Министерство образования Украины

Запорожский государственный университет

Факультет математики
и экономической кибернетики
Кафедра ММ и ИТ

ДИПЛОМНАЯ РАБОТА

на тему: Разработка концептуальной модели серверной части информационной интегрированной системы “Студент”

Выполнил Филобок Артур Петрович

ст. Группы 8222-1
/шифр/ /подпись и дата/ /ф., и., о./

Руководитель доцент кафедры ММ и ИТ Ермолаев В. А.
/должность/ /подпись и дата/ /ф., и., о./

Нормоконтролер Бакурова А. В.
/подпись/ /ф., и., о./

Запорожье

РЕФЕРАТ

Описание текста дипломной работы.

Данный текст содержит страниц, глав, рисунков, таблицы, терминов, источника, приложение.

Объектом исследования дипломной работы является архитектура и принципы построения корпоративных систем.

Цель данной дипломной работы состоит в выборе оптимальной архитектуры для создания корпоративных информационных систем, разработки концептуальной модели данных серверной части информационной интегрированной системы “Студент”.

В результате дипломной работы была создана концептуальная модель серверной части интегрированной информационной системы “Студент”, позволяющая построить на ее основе приложение обслуживающее работу ВУЗа. Данная концептуальная модель использует подход федеративных баз данных и позволяет использовать уже работающие приложения.

КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ, ФЕДЕРАТИВНЫЕ СИСТЕМЫ БАЗ ДАННЫХ, ОБЩАЯ ИНТЕГРАЦИОННАЯ МОДЕЛЬ, ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ, СХЕМА ТРАНСФОРМАЦИИ, СХЕМА ИНТЕГРАЦИИ, КЛЮЧ, ДОМЕН, АТРИБУТ.

СОДЕРЖАНИЕ

Введение

Во многих больших организациях функционируют различного рода системы обработки данных. Эти системы обработки данных и сами данные развиваются отдельно друг от друга. Системы обработки данных, а вследствие чего и системы управления базами данных отличаются по нескольким аспектам, таким как модель данных, язык запросов, архитектура систем, а также структурой и семантикой моделей данных. Как правило, приложения в силу определенных проблем остаются основанными на прежних системах. Другим приложениям требуется доступ к разработанным данным, но эти попытки обычно безуспешны из-за гетерогенности вышеупомянутых данных. Для решения этих проблем был разработан подход так называемых федеративных баз данных[SL90, LMR90, BHP92, PBE95]. Федеративные базы данных содержат несколько схем, которые формируют архитектурную схему федеративных баз данных как показано в [SL90]:

1.Локальная схема: локальная схема это концептуальная схема компонентной системы баз данных, которая существует в локальной модели данных этой компонентной системы баз данных;

2. Компонентная схема: компонентная схема это локальная схема трансформированная в общую модель данных федеративного типа;

3. Экспортная схема: экспортная схема отличается от компонентной и определяет интерфейс к локальным данным, которой позволяет получить федеративность;

4. Федеративная схема: федеративная (или интеграционная) схема это результат интеграции множества экспортных схем, тем самым образуя унифицированный интерфейс для глобального приложения;

5. Внешняя схема: внешняя схема это специальный вид федеративной схемы. Он может быть основан на специфической модели данных, отличающейся от общей модели данных. Например, внешняя схема резервируется как специфический интерфейс для глобального приложения.

Общая модель данных варьируется от семантически бедной модели данных, т.е. реляционной модели данных, до семантически богатых моделей данных, т.е. объектно-ориентированной модели данных. Но только семантически богатая модель данных может быть общей моделью данных, потому что, не теряется или почти не теряется семантика в течении трансформации из объектно-ориентированной модели данных в общую. С другой стороны семантически богатая модель данных оказывается более гетерогенной на уровне схемы, чем может токовой оказаться семантически бедная модель данных. Эта гетерогенность на уровне схемы осложняет процесс интеграции. Поэтому будет использоваться семантически бедная модель данных. Для облегчения процесса интеграции была разработана модель данных называемая общая интеграционная модель GIM (представленная в [Sch95]).

Из-за семантических свойств, GIM не может использоваться как интерфейс приложений. Поэтому внешняя схема должна быть экспортирована в отличную от GIM модель данных и должна являться результатом интегрированной GIM-схемы. Процесс перехода от семантически бедной модели данных к объектно-ориентированной модели данных описана в [BKNW91].

GIM это графическое средство, не относящееся к федеративным базам данных, с помощью которого можно провести процесс интеграции в графическом виде. GIM-схема представляется в виде диаграмм.

В Запорожском государственном университете разработаны и эксплуатируются следующие информационные системы:

- "АРМ технического секретаря приемной комиссии";
- "АРМ ответственного секретаря приемной комиссии";
- "Бухгалтерия".

Целью данной дипломной работы является разработка модели данных, которая бы включала в себя модели данных уже разработанных и используемых приложений, а также дополнительные данные о студентах, их успеваемости, медицинские сведения, данные профкома и т.д.. Построенная на базе подхода к созданию федеративных систем, данная модель учитывает корпоративную целостность данных, является логическим интерфейсом к ранее разработанным данным, и тем не менее остается прозрачной для уже функционирующих приложений.

1 Обзор современных методов и архитектур для создания корпоративных информационных систем

1.1 Модель вычислений с использованием большой или мини-ЭВМ

Сначала вернемся назад, в те времена, когда для вычислений в организациях применялись централизованные *хост-компьютеры* (то есть большая ЭВМ или мини-ЭВМ организации). При такой схеме пользователь мог поработать с приложением на большой машине, просто придвинув стул к подключенному к машине *неинтеллектуальному терминалу.* Терминал назывался неинтеллектуальным, потому что он не обладал никакими вычислительными возможностями, и просто передавал и выводил на экран информацию, посылаемую ему машиной.

Заметим, во-первых, что приложение на большой ЭВМ — это единый компонент, отвечающий за взаимодействие с пользователем и управление данными в многопользовательской среде. Довольно быстро стало очевидным, что такая стратегия разработки приложений неэффективна, поскольку для каждого приложения разработчикам приходилось создавать один и тот же компонент управления данными. Таким образом, приложения на больших и мини-ЭВМ эволюционировали и были разделены на две части: *внешний интерфейс,* отвечающий за взаимодействие с пользователем, и *внутренний компонент,* отвечающий за управление данными. Этот внутренний компонент, система управления базами данных (СУБД), представляет собой центральный модуль, используемый с каждым новым интерфейсным приложением. Так как множество внешних компонентов смогли получить доступ к базе данных, управляемой одним внутренним модулем СУБД, разделение приложения на внешний и

внутренний компоненты обеспечило системам большую гибкость.

Как и все прочее, модель вычислений на базе хост-машины имеет свои преимущества и недостатки. Плюсом является централизация в большой ЭВМ. Таким образом, системные администраторы могут надежно управлять одной машиной и обеспечивать доступность данных, когда они требуются пользователям, а также для защиты архивировать их. централизованные системы позволяют совместно использовать периферию, диски, принтеры и модемы. Однако такая модель имеет и многие отрицательные качества. Например, чем большему числу сотрудников необходим доступ к большой ЭВМ, тем большая вычислительная мощность требуется для обслуживания потребностей организации. Сложилось так, что индустрию больших и мини-ЭВМ контролировали лишь несколько компаний. Это означало, что соответствующие ОС, процессоры, приложения и память на диске, необходимые для обслуживания организации, стоили очень дорого. Чем больше вам было нужно, тем больше приходилось платить.

1.2 Модель с автономными персональными вычислениями

В 80-е годы произошло то, что навсегда изменило характер вычислений в организации: появились персональные компьютеры и рабочие станции. С тех пор как IBM создала PC и ОС DOS, Apple — Macintosh, а позднее появились рабочие станции UNIX таких компаний как Hewlett-Packard и Sun Microsystems, независимые друг от друга рабочие станции быстро стали доминировать в организациях, положив конец централизованному контролю над данными компании больших машин. Такую популярность персональные рабочие станции приобрели благодаря тому, что они имеют над большими ЭВМ несколько преимуществ:

- Персональные рабочие станции — это недорогие и простые в

использовании компьютеры, предоставляющие вычислительные возможности и производительность, сопоставимые с дорогими большими машинами.

- Пользователь может сделать компьютерную рабочую станцию "персональной", выбрав тип рабочей станции, ОС и приложения, лучше отвечающие его потребностям.

- ПК-приложения (например, текстовые процессоры, электронные таблицы, графические программы и СУБД) предлагаются в большом ассортименте и обычно очень недороги и вполне доступны для покупки. Пользователь, которому не удается найти отвечающее его потребностям приложение, может с помощью простого в использовании средства разработки создать собственное.

- Данные рабочей станции представляют собой автономный массив информации, который также персонален. Каждая рабочая станция сама отвечает за управление данными, их архивацию и защиту. Отдельные пользователи сами управляют своими ПК, не прибегая к дорогостоящим услугам инженеров вычислительного центра.

