



Et vos données PostgreSQL
voyagent dans le temps !

L'acronyme de
« *Enregistrement des Mises A Jour* »

E-Maj, ça sert à quoi ?

- E-Maj permet de **déplacer dans le temps** des contenus de données, avec une granularité de niveau table
- En **enregistrant les mises à jours** sur des ensembles de tables applicatives, on peut
 - les **dénombrer** (fonction statistique),
 - les **consulter** facilement (fonction d'audit),
 - les **annuler** (fonction de « rollback »),
 - les **rejouer** (génération de script, ou annulation d'une annulation...)
- Utilisable avec
 - des applications en test ou en production
 - des bases de données de toute taille

La plus-value

- En environnement de **test**
 - Aide l'organisation des tests applicatifs en fournissant un moyen rapide
 - d'examiner les mises à jour générées par l'application
 - d'annuler les mises à jour issues d'une exécution de programmes et de pouvoir ainsi répéter facilement des tests
- En environnement de **production**
 - Permet d'annuler des traitements
 - sans devoir sauver et restaurer l'instance par `pg_dump/pg_restore` ou copie physique
 - avec une granularité plus fine
 - Évite de perdre des nuits complètes de traitement batch, en facilitant les reprises sur incident
 - D'autant plus intéressant que les tables sont volumineuses et les mises à jour peu nombreuses

Les composants

- **E-Maj**, le cœur
 - Une extension PostgreSQL
 - Open Source, sous licence GPL
 - Téléchargeable depuis [pgxn.org](https://pgxn.org/dist/e-maj/) - <https://pgxn.org/dist/e-maj/>
 - Sources disponibles sur [github.com](https://github.com/dalibo/emaj) - <https://github.com/dalibo/emaj>
- **1 clients web**
 - Emaj_web - https://github.com/dalibo/emaj_web
- Une **documentation** en ligne
 - En français (ou anglais) - <https://emaj.readthedocs.io/fr/latest/>



Les caractéristiques qui ont guidé le design

- **Fiabilité**
 - Intégrité absolue des données après annulation de mises à jour
 - Gestion de tous les objets usuels (tables, séquences, contraintes,...)
- **Facilité** d'utilisation pour les DBAs, exploitants, développeurs et testeurs d'applications, ...
 - Facilement compréhensible et utilisable
 - Facile à automatiser (donc scriptable)
- **Performance**
 - Surcoût du log limité
 - Durée de « rollback » acceptable
- **Sécurité**
- **Maintenabilité**

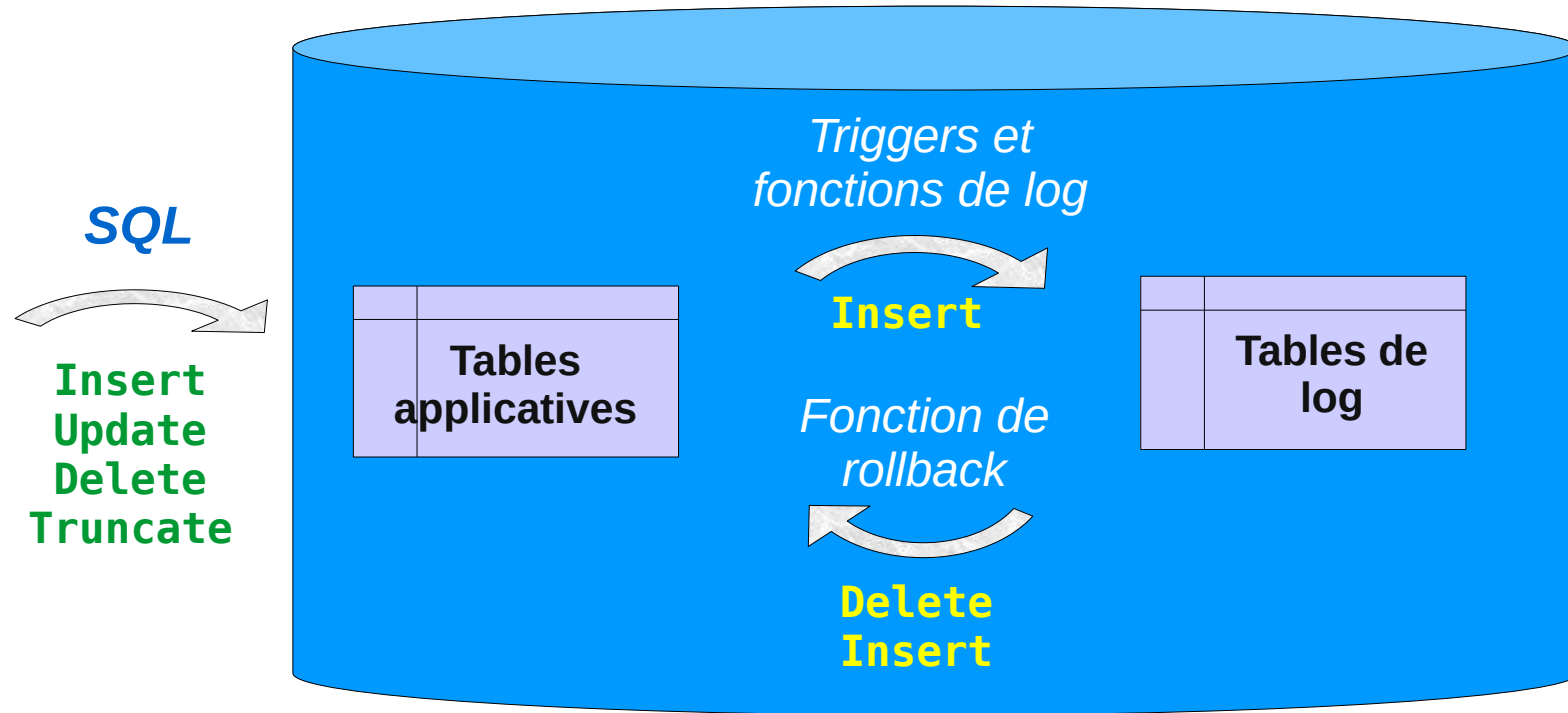
Concepts

- **Groupe de tables** = ensemble de tables et/ou séquences d'une base de données, appartenant à un ou plusieurs schémas, et ayant le même rythme de vie ; c'est le principal objet manipulé par l'utilisateur
- **Marque** = point stable dans la vie d'un « groupe de tables », et dont on peut retrouver l'état ; elle est identifiée par un nom
- **Rollback E-Maj** = positionnement d'un « groupe de tables » à l'état dans lequel il se trouvait lors de la prise d'une « marque »
 - NB : ce concept est différent du rollback des transactions effectué par le SGBD
 - le « rollback du SGBD » annule la transaction courante
 - le « rollback E-Maj » annule les mises à jour de multiples transactions « commitées »

