



E-Maj

Let your PostgreSQL data
travel back in time

French acronym for
*"**E**nregistrement des **M**ises **A** **J**our"*
i.e. "updates recording"

What is E-Maj for?

- E-Maj allows the data content to **travel back in time**, with a **table level granularity**.
- By recording changes on sets of application tables, it is possible to
 - **Count** them (statistic function).
 - Easily **view** them (audit function).
 - **Revert** them ("rollback" function).
 - **Replay** them (script generation, or revert a revert...).
- Usable with:
 - Applications in test or in production.
 - Databases of all sizes.

Key Benefits

- In **Test** Environment
 - **Testing**: Cancel changes generated by the application to easily repeat tests.
 - **Debugging**: Quickly examine data changes.
- In **Production** Environment
 - **Recover from failures**: Roll back processes without restoring the entire instance.
 - **Fine-grained recovery**: Avoid losing entire batch processing nights by recovering from failures at the right stable point in time.
 - **Efficient for large tables** with few updates.

Components

- **E-Maj** core
 - PostgreSQL extension
 - Open Source (GPL licence)
 - Download from [pgxn.org](https://pgxn.org/dist/e-maj/) - <https://pgxn.org/dist/e-maj/>
 - Sources available on [github.com](https://github.com/dalibo/emaj) - <https://github.com/dalibo/emaj>
- **Emaj_web**
 - Web-based client - https://github.com/dalibo/emaj_web
- **Documentation**
 - Available in English and French - <https://emaj.readthedocs.io/en/latest/>



Design Principles

- **Reliability**
 - Guaranteed data integrity after changes cancellation.
 - Support all standart objects (tables, sequences, constraints, etc.).
- **Ease of Use**
 - Intuitive for DBAs, production teams, application developpers and testers.
 - Scriptable for automated production workflows.
- **Performance**
 - Limited logging overhead.
 - Acceptable “rollback” duration.
- **Security**
- **Maintenability**
 - PL/pgSQL (no changes in Postgres core).

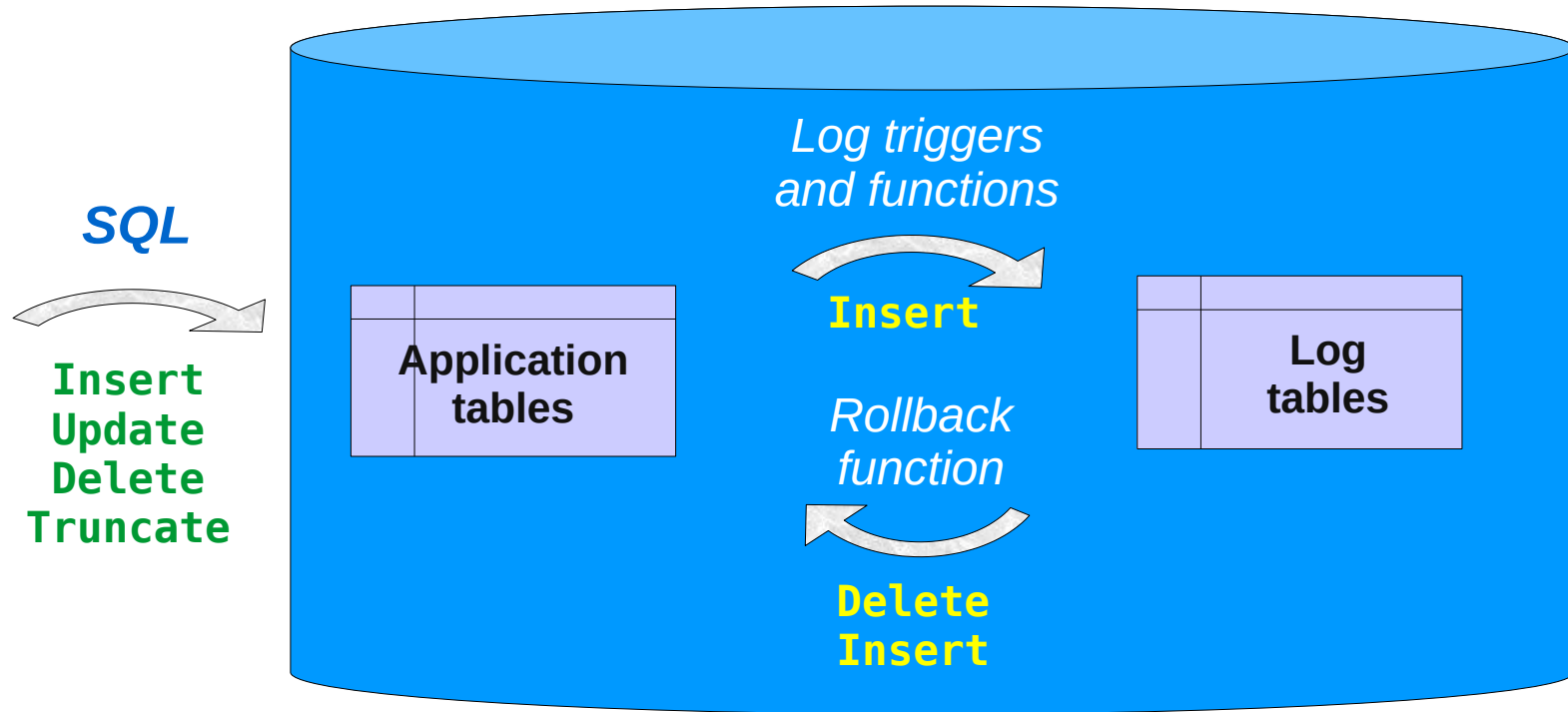
Core Concepts

- **Table Group**
 - Set of tables/sequences (from one or several schemas) with the **same life cycle**.
- **Mark**
 - **Stable point** in the life of a “table group”.
 - Identified by a name.
 - Used to restore the group to a previous state.
- **E-Maj Rollback**
 - **Restores** a “table group” to a previously set “mark”.
 - Different from RDMS transaction rollbacks.
 - RDBMS-rollback: cancels the current transaction.
 - E-Maj rollback: cancels changes from several committed transactions

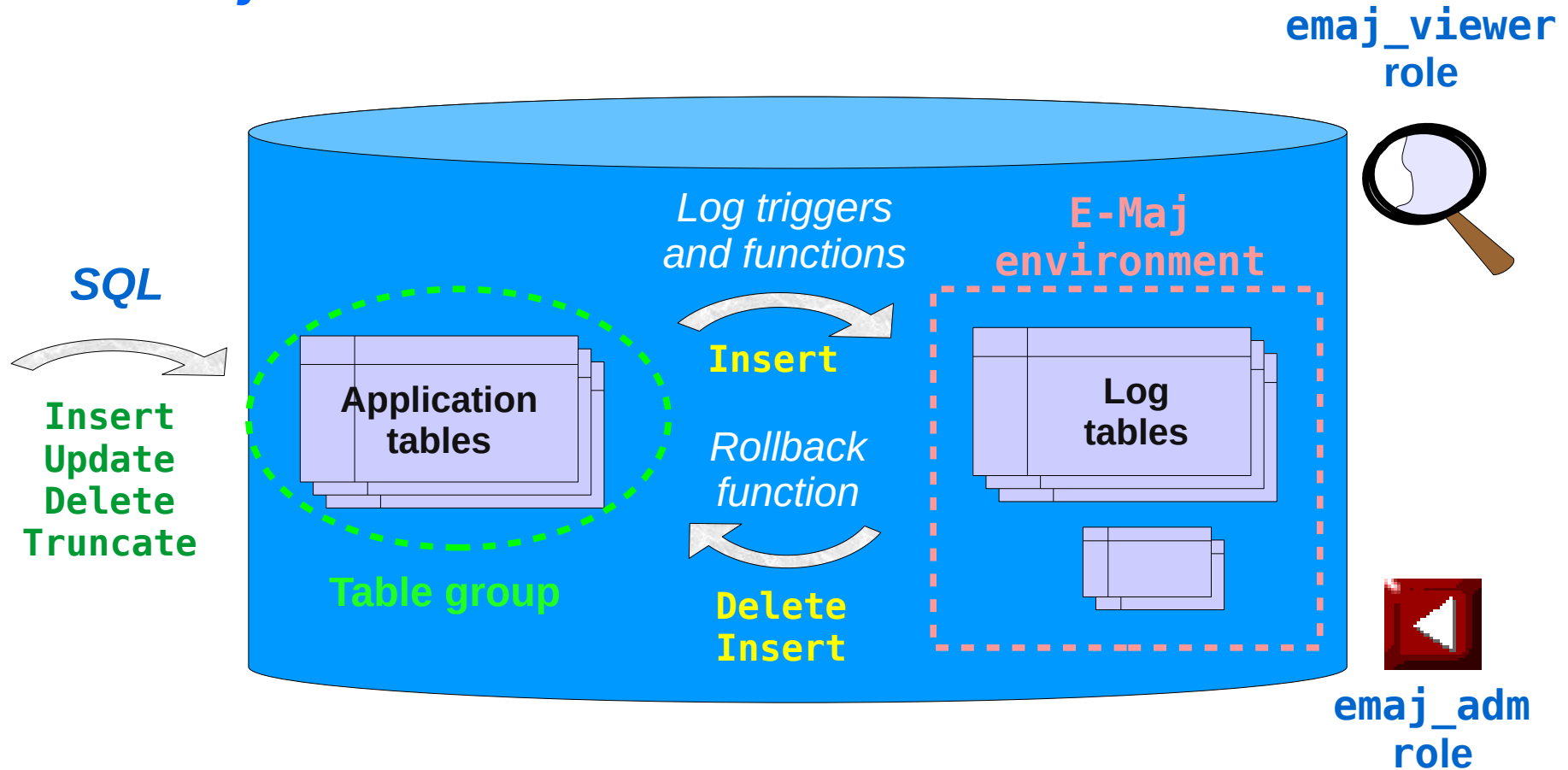
Concepts (2)

