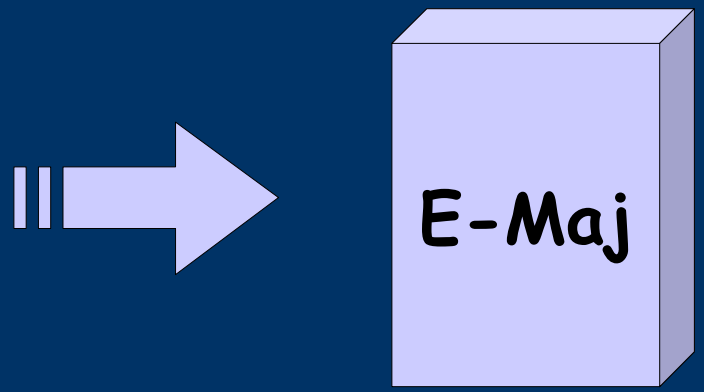
E-Maj 1.0.1

-

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 more “industrial” solution



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

Components

- E-Maj
  - PostgreSQL extension
  - Open Source (license GPL)
  - Available on
    - pgfoundry.org
    - pgxn.org
    - github

- Plugin for phpPgAdmin
  - A version with phpPgAdmin 5.0.4 available on demand

E-Maj objectives

- Record application tables updates in order to:
  - look at them (audit)
  - rollback them if needed
- Usable
  - with applications in test or in production
  - with database of various size

