УДК 004.414.28

МОДЕЛЬ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ ПАРАЛЛЕЛЬНОЙ СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ДЛЯ ГРИД

А.В. Лепихов

Статья посвящена вопросам проектирования параллельных систем управления базами данных для грид-систем. Описаны общие требования и структура ПСУБД. Описана модель вариантов использования системы, приведены спецификации и диаграммы вариантов использования подсистем «Клиент» и «Координатор».

Ключевые слова: *варианты использования, грид, система баз данных, параллельная обработка*

Введение

В связи со значительным ростом хранимой в электронном виде информации, в настоящее время системы баз данных зачастую используются для управления петабайтами данных. Типичным примером таких сверхбольших баз данных являются научные базы данных, накапливающие информацию, поставляемую ускорителями элементарных частиц. Подобные базы данных могут эффективно поддерживаться и обрабатываться параллельными системами управления базами данных (ПСУБД).

По этой причине в настоящее время одним из перспективных направлений в области параллельных систем баз данных является разработка систем управления базами данных для аппаратных платформ с многопроцессорной иерархической архитектурой и грид-систем. В такой системе процессорные устройства, память, диски и проч. связываются друг с другом в соответствии с некоторой иерархией. На первом уровне иерархии находятся процессорные ядра, размещенные на одном кристалле. На втором уровне находятся многоядерные процессоры, объединенные в многопроцессорные модули с общей памятью (SMP). На третьем уровне SMP-модули объединяются в кластер с помощью высокоскоростной соединительной сети. Четвертый уровень представляют грид-системы, включающие в себя несколько кластеров. Корпоративные грид-системы могут объединяться в кооперативные грид-объединения на базе Интернет.

Параллельным системам управления базами данных посвящено большое количество работ, обзор которых можно найти в [1–3]. Однако проблематика систем баз данных для грид исследована мало.

Целью настоящей статьи является описание спецификации ПСУБД ориентированной на использование в грид. Спецификация параллельной СУБД для грид разработана с учетом распределенного характера системы, неоднородности вычислительных узлов и коммуникационной сети. ПСУБД проектируется таким образом, чтобы изолировать пользователя от сложной структуры и технологий, лежащих в основе грид-системы. Требования к ПСУБД задаются при помощи модели вариантов использования [4], построенной на основе языка моделирования UML версии 2.0 [5].

Статья организована следующим образом. В разделе 1 описан механизм обработки запросов в параллельных СУБД и рассмотрена проблема появления перекосов. Приведены

основные требования, предъявляемые к разрабатываемой ПСУБД для грид-систем. В Разделе 2 рассмотрена структура ПСУБД, дано описание подсистем «Клиент», «Координатор» и «Ядро ПСУБД». Приводится диаграмма состояний подсистемы «Клиент». Раздел 3 посвящен описанию вариантов использования ПСУБД. Специфицированы варианты использования подсистем «Клиент» и «Координатор», представлены диаграммы вариантов использования. В разделе 4 дано описание форматов входных и выходных данных ПСУБД. В заключении суммируются основные полученные результаты.

1. Организация параллельной обработки запросов

1.1. Фрагментный параллелизм

Вопросам организации параллельной обработки запросов в многопроцессорных системах посвящен целый ряд работ (см. [6–8]). Одной из основных форм параллельной обработки запросов является *фрагментный параллелизм*. Фрагментный параллелизм предполагает *фрагментацию* – разбиение на непересекающиеся части каждого отношения базы данных. Фрагменты отношения распределяются по различным процессорным узлам многопроцессорной системы. Основная идея заключается в том, чтобы преобразовать запрос в набор параллельных агентов, каждый из которых будет обрабатывать свою часть запроса на выделенном для него процессорном узле. При этом он будет использовать расположенный на данном узле фрагмент и (возможно) результаты работы других агентов. В работе [6] преобразование запроса в набор параллельных агентов достигается путем вставки специального оператора **exchange** в дерево запроса. Обработка запроса в таком случае происходит следующим образом [9]. Пусть система состоит из n процессорных узлов. Запрос на языке SQL передается пользователем на host-машину, где он транслируется в некоторый *последовательный физический план*. Данный физический план преобразуется в *параллельный план*, представляющий собой совокупность параллельных агентов. Это достигается путем вставки параллельного оператора **exchange** в соответствующие места дерева запроса. Оператор **exchange** организует взаимодействие параллельных агентов и обеспечивает пересылку кортежей между агентами в тех случаях, когда это необходимо.

На следующем этапе параллельные агенты пересылаются с host-машины на соответствующие процессорные узлы, где интерпретируются исполнителем запросов. Результаты выполнения агентов объединяются корневым оператором **exchange** на выделенном процессорном узле.