- Table Group Type
 - By default, **rollbackable**
 - May be created as **audit-only**
 - E-Maj rollbacks are not possible
 - Useful to capture data changes for tables without PRIMARY KEY or of type UNLOGGED
- **Log session**
 - **Time interval** during which a table group **records changes**
 - Bounded by the table group **start** and **stop** actions.

How E-Maj Works: Trigger-Based Logging



Main Objects



Management of Application Sequences

- Sequence increments are not individually recorded
- At set mark time
 - The state of each sequence of the group is stored into an internal table
- At E-Maj rollback time
 - Each sequence is reset to its state recorded at the targeted mark

Install E-Maj

- Standard install
 - `pgxn install E-Maj --sudo`
 - Then as a superuser
 - `CREATE EXTENSION emaj CASCADE;`
- Install without superuser privileges (with a few limits)
 - Download from pgxn.org and unzip the extension
 - `psql ... -f sql/emaj-<version>.sql`
- Adds to the database
 - Required extensions (`dblink` and `btree_gist`, if needed)
 - 2 roles (`emaj_adm`, `emaj_viewer`)
 - The '`emaj`' schema, with: 21 technical tables, 12 types, 2 views, 1 sequence, 3 event triggers and more than 200 functions

Initialization

- 1) Create a table group
 - `SELECT emaj_create_group (group, is_rollbackable [,comment]);`
- 2) Assign tables and sequences
 - `SELECT emaj_assign_tables (schema, inclusion regexp, exclusion regexp, group);`
 - `SELECT emaj_assign_sequences (schema, inclusion regexp, exclusion regexp, group);`
 - Ex: all tables of a schema except those suffixed by sav:
`'.*', 'sav$'`
 - Creates for each application table: 1 log table, 1 log sequence, 1 log trigger and its function
- 3) Drop a table group
 - `SELECT emaj_drop_group (group)`

The 3 Main Functions to Manage Groups

- “Start” a group
 - `emaj_start_group (group [,mark] [,delete.logs] [logging.group.allowed])`
activates log triggers and sets a first mark
- Set a mark
 - `emaj_set_mark_group (group [,mark] [,comment])`
sets an intermediate mark
- “Stop” a group
 - `emaj_stop_group (group [,mark] [,delete.logs] [idle.group.allowed])`
deactivates the log triggers
 - A rollback is not possible anymore
- % in mark name = current time (e.g., M1_23.59.59.123456)

Examining Logs

- May largely help the application debugging.
- Each application table has its own **log table**.
 - `emaj_<schema>.<table>_log`
- A log table contains:
 - The same columns as its related application table.
 - And some technical columns.
- Log entries per row change:
 - **INSERT**: 1 row (image of the **NEW** row).
 - **DELETE, TRUNCATE**: 1 row (image of the **OLD** row).
 - **UPDATE**: 2 rows (image of the **OLD** and the **NEW** rows).
- A TRUNCATE generates also a single log row.

Log Tables Technical Columns

- 6 technical columns
 - `emaj_verb` : SQL statement type (INS, UPD, DEL, TRU)
 - `emaj_tuple` : Row type (OLD, NEW)
 - `emaj_gid` : Internal sequence number
 - `emaj_changed` : Timestamp (`clock_timestamp()`)
 - `emaj_txid` : Transaction ID (`txid_current()`)
 - `emaj_user` : Role that performed the change (`session_user`)
- ... and some others can be added
- It is possible to identify clients and transactions, and analyze the timing of the program execution

Support of Generated Columns on Application Tables (*GENERATED ALWAYS AS expression*)

Type	Physical (STORED)	Virtual
Minimum Postgres version	12	18
Structure in log table	A standart column	
Content in log table	Expression result	NULL
SQL visualisation of the changes impact	Direct examination of the log table column	Use the expression
Expression change (<i>ALTER TABLE ... ALTER COLUMN ... SET EXPRESSION ...</i>)	Blocked by E-Maj (*)	Possible
Expression drop (<i>ALTER TABLE ... ALTER COLUMN ... DROP EXPRESSION</i>)	Possible, but risky for data integrity at subsequent E-Maj rollback operations	Not applicable (forbidden by PostgreSQL)
Column transformation: generated $\leftrightarrow$ standart (<i>ALTER TABLE ... DROP COLUMN ..., ADD COLUMN ...</i>)	Detected by E-Maj at mark set and rollback attempts (*)	

(*) the table must be temporarily removed from its group

Counting Data Changes – Group Level Statistics

- At **table group** level and for a given **marks interval**
- `emaj_log_stat_group (group, start_mark, end_mark)`
 - Quickly returns an **estimate** of recorded changes per table
- `emaj_detailed_log_stat_group (group, start_mark, end_mark)`
 - Scans log tables and returns **precise** statistics on their content, per table, statement type (INSERT / UPDATE / DELETE / TRUNCATE) and ROLE
- `emaj_sequence_stat_group (group, start_mark, end_mark)`
 - Returns the number of increments per sequence

Counting Data Changes – Elementary Level Statistics

- Quickly return an estimate of changes for **one table** for each elementary marks interval on a given time frame
 - `emaj_log_stat_table (schema, table, start_date_time, end_date_time)`
 - `emaj_log_stat_table (schema, table, start_group, start_mark, end_group, end_mark)`
- Return the number of increments for **one sequence** for each elementary marks interval on a given time frame
 - `emaj_log_stat_sequence (schema, sequence, start_date_time, end_date_time)`
 - `emaj_log_stat_sequence (schema, sequence, start_group, start_mark, end_group, end_mark)`

