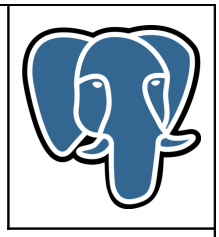


Extension PostgreSQL « E-Maj »

-

Manuel d'utilisation

Version 1.0.2

Mise à jour le dim. 10 mars 2013

Table des matières

1 - Introduction 5
1.1 - Contenu du document 5
1.2 - Licence 5
1.3 - Objectifs d'E-Maj 5

2 - Principes de fonctionnement 7
2.1 - Concepts 7
2.1.1 - Groupe de Tables 7
2.1.2 - Marque 7
2.1.3 - Rollback 7
2.2 - Architecture 8
2.2.1 - Les requêtes SQL tracées 8
2.2.2 - Les objets créés 8
2.2.3 - Norme de nommage des objets E-Maj 10
2.2.4 - Les schémas créés 10
2.2.5 - Les tablespaces utilisés 11

3 - Installation d'E-Maj 12
3.1 - Téléchargement et décompression de l'extension 12
3.1.1 - Téléchargement 12
3.1.2 - Décompression 12
3.2 - Installation de l'extension E-Maj 13
3.2.1 - Opérations préliminaires 14
3.2.2 - Installation des composants E-Maj 14
3.2.3 - Adaptation du fichier de configuration postgresql.conf 15
3.2.4 - Test et démonstration 15
3.3 - Mise à jour d'une version existante 16
3.3.1 - Démarche générale 16
3.3.2 - Dé-référencement d'une extension E-Maj antérieure 16
3.3.3 - Migration par désinstallation puis réinstallation 17
3.3.4 - Migration de E-Maj 0.10.0 à 0.10.1 18
3.3.5 - Migration de E-Maj 0.10.1 à 0.11.0 18
3.3.6 - Migration de E-Maj 0.11.0 à 0.11.1 20
3.3.7 - Migration de E-Maj 0.11.1 à 1.0.0 21
3.3.8 - Migration de E-Maj 1.0.0 à 1.0.1 21
3.3.9 - Migration de E-Maj 1.0.1 à 1.0.2 22
3.4 - Désinstallation d'E-Maj 23

4 - Utilisation d'E-Maj 24
4.1 - Mise en place de la politique d'accès à E-Maj 24
4.1.1 - Les rôles E-Maj 24
4.1.2 - Attribution des droits E-Maj 24
4.1.3 - Attribution des droits sur les tables et objets applicatifs 25
4.1.4 - Synthèse 25
4.2 - Fonctions principales 26
4.2.1 - Enchaînement des opérations 26
4.2.2 - Définition des groupes de tables 27
4.2.3 - Création d'un groupe de tables 29
4.2.4 - Démarrage d'un groupe de tables 30
4.2.5 - Pose d'une marque intermédiaire 31
4.2.6 - Rollback simple d'un groupe de tables 32
4.2.7 - Rollback annulable d'un groupe de tables 33
4.2.8 - Arrêt d'un groupe de tables 35
4.2.9 - Suppression d'un groupe de tables 36
4.2.10 - Modification d'un groupe de tables 36
4.3 - Fonctions multi-groupes 38
4.3.1 - Généralités 38
4.3.2 - Liste des fonctions multi-groupes 38
4.3.3 - Syntaxes pour exprimer un tableau de groupes 38
4.3.4 - Autres considérations 39
4.4 - Fonctions de gestion des marques 40
4.4.1 - Commentaires sur les marques 40
4.4.2 - Recherche de marque 40
4.4.3 - Renommage d'une marque 41
4.4.4 - Suppression d'une marque 41
4.4.5 - Suppression des marques les plus anciennes 42
4.5 - Fonctions statistiques 43
4.5.1 - Statistiques générales sur les logs 43
4.5.2 - Statistiques détaillées sur les logs 44
4.5.3 - Estimation de la durée d'un rollback 45
4.6 - Fonctions d'extraction de données 47
4.6.1 - Vidage des tables d'un groupe 47
4.6.2 - Vidage des tables de log d'un groupe 48
4.6.3 - Génération de scripts SQL rejouant les mises à jour tracées 49
4.7 - Autres fonctions 52
4.7.1 - Réinitialisation des tables de log d'un groupe 52
4.7.2 - Commentaires sur les groupes 52
4.7.3 - Vérification de la consistance de l'environnement E-Maj 52
4.7.4 - Arrêt forcé d'un groupe de tables 53

4.7.5 - Suppression forcée d'un groupe de tables....54
4.8 - Rollback avec parallélisme....55
4.8.1 - Sessions....55
4.8.2 - Préalables....55
4.8.3 - Syntaxe....55
4.8.4 - Exemples....56
5 - Considérations diverses....58
5.1 - Paramétrage....58
5.2 - Contrôles internes....59
5.3 - Traçabilité des opérations....59
5.4 - Impacts sur l'administration du cluster et de la base de données....61
5.4.1 - Arrêt/relance du cluster....61
5.4.2 - Sauvegarde et restauration....62
5.4.3 - Chargement de données....63
5.4.4 - Réorganisation des tables de la base de données....63
5.4.5 - Utilisation d'E-Maj avec de la réplication....64
5.4.6 - Changement de version de PostgreSQL....64
5.5 - Sensibilité aux changements de date et heure système....65
5.6 - Performances....65
5.6.1 - Surcoût de l'enregistrement des mises à jour....65
5.6.2 - Durée d'un rollback E-Maj....66
5.6.3 - Optimiser le fonctionnement d'E-Maj....66
5.7 - Limites d'utilisation....67
5.8 - Responsabilités de l'utilisateur....68
5.8.1 - Constitution des groupes de tables....68
5.8.2 - Exécution appropriée des fonctions principales....68
5.8.3 - Gestion des triggers applicatifs....68
5.8.4 - Modification des tables et séquences internes d'E-Maj....69
6 - Plugin phpPgAdmin....70
6.1 - Présentation générale....70
6.2 - Utilisation....70
6.2.1 - Comment accéder à E-Maj dans l'interface phpPgAdmin....70
6.2.2 - Liste des groupes de tables....71
6.2.3 - Composition des groupes de tables....73
6.2.4 - Détail d'un groupe de tables....74
6.2.5 - Statistiques pour les rollbacks....76
6.2.6 - Statistiques sur le contenu des tables de log....76
7 - Annexes....78
7.1 - Liste des fonctions E-Maj....78

1 INTRODUCTION

1.1 CONTENU DU DOCUMENT

Le présent document constitue le manuel d'utilisation de l'extension PostgreSQL E-Maj.

Le chapitre 2 présente les concepts utilisés par E-Maj puis l'architecture générale de l'extension.

Le chapitre 3 décrit les procédures d'installation, de changement de version et de désinstallation d'E-Maj.

Le chapitre 4 détaille le façon d'utiliser E-Maj. Chaque fonction y est décrite.

Le chapitre 5 apporte quelques compléments nécessaires à la bonne compréhension du fonctionnement de l'extension.

Enfin, le chapitre 6 présente l'extension E-Maj de l'outil d'administration phpPgAdmin.

1.2 LICENCE

Cette extension et toute la documentation qui l'accompagne sont distribuées sous licence GPL (GNU - General Public License).

1.3 OBJECTIFS D'E-MAJ

E-Maj est l'acronyme français de « Enregistrement des Mises A Jour ».

L'objectif principal d'E-Maj est de permettre des restaurations logiques du contenu d'un ensemble de tables dans un état prédéfini, sans restauration physique de l'ensemble des fichiers d'une instance (cluster) PostgreSQL, ni rechargement complet de l'ensemble des tables concernées.

Mais E-Maj peut également servir à tracer les mises à jours effectuées sur le contenu de tables par des traitements.
Il constitue une bonne solution pour :

- positionner à des moments précis des points de sauvegarde sur un groupe de tables,
- restaurer si nécessaire ce groupe de tables dans un état stable, sans arrêt du cluster,
- gérer plusieurs points de sauvegarde, chacun d'eux étant utilisable à tout moment comme point de restauration.

Ainsi, dans un environnement de production, E-Maj peut permettre de simplifier l'architecture technique utilisée, en offrant une alternative souple et efficace à des sauvegardes intermédiaires longues (*pg_dump*) et/ou coûteuses en espace disque (disques miroirs). E-Maj peut également apporter une aide au débogage, en offrant la possibilité d'analyser de façon précise les mises à jour effectuées par un traitement suspect sur les tables applicatives.

Dans un environnement de test, E-Maj permet également d'apporter de la souplesse dans les opérations. Il est ainsi possible de repositionner très facilement les bases de données dans des états stables prédéfinis afin de répéter autant de fois que nécessaire des tests de traitement.

2 PRINCIPES DE FONCTIONNEMENT

2.1 CONCEPTS

E-Maj s'appuie sur trois concepts principaux.

2.1.1 Groupe de Tables

Le « *groupe de tables* » (tables group) représente un ensemble de tables applicatives qui vivent au même rythme, c'est-à-dire dont, en cas de besoin, le contenu doit être restauré comme un tout. Il s'agit typiquement de toutes les tables mises à jour par un ou plusieurs traitements. Chaque groupe de tables est défini par un nom unique pour la base de données concernée. Par extension, un groupe de tables peut également contenir des séquences applicatives (au sens du SGBD). Les tables et séquences qui constituent un groupe peuvent appartenir à des schémas différents de la base de données.

A un instant donné, un groupe de tables est soit dans un état « *actif* », soit dans un état « *inactif* ». L'état actif signifie que les mises à jour apportées aux tables du groupe sont enregistrées.

Un groupe de tables est soit de type « *rollbackable* » (cas standard), soit de type « *audit_only* ». Dans ce second cas, il n'est pas possible de procéder à un rollback du groupe. En revanche, cela permet d'enregistrer à des fins d'observation les mises à jour du contenu de tables ne possédant pas de clé primaire.

2.1.2 Marque

Une « *marque* » (mark) est un point particulier dans la vie d'un groupe de tables correspondant à un état stable des tables et séquences du groupe. Elle est positionnée de manière explicite au travers d'une intervention de l'utilisateur. Une marque est définie par un nom unique au sein du groupe de tables.

2.1.3 Rollback

L'opération de « *rollback* » consiste à remettre toutes les tables et séquences d'un groupe dans l'état dans lequel elles se trouvaient lors de la pose d'une marque.

Il existe en fait deux types de rollback :

- avec un « *unlogged rollback* », aucune trace des mises à jour annulées par l'opération de rollback n'est conservée : il n'y a pas de mémoire de ce qui a été effacé,

- au contraire, dans une opération de « *logged rollback* », les annulations de mises à jour sont elles-mêmes tracées dans les tables de log, offrant ainsi la possibilité d'annuler l'opération de rollback elle même.

2.2 ARCHITECTURE

Pour mener à bien l'opération de rollback sans avoir conservé au préalable une image physique des fichiers du cluster PostgreSQL, il faut pouvoir enregistrer les mises à jour effectuées sur les tables applicatives de manière à pouvoir les annuler.

Avec E-Maj, cela prend la forme suivante.

2.2.1 Les requêtes SQL tracées

Les opérations de mises à jour enregistrées concernent les verbes SQL suivants :

- insertions de lignes :
  - ✔ *INSERT* élémentaires (*INSERT ... VALUES*) ou ensemblistes (*INSERT ... SELECT*)
  - ✔ *COPY ... FROM*
- mises à jour de lignes :
  - ✔ *UPDATE*
- suppression de lignes :
  - ✔ *DELETE*
- vidage de table :
  - ✔ *TRUNCATE* (à partir de PostgreSQL 8.4)

Pour les requêtes qui traitent plusieurs lignes, chaque création, modification ou suppression est enregistrée individuellement. Ainsi par exemple, une requête « *DELETE FROM <table>* » portant sur une table d'1 million de lignes génèrera l'enregistrement d'1 million de suppressions de ligne.

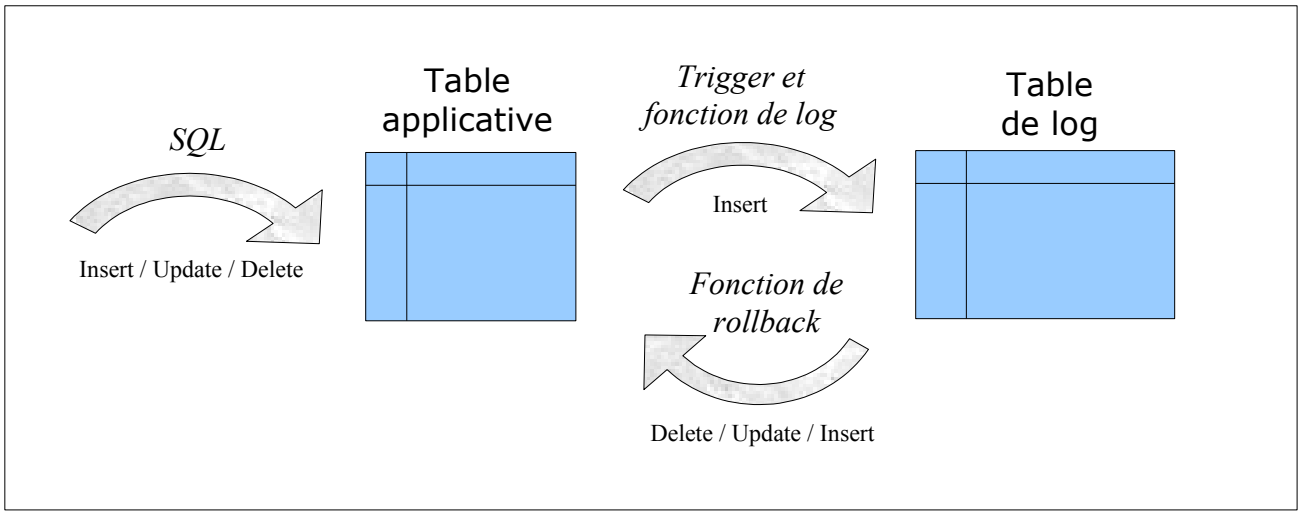
Le cas des verbes SQL *TRUNCATE* est spécifique. Comme aucun trigger de niveau ligne (*FOR EACH ROW*) n'est activable pour ce verbe, les conséquences d'un *TRUNCATE* ne peuvent pas être annulées par E-Maj. C'est pourquoi son exécution n'est autorisée que pour les groupes de tables créés en mode « *audit_only* ». Dans ce cas, seule l'exécution du verbe est enregistrée.

