E-Maj

-

Avec cette extension PostgreSQL, vos données voyagent dans le temps !

L'acronyme de
« *E*nregistrement des *M*ises *A* *J*our »

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/
  - Sources disponibles sur github.com - https://github.com/beaud76/emaj
- 2 **clients web**
  - L'application Emaj_web - https://github.com/beaud76/emaj_web
  - Un plug-in pour phpPgAdmin - https://github.com/beaud76/emaj_ppa_plugin
- Une **documentation** en ligne
  - En français (ou anglais) - https://emaj.readthedocs.io/fr/stable/

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**

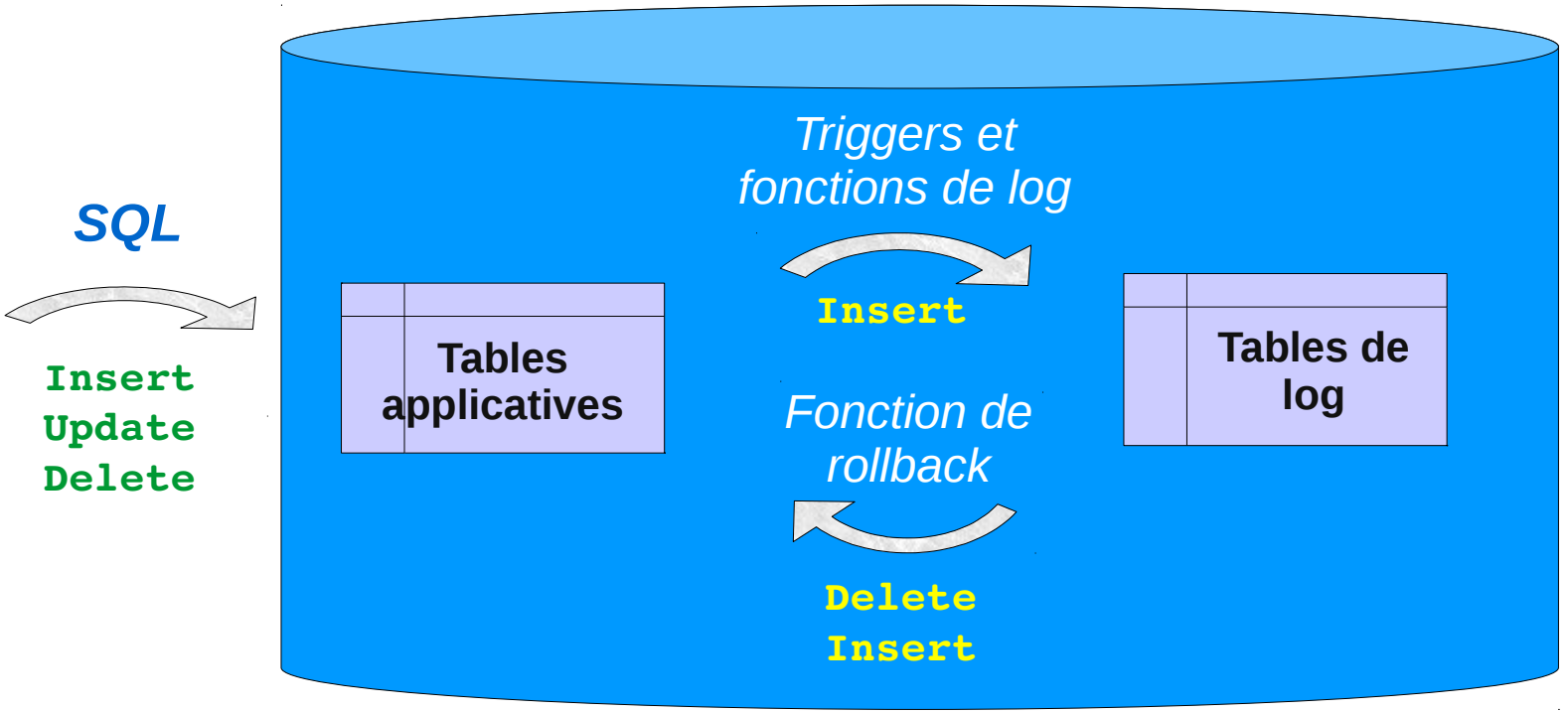
Concepts

- **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 »