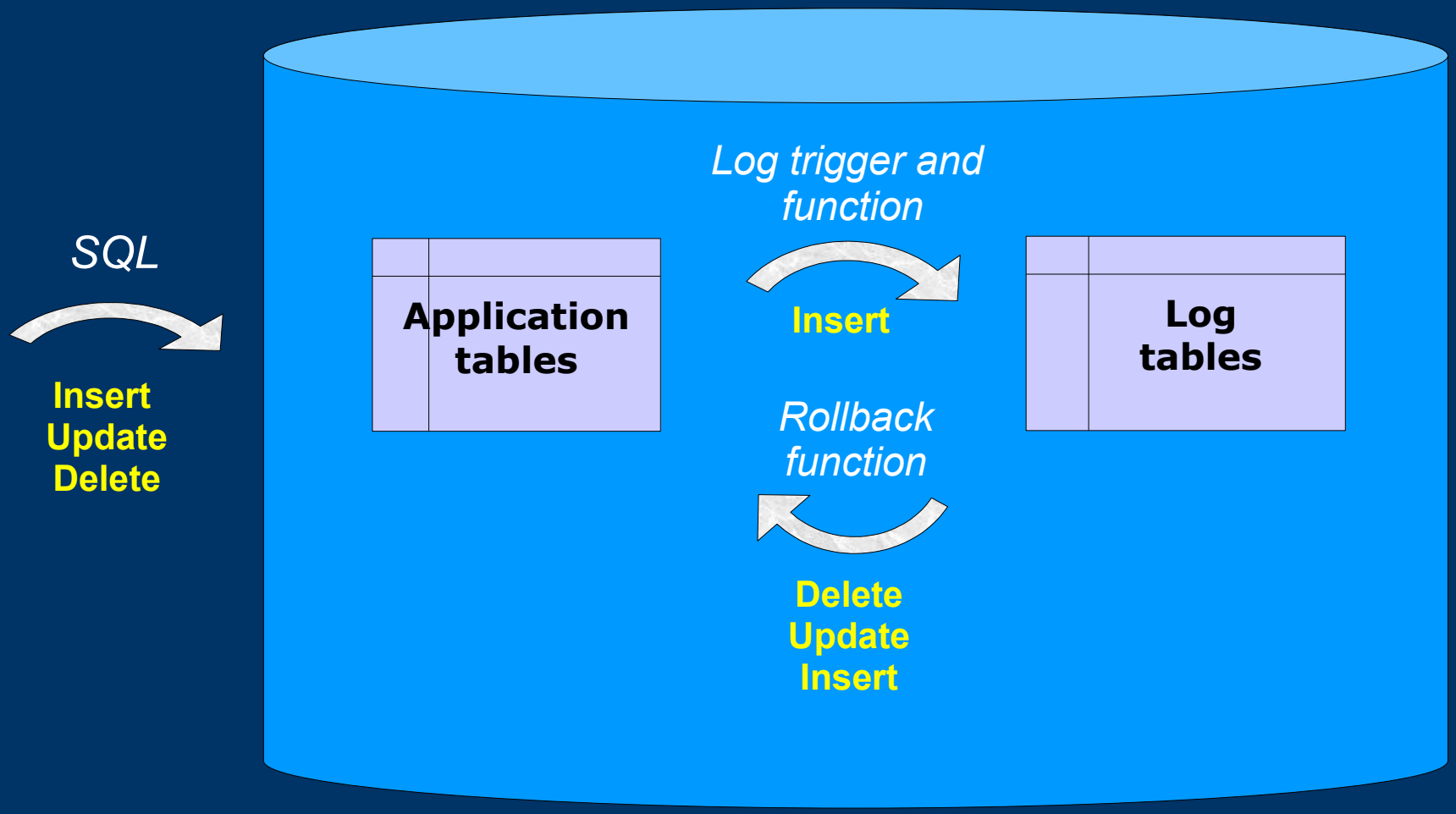
E-Maj Requirements

- Reliability:
  - Absolute integrity of databases after « rollbacks »
  - Manage all usual objects (tables, sequences, constraints,...)
- Ease of use for all users (DBA, production people, application developers and testers,...):
  - Easy to understand and use
  - Easy to integrate into an automatized production (« script-able »)
- Performance:
  - Limited overhead of the log (a few percent)
  - Acceptable « rollback » duration
- Maintainability
- Security