Rolling back Changes - the “Simple” Rollback

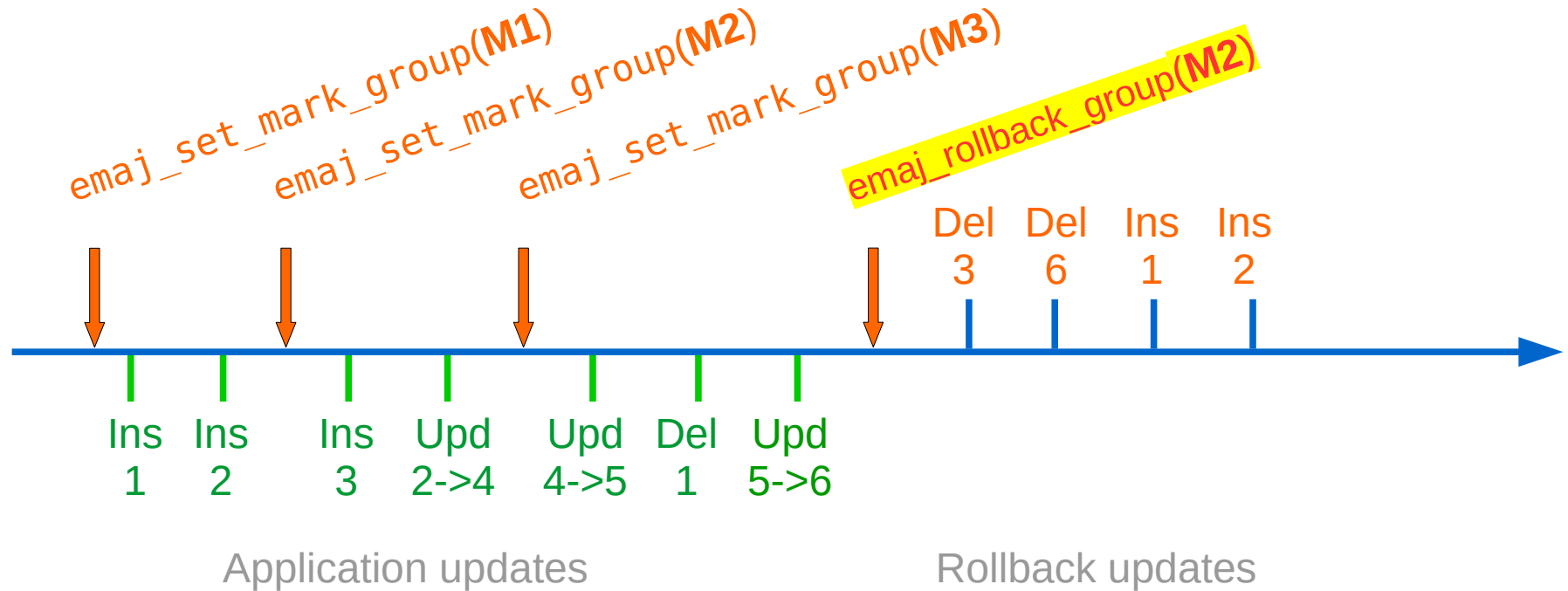
- To reset a table group in the state at a given mark
 - `emaj_rollback_group (group, mark [, false [, comment]])`
- How it works
 - Log triggers are disabled during the operation
 - Each table is restored to its state at target mark
 - Application sequences are reset to their state at the mark
 - Foreign keys are respected
 - Logs and marks after the target are deleted
 - => all what is after the rollback target mark is forgotten
- Tables remain accessible in read mode during the rollback

Rolling back Changes : the “Logged” Rollback

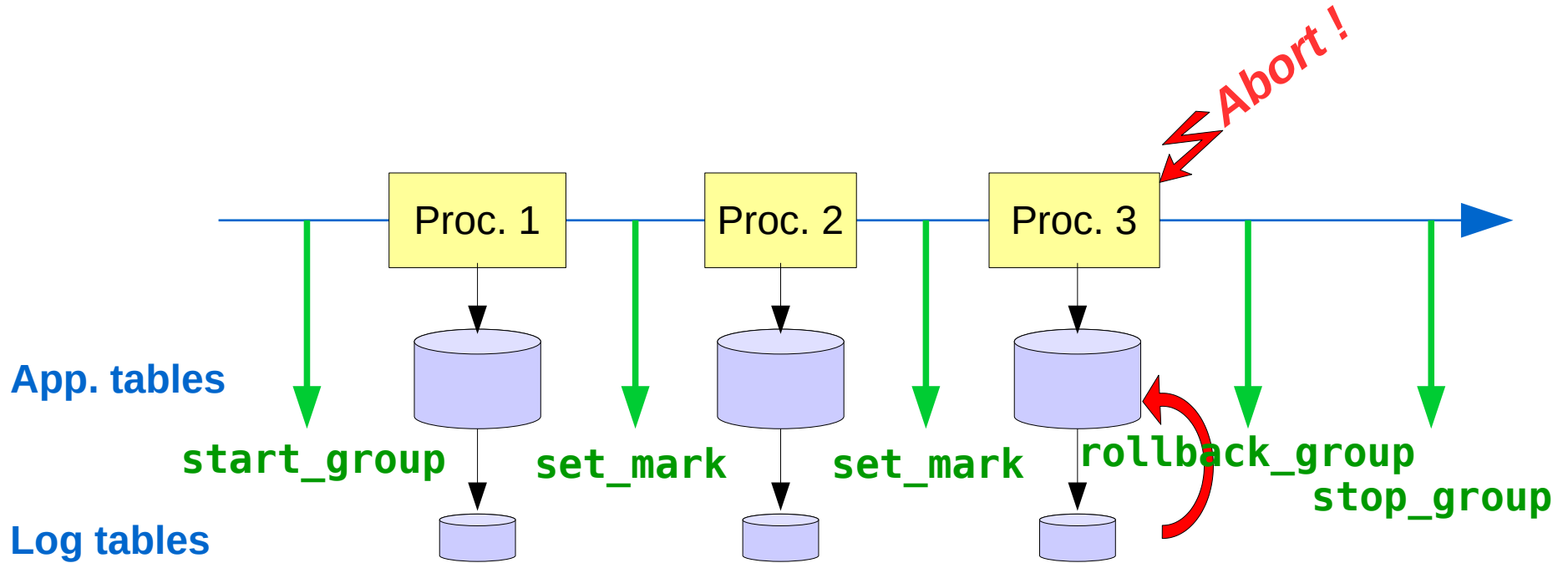
- `emaj_logged_rollback_group (group, mark[, false [, comment]])`
- Key differences with the “simple” rollback:
 - Log **triggers remain active** during the operation
=> Rollback changes are recorded
 - **Logs and marks are preserved.**
 - **2 automatic marks** are set before and after the rollback
 - `RLBK_<rollback id>_START`
 - `RLBK_<rollback id>_DONE`
- Allows reverting the E-Maj rollback ! And more generally let a table group travel back and forth in time !

Optimised Rollback Algorithm

- Processes each **primary key** value only **once**

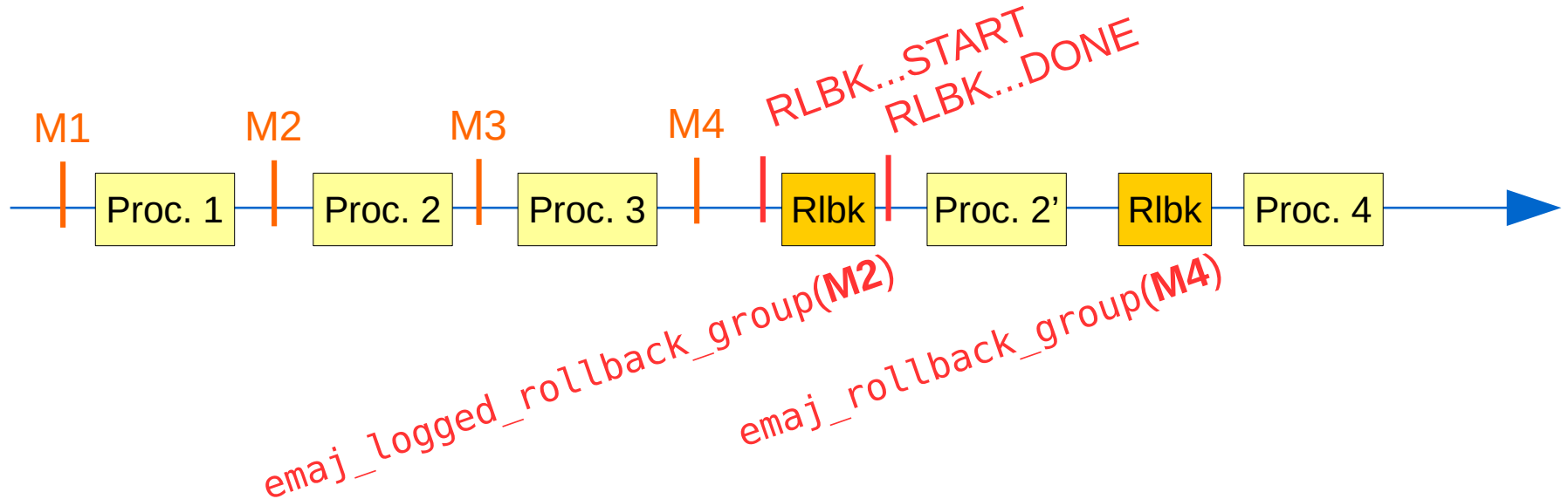


Typical Use Case in a Production Batch Processing



Typical Use Case in Test Environment

- Goal: 4 processings to test in sequence.
- After test 3, a new version of processing 2 must be re-tested.
- Then perform the remaining tests.



