Отображение графовой модели данных в каноническую объектно-фреймовую информационную модель при создании систем интеграции неоднородных информационных ресурсов

Сергей Ступников

Институт проблем информатики, Российская академия наук

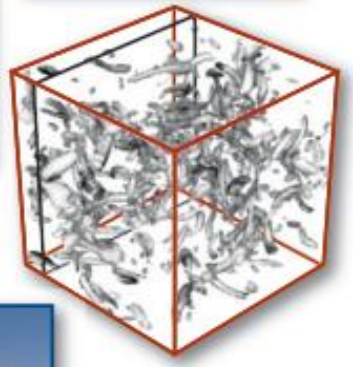
План

- Четвертая научная парадигма
- Графовые модели данных и их унификация
- Модель данных атрибутированных графов
- Отображение модели атрибутированных графов в каноническую информационную модель
  - Отображения языка определения данных
  - Отображение языка манипулирования данными
- Сохранение информации и семантики операций ЯМД при отображении
- Родственные исследования и направления дальнейшей работы

Science Paradigms

- Thousand years ago:
  science was **empirical**
  describing natural phenomena
- Last few hundred years:
  theoretical branch
  using models, generalizations
- Last few decades:
  a **computational** branch
  simulating complex phenomena
- Today: **data exploration** (eScience)
  unify theory, experiment, and simulation
  - Data captured by instruments or generated by simulator
  - Processed by software
  - Information/knowledge stored in computer
  - Scientist analyzes database/files using data management and statistics

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G p}{3} - \mathrm{K}\frac{c^2}{a^2}$$



Jim Gray on eScience

Четвертая парадигма

- Интенсивное использование данных, данные как доминирующий фактор

- Необходимы
  - новые подходы к концептуализации, организации и реализации информационных систем
  - методы и средства оперирования данными, объемы которых выходят за рамки возможностей современных СУБД
  - подходы, позволяющих справляться с разнообразием массово и хаотично развивающихся языков и моделей данных
    - NoSQL-модели
    - онтологические и семантические модели
    - модели, основанные на многомерных массивах
    - графовые модели
    - …

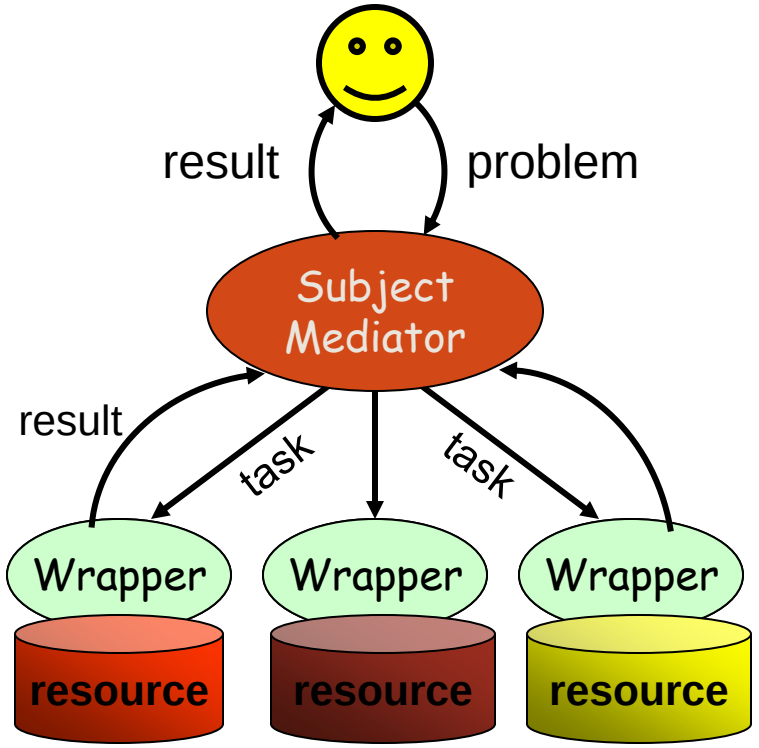
Цель работы

- Унификация графовой модели данных для *виртуальной* или *материализованной* интеграции ресурсов при создании федеративных баз данных или хранилищ данных

