Virtual Integration of Heterogeneous Data and Data Model Unification

Subject Mediators

Sergey Stupnikov

Institute of Informatics Problems, RAS

sstupnikov@ipi.ac.ru

Outline

- Mediators according to Wiederhold
- Виды посредников, создание посредников
- Архитектура исполнительных механизмов слоя предметных посредников
- Пример схемы посредника, запрос к ней и семантических отображений
- Основные принципы планирования исполнения запросов
- Архитектура адаптеров
- Решение задач в среде посредников

History: What is Mediation [Wiki]

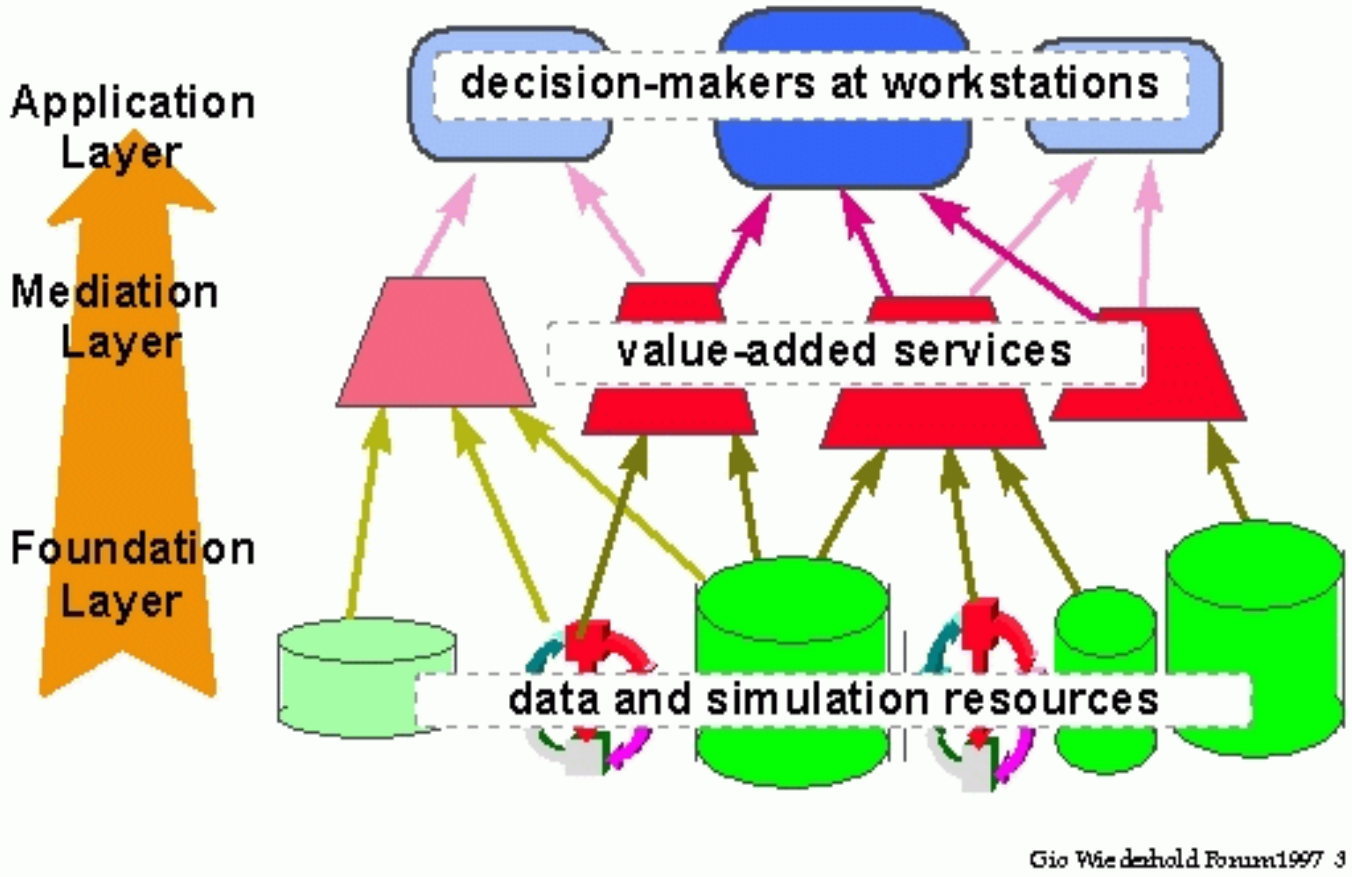
- *Mediation*, a form of *appropriate dispute resolution*, aims to assist two (or more) disputants in reaching an agreement
  - Disputants may use mediation in a variety of disputes, such as *commercial*, *legal*, *diplomatic*, *workplace*, *community and family matters*
  - The activity of mediation in itself appeared in very ancient times. Historians presume early cases in Phoenician commerce (but suppose its use in Babylon, too). The practice developed in Ancient Greece, then in Roman civilization
  - The Romans called mediators by a variety of names, including *internuncius*, *medium*, *intercessor, philantropus, interpolator, conciliator, interlocutor, interpres*, and finally *mediator*.
  - Some cultures regarded the mediator as a *sacred figure*, worthy of particular respect; and the role partly overlapped with that of traditional *wise men* or *tribal chief*
- "Conciliation" (примирение, улаживание разногласий) sometimes serves as an umbrella-term that covers all mediation and facilitative and advisory dispute-resolution processes. One significant difference between conciliation and mediation lies in the fact that conciliators possess expert knowledge of the domain in which they conciliate.

Mediator according to Gio Wiederhold (Stanford Un.)

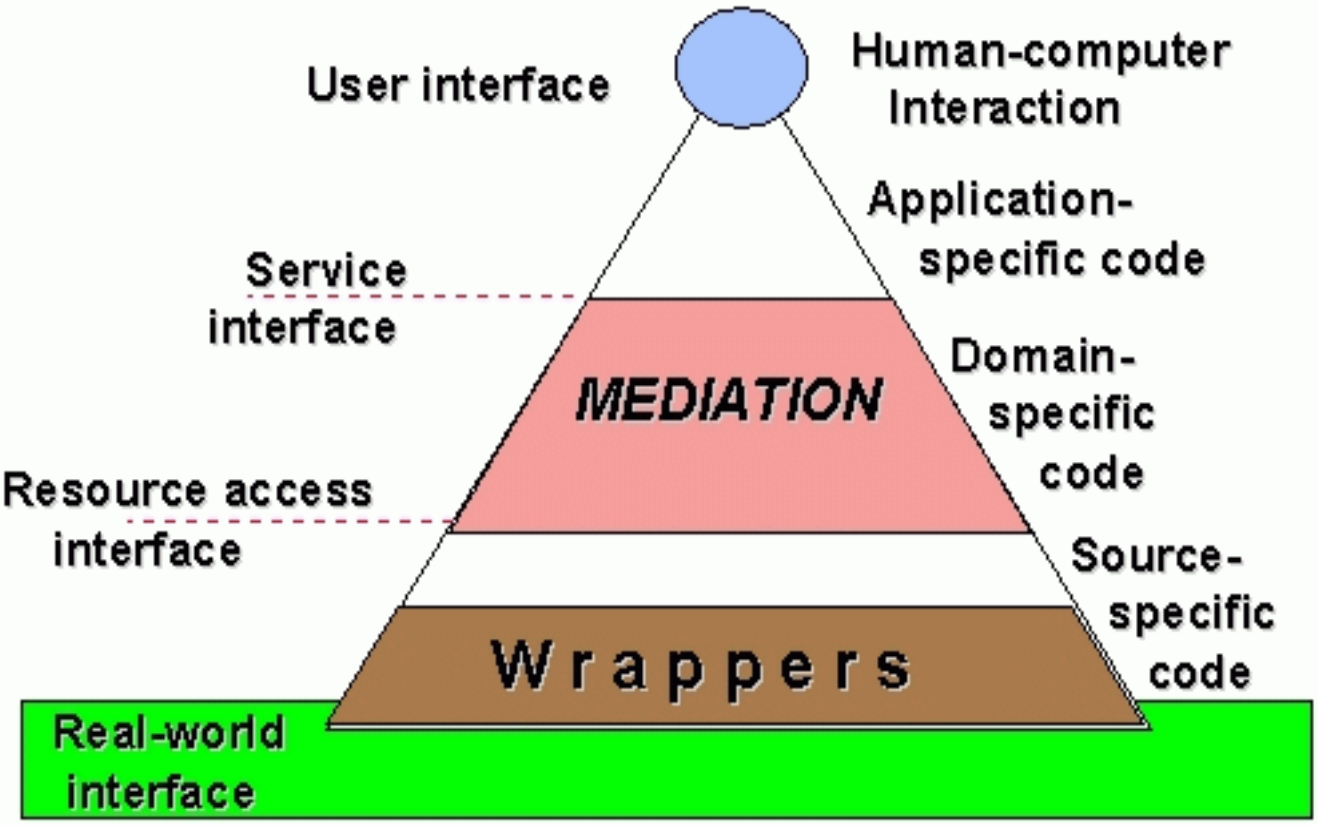
- **Data describes specific instances and events**. Data may gathered automatically or clerically. The correctness of data can be verified vis-a-vis the real world
- **Knowledge describes abstract classes**. Each class typically covers many instances. Experts are needed to gather and formalize knowledge. Data can be used to disprove knowledge
- A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications

- An important objective of the architecture is the ability to utilize a variety of information sources without demanding that they be brought into a common format and with only minimal requirements on their interfaces
- Mediator modules comprise a layer of intelligent middleware services in information systems linking data resources and application programs