2.2.2 Les objets créés

Pour chaque table applicative sont créés :

- une table de log dédiée, qui contient les données correspondant aux mises à jour effectuées,
- un trigger et une fonction spécifique, permettant, lors de chaque création (*INSERT, COPY*), mise à jour (*UPDATE*) ou suppression (*DELETE*) de ligne, d'enregistrer dans la

table de log toutes les informations nécessaires à l'annulation ultérieure de l'action élémentaire,
- une fonction de rollback, qui permet d'annuler tout ou partie des mises à jour enregistrées pour la table, (sauf pour les tables appartenant à un groupe créé en mode « *audit_only* »),
- à partir des versions de PostgreSQL 8.4, un autre trigger permettant de bloquer toute exécution d'un verbe SQL *TRUNCATE* alors que les triggers de logs sont activés,
- une séquence qui permet de dénombrer très rapidement le nombre de mises à jour enregistrées dans les tables de log entre 2 marques.



Une table de log a la même structure que la table applicative correspondante. Elle comprend néanmoins quelques colonnes techniques supplémentaires :
- un identifiant unique, sous la forme d'un entier associé à une séquence globale,
- la date et l'heure précise de la mise à jour,
- le type d'opération SQL effectuée : INS pour *INSERT*, UPD pour *UPDATE* et DEL pour *DELETE*,
- un attribut 'OLD' ou 'NEW' permettant de distinguer les anciennes et nouvelles valeurs des lignes mises à jour,
- le numéro interne de la transaction à l'origine de la mise à jour (tx*id* PostgreSQL),
- le rôle de connexion à l'origine de la mise à jour,
- l'adresse ip de l'utilisateur à l'origine de la mise à jour.

Pour le bon fonctionnement d'E-Maj, un certain nombre d'objets techniques sont également créés à l'installation de cette extension :
- 10 tables,
- 3 types,
- 75 fonctions techniques, dont 35 directement appelables par les utilisateurs,
- 1 séquence, nommée *emaj_global_seq*, permettant d'associer à chaque mise à jour enregistrée dans une table de log quelconque de la base de données un identifiant unique de valeur croissante au fil du temps,
- 1 schéma spécifique, nommé *emaj*, qui contient tous ces objets,

- 2 rôles de type groupe (sans possibilité de connexion) : *emaj_adm* pour administrer les composants E-Maj, et *emaj_viewer* pour uniquement consulter les composants E-Maj.

Les quelques tables techniques dont il peut être utile de connaître la structure sont décrites dans les chapitres suivants (*emaj_group_def* est décrite dans le §4.2.2, *emaj_param* est décrite dans le §5.1 et *emaj_hist* est décrite dans le §5.3)

2.2.3 Norme de nommage des objets E-Maj

Les objets associés aux tables applicatives portent des noms construits en utilisant le nom de la table et de son schéma d'appartenance. Ainsi :

- le nom de la table de log est :
  <nom.du.schema>_<nom.de.la.table>_log

- le nom de la fonction de log est :
  <nom.du.schema>_<nom.de.la.table>_log_fnct

- le nom du trigger de log est :
  <nom.du.schema>_<nom.de.la.table>_emaj_log_trg

- le nom du trigger de blocage des verbes *TRUNCATE* est :
  <nom.du.schema>_<nom.de.la.table>_emaj_trunc_trg

- le nom de la fonction de rollback est :
  <nom.du.schema>_<nom.de.la.table>_rlbk_fnct

- le nom de la séquence associée à la table de log est :
  <nom.du.schema>_<nom.de.la.table>_log_seq

Le nom des autres fonctions E-Maj est aussi normalisé :
- les fonctions dont les noms commencent par 'emaj_' sont appelables par les utilisateurs,
- les fonctions dont les noms commencent par '_' sont des fonctions internes qui ne doivent pas être appelées directement.

2.2.4 Les schémas créés

Tous les objets techniques créés lors de l'installation de l'extension sont localisés dans le schéma *emaj*.

Par défaut, tous les objets liés aux groupes de tables sont créés dans le schéma principal *emaj*. Mais, au travers du paramétrage des groupes de tables, il est possible de localiser ces objets dans un ou plusieurs schémas secondaires. Le nom des schémas secondaires commencent par « emaj », seul leur suffixe étant paramétrable (voir le §4.2.2).

2.2.5 Les tablespaces utilisés

Lors de l'installation de l'extension et lors de la création des tables de log, E-Maj peut utiliser le tablespace par défaut.

Mais il est également possible de créer un tablespace dédié nommé *tspemaj*. S'il existe lors de l'installation ou de la création des tables de log, il sera utilisé comme support des tables créées.

Au travers du paramétrage des groupes de tables, il est aussi possible de créer les tables de log et leur index dans des tablespaces spécifiques (voir le §4.2.2).

3 INSTALLATION D'E-MAJ

Dans cette partie, nous allons décrire comment installer l'extension E-Maj. Un dernier chapitre est consacré à sa désinstallation.

3.1 TÉLÉCHARGEMENT ET DÉCOMPRESSION DE L'EXTENSION

Dans un premier temps, il faut se procurer E-Maj pour pouvoir ensuite l'installer.

3.1.1 Téléchargement

E-Maj est disponible en téléchargement sur deux sites Internet :

- PGXN (http://pgxn.org)
- pgFoundry.org (http://pgfoundry.org/projects/emaj/).

3.1.2 Décompression

L'extension est fournie sous la forme d'un unique fichier compressé. Pour pouvoir être utilisé, ce fichier doit donc être décompressé.

Sous Windows, vous pouvez utiliser votre utilitaire de décompression préféré. Sous Unix/Linux, une commande du type :

```
tar -xvzf emaj-<version>.tar.gz
```

peut être utilisée pour un fichier *.tar.gz* ou

```
unzip emaj-<version>.zip
```

pour un fichier zip.

On dispose maintenant d'un répertoire emaj-<version> comprenant l'arborescence suivante :

- sql/emaj.sql — script psql d'installation des composants E-Maj
- sql/emaj-1.0.1-to-1.0.2.sql — script de migration d'E-Maj d'une version 1.0.1 à 1.0.2
- sql/emaj-1.0.0-to-1.0.1.sql — script de migration d'E-Maj d'une version 1.0.0 à 1.0.1
- sql/emaj-0.11.1-to-1.0.0.sql — script de migration d'E-Maj d'une version 0.11.1 à 1.0.0

- sql/emaj-0.11.0-to-0.11.1.sql script de migration d'E-Maj d'une version 0.11.0 à 0.11.1
- sql/check-0.10.1-to-0.11.0-conditions.sql script psql de vérification des conditions de migration de 0.10.1 vers 0.11.0
- sql/emaj-0.10.1-to-0.11.0.sql script de migration d'E-Maj d'une version 0.10.1 à 0.11.0
- sql/emaj-0.10.0-to-0.10.1.sql script de migration d'E-Maj d'une version 0.10.0 à 0.10.1
- sql/emaj--0.10.1--unpackaged script de « déconstruction » d'une extension E-Maj installée en 0.10.1
- sql/emaj--0.10.0--unpackaged script de « déconstruction » d'une extension E-Maj installée en 0.10.0
- sql/demo.sql script psql de démonstration d' E-Maj
- sql/prep-pr.sql script psql de test pour les rollbacks parallélisés
- sql/uninstall.sql script psql de désinstallation
- README documentation réduite de l'extension
- CHANGES notes de versions
- LICENSE information sur la licence utilisée pour E-Maj
- AUTHORS identification des auteurs
- META.json données techniques destinées à PGXN
- doc/Emaj.<version>_doc_en.pdf documentation en anglais de l'extension E-Maj
- doc/Emaj.<version>_doc_fr.pdf documentation en français de l'extension E-Maj
- doc/Emaj.<version>_pres_en.pdf présentation en anglais de l'extension E-Maj
- doc/Emaj.<version>_pres_fr.pdf présentation en français de l'extension E-Maj
- php/emajParallelRollback.php outil de rollback parallélisé

3.2 INSTALLATION DE L'EXTENSION E-MAJ

Si une version d'E-Maj est déjà installée dans la base de données, allez au chapitre §3.3.

Dans les versions PostgreSQL 9.1 et suivantes, il existe une gestion intégrée des extensions, destinée à simplifier l'installation et la gestion de composants additionnels au SGBD. Malheureusement les caractéristiques d'E-Maj ne permettent pas d'utiliser de manière fiable cette gestion intégrée des extensions.

Quelques opérations préliminaires sont requises.

3.2.1 Opérations préliminaires

Pour ces opérations, l'utilisateur doit se connecter à la base de données concernée en tant que super-utilisateur, en utilisant par exemple *psql*.

Si le langage PL/PGSQL n'est pas activé (il n'est pas activé par défaut dans les versions de PostgreSQL antérieures à 9.0), il faut l'activer par la commande SQL suivante :

```
CREATE LANGUAGE plpgsql;
```

La seconde opération préliminaire est optionnelle. Elle consiste à créer un tablespace nommé *tspemaj*. S'il existe, les tables et index créées par E-Maj seront implantés dans ce tablespace, sauf paramétrage spécifique des groupes de tables (voir §4.2.2). Une fois créé, ce tablespace est partagé par toutes les bases de données du cluster PostgreSQL.

Pour créer le tablespace *tspemaj*, il faut d'abord créer l'espace de stockage qui sera associé, un répertoire pour Unix/Linux ou un dossier pour Windows, en le laissant vide de tout fichier. Puis il faut exécuter la commande SQL suivante :

```
CREATE TABLESPACE tspemaj LOCATION '<localisation.du.tablespace>';
```

Pour des questions de performance, il est recommandé d'implanter le tablespace *tspemaj* sur un espace disque distinct de celui qui supporte les tables applicatives.

3.2.2 Installation des composants E-Maj

Les composants E-Maj peuvent maintenant être installés dans la base de données, en exécutant depuis psql le script *emaj.sql* fourni.

```
\i <répertoire_emaj>/sql/emaj.sql
```

Le script commence par vérifier que la version de PostgreSQL est supérieure ou égale à la version 8.2, que le rôle qui exécute le script est bien un super-utilisateur.

Le script crée alors le schéma *emaj* avec ses tables techniques, ses types et ses fonctions.

Le schéma *emaj* ne doit contenir que des objets liés à E-Maj.

S'ils n'existent pas déjà, les 2 rôles *emaj_adm* et *emaj_viewer* sont également créés.

Enfin, le script d'installation examine la configuration du cluster. Le cas échéant, il affiche un message concernant le paramètre *-max_prepared_statements* (voir §4.8.2).

A la fin de son exécution, le script affiche le message :

>>> E-Maj objects successfully created

3.2.3 Adaptation du fichier de configuration postgresql.conf

Les fonctions principales d'E-Maj posent un verrou sur chacune des tables du groupe traité. Si le nombre de tables constituant le groupe est élevé, il peut s'avérer nécessaire d'augmenter la valeur du paramètre *max_locks_per_transaction* dans le fichier de configuration *postgresql.conf*. Ce paramètre entre dans le dimensionnement de la table en mémoire qui gère les verrous du cluster. Sa valeur par défaut est de 64. On peut le porter à une valeur supérieure si une opération E-Maj échoue en retournant un message d'erreur indiquant clairement que toutes les entrées de la table des verrous sont utilisées.

De plus, si l'utilisation de l'outil de rollback en parallèle est envisagée (voir § 4.8), il sera probablement nécessaire d'ajuster le paramètre *max_prepared_transaction*.

3.2.4 Test et démonstration

Il est possible de tester le bon fonctionnement des composants E-Maj installés et d'en découvrir les principales fonctionnalités en exécutant un script de démonstration. Toujours sous psql, il suffit d'exécuter le script *demo.sql* fourni avec l'extension.

```
\i <répertoire_emaj>/sql/demo.sql
```

Si aucune erreur n'est rencontrée, le script affiche ce message final :

This ends the E-Maj demo. Thank You for using E-Maj and have fun!

