УДК 004.658.6:004.042

АЛЬТЕРНАТИВНЫЙ СПОСОБ РАЗРЕШЕНИЯ КОНФЛИКТОВ РЕПЛИКАЦИИ В РАСПРЕДЕЛЕННЫХ БАЗАХ ДАННЫХ ORACLE

Базилевский Е.В., Гришмановский П.В.

ГОУ ВПО «Сургутский государственный университет Ханты-Мансийского автономного округа – Югры», Сургут, Россия (628400, Тюменская обл., г. Сургут, ул. Ленина, 1), e-mail: chester917@yandex.ru

Распределенные базы данных, в отличие от централизованных, призваны сократить объемы информации, передаваемые по телекоммуникационным линиям, и повысить оперативность доступа к актуальным данным. Важной задачей является обеспечение оперативного устранения конфликтов репликации, синхронизации данных и обеспечении бесперебойного процесса репликации данных между узлами базы данных. Время, затраченное на решение данных задач, напрямую зависит от возможностей используемого программного обеспечения. Компания Oracle Corp. предлагает в составе СУБД Oracle решение, позволяющее управлять реплицированием данных в распределённых базах данных. На сегодняшний день это решение является недостаточно функциональным и эффективным в области разрешения конфликтных транзакций. В данной статье предложен альтернативный способ разрешения конфликтов репликации, который является более эффективным по сравнению с Oracle Enterprise Manager.

Ключевые слова: база данных, репликация данных, конфликты репликации, Oracle.

AN ALTERNATIVE WAY OF REPLICATION CONFLICTS RESOLVING IN DISTRIBUTED ORACLE DATABASE

Bazilevskiy E.V., Grishmanovsky P.V.

Surgut State University, Surgut, Russia (628400, Tyumen region, Surgut, Lenina st., 1), e-mail: chester917@yandex.ru

Unlike the centralized, distributed databases are designed to reduce the amount of information, transmitted over telecommunication lines and to improve the efficiency of access to relevant data. The important task is to ensure the operative replication conflicts' resolving, synchronize data and ensure an uninterrupted replication process between database nodes. Time, spent on the decision of these tasks, directly depends on possibilities of used software. Oracle Corp. offers a decision, which is included into Oracle DBMS and allow to operate the data replication at distributed databases. This solution is insufficiently functional and effective at conflict transactions' resolving sphere for today. This article suggests an alternative way of replication conflicts' resolving, which is more effective then Oracle Enterprise Manager.

Keywords: database, data replication, replication conflicts, Oracle.

Введение

В распределенной базе данных конфликт репликации возникает при одновременном (условно) изменении одной записи в результате выполнения транзакций разными серверами над локальными репликами. Однако конфликт будет обнаружен только при синхронизации данных. Все распределенные СУБД идентифицируют конфликты, но не всегда могут обеспечить автоматическое их устранение – отмену всех изменений или применение одного из них, т.к. эта задача требует вмешательства человека (администратора) из-за необходимости учета специфики назначения и структуры данных, конкретных значений, объектов и характера конфликта. Эффективность разрешения конфликтов, а с другой стороны – потери из-за отказов в выполнении запросов к заблокированным объектам базы

данных зависят от функциональных и эксплуатационных характеристик предоставляемого администратору инструментария СУБД.

Обработка конфликтов репликации в СУБД Oracle

В СУБД Oracle при возникновении конфликтной транзакции соответствующие данные помещаются в системную таблицу system.def$_aqerror. Основными полями, описывающими конфликтные транзакции, являются [2]:

- ENQ_TID – порядковый номер конфликтной транзакции;
- STEP_NO – номер вызова в транзакции;
- USER_DATA – непосредственно информация об изменении объекта БД.

Значение поля USER_DATA представляет собой массив байт, для хранения которого в Oracle используется тип BLOB. Байты в массиве располагаются следующим образом (рис. 1). Первый байт определяет тип аргумента, над которым пользователем были произведены изменения. Каждому типу аргумента соответствует определенное количество байт. Следующий байт определяет количество байт, в соответствие с указанным типом аргумента, в которых закодированы изменения аргумента пользователями. Далее в указанном количестве байт содержится информация об исходном значении аргумента до внесения изменений пользователями. Следующая группа байт такого же размера содержит модифицированное значение аргумента. Таким образом, алгоритм сводится к последовательному разбору массива байт, хранящегося в поле USER_DATA по каждому вызову в конфликтной транзакции. Текущее значение изменяемого аргумента определяется непосредственно из БД по SQL-запросу.

