Шахгельдян К.И.

Автоматическая интеграция данных на базе онтологического подхода

Введение

Корпоративная информационная среда (КИС) вуза из средства предоставления доступа к информации превратилась в настоящее время в обязательный компонент жизнедеятельности вуза на всех уровнях: управление, исполнение, обучение. Интеграция данных одна из самых важных задач построения КИС, связана с многообразием объектов среды: серверов, баз данных, приложений.

КИС вуза обычно содержит от 20 до 100 серверов, часть из которых представляют собой серверы баз данных. Наличие множества серверов баз данных объясняется несколькими причинами:

1. использование в КИС различных информационных систем (ИС) с разными системами управления базами данных (СУБД);
2. распределение нагрузки на несколько серверов для повышения производительности;
3. использование серверов в удаленных зданиях (часто в разных городах), соединенных каналом с ограниченной пропускной способностью.

В большинстве случаев в связи с наличием множества серверов СУБД и ИС приходится решать проблемы интеграции данных. Среди этих проблем, прежде всего, возникает проблема репликации данных. В сложной КИС число репликаций доходит до нескольких сотен. При этом внутри каждой репликации число реплицируемых объектов базы данных в среднем составляет около десятка. Таким образом, в рамках сложной КИС с несколькими филиалами, в которых функционируют собственные серверы баз данных, число реплицируемых объектов

достигает нескольких тысяч. В результате возникают три основные проблемы:

1. как корректно реплицировать объекты, так как многие их них связаны друг с другом;

2. каким образом отслеживать все требуемые к репликации объекты;

3. каким образом обеспечить синхронизацию реплицируемых объектов.

Для организации репликаций данных в основном используются встроенные средства СУБД. Но для КИС вузов характерна гетерогенность, что затрудняет использование встроенных средств репликации. В тех же случаях, когда встроенные средства могут применяться, они не позволяют в полной мере решить проблемы синхронизации объектов, а так же не решают проблем обеспечения качества данных, связанных с рассогласованием используемых справочников.

В последнее время появились работы по применению онтологического подхода для интеграции данных /1, 2/. В этих работах рассматривается проблема организации хранилища структурированной информации. Использование онтологического подхода здесь позволяет создавать единый репозиторий данных, а так же обеспечивать понимание семантики хранящейся в различных базах данных информации.

Одна из проблем сопровождения и эксплуатации сложной КИС вуза связана с большим количеством и разнообразием объектов, используемых в КИС. Стратегия борьбы с проблемой количества и многообразия состоит в обеспечении автоматического выполнения всех процедур, которые могут быть выполнены автоматически.

В этой работе предлагается использовать онтологии для реализации автоматической репликации данных в гетерогенной КИС.

Онтологии

Остановимся кратко на онтологическом подходе, используемом при создании ИС /3-5/.

Онтологии определяют термины для представления или описания некоторой предметной области. Онтологии включают определенные пользователем формальные базовые концепции предметной области и их взаимоотношения. Компонентами онтологии являются концепты или *классы*, которые описывают понятия предметной области, *свойства* класса, которые описывают различные характеристики концепта, и *ограничения на свойства*. Класс может иметь подклассы, которые представляют собой более специализированные концепты, чем надклассы.

Свойства классов представляют собой либо атрибуты, либо отношения между классами. Онтологии имеют мощный инструмент формирования произвольных отношений между классами. Собственно отношения являются свойствами классов. Свойства (а, следовательно, отношения) могут иметь характеристики.

Возможность формировать собственные отношения между классами, т.е. добавлять в разные классы взаимозависимые свойства и накладывать на них характеристики симметричности, транзитивности, инверсности, функциональной эквивалентности и обратной к ней, позволяет автоматизировать вывод новых знаний.

На отношения наследования наложены по умолчанию свойства транзитивности, т.е. если класс Y является подклассом X, а класс X является подклассом Z, то класс Y, является подклассом Z. Транзитивность так же является характеристикой свойства агрегации,

т.е. отношений быть частью. Если класс X содержит класс Y, а класс Y содержит класс Z, то класс X содержит класс Z.

С другой стороны, если класс X имеет свойство быть частью класса Y, то класс Y может иметь инверсное свойство содержит экземпляры класса X.

В онтологии есть два типа явных знаний: аксиомы и общие правила. Аксиомы это стандартные (по умолчанию) множества правил, такие как правила для симметричных, транзитивных или инверсных свойств. Например, если X имеет отношение с Y, а Y имеет те же отношения с Z, и при этом эти отношения транзитивны, то онтологическая система делает заключение, что X имеет отношения с Z и такие отношения не требуют явного описания. Общие правила – это правила некоторой предметной области. Например, если X содержит Y, а Y имеет некоторое отношение к Z, то можно заключить, что X так же имеет некоторое отношение к Z.

Аксиомы и правила используются для того, чтобы вывести новые знания.

На свойства могут накладываться ограничения. Самым простым ограничением являются ограничения типов данных, накладываемые на атрибуты. Второй тип ограничений обычно связан с отношениями. В этот тип ограничений входят ограничения на допустимые значения экземпляра класса y в отношениях $P(x, y)$. Допустимость значений определяется множеством существующих экземпляров класса y. Ограничения могут накладываться и на мощность множества экземпляров класса y в отношениях $P(x, y)$. Допускается указывать минимальное, максимальное и точное число в множестве. К

ограничениям относится так же требование на равенство y в отношениях $P(x, y)$ некоторому определенному экземпляру класса.

Использование онтологий дает большие преимущества. Во-первых, онтологии обеспечивают возможность понимания различными системами одних и тех же вопросов предметной области (это преимущество используются при построении хранилищ данных в работах /1, 2/). Во-вторых, онтологии обеспечивают понимание предметной области информационными системами. В-третьих, онтологии обеспечивают возможность выполнения проверок, корректировок различных ситуаций для обеспечения эффективного и корректного функционирования информационных систем и поддержку качества данных. Последние два преимущества онтологий используется в данной работе для обеспечения автоматического поддержания интеграции данных.

В КИС вуза предлагается определить базовый класс Object и любые пользовательские классы и отношения являются производными от класса Object.

Отношения между понятиями

В КИС между понятиями поддерживаются произвольные отношения, которые могут определяться не только разработчиками, но и бизнес-аналитиками. Класс X имеет произвольные отношения с классом Y, при этом отношения могут быть представлены

- свойством класса X с указанием на класс Y;
- классом Z, которое имеет отношения с классами X и Y.

