UDC 004.031

Włodzimierz Khadzhynov, Mateusz Maksymiuk
Technical University of Koszalin, Department of Electronics & Informatics
ul. Śniadeckich, 2, 75-453 Koszalin, Poland
e-mail: hadginov@ie.tu.koszalin.pl; mateuszmaksymiuk@wp.pl

Execution of replication in heterogeneous database systems

Database replication is a mechanism which allows distributed databases stay in synchronized and consistent state. In heterogeneous case there is a set of databases with different platforms, what additionally complicates this process. High demand for this technique causes that replication issue is developed very fast nowadays. Database replication mechanism consists in intercepting data modifications and propagating them to other databases where they are executed again. This process is very complex and there are many variants and types of replication. In this paper are discussed database replication classification (synchronous, asynchronous, multiple master, master slave), kinds of replication conflicts with resolving strategies, data modification capturing and publication-subscription model. There is introduced an example for realization of replication in heterogeneous environment. Also are discussed aspects of independence database assess mechanism from specific platform, what can be used in implementation of heterogeneous database replication.

Key words *Replication mode, Two-Phase Commit protocol, Master-Slave replication model, Multiple Master replication model, Replication conflicts, Hibernate package.*

1. Introduction

Precise definition of term «replication» was agreed in the middle of 90s. It describes database replication as a process of generating a distinct copy of data from one or more sources and delivering them to one or more destination nodes. Term «node» represents a single database, which is a part of distributed database system and takes part in replication process. Term «group of nodes» means a set of all nodes that take part in replication process.

Database replication is applied in many areas:

— synchronization of distributed databases;

— database security achieved through increase of data (data excess);

— increasing of system availability in the case of node failure;
— load distribution;
— data transfer from OLTP to DSS in order to increase efficiency;
— location of data in place where it is the most needed (limits communication cost).

2. Replication profile

An ideal model of database replication should meet the following criteria:
— data integration and consistency with transaction support (1-copy-serializability [1, 5]);
— synchronization (immediate transaction execution at all nodes);
— symmetrics (modifications allowed in every node);
— transparency;
— efficiency;
— scalability (efficient work regardless of a number of nodes);
— failure resistance.

Implementation of an ideal model is impossible because of physical limitations. Therefore there were created many different variants of database replication. Everyone of them offers only a subset of above-mentioned requirements.

2.1. Synchronous replication mode (Eager replication)

Synchronous replication realizes immediate execution of a transaction in all nodes [4]. Most often this mode uses such protocols as Two-Phase Commit — 2PC (Fig. 1). Besides there are many other protocols [2], which are generally named RMC protocols (Replica Majority Control). In the first phase of transaction commitment of 2PC protocol, there is made a common decision: commit or abort. In the second phase, the choosen decision is executed at all nodes. Unfortunately the main disadvantage of such protocols as 2PC, 3PC and others similar is large time consumption, which greatly decreases system efficiency [2, 3].

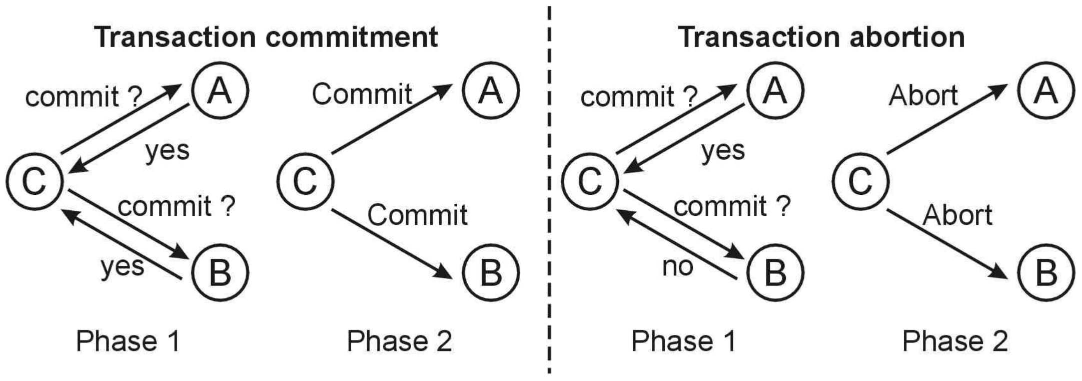

Fig. 1. Two-Phase Commit protocol (2PC)

In order to ensure concurrent access to data — there are used various resource blocking protocols. The most common used is Two-Phase Blocking protocol — 2PL (Fig. 2).

