

E-Maj 0.10.0

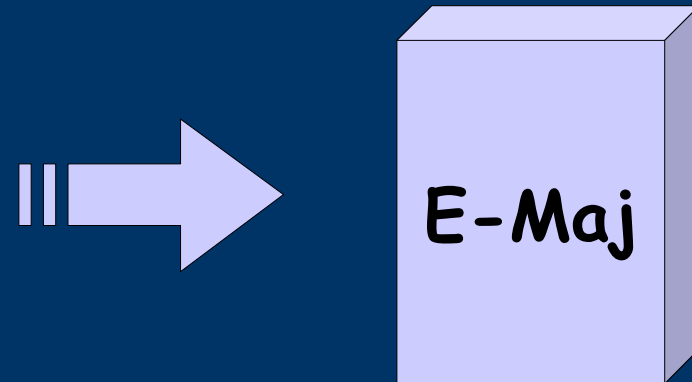
-

a PostgreSQL extension



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



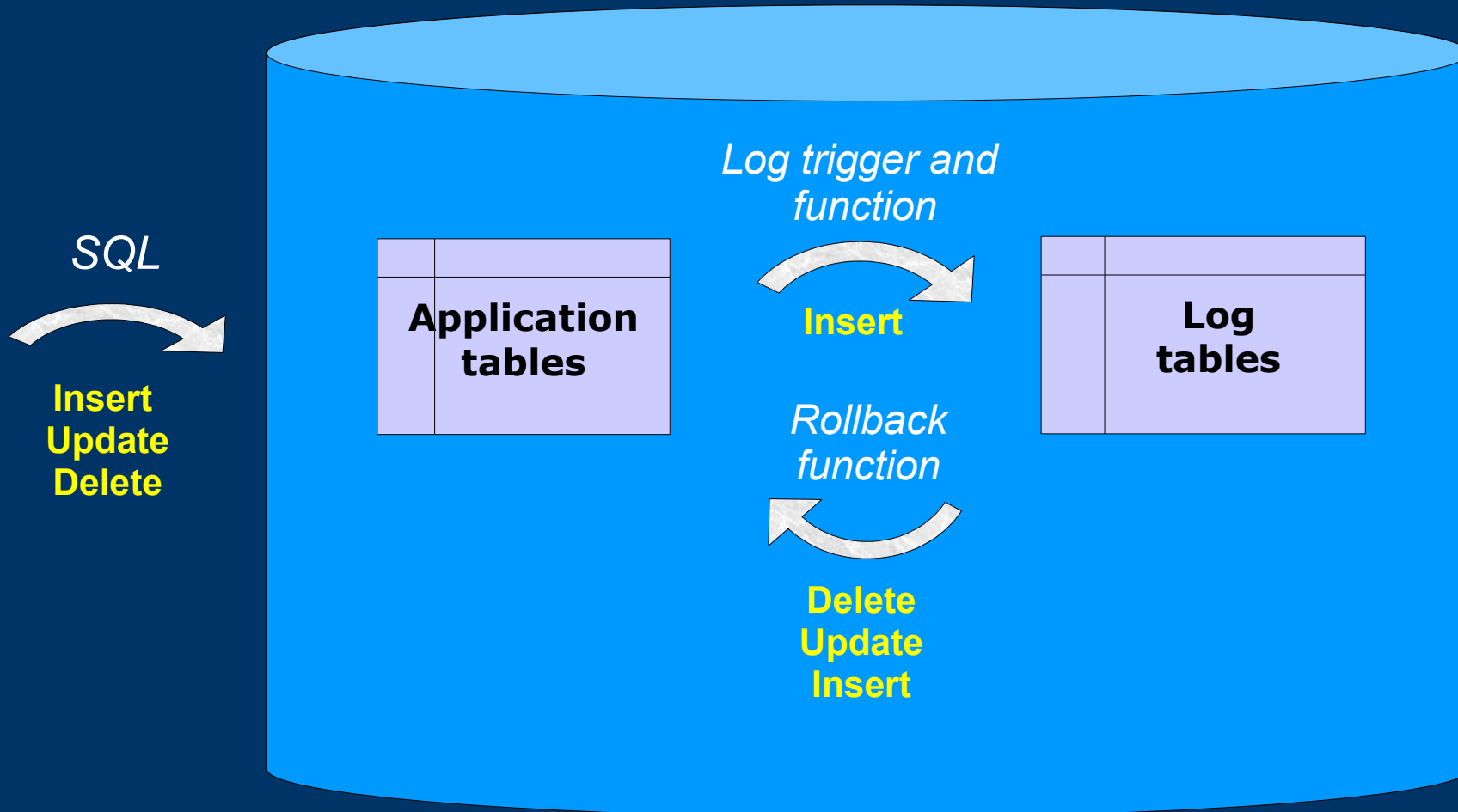
French acronym for
« Enregistrement des Mises A Jour »,
i.e. Updates recording