1.2. Балансировка загрузки

Большое негативное влияние на производительность параллельных СУБД для многопроцессорных систем оказывает проблема перекосов [9]. Суть проблемы заключается в следующем. При параллельной обработке запроса каждый агент обрабатывает свою часть запроса на отдельном процессорном узле. В этом случае возникает ситуация, когда один или несколько агентов завершают обработку, в то время как остальные (загруженные) агенты активно работают. Таким образом, при обработке запроса часть процессорных узлов может простаивать, что уменьшает эффект от распараллеливания и увеличивает время обработки запроса. Подобные ситуации могут возникать вследствие неоднородности процессорных узлов, при неравномерном распределении базы данных по процессорным узлам и т.д.

Одним из наиболее известных способов решения данной проблемы является использование алгоритмов балансировки загрузки, основанных на репликации базы данных. В работе [10] в качестве стратегии репликации предлагается метод частичного зеркалирования. При использовании данной стратегии каждый фрагмент на логическом уровне разбивает-

ся на равномощные сегменты, являющиеся единицами репликации данных и балансировки загрузки. Фрагмент, расположенный на одном узле, частично реплицируется на остальных узлах многопроцессорной системы. Размер реплицируемой части фрагмента на каждом узле задается некоторым коэффициентом репликации.

Указанный метод позволяет нейтрализовать негативный эффект ситуации перекоса: уменьшить время простоя процессорного узла и время обработки запроса.

1.3. Общие требования к параллельной СУБД

Параллельная система управления базами данных (ПСУБД) представляет собой программную систему, которая предназначена для параллельной обработки запросов в распределенных вычислительных системах.

ПСУБД поддерживает фрагментный параллелизм и имеет механизм балансировки загрузки, основанный на стратегии репликации, называемый методом частичного зеркалирования [10]. Заложенный в ПСУБД параллелизм обработки запросов прозрачен для пользователя.

ПСУБД поддерживает многопользовательский режим работы, т.е. одновременно система может обрабатывать запросы нескольких пользователей. Для организации приема запросов от пользователей в ПСУБД использован реляционный язык запросов RQL (см. раздел 4.1). Запрос, записанный на языке RQL, передается ПСУБД в виде текстового файла. После окончания обработки запроса ПСУБД передает пользователю *результирующую таблицу* и *лог-файл*.

Результирующая таблица представляется в виде текстового файла в формате CSV (Comma Separated Value) [11]. Если результатом запроса является пустая таблица, то текстовый файл с результирующей таблицей будет пустым.

Лог-файл передается пользователю в виде текстового файла в формате CSV. Лог-файл содержит диагностические сообщения о ходе выполнения запроса. Структура и семантика сообщений лог-файла определены в разделе 4.2.

2. Структура ПСУБД

Параллельная СУБД состоит из объектов четырех классов: *Пользователь Клиент*, *Координатор* и *Ядро ПСУБД*. Пример, демонстрирующий статическую структуру ПСУБД, показан на рис. 1.

Пользователь представляет собой человека, взаимодействующего с ПСУБД посредством Клиента, однако Клиент допускает вызов в формате командной строки и может быть использован программными системами, являющимися внешними по отношению к ПСУБД.

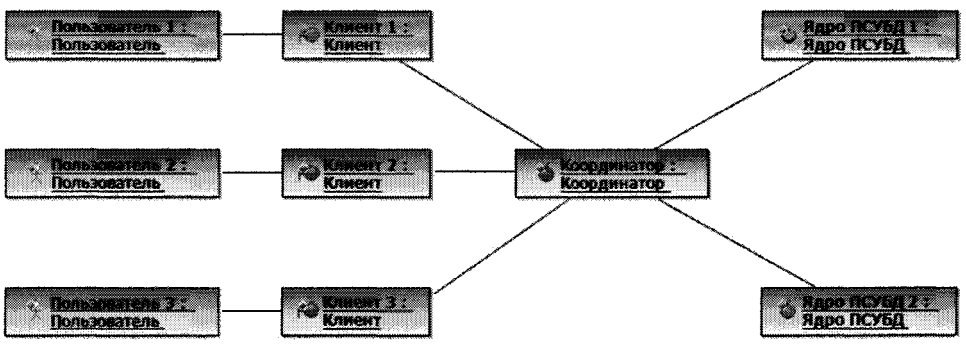

Рис. 1. Диаграмма объектов для случая с тремя Пользователями и двумя Ядрами ПСУБД

Клиент представляет собой подсистему ПСУБД, реализующую интерфейс Пользователя. Для каждого Пользователя ПСУБД порождается отдельный экземпляр Клиента. Поступающие от пользователя запросы Клиент передает Координатору, который обеспечивает совместную обработку запроса Ядрами ПСУБД.