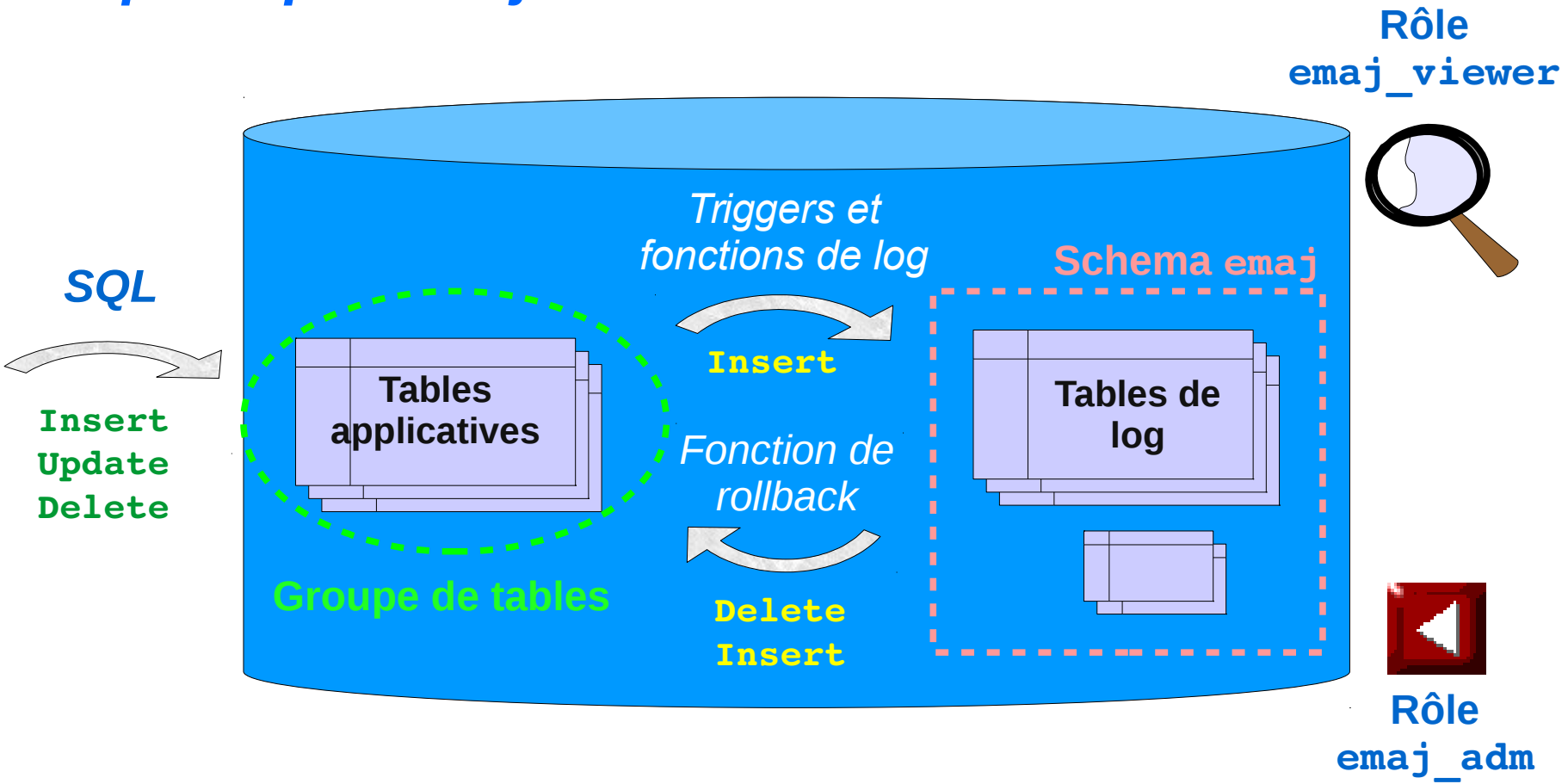
Concepts (complément)

- 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
    - les TRUNCATE sont enregistrés et non bloqués
    - Il n’est pas indispensable que chaque table ait une PRIMARY KEY

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
- Installer : `sudo make install`
- Se connecter à la base ciblée en tant que super-utilisateur et exécuter
  - `CREATE EXTENSION IF NOT EXISTS dblink;`
  - `CREATE EXTENSION IF NOT EXISTS btree_gist;`
  - `CREATE EXTENSION emaj;`
- L'installation ajoute à la database
  - 1 schema 'emaj' avec environ 110 fonctions, 15 tables techniques, 8 types, 1 vue, 1 séquence, 2 event triggers
  - 2 rôles

