



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"

E-Maj, what is it for?

- E-Maj allows the data content to **travel back in time**, with a table level granularity
- By recording updates 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

The gains

- In **test** environment
 - Helps the application tests management by providing a quick way to
 - Examine updates generated by the application, for debugging purpose
 - Cancel updates generated by the application in order to easily repeat tests
- In **production** environment
 - Allows to cancel processings
 - Without being obliged to save and restore the instance by `pg_dump/pg_restore` or by physical copy
 - With a finer granularity
 - Avoids to loose entire batch processing nights by helping the recovery after failure
 - Very interesting with large tables and few updates

The components

- **E-Maj**, the heart
 - A PostgreSQL extension
 - Open Source, under 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**
 - A web client - https://github.com/dalibo/emaj_web
- The online **documentation**
 - In English (or French) - <https://emaj.readthedocs.io/en/latest/>



The characteristics which drove the design

- **Reliability**
 - Absolute data integrity after updates cancellation
 - Management of all usual objects (tables, sequences, constraints,...)
- **Ease of use** for DBAs, production people, application developpers and testers,...
 - Easy to understand and use
 - Easy to integrate into an automatized production (thus scriptable)
- **Performance**
 - Limited log overhead
 - Acceptable “rollback” duration
- **Security**
- **Maintainability**

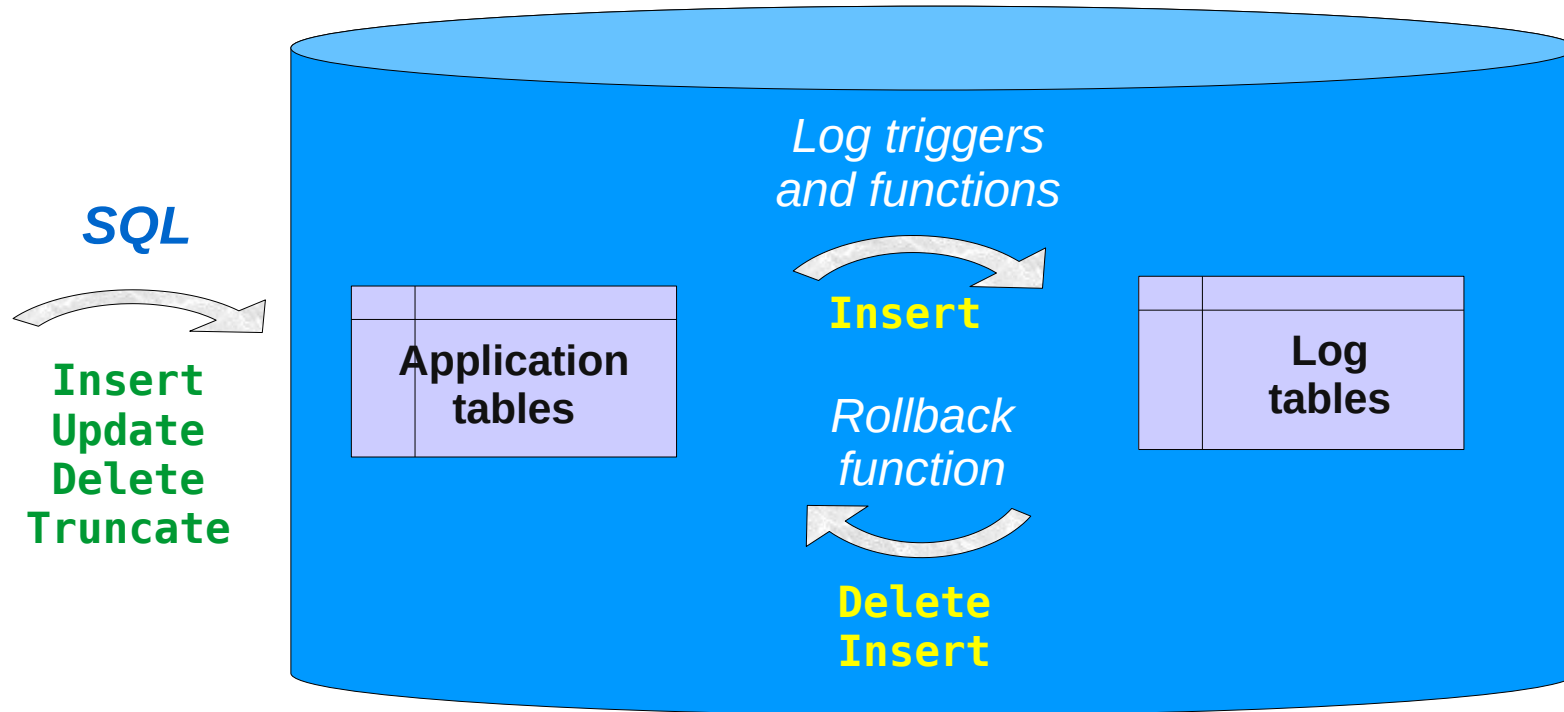
Concepts

- **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, whose state can be set back ; identified by a name
- **E-Maj Rollback** = positioning of a tables group at a previously set mark state
 - NB: this concept is different from the transaction rollbacks performed by the RDBMS
 - a “RDBMS-rollback” cancels the current transaction
 - a “E-Maj rollback” cancels updates from several committed transactions

