DaRe explained

The official

DaRe (Data Replication)

documentation

Jarle (jgaa) Aase
Jgaa’s Internet

jgaa@jgaa.com

DaRe explained: The official DaRe (Data Replication) documentation

by Jarle (jgaa) Aase

Published v0.01.03 2002/06/16

This manual is produced using DocBook (http://www.docbook.org/). The original manuscript is included in the source code distribution. You can browse the on-line version at dare.jgaa.com (http://dare.jgaa.com/) or download the Adobe pdf version (http://dare.jgaa.com/dare-manual.pdf).

Table of Contents

List of Tables

About this Manual

1. Purpose / Scope of this Document

This is the official documentation on *the DaRe program*. It is primarily targeted against system administrators.

2. Copyrights and Trademarks

Chapter 1. Introduction to DaRe

1.1. Background

DaRe is a handy program to replicate data from one database to another. It was originally written to replicate data from some Microsoft SQL Server 2000 databases in enterprises to MySQL under Linux. The goal was to allow easy and secure access to selected tables from Apache/PhP. The advantages by this approach was to save money on expensive Microsoft SQL Licenses needed if the data was to be used from IIS, and to improve the security. Apache is harder to break into than IIS, and even if someone manages to break into the webserver - the original database is still safe.

DaRe is written to be platform-independent. It is however currently only tested for replication from MS SQL Server to MySQL. It is design to make maintainance easy, and will create special tables on the source database and the destination tables automatically. It can also do a fresh re-synch where all the data is refreshed.

1.2. How it works

DaRe is actually quite simple to understand. When it is started, it reads the configuration file, and tries to open the ODBC connections specified. Then it checks if a special table, specified in the configuration file, exists in the source database. If it don’t exist, it is created. This table, usually named `DareRepl` is used to store a timestamp value and the key identifier for each data-row in each table that is replicated. When this is done, DaRe investigates the source tables specified in the configuration file, and checks if the tables and columns that are to be replicated, exists in the destination database. It they don’t, they are automatically created. When all this initialization is completed, DaRe starts on it’s real task: It compares all the rows in all the tables with the key-values stored in the special `DareRepl` table, and sends all new or modidied data-rows to the destination database. Then it deletes all data-rows on the destination database that don’t exist on the source database.

How efficient this is depends on the size of the source database. Since each data-row is accessed and compared during the replication process, the operation takes time and resources on the server on large databases. If you experience problems with the performance, you may want to use a view in stead of a table on the destination database to limit the number of records and columns to what you actually need on the destination database.

1.3. Platforms

The following platforms are supported:

- Windows NT, 2000 and successors. Access to MySQL is done trough the MySQL ODBC driver for Windows.

1.4. Licensing

The program is released under the Gnu Public License (http://www.gnu.org/copyleft/fdl.html) (GPL). The program and source code is in other words free. There is no fee, no user-registration, no royalties, no nothing.

1.5. Download

The latest version of DaRe can be downloaded from sourceforge.net/projects/dare/ (http://sourceforge.net/projects/dare/) The file section there contains both binary and source tarballs. The files can of course also be downloaded from ftp.jgaa.com (ftp://ftp.jgaa.com).

1.6. Obtaining support and reporting bugs

DaRe is supported at support.jgaa.com (http://support.jgaa.com/). This site has a modern bug-reporting and bug-tracking facility, and you can also suggest features there, and subscribe to mailing lists. There is also a newsgroup: alt.comp.jgaa where you can ask for support.

If you have discovered a bug that can compromise the security in DaRe please contact <`jgaa@jgaa.com`> directly. If you don’t get a reply within 12 hours, please resend the message.

1.7. Author

DaRe is written by Jarle (jgaa) Aase, best known for the original free FTP server for windows, War FTP Daemon. You can visit my homepage at www.jgaa.com (http://www.jgaa.com/). The development is sponsored by Lerøen Consultinng AS www.ldp.no (http://www.ldp.no).

