počtu atributů (a kategorií)
- Při omezení délky hypotézy na *k* už „jen" s mocninou *k*
- Počet atributů hlavní tabulky ovšem silně zvyšují virtuální atributy...

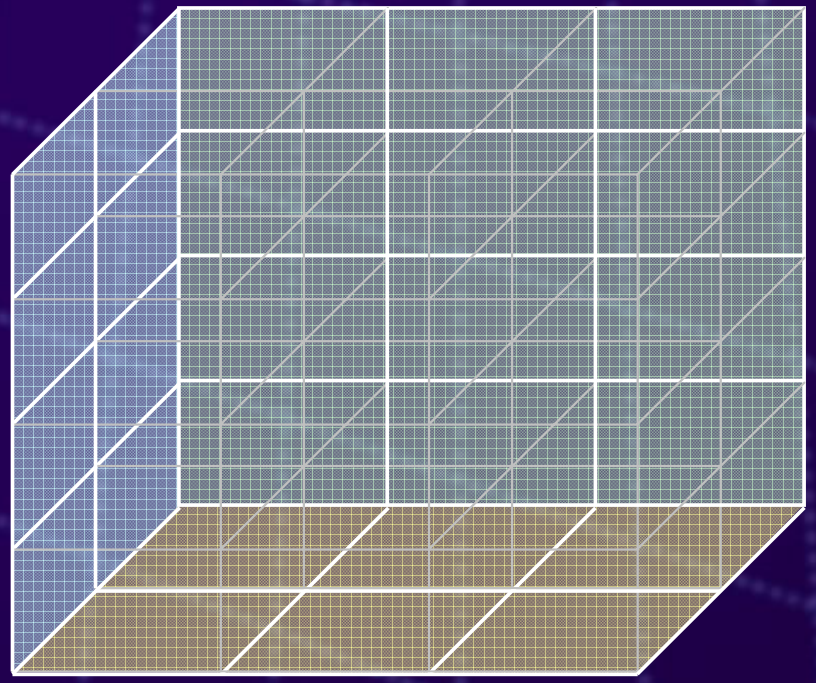



- 10 atributů podřízené tabulky => ~100 hypotéz (virtuálních atributů)
- počet hypotéz z hlavní tabulky je třeba násobit ~100, i když připustíme maximálně 1 virtuální atribut do každé hypotézy

# Prostor všech hypotéz (2)

- Cedent si lze představit jako mnohorozměrný kvádr
  - dimenze = atributy
  - hodnoty v každé dimenzi = **koeficienty** + speciální „NOT USED"
- Př.: A(2) & B(3) & C(1) omezení délky cedentu na 2



# Prostor všech hypotéz (3)

- Kvádr se „snadno řeže“ pomocí nadrovin na poloprostory
  - např. C(=1), B(<3)
- Tomu odpovídá dělení úlohy na disjunktní menší části
- Problém nalézt „optimální“ dělení
  - rekurzivní dělení (binární nebo mnohočetné)
  - volba vhodné nadroviny
  - počítání velikosti celé úlohy a části úlohy
  - virtuální atributy = primární kandidáti na dělení

# Počítání hypotéz

- Počet koeficientů
- Počet parciálních cedentů, celých cedentů
- Počet hypotéz

- Nutno vzít v potaz omezení:
  - max. délka každého parciálního cedentu
  - max. délka celého cedentu
  - max. počet použitých virtuálních atributů