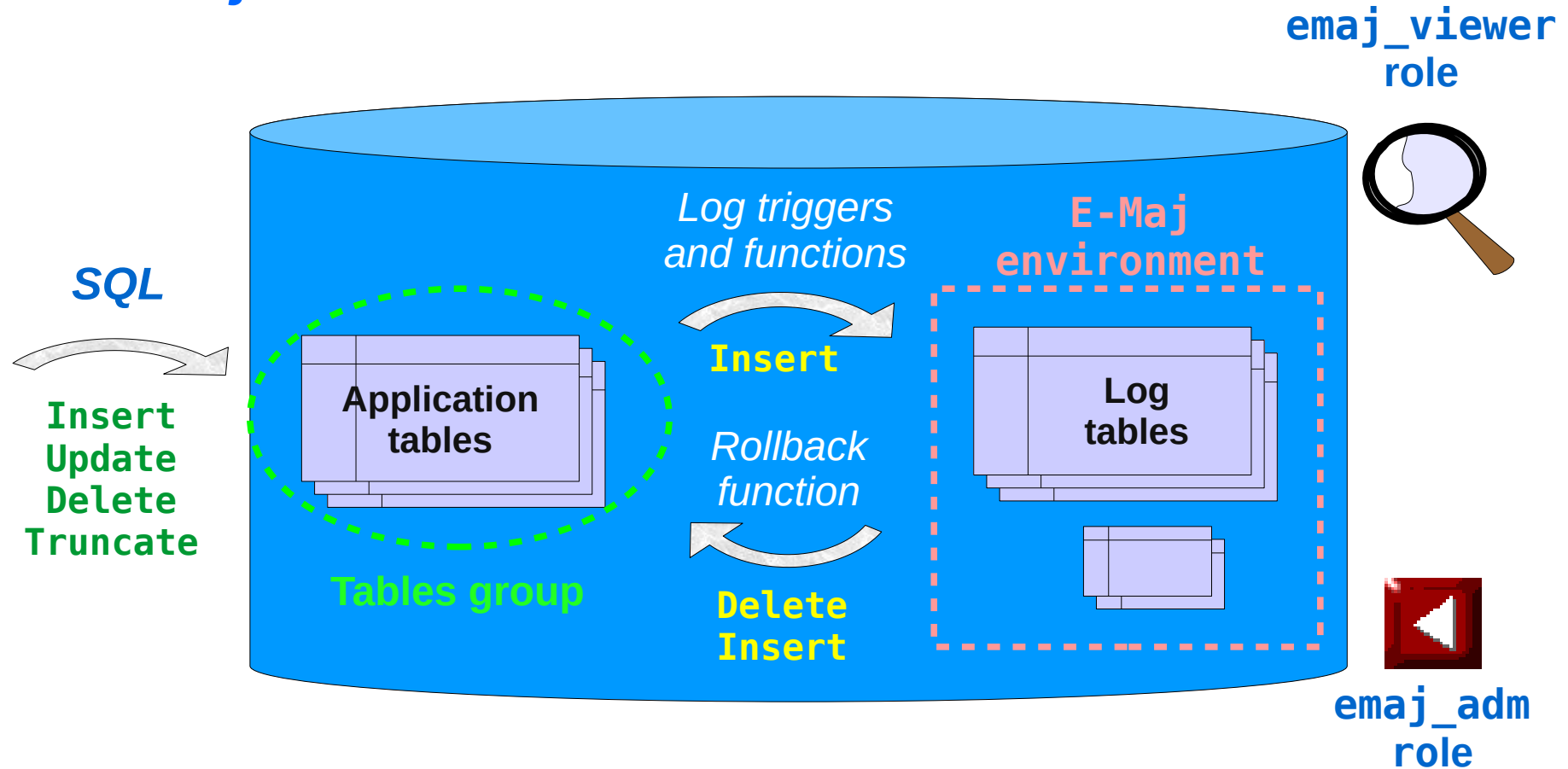
Concepts (2)

- By default, a tables group is created as **rollbackable**
- A tables group 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 when a tables group capture data changes ; it is bounded by the tables group **start** and **stop** actions.

An updates recording based on triggers



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

- Standart install
 - `pgxn install E-Maj --sudo`
 - Log on the target database as super-user and execute
 - `CREATE EXTENSION emaj CASCADE;`
- Install on DBaaS cloud environment
 - Download from pgxn.org and unzip the extension
 - `psql ... -f sql/emaj-<version>.sql`
- This adds to the database
 - the extensions `dblink` et `btree_gist` if needed
 - 1 schema, named 'emaj', with about 190 functions, 19 technical tables, 12 types, 1 view, 1 sequence, 3 event triggers
 - 2 roles

Initialization

- For each group:
 - 1) Create an empty group
`SELECT emaj_create_group (group, is_rollbackable [,comment]);`
 - 2) Add 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$'`
 - Create for each application table: 1 log table, 1 log sequence, 1 log trigger and its function
- NB: `SELECT emaj_drop_group (group)`
 - ... drop an existing group

The 3 main functions to manage groups

- “Starting” a group
 - `emaj_start_group (group, mark)`
activates the log triggers and sets a first mark
- Setting a mark
 - `emaj_set_mark_group (group, mark [,comment])`
sets an intermediate mark
- “Stopping” a group
 - `emaj_stop_group (group [,mark])`
deactivates the log triggers => a rollback is not possible anymore
- The % character in a mark name represents the current date and time

Examine logs

- Examining log tables 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
- A single row change in an application table generates
 - 1 log row for an INSERT (image of the new row)
 - 1 log row for a DELETE or a TRUNCATE (image of the old row)
 - 2 log rows for an UPDATE (image of the old and the new rows)
- A TRUNCATE generates also a single log row

Log tables technical columns

- 6 technical columns at the end of each log row
 - `emaj_verb` : SQL statement type - INS/UPD/DEL/TRU
 - `emaj_tuple` : row type - OLD/NEW
 - `emaj_gid` : internal sequence number
 - `emaj_changed` : time of the update - `clock_timestamp()`
 - `emaj_txid` : transaction identifier - `txid_current()`
 - `emaj_user` : connection role of the client - `session_user`
- ... and some others can be added
- It is possible to identify clients and transactions, and analyze the timing of the program execution

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 recorded data changes

- 3 statistical functions, at **tables 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 recorded data changes (2)