Chapter 2. Compilation and installation

2.1. Source code or precompiled binary?

DaRe is distributed as source code, and as a compiled binary program for Windows. If you happen to use Windows, this will be the most convenient way to install the package. If not, you are probably using some Unix/Posix compliant operating system, - and you need to compile the source code and sort out compatibility issues.

2.2. Compiling under Windows

Normally, you will use the binary Windows distribution. If you by some reason need to compile your own binary, you need Microsoft Visual C++ version 6, SP 3 (or better).

Download the latest source code from sourceforge.net/projects/dare/ (https://sourceforge.net/projects/dare/) and unpack it to a suitable location on your disk. Open the dare project with Microsoft Visual C++. Compile either the `Win32 Debug` build or the `Win32 Release` build, depending on your requirements. If you compile the `Win32 Debug` build, you can trace the program 100% in the debugger.

Chapter 3. Configuration

3.1. Overview

DaRe is implemented as a command-line utility that reads a configuration-file when it start us, and then replicates whatever tables that are specified in that file. It connects to the databases with ODBC. I've used ssh to create a secure tunnel for the ODBC connection to the MySQL server - but any encrypted tunnelling software or VPN protocol shold be fine. You can of cource also use an unsecured TCP/IP connection - but I would not reccommend it as it will open for all kinds of security problems.

3.2. The Configuration file

DaRe depends on a configuration file; a textfile with simple directives. If you are familiar with Windows, you can think of this file as an extended `.ini` file. If you are familiar with Unix, you know what a configuration file is ;)

The configuration file can be specified by a command-line argument when DaRe is started (**--conf-file=path**). If it is not specified, DaRe will look for it as `...Application Data\dare\dare.conf` (Windows), or `~/.dare.conf` (Unix).

The configuration file consist of two sections; the general section, and the rule/Table section.

3.2.1. General Section

Table 3-1. Options in the general section

log-file	The path to a log-file. If this option is set, DaRe will maintain a log. If not, no log will be written (unless the --log-file command-line argument is given when DaRe starts up).
src-odbc-connect	ODBC connect statement for the source database. Can be something like: DSN=data; where data is a local (system) ODBC name.
dst-odbc-connect	ODBC connect statement to a remote database; typically a MySQL ODBC connection.
dare-table	The name of a table to use for syncronization. This is a table exlusively used by DaRe to store timestamp information (when a local record was last updated). DaRe will create the table if it don’t exist. The only thing you have to worry about is that the name is not in use by your application(s).
sql-timeout	Sets a time-out on the SQL queries to the server. I have experienced problems where Microsoft SQL Server have deadlocked DaRe and other processes - without detecting them. We actually had to use the management console to resolve the conflicts(!). This option has solved this problem.

3.2.2. Rule/Table Section

The `[Table:{name}]` sections are used to specify what tables to replicate. The `{name}` is the name of one table. There can be any number of table-sections. If the `{name}` is using characters other than a-z, digits and underscore, the name is translated to a safe substitute for the destination table. This means that you can replicate names with spaces, national characters and all sorts of dollars and other signs database maintainers put into their databases to look skilled.

Table 3-2. Options in the Rule/User section

table-id	This is a unique number identifying this table. The number is used internally by DaRe. I usually start by numbering the first table 1, the second 2 and so on.
key-col	The name of a column in the table that can be used to identify the data-row. This column must be indexed (if not, things will get really slow) and unique. The data-type must be integer. The reason that DaRe require a integer key-column is speed. Comparing and copying large character or binary key's would slow things down. If you don't have an integer key-column in your table, add one with auto-numbering. Your applcation can then use the old key-column, and DaRe can use the new one.
timestamp-col	The name of a column of type timestamp. If your table don't have a timestamp column, just add one. The timestamp is used to determine if the data-row must be replicated.

col-names	The names of the columns to replicate. Pattern-matching works, and a single star (*) instructs DaRe to replicate everything (except the timestamp column).

3.2.3. Sample configuration file

```
# Configuration sample fo DaRe
# by Jarle (jgaa) Aase, 2001

[General]
    log-file = dare.log

    # Information about the source configuration
    src-odbc-connect = DSN=indata;

    # Information about the remote server
    dst-odbc-connect = DSN=outdata;

    # The syncronizing table on the source database
    dare-table = DareRepl

    # Set a sane timeout on SQL queries.
    # I have experienced deadlocks
    # on MS SQL Server 2000, and the timeout
    # should resolve this.
    sql-timeout = 30

[Table:Sfi]
    table-id = 1
    key-col = Nummer
    timestamp-col = upsize_ts
    col-names = *
```

```
[Table:Tilbud]
    table-id = 2
    key-col = TilbudId
    timestamp-col = upsize_ts
    col-names = *

[Table:Jobb]
    table-id = 3
    key-col = JobbId
    timestamp-col = upsize_ts
    col-names = *

#[Table:Element]
#    table-id = 4
#    key-col = Nummer
#    timestamp-col = upsize_ts
#    col-names = *
```

> **Note:** The lines may be wrapped in this manual. In the configuration-file, the "token = value" sequence must be on one single line. This also applies for the "[User: ...]" sequence.

Lines beginning with a grid (#) are treated as comments and ignored.

Appendix A. Command Line Syntax and Options

A.1. Overview

DaRe understand Posix-style command-line options. In environments with a complete Gnu c library (like Linux), the Posix compatibility should be pretty good - but under Windows and other systems that don't support the getopt_long() library function, DaRe use a simple substitute I wrote some time ago. This substitute understand the `--option` and `--` flags, but will not reorganize options. In English, this means that options must be given before other arguments.

DaRe usually only needs the option to specify a configuration-file. You can however override some of the options in the configuration-file from the command-line, and you can specify verbose reporting to ease troubleshooting.

A.2. Getting help

Use **dare** `--help` to get a brief list and explanation of all the supported command-line options.

```
C:\devel\current\dare\Release>dare --help
 dare [options]
Options:
 -c  --conf-file=name   Specifies the configuration file to use
 -v  --version          Show version number and quit
     --verbose          Turn on verbose mode
     --debug            Turn on debug messages
 -l  --log-file=name    Name of logfile
 -h  --help             Show this help
     --recreate-all     Re-create tables and refresh all data
```

```
-t  --sql-timeout=#    Timeout in seconds on SQL statements.
-L  --license          Show licensing terms (GPL)
```

The actual options that are available may vary from version to version, and from operating system to operating system.

A.3. The options

Table A-1. Command line options

--conf-file	Specifies the configuration-file to use.
--version	Prints the program name and version number to the console and quits.
--verbose	Prints more verbose information to the console.
--debug	Prints lots of information to the console. Used to debug the internals of the program.
--log-file	Specifies a log-file. If not set, no log will be written, (unless the log-file directive is specified in the configuration file).
--help	Prints a short help-text to the console.
--recreate-all	When this option is activated, the target-tables are deleted and recreated, and all the data is replicated. This can take some time on large databases.

--sql-timeout	Specifies the max duration time for SQL queries in seconds. If the time-limit is reached, the query is aborted, and DaRe must be started again to do a replication. This option was added to handle a bug (deadlock) in Microsoft SQL Server 2000.
--license	Prints the GNU license and exits.

Appendix B. Firewalls

When it comes to legitim network traffic, firewalls are hell. Many new protocols today encapsulate themselves into the http (www) protocol to bypass firewalls. This may seem like a good idea, initially - as one avoid some problems with troublesome firewall administrators. But it also undermines the security, since the same firewall administrators loose the ability to decide what to let through to the internal network. DaRe makes no attempts to masquerade itself as something else. It use ODBC connections, and connects to the databases trough the means specified in the ODBC configuration. I usually connect trough a ssh port-forwarding tunnell to handle authentication and encryption. The firewalls on our systems are configured to allow ssh connections between the server where DaRe runs, and the Linux server with MySQL.

If you use **ssh** to secure the connection, you must open port TCP/22 in the firewalls. If you use some kind of VPN and just wants to connect to MySQL directly trough the ODBC connector, you must open port TCP/3306.