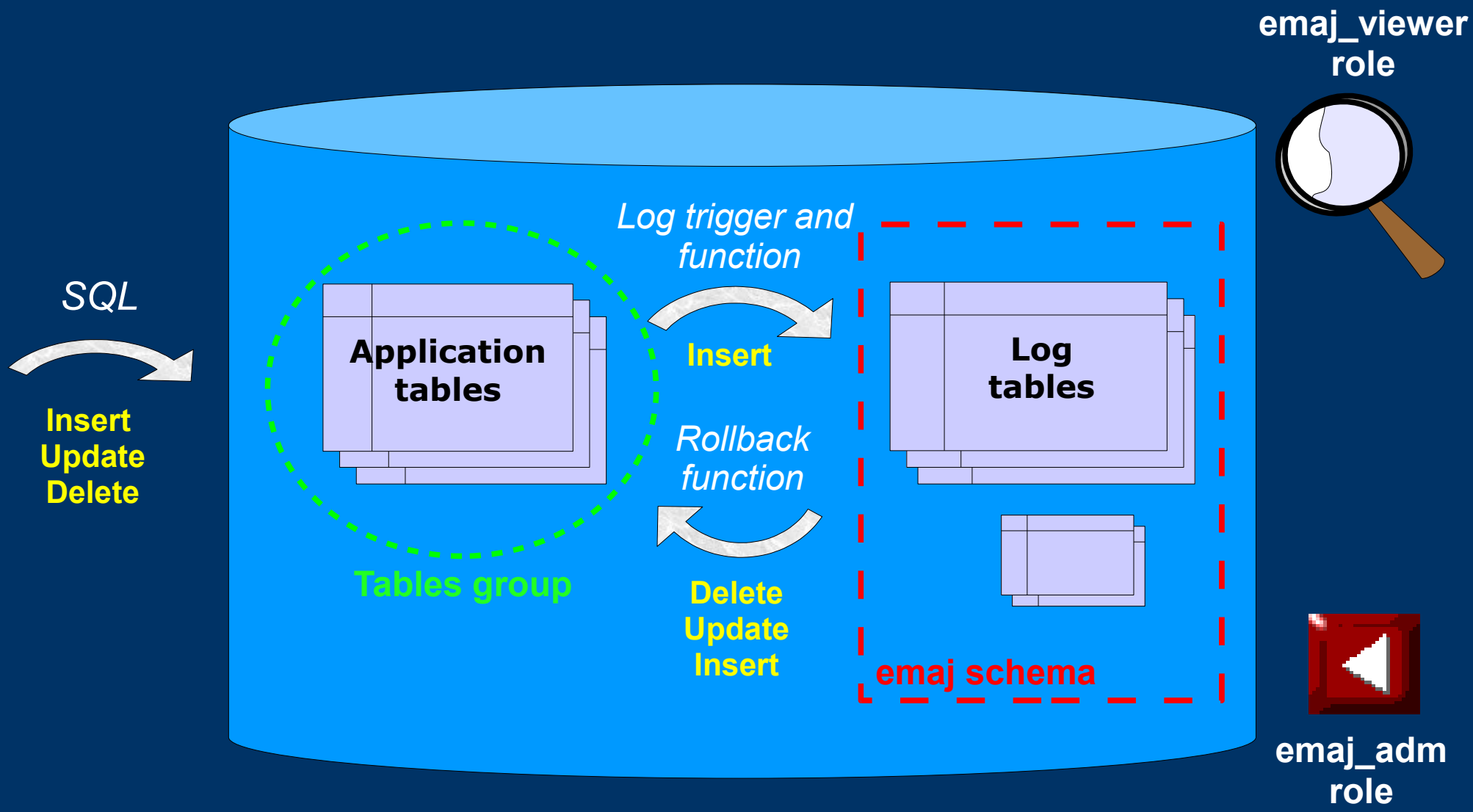
E-Maj Concepts

- Tables group = a set of tables and/or sequences belonging to one or several schemas and having the same life cycle ; it's the only object manipulated by users
- Mark = stable point in the life of a tables group, identified by a name and whose state can be set back
- Rollback = positioning of a tables group at its state when a mark was previously set

The basics of updates logging



E-Maj: general principles



E-Maj Installation

- Database preliminary operation:
  - CREATE LANGUAGE plpgsql; (if pg < 9.0)
- Then, as super-user:
  - \i .../sql/emaj.sql
- The installation in a database adds:
  - 1 schema 'emaj' with 75 functions, 10 technical tables and 3 types
  - 2 roles