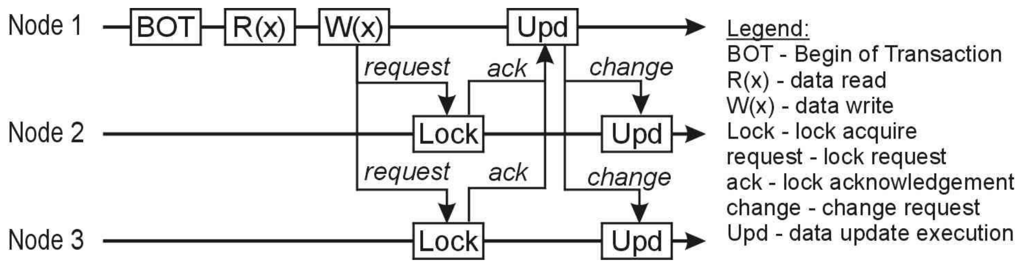

Fig. 2. Two-Phase Locking protocol (2PL)

Read of data is executed locally, therefore lock for reading is acquired only in local node. Write of data is performed at all nodes in a group, therefore locks for writing are acquired in every node. This technique is known as «Read one/Write all» (ROWA). Its disadvantage is requirement of all nodes availability. Because of this, there was developed another method named ROWAA, which means «Read one/Write all available». In this method the write is executed only at those nodes, which are available at the moment. Unfortunately, the use of blocking causes existence of deadlocks, which greatly decrease system efficiency [3].

Advantages:

— data consistency and integrity (1-copy-serializability);

— real-time data up-to-date.

Disadvantages:

— low efficiency caused by use of 2PC and 2PL protocols;

— low scalability;

— low failure resistance.

2.2. Asynchronous replication mode (Lazy replication)

Asynchronous replication (Fig. 3) does not require permanent network connection between all nodes. Each node operates separately and communication occurs periodically [4]. All reads and writes are performed locally, but every data change is captured and stored in appropriate buffers (usually FIFO queues). Each item stored in buffer includes significant informations, which allows to re-execute given modification in other nodes. Some of these informations are: modification time, table name, record key value, record field values before and after modification. Next, there occurs data synchronization at all nodes in the group. It may be executed periodically, by user demand or soon after modification occurs (case of asynchronous replication, which is near real-time). In synchronization process, all captured data changes in every node are propagated (transmitted) to all other nodes, where they are re-executed.

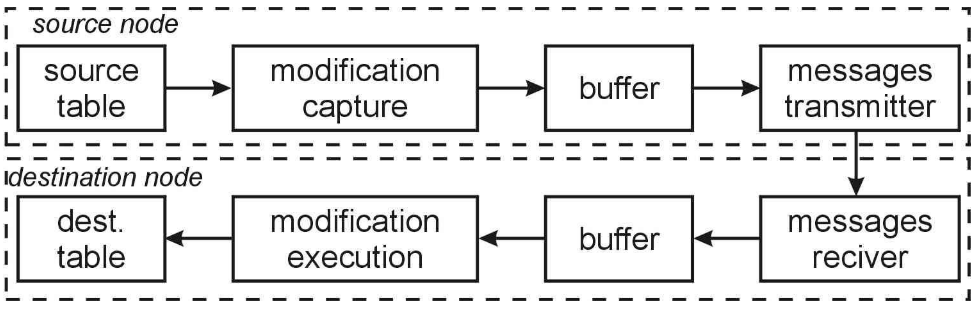

Fig. 3. Asynchronous replication

Asynchronous mode is frequently used with publication-subscription model. It allows selection only a subset of database tables which are included in replication process. The main disadvantage is lack of data consistency and integrity between consecutive replication executions. Furthermore, during the replication there may occur replication conflicts. This issue is described in the further paragraph.