К сожалению, переход к независимым персональным вычислениям, по сравнению с централизованными вычислениями на больших ЭВМ, не только дал преимущества, но и породил проблемы. Больше всего бросается в глаза то, что информация предприятия, централизованная и доступная на большой ЭВМ всем сотрудникам, становится распределенной между большой машиной и персональными рабочими станциями. Таким образом, выигрыш в отношении "цена/производительность" и в простоте использования, который дают персональные вычисления, легко может быть сведен на нет потерей продуктивности труда коллективов, которым необходим доступ к

распределенной по предприятию информации. Кроме невозможности совместной работы с данными, пользователи несвязанных персональных рабочих станций не могут совместно работать с другими дорогими ресурсами, доступными пользователям большой ЭВМ — дисками, принтерами, модемами и прочими периферийными устройствами.

1.3 Модель вычислений с сетью и файловым сервером

Проблемы совместного использования данных и периферийных устройств персональных компьютеров и рабочих станций быстро породили *модель вычислений* с *сетью и файловым сервером.* Фактически, если сегодня вы используете на работе персональный компьютер или рабочую станцию, то скорее всего ваш компьютер подключен к *локальной вычислительной сети* (LAN). Локальная сеть дает преимущества коллективных вычислений, сохраняя простоту использования ПК, но позволяя совместно использовать данные и периферию, как в системах с большой ЭВМ.

Для совместного использования данных в сети работающие в ней хранят файлы на *файловом сервере.* Файловый сервер — центральный *узел* (компьютер в сети), который хранит файлы данных, доступные всем пользователям. Обычно файловый сервер в локальной сети является также центральным концентратором для совместного использования периферийных устройств, таких как принтеры, очереди печати и модемы. Так как файловый сервер является независимым компьютером сети, то лучше специализировать его для выполняемых им функций, установив большой объем дисковой памяти.

В локальной сети функционирующее на рабочей станции приложение считывает и записывает файлы, обмениваясь ими с сетевым файловым сервером.

Во многих случаях по сети для выполнения операций на ее локальных
ПК файлы передаются целиком. Файловый сервер не принимает участия в обработке приложения. Он просто хранит файлы для выполняемых на ПК программ. Например, на ПК локальной сети у вас может работать персональный администратор базы данных.

Сначала вы запускаете персональный администратор базы данных, а затем запрашиваете информацию в файле на файловом сервере. Сервер посылает весь файл данных или его часть, передавая его по сети на вашу рабочую станцию. В работе персонального администратора базы данных и самой базы сервер не участвует. При сохранении файла вы копируете данные по сети обратно на файловый сервер.

К сожалению, характеристики модели вычислений с сетью и файловым сервером не позволяют ей адекватно обслуживать требующие высокой производительности многопользовательские приложения с разделяемыми данными, которые легко поддерживают большие ЭВМ. Системы с файловым сервером имеют два недостатка, не позволяющие им обслуживать многопользовательские приложения. Во-первых, модель с файловым сервером не обеспечивает необходимой многопользовательским приложениям *согласованности данных* (одновременного доступа к одному набору данных множества пользователей). Это связано с тем, что файловый сервер работает с файлами — очень большими наборами данных, и не позволяет пользователю обращаться к нему совместно с другими, поскольку файл блокируется. Короче говоря, пользователи, работающие с одними и теми же данными, обычно мешают друг другу и вынуждены ждать доступа к файлу. К тому же, если множество файлов запрашивают и передают по сети сразу много рабочих станций, то сеть быстро насыщается, и трафик становится узким местом, ухудшая производительность системы.

1.4 Модель вычислений клиент/сервер

Присущие локальной сети проблемы породили *модель вычисления клиент/сервер.* Вычисления клиент/сервер (которые называют также *распределенными вычислениями или кооперативной обработкой приложения)* дают преимущества модели сетевых вычислений с доступом к совместно используемым данным и высокие характеристики производительности, присущие модели вычислений с хост-машиной.

Системы клиент/сервер имеют три различных компонента, каждый из которых выполняет конкретную работу: сервер базы данных, клиентное приложение и сеть. *Сервер* ("внутренний компонент") эффективно управляет ресурсом (таким как информационная база данных). Основной функцией сервера является оптимальное управление ресурсом для множества клиентов, которые одновременно у него этот ресурс запрашивают. Серверы баз данных выполняют такие задачи, как:

- Управление одной информационной базой данных, с которой совместно работают множество пользователей.

- Управление доступом к базе данных и другими требованиями защиты.

- Защита информации в базе данных с помощью средств архивации/восстановления и создания резервных копий.

- Централизованное задание для всех клиентных приложений правил глобальной целостности данных.

Клиентное приложение ("внешний интерфейс") — это часть системы, которую пользователь использует для взаимодействия с данными. Клиентные

приложения в СУБД клиент сервер выполняют следующие задачи:

- Представление интерфейса, с помощью которого пользователь может выполнять свою работу.

- Управление логикой приложения, например, всплывающими списками в форме ввода данных или столбчатыми диаграммами в графическом представлении данных.

- Выполнение логики приложения, например, вычисление полей в форме ввода данных.

- Проверка допустимости данных.

- Запрос и получение информации о сервере базы данных.

Наконец, средствами передачи данных между клиентом и сервером в системе являются *сеть и коммуникационное программное обеспечение,* имеющееся у клиента и на сервере и позволяющее им взаимодействовать через сеть.

Поскольку клиентное приложение и сервер базы данных работают совместно и распределяют загрузку приложения (отсюда и термин "распределенная обработка приложения"), система клиент сервер может обеспечить лучшую производительность, чем система с файловым сервером. Сервер управляет для нескольких клиентов базой данных, а клиенты посылают, получают и анализируют полученные с сервера данные. В приложении клиент/сервер клиентное приложение работает с небольшими специальными наборами данных, например, строками таблицы, а не с целыми файлами, как в системе с файловым сервером. Сервер базы данных здесь является

интеллектуальным. Он блокирует и возвращает строки по запросам клиентов, что обеспечивает параллельность, минимальный сетевой трафик и улучшенную производительность системы.

Некоторые преимущества модели клиент/сервер определяются тем фактом, что клиентная и серверная часть системы работают обычно на разных компьютерах. Во-первых, каждый компьютер в системе можно выбрать таким образом, чтобы он лучше всего отвечал требованиям каждого компонента. Например, для сервера базы данных лучше использовать компьютер с мощным процессором (или процессорами), большим объемом ОЗУ и памяти на дисках. Благодаря этому, такой сервер сможет хранить большие объемы данных и адекватно обрабатывать множество одновременных запросов клиентов. Для выполнения же клиентного приложения лучше использовать менее дорогой компьютер с минимальной памятью на диске и оперативной памятью, мышью и хорошими графическими возможностями. Таким образом, организация может при минимальных затратах предоставить пользователям простое в применении инструментальное средство для ввода и анализа данных.

Во-вторых, такая система обладает хорошей адаптируемостью и гибкостью в случае неизбежных изменений в программном и аппаратном обеспечении. Предположим, например, что появился новый тип компьютера, дающего при вдвое меньшей цене удвоенную по сравнению с имеющимся сервером производительность. В системах клиент/сервер легко заменить старый сервер на новый, не нарушая функциональности клиентных приложений и продуктивности работы пользователей.

В-третьих, легко масштабировать систему, приспособив ее к изменениям в рабочей группе. Например, если в отделе появляются новые сотрудники, их можно с помощью новых клиентных рабочих станций сразу подключить к сетевой системе.

Другим преимуществом системы клиент сервер является то, что каждый функциональный компонент системы можно специализировать для наилучшего выполнения тех или иных операций. Например, для разработки клиентного приложения программист сосредотачивает свои усилия на представлении и анализе данных. Тем временем управлением данными занимается сервер базы данных. Таким образом , разработчику при создании нового приложения не нужно каждый раз проектировать код СУБД.

Однако вычисления клиент/сервер имеют и присущие им недостатки. Во-первых, ожидаемую экономию затрат реально можно получить не всегда. При проектировании стоимости компьютерной системы следует учитывать множество факторов, а не только затраты на аппаратуру. Например, при оценке затрат важными показателями является продуктивность пользователей, включая пользователей приложения, разработчиков и администраторов. Разработчики могут улучшить продуктивность благодаря доступным в системах СУБД клиент/сервер GUI и инструментальным средствам *автоматизированной разработки программного обеспечения* (CASE). Однако пользователи и администраторы могут фактически столкнуться со снижением производительности. Почему? Системе клиент/сервер, представляющей собой сочетание независимо разработанных различными производителями и управляемых аппаратных и программных компонентов, присуща меньшая надежность, чем однородным и централизованно управляемым большим или мини-ЭВМ. К кому вы обратитесь за технической поддержкой, если что-то не так? Неработоспособность из-за ненадежности системы снижает продуктивность работы пользователей и администраторов.