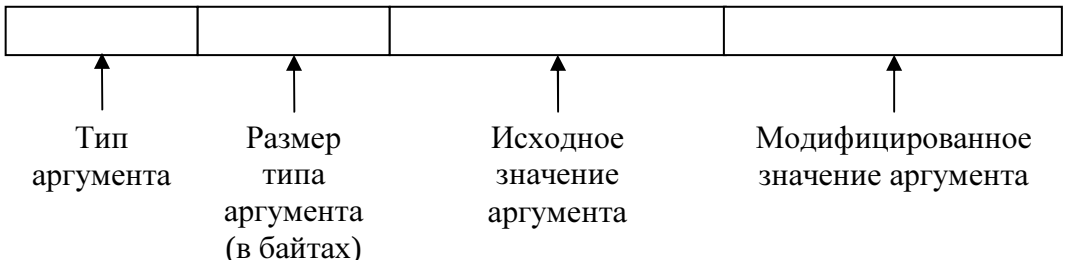

Рис. 1. Массив байт поля USER_DATA.

Таким образом, каждая транзакция содержит в себе определённое количество вызовов, которые в свою очередь содержат набор аргументов. Количество аргументов зависит от определённой таблицы, над которой были произведены действия пользователем, и если количество полей в таблице велико, то соответственно количество аргументов также будет большим.

Разработчиками СУБД Oracle предлагается администратору баз данных использовать функционал приложения Oracle Enterprise Manager (OEM) для разрешения конфликтных транзакций [5]. OEM последовательно извлекает конфликтные транзакции из очереди и выполняет их разбор при помощи системного (для СУБД Oracle) пакета dbms_defer_query. Полученная в результате разбора информация отображается для принятия решения администратором БД, после чего конфликтная транзакция удаляется из очереди. Общий алгоритм разбора конфликтной транзакции в OEM показан на рис. 2.

Разбор конфликтных транзакций в OEM реализован при помощи пакета dbms_defer_query. Данный пакет содержит следующие функции, которые позволяют получать данные, необходимые для разбора вызовов в конфликтных транзакциях [5]:

– get_arg_form – возвращает набор символов для указанного аргумента в вызове конфликтной транзакции, который используется в последующей обработке;
– get_arg_type – определяет тип аргумента в вызове;
– get_datatype_arg – определяет значение аргумента в вызове в соответствии с одним из следующих типов: NUMBER, VARCHAR2, CHAR, DATE, RAW, ROWID, BLOB, CLOB, NCLOB, NCHAR, NVARCHAR2, TIMESTAMP.

Текущее значение данных изменяемой таблицы определяется непосредственно путём обращения к таблице и считывания необходимых данных по каждому аргументу вызова конфликтной транзакции.

Языком разработки OEM является Java. В отличие от многих языков программирования, Java – это еще и программная платформа, включающая большой объём кода мощной библиотеки и среду для выполнения программ (Java Virtual Machine), которая обеспечивает безопасность, независимость от операционной системы и автоматическую «сборку мусора».

В руководстве по языку Java декларируются такие его свойства, как: простой, объектно-ориентированный, распределённый, надёжный, безопасный, независимый от архитектуры компьютера, переносимый, интерпретируемый, высокопроизводительный, многопотоковый, динамичный [3]. Такие характеристики, как независимость от архитектуры компьютера и производительность, требуют более подробного рассмотрения.

Компилятор Java генерирует объектный файл, формат которого не зависит от архитектуры компьютера. Объектный файл содержит байтовый код, разработанный таким образом, чтобы его можно было легко интерпретировать с помощью виртуальной машины [4]. Интерпретация байтового кода всегда будет выполняться медленнее, чем выполнение эквивалентного машинного кода. Однако на многих платформах возможна так называемая синхронная компиляция (just-in-time compilers – JIT) байтового кода, но даже в этом случае в

итоге не обеспечивается такая же производительность, как при компиляции, ориентированной на конкретный тип процессора [3].