Advantages:

— efficiency — lack of protocols such as 2PC and 2PL;

— scalability;

— no permanent connection requirement;

— failure tolerance;

— optimization possibility (multiple modifications of the same record are replaced by only one).

Disadvantages:

— lack of full consistency and integrity of data;

— occurrence of replication conflicts.

2.3. Master-Slave replication model (Primary Copy)

In Primary Copy model [5, 6] all data modifications are executed at only one node (so-called Master node). At all other nodes (so-called Slave nodes) there are performed only data read operations. Every data modification submitted at Slave nodes, is redirected to Master node, where it is executed. After execution in Master node its result is propagated (usually asynchronous) to Slave nodes (Fig. 4).

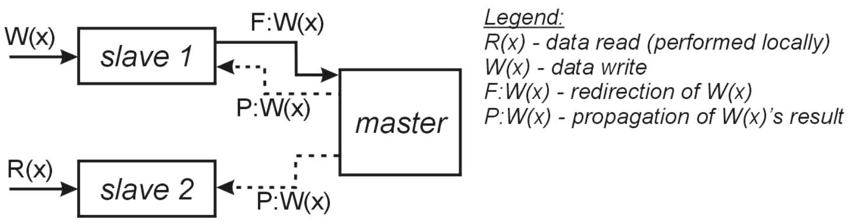
W(x)
slave 1
F:W(x)
P:W(x)
master
R(x)
slave 2
P:W(x)
Legend:
R(x) - data read (performed locally)
W(x) - data write
F:W(x) - redirection of W(x)
P:W(x) - propagation of W(x)'s result

Fig. 4. Master-Slave replication model

This model realizes unidirectional replication (from Master to Slave) which is asymmetrical (Master can read and write, Slave can only read). Such a conception significantly simplifies execution of replication process, because of exclusion the most important problems:

— better efficiency caused by elimination of RMC protocols (2PC & 2PL);

— lack of replication conflicts in asynchronous mode.

On the other hand, system disadvantages are:

— bottleneck of Master node;

— sensitivity to Master node failures;

— lack of full concurrency in data access (only read).

Unidirectional replication model is often used for increment data security (data multiplication). Also it is applied for data transfer from OLTP to DSS systems for better efficiency achievement.

2.4. Multiple Master replication model (Update Everywhere)

Multiple Master model [5, 6] does not require that data modifications are executed only in a single node. Every modification can be executed in any node in the group. This makes the system symmetric (all nodes have full read/write access to data), and replication is bidirectional. A diagram of Multiple Master model is presented below (Fig. 5).

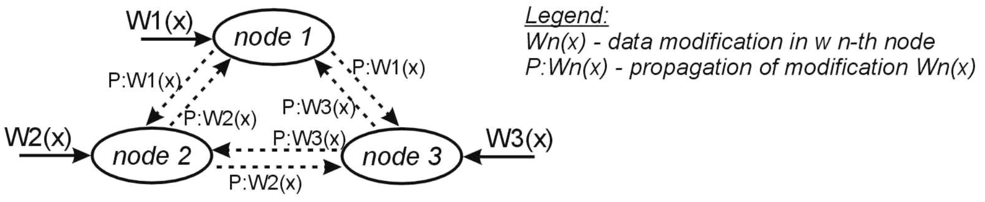
W1(x)
node 1
P:W1(x)
P:W1(x)
P:W2(x)
P:W3(x)
W2(x)
node 2
P:W3(x)
P:W2(x)
node 3
W3(x)
Legend:
Wn(x) - data modification in w n-th node
P:Wn(x) - propagation of modification Wn(x)

Fig. 5. Multiple-Master model

Because every node has write access to data, there can appear conflicts. Conflict occurs when at least two nodes execute modification of the same record. For synchronous mode it will be a locking conflict and for asynchronous mode — a replication conflict.