Estimating E-Maj Rollback Duration

- Helps decide if a rollback is feasible within the available time.
- A function estimates the time needed to rollback a group to a given mark
 - `emaj_estimate_rollback_group (group, mark)`

Executing a Parallel E-Maj Rollback

- A Perl client for parallel rollbacks
 - `emajParallelRollback.pl -d <database> -h <host> -p <port> -U <user> -W <password> -g <group_name or groups_list> -m <mark> -s <nb_sessions> [-l] [-c comment>]`
- Features:
 - Distributes tables across parallel sessions.
 - All sessions belong to a single transaction (2PC)
=> `max_prepared_transactions` >= nb sessions.
- Needs Perl + its PostgreSQL extension.

Monitoring in Progress E-Maj Rollbacks

- A function
 - `SELECT * FROM emaj.emaj_rollback_activity ();`
 - Returns
 - Rollback characteristics (group, mark, etc).
 - Current state and current duration.
 - Estimated remaining time and progress (%)
- Needs to setup the “`dblink_user_password`” parameter

Monitoring E-Maj Rollbacks

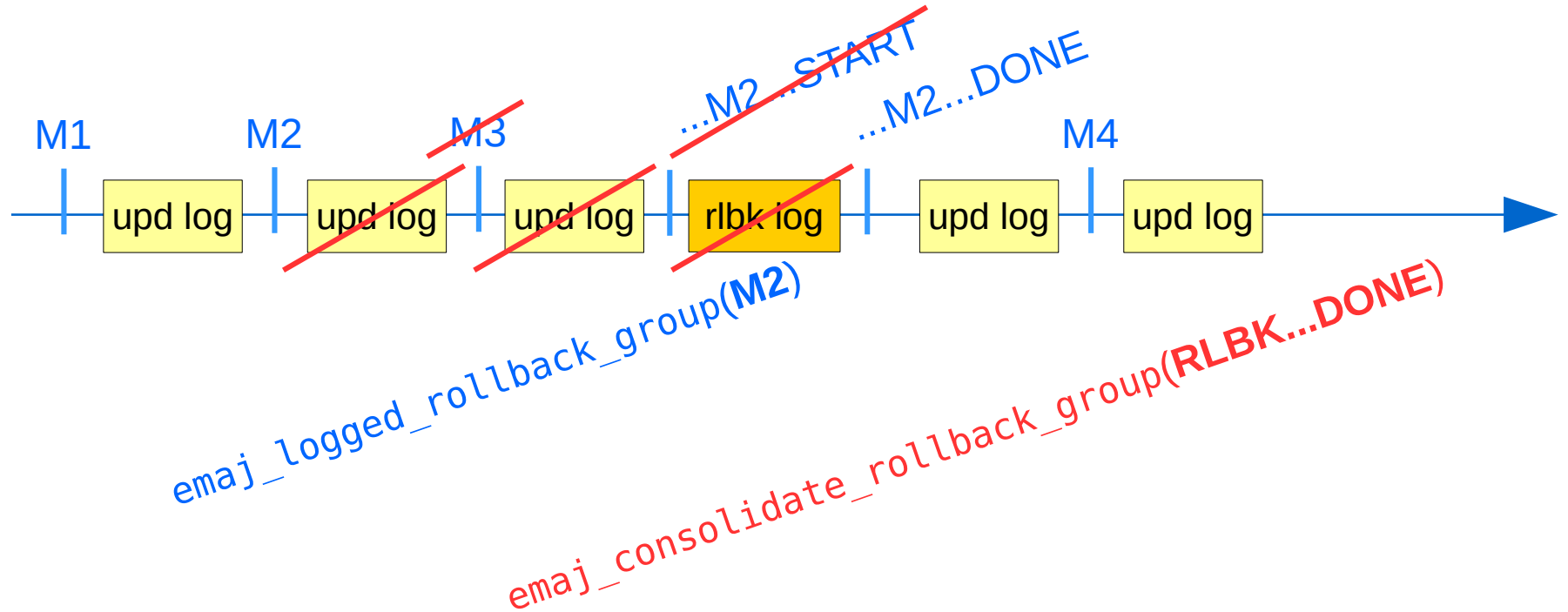
- A Perl client to monitor the in progress or completed rollbacks
 - `emajRollbackMonitor.pl -d <database> -h <host> -p <port> -U <user> -W <password> -n <nb_iterations> -i <refresh_rate_in_seconds> -l <nb_completed rollbacks> -a <completed_rollbacks_history_depth_in_hours>`

```
E-Maj (version 4.2.0) - Monitoring rollbacks activity
-----
21/03/2023 - 08:31:23
** rollback 34 started at 2023-03-21 08:31:16.777887+01 for groups {myGroup1}
   status: COMMITTED ; ended at 2023-03-21 08:31:16.9553+01
** rollback 35 started at 2023-03-21 08:31:17.180421+01 for groups {myGroup1}
   status: COMMITTED ; ended at 2023-03-21 08:31:17.480194+01
-> rollback 36 started at 2023-03-21 08:29:26.003502+01 for groups {group20101}
   status: EXECUTING ; completion 85 %; 00:00:20 remaining
```

Consolidating a “Logged” Rollback

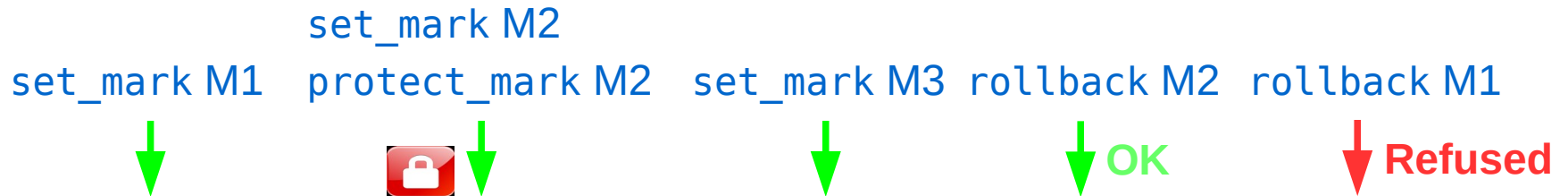
- Transform a “logged rollback” into a “simple rollback”, to free up log space.
- `emaj_consolidate_rollback_group (group, end_rollback_mark)`
- Intermediate logs and marks are deleted.
- Tables can be updated during the consolidation.
- To list of consolidable rollbacks:
 - `emaj_get_consolidable_rollbacks ()`

Example of E-Maj Rollback Consolidation



Protecting Against Unattended E-Maj Rollbacks

- Protect a table group.
 - `emaj_protect_group (group)`
 - `emaj_unprotect_group (group)`
- Protect a mark = block rollbacks to earlier marks.
 - `emaj_protect_mark_group (group, mark)`
 - `emaj_unprotect_mark_group (group, mark)`