Одними из важных отношений между классами в КИС вуза являются отношения наследования и проекции. В рамках модели, рассматриваемой в работе, наследования между класами X и Y

$P'(X,Y)$ определено в случае, если класс Y имеет все те же атрибуты, что и класс X, но на некоторые из которых могут быть наложены более жесткие ограничения, чем на атрибуты класса X, при этом у Y так же могут быть дополнительные атрибуты. В этом случае класс Y может наследоваться от X. В случае, если в производном классе Y добавлены атрибуты по сравнению с классом X, то определено представление $|Y|_X$, которое позволяет выделять в экземпляре класса Y часть, которая определяется только семантикой X.

Наложение более жестких ограничений на атрибуты приводит к уменьшению мощности множества экземпляров производных классов по сравнению с базовыми.

Отношение проекции описывают представление классов предметных областей в области информационных технологий (ИТ). Проекция – это отношения, описывающие соответствие между понятиями предметной области и классом Источник данных.

Отношения проекции описаны между базовым классом КИС Object и классом Источник данных и имеют атрибутами тип проекции. Отношения проекции определяют связи между классом X и источником данных A: $P(X,A)$. Источник данных представляет собой таблицу или представление в системах с реляционной моделью данных. Для систем с объектной моделью понятие предметной области имеет отношения проекции с классом базы данных, т.е. источник данных представляет собой некоторый объект.

При наличии единой базы данных одно понятие предметной области имеет отношения проекции с одним экземпляром класса источника данных. В КИС вуза база данных не единая, могут иметь

место многочисленные реплики. В этом случае определяются отношения классов с несколькими источниками данных. Отношения проекции должны иметь атрибут, определяющий ограничения доступа. В общем случае выделяем 3 типа отношений проекция:

- 0 – отсутствуют отношения проекции между классами и источником (в большинстве случае не используется, за исключением ситуации, возникающей при переопределении источников в иерархии классов);
- 1 – источник является контейнером всех экземпляров класса, при этом разрешено как чтение, так и запись экземпляров;
- 2 – источник является контейнером экземпляров класса с правом только на чтение.

Отношения проекции кроме типа имеют множество соответствий между атрибутами классов и атрибутами источника данных. В отношении проекция, описанном в базовом классе Object, единственным атрибутом является тип отношений проекции. В производных классах тип отношений может быть переопределен, а так же добавлено множество соответствий между атрибутами класса и атрибутам источника данных.

Отношения проекции между одним классом и разными источниками допускает определения соответствия не со всеми атрибутами класса. Часть атрибутов в отдельных источниках может не использоваться, и кроме того, онтологическая модель КИС позволяет понятию иметь виртуальные атрибуты, которые определяются как регулярное выражение из других атрибутов понятия и могут не иметь проекции на область ИТ (для оперативных данных) или иметь проекцию (для агрегированного хранилища).

Рассмотрим пример отношений проекции между классами Подразделение X, Подразделение головного вуза Y, Подразделение филиала Z и источниками данных, расположенными на серверах A (корпоративный сервер в головном вузе), B (сервер филиала), C (сервер, используемый для чтения информации и представлении ее в портале вуза, содержит не все атрибуты класса Подразделение).

Класс Подразделение является базовым в отношении классов Подразделения головного вуза $P'(X,Y)$ и Подразделения филиала $P'(X,Z)$. Подразделения головного вуза имеют определенное значение атрибута бизнес-единицы (BUID), так же как и подразделения филиала. На сервере A (далее источник A) хранятся все экземпляры всех подразделений вуза, но при этом подразделения головного вуза вносятся непосредственно в источник A, а подразделения филиала в источник, расположенный на сервере B (далее источник B). Источник B содержит только подразделения филиала. На сервере C (далее источник C) хранятся все подразделения, но данные туда не вносятся, а лишь используются в других системах, при этом некоторые атрибуты, например, BUID, не используются.

Для полного описания отношений проекция между понятиями предметной области и ИТ-области необходимо:

- описать отношения проекции между Подразделениями X и источником A с типом 2 $P(X,A\,|\,2)$ и определить соответствия между атрибутами Подразделения (BUID, ID, Name и т.п.) и атрибутами источника данных;

- описать отношения проекции между Подразделениями и источником C с типом 2 $P(X, C \mid 2)$ и определить соответствия между атрибутами Подразделения (ID, Name) и атрибутами источника данных;
- описать отношения проекции между Подразделениями головного вуза и источником A с типом 1 $P(Y, A \mid 1)$, никаких дополнительных соответствий не определять, используются соответствия отношения проекции между базовым классом Подразделения;
- описать отношения проекции между Подразделениями филиала и источником B с типом 1 $P(Z, B \mid 1)$ и определить соответствия между атрибутами Подразделения филиала (BUID, ID, Name и т.п.) и атрибутами источника данных.

Пусть определены классы X, Y, между которыми установлены отношения наследования $P'(X, Y)$. Пусть определены так же отношения проекции с типом 2 $P(X, A \mid 2)$. Онтологическое правило позволяет сделать вывод о существовании в этом случае отношений $P(|Y|_A, A \mid 2)$.

В иерархии любого понятия предметной области должны быть отношения проекции с типом 1 с одним источником данных. Это ограничение обосновано необходимостью создавать экземпляры любого понятия предметной области средствами КИС и иметь их отражение на ИТ-область.

Если определены отношения $P'(X, Y)$, $P'(X, Z)$ и $P(Y, A \mid 1)$, при этом $Y \cap Z = \varnothing$, то существуют отношения проекции $P(Z, B \mid 1)$,

при этом A и B могут не совпадать. Если такое отношение проекции отсутствует, то это свидетельствует о наличии экземпляров класса X, которые не имеют средств редактирования в КИС.

Отношения между классами так же имеют проекцию на экземпляры класса источник данных. Если источник данных описывает таблицу или представление, то отношения между классами проецируются на уровень связей в СУБД (внешние ключи). В гетерогенной информационной среде, где присутствуют различные СУБД, не всегда одинаковой архитектуры, невозможно описать все отношения между классами на уровне СУБД, так как часто отсутствует возможность использовать внешние ключи. В этом случае описание отношений между классами в полной мере лежит на онтологиях. Такое разделение созвучно понятию связей между данными в базах данных на физическом и логическом уровне. Физический уровень связей предполагает наличие связей на уровне СУБД (внешние ключи), логический уровень – описание связей на уровне программ.

Отношения проекции между классами и источниками данных транслируют класс в таблицы или представления базы данных, а атрибуты класса в поля таблицы или представления.