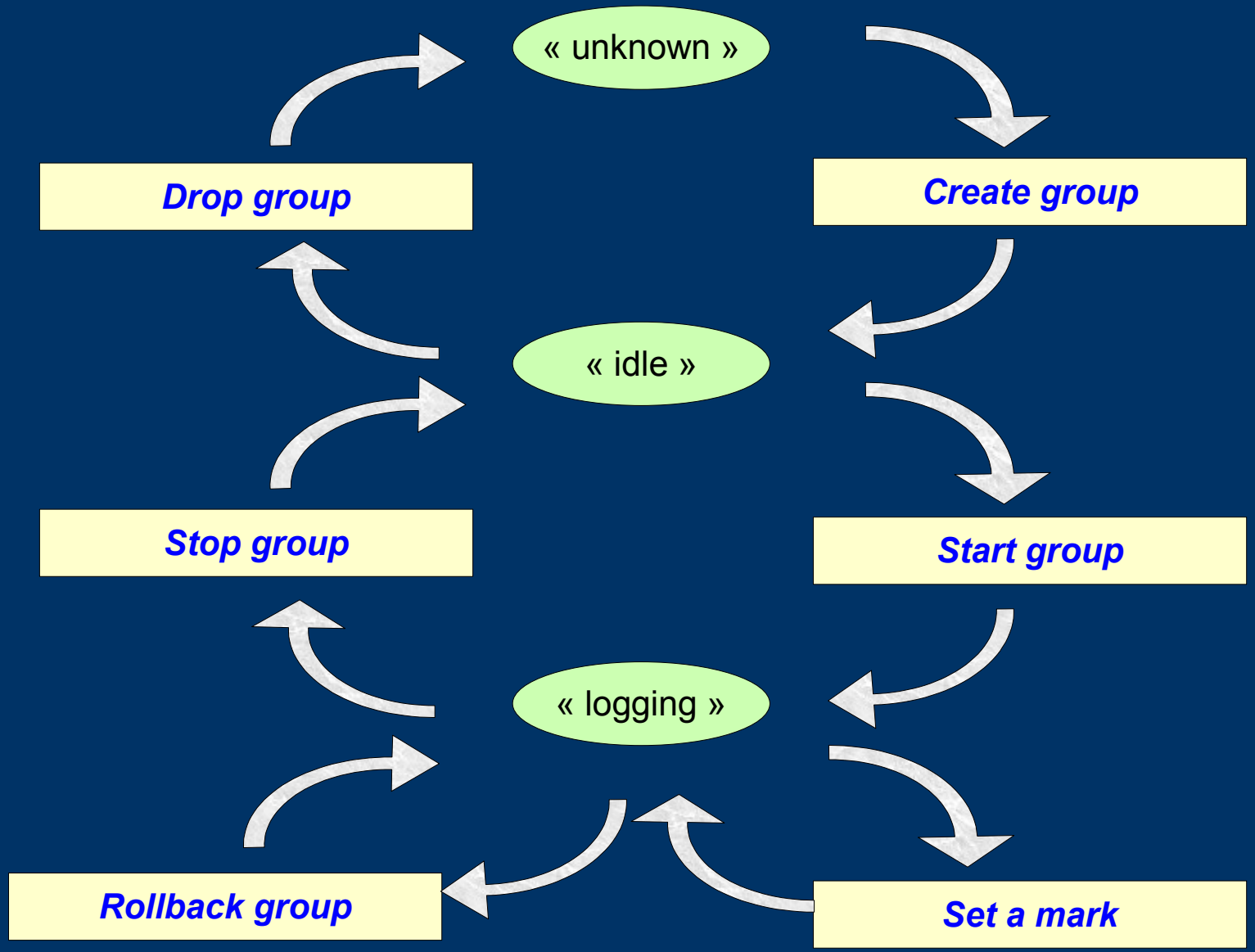
E-Maj Initialisation

- 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

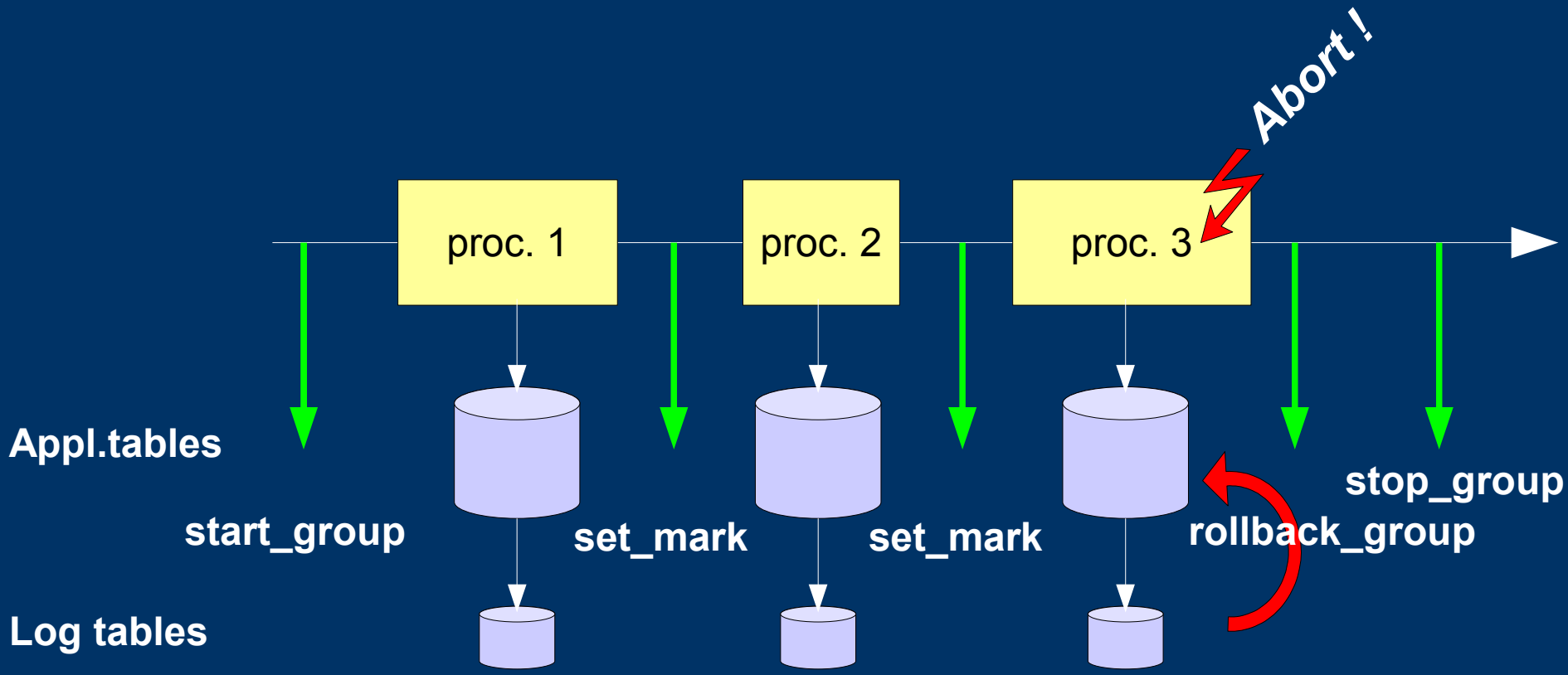
E-Maj: Main functions

- 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)
