Progress®

Learn About Pro2®

OpenEdge®

Table of Contents

Copyright

© **2019 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.**

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Deliver More Than Expected, Icenium, Kendo UI, Kinvey, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, Rollbase, SequeLink, Sitefinity (and Design), Sitefinity, SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. Analytics360, AppServer, BusinessEdge, DataDirect Autonomous REST Connector, DataDirect Spy, SupportLink, DevCraft, Fiddler, JustAssembly, JustDecompile, JustMock, Kinvey Chat, NativeChat, NativeScript Sidekick, OpenAccess, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

February 2019

Last updated with new content: Release 12.0

Preface

Purpose

This manual explains the basic concepts of OpenEdge® Pro2. It provides basic details about Pro2 like what is Pro2, its uses, features and its advanced capabilities. It also contains information on Pro2 architecture over LAN and WAN. Additionally, it discusses disaster recovery planning for Pro2 and the role of the Replicated SQL database.

What is Pro2 introduces you to OpenEdge Pro2 and discusses how this tool is used. Subsequent topics provide additional information about replication, real time replication, architecture, concepts, advanced capabilities, and disaster recovery.

For the latest documentation updates see the OpenEdge Product Documentation Overview page on Progress Communities:
https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx.

Audience

This book is intended for database administrators (DBA's) and consultants to help them to have an easy and simple way of configuring the replication process. It assumes a fundamental knowledge of both OpenEdge and DataServer for MS SQL and Oracle.

Organization

What is Pro2

Describes what is Pro2, where it is needed and what is it used for.

Pro2 architecture

Describes the basic architecture of Pro2 over LAN and WAN.

Pro2 concepts

Discusses the basic concepts of pro2 which include Pro2 replication triggers, processor procedure generator, replication processor, repl database and its properties. It also discusses about change data capture (CDC) and second pass replication.

Pro2 advanced capabilities

Describes the advanced features of Pro2 which include various customisations in Pro2 and the replication process. Apart from that, it also discusses vasious aspects of bulk-loading.

Disaster recovery planning

Discusses the impact on business if the database or server(s) or both are lost. It also describes how to prevent the same.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library, including information about the following:

- Using ABL documentation
- Examples of syntax descriptions
- OpenEdge messages

1

Learn about Pro2

OpenEdge Pro2 provides replication of OpenEdge databases to Microsoft SQL server, Oracle, or other OpenEdge databases. Pro2 supports enterprises in delivering the critical business data needed to support analytics and business intelligence initiatives. It removes connectivity limitations without disruption to normal business operations or risk to the transactional database. Pro2 provides maximum flexibility and excellent performance since Pro2 is native to OpenEdge.

Pro2 includes a web user interface which allows intuitive configuration and monitoring of performance data and analytics. Pro2 improves productivity by providing functionality that allows the creation of new replication processes and management of bulk load from the web user interface. Pro2 can be used with the OpenEdge database Change Data Capture (CDC) functionality to deliver near real-time replication.

This topic describes the architecture details of Pro2 along with the concepts and terminology needed for proper setup and use.

For details, see the following topics:

- What is Pro2?
- Pro2 architecture
- Pro2 concepts
- Pro2 advanced capabilities
- Disaster recovery planning

What is Pro2?

OpenEdge Pro2 is a lightweight, configurable tool that provides near real-time replication of an OpenEdge database to another database, typically Microsoft SQL Server, Oracle database, or another OpenEdge database instance). The business solutions that may require near real-time replication include:

- Heterogeneous application integration
- Reporting server
- Data archival
- Business intelligence

Pro2 can replicate one or more OpenEdge databases to one or more target databases. You can replicate duplicate entire databases, or limit replication to only specific tables and fields.

Real time replication

Pro2 allows you to configure the time interval between the replication sweeping operations. You can customize the interval in real time according to your needs. For example, every "x" seconds, hourly, twice a day, or daily depending on your specific business problem.

Additionally, you can suspend the entire replication process temporarily. You may also choose to discontinue specific table replication temporarily if needed. Usually, this is done to reduce load on the production system during peak timings such as end-of-quarter processing.

You can restart the suspended replication after the peak timings have passed. However, while the replication process is suspended, the users of the target system do not have access to the records that have been updated in your source OpenEdge database until the replication process is restarted and the replication queue is completely swept.

What is replication?

The data replication process uses a single replication table consisting of minimal data and indexing and a single sequence for process control. No raw data is captured during database events (Record, Write and Delete events). However, the replication triggers fire and capture the updated record information to any database replication table (queue) as the create, update and delete events take place. Capturing of database, table, transaction, and event type information is responsible for significantly reducing the production (replication) overhead.

All replication data captured is maintained in the following database tables:

- ReplControl
- ReplCustDefs
- ReplCustFlds
- ReplDBXRef
- ReplFieldXref
- ReplProperties
- ReplQueue
- ReplTableXRef
- ReplThreadControl

These tables can be in a standalone repl database or embedded into one of the source OpenEdge databases. There is a replication processor that cycles through the replication records periodically based on user configuration and replicates the table data directly to the target database via the OpenEdge DataServer.

Pro2 architecture

Pro2 architecture varies over local area network (LAN) and wide area network (WAN). It includes:

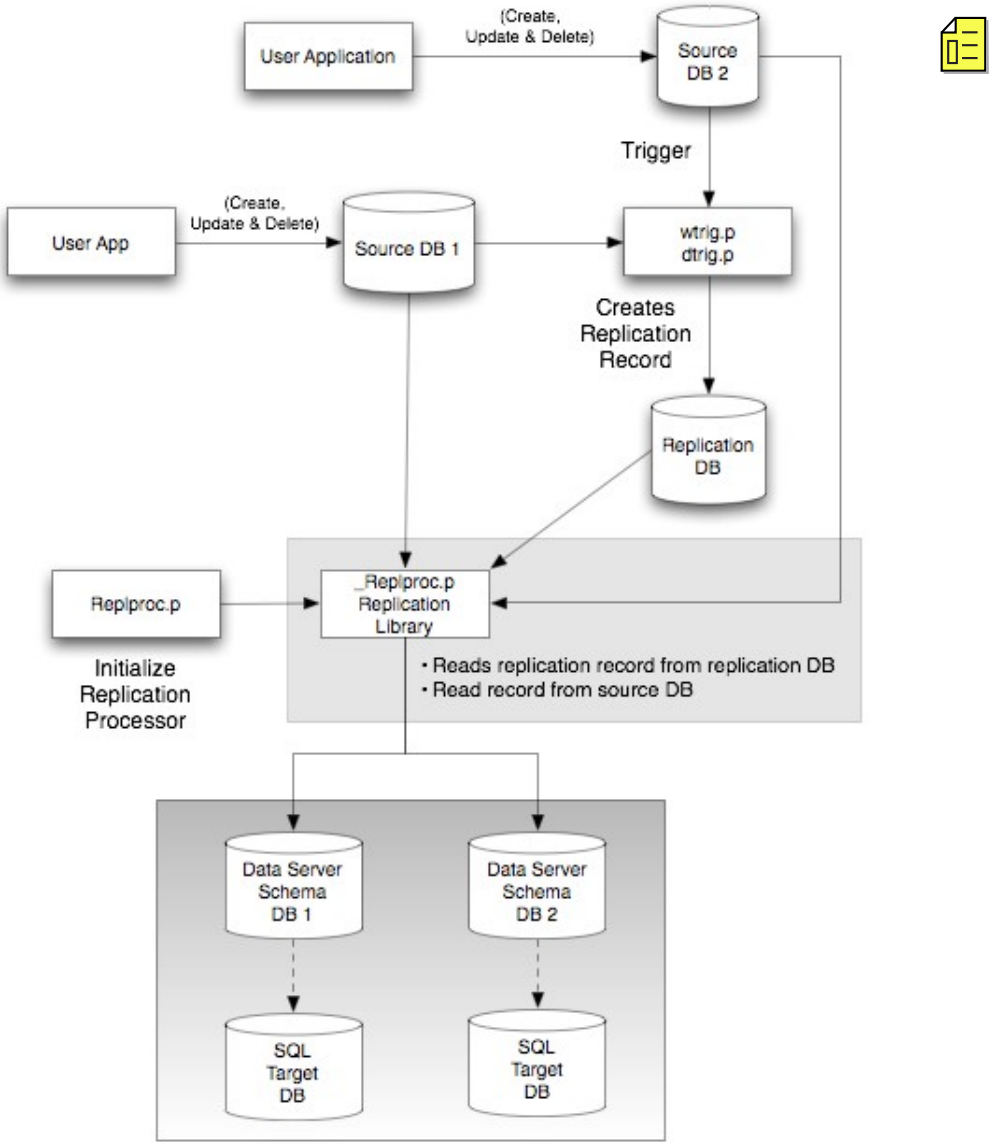
- An OpenEdge database (also known as source database). There is usually a user application tied to the source database.
- A repl database where the Pro2 database tables are stored. These tables control replication.
- A target database (MS SQL Server, Oracle database or OpenEdge database)

Pro2 architecture over LAN

The replication functionality consists of the following components:

- A source database which is an OpenEdge database. The user application is tied to this database.
- A repl database which stores database tables. (For more details, refer to the What is replication? section.)
- A schema holder database which contains information about the data definitions for one or more databases.
- Pro2 application
- A target database which can be either MS SQL Server, Oracle or an OpeEdge database.

Note: As an alternate setup, the repl tables can be embedded directly in the source database.



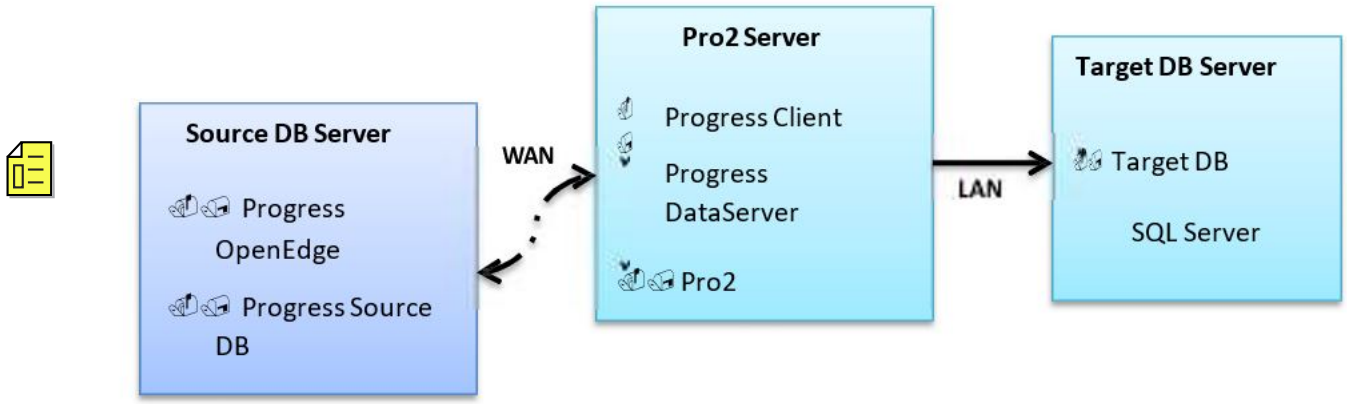
Pro2 architecture over WAN

When the target database is in a different geographic location than the source OpenEdge database, configure Pro2 to include a Progress Application Server on the source database server in addition to the standard configuration on the Pro2 server.

Instead of running the replication procedures using typical client/server mode connections to the source and repl databases, Pro2 sends requests to PAS for OpenEdge on the database server which returns data to the calling procedure on the Pro2 server.

Note: The Pro2 server must be on the same local area network as the target database server.

The following figure describes how this topic refers to each component of the Pro2 replication configuration:



Pro2 concepts

You need to understand the following concepts in order to use Pro2 and use it to its full capablities:

- Pro2 replication triggers
- Components of replication triggers
- Processor procedure generator
- Change Data Capture and its limitations
- Replication processor
- Processor library
- Target schema
- Repl database
- Repl properties and Repl queue
- Second pass replication

Pro2 replication triggers

Pro2 replication triggers are used with the source OpenEdge database. You can use these triggers to:

- Create the replication record based on the database, table, and ROWID.
- Record the type of event (i.e. create, write, delete the transaction number).
- Set the Applied flag to false (default).

All replication triggers are compression triggers by default.

Note: You cannot use the compression trigger template when auditing is implemented. In this case, you are required to override the default triggers.

Schema trigger definitions

Schema triggers are fired when schema-level objects (tables) are modified and when the user logon or logoff events occur. Some of these definitions are:

- CREATE—To create and enable a database trigger.
- DELETE—To remove a database trigger from the database.
- WRITE—To write and edit a database trigger.
- RE-DEL
- RP-WRI

The event type RP-DEL stands for REPLICATION-DELETE and RP-WRI for REPLICATION-WRITE, includes both create and update events. The procedure specified is run when this type of event occurs. You can get a trigger definition report from the Procedure Editor (The Procedure Editor is an OpenEdge ABL code editor). Run Tools→ Data Administration. From the Data Administration window, select Database → Reports → Triggers

An example of the output from the Progress trigger report is shown below. This example is of a Customer table in the sports database after Pro2 triggers have been inserted:

Table 1: Progress trigger report

Table/Field name	Event	CRC	Flags	Procedure
CUSTOMER	CREATE	No		C:\dlc101c\sports\crcust.p
	DELETE	No		C:\dlc101c\sports\delcust.p
	RP-DEL	No		.\bprepl\repl_d\dsports_Customer.t
	RP-WRI	No		.\bprepl\repl_w\wsports_Customer.t
	WRITE	No		C:\dlc101c\sports\wrcust.p

Alternatively, these trigger definitions can be displayed by the `replTrigDisp.p` procedure (to screen) and the `replTrigDisp2.p` procedure (to file).

Trigger procedures

The Pro2 trigger procedures are found by the client executable in a location specified in the PROPATH. The replication trigger location can be in the same directory as the other triggers (If there are other triggers defined for the database). However, these triggers must be already included in the PROPATH. You must take care to place the trigger procedures on the database server and on any application server boxes from which end users run the application.

Note that the pathname specified for the Pro2 replication triggers in the schema trigger definition is a relative path starting with bprepl. Typically, a bprepl directory is created with the following sub-directories in a location already in the PROPATH:

- repl_d (delete triggers)
- repl_w (write triggers)

In any alternate scenario, the PROPATH is modified to include the location of the bprepl directory.

When a replication trigger is defined in the schema for a table, the client attempts to run the specified trigger procedure. This happens when a client executable attempts to update a record in the source database.

The trigger procedures should be located on the database server if the user is connected as a local direct connect client (whether an end-user or an Progress Application Server servicing a remote client). Whereas, the trigger procedures should be on the client system if they are remote and are connected to the database via client/server.

For optimal performance, the replication triggers should be compiled (`. r code`). To compile the triggers, the standard OpenEdge application compiler is used in a Procedure Editor session connected to both the source database and the repl database.

Merged triggers

The Pro2 replication triggers need to be merged or integrated with the original triggers if the pre-existing replication triggers (REPLICATION-DELETE or REPLICATION-WRITE) exist in the source database. You can merge the triggers by adding a RUN statement in the original replication trigger procedure that calls the Pro2 trigger or by adding the Pro2 replication trigger logic to the original trigger.

Note:

The types of triggers in Progress are as follows:

- Primary triggers—These types of triggers are not to be integrated with Pro2 triggers. CREATE, WRITE, and DELETE are primary triggers and are not impacted by Pro2 replication.
- Replication triggers—These are executed after the primary triggers and need to be merged with the Pro2 triggers. These triggers are REPLICATION-DELETE or REPLICATION-WRITE.

Components of replication triggers

The following are the components of replication triggers:

- Definition of the replication trigger in the schema of the tables to be replicated that indicates which procedure is to be run whenever there is a Create, Update, or Delete event on a record. In the OpenEdge database architecture, the triggers are executed by the client executable (versus the trigger being executed by the database server executable as in most other database systems).
- The trigger procedure specified by the schema trigger definition. The Pro2 trigger procedures are very lightweight. They do not alter the application flow in any way and are transparent to both the application and the end-users.

In cases where replication triggers already exist in the source database for particular events and tables, the Pro2 replication logic needs to be integrated with or merged into the pre-existing triggers.

Processor procedure generator

The processor procedure generator creates a single procedure library that contains an internal procedure for each of the replication tables. These internal procedures encapsulate the replication logic specific to a specific table. The resulting procedure library is run persistently while the replication processor is running. If this procedure is run from the Progress editor, all databases, namely source database, schema holder and target database must be connected.

Additionally, create the following database aliases:

- SourceDB for the source database
- SchemaDB for the OpenEdge schema holder database.
- TargetDB for the target database (which can be an MSS SQL Server, Oracle or an OpenEdge database).

Change data capture and its limitations

OpenEdge provides support for a feature called Change Data Capture(CDC). CDC is an OpenEdge RDBMS feature that identifies and captures data that has changed in tables of a source database, as a result of CREATE, UPDATE, and DELETE (CUD) operations.

CDC is an industry term that describes the process of duplicating subsets of OLTP data in an external data source with a relatively up to date version of relational data. The OpenEdge implementation of CDC provides a flexible and scalable capture process to facilitate the data extraction, transformation, and eventually the loading of the data to an external data source.

Pro2 leverages this feature and replaces the ABL triggers with CDC. Pro2 CDC implementation also supports a feature called thread#0. When this feature is used, the CDC Admin thread converts all CDC_change_tracking records irrespective of which thread the table is mapped to. This provides the convenience of having a single CDC thread that converts all CDC records to replqueue records.

The CDC replication process is faster than using ABL triggers and can be managed online. This results in relatively low database downtime. The client application connection remains unchanged and there is no need to connect to the repl database. You can create a copy of the CDC batch program (CDCBatch.p) and append the file name with the thread number to run multiple threads. For example, CDCBatch0, CDCBatch1, CDCBatch2 and so on.

Limitations of using Pro2 with CDC

The limitations of using Pro2 with CDC are as follows:

- Pro2 handles only Level 0 policies and policies should be created with the suffix Pro2 to be recognized by Pro2. For example, **<tablename>_pro2.**
- Pro2 does not replicate any changes that are generated from Non-Pro2 policies.
- Pro2 uses the **_userMisc** field in the **change_tracking_table** to determine whether a record is read or not.

Replication processor

The replication processor, also known as the thread, is a batch program that periodically cycles through and processes any replication queue records that have not been applied to the SQL database. The name of the thread is the timestamp and the number of the thread.

The cycle period is controlled by the sleep interval set by the Settings button on the Dashboard window of Pro2 user interface. This value represents the number of seconds that the processor sleeps between cycles. If the value is set to 0 (zero), then the batch session exits and needs to be restarted to resume processing of replication records.

Replication log files

Each replication processor writes to two log files - replproc(YEARMMDD-thread#).log and repllog(YEARMMDD-thread#).log, that are located in the operating system directory specified by the `LOG_DIRECTORY` value in the properties table (by default bprepl\repl_log folder). Any errors encountered and general information such as cycle start and stop time and number of records processed are captured here.

These files should be reviewed periodically and require periodic truncation/deletion or archiving. When the logging level option is set to anything other than None, the replication processor generates basic log entries, such as when procedure libraries are executed and when the processor queue is started and stopped. Each level builds upon the previous level. For example, the Moderate setting logs errors as well as the basic entries generated by the Minimum setting.

Replication threads

There can be up to five separate threads of replication processors running. For each thread, a separate replBatch.bat from bprepl\Scripts is run (For example: replBatch.bat, replBatch2.bat, replBatch3.bat corresponding to each thread.) Tables are assigned to specific threads in the Properties window of the Pro2 web user interface.

Note: To increase the number of replication threads, please contact your administrator to modify your license and get additional threads.

For most implementations, one thread is typically enough to maintain pace with the changes to be replicated. For implementations with very high transaction numbers, tables can be spread across up to five queues to manage performance. Individual tables that have exceedingly high transaction volume or are of very high priority for users of the target database can be assigned their own thread.

Other reasons to have tables in separate threads include running static reports on the target database for a subset of tables. These tables are put in their own thread. When it is required to run the static report on the target database, replication for that thread is stopped. Changes for these tables continue to accumulate. When the report is completed, the replication processor for this thread restarts, catches up with the changes in the queue, and then resumes processing of new changes.

Split replication threads

You can split each replication thread into multiple split threads regardless of whether it serves replication of a single table or group of tables. You can split a single thread into a maximum of 10 split threads.

When you split a thread into n number of parts, it provides n-1 split threads. For example, if you split the thread 5 (replBatch5.bat) into four parts, it provides three split threads 51, 52, and 53 along with the original thread 5. This means, the four parts are threads 5, 51, 52, and 53. Pro2 determines the number of a split thread dynamically based on the `RECID` of the source record, and groups together all the split threads that belong to the same record or `RECID`. This allows you to run multiple processes without adding queue compression and prevents out-of-sequence processing of more changes to the same `ROWID` or `RECID`.

Processor library

The processor library is a set of procedures used to handle the actual replication of individual records in a given source table over to their corresponding target records. These procedures are generated as a part of replication process in the Manage Replication window of the Pro2 user interface. In the Manage Replication window, select Generate Code tab, select the required checkbox(es) for libraries to be generated and then click Generate Code. These procedures are stored in the bprepl\repl_proc folder.

To generate the processor library procedures, Pro2 evaluates the field and tables specified for replication and creates logic that executes writing the data to the target database.

Part of the process of generating the processor library is to create an individual "assignment" procedure for each replicated table. This assignment procedure gets stored in the bprepl\repl_inc folder using a naming convention of 'rp_<dbName>_<tableName>.i'. This solution enables for extensible functionality because users may add customized business logic to these assignment procedures to solve business problems.

For example, if the source records need to be accumulated in some way prior to replication, these assignment procedures can be modified to do so. You must take care while editing the assignment procedures. If not executed properly, there is a risk of breaking replication. Any customized assignment procedures should be backed up in the case that the Pro2 product needs to be reinstalled or reinitialized.

The replication procedure library (ReplLib.p) contains internal procedures and functions that handle general utility functions such as reading from and writing to configuration files, map files, and the replication properties and controls tables. It runs persistently and is added to either the session or procedure SUPER stack.

Target schema

Pro2 reads the source OpenEdge database schema and generates scripts that are then executed on the target database server to create the target schema. By default, all tables and fields are created in the target. The rest of the files (drop.sql, index.sql, trunc.sql, errors.log and warnings.log) are created separately. These files can be executed by the user as per their requirements. Typically, tables which are not replicated remain unmapped. Edit the generated scripts if you dont want the tables to exist at all on the target side.

Target side only tables and fields

The following tables are added to the target database for Pro2 use:

- **Pro2sql:** used by Pro2 to verify connection to target database.
- **P2mismatch:** used in customized implementations to monitor column truncation.

The following fields are added to all target database tables for Pro2 use:

- **PRROWID** —Unique key that corresponds to the source Progress ROWID.
- **Pro2created**—Date/timestamp record was first created
- **Pro2modified**—Date/timestamp record was last modified
- **Pro2SrcPDB**—Name of the source database
- **Progress_RECID**—Used by Pro2 for Progress versions before 10.2
- **Progress_RECID_Ident** —Used by Pro2 for Progress versions before 10.2

Repl database

Pro2 replication process uses the following OpenEdge database tables:

- ReplControl
- ReplCustDefs
- ReplCustFlds
- ReplDBXRef
- ReplFieldXref
- ReplProperties
- ReplQueue
- ReplTableXRef
- ReplThreadControl

These tables can be embedded in one of the source databases or they can be configured as a stand-alone OpenEdge database. See the Repl tables schema objects section in the Reference book for the description of the repl tables.

Stand-alone repl database

Standard OpenEdge database utilities are used to create the repl database. The structure file (.st file) and schema definition file (.df file) used to create the repl database can be found in the root Pro2 install folder. All end-user configuration files, parameter (.pf) files, and scripts need to be modified to include a connection to the repl database wherever there is a connection made to the source database.

The `ubroker.properties` file needs to be modified for PAS for OpenEdge that update the source database to add a connection to the repl db. Another point to consider is `conmgr.properties` modifications, start/shutdown scripts and backups. A stand-alone repl database requires standard database administration. This includes `conmgr.properties` modifications, startup/shutdown scripts and backups. *See the Pro2 Disaster Recovery topic for more information.*

Embedding the repl Tables

Embedding the repl tables into the source database can simplify implementation. If the repl tables are embedded, no additional database connection is required. The repl is usually used as a standalone to simplify schema updates made by the application provider to the source database. Pro2 works the same whether the repl tables are embedded or in a separate stand-alone database.

Repl database startup parameters

The repl database startup parameters are as follows:

- Max Number of Users (-n, `maxusers`)—All end-users must connect to the repl database as well as to the source database. For this reason, the max number of users (the -n parameter when starting the database server with the proserve command or maxusers when using dbman) specified for the repl database should not be lower than that of the source database. If this parameter is not set high enough, end-users may not be able to run the application.
- Max Lock Table Entries (-L, `locktableentries`)—Each update to the source database, which requires an Exclusive-Lock on the source record, results in a record being written to the replqueue in the repl database which also requires an Exclusive-Lock. Thus, the max number of lock table entries (the -L when starting the database server with the proserve command or lock table entries setting in the `conmgr.properties` file when using dbman) specified for the repl database should not be lower than that of the source database. If this parameter is not set high enough, end-users may encounter errors while trying to make updates and those updates fail.
- Max Number of Servers (-Mn, `maxservers`)—If there are a high number of remote users connecting to the source database, the default of five servers to handle all remote connections to the repl database may not be sufficient. The max number of servers for the repl database (the -Mn when starting the database server with the proserve command or maxservers setting in the `conmgr.properties` file when using dbman) should not be lower than that of the source database. If this parameter is not set high enough, end-users connected as remote clients may not be able to run the application.
- Before-Image File Parameters—For optimal performance, the repl database before-image (BI) file should be truncated at initial implementation to set the BI block size (-biblocksize) and the BI cluster size (-bi). for example, proutil Repl –C truncate bi –biblocksize 16 –bi 16384.

Mapping information

Mapping information is required to establish relationships between separate databases. It shows the mapped data fields from a source file to their related target fields. The repl database contains all the mapping information for the replication of the databases, tables and fields between the source and target. This information is stored in the `ReplDBXref, ReplTableXref,` and `ReplFieldXref` tables. Mapping information can be saved to and loaded from a text file.

Depending on the number, schema, and primary keys and foreign keys of the databases, the data sources, and the databse mapping information has a varying degree of complexity.

Repl properties and repl queue

The OpenEdge replication source database and target database tables are stored in property files. There are configuration settings whose values you set in these tables which control different aspects of your replication environment. The following are the tables for Pro2 replication environment:

- `ReplProperties`—Configuration settings such as log file location, logical delete tables and specification of procedure templates are stored in the `ReplProperties` table. Configuration settings can be saved to and loaded from a text file from the New button in the Manage Replication window on the Pro2 web user interface.
- `ReplQueue`—Information on change events is stored in the `ReplQueue` table. This information includes the `ROWID` of the record changed, event date/time, and queue thread. Typically, repl queue records represent updates made to the source database that are waiting to be written to the target SQL database.

Second pass replication

Second pass replication helps you to perform audit data warehouse tasks. It helps track all data changes to a table during a time interval for any or all fields. Second pass replication creates a copy of the source table in the second pass database and inserts a record into this table every time there is a change in the source table (for UPDATE, DELETE, INSERT).

A new record is created in the target (or staging) table everytime there is a change in the table irrespective of whether the change happens to the same field. The replicated table holds a new record for every change in the source table. A new property, `AUDIT_PASS` is added to the `predefs.i` file. This property supports second pass Replication and the default value of this parameter is set to NO. To start Second Pass Replication for a setup, this property should be set to YES for that specific setup.

Pro2 advanced capabilities

Pro2 contains several advanced capablities or features that allow you to perform advanced database administration tasks. This topic describes the folllowing capablities:

- ReplLog Check functionality
- Standard data type conversion
- Pro2 customization
- Replication customization
- Bulk-loading
- Multi-table Bulk-load criteria specification
- ASCII Bulk-load and export

ReplLog Check functionality

The ReplLog Check functionality allows you to scan all the log files for any errors and reports them through an email. To execute this functionality, run the `RunReplLogChk.bat` file from the \Pro2\bprepl\Scripts folder. If you are running this functionality for the first time,an `ErrorTrigger.lst` file is created (available in the repl_log folder). Add your error list that needs to be scanned/excluded in this file separated by comma. This scans all the log files, excluding the unwanted errors and reporting back the error instances found in an email.

You can log the errors in the `ErrorTrigger.lst` file as shown below:

```
[ADMIN_LOG]
error,[SQL Server],
[ADMIN_LOG-EXCLUDES]
Warning
[REPL_LOG]
Error,[SQL Server],Maximum number of client connections,Application server connect
failure,Appserver did not connect,Transport resources unavailable,
[REPL_LOG_EXCLUDES]
Warning,Closing Replication Log,Re-Opened Replication Proc Log,Openining Replication
[APPSRV_LOG]
Error,[SQL Server],
[APPSRV_LOG_EXCLUDES]
Warning,Closing Replication Log
[BULK_LOAD_LOG]
Error,[SQL Server],
[BULK_LOAD_LOG_EXCLUDES]
Warning
```

Note: If the `ErrorTrigger.lst` file is not found in the expected directory, it will be created in repl_log directory with an empty skeleton.

Standard data type conversions

Pro2 performs some data type conversions automatically to get past issues that would typically break other types of ODBC data transfers from OpenEdge to an MS SQL database server.

- Date conversions— Some versions of MS SQL server and Oracle cannot handle dates with a value before January 1, 1753. To work around this situation, for any dates before 1/1/1753, Pro2 preserves the month and the date and set the year to 1753.
- Column width—In OpenEdge databases, data is stored in variable length fields regardless of the field width definition. However, in MS SQL server and Oracle, column width is fixed. To work around this, Pro2 truncates data to the column width defined in the target schema.
- Logical fields—OpenEdge database recognizes three possible values for logical fields: True (1), False (0), and Unknown (?). The value "Unknown" is not recognized in foreign databases. To work around this, if a logical value is True, Pro2 sets the target field to True, otherwise it is set it to False.

Pro2 customization

Pro2 can be easily customized to suit your needs. You can generate Pro2 specific fields, can add and delete index information on the target database and change table and field names as per your needs. The following customizations can be done in Pro2:

- Customizable by table, by field/column, by record/row—All tables or a subset of tables can be replicated. In addition, not all fields within the tables selected for replication need to be replicated. In addition, logic can be added to "filter" records in selected tables so that only those records that meet the specified criteria are replicated.

- Custom transformations using ABL supported logic—Data can be manipulated with ABL logic before being written to the target database. For example, field values can be modified based on business logic or target side only columns can be populated with data consolidated from multiple source fields.
- Generating Pro2 specific fields—Pro2 provides an option Delta DF Pro2Pro to generate only Pro2 specific fields for your target database. In cases where the source schema is already replicated to your target database, you can use Delta DF Pro2Pro to generate only Pro2 specific fields, and thereby avoid unnecessary replication of source schema.

  Oracle Specialization: Use the following properties to generate Oracle specific output:

  - **ORACLE_USE_LOGICAL**—This property is responsible for converting logical field to number (1).

  - **ORACLE_USE_SCALE**—This property is responsible for adding precision and scale for decimal fields.

  By default, the value of these properties is set to NO. To use these properties, set the value to YES.

- Target side index information—Customers can add and delete index information on the target database without interfering with Pro2 Replication as long as no unique keys are added and as long as the `PRROWID` key is not modified. Pro2 compiles the CREATE INDEX statements into a separate file named <dbname>-<targettype>-index.sql file. These index statements are applied to the target only if the customer wants them.
- Standard modifications to table and field names— You can modify table and field names in Pro2.

  - Extent fields—Array fields from the OpenEdge source database are created as individual fields in the target database schema.
  - Reserved words—Any table or column names that are reserved words in the target database server or in OpenEdge must be renamed. Often this is done by appending an underscore "_" to the end of the column name, however, the column can be renamed to anything that is not a reserved word.

Replication customization

Pro2 is written completely in the ABL and has been designed for scalability and customization. you can customize auditing, source changes capturing and record maintainance. Apart from that you can also customize datatype and name transformation

- Capture source changes by SQL clients—The standard Pro2 replication triggers capture changes made by ABL clients. Pro2 can also capture changes made to the source OpenEdge database by SQL clients. To do this, java triggers can be implemented in the OpenEdge source database.

- Audit database—In addition to replication, Pro2 can be extended to write to a second target database to maintain audit records. Whereas each row is unique in the primary replication database (via `PRROWID` column), the audit database has multiple rows corresponding to each source database record, with a new row created for each change made to the source record. Auditing is implemented by having a second pass

through the repl queue by a separate replication processor designated for the audit database. Typically, when auditing is implemented, a subset of the replicated tables is audited, however, it is possible to audit all tables.

- Logical deletes—There are times when auditing is not required however it is desired to maintain records in the target database after they are deleted from the source database. In this case, instead of being deleted, rows are designated as logical deletes (via `PRROWID` modification) and the data is kept in the source database.

Data type and name transformation

Data types on the target side can be the same as on the source database or they can be different. For example, precision of decimal types can be decreased, logical fields can be changed to character, and integers can be changed to logical.

If you do not need the default conversion of datatypes, enable the `TGT_DATATYPE_OVERRIDE` property. This property retains the value of a datatype as is and does not convert it to meet the target schema requirements. However, you must recheck the validations as they may fail.

Table and column names on the target database can be the same or different from those on the source database(s). If they are the same, the Pro2 Auto-map function is used. Otherwise, the tables are manually mapped; however, mapping needs to be done once only and saved for use in subsequent implementations.

If you want to use literal table and column names for your target database (SQL only), you can enable the `TGT_USE_LITERAL_NAMES` property. This property allows you to use the same naming convention as the source database for tables and columns without modifying them to meet the requirements of the target database. Another property `GEN_SQL_DEC_FORMAT`, is used to set the format for decimal fields in both schema and differential files. If the property is set to YES, the format is same as in the source database. If it is set to NO then the default format [decimal(17,2) null] is used. The default value for this property is NO.

Bulk-loading

Bulk-loading is a process where one can load large amounts of data into a database in a relatively short period of time. Database indexes are typically optimized for inserting rows one at a time. However, when you need to load a lot of data all at once, inserting rows of data one at a time is often very slow and inefficient.

Bulk-loading is used when you need to import or export large amounts of data relatively quickly. With bulk loading operations, you don't just insert data one row at a time; data is instead inserted through a variety of more efficient methods based on the structure of the specific database.

The Bulk-load utility can be run from the Actions tab on the Pro2 user interface. Click Actions and then select Run Bulk Loads. It can also be run from the RunBulkLoader shortcut in the bprepl\Scripts folder.

Note: The Bulk-Load Utility always runs the standard bulk copy procedures in bprepl\repl_mproc. It does not run the Direct SQL bulk load procedures in bprepl\SQL_mproc. The SQL_mproc procedures must be run from a Pro2 Editor session.

Pro2 bulk-load processor

The bulk load processor is a collection of special replication programs that perform replication for an entire table which is done for an initial sync of the source table to the target table. A bulk-load procedure basically executes a query such as "**`FOR EACH`** table-name: **`BUFFER-COPY`** table-name TO Target-Name." This process can be run while replication is turned on. Run it again to sync an individual table. the only exception is of deletes on the target; if source deleted rows exist in the target, these need to be removed manually. To always be certain this does not happen, truncate the data from the target table before performing a bulk-load).

Bulk-load procedures

The bulk-Load procedures can be generated after table mapping has been done from the Manage Replication window in the Pro2 user interface. Select Manage Replication and then click Generate code. Select Bulk-Copy Processor. The bulk-load procedures are generated in the directory specified by the `MASS_LOAD_DIRECTORY` property (bprepl/repl_mproc by default).

The template file specified by the `MASS_LOAD_TEMPLATE` property is used in the bulk-load procedure generation. By default, this procedure is tmpl_mreplproc_restart_auto-push.p. Bulk-load procedures generated using this template will pick up where they left off if they are stopped and restarted. In addition, if a source record is locked and cannot be copied, the procedures add the record to the replqueue to be written by the replication processor and the bulk-load procedure moves on to the next record.

The `Bulk_Max_Cache` property in the tmpl_mreplproc_restart-auto-push template stores the last processed row-ids in the cache. In general, if the loading fails or stops during the bulk-load process and then restarted, the loading starts from the beginning. In such cases, the `Bulk_Max_Cache` property allows you to start the bulk-load process from the last processed row-ids stored in the cache instead of loading from the beginning. This property defaults to 25 and you can modify its value using the Properties tab in the Pro2 user interface.

During the bulk-load process, the `Oracle_Bulk_Transaction_Count` property (Oracle only) commits multiple records at a time per transaction. This property defaults to 1000. You can add this property and then modify its value using the Properties tab in Pro2 user interface. When setting this property for LAN configuration, ensure that the `MASS_LOAD_TEMPLATE` property must use the tmpl_mreplproc_oracle.p procedure.

For WAN, ensure that the `APPSRV_MASS_LOAD_TEMPLATE` property uses the tmpl_ASmproc_oracle.p procedure, and the `APPSRV_FETCH_COUNT_MASS_BULK` property has the same value as the `Oracle_Bulk_Transaction_Count` property.

You can use the `INCLUDE_LOB` property to choose whether to include LOBs in or exclude them from replication and bulkload. The property is set to YES by default, which transfers LOB data to the target database. However, to exclude LOBs from replication and bulkload, set this property to NO. This is applicable to both LAN and WAN environments.

All bulk_load templates have ElapsedTime property which indicates the total time taken for the bulk-load process to complete. The elapsed time information is provided in the log files generated by the bulk-load procedures. You can find this information on each log entry per 100K rows and as well as at the end of the loading process.

The bulk-load procedures write to individual log files in the folder specified by the `LOG_DIRECTORY` property (bprepl\repl_log folder). The Bulk-Load procedures can be run individually from the Procedure Editor or in a group by the mr<source-database>.p procedure which runs each one in turn. Multiple bulk-loads can also be launched at once from the Bulk-Load utility.

Multi-table bulk-load criteria specification

The Bulk-Load utility automatically skips over tables that have been flagged to not generate queue records (genqrec = FALSE) or to not process queue records (procqrec = FALSE). All other criteria are user-selected.

- For mapped bulk-load runners—Increase Run Flag If more than one instance of a Bulk-Load Utility is run concurrently, each instance must be given a separate flag (typically letters A through Z) to distinguish temp files created and used by each instance. Each instance of the Bulk-Load Utility can have up to five bulk-load threads. The number of bulk-load utility instances and associated threads is limited by the Progress license on the Pro2 box. For example, two Bulk-Load runners set for five threads each equals a total of twelve Progress license sessions running.
- For mapped repl thread—Tables mapped to the replication thread entered are selected; zero (0) specifies all threads.
  - Start at table—Indicates what table to start from. Format: {dbname}.{tablename} Example "sports.item" starts at the "item" table of the "sports" database, skipping over tables with names alphabetically before "item". A period (.) indicates to start from the beginning.
  - Multi-thread bulk-loads—The number bulk-load processes or "threads" to run at once. The maximum is five bulk-load threads per bulk-load launcher.

Note: Bulk-load threads are distinct from replQueue threads.

  - Re-load table if already bulk-loaded once—Pro2 track of the tables that have been bulk- loaded stores and stores information in the replControl table. It uses this data to restart a load if it is stopped. It also skips loading a table that is already completely loaded. If you set this flag to yes it deletes any "COMPLETED" control records, so the table re-loads from the beginning.
  - Table include list—A comma separated list of tables to include. Patterns can be specified using "*" as a wild card. Example: "ca*,da*,e*,z*". This performs a bulk-copy of all tables starting with ca, da, e, and z.
  - Table exception list—A comma separated list of tables to exclude or skip. Patterns can be specified using "*" as a wild card. Example: "ca*,da*,e*,z*". This performs a bulk-copy of all tables except those starting with ca, da, e, and z.

ASCII Bulk-load and export

For WAN implementations, the ASCII bulk-export/import was designed to sync replication tables from Source to Target. For very large tables (typically over fifty million rows), the ASCII Bulk-Export/Load utility can be used to decrease the time of the bulk load. The process uses an ASCII dump export process on the Source server. This process creates the dump procedures and the SQL load procedures and then dumps the data to ASCII flat files. Once this is finished the dumped ASCII data files and the generated SQL load procedures are transferred to the Target database server. The SQL load procedures are then run from the SQL query-editor.

Disaster recovery planning

Note: Ideally Pro2 is not designed for disaster recovery.

The role of the Replicated SQL database in your business is one of the major points of consideration in planning for Disaster Recovery (D/R) in a Pro2 replication environment. Another point to consider is the size of your source and target databases and the length of time required to restore and to reload the data to the SQL target.

Pro2 Disaster Recovery Planning should include responses to the question, how would it impact our business if we lost the:

- Source database
- Repl database (assuming repl tables are not embedded in a source database and are in a standalone repl database)
- SQL target database
- Pro2 server
- SQL server
- OpenEdge database server

When is it necessary to reset the SQL target database

Generally, the target SQL database must be loaded whenever the source database or repl database need to be restored from backup. Bulk-loading the data from the source OpenEdge database to the target SQL database can take a long time. This time may range from days or weeks depending upon the size of the database. Your disaster recovery plan should be clear on what situations require a full re-bulkload of the data and the steps required to do so.

To answer the question of when to reset, it is important to understand the role of `ROWID` in the replication process. The `ROWID` is the unique record identifier in the OpenEdge database. Pro2 uses the `ROWID` to store and identify the corresponding record in the SQL database. Two important facts about how `ROWID`'s are assigned in the OpenEdge database are:

- The order in which records are created and deleted determines their `ROWID`.
- When a record is deleted from the OpenEdge database, its `ROWID` can be used again for a subsequently created record.

Thus, even if you know exactly what records were created and deleted during the time the source database or repl database were lost and re-apply those changes to a restored version of the database, the `ROWID`'s could be completely different than the original values stored in the target SQL database and any subsequent changes result in invalid data in the target SQL database.

Why back-up the repl database?

The repl database contains a lot of valuable information and understanding the roles of the various tables in the repl database is helpful in determining a backup strategy for the repl database. The repl database contains many tables and its essential to back up the database to proper functioning of the too.

Some of the important reasons to back up the database are:

- The repl database contains all the mapping information for the replication of the databases, tables and fields between the source and target.
- The repl properties contain configuration settings such as log file location, logical delete tables and specification of procedure templates which are stored in the ReplProperties table.
- The replqueue table stores information on change events. This information includes the `ROWID` of the record changed, event date/time, and queue thread.

Restrictions with online backups

You must be cautious while using online backups as part of your disaster recovery solution. Some of the important points to be taken care of are:

- The backups of the source database and the repl database could be out-of-sync depending on the timing of the online backups for the repl database and the source database.
- The ReplQueue may have transactions that were not included in the backup of the source database if the source database backup occurred first.
- The ReplQueue in the backup of the repl database might be missing committed transactions if the repl database backup occurred first.

Progress recommends that online backups are used only in conjunction with after-imaging.

Refer to *Progress OpenEdge Data Management* documentation for information on Progress backup utilities.

Pro2 and after-imaging

After-imaging can be enabled and running on the source database concurrently with Pro2 replication. If after-imaging is enabled and running on the source database, it does not need to be enabled on the repl database. However, if after-imaging is not enabled on the repl database, the disaster recovery plan must include re-seeding the target SQL database.

Refer to *Progress OpenEdge Data Management* documentation for information on how to enable after-imaging.

Pro2 and OpenEdge replication

OpenEdge Replication is a Progress product that maintains a replicated copy of the source database as part of a disaster recovery scheme. Pro2 replication can be running concurrently with OpenEdge replication. If the target SQL database and the repl database are part of the disaster recovery solution, the OpenEdge replication configuration needs to be modified to include them.

Refer to *Progress OpenEdge replication* documentation for more information.