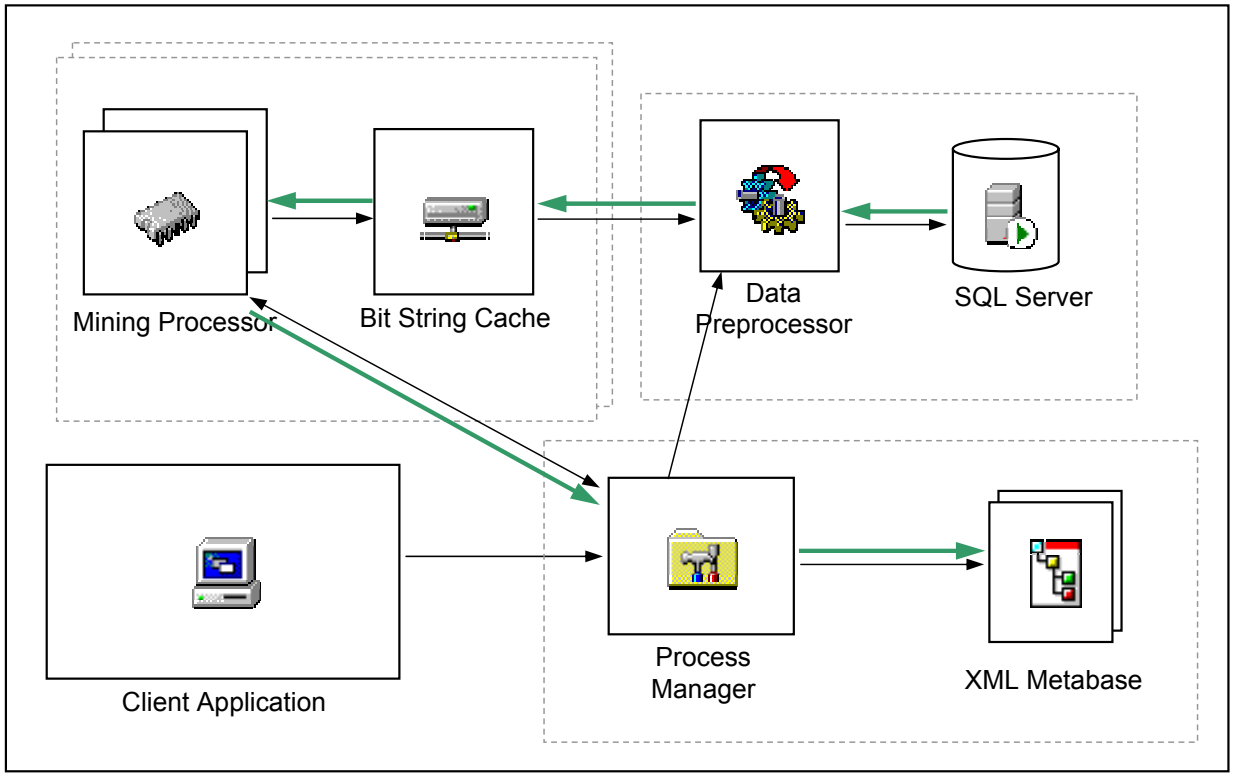
# Přeskakování hypotéz a několik souvisejících konceptů

- $A(1) \Rightarrow_{100\%} B(1)$, pak neprodlužovat antecedent
- $A(1) \Rightarrow_{0\%} B(1)$, pak neprodlužovat sukcedent
- neprodlužovat antecedent, pokud počet objektů, které jej splňují, klesne pod *Base*
- Když máme $A(1) \Rightarrow_{90\%} C(1)$ a $A(1)$ & $B(2) \Rightarrow_{91\%} C(1)$, která je cennější (a možná máme dalších 1000 podobných)?
- Když nechce být uživatel zahlcen příliš velkým množstvím hypotéz, jaká kritéria hodnocení jsou důležitá?
- Adaptace 4ft-UF Filter (utility function filter)
- Adaptace 4ft-IL (literal importance)
- Adaptace chí-kvadrát testu z KEXu

# Architektura systému (1)

- Veškerá funkcionalita napsaná do knihoven
- .NET Remoting umožňuje některé třídy „vystavit“ na serveru pro vzdálené vytvoření a volání
  - distribuované počítání „zdarma“
  - použití služeb (NT service)
- Snadná možnost zabalit kód ProcessManageru do webových služeb
  - klientská aplikace lze snadno napsat jako web

# Architektura systému (2)



# Reprezentace úlohy

- Task
- TaskFragment
- CedentSetting
- PartialCedentSetting
- LiteralSetting
- CoefficientSetting