Основной задачей Координатора является доставка запроса от Клиента на Ядра ПСУБД. Координатор принимает запрос от каждого обратившегося к нему Клиента и передает его Ядрам ПСУБД, которые будут выполнять обработку данного запроса. В рамках рассматриваемого примера совместная обработка запросов осуществляется двумя Ядрами ПСУБД и управляется одним Координатором. В целях повышения производительности в распределенной вычислительной системе может быть запущено несколько Координаторов. В таком случае Клиент может обращаться с запросами к любому доступному Координатору. Необходимость введения дополнительного Координатора может возникать, например, при увеличении количества Клиентов или объемов поступающих от Ядер ПСУБД результатов.

Ядро ПСУБД непосредственно выполняет обработку запроса и передает результат тому Координатору, от которого поступил запрос.

На рис. 2 показана типовая конфигурация размещения подсистем ПСУБД. Клиент запускается на рабочей станции пользователя (в данной кофигурации на консоли). Если с ПСУБД на одной рабочей станции взаимодействует несколько пользователей, то для каждого пользователя на данной рабочей станции запускается отдельный экземпляр Клиента.

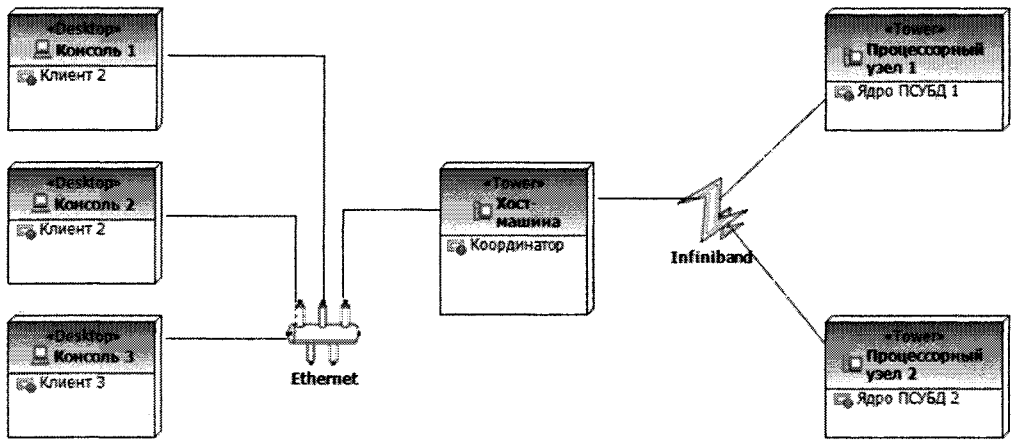

Рис. 2. Диаграмма размещения подсистем ПСУБД для случая с тремя Пользователями и двумя Ядрами ПСУБД

Координатор размещается в распределенной вычислительной системе на выделенном узле (хост-машине). Координатор взаимодействует с Клиентами через локальную коммуникационную сеть (в данном случае Ethernet). Координатор взаимодействует с ядрами через коммуникационную сеть (в данном случае Infiniband). Ядро ПСУБД размещается на выделенном ему процессорном узле. Процессорные узлы системы соединяются между собой коммуникационной сетью (Infiniband).

2.1. Подсистема «Клиент»

Клиент предоставляет пользователю интерфейс доступа к ПСУБД: обеспечивает передачу запросов ПСУБД, мониторинг процесса обработки запроса и доступ к результату его выполнения. В каждый момент времени Клиент может обрабатывать только один запрос пользователя. Клиент может использоваться в диалоговом и пакетном режимах. При

использовании Клиента в диалоговом режиме пользователю предоставляется диалоговое окно, основными элементами которого являются строка ввода запроса, область просмотра результата и область просмотра диагностических сообщений. В пакетном режиме работы Клиенту передается имя файла с запросом и имя файла, в который будет записана результирующая таблица.

На рис. 3 показана диаграмма состояний Клиента. Клиент может находиться в одном из семи состояний: «Ожидание запроса», «Выполнение запроса», «Сохранение результата», «Сохранение лог-файла», «Прием сообщения», «Прием результата», и «Прерывание запроса».