Ключевым фактором в оценке экономии затрат является выбор для работы в системе клиент/сервер приложения конкретного типа. Например, выполнять и управлять в системе клиент/сервер крупной системой заказа

авиабилетов с учетом того, что она имеет сотни и тысячи терминалов и распределенные по всему миру узлы, нереально. Но она подойдет для выполнения локализированных приложений бухгалтерского учета и производственных задач подразделения предприятия. Вычисления клиент/сервер составляют очень важную часть общей информационной стратегии предприятия, но их нельзя считать верным выбором для каждого приложения.

1.5 Модель федеративных баз данных

Рассматривая сложившуюся ситуацию в различных организациях нетрудно заметить, что различные системы управления базами данных используются различными приложениями в разных подразделениях организаций. Так, каждое подразделение обычно имеет свою точку зрения на использование баз данных и выбор СУБД. Таким образом, различное число отличающихся систем баз данных в различных подразделениях создавались и развивались отдельно друг от друга.

Интегрирование отдельно существующих баз данных в одну логическую базу данных это многообещающий путь, который сможет помочь предприятию остаться работоспособным при всё увеличивающим темп прогрессе. Концепция федеративных[HM85, SL90] баз данных состоит в том чтобы интегрировать отдельно существующие базы данных в унифицированное представление данных на одном глобальном уровне. Федеративные системы баз данных это системы состоящие из набора совместных, но (частично) автономных и возможно гетерогенных компонентных систем баз данных. Основная идея федеративных систем баз данных состоит в создании унифицированного интерфейса для гетерогенных исходных данных отдельно развитых систем. Так, новое (глобальное) приложение имеет возможность доступа к федеративным данным и существующим (локальным) операциям приложений, при том что специфические компонентные системы баз данных могут оставаться неизмененными.

Выводы

В данной главе были рассмотрены различные методы построения корпоративных систем баз данных. Сравнительный анализ архитектур для создания корпоративных систем показал, что при построении интегрированной информационной системы наиболее выгодно использовать федеративную методику проектирования моделей данных. Следующая глава будет посвящена рассмотрению базовых понятий проектирования федеративных систем, а также вспомогательных средств, используемых при разработке федеративной системы.

2 Метод проектирования моделей данных федеративных информационных систем

2.1 Базовые понятия федеративных информационных систем

При разработке федеративной модели данных “СТУДЕНТ” будем придерживаться общепринятой пятиуровневой архитектурной схемы для федеративных систем баз данных, которая содержит следующие уровни/компоненты:

1. Локальная схема: локальная схема это концептуальная схема компонентной системы баз данных, которая существует в локальной модели данных этой компонентной системы баз данных;

2. Компонентная схема: компонентная схема это локальная схема трансформированная в общую модель данных федеративного типа;

3. Экспортная схема: экспортная схема отличается от компонентной и определяет интерфейс к локальным данным, которой позволяет получить федеративность;

4. Федеративная схема: федеративная (или интеграционная) схема это результат интеграции множества экспортных схем, тем самым образуя унифицированный интерфейс для глобального приложения;

5. Внешняя схема: внешняя схема это специальный вид федеративной схемы. Он может быть основан на специфической модели данных, отличающейся

от общей модели данных. Например, внешняя схема резервируется как специфический интерфейс для глобального приложения.

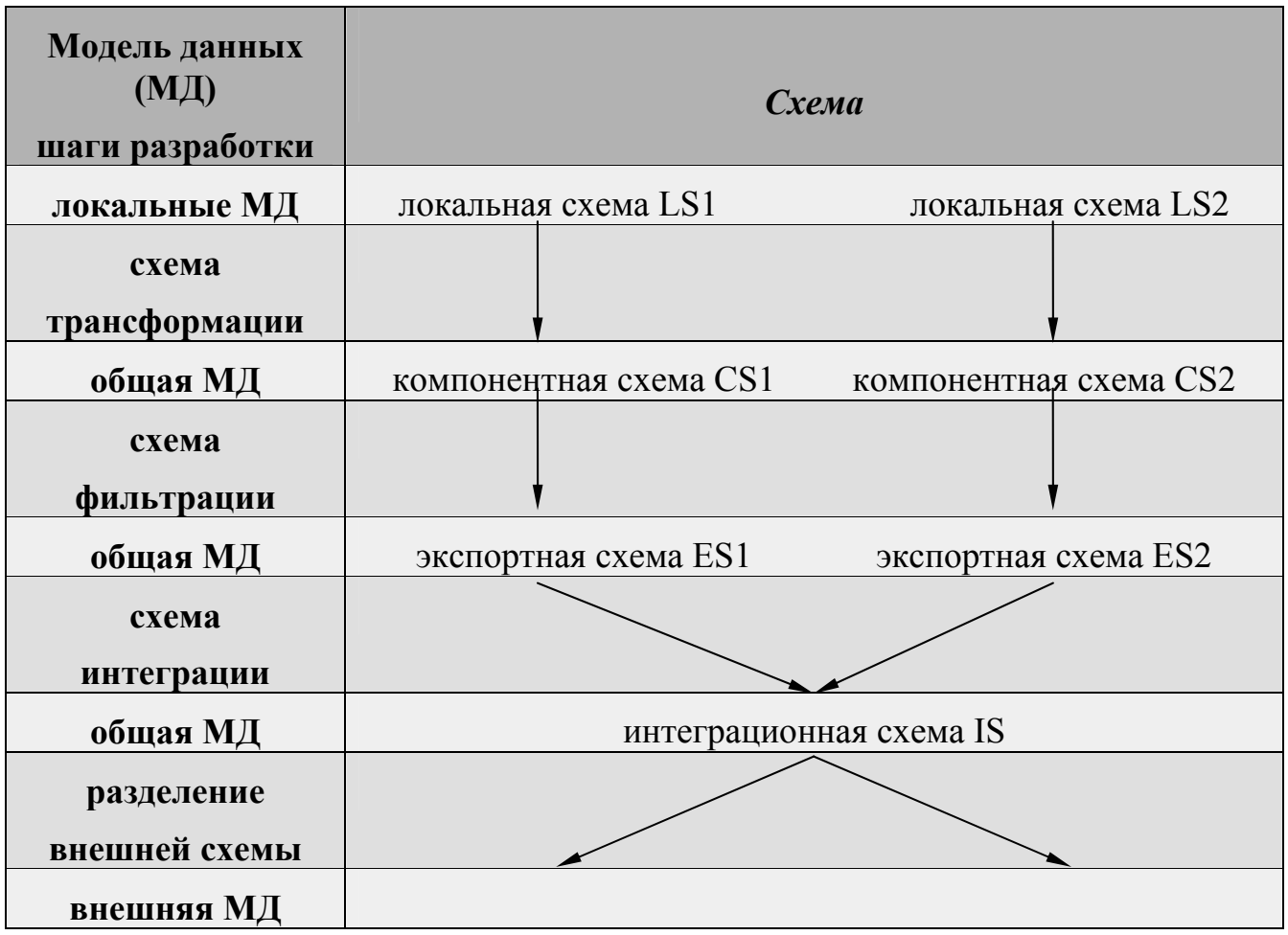


Рисунок 1.1

На рисунке 1.1 вы видите общие компоненты процесса разработки федеративной базы данных. Чрезвычайно важным пунктом в процессе разработки федеративной базы данных является правильный выбор общей модели данных. В [BLN86, SCG91, PBE95], общие модели данных варьируются от “семантически богатых” модели данных (аналогична объектно-ориентированной модели данных) до “семантически бедных” моделей данных (аналогична реляционной модели данных).

Локальные схемы различных (гетерогенных) компонентных систем баз данных является стартовой точкой построения федеративной базы данных. На первом шаге локальная схема, определенная моделью данных компонентной системы баз данных, трансформируется в общую модель данных федеративной базы данных. Процесс трансформации является необходимым для процесса фильтрации, но эти два шага можно поменять местами. Результатом этих двух шагов является экспортная схема, которая может интегрирована в федеративную схему. Схема интеграционного процесса, собственно, очень сложна и разделяется на несколько шагов: предварительное интегрирование, сравнение, разграничение и реструктуризация схем подлежащих интегрированию(более детально описано в [BLN86]). На последнем шаге, внешняя схема, которая определена как специальный вид федеративной системы, разделяется. Как правило внешняя схема должна быть представлена моделью данных отличной от общей модели данных, и по этому дополнительный шаг трансформации обязателен.

2.2 Классификация ограничений целостности

Как было показано выше, ограничения целостности играют важную роль в процессе создания федеративных баз данных. Несколько свойств ограничений целостности могут быть рассмотрены и разделены на категории.