HypothesisTrace
CedentTrace
PartialCedentTrace
LiteralTrace
CoefficientTrace

# XML metabáze – definice dat (1)

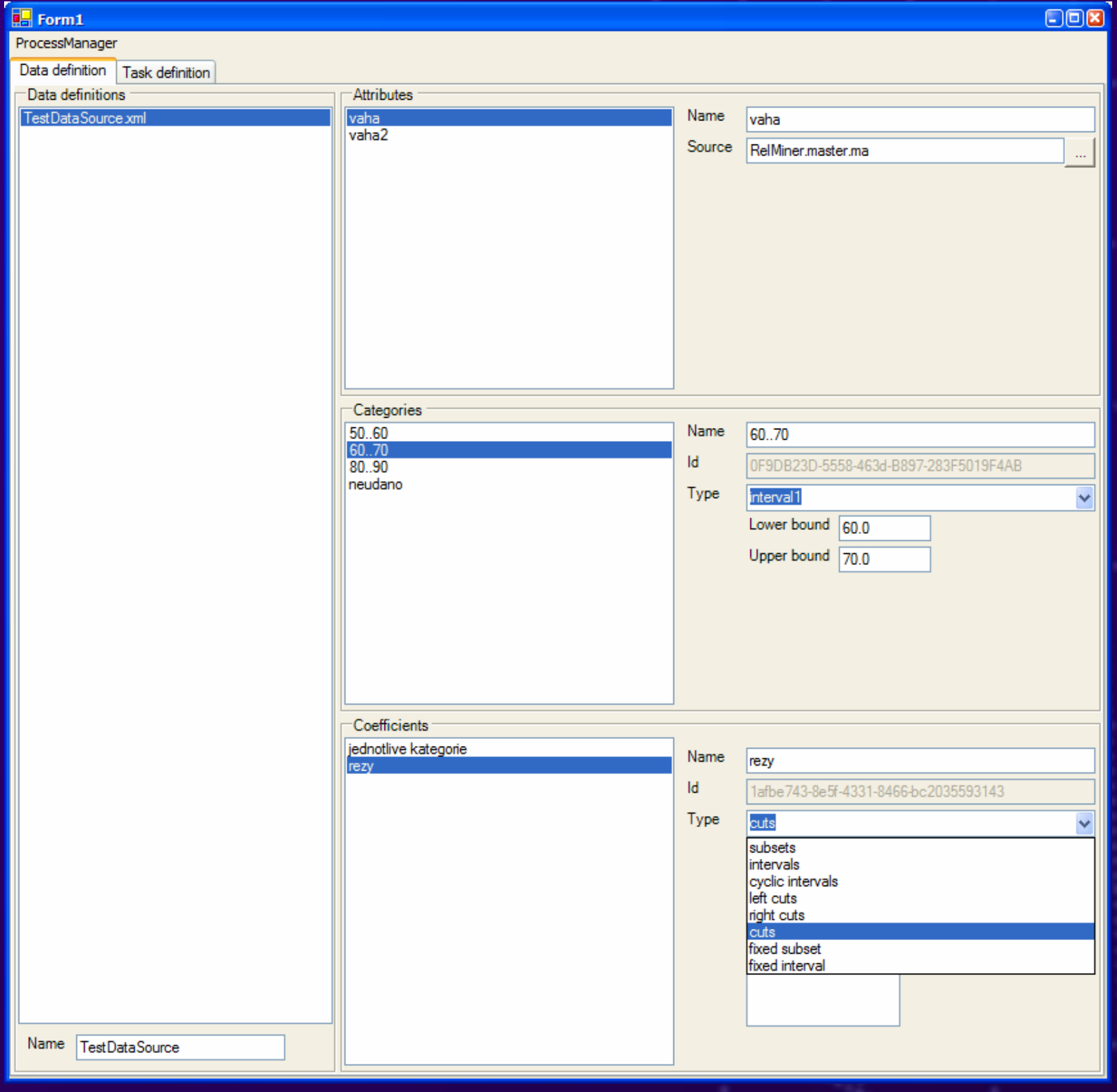
```
<Database name="stulong">
    <ConnectionString>Provider=SQLOLEDB;Data Source=(local);Initial Catalog=RelMiner;
                      Integrated Security=SSPI;</ConnectionString>
    <Table name="pacienti" id="D039663E-91E0-45e3-AFA3-752EEE59C5D0">
        <Column name="pacient_id" primaryKey="true" />
        <Column name="vaha" type="real" />
        <Column name="vyska" type="real" />
        <Column name="vek" type="real" />
        <Column name="stav" type="class" />
                <!-- 1=svobodny 2=zenaty 3=rozvedeny 4=vdovec 5=neudano -->
        <Column name="vzdelani" type="ordinal" />
                <!-- 1=zakladni 2=vyuceny 3=stredni 4=vyssi 5=vysoke -->
    </Table>

    <Table name="prohlidky" id="31791B8E-99FA-4162-9400-351AA8E05ADD">
        <Column name="prohlidka_id" primaryKey="true" />
        <Column name="pacient_id" foreignKey="pacienti.pacient_id" />
        <Column name="poradi" type="ordinal" />
        <Column name="tlak" type="real" />
        <Column name="cholesterol" type="real" />
    </Table>
</Database>
```

