Проблема интеграции филиалов и КИС ВУЗа на уровне данных

Шахгельдян Карина Иосифовна

Князев Вячеслав Александрович

Владивостокский Государственный Университет Экономики и Сервиса
Владивосток, e-mail: carinash@vvsu.ru, vyacheslav.knyazev@vvsu.ru

В условиях растущей информатизации крупных предприятий все чаще возникает вопрос об интеграции данных, находящихся в различных базах данных (БД) на различных серверах. Это способствует поиску решения таких задач, как распределение нагрузки между серверами, объединение работы различных сервисов в корпоративную информационную среду (КИС) предприятия, создание хранилищ с избыточной информацией для анализа объединенных данных, связь КИС предприятия с КИС его филиалов и т.д.

Существует множество работ, посвященных исследованию проблемы репликации данных, и каждая известная система управления базами данных (СУБД) уже имеет встроенную возможность репликации данных [1]. Также существуют разработки сторонних производителей [2], решающие нестандартные ситуации в задачах интеграции на уровне данных. Несмотря на попытки стандартизировать механизм репликации, например, встроенными репликаторами в ведущих СУБД, каждое из решений задач репликации на уровне данных охватывает только ограниченный круг ситуаций, где оно может быть применено.

Репликация (replication) определяется как процесс автоматического распределения копий данных и объектов баз данных между экземплярами сервера БД с одновременной синхронизацией всей распространяемой информации. С некоторой долей условностей данные в корпоративной БД можно разделить на справочники и оперативные данные. Справочники – это относительно статическая информация, 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.

Сформулируем одну из задач репликации, возникающую во время интеграции КИС вуза с его филиалами на уровне данных. Из-за больших нагрузок на сервера баз данных и низкой пропускной способности канала связи с филиалами Владивостокского государственного университета экономики и сервиса (ВГУЭС) было принято решение распределить корпоративные БД ВГУЭС на центральные и базы данных филиалов. При этом необходимо реплицировать справочную информацию из центральной БД в БД филиалов и оперативные данные из БД филиалов в центральную БД. Поскольку внедрение корпоративного программного обеспечения в филиалах производится постепенно, то требуется возможность постепенного расширения объема реплицируемых данных.

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 из филиала 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 филиалов заменяются на те, которые присвоились этим записям в корпоративной базе данных. Подход опасен тем, что при задержке в репликации возможно использования неизменных 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 центральной базе данных соответствия идентификационных данных филиалов с данными головного вуза. Поля таблицы соответствия заполняются после выполнения репликации.

Предпочтительными для КИС являются два подхода: первый и последний. Первый подход позволяет определить для каждого филиала 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.

Встроенные механизмы репликации СУБД [1] описанной выше проблемы не учитывают, поэтому не могут быть применены для решения поставленной задачи. Корпоративные базы данных ВГУЭС созданы раньше возникновения проблемы интеграции с филиалами, поэтому алгоритм асинхронной репликации [2], который предъявляет требования к первичному ключу быть 64-х разрядным и составным, также не подходит. Помимо этого, по данному алгоритму можно производить репликацию полностью базы данных, а не частичного набора таблиц.

На первом этапе интеграции КИС ВГУЭС с филиалами было разработано и применено частичное решение сформулированной проблемы. Данное решение не позволяет удалять в подчиненных БД данные, удаленные в реплицируемых БД. Решение реализовано в виде нескольких алгоритмов.

Условия работы данного метода: у всех таблиц есть первичный ключ, в таблицах нет составных первичных ключей, нет связи «многие ко многим», никакие две таблицы не могут ссылаться друг на друга (иметь внешние ключи друг друга), либо ссылаться друг на друга посредством промежуточных таблиц (например, таблица a ссылается на таблицу b, таблица b ссылается на таблицу c, а таблица c ссылается на таблицу a).

Перед настройкой репликации необходимо на сервере филиала развернуть базу данных, структура которой будет соответствовать центральной базе. Все таблицы делятся на два вида: справочники и оперативные.