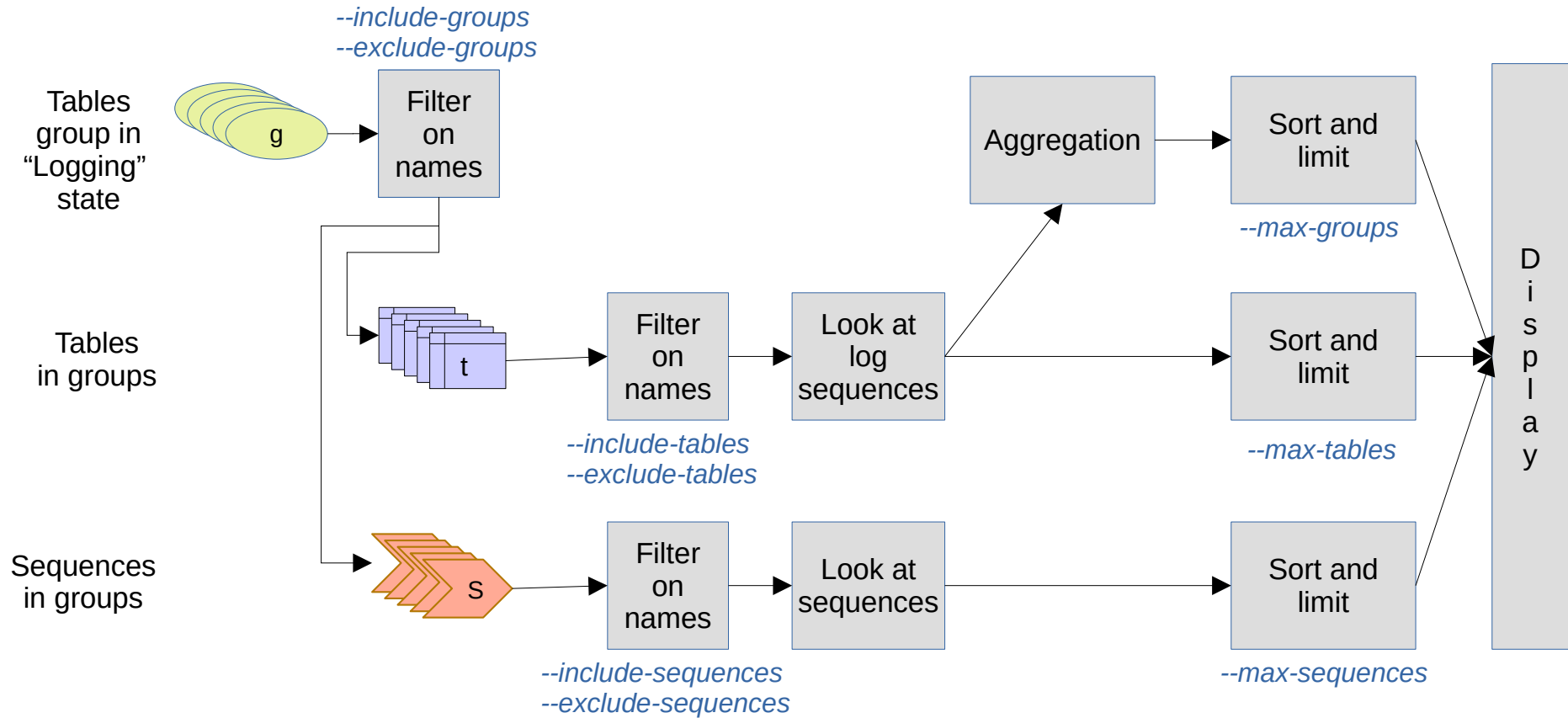
Monitoring the Changes Recording: emajStat

- A Perl client.
- Displays #changes on tables and sequences since the latest mark and the previous display, in absolute value and changes/sec.
- Many options to filter groups, tables and sequences, define the refresh parameters, ...
 - For the details: `emajStat.pl --help`

```
E-Maj (version 4.5.0) - Monitoring logged changes on database regression (@127.0.0.1:5412)
-----
2024/08/15 08:12:59 - Logging: groups=2/3 tables=11/11 sequences=4/4 - Changes since 1.004 sec: 0 (0.000 c/s)

Group name + Latest mark + Changes since mark + Changes since prev.
myGroup1 | Multi-1 (2024/08/15 08:12:38) | 359 (17.045 c/s) | 0 (0.000 c/s)
Table name + Group + Changes since mark + Changes since prev.
myschema1.mytbl1 | myGroup1 | 211 (10.018 c/s) | 0 (0.000 c/s)
myschema1.mytbl3 | myGroup1 | 60 ( 2.849 c/s) | 0 (0.000 c/s)
myschema1.mytbl2b | myGroup1 | 52 ( 2.469 c/s) | 0 (0.000 c/s)
myschema1.mytbl2 | myGroup1 | 27 ( 1.282 c/s) | 0 (0.000 c/s)
myschema1.mytbl4 | myGroup1 | 9 ( 0.427 c/s) | 0 (0.000 c/s)
Sequence name + Group + Changes since mark + Changes since prev.
myschema1.mytbl2b_col20_seq | myGroup1 | -5 (-0.237 c/s) | 0 (0.000 c/s)
myschema1.mytbl3_col31_seq | myGroup1 | -20 (-0.950 c/s) | 0 (0.000 c/s)
```

emajStat Logic and Parameters



Analysing Recorded Data Changes

- Process logs between 2 marks for all or some tables/sequences of a group.
- Export log table extracts and sequences on **files** via COPY
 - `emaj_dump_changes_group (group, start_mark, end_mark, options_list, tables/seq_array, directory)`
- Generate **SQL** to export changes.
 - In a file on the instance disk space:
`emaj_gen_sql_dump_changes_group (group, start_mark, end_mark, options_list, tables/seq_array, file)`
 - In an `emaj_temp_sql` temporary table, for any use by any client:
`emaj_gen_sql_dump_changes_group (group, start_mark, end_mark, options_list, tables/seq_array)`

Analysing Data Changes: the Options

- Common to `emaj_dump_changes_group()` and `emaj_gen_sql_dump_changes_group()`
 - **CONSOLIDATION** = NONE (default) | PARTIAL | FULL.
 - **EMAJ_COLUMNS** = ALL | MIN | (list): selects E-Maj technical columns.
 - **COLS_ORDER** = LOG_TABLE | PK: sets the order of delivered columns.
 - **ORDER_BY** = PK | TIME: sets the order of delivered rows, by PK or emaj_gid.
 - **SEQUENCES_ONLY**: excludes tables.
 - **TABLES_ONLY**: excludes sequences.
- For `emaj_dump_changes_group()` only
 - **COPY_OPTIONS** = (options list): for the COPY TO generation.
 - **NO_EMPTY_FILES**: removes empty files (tables without changes).
- For `emaj_gen_sql_dump_changes_group()` only
 - **PSQL_COPY_DIR** = directory : generates a \copy for each statement, with this directory.
 - **PSQL_COPY_OPTIONS** = (liste options) : sets the \copy options.
 - **SQL_FORMAT** = RAW | PRETTY : formats each statement on 1 or several lines.

Analysing Data Changes: Consolidated View of Changes

- **Net outcome** of recorded changes, for a given time range and for each primary key.
- At most: 1 “**OLD**” row (the initial state) and 1 “**NEW**” row (the final state).
- Ex: if UPDATE ‘A’ → ‘B’ then UPDATE ‘B’ → ‘C’ => OLD = ‘A’ and NEW = ‘C’.
- Requires an **explicit PRIMARY KEY**.
- 2 consolidation modes:
 - “**Partial**”: ignore columns content.
 - “**Full**”: compares columns content
 - For a given PK, no change reported if “OLD” row = “NEW” row
 - Ex: no change reported if UPDATE ‘A’ → ‘B’ then UPDATE ‘B’ → ‘A’, or if INSERT then DELETE.
- **Sequences**
 - 1 “OLD” row and 1 “NEW” row for the initial and final sequence’s characteristics
 - In “Full consolidation” mode, no row if the sequence has not changed