# XML metabáze – definice dat (2)

```
<Attribute name="vaha" id="DA66111A-7355-4ADF-BBE9-15F1AEE4E781" source="stulong.pacienti.vaha">
    <!-- typy kategorii:
        enum - pouze na sloupcich typu class, ordinal
        interval1 - <x, y)
        interval2 - <x, y>
        interval3 - (x, y>
        interval4 - (x, y)
        null

        poznamky:
        - intervaly se daji pouzit u sloupcu typu real a ordinal
        - enum lze pouzit u sloupcu typu class
        - null se muze pridat k obema typum
        - intervaly musi byt definovane tak, aby pokryly vsechny hodnoty (-INF, INF)
        - intervaly se nesmi prekryvat
    -->
    <Categories>
        <Category name="50..60" type="interval4" lowerBound="-INF" upperBound="60.0" id="CC2AC97E-76BF-4a4a-A167-94D3B9060923" />
        <Category name="60..70" type="interval1" lowerBound="60.0" upperBound="70.0" id="0F9DB23D-5558-463d-B897-283F5019F4AB" />
        <Category name="70..80" type="interval1" lowerBound="70.0" upperBound="80.0" id="A671798B-8B28-4dd6-9C96-826154AC9F96" />
        <Category name="80..90" type="interval1" lowerBound="80.0" upperBound="90.0" id="B52949C6-42A6-4ddb-8EDA-4502357225D0" />
        <Category name="90..100" type="interval1" lowerBound="90.0" upperBound="INF" id="CE0F95C1-58A1-42aa-AAB4-5581439395DF" />
        <Category name="neudano" type="null" id="17DB3192-9898-4fe3-9842-B50271DD7C5B" />
    </Categories>
    <Coefficients>
        <Coefficient name="jednotlive kategorie" type="subsets" minSize="1" maxSize="1" id="3CC21F6E-55EF-4c44-AE7B-EA07534F6F40" />
        <Coefficient name="intervaly" type="intervals" minSize="1" maxSize="2" id="7B81583C-C9BE-470b-87D9-E857C67F7F57" />
        <Coefficient name="rezy" type="cuts" minSize="1" maxSize="3" id="B990CE75-E6D7-40a1-91C7-63EF5E3EC8DE" />
    </Coefficients>
</Attribute>
```

