Transbase®

WHITE PAPER

Update Distribution and Replication

(Version 6.9 and later)

1. Terminology

Both Transbase® and Transbase® CD provide mechanisms for distributing changes made to a server database to client databases. However, the motivation and the goals are different.

While Transbase® CD provides several methods to distribute updates, Transbase® also provides replication support for online databases aiming particularly to increase database availability. The replication methods could also be used to distribute updates from a single master database to several slave databases.

In the following, we describe both approaches and their particular advantages for the problem of periodic updates.

2. Prerequisites

In order to distribute or replicate updates into client databases, several prerequisites must be matched.

First of all, the Transbase® authoring database must be changed as few as possible; in particular, primary keys that are used to identify rows in the database must be kept as stable as possible. If the database (including primary keys) would be generated newly for each distribution, key stability is not given. A single delete of a record may change all keys requiring the whole database to be updated. Thus, updates would become so large that they cannot be published online.

Instead, the Transbase® authoring database should be kept online and any updates should be applied to this database as they occur. E.g. to change a price for a part, the corresponding row should be updated (not deleted and inserted), and no other rows should be touched. To insert a new part, a new primary key should be selected without changing keys of existing parts. To delete a part, the primary key of this part should not be reused.

If this approach is followed, Transbase® keeps track of the changes and, at any appropriate point in time, will be capable to publish all updates in a single operation so that they can be transferred and applied to client databases. The volume of such updates obviously depends on the number and amount of changes. The more frequently updates are published, the smaller the volume typically will be.

3. Update Distribution using Transbase® CD

Transbase® CD is used to run databases on compressed read-only files (so-called ROMfiles). In addition, it provides a mechanism to update those databases virtually by relocating pages from read-only files to read-write files.

Updates can be replicated from server to client side in a compressed form, too. Various techniques are available here and are described below.

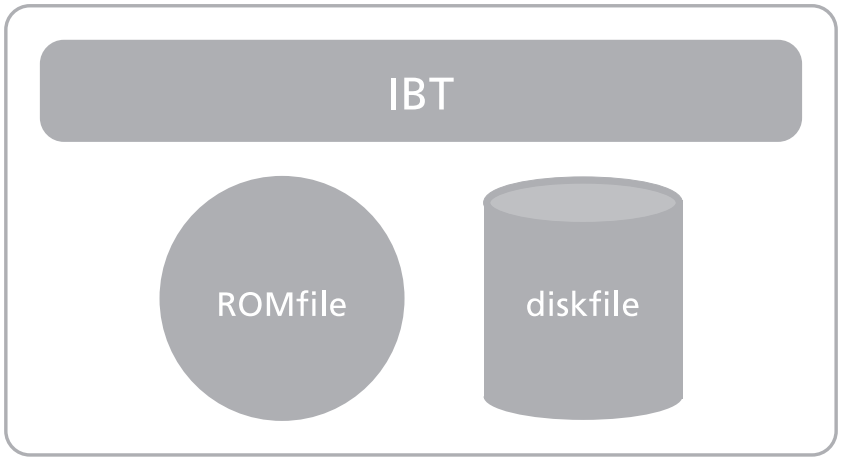
The replication goals are:

- Highly compressed volume of updates in order to distribute them online
- Fast application of updates at client sites
- Robust processes (updates may be skipped, …)
- Database stability

3.1 Updates in Transbase® CD

Transbase® CD can update databases even if they are on read/only media or in compressed files (so called ROMfiles). This is achieved by adding a read/write file (so called diskfile) to the database where all updates go to and a data structure that records which page has been updated and where it has been relocated. This data structure is called IBT (indirection B-tree) and keeps track of updated pages. It contains one record for each updated page (app. 12 bytes).