Таблицы справочники (далее справочники) заполняются в центральной базе данных и во время репликации данных в филиалы записи полностью перезаписываются. В базах данных филиалов справочники нельзя изменять. Если первичный ключ справочников является счетчиком, то в БД филиалов он должен целочисленного типа (но не счетчиком). В справочниках не может быть внешних ключей из оперативных таблиц.

Оперативные таблицы в филиалы не реплицируются, вся информация, содержащаяся в них, заносится в филиалах.

В задачу интеграции с филиалами на уровне данных входит регулярное обновление справочников в филиалах из центральной БД и репликация оперативной информации из БД филиалов в центральную БД. Поэтому задача разбита на две подзадачи: репликации справочников и репликации оперативных таблиц.

В обоих случаях перед репликацией данных необходимо упорядочить список реплицируемых таблиц. Каждая из подзадач обладает своим алгоритмом обновления данных в таблице. Ниже приводится алгоритм упорядочивания реплицируемых таблиц:

Алгоритм упорядочивания таблиц для репликации данных

В алгоритме используются два списка: таблиц, которые надо обновить и обновленных таблиц. В начале алгоритма в первом списке содержатся все таблицы для репликации, второй список пуст.

Введем понятие: различными внешними ключами реплицируемых таблиц являются внешние ключи, ссылающиеся на таблицы из списка реплицируемых и не на саму себя.

1. На первом шаге упорядочим список таблиц по количеству различных внешних ключей в них по возрастанию.
2. К таблицам, у которых нет различных внешних ключей, применяем алгоритм обновления данных. Удаляем обновленные таблицы из первого списка и записываем их во второй.
3. Выделяем те таблицы, различные внешние ключи которых являются первичными уже обновленных таблиц. К ним так же относятся таблицы, удовлетворяющие этому правилу, у которых есть внешний ключ/ключи собственного первичного ключа. Обновляем эти таблицы по алгоритму обновления. Переводим их из первого списка во второй.
4. Если первый список не пуст, то переходим к шагу три.

Условия для выполнения данного алгоритма: нет ни одной реплицируемый таблицы, которая имеет внешний ключ нереплицируемой таблицы.

Обновление справочников филиалов.

Во многих КИС работа со справочниками решена следующим образом. Справочники редактируются только в центральной базе данных, а в филиалы реплицируется весь справочник или некоторое подмножество его кортежей. Изменения справочников в филиалах запрещено.

Проблема возникает в связи с возможным нарушением целостности данных при репликации. Например, в справочнике базы *X* ранее существовала запись, которую удалили в связи с тем, что не использовали в настоящем и не собираются использовать в будущем. Но в базе филиала *Z* могли внести информацию, связанную с этой записью. При попытке реплицировать обновленный справочник из *X* в *Z* произойдет ошибка, и репликация не будет выполнена. Решений этой проблемы может быть несколько.

1. Удаление всех связных данных в базе данных *Z* с записью, которую удалили в справочнике в базе *X*.
2. Реплицировать новые элементы в справочнике и не удалять существующие.
3. Замена связей с удаленной записью на некоторую заранее определенную связь (в том числе null или определенную запись из справочника в *X*) с уведомлением администратора.
4. Откат 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 справочники в филиалах и центральной базе перестают быть синхронными и это влечет за собой дальнейшие ошибки при репликации оперативных данных из филиала в центральную

базу данных. Однако, его существование допустимо, если риск асинхронности справочников невелик. Ниже приводится алгоритм обновления таблицы справочника, отражающий второе решение.

Алгоритм обновления таблицы справочника

1. Отсортировать записи в справочнике центральной БД по столбцу, являющемуся первичным ключом, по возрастанию.
2. Запустить цикл по отсортированным записям справочника центральной БД. На каждом шаге цикла выполнять шаг 3.
3. Если в справочнике филиала присутствует запись с соответствующим значением первичного ключа, то обновить значение всех полей данной записи, кроме первичного ключа. Иначе добавить в справочник филиала новую запись.

Условия для выполнения данного алгоритма: если в справочнике присутствуют внешние ключи, ссылающиеся на первичный ключ самого справочника, то их значения не могут быть больше, чем значение первичного ключа соответствующей записи (т.е. запись в справочнике не может ссылаться не ещё не существующую запись).