В отношении проекции определены соответствия для ограничений:

1. ограничение на точную мощность множества атрибута или свойства = 1 (2,3 и т.п.) соответствует обязательному присутствию атрибута(ов) с ограничением значения поля not null;
2. ограничение на максимальную мощность множества =1 соответствует атрибуту, возможно со значением null;

3. ограничение на максимальную мощность множества атрибутов > 1 предполагает наличие в таблице соответствующего числа столбцов, возможно со значением null в соответствующих полях;
4. ограничение на точную или максимальную мощность множества y в отношении $P(x, y)$ > 1 соответствует использованию дополнительной таблицы с полями идентификатор экземпляра понятия и значения атрибута;
5. допустимое множество значений для атрибутов, диапазон значений, регулярные выражения транслируются в ограничения на поля таблицы (CHECK).

Отношения проекции с реляционными источниками в КИС могут описываться на основании уже существующих таблиц и представлений, поэтому все возможные ограничения на поля таблиц могут быть обратно транслированы в описания понятия и отношений в КИС.

1. Если таблица A имеет внешний ключ с таблицей B, то понятие, которое имеют проекцию с источником данных A связано некоторыми отношениями с понятием, которое имеет отношение проекции с источником данных B.
2. Если ограничения на столбцы таблицы A включают ограничения not null, то это транслируется в ограничения на точную мощность множества атрибутов понятия, имеющего отношения проекции с источником A, равную 1.
3. Если ограничения на столбцы таблицы A включают ограничения возможно null, то это транслируется в ограничения на максимальную мощность множества атрибутов, равную 1.

4. Если в описании таблицы присутствует CHECK, то это может транслироваться в соответствующие ограничения на атрибуты понятия.

Отношения проекции можно описать и следующим образом

$$E^{DB} = \{e_{ij}\} = \begin{cases} 0,\ b_j \text{ источник данных не имеет} \\ \quad \text{отношений проекции с } d_i \text{ понятием} \\ 1,\ b_j \text{ источник данных является контейнером} \\ \quad \text{экземпляров } d_i \text{ понятия с правами} \\ \quad \text{чтение/запись} \\ 2, b_j \text{ источник данных является контейнером} \\ \quad \text{экземпляров } d_i \text{ понятия с правами чтение} \end{cases} \quad (1)$$

Рассмотрим, каким образом, использование модели отношений проекции между понятиями позволяют реализовать алгоритмы репликации данных.

Репликация данных

Репликация определяется как процесс автоматического распределения копий данных и объектов баз данных между экземплярами сервера баз данных с одновременной синхронизацией всей распространяемой информации.

Проблема, которая встает перед разработчиками в сложной гетерогенной КИС, состоит в эффективной организации репликций с учетом необходимости согласования между ними, с учетом высокоуровневого управления репликациями и с необходимостью увязывать репликации с различными требованиями и ограничениями в КИС.

С некоторой долей условности данные в корпоративной базе данных можно разделить на справочники и оперативные данные.

Определим справочники как относительно статическую информацию, которая, может меняться (удаляться, добавляться, редактироваться), но это происходит не часто, и сами по себе данные в справочниках не представляют значимую информацию, она всегда используется в оперативных данных либо непосредственно, либо через связи с другими справочниками. Кроме того, зависимость между справочниками и оперативными данными односторонняя, т.е. оперативные данные зависят от справочников, но не наоборот.

Данные в КИС вуза реплицируются в нескольких случаях.

1. Данные из серверов корпоративных баз данных реплицируются на серверы, используемые для чтения данных (Рис.1; $X \Rightarrow A$, $Y \Rightarrow A$). Основная причина такой репликации – необходимость размещать серверы базы данных в демилитаризованной зоне (ДМЗ) КИС, а так же использование отдельных серверов для чтения информации, например, во внешнем портале вуза.
2. Данные из одних корпоративных баз данных реплицируются на другие корпоративные серверы баз данных (Рис.1; $X \Rightarrow Y$). В этом случае причина репликации связана с повышением производительности, так как запросы одновременно к двум серверам выполняются медленнее, чем, если базы данных будут храниться на одном сервере. Такие репликации организуются для задач, где необходима высокая производительность или установлены требования реального времени.
3. Справочники из корпоративных баз данных реплицируются на серверы филиалов (Рис.1; $X \Rightarrow Z$, $Y \Rightarrow Z$). Репликации связаны с необходимостью построения распределенной базы данных, когда информация по филиалам вводится в филиалах,

но при этом используются справочники, заполняемые в основной базе данных. Эти справочники реплицируются в филиалы.

4. Оперативные данные из серверов филиалов реплицируются в центральные корпоративные базы данных (Рис.1; $Z \Rightarrow X$, $Z \Rightarrow Y$). Репликации связаны с необходимостью построения распределенной базы данных, когда данные по филиалам вводятся в филиалах, а затем, для дальнейшей работы реплицируются в основную базу. Еще одной причиной репликации в этом случае является минимизация трафика между серверами в рабочие часы.
5. Данные из корпоративной базы данных реплицируются на зеркальный сервер, чтобы обеспечить надежность функционирования системы в случае выхода из строя корпоративного сервера (Рис.1; $X \Rightarrow W$, $Y \Rightarrow W$).
6. Данные реплицируются из одной базы данных в другую в виду разной архитектуры этих баз данных или разной модели данных (Рис.1; $C \Rightarrow A$). Решение обычно состоит в выборе одного из справочников, в качестве базового классификатора и репликации его в другие базы данных. Например, в КИС Владивостокского государственного университета экономики и сервиса (ВГУЭС) данные об организационной структуре и сотрудниках реплицируются из MS SQL Server в Lotus Notes для реализации информационной системы планирования и составления отчетов сотрудниками вуза, а так же данные реплицируются в LDAP-серверы для управления сетевой инфраструктурой (Рис.1; $X \Rightarrow Empl, X \Rightarrow Branch, Y \Rightarrow Stud, Y \Rightarrow Branch$).

7. Данные реплицируются на другой сервер для интеграции стороннего приложения (например, портала) в КИС, не имеющего средств интеграции на лету (Рис.1; $A \Rightarrow B$).

8. Данные реплицируются на мобильные рабочие места (Рис.1; $X \Rightarrow V$) с целью их использования в местах без сетевой инфраструктуры; данные с мобильного рабочего места реплицируются обратно (Рис.1; $V \Rightarrow X$). Примером такого использования могут служить автоматизация задачи инвентаризации с использованием штрих-кодов в местах без доступа к сети.