Мы проклассифицируем ограничения целостности в двух ортогональных направлениях: экстенсивном и интенсивном. В экстенсивном направлении, мы определимся между ограничения целостности которые должны соответствовать только одному объекту и которые должны соответствовать множеству объектов. В нашей терминологии мы сформируем один называемый одно-объектные ограничения целостности и много объектные ограничения целостности. В интенсивном пути, ограничения целостности изменятся в количестве (ноль, один и более одного объекта). Есть два за почему мы выбрали эту классификацию. Во-

первых, эта классификация соответствует концепции нашей интеграционной модели. Во-вторых, как мы увидим позже эта классификация позволит нам понять как ограничения целостности могут изменять свои характеристики в течении создания федеративной базы данных. С помощью комбинирования этих двух направлений мы получаем шесть классов ограничений целостности (рис. 2.1).

Ограничения целостности (ОЦ) *Классификация*		*Внешняя размерность*	
		1 объект	m объектов (m>1)
внутренняя размерность	0 атрибутов	IC_{10}	IC_{m0}
	1 атрибут	IC_{11}	IC_{m1}
	n атрибутов	IC_{1n}	Ic_{mn}

Рисунок 2.1

Ниже приведена классификация этих классов:

• IC_{10}: Это ограничение целостности может быть сопоставлена только одному объекту, который не содержит ни одного атрибута этого объекта. Например, такого плана ограничения есть включающие ограничения, так как объект подкласса должен принадлежать своему суперклассу. В этом случае мы должны проверить каждый объект подкласса на принадлежность его к суперклассу.

• IC_{11}: Примером этого класса ограничений может служить ненулевое условие, ограничивающие ряд атрибутов.

• IC_{1n}: Типичный пример данного класса - ограничение целостности которые обеспечивают поддержку между несколькими атрибутами одного объекта.

• IC_{m0}: Также как и IC_{10}, наличие атрибутов на уровне объекта не является обязательным условием валидности данного ограничения. Примером ограничения этого класса может служить условие ограничивающее число экземпляров класса.

• IC_{m1}: Хорошо известное условие уникальности только одного атрибута класса может служить примером ограничения этого типа.

•IC_{mn}: Примером ограничения целостности которое должно быть справедливым на множестве атрибутов множества объектов - это условие уникальности наложенное на совокупность нескольких атрибутов объекта класса.

Для того чтобы закончить приведенную выше классификацию мы опишем некоторые хорошо известные характеристики ограничений целостности.

- Ограничения целостности интра-класса и интер-класса. Ограничения целостности разделяются на ограничения интра-класса и интер-класса[JQ92]. Где ограничения интра-класса относятся к объектам только одного класса, а ограничения интер-класса относятся к объектам различных классов.

- Статические и динамические ограничения целостности. Ограничения целостности обычно делят на статические и динамические. Статические ограничения относятся к возможным значениям атрибутов объектов и

поэтому характеризуют возможные состояния базы данных.

Динамические ограничения характеризуют эволюцию базы данных во времени и могут быть разделены на переходные и временные. Временные ограничения определяют набор возможных состояний базы данных, существование отличных от этих состояний не допускается. Переходные ограничения дают условия перехода базы данных из одного состояния к другому[LGS94].

- Неотъемлемые от модели данных и внешние ограничения целостности. Ограничения целостности также можно разделить на ограничения неотъемлемые от модели данных и ограничения сформулированные из внешних условий.

- Описательные и процедурные ограничения целостности. Ограничения целостности разделяют также по способам их формулирования, они могут быть описательными и процедурными. В первом случае ограничения декларативно описываются на языках высокого уровня таких как, например, SQL. Процедурные же ограничения описываются при программировании приложений или в телах методов (в случае объектно-ориентированных систем).

- Возникающие на определенных шагах создания Ф.Б.Д.

    - локальные ограничения целостности (LIC);
    - трансформационные ограничения целостности (TIC);
    - экспортные ограничения целостности (EIC);
    - интеграционные ограничения целостности (IIC);

- глобальные ограничения целостности (GIC);
- внешние ограничения целостности (XIC).

Для определения ограничений целостности предлагается следующая формальная запись:

<схема><номер>$_{<xIC>}$[атр][exts];<условие>(<характеристика>),

где:

схема - название схемы;

номер - ограничения целостности внутри одной схемы могут быть определены дополнительным идентификационным номером;

xIC - LIC, TIC, EIC, IIC, GIC, XIC;

атр, exts - здесь перечисляются атрибуты и extensions;

условие - условие на значение, домен , cardinality атрибутов или extensions;

характеристика - характеристика ограничения целостности, т.е. класс или динамический аспект.

2.3 Общая интеграционная модель

Общая модель данных варьируется от семантически бедной модели данных, т.е. реляционной модели данных, до семантически богатых моделей данных, т.е. объектно-ориентированной модели данных. Но только семантически

богатая модель данных может быть общей моделью данных, потому что, не теряется или почти не теряется семантика в течении трансформации из объектно-ориентированной модели данных в общую. С другой стороны семантически богатая модель данных оказывается более гетерогенной на уровне схемы, чем может токовой оказаться семантически бедная модель данных. Эта гетерогенность на уровне схемы осложняет процесс интеграции. Данная проблема существует в объектно-ориентированной интеграционной методологии, как описано в [SPD92, RPRG94, PBE95, GSC95]. Поэтому будет использоваться семантически бедная модель данных. Для облегчения процесса интеграции была разработана модель данных называемая общая интеграционная модель GIM (представленная в [Sch95]). Ниже описываются простейшие концепции GIM:

- Простые типы данных: концепция отношений первой нормальной формы адаптирована из реляционных баз данных как класс GIM;

- Класс не перекрывающихся частей: Части любых двух классов должны разделяться и подвергаться идентификации. Подмножества отношений и частично перекрытия между классами не допускаются;

- Идентификаторы объектов: В GIM используется концепция идентификаторов объектов[KC86];

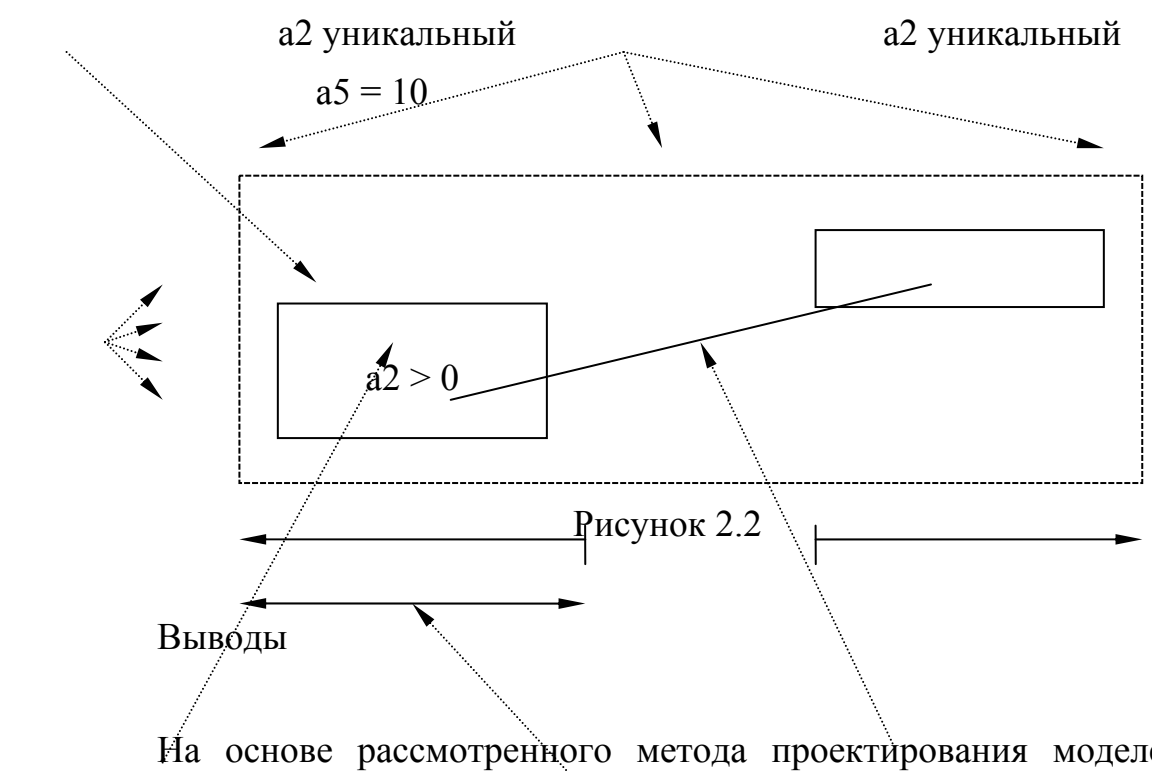
- Двунаправленные двоичные отношения: GIM поддерживает различные типы отношений (1:1, 1:n,m:n) между классами. Все отношения в GIM двунаправленные, т.е. для каждого отношения существует обратное ему отношение.

- Ограничения целостности: В GIM поддерживаются такие же ограничения целостности как и SQL2, т.е. ограничения отношений, статические и

динамические ограничения, ограничения уникальности, и ограничения целостности фиксирующие значения атрибутов всех объектов.