Обновление оперативных данных из филиалов.

Для решения проблемы идентификации ключевых полей был выбран способ сохранения в центральной базе данных соответствия идентификационных данных филиалов с данными головного вуза. Разработанный метод работает только для таблиц, чьи первичные ключи одного типа данных и в случае, если первичные ключи генерируются автоматически. В случае разнотипных первичных ключей возможно расширить систему таблиц, хранящих соответствие записей филиалов и центральной БД.

Рекомендуется использовать две таблицы-справочника: справочник филиалов и справочник реплицируемых оперативных таблиц.

В КИС ВГУЭС была использована следующая система таблиц:

1. Таблица справочник списка реплицируемых таблиц: filial_link_tables (id int PK, table_name varchar(100))
2. Таблица справочник списка филиалов: filial_link_filials (id int PK, filial_name varchar(100))
3. Таблица соответствия первичных ключей: filial_links (inner_id int, filial_id int, table_id int FK, filial_id int FK)

Алгоритм обновление оперативных данных из филиала

1. Запустить цикл по записям оперативной таблицы.
2. Если для записи таблицы филиала есть запись первичного ключа (filial_id) в таблице соответствий, то обновляем значения полей таблицы центральный БД с первичным ключом, значение которого равно значению inner_id, взятому из таблицы соответствия.
3. Иначе добавляем новую запись в оперативную таблицу центральной БД. В таблицу соответствий в поле inner_id добавляем полученный первичный ключ новой записи.

Недостатки этого метода описаны выше, среди достоинств можно отметить, что данный метод решения не привязан к конкретной СУБД, не требует модификации структуры таблиц и не нуждается в их описании. Также данное решение можно легко масштабировать на следующих этапах внедрения КПО в филиалы.

На следующем этапе интеграции вуза с филиалами на уровне данных требуется устранить главный недостаток предыдущего метода - добавить обработку удаления реплицируемых данных.

Чтобы учитывать удаление данных из справочников и оперативных таблиц при репликации, необходимо использовать описание логической взаимосвязи таблиц. Такое описание можно реализовать с помощью модели обобщенного репозитория метаданных (ОРМД) корпоративной информационной среды вуза [3, 4].

В этом случае можно привести следующее краткое описание методики учета удаленных данных, рассмотрев случай удаления справочных значений. При репликации справочника, необходимо выделять все записи справочника филиала, которым нет соответствующих по первичному ключу записей в соответствующем справочнике центральной БД. После этого заносить информацию об этих записях в таблицу удаленных справочных данных, расположенную в БД филиала. После этого периодически запускать процедуру очистки удаленных справочных значений. Эта процедура будет согласно описанию данных сортировать удаляемые значения по степени ссылки друг на друга таблиц, в которых они содержатся, затем проверять, ссылаются ли ещё какие-нибудь таблицы на удаляемую запись. Если нет, то запись удаляется. Этот принцип напоминает принцип корзины в ОС Windows с автоматическим чистильщиком. Также можно обрабатывать и удаление оперативных данных в центральной БД, реплицированных с филиалов.

Использование полного описания репликаций в ОРМД, а так же алгоритма репликации справочников позволяет легко корректировать репликацию введением в описание ОРМД новых классов, соответствующих справочникам, не изменяя при этом кода репликации. Такой механизм позволят реализовывать репликацию без необходимости использования SQL-запросов.

Библиография

1. Официальный учебный курс Microsoft Репликация баз данных // Официальный учебный курс Microsoft Администрирование MS SQL Server 2000 (+CD). 2-е изд.: 2006. – Гл. 15. // http://www.sql.ru/articles/mssql/2006/050201DatabaseReplication.shtml
2. Губаев Марат Алгоритм асинхронной репликации// http://repl2.narod.ru/
3. Шахгельдян К.И. Модель обобщенного репозитория метаданных корпоративной информационной среды вуза//Системы управления и информационные технологии.-2006.-№2.1(24).-с.201-204.
4. Шахгельдян К.И. Корпоративная информационная среда: подход, основанный на понятиях//Информационные технологии моделирования и управления.-2006.-№4(29).-с.503-510.