

E-Maj 2.1.0

-

a PostgreSQL extension

French acronym for

Enregistrement des Mises A Jour

i.e. "updates recording"

Components

- E-Maj
 - PostgreSQL extension
 - Open Source (GPL license)
 - Available on
 - pgxn.org
 - [github](https://github.com/beaud76/emaj) (<https://github.com/beaud76/emaj>)
- 2 web clients
 - Plug-in for phpPgAdmin 5.1+ (https://github.com/beaud76/emaj_ppa_plugin)
 - Emaj_web application (https://github.com/beaud76/emaj_web)
- Documentation source also available on [github](https://github.com/beaud76/emaj_doc) (https://github.com/beaud76/emaj_doc)



E-Maj objectives

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

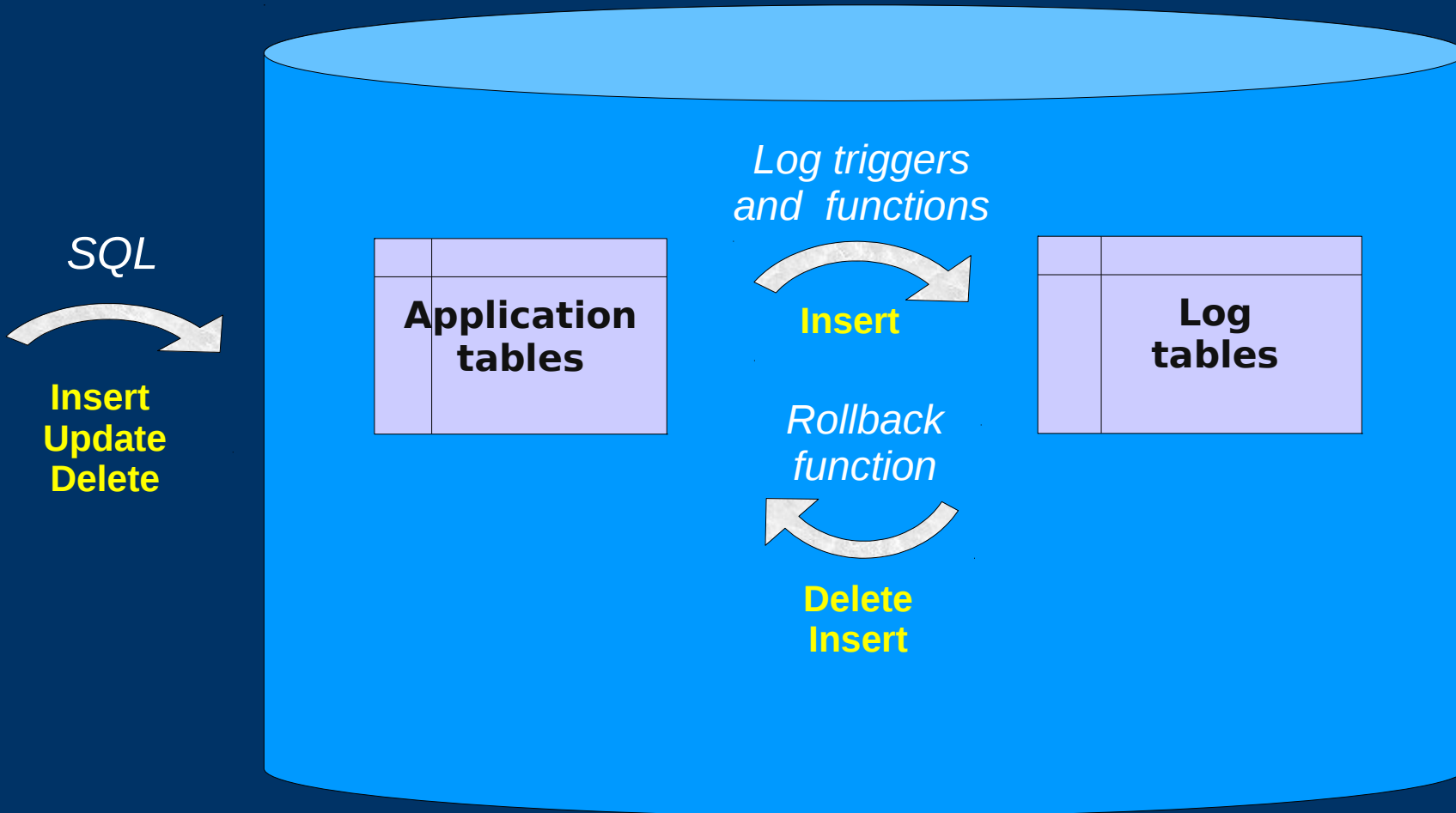
E-Maj Requirements