Analyse Data Changes: emaj_temp_sql Temporary Table Structure

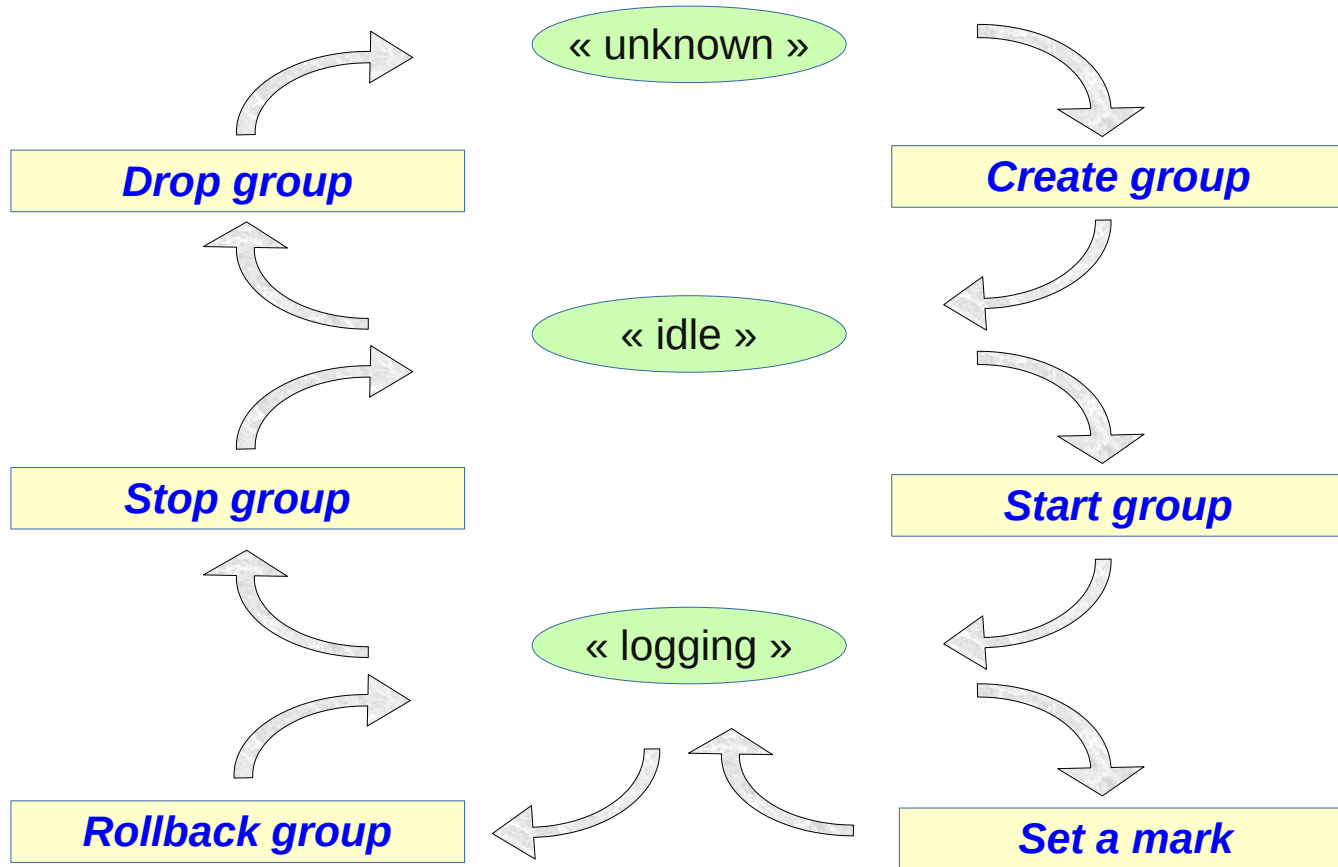
```
CREATE TEMP TABLE emaj_temp_sql (  
    sql_stmt_number      INT,           -- Statement number  
                                   -- (0 for the initial comment)  
    sql_line_number      INT,           -- Line number within the statement  
                                   -- (0 for the initial comment of the statement)  
    sql_rel_kind          TEXT,          -- Relation kind: "table" or "sequence"  
    sql_schema            TEXT,          -- Schema name  
    sql_tblseq            TEXT,          -- Table or sequence name  
    sql_first_mark        TEXT,          -- First mark name (for the table/sequence)  
    sql_last_mark         TEXT,          -- Last mark name (for the table/sequence)  
    sql_group             TEXT,          -- Table group owning the relation  
    sql_nb_changes         BIGINT,        -- Estimated number of changes to process  
    sql_file_name_suffix  TEXT,          -- File name suffix  
    sql_text              TEXT,          -- SQL statement text  
    sql_result            BIGINT         -- Column dedicated to the caller for its operations  
                                   -- (some other can be added with ALTER TABLE)  
);
```

An index on the 2 first columns

Replaying Data Changes

- **Generate a SQL script** to replay the elementary changes between 2 marks, for some or all tables and sequences of a group:
 - In the instance disk space:
`emaj_gen_sql_group (group, start_mark, end_mark,
dest_file [,tables/seq_list])`
 - Anywhere, via psql:
`SELECT emaj_gen_sql_group (group, start_mark,
end_mark, NULL [,tables/seq_list])
\copy (SELECT * FROM emaj_sql_script) TO 'dest_file'`
- **Use case:** Replicate changes in a test environment.

Table Group Lifecycle



Dynamic Adjustment of Table Groups

- **Add one or several tables:**
 - `emaj_assign_table(schema, table, group [, properties] [, mark])`
 - `emaj_assign_tables(schema, tables list, group [, properties] [, mark])`
 - `emaj_assign_tables(schema, selection filter, exclusion filter, group [, properties] [, mark])`
- **Properties** (in JSON format), to sets the priority and the tablespaces for log data and index (e.g., {"priority": 1}).
- **Selection and exclusion filters:** RegExp.
 - `emaj_assign_tables('myschema', 'tbl.*', '_sav$', 'mygroup')`
assigns to the group 'mygroup' all tables of schema 'myschema' starting with 'tbl' and not ending with '_sav'.

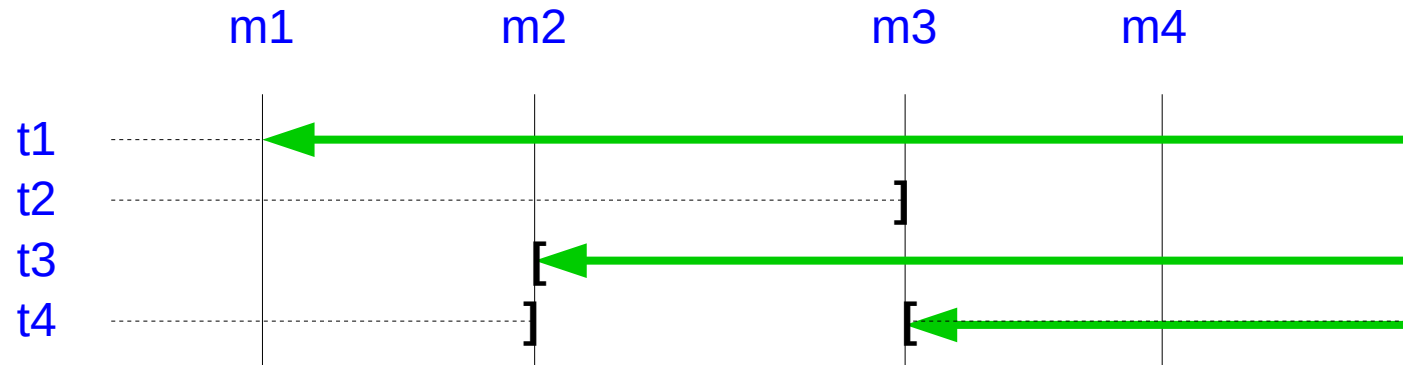
Table Groups Dynamic Adjustment

- Similarly:
 - `emaj_assign_sequence()` and `emaj_assign_sequences()`
 - `emaj_modify_table()` and `emaj_modify_tables()`
 - `emaj_move_table()` and `emaj_move_tables()`
 - `emaj_move_sequence()` and `emaj_move_sequences()`
 - `emaj_remove_table()` and `emaj_remove_tables()`
 - `emaj_remove_sequence()` and `emaj_remove_sequences()`
- To know the group a table or sequence is assigned to
 - `emaj_get_assigned_group_table()`
 - `emaj_get_assigned_group_sequence()`

Impact of Logging Group Structure Changes on Rollbacks

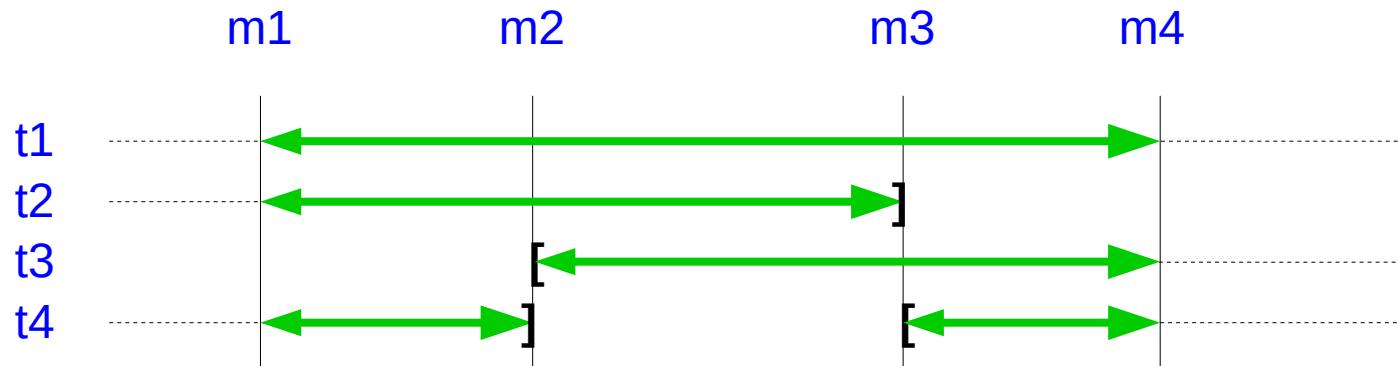
Table t2 removed at mark m3, t3 added at m2, t4 removed at m2 and added at m3