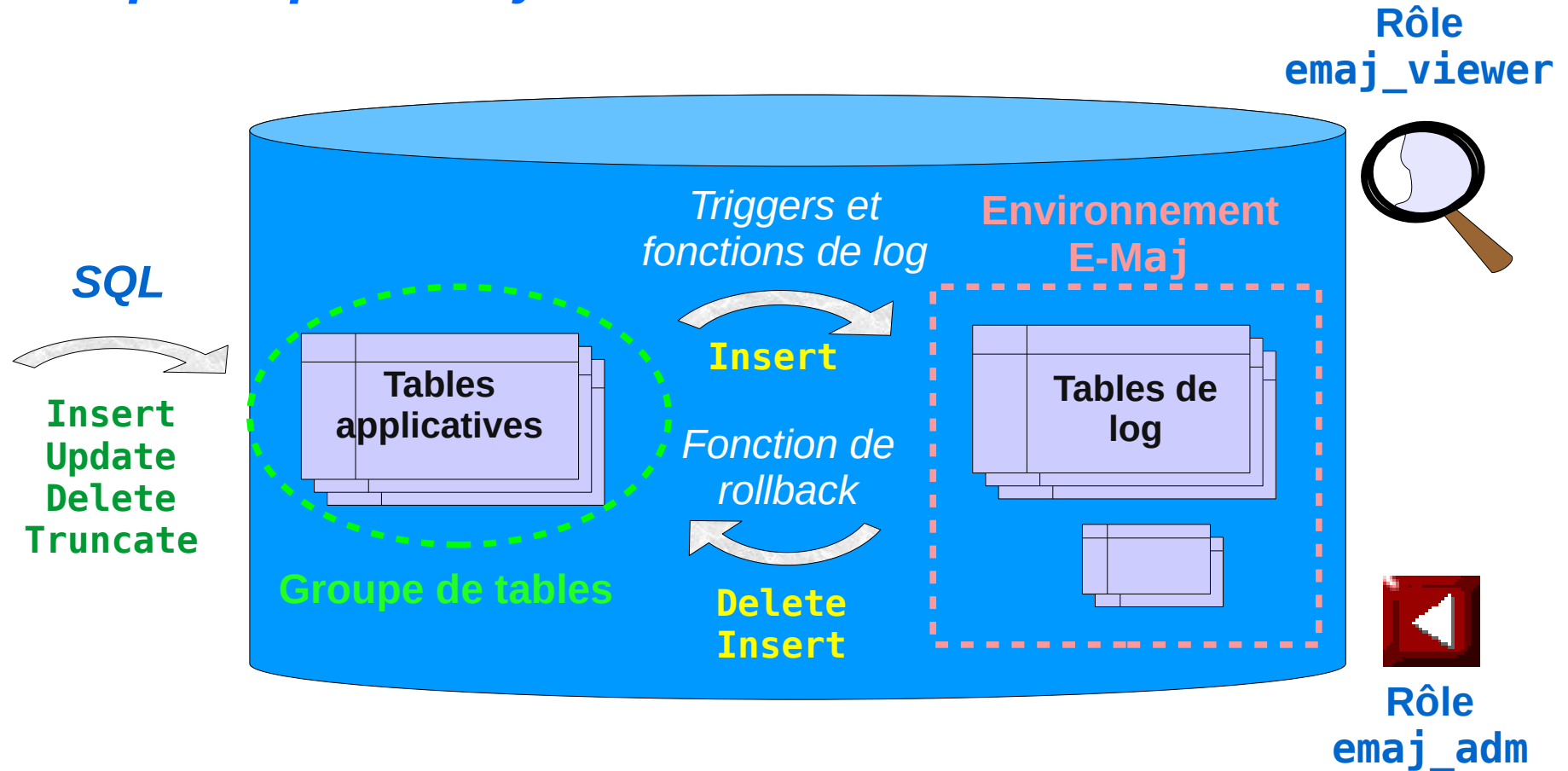
Concepts (2)

- Par défaut, un groupe de tables est créé « **rollbackable** »
- Un groupe de tables peut être créé « **audit-only** »
 - Pas de rollback E-Maj possible
 - Mais
 - Il n'est pas indispensable que chaque table ait une PRIMARY KEY
 - Les tables peuvent être de type UNLOGGED ou WITH OIDS

Un log des mises à jour basé sur des triggers



Les principaux objets



Gestion des séquences applicatives

- Les incréments des séquences ne sont pas enregistrées individuellement
- Pose d'une marque sur un groupe de tables
 - L'état de chaque séquence du groupe est enregistré dans une table interne
- Rollback E-Maj
 - Chaque séquence est remise dans l'état enregistré à la pose de la marque ciblée

Installation

- Télécharger et décompresser l'extension
- Installation standard
 - Copier les fichiers emaj.control et sql/*.sql dans \$SHAREDIR/extension
 - Se connecter à la base ciblée en tant que super-utilisateur puis
 - `CREATE EXTENSION emaj CASCADE;`
- Installation sur environnements type cloud DBaaS
 - `psql ... -f sql/emaj-<version>.sql`
- L'installation ajoute à la database
 - les extensions `dblink` et `btree_gist` si besoin
 - 1 schema 'emaj' avec environ 120 fonctions, 15 tables techniques, 8 types, 1 vue, 1 séquence, 3 event triggers
 - 2 rôles

Initialisation

- Pour chaque groupe :
 - 1) Création du groupe vide
`SELECT emaj_create_group (groupe, est_rollbackable);`
 - 2) Ajout de tables et séquences
`SELECT emaj_assign_tables (schéma, regexp inclusion, regexp exclusion, groupe);`
`SELECT emaj_assign_sequences (schéma, regexp inclusion, regexp exclusion, groupe);`
 - Ex : toutes les tables d'une schéma sauf celles suffixées par sav :
`'.*', 'sav$'`
 - crée pour chaque table applicative : 1 table de log + 1 séquence de log + 1 trigger de log et sa fonction
- NB : `SELECT emaj_drop_group (groupe)`
 - ... supprime un groupe existant

Les 3 fonctions principales de gestion des groupes

- « Démarrage » d'un groupe
 - `emaj_start_group (groupe, marque)`
active les triggers de log et pose une marque initiale
- Pose une marque intermédiaire
 - `emaj_set_mark_group (groupe, marque)`
pose une marque intermédiaire
- « Arrêt » d'un groupe
 - `emaj_stop_group (groupe [,marque])`
désactive les triggers de log => le rollback n'est plus possible
- Le caractère % dans le nom d'une marque représente la date et l'heure courante

Examiner les logs

- L'examen des tables de log peut grandement aider le debugging des applications
- Chaque table applicative a sa table de log
 - `emaj_<schéma>.<table>_log`
- Une table de log contient
 - les mêmes colonnes que la table applicative associée
 - et quelques colonnes techniques
- Une ligne mise à jour dans une table applicative génère
 - 1 ligne de log pour un INSERT (nouvelle ligne)
 - 1 ligne de log pour un DELETE ou TRUNCATE (ancienne ligne)
 - 2 lignes de log pour un UPDATE (ancienne et nouvelle lignes)
- Un TRUNCATE génère aussi une ligne de log