Conflicts occurred in replication process require appropriate resolving strategies. Deep analysis of this issue is described in further paragraphs.

Advantages:

— system symmetrics – data read and write in all nodes;

— less failure susceptibility.

Disadvantages:

— reduced efficiency, caused by requirement RMC protocols usage in synchronous mode;

— occurrence of replication conflicts.

2.5. Database platform uniformity — homogenous and heterogeneous replication

Each node, which is taking part in replication process, represents a specific database platform. Case, where all nodes represent an identical database platform, is called

homogenous replication. Much more complicated is a heterogeneous replication case (Fig. 6), where all nodes represent on different database platforms. This causes various problems:
— different binary formats of transaction log;
— incompatibility of data types;
— incompatibility in SQL syntax (and its procedural extensions).

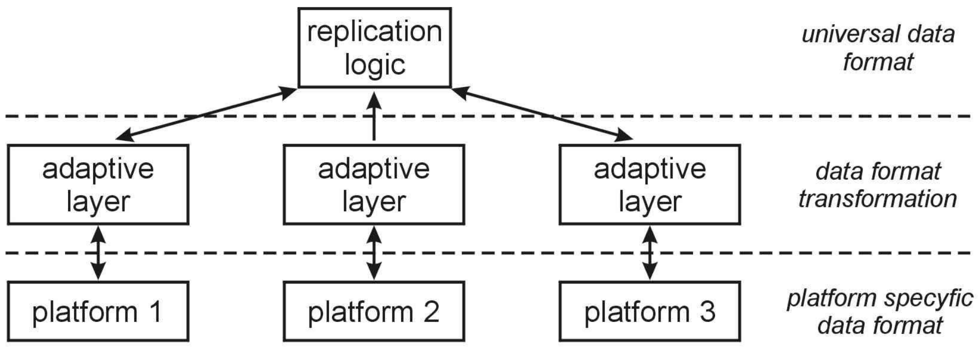

Fig. 6. Heterogenous replication

In order to solve above problems, there are required special mechanisms, which must be created separately for every database platform, taking part in replication process.

Among them, we can distinguish the following elements:
— database connection class libraries;
— transaction log reading mechanism;
— data type translation modules (between different platforms);
— SQL generation procedures.

To solve these problems, replication system should be divided into two layers:
— system logic layer (independent from any specific platform);
— adaptive layer (specific for given platform).

In system logic layer, there are defined replication algorithms, universal data types, abstract classes and interfaces, which are implemented in an adaptive layer. Every element in this layer is not associated with any specific database platform.

An adaptive layer is responsible for specific database platform access. It can read transaction log, some system tables, generate and execute SQL code or translate platform specific data types to universal data types defined in a system logic layer. Such architecture ensures independency from specific database platform and could be easily extended to support other platforms, through definition of new adaptive layers.

2.6. Data modifications capturing

Process of capturing data modifications executed in database is essential part of replication issue and it could be realized in various methods:
— transaction log reading;
— usage of triggers or rules;
— usage of network sniffer for capturing SQL commands sended to database;

— creation own database layer access (eg. Jdbc driver).

— creation proxy application in database access.

Transaction log reading ensures good efficiency and full system transparency, but is tightly connected to specific database platform and very difficult. Triggers are easy in implementation and universal, but cause large overhead and decrease efficiency. Creating own database driver causes small overhead but is not easy. The proxy application, which receives database commands, does not ensure system transparency.

2.7. Asynchronous replication conflicts

There are several kinds of asynchronous replication conflicts:

— Insert-Insert;

— Update-Update;

— Update-Delete.

Insert-Insert conflict occurs when two or more nodes try to insert record with the same primary key value. This will violate uniqueness constraint and prevents effective execution of these inserts.

Update-update conflict occurs when two or more nodes try to update the same record. To solve this problem it is necessary to make decision about final values of record that will be written to database.

Update-delete conflict occurs when the same record is updated in one node and deleted in another node. To solve the conflict it is necessary to make decision what action should be done, modification or deletion?