- СУБД, основанные на графовых моделях - новый вид ресурсов для интеграции вместе с привычными ресурсами – реляционными и объектными СУБД, веб-сервисами, …
- Графовые модели
  - Информация о взаимосвязях между данными или их топологии является более важной (или настолько же важной), как сами данные
  - Данные и/или схема – граф
  - Манипулирование данными
    - трансформация графов
    - пути, подграфы, связность …
  - *Применения* - системы управления и анализа сложных сетей – социальных, биологических, информационных, транспортных, телекоммуникационных …

Виртуальная интеграция в предметных посредниках

- Задача формулируется в терминах схемы посредника, затем
- трансформируется в набор подзадач (запросов) к ресурсам, зарегистрированным в посреднике;
- подзадачи исполняются на ресурсах, результаты возвращаются в посредник;
- результаты объединяются и представляются пользователю.



Унификация информационных моделей

- *Каноническая информационная модель* - общий язык, унифицирующий разнообразные модели ресурсов
- *Унификация* исходной модели данных - ее отображение в каноническую модель, сохраняющее информацию и семантику операций языка манипулирования данными (ЯМД)
  - унификация должна быть доказуемо правильной
  - унификация моделей ресурсов является необходимым предусловием для регистрации ресурсов в посреднике
- В качестве канонической модели в данной работе рассматривается язык СИНТЕЗ - комбинированная слабоструктурированная и объектная модель данных, нацеленная на разработку предметных посредников для решения задач в средах неоднородных ресурсов
  - Разработан прототип программных средств для поддержки среды предметных посредников
  - Проведена унификация структурированных, онтологических, сервисных, процессных моделей