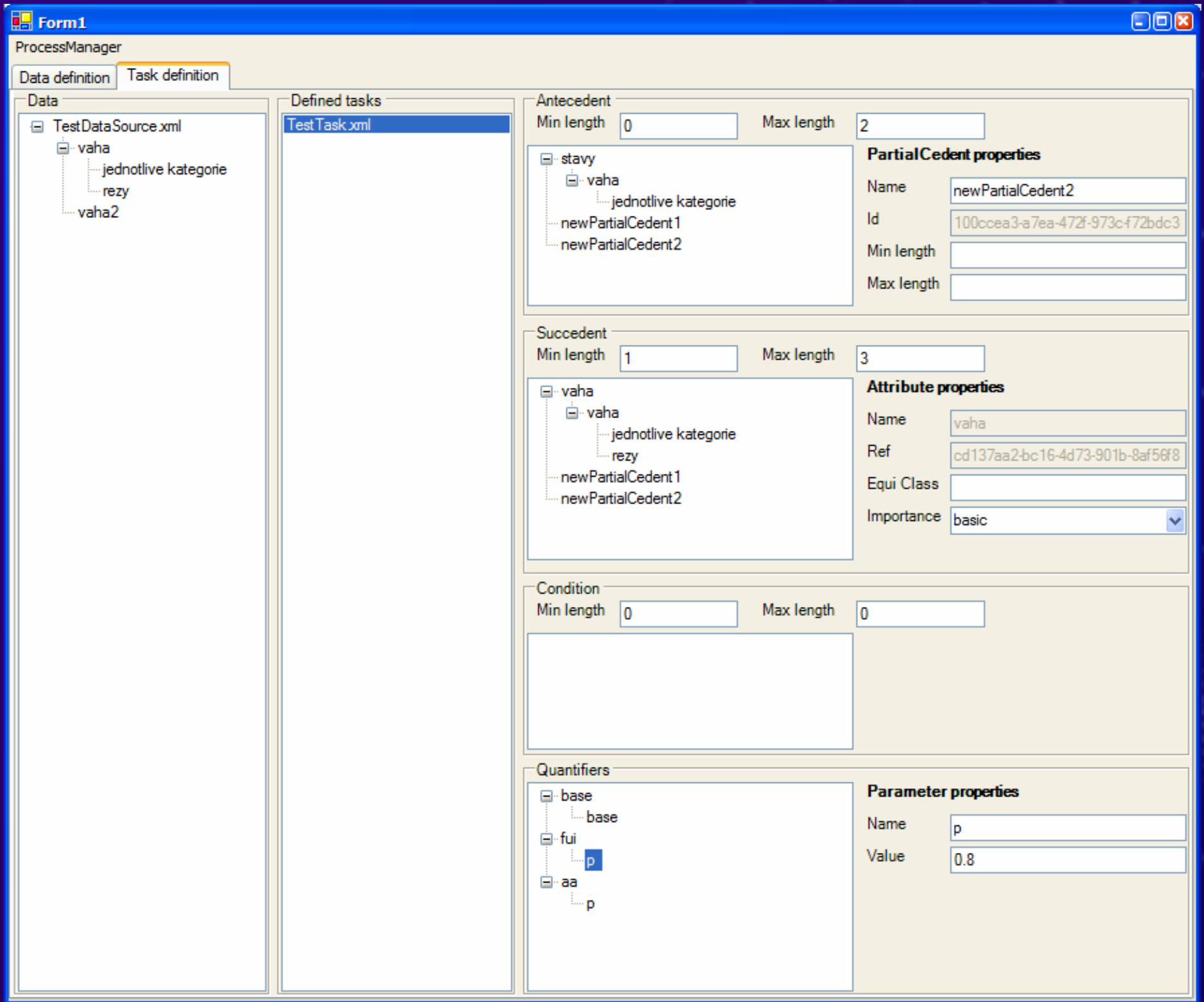
# XML metabáze – definice dat (3)

```
<VirtualAttributes>
  <AggregateAttribute name="prumer tric auto" id="51bce90a-74cb-4546-a6cc-661fcf28166d" source="stulong.prohlidky.tric" function="avg">
    <Categories>
      <Autocreate method="equifrequency">
        <Parameter name="intervals" value="5" />
        <IdPool>
          <Category id="442723FC-E621-405d-B5F6-86C8C853C9AF" /><!-- TODO: category names? -->
          <Category id="CDA69477-1C51-4b16-A349-5BC599F99F63" />
          <Category id="0475E378-1BB6-432d-8472-ED96D8D46CCC" />
          <Category id="375D040C-A2D5-40c9-9493-55D0C97FC488" />
          <Category id="54A7254D-5213-41a5-9187-EAA397C454AD" />
        </IdPool>
      </Autocreate>
    </Categories>
    <Coefficients>
      <Coefficient name="intervaly" type="intervals" minSize="1" maxSize="2" id="E0F10AC1-72E7-4b4a-8F9A-48EE2755F7BB" />
    </Coefficients>
  </AggregateAttribute>

  <ExistentialAttribute name="smrtelny tlak" id="cd26bf19-af4e-40f1-abc9-ee386fdb5bf3" source="stulong.prohlidky.tlak" predicate="tlak &gt; 200.0" />

  <AggregateAttribute name="prumer tric rucne" id="f7ee27f9-64d0-459a-9dde-a955fde67951" source="stulong.prohlidky.tric" function="avg">
    <Categories>
      <Category name="mala" type="interval4" lowerBound="-INF" upperBound="8" id="8233DAA3-47B0-4d89-A556-2B78F5381496" />
      <Category name="stredni" type="interval1" lowerBound="8" upperBound="11" id="D374C2C7-1325-4b95-B3A2-50BB55D170D0" />
      <Category name="velka" type="interval1" lowerBound="11" upperBound="INF" id="EE686405-1060-4cc6-B7CF-7317E06B5FF2" />
    </Categories>
    <Coefficients>
      <Coefficient name="jednotlive kategorie" type="subsets" minSize="1" maxSize="1" id="9591D803-1C41-41ae-BA5F-18A5BCF68021" />
    </Coefficients>
  </AggregateAttribute>
</VirtualAttributes>
```