Initially, IBT is empty. Once a page is updated for the first time, a new entry is generated that maps the pair (original page number, original segment number) to a new page number on diskfile. For each page access IBT is searched first to see whether the page is relocated or not. If it is not relocated (i.e. if the IBT contains no entry for this page) the original page is fetched from read/only media; otherwise the relocated page is fetched from read/write disk. If the same page is updated several times, no further changes to IBT are required. The following picture shows the structure at retrieval site:



Picture1: Transbase® CD

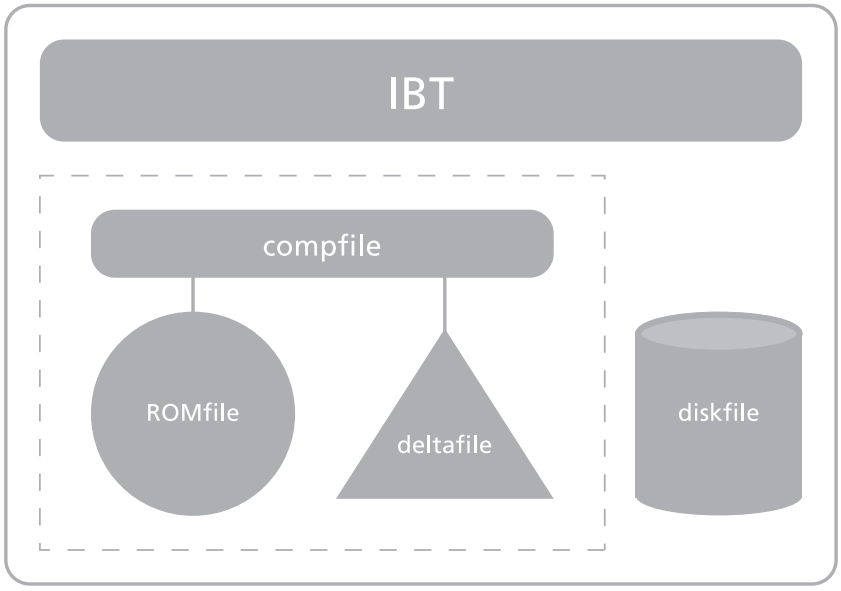
All changes in IBT are executed under transaction control, i.e. IBT is locked eventually thereby limiting concurrent update operation on Transbase® CD databases. In particular, changes to IBT are committed with the requesting transaction. Therefore the information in IBT is persistent like any transactional update.

As the relocated pages may be updated multiple times they must be stored in an uncompressed way. By using this mechanism heavily, database size may increase significantly compared to the originally compressed size. The amount of local storage may limit the updates that can be received by a database.

In many cases, however, Transbase® CD is used to distribute updates periodically. Updates can be prepared at server site and shipped as a set of files to the retrieval sites where they are simply „included" into the database.

Those updates can be and typically are compressed. A further data structure is provided that, for each compressed file, records which page is located at what offset. This data structure is called compfile.

On top of that, on-site updates can be performed as before by adding a diskfile along with an IBT. The structure is sketched by following picture:



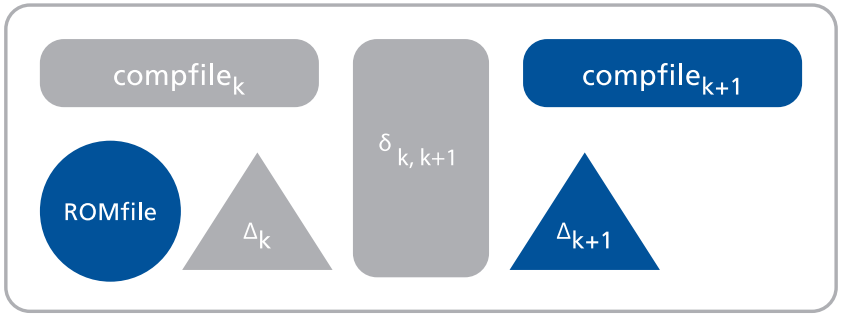
Picture 2: Transbase® CD Delta updates

When this procedure is iterated, it becomes clear that the first delta contains only necessary pages, while later deltas contain many pages that have already been contained in previous deltas. This is obviously not optimal in terms of transfer volume and therefore has been the reason to develop so-called transitional deltas.

3.2 Transitional deltas

In the following we will denote by Δ_1, Δ_2, … the set of pages that have changed, compared to the originally distributed romfile. If we further define δ_{12} as the set of pages contained in Δ_2 but not in Δ_1, we can write $\Delta_2 = \Delta_1 + \delta_{12}$, $\Delta_3 = \Delta_2 + \delta_{23} = \Delta_1 + \delta_{12} + \delta_{23}$, and so on.