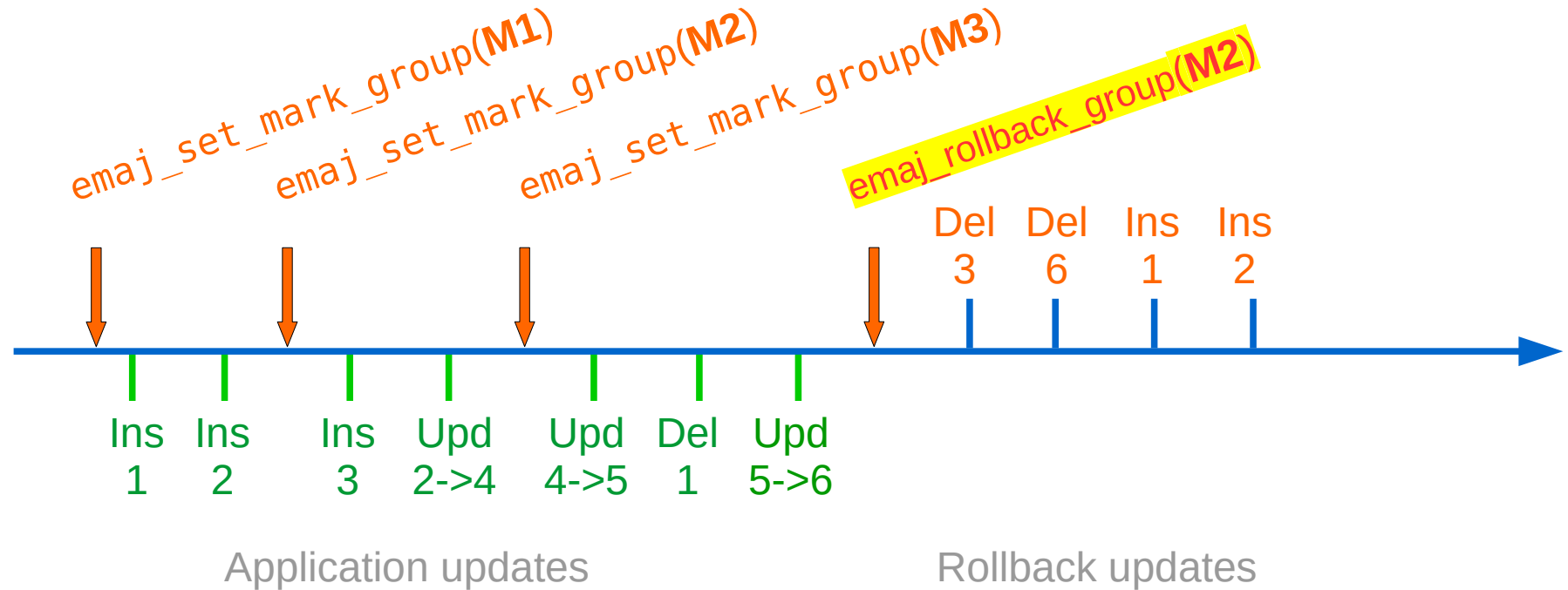
- 2 functions quickly return an estimate of recorded changes for **one table** for each elementary marks interval on a given time frame
 - `emaj_log_stat_table (schema, table, start_date_time, end_date_time)`
 - `emaj_log_stat_table (schema, table, start_group, start_mark, end_group, end_mark)`
- 2 functions 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)`

Cancel updates : the “simple” rollback

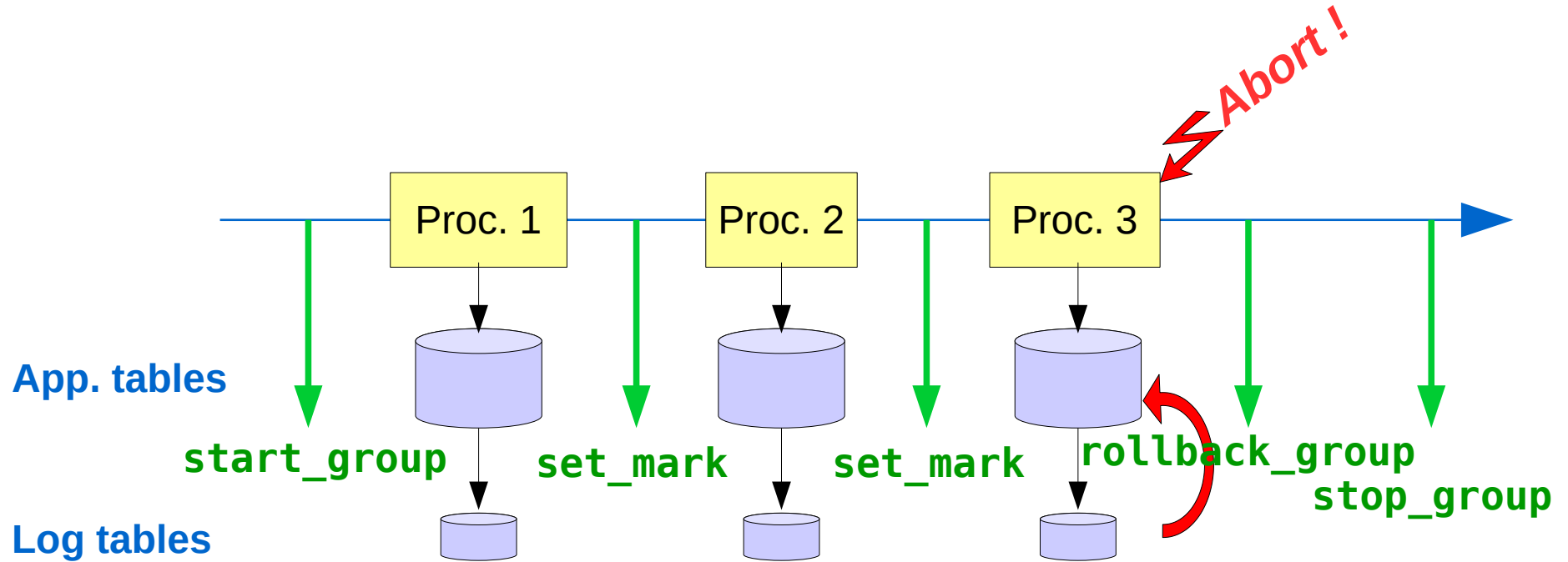
- A “rollback” function allows to reset a tables group in the state it had at a given mark
 - `emaj_rollback_group (group, mark [, false [, comment]])`
- How this works
 - Log triggers are deactivated during the operation
 - Each table is reset to its mark state using an optimised algorithm
 - Application sequences are reset to their mark state
 - Takes into account the foreign keys, if any
 - The canceled logs and marks are deleted
 - => all what is after the rollback mark is forgotten