Из-за семантических свойств перечисленных выше концепций, GIM не может использоваться как интерфейс приложений. Поэтому внешняя схема должна быть экспортирована в отличную от GIM модель данных и должна являться результатом интегрированной GIM-схемы. Процесс перехода от семантически бедной модели данных к объектно-ориентированной модели данных описана в [BKNW91].

GIM это графическое средство, не относящееся к федеративным базам данных, с помощью которого можно провести процесс интеграции в графическом виде. GIM-схема представляется в виде диаграмм. Чтобы понять диаграммы вы должны усвоить семантику графических элементов.



На основе рассмотренного метода проектирования моделей данных федеративных информационных систем можно выделить ряд требований накладываемых на разрабатываемую концептуальную модель:

1. Данная концептуальная модель должна содержать информацию о студентах взятую из АРМа ответственного секретаря, “Бухгалтерии”.

2. Построить интегрированную модель для описанных выше подразделений вуза.

3. Исключить в модели семантическое дублирование данных.

Для достижения описанных выше целей необходимо, используя метод проектирования моделей данных федеративных информационных систем произвести построение федеративной модели данных.

Таким образом, формально задачу разработки отмеченной модели данных можно сформулировать следующим образом:

требуется создать модель данных, удовлетворяющую следующим требованиям:

1. Данная модель должна учитывать корпоративную целостность данных.

2. Построенная модель должна является логическим интерфейсом к ранее разработанным данным.

3. Полученная модель должна оставаться прозрачной для уже функционирующих приложений

3 Разработка концептуальной модели корпоративной системы “Студент”

3.1 Описание локальных схем данных

Запишем некоторые из локальных схем данных разработанных приложений на языке высокого уровня SQL. Ниже приведены базы данных АРМа “Бухгалтерия”:

Схема А - Create table student (TN NUMERIC(4,0) primary key, FM CHAR(15), IM CHAR(15), OT CHAR(15), POL CHAR(1), GOD NUMERIC(4,0), ADDRES1 CHAR(15), ADDRES2 CHAR(15), ADDRES3 CHAR(15), TELEFON CHAR(9))

Схема А - Create table univer (TN NUMBER(4,0), UCH_SEZ NUMERIC(5,0), KOL NUMBER(1,0), N_DOG CHAR(10), DATE_Z DATE, VID NUMERIC(1,0), OBUCH NUMERIC(1,0), FAK_N NUMERIC(2,0), SPETS_N NUMERIC(2,0), SEM NUMERIC(2,0), PREDPR_N NUMERIC(2,0), SUM_ALL NUMERIC(10,2), NDS_ALL NUMERIC(10,2), N_PP NUMERIC(7,0), DATE_PP DATE, SUM_OPL NUMERIC(10,2), NDS_OPL NUMERIC(10,2), OTKL NUMERIC(10,2), NDS_OTKL NUMERIC(10,2), NPR_ISKL CHAR(10), DATE_ISKL DATE, FOREIGN KEY (TN) REFERENCES student)

Схема А - Create table lkarta (TN NUMERIC(6,0) PRIMARY KEY, FM CHAR(15), IM CHAR(15), OT CHAR(15), BAL NUMERIC(4,2), FIO CHAR(20), KOD NUMERIC(4,0), NALOG NUMERIC(1,0), PROF NUMERIC(1,0), UVOL NUMERIC(1,0), COMENT CHAR(68), DZAP DATE)

Схема А - Create table pricaz (TN NUMERIC(6,0), VO NUMERIC(3,0), PROCENT NUMERIC(6,4), DZAP DATE, FOREIGN KEY (TN) REFERENCES lkarta)

Теперь приведем основную базу данных “АРМа технического секретаря”:

Схема В - Create table base (FO NUMERIC(1,0), FAK NUMERIC(2,0), SPE NUMERIC(2,0), FAMIL CHAR(30), N_REG NUMERIC(4,0), PLAT CHAR(1), NATION CHAR(1), SEMPOL NUMERIC(1), HARZHIT NUMERIC(1), H_SCHOOL NUMERIC(3,0), MESTOOK NUMERIC(3,0), Y_OKZAV NUMERIC(2,0), UCHZAV NUMERIC(3,0), L_SCHOOL NUMERIC(1,0), L_INOSTR NUMERIC(1,0), LGOT NUMERIC(2,0), NAGRAD NUMERIC(2,0), ARMY NUMERIC(1,0), D_ARMYEND NUMERIC(2,0), ST_TRUDOB NUMERIC(2,0), SPECDOVUZ CHAR(10), DOMADR CAHR(50), D_ZAPOLN DATE, D_ROZHD DATE, SOCPOLOZH NUMERIC(1,0), ST_SPEC NUMERIC(2,0), SEX NUMERIC(1,0), B_EKZ1 NUMERIC(1,0), B_EKZ2 NUMERIC(1,0), B_EKZ3 NUMERIC(1,0), B_ATT1 NUMERIC(3,1), B_ATT2 NUMERIC(3,1), B_ATT3 NUMERIC(3,1), B_ATT4 NUMERIC(3,1), P_ATT1 NUMERIC(2,0), P_ATT2 NUMERIC(2,0), P_ATT3 NUMERIC(2,0), P_ATT4 NUMERIC(2,0), MKEY CHAR(6) PRIMARY KEY, P_EKZ1 NUMERIC(2,0), P_EKZ2 NUMERIC(2,0), P_EKZ3 NUMERIC(2,0), ATTMIDAR NUMERIC(4,2), SCORE NUMERIC(5,2), PRIOR NUMERIC(3,0), PRIOR2 NUMERIC(3,0), SOBES NUMERIC(1,0), PRIORVIRT NUMERIC(3,0))

Концептуальная модель данных интегрированной информационной системы “Студент”, описанная в дипломной работе “Создание концептуальной модели данных клиентской части интегрированной информационной системы “Студент”” не приводится из-за ее громоздкости, а также из-за того, что она сохраняет модель данных системы “Абитуриент”, основанную на ER подходе. В

ходе процесса федерации мы будем ссылаться на модель данных системы “Студент”.

Исходя из описанных выше схем можно получить диаграмму перекрывания атрибутов локальных баз данных, как показано на рисунке 2.3

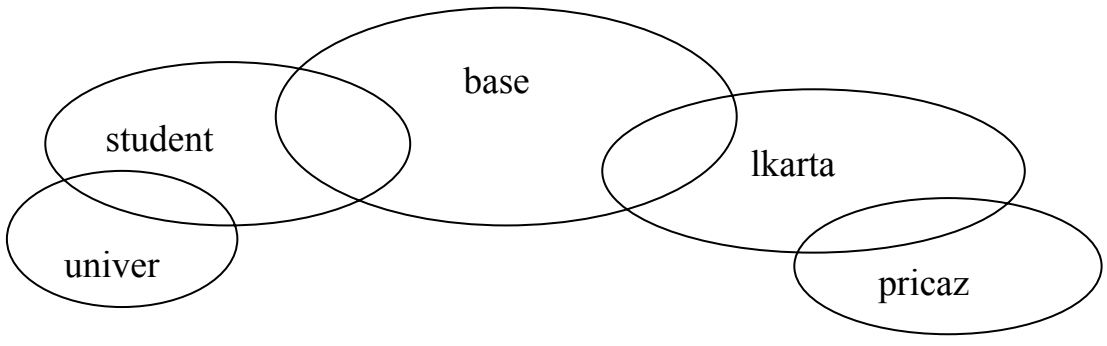


Рисунок 3.1

Мы будем приводить только те ограничения целостности, которые нам будут интересны с точки зрения глобальной целостности данных в перекрывающихся частях отношений.

Вначале рассмотрим ограничения ключей:

$A1_{LIC}$[TN][student]:TN первичный ключ (IC_{m1}, интра-класс, статическое)

$A2_{LIC}$[TN][lkarta]: TN первичный ключ (IC_{m1}, интра-класс, статическое)

$B1_{LIC}$[MKEY][base]: MKEY первичный ключ (IC_{m1}, интра-класс, статическое)

Затем обратим внимание на размерности одинаковых полей различных баз данных с целью предотвращения конфликтов данного типа:

$A3_{LIC}$[FM][student]: FM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A4_{LIC}$[FM][lkarta]: FM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A5_{LIC}$[IM][student]: IM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A6_{LIC}$[IM][lkarta]: IM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A7_{LIC}$[OT][student]: OT $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A8_{LIC}$[OT][lkarta]: OT$\leq$ 15(IC_{1n}, интра-класс, статическое)

$A9_{LIC}$[FIO][lkarta]: Fio $\leq$ 20(IC_{1n}, интра-класс, статическое)

$B2_{LIC}$[FAMIL][base]: FAMIL $\leq$ 15(IC_{1n}, интра-класс, статическое)

3.2 Схема трансформации в общую интеграционную модель

Как было описано ранее локальные схемы должны быть трансформированы в общую модель данных. В качестве общей модели данных была выбрана общая интеграционная модель. Каждая модель данных должна быть трансформирована в общую интеграционную модель отдельно. Т.к. все модели данных подлежащие трансформации основаны на ER подходе, близком по структуре с моделью данных GIM, то собственно в процессе трансформации необходимость отпадает. Поэтому перейдем непосредственно к описанию ограничений целостности.