Les colonnes techniques des tables de log

- 6 colonnes techniques en fin de chaque ligne de log
 - `emaj_verb` : type de mise à jour - INS/UPD/DEL/TRU
 - `emaj_tuple` : type de ligne - OLD/NEW
 - `emaj_gid` : numéro de séquence interne
 - `emaj_changed` : heure de la mise à jour - `clock_timestamp()`
 - `emaj_txid` : numéro de la transaction - `txid_current()`
 - `emaj_user` : rôle de connexion du client - `session_user`
- ... et on peut en ajouter d'autres
- On peut ainsi identifier les clients, le découpage des transactions et analyser le timing d'exécution du traitement

Compter les mises à jour

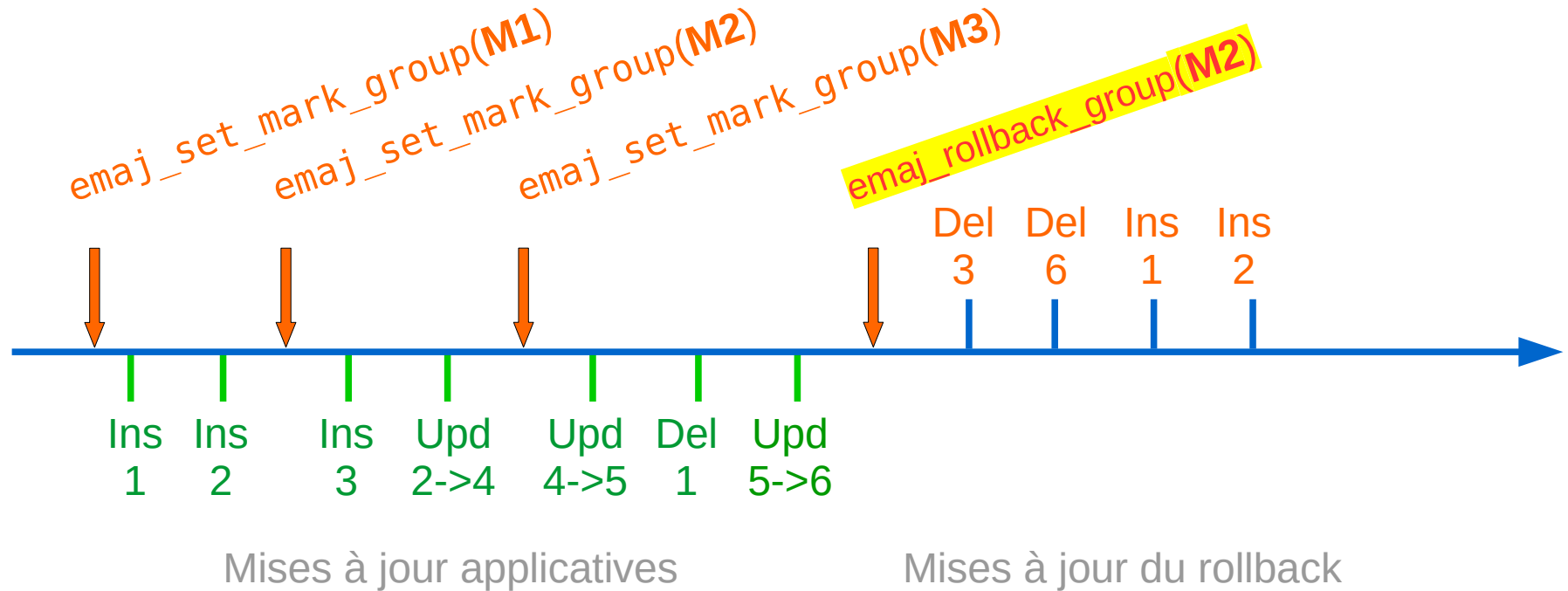
- 2 fonctions statistiques
 - `emaj_log_stat_group (groupe, marque_début, marque_fin)`
retourne rapidement des estimations du nombre de mises à jour enregistrées
 - par table
 - entre 2 marques (ou entre 1 marque et la situation courante)
 - `emaj_detailed_log_stat_group (groupe, marque_début, marque_fin)`
parcourt les tables de log et retourne des statistiques précises sur leur contenu
 - par table
 - par type de requête (INSERT / UPDATE / DELETE / TRUNCATE)
 - par ROLE à l'origine des mises à jour
 - entre 2 marques (ou entre 1 marque et la situation courante)

Annuler des mises à jour : le rollback « simple »

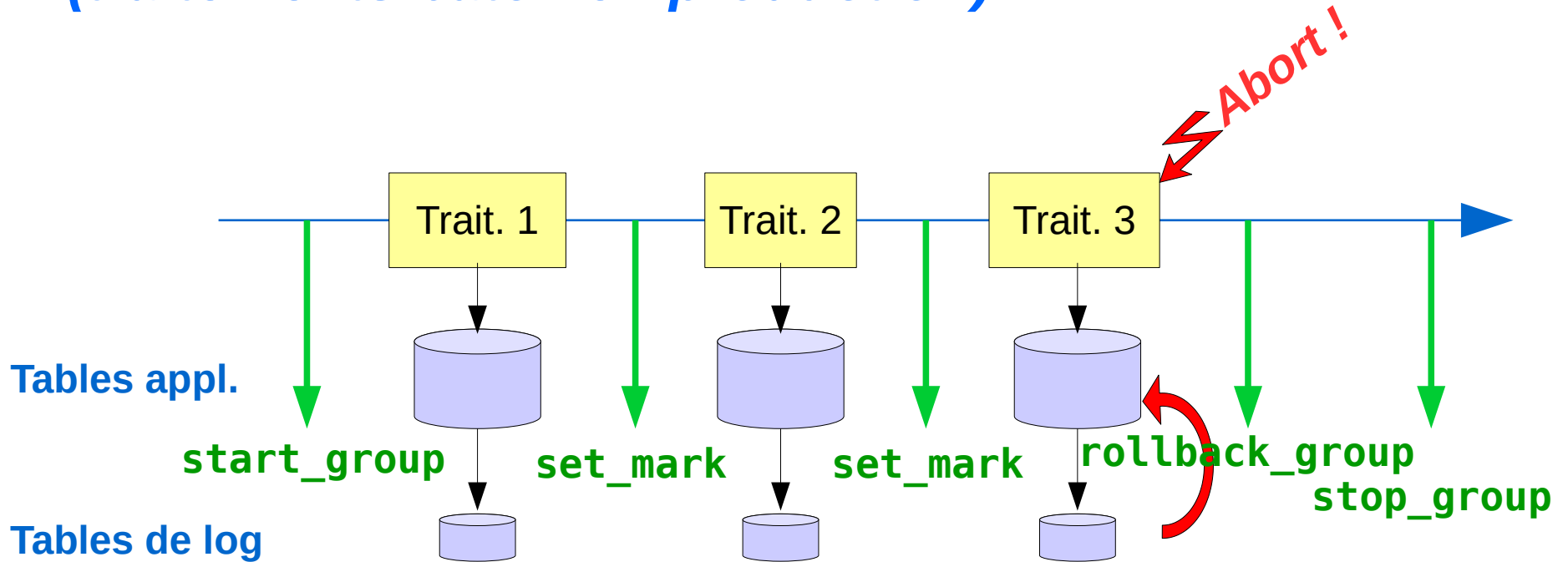
- Une fonction de « rollback » permet de remettre un groupe de tables dans l'état dans lequel il se trouvait à une marque donnée
 - `emaj_rollback_group (groupe, marque, false)`
- Fonctionnement
 - Les triggers de log sont désactivés le temps de l'opération
 - Chaque table est remise à l'état correspondant à la marque par un algorithme optimisé
 - Les séquences applicatives sont remises à l'état correspondant à la marque
 - Prend en compte les éventuelles clés étrangères
 - Les logs et les marques annulés sont supprimés
 - => tout ce qui est postérieur à la marque de rollback est « oublié »