The principal improvement of transitional deltas is that only $\delta_{n-1,n}$ is transferred to the client, so the transfer volume is minimized. However, the client has to generate Δ_n by merging romfile, Δ_{n-1} , and $\delta_{n-1,n}$. This operation may take some CPU and IO resources. After Δ_n is produced, Δ_{n-1} may be deleted (by default, it is kept, however). But for the interim both files are needed. The process is sketched by the following picture:



Picture 3: Transition from k to k+1
After the transition, the gray files may be deleted

It is obvious that a client that has skipped some deltas may directly upgrade from Δ_{n-k} to Δ_n for some k>1 when this specific delta $\delta_{n-k,n}$ can be provided.

Transbase® CD provides tools to merge a sequence of $\delta_{k-1,k} \circ \delta_{k,k+1}$ into a single delta $\delta_{k-1,k+1}$. Thus, it is easy to provide the newest deltas for each possible client state, namely: δ_{1n} , δ_{2n} , δ_{3n} , …, $\delta_{n-1,n}$

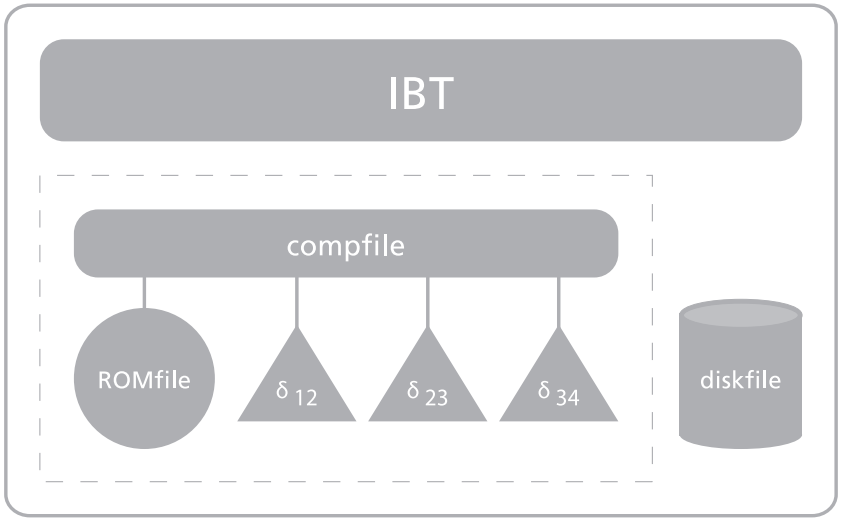
When a new delta $\delta_{n,n+1}$ is generated, this delta is applied to the sequence and produces $\delta_{1,n} \circ \delta_{n,n+1}$, $\delta_{2,n} \circ \delta_{n,n+1}$, $\delta_{3,n} \circ \delta_{n,n+1}$, …, $\delta_{n-1,n} \circ \delta_{n,n+1}$ yielding $\delta_{1,n+1}$, $\delta_{2,n+1}$, $\delta_{3,n+1}$, …, $\delta_{n-1,n+1}$.

Together with the delta $\delta_{n,n+1}$ the new sequence is complete and can bring any client from any state to the newest state n+1.

The main disadvantage of this transitional delta technique is the need of CPU and IO resources after reception of a new delta file. This leads to the development of so-called generation deltas.

3.3 Generation deltas

The main disadvantage of transitional deltas can be avoided by omitting the construction of a single deltafile containing all deltas because this takes local CPU and IO resources. Instead, the compfile is modified that it knows for each page where to locate it: either in ROMfile or in one of the $\delta_{k, k+1}$ files for some k. This behavior is shown by the picture below. As before, a diskfile can be added along with an IBT to accept local changes to the database:



Picture 4: Transbase CD® generation deltas

Note that all $\delta_{k, k+1}$ files must be kept. Technically, those files are stored in a directory together with the corresponding compfile. However, only the compfile of the newest δ file is being used.