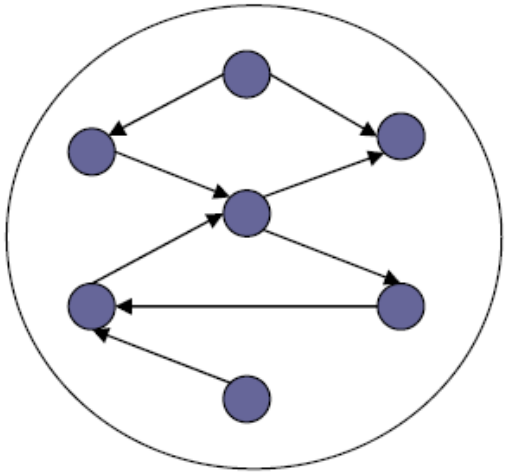
Графы, гипервершины, гиперграфы

N = set of simple nodes

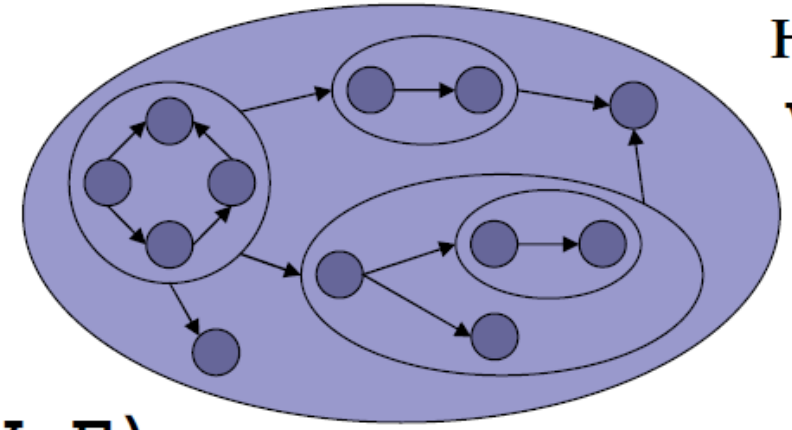
H = set of hypernodes

Graph

$V \subseteq N$ $E \subseteq V \times V$

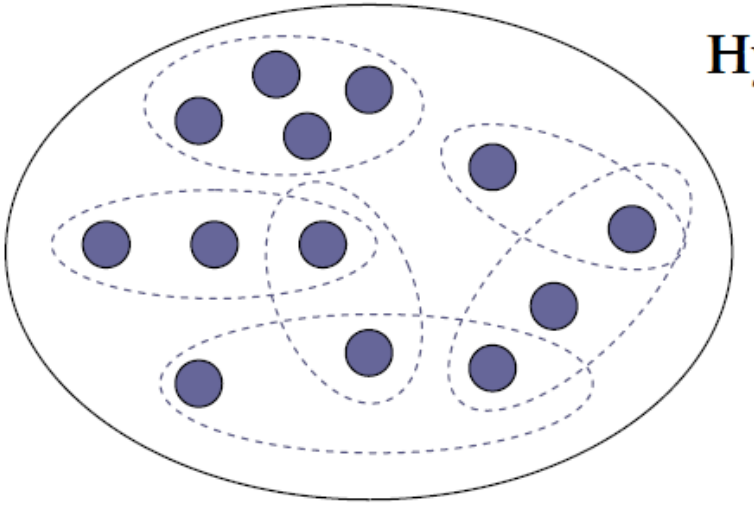


$G = (V, E)$



Hypernode

$V = N \cup H$

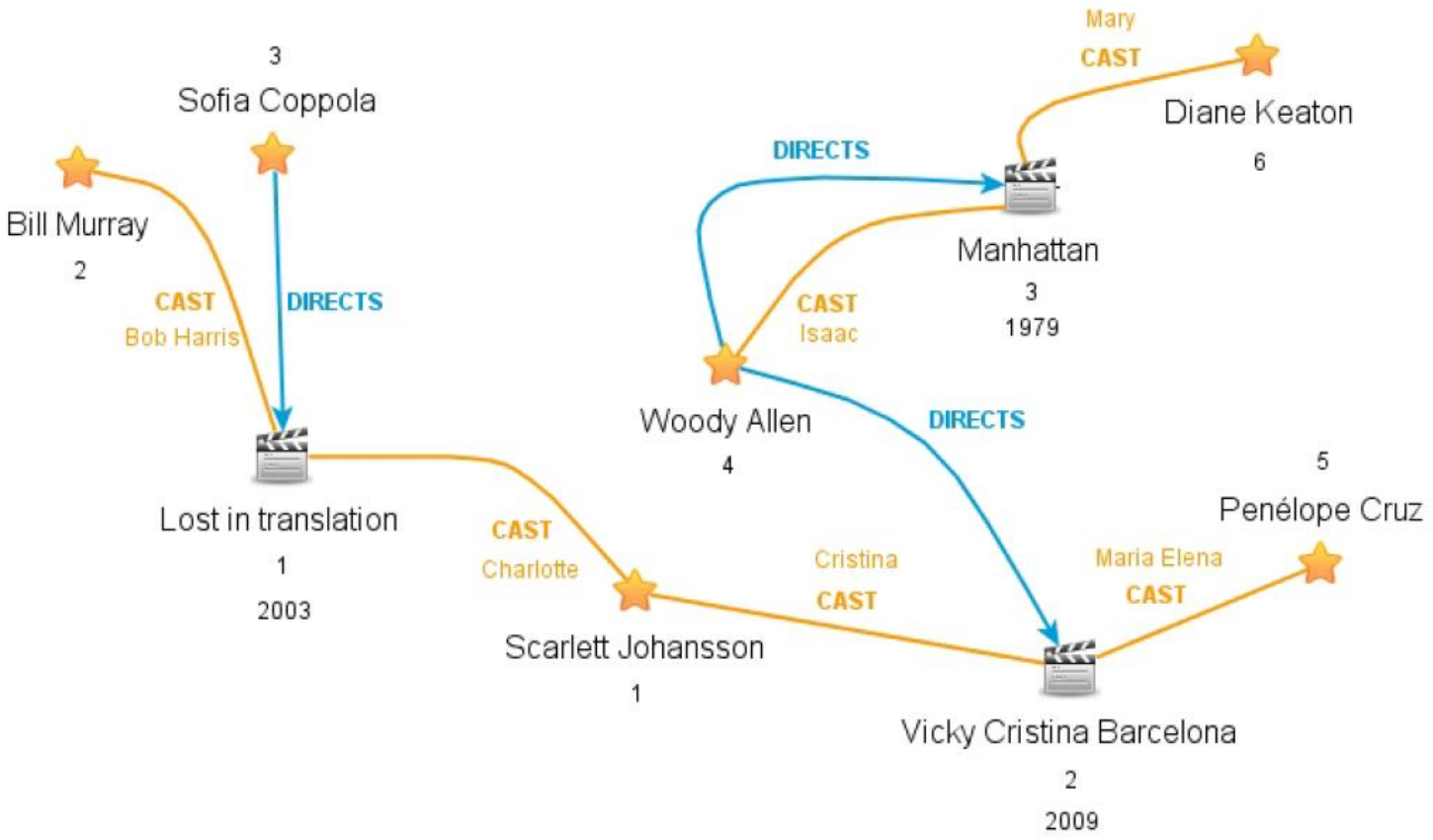


Hypergraph

$E \subseteq P(V)$

Модель данных атрибутированных графов

- *Атрибутированный граф* - атрибуты (свойства) приписываются ребрам и/или вершинам графа
  - Neo4j, Dex, InfiniteGraph, OrientDB, VertexDB, Filament, OQGraph, Horton, InfoGrid