Form1
ProcessManager
Data definition
Task definition
Data definitions
TestDataSource.xml
Name
TestDataSource
Attributes
vaha
vaha2
Name
vaha
Source
RelMiner.master.ma
...
Categories
50..60
60..70
80..90
neudano
Name
60..70
Id
0F9DB23D-5558-463d-B897-283F5019F4AB
Type
interval1
Lower bound
60.0
Upper bound
70.0
Coefficients
jednotlive kategorie
rezy
Name
rezy
Id
1afbe743-8e5f-4331-8466-bc2035593143
Type
cuts
subsets
intervals
cyclic intervals
left cuts
right cuts
cuts
fixed subset
fixed interval

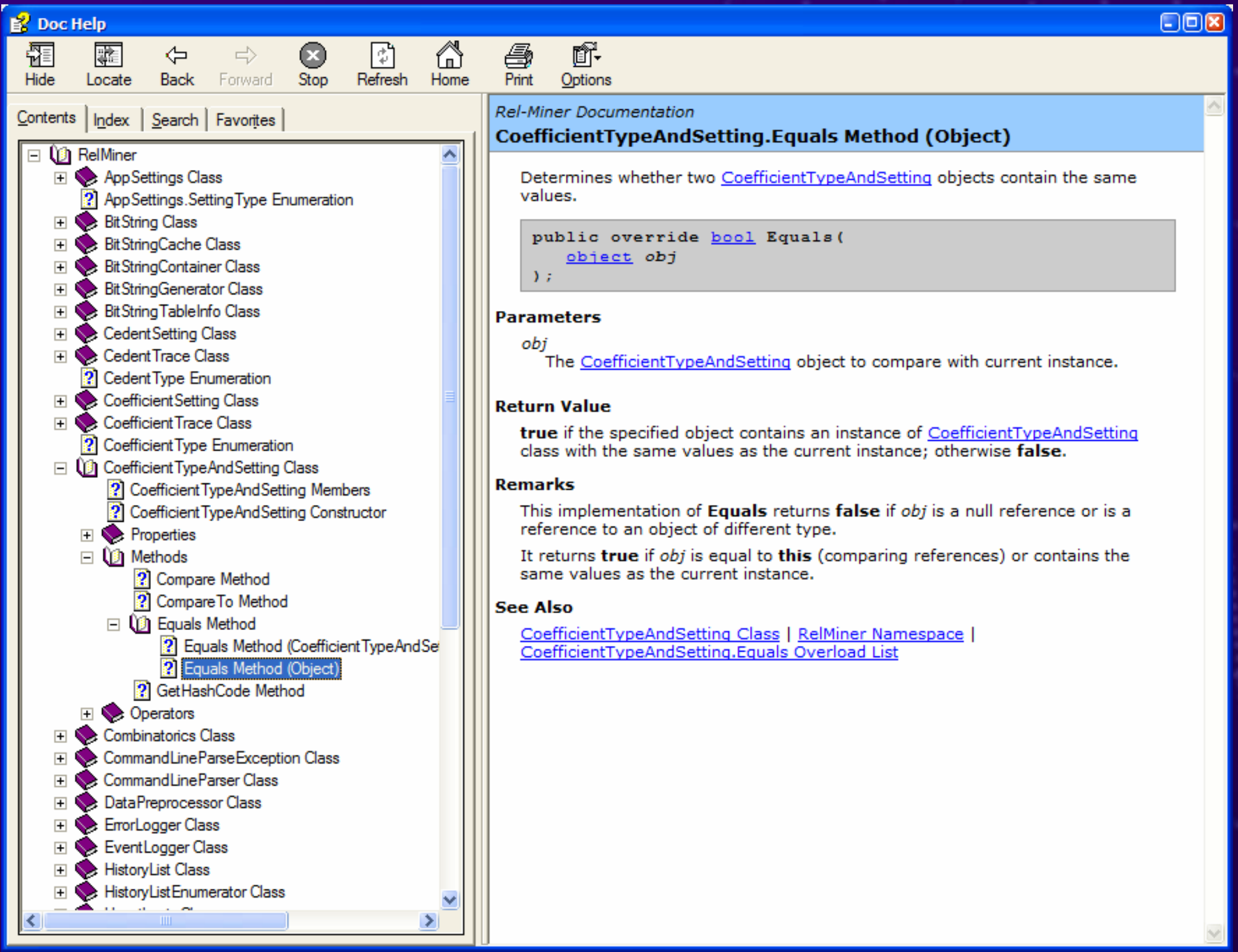
# XML metabáze – definice úlohy

```
<Antecedent minLength="0" maxLength="2">
    <PartialCedent name="stav" >
        <Attribute name="stav" ref="386dcb67-5267-47e7-a975-e9ddb9250c50" >
            <Coefficient name="jednotlive kategorie" ref="C8C9C620-20FA-402d-B900-6C97BE429EAA"/>
        </Attribute>
    </PartialCedent>
    <PartialCedent name="detaily" >
        <AggregateAttribute name="prumer tric auto" ref="51bce90a-74cb-4546-a6cc-661fcf28166d" equivalenceClass="prumerny tric">
            <Coefficient name="intervaly" ref="E0F10AC1-72E7-4b4a-8F9A-48EE2755F7BB" />
        </AggregateAttribute>
        <ExistentialAttribute name="smrtelny tlak" ref="cd26bf19-af4e-40f1-abc9-ee386fdb5bf3" />
        <HypothesesAttribute name="vysoky tlak implikuje cholesterol" ref="847FEC58-4578-4b8f-A633-C7DE1EDE290B" />
    </PartialCedent>
</Antecedent>
<Succedent>
    <PartialCedent name="vaha" maxLength="1">
        <Attribute name="vaha" ref="DA66111A-7355-4ADF-BBE9-15F1AEE4E781">
            <Coefficient name="jednotlive kategorie" ref="3CC21F6E-55EF-4c44-AE7B-EA07534F6F40" />
            <Coefficient name="rezy" ref="B990CE75-E6D7-40a1-91C7-63EF5E3EC8DE" />
        </Attribute>
        <Attribute name="vaha_3" ref="4064C9AD-4C04-4D5B-A5BE-8C5E996DF4A6">
            <Coefficient name="jednotlive kategorie" ref="821A7123-D33D-43c2-9B4A-96ACEDC42BEE" />
        </Attribute>
    </PartialCedent>
</Succedent>
<Condition />
```