2.7.1. Conflict avoidance

2.7.1.1. Globally unique identifiers generation

The globally unique identifier problem [7] is essential for elimination of occurrence the insert-insert conflicts. The most frequently used methods that solve this problem are:

— identifier division into two parts : database identifier and sequence value;

— assignment for each node, sequence with separate value ranges.

The first technique divides identifier into two parts: unique node (database) identifier within nodes group and sequence value (same sequence for each node). These two values together guarantee globally identifier uniqueness.

The second way is assignment for each node, appropriate sequence with separate value ranges, ie. {0,…, 9999999}, {10000000,…, 19999999} etc.

Another method is usage of randomly generated GUID (Globally Unique Identifier) numbers, but it is seldom used and its main disadvantage is low efficiency.

2.7.1.2. Data ownership

In this method [8] every node is allows to modify only this data, which it owns. Every new-inserted record has strict membership to node, in which insertion has taken place (mother node). During replication process each record is replicated to all nodes in the group, but given record can be modified only in its mother node. In other nodes, record can be only read. For asynchronous mode such technique guarantees elimination of update-update and update-delete conflicts. It is achieved, because each record can be modified or deleted only in one node (mother node). Although this method is very effective its usage is limited. In many cases there is requirement for full data access concurrency, which excludes usage of this method.

2.7.2. Conflict resolving

These conflicts, which were not avoided, must be solved using appropriate strategies (algorithms). In this paragraph many various methods are discussed.

2.7.2.1. Primary key uniqueness conflict (Insert-Insert)

In case of primary key uniqueness violation, most often is used one of the following methods:

— addition (concatenation) to key field, the server name (for text fields);
— addition (concatenation) to key field, sequence value;
— insertion abort.

2.7.2.2. Methods of executing UPDATE modification

Important issue during Update-Update conflict resolving is the way of record modification. There are several options possible:

— modification applied to only changed columns;
— modification applied to defined groups of columns;
— modification of entire record (all columns).

First method enables modification of separate columns in different nodes. Changes will be consolidated into single UPDATE statement. In second method there are defined related column groups which must be updated simultaneously (ie. «postal code», «city» and «street»). Modification of entire record, according to values from the given node, discards any changes from other nodes.

2.7.2.3. Time-stamps

In this popular technique [7], decision about approvement or abortion of modification bases on modification occurrence timestamp. On this basis it could be decided, which modification would be accepted and which one would be discarded. The algorithm chooses only the most earliest or most latest modification. It could also sort all modifications by their timestamps and then execute each of them in proper order.

2.7.2.4. Node priority

Another technique bases on priorities assigned for each node [7]. Precedence of record modification has node which has the highest priority. Disadvantage of this technique is necessity of priorities (the best different) assignment for each node, what could be not easy.

2.7.2.5. Mathematical functions

Usage of mathematical functions [7] is frequently used for number fields (ie. integer, float). There are used various functions, such as:

— maximum;
— minimum;
— average;
— other algorithms.

3. Example implementation of asynchronous replication system in heterogeneous environment

In this paragraph there is discussed an application named «Universal Database Replicator». It realizes database replication in asynchronous mode. The project is developed within the confines of research in WEiPK at database laboratory on master's studies. Replicator realizes replication in heterogeneous environment, supporting three platforms: Sybase SQL Anywhere, MS SQL Server and PostgreSQL. Because of modular architecture and broad configuration's range, there is a possibility to extend the system with other platforms support.

3.1. Guidelines

During the system designing, there were taken the following guidelines:

— asynchronous replication, based on publication-subscription model;

— update everywhere model (multiple master);

— heterogeneous environment (SQL Server, Sybase SQL Anywhere, PostgreSQL);

— usage of conflict resolution strategies (timestamps, priorities);

— generation of globally unique identifier for primary key fields.

3.2. Implementation

The system was created on Java platform. Furthermore, there were used: DDL Utils class library and Hibernate environment. Because of this there was achieved isolation of a system logic from a database layer.