ЯМД графовой модели

- API, скриптовые языки
  - доступ к структуре графа
  - методы обхода графа
  - алгоритмы на графах

- Декларативные языки (Cypher)
  - Поддержан Neo4j
  - Родство с SQL и SPARQL
  - Возможности
    - смежность вершин и ребер
    - достижимость по путям фиксированной длины
    - достижимость по простым регулярным путям
    - поиск кратчайших путей
    - поиск подграфов по образцу
    - …

Отображение ЯОД (I)

Графовая модель

VT(Cinema) = { people, movie }

ET(Cinema) = { cast, directs }

A(movie) = {
<id, long>,
<title, string>,
<year, integer>}

A(people) = {
<id, long>,
<name, string>}

cast = <{<character, string>}, false, false, undefined, undefined>

directs = < ∅, true, true, people, movie>

Каноническая объектная модель

{ Cinema; in: module;
{ vertices; in: class; … },
{ edges: in: class; … };
{ movie; in: class; superclass: vertices;
instance_type: {
id: long;
title: string;
year: integer; }; }
{ directs; in: class; superclass: edges;
instance_type: {
edgeConstr: {in: invariant;
{{ all e/directs.inst (directs(e) ->
people(e.startVertex) &
movie(e.endVertex)) }}
}; };
} }

Отображение ЯОД (II)

- *Схема* отображается в одноименный модуль, включающий классы, содержащие вершины и ребра графа
- *Тип вершины* представляется одноименным классом - подклассом класса всех вершин vertices
- *Тип ребра* представляется одноименным классом - подклассом класса всех вершин edges
- *Атрибуты* типов вершин и ребер представляются атрибутами типа экземпляров соответствующих классов
- Между встроенными типами графовой модели (int8, int64, double и т.д.) и встроенными типами объектной модели (short, long, double) устанавливается взаимно-однозначное соответствие
- …

Виртуальная и материализованная интеграция

- При *виртуальной* интеграции отображение ЯМД обеспечивает возможность трансляции программ на языке посредника в запросы на языке ресурсов
- При *материализованной* интеграции данные извлекаются из ресурса и представляются в хранилище в канонической модели. При этом программы на языке канонической модели исполняются непосредственно на данных.
  - Отображение ЯМД нужно лишь для того, чтобы убедиться, что отображение моделей сохраняет информацию и семантику операций.
  - Семантически правильное отображение служит базой для процесса Извлечения-Преобразования-Загрузки (ETL), формирующего из данных ресурса данные хранилища: ETL-процесс может быть выражен только в терминах канонической модели.

Отображение ЯМД

Каноническая объектная модель

- Datalog-подобный язык в объектной среде

```
q([colleague_name]) :-
people(scarlett/[name]),
movies(m),
people(colleague/
    [collegue_name: name]),
cast(c1),
cast(c2),
c1.isValidEdge(m, scarlett),
c2.isValidEdge(m, colleague),
scarlett.name =
  “Scarlett Johansson”,
colleague.name.like(“*Cruz*”).
```

Графовая модель

- Cypher

```
START scarlett =
  node:node_auto_index(
      name = 'Scarlett Johansson')
MATCH
  m-[c1:cast]-scarlett,
  m-[c2:cast]-collegue
WHERE
  collegue.name =~ /*Cruz*/
RETURN collegue.name
```

Вернуть имена актеров по фамилии Круз, игравших в фильмах вместе со Скарлетт Йохансон

Сохранение информации и семантики операций ЯМД при отображении

- AMN - теоретико-модельная нотация, основанная на теории множеств и типизированном языке первого порядка
  - Спецификации AMN - абстрактные машины
  - Интегрированно рассматриваются спецификация пространства состояний и поведения машины
  - Формализуется отношение *уточнения* (спецификация B уточняет спецификацию A, если пользователь может использовать B вместо A, не замечая факта замены A на B)