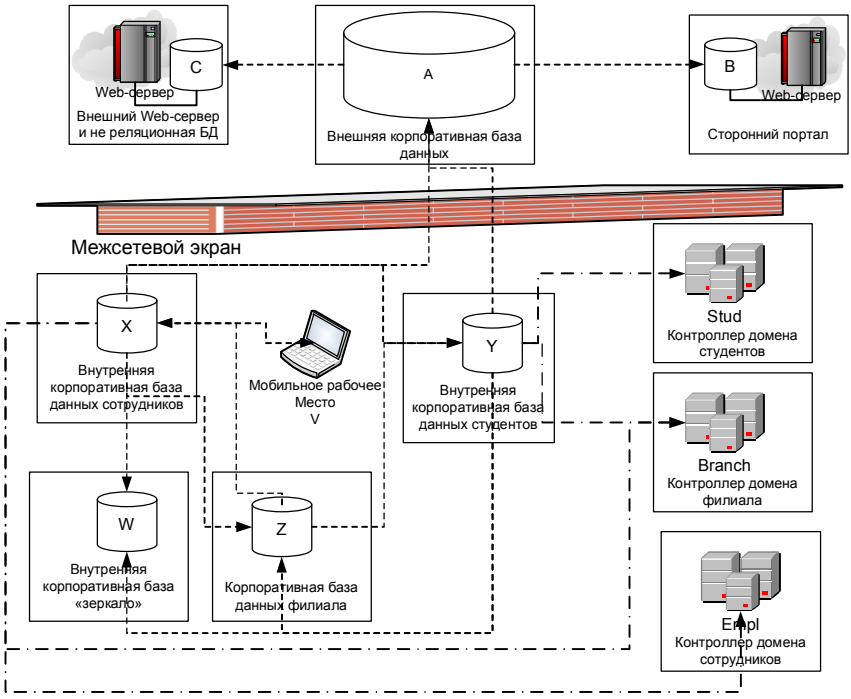

Рис.1. Общая схема репликации данных в КИС

Репликации организуются так часто как необходимо (при изменении источника, один или несколько раз в час, один раз в день, один раз в неделю, один раз в месяц и т.п.).

Для каждого понятия существует единственный источник данных с первичными данными, т.е.

$$\forall i \; \exists! \, j \; e_{ij}^{DB} = 1. \qquad (2)$$

Выделение первичного источника данных (т.е. источника, в котором данные создаются) позволяет избежать проблем двойного ввода. Понятие первичного источника данных включает два ограничения. Во-первых, в этом источнике хранятся данные, которые введены в одной системе, а все другие системы только пользуются этими данными, никак их не меняя. Во-вторых, данные из этого источника реплицируются в другие источники, где они могут использоваться в режиме для чтения, т.е. там, где $e_{ij}^{DB} = 2$.

За ввод экземпляров некоторого понятия отвечает единственное приложение. В модели интегрированных приложений КИС /6/ за создание в источнике данных проекции экземпляра некоторого понятия отвечает единственная серверная компонента. Такое ограничение позволяет локализовать поиск мест для исправления ввиду изменения семантики понятия и для поиска ошибок ввода. В некоторых случаях это условие не выполнятся, но число таких систем должно быть ограничено.

Репликации бывают синхронные и асинхронные. Синхронные репликации выполняются одна после другой. Т.е. для синхронных репликаций действует понятие «раньше». Понятие «раньше» означает, что репликация A выполняется после окончания репликации B, при этом репликация A связана по крайне мере с

одним из серверов, с которым связана репликация B. Асинхронные репликации выполняются независимо друг от друга и понятия «раньше» там не существует.

В больших КИС, где много серверов, возникают проблемы некорректно выполненных репликаций. Например, реплицировать данные из корпоративного сервера в ДМЗ имеет смысл только после того, как прошла репликация из филиалов в корпоративную базу данных, т.е. $Z \Rightarrow X$ и $Z \Rightarrow Y$ должны быть выполнены раньше $X \Rightarrow A$ и $Y \Rightarrow A$. При этом репликация $Z \Rightarrow X$ должна быть выполнена раньше репликации $Z \Rightarrow Y$, а репликация $X \Rightarrow A$ - раньше $Y \Rightarrow A$. Так как репликации находятся на разных серверах, то стандартными средствами синхронизации СУБД эту проблему решить нельзя.

Для выполнения сложной синхронизированной репликации необходимо описания общей схемы репликации в КИС. Такая схема позволяет, во-первых, корректно выполнять репликации, во-вторых, администратору КИС корректно отрабатывать ошибки в цепочке синхронных репликаций. Так как КИС постоянно изменяется, то такая схема должна быть создана и поддерживаться в актуальном состоянии автоматически.

Алгоритм автоматической репликации данных

В сложной КИС количество репликаций может переходить за сотни и корректно организовать их исполнение (синхронизацию, содержимое, время) часто достаточно сложно. Использование онтологического подхода позволяет автоматизировать организацию репликаций в КИС на базе правил.

Понятия КИС имеют отношения проекции с различными источниками данных. Единственное требование состоит в том, что для одного класса отношения проекции с типом доступа на чтение/запись существуют с единственным источником данных (2).

Сформулируем правила генерации автоматической репликации.

1. Если класс X имеет отношения проекции с источником A с правом доступа на чтение $P(X, A \mid 2)$, класс Y, производный от X (т.е. определены отношения $P'(X, Y)$), имеет отношения проекции с источником B с правом доступа на чтение/запись $P(Y, B \mid 1)$ и $X \neq Y$, то должна существовать репликация всех объектов из источника B в источник A (Рис. 2), т.е. отношения репликации между источниками $P''(B, A)$. В общем случае понятие Y может содержать дополнительные атрибуты по сравнению с базовым понятием X. В этом случае при автоматической репликации дополнительные атрибуты не реплицируются.