Form1
ProcessManager
Data definition
Task definition
Data
TestDataSource.xml
vaha
jednotlive kategorie
rezy
vaha2
Defined tasks
TestTask.xml
Antecedent
Min length
0
Max length
2
stavy
vaha
jednotlive kategorie
newPartialCedent1
newPartialCedent2
PartialCedent properties
Name
newPartialCedent2
Id
100ccea3-a7ea-472f-973c-f72bdc3
Min length
Max length
Succedent
Min length
1
Max length
3
vaha
vaha
jednotlive kategorie
rezy
newPartialCedent1
newPartialCedent2
Attribute properties
Name
vaha
Ref
cd137aa2-bc16-4d73-901b-8af56f8
Equi Class
Importance
basic
Condition
Min length
0
Max length
0
Quantifiers
base
base
fui
p
aa
p
Parameter properties
Name
p
Value
0.8

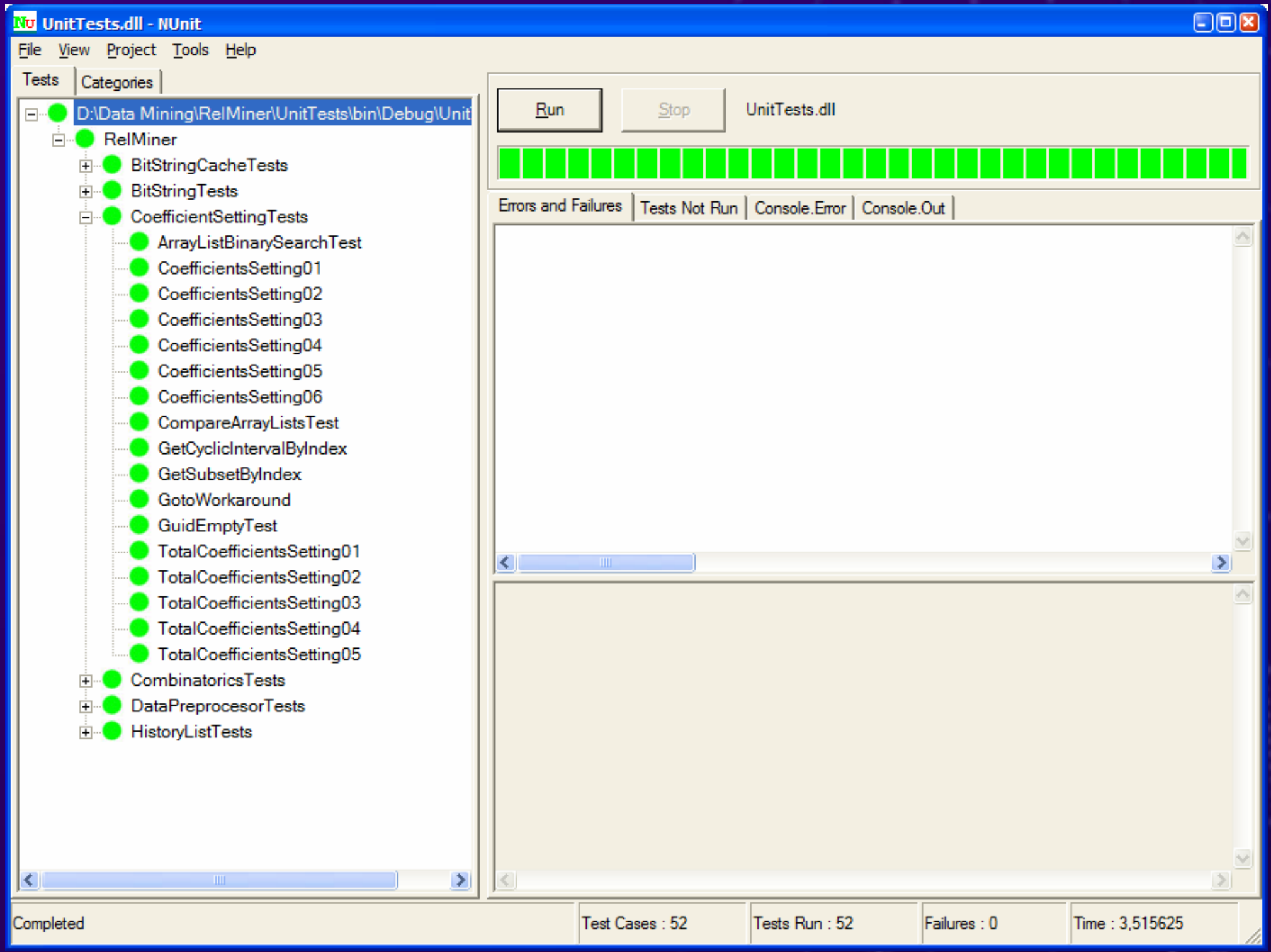
# Generování hypotéz – použití iterátorů (C#)

```
for (int i = 0; i < coefficients.Count; i++)
{
    CoefficientTypeAndSetting current = coefficients[i];
    for (int j = current.MinLength; j <= current.MaxLength; j++)
    {
        switch (current.CoefficientType)
        {
            case CoefficientType.Subsets:
                long count = Combinatorics.BinomialCoefficient(_coefficientSetting._categories.Count, j);
                for (k = 0; k < count; k++)
                {
                    yield return Combinatorics.GetSubsetByIndex(_coefficientSetting._categories, j, k);
                }

            case ...:

        }
    }
}
```