E-Maj objectives

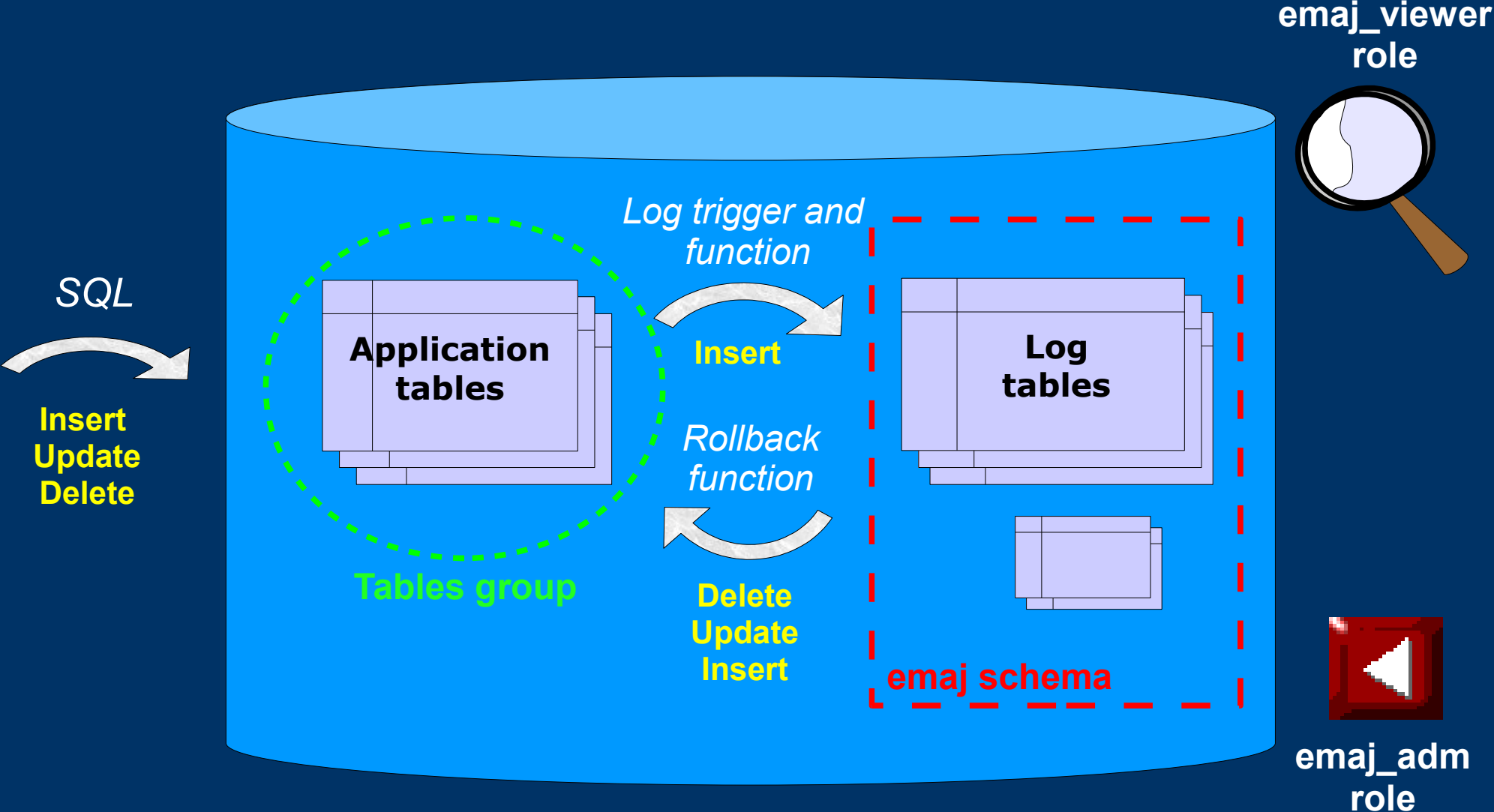
- Record application tables updates in order to:
 - Look at them (audit)
 - Rollback them if needed
- Open Source Extension available on pgFoundry.org