Après l'exécution du script, l'environnement de démonstration est laissé en l'état. On peut alors l'examiner et jouer avec. Pour le supprimer, exécuter la fonction de nettoyage qu'il a généré :

```
SELECT emaj.emaj_demo_cleanup();
```

Ceci supprime le schéma *emaj_demo_app_schema* et les deux groupes de tables '*emaj demo group 1*' et '*emaj demo group 2*'.

3.3 MISE À JOUR D'UNE VERSION EXISTANTE

3.3.1 Démarche générale

La procédure de mise à jour de la version d'E-Maj varie en fonction de la version E-Maj installée et du mode d'installation qui a été utilisé.

Pour les versions d'E-Maj antérieures à 0.10.0, il n'existe pas de procédure spécifique de mise à jour. On procèdera donc à une simple désinstallation puis réinstallation de l'extension, tel que décrit dans le chapitre §3.3.3. Cette démarche peut d'ailleurs être utilisée quelle que soit la version d'E-Maj installée. Elle présente néanmoins l'inconvénient de devoir supprimer tous les logs enregistrés, perdant ainsi toute capacité ultérieure de rollback ou d'examen des mises à jour enregistrées.

Pour les versions d'E-Maj installées 0.10.0 et suivantes, il est possible de procéder à une migration sans désinstallation. Suivant les cas, il pourra être nécessaire de procéder en une ou en plusieurs étapes :

- la migration de 0.10.0 à 0.10.1 est décrite dans le chapitre §3.3.4
- la migration de 0.10.1 à 0.11.0 est décrite dans le chapitre §3.3.5
- la migration de 0.11.0 à 0.11.1 est décrite dans le chapitre §3.3.6
- la migration de 0.11.1 à 1.0.0 est décrite dans le chapitre §3.3.7
- la migration de 1.0.0 à 1.0.1 est décrite dans le chapitre §3.3.8
- la migration de 1.0.1 à 1.0.2 est décrite dans le chapitre §3.3.9

Mais si E-Maj 0.10.0 ou 0.10.1 a été installé avec le gestionnaire d'extensions intégré (par une requête *CREATE EXTENSION*), il faudra au préalable dé-référencer l'extension, tel que décrit dans le chapitre §3.3.2.

3.3.2 Dé-référencement d'une extension E-Maj antérieure

Si E-Maj 0.10.0 ou 0.10.1 a été installé par une commande *CREATE EXTENSION* il est nécessaire de dé-référencer E-Maj du système de gestion d'extensions intégré.

Pour ce faire, il suffit d'enchaîner les 2 commandes suivantes :

```
\i <répertoire_emaj>/sql/emaj--<version_emaj>--unpackaged.sql

DROP EXTENSION emaj;
```

où <version_emaj> peut prendre les valeurs 0.10.0 ou 0.10.1 selon la version d'E-Maj installée.

A l'issue de cette opération, l'*extension emaj* n'existe plus, mais tous les composants qu'elle contenait (tables, fonctions, types,..) sont toujours présents.

3.3.3 Migration par désinstallation puis réinstallation

Pour ce type de migration, il n'est pas nécessaire d'utiliser la procédure de désinstallation complète présentée §3.4. Les *tablespaces* et les rôles peuvent notamment rester en l'état. En revanche, il peut s'avérer judicieux de sauvegarder quelques données utiles. C'est pourquoi, la démarche suivante est proposée.

3.3.3.1 Arrêt des groupes de tables

Si certains groupes de tables sont encore actifs, il faut au préalable les arrêter à l'aide de la fonction *emaj_stop_group()* (voir § 4.2.8), ou de la fonction *emaj_force_stop_group()* (voir §4.7.4) si *emaj_stop_group()* retourne une erreur..

3.3.3.2 Sauvegarde des données utilisateurs

Il peut en effet être utile de sauvegarder le contenu de la table *emaj_group_def* pour un rechargement facile après le changement de version, par exemple en la copiant sur un fichier par une commande *\copy,* ou en dupliquant la table en dehors du schéma *emaj* avec une requête SQL :

```
CREATE TABLE public.sav_group_def AS SELECT * FROM emaj.emaj_group_def;
```

De la même manière, si l'administrateur E-Maj a modifié des paramètres dans la table *emaj_param*, il peut être souhaitable d'en conserver les valeurs, avec par exemple :

```
CREATE TABLE public.sav_param AS SELECT * FROM emaj.emaj_param
WHERE param_key <> 'emaj_version';
```

3.3.3.3 Suppression et réinstallation d'E-Maj

Une fois connecté en tant que super-utilisateur, il suffit d'enchaîner les scripts de désinstallation *uninstall.sql* de la version en place puis d'installation de la nouvelle version, *emaj.sql*.

```
\i <répertoire_ancien_emaj>/sql/uninstall.sql

\i <répertoire_emaj>/sql/emaj.sql
```

3.3.3.4 Restauration des données utilisateurs

Les données sauvegardées au préalable peuvent alors être restaurées dans les tables E-Maj, par exemple avec des requêtes de type INSERT SELECT.

```
INSERT INTO emaj.emaj_group_def SELECT * FROM public.sav_group_def;

INSERT INTO emaj.emaj_param SELECT * FROM public.sav_param;
```

Une fois les données copiées, les tables ou fichiers temporaires peuvent être supprimés.

3.3.4 Migration de E-Maj 0.10.0 à 0.10.1

Si une version 0.10.0 d'E-Maj est installée, il est possible de procéder à une simple mise à jour de cette version pour passer en 0.10.1.

Cette opération peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même être actifs au moment de la migration. Ceci signifie en particulier :

- que des mises à jour peuvent être enregistrées avant puis après la migration, sans que les groupes de tables soient arrêtés,
- et donc qu'après la migration, un *rollback* à une marque posée avant cette migration est possible.

Cette migration est très rapide. Elle ne consiste qu'en un ajout ou une modification de quelques fonctions.

Pour migrer de la version 0.10.0 vers la version 0.10.1 d'E-Maj, il suffit d'exécuter le script psql *emaj-0.10.0-to-0.10.1.sql* fourni :

```
\i <répertoire_emaj>/sql/emaj-0.10.0-to-0.10.1.sql
```

Le script liste les groupes de tables qui nécessiteront une recréation pour bénéficier de toutes les améliorations liées à cette version 0.10.1. Mais la migration suivante vers E-Maj 0.11.0 procèdera implicitement à cette recréation.

A la fin de la migration le message suivant est affiché :

>>> E-Maj successfully migrated to 0.10.1

3.3.5 Migration de E-Maj 0.10.1 à 0.11.0

Si une version 0.10.1 d'E-Maj est installée, il est possible, sous certaines conditions (voir §3.3.5.1), de procéder à une simple mise à jour de cette version pour passer en 0.11.0.

Dans ce cas, cette opération peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même rester actifs au moment de la migration. Ceci signifie en particulier :

- que des mises à jour peuvent être enregistrées avant puis après la migration, sans que les groupes de tables soient arrêtés,
- et donc qu'après la migration, un *rollback* à une marque posée avant cette migration est possible.

Mais cette opération peut être longue. En effet, avec cette version 0.11.0, quelques fonctions sont créées ou modifiées, la table *emaj_mark* est modifiée, mais surtout la structure des tables de log change. Le script de migration doit donc en particulier recréer toutes ces tables de log. La durée de l'opération dépend donc directement du volume de log présent. Pour limiter cette durée, et si cela ne présente pas d'inconvénient, il peut donc être préférable de supprimer les marques les plus anciennes, voire d'arrêter les groupes de tables puis purger les logs (fonctions *emaj_stop_group()* et *emaj_reset_group()*), voire même de supprimer les groupes de tables avant la migration et de les recréer après cette migration.

Pour que les tables de log ne soient pas mises à jour par d'autres traitements durant cette migration, un verrou de type exclusif est posé sur chaque table de chaque groupe de tables actif. Cela signifie que la migration ne peut s'effectuer qu'à une période où il n'y a aucune activité sur les tables protégées par E-Maj.

3.3.5.1 Validation des conditions de migration

La migration va attribuer un nouveau numéro de séquence à chaque ligne de log, numéro maintenant unique au sein de la base de données. Pour garantir l'intégrité des données enregistrées par E-Maj, il est essentiel qu'aucun changement d'heure système passé n'empêche une remise en séquence fiable des lignes de toutes les tables de log.

Un script psql, nommé *check-0.10.1-to-0.11.0-conditions.sql*, est fourni avec la version. Il permet d'analyser l'état de l'environnement E-Maj et indique s'il est possible de procéder à la simple mise à jour de la version.

```
\i <répertoire_emaj>/sql/check-0.10.1-to-0.11.0-conditions.sql
```

Des messages de types « *warning* » ou « *notice* » peuvent être générés par l'exécution du script. Les messages de types « *notice* » sont de simple détections de désynchronisation de 2 lignes de log successives, mais qui ne sont pas gênants pour la migration. En revanche, les messages de type « *warning* » matérialisent des cas bloquants pour la migration.

La fonction exécutée restitue un message textuel indiquant le résultat de l'analyse. Si la migration est possible, le message suivant est retourné :

This E-Maj environment can be migrated into 0.11.0.

Dans le cas contraire; on obtient le message :

This E-Maj environment can NOT be migrated into 0.11.0.

Dans ce second cas, deux solutions sont possibles :
- soit supprimer des anciennes marques pour éliminer les périodes de temps qui posent problème
- soit procéder à une suppression puis recréation des groupes de tables.

Le script de migration proprement dit intègre lui aussi ces contrôles.

3.3.5.2 Mise à jour d'E-Maj

Si le test présenté au chapitre précédent indique que la migration de la version 0.10.1 d'E-Maj vers la version 0.11.0 est possible, on peut alors exécuter le script psql *emaj-0.10.1-to-0.11.0.sql* fourni :

```
\i <répertoire_emaj>/sql/emaj-0.10.1-to-0.11.0.sql
```

A la fin de la migration le message suivant est affiché :

>>> E-Maj successfully migrated to 0.11.0

3.3.6 Migration de E-Maj 0.11.0 à 0.11.1

Si une version 0.11.0 d'E-Maj est installée, il est possible de procéder à une simple mise à jour de cette version pour passer en 0.11.1.

Cette opération peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même être actifs au moment de la migration. Ceci signifie en particulier :
- que des mises à jour peuvent être enregistrées avant puis après la migration, sans que les groupes de tables soient arrêtés,
- et donc qu'après la migration, un *rollback* à une marque posée avant cette migration est possible.

Cette migration est très rapide.

Pour migrer de la version 0.11.0 vers la version 0.11.1 d'E-Maj, il suffit d'exécuter le script psql *emaj-0.11.0-to-0.11.1.sql* fourni :

```
\i <répertoire_emaj>/sql/emaj-0.11.0-to-0.11.1.sql
```

A la fin de la migration le message suivant est affiché :

>>> E-Maj successfully migrated to 0.11.1

3.3.7 Migration de E-Maj 0.11.1 à 1.0.0

Si une version 0.11.1 d'E-Maj est installée, il est possible de procéder à une simple mise à jour de cette version pour passer en 1.0.0.

Cette opération peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même être actifs au moment de la migration. Ceci signifie en particulier :

- que des mises à jour peuvent être enregistrées avant puis après la migration, sans que les groupes de tables soient arrêtés,
- et donc qu'après la migration, un *rollback* à une marque posée avant cette migration est possible.

Cette migration est très rapide.

Pour migrer de la version 0.11.1 vers la version 1.0.0 d'E-Maj, il suffit d'exécuter le script psql *emaj-0.11.1-to-1.0.0.sql* fourni :

```
\i <répertoire_emaj>/sql/emaj-0.11.1-to-1.0.0.sql
```

A la fin de la migration le message suivant est affiché :

>>> E-Maj successfully migrated to 1.0.0

3.3.8 Migration de E-Maj 1.0.0 à 1.0.1

Si une version 1.0.0 d'E-Maj est installée, il est possible de procéder à une simple mise à jour de cette version pour passer en 1.0.1.

Cette opération peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même être actifs au moment de la migration. Ceci signifie en particulier :

- que des mises à jour peuvent être enregistrées avant puis après la migration, sans que les groupes de tables soient arrêtés,
- et donc qu'après la migration, un *rollback* à une marque posée avant cette migration est possible.

Cette migration est très rapide.

Pour migrer de la version 1.0.0 vers la version 1.0.1 d'E-Maj, il suffit d'exécuter le script psql *emaj-1.0.0-to-1.0.1.sql* fourni :

```
\i <répertoire_emaj>/sql/emaj-1.0.0-to-1.0.1.sql
```

A la fin de la migration le message suivant est affiché :

>>> E-Maj successfully migrated to 1.0.1

3.3.9 Migration de E-Maj 1.0.1 à 1.0.2

Si une version 1.0.1 d'E-Maj est installée, il est possible de procéder à une simple mise à jour de cette version pour passer en 1.0.2.

Cette opération peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même être actifs au moment de la migration. Ceci signifie en particulier :

- que des mises à jour peuvent être enregistrées avant puis après la migration, sans que les groupes de tables soient arrêtés,
- et donc qu'après la migration, un *rollback* à une marque posée avant cette migration est possible.

Cette migration est très rapide.