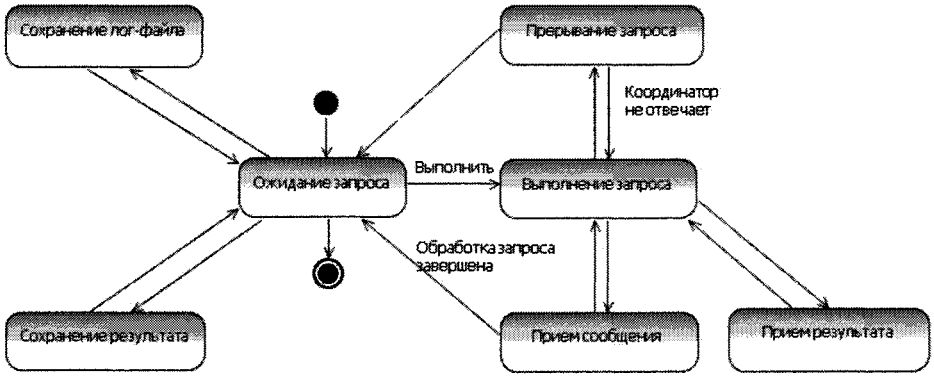

Рис. 3. Диаграмма состояний подсистемы «Клиент»

Рабочий цикл Клиента начинается с состояния «Ожидание запроса», в котором Клиент ожидает команд от Пользователя. Из данного состояния Клиент может перейти в состояние «Выполнение запроса», «Сохранение результата» или «Сохранение лог-файла». Пользователь может также корректно завершить Клиента, вызвав функцию «Завершить работу».

В состоянии «Выполнение запроса» Клиент ожидает результатов обработки запроса от Координатора. При поступлении от Координатора результирующей таблицы или диагностического сообщения, Клиент переходит в состояние «Прием результата» или «Прием сообщения» соответственно. Пользователь может спровоцировать прерывание обработки запроса, вызвав функцию Клиента «Прервать». При этом, Клиент переходит в состояние «Прерывание запроса».

В состоянии «Сохранение результата» Клиент помещает результирующую таблицу в заданный пользователем файл. При нормальном завершении операции Клиент переходит в состояние «Ожидание запроса».

В состоянии «Сохранение лог-файла» Клиент помещает полученные в ходе обработки запроса диагностические сообщения в указанный пользователем лог-файл. При нормальном завершении операции Клиент переходит в состояние «Ожидание запроса».

В состоянии «Прием сообщения» Клиент принимает диагностическое сообщение от Координатора Если полученное сообщение уведомляет об окончании обработки запроса, то Клиент переходит в состояние «Ожидание запроса». В противном случае он переходит в состояние «Выполнение запроса».

В состоянии «Прием результата» Клиент принимает результирующую таблицу от Координатора. При нормальном завершении операции Клиент переходит в состояние «Ожидание запроса».

В состоянии «Прерывание запроса» Клиент передает Координатору сообщение о прерывании обработки запроса. Если сообщение удается передать Координатору, прерывание запроса заканчивается успешно и Клиент переходит в состояние «Ожидание запроса». Если

Координатор оказывается недоступен, Клиент переходит в состояние «Выполнение запроса».

2.2. Подсистема «Координатор»

Координатор организует параллельное выполнение запроса на множестве Ядер ПСУБД. При этом, одно Ядро ПСУБД одновременно может обрабатывать несколько запросов. Координатор может взаимодействовать с несколькими Клиентами. Во время работы Координатор ведет лог-файл, в который заносятся диагностические сообщения о работе Координатора.

2.3. Подсистема «Ядро ПСУБД»

Ядро ПСУБД работает на отдельном SMP-узле и выполняет обработку поступающих запросов в асинхронном режиме. На одном SMP-узле может работать только одно Ядро ПСУБД. Ядро ПСУБД поддерживает межтранзакционный параллелизм, т.е. одновременно может обрабатывать несколько запросов.

3. Модель вариантов использования

В данном разделе приведены варианты использования подсистем «Клиент» и «Координатор». Вариант использования описывается в соответствии со следующей дисциплиной. Во-первых, приводится краткое описание варианта использования и предварительные условия, необходимые для начала его выполнения. Во-вторых, приводится описание основных шагов, составляющих поток событий варианта использования. В-третьих, обсуждаются альтернативные потоки событий, приводятся дополнительные требования.

3.1. Варианты использования подсистемы «Клиент»

Для подсистемы «Клиент» нами выделяются следующие варианты использования, приведенные на рис. 4: *«Выполнить»*, *«Прервать»*, *«Передать сообщение»*, *«Пополнить результат»*, *«Сохранить результат»*, *«Сохранить лог-файл»* и *«Завершить работу»*. Варианты использования предусматривают диалоговый и пакетный режимы работы Клиента. Далее в этом разделе приводится детальное описание каждого варианта использования.

3.1.1. Вариант использования «Выполнить»

Данный вариант использования передает запрос от Пользователя к Координатору. Вариант использования начинается, когда Пользователь указывает путь к текстовому файлу с запросом и активирует функцию «Выполнить». При этом Клиент должен находиться в состоянии ожидания запроса и текстовый файл с запросом должен существовать на узле Клиента.