The basics of updates logging



E-Maj: general principles



E-Maj Requirements

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

E-Maj Concepts

- **Tables group** = a set of tables and/or sequences belonging to one 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 tables group, whose state can be set back ; is identified by a name
 - **Rollback** = positionning of a tables group at its state when a mark was previously set
-
-

E-Maj Installation

- Cluster preliminary operation:
 - CREATE TABLESPACE tspemaj LOCATION...
 - Database preliminary operation:
 - CREATE LANGUAGE plpgsql; (pg < 9.0)
 - Then, as super-user:
 - \i ../sql/emaj.sql or
 - CREATE EXTENSION emaj; (pg 9.1+)
 - The installation in a database adds:
 - 1 schema 'emaj' with 61 functions, 10 technical tables and 2 types
 - 2 roles for E-Maj management
-
-

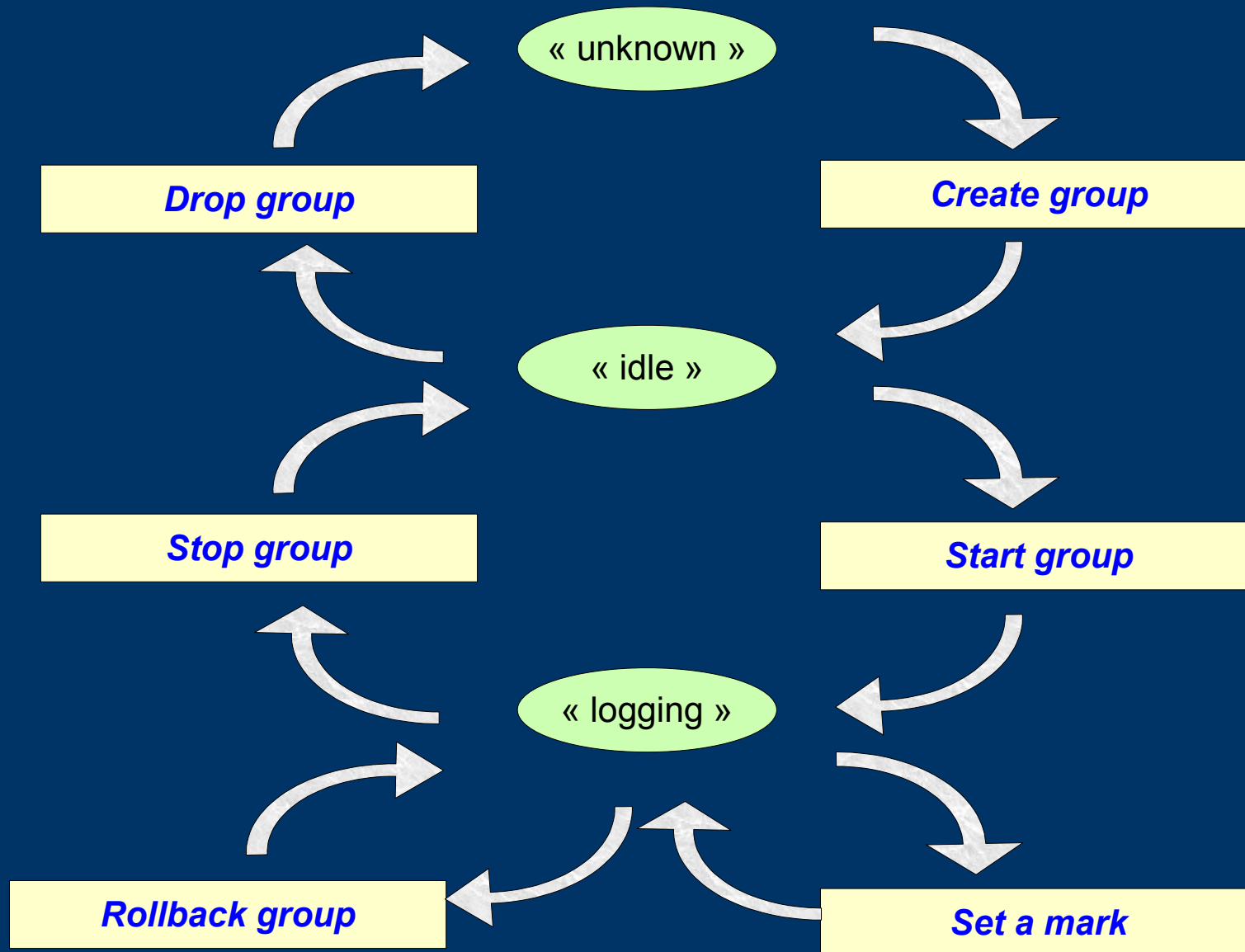
E-Maj Initialisation