$A1_{TIC}$[TN][student]:TN уникален (IC_{m1}, интра-класс, статическое)

$A2_{TIC}$[TN][student]:TN не пуст (IC_{11}, интра-класс, статическое)

$A3_{TIC}$[TN][lkarta]: TN первичный ключ (IC_{m1}, интра-класс, статическое)

$A4_{TIC}$[TN][lkarta]: TN не пуст (IC_{11}, интра-класс, статическое)

$B1_{TIC}$[MKEY][base]: MKEY уникален (IC_{m1}, интра-класс, статическое)

$B2_{TIC}$[MKEY][base]: MKEY не пуст (IC_{11}, интра-класс, статическое)

$A5_{TIC}$[FM][student]: FM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A6_{TIC}$[FM][lkarta]: FM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A7_{TIC}$[IM][student]: IM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A8_{TIC}$[IM][lkarta]: IM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A9_{TIC}$[OT][student]: OT $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A10_{TIC}$[OT][lkarta]: OT$\leq$ 15(IC_{1n}, интра-класс, статическое)

$A11_{TIC}$[FIO][lkarta]: Fio $\leq$ 20(IC_{1n}, интра-класс, статическое)

$B3_{TIC}$[FAMIL][base]: FAMIL $\leq$ 30(IC_{1n}, интра-класс, статическое)

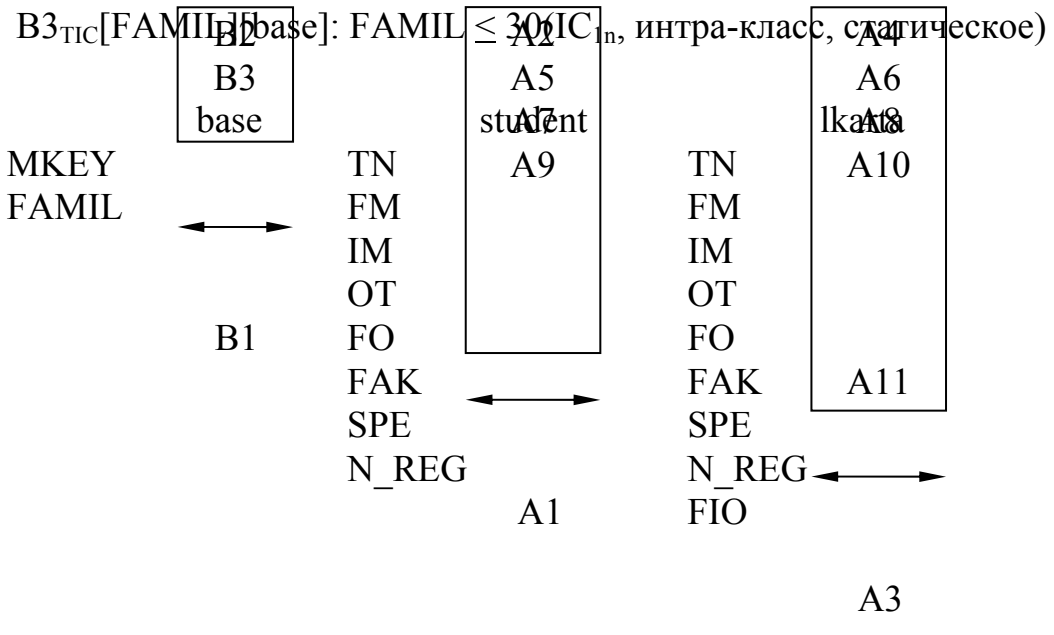


Рисунок 3.2

3.3 Интеграционная схема

На этом этапе локальные схемы уже трансформированы в общую модель данных и мы приступим к процессу интеграции. Сам процесс интеграции очень сложен. На этом шаге возникает несколько типов конфликтов:

1. мета конфликты;

2. конфликты атрибутов;

3. внутренние конфликты;

4. внешние конфликты.

Мета конфликты возникают в том случае, когда объекты одного класса имеют различные значения и при пересечении объектов различных схем отличающихся определением классов.

Однозначная идентификация студента производится по следующим атрибутам: FO FAK SPE и N_REG. Для синхронизации ключей TN и MKEY в базы данных student и lkarta необходимо ввести вышеперечисленные атрибуты.

$A1_{IIC}$[FO,FAK,SPE,N_REG][student]: FO+FAK+SPE+N_REG уникально (IC_{m1}, интра-класс, статическое)

$A1_{IIC}$[FO,FAK,SPE,N_REG][lkarta]: FO+FAK+SPE+N_REG уникально (IC_{m1}, интра-класс, статическое)

$B1_{IIC}$[MKEY][base]: MKEY уникален (IC_{m1}, интра-класс, статическое)

Для обнаружения конфликтов атрибутов необходимо сравнить атрибуты различных схем. Два или более атрибута являются одним атрибутом в интегрированной схеме в том случае если они являются семантическими эквивалентами. В нашем случае все конфликты атрибутов были описаны на шаге трансформации и ограничения целостности могут быть применены здесь.

$A2_{TIC}$[FM][student]: FM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A2_{TIC}$[FM][lkarta]: FM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A3_{TIC}$[IM][student]: IM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A3_{TIC}$[IM][lkarta]: IM $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A4_{TIC}$[OT][student]: OT $\leq$ 15(IC_{1n}, интра-класс, статическое)

$A4_{TIC}$[OT][lkarta]: OT$\leq$ 15(IC_{1n}, интра-класс, статическое)

$A5_{TIC}$[FIO][lkarta]: Fio $\leq$ 20(IC_{1n}, интра-класс, статическое)

$B2_{TIC}$[FAMIL][base]: FAMIL $\leq$ 30(IC_{1n}, интра-класс, статическое)

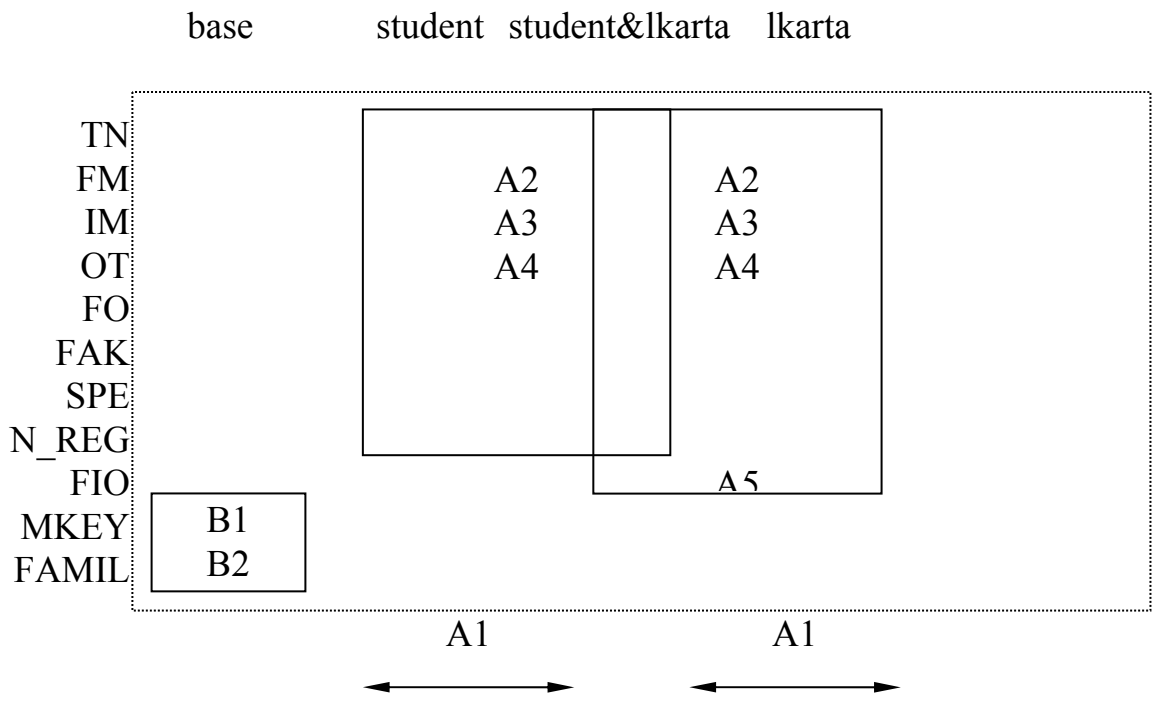


Рисунок 3.3

Для разрешения внутренних конфликтов применяется внутренняя декомпозиция. Внутренняя декомпозиция заключается в группировке атрибутов. Другими словами все атрибуты группы должны быть атрибутами одного класса. Если к классу применяется внутренняя декомпозиция, то расширения данного класса остаются неизмененными, а группы атрибутов подвергаются операции дизъюнкции. Внутренняя декомпозиция не вносит никаких ограничений целостности, потому что они определяется в расширениях классов.