Transform Data to Information

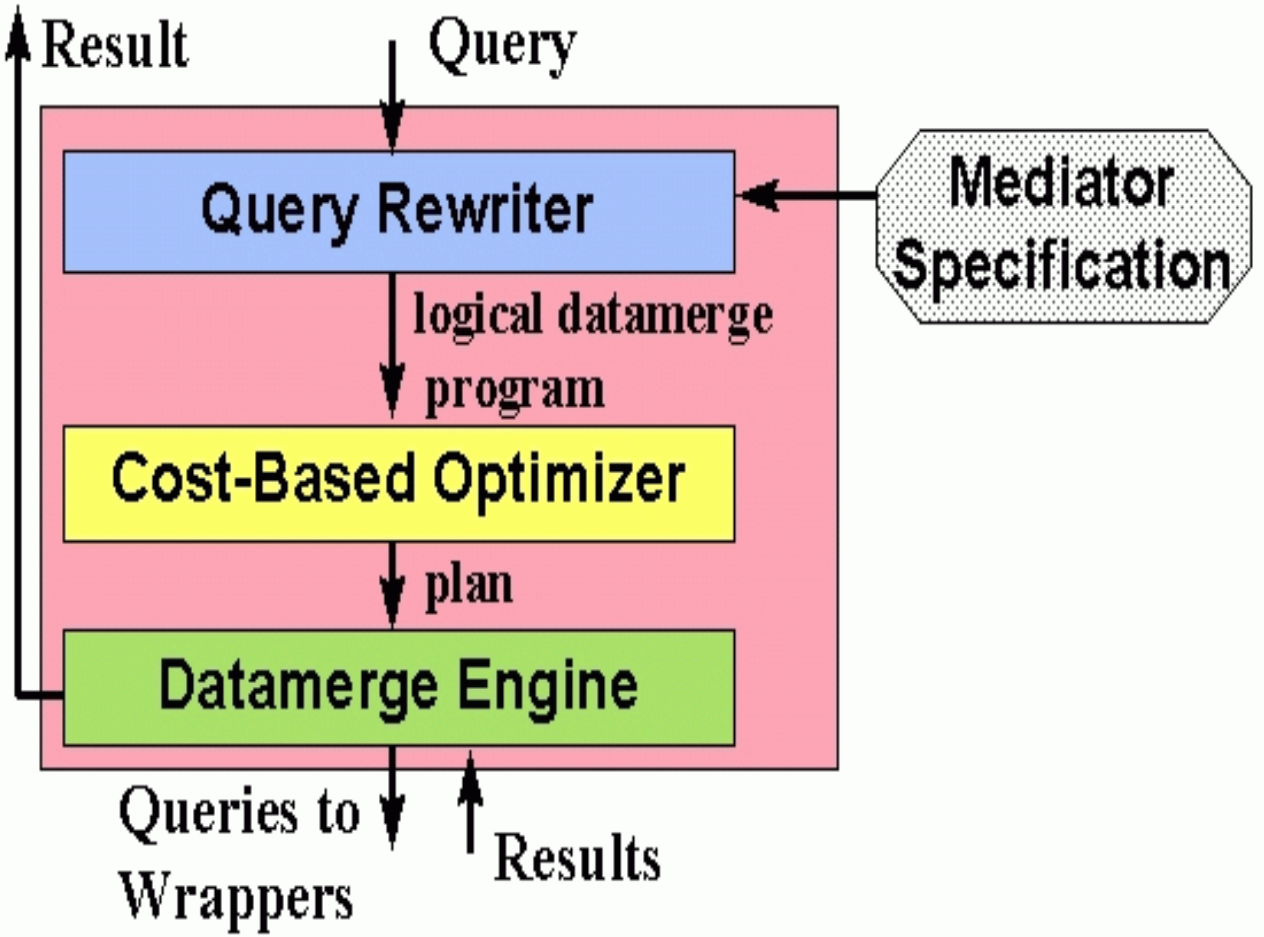


Functional Layer



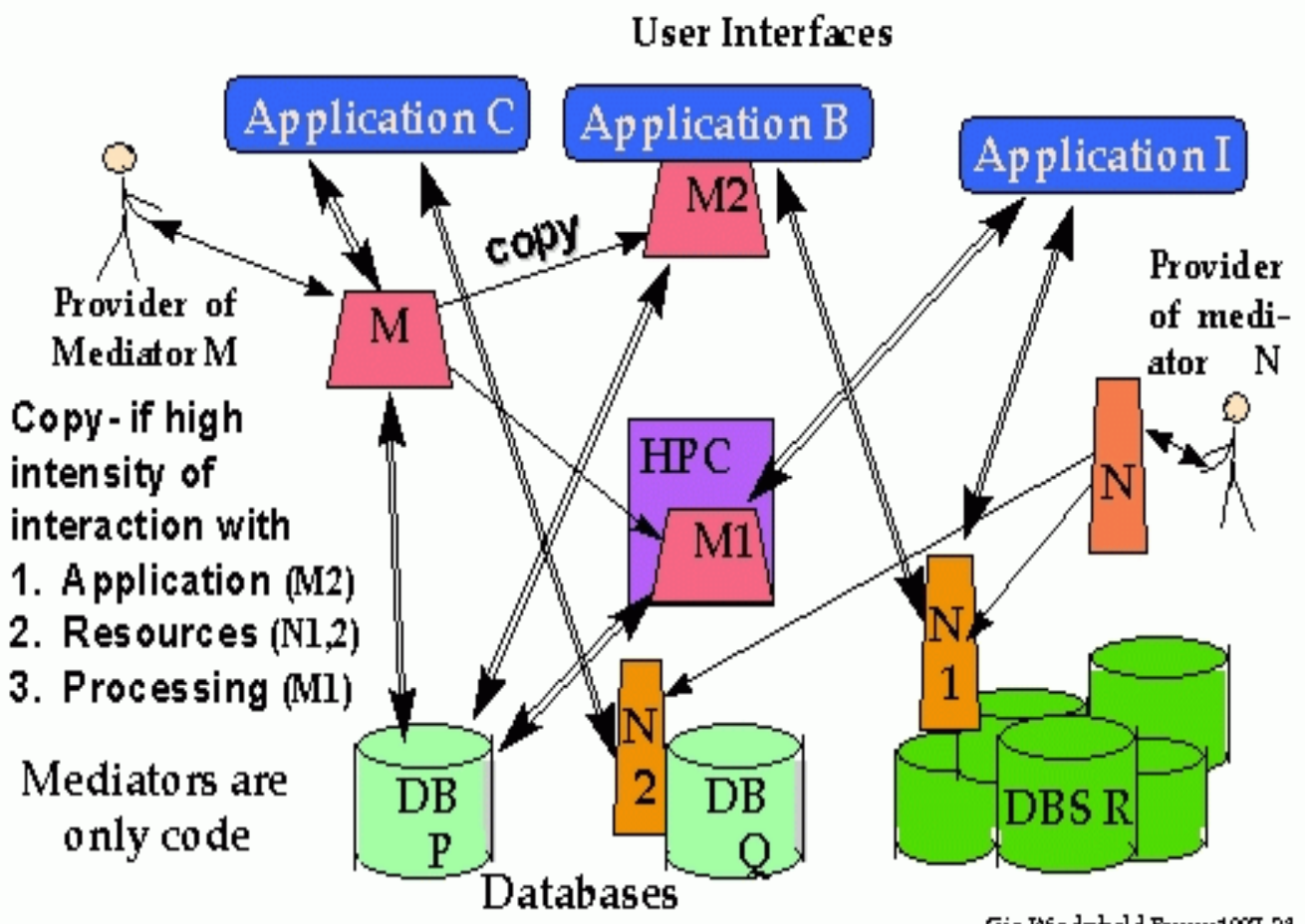
Gio Wiederhold Forum1997 4

Mediator Specification Interpreter Architecture



Gio Wiederhold Forum1997 14

Allocation Flexibility



Summary

- Mediation bridges Applications and Sources
- Mediator technology transforms data to information by applying an expert's knowledge
- Information is abstracted from data view integration
- View objects can be populated from relational DBs
- Mediation permits incremental system growth ($n \log n$)
- Mediatiors contain many domain-specific objects
- A consistent domain has a common ontology
- Composing domain ontologies requires an algebra
- Mediators provide a service-model on the networks

Mediation for Security

Public Health, . . ., Application

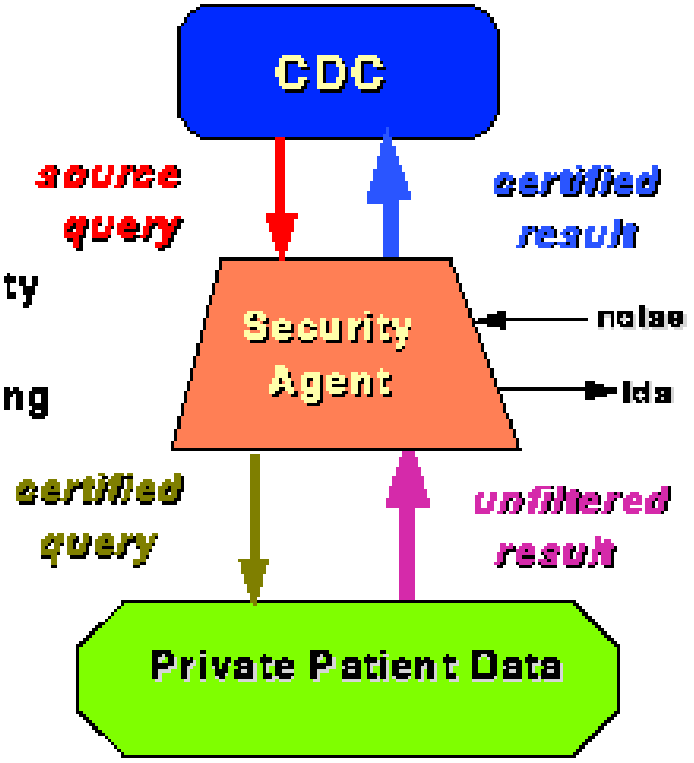
- Valid statistical data
- No access to private data

Security Mediator

- Owned by independent security officer
- does aggregation and screening
- Manual . . . to . . . automation

Physicians' Databases

- Valuable resources
- Need to be aggregated for significance



22

Два подхода к представлению в посреднике множества источников данных

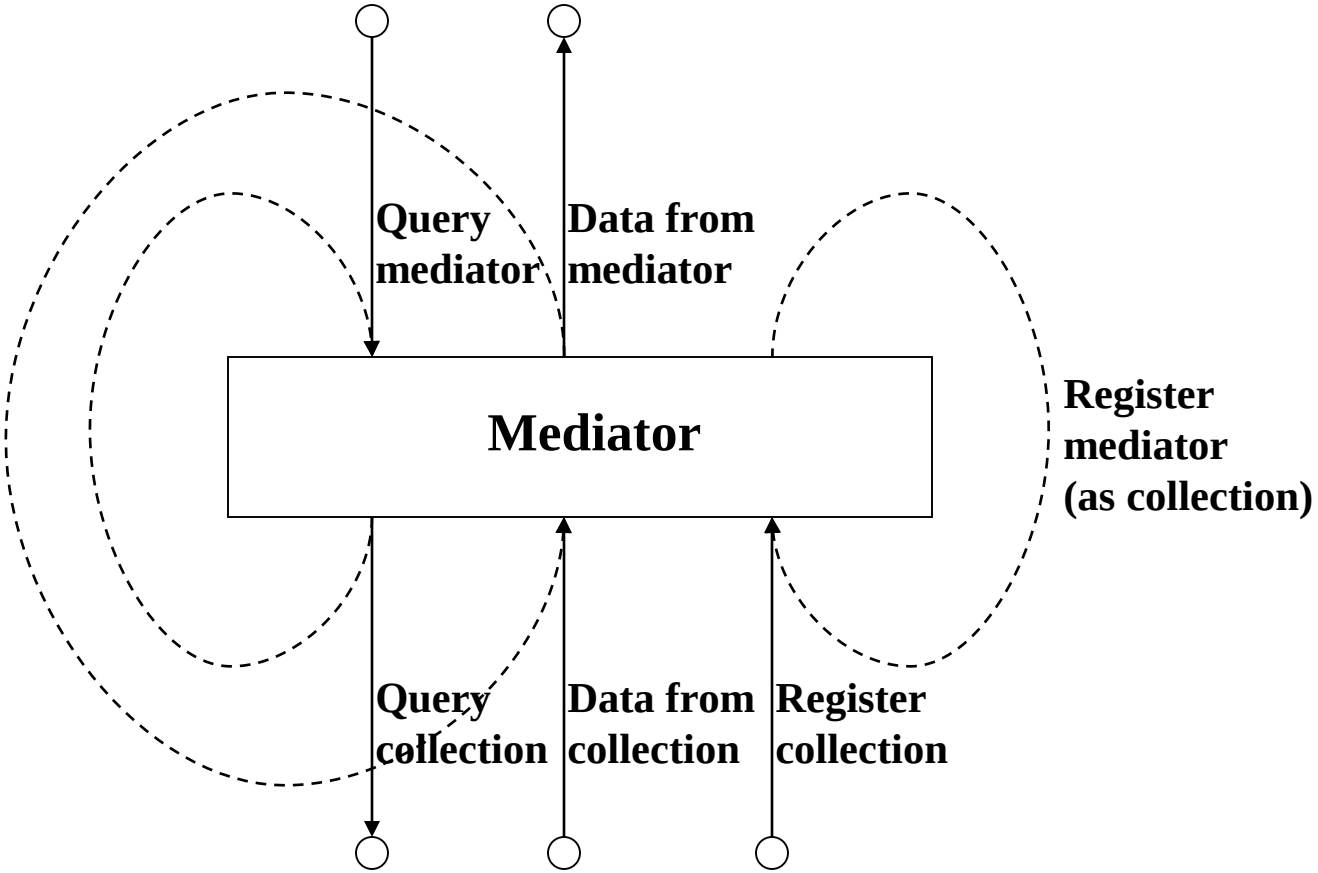
Посредники образуют слой между потребителями и источниками данных. Каждый посредник внешне выглядит как источник данных и может являться источником для посредников более высокого уровня.

- Подход, движимый имеющимися и разрабатываемыми источниками данных
  - Создается интегрированное представление множества источников независимо от задач приложений
  - Трудно обеспечить масштабируемость по числу источников
  - Интегрированную схему совокупности источников приходится изменять при включении в рассмотрение каждого нового источника
  - Глобальная схема может стать необозримой для пользователей приложений
- Подход, движимый приложениями, задачами системы
  - Создается описание предметной области класса задач (независимо от существующих источников), в которое отображаются релевантные задачам источники

Advantages of subject domain mediation

1. **Semantic integration** of heterogeneous information collections is reached
2. Users should know only **subject definitions** as defined by a community
3. Information providers can **disseminate** their information for integration **independently** of each other and **at any time**.
4. Autonomous information collections are absolutely **independent** on the mediator and its consolidated metainformation definitions
5. Users have **integrated access to all information registered** up to the moment of a query.
6. Mediators form **recursive structure**. Multiple subjects can be semantically integrated defining mediators of the higher level.