  - Rollbacks 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 cancelled (rolled-back!)
- emaj_stop_group (group [,mark])
  - Deactivates log triggers => rollback no longer possible

E-Maj: tables group life cycle

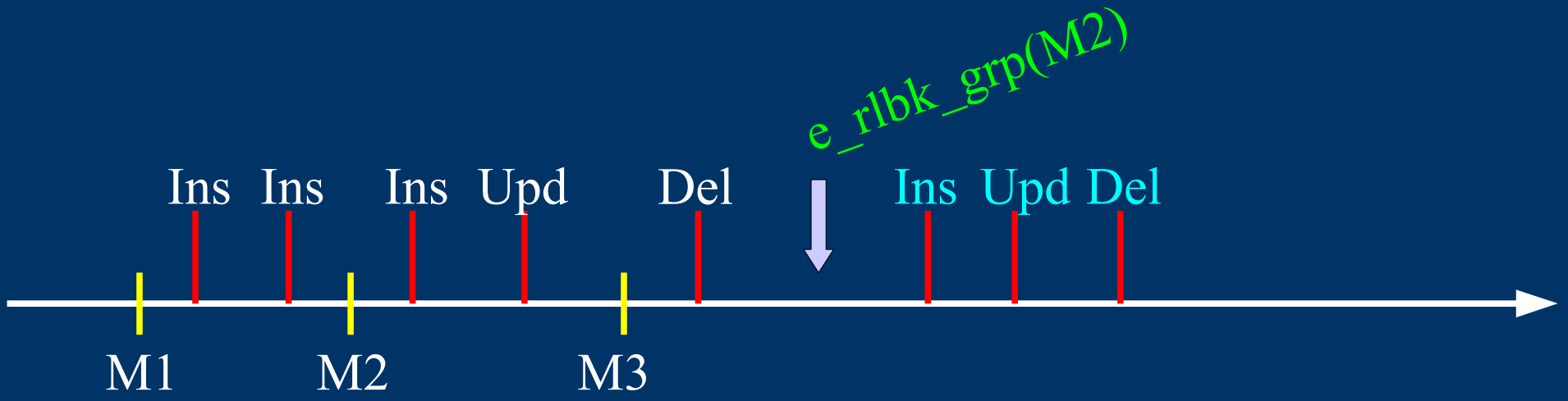


A typical E-Maj sequence ...



