Meta-set calculus as mathematical basis for creating abstract, structured data store querying technology

Logics Research Centre SIA
Latvia University of Agriculture
Abstraction of object set: Meta-set

- Metaset consists of:
  <type constraints>
  - Defines types in object databases
  - Defines tables in relational databases
  - Defines basic structures in other databases

[property-value constraints] are combinations of metadata and value (data);
- In relational context it can be interpreted as property-value constraint;
- In key-value stores it can be interpreted as key-value pair;

{object set constraints} which defines relationships between 2 sets

- Example:
  <Person>[FirstName="Mikus", LastName="Vanags"]

- Meta-set can be interpreted as query to data store. Equivalent query to relational database:
  SELECT * FROM PERSONS
  WHERE FirstName="Mikus" AND LastName="Vanags"
Mostly people use many abstractions, but do not interpret them as abstractions.

\[
\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \cdots + \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n + 1)!}
\]

\[
\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \cdots + \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}
\]

Trigonometric functions are abstractions of infinite Tailor series. Without these abstractions many things would not be possible!
Why meta-sets are so important?

• **Meta-set describes set of unknown number of objects** (theoretically it could be even infinity).

• Second order predicate logic engine could work without meta-sets, but it still could ended with loading in memory all database content during deduction process.
Querying process in Decentralized Deduction Engine

Result
- Objects

Deduction Process
- Facts containing Meta-sets
- Rules

Deduction Process Result
- Meta-sets

Server
- Facts and rules
- Business Objects

Client
- Question

http://www.getlang.org
Meta-set calculi similarity to constraint logic programming

• In addition to constraint logic programming

Meta-set calculi:
  – Contain type information
  – Support object set abstractions
  – work with many constraint stores

• Both CLP and MSC requires modifications in logic programming engine.
Physical model used in all examples
Meta-set matching and unification with meta-sets

- Meta-set matching differs from object matching, because meta-sets are like small parts of larger query that is being built and not all differences in meta-sets are considered as failures in matching. For example:

  1) something(<Dog>) matches with something(<Dog>)
  2) something(<Dog>) matches with something(<Pet>)
  3) something(<Dog>) matches with something(<Animal>)
  4) something(<Dog>) does not match with something(<Cat>)
  5) something(<Dog>) does not match with something(<Person>)
  6) something(<Dog>) matches with something(x).

- In unification, when meta-set type constraints matches and if variable was used in matching, the meta-set, to which the variable references, will contain updated list with the most specific type constraints from both meta-sets, merged lists of both meta-set property-value constraints and set-constraints.
Difference between TermNode syntax and TermExpression syntax

• **TermNode** syntax is not type safe, but expressions are processed at compile time.

• **TermExpression** syntax is more type safe (still not fully type safe), but TermExpressions are evaluated at runtime.

• We wanted to design **general purpose language extensions** to support meta-sets (and get performance + full type safety), but discovered, that large software vendors can just ignore us, we needed orthogonal (independent solution)...

http://www.getlang.org
Proposed Solution

• Abstract data querying language: “Get” or more googlable form “GetLang”

• which is based on our invented calculus: “Meta-set calculus” – extension of second order predicate calculus

• and our calculus implementation is named: “Decentralized Deduction Engine” or simply DDE
GetLang use cases

Class declarations according to available metadata

GetLang code

Code generation

C# code

Java code

Python code

... (abstractions of queries)

Query execution

Query

Query generation

Logical deduction

Db connection

Logic interpretation

Query rules in form suitable for logical deduction

Code execution

Code generation

GetLang code

Resulting objects

http://www.getlang.org

Logics Research Centre
GetLang example for 4 queries reusing common query parts

using DataStructures;

metaset Invoice a;
metaset TransportationInvoice b;
metaset AcceptanceInvoice c;
metaset b,c d;
metaset a,d e;

parameter Warehouse Warehouse;
parameter DateTime DateFrom;
parameter DateTime DateTo;

inPeriod(e) : e.DealDate >= DateFrom, e.DealDate < DateTo;
atWarehouse(a) : a.Warehouse = Warehouse;
fromWarehouse(d) : d.WarehouseFrom = Warehouse;
toWarehouse(d) : d.WarehouseTo = Warehouse;
order(d) : OrderAscending(d.DealDate, d.DealNumber);

buyingAtWarehouseInPeriod(a) : atWarehouse(a), inPeriod(a), order(a);
transportationFromWarehouseInPeriod(b) : fromWarehouse(b), inPeriod(b);
toWarehouseInPeriod(d) : toWarehouse(d), inPeriod(d);

BuyingAtWarehouseInPeriod = buyingAtWarehouseInPeriod(a);
TransportationFromWarehouseInPeriod = transportationFromWarehouseInPeriod(b);
TransportationToWarehouseInPeriod = toWarehouseInPeriod(b);
AcceptedAtWarehouseInPeriod = toWarehouseInPeriod(c);

From this code will be possible to generate code in general purpose programming languages like C#, Java and others...
Comparison of db4o querying technologies
(integrated in general purpose programming languages)

// soda query definition execution example
var query = _db.Query();
query.Constrain(typeof(Invoice));
query.Descend("_warehouse").Constrain(_warehouse);
query.Descend("_dealDate").Constrain(dateFrom).Greater().Equal();
query.Descend("_dealDate").Constrain(dateTo).Smaller();
query.Descend("_dealDate").OrderAscending();
query.Descend("_dealNumber").OrderAscending();
var results = query.Execute().OfType<Invoice>();

// linq query definition and execution example
var results = (from Invoice invoice in _db
where
    invoice.Warehouse == _warehouse &&
    invoice.DealDate >= dateFrom &&
    invoice.DealDate < dateTo
orderby invoice.DealDate, invoice.DealNumber
select invoice).ToList();

// dde query execution example
// _dde is instance of QueryingLogic class generated from GetLang code
var results = _dde.BuyingAtWarehouseInPeriod(_warehouse, dateFrom, dateTo);

SODA queries:
- performs excellent,
- are not type safe,
- can’t reuse existing query parts,
- are difficult to serialize

LINQ queries:
- not always performs excellent,
- are strongly typed,
- can’t reuse existing query parts,
- are difficult to serialize

DDE queries:
- performs as fast as SODA,
- reuses existing query parts,
- are strongly typed,
- can be easily serialized,
- and used in distributed systems
Layered structure of DDE and our responsibilities

Our responsibility

Hosting company responsibility

Other programmers responsibility

DB vendor responsibility