Mediator's Recursion



Образование схемы посредника, движимого приложениями

- Схема посредника определяется **независимо от интегрируемых в посреднике источников** в результате **консолидации требований сообщества** пользователей конкретных приложений посредника
- Пользователи (задачи) должны знать только **определения предметной области** в посреднике (определения понятий, структуры и поведения сущностей предметной области)
- **Запросы** при решении задач формулируются **независимо от фактического набора источников данных**, интегрированных в посреднике
- **Публикация** информации о вновь разработанных источниках данных может осуществляться **в любое время** и независимо от действующих к этому времени предметных посредников (в результате публикации могут быть инициированы действия по интеграции новых источников в посредниках, которым они релевантны). Появление новых источников не оказывает влияния на схему посредника.

Интеграция схем источников в посредниках (1)

- Согласование прикладного контекста источника с прикладным контекстом посредника
  - Установление соответствия сущностей и свойств источника сущностям и свойствам посредника
    - с помощью онтологических спецификаций контекстов, либо
    - полагаться на интуицию экспертов при решении вручную
  - *Онтологическая спецификация* - множество определений понятий конкретной предметной области
    - играют роль 'клея' фрагментов схем исочников и схем посредников для их семантического соответствия и композиции
    - онтологические определения аннотируют элементы схем посредника и источников
    - возможно использование вербальной онтологии (задаваемую подобно определениям терминов в толковом словаре). Вербальное представление онтологии может быть дополнено более формальными спецификациями в онтологических моделях (например, дескриптивных логиках)

Correlation of Ontological Concepts

- evaluation of descriptor weights
  - W_{Xk} – normalized weight of term k in concept X
  - N – number of concepts
  - n_k - number of concepts containing term k
  - f_{Xk} – frequency of term k in X
  - V_x – vector of all terms of X

$$W_{Xk} = \frac{f_{Xk} \log \frac{N}{n_k}}{\sqrt{\sum_{i \in V_X} \left(f_{Xi} \log \frac{N}{n_i} \right)^2}}$$

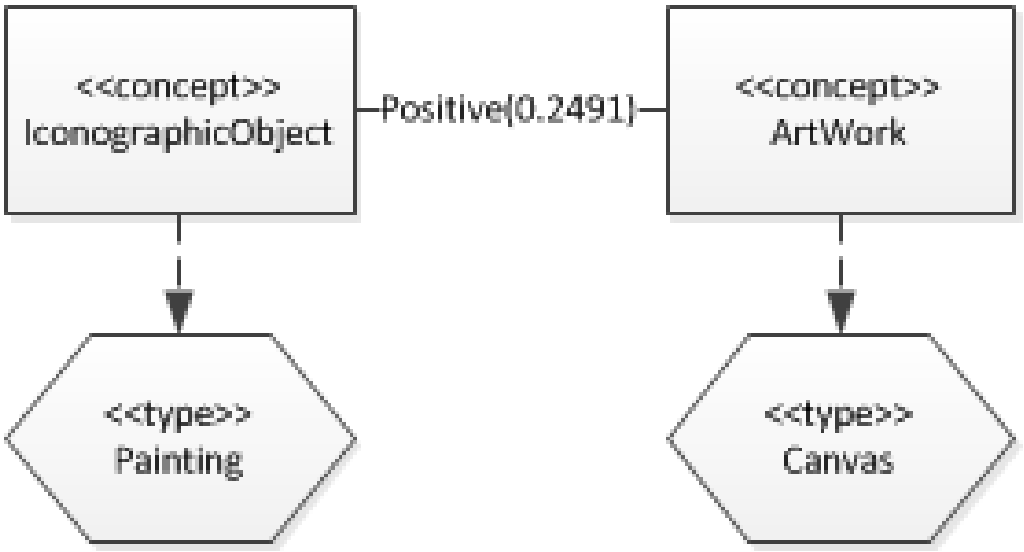
  - ❑ significance (weight) is higher for terms with high frequence in a concept
  - ❑ significance is higher for terms that occur in small number of concepts

- establishing intercontext relationships between concepts

$$sim(X,Y) = \frac{\sum_{k \in V_X \cup V_Y} (W_{Xk} \cdot W_{Yk})}{\sqrt{\sum_{k \in V_X} (W_{Xk})^2 \cdot \sum_{k \in V_Y} (W_{Yk})^2}}$$

Example of Ontological Correlation

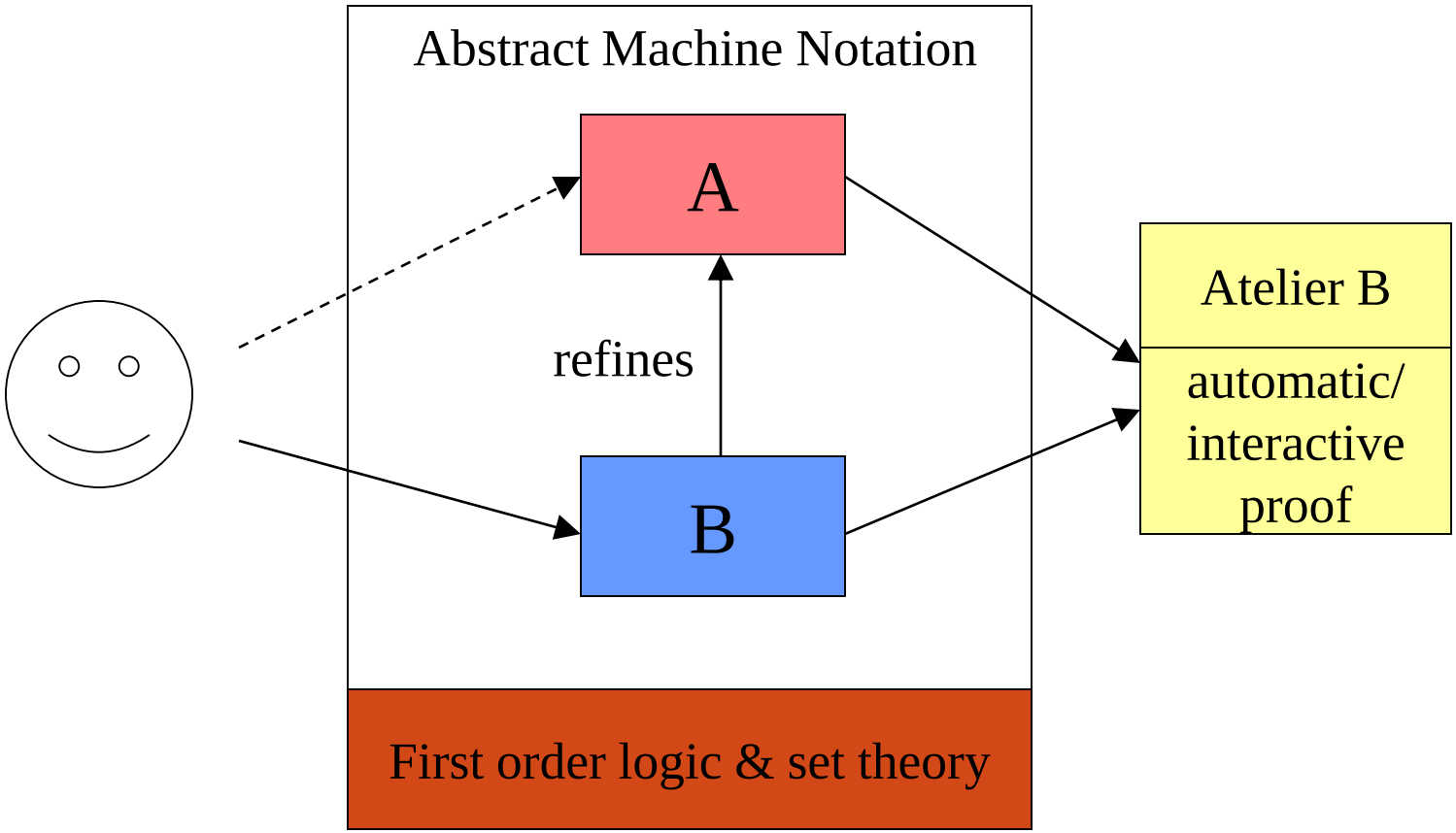
- { IconographicObject; in: concept;
  - def: “This entity comprises objects which are designed primarily or in addition to another functionality to represent or **depict** something in an optical manner, be it concrete or abstract. This entity has a certain pragmatic value in the fine **arts** since it conveniently groups together objects such as **paintings**, **drawings**, watercolors and other similar objects.” }
- { ArtWork; in: concept;
  def: “**Art** such as sculpture, **drawings** or **paintings** **depicting** something.” }
- Sim(IconographicObject, ArtWork) = 0.2491



Интеграция схем источников в посредниках вида ПП (2)

- Регистрация схемы исочника в схеме посредника
  - Регистрация - процесс целенаправленной трансформации спецификаций, включающий
    - декомпозицию схемы посредника на непротиворечивые фрагменты
    - поиск среди схем интегрируемых источников подходящих типов сущностей - кандидатов для уточнения ими типов сущностей посредника
    - выявление конфликтов представления значений, структур и поведения одного и то же свойства сущности в источнике и в посреднике и формирование правил устранения конфликтов для различных свойств
    - построение правил, определяющих классы сущностей источников в виде композиции классов сущностей посредника
  - Для манипулирования спецификациями используется исчисление спецификаций
    - операция выявления наибольшего общего фрагмента спецификаций типов сущностей в схеме источника и схеме посредника
    - фрагмент схемы источника должен *уточнять* фрагмент схемы посредника
      - Неформально: спецификация *B уточняет* спецификацию *A,* если пользователь может использовать *B* вместо A, не замечая разницы
      - Формально: отношение уточнения в языке Abstract Machine Notation (AMN) в виде набора теорем, доказательство при помощи Atelier B