An optimised rollback algorithm

- It processes each primary key value only once



A typical E-Maj usage (production batch processing)

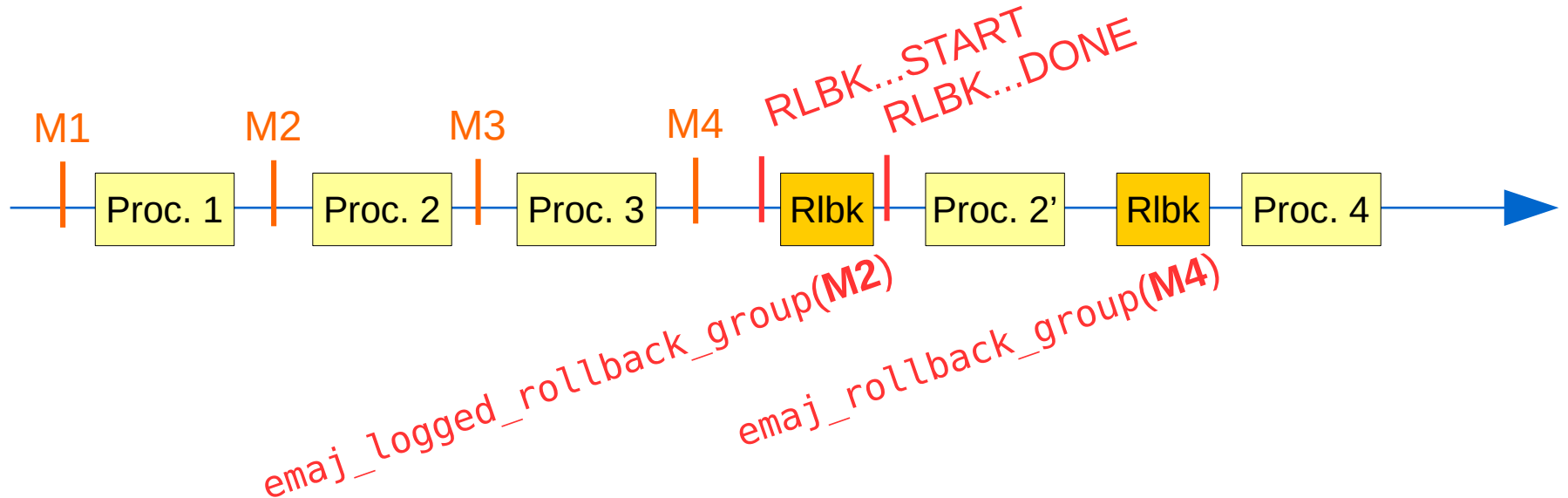


Cancelling updates : the “logged” rollback

- `emaj_logged_rollback_group (group, mark[, false [, comment]])`
- Different from the “simple” rollback
 - Log triggers are NOT deactivated during the operation
=> the updates generated by the rollback are recorded
 - Cancelled logs et marks are NOT deleted
- So we can revert an E-Maj rollback ! And more generally let a tables group travel back and forth in time !
- 2 marks are automatically set before and after the rollback
 - `RLBK_<rollback id>_START` and `RLBK_<rollback id>_DONE`
- During the rollback, tables remain accessible in read mode

A typical E-Maj usage in test environment

- 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 an E-Maj rollback duration

- In order to know if we have enough time to perform the operation or if another way to recover would be more efficient
- 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 performs rollbacks with parallelism
 - `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>]`
- Automatically spreads the tables to process into a given number of parallel sessions
- All sessions belong to a single transaction (2PC)
 - => `max_prepared_transactions` >= nb sessions
- Needs perl software with its PostgreSQL extension

Monitoring E-Maj rollbacks in execution

- A function
 - `SELECT * FROM emaj.emaj_rollback_activity ();`
 - returns
 - The characteristics of rollbacks (group, mark...)
 - Their state
 - Their current duration
 - An estimate of the remaining duration and the already executed %
- Needs to setup the value of the “`dblink_user_password`” parameter in the `emaj_param` table