Also note that a database can be reset from a state k to any previous state j<k by simply deleting the directories for state >j and using the compfile of state j. This technique minimizes both the transport volume and the time and local resources to update a database from state k to k+1.

The performance overhead caused by the n-way branch of compfiles is not significant.

4. Replication using Transbase®

A completely different mechanism is available for Transbase® but may be applicable for periodic updates as well. The original motivation, however, was to provide improved backup features for Transbase®. This mechanism has been extended to support replication on a physical (i.e. page) level. In particular, one master database is able to replicate its updates to one or many slave databases. The goals of this kind of replication are:

- Keep a database mirror as much as close to the original database to allow an extremely fast hand-over in case of failures
- Try to minimize any data loss in such cases

For deployment databases this mechanism could be used to distribute updates regularly into slave (retrieval) databases. However, there are subtle differences that have to be considered.

In the following we use the terms master database and slave database rather than editorial and retrieval database as used with Transbase® CD.

4.1 Prerequisites

There are three essential prerequisites to be observed:

- The slave database must be strictly read-only.
- Master and slave databases must be operated with „delta logging“ (rather than „before image logging“ which is the default for Transbase® CD).
- Master and slave databases must run on the same CPU architecture, e.g. both on little-endian or both on big-endian architecture.

The latter could be released, if needed, by changing the physical page structure of Transbase® databases into a platform-independent shape (as in Transbase® CD). Currently, the database format is platform-dependent.

4.2 Description

When the master database is updated a so-called delta log records the changes in a page-oriented manner. In particular, the log contains the following information:

- start of a transaction
- end of a transaction (commit/rollback)
- allocation of a new page
- deletion of a page
- update of a page

The most frequent records are the update records. To keep them as compact as possible not the whole (new) page is recorded but a bitwise delta between the new version and the old version of the page. For the xor operation, the following equations hold:

delta = old **xor** new
new = old **xor** delta
old = new **xor** delta

Therefore, the log can be used as a redo log as well as an undo log. For replication purposes, the log almost always is used as redo log. Only when aborted transactions are part of the log, it is also used for undo.

The master database continuously produces log records as update transactions are being processed. The log has to be written persistently to disk on two occasions:

- when a transaction ends (commits or aborts)
- when a page is written to disk before it is committed (typically because the database cache is too small to hold the update).

The replication component of Transbase®, called tbrepl, is an additional process on the master database. It can be started at any point in time (usually at startup of the database) and it can stop at any point in time, either deliberately or occasionally.

When the process is up, it can be connected by slave databases in order to pass the log from a given point in the log. When the end of the log is reached, the process waits and provides the next portion upon end of each transaction finished at master.

At slave databases, the same process tbrepl can be started (again at any time). Once it connects to the master tbrepl process, it receives logs and immediately processes them on the slave database by applying the page deltas. Whenever it receives a commit record it commits the changes. Therefore the slave database is only a little behind the master database.

While updates are applied, the slave database may be operational without restriction. Updates will become visible once they are committed. If the log contains several transactions, also several intermediate states become visible. If the log contains only one large transaction, no intermediate states will be visible: before the transaction ends, the original database state will be visible, after it commits, the new database state will be visible. Note here, that active read only transactions may hold locks on the database that might prevent the update transaction from committing its changes.

The slave tbrepl process can be stopped intentionally or not at any time; after restart it will continue from where it stopped. Transbase® guarantees that updates will be processed once and only once. The proper sequencing of logs is also checked by Transbase®.

4.3 Performance

Applying a log is a comparably fast operation. In particular, the log is being processed already, while it is received. Usually the log is processed much faster than the original transaction as no SQL statement processing is involved, and only the changes have to be processed.

The volume of the log to be transferred from master to slave is reduced by compression. However, pages that are frequently changed (so called hot spot pages) will contribute much to the size of the log. Those multiple changes could be summarized at master side before the log is being transferred, but those logical compression currently is not available.

Contact

Transaction Software GmbH
Willy-Brandt-Allee 2
81829 München

Tel.: +49 89 / 627 09 - 0
Fax: +49 89 / 627 09 - 11

info@transaction.de
www.transaction.de
www.transbase.de