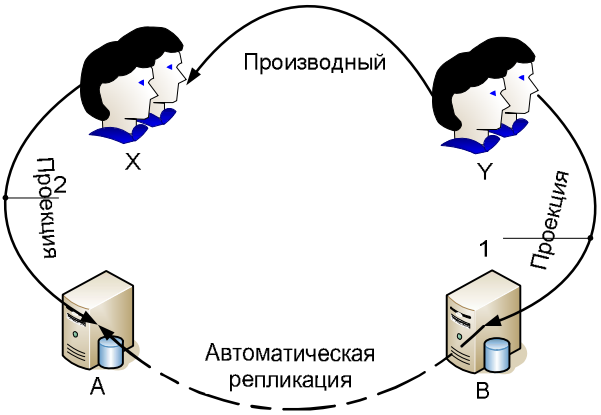

Рис. 2. Автоматическая репликация данных на базе отношений наследования и проекции (из производного класса в базовый)

2. Если класс X имеет проекции с источниками A и B с типом доступа чтения и чтения/записи соответственно $P(X, A \mid 2)$, $P(X, B \mid 1)$, то должна существовать репликация всех экземпляров класса X, хранящихся в источнике *B*, в источник A $P''(B, A)$ (Рис.3).

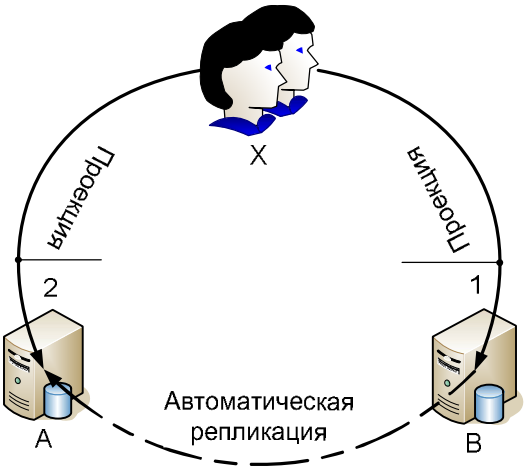

Рис.3. Автоматическая репликация данных на базе отношений проекции

3. Если класс X имеет отношения проекции с источником A с правом доступа на чтение/запись $P(X, A \mid 1)$ и класс Y – производный от класса X $P'(X, Y)$, имеет отношения проекции с источником B с правом доступа на чтение $P(Y, B \mid 2)$ и $X \neq Y$, то

должна существовать репликация всех объектов класса Y из источника A в источник B $P''(A,B)$ (Рис.4.).

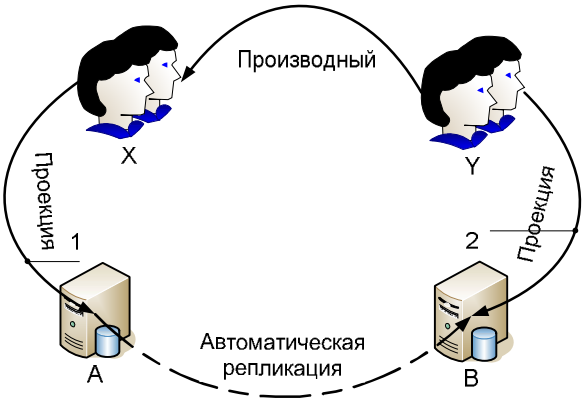

Рис.4. Автоматическая репликация данных на базе отношений проекции и наследования (из базового класса в производный)

4. Наличие нескольких источников данных, имеющих отношения проекции с правами на чтение с базовым понятием $P(X,A\,|\,2), P(X,C\,|\,2)$ и наличие отношений проекции между производным классом $P(Y,B\,|\,1), P'(X,Y)$, приводит к репликациям вида $P''(B,A), P''(B,C)$ (Рис.5).

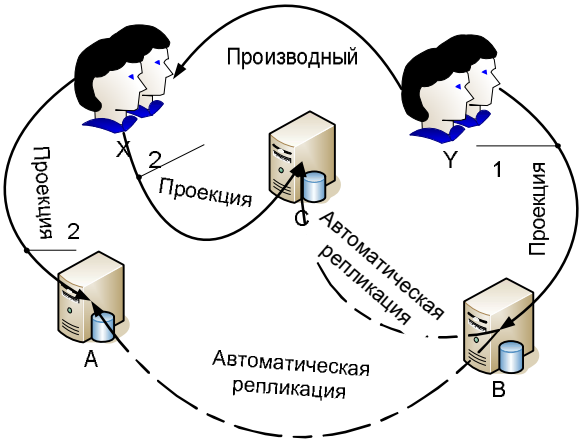

Рис.5. Автоматическая репликация данных в несколько источников на базе отношений

Алгоритм репликации состоит в следующем.

1. Определяются отношения проекции с правами чтения/запись для всех классов в некоторой иерархии. Поиск понятий начинается от базового понятия Object.
2. Для каждого понятия X, имеющего отношения проекции с некоторым источником A_X с правами чтения/запись $P(X, A_X \mid 1)$, определяются все источники, с которым у X есть отношения проекции с правами на чтение $P(X, B_X^i \mid 2)_{i=\overline{1,N_X}}$. Согласно правилу 2 генерируются репликации вида $P''(A_X, B_X^i)_{i=\overline{1,N_X}}$.
3. Все производные классы от X (если он имеет отношения проекции с типом на чтения/запись) должны иметь отношения проекции только с правами на чтение

$P'(X, Y_i)_{i=\overline{1,N_Y}}, P(Y_i, C_i \mid 2)_{i=\overline{1,N_Y}}$, отсюда следует, что определена автоматическая репликация $P''(A, C_i)_{i=\overline{1,N_Y}}$.

4. Если первый класс, который встретилось в иерархии, имеет отношения проекции только с правами на чтение $P(X, A_i \mid 2)_{i=\overline{1,N_X}}$, то происходит дальнейший поиск по иерархии с целью обнаружения производных классов, имеющих отношения проекции с правами на чтения/запись $P'(X, Y_i)_{i=\overline{1,N_Y}}, P(Y_i, B_i \mid 1)_{i=\overline{1,N_Y}}$. Для таких классов, во-первых, применяется правило 2 и определяется репликация в источники с отношениями проекции только на чтение $P(Y_i, C_{ij})_{i=\overline{1,N_Y}, j=\overline{1,N_{Y_i}}}$ $P_2(B_i, C_{ij})_{i=\overline{1,N_Y}, j=\overline{1,N_{Y_i}}}$. Во-вторых, для базового класса X рассматриваются источники, с которыми у X есть отношения проекции с правами только на чтение $P(X, A_i \mid 2)_{i=\overline{1,N_X}}$. В общем случае должны существовать репликации вида $P_2(B_j, A_i)_{j=\overline{1,N_Y}, i=\overline{1,N_X}}$. Но в более частном случае часть источников B_j, A_i могут совпадать. В случае совпадения источников репликация не генерируется.