- 1) Populate emaj_group_def table to define groups and the tables/sequences they contain
 - 2) For each group :
 - SELECT **emaj_create_group** (group, is_rollbackable);
 - Creates for each application table:
 - 1 log table into schema emaj and tablespace tspemaj
 - 1 trigger + 1 function to log table updates
 - 1 function to « rollback » the updates on the application table (if rollbackable group)
 - SELECT **emaj_drop_group** (group)
... drops a previously created group
-
-

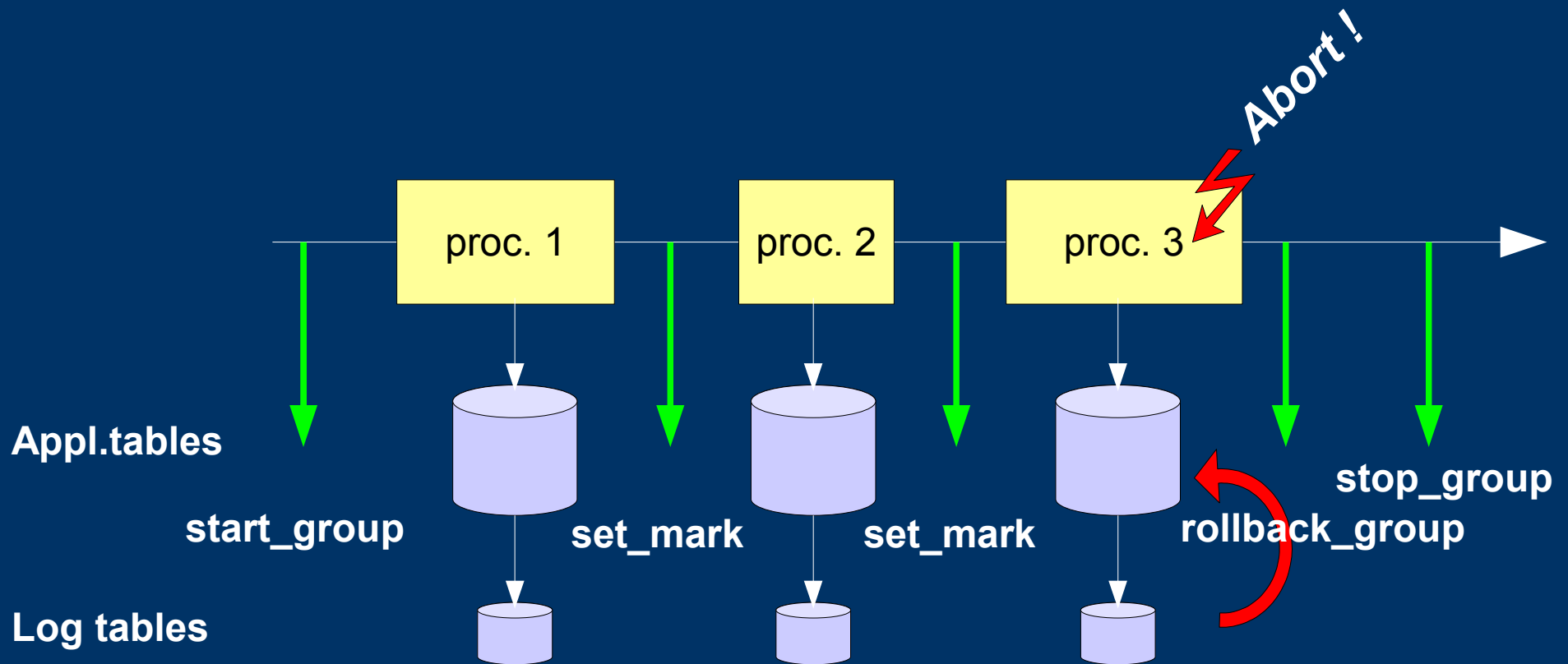
E-Maj: Main functions

- **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)
 - Rolls back tables and sequences of the group to their state at mark set
 - **emaj_logged_rollback_group** (group, mark)
 - Similar as emaj_rollback_group function but the rollback can be later canceled (rollbacked!)
 - **emaj_stop_group** (group)
 - Deactivates log triggers => rollback no longer possible
-
-

E-Maj: tables group life cycle



A typical E-Maj sequence ...