Поток событий состоит из следующих шагов. На первом шаге Клиент переходит в состояние «Выполнение запроса» и подготавливает систему к обработке запроса: очищает область просмотра результирующей таблицы, очищает область просмотра сообщений, формирует пустой файл результата и пустой лог-файл. На втором шаге Клиент передает текстовый файл с запросом Координатору. Вариант использования завершается.

Вариант использования предусматривает один альтернативный поток событий. Если попытка Клиента передать запрос Координатору завершается неудачей, то Клиент выводит в области просмотра соответствующее сообщение об ошибке, записывает его в лог-файл и переходит в состояние «Ожидание запроса». Вариант использования завершается.

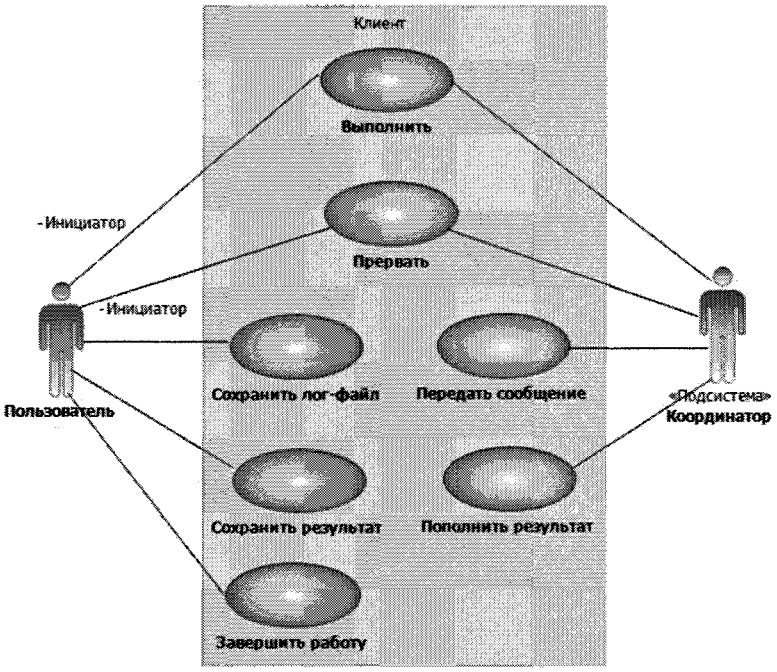

Рис. 4. Диаграмма вариантов использования подсистемы «Клиент»

3.1.2. Вариант использования «Прервать»

Данный вариант использования осуществляет прерывание выполнения запроса (досрочное завершение) по требованию Пользователя. Вариант использования начинается, когда Пользователь вызывает функцию «Прервать». При этом Клиент должен находиться в состоянии «Выполнение запроса».

Поток событий состоит из следующих шагов. На первом шаге Клиент переходит в состояние вывода сообщения и посылает Координатору сообщение «Прервать». Если передача сообщения завершается успешно, то на втором шаге Клиент формирует сообщение о досрочном прерывании запроса, выводит его в области просмотра сообщений и записывает в лог-файл. На последнем шаге Клиент переходит в состояние «Ожидание запроса». Вариант использования завершается.

Если попытка Клиента передать сообщение Координатору завершается неудачей, то он формирует сообщение об ошибке, выводит его в области просмотра и записывает в лог-файл. После этого Клиент переходит в состояние «Выполнение запроса». Вариант использования завершается.

3.1.3. Вариант использования «Передать сообщение»

Данный вариант использования уведомляет Клиента о возникновении некоторого существенного события в ходе выполнения запроса. Вариант использования начинается, когда Координатор вызывает функцию «Передать сообщение». При этом Клиент должен находиться в состоянии «Выполнение запроса».

Поток событий состоит из следующих шагов. На первом шаге Клиент переходит в состояние «Прием сообщения» и получает от Координатора диагностическое сообщение. На втором шаге Клиент выводит полученное сообщение в области просмотра сообщений и записывает его в лог-файл. На третьем шаге Клиент переходит в состояние «Выполнение запроса». Вариант использования завершается.

Вариант использования предусматривает альтернативный поток событий. Если Кли-

ент на первом шаге получает от Координатора сообщение с кодом фатальной ошибки или кодом, соответствующим нормальному завершению, то выполняется следующая последовательность шагов. На первом шаге Клиент выводит полученное сообщение в области просмотра сообщений и записывает его в лог-файл. На втором шаге Клиент переходит в состояние «Ожидание запроса», после чего вариант использования завершается.

3.1.4. Вариант использования «Пополнить результат»

Данный вариант использования передает очередную часть результирующей таблицы от Координатора Клиенту. Вариант использования начинается, когда Координатор вызывает функцию «Пополнить результат».