`emaj_rollback_group(<groupe>,'m1', true)` would process:



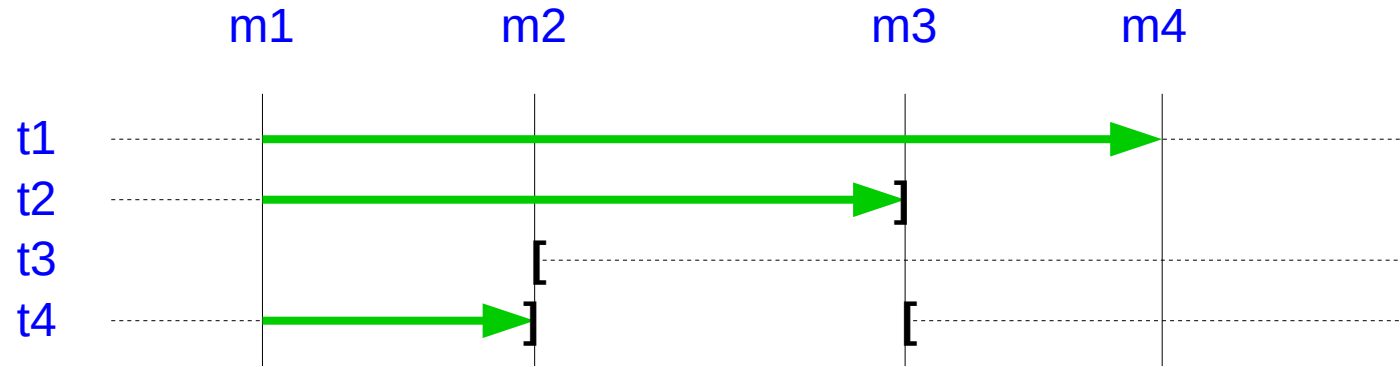
Impact of Logging Group Structure Changes on Statistics and Content Changes Extracts

`emaj_log_stat_group(<groupe>,'m1','m4')` and
`emaj_dump_changes_group(<groupe>,'m1','m4',...)` would report:



Impact of Logging Group Structure Changes on the SQL Scripts Generation

`emaj_gen_sql_group(<group>,'m1','m4')` would process:



Modifying a Table Structure in a LOGGING Group

- **No direct ALTER TABLE** if the group is LOGGING and if the action impacts the **log table structure**.
 - e.g. Rename the table, change its schema, add/drop/rename a column, change a column type, split/merge partitions.
- Solution:
 - **Remove** the table from its group.
 - **ALTER TABLE**.
 - **Re-add** the table to the group.
- Warning: **no E-Maj rollback** to a mark **prior** the structure change.
- Idem to rename a **sequence** or change its schema.

Processing Several Groups in a Single Operation

- “Multi-groups” variants of functions
 - `emaj_start_groups (groups_array, ... )`
 - `emaj_stop_groups (groups_array, ... )`
 - `emaj_set_mark_groups (groups_array, ... )`
 - `emaj_rollback_groups (groups_array, ... )`
 - `emaj_logged_rollback_groups (groups_array, ... )`
 - `emaj_log_stat_groups (groups_array, ... )`
 - `emaj_gen_sql_groups (groups_array, ... )`
- **Shared mark** across groups.
- Both PostgreSQL **syntaxes** for groups arrays
 - `ARRAY['group 1', 'group 2', ... ]`
 - `'{"group 1", "group 2", ... }'`

Exporting / Importing Groups Configurations

- In JSON format, direct or on file.
- Get the configuration:
 - `emaj_export_groups_configuration (NULL, groups_array)`
- Export the configuration into a file:
 - `emaj_export_groups_configuration (file, groups_array)`
- Import a configuration:
 - `emaj_import_groups_configuration (JSON_description, groups_array, ...)`
- Import a configuration from a file:
 - `emaj_import_groups_configuration (file, groups_array, ...)`

Managing marks

- Comment a mark for a group:
 - `emaj_comment_mark_group (group, mark, comment)`
- Rename a mark:
 - `emaj_rename_mark_group (group, old_name, new_name)`
- Delete a mark:
 - `emaj_delete_mark_group (group, mark)`
 - If the deleted mark is the first one, logs prior the second mark are deleted.
- Delete all marks prior a given mark:
 - `emaj_delete_before_mark_group (group, mark)`
 - Deletes logs prior the mark (it may take a long time...)

Managing mark (2)

- Search for marks:
 - `emaj_find_previous_mark_group (group, date-time)` returns the mark immediately preceeding a given timestamp.
 - `emaj_find_previous_mark_group (group, mark)` returns the mark immediately preceeding a given mark.
- “`EMAJ_LAST_MARK`” represents the last set mark for a group.
 - Usable for all parameters defining an existing mark.

Other Actions on Groups

- Comment a group (add/modify/suppress):
 - `emaj_comment_group (group, comment)`
- Purge log tables of a stopped group (anticipating its next restart):
 - `emaj_reset_group (group)`
- Force a group stop (if the normal stop fails):
 - `emaj_force_stop_group (group)`
- Snapshot a group (dump tables and sequences on files):
 - `emaj_snap_group (group, directory, copy_options)`
- Erase histories about a dropped table group:
 - `emaj_forget_group (group)`

E-Maj Parameters

- 3 general parameters:
 - ‘history_retention’ (def = 1 year) : traces retention delay.
 - ‘dblink_user_password’ : for E-Maj rollbacks.
 - ‘alter_log_table’ : to add columns to log tables.
- 6 parameters for the E-Maj rollback durations estimate model:
 - ‘fixed_step_rollback_duration’ (def = 2,5ms).
 - ‘fixed_dblink_rollback_duration’ (def = 4ms).
 - ‘fixed_table_rollback_duration’ (def = 1ms).
 - ‘avg_row_rollback_duration’ (def = 100μs).
 - ‘avg_row_delete_log_duration’ (def = 10μs).
 - ‘avg_fkey_check_duration’ (def = 20μs).

Manage E-Maj Parameters

- Look at parameters:
 - `emaj_all_param` view for administrators.
 - `emaj_visible_param` view for `emaj_viewer` roles.
- Modify a parameter:
 - `emaj_set_param (key, value)`
- Export or import parameters configurations:
 - `emaj_export_parameters_configuration ()`
 - `emaj_import_parameters_configuration ()`
 - In JSON format, direct or on file.

Useful Functions to Write Idempotent Scripts

- To test states:
 - When creating/dropping a table group:
 - `emaj_does_exist_group (group)`
 - When setting a mark:
 - `emaj_is_logging_group (group)`
 - `emaj_does_exist_mark_group (group, mark)`
- To build table groups arrays:
 - `emaj_get_groups(include regexp filter, exclude regexp filter)`
 - `emaj_get_logging_groups(include regexp filter, exclude regexp filter)`
 - `emaj_get_idle_groups(include filter, regexp exclude regexp filter)`

Other Actions

- Getting the current emaj extension version or drop the extension.
 - `emaj_get_version ()`
 - `emaj_drop_extension ()`
- Verifying the good health of the E-Maj installation.
 - `emaj_verify_all ()`
- Getting the current log table of a given application table.
 - `emaj_get_current_log_table ()`
- Manualy purging obsoletes traces.
 - `emaj_purge_histories ()`
- Creating/modifying/deleting a comment on a rollback.
 - `emaj_comment_rollback ()`

Temporary or permanent logging?