- 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
-
-

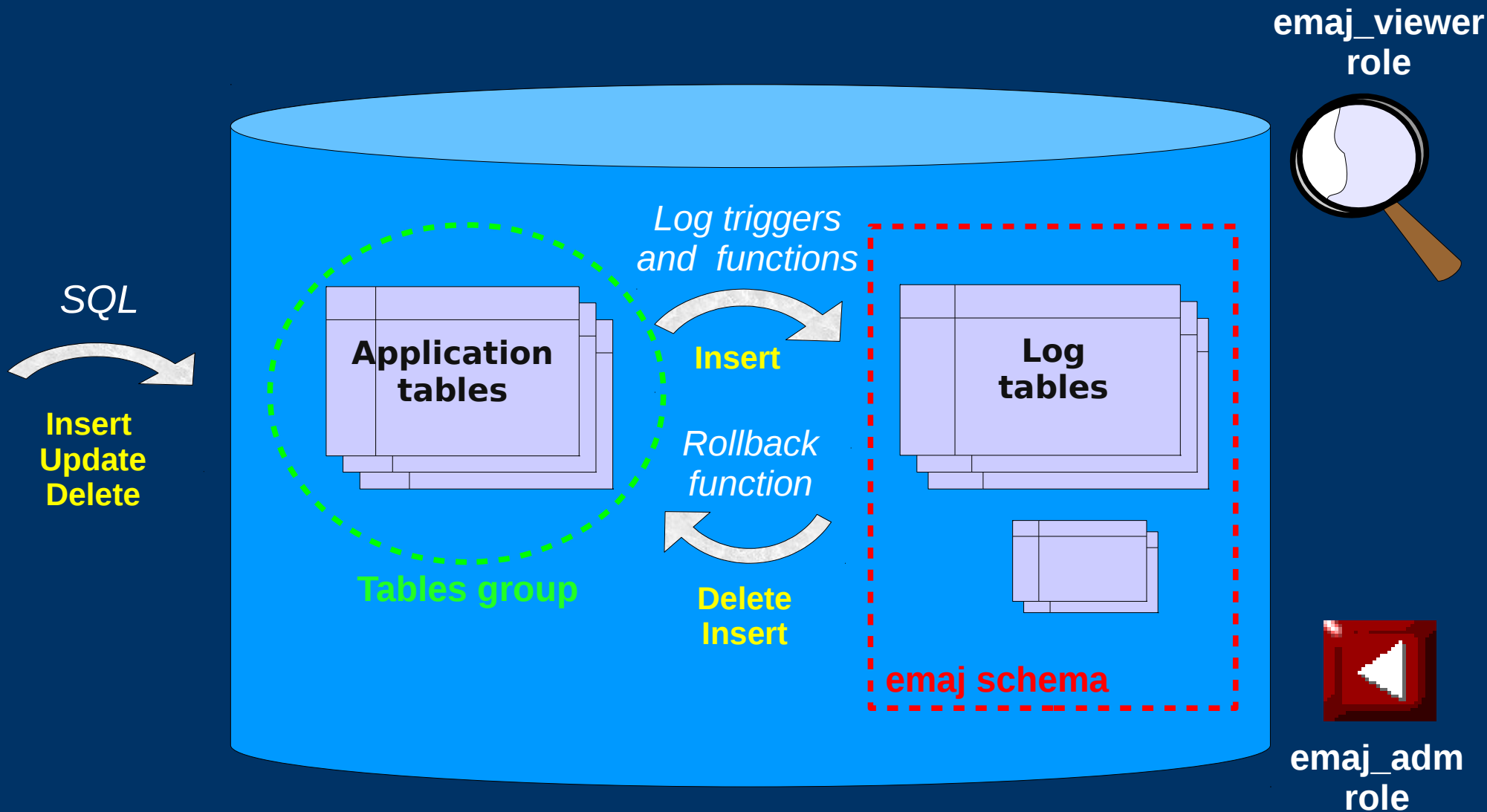
E-Maj Concepts

- **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

- Download and install the extension in the *share/postgres/extension* directory of the PostgreSQL software
 - Copy and adapt the *sql/emaj.control* file directly into the *share/postgres/extension* directory
 - Connect to the target database as superuser and execute
 - **CREATE EXTENSION IF NOT EXISTS DBLINK;** (recommended)
 - **CREATE EXTENSION EMAJ;**
 - The installation in the database adds:
 - 1 schema 'emaj' with about 90 functions, 12 technical tables, 7 types, 1 view and 1 sequence
 - 2 event triggers
 - 2 roles
-
-