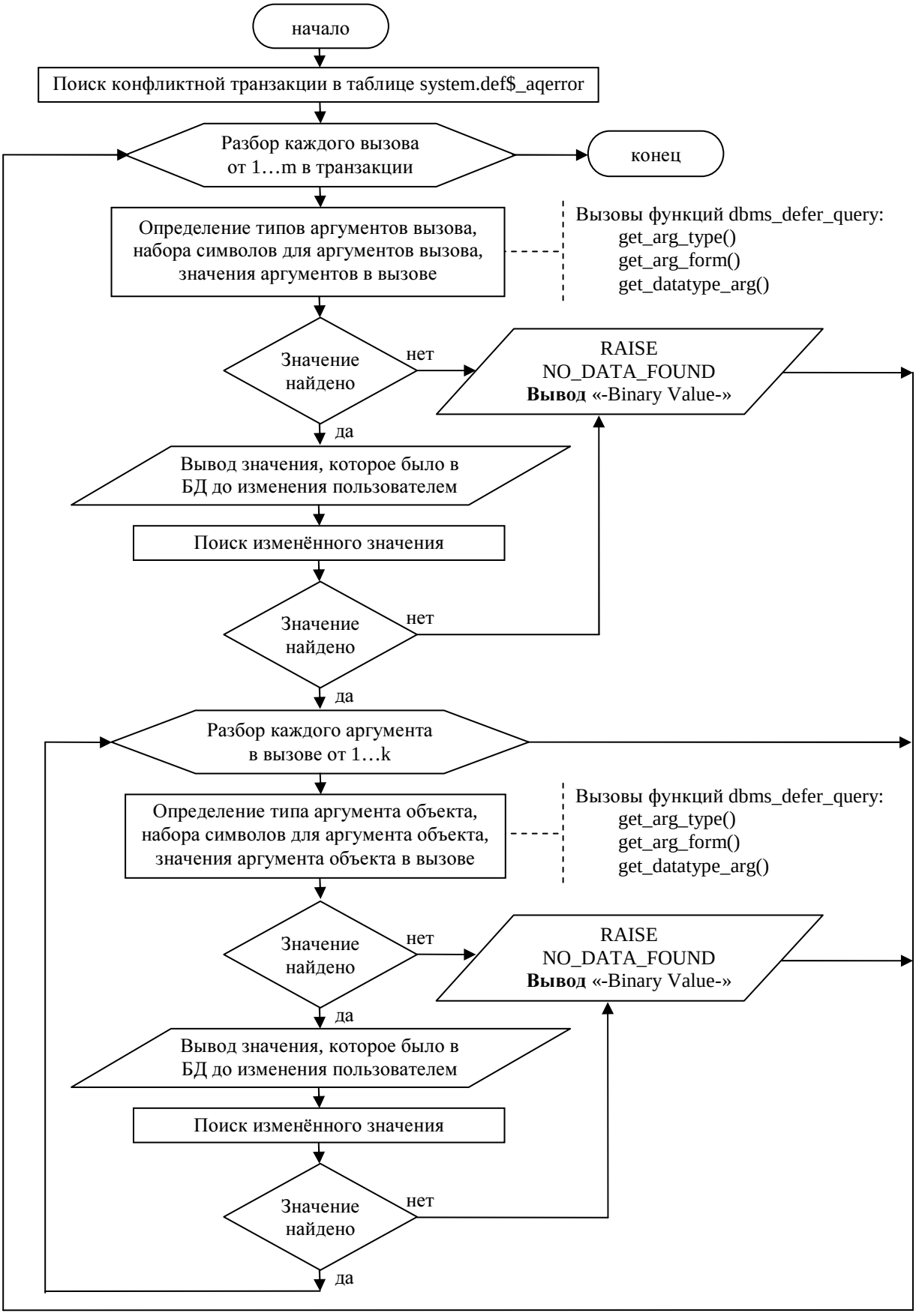

Рис. 2. Блок-схема разбора вызовов в OEM.

Таким образом, независимость от архитектуры компьютера является в то же время и слабой стороной Java, обуславливающей снижение производительности программ, разработанных на этой платформе. Тот факт, что OEM разработан на Java, является одним из факторов, снижающих производительность этого ПО.

Выполнение различных операций в OEM сопровождается множеством фоновых задач, выполняемых JVM, что существенно увеличивает время получения результата. Взаимодействие OEM с базой данных осуществляется посредством JDBC (Java Database Connectivity – интерфейс взаимодействия Java-приложений с базами данных) [5].

Для разрешения конфликтных транзакций в распределённой БД необходимо получить список конфликтных транзакций и детальную информацию по каждому вызову транзакции. Это реализуется посредством вызова упомянутых выше функций, исполняемых JVM, которая формирует ряд SQL-запросов к конкретной БД и получает результат их выполнения. После выполнения обработки полученной информации по заложенным в OEM алгоритмам JVM выполняет преобразование информации в удобочитаемую форму.

При таком взаимодействии OEM и JVM наиболее критическими факторами с точки зрения производительности являются следующие:

– время обработки действий администратора в OEM, время формирования соответствующих SQL-запросов к интересующей БД, время преобразования полученного результата из БД, а также время предоставления обработанных данных непосредственно администратору определяется не только быстродействием платформы, но и эффективностью интерпретации байт-кода;
– в силу большого объема данных, полученных из БД, и ограниченности ресурсов рабочего места администратора их оперативная обработка в OEM посредством JVM и отображение результата в OEM иногда оказываются невозможными (приложение «зависает»). По причине постоянного роста промышленных БД подобные ситуации довольно часты, но являются крайне нежелательными, особенно при разборе конфликтных транзакций и синхронизации данных между различными узлами распределённой БД;
– получение «старых», «текущих» и «новых» значений изменяемых объектов БД при разборе вызовов в конфликтных транзакциях происходит посредством выполнения соответствующих функций и процедур из системного пакета dbms_defer_query: get_arg_form, get_datatype_arg и get_arg_type. При большом количестве конфликтных транзакций с большим количеством вызовов в каждой из них серверу БД необходимо больше времени на обработку данных, что не позволяет оперативно получить информацию, необходимую для анализа и принятия решения по разрешению

конфликтных транзакций, тем самым синхронизировать данные между различными узлами распределенной БД.

Альтернативный способ разбора конфликтных транзакций

Описанные выше проблемы, свойственные OEM, устраняются при использовании утилиты ErrManager, разработанной автором в среде Embarcadero Delphi. ErrManager получает необходимые данные из БД путём SQL-запросов. Время получения данных ограничено только загруженностью БД и связью между клиентской машиной и сервером БД, а разбор конфликтной транзакции осуществляется на стороне клиента с существенно меньшим, в отличие от OEM, использованием ресурсов сервера БД [1]. Алгоритм разбора показан на рис. 3.

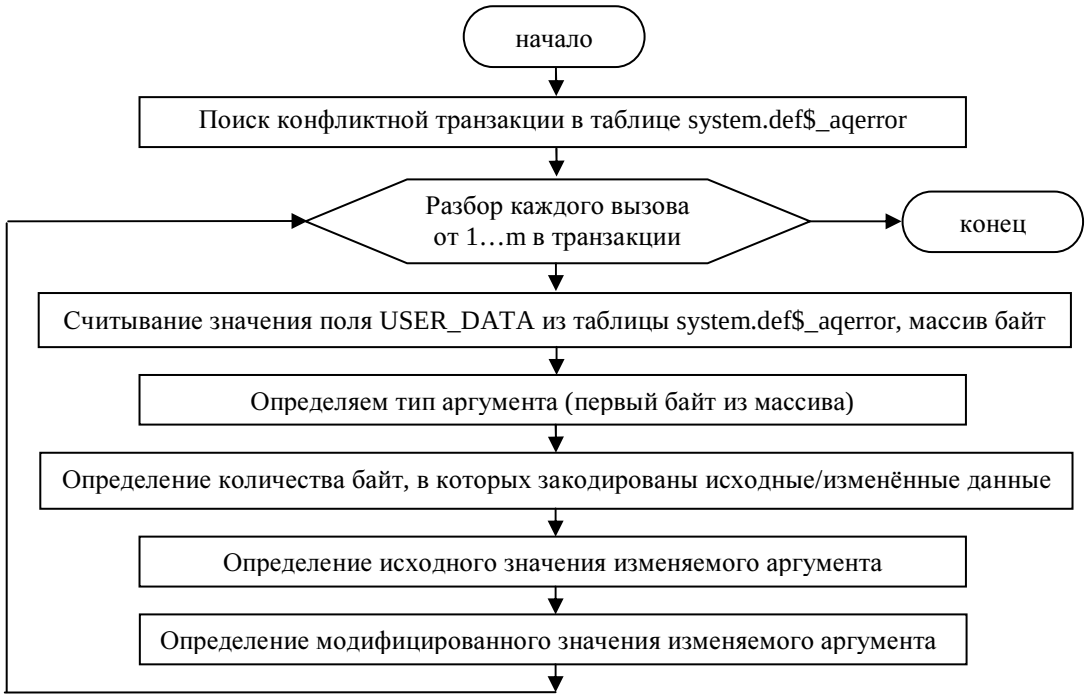

Рис. 3. Блок схема разбора вызовов в ErrManager.

ErrManager разбирает BLOB-значения без использования системного пакета dbms_defer_query, тем самым получая необходимые исходные значения изменяемых объектов БД. Это значительно снижает время получения старых, новых и текущих значений за счёт снижения количества и сокращения времени обращений к БД, тем самым значительно уменьшая время разрешения конфликта.

Таким образом, на основе анализа выше описанного алгоритма разбора вызовов в конфликтных транзакциях можно заключить, что время разбора значительно сокращается за счёт того, что для каждого вызова в транзакции анализируется фактически только однократно извлекаемое значение поля USER_DATA таблицы system.def$_aqerror. Разбор

массива байт путём считывания и преобразования в число или текст является более эффективным и не требует использования ресурсов БД, по сравнению с повторяющимися многократно вызовами набора функций из системного пакета при разборе вызовов в OEM.

Разработанная утилита ErrManager имеет вид приложения пользователя (рис. 4).

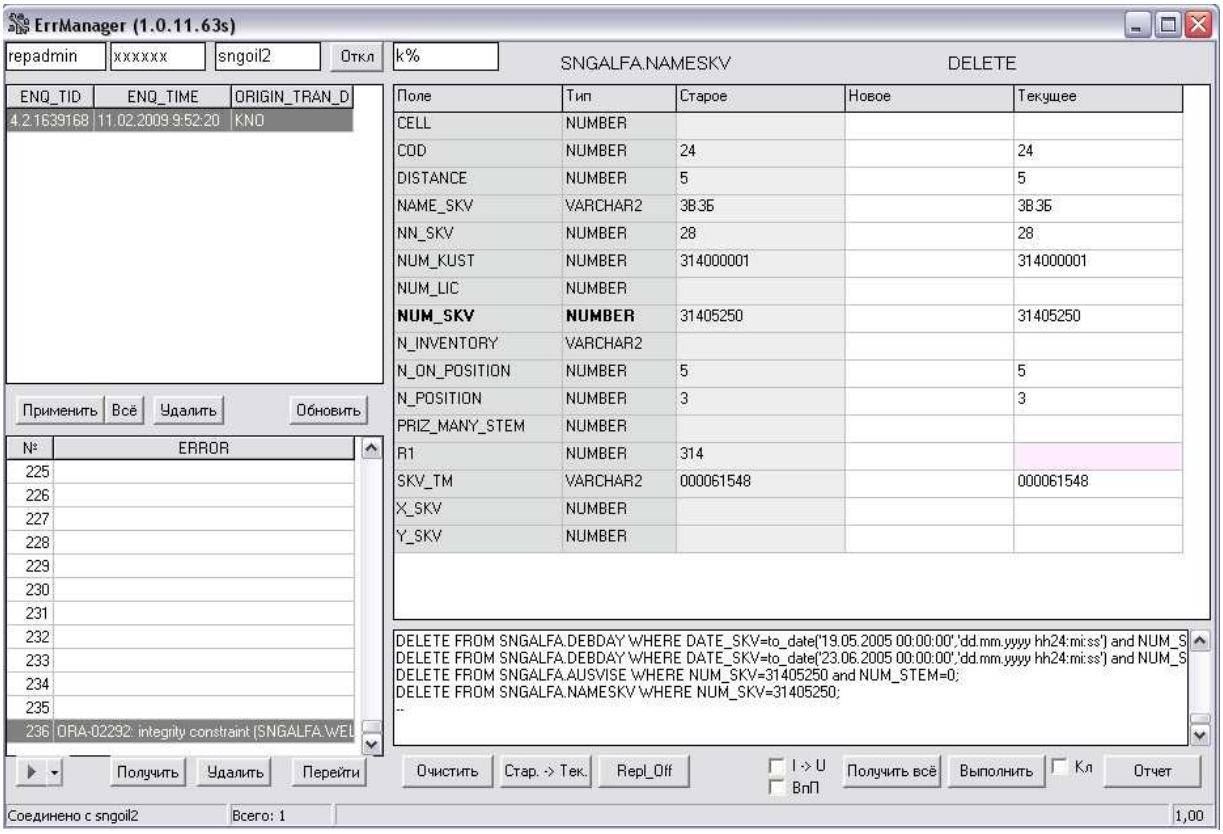

Рис. 4. Общий вид приложения ErrManager.

Функции, реализованные в утилите ErrManager, позволяют [1]:

- получать информацию о конфликтной транзакции и отображать информацию о состоянии данных в БД-источнике перед внесением изменения, данные, измененные пользователем, и текущие данные на БД-приемнике;
- преобразовывать изменения в SQL-запрос для повтора этих действий на приемнике;
- приводить запись на приемнике к такому же виду, какой она была на источнике до внесения изменений, что позволяет разрешить конфликт путём простого повтора выполнения ошибочной транзакции;
- повторять выполнение конфликтных транзакций и удалять их группами;
- автоматически разрешать конфликтные транзакции указанным способом (из числа приведенных выше) и очищать очередь транзакций после выполненных действий;
- отфильтровывать очередь конфликтных транзакций с разных БД при отображении в рабочей области программы;
- автоматически выявлять конфликтный вызов в транзакции;

- отключать репликацию на другие узлы распределённой БД, при синхронизации данных;
- автоматически преобразовывать SQL-запросы на вставку в запросы на изменение при наличии уже существующей записи в БД-приемнике;
- автоматически получать SQL-скрипт разрешения конфликтной транзакции и выполнять его на БД-приемнике;
- формировать отчет в форме HTML по указанным конфликтным транзакциям как в форме OEM, так и в собственной форме;
- позволяет вести мониторинг наличия конфликтных транзакций на указанном узле распределённой БД.

Заключение

Проведенное тестирование утилиты ErrManager включало в себя временную оценку процесса разрешения конфликтных транзакций и синхронизации данных при реплицировании информации между различными узлами распределённой БД. Результаты тестирования показали, что использование ErrManager является более эффективным по сравнению с Oracle Enterprise Manager.

Список литературы

1. Базилевский Е.В. Автоматизация деятельности администратора баз данных при работе с конфликтами репликаций в системах управления баз данных ORACLE // Наука и инновации XXI века : мат-лы X Юбил. окр. конф. молодых учёных, Сургут, 26-27 нояб. 2009 г. : в 2 т. / Сургут. гос. ун-т ХМАО – Югры. – Сургут : ИЦ СурГУ, 2010. – Т. 1. – С. 21-23.
2. Гаршин И.К. Практика работы с Oracle: генерация, администрирование, репликация. – М. : ПОЛТЕКС, 2000. – 250 с. : ил.
3. Хорстманн К., Корнелл Г. Java 2. Библиотека профессионала, том 1. Основы. – 7-е изд. : Пер. с англ. – М. : Издательский дом «Вильямс», 2006. – 286 с. : ил. – Парал. тит. англ.
4. James Gosling, Henry McGilton. The Java Language Environment: Contents [Электронный ресурс] / A White Paper // May 1996. – URL: http://java.sun.com/docs/white/ langenv (дата обращения: 27.02.2012).
5. Oracle [Электронный ресурс] / Oracle Replication. – URL: http://www.oracle.com/global/ru/pdfs/tech/tg_oracle_replication.pdf (дата обращения: 29.02.2012).

Рецензенты

Бадулин Н.Н., д.т.н., профессор кафедры радиоэлектроники ГОУ ВПО «Сургутский государственный университет Ханты-Мансийского автономного округа – Югры», г. Сургут.

Кожухов С.Ф., д.физ.-мат.н., профессор, заведующий кафедрой высшей математики факультета информационных технологий ГОУ ВПО «Сургутский государственный университет Ханты-Мансийского автономного округа – Югры», г. Сургут.