Monitoring E-Maj rollbacks

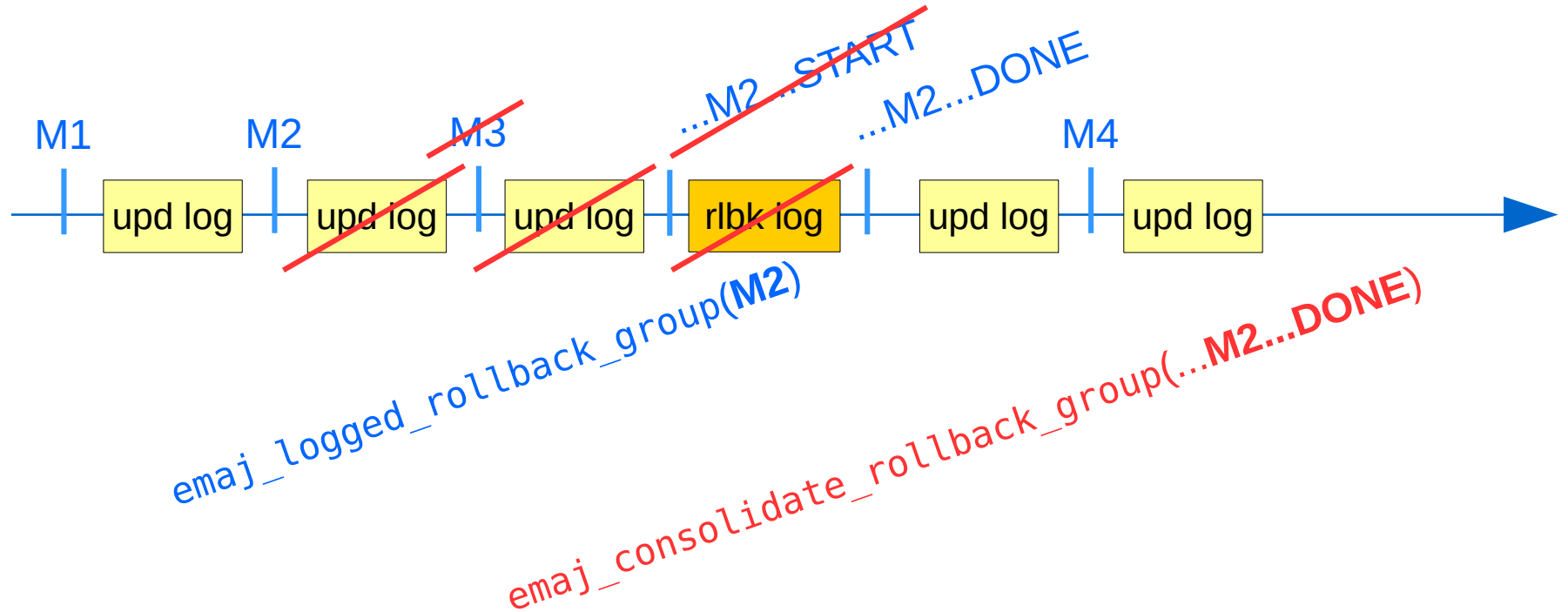
- A perl client to monitor the executing or completed rollbacks
 - `emajRollbackMonitor.pl -d <database> -h <host> -p <port> -U <user> -W <password> -n <nb_iterations> -i <refresh_rate_in_seconds> -l <nb_completed rollbacks> -a <completed_rollbacks_history_depth_in_hours>`

```
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
```

Consolidate a “logged” rollback

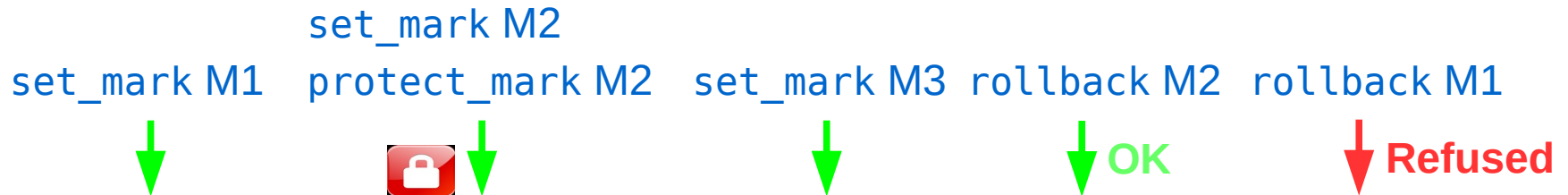
- “Consolidate” a rollback means transform a “logged rollback” into a “simple rollback”
- Intermediate logs and marks are deleted, recovering some place in the logs
 - `emaj_consolidate_rollback_group (groups, end_rollback_mark)`
- Tables can be updated during the consolidation
- A function returns the list of consolidable rollbacks
 - `emaj_get_consolidable_rollbacks ()`

Example of E-Maj rollback consolidation



Being protected against unattended E-Maj rollbacks

- 2 functions to manage the protection of a tables group
 - `emaj_protect_group (group)`
 - `emaj_unprotect_group (group)`
- 2 functions to manage the protection of a mark
 - `emaj_protect_mark_group (group, mark)` blocks any attempt to rollback to a mark prior the protected mark
 - `emaj_unprotect_mark_group (group, mark)`