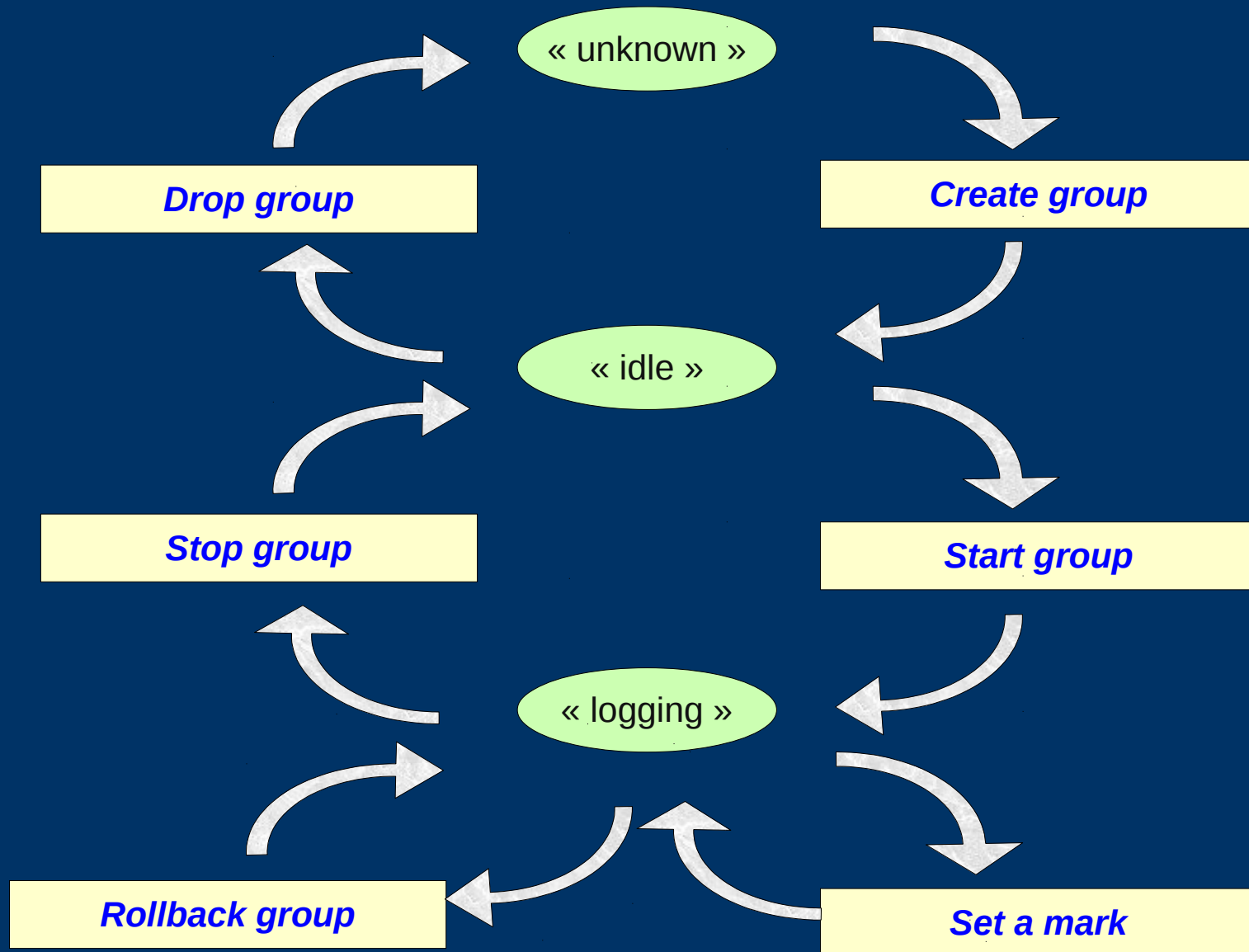
E-Maj Initialisation

- 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 + 1 sequence into an 'emaj' schema
 - 1 trigger + 1 function to log table updates
 - SELECT **emaj_drop_group** (group)
... drops a previously created group