Initialisation

- Alimentation de la table `emaj_group_def` pour définir le contenu des groupes de tables
  - 1 ligne par table/séquence applicative
  - Au moins les colonnes `grpdef_group`, `grpdef_schema` et `grpdef_tblseq`
- Pour chaque groupe :
  - `SELECT emaj_create_group (groupe, est_rollbackable);`
  - crée pour chaque table applicative :
    - 1 table de log + 1 séquence
    - 1 trigger + 1 fonction de log
  - 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 debuging des applications
- Chaque table applicative a sa table de log
  - par défaut `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` (ancienne ligne)
  - 2 lignes de log pour un `UPDATE` (ancienne et nouvelle lignes)
- Un `TRUNCATE` génère une ligne de log

Les colonnes techniques des tables de log

- 8 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`
  - `emaj_user_ip` : adresse ip du client - `inet_client_addr()`
  - `emaj_user_port` : port ip du client – `inet_client_port()`
- On peut ainsi identifier les clients, le découpage des transactions et analyser le timing d'exécution du traitement

Compter les mises à jour

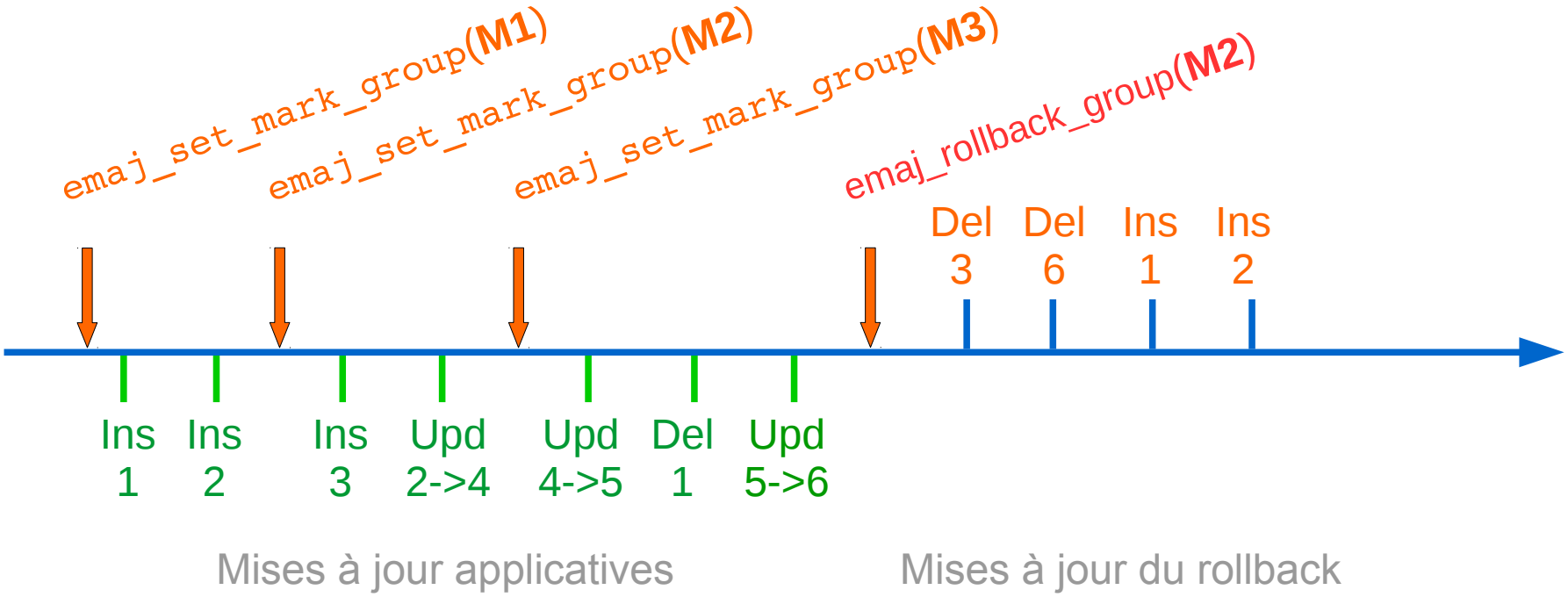
- 2 fonctions statistiques
  - `emaj_log_stat_group (`*`groupe`*`, `*`marque_début`*`, `*`marque_fin`*`)`
    retourne rapidement des estimations du nombre de mises à jour enregistrées
    - par table
    - entre 2 marques (ou entre 1 marque et la situation courante)
  - `emaj_detailed_log_stat_group (`*`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)
    - 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 »