Разрешение внешних конфликтов более сложно, чем разрешение внутренних. Для разрешения внешних конфликтов применяется декомпозиция расширений перекрывающихся классов. Допустим что класс A и класс B перекрываются внешне. Тогда в результате декомпозиции классов A и B мы получим три класса A-B, B-A и A&B, где - и & - логические операции над

множествами. При применении декомпозиции расширений к схеме на рисунке 2.5 мы получим интеграционную схему, изображенную на рисунке 2.6

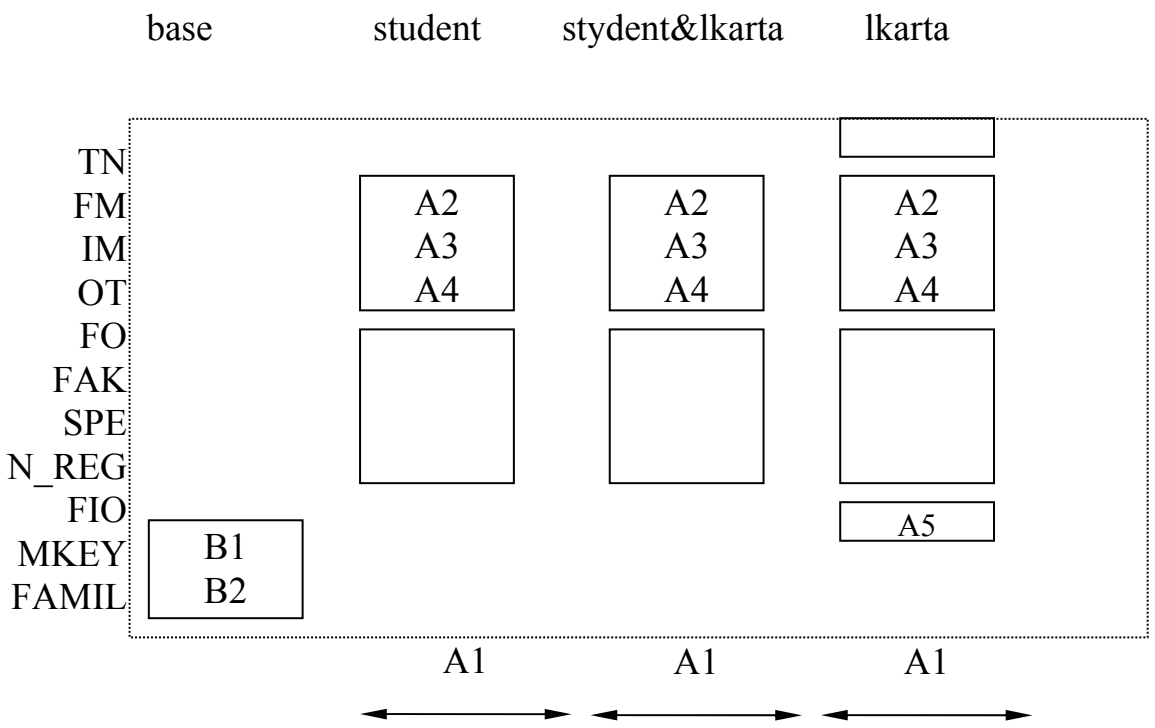


Рисунок 3.4

3.4 Разделение на внешние схемы

Разделение на внешние схемы является другим очень важным процессом при разработке федеративных баз данных. Глобальные приложения нуждаются в специфическом представлении данных, поэтому данный процесс поддерживает различные схемы. Процесс разделения на внешние схемы имеет несколько шагов:

1. разделение внешней схемы, которая эквивалентна локальной схеме должна включать в себя инвертированные операции трансформации и интеграции, примененные к интеграционной схеме;

2. интеграционная схема представляет собой законченное представление федеративных данных. В силу того что GIM не может выступать в роли внешней модели данных, интеграционная схема должна быть трансформирована в другую модель данных.

3. представления могут быть разделены путем исключения некоторых элементов схемы и добавлением новых ограничений целостности.

В дополнение к выше перечисленным задачам, при генерации различно структурированных представлений может быть применена дополнительно операция реструктурирования. Ниже мы будем уделять внимание созданию полной внешней схемы, основанной на объектно-ориентированной модели данных. Таким образом мы опишем как объединить классы GIM. Существует два различных метода композиции, внутренняя и внешняя композиция.

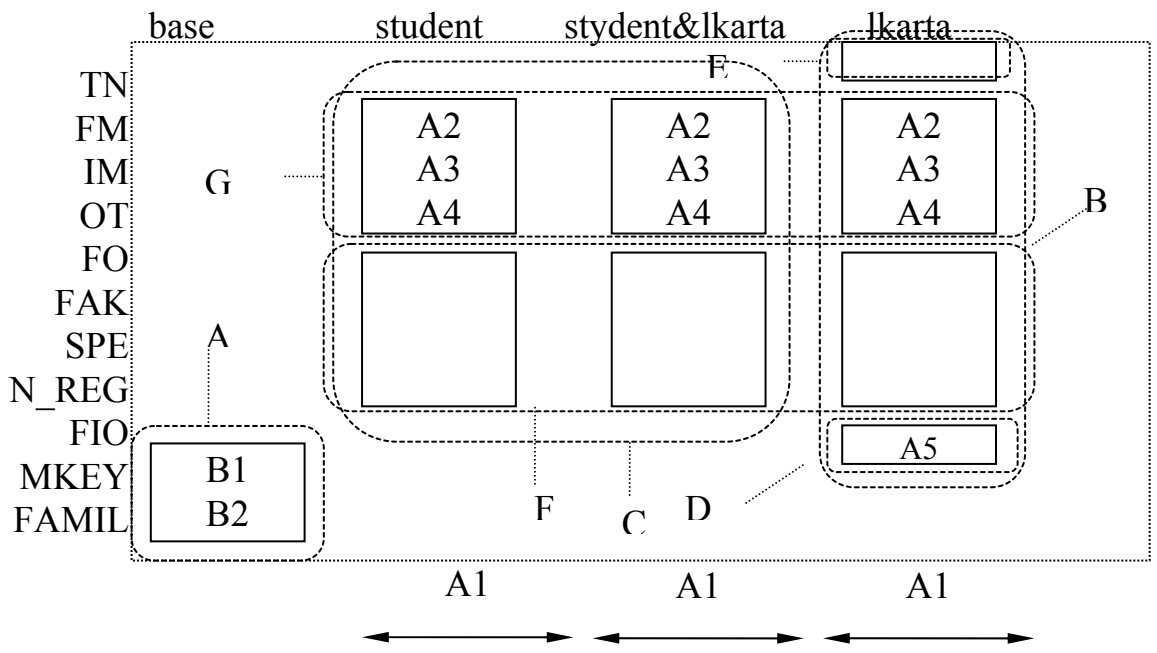
Только классы с одинаковыми расширениями могут быть объединены внутренне. Если определено ограничение целостности интер-класса при описании зависимостей между атрибутами объединяемых классов, то это ограничение целостности становится ограничением интра-класса.

Внешняя декомпозиция более сложна чем внутренняя. Проблемы возникают в случае, если объединяемые классы определяются различными ограничениями целостности. В принципе, существуют два возможных метода композиции различных ограничений целостности:

• Конъюнктивная композиция. Конъюнктивная композиция ограничений целостности состоит в том, что более слабое ограничение отбрасывается. Таким образом не все классы GIM могут быть доступны глобальному приложению;

• Дизъюнктивная композиция. Дизъюнктивная композиция ограничений целостности состоит в том, что более сольное ограничение отбрасывается. Таким образом все классы GIM могут быть доступны глобальному приложению.

Теперь, после описания основных принципов композиции классов GIM



мы приступим к применению данной методологии.