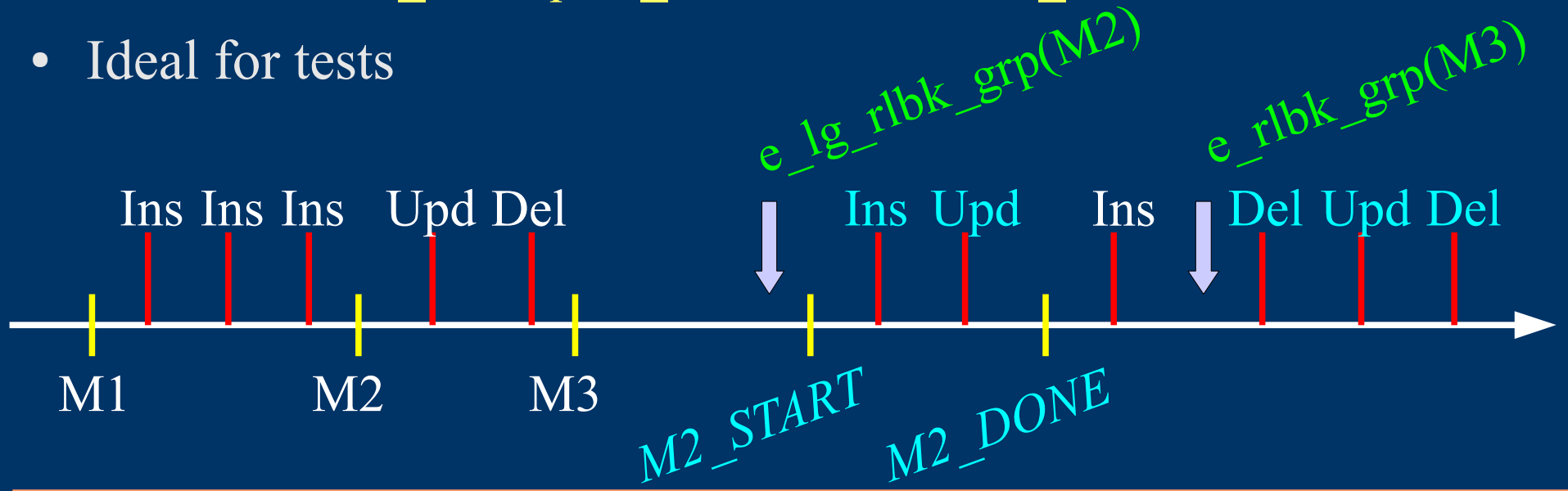
« Simple Rollback »

- Log triggers are de-activated
- INSERTs are cancelled by DELETEs, DELETEs by INSERTS and UPDATEs by … UPDATEs,
- Applied in reverse order
- Cancelled logs and marks are deleted



« Logged Rollback »

- Log triggers are NOT de-activated
- Cancelled logs and marks are kept
- A mark before and a mark after the rollback are automatically set
  - RLBK_<marque>_<HH.MI.SS.MS>_START
  - RLBK_<marque>_<HH.MI.SS.MS>_DONE
- Ideal for tests



E-Maj possible usages

- Can largely help application tests in providing a way to quickly rollback updates issued by a run and repeat those tests
- In production, provides a rollback capability on batch processing without being obliged to either pg_dump / restore tables or physically save and restore the entire cluster disk space
  - All the more interesting as tables are large, with relatively limited updates

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

- To manage several groups in a single transaction:
  - emaj_start_groups (groups array, mark)
  - emaj_stop_groups (groups array)
  - emaj_set_mark_groups (groups array, mark)
  - emaj_rollback_groups (groups array, mark)
  - emaj_logged_rollback_groups (groups array, mark)
- 2 syntaxes for a groups array
  - ARRAY['group 1','group 2',...]
  - '{"group 1", "group 2",…}'

Marks management functions

- emaj_comment_mark_group (group, mark)
  - Sets, modifies or deletes a comment on a mark
- emaj_delete_mark_group (group, mark)
  - Suppress a mark
- emaj_delete_before_mark_group (group, mark)
  - Suppress all marks preceeding the supplied mark
- emaj_rename_mark_group (group, old mark, new mark)
  - Renames a mark