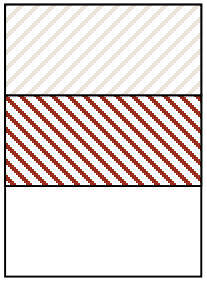
Refinement Formalization



The Idea of Registration Process

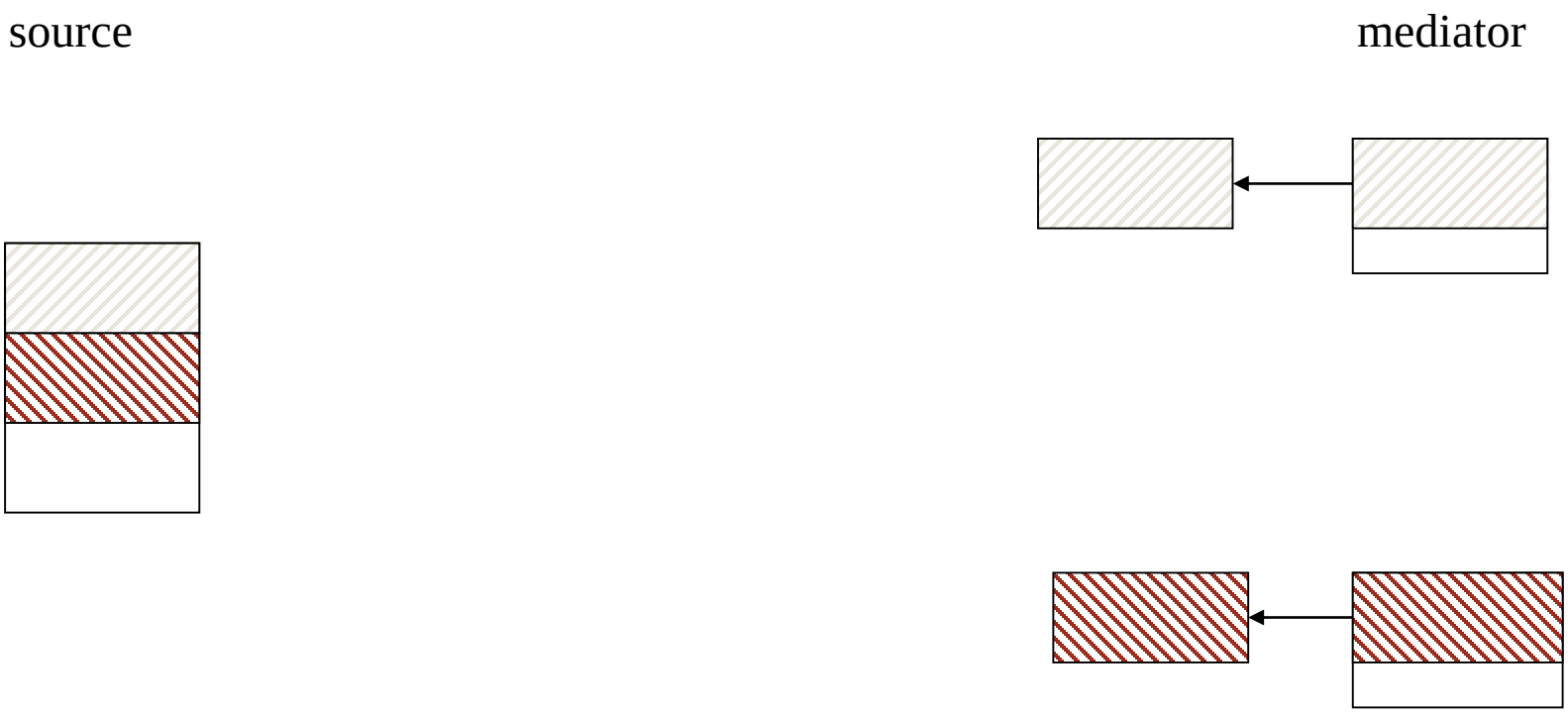
The Idea of Registration Process

source

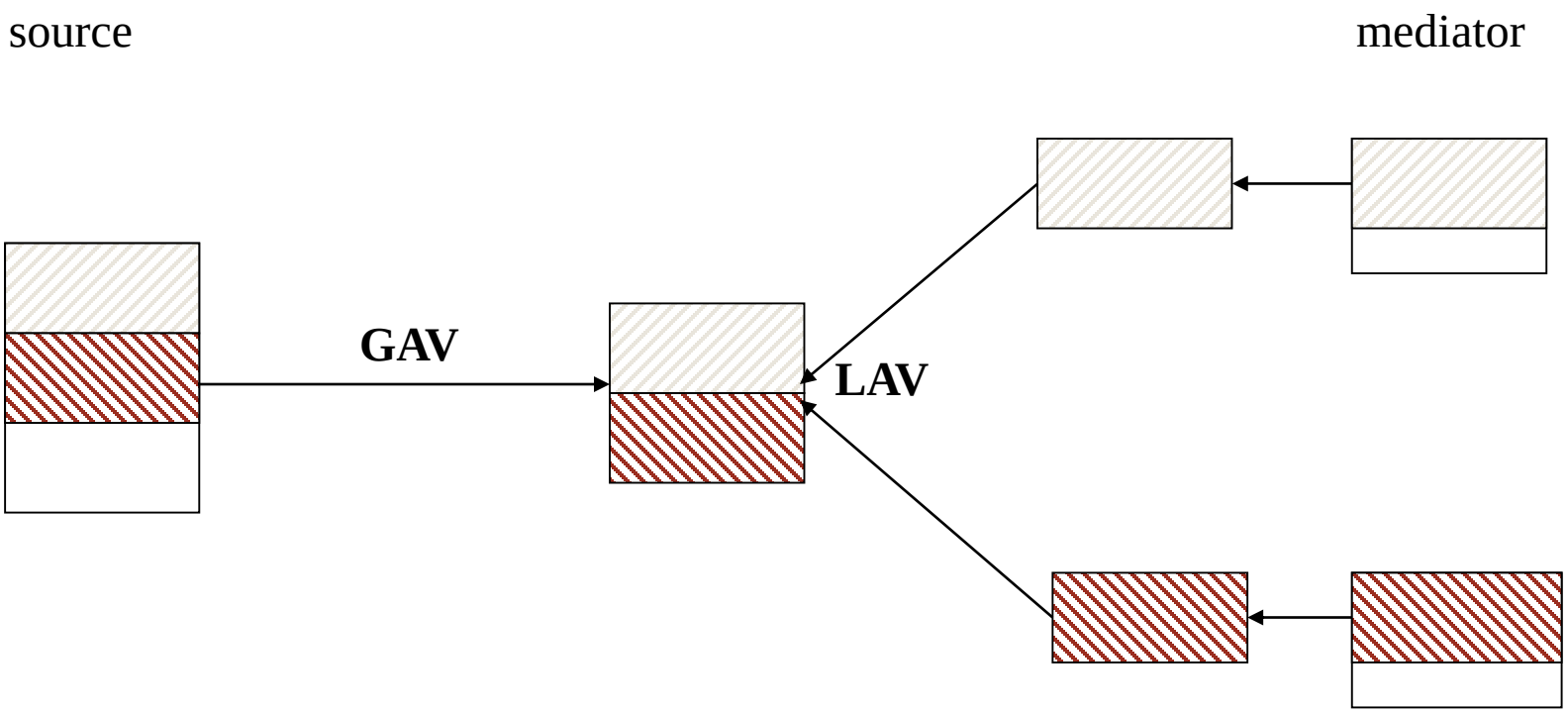


mediator

The Idea of Registration Process



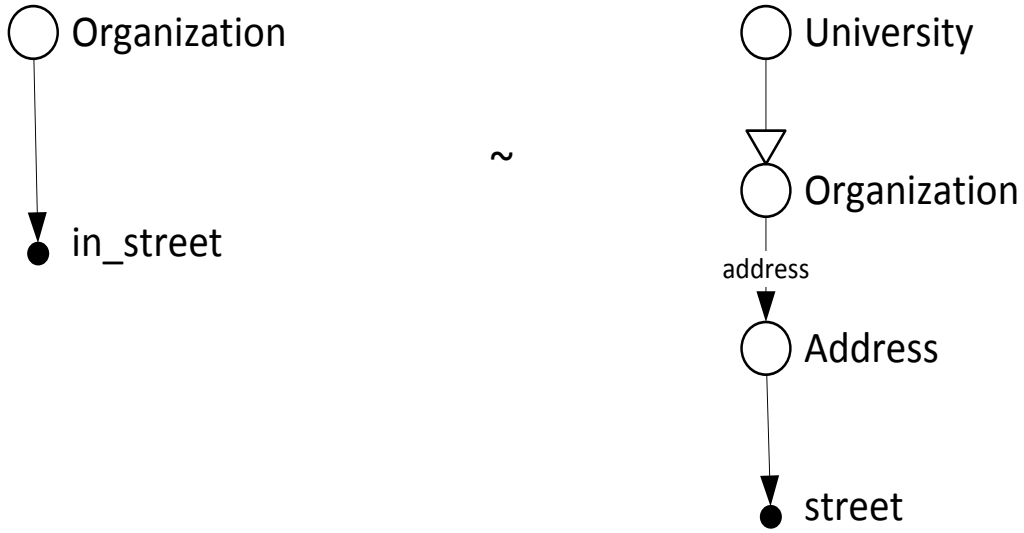
The Idea of GLAV Registration Process



Виды конфликтов

- Структурные конфликты
  - конфликт имен
    - типы Proposal и Submission имеет одинаковый смысл
  - использование различных типов атрибутов
    - атрибут budget может быть как типа integer, так и типа real
  - использование разных уровней абстракции представления информации
    - адрес в типе Organization задается посредством атрибутов city и street
    - используется дополнительный тип Address, содержащий атрибуты city и street
- Конфликты значений
  - использование разных отношений на экстенсионалах классов (эквивалентность, включение, пересечение, непересекающиеся)
    - класс employee включается в класс person
  - представление атрибутов в различных единицах измерения
    - атрибут budget может задаваться в рублях и в долларах
- Конфликты поведения
  - использование различных утверждений и ограничений целостности (например, ключи)
  - использование различных реализаций функций

Пример разрешения структурного конфликта




```
get_in_street: {in: function;
  params: {+y/University, -returns/string};
  {{ returns = y.address.street }}
}
```

Конструирование GLAV взглядов

- The process of the registration of heterogeneous data sources registration in the subject mediator is based on GLAV approach
- Basic technique is LAV
- The GAV approach additionaly allows
  - to specify functions resolving various kinds of conflicts between mediator and local source specifications
  - to define rules for transformation of query results form local source to mediator.

- General form of GLAV view

$$S_1(w_1/T_{w1}) , \dots , S_n(w_n/T_{wn}), F_{s1}(X_1), \dots , F_{sl}(X_l), B_s \subseteq$$
$$V(v/T) \subseteq$$
$$M_1(v_1/T_{v1}) , \dots , M_p(v_p/T_{vp}), F_{m1}(Y_1), \dots , F_{mq}(Y_m), B_m$$

  - S_i are local (resource) classes, F_{si} are local functions
  - M_j are mediator classes, F_{mj} are mediator functions
  - B_s and B_m are conditions

Partial Source View Construction (Example)

- For each local class (*canvas*) and for each its relevant federated class (*painting* and *creator*) we construct a partial view (*v_canvas_painting*, *v_canvas_creator*) expressing local class in terms of federated class
- *R_Painting_Canvas = MCR(Painting, Canvas)*, *CR_Painting_Canvas* is the corresponding reduct of *Canvas*, refining *R_Painting_Canvas*

- LAV rule:

  $v\_canvas\_painting(p/R\_Painting\_Canvas) \subseteq$
  $painting(p/R\_Painting\_Canvas)$ &
  $p.in\_collection.in\_repository.name = 'Uffizi'$
- GAV rule:

  $v\_canvas\_painting(p/R\_Painting\_Canvas) \supseteq canvas(p/CR\_Painting\_Canvas)$

- LAV rule:

  $v\_canvas\_creator(c/R\_Creator\_Canvas) \subseteq$
  $creator(c/R\_Creator\_Canvas)$ &
  $in(w, c.works)$ & $w.in\_collection.in\_repository.name = 'Uffizi'$
- GAV rule:

  $v\_canvas\_creator(p/R\_Creator\_Canvas) \supseteq canvas(p/CR\_Creator\_Canvas)$

Conflict Reconciliation in Views

v_university_organization(p/R_Organization_University) ⊆
organization(p/R_Organization_University).
v_university_organization(p/R_Organization_University[in_street]) ⊇
university(p/CR_Organization_University) &
in_street = get_in_street(p).

```
{CR_Organization_University;  in: c_reduct;
  metaslot
      of: University;
      taking: {title, address};
      reduct: R_Organization_University;
     simulating: {
         R_Organization_University.title ~ CR_Organization_University.title;
         R_Organization_University.in_street  ~ CR_Organization_University.get_in_street;
     };
  end;
get_in_street: {in: function;
      params: {+ext/University, -returns/string};
     {{ returns = ext.address.street }}
}
```

Source View Composition (Example)

- For each local class (*canvas*) construct a view (*v_canvas*) as a composition of its partial views (*v_canvas_painting*, *v_canvas_creator*).