Поток событий состоит из следующих шагов. На первом шаге Клиент переходит в состояние «Прием результата» и принимает от Координатора очередную часть результирующей таблицы. На втором шаге Клиент выводит полученную часть результирующей таблицы в области просмотра результата и записывает ее в результирующий файл. На третьем шаге Клиент переходит в состояние выполнения запроса. Вариант использования завершается.

3.1.5. Вариант использования «Сохранить результат»

Данный вариант использования помещает результат запроса в текстовый файл в формате CSV. Вариант использования начинается, когда Пользователь вызывает функцию «Сохранить результат». При этом Клиент должен находиться в состоянии ожидания нового запроса.

Поток событий состоит из следующих шагов. На первом шаге Клиент переходит в состояние «Сохранение результата», получает от Пользователя имя файла, в который будет помещена результирующая таблица. На втором шаге Клиент сохраняет результирующую таблицу в текстовом файле, указанном пользователем, в формате CSV. На третьем шаге Клиент переходит в состояние «Ожидание запроса». Вариант использования завершается.

Полученная Клиентом результирующая таблица хранится во временном файле до тех пор, пока Пользователь не запустит выполнение нового запроса.

3.1.6. Вариант использования «Сохранить лог-файл»

Данный вариант использования сохраняет лог-файл запроса в текстовом файле в формате CSV. Вариант использования начинается, когда Пользователь вызывает функцию «Сохранить лог-файл». При этом Клиент должен находиться в состоянии «Ожидание запроса».

Поток событий состоит из следующих шагов. На первом шаге Клиент переходит в состояние «Сохранение лог-файла», запрашивает у Пользователя путь и имя файла для сохранения лог-файла. На втором шаге Клиент сохраняет лог-файл в текстовом файле, указанном пользователем. На третьем шаге Клиент переходит в состояние «Ожидание запроса». Вариант использования завершается.

Полученные Клиентом от Координатора сообщения хранятся во временном файле до тех пор, пока Пользователь не запустит выполнение нового запроса.

3.1.7. Вариант использования «Завершить работу»

Данный вариант использования завершает сеанс работы Пользователя с ПСУБД. Вариант использования начинается, когда Пользователь вызывает функцию «Завершить работу». При этом Клиент должен находиться в состоянии «Ожидание запроса».

Поток событий состоит из следующих шагов. На первом шаге Клиент освобождает используемые структуры данных. На втором шаге Клиент завершает свою работу (возвращает

поток управления операционной системе). Вариант использования завершается.

3.2. Варианты использования подсистемы «Координатор»

Для подсистемы «Координатор» нами были разработаны следующие варианты использования, приведенные на рис. 5: *«Передать запрос»*, *«Передать результат»*, *«Передать сообщение»*, *«Прервать»*. Координатор взаимодействует с множеством Клиентов и должен представлять собой систему высокой готовности. Поэтому варианты использования спроектированы таким образом, чтобы максимально уменьшить время отклика Координатора. Далее в этом разделе приводится детальное описание каждого варианта использования.

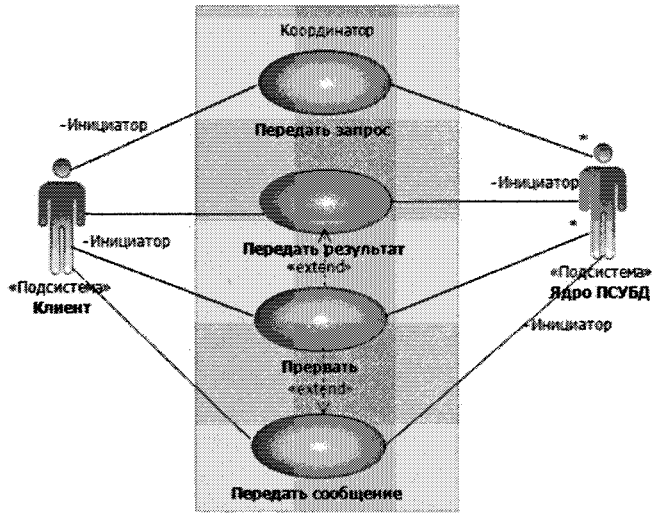

Рис. 5. Диаграмма вариантов использования подсистемы «Координатор»

3.2.1. Вариант использования «Передать запрос»

Передает запрос Ядрам ПСУБД для дальнейшей обработки. Вариант использования начинается, когда Клиент вызывает функцию «Передать запрос».

Поток событий состоит из следующих шагов. На первом шаге Координатор получает запрос от Клиента как параметр функции «Передать запрос». На втором шаге Координатор панализирует файл конфигурации ПСУБД и формирует список Ядер ПСУБД, которые будут выполнять запрос. На третьем шаге Координатор передает запрос Ядрам ПСУБД в асинхронном режиме. Вариант использования завершается.