Пример организации репликации приведен на

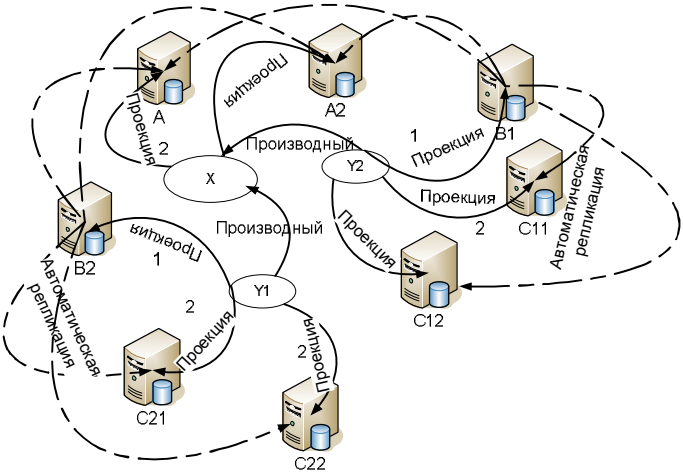

Рис.6.

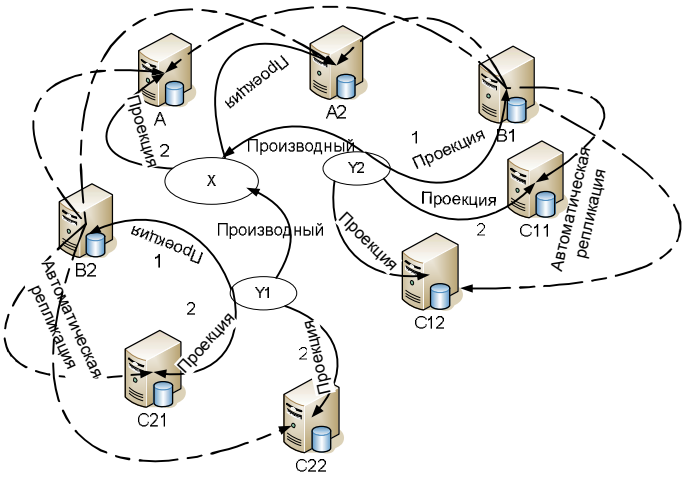

Рис.6. Пример автоматических репликаций

При выполнении автоматической репликации может возникнуть необходимость не в простом копировании значений, а в выполнении некоторого преобразования. Например, класс X, состоит из двух атрибутов $X = \{x_1, x_2\}$, а при репликации необходимо сохранять класс с одним атрибутом, равным сумме двух атрибутов X. Для реализации это идеи формируется класс $Y : P'(X, Y), Y = \{x_1, x_2, y = x_1 + x_2\}$. При определении отношений проекции для $P(Y, B \mid 2)$ определяется соответствие только для атрибута y. В результате репликации в источнике B хранятся агрегированные по атрибутам экземпляры класса X.

Использование иерархии классов позволяет определять условия выборки множества объектов для репликации. Например, если

необходимо реплицировать не всех студентов, а только студентов высшего профессионального образования (ВПО), то используется иерархия классов – базовый класс Студент и производный класс – Студент ВПО, при этом при формировании отношений иерархии задаются условия отношений /6/. Эти условия используются затем при реализации репликации для выборки всех студентов ВПО из общего множества студентов.

Если в производном классе имеются атрибуты, которые отсутствуют в базовом и определена репликация из источниками базового класса в источник производного, то атрибуты производного класса, которые не включены в базовый класс могут быть определены на основании значений по умолчанию – ограничений для атрибутов классов в КИС. Значения по умолчанию представляют собой некоторые регулярные выражения. Если значение не определено, то оно устанавливается в null, если это атрибут класса. Для свойств, которые не определены в базовом классе и не имеют значений по умолчанию в производном классе, при репликации экземпляры отношений не создаются.

Проблема, которая возникает в задачах репликаций, решаемых другими средствами /2/, связана с разными именами исходной и целевой таблицы и полей. Это проблема легко решается на основании описания отношений проекция одних и тех же атрибутов реплицируемого понятия на источники данных и поля с разными именами.

Еще одной проблемой репликаций является проблема разных типов данных в источнике и приемнике репликации. Решение этой проблемы зависит от того, намеренные или ошибочные эти расхождения. В КИС в модуле контроля корректности описания классов выполняются проверки на совпадения типов атрибутов

источников данных, которые являются проекцией одних и тех же атрибутов классов в разных источниках. При нахождении различий выполняется уведомление администратора проекта.

В отдельных случаях такие расхождения определены намерено, и для осуществления репликации требуется преобразования типов. Для преобразования могут использоваться либо стандартные средства СУБД, либо описаны специфические преобразования одних типов в другие. Но здесь так же используются функции или операторы SQL.

Простые преобразования так же могут быть связаны с некоторым упрощением исходной информации. Например, исходные поля проецируют атрибут даты в формате год/месяц/день/час/секунда. Атрибуты результирующего класса проецируются на поле с типом данных строка, содержащей информацию о годе/месяце/дне. Такое преобразование может быть осуществлено автоматически средствами СУБД, но при этом требуется указать необходимость урезания данных до даты. Такие преобразования могут быть подставлены в репликации автоматически при нахождении расхождения в типах данных.

Каждый экземпляр класса Источник данных связан с одной стороны с сервером и базой данных, с другой – отношениями проекция, - с понятиями предметной области. Таким образом, всегда можно оценить, какие понятия предметной области, каким образом, на какие серверы и когда реплицируются.

Обеспечение синхронизации данных

Если речь идет об одном сервере, на котором выполняются репликации, то они легко синхронизируются по успешному окончанию работы предыдущей репликации стандартными средствами синхронизации СУБД. Но в целом репликации копируют данные с разных серверов (которые, кроме того, могут иметь различную

архитектуру), поэтому обычными средствами СУБД для репликации данных выполнить их по успешному завершению нельзя.

Для реализации синхронных репликаций предлагается использовать автоматически сгенерированные репликации и отношения между классами. Как рассматривалось выше, на основании отношений проекции и наследования могут быть сгенерированы репликации. Но порядок их следования должен быть уточнен.

Каждая автоматически сгенерированная репликация содержит копирования экземпляров одного класса.