- LAV rule:

v_canvas(p/CT_ Painting_Creator) $\subseteq$

painting(p/R_Painting_Canvas) &

creator(p/R_Creator_Canvas) &

p.in_collection.in_repository.name = 'Uffizi' &

in(w, p.works) & w.in_collection.in_repository.name = 'Uffizi'

- GAV rule:

v_canvas(p/CT_Painting_Creator) $\supseteq$ *canvas(p/CT_Canvas)*

- Types:

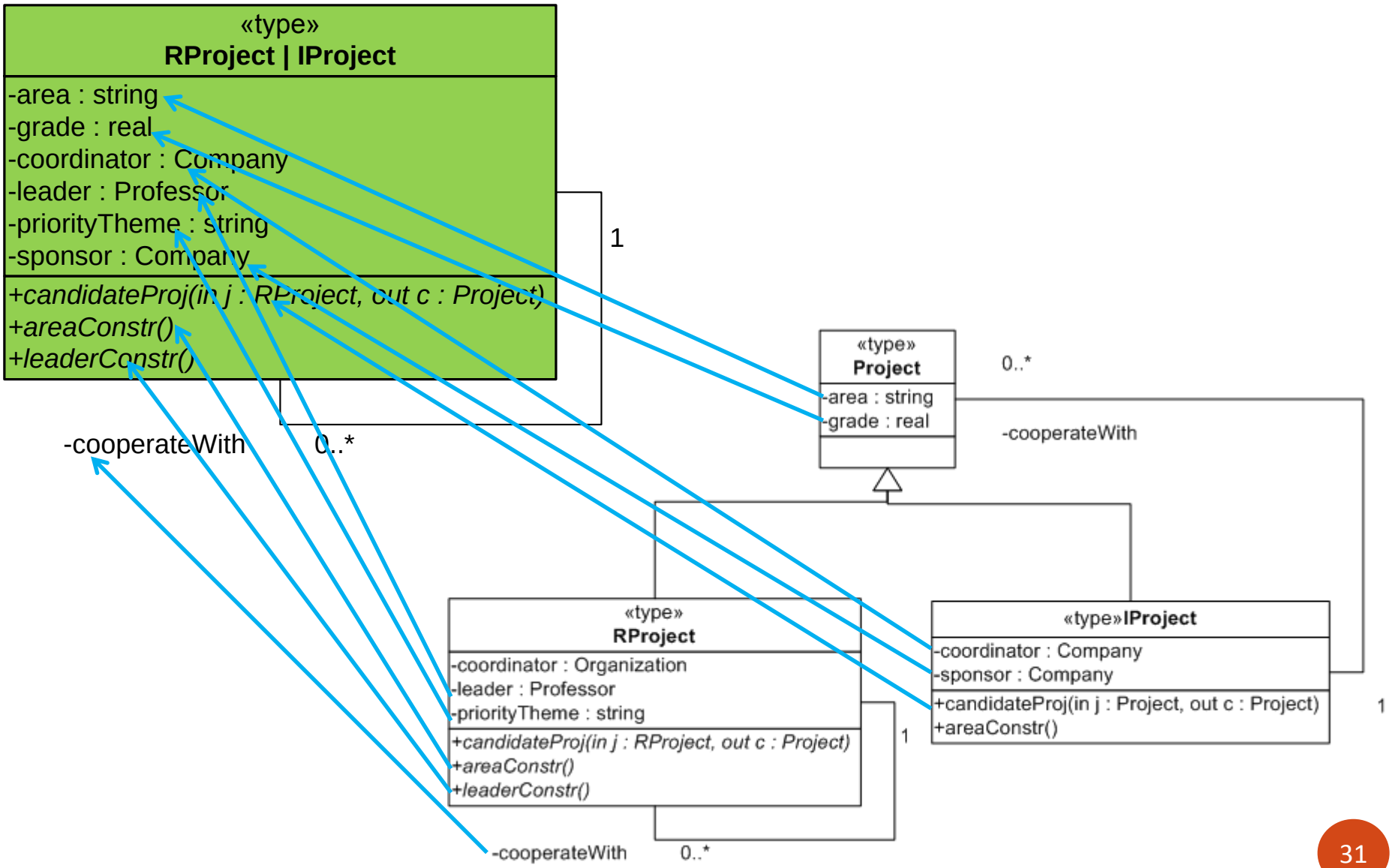
CT_Canvas = CR_Painting_Canvas | CR_Creator_Canvas

CT_Painting_Creator = R_Painting_Canvas | R_Creator_Canvas

Операция соединения типов (join)

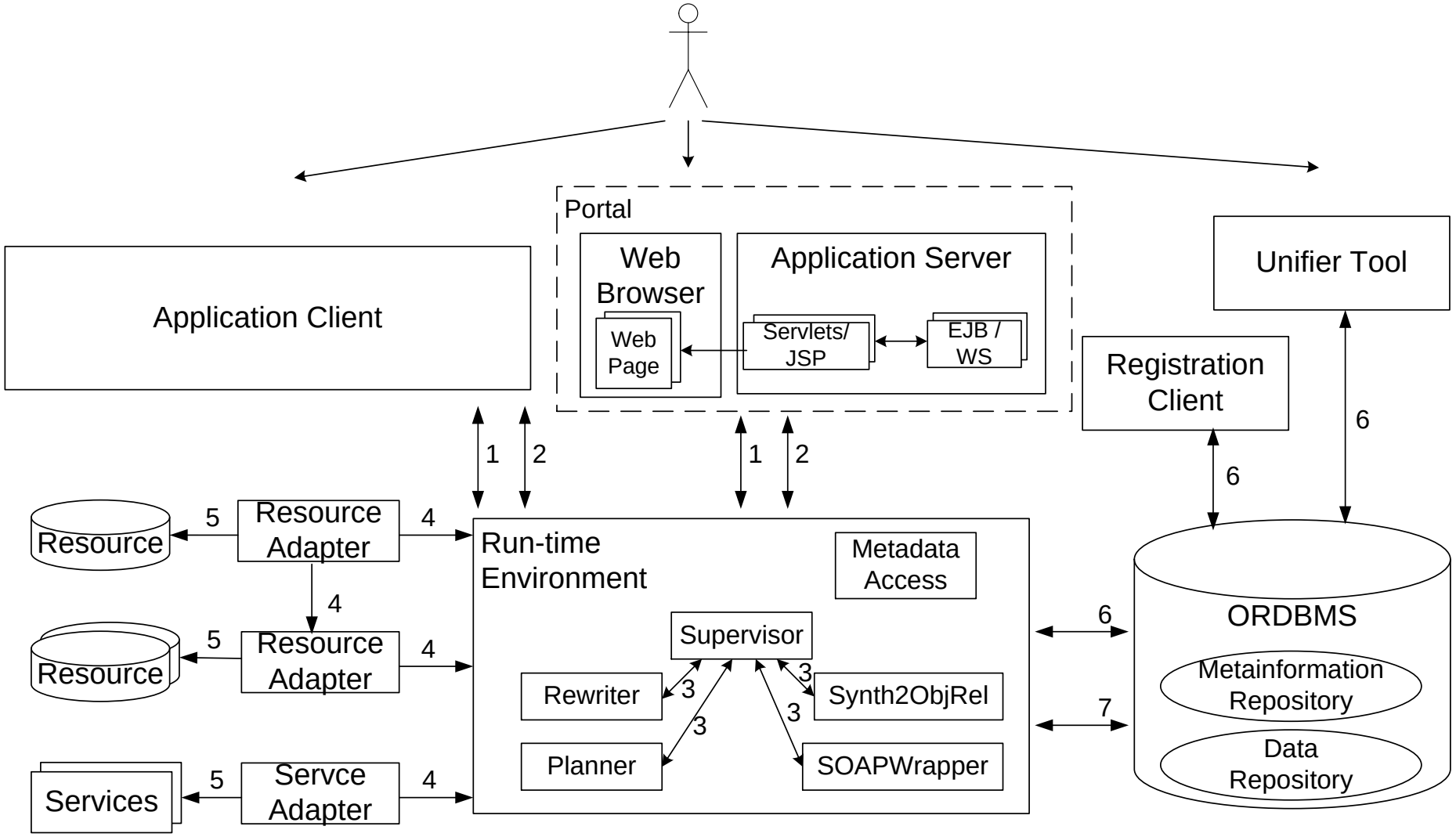
- An operation T1 | T2 produces a type T as is a **join** of specifications of the operand types
  - Generally T includes a merge of specifications of T1 and T2
  - Common elements of specifications of T1 and T2 are included into the resulting type only once

Type Join Example



Архитектура исполнительных механизмов слоя предметных посредников

Архитектура (1)



Архитектура (2)

- Клиент приложения
- Портал
- Посредник
- Супервизор
- Средства переписывания запросов (Rewriter)
- Планировщик (Planner)
- SynthesisToObjectRelational
- Адаптеры
- База данных и метаинформации

Клиент приложения

- просмотр метаданных
  - в графическом виде (в виде UML диаграмм)
  - в текстовом виде (спецификация в канонической модели посредников – язык СИНТЕЗ)
- задание запросов к данным зарегистрированных источников
- представление результатов запросов
- поддержка различных языков запросов: Syfs, ADQL

- портал: все тоже самое, только посредством Веб

Посредник

- представляет собой веб-сервис
- обеспечивает:
  - формирование и ведение интегрированной базы метаинформации для данной предметной области
  - регистрацию разнообразных источников данных
  - унифицированное представление информации посредством канонической модели посредников
  - задание и выполнение запросов над интегрированной схемой посредника (компонент Supervisor)

Supervisor

- является связующим компонентом исполнительной системы посредников и фактически определяет ее интерфейс
- обрабатывает запросы пользователя и для их выполнения взаимодействует с другими компонентами (Rewriter, Planer, SynthesisToOracle, …)
  - openSession
  - executeQuery
  - getResult
  - getStatus
  - cancelQuery
  - closeSession

Средства переписывания запросов (Rewriter)

- Компонент Rewriter реализует функцию переписывания запроса пользователя на языке Syfs над федеративной схемой в запрос над объединенной схемой источников, который выражается на языке Asyfs и передается компоненту Planner для дальнейшей обработки

Планировщик (Planner)

- Компонент Planner реализует функцию планирования запроса над источниками
- В результате планирования строится распределенный запрос, который составляют удаленные запросы к отдельным источникам
- Последовательность выполнения удаленных запросов определяет структура, которая называется «деревом выполнения»

SynthesisToObjectRelational

- Реализует runtime-поддержку канонической модели данных посредника над объектно-реляционной базой данных
- Позволяет загружать и выгружать данные, которые в канонической модели представляются как набор классов, что транслируется в набор таблиц базы данных
- Данные загружаются и выгружаются в формате VOClass
- Реализует поддержку выполнения запросов на языке Asyfs, транслируя их в язык SQL

Адаптеры

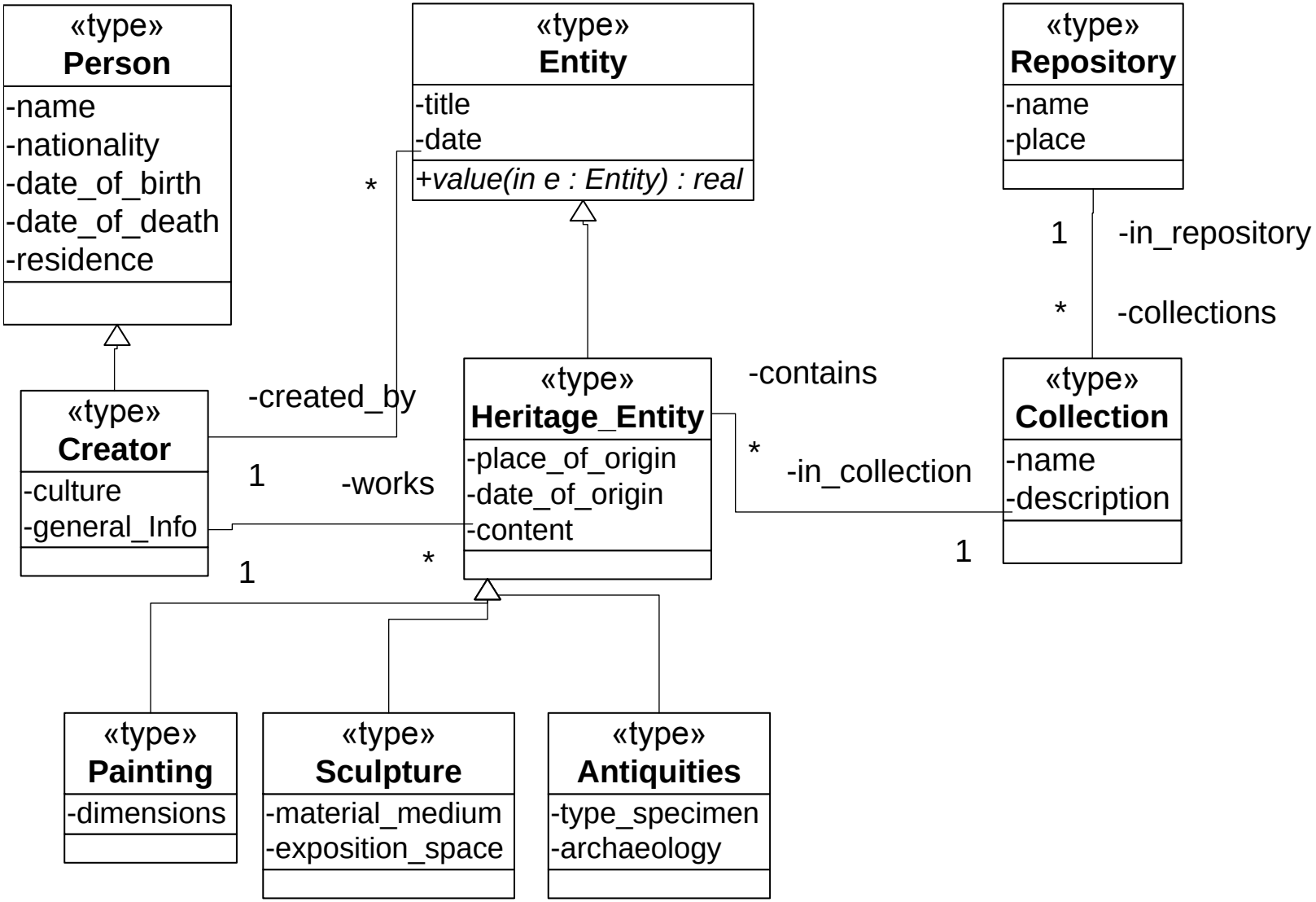
- Предоставляет доступ к данным источника в терминах канонической модели и языка запросов Asyfs
- Транслирует запрос на языке Asyfs в язык запросов источника (SQL, Xquery, command line, etc.)
- Преобразует результат полученный от источника, в формат канонической модели данных (VOClass)
- Дополнительно, если это позволяет источник, реализует возможность загрузки в источник временных классов, которые потом использует выполняемый адаптером запрос