Un algorithme de rollback optimisé

- Ne traite qu'une seule fois chaque valeur de clé primaire



Un enchaînement E-Maj typique (traitements batch en production)

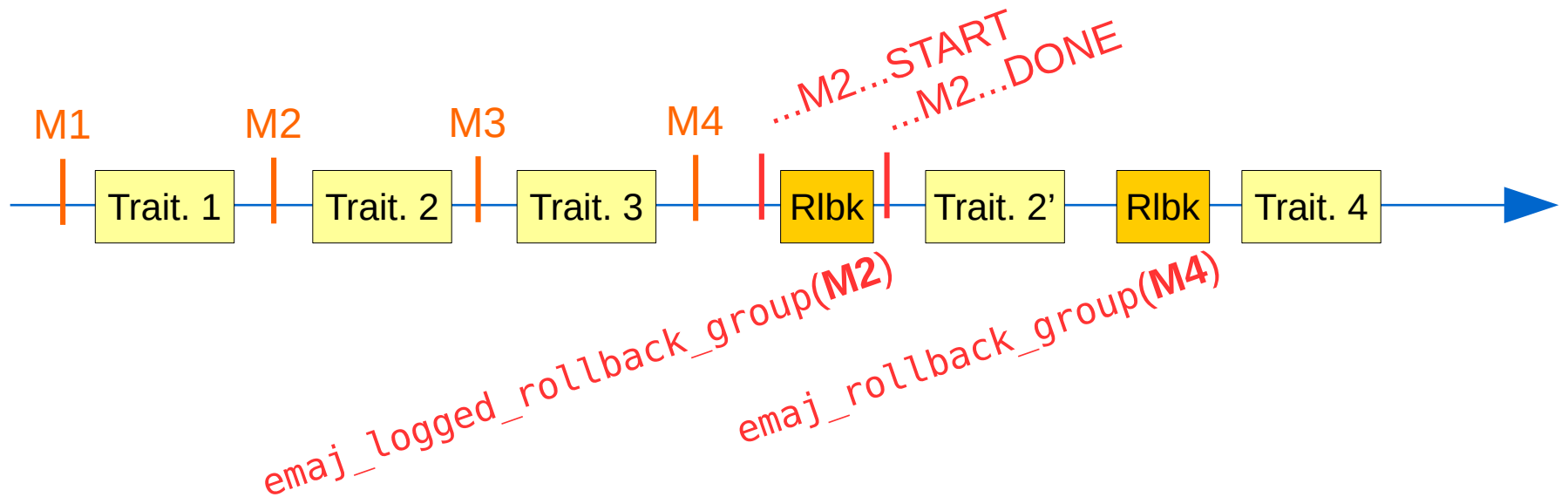


Annuler des mises à jour : le rollback « tracé »

- `emaj_logged_rollback_group` (*groupe, marque, false*)
- Il se distingue du rollback « simple » par le fait que
 - les triggers de logs ne sont PAS désactivés lors de l'opération
=> les mises à jour générées par le rollback sont enregistrées
 - les logs et marques annulées ne sont PAS supprimés
- On pourra donc annuler un rollback E-Maj ! Et plus généralement faire voyager un groupe de tables dans le temps !
- 2 marques sont automatiquement posées avant et après le rollback
 - `RLBK_<marque cible>_<HH.MI.SS.MS>_START`
 - `RLBK_<marque cible>_<HH.MI.SS.MS>_DONE`
- Pendant le rollback les tables restent accessibles en lecture

Un enchaînement E-Maj typique en environnement de test

- Un enchaînement de 4 traitements à tester
- Après le test 3, une nouvelle version du traitement 2 à retester
- Puis poursuite des tests restants



Estimer la durée d'un rollback E-Maj

- Pour savoir si on a le temps de réaliser l'opération ou si un autre moyen de remise en état ne serait pas plus rapide
- Une fonction estime la durée nécessaire pour rollbacker un groupe à une marque donnée
 - `emaj_estimate_rollback_group (groupe, marque)`

Paralléliser un rollback E-Maj

- Un client php ou perl effectue des rollbacks avec parallélisme
 - `emajParallelRollback.php -d <database> -h <host> -p <port> -U <user> -W <password> -g <nom_groupe ou listes_groupes> -m <marque> -s <nb_sessions> [-l]`
- Répartit automatiquement les tables à traiter dans un nombre donné de sessions
- Toutes les sessions appartiennent à une seule transaction (2PC)
 - => `max_prepared_transactions` >= nb sessions
- Nécessite php ou perl avec son extension PostgreSQL

Suivre les rollbacks E-Maj en exécution

- Une fonction
 - `SELECT * FROM emaj.emaj_rollback_activity ();`
 - restitue
 - les caractéristiques des rollbacks (groupe, marque...)
 - leur état
 - leur durée écoulée
 - une estimation de la durée restante et du % réalisé
- Nécessite la valorisation du paramètre « `dblink_user_password` » dans la table `emaj_param`

Suivre les rollbacks E-Maj

- Un client php ou perl pour suivre les rollbacks en cours / terminés
 - `emajRollbackMonitor.php -d <database> -h <host> -p <port> -U <user> -W <password> -n <nb_itérations> -i <rafraichissement_en_secondes> -l <nb_rollbacks_terminés> -a <historique_rollbacks_terminés_en_heures>`

```
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
```