Рисунок 3.5

Классы Внешней Схемы		Внешняя Размерность	Внутренняя Размерность	Ограничения Целостности
A	base	1	10,11	X2,X7
B	lkarta	4	1,2,3,4,5,6,7,8,9	
C	student	2,3	2,3,4,5,6,7,8	X6
D	абстрактный класс	4	9	
E	абстрактный класс	4	1	
F	абстрактный класс	2,3	5,6,7,8	X1
G	абстрактный класс	2,3,4	2,3,4	X3,X4,X5

Рисунок 3.6

$X1_{IIC}$[FO,FAK,SPE,N_REG][F]: FO+FAK+SPE+N_REG уникально (IC_{m1}, интра-класс, статическое)

$X2_{IIC}$[MKEY][base]: MKEY уникален (IC_{m1}, интра-класс, статическое)

$X3_{TIC}$[FM][G]: FM $\leq$ 7(IC_{1n}, интра-класс, статическое)

$X4_{TIC}$[IM][G]: IM $\leq$ 6(IC_{1n}, интра-класс, статическое)

$X5_{TIC}$[OT][G]: OT $\leq$ 7(IC_{1n}, интра-класс, статическое)

$X6_{TIC}$[FIO][D]: Fio $\leq$ 20(IC_{1n}, интра-класс, статическое)

$X7_{TIC}$[FAMIL][base]: FAMIL ≤ 20(IC_{1n}, интра-класс, статическое)

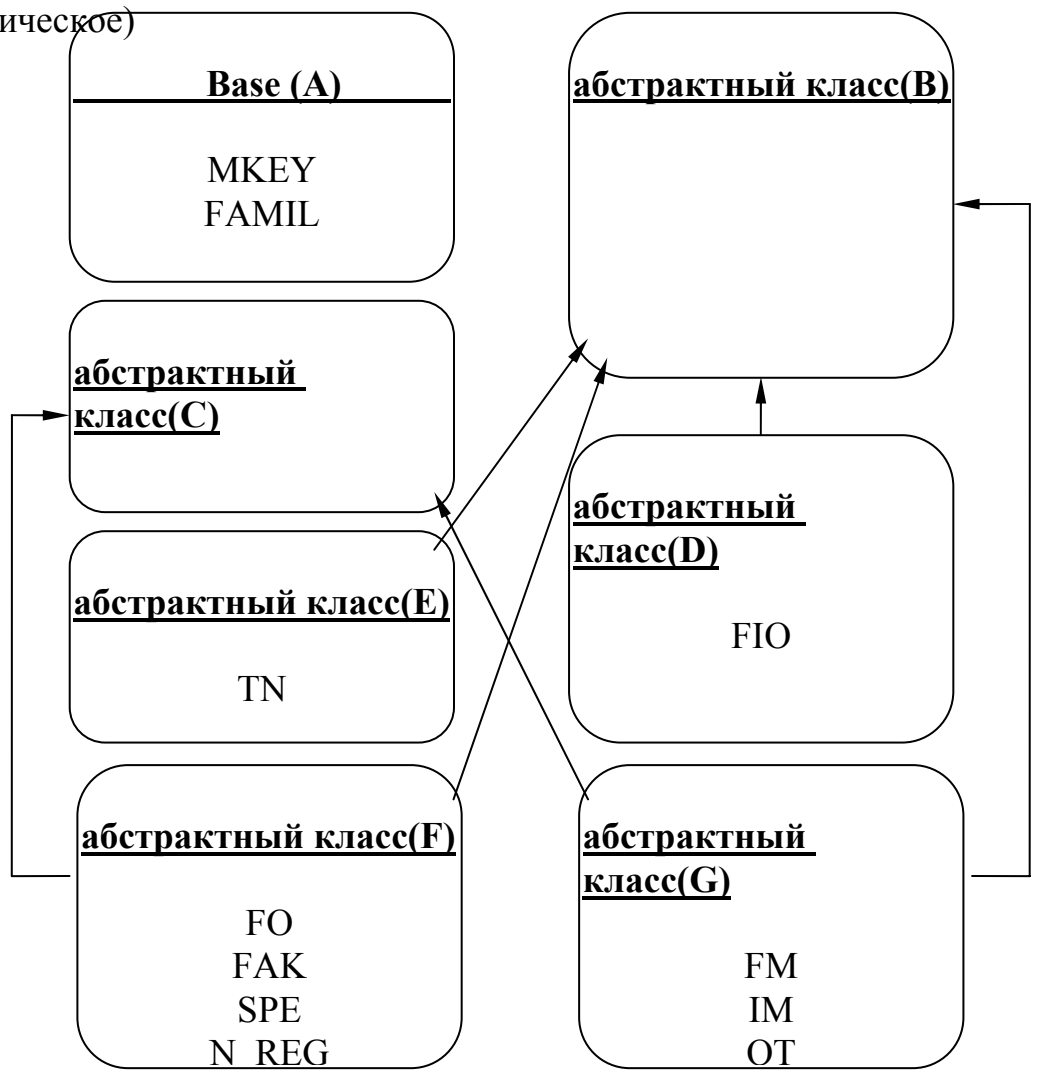


Рисунок 3.7

Как видно из рисунка 3.7 мы получили объектно-ориентированную внешнюю схему. Далее преобразовуя полученную схему из объектно-ориентированной в реляционную и добавляя необходимые атрибуты мы получим федеративную модель данных интегрированной информационной системы “Студент”.

Выводы

В результате построения федеративной модели данных можно сделать следующие выводы:

1.Информационная модель данных системы “Бухгалтерия” имеет дублирующиеся данные. В тоже время эта модель является неполной с точки зрения корпоративной целостности и в данную модель требуется введение дополнительных атрибутов.

2.Построенная федеративная модель данных интегрированной информационной системы “Студент” учитывает выше перечисленные недостатки и на ее базе может быть построено приложение, автоматизирующие работу отделов ВУЗа работающих с информацией о студентах.

ЗАКЛЮЧЕНИЕ

В процессе дипломной работы была разработана концептуальная модель серверной части информационной интегрированной системы “Студент”, и были выполнены следующие задачи:

- исследованы современные архитектуры для создания корпоративных информационных систем;
- выбрана оптимальная методика построения интегрированных информационных систем;
- на основе была выбранной методики была разработана концептуальная модель данных.

Таким образом в данной дипломной работе были вынесены рекомендации по усовершенствованию модели данных системы “Бухгалтерия” и создана концептуальная модель предметной области при помощи которой можно разработать приложение для автоматизации деятельности отделов ВУЗа работающих с информацией о студентах.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

BHP92 M. Bright, A. Hurson, and S. Pakzad. A Taxonomy and Current Issues in Multidatabase Systems. IEEE Computer, 25(3):50-56, March 1992.

BKNW91 T. Barsalou, A. Keller, Siambela N., and G. Wiederhold. Upgdating Relational Database through Object-Based Views.//J. Clifford and R. King, editors, Proc. Of the 1991 ACM SIGMOD Int. Conf. On Management of Data, Denver, Colorado, SIGMOD RECORD 20(2), pages 248-257. ACM Press, June 1991.

BLN86 C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schemata Integration. //ACM Computing Surveys, 18(4):323-364, December 1986.

GSC95 M. Garcia-Solaco, F. Saltor, and M. Castellanos. A Structure Based Schemata Integration Methodologies.//P. S. Yu and A. L. P. Chen, editors, Proc. Of the 11th IEEE Int. Conf. On Data Engineering (ICDE'95), Taipei, Taiwan, pages 505-512. IEEE Computer Society Press, March 1995.

HM85 D. Heimbinger and D. McLeod. A Federated Architecture for Information Managment.//ACM Transactions on Office Information Systems, 3(3):253-278, 1985.

JQ92 H. V. Jagadish and X. Qian. Integrity Maintenance in an OODB.//L. Y. Yuan, editors, Proc. of the 18th Int. Conf. On Very Large Data Bases (VLDB'92), Vancouver, Canada, pages 469-480. Morgan Kaufmann Publishers, August 1992.

KC86 S. N. Khoshafian and G. P. Copeland. Object Identity.//N. Meyrowitz,

editors, Proc of the 1st Int Conf. On Object Oriented

Programming Systems, Languages and Applications (OOPSLA'86), Portland, Oregon, SIGPLAN Notices 21(11), pages 406-416. ACM Press, November 1986.

LMR90 W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of Multiple Autonomous Database.//ACM Computing Surveys, 22(3):267-293, September 1990.

PBE95 E. Pitoura, O. Bukhers, and A. K. Elmagarmid. Object Orientation in Multidatabases Systems.//ACM Computing Surveys, 27(2):141-195, June 1995.

RPRG94 M. P. Reddy, B. E. Prasad, and A. Gupta. Formulating Global Integrity Constrains during Derivation of Global Schema.// Data & Knowledge Engineering, 16(3):241-268, July 1995.

SCG91 F. Saltor, M. Castellanos, and M. Garcia-Solaco. Suitabikity of Data Models as Canonical Models for Federated Databases.//J. Clifford and R King, editors, Proc. of the 1991 ACM SIGMOD Int. Conf. On Manadement of Data, Denver, Colorado, SIGMOD RECORD 20(2), pages 44-48. ACM Press, June 1991.

Sch95 I. Schmitt. Flexible Integration and Derivation of Heterogeneous Schemata in Fedarated Database Systems. //Preprint Nr. 10, Fakultät für Informatik, Universität Magdeburg, November 1995.

SL90 A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogenous, and Autonomous Database. ACM Computing Surveys, 22(3):183-236, September 1990.

SPD92 S. Spaccapeitra, C. Parent, and Y. Dupont. Model Independed Assertions

for Integration of Heterogeneous Schemas.//The VLDB Juornal, 1(1):81- 126, 1992.