База данных и метаинформации

- База используется для хранения метаинформации посредника
  - спецификации посредника
  - спецификации источников данных
  - онтологии
  - взгляды
- и для хранения промежуточных результатов запросов
- База реализована на объектно-реляционной СУБД

Пример посредника

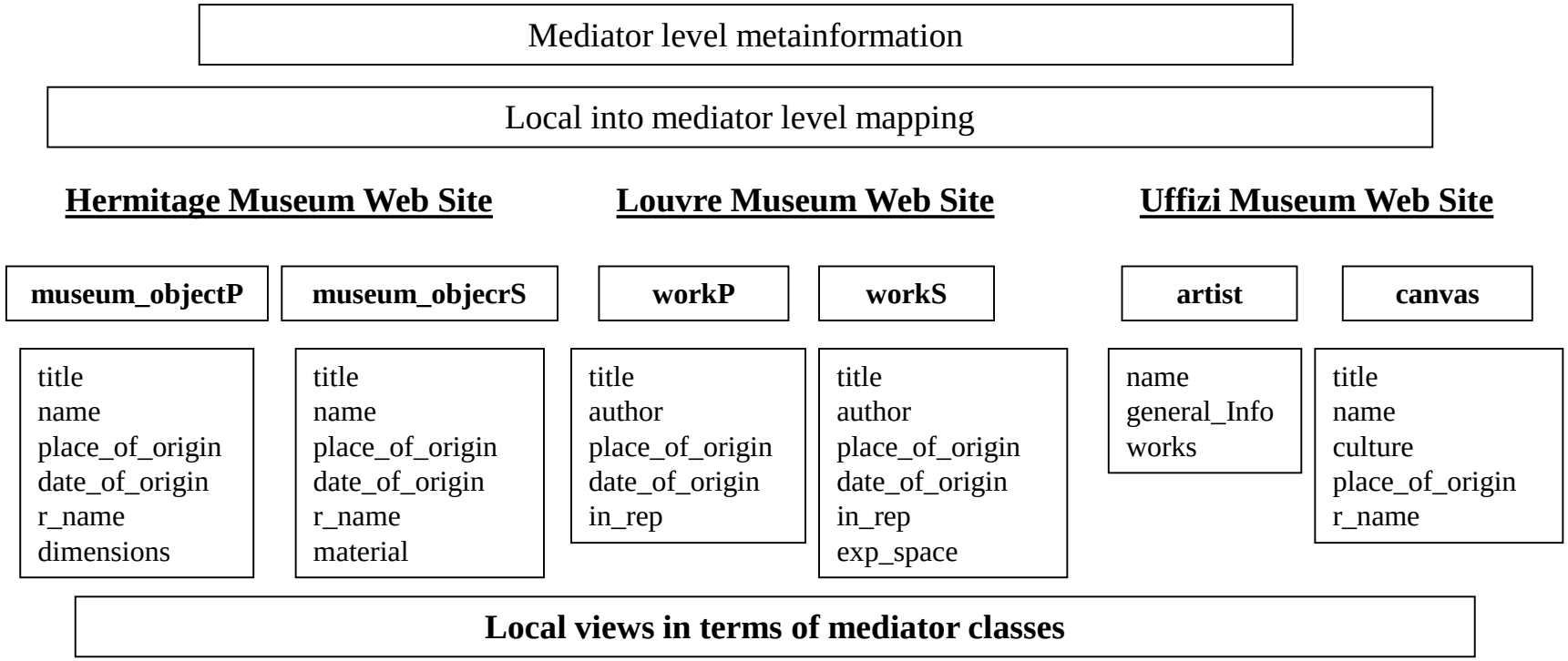
Пример схемы посредника



Пример запроса к схеме посредника

```
valuable_Italian_heritage_entities(x/[title, c_name_first, c_name_last, r_name, v]) :-
heritage_entity(h/[title,
c_name_first: created_by.name.first_name,
c_name_last: created_by.name.last_name,
place_of_origin, date_of_origin,
r_name: in_collection.in_repository.name])
& value(h) = v & value(h) >= 200000
& date_of_origin >= 1500 & date_of_origin < 1750
& place_of_origin = 'Italy'.
```

Sources Representation at Subject Mediator



Local views in terms of mediator classes

museum_objectP(p/Museum_Object[title, name, place_of_origin, date_of_origin, r_name]) ⊆ painting(p/Painting[title, name:created_by.name, place_of_origin, date_of_origin, r_name: in_collection.in_repository.name]), date_of_origin < 1700, date_of_origin > 1600

…

workP(p/Work[title, author, place_of_origin, date_of_origin, in_rep]) ⊆ painting(p/Painting[title, author: created_by.name, place_of_origin, date_of_origin, in_rep: in_collection.in_repository.name]), in_rep = 'Louvre'

…

amount(h/Entity[title, name: created_by.name], v/real) ⊆ value(h/Entity[title, name: created_by.name], v/real)

canvas(p/Canvas[title, name, culture, place_of_origin, r_name]) ⊆ painting (p/Painting[title, name: created_by.name, place_of_origin, date_of_origin, r_name: in_collection.in_repository.name]), creator (c/Creator[name, culture]), r_name = 'Uffizi', date_of_origin >= 1550, date_of_origin < 1700

artist (a/Artist[name, general_Info, works]) ⊆ creator(a/Creator[name, general_Info, works/{set-of:Painting}])

Semantic mappings (views) for Uffizi (I)
(slightly changed)

- **LAV**

Views.v_Uffizi_Painting(x/[title, first_name, last_name, culture,
    place_of_origin, date_of_origin, r_name]):-
painting(y1/[title, place_of_origin, date_of_origin,
    first_name: created_by.name.first_name,
    last_name: created_by.name.last_name,
    r_name: in_collection.in_repository.name]) &
creator(y2/[culture, first_name: name.first_name,
    last_name: name.last_name]) &
    r_name = 'Uffizi' &
    date_of_origin >= 1550 & date_of_origin < 1700.

Semantic mappings (views) for Uffizi (II)
(slightly changed)

- **GAV**

  v_Uffizi_Painting(x/[title, first_name, last_name, culture,
  place_of_origin, date_of_origin, r_name]) :-
  Uffizi.canvas(x/[title, painter, culture, place_of_origin: place,
  date_of_origin: date]) &
  I_FUNC.nameUffizi(r_name) &
  I_FUNC.getFirstName(painter, first_name) &
  I_FUNC.getLastName(painter, last_name).

Asyfs Query Rewritten

```
MOVE{session1.valuable_Italian_heritage_entities}
UNION plan to Hermitage
UNION plan to Louvre
UNION
 PROJECT{[title, c_name_first, c_name_last, r_name, v]}
  SELECT{v >= 200000.0 & date_of_origin >= 1500.0 & date_of_origin < 1750.0 &
   place_of_origin = 'Italy'}
    APPEND{Uffizi.amount(title, v)}
     PROJECT{[title/string: title, c_name_first/string: first_name,
        c_name_last/string: last_name, place_of_origin/string:
        place_of_origin,          date_of_origin/real: date_of_origin,
        r_name/string: r_name]}
       PROJECT{[title, first_name, last_name, culture, place_of_origin,
              date_of_origin, r_name]}
        APPEND{I_FUNC.getLastName(painter, last_name)}
         APPEND{I_FUNC.getFirstName(painter, first_name)}
          APPEND{I_FUNC.nameUffizi(r_name)}
           PROJECT{[title, painter, culture, place_of_origin: place,
                    date_of_origin: date]}
                  Uffizi.canvas
```

Основные принципы планирования исполнения запросов

Планирование запросов как общая для всех СУБД задача

- Задачи планировщика:
  - определить алгоритм получения результата (задаваемого запросом)
  - алгоритм вычисления результата должен быть эффективным (оптимальным)

- Пространство возможных алгоритмов вычисления результата любого конкретного запроса определяется исключительно архитектурой системы
- Чем сложнее система, тем больше пространство в котором нужно найти эффективный алгоритм

Подходы к оценке эффективности алгоритмов вычисления запросов

- по ожидаемому времени исполнения
- по использованию различных ограниченных ресурсов
  - оперативной памяти
  - дискового пространства
  - процессорного времени
  - нагрузки на линии передачи данных (для распределенных систем)
  - времени подсистемы ввода-вывода
- по времени, затраченному на поиск конкретного алгоритма

- В реальной ситуации решение всегда будет компромиссом, т.к. оптимизировать все эти критерии одновременно – невозможно

Архитектуры интеграционных сред

- Системы с централизованным хранилищем данных (в хранилище содержатся и периодически обновляются точные копии интегрируемых источников)
  - легко планировать запросы
  - высокая эффективность выполнения запросов
  - сложно обслуживать
  - не всегда возможно сделать копию источника целиком (технические/юридические препятствия)
- Множество источников данных, объединенных общей средой выполнения запросов. Каждый источник может выполнять запросы лишь над своими данными и возвращать результат (отсутствует возможность передачи данных из одного источника в другой)
  - средняя сложность планирования запросов
  - низкая эффективность при больших объемах данных
- Множество информационных источников с возможностью передачи данных из одного источника в другой
  - высокая сложность планирования запросов.
  - средняя эффективность выполнения запросов.