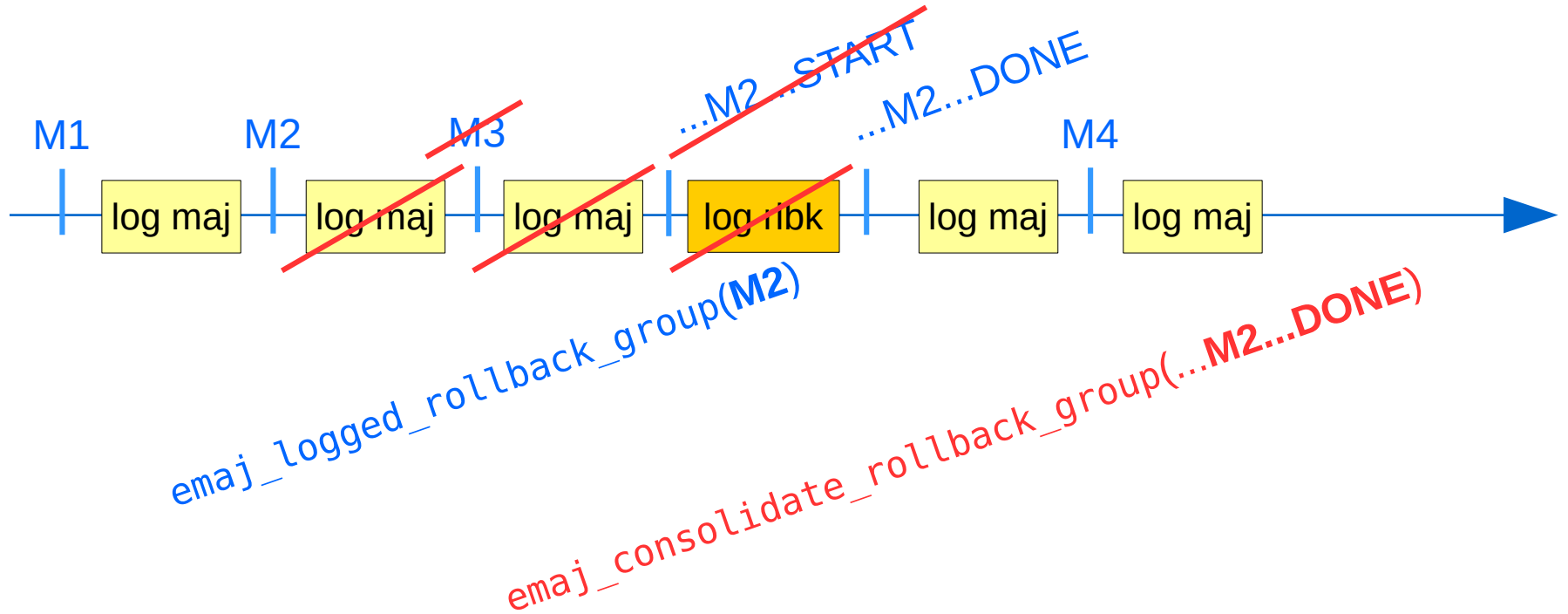
```
** rollback 35 started at 2023-03-21 08:31:17.180421+01 for groups {myGroup1}  
status: COMMITTED ; ended at 2023-03-21 08:31:17.480194+01
```

```
-> rollback 36 started at 2023-03-21 08:29:26.003502+01 for groups {group20101}  
status: EXECUTING ; completion 85 %; 00:00:20 remaining
```

Consolider un rollback « tracé »

- « Consolider » un rollback, c'est transformer un « rollback tracé » en « rollback simple »
- Les logs et marques intermédiaires sont supprimés, permettant de récupérer de la place dans les logs
 - `emaj_consolidate_rollback_group (groupe, marque_fin_de_rollback)`
- Les tables peuvent être mises à jour pendant la consolidation
- Une fonction restitue la liste des rollbacks consolidables
 - `emaj_get_consolidable_rollbacks ()`

Exemple de consolidation de rollback E-Maj



Se protéger contre des rollbacks E-Maj accidentels

- 2 fonctions pour gérer la protection d'un groupe de tables
 - `emaj_protect_group (groupe)`
 - `emaj_unprotect_group (groupe)`
- 2 fonctions pour gérer la protection d'une marque
 - `emaj_protect_mark_group (groupe, marque)` bloque toute tentative de rollback à une marque antérieure à la marque protégée
 - `emaj_unprotect_mark_group (groupe, marque)`



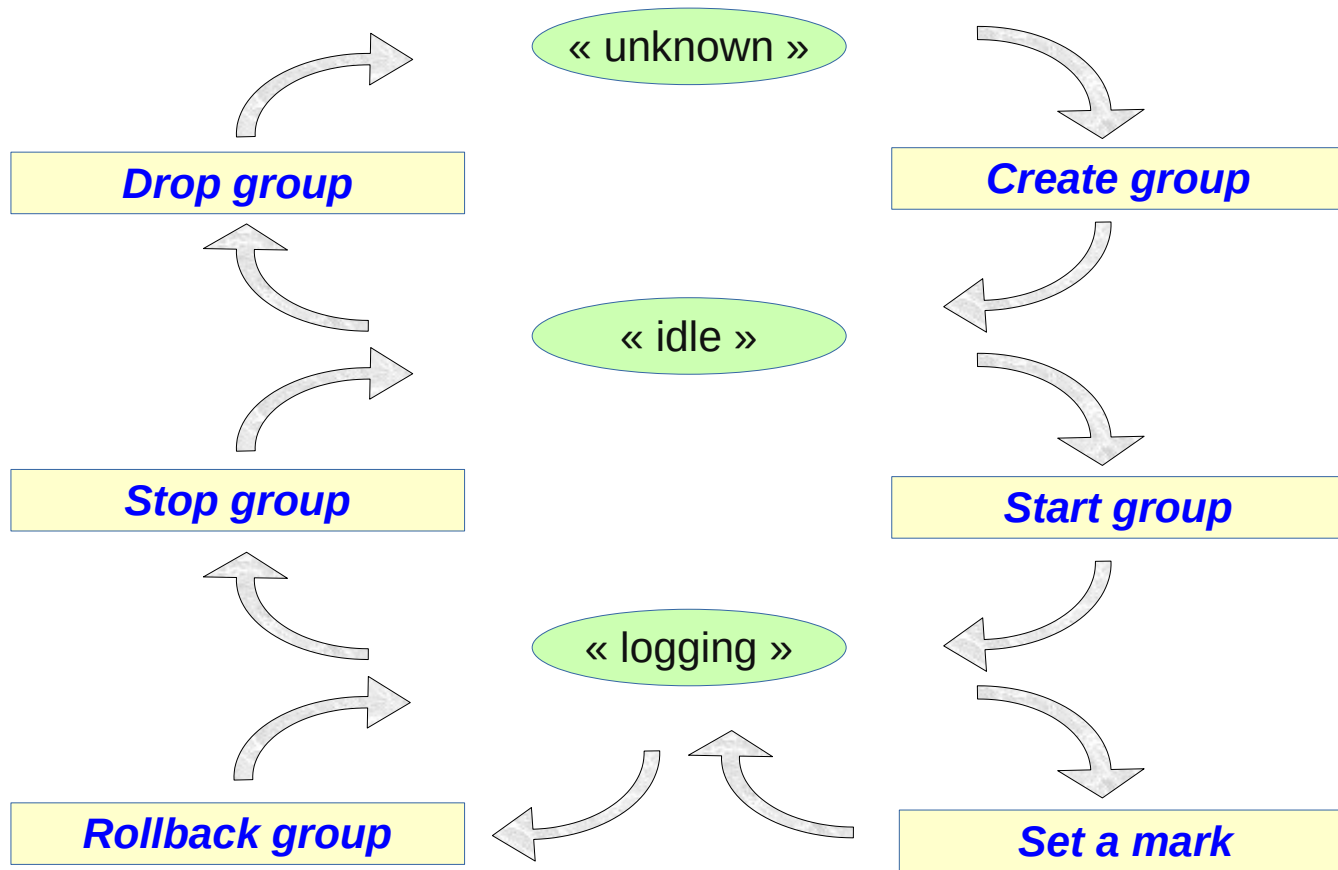
Exporter à partir d'un environnement E-Maj