E-Maj 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
- Can also help application tests in providing a way to quickly rollback updates issued by a run and repeat those tests



Marks usage strategies (1/2)

- « 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)
-
-

Marks usage strategies (2/2)

- Permanent logging and regular cancellation of oldest marks (« rolling log »)
 - repeat
 - processing #i
 - emaj_set_mark (group, mark #i+1)
 - emaj_delete_before_mark (group, mark #j)
- (warning, marks deletion may be costly)



Multi-groups functions

- **emaj_start_groups** (groups array, mark)
 - Starts several groups at once and set a common mark
 - **emaj_set_mark_groups** (groups array, mark)
 - Sets a common mark for several groups
 - **emaj_rollback_groups** (groups array, mark)
 - Rollsbacks several groups at once (i.e. in a single transaction) to a common mark
 - **emaj_logged_rollback_groups** (groups array, mark)
 - Similar as emaj_rollback_groups function but the rollback can be later canceled
 - **emaj_stop_groups** (groups array)
 - Stops several groups at once
-
-

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 (1/3)

- **emaj_estimate_rollback_duration** (group, mark)
 - Estimates the time needed to rollback a group to a mark
 - **emaj_rollback_and_stop_group** (group, mark)
 - Chains the calls to `rollback_group` and `stop_group` functions - this allows to differ the rows deletion from log tables in order to get quicker rollback
 - **emaj_comment_group** (group, comment)
 - Sets, modifies or deletes a comment on a group
 - **emaj_reset_group** (group)
 - Purges log tables before the next `emaj_start_group` call (and reclaims disk space)
-
-

Other secondary functions (2/3)

- **emaj_comment_mark_group** (group, mark)
 - Sets, modifies or deletes a comment on a mark
 - **emaj_find_previous_mark_group** (group, timestamp)
 - Retrieves the mark name immediately preceding a point in time
 - **emaj_delete_mark_group** (group, mark)
 - Suppress a mark
 - **emaj_delete_before_mark_group** (group, mark)
 - Suppress all marks preceding the supplied mark
 - **emaj_rename_mark_group** (group, old mark, new mark)
 - Renames a mark
-
-

Other secondary functions (3/3)

- **emaj_force_drop_group** (group)
 - Forces 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
 - **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 client

- A php module performs parallel restore
 - Acts as a client for the database
 - Automatically spreads the group(s) to rollback into a given number of sessions
 - Performs the parallel rollback in a unique transaction (2PC) (→ `max_prepared_transaction >= #sessions`)
 - `emajParallelRollback.php` `-d <database> -h <host> -p <port> -U <user> -W <password> -g <group_name or groups_list> -m <mark> -s <#sessions> [-1]`
 - Other options: `--help`, `-v`, `--version`
 - Needs php with the PostgreSQL extension
-
-

Reliability

- Many checks, in particular at `start_group` and `rollback_group` time
 - Do all listed tables and sequences exist ?
 - Do the triggers and log tables exist with the right columns and types ?
 - Are we sure the table structures have not changed between `emaj_start_group` and `eamj_rollback_group` functions call
 - Strong locks on tables at `start_group`, `set_mark_group` and `rollback_group` times to be sure no transaction are currently accessing/updating application 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 its related tables and functions
 - Log triggers are created as « SECURITY DEFINER »
 - No need to grant extra rights on application tables
 - Protection against SQL injections
-
-

Performances

- Log overhead
 - Highly depends on hardware and sql read/write ratio
 - Typically a few % on elapse times
 - Rollback duration
 - Highly depends on hardware and database structure (row sizes, indexes, constraints...)
 - Measured on recent hardware with a real application: about 10Gb of log in 1 hour
-
-

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
- Ask for it, if you want to try...



Current limits

- PostgreSQL version : from 8.2 up to 9.1
- Every application table belonging to a rollbackable 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.
 - TRUNCATEs are just blocked when pg version $>$ 8.3

To conclude...

- More information in the documentation + README and CHANGES files
 - Many thanks for their help to :
 - Andreas Scherbaum
 - Jean-Paul Argudo and Dalibo team
 - CNAF DBAs team
 - People who already contacted me for comments, requests...
 - Feel free to email: [phb<dot>emaj<at>free<dot>fr](mailto:phb@emaj.free.fr)
-
-