УДК: 004.652.5:004.415.52

ОРГАНИЗАЦИЯ БАЗЫ ДАННЫХ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ КОНТРОЛЯ

Конокбаева А. К.
(Аспирант кафедры автоматического управления КГТУ им. И. Раззакова)

Статья посвящена распределенным базам данных в современном обществе, которые вносят новые информационные технологии в том числе и в образование. Показано, что распределенные базы данных играют в обществе важную роль, которое требует креативности, или же развивать креативное, поисковое, навигаторское мышление. Главное достоинство новой модели - открытость для каждого пользователя в системе и возможность самоорганизации.

Введение. Современные системы автоматизации строятся, как правило, с применением цифровых средств. В таких системах в качестве цифрового регулятора и устройства обработки информации используются микроконтроллеры и микро ЭВМ. Распределенные системы автоматизации строятся как двух- или трехуровневые. На верхних уровнях распределенных систем устройством сбора, преобразования, хранения, обработки информации и принятия решения являются промышленные или универсальные ЭВМ, например персональный компьютер, а на нижних уровнях - программируемые логические контроллеры (ПЛК) и промышленные ЭВМ. Информационное обеспечение распределенных систем автоматизации также является распределенной, оно должно быть организовано в виде распределенной базы данных. Для создания распределенной базы данных необходимо, прежде всего, разработать информационное обеспечение распределенной системы контроля [1-3].

Распределённая система - система, для которой отношения местоположений элементов (или групп элементов) играют существенную роль с точки зрения функционирования системы, а, следовательно, и с точки зрения анализа и синтеза системы.

Под распределенной информационной базой понимается неограниченное количество баз данных, дистанционно отдаленных друг от друга и имеющих ряд общих характеристик:

- функционирующих по единым правилам, определенным централизованно для всех баз данных, входящих в распределенную информационную базу;
- обмен данными осуществляется по правилам, также определенным централизованно.

Организация распределенной базы необходима для распределенных систем контроля и управления.

Архитектура распределенной системы: каждый компьютер является автономным модулем, состоящим из ЦП, памяти и периферийных устройств. Соответствие модели не нарушается даже несмотря на то, что компьютер не располагает локальной файловой системой: он должен иметь периферийные устройства для связи с другими машинами, а все принадлежащие ему файлы могут располагаться и на ином компьютере. Физическая память, доступная каждой машине, не зависит от процессов, выполняемых на

других машинах. Этой особенностью распределенные системы отличаются от сильно связанных многопроцессорных систем. Соответственно, и ядро системы на каждой машине функционирует независимо от внешних условий эксплуатации распределенной среды.

База данных физически распределяется по узлам данных на основе **фрагментации** и **репликации** данных. При наличии схемы реляционной базы данных каждое отношение фрагментируется на горизонтальные или вертикальные разделы. Горизонтальная фрагментация реализуется при помощи операции селекции, которая направляет каждый кортеж отношения в один из разделов, руководствуясь предикатом фрагментации. Фрагменты данных могут также реплицироваться на основе характера доступа к ним. Это полезно, если доступ к одним и тем же данным производится из приложений, выполняющихся на разных узлах. В таком случае, с точки зрения экономии затрат, более эффективно дублировать данные в ряде узлов, чем непрерывно пересылать данные между узлами.

ОТ идеальной распределенной базы данных (DDB) требуют [4] следующие 12 свойств или качеств DDB:

- локальная автономия (local autonomy);
- независимость от центрального узла (no reliance on central site);
- непрерывные операции (continuous operation);
- прозрачность расположения (location independence);
- прозрачность фрагментации (fragmentation independence);
- прозрачность тиражирования (replication independence);
- обработка распределенных запросов (distributed query processing);
- обработка распределенных транзакций (distributed transaction processing);
- независимость от оборудования (hardware independence);
- независимость от операционных систем (operationg system independence);
- прозрачность сети (network independence);
- независимость от баз данных (database independence).