- Utiles en test pour comparer plusieurs exécutions d'un même traitement ou « répliquer » les mises à jour d'un traitement
- Générer un script sql rejouant les mises à jour enregistrées entre 2 marques, pour tout ou partie des tables et séquences d'un groupe
 - Dans l'espace disque de l'instance :
`emaj_gen_sql_group (groupe, marque_début, marque_fin, fichier [,liste_tables/seq])`
 - N'importe où, avec psql :
`SELECT emaj_gen_sql_group (groupe, marque_début, marque_fin, NULL [,liste_tables/seq])
\copy (SELECT * FROM emaj_sql_script) TO 'fichier'`

Exporter à partir d'un environnement E-Maj (2)

- Vider sur fichiers dans un répertoire, par COPY, toutes les tables et séquences d'un groupe
 - *emaj_snap_group* (*groupe, directory, options_copy*)
- Vider sur fichiers dans un répertoire, par COPY, une partie des tables de log et des séquences d'un groupe
 - *emaj_snap_log_group* (*groupe, marque_début, marque_fin, directory, options_copy*)

Le cycle de vie d'un groupe de tables



Ajustement dynamique des groupes de tables

- Pour ajouter une ou plusieurs tables
 - `emaj_assign_table(schéma, table, groupe, propriétés [, marque])`
 - `emaj_assign_tables(schéma, liste de tables, groupe, propriétés [, marque])`
 - `emaj_assign_tables(schéma, filtre de sélection, filtre d'exclusion, groupe, propriétés [, marque])`
- Propriétés :
 - Format JSON
 - Pour définir la priorité et les tablespaces de data et index de log
- Filtres de sélection et d'exclusion : des RegExp

Ajustement dynamique des groupes de tables (suite)

- Exemple

- `emaj_assign_tables('monschéma', 'tbl.*', '_sav$', 'monGroupe', '{"priority":1}')`

affecte au groupe 'monGroupe' et avec une priorité 1 toutes les tables du schéma 'monschema' dont le nom commence par 'tbl' et ne se termine pas par '_sav'

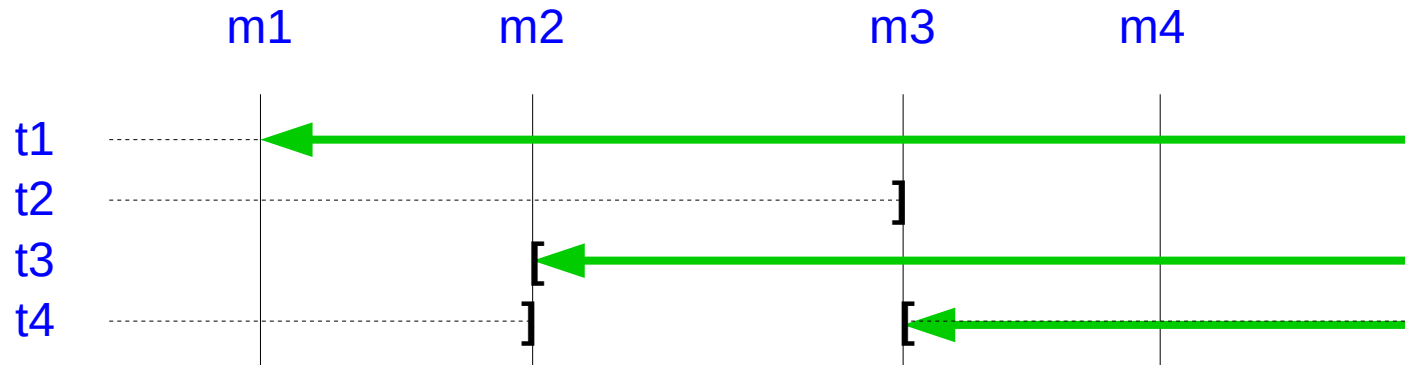
Ajustement dynamique des groupes de tables (suite)

- Sur le même principe :
 - `emaj_assign_sequence()` et `emaj_assign_sequences()`
 - `emaj_modify_table()` et `emaj_modify_tables()`
 - `emaj_move_table()` et `emaj_move_tables()`
 - `emaj_move_sequence()` et `emaj_move_sequences()`
 - `emaj_remove_table()` et `emaj_remove_tables()`
 - `emaj_remove_sequence()` et `emaj_remove_sequences()`

Impact des changements de structure de groupes actifs sur les rollbacks

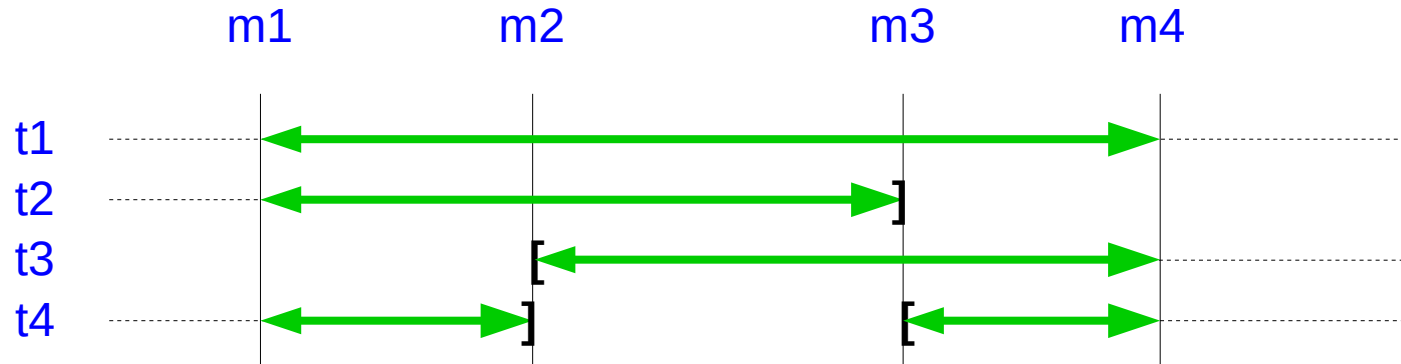
Table t2 sortie à la marque m3, t3 entrée à m2, t4 sortie à m2 et entrée à m3

`emaj_rollback_group(<groupe>,'m1')` traiterait :



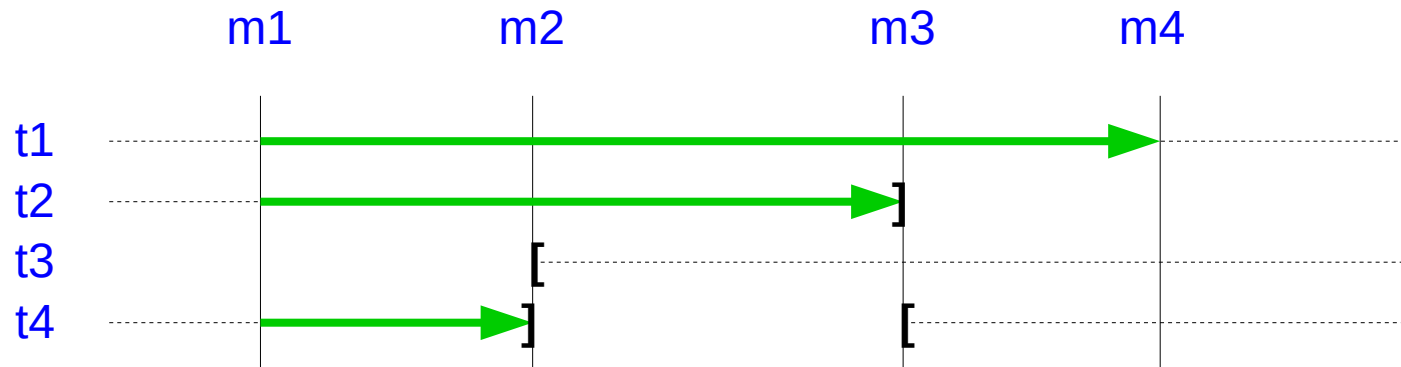
Impact des changements de structure de groupes actifs sur les statistiques

`emaj_log_stat_group(<groupe>,'m1','m4')` produirait :



Impact des changements de structure de groupes actifs sur la génération de script

emaj_gen_sql_group(<groupe>,'m1','m4') traiterait :



Modifier la structure d'une table d'un groupe actif

- Pour des actions comme : renommer la table, changer son schéma, ajouter/supprimer/renommer une colonne, changer le type d'une colonne
- La table de log doit aussi changer de structure
- 3 étapes
 - Sortie de la table de son groupe de tables
 - ALTER TABLE
 - Entrée de la table dans son groupe de tables
- Contrainte : un rollback E-Maj à une marque antérieure ne pourra pas remonter au-delà du changement de structure
- Idem pour renommer une séquence ou changer son schéma

Traiter plusieurs groupes en une seule opération

- Quelques variantes « multi-groupes » de fonctions
 - `emaj_start_groups (tableau_de_groupes, ...)`
 - `emaj_stop_groups (tableau_de_groupes, ...)`
 - `emaj_set_mark_groups (tableau_de_groupes, ...)`
 - `emaj_rollback_groups (tableau_de_groupes, ...)`
 - `emaj_logged_rollback_groups (tableau_de_groupes, ...)`
 - `emaj_log_stat_groups (tableau_de_groupes, ...)`
 - `emaj_gen_sql_groups (tableau_de_groupes, ...)`
- Permettent d'avoir des marques communes à plusieurs groupes
- Les 2 syntaxes PostgreSQL pour valoriser un tableau de groupes
 - `ARRAY['groupe 1', 'groupe 2', ...]`
 - `'{"groupe 1", "groupe 2", ... }'`

Gérer les marques

- Commenter une marque d'un groupe (ajout/modification/suppression)
 - `emaj_comment_mark_group (groupe, marque, commentaire)`
- Renommer une marque
 - `emaj_rename_mark_group (groupe, ancien_nom, nouveau_nom)`
- Supprimer une marque
 - `emaj_delete_mark_group (groupe, marque)`
 - Si la marque supprimée est la 1ère, les logs antérieurs à la 2ème sont effacés
- Supprimer toutes les marques antérieures à une marque donnée
 - `emaj_delete_before_mark_group (groupe, marque)`
 - Efface les logs antérieurs à la marque (ça peut être long !)

Gérer les marques (2)

- Rechercher des marques
 - `emaj_find_previous_mark_group (groupe, date-heure)` retourne la marque qui précède immédiatement la date et heure donnée
 - `emaj_find_previous_mark_group (groupe, marque)` retourne la marque qui précède immédiatement une marque donnée
- « `EMAJ_LAST_MARK` » représente la dernière marque posée pour un groupe
 - Utilisable pour tous les paramètres qui définissent une marque existante

Autres actions sur les groupes

- Commenter un groupe (ajout/modification/suppression)
 - `emaj_comment_group (groupe, commentaire)`
- Purger les tables de log d'un groupe arrêté (avant son prochain démarrage)
 - `emaj_reset_group (groupe)`
- Exporter / importer des configurations de groupes de tables
 - `emaj_export_groups_configuration ()`
 - `emaj_import_groups_configuration ()`
- Forcer l'arrêt d'un groupe (en cas de problème avec la fonction d'arrêt normale)
 - `emaj_force_stop_group (groupe)`

Actions diverses

- Vérifier la bonne santé de l'installation E-Maj
 - `emaj_verify_all ()`
- Récupérer l'identité de la table de log courante d'une table
 - `emaj_get_current_log_table ()`
- Purger manuellement les traces obsolètes
 - `emaj_purge_histories ()`
- Exporter/importer les configurations de paramètres
 - `emaj_export_parameters_configuration ()`
 - `emaj_import_parameters_configuration ()`

Log temporaire ou log permanent ?

- **Log temporaire** = Enchaînement du type
 - `emaj_start_group()`
 - répéter
 - traitement
 - `emaj_set_mark()`
 - `emaj_stop_group()`
 - Au redémarrage suivant les anciens logs sont purgés
 - Mais les arrêts et relances posent des verrous très lourds
- **Log permanent** = pas d'arrêt/relance régulier des groupes
 - Il faut régulièrement vider les logs des données obsolètes, avec la fonction `emaj_delete_before_mark()`
 - La suppression peut être coûteuse si le volume de log à effacer est important

Pour les grosses bases de données...

- Possibilité de stocker les tables de log et leur index dans des **tablespaces**
 - 2 propriétés optionnelle à l'assignation des tables dans les groupes

Pour garantir la fiabilité

- Aucune modification du moteur PostgreSQL
- Nombreux **contrôles** systématiques, en particulier au démarrage d'un groupe, à la pose d'une marque ou à un rollback :
 - Existence de toutes les tables, séquences, fonctions et triggers ?
 - Cohérence des colonnes entre les tables applicatives et les tables de log (existence, type) ?
- **Verrous** forts sur les tables lors des `start_group`, `set_mark_group` et `rollback_group`, pour être sûr qu'aucune transaction n'est en train de mettre à jour les tables applicatives
 - On peut influencer l'ordre de pose des verrous en définissant un niveau de priorité pour chaque table dans la table `emaj_group_def`
- Rollback de toutes les tables et séquences dans une seule **transaction**

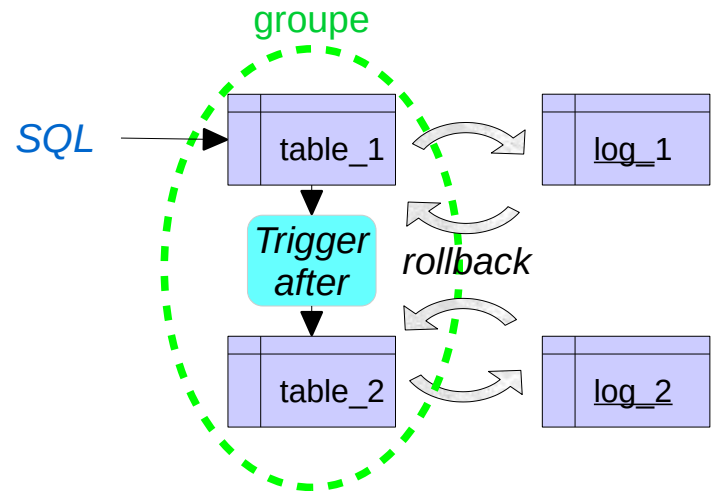
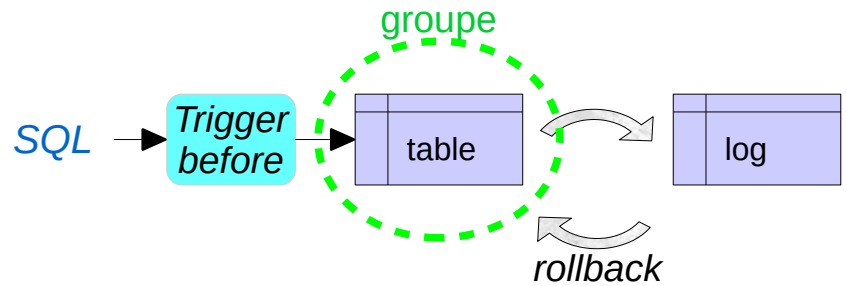
Pour garantir la fiabilité (suite)

- Des « **event triggers** » bloquent la suppression intempestive ou certaines modifications de composants (tables, séquences, fonctions...)
 - 2 fonctions pour désactiver/ré-activer le blocage
 - `emaj_disable_protection_by_event_triggers ()`
 - `emaj_enable_protection_by_event_triggers ()`

Impact des triggers applicatifs sur les rollbacks

E-Maj

- Trigger de type *BEFORE* sur une table appartenant à un groupe de tables
 - Les valeurs réellement écrites en base sont enregistrées dans les logs
 - => à désactiver au rollback E-Maj
- Trigger de type *AFTER* écrivant dans une autre table du même groupe de tables
 - Le rollback remettra les 2 tables dans le bon état
 - => à désactiver au rollback E-Maj



- Autres cas : Étudier les impacts



Gérer les triggers applicatifs

- Par défaut, les triggers applicatifs sont désactivés automatiquement lors des rollback E-Maj
- Un trigger peut être laissé dans son état lors du rollback s'il est enregistré comme tel
- 2 propriétés utilisables par les fonctions `emaj_assign_table()`, `emaj_assign_tables()`, `emaj_modify_table()` et `emaj_modify_tables()` pour spécifier les triggers à ne pas désactiver automatiquement
 - `"ignored_triggers"`: `["trg1","trg2",...]` liste les noms de triggers
 - `"ignored_triggers_profiles"`: `["regexp1","regexp2",...]` liste des expressions rationnelles de sélection des triggers

Pour concourir à la sécurité

- 2 rôles NOLOGIN dont les droits peuvent être donnés :
 - `emaj_adm` pour l'administration E-Maj
 - `emaj_viewer` pour la simple consultation des objets E-Maj (logs, marques, statistiques)
- Les objets E-Maj ne sont créés et manipulés que par un super-utilisateur ou un membre de `emaj_adm`
- Aucun autre droit n'est donné sur les schémas, tables et fonctions d'E-Maj
- Les triggers de log sont créés en « SECURITY DEFINER »
- Pas besoin de donner des droits supplémentaires sur les tables ou séquences applicatives

Performances

- Surcoût du log
 - Dépend largement du matériel et de la part des mises à jour SQL dans les traitements
 - Typiquement quelques % sur les durées de traitement
 - Mais beaucoup plus sur des chargements purs de données
- Durée de rollback
 - Dépend évidemment du nombre de mises à jour à annuler
 - Dépend aussi largement
 - de la configuration du serveur,
 - de la structure des tables (taille des lignes, index, clés étrangères et autres contraintes...)
 - Mais sauf cas particulier, toujours moins long qu'une restauration logique

Emaj_web

- Pour les administrateurs et utilisateurs
- Tous les objets E-Maj (groupes, marques...) et leurs attributs
- (presque) toutes les actions possibles sur les objets E-Maj

Connexion : localhost:5411 - rôle « postgres » SQL | Historique | Déconnexion Français

Emaj_web > Pg 11 > postgres

Groupes Schémas Triggers Rollbacks E-Maj E-Maj

Groupes de tables en état "démarré" ⓘ

	Y	Groupes	Création	Tables	Séquences	Type	Marques	Actions	Commentaire
<input type="checkbox"/>		myGroup1	22 juil. 2020 07:56:51	5	1	🔄	3	🔒 ⏪ 🗑️	Useless comm...
<input type="checkbox"/>		myGroup2	22 juil. 2020 07:56:51	4	2	🔄	4	🔒 ⏪ 🗑️	

Sélectionner Actions sur les objets (0)
Tous / Visibles / Aucun / Inverser ⏪ ⏩ 🗑️

Groupes de tables en état "arrêté" ⓘ

	Y	Groupes	Création	Tables	Séquences	Type	Marques	Actions	Commentaire
<input type="checkbox"/>		phil's group#3	22 juil. 2020 07:56:51	2	1	🛑	0	▶️ 🗑️ 🗑️	

Sélectionner Actions sur les objets (0)
Tous / Visibles / Aucun / Inverser ▶️ 🗑️ 🗑️

Nouveau groupe Exporter Importer

Liste des groupes de tables

Emaj_web : détail d'un groupe de tables

Connexion : localhost:5411 - rôle « postgres » SQL | Historique | Déconnexion Français

Emaj_web > Pg 11 > postgres > myGroup1

Propriétés Statistiques log Contenu

Propriétés du groupe de tables "myGroup1"

Etat	Création	Type	Tables	Séquences	Marques	Taille log
	mer. 22 juil. 07:56:51		5	1	3	144 kB

Commentaire : *Useless comment!*

Poser une marque
Protéger
Arrêter
Commenter

Marques du groupe de tables "myGroup1"

	Marque	Etat	Posée à	Mises à jour	Cumul mises à jour i	Actions	Commentaire
<input type="checkbox"/>	MARK3		mer. 22 juil. 07:56:52	0	0	1	
<input type="checkbox"/>	MARK2		mer. 22 juil. 07:56:52	7	7	1	End of 1st p...
<input type="checkbox"/>	MARK1		mer. 22 juil. 07:56:51	19	26		

Sélectionner Actions sur les objets (0)

Tous / Visibles / Aucun / Inverser

Limitations actuelles

- Depuis E-Maj 4.2, la version PostgreSQL minimum requise est la **11**
- Les tables applicatives appartenant à un groupe « rollbackable » doivent avoir une **PRIMARY KEY**
- Les requêtes de **DDL** ne peuvent pas être gérées par E-Maj

Pour conclure...

- Beaucoup plus d'**informations** dans la documentation et dans les fichiers README et CHANGES
- Grand **merci** pour leur aide à :
 - Andreas Scherbaum (qui a fait germer l'idée), Jean-Paul Argudo et l'équipe Dalibo (qui ont incité à poursuivre), les DBA de la CNAF (utilisateurs fidèles), Don Levine (qui a aidé dans la traduction anglaise)
 - Tous ceux qui m'ont contacté pour m'adresser leur commentaires ou doléances...
- N'hésitez pas à faire des **retours** sur github ou par email (phb.emaj@free.fr)