- Метод доказательства сохранения информации и семантики операций
  - θ - отображение модели исходной модели S в целевую модель T
  - семантика моделей представляется в виде абстрактных машин AMN M_S и M_T
    - структуры данных моделей – классы, массивы представляются переменными машин
    - структур данных представляются инвариантами машин
    - операции моделей данных представляются операциями машин
  - отображение θ сохраняет информацию и семантику операций, если машина M_S уточняет машину M_T

Семантика объектной модели в AMN

```
REFINEMENT  ObjectDM
CONSTANTS
  c_edges, c_vertices,  a_startVertex, a_endVertex, c_edges_instance_type
ABSTRACT_VARIABLES
   m_directed, m_restricted, m_startVertexType, m_endVertexType, isValidEdge, ...
INVARIANT
  c_edges: classNames & c_vertices: classNames &
  isValidEdge: objectsOfClass(c_edges) * objectsOfClass(c_vertices) * objectsOfClass(c_vertices) --> BOOL &
   …
OPERATIONS
     deleteVertex(attr, cond) =
     PRE
        attr : dom(attributeNames) &  cond : INT --> BOOL & attributeType(attr) = Integer
     THEN
          objectsOfClass(c_vertices) :=
          objectsOfClass(c_vertices) -
          { vert | vert: objectsOfClass(c_vertices) & vert: dom(adtAttributeValue(attr)) &
                    cond(integerAttributeValue(attr)(vert)) = TRUE }
     END
END
```

Семантика графовой модели в AMN

```
REFINEMENT GraphDM
REFINES ObjectDM
ABSTRACT_VARIABLES
   vertexTypeIDs, edgeTypeIDs, attributeIDs, attributes, vertices, vertixType, edges, edgeType, ...
INVARIANT
   vertexTypeIDs: POW(NAT) & edgeTypeIDs: POW(NAT) & attributeIDs: POW(NAT) &
   attributes: vertexTypeIDs & vertices: POW(NAT) & vertixType: vertices --> vertexTypeIDs &
   edges: POW(NAT) & edgeType: edges --> edgeTypeIDs &
    ...
OPERATIONS
deleteVertex(attr, cond) =
     PRE
        attr: attributeIDs & cond: INT --> BOOL & attributeTyping(attr) = Integer
     THEN
          vertices := vertices -
          { vert | vert: vertices & attr: attributes(vertixType(vert)) &
                   cond(g_integerAttributeValue(vert, attr)) = TRUE }
     END
END
```

Инвариант уточнения

- Связывает переменные уточняемой и уточняющей машин, добавляется в инвариант уточняющей машины

- Множество имен типов графовой модели совпадает с множеством имен классов объектной модели (за исключением предопределенных классов c_edges, c_vertices)

  ran(typeName) = classNames - {c_edges, c_vertices}

- Вершины и ребра графовой базы данных соответствуют объектам классов c_vertices и c_edges:

  vertices = objectsOfClass(c_vertices) & edges = objectsOfClass(c_edges)

- Значения атрибутов вершин и ребер графовой модели совпадают со значениями соответствующих атрибутов соответствующих объектов

```
!(vert, attr).(vert: vertices & attr: attributeIDs =>
  ((vert |-> attr) : dom(g_integerAttributeValue) =>
      g_integerAttributeValue(vert, attr) = integerAttributeValue(attr)(vert)) )
!(edg, attr).(edg: edges & attr: attributeIDs =>
  ((edg |-> attr) : dom(g_integerAttributeValue) =>
      g_integerAttributeValue(edg, attr) = integerAttributeValue(attr)(edg))   )
```

Доказательство уточнения

- Машины ObjectDM и GraphDM загружены в инструментальное средство доказательства уточнения Atelier B
- Автоматически сгенерированы теоремы уточнения
  - для операции deleteVertex – 15 теорем
- Теоремы доказываются автоматически или интерактивно
  - для операции deleteVertex – все теоремы доказаны автоматически
- Доказательство проводится для всех операций ЯМД

Дальнейшая работа

- выбор конкретных графовых моделей, основанных на простых и атрибутированных графах
  - построение трансформаций, реализующих отображение
- расширение инструментальных средств поддержки предметных посредников для виртуальной интеграции графовых баз данных
- применение технологии предметных посредников для решения научных задач над множеством неоднородных ресурсов, включающим графовые базы данных