3.2.1. DDL Utils

DDL Utils library includes a set of classes designed for database's metadata management. It supports various database platforms, what ensures much flexibility and easy migration between different platforms. The library performs such operations as metadata reading, database objects creation, modification or deletion. DDL Utils enables easy and simple reading of database model into object oriented model. Because of this, there can be obtained necessary information about database and its tables. The main usage of this library is creation of appropriate log tables (required for asynchronous replication) — for original database tables.

3.2.2. Hibernate

Hibernate package realizes relational database mapping to its object equivalent. It gives easy database access, reduces application code length and eliminates necessity of SQL statements creation. Hibernate implements the universal data access layer, which is separated from application business logic. It is used for accessing (read/write) replicator system tables. Because Hibernate supports various database platforms — it gives much flexibility.

3.3. Structure and processing

There can be distinguished several parts in replication system:

— replicator application;

— replicator database part — includes its system tables;

— user database part — includes user tables, taking part in replication process.

Replicator application contains many modules, which include classes responsible for performing various tasks (Fig. 7).

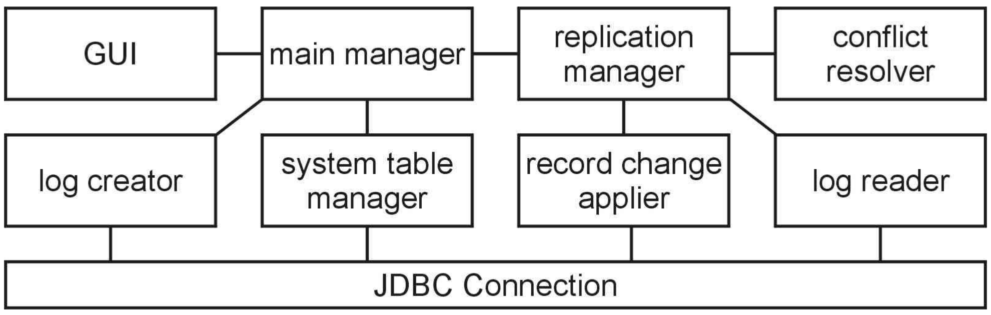

Fig. 7. Replicator structure — modules

To execute a replication process, there must be completed the following steps in each node (database):

1) database preparation;

2) publication and subscription of database tables;

3) synchronization execution.

The first phase includes creation of replicator's system tables (see Table) in a given database. This step is realized automatically by Hibernate environment. These tables contain informations about published and subscribed tables and remote databases which are taking part in replication process.

Replicator system tables

Table content	Table fields
log table list	log table id, original table name, log table name, trigger name, stored procedure name, information about publication
remote databases list	remote database id, host address, database type, database name, login, password, port, instance name
subscriptions list	remote table id, remote table name, remote database reference, local table name, index of last imported change identifier
main table log	change id, change kind, change time, log table id
dedicated table log	change id, record fields values before and after change

In the second step, there are performed publication and subscription of database tables. Publication of tables makes them available to other nodes (databases) for subscription. Subscription of table means registering for selected publication, which enables import of data from remote databases. All modifications performed on published or subscribed tables must be captured and saved. They are captured by table triggers and saved to appropriate log tables. Triggers are created on original database tables, which are selected by user for replication. Log tables are special replicator's system tables, designed for storing modifications. We can distinguish two kinds of log tables:

— main log — occurs in single number;

— dedicated table log — occurs in plural number — one log for each published/subscribed table.

Main log contains informations common to all tables. Dedicated table log contains informations specific for a given table, which it concerns.

Dedicated log is created separately for each published or subscribed table with usage of DDL Utils library. SQL code of triggers and stored procedures is defined in parametrized text files. This separates SQL code from application code, so it can be easily modified without recompilation requirement.

The third stage is the database synchronization process. The current node reads all changes from log tables which are subscribed. There is established connection with each remote database, which is defined in replicator's system table. After connection establishing, there occurs reading of modifications executed on tables which are subscribed. When all modifications from remote databases are read, the synchronization process in a local node (database) begins. Because the possibility of replication conflict occurrence, there are applied several methods of conflicts resolving, mentioned earlier.