Локальная автономия. Это качество означает, что управление данными на каждом из узлов распределенной системы выполняется локально. База данных, расположенная на одном из узлов, является неотъемлемым компонентом распределенной системы. Будучи фрагментом общего пространства данных, она, в то же время функционирует как полноценная локальная база данных; управление ею выполняется локально и независимо от других узлов системы.

Независимость от центрального узла. В идеальной системе все узлы равноправны и независимы, а расположенные на них базы являются равноправными поставщиками данных в общее пространство данных. База данных на каждом из узлов самодостаточна - она включает полный собственный словарь данных и полностью защищена от несанкционированного доступа.

Непрерывные операции. Это качество можно трактовать как возможность непрерывного доступа к данным в рамках DDB вне зависимости от их расположения и вне зависимости от операций, выполняемых на локальных узлах. Это качество можно выразить лозунгом "данные доступны всегда, а операции над ними выполняются непрерывно".

Прозрачность расположения. Это свойство означает полную прозрачность расположения данных. Пользователь, обращающийся к DDB, ничего не должен знать о реальном, физическом размещении данных в узлах информационной системы. Все операции над данными выполняются без учета их местонахождения. Транспортировка запросов к базам данных осуществляется встроенными системными средствами.

Прозрачность фрагментации. Это свойство трактуется как возможность распределенного размещения данных, логически представляющих собой единое целое.

Прозрачность тиражирования. Тиражирование данных - это асинхронный в общем случае процесс переноса изменений объектов исходной базы данных в базы, расположенные на других узлах распределенной системы. В данном контексте прозрачность тиражирования означает возможность переноса изменений между базами данных средствами, невидимыми пользователю распределенной системы. Данное свойство означает, что тиражирование возможно и достигается внутрисистемными средствами.

Обработка распределенных запросов. Это свойство DDB трактуется как возможность выполнения операций выборки над распределенной базой данных, сформулированных в рамках обычного запроса на языке SQL. То есть операцию выборки из DDB можно сформулировать с помощью тех же языковых средств, что и операцию над локальной базой данных.

Независимость от оборудования. Это свойство означает, что в качестве узлов распределенной системы могут выступать компьютеры любых моделей и производителей - от мэйнфреймов до персональных компьютеров.

Независимость от операционных систем. Это качество вытекает из предыдущего и означает многообразие операционных систем, управляющих узлами распределенной системы.

Прозрачность сети. Доступ к любым базам данных может осуществляться по сети. Спектр поддерживаемых конкретной СУБД сетевых протоколов не должен быть ограничением системы с распределенными базами данных. Данное качество формулируется максимально широко - в распределенной системе возможны любые сетевые протоколы.

Независимость от баз данных. Это качество означает, что в распределенной системе могут мирно сосуществовать СУБД различных

производителей, и возможны операции поиска и обновления в базах данных различных моделей и форматов.

Рассмотрим механизмы обеспечения некоторых свойств DDB.

Целостность данных. В DDB поддержка целостности и согласованности данных, ввиду свойств 1-2, представляет собой сложную проблему. Ее решение - синхронное и согласованное изменение данных в нескольких локальных базах данных, составляющих DDB - достигается применением протокола двухфазной фиксации транзакций. Если DDB однородна - то есть на всех узлах данные хранятся в формате одной базы и на всех узлах функционирует одна и та же СУБД, то используется механизм двухфазной фиксации транзакций данной СУБД. В случае же неоднородности DDB для обеспечения согласованных изменений в нескольких базах данных используют менеджеры распределенных транзакций. Это, однако, возможно, если участники обработки распределенной транзакции - СУБД, функционирующие на узлах системы, поддерживают XA-интерфейс, определенный в спецификации DTP консорциума X/Open. В настоящее время XA-интерфейс имеют CA-OpenIngres, Informix, Microsoft SQL Server, Oracle, Sybase.