Задачи планировщика в среде предметных посредников

- Входные параметры планировщика
  - Запрос на языке ASYFS в терминах *объединенной* схемы
  - *Объединенная схема* – объединение локальных схем источников, в котором каждый источник представлен отдельным модулем
    - отдельный модуль функций разрешения конфликтов
- Результат планирования
  - Множество подзапросов, каждый из которых либо
    - целиком состоит из операций и выполняется над данными конкретного источника, либо
    - содержит ссылки на результаты выполнения других подзапросов
      - результаты должны быть предварительно загружены в адаптер этого источника, после чего подзапрос может быть выполнен целиком на данном источнике

Базовые принципы планирования

- Упрощение исходного запроса - разбиение сложных операций на множества элементарных с сохранением семантики
  - выполнимость каждой операции хотя бы на одном источнике целиком
  - минимальность количества атрибутов, которые эта операция использует
- Перестановка операций union
- Построение зависимостей операций (для определения пространства семантически эквивалентных перестановок операций)
- Разбиение на подзапросы

Пример упрощения запроса

Исходный запрос

```
MOVE{result}
PROJECT{[name, age, addressId]}
SELECT{(y.getMaxAge() - x.getAge()) = 5.0 & y.getMaxAge() = x.getAge() &
   country = 'Russia' & city = 'Moscow' & age > 20.0 &
   RemoteModule0.fnNegate(y.isGood())}
JOIN
  PROJECT{[name, age, addressId, x]} RemoteModule0.Person(x)
  PROJECT{[addressId, country, city, y] RemoteModule1.Address(y)
```

Результат упрощения

```
MOVE{result}
  PROJECT{[name, age, addressId]}
    SELECT{@"@simple?#3" = 5.0}
      SELECT{@"@simple?#7" = true}
        APPEND{id(@"@simple?#8", @"@simple?#7")}
        APPEND{RemoteModule0.fnNegate(@"@simple?#6", @"@simple?#8")}
        APPEND{isGood(y, @"@simple?#6")}
          SELECT{age > 20.0}
            SELECT{city = 'Moscow'}
              SELECT{country = 'Russia'}
                SELECT{@"@simple?#4" = @"@simple?#5"}
                  APPEND{getAge(x, @"@simple?#5")}
                  APPEND{getMaxAge(y, @"@simple?#4")}
                  APPEND{id((@"@simple?#1" - @"@simple?#2"), @"@simple?#3")}
                  APPEND{getAge(x, @"@simple?#2")}
                  APPEND{getMaxAge(y, @"@simple?#1")}
                  JOIN
                   PROJECT{[name, age, addressId, x]} RemoteModule0.Person(x)
                   PROJECT{[addressId, country, city, y]} RemoteModule1.Address(y)
```

Разбиение на подзапросы

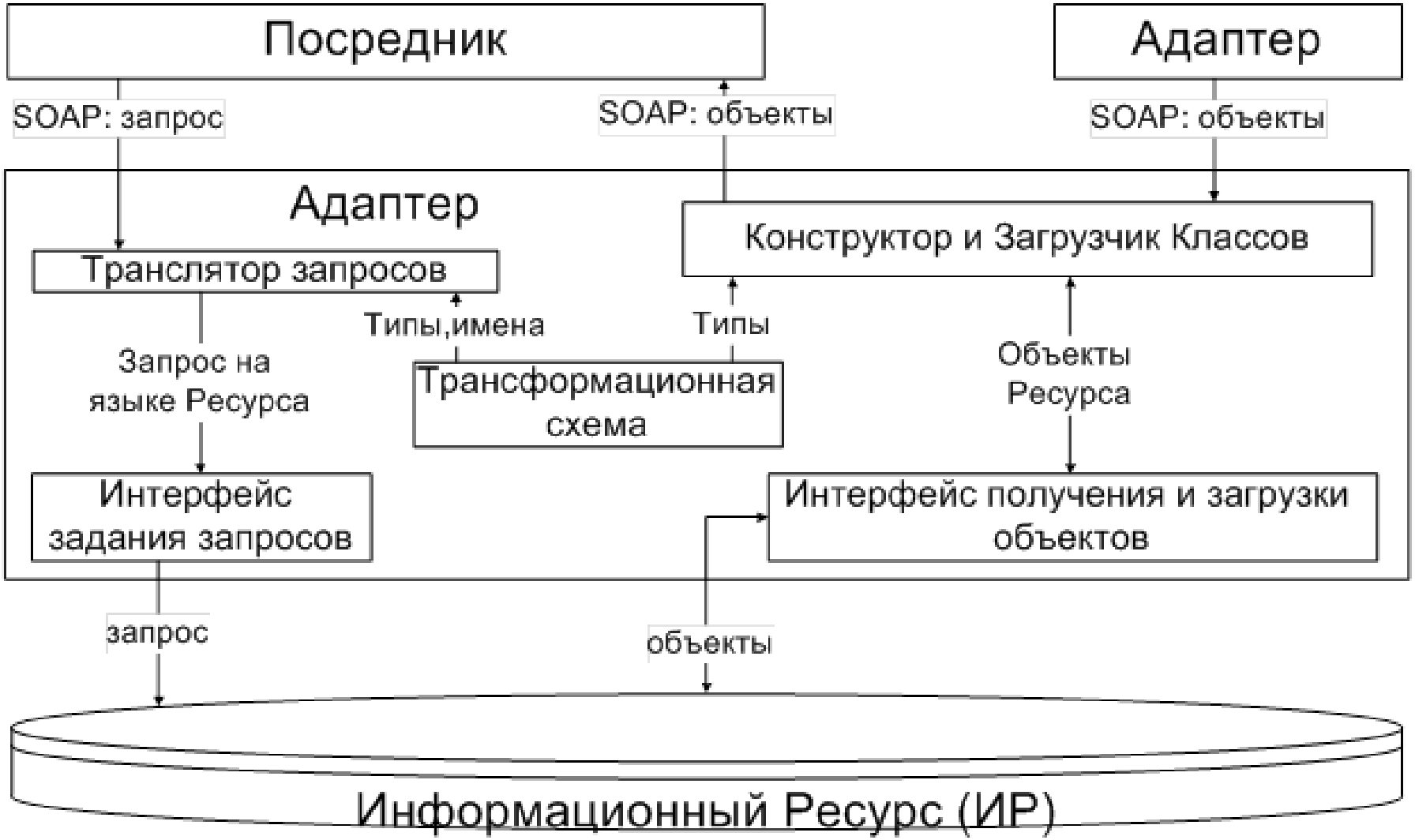
- Разбиение на подзапросы – формирование группы подмножеств удовлетворяющих требованиям:
  - ограничения на порядок выполнения операций
  - ограничений на выполнимость определенных операций на конкретных источниках
  - эффективность
- Каждое полученное таким образом подмножество состоит из операций выполнимых в определенном порядке на одном конкретном источнике над конкретными наборами данных (возможно, результатами выполнения других подмножеств)

Архитектура адаптеров

Функции адаптеров

- преобразование запросов, поступающих от посредника, в запросы к источникам данных
- инициирование выполнения запроса на источнике и получение результата от источника
- преобразование объектов результирующего множества, полученных от источника, в объекты канонической модели данных посредника
- прием данных от других адаптеров

Архитектура адаптеров



Компоненты адаптеров (1)

- Транслятор
  - получает запрос на языке запросов посредника и осуществляет преобразование в запрос на языке запросов источника
  - после преобразования запрос возвращается на языке модели данных источника и выполняется посредством интерфейсных компонентов
- Трансформационная схема
  - предназначена для преобразования имен и типов канонической модели посредника к именам и типам модели данных источника в запросе, получаемом от посредника и
  - обратного преобразования из значений результата в каноническую форму для дальнейшей передачи результата посреднику
- Интерфейс
  - взаимодействие с источником - передача запроса источнику, получение результата выполнения запроса, загрузка данных в источник

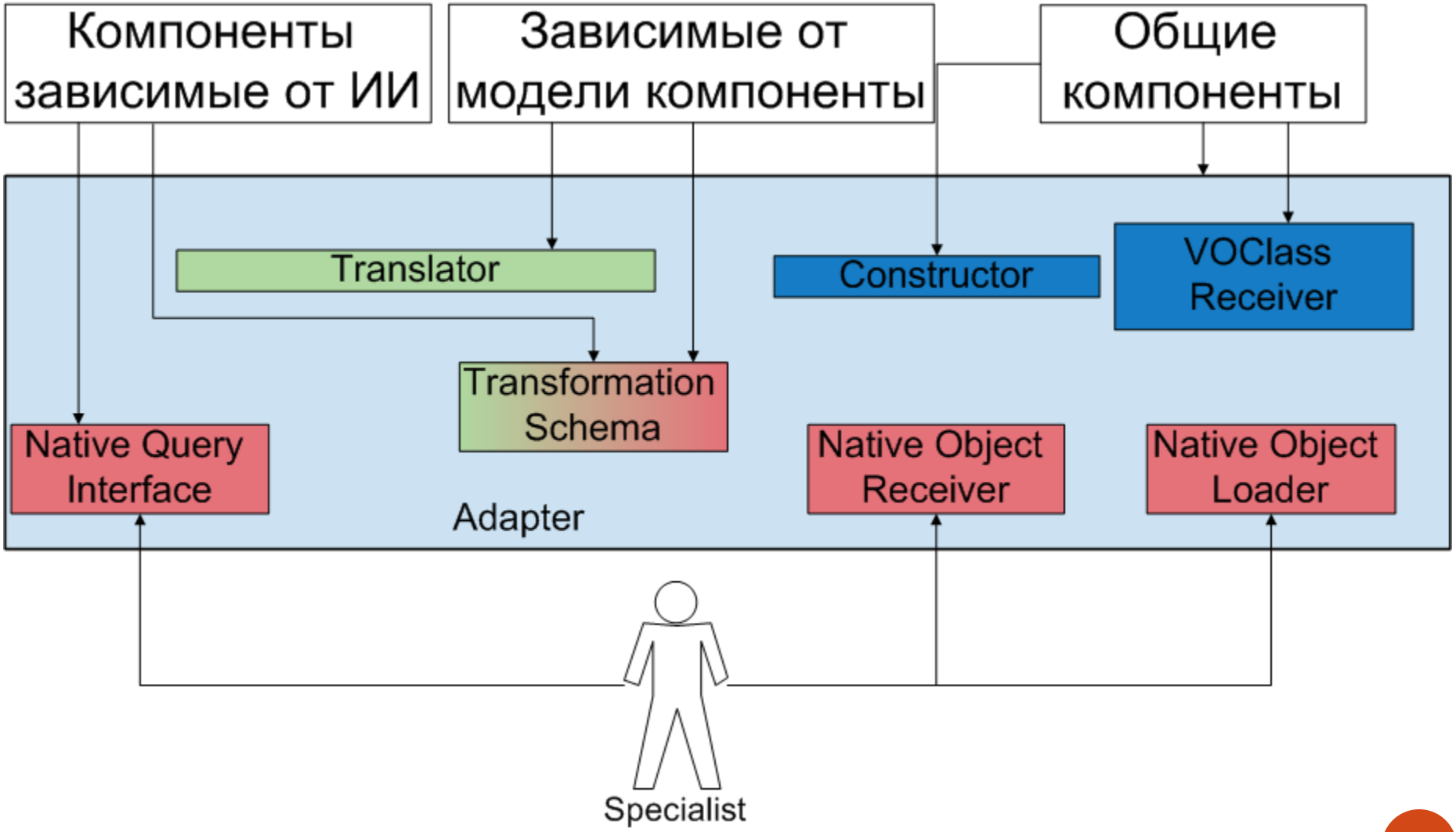
Компоненты адаптеров (2)

- Конструктор классов
  - преобразует данные, полученные от исочника, в формат обмена данными между компонентами посредников
- Загрузчик классов
  - преобразует данные из формата обмена данными между компонентами посредников в формат источника
- Адаптер
  - обеспечивает взаимодействие компонентов архитектуры, определяя логику функционирования адаптера и его интерфейс

Требования к адаптерам

- все методы адаптера должны допускать одновременное использование из нескольких потоков (thread safe)
- поддержка возможности выполнения запроса на источнике, и получение результата этого запроса
- поддержка изолированных друг от друга сессий с изолированными ресурсами
- поддержка возможности повторного использования результата запроса
- поддержка возможности управления загрузкой данных в адаптер супервизором, а также возможности материализации запросов в адаптерах
- поддержка Capabilities адаптеров
- поддержка оценочных запросов
- поддержка возможности задания запроса над ограниченной выборкой
- поддержка статистики для эффективного планирования

Компоненты адаптеров



Решение задач в среде посредников

Выбор вторичных стандартов при фотометрии оптического послесвечения гамма-всплесков

- Гамма-всплески (GRB)
  - Кратковременные вспышки (t ~ 0.2-100 с) гамма-излучения
  - Источники лежат на космологических расстояних (zmax = 8.26, GRB 090423)
  - Образуют послесвечения во всех диапазонах спектра
  - Связаны с особым типом сверхновых – гиперновыми (длинные всплески) и слиянием компактных релятивистских объектов (короткие всплески)
  - Очень «быстрые» события
- Необходима быстрая и точная фотометрия для построения общей кривой блеска
  - необходимы унифицированные вторичные стандарты
- Вторичный стандарт (звезда)
  - не переменная
  - достаточно изолированной от других объектов для корректной фотометрии
  - с известной (из каталога или другим образом) звёздной величиной в нескольких оптических фильтрах
  - со спектром, близким к спектру оптического послесвечения GRB