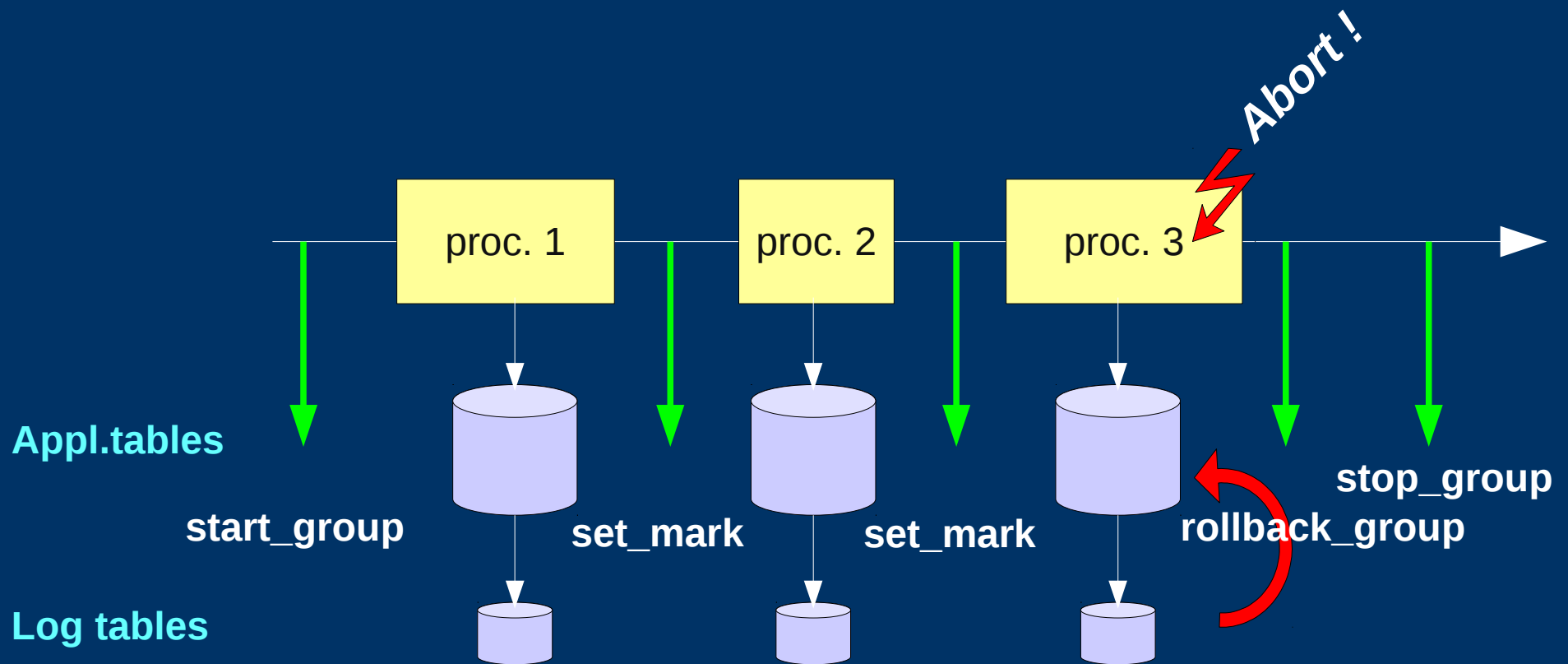
E-Maj: Main functions

- **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, false)
 - Rolls back tables and sequences of the group to their state at mark set
 - **emaj_logged_rollback_group** (group, mark, false)
 - 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 ...



Log tables

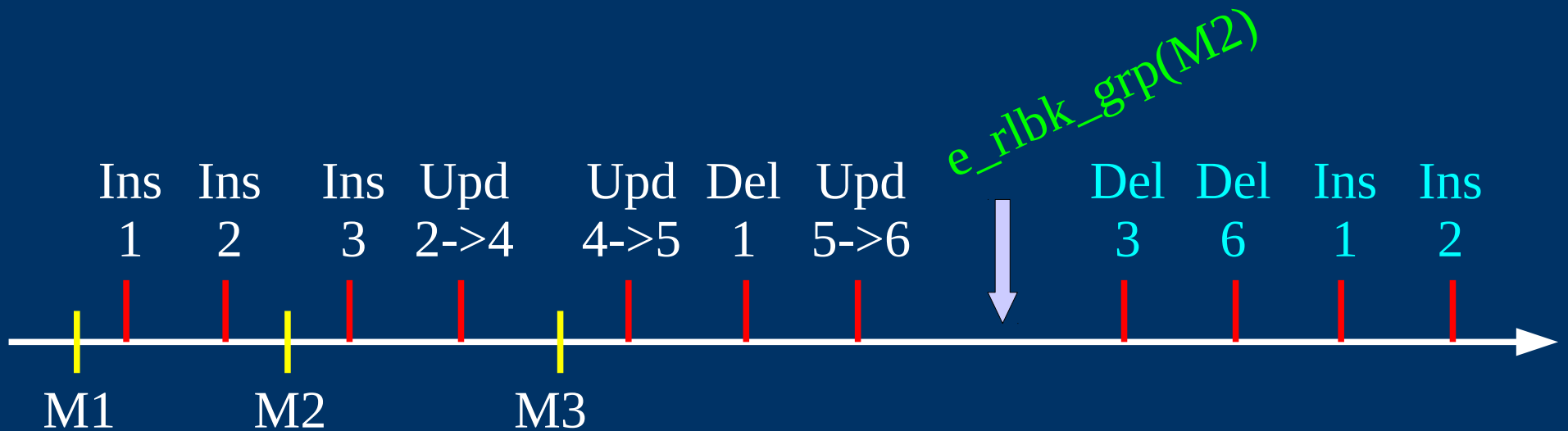
- Examining log tables may largely help application debugging
 - A log table contains
 - The same columns as the associated application table
 - And some technical columns
 - A changed row in an application table generates
 - 1 log row for an INSERT (new row)
 - 1 log row for a DELETE (old row)
 - 2 log rows for an UPDATE (old and new rows)
 - A TRUNCATE generates 1 log row
-
-

Technical columns of log tables

- 8 technical columns at the end of each log row
 - emaj_verb : type of change - INS/UPD/DEL/TRU
 - emaj_tuple : type of log row - OLD/NEW
 - emaj_gid : internal sequence number
 - emaj_changed : change timestamp - clock_timestamp()
 - emaj_txid : transaction identifier - txid_current()
 - emaj_user : client connection role - session_user
 - emaj_user_ip : client ip address - inet_client_addr()
 - emaj_user_port : client ip port - inet_client_port()
-
-