Если в DDB предусмотрено тиражирование данных, то это сразу предъявляет дополнительные жесткие требования к дисциплине поддержки целостности данных на узлах, куда направлены потоки тиражируемых данных. Проблема в том, что изменения в данных инициируются как локально - на данном узле - так и извне, посредством тиражирования. Неизбежно возникают конфликты по изменениям, которые необходимо отслеживать и разрешать.

Прозрачность расположения. Это качество DDB в реальных продуктах должно поддерживаться соответствующими механизмами. Разработчики СУБД придерживаются различных подходов. Рассмотрим пример из Oracle. Пусть DDB включает локальную базу данных, которая размещена на узле в Лондоне. Создадим вначале ссылку (database link), связав ее с символическим именем (london_unix), транслируемым в IP-адрес узла в Лондоне:

CREATE PUBLIC DATABASE LINK london.com CONNECT TO london_unix USING oracle_user_ID.

Теперь можно явно обращаться к базе данных на этом узле, запрашивая, например, в операторе SELECT таблицу, хранящуюся в этой базе:

SELECT customer.cust_name, order.order_date FROM customer@london.com, order WHERE customer.cust_number = order.cust_number.

Очевидно, что запрос зависит от расположения базы данных. Определим customer и customer@london.com как синонимы:

CREATE SYNONYM customer FOR customer@london.com.

Теперь можно написать полностью независимый от расположения базы данных запрос:

SELECT customer.cust_name, order.order_date FROM customer, order WHERE customer.cust_number = order.cust_number

Задача решается с помощью оператора SQL CREATE SYNONYM, который позволяет создавать новые имена для существующих таблиц. При этом оказывается возможным обращаться к другим базам данных и к другим компьютерам. Так, запись в СУБД Informix

CREATE SYNONYM customer FOR client@central:smith.customer

означает, что любое обращение к таблице customer в открытой базе данных будет автоматически переадресовано на компьютер central в базу данных client к таблице customer. Оказывается возможным переместить таблицу из одной базы данных в другую, оставив в первой базе ссылку на ее новое местонахождение, при этом все необходимые действия для доступа к содержимому таблицы будут сделаны автоматически.

Во многих СУБД задача управления именами объектов DDB решается путем использования глобального словаря данных, хранящего информацию о DDB: расположение данных, возможности других СУБД (если используются шлюзы), сведения о скорости передачи по сети с различной топологией и т.д.

Обработка распределенных запросов. Выше уже упоминалось это качество DDB. Обработка распределенных запросов (Distributed Query -DQ) - задача, более сложная, нежели обработка локальных и она требует интеллектуального решения с помощью особого компонента - оптимизатора DQ. Обратимся к базе данных, распределенной по двум узлам сети. Таблица detail хранится на одном узле, таблица supplier - на другом. Размер первой таблицы - 10000 строк, размер второй - 100 строк (множество деталей поставляется небольшим числом поставщиков). Допустим, что выполняется запрос:

SELECT detail_name, supplier_name, supplier_address

FROM detail, supplier

WHERE detail.supplier_number = supplier.supplier_number;

Результирующая таблица представляет собой объединение таблиц detail и supplier, выполненное по столбцу detail.supplier_number (внешний ключ) и supplier.supplier_number (первичный ключ).

Данный запрос - распределенный, так как затрагивает таблицы, принадлежащие различным локальным базам данных. Для его нормального выполнения необходимо иметь обе исходные таблицы на одном узле. Следовательно, одна из таблиц должна быть передана по сети. Очевидно, что это должна быть таблица меньшего размера, то есть таблица supplier. От интеллекта оптимизатора DQ впрямую зависит скорость выполнения распределенных запросов.