Рассмотрим пример. Пусть определены отношения проекции $P(X, A_X \mid 1), P(X, B_X \mid 2), P(X, C_X \mid 2)$. Онтологическая машина делает заключение, что должны существовать отношения репликации $P''(A_X, B_X), P''(A_X, C_X)$. Пусть существует понятие Y, имеющие свойство, связывающее его с понятием X: $P_1(Y, X)$. Для понятия Y определены отношения наследования: $P'(Y, Y_1), P'(Y, Y_2)$ и отношения проекции $P(Y_1, A_Y \mid 1), P(Y_2, B_Y \mid 1), P(Y, A_Y \mid 2), P(Y, C_Y \mid 2)$. Онтологическая машина делает заключение о необходимости репликаций $P''(B_Y, A_Y), P''(A_Y, C_Y)$.

Таким образом, онтологическая машина на основании отношений проекции и наследование сгенерировала отношения репликации

$$P''(A_X, B_X), P''(A_X, C_X), P''(B_Y, A_Y), P''(A_Y, C_Y).$$

Для корректного выполнения репликаций необходимо определить порядок их следования и возможность параллельного выполнения.

Порядок следования репликаций с различными источниками-приемникам, определяется наличием отношений между классами.

Отношения $P_1(Y,X)$ определяют свойства класса Y, поэтому независимым считается класс X. Таким образом, первым реплицируются экземпляры класса X: $P''(A_X,B_X), P''(A_X,C_X)$. Класс Y связан единственным свойством $P_1(Y,X)$ с уже реплицированным классом, поэтому следующим блоком будет блок репликаций класса Y. Поскольку источники репликаций различны, то они не могут выполняться параллельно. Последовательность репликаций определяется тем, что один из источников является приемником в другой репликации, поэтому последовательность следующая – первой выполняется репликация $P''(B_Y,A_Y)$, а второй - $P''(A_Y,C_Y)$.

Алгоритм синхронизации репликаций состоит в следующем. Из всех классов, участвующих в репликациях, выбираются те классы, которые не имеют никаких отношений ни с одним другим классом, участвующим в репликациях. Фактически, на первом этапе выбираются независимые справочники. При выполнении репликаций на очередной итерации выбираются те репликации, которые связаны с классами либо не имеющими свойств (т.е. независимыми справочниками), либо их свойства связаны с уже реплицированными классами. Внутри блока репликаций одного класса в случае одного источника для нескольких репликаций, репликации могут выполняться параллельно. В противном случае порядок репликаций внутри блока репликаций одного класса определяется на базе источников и приемников репликации. В первую очередь выполняются те репликации, приемники в которых не являются источниками в других репликаций внутри блока.

Процесс выполнения автоматических репликаций с синхронизацией, как описано выше выполняется специализированным приложением. Но при наличии большого числа репликаций узким местом может оказаться сервер приложений, на котором будут выполняться приложение. Для решения проблемы производительности можно использовать другие серверы приложений с распределением на них работы.

Но здесь возникает проблема синхронизации приложений в распределенной системе. Одним из инструментов синхронизации является абстракция событие.

Событие определяется как значимое отражение изменения в состоянии некоторого ресурса и оно генерируется, чтобы известить об этом изменении /5/. С точки зрения онтологического подхода определим класс событие как отражение значимого изменения в состоянии некоторого экземпляра класса КИС или в связи с появлением или удалением экземпляра класса, а так же с ограничением на число экземпляров класса.

Между репликациями и событиями существуют отношения генерации $P_G\left(P''\left(A_X, B_X\right), \varepsilon_X\right)$, которые в зависимости от атрибутов определяют начало или завершение репликации.

С помощью установленных событий приложения, реализующие репликации зависимых классов на различных серверах, узнают о завершенной репликации тех классов, которые связаны с подготовленным к репликации классом.

В некоторых случаях репликация $P''\left(A_Y, C_Y\right)$ имеет смысл даже, если репликация $P''\left(B_Y, A_Y\right)$ не была выполнена. В этом случае на C_Y будут перенесены только те изменения, которые были сделаны за

предыдущий период в источнике данных A_Y, без учета изменений в источнике B_Y. Это возможность определяется наличием отношений проекции с правом на чтение/запись $P(Y_1, A_Y \mid 1)$. Приложение, выполняющее репликацию, определяет невозможность выполнить репликацию $P''(B_Y, A_Y)$, фиксирует неуспех репликации и выполняет репликацию $P''(A_Y, C_Y)$ с уведомлением администратора о неполных данных по экземплярам класса Y в источнике C_Y.

Проблема уникальных ключей

Одна из проблем, возникающих при репликации данных - это проблема идентификационных ключевых полей. Так как физически базы данных разные, то возникает необходимость синхронизировать между собой первичные ключи таблиц. Существует несколько подходов к решению этой проблемы.

1. Определяется для каждой базы данных диапазон изменения уникальных идентификаторов, при репликации данных в виду разных идентификаторов наложения не произойдет. Но так как на поле первичного ключа может быть наложено требования авто увеличения, то данное решение не будет корректным – диапазон будет нарушен после первой же репликации.
2. Разновидностью первого пункта может быть второй пункт, когда в каждой базе данных идентификаторы вычисляется по некоторой формуле. Например, всего филиалов не более 10, тогда остаток от деления на 10 идентификатора определяет, в какой базе генерируется идентификатор. Подход хорошо работает, если это учитывалось с самого начала построения КИС. Но так как центральные базы данных в корпоративной сети вузов

существовали задолго до образования филиалов, то все идентификаторы уже определены и смена их невозможна в виду того, что они используются в различных задачах (например, уникальный идентификатор студента наносится на индивидуальную пластиковую карту в виде штрих-кода).

3. Использование дополнительного идентификатора для определения базы данных. Подход требует изменения процедур идентификации уже существующих данных, что не всегда возможно.
4. Изменение кода в базе данных филиала в соответствии с кодами головного вуза. В каждой базе ведется нумерация в соответствии с общими правилами. Но при репликации данных идентификационные поля в базе данных источника репликации заменяются на те, которые присвоились этим записям в приемнике. Подход опасен тем, что при задержке в репликации возможно использования неизмененных кодов в других задачах. Например, на основании только что введенных данных по студенту ему распечатывают идентификационную пластиковую карту с идентификационным кодом, который еще не был заменен.
5. Разновидность 4-го пункта – сохранение в приемнике репликации соответствия идентификационных данных источника и приемника. Поля таблицы соответствия заполняются после выполнения репликации.

Предпочтительными для КИС являются два подхода: первый и последний. Первый подход позволяет определить для каждого источника репликации свой диапазон идентификаторов для тех случаев, где не используется ограничения по автоувеличению для идентификатора записи. Для случаев с автоувеличением

идентификаторов используется пятый подход с введением в таблицы дополнительных полей.