Below there is presented a diagram of replicator operation (Fig. 8).

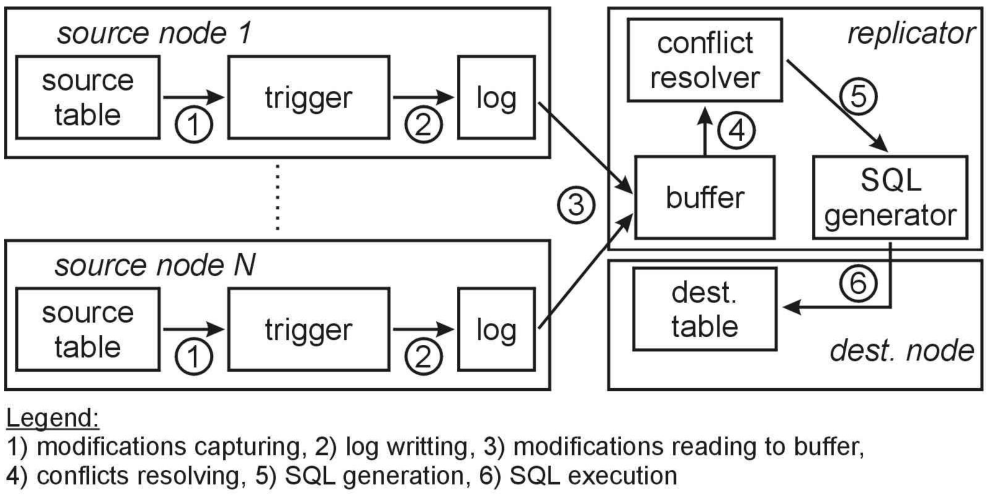

Fig. 8. Replicator operation

4. Summary

Database replication is currently in very dynamic development phase. There are still created new models and replication techniques, which offer possibilities that were impossible so far. The leading solutions existed few years ago, nowadays are obsolete and unusable. The globalization of informatic systems and commonly used distributed systems is undoubtedly a stimulus for replication developing. The main goal is developing such techniques which will give well consistency and integrity with preservation of good efficiency and scalability. The Universal Database Replicator, proposed above, was tested on various platforms with distributed database systems: Sybase, PostgreSQL, MS SQL Server. There was shown efficiency of asynchronous replication mode. Usage of Java platform with Hibernate and DDL Utils library enables replicator independence from database and operating system platform.

1. *Manassiev K:* Scalable and Highly Available Database Replication through Dynamic Multiversioning. — University of Toronto, 2005. — P. 87. — http://www.cs.toronto.edu/~kaloianm/docs/ut-thesis.pdf

2. *Guerraoui R., Oliveira R., Schiper A.* Atomic Updates of Replicated Data. — 1996. — P. 16. — http://citeseer.ist.psu.edu/guerraoui96atomic.html

3. *Bausch W.* Integrating Synchronous Update-Everywhere Replication into the PostgreSQL Database Engine. — Swiss Federal Institute of Technology (ETH). — Zurich. — 1999. — P. 55. — http://www.iks.inf.ethz.ch/publications/files/bausch_postgres_r.pdf

4. *Kemme B.* Database Replication for Clusters of Workstations. — Swiss Federal Institute of Technology (ETH). — Zurich. — 2000. — P. 155. — http://www.cs.mcgill.ca/~kemme/papers/phd-dina4.pdf

5. *Lin Y.* Database Replication in Wide Area Networks. — 2005. — P. 20. — http://www.cs.mcgill.ca/~ylin30/paper/proposal.pdf

6. *Wiesmann M., Pedone F., Schiper A., Kemme B., Alonso G.* Database Replication Techniques: a Three Parameter Classification. — Swiss Federal Institute of Technology (ETH). — Zurich. — 2000. — P. 10. — http://citeseer.ist.psu.edu/wiesmann00database.html

Received 06.03.2006