The emajStat client to monitor changes recording

- A perl client to count changes on tables and sequences since the latest mark of their group and since the previous display, in absolute value and changes per second
- 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)
```

The flowchart illustrates the data processing pipeline for the "Logging" state. It starts with three input sources on the left: "Tables group in 'Logging' state" (represented by a green cylinder labeled 'g'), "Tables in groups" (represented by a stack of blue rectangles labeled 't'), and "Sequences in groups" (represented by a stack of orange rectangles labeled 's').

The pipeline consists of several processing blocks:

- The "Tables group in 'Logging' state" input feeds into a "Filter on names" block, which is associated with the flags `--include-groups` and `--exclude-groups`.
- The "Tables in groups" input feeds into a "Filter on names" block, which is associated with the flags `--include-tables` and `--exclude-tables`.
- The "Sequences in groups" input feeds into a "Filter on names" block, which is associated with the flags `--include-sequences` and `--exclude-sequences`.
- The output of the first "Filter on names" block feeds into the "Look at log sequences" block.
- The output of the second "Filter on names" block feeds into the "Look at log sequences" block.
- The output of the third "Filter on names" block feeds into the "Look at sequences" block.
- The "Look at log sequences" block feeds into the "Aggregation" block.
- The "Look at sequences" block feeds into the "Sort and limit" block, which is associated with the flag `--max-sequences`.
- The "Aggregation" block feeds into the "Sort and limit" block, which is associated with the flag `--max-groups`.
- The output of the "Sort and limit" block (from the "Look at log sequences" block) feeds into the "Display" block.
- The output of the "Sort and limit" block (from the "Aggregation" block) feeds into the "Display" block.
- The output of the "Sort and limit" block (from the "Look at sequences" block) feeds into the "Display" block.

The final output of the pipeline is the "Display" block, which shows the results of the processing.

Analyse recorded data changes

- Dump on files, by COPY, in a given directory, a log tables extracts and sequences of a group
 - `emaj_dump_changes_group (group, start_mark, end_mark, options_list, tables/seq_array, directory)`
- Generate SQL to extract recorded changes between 2 marks for all or some tables or sequences of a group
 - In 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)`

Analyse 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()`
 - **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()`
 - **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

Analyse data changes: the consolidated vision of changes

- The consolidated vision of changes provides a 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’, row OLD = ‘A’ and row NEW = ‘C’
- Therefore each examined table must have an explicit PK
- 2 consolidation kinds
 - “Partial consolidation”: without taking into account the columns content
 - “Full consolidation”: examining the changed data
 - For a given PK, no change is reported if all columns of both “OLD” and “NEW” rows are equal
 - Ex: no change reported for a given PK 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 is returned if the sequence has not been changed

Analyse data changes: emaj_temp_sql temporary table structure

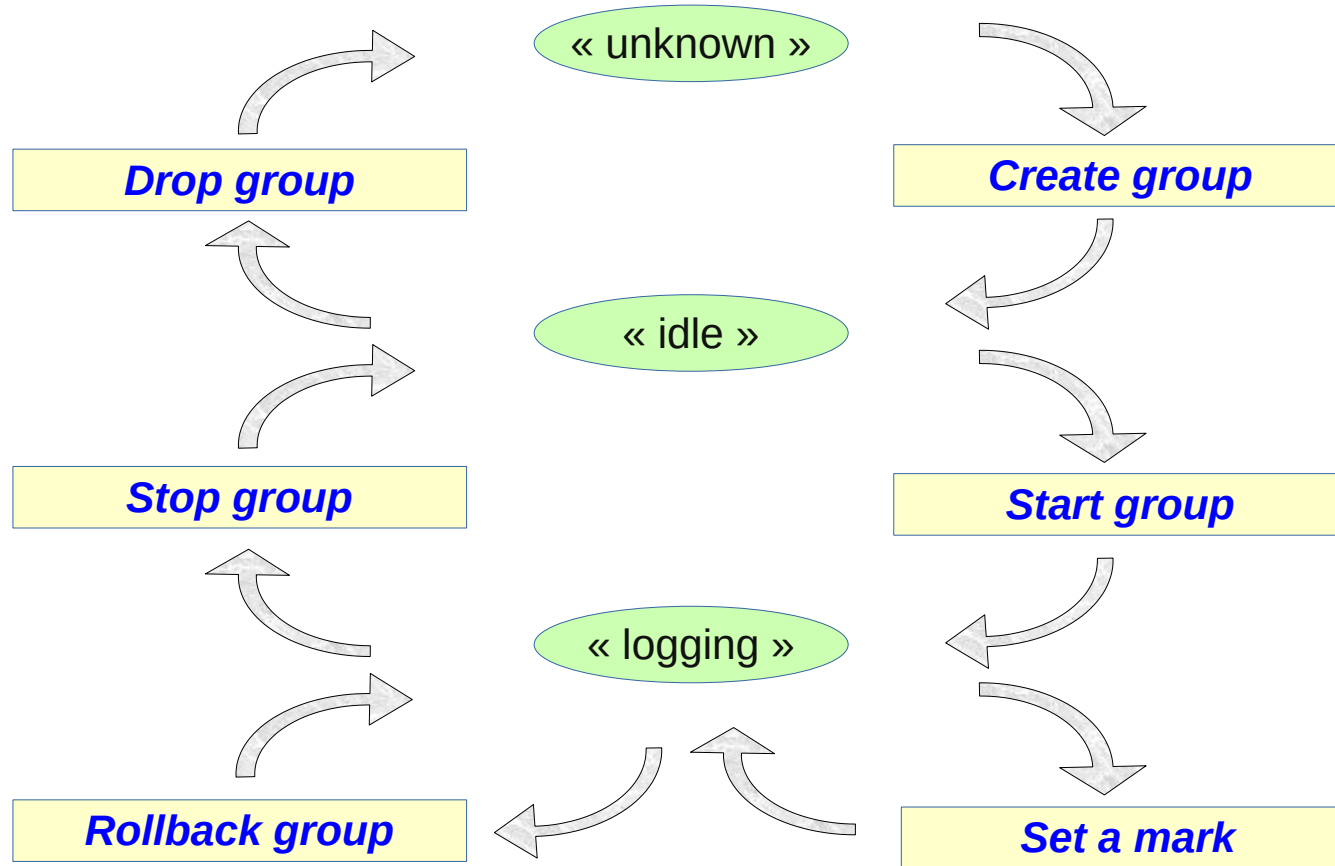
```
CREATE TEMP TABLE emaj_temp_sql (  
  sql_stmt_number      INT,          -- Statement number  
                                -- (0 for the initial comment)  
  sql_line_number      INT,          -- Line number within the statement  
                                -- (0 for the initial comment 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,         -- Tables 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

Replay data changes

- Generate a sql script replaying the elementary recorded 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, with psql:
`SELECT emaj_gen_sql_group (group, start_mark,
end_mark, NULL [,tables/seq_list])
\copy (SELECT * FROM emaj_sql_script) TO 'dest_file'`
- Useful in test environment to “replicate” the changes produced by a processing

The tables group life cycle



Tables groups dynamic adjustment

- To 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:
 - JSON format
 - To define the priority and the tablespaces for log data and index
- Selection and exclusion filters: RegExp

Tables groups dynamic adjustment

- Example
 - `emaj_assign_tables('myschema', 'tbl.*', '_sav$', 'mygroup',
'{"priority":1}':::json)`
assigns to the group 'mygroup' and with the priority 1 all tables of the schema 'myschema' whose name starts with 'tbl' and doesn't end with '_sav'

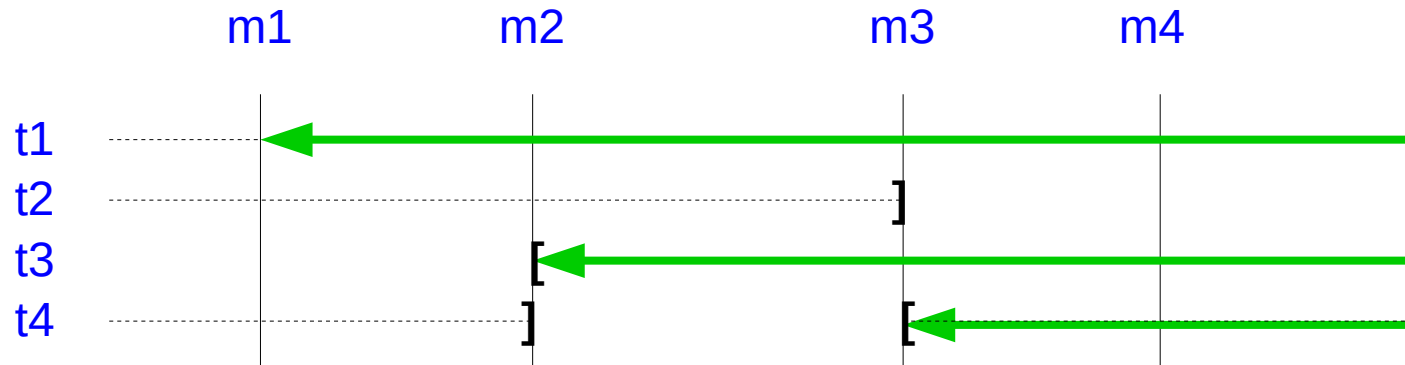
Tables groups dynamic adjustment

- 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()`

Impact of logging group structure changes on rollbacks

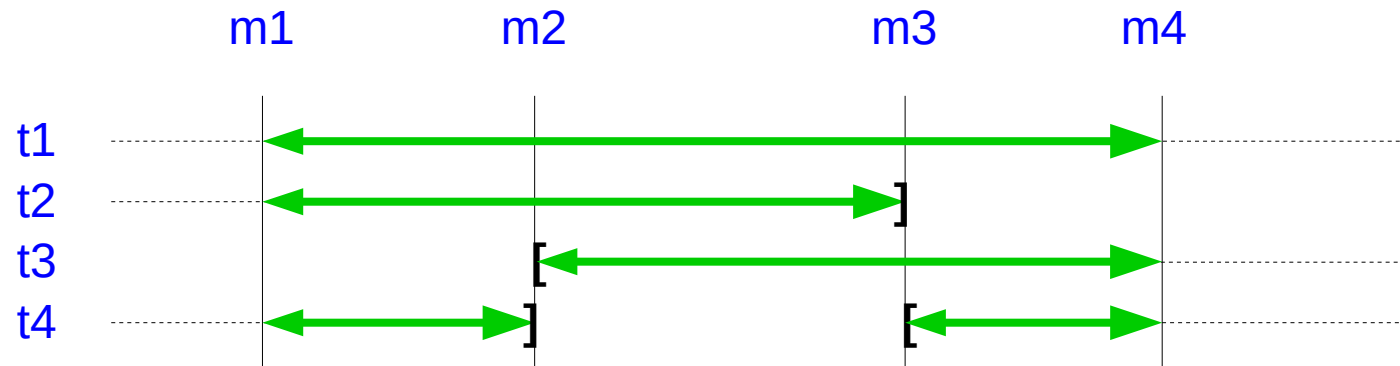
Table t2 removed at mark m3, t3 added at m2, t4 removed at m2 and added at m3

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



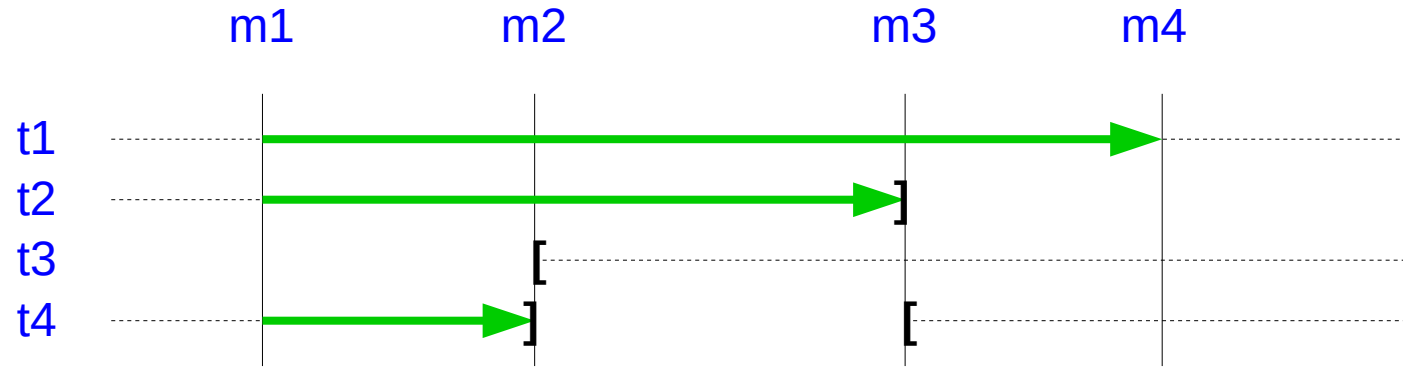
Impact of logging group structure changes on statistics and content changes extracts

`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:



Modify the structure of a table in a LOGGING group

- For actions like: rename the table, change its schema, add/drop/rename a column, change a column type
- The log table structure is impacted
- 3 steps
 - Remove the table from its tables group
 - ALTER TABLE
 - Add the table into its tables group
- Constraint: an E-Maj rollback to a prior mark will not be able to go beyond the structure change
- Idem to rename a sequence or change its schema

Processing several groups in a single operation

- Some “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, ...)`
- Allows to get marks shared by several groups
- Both PostgreSQL syntaxes for groups arrays
 - `ARRAY['group 1', 'group 2', ...]`
 - `'{"group 1", "group 2", ... }'`

Managing marks

- Comment a mark for a group (add/modify/suppress)
 - `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 one 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 date and time
 - `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)`
- Export / import tables groups configurations
 - `emaj_export_groups_configuration ()`
 - `emaj_import_groups_configuration ()`
- Force a group stop (in case of problem with the normal stop function)
 - `emaj_force_stop_group (group)`

Other actions on groups

- Snap on files in a given directory, by COPY, all tables and sequences of a group
 - `emaj_snap_group (group, directory, copy_options)`
- Erase histories about a dropped tables group
 - `emaj_forget_group (group)`

Other actions

- Get the current emaj extension version or drop the extension
 - `emaj_get_version ()`
 - `emaj_drop_extension ()`
- Verify the good health of the E-Maj installation
 - `emaj_verify_all ()`
- Get the current log table of a given application table
 - `emaj_get_current_log_table ()`
- Manualy purge obsoletes traces
 - `emaj_purge_histories ()`
- Create/modify/delete a comment on a rollback
 - `emaj_comment_rollback ()`
- Export or import parameters configuration
 - `emaj_export_parameters_configuration ()`
 - `emaj_import_parameters_configuration ()`

Temporary or permanent logging?