- **Temporary logging**
 - Start/stop group repeatedly.steps like
 - `emaj_start_group()`
 - repeat
 - processing
 - `emaj_set_mark()`
 - `emaj_stop_group()`
 - By default, logs and marks are automatically deleted at next start
 - But stops and starts set very heavy locks
- **Permanent logging** = no repeated group stop/restart
 - Obsolete logs and marks must be regularly deleted, using the `emaj_delete_before_mark()` function
 - The deletion can be costly if the volume of log to delete is big

Temporary or permanent logging?

Temporary logging	Permanent logging
Start/stop group repeatedly	No repeated stop/restart
Old logs are purged on restart	Obsolete logs must be manually deleted (emaj_delete_before_mark())
Heavy locks on start/stop	No locks, but storage overhead

To Ensure the Reliability

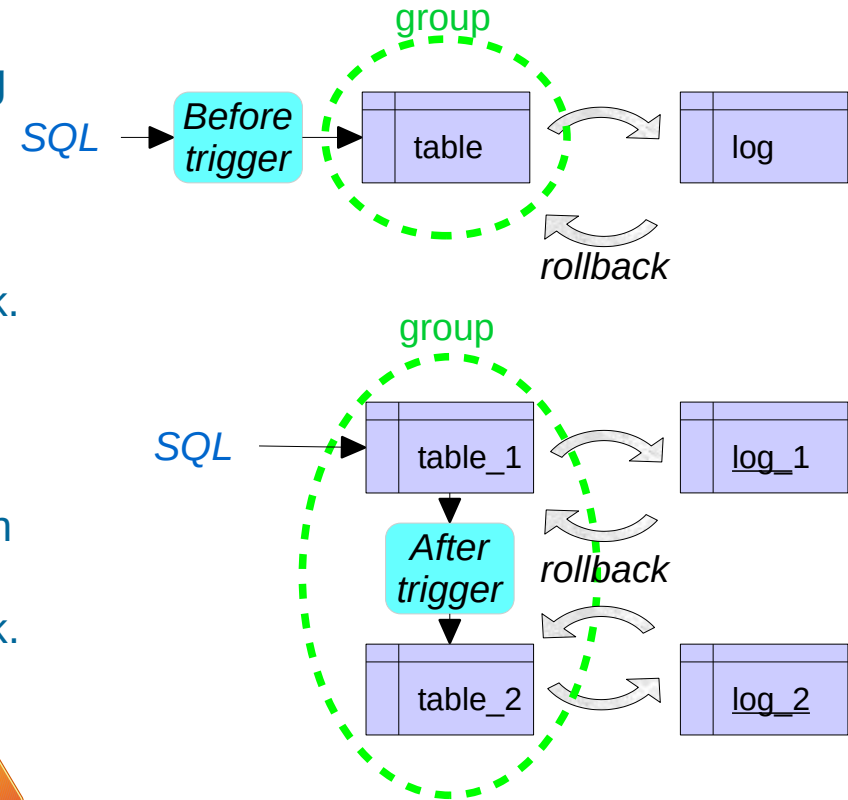
- **No change** in the **PostgreSQL** engine.
- Many systematic **checks**, in particular at group start, mark set or rollback times:
 - Do all required tables, sequences, functions and triggers exist?
 - Consistency of columns between the application tables and the related log tables (existence, type)?
- Heavy **locks** on tables at **start_group**, **set_mark_group** and **rollback_group**, to be sure that no transaction is currently updating application tables.
 - The order of lock setting can be influence by a priority level defined for each table.
- Rollback all tables and sequences in a single **transaction**.

To Ensure the Reliability (2)

- Use internal **sequences** instead of clock timestamp to be insensitive to server time shift.
- “**event triggers**” block unintentional drops or some component changes (tables, sequences, functions...).
- 2 functions to deactivate/reactivate the lock-in:
 - `emaj_disable_protection_by_event_triggers ()`
 - `emaj_enable_protection_by_event_triggers ()`

Impact of Application Triggers on E-Maj Rollbacks

- BEFORE triggers on a table belonging to a table group:
 - Values really inserted into the database are recorded into the log.
 - => must be disabled at E-Maj rollback.
- AFTER triggers updating a table belonging to the same table group:
 - The rollback will reset both tables with the right content.
 - => must be disabled at E-Maj rollback.
- Other cases : study the impacts.



Impact of Application Triggers on E-Maj Rollbacks

- By **default**, application triggers are automatically **disabled** by E-Maj rollbacks.
- A trigger may be left in its state at rollback time if it is registered as is.
- 2 properties for `emaj_assign_table()`, `emaj_assign_tables()`, `emaj_modify_table()` and `emaj_modify_tables()` functions to specify the triggers that must be ignored by the E-Maj rollback processing:
 - `"ignored_triggers": ["trg1","trg2",...]` lists trigger names.
 - `"ignored_triggers_profiles": ["regexp1","regexp2",...]` lists regular expressions that select trigger names.

Security features

- 2 E-Maj roles (NOLOGIN):
 - Administration: `emaj_adm`.
 - Read-only: `emaj_viewer` (to look at logs, marks, statistics).
- **No additional rights** needed on **E-Maj objects**.
- => Just **GRANT** `emaj_adm` / `emaj_viewer` to your **authorized roles**.
- **Log triggers** are created “SECURITY DEFINER”.
- => **No additional rights** needed on application **tables** or **sequences**.

Performance Considerations

- Log overhead:
 - Highly depends on hardware and on the application read/write SQL ratio.
 - Typically **a few %** on **elapse times**.
 - Higher on pure data loading operations.
- Rollback duration:
 - Mainly depends on the number of changes to cancel.
 - Also depends on:
 - The hardware configuration.
 - Tables structure (row sizes, indexes, foreign keys, other constraints...).
 - Almost always **shorter than a logical restore**.

Emaj_web

- User-friendly UI.
- For administrators and users.
- View all E-Maj objects (groups, marks...) and their attributes.
- Perform (almost) all actions on E-Maj objects.

Connection: localhost:5415 - role "postgres" SQL | History | Logout English

Emaj_web > Pg 15 > postgres

Groups Schemas Triggers E-Maj Rollbacks E-Maj

Tables groups in "LOGGING" state

	Group	Created at	Tables	Sequences	Type	Marks	Actions	Comment
☐	myGroup1	12 Apr 2024 15:51:04	5	1		3		Useless comm...
☐	myGroup2	12 Apr 2024 15:51:04	4	2		4		

Select Actions on objects (0)

All / Visible / None / Invert

Tables groups in "IDLE" state

	Group	Created at	Tables	Sequences	Type	Marks	Actions	Comment
☐	phil's group#3	12 Apr 2024 15:51:04	2	1		0		

Select Actions on objects (0)

All / Visible / None / Invert

New group Export Import

Old dropped tables groups

No old dropped tables groups.

Table groups list

Emaj_web : Table Group Details

Connection: localhost:5415 - role "postgres" SQL | History | Logout English

Emaj_web > Pg 15 > postgres > myGroup1

Properties

Changes statistics

Content

History

Tables group "myGroup1" properties

Created at	Type	Tables	Sequences	State	Started at	Marks	Log size
12 Apr 2024 15:51:04		5	1		12 Apr 2024 15:51:05	3	144 kB

Comment: Useless comment!

Set a mark

Protect

Stop

Set a comment

Tables group "myGroup1" marks

		Mark	State	Set at	Row changes	Cumulated changes	Actions	Comment
☐		MARK3		Fri 12 Apr 15:51:05	0	0		
☐		MARK2		Fri 12 Apr 15:51:05	7	7		End of 1st prog...
☐		MARK1		Fri 12 Apr 15:51:05	19	26		

Select

Actions on objects (0)

All / Visible / None / Invert

Current Limitations

- **Minimum PostgreSQL version = 14.**
- Every application table in a **rollbackable group** needs a **PRIMARY KEY**.
- **DDL** statements cannot be logged or rolled back by E-Maj.
 - A table structure change requires temporarily removing the table from its group.
- **Foreign keys on partitioned tables** are incompatible with E-Maj rollbacks.
 - Workaround: define foreign keys on each partition.

Conclusion

- Many more informations in:
 - the **documentation**:
<https://emaj.readthedocs.io/en/latest/index.html>
 - the README et CHANGES files.
- **Thanks** to all contributors and users!
- **Feedback**: Contribute via GitHub or email (phb.emaj@free.fr).