Общий процесс решения задачи

- Построение глоссария предметной области
- Построение онтологии предметной области
- Определение концептуальной схемы
- Создание посредников
- Регистрация источников
- Получение результата

Пример вербальных определений для создания онтологии

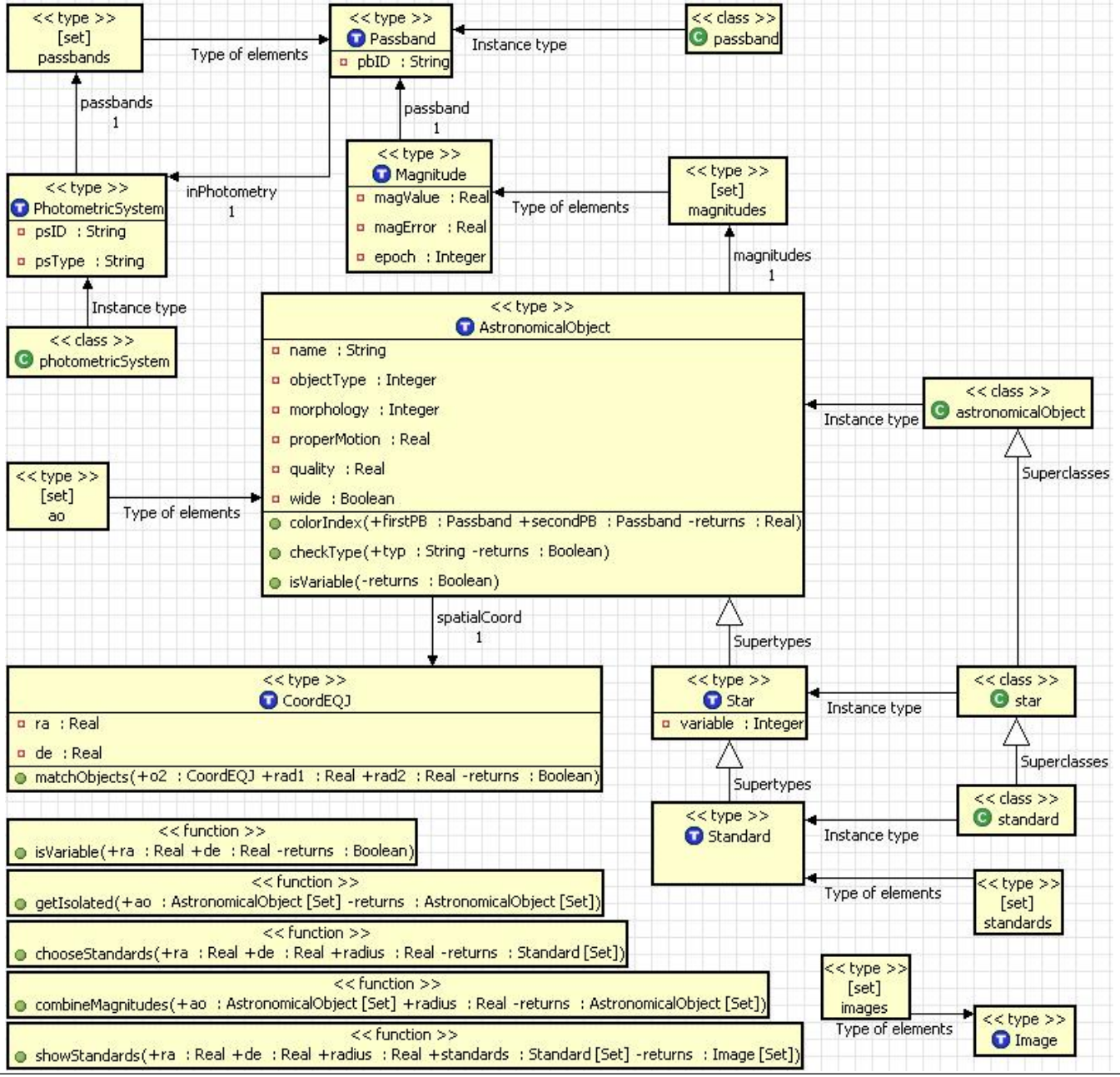
- Photometric System – a set of passbands with particular characteristics
  - "Слоановской фотометрической системы (ugriz)"
  - "Джонсона-Кузинса (UBVRI)"
  - "HJK"
- Passband – the portion of spectrum, between limiting wavelengths, that is transmitted with minimum relative loss or maximum relative gain by a filtering device
  - U, B, V, R, I,
  - H, J, K,
  - u, g, r, i, z
- Astronomical Object – significant physical entity on the sky having electro-magnetic emission in particular passbands

Пример спецификации онтологических понятий

```
{ AstronomicalObject;
    metaframe
    in: metaproperty, entity;
    end
  in: type, concept;
  def: {significant physical entity on the sky having
       electro-magnetic emission in particular passbands};
  morphology: {enum; enum_list: {point, fuse, elliptic}};
  coordinate: Coordinate;
  magnitudes: {set; type_of_element: Magnitude;};
  properMotion: ProperMotion;
  colorIndex: ColorIndex;
  isolated: Isolation;
  quality: Quality;
}
```

На основе онтологических понятий конструируется концептуальная схема

Концептуальная схема для задачи

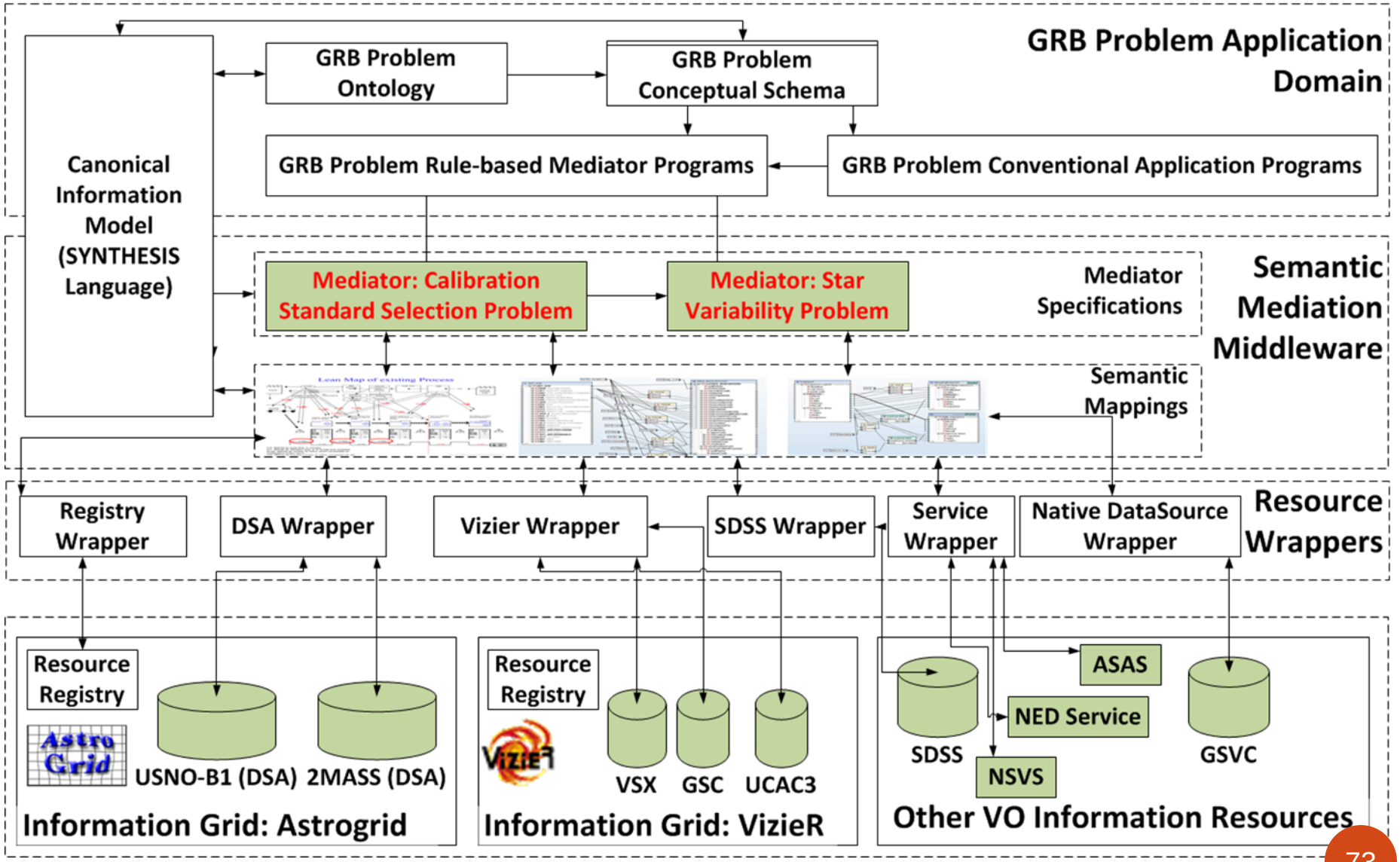


Тип: Астрономический объект

В типе AstronomicalObject определяется:

- идентификатор или название объекта (name)
- тип объекта (objectType: star, galaxy, comet, asteroid, planet, quasar, ghost, artifact, unknown)
- координаты объекта (coordEQJ)
- морфология объекта (morphology: point, fuse, elliptic)
- множество магнитуд в разных полосах известных для данного объекта (magnitudes)
- протяженность объекта (wide)
- характеристику движения (propertyMotion)
- качество данных (quality)
- метод вычисления индекса цвета (colorIndex)
- метод определения типа объекта (objectType) по некоторому эталонному каталогу (каталогам) (checkType)
- метод , определяющий является ли объект переменным на основе данных из ряда источников (isVariable)

Инфраструктура среды предметных посредников для решаемой задачи



Автоматический процесс решения задачи

1) Извлечение объектов в некоторой площадке из всех зарегистрированных каталогов
(USNO-B1, SDSS, 2MASS) и их кросс-идентификация

2) Отбор изолированных и непеременных объектов (VSX, GSC, UCAC3, NSVS, ASAS, GSVC)

3) Отбор среди объектов звезд с необходимыми спектральными параметрами, достаточно качественными данными и малым собственным движением

4) Отображение выбранных стандартов

Пример правила программы над посредником

```
standard(x/[ra, de, magnitudes, objectType, properMotion,
  quality])
:-
astronomicalObject(x1/[ra: spatialCoord.ra, de:
  spatialCoord.de, objectType, properMotion, quality,
  magnitudes])
& ra < queryRA + radius & ra > queryRA - radius
& de < queryDE + radius & de > queryDE - radius
& isVariable(x1, isVar) & isVar = false
& isIsolated(x1, isolated) & isolated = true
& checkType(x1, 'G', nType) & nType = false
& objectType = Star
& properMotion < 0.01
& quality  < 0.01
```

- Правило по существу решает задачу
- За предикатом astronomicalObject скрываются зарегистрированные в посреднике каталоги

Пример результата grb090621b-refstars

center RA = 20:53:52.92, Dec = +69:01:42.4

Были отобраны 12 объектов:

Name	ra	de
1590-0172130	20:53:41.83944	+69:01:45.4224
20534605+6857386	20:53:46.056	+68:57:38.6892
20534672+6855471	20:53:46.72296	+68:55:47.1756
20540713+6903140	20:54:07.13088	+69:03:14.0724
20533385+6903080	20:53:33.85464	+69:03:08.0316
20541305+6858245	20:54:13.05552	+68:58:24.528
20540518+6902392	20:54:05.18232	+69:02:39.2208
20533014+6902038	20:53:30.14448	+69:02:03.8148
20533149+6905218	20:53:31.49472	+69:05:21.8436
20534225+6905546	20:53:42.25752	+69:05:54.6648
20540218+6902035	20:54:02.18856	+69:02:03.57
20540401+6905145	20:54:04.01856	+69:05:14.5392