- **Temporary logging** = steps like
 - `emaj_start_group()`
 - repeat
 - processing
 - `emaj_set_mark()`
 - `emaj_stop_group()`
 - At next start, old logs are purged
 - But stops and starts set very heavy locks
- **Permanent logging** = no repeated group stop/restart
 - Obsolete data in log tables 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

For large databases...

- Log tables and indexes can be stored into **tablespaces**
 - 2 optional properties set when assigning tables to groups

To ensure the reliability

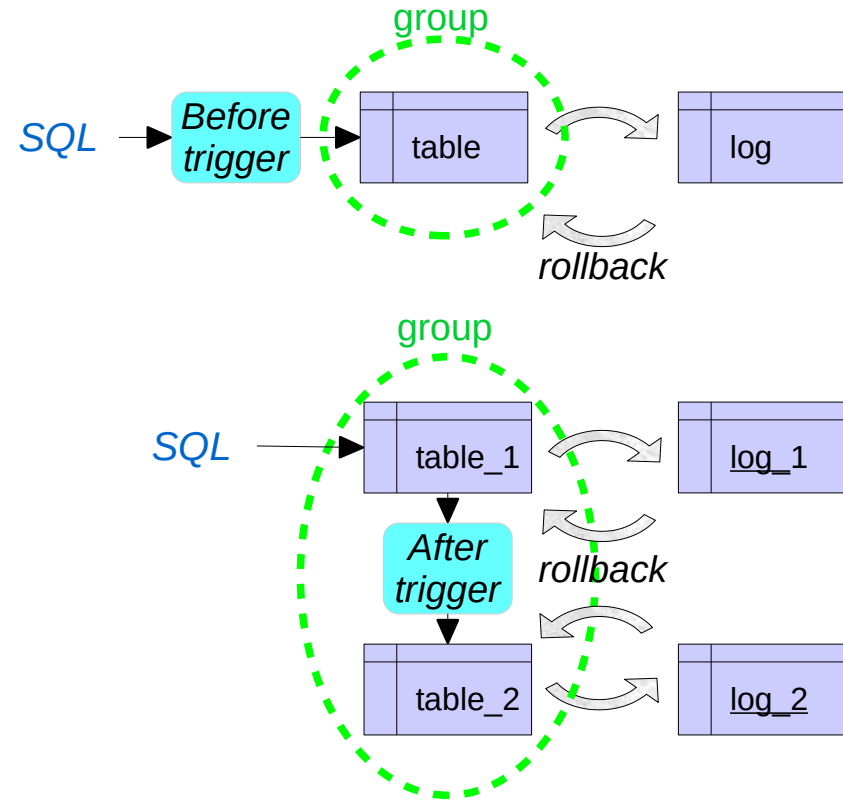
- No change in the PostgreSQL engine
- Many systematic **checks**, in particular at group start, mark set or rollback times:
 - Do all required tables, sequences, functions and triggers exist?
 - Consistency of columns between the application tables and the related log tables (existence, type)?
- Heavy **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 by a single **transaction**

To ensure the reliability (2)

- “**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

- Triggers of type *BEFORE* on a table belonging to a tables group
 - Values really inserted into the database are recorded into the log
 - => to be disabled at E-Maj rollback
- Triggers of type *AFTER* updating a table belonging to the same tables group
 - The rollback will reset both tables with the right content
 - => to 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

To contribute to the security

- 2 NOLOGIN roles whose rights may be granted:
 - `emaj_adm` for the E-Maj administration
 - `emaj_viewer` to just look at E-Maj objects (logs, marks, statistics)
- E-Maj objects are only created and handled by a super-user or a member of the `emaj_adm` role
- No other right has to be granted on E-Maj schemas, tables and functions
- Log triggers are created with the “SECURITY DEFINER” attribute
- No need to give additional rights to application tables or sequences

Performances

- Log overhead
 - Highly depends on hardware and on the application read/write SQL ratio
 - Typically a few % on elapse times
 - But can be much higher on pure data loading
- Rollback duration
 - Of course depends on the number of updates to cancel
 - Also highly depends on
 - The hardware configuration
 - Tables structure (row sizes, indexes, foreign keys, other constraints...)
 - But almost always shorter than a logical restore

Emaj_web

- For administrators and users
- All E-Maj objects (groups, marks...) and their attributes
- (almost) all possible 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
<input type="checkbox"/>	myGroup1	12 Apr 2024 15:51:04	5	1		3		Useless comm...
<input type="checkbox"/>	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
<input type="checkbox"/>	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.

Tables groups list

Emaj_web : tables 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
<input type="checkbox"/>		MARK3		Fri 12 Apr 15:51:05	0	0		
<input type="checkbox"/>		MARK2		Fri 12 Apr 15:51:05	7	7		End of 1st prog...
<input type="checkbox"/>		MARK1		Fri 12 Apr 15:51:05	19	26		

Select

Actions on objects (0)

All / Visible / None / Invert

Current limitations

- Minimum PostgreSQL version = **12**
- Every application table belonging to a rollbackable group needs a **PRIMARY KEY**
- **DDL** statements cannot be logged or cancelled by E-Maj
 - Changing a table's structure requires to temporarily remove the table from its group
- **FOREIGN KEYs** defined on **partitionned tables** are incompatible with E-Maj rollbacks
 - => define them on each partition

To conclude...

- Many more **informations** in
 - the documentation:
<https://emaj.readthedocs.io/en/latest/index.html>
 - the README et CHANGES files
- Many thanks to all contributors and users
- Feel free to give any **feedback** through github or email (phb.emaj@free.fr)