Межоперабельность. В контексте DDB межоперабельность означает две вещи. Во-

первых, - это качество, позволяющее обмениваться данными между базами данных различных поставщиков. Как, например, тиражировать данные из базы данных Informix в Oracle и наоборот? Известно, что штатные средства тиражирования в составе данной конкретной СУБД позволяют переносить данные в однородную базу. Так, средствами CA-Ingres/Replicator можно тиражировать данные только из Ingres в Ingres. Как быть в неоднородной DDB? Ответом стало появление продуктов, выполняющих тиражирование между разнородными базами данных.

Во-вторых, это возможность некоторого унифицированного доступа к данным в DDB из приложения. Возможны как универсальные решения (стандарт ODBC), так и специализированные подходы. Очевидный недостаток ODBC - недоступность для приложения многих полезных механизмов каждой конкретной СУБД, поскольку они могут быть использованы в большинстве случаев только через расширения SQL в диалекте языка данной СУБД, но в стандарте ODBC эти расширения не поддерживаются.

Специальные подходы - это, например, использование шлюзов, позволяющее приложениям оперировать над базами данных в "чужом" формате так, как будто это собственные базы данных. Вообще, цель шлюза - организация доступа к унаследованным (legacy) базам данных и служит для решения задач согласования форматов баз данных при переходе к какой-либо одной СУБД. Так, если компания долгое время работала на СУБД IMS и затем решила перейти на Oracle, то ей очевидно потребуется шлюз в IMS. Следовательно, шлюзы можно рассматривать как средство, облегчающее миграцию.

Технология тиражирования данных. Принципиальная характеристика тиражирования данных (Data Replication - DR) заключается в отказе от физического распределения данных. Суть DR состоит в том, что любая база данных всегда является локальной; данные размещаются локально на том узле сети, где они обрабатываются; все транзакции в системе завершаются локально. Тиражирование данных - это асинхронный перенос изменений объектов исходной базы данных в базы, принадлежащим различным узлам распределенной системы. Функции DR выполняет, как правило, специальный модуль СУБД - сервер тиражирования данных, называемый репликатором (так устроены СУБД CA-OpenIngres и Sybase). В Informix-OnLine Dynamic Server репликатор встроен в сервер, в Oracle 7 для использования DR необходимо приобрести дополнительно к Oracle7 DBMS опцию Replication Option.

Простейший вариант DR - использование "моментальных снимков" (snapshot). Рассмотрим пример из Oracle:

```
CREATE SNAPSHOT unfilled_orders
REFRASH COMPLETE
START WITH TO_DATE ('DD-MON-YY HH23:MI:55')
NEXT SYSDATE + 7
AS SELECT customer_name, customer_address, order_date
FROM customer@paris, order@london
WHERE customer.cust_name = order.customer_number AND
order_complete_flag = "N";
```

"Моментальный снимок" в виде горизонтальной проекции объединения таблиц customer и order будет выполнен в 23:55 и будет повторятся каждые 7 дней. Каждый раз будут выбираться только завершенные заказы.

Реальные схемы тиражирования, разумеется, устроены более сложно. В качестве базиса для тиражирования выступает транзакция к базе данных. В то же время возможен перенос изменений группами транзакций, периодически или в некоторый момент времени, что дает возможность исследовать состояние принимающей базы на определенный момент времени.

Литература

1. Акматбеков Р. А. Концепция распределенной системы автоматизации очистной установки на базе информационных технологий / Телекоммуникационные и информационные технологии. Состояние и перспективы развития. КГТУ им. И. Раззакова – Б.: ОАО «Кыргызтелеком», 2008. - с.189-195.
2. Акматбеков Р. А. Системы автоматизации и управления: Учебник. - Б.: ИЦ “Техник”, 2013. - 206 с. (Учебник с грифом МОН КР).
3. Акматбеков Р. А., Конокбаева А. К. Информационное обеспечение распределенной системы контроля биологической очистки бытовых сточных вод.
4. Date C.J. An Introduction to Database Systems. - Addison-Wesley Professional, 2003. - 1024 p.