Doc Help
Hide
Locate
Back
Forward
Stop
Refresh
Home
Print
Options
Contents
Index
Search
Favorites
RelMiner
AppSettings Class
AppSettings.SettingType Enumeration
BitString Class
BitStringCache Class
BitStringContainer Class
BitStringGenerator Class
BitStringTableInfo Class
CedentSetting Class
CedentTrace Class
CedentType Enumeration
CoefficientSetting Class
CoefficientTrace Class
CoefficientType Enumeration
CoefficientTypeAndSetting Class
CoefficientTypeAndSetting Members
CoefficientTypeAndSetting Constructor
Properties
Methods
Compare Method
CompareTo Method
Equals Method
Equals Method (CoefficientTypeAndSe
Equals Method (Object)
GetHashCode Method
Operators
Combinatorics Class
CommandLineParseException Class
CommandLineParser Class
DataPreprocessor Class
ErrorLogger Class
EventLogger Class
HistoryList Class
HistoryListEnumerator Class
Rel-Miner Documentation
CoefficientTypeAndSetting.Equals Method (Object)
Determines whether two CoefficientTypeAndSetting objects contain the same values.
public override bool Equals(
object obj
);
Parameters
obj
The CoefficientTypeAndSetting object to compare with current instance.
Return Value
true if the specified object contains an instance of CoefficientTypeAndSetting class with the same values as the current instance; otherwise false.
Remarks
This implementation of Equals returns false if obj is a null reference or is a reference to an object of different type.
It returns true if obj is equal to this (comparing references) or contains the same values as the current instance.
See Also
CoefficientTypeAndSetting Class | RelMiner Namespace | CoefficientTypeAndSetting.Equals Overload List

# Dokumentace ve zdrojáku (XML)

```
/// <summary>
/// Determines whether two <see cref="RelMiner.CoefficientTypeAndSetting">CoefficientTypeAndSetting</see> objects contain the same values.
/// </summary>
/// <param name="obj">The <see cref="RelMiner.CoefficientTypeAndSetting">CoefficientTypeAndSetting</see> object to compare with current
        instance.</param>
/// <returns><b>true</b> if the specified object contains an instance of <see
        cref="RelMiner.CoefficientTypeAndSetting">CoefficientTypeAndSetting</see> class with the same values as the current instance; otherwise
        <b>false</b>.</returns>
/// <remarks>
/// <para>This implementation of <b>Equals</b> returns <b>false</b> if <i>obj</i> is a null reference or is a reference to an object of different
        type.</para>
/// <para>It returns <b>true</b> if <i>obj</i> is equal to <b>this</b> (comparing references) or contains the same values as the current
        instance.</para>
/// </remarks>
public override bool Equals(object obj)
{
    CoefficientTypeAndSetting that = obj as CoefficientTypeAndSetting;
    if ((object) that != null)
    {
        return ((object) this == (object) that) || (this.CoefficientType == that.CoefficientType) && (this.MinLength == that.MinLength) &&
        (this.MaxLength == that.MaxLength);
    }
    else
    {
        return false;
    }
}
```


UnitTests.dll - NUnit
File View Project Tools Help
Tests Categories
D:\Data Mining\RelMiner\UnitTests\bin\Debug\Unit
RelMiner
BitStringCacheTests
BitStringTests
CoefficientSettingTests
ArrayListBinarySearchTest
CoefficientsSetting01
CoefficientsSetting02
CoefficientsSetting03
CoefficientsSetting04
CoefficientsSetting05
CoefficientsSetting06
CompareArrayListsTest
GetCyclicIntervalByIndex
GetSubsetByIndex
GotoWorkaround
GuidEmptyTest
TotalCoefficientsSetting01
TotalCoefficientsSetting02
TotalCoefficientsSetting03
TotalCoefficientsSetting04
TotalCoefficientsSetting05
CombinatoricsTests
DataPreprocesorTests
HistoryListTests
Run
Stop
UnitTests.dll
Errors and Failures
Tests Not Run
Console.Error
Console.Out
Completed
Test Cases : 52
Tests Run : 52
Failures : 0
Time : 3,515625