3.2.2. Вариант использования «Передать результат»

Передает результирующую таблицу, выдаваемую Ядром ПСУБД, Клиенту. Вариант использования начинается, когда Ядро ПСУБД вызывает функцию «Передать результат».

Поток событий состоит из следующей последовательности шагов. На первом шаге Координатор получает результирующую таблицу в виде текстового файла от Ядра ПСУБД в синхронном режиме. На втором шаге Координатор определяет Клиента, который запустил данный запрос и вызывает функцию «Пополнить результат» (передает Клиенту файл с результатом, полученный от Ядра ПСУБД). Вариант использования завершается.

Вариант использования включает в себя следующие два альтернативных потока.

Если на шаге 2 основного потока Координатору не удается передать результирующую таблицу Клиенту, то выполняются следующие действия. На первом шаге Координатор запи-

сывает в лог-файл сообщение «Клиент недоступен». На втором шаге Координатор вызывает функцию «Прервать». Вариант использования завершается.

Если на шаге 2 основного потока Координатор обнаруживает, что Клиент прервал выполнение запроса, то Координатор удаляет результирующую таблицу и вариант использования завершается.

3.2.3. Вариант использования «Передать сообщение»

Уведомляет Координатор о возникновении некоторого существенного события, произошедшего с Ядром ПСУБД. Вариант использования начинается, когда Ядро ПСУБД вызывает функцию «Передать сообщение».

Основной поток событий состоит из следующей последовательности шагов. На первом шаге Координатор получает сообщение от Ядра ПСУБД. На втором шаге Координатор интерпретирует полученное сообщение и формирует диагностическое сообщение для клиента. На последнем шаге Координатор передает данное диагностическое сообщение Клиенту. Вариант использования завершается.

Вариант использования включает в себя три альтернативных потока.

Если Координатору на третьем шаге основного потока событий не удается передать диагностическое сообщение Клиенту, то начинается выполнение альтернативного потока, состоящего из следующих шагов. На первом шаге Координатор помещает в лог-файл, сообщение «Клиент недоступен». На втором шаге Координатор вызывает функцию «Прервать». Выполнение варианта использования завершается.

Если на шаге 2 основного потока Координатор обнаруживает, что получено сообщение «Обработка запроса завершена» то выполняется следующая последовательность шагов. На первом шаге Координатор удаляет ядро из списка Ядер ПСУБД, выполняющих данный запрос. На втором шаге Координатор передает Клиенту диагностическое сообщение о том, что Ядро ПСУБД завершило обработку запроса. Вариант использования завершается.

Если на шаге 1 альтернативного потока 2 Координатор обнаруживает, что список Ядер ПСУБД, обрабатывающих запрос, пуст, то выполняется следующая последовательность шагов. На первом шаге Координатор передает Клиенту диагностическое сообщение о том, что Ядро ПСУБД завершило обработку запроса. На втором шаге Координатор передает Клиенту диагностическое сообщение «Обработка запроса завершена». Вариант использования завершается.

3.2.4. Вариант использования «Прервать»

Останавливает выполнение запроса на ядрах ПСУБД по требованию Клиента. Вариант использования начинается, когда Клиент вызывает функцию «Прервать».

Основной поток событий состоит из следующей последовательности шагов. На первом шаге Координатор получает от Клиента сообщение «Прервать». На втором шаге Координатор посылает всем ядрам ПСУБД, задействованным в обработке запроса, сообщение «Прервать». Вариант использования завершается.

4. Форматы входных и выходных данных

4.1. Спецификация языка RQL

Язык запросов RQL (Relational Query Language) базируется на реляционной алгебре. Описанная здесь нотация включает в себя операции простой выборки и эквисоединения. Простая выборка (`Restrict`) допускает только условия следующего вида: `<Атрибут>`

<Операция сравнения> <Константа>. Операция эквисоединения (eQuijoin) выполняет соединение по равенству одной пары атрибутов. Предложенная нотация может быть легко распространена на другие реляционные операции.

В качестве разделителя лексем могут использоваться пробелы, символы табуляции, перевода строки и возврата каретки в любом количестве и сочетании.

Грамматика языка RQL, описанная с помощью нотации Бэкуса-Наура представлена ниже:

```
<Запрос> ::= <Оператор>
<Запрос> ::= <Запрос>;<Оператор>
<Оператор> ::= <Метка> <Реляционная операция>
<Реляционная операция> ::= <Выборка> | <Эквисоединение>
<Выборка> ::= R <Условие> <Таблица>
<Условие>::= <Номер атрибута> <Операция сравнения> <Константа>
<Эквисоединение> ::= Q <Номер атрибута> <Номер атрибута> <Таблица> <Таблица>
<Таблица> ::= <Метка> | <Идентификатор хранимой таблицы>
<Метка> ::= <Целое без знака>
<Номер атрибута> ::= <Целое без знака>
<Операция сравнения> ::= > | < | = <Идентификатор хранимой таблицы> ::= #<Целое
без знака>
```

4.2. Спецификация лог-файла

Лог-файл содержит диагностические сообщения, выдаваемые ядрами ПСУБД. В качестве разделителя лексем могут использоваться пробелы, символы табуляции, перевода строки и возврата каретки в любом количестве и сочетании. Описание структуры лог-файла в нотации Бэкуса-Наура представлена ниже:

```
<Лог-файл> ::= <Сообщение>{<Сообщение>}
<Сообщение> ::= <Ядро> <Результат> <Комментарий>
<Ядро> ::= <Целое без знака>
<Результат> ::= <Целое без знака>
<Комментарий> ::= <Строка>
```

5. Заключение

В данной работе были описаны спецификации параллельной системы управления базами данных для распределенных грид-систем. Спецификации разрабатывались при помощи модели вариантов использования на базе языка моделирования UML 2.0. В качестве введения в тематику параллельных СУБД был рассмотрен механизм параллельной обработки запросов, в основе которого лежит фрагментация базы данных по узлам распределенной вычислительной системы. Была рассмотрена проблема возникновения перекосов и возможные методы ее устранения. Описаны основные требования к параллельным СУБД.

Подробно рассмотрена структура ПСУБД, представлены основные подсистемы и схема их взаимодействия. Для описания структуры ПСУБД были разработаны диаграмма объектов и диаграмма размещения. Комментарии к диаграммам подробно описывают структуру ПСУБД и конфигурацию размещения подсистем на процессорных узлах распределенной вычислительной системы. Для уточнения рабочего цикла подсистемы «Клиент» была разработана диаграмма состояний. Для каждого состояния приведены комментарии, описывающие его назначение и условия перехода в другие состояния.

Детально описаны варианты использования для подсистем «Клиент» и «Координатор». Приведены диаграммы вариантов использования данных подсистем. Представлены

форматы входных и выходных данных параллельной ПСУБД.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 06-07-89148).

Литература

1. Gray, J. Query Evaluation Techniques for Large Databases / J. Gray // ACM Computing Surveys. – 1993. – Vol. 25, №2. – P. 73 – 169.
2. Mehta, M. Data Placement in Shared-Nothing Parallel Database Systems / M. Mehta, D.J. DeWitt // The VLDB J. – 1997. – Vol. 6, №1. – P. 53 – 72.
3. Williams, M.H. Data Placement in Parallel Database Systems / M.H. Williams, S. Zhou // Parallel Database Techniques. – 1998. – P. 203 – 218.
4. Rumbaugh, J. Getting started – Using Use Cases to Capture Requirements / J. Rumbaugh // Journal of Object Oriented Programming. – 1994. – Vol. 7, №5. – P. 8 – 12.
5. Selic, B. UML 2: a model-driven development tool / B. Selic // IBM Syst. J. – 2006. – Vol. 45, №3. – P. 607 – 620.
6. Соколинский, Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой / Л.Б. Соколинский // Программирование. – 2001. – №6. – С. 13 – 29.
7. The Grid-DBMS: Towards Dynamic Data Management in Grid Environments / G. Aloisio, M. Cafaro, S. Fiore, M. Mirto // Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05). – 2005. – Vol. 2. – P. 199 – 204.
8. An adaptive parallel query processing middleware for the grid / DaSilva V.F., Dutra M.L., Porto F., Schulze B., Barbosa A. C., de Oliveira J.C. // Concurr. Comput. : Pract. Exper. –2006. – Vol. 18, №6. – P. 621 – 634.
9. Костенецкий, П.С. Технологии параллельных систем баз данных для иерархических многопроцессорных сред / П.С. Костенецкий, А.В. Лепихов, Л.Б. Соколинский // Автоматика и телемеханика. –2007. – Т. 68, №5. – С. 847 – 859.
10. Лепихов, А.В. Стратегия размещения данных в многопроцессорных системах с симметричной иерархической архитектурой / Лепихов А.В., Соколинский Л.Б. // Научный сервис в сети Интернет: технологии параллельного программирования: тр. Всерос. науч. конф. (18 – 23 сент. 2006 г., г. Новороссийск). – М., 2006. – С. 39 – 42.
11. RFC4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files: (http://tools.ietf.org/html/rfc4180), October 2005.

Кафедра системного программирования,
Южно-Уральский государственный университет
lepihov@susu.ru

Поступила в редакцию 10 марта 2008 г.