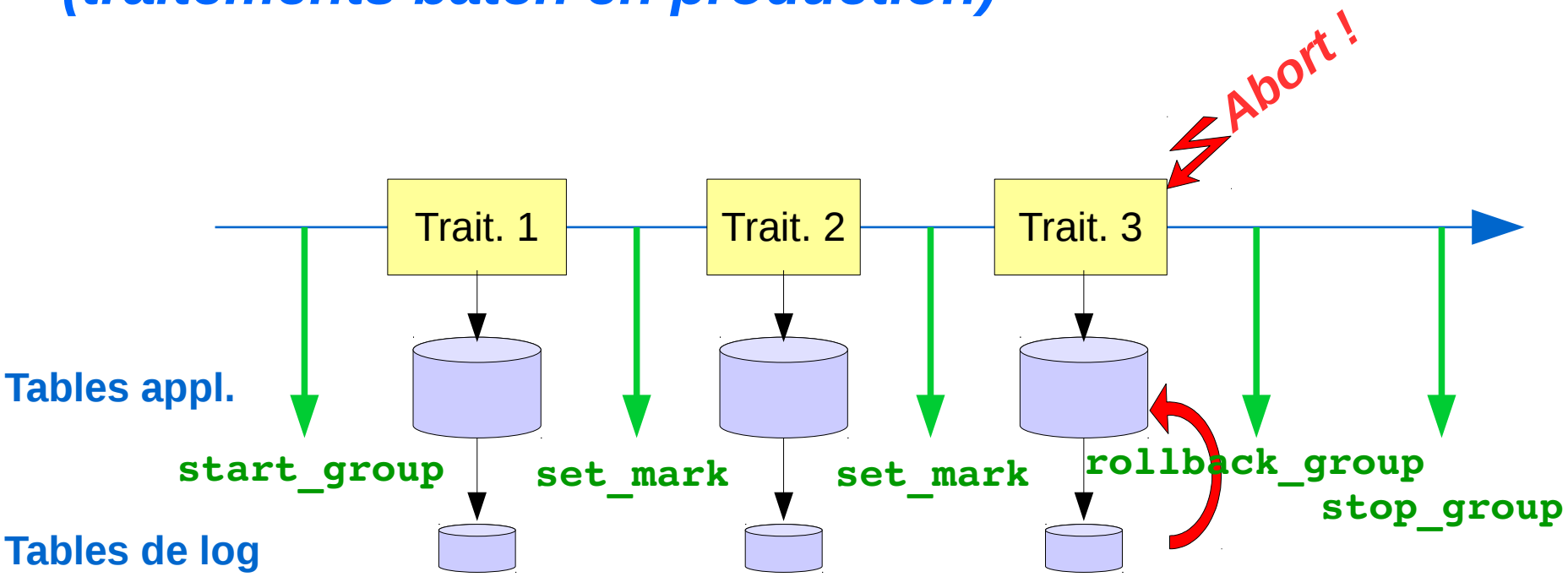
- Une fonction de « rollback » permet de remettre un groupe de tables dans l'état dans lequel il se trouvait à une marque donnée
  - `emaj_rollback_group (groupe, marque, false)`
- Fonctionnement
  - Les triggers de log sont désactivés le temps de l'opération
  - Chaque table est remise à l'état correspondant 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)

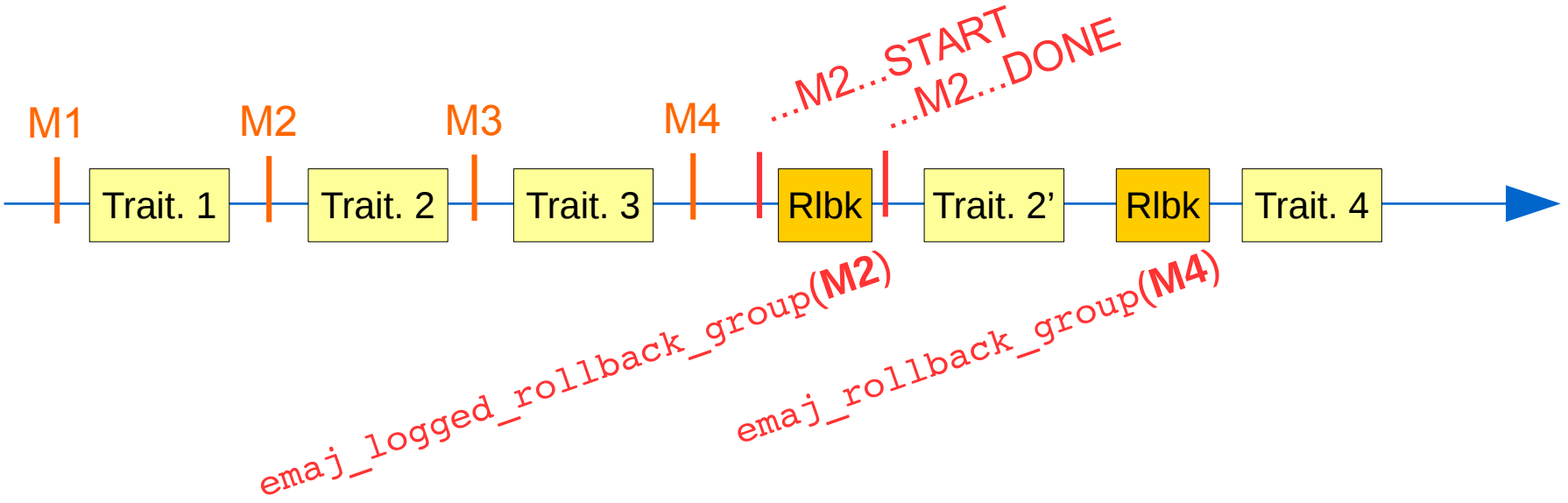


Annuler des mises à jour : le rollback « tracé »

- 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 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_transaction >= nb sessions
- Nécessite php 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 module php pour suivre les rollbacks en cours ou 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>`

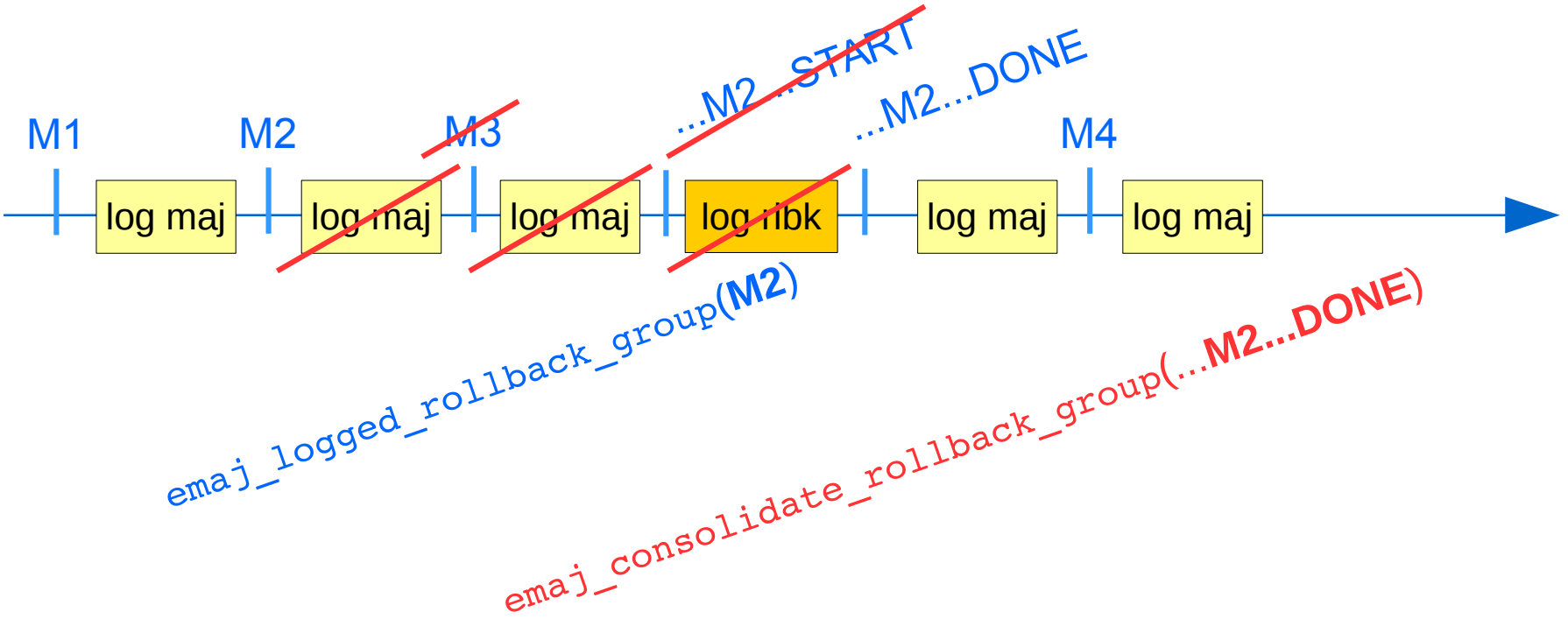
```
 E-Maj (version 2.2.0) - Monitoring rollbacks activity
------------------------------------------------------------

04/09/2017 - 12:07:17
** rollback 35 started at 2017-09-04 12:06:21.474217+02 for groups {myGroup1}
   status: COMMITTED ; ended at 2017-09-04 12:06:21.787615+02
-> rollback 36 started at 2017-09-04 12:04:31.769992+02 for groups {group1232}
   status: EXECUTING ; completion 89 % ; 00:00:20 remaining
-> rollback 37 started at 2017-09-04 12:04:21.894546+02 for groups {group1233}
   status: LOCKING ; completion 0 % ; 00:22:20 remaining
```

Consolider un rollback « tracé »

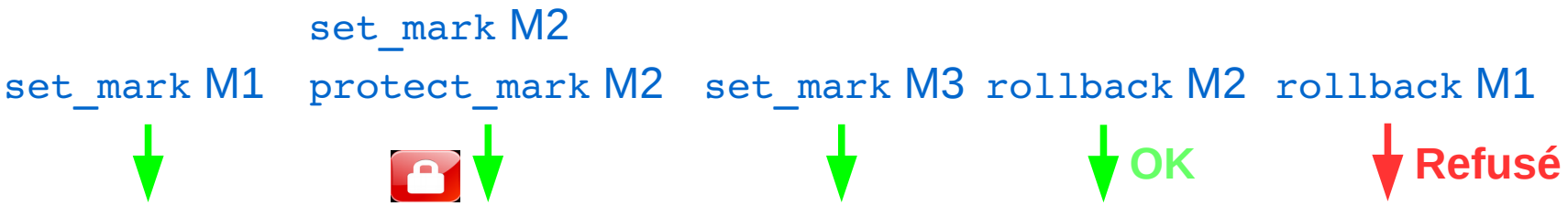
- « Consolider » un rollback, c'est transformer un « rollback tracé » en « rollback simple »
- Les logs et marques intermédiaires sont supprimés, permettant de récupérer de la place dans les logs
  - `emaj_consolidate_rollback_group (`*`groupe`*`, `*`marque_fin_de_rollback`*`)`
- Les tables peuvent être 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

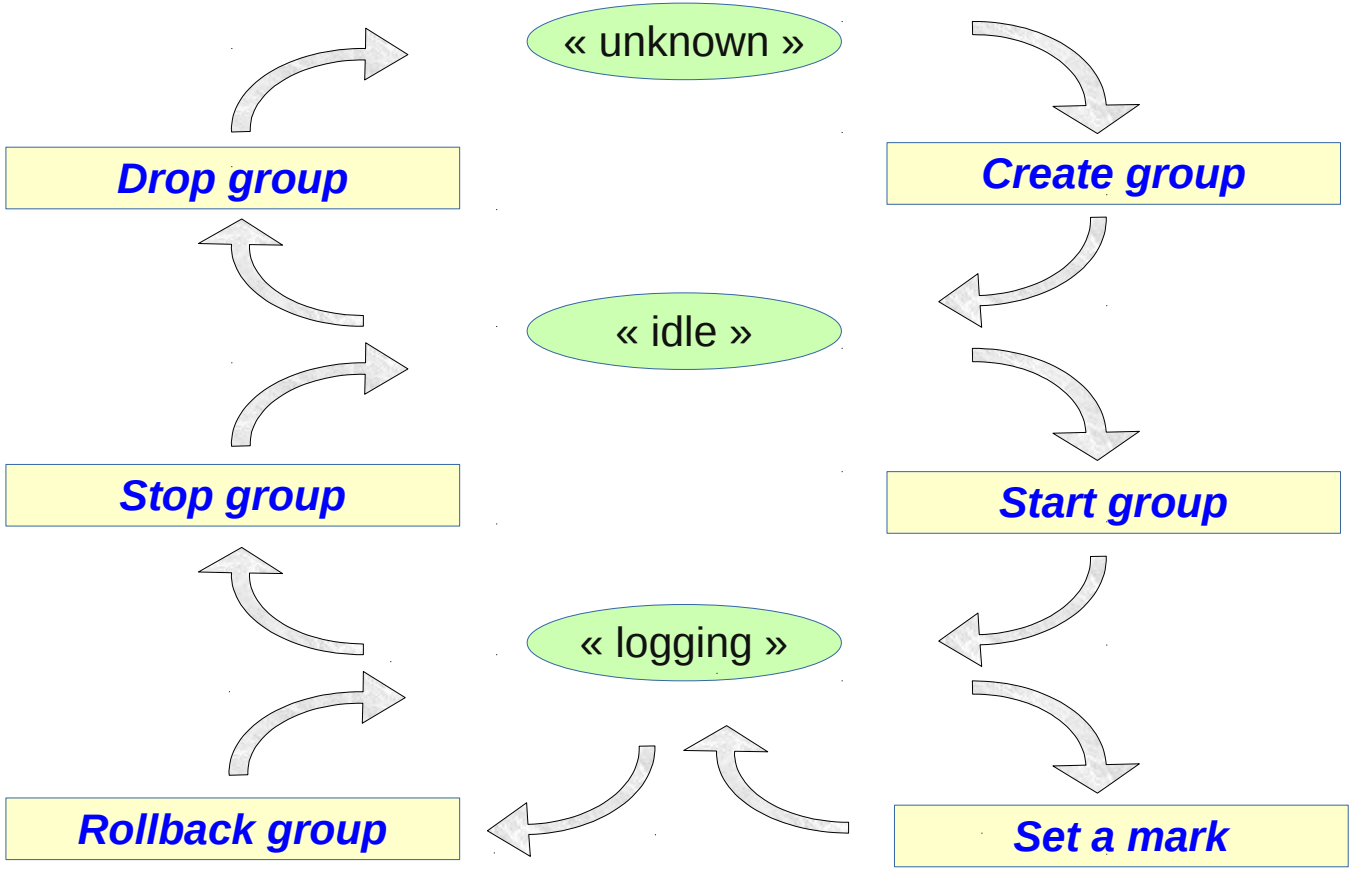
- 2 fonctions pour gérer la protection d'un groupe de tables
  - `emaj_protect_group (`*`groupe`*`)`
  - `emaj_unprotect_group (`*`groupe`*`)`
- 2 fonctions pour gérer la protection d'une marque
  - `emaj_protect_mark_group (`*`groupe`*`, `*`marque`*`)` bloque toute tentative de rollback à une marque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



Exporter à partir d'un environnement E-Maj

- Générer un script sql rejouant les mises à jour enregistrées entre 2 marques, pour tout ou partie des tables et séquences d'un groupe
  - `emaj_gen_sql_group (groupe, marque_début, marque_fin, fichier [,liste_tables/seq])`
- 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)`
- Utiles en test pour comparer plusieurs exécutions d'un même traitement ou « répliquer » les mises à jour d'un traitement

Le cycle de vie d'un groupe de tables



Modifier la structure des groupes

- Il faut :
  - Modifier le contenu de la table `emaj_group_def` (ajout/suppression de ligne, modification d'attributs)
  - Puis appeler la fonction `emaj_alter_group (groupe)`
- Le groupe de tables doit être arrêté avant l'appel de la fonction pour
  - l'ajout d'une table ou séquence
  - la modification de structure d'une table applicative
- Le groupe peut rester actif pour
  - les modifications d'attributs dans `emaj_group_def`
  - La sortie d'une table ou séquence du groupe de tables

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_gen_sql_groups (`*`tableau_de_groupes`*`, … )`
  - `emaj_alter_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)`
- 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 (çà 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)`
- 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 ()`

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**
  - Configurable pour chaque table dans `emaj_group_def`
- Possibilité de mettre les objets de log dans des **schémas secondaires** dédiés
  - Configurable pour chaque table dans `emaj_group_def`
  - Les schémas sont créés et supprimés automatiquement par E-Maj

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
- Rollback de toutes les tables et séquences dans une seule **transaction**

Pour garantir la fiabilité (suite)

- Les requêtes **`TRUNCATE`** sont bloquées pour les groupes « rollbackables » actifs
- Pour les versions de PostgreSQL ≥ 9.3, 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 ()`

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

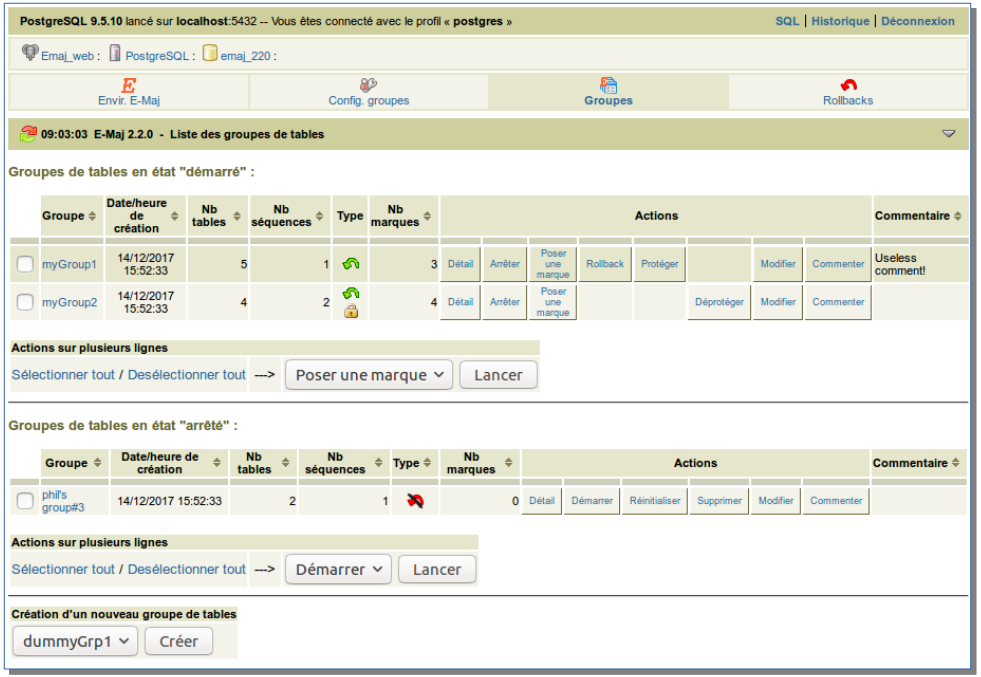
Performances

- 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

2 clients web

- 2 clients aux fonctionnalités identiques, pour aider les administrateurs et utilisateurs
  - Client indépendant **Emaj_web**
  - **Plug-in** complètement intégré dans **phpPgAdmin** (5.1+)

- Montre tous les objets E-Maj (groupes, marques...) et leurs attributs
- Permet toutes les actions possibles sur les objets E-Maj



Liste des groupes de tables

Emaj_web : détail d'un groupe de tables

PostgreSQL 9.5.10 lancé sur **localhost**:5432 -- Vous êtes connecté avec le profil « **postgres** » SQL | Historique | Déconnexion

Emaj_web : PostgreSQL : emaj_220 : E-Maj :

Propriétés | Statistiques log | Contenu

09:19:59 E-Maj 2.2.0 - Propriétés et marques du groupe de tables "myGroup1"

Propriétés du groupe de tables "myGroup1"

Etat	Date/heure de création	Nb tables	Nb séquences	Type	Nb marques	Taille du log
	2017-12-14 15:52:33.065525+01	5	1		3	160 kB

Commentaire : Useless comment!

Arrêter | Poser une marque | Modifier | Protéger | Commenter

Marques du groupe de tables "myGroup1"

Marque	Date-Heure	Etat	Nb mises à jour	Cumul mises à jour	Actions						Commentaire
MARK3	2017-12-14 15:52:33.552386+01		0	0	Rollback	Renommer	Effacer	Première marque	Protéger	Commenter	
MARK2	2017-12-14 15:52:33.504214+01		7	7	Rollback	Renommer	Effacer	Première marque	Protéger	Commenter	End of 1st program
MARK1	2017-12-14 15:52:33.396047+01		19	26	Rollback	Renommer	Effacer		Protéger	Commenter	

Propriétés et liste des mSarques d'un groupe de tables

Limitations actuelles

- Depuis E-Maj 2.2, la version PostgreSQL minimum requise est la **9.2**
- Les tables applicatives appartenant à un groupe « rollbackable » doivent avoir une **`PRIMARY KEY`**
- Les requêtes de **`TRUNCATE`** de table ne peuvent pas être annulées
- 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)