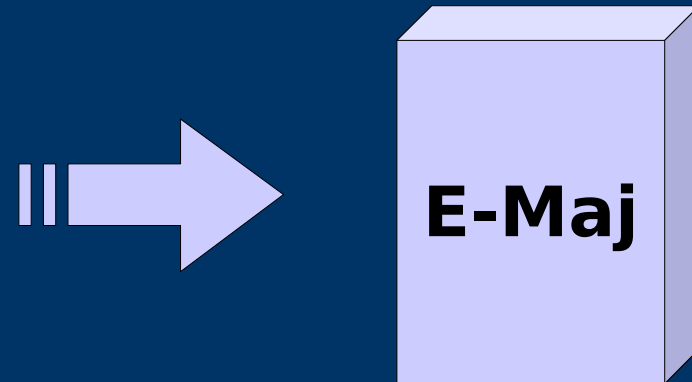


E-Maj 0.9: a PostgreSQL contrib

Ph.Beaudoin – January 2011

From the idea of logical restore to ... E-Maj

- Original idea = table_log contrib from Andreas Scherbaum
 - 1 trigger per table to log all updates into a log table
 - 1 function to cancel the updates
- Development of plpgsql functions extending the concept to build a solution usable on production



French acronym for
« Enregistrement des Mises A Jour »,
i.e. Updates recording

Requirements

- Reliability:
 - Absolute integrity of databases after « rollbacks »
 - Manage all objects (tables, séquences, contraintes,...)
 - Ease of use for DBAs and production people:
 - Easy to understand and use
 - Easy to automatize (« scriptable »)
 - Performance:
 - Limited overhead of the log
 - Acceptable « rollback » duration
 - Maintainability
 - Security
-
-

Concepts

- **Table_group** = a set of tables and/or sequences belonging to a unique schema or several schemas and having the same life cycle ; it's the object on which « marks » and « rollbacks » are applied ; it's the only object manipulated by users
 - **Mark** = stable point in the life of a table_group, whose state can be set back ; is identified by a name
 - **Rollback** = positioning of a table_group at its state when a mark was previously set
-
-

Installation

- Preliminary operations:
 - plpgsql language has to be created in the database
 - a tablespace, named tspemaj, must have been created in the cluster
 - Installation done with a unique script, named emaj.sql ; to be launched using a super-user ROLE
 - The installation in a database adds :
 - 1 schema (emaj) containing
 - 36 plpgsql functions and
 - 10 technical tables and 2 types
-
-

Initialisation

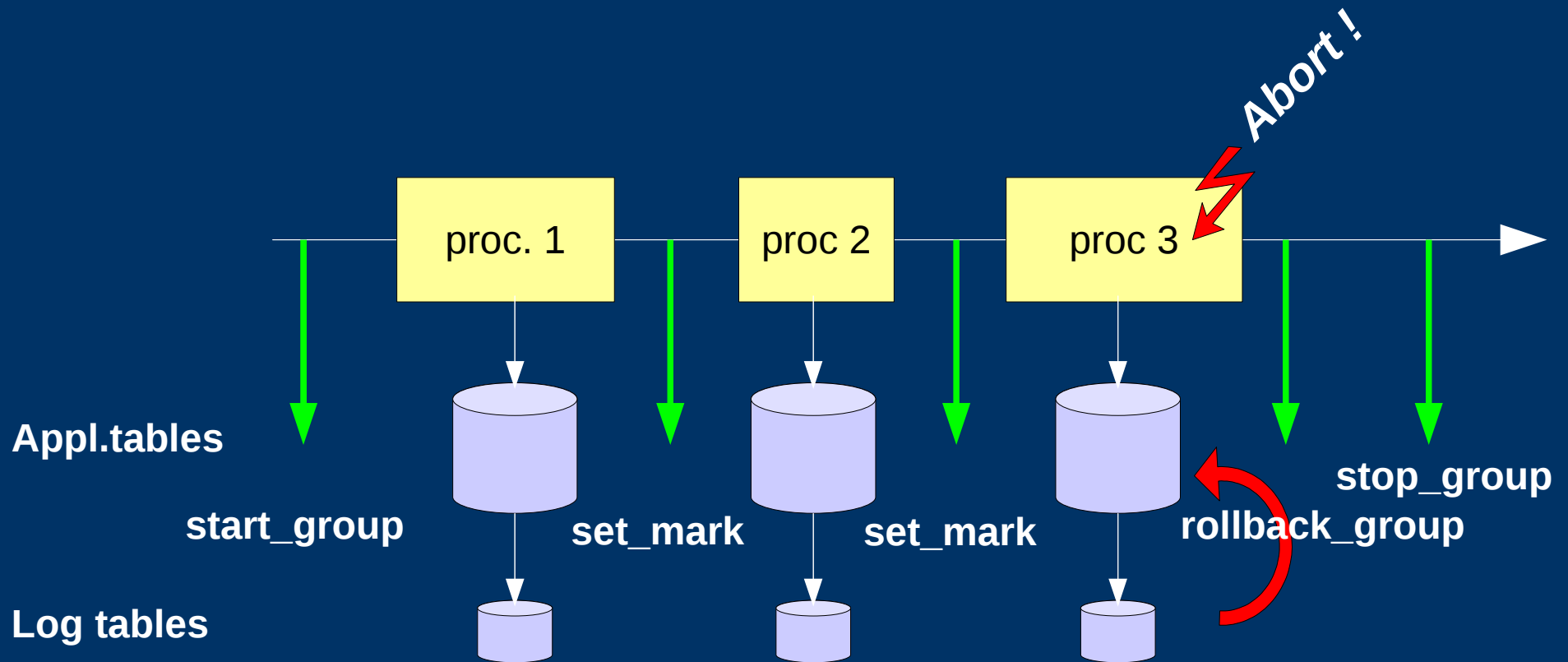
- 1) Populate emaj_group_def table to define groups and the tables/sequences they contain
 - 2) For each group :
 - SELECT **emaj_create_group** (group);
 - => creates for each application table:
 - 1 trigger associated to table updates
 - 1 log table into tablespace tspemaj
 - 1 function to « rollback » the updates on the application table
 - A **emaj_drop_group** (group) function ... drops a previously created group
-
-

Main functions

- **emaj_start_group** (group, mark)
 - Activates log triggers and set an initial mark
- **emaj_set_mark_group** (group, mark)
 - Sets an intermediate mark
- **emaj_rollback_group** (group, mark)
 - Rollbacks tables et sequences of the group to their state at mark set
- **emaj_stop_group** (group)
 - Deactivates log triggers => rollback no longer possible



A typical E-Maj sequence ...



Possible usages

- Provides a rollback capability on batch processing without being obliged to either pgdump/restore tables or physically save and restore the entire cluster disk space
- All the more interesting as:
 - tables are large with relatively limited updates
 - several tables groups / databases share the same cluster
- Can also help application tests in providing a way to quickly rollback updates issued by a run and repeat those tests



Marks usage strategies

- « mono-mark » usage to minimise disk space use
 - repeat
 - start_group (group, mark)
 - processing i
 - stop_group (group)
- « multi-marks » usage for more flexibility in rollbacks
 - start_group (group, mark1)
 - repeat
 - processing i
 - emaj_set_mark (group, mark i+1)
 - stop_group (group)

Statistic functions

- **emaj_log_stat_group** (group, begin_mark, end_mark)
 - Quickly provides per table statistics about the number of rows in log tables between 2 marks or between a mark and the current situation
- **emaj_detailed_log_stat_group** (group, begin_mark, end_mark)
 - Delivers statistics from log tables on updates between 2 marks, per table, per statement type (INSERT / UPDATE / DELETE) and per ROLE that initiated the updates

Other secondary functions (1/2)

- **emaj_estimate_rollback_duration** (group, mark)
 - Estimate the time needed to rollback a group to a mark
- **emaj_rollback_and_stop_group** (group, mark)
 - Chains the calls to rollback_group et stop_group functions - this allow to differ the rows deletion from log tables in order to get quicker rollback
- **emaj_reset_group** (group)
 - Purges log tables before the next emaj_start_group call. This is a way to reclaim disk space if needed

Other secondary functions (2/2)

- `emaj_delete_mark_group` (group, mark)
 - Suppress a mark
- `emaj_rename_mark_group` (group, old mark, new mark)
 - Renames a mark
- `emaj_force_drop_group` (group)
 - Force the suppression of a group (in case `emaj_drop_group` function is not usable)
- `emaj_verify_group` (group)
 - Verifies the E-Maj internal consistency of a group

Test functions

- `emaj_snap_group` (group, directory)
 - Snaps all tables and sequences of a group on individual files located on a directory
 - Rows are ordered by primary keys
 - Snap files can be diff with a reference to be sure the log and rollback operations worked properly

Parallel rollback extension

- A php module performs parallel restore
 - Acts as a client of the database
 - Automatically spreads the group to rollback into a given number of subgroups
 - Performs the parallel rollback in a unique transaction (2PC) ($\Rightarrow \text{max_prepared_transaction} \geq \text{\#subgroups}$)
 - **emajParallelRollback.php** -d <database> -h <host> -p <port> -U <user> -W <password> -g <group_name> -m <mark> -s <\#subgroups>
 - Other options: --help, -v, --version
 - Needs php with the PostgreSQL extension
-
-

Reliability

- Many checks in particular at emaj_start_group and emaj_rollback_group time
 - Do all listed tables and sequences exists ?
 - Do the triggers and log tables exist with the right columns and types ?
 - Are we sure the table stuctures have not changed between start_group and rollback_group functions
 - Exclusive lock on tables at start_group and rollback_group time to be sure no transaction are currently accessing the tables
 - Rollback all tables et sequences in a single transaction
-
-

Security

- 2 roles that can be granted :
 - emaj_adm for ... emaj administrators
 - emaj_viewer to just be able to look at log tables
 - E-Maj objects are only created by a super-user or a member of emaj_adm
 - No other right is granted on the emaj schema and all related tables and functions
 - Log triggers are created as « SECURITY DEFINER »
 - No need to grant extra rights on application tables
 - Protection against SQL injections
-
-

PhpPgAdmin plugin

- A plugin for phpPgAdmin 5 is available to help administrator or viewer
 - Shows all E-Maj objects and their attributes
 - Allows all possible actions on E-Maj objects

Current limits

- PostgreSQL version : from 8.2 up to 9.0
- Every application table belonging to a group needs a **PRIMARY KEY**
- Schema name length + application table name length \leq 52 characters
- **DDL** or **TRUNCATE** operations cannot be managed by E-Maj.
 - TRUNCATE are just blocked when pg version $>$ 8.3

To conclude...

- More information in the readme.txt and releaseNotes.txt files
- Many thanks for their help to :
 - Andreas Scherbaum
 - Jean-Paul Argudo and Dalibo team