Рассмотрим, как решается проблема уникального ключа для автоматической репликации на базе онтологического подхода. Пусть существует базовый класс Студент X, который имеет отношения проекции с источником данных A в базе данных головного вуза, при этом доступ определен для чтения $P(X, A \mid 2)$. Студент головного вуза, как производный от класса Студент имеет отношение проекции с тем же источником A, но с доступом для чтения/записи $P'(X, Y), P(Y, A \mid 1)$. Студент филиала, как производный от класса Студент, имеет отношения проекции с источником B, расположенным в базе филиала и с доступом чтения/записи $P'(X, Z), P(Z, B \mid 1)$. На основании описания этих классов и правила 1 формируется репликация, которая обеспечивает копирование студентов филиалов в источник данных, расположенный в головном вузе A $P''(B, A)$.

При автоматическом создании репликации используется следующий подход. В первую очередь анализируются ограничения наложенные на уникальные идентификаторы в классов X, Y, Z. Ограничения на атрибуты производных классов могут отличаться от ограничений на атрибуты базовых классов. Если среди ограничений присутствуют ограничения по диапазону, то выполняется проверка на пересечение диапазонов идентификаторов Y, Z. Если диапазоны не пересекаются, то используется первый подход, если диапазоны в ограничениях не определены или диапазоны пересекаются, то используется пятый подход, который требуется наличия соответствия

между экземплярами классов, имеющих проекцию на разные источники данных.

Первый подход не требует никаких дополнительных описаний, так как при репликации никаких изменений в объектах не происходит. Пятый подход несколько сложнее. При определении отношений проекция для одного и того же класса (или классов в одной иерархии) на различные источники данных требуется определение источника для проекции соответствия между экземплярами классов. Источники, отвечающие за соответствия, определяются либо автоматически (т.е. выполняется их генерация в виде специализированных таблиц) и располагаются они в обеих база данных, либо некоторые, уже существующие таблицы определяются как источники соответствий.

Проблема целостности данных

Проблема возникает в связи с возможным нарушением целостности данных при репликации. Например, в источнике данных A_X ранее существовал экземпляр класса X, который удалили в связи с тем, что не использовали в настоящем и не собираются использовать в будущем. Но в базе филиала C_Y могли внести информацию, связанную с этим экземпляром (т.е. существуют отношения $P_1(Y,X), P(Y,C_Y)$). При попытке реплицировать обновленный справочник $P''(A_X, C_X)$ произойдет ошибка, и репликация не будет выполнена. Ошибка возникает в связи с необходимостью удалить экземпляр класса X в C_X, при наличии экземпляра Y в C_Y, имеющего отношения $P_1(Y,X)$.

Решений этой проблемы может быть несколько.

1. Удалить экземпляры C_Y, соответствующие тем, которые затем удалить из C_X.
2. Реплицировать только новые экземпляры X в C_X и не удалять те, которые используются в C_Y.
3. Полностью заменить C_X на A_X и заменить связи с несуществующими экземплярами на некоторые заранее определенные связи (это может быть null или определенный экземпляр из A_X) с уведомлением администратора.
4. Откатить репликации с уведомлением администратор о причине.

Первое решение нельзя признать корректным, так как может быть удалена нужная информация. Второе решение так же нельзя признать корректным в связи с тем, что реплики перестают быть синхронными и это влечет за собой дальнейшие ошибки при репликации оперативных данных.

Наилучшим решением является 3-ье решение, если его можно выполнить автоматически. Для поддержки автоматической замены используется значения по умолчанию для атрибутов классов.

Пусть описан класс Y, имеющий отношение с классом X $P_1(Y, X)$. При этом один из атрибутов класса Y $Y \to c$, отвечающий за эту связь, принимает значения из области атрибута $D = \{d_i\}_{i=1}^{N}$ класса X. Необходимо определить, что атрибут $Y \to c$ в случае выхода из области D принимает определенное значение из D, например, d_1, т.е. $if\ \exists i\ y_i \to c \notin D \Rightarrow y_i \to c_i = d_1$. Для такого определения используется механизм описания ограничений на атрибуты – установки по умолчанию. В установках по умолчанию

могут использоваться любые регулярные выражения, включающие константы (в том числе null) и атрибуты этого или другого класса.

В случае, когда выполнить замену нельзя (отсутствуют или значения по умолчанию, или составляющие регулярного выражения, определяющего значение по умолчанию), то уведомление об ошибке репликации с указанием причины отправляется администратору и выполняется откат репликации.

Заключение

Онтологический подход требует описания классов КИС и отношений между ними. Одним из важных преимуществ применения онтологий является то, что эти описания не ограничиваются одной задачей, например, репликацией данных. Одни и те же описания используются для реализации интеграции на лету, организации хранилища данных, для извлечения экземпляров классов, для обеспечения качества данных в КИС, для реализации ограничений доступа к ресурсам, для реализации управления средним слоем и для обеспечения безопасности.

В рамках рассматриваемой в работе задачи использование онтологического подхода позволяет не только автоматически генерировать репликации, но и легко их корректировать, при появлении новых классов и источников данных, изменении существующих классов и источников, не меняя при этом кода выполнения репликации.

Литература

1. В.А. Виттих, Д.В. Волхонцев, А.Н. Гинзбург, М.А.Караваев, П.О.Скобелев, О.Л.Сурнин, М.А.Шамашов. Распределенные онтологии и их применение в решении задач интеграции данных // http://www.kg.ru/support/library/dataintegration/

2. Бездушный А.А., Бездушный А.Н., Нестеренко А.К., Серебряков В.А., Сысоев Т.М. Интеграция распределенных данных на основе технологий Semantic Web и рабочих

процессов. //Сборник докладов Шестой Всероссийской конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции», Санкт-Петербург, 2004

3. Клещев А.С., Шалфеева Е.А. Классификация свойств онтологий. Онтологии и их классификации. Препринт, Владивосток, ИАПУ ДВО РАН, 2005, 19с.

4. OWL Web Ontology Language. Overview// http://www.w3.org/TR/owl-features/

5. Stojanovic L., Schneider J., Maedche A., Libischer S., Studer R., Lumpp Th., Abecker A., Breiter G., Dinger J.. The role of ontologies in autonomic computing systems//IBM Research Journal. - 2004.-vol.43.-№4.

6. Крюков В.В., Шахгельдян К.И. Корпоративная информационная среда вуза: методология, модели, решения. Монография//Дальнаука.- 2007. - 308 с.