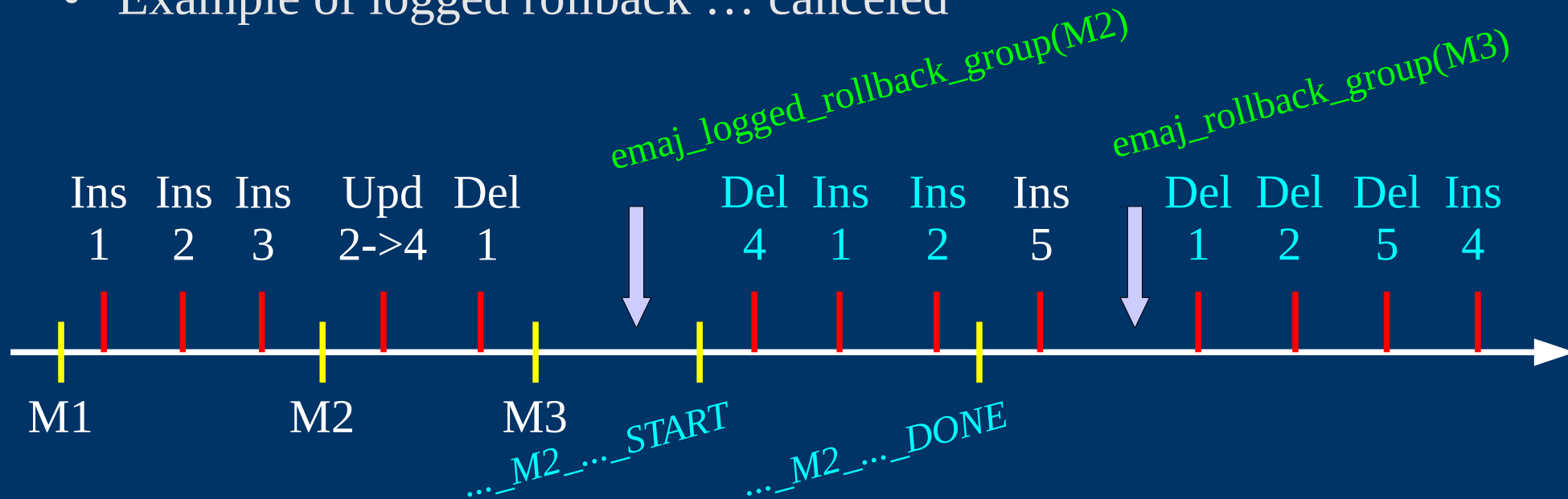
« Simple Rollback »

- Log triggers are de-activated
- Each table is set to its correct state using an optimized algorithm
 - Processes only once each primary key
 - Takes into account potential foreign keys
- Cancelled logs and marks are deleted



« *Logged Rollback* » (1/2)

- Log triggers are NOT de-activated
- Cancelled logs and marks are kept
- Mark automatically set before and after the rollback
 - `RLBK_<marque>_<HH.MI.SS.MS>_START`
 - `RLBK_<marque>_<HH.MI.SS.MS>_DONE`
- Example of logged rollback ... canceled



« *Logged Rollback* » (2/2)

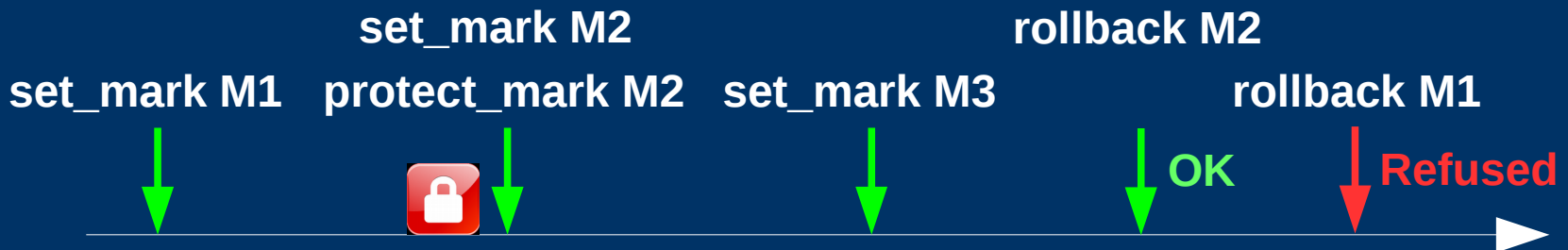
- Ideal for tests : avoid numerous intermediate saves to replay old tests
 - During the rollback operation, tables remain accessible for reads
 - A logged rollback can be later transformed into a simple rollback => "rollback consolidation"
 - Intermediate logs and marks are deleted, reclaiming log space
 - `emaj_consolidate_rollback_group(group, end_rollback_mark)`
 - Tables may be updated in parallel
-
-

Monitor in progress rollbacks

- Needs dblink, and the setting of the “dblink_user_password” parameter in the emaj_param table
 - `SELECT * FROM emaj.emaj_rollback_activity();`
 - Returns
 - Rollback characteristics (group, mark...)
 - Rollback state
 - Elapse time
 - Estimate of the remaining duration and the % done
-
-

Protection against accidental rollbacks

- 2 functions to manage the tables group protection
 - `emaj_protect_group` (group)
 - `emaj_unprotect_group` (group)
- 2 functions to manage the marks protection
 - `emaj_protect_mark_group` (group, mark)
 - `emaj_unprotect_mark_group` (group, mark)



E-Maj possible usages

- Largely helps **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 if the logs part to erase is important)



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, false)
 - `emaj_logged_rollback_groups` (groups array, mark, false)
 - 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_rename_mark_group` (group, old mark, new mark)
 - Renames a mark
 - `emaj_delete_mark_group` (group, mark)
 - Suppress a mark
 - `emaj_delete_before_mark_group` (group, mark)
 - Suppress all marks preceding the supplied mark
-
-

Other groups management functions

- `emaj_alter_group` (group) / `emaj_alter_groups` (groups)
 - Processes the changes about the definition of one or several tables groups
 - `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
-
-

Other rollbacks management functions

- `emaj_estimate_rollback_group` (group, mark)
 - Estimates the time needed to rollback a group to a mark
 - `emaj_consolidate_rollback_group` (group, mark)
 - Consolidate a logged rollback identified by the tables group and the generated end rollback mark. It transforms an unlogged rollback into a logged rollback by deleting all marks and logs between the rollback target mark and the end rollback mark.
 - `emaj_get_consolidable_rollbacks` ()
 - List rollback operations that may be consolidated
-
-

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_gen_sql_group** (group, start_mark, end_mark, file_pathname [, tables/seq_list])
emaj_gen_sql_groups (groups,...)
 - Generates a sql script replaying updates recorded between 2 marks for all or several tables and sequences of a tables group
-
-

Other functions

- `emaj_find_previous_mark_group` (group, timestamp) or `emaj_find_previous_mark_group` (group, mark)
 - Retrieves the mark name immediately preceding a point in time or another mark
- `emaj_verify_all` ()
 - Verifies the E-Maj environment consistency

For large databases...

- Dedicated tablespaces may be used for log tables and indexes
 - To be configured in emaj_group_def table
 - Must have been created previously
 - Secondary E-Maj schemas may contain log objects
 - To be configured in emaj_group_def table
 - Schemas are created and dropped by E-Maj
-
-

Parallel rollback client

- A php module performs parallel restore
 - Acts as a client for the database
 - Automatically spreads the tables to rollback into a given number of sessions
 - Performs the parallel rollback in a unique transaction (→ `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
-
-

Rollbacks monitoring client

- A php module to monitor in progress or recently completed rollback operations
- `emajRollbackMonitor.php` -d <database> -h <host> -p <port> -U <user> -W <password> -n <#iterations> -i <refresh_interval_in_seconds> -l <#completed_rollbacks> -a <completed_rollbacks_history_in_hours>
- Other options : --help, -v, --version

```
E-Maj (version 1.1.0) - Monitoring rollbacks activity
```

```
-----  
04/07/2013 - 12:07:17  
** rollback 35 started at 2013-07-04 12:06:21.474217+02 for groups {myGroup1}  
   status: COMMITTED ; ended at 2013-07-04 12:06:21.787615+02  
-> rollback 36 started at 2013-07-04 12:04:31.769992+02 for groups {group1232}  
   status: EXECUTING ; completion 89 % ; 00:00:20 remaining  
-> rollback 37 started at 2013-07-04 12:04:21.894546+02 for groups {group1233}  
   status: LOCKING ; completion 0 % ; 00:22:20 remaining
```


Reliability (1/2)

- 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
-
-

Reliability (2/2)

- **TRUNCATE** statements are blocked for logging rollbackable groups
- For the most recent PostgreSQL versions (9.3+), some “event triggers” block some unattended component drops or changes (tables, sequences, functions...)
 - 2 functions to disable/re-enable the blocking
 - `emaj_disable_protection_by_event_triggers()`
 - `emaj_enable_protection_by_event_triggers()`

Security

- 2 roles that can be granted :
 - emaj_adm for ... E-Maj administration
 - emaj_viewer to just be able to look at E-Maj objects (logs, marks, statistics)
 - 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...)

2 web clients

- PhpPgAdmin plug-in
 - Fully integrated into phpPgAdmin 5.1+
- Emaj_web independent application
- Share the same functionalities
 - Help administrators and viewers
 - Show all E-Maj objects (groups, marks...) and their attributes
 - Allow all possible actions on E-Maj objects

The screenshot displays the phpPgAdmin interface for PostgreSQL 9.1.1. The top navigation bar includes 'E-Maj env.', 'Groups conf.', 'Groups', and 'Rollback op.'. The main content area is titled 'Tables groups list' and shows two sections: 'Tables groups in "LOGGING" state' and 'Tables groups in "IDLE" state'. Each section contains a table with columns for Group, Creation date/time, # tables, # sequences, Type, # marks, Actions, and Comment. The 'LOGGING' section lists 'myGroup1' and 'myGroup2', both of type 'ROLLBACK-ABLE'. The 'IDLE' section lists 'phil's group#3' of type 'AUDIT-ONLY'. Below each table is an 'Actions on multiple lines' section with buttons for 'Select all / Unselect all', 'Set a mark', and 'Execute'. At the bottom, there is a 'Creation of a new tables group' section with a dropdown menu set to 'dummyGrp1' and a 'Create' button.

Group	Creation date/time	# tables	# sequences	Type	# marks	Actions	Comment
<input type="checkbox"/> myGroup1	09/08/2013 12:52:38	5	1	ROLLBACK-ABLE	3	Detail Stop Set a mark Rollback Set a comment	Useless comment!
<input type="checkbox"/> myGroup2	09/08/2013 12:52:38	4	2	ROLLBACK-ABLE	2	Detail Stop Set a mark Rollback Set a comment	

Group	Creation date/time	# tables	# sequences	Type	# marks	Actions	Comment
<input type="checkbox"/> phil's group#3	09/08/2013 12:52:38	2	1	AUDIT-ONLY	0	Detail Start Reset Drop #!er Set a comment	

Current limits

- Since E-Maj 2.0.0, the minimum required PostgreSQL version is **9.1**
 - Every application table belonging to a rollbackable group needs a **PRIMARY KEY**
 - **DDL** statement cannot be managed by E-Maj
-
-

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, Don Levine
 - People who already contacted me for comments, requests...
 - Feel free to email: [phb<dot>emaj<at>free<dot>fr](mailto:phb@emaj.free.fr)
-
-