Other groups management functions

- 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)
- emaj_force_stop_group (group)
  - Forces a group stop

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

Export functions

- emaj_snap_group (group, directory, copy_options)
  - Snaps all tables and sequences of a group on individual files into a directory
- emaj_snap_log_group (group, start_mark, end_mark, directory, copy_options)
  - Snaps part of all log tables and sequences of a group on individual files into a directory
- emaj_generate_sql (group, start_mark, end_mark, file_pathname)
  - Generates a sql script replaying updates recorded between 2 marks

Other functions

- emaj_find_previous_mark_group (group, timestamp) or emaj_find_previous_mark_group (group, mark)
  - Retrieves the mark name immediately preceeding a point in time or another mark
- emaj_verify_all ()
  - Verifies the E-Maj environment consistency
- emaj_estimate_rollback_duration (group, mark)
  - Estimates the time needed to rollback a group to a mark

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> [-l]
- Other options: --help, -v, --version
- Needs php with the PostgreSQL extension

For large databases...

- Dedicated tablespaces may be used for log tables and indexes
  - tspemaj tablespace used by default if it exists
  - To use other tablespaces,
    - Create them
    - Configure its use in emaj_group_def table
- Secondary E-Maj schemas may contain log objects
  - To be configured in emaj_group_def table
  - Schemas are created and dropped by E-Maj

Reliability

- Many checks, in particular at start_group, set_mark_group and rollback_group times:
  - Do all tables, sequences, functions, triggers exist ?
  - Are we sure that all application tables and their log tables are consistent (columns names and types) ?
- 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

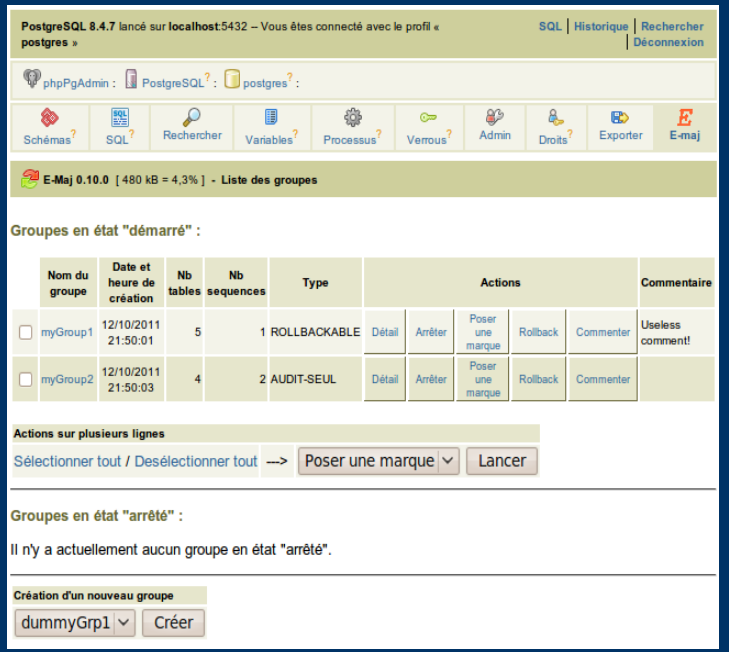
- 2 roles that can be granted :
  - emaj_adm for ... emaj administrators
  - emaj_viewer to just be able to look at log tables
- E-Maj objects are only created by a super-user or a member of emaj_adm
- No other right is granted on the E-Maj schemas, tables and functions
- Log triggers are created as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

Performances

- Log overhead
  - Highly depends on hardware and on the application read/write SQL 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

- Minimum PostgreSQL version = 8.2
- Every application table belonging to a rollbackable group needs a PRIMARY KEY
- Schema name length + application table name length <= 52 characters
- DDL or TRUNCATE operations cannot be managed by E-Maj.
  - TRUNCATEs are just blocked when pg version >= 8.4

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, Ronan Dunklau
  - People who already contacted me for comments, requests...
- Feel free to email: phb<dot>emaj<at>free<dot>fr