Pour migrer de la version 1.0.1 vers la version 1.0.2 d'E-Maj, il suffit d'exécuter le script psql *emaj-1.0.1-to-1.0.2.sql* fourni :

```
\i <répertoire_emaj>/sql/emaj-1.0.1-to-1.0.2.sql
```

A la fin de la migration le message suivant est affiché :

>>> E-Maj successfully migrated to 1.0.2

3.4 DÉSINSTALLATION D'E-MAJ

Si certains groupes de tables sont encore actifs, il faut au préalable les arrêter à l'aide de la fonction *emaj_stop_group()* (voir § 4.2.8), ou de la fonction *emaj_force_stop_group()* (voir §4.7.4) si *emaj_stop_group()* retourne une erreur..

Pour désinstaller E-Maj d'une base de données, l'utilisateur doit se connecter à cette base avec psql, en tant que super-utilisateur.

Si on souhaite supprimer les rôles *emaj_adm* et *emaj_viewer*, il faut au préalable retirer les droits donnés sur ces rôles à d'éventuels autres rôles, à l'aide de requêtes SQL *REVOKE*.

```
REVOKE emaj_adm FROM <role.ou.liste.de.rôles>;
REVOKE emaj_viewer FROM <role.ou.liste.de.rôles>;
```

Si ces rôles *emaj_adm* et *emaj_viewer* possèdent des droits d'accès sur des tables ou autres objets relationnels applicatifs, il faut également supprimer ces droits au préalable à l'aide d'autres requêtes SQL *REVOKE*.

Ensuite, il suffit d'exécuter le script *uninstall.sql* fourni avec la version d'E-Maj installée.

```
\i <répertoire_emaj>/sql/uninstall.sql
```

Ce script supprime les schémas de l'extension (le schéma principal *emaj* et les éventuels schémas secondaires) et tout ce qu'ils contiennent. Il supprime également les objets créés par le script *demo.sql*, et qui n'auraient pas été déjà supprimés.

Si les rôles *emaj_adm* et *emaj_viewer* ne sont pas associés à d'autres rôles ou à d'autres bases de données du cluster et ne possèdent pas de droits sur d'autres tables, ils sont supprimés.

En revanche, s'ils existent, le tablespace *tspemaj* et les éventuels autres tablespaces créés pour supporter les tables de log ne sont PAS supprimés par le script.

4 UTILISATION D'E-MAJ

4.1 MISE EN PLACE DE LA POLITIQUE D'ACCÈS À E-MAJ

Une mauvaise utilisation d'E-Maj peut mettre en cause l'intégrité des bases de données. Aussi convient-il de n'autoriser son usage qu'à des utilisateurs qualifiés et clairement identifiés comme tels.

4.1.1 Les rôles E-Maj

Pour utiliser E-Maj, on peut se connecter en tant que super-utilisateur. Mais pour des raisons de sécurité, il est préférable de tirer profit des deux rôles créés par la procédure d'installation :

- *emaj_adm* sert de rôle d'administration ; il peut exécuter toutes les fonctions et accéder à toutes les tables d'E-Maj, en lecture comme en mise à jour,
- *emaj_viewer* sert pour des accès limités à de la consultation ; il ne peut exécuter que des fonctions de type statistique et n'accède aux tables d'E-Maj qu'en lecture.

Tous les droits attribués à *emaj_viewer* le sont aussi à *emaj_adm*.

Mais lors de leur création, ces deux rôles ne se sont pas vus attribuer de capacité de connexion (aucun mot de passe et option *NOLOGIN* spécifiés). Il est recommandé de NE PAS leur attribuer cette capacité de connexion. A la place, il suffit d'attribuer les droits qu'ils possèdent à d'autres rôles par des requêtes SQL de type *GRANT*.

4.1.2 Attribution des droits E-Maj

Pour attribuer à un rôle donné tous les droits associés à l'un des deux rôles *emaj_adm* ou *emaj_viewer*, et une fois connecté en tant que super-utilisateur pour avoir le niveau de droit suffisant, il suffit d'exécuter l'une des commandes suivantes :

```
GRANT emaj_adm TO <mon.rôle.administrateur.emaj>;
GRANT emaj_viewer TO <mon.rôle.de.consultation.emaj>;
```

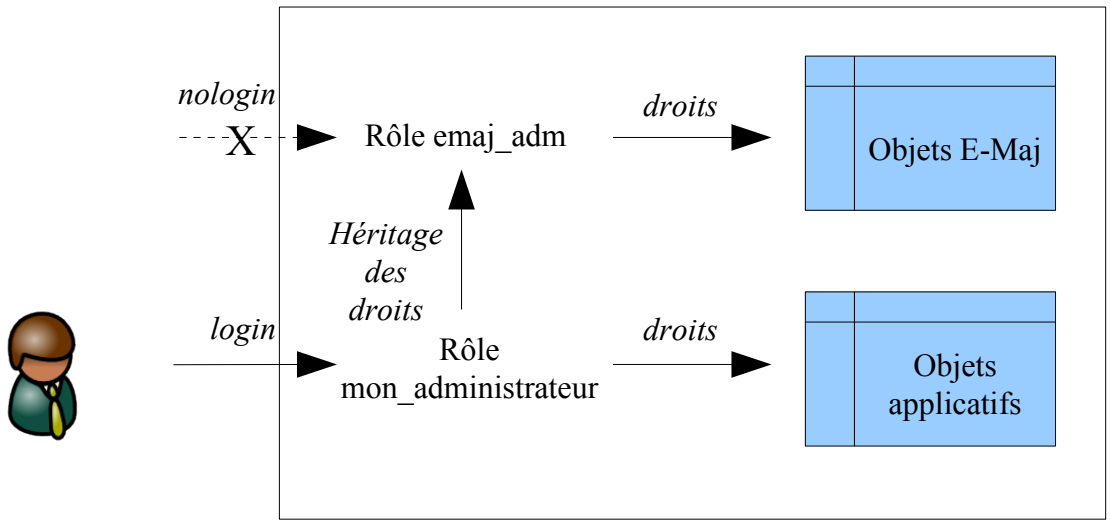
Naturellement, plusieurs rôles peuvent se voir attribuer les droits *emaj_adm* ou *emaj_viewer*.

4.1.3 Attribution des droits sur les tables et objets applicatifs

Pour qu'un administrateur E-Maj puisse également accéder à des tables ou à d'autres objets applicatifs (schémas, séquences, vues, fonctions,...), on peut attribuer aux rôles *emaj_adm* ou *emaj_viewer* des droits d'accès à ces objets. Mais il est préférable d'affecter ces droits directement et uniquement aux rôles qui héritent des droits d'*emaj_adm* ou *emaj_viewer*, en ne laissant à ces derniers que des droits sur les tables et objets E-Maj.

4.1.4 Synthèse

Le schéma ci-dessous symbolise l'attribution recommandée des droits pour un administrateur E-Maj.



Bien évidemment, ce schéma s'applique également au rôle *emaj_viewer*.

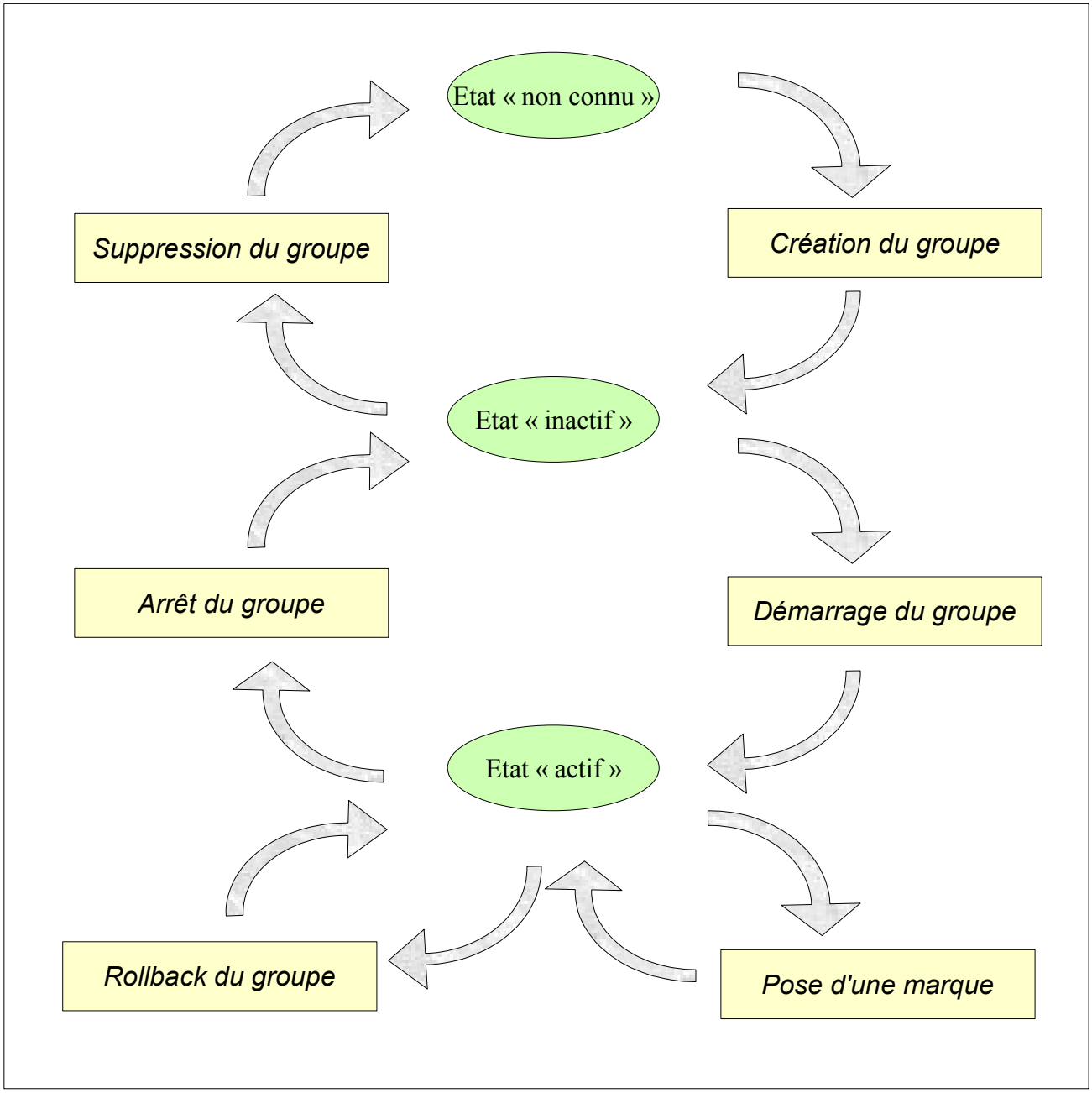
Sauf indication contraire, les opérations qui suivent vont pouvoir être exécutées indifféremment avec un rôle super-utilisateur ou un rôle du groupe *emaj_adm*.

4.2 FONCTIONS PRINCIPALES

Avant de décrire chacune des principales fonctions d'E-Maj, il est intéressant d'avoir un aperçu global de l'enchaînement typique des opérations.

4.2.1 Enchaînement des opérations

L'enchaînement des opérations possibles pour un groupe de tables peut se matérialiser par ce synoptique.



4.2.2 Définition des groupes de tables

4.2.2.1 La table emaj_group_def

Le contenu du ou des groupes de tables que l'on souhaite gérer se définit en garnissant la table *emaj.emaj_group_def*. Il faut insérer dans cette table une ligne par table ou séquence applicative à intégrer dans un groupe. Cette table *emaj.emaj_group_def* a la structure suivante :

Colonne	Type	Description
grpdef_group	TEXT	nom du groupe de tables
grpdef_schema	TEXT	nom du schéma contenant la table ou la séquence applicative
grpdef_tblseq	TEXT	nom de la table ou de la séquence applicative
grpdef_priority	INT	niveau de priorité de la table ou de la séquence dans les traitements E-Maj (optionnel)
grpdef_log_schema_suffix	TEXT	suffixe permettant de construire le nom du schéma contenant les objets E-Maj de la table (optionnel)
grpdef_log_dat_tsp	TEXT	nom du tablespace hébergeant la table de log (optionnel)
grpdef_log_idx_tsp	TEXT	nom du tablespace hébergeant l'index de la table de log (optionnel)

L'administrateur peut alimenter cette table par tout moyen usuel : verbe SQL *INSERT*, verbe SQL *COPY*, commande psql *\copy*, outil graphique, etc.

Le contenu de la table *emaj_group_def* est sensible à la casse. Les noms de schéma, de table, de séquence et de tablespace doivent correspondre à la façon dont PostgreSQL les enregistre dans son catalogue. Ces noms sont le plus souvent en minuscule. Mais si un nom est encadré par des double-guillemets dans les requêtes SQL, car contenant des majuscules ou des espaces, alors il doit être enregistré dans la table *emaj_group_def* avec ces mêmes majuscules et espaces.

Pour garantir l'intégrité des tables gérées par E-Maj, il est fondamental de porter une attention particulière à cette phase de définition des groupes de tables. Si une table était manquante, son contenu se trouverait bien sûr désynchronisé après une opération de rollback sur le groupe de tables auquel elle aurait dû appartenir. En particulier, lors de la création ou de la suppression de tables applicatives, il est important de tenir à jour le contenu de cette table *emaj_group_def*.

4.2.2.2 Les colonnes principales

Un nom de groupe de tables (colonne ***grpdef_group***) doit contenir au moins un caractère. Il peut contenir des espaces et/ou des caractères de ponctuation. Mais il est recommandé d'éviter les caractères virgule, guillemet simple ou double.

Une table ou une séquence d'un schéma donné (colonnes ***grpdef_schema*** et ***grpdef_tblseq***) ne peut pas être affectée à plusieurs groupes de tables. Toutes les tables d'un schéma n'appartiennent pas nécessairement au même groupe. Certaines peuvent appartenir à des groupes différents. D'autres peuvent n'être affectées à aucun groupe.

Toute table appartenant à un groupe de tables non créé en mode « audit_only » doit posséder une clé primaire explicite (clause *PRIMARY KEY* des *CREATE TABLE* ou *ALTER TABLE*).

Si une séquence est associée à une table applicative, il faut explicitement la déclarer dans le même groupe que sa table. Ainsi, lors d'une opération de rollback, elle sera remise dans l'état où elle se trouvait lors de la pose de la marque servant de référence au rollback.

En revanche, les tables de log et leur séquence NE doivent PAS être référencées dans un groupe de tables !

4.2.2.3 Les colonnes optionnelles

La colonne ***grpdef_priority*** est de type entier (*INTEGER*) et peut prendre la valeur NULL, Elle permet de définir un ordre de priorité dans le traitements des tables par les fonctions d'E-Maj. Ceci peut-être utile pour faciliter la pose des verrous. En effet, en posant les verrous sur les tables dans le même ordre que les accès applicatifs typiques, on peut limiter le risque de *deadlock*. Les fonctions E-Maj traitent les tables dans l'ordre croissant de *grpdef_priority*, les valeurs *NULL* étant traitées en dernier. Pour un même niveau de priorité, les tables sont traitées dans l'ordre alphabétique de nom de schéma puis de nom de table.

Pour les installations E-Maj comportant un très grand nombre de tables, il peut s'avérer pratique de répartir tous les objets créés par l'extension dans plusieurs schémas, au lieu de les concentrer dans l'unique schéma *emaj*. La colonne ***grpdef_log_schema_suffix*** sert à spécifier le nom du schéma dans lequel la table de log, la séquence de log et les fonctions de log et de rollback seront créées.

Si cette colonne *grpdef_log_schema_suffix* contient une valeur NULL ou une chaîne vide, le schéma principal *emaj* sera utilisé. Dans le cas contraire, un schéma secondaire sera utilisé. Son nom est la concaténation de 'emaj' et de la valeur de la colonne.

La création et la suppression des schémas secondaires sont gérées exclusivement par les fonctions E-Maj. Ils ne devront PAS contenir d'objets autres que ceux créés par E-Maj.

Pour les séquences, la colonne *grpdef_log_schema_suffix* doit rester NULL.

Pour optimiser les performances des installations E-Maj comportant un très grand nombre de tables, il peut s'avérer intéressant de répartir les tables de log et leur index dans plusieurs tablespaces La colonne ***grpdef_log_dat_tsp*** sert à spécifier le nom du tablespace à utiliser pour la table de log d'une table applicative. De la même manière, la colonne ***grpdef_log_idx_tsp*** sert à spécifier le nom du tablespace à utiliser pour l'index de la table de log.

Si une colonne *grpdef_log_dat_tsp* ou *grpdef_log_idx_tsp* contient une valeur NULL (valeur par défaut), le tablespace utilisé lors de la création du groupe sera *tspemaj*, s'il existe, ou le tablespace par défaut de la session courante.

Si une colonne *grpdef_log_dat_tsp* ou *grpdef_log_idx_tsp* contient une valeur non nulle, le tablespace ainsi cité devra pré-exister au moment de la création du groupe.

Pour les séquences, les colonnes *grpdef_log_dat_tsp* et *grpdef_log_idx_tsp* doivent rester NULL.

4.2.3 Création d'un groupe de tables

Une fois la constitution d'un groupe de tables définie, E-Maj peut créer ce groupe. Pour ce faire, il suffit d'exécuter la requête SQL suivante :

```
SELECT emaj.emaj_create_group('<nom.du.groupe>', <est.rollbackable>);
```

ou encore, dans sa forme abrégée :

```
SELECT emaj.emaj_create_group('<nom.du.groupe>');
```

Le second paramètre, de type booléen, indique si le groupe est de type « *rollbackable* » avec la valeur vrai ou de type « *audit_only* » avec la valeur fausse. Si le second paramètre n'est pas fourni, le groupe à créer est considéré comme étant de type « *rollbackable* ».

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Pour chaque table du groupe, cette fonction crée la table de log associée, la fonction et le trigger de log, ainsi que le trigger bloquant les exécutions de requêtes SQL *TRUNCATE* (à partir de PostgreSQL 8.4). Si le groupe est de type « rollbackable », elle crée aussi la fonction de rollback.

La fonction crée également les éventuels schémas E-Maj secondaires nécessaires.

En revanche, si des tablespaces spécifiques pour les tables de log ou pour leurs index, sont référencés, ceux-ci doivent déjà exister avant l'exécution de la fonction.

La fonction *emaj_create_group()* contrôle également l'existence de « triggers applicatifs » impliquant les tables du groupe. Si un trigger existe sur une table du groupe, un message est retourné incitant l'utilisateur à vérifier que ce trigger ne fait pas de mises à jour sur des tables n'appartenant pas au groupe.

Si une séquence du groupe est associée à une colonne de type *SERIAL* ou *BIGSERIAL* et que sa table d'appartenance ne fait pas partie du groupe, la fonction génère également un message de type *WARNING*.

Toutes les actions enchaînées par la fonction *emaj_create_group() s*ont exécutées au sein d'une unique transaction. En conséquence, si une erreur survient durant l'opération, toutes les tables, fonctions et triggers déjà créés par la fonction sont annulées.

En enregistrant la composition du groupe dans la table interne *emaj_relation*, la fonction *emaj_create_group()* en fige sa définition pour les autres fonctions E-Maj, même si le contenu de la table *emaj_group_def* est modifié entre temps.

Un groupe créé peut être modifié par la fonction *emaj_alter_group()* (voir §4.2.10) ou supprimé par la fonction *emaj_drop_group()* (voir §4.2.9).

4.2.4 Démarrage d'un groupe de tables

Démarrer un groupe de table consiste à activer l'enregistrement des mises à jour des tables du groupe. Pour ce faire, il faut exécuter la commande :

```
SELECT emaj.emaj_start_group('<nom.du.groupe>', '<nom.de.marque>' ,
<anciens.logs.à.effacer?>);
```

ou encore, dans sa forme abrégée :

```
SELECT emaj.emaj_start_group('<nom.du.groupe>', '<nom.de.marque>');
```

Le groupe de tables doit être au préalable en état arrêté (*IDLE*).

Un nom de marque doit être spécifié. Elle constituera la première marque sur laquelle on pourra effectuer un rollback.

Le nom de la marque peut contenir un caractère générique '%'. Ce caractère est alors remplacé par l'heure de début de la transaction courante, au format « hh.mn.ss.mmm »,

Si le paramètre représentant la marque est vide ou *NULL*, un nom est automatiquement généré : « *MARK_%* », où le caractère '%' représente l'heure de début de la transaction courante.

Le paramètre <*anciens.logs.à.effacer*> est un booléen optionnel. Par défaut sa valeur est égal à vrai (*true*), ce qui signifie que les tables de log du groupe de tables sont purgées de toutes anciennes données avant l'activation des triggers de log. Si le paramètre est explicitement positionné à « faux » (*false*), les anciens enregistrements sont conservés dans les tables de log. De la même manière, les anciennes marques sont conservées, même si ces dernières ne sont alors plus utilisables pour un éventuel rollback (des mises à jour ont pu être effectuées sans être tracées alors que le groupe de tables était arrêté).

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Pour être certain qu'aucune transaction impliquant les tables du groupe n'est en cours, la fonction *emaj_start_group()* pose explicitement un verrou de type *ACCESS EXCLUSIVE* sur chacune des tables du groupe. Si des transactions accédant à ces tables sont en cours, ceci peut se traduire par la survenue d'une étreinte fatale (*deadlock*). Si la résolution de l'étreinte fatale impacte la fonction E-Maj, le *deadlock* est intercepté et la pose de verrou est automatiquement réitérée, avec un maximum de 5 tentatives.

La fonction procède également à la purge des événements les plus anciens de la table technique *emaj_hist* (voir §5.3).

A l'issue du démarrage d'un groupe, celui-ci devient actif en prenant l'état « *LOGGING* ».

4.2.5 Pose d'une marque intermédiaire

Lorsque toutes les tables et séquences d'un groupe sont jugées dans un état stable pouvant servir de référence pour un éventuel rollback, une marque peut être posée. Ceci s'effectue par la requête SQL suivante :

```
SELECT emaj.emaj_set_mark_group('<nom.du.groupe>', '<nom.de.marque>');
```

Le groupe de tables doit être en état démarré (*LOGGING*).

Une marque de même nom ne doit pas déjà exister pour le groupe de tables.

Le nom de la marque peut contenir un caractère générique '%'. Ce caractère est alors remplacé par l'heure de début de la transaction courante, au format « hh.mn.ss.mmm »,

Si le paramètre représentant la marque est vide ou *NULL*, un nom est automatiquement généré : « *MARK_%* », où le caractère '%' représente l'heure de début de la transaction courante.

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

La fonction *emaj_set_mark_group()* enregistre l'identité de la nouvelle marque, avec l'état des séquences applicatives appartenant au groupe, ainsi que l'état des séquences associées aux tables de log. Les séquences applicatives sont traitées en premier, pour

enregistrer leur état au plus près du début de la transaction, ces séquences ne pouvant pas être protégées des mises à jour par des verrous.

Notez qu'il est possible d'enregistrer deux marques consécutives sans que des mises à jour de tables aient été enregistrées entre ces deux marques.

La fonction *emaj_set_mark_group()* pose des verrous de type *ROW EXCLUSIVE* sur chaque table du groupe. Ceci permet de s'assurer qu'aucune transaction ayant déjà fait des mises à jour sur une table du groupe n'est en cours. Néanmoins, ceci ne garantit pas qu'une transaction ayant lu une ou plusieurs tables avant la pose de la marque, fasse des mises à jours après la pose de la marque. Dans ce cas, ces mises à jours effectuées après la pose de la marque seraient candidates à un éventuel rollback sur cette marque.

4.2.6 Rollback simple d'un groupe de tables

S'il est nécessaire de remettre les tables et séquences d'un groupe dans l'état dans lequel elles se trouvaient lors de la prise d'une marque, il faut procéder à un rollback. Pour un rollback simple (« *unlogged* »), il suffit d'exécuter la requête SQL suivante :

```
SELECT emaj.emaj_rollback_group('<nom.du.groupe>', '<nom.de.marque>');
```

Le groupe de tables doit être en état démarré (*LOGGING*) et la marque indiquée doit être toujours « active », c'est à dire qu'elle ne doit pas être marquée comme logiquement supprimée (*DELETED*).

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque pour indiquer la dernière marque posée.

La fonction retourne le nombre de tables et de séquences **effectivement** modifiées par l'opération de rollback.

Pour être certain qu'aucune transaction concurrente ne mette à jour une table du groupe pendant toute la durée du rollback, la fonction *emaj_rollback_group()* pose explicitement un verrou de type *EXCLUSIVE* sur chacune des tables du groupe. Si des transactions accédant à ces tables en mise à jour sont en cours, ceci peut se traduire par la survenue d'une étreinte fatale (*deadlock*). Si la résolution de l'étreinte fatale impacte la fonction E-Maj, le *deadlock* est intercepté et la pose de verrou est automatiquement réitérée, avec un maximum de 5 tentatives. En revanche, les tables du groupe continuent à être accessibles en lecture pendant l'opération.

Si des tables du groupe à « rollbacker » possèdent des triggers, il peut être nécessaire de les désactiver avant le rollback et de les réactiver à l'issue de l'opération (voir §5.8.3).

Si une table impactée par le rollback possède une clé étrangère (*foreign key*) ou est référencée dans une clé étrangère appartenant à une autre table, alors la présence de cette clé étrangère est prise en compte par l'opération de rollback. Si le contrôle des clés

créées ou modifiées par le rollback ne peut être différé en fin d'opération (contrainte non déclarée *DEFERRABLE*), alors cette clé étrangère est supprimée en début de rollback puis recréée en fin de rollback.

A l'issue de l'opération de rollback, se trouvent effacées :

- les données des tables de log qui concernent les mises à jour annulées,
- toutes les marques postérieures à la marque référencée dans la commande de rollback.

Il est alors possible de poursuivre les traitements de mises à jour, de poser ensuite d'autres marques et éventuellement de procéder à un nouveau rollback sur une marque quelconque.

Par nature, le repositionnement des séquences n'est pas « annulable » en cas de rollback de la transaction incluant l'exécution de la fonction *emaj_rollback_group()*. Pour cette raison, le traitement des séquences applicatives est toujours effectué après celui des tables. Néanmoins, même si le temps de traitement des séquences est très court, il n'est pas impossible qu'un problème surgisse lors de cette dernière phase. La relance de la fonction *emaj_rollback_group()* mènera à bien l'opération de manière fiable. Mais si cette fonction n'était pas ré-exécutée immédiatement, il y aurait risque que certaines séquences aient été repositionnées, contrairement aux tables et à d'autres séquences.

4.2.7 Rollback annulable d'un groupe de tables

Un autre fonction permet d'exécuter un rollback de type « *logged* », Dans ce cas, les triggers de log sur les tables applicatives ne sont pas désactivées durant le rollback, de sorte que durant le rollback les mises à jours de tables appliquées sont elles-mêmes enregistrées dans les tables de log. Ainsi, il est ensuite possible d'annuler le rollback ou, en quelque sorte, de « rollbacker le rollback ».

Pour exécuter un « *logged rollback* » sur un groupe de tables, il suffit d'exécuter la requête SQL suivante :

```
SELECT emaj.emaj_logged_rollback_group('<nom.du.groupe>',
'<nom.de.marque>');
```

Les règles d'utilisation sont les mêmes que pour la fonction *emaj_rollback_group()*,

Le groupe de tables doit être en état démarré (*LOGGING*) et la marque indiquée doit être toujours « active », c'est à dire qu'elle ne doit pas être marquée comme logiquement supprimée (*DELETED*).

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque pour indiquer la dernière marque posée.

La fonction retourne le nombre de tables et de séquences **effectivement** modifiées par l'opération de rollback.

Pour être certain qu'aucune transaction concurrente ne mette à jour une table du groupe pendant toute la durée du rollback, la fonction *emaj_rollback_group()* pose explicitement un verrou de type *EXCLUSIVE* sur chacune des tables du groupe. Si des transactions accédant à ces tables en mise à jour sont en cours, ceci peut se traduire par la survenue d'une étreinte fatale (*deadlock*). Si la résolution de l'étreinte fatale impacte la fonction E-Maj, le *deadlock* est intercepté et la pose de verrou est automatiquement réitérée, avec un maximum de 5 tentatives. En revanche, les tables du groupe continuent à être accessibles en lecture pendant l'opération.

Si des tables du groupe à rollbacker possèdent des triggers, il peut être nécessaire de les désactiver avant le rollback et de les réactiver à l'issue de l'opération (voir §15.8.3).

Si une table impactée par le rollback possède une clé étrangère (*foreign key*) ou est référencée dans une clé étrangère appartenant à une autre table, alors la présence de cette clé étrangère est prise en compte par l'opération de rollback. Si le contrôle des clés créées ou modifiées par le rollback ne peut être différé en fin d'opération (contrainte non déclarée *DIFERRABLE*), alors cette clé étrangère est supprimée en début de rollback puis recréée en fin de rollback.

Contrairement à la fonction *emaj_rollback_group()*, à l'issue de l'opération de rollback, les données des tables de log qui concernent les mises à jour annulées, ainsi que les éventuelles marques postérieures à la marque référencée dans la commande de rollback sont conservées.

De plus, en début et en fin d'opération, la fonction pose automatiquement sur le groupe deux marques, nommées :

- 'RLBK_<marque.du.rollback>_<heure_du_rollback>_START'
- 'RLBK_<marque.du.rollback>_<heure_du_rollback>_DONE'

où <heure_du_rollback> représente l'heure de début de la transaction effectuant le rollback, exprimée sous la forme « heures.minutes.secondes.millisecondes ».

A l'issue du rollback, il est possible de poursuivre les traitements de mises à jour, de poser d'autres marques et éventuellement de procéder à un nouveau rollback sur une marque quelconque, y compris l'a marque automatiquement posée en début de rollback, pour annuler ce dernier, ou encore une ancienne marque postérieure à la marque utilisée pour le rollback.

Des rollbacks de différents types (*logged* / *unlogged*) peuvent être exécutés en séquence.

En guise d'exemple, on peut ainsi procéder à l'enchaînement suivant :

Pose de la marque M1
…
Pose de la marque M2
…

Logged rollback à M1
générant les marques RLBK_M1_<heure>_STRT,
puis RLBK_M1_<heure>_DONE
…
Rollback à RLBK_M1_<heure>_DONE
(pour annuler le traitement d'après rollback)
…
Rollback à RLBK_M1_<heure>_STRT
(pour finalement annuler le premier rollback)

4.2.8 Arrêt d'un groupe de tables

Lorsqu'on souhaite arrêter l'enregistrement des mises à jour des tables d'un groupe, il est possible de désactiver le log par la commande SQL :

```
SELECT emaj.emaj_stop_group('<nom.du.groupe>', '<nom.de.marque'>);
```

ou encore, dans sa forme abrégée :

```
SELECT emaj.emaj_stop_group('<nom.du.groupe>');
```

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

La fonction pose automatiquement une marque correspondant à la fin de l'enregistrement. Si un nom de marque est fournie, elle utilise ce nom. Sinon, elle prend par défaut le nom de marque :

STOP_<heure_d'arrêt>

où <heure_d'arrêt> est exprimée sous la forme « heures.minutes.secondes.millisecondes ».

Mais si le nom de marque fourni est NULL ou es une chaîne vide, la marque prend le nom :

MARK_<heure_d'arrêt>

L'arrêt d'un groupe de table désactive simplement les triggers de log des tables applicatives du groupe. La pose de verrous qu'entraine cette opération peut se traduire par la survenue d'une étreinte fatale (*deadlock*). Si la résolution de l'étreinte fatale impacte la fonction E-Maj, le *deadlock* est intercepté et la pose de verrou est automatiquement réitérée, avec un maximum de 5 tentatives.

En complément, la fonction *emaj_stop_group()* passe le statut des marques à l'état « supprimé » (*DELETED*). Il n'est dès lors plus possible d'exécuter une commande de rollback, même si aucune mise à jour n'est intervenue sur les tables entre l'exécution des deux fonctions *emaj_stop_group()* et *emaj_rollback_group()*.

Pour autant, le contenu des tables de log et des tables internes d'E-Maj peut encore être visualisé.

A l'issue de l'arrêt d'un groupe, celui-ci redevient inactif en prenant l'état « *IDLE* ».

Exécuter la fonction *emaj_stop_group()* sur un groupe de tables déjà arrêté ne génère pas d'erreur. Seul un message d'avertissement est retourné.

4.2.9 Suppression d'un groupe de tables

Pour supprimer un groupe de tables créé au préalable par la fonction *emaj_create_group()*, il faut que le groupe de tables à supprimer soit déjà arrêté. Si ce n'est pas le cas, il faut utiliser la fonction *emaj_stop_group()* (voir § 4.2.8).

Ensuite, il suffit d'exécuter la commande SQL :

```
SELECT emaj.emaj_drop_group('<nom.du.groupe>');
```

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Pour ce groupe de tables, la fonction *emaj_drop_group()* supprime tous les objets qui ont été créés par la fonction *emaj_create_group()* : tables de log, fonctions de log et de rollback, triggers de log.

Les éventuels schémas E-Maj secondaires qui deviennent inutilisés sont également supprimés.

La pose de verrous qu'entraine cette opération peut se traduire par la survenue d'une étreinte fatale (*deadlock*). Si la résolution de l'étreinte fatale impacte la fonction E-Maj, le *deadlock* est intercepté et la pose de verrou est automatiquement réitérée, avec un maximum de 5 tentatives.

4.2.10 Modification d'un groupe de tables

Deux types d'événements peuvent rendre nécessaire la modification d'un groupe de tables :

- la composition du groupe de tables change, avec l'ajout ou la suppression de tables ou de séquence dans le groupe, ou avec le changement d'un des paramètres liés à une table (priorité, schéma de log ou tablespace),
- une ou plusieurs tables applicatives appartenant au groupe de tables voient leur structure évoluer (ajout ou suppression de colonnes, changement de type de colonne), ceci ayant un impact sur la structure des tables de log associées.

Dans les deux cas, la démarche à suivre est la suivante :

- arrêter le groupe s'il est dans un état actif, avec la fonction *emaj_stop_group()*,
- adapter le contenu de la table *emaj_group_def* et/ou modifier la structure des tables applicatives pour refléter l'évolution souhaitée,
- supprimer puis recréer le groupe avec les fonctions *emaj_drop_group()* et *emaj_create_group()*.

Mais l'enchaînement des deux fonctions *emaj_drop_group()* et *emaj_create_group()* peut être remplacé par l'exécution de la fonction *emaj_alter_group()*, avec une requête SQL du type :

```
SELECT emaj.emaj_alter_group('<nom.du.groupe>');
```

La fonction retourne le nombre de tables et de séquences dorénavant contenues dans le groupe de tables.

La fonction *emaj_alter_group()* recrée également les objets E-Maj qui pourraient manquer (table de log, fonction, …).

La fonction supprime et/ou crée les schémas de log secondaires, en fonction des besoins.

A l'issue de la modification d'un groupe, celui-ci garde son état « *IDLE* » mais le contenu de ses tables de log est purgé.

Le caractère « rollbackable » ou « audit_only » du groupe de tables ne peut être modifié par cette commande. Pour changer cette caractéristique, il faut supprimer puis recréer le groupe de tables, en utilisant respectivement les fonctions *emaj_drop_group()* et *emaj_create_group()*.

Toutes les actions enchaînées par la fonction *emaj_alter_group() s*ont exécutées au sein d'une unique transaction. En conséquence, si une erreur survient durant l'opération, le groupe de tables se retrouve dans son état initial.

Dans la plupart des cas, l'exécution de la fonction *emaj_alter_group()* est nettement plus rapide que l'enchaînement des deux fonctions *emaj_drop_group()* et *emaj_create_group()*.

Il est possible d'anticiper la mise à jour de la table *emaj_group_def*, alors que le groupe de tables est encore actif. Cette mise à jour ne prendra bien sûr effet qu'à l'issue de l'exécution de la fonction *emaj_alter_group()*.

En cas de déphasage entre la structure des tables applicatives et celle des tables de log, E-Maj génère une erreur lors du démarrage du groupe, de la pose d'une marque ou d'une demande de rollback.

4.3 FONCTIONS MULTI-GROUPES

4.3.1 Généralités

Pour pouvoir synchroniser les opérations courantes de démarrage, arrêt, pose de marque et rollback entre plusieurs groupes de tables, les fonctions usuelles associées disposent de fonctions jumelles permettant de traiter plusieurs groupes de tables en un seul appel.

Les avantages qui en résultent sont :

- ✔ de pouvoir traiter tous les groupes de tables dans une seule transaction,
- ✔ d'assurer un verrouillage de toutes les tables à traiter en début d'opération, et ainsi minimiser les risques d'étreintes fatales.

4.3.2 Liste des fonctions multi-groupes

Les fonctions suivantes traitent plusieurs groupes :

- *emaj.emaj_start_groups(<tableau.de.groupes>,<marque.de.début>[,<reset.log>])* démarre plusieurs groupes,
- *emaj.emaj_stop_groups(<tableau.de.groupes> [,<marque>])* arrête plusieurs groupes,
- *emaj.emaj_set_mark_groups(<tableau.de.groupes>,<marque>)*, pose une marque sur plusieurs groupes,
- *emaj.emaj_rollback_groups(<tableau.de.groupes>,<marque>)* effectue un rollback simple sur plusieurs groupes,
- *emaj.emaj_logged_rollback_groups(<tableau.de.groupes>,<marque>)* effectue un rollback annulable sur plusieurs groupes.

4.3.3 Syntaxes pour exprimer un tableau de groupes

Le paramètre <tableau de groupes> passé aux fonctions multi-groupes est de type SQL TEXT[ ], c'est à dire un tableau de données de type TEXT.

Conformément au langage SQL, il existe deux syntaxes possibles pour saisir un tableau de groupes, utilisant soit les accolades { }, soit la fonction ARRAY.

Lorsqu'on utilise les caractères {}, la liste complète est entre simples guillemets, puis les accolades encadrent la liste des éléments séparés par une virgule, chaque élément étant délimité par des doubles guillemets. Par exemple dans notre cas, nous pouvons écrire :

```
' { "groupe 1" , "groupe 2" , "groupe 3" } '
```

La fonction SQL ARRAY permet de construire un tableau de données. La liste des valeurs est entre crochets et les littéraux sont séparés par une virgule. Par exemple dans notre cas, nous pouvons écrire :

```
ARRAY [ 'groupe 1' , 'groupe 2' , 'groupe 3' ]
```

Ces deux syntaxes sont équivalentes, et le choix de l'une ou de l'autre est à l'appréciation de chacun.

4.3.4 Autres considérations

L'ordre dans lequel les groupes sont listés n'a pas d'importance. L'ordre de traitement des tables dans les opérations E-Maj dépend du niveau de priorité associé à chaque table, et pour les tables de même priorité de l'ordre alphabétique de nom de schéma et nom de table, tous groupes confondus.

Il est possible d'appeler une fonction multi-groupes pour traiter une liste … d'un seul groupe, voire une liste vide. Ceci peut permettre une construction ensembliste de la liste (par exemple avec la fonction *array_agg()* disponible à partir de la version de PostgreSQL 8.4).

Les listes de groupes de tables peuvent contenir des doublons, des valeurs NULL ou des chaînes vides. Ces valeurs NULL et ces chaînes vides sont simplement ignorées. Si un nom de groupe de tables est présent plusieurs fois, une seule occurrence du nom est retenue. Dans tous ces cas, et dans le cas où la liste est vide, un message d'avertissement est généré.

Le formalisme et l'usage des autres paramètres éventuels des fonctions est strictement le même que pour les fonctions jumelles mono-groupes.

Néanmoins, une condition supplémentaire existe pour les fonctions de rollbacks, La marque indiquée doit strictement correspondre à un même moment dans le temps pour chacun des groupes. En d'autres termes, cette marque doit avoir été posée par l'appel d'une même fonction *emaj_set_mark_groups()*.

4.4 FONCTIONS DE GESTION DES MARQUES

4.4.1 Commentaires sur les marques

Il est possible de positionner un commentaire sur une marque quelconque. Pour se faire, il suffit d'exécuter la requête suivante :

```
SELECT emaj.emaj_comment_mark_group('<nom.du.groupe>',
'<nom.de.marque>', '<commentaire>');
```

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque à commenter pour indiquer la dernière marque posée.

La fonction ne retourne aucune donnée.

Pour modifier un commentaire, il suffit d'exécuter à nouveau la fonction pour le même groupe de tables et la même marque, avec le nouveau commentaire.

Pour supprimer un commentaire, il suffit d'exécuter la fonction avec une valeur NULL pour le paramètre commentaire.

Les commentaires sont stockés dans la colonne *mark_comment* de la table *emaj.emaj_mark* qui décrit les marques.

Les commentaires sont surtout intéressants avec l'utilisation du plugin E-Maj pour phpPgAdmin (voir §6). En effet, ce dernier les affiche systématiquement dans le tableau des marques d'un groupe.

4.4.2 Recherche de marque

La fonction *emaj_get_previous_mark_group()* permet de connaître, pour un groupe de tables, le nom de la dernière marque qui précède soit une date et une heure donnée, soit une autre marque.

```
SELECT emaj.emaj_get_previous_mark group('<nom.du.groupe>',
'<date.et.heure>');
```

ou

```
SELECT emaj.emaj_get_previous_mark group('<nom.du.groupe>', '<marque>');
```

Dans la première forme, la date et l'heure doivent être exprimées sous la forme d'un *TIMESTAMPTZ, par exemple le littéral '2011/06/30 12:00:00 +02*'. Si l'heure fournie est strictement égale à l'heure d'une marque existante, la marque retournée sera la marque précédente.

Dans la seconde forme, le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque pour indiquer la dernière marque posée.

Cette fonction est particulièrement utile quand elle est utilisée conjointement avec la fonction *emaj_delete_before_mark_group()*. Ainsi par exemple, pour supprimer toutes les marques (et les logs associés) posées depuis plus de 24 heures, on peut exécuter la requête :

```
SELECT emaj.emaj_delete_before_mark_group('<groupe>',
emaj.emaj_get_previous_mark_group('<groupe>', current_timestamp - '1
DAY'::INTERVAL));
```

4.4.3 Renommage d'une marque

Une marque précédemment posée par l'une des fonctions *emaj_create_group()* ou *emaj_set_mark_group()* peut être renommée avec la commande SQL :

```
SELECT emaj.emaj_rename_mark_group('<nom.du.groupe>',
'<nom.de.marque>', '<nouveau.nom.de.marque>');
```

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque à renommer pour indiquer la dernière marque posée.

La fonction ne retourne aucune donnée.

Une marque portant le nouveau nom souhaité ne doit pas déjà exister pour le groupe de tables.

4.4.4 Suppression d'une marque

Une marque peut également être supprimée par l'intermédiaire de la commande SQL :

```
SELECT emaj.emaj_delete_mark_group('<nom.du.groupe>',
'<nom.de.marque>');
```

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque pour indiquer la dernière marque posée.

La fonction retourne la valeur 1, c'est à dire le nombre de marques effectivement supprimées.

Pour qu'il reste au moins une marque après l'exécution de la fonction, la suppression d'une marque n'est possible que s'il y a au moins 2 marques pour le groupe de tables concerné.

Si la marque supprimée est la première marque pour le groupe, les lignes devenues inutiles dans les tables de log sont supprimées.

4.4.5 Suppression des marques les plus anciennes

Pour facilement supprimer en une seule opération toutes les marques d'un groupe de tables antérieures à une marque donnée, on peut exécuter la requête :

```
SELECT emaj.emaj_delete_before_mark_group('<nom.du.groupe>',
'<nom.de.marque>');
```

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque pour indiquer la dernière marque posée.

La fonction supprime les marques antérieures à la marque spécifiée, cette dernière devenant la nouvelle première marque. Elle supprime également des tables de log toutes les données concernant les mises à jour de tables applicative antérieures à cette marque.

La fonction procède également à la purge des événements les plus anciens de la table technique *emaj_hist* (voir §5.3).

Cette fonction permet ainsi d'utiliser E-Maj sur de longues périodes sans avoir à arrêter et redémarrer les groupes, tout en limitant l'espace disque utilisé pour le log.

Néanmoins, comme cette suppression de lignes dans les tables de log ne peut utiliser de verbe SQL *TRUNCATE*, la durée d'exécution de la fonction *emaj_delete_before_mark_group()* peut être plus longue qu'un simple arrêt et relance de groupe. En contrepartie, elle ne nécessite pas de pose de verrou sur les tables du groupe concerné. Son exécution peut donc se poursuivre alors que d'autres traitements mettent à jour les tables applicatives. Seules d'autres actions E-Maj sur le même groupe de tables, comme la pose d'une nouvelle marque, devront attendre la fin de l'exécution d'une fonction *emaj_delete_before_mark_group()*.

4.5 FONCTIONS STATISTIQUES

Deux fonctions permettent d'obtenir des statistiques sur le contenu des tables de log :

- *emaj_log_stat_group()* permet d'avoir rapidement une vision du nombre de mises à jour enregistrées entre deux marques, ou depuis une marque, pour chaque table d'un groupe,
- *emaj_detailed_log_stat_group()* permet d'avoir, pour un groupe de tables, une vision détaillée du nombre de mises à jour enregistrées entre deux marques, ou depuis une marque, par table, type de verbe (INSERT/UPDATE/DELETE) et rôle de connexion.

En complément, E-Maj fournit une fonction, *emaj_estimate_rollback_duration(),* qui permet d'estimer la durée que prendrait un éventuel rollback d'un groupe à une marque donné.

Enfin, une fonction, *emaj_get_previous_mark_group()*, retourne pour un groupe le nom de la marque qui précède une date et une heure donnée.

Toutes ces fonctions statistiques sont utilisables par tous les rôles E-Maj : *emaj_adm* et *emaj_viewer*.

4.5.1 Statistiques générales sur les logs

On peut obtenir les statistiques globales complètes à l'aide de la requête SQL :

```
SELECT * FROM emaj.emaj_log_stat_group('<nom.du.groupe>',
'<marque.début>', '<marque.fin>');
```

La fonction retourne un ensemble de lignes, de type *emaj.emaj_log_stat_type* et comportant les colonnes suivantes :

- *stat_group* : nom du groupe de tables (type TEXT),
- *stat_schema* : nom du schéma (type TEXT),
- *stat_table* : nom de table (type TEXT),
- *stat_rows* : nombre de modifications de lignes enregistrées dans la table de log associée à la table (type BIGINT)

Une valeur NULL ou une chaine vide (''), fournie comme marque de début, représente la plus ancienne marque accessible.

Une valeur NULL fournie comme marque de fin représente la situation courante.

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque. Il représente alors la dernière marque posée.

La fonction retourne une ligne par table, même si aucune mise à jour n'est enregistrée pour la table entre les deux marques. Dans ce cas, la colonne *stat_rows* contient la valeur 0.

Il est possible aisément d'exécuter des requêtes plus précises sur ces statistiques. Ainsi par exemple, après exécution du script de test *test-emaj-2.sql*, on peut obtenir le nombre de mises à jour par schéma applicatif avec une requête du type :

```
postgres=# SELECT stat_schema, sum(stat_rows)
FROM emaj.emaj_log_stat_group('myAppl1', NULL, NULL)
GROUP BY stat_schema;
 stat_schema | sum
-------------+-----
 myschema    |  41
(1 row)
```

L'obtention de ces statistiques ne nécessite pas le parcours des tables de log. Elles sont donc restituées rapidement.

Mais, les valeurs retournées peuvent être approximatives (en fait surestimées). C'est en particulier le cas si, entre les deux marques citées, des transactions ont mis à jour des tables avant d'être annulées.

4.5.2 Statistiques détaillées sur les logs

Le parcours des tables de log permet d'obtenir des informations plus détaillées, au prix d'un temps de réponse plus long. Ainsi, on peut obtenir les statistiques détaillées complètes à l'aide de la requête SQL :

```
SELECT * FROM emaj.emaj_detailed_log_stat_group('<nom.du.groupe>',
'<marque.début>', '<marque.fin>');
```

La fonction retourne un ensemble de lignes, de type *emaj.emaj_detailed_log_stat_type* et comportant les colonnes suivantes :

- *stat_group* : nom du groupe de tables (type TEXT),
- *stat_schema* : nom du schéma (type TEXT),
- *stat_table* : nom de table (type TEXT),
- *stat_role* : rôle de connexion (type VARCHAR(32)),
- *stat_verb* : verbe SQL à l'origine de la mise à jour (type VARCHAR(6), avec les valeurs *INSERT* / *UPDATE* / *DELETE*),
- *stat_rows* : nombre de modifications de lignes enregistrées dans la table de log associée à la table (type *BIGINT*)

Une valeur NULL ou une chaine vide ('), fournie comme marque de début représente la plus ancienne marque accessible.

Une valeur NULL fournie comme marque de fin représente la situation courante.

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque. Il représente alors la dernière marque posée.

Contrairement à la fonction *emaj_log_stat_group*, *emaj_detailed_log_stat_group* ne retourne aucune ligne pour les tables sans mise à jour enregistrée sur l'intervalle de marques demandées. La colonne *stat_rows* ne contient donc jamais de valeur 0.

Il est possible aisément d'exécuter des requêtes plus précises sur ces statistiques. Ainsi par exemple, après exécution du script de test *test-emaj-2.sql*, on peut obtenir le nombre de mises à jour pour une table donnée, ici mytbl1, par type de verbe exécuté, avec une requête du type :

```
postgres=# SELECT stat_table, stat_verb, stat_rows
FROM emaj.emaj_detailed_log_stat_group('myAppl1', NULL, NULL)
WHERE stat_table='mytbl1';
 stat_table | stat_verb | stat_rows
------------+-----------+-----------
 mytbl1     | DELETE    |         1
 mytbl1     | INSERT    |         6
 mytbl1     | UPDATE    |         2
(3 rows)
```

4.5.3 Estimation de la durée d'un rollback

La fonction *emaj_estimate_rollback_duration()* permet d'obtenir une estimation de la durée que prendrait le rollback d'un group à une marque donnée. Elle peut être appelée de la façon suivante :

```
SELECT emaj.emaj_estimate_rollback_duration('<nom.du.groupe>',
'<nom.de.marque>');
```

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme nom de marque. Il représente alors la dernière marque posée.

La fonction retourne un donnée de type *INTERVAL*.

Le groupe de tables doit être en état démarré (*LOGGING*) et la marque indiquée doit être utilisable pour un rollback, c'est à dire qu'elle ne doit pas être marquée comme logiquement supprimée (*DELETED*).

L'estimation de cette durée n'est qu'approximative. Elle s'appuie sur :

- le nombre de lignes à traiter dans les tables de logs, tel que le retourne la fonction *emaj_log_stat_group()*,
- des relevés de temps issus d'opérations de rollback précédentes pour les mêmes tables
- 5 paramètres génériques (voir § 5.1) qui sont utilisés comme valeurs par défaut, lorsqu'aucune statistique n'a été enregistrée pour les tables à traiter.

Compte tenu de la répartition très variable entre les verbes INSERT, UPDATE et DELETE enregistrés dans les logs, et des conditions non moins variables de charge des serveurs lors des opérations de rollback, la précision du résultat restitué est faible. L'ordre de grandeur obtenu peut néanmoins donner une bonne indication sur la capacité de traiter un rollback lorsque le temps imparti est contraint.

Sans statistique sur les rollbacks précédents, si les résultats obtenus sont de qualité médiocre, il est possible d'ajuster les paramètres listés au chapitre 5.1. Il est également possible de modifier manuellement le contenu de la table *emaj.emaj_rlbk_stat* qui conserve la durée des rollbacks précédents, en supprimant par exemple les lignes correspondant à des rollbacks effectués dans des conditions de charge inhabituelles.

4.6 FONCTIONS D'EXTRACTION DE DONNÉES

Trois fonctions permettent d'extraire des données de l'infrastructure E-Maj et de les stocker sur des fichiers externes.

4.6.1 Vidage des tables d'un groupe

Il peut s'avérer utile de prendre des images de toutes les tables et séquences appartenant à un groupe, afin de pouvoir en observer le contenu ou les comparer. Une fonction permet d'obtenir le vidage sur fichiers des tables d'un groupe :

```
SELECT emaj.emaj_snap_group('<nom.du.groupe>', '<répertoire.de.stockage>',
'<options.COPY>');
```

Le nom du répertoire fourni doit être un chemin absolu. Ce répertoire doit exister au préalable et avoir les permissions adéquates pour que le cluster PostgreSQL puisse y écrire.

Le troisième paramètre précise le format souhaité pour les fichiers générés. Il prend la forme d'une chaîne de caractères reprenant la syntaxe précise des options disponibles pour la commande SQL COPY TO.

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Cette fonction *emaj_snap_group()* génère un fichier par table et par séquence appartenant au groupe de tables cité. Ces fichiers sont stockés dans le répertoire ou dossier correspondant au second paramètre de la fonction. D'éventuels fichiers de même nom se trouveront écrasés.

Le nom des fichiers créés est du type :

<nom.du.schema>_<nom.de.table/séquence>.snap

Les fichiers correspondant aux séquences ne comportent qu'une seule ligne, qui contient les caractéristiques de la séquence.

Les fichiers correspondant aux tables contiennent un enregistrement par ligne de la table, dans le format spécifié en paramètre. Ces enregistrements sont triés dans l'ordre croissant de la clé primaire.

En fin d'opération, un fichier *_INFO* est créé dans ce même répertoire. Il contient un message incluant le nom du groupe de tables et la date et l'heure de l'opération.

Il n'est pas nécessaire que le groupe de tables soit dans un état inactif, c'est-à-dire qu'il ait été arrêté au préalable.

Comme la fonction peut générer de gros ou très gros fichiers (dépendant bien sûr de la taille des tables), il est de la responsabilité de l'utilisateur de prévoir un espace disque suffisant.

Avec cette fonction, un test simple de fonctionnement d'E-Maj peut enchaîner :

- emaj_create_group(),
- emaj_start_group(),
- emaj_snap_group(<répertoire_1>),
- mises à jour des tables applicatives,
- emaj_rollback_group(),
- emaj_snap_group(<répertoire_2>),
- comparaison du contenu des deux répertoires par une commande diff par exemple.

4.6.2 Vidage des tables de log d'un groupe

Il est également possible d'obtenir le vidage total ou partiel sur fichiers des tables de log d'un groupe de tables. Ceci peut permettre de conserver une trace des mises à jour effectuées par un ou plusieurs traitements, à des fins d'archivage ou de comparaison entre plusieurs traitements. Pour ce faire, il suffit d'exécuter une requête :

```
SELECT emaj.emaj_snap_log_group('<nom.du.groupe>', '<marque.début>',
'<marque.fin>', '<répertoire.de.stockage>', '<options.COPY>');
```

Un *NULL* ou une chaîne vide peuvent être utilisés comme marque de début. Ils représentent alors la première marque connue.
Un *NULL* ou une chaîne vide peuvent être utilisés comme marque de fin. Ils représentent alors la situation courante.

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme marque de fin. Il représente alors la dernière marque posée.

Le nom du répertoire fourni doit être un chemin absolu. Ce répertoire doit exister au préalable et avoir les permissions adéquates pour que le cluster PostgreSQL puisse y écrire.

Le cinquième paramètre précise le format souhaité pour les fichiers générés. Il prend la forme d'une chaîne de caractères reprenant la syntaxe précise des options disponibles pour la commande SQL COPY TO.

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Cette fonction *emaj_snap_log_group()* génère un fichier par table de log, contenant la partie de cette table correspond aux mises à jour effectuées entre les deux marques citées ou la marque de début et la situation courante. Le nom des fichiers créés pour chaque table est du type :

<nom.du.schema>_<nom.de.table>_log.snap

La fonction génère également deux fichiers, contenant l'état des séquences applicatives lors de la pose respective des deux marques citées, et nommés :

<nom.du.groupe>_sequences_at_<nom.de.marque>

Si la borne de fin représente la situation courante, le nom du fichier devient :

<nom.du.groupe>_sequences_at_<heure>

l'heure étant exprimée avec un format HH.MM.SS.mmm

Ces fichiers sont stockés dans le répertoire ou dossier correspondant au quatrième paramètre de la fonction. D'éventuels fichiers de même nom se trouveront écrasés.

En fin d'opération, un fichier *_INFO* est créé dans ce même répertoire. Il contient un message incluant le nom du groupe de tables, les marques qui ont servi de bornes et la date et l'heure de l'opération.

Il n'est pas nécessaire que le groupe de tables soit dans un état inactif, c'est-à-dire qu'il ait été arrêté au préalable.

Comme la fonction peut générer de gros voire très gros fichiers (en fonction du volume des tables), il est de la responsabilité de l'utilisateur de prévoir un espace disque suffisant.

Les tables de log ont une structure qui découlent directement des tables applicatives dont elles enregistrent les mises à jour. Elles contiennent les mêmes colonnes avec les mêmes types. Mais elles possèdent aussi quelques colonnes techniques complémentaires :

- *emaj_verb* type de verbe SQL ayant généré la mise à jour (INS, UPD, DEL)
- *emaj_tuple* version des lignes (OLD pour les DEL et UPD ; NEW pour INS et UPD)
- *emaj_gid* identifiant de la ligne de log
- *emaj_changed* date et heure de l'insertion de la ligne dans la table de log
- *emaj_txid* identifiant de la transaction à l'origine de la mise à jour
- *emaj_user* rôle de connexion à l'origine de la mise à jour
- *emaj_user_ip* adresse ip du client à l'origine de la mise à jour (si le client est connecté avec le protocole ip)

4.6.3 Génération de scripts SQL rejouant les mises à jour tracées

Les tables de log contiennent toutes les informations permettant de rejouer les mises à jour. Il est dès lors possible de générer des requêtes SQL correspondant à toutes les mises à jour,intervenues entre 2 marques particulières ou à partir d'une marque, et de les enregistrer dans un fichier. C'est l'objectif de la fonction *emaj_generate_sql()*,.

Ceci peut permettre de ré-appliquer des mises à jour après avoir restauré les tables du groupe dans l'état correspondant à la marque initiale, sans avoir à ré-exécuter aucun traitement applicatif.

Pour générer ce script SQL, il suffit d'exécuter une requête :

```
SELECT emaj.emaj_generate_sql('<nom.du.groupe>', '<marque.début>',
'<marque.fin>', '<fichier>');
```

Un *NULL* ou une chaîne vide peuvent être utilisés comme marque de début. Ils représentent alors la première marque connue.
Un *NULL* ou une chaîne vide peuvent être utilisés comme marque de fin. Ils représentent alors la situation courante.

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé comme marque de fin. Il représente alors la dernière marque posée.

Le nom du fichier de sortie doit être exprimé sous forme de chemin absolu. Le fichier doit disposer des permissions adéquates pour que le cluster postgreSQL puisse y écrire. Si le fichier existe déjà, son contenu sera écrasé.

La fonction retourne le nombre de requêtes générées (hors commentaire et gestion de transaction).

Il n'est pas nécessaire que le groupe de tables soit dans un état inactif, c'est-à-dire qu'il ait été arrêté au préalable.

La fonction n'est disponible qu'à partir de la version de PostgreSQL 8.3.

Pour que le script puisse être généré, toutes les tables doivent avoir une clé primaire explicite (*PRIMARY KEY*).

Toutes les requêtes, *INSERT*, *UPDATE*, *DELETE* et *TRUNCATE* (pour les groupes de tables de type *audit_only*), sont générées dans l'ordre d'exécution initial.

Elles sont insérées dans une transaction. Elles sont entourées d'une requête *BEGIN TRANSACTION;* et d'une requête *COMMIT;*. Un commentaire initial rappelle les caractéristiques de la génération du script : la date et l'heure de génération, le groupe de tables concerné et les marques utilisées.

Les requêtes de type *TRUNCATE* enregistrées pour des groupes de tables de type *audit_only* sont également insérées dans le script.

Enfin, les séquences appartenant au groupe de tables sont repositionnées à leurs caractéristiques finales en fin de script.

Le fichier généré peut ensuite être exécuté tel quel par l'outil *psql*, pour peu que le rôle de connexion choisi dispose des autorisations d'accès adéquates sur les tables et séquences accédées.

La technique mise en œuvre aboutit à avoir des caractères antislash doublés dans le fichier de sortie. Il faut alors supprimer ces doublons avant d'exécuter le script, par exemple dans les environnement Unix/Linux par une commande du type :
sed 's/\\\\/\\/g' <nom_fichier> | psql ...

Comme la fonction peut générer un gros voire très gros fichier (en fonction du volume des logs), il est de la responsabilité de l'utilisateur de prévoir un espace disque suffisant.

Il est aussi de la responsabilité de l'utilisateur de désactiver d'éventuels triggers avant d'exécuter le script généré.

4.7 AUTRES FONCTIONS

4.7.1 Réinitialisation des tables de log d'un groupe

En standard, et sauf indication contraire, les tables de log sont vidées lors du démarrage du groupe de tables auquel elles appartiennent. En cas de besoin, il est néanmoins possible de réinitialiser ces tables de log avec la commande SQL suivante :

```
SELECT emaj.emaj_reset_group('<nom.du.groupe>');
```

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Pour réinitialiser les tables de log d'un groupe, ce dernier doit bien sûr être à l'état inactif (« *IDLE* »).

4.7.2 Commentaires sur les groupes

Il est possible de positionner un commentaire sur un groupe quelconque. Pour se faire, il suffit d'exécuter la requête suivante :

```
SELECT emaj.emaj_comment_group('<nom.du.groupe>', '<commentaire>');
```

La fonction ne retourne aucune donnée.

Pour modifier un commentaire, il suffit d'exécuter à nouveau la fonction pour le même groupe de tables, avec le nouveau commentaire.

Pour supprimer un commentaire, il suffit d'exécuter la fonction avec une valeur NULL pour le paramètre commentaire.

Les commentaires sont stockés dans la colonne *group_comment* de la table *emaj.emaj_group* qui décrit les groupes.

4.7.3 Vérification de la consistance de l'environnement E-Maj

Une fonction permet de vérifier la consistance de l'environnement E-Maj. Cela consiste à vérifier l'intégrité de chaque schéma d'E-Maj et de chaque groupe de tables créé. Cette fonction s'exécute par la requête SQL suivante :

```
SELECT * FROM emaj.emaj_verify_all();
```

Pour chaque schéma E-Maj (*emaj* et les éventuels schémas secondaires), la fonction vérifie :

- que toutes les tables, fonctions et séquences et tous les types soit sont des objets de l'extension elle-même, soit sont bien liés aux groupes de tables créés,
- qu'il ne contient ni vue, ni « foreign table », ni domaine, ni conversion, ni opérateur et ni classe d'opérateur.

Ensuite, pour chaque groupe de tables créé, la fonction procède aux mêmes contrôles que ceux effectués lors des opérations de démarrage de groupe, de pose de marque et de rollback (voir §5.2).

La fonction retourne un ensemble de lignes qui décrivent les éventuelles anomalies rencontrées. Si aucune anomalie n'est détectée, la fonction retourne une unique ligne contenant le message :

'*No error detected*'

La fonction *emaj_verify_all()* peut être exécutée par les rôles membres de *emaj_adm* et *emaj_viewer*.

Si des anomalies sont détectées, par exemple suite à la suppression d'une table applicative référencée dans un groupe, les mesures appropriées doivent être prises. Typiquement, les éventuelles tables de log ou fonctions orphelines doivent être supprimées manuellement.

4.7.4 Arrêt forcé d'un groupe de tables

Il peut arriver qu'un groupe de tables endommagé ne puisse pas être arrêté. C'est par exemple le cas si une table applicative du groupe de tables a été supprimée alors que ce dernier était actif. Si les fonctions usuelles *emaj_stop_group()* ou *emaj_stop_groups()* retournent une erreur, il est possible de forcer l'arrêt d'une groupe de tables à l'aide de la fonction *emaj_force_stop_group()*.

```
SELECT emaj.emaj_force_stop_group('<nom.du.groupe>');
```

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Cette fonction *emaj_force_stop_group()* effectue le même traitement que la fonction *emaj_stop_group()*, Elle présente néanmoins les différences suivantes :

- elle gère les éventuelles absences des tables et triggers à désactiver, des messages de type « *Warning* » étant générés dans ces cas,
- elle ne pose pas de marque d'arrêt.

Une fois la fonction exécutée, le groupe de table est en état « *IDLE* ». Il peut alors être supprimé ou modifié avec les fonctions *emaj_drop_group()* ou *emaj_alter_group()*.

Il est recommandé de n'utiliser cette fonction qu'en cas de réel besoin.

4.7.5 Suppression forcée d'un groupe de tables

Il peut arriver qu'un groupe de tables endommagé ne puisse pas être arrêté. Mais n'étant pas arrêté, il est impossible de le supprimer. Pour néanmoins pouvoir supprimer une table avec un log actif, une fonction spéciale est disponible.

```
SELECT emaj.emaj_force_drop_group('<nom.du.groupe>');
```

La fonction retourne le nombre de tables et de séquences contenues dans le groupe.

Cette fonction *emaj_force_drop_group()* effectue le même traitement que la fonction *emaj_drop_group()*, mais sans contrôler l'état du groupe au préalable. Il est recommandé de n'utiliser cette fonction qu'en cas de réel besoin.

NB : depuis la création de la fonction *emaj_force_stop_group()*, cette fonction *emaj_force_drop_group()* devient en principe inutile. Elle est susceptible de disparaître dans une future version d'E-Maj.

4.8 ROLLBACK AVEC PARALLÉLISME

Sur les serveurs équipés de plusieurs processeurs ou cœurs de processeurs, il peut être intéressant de réduire la durée des rollbacks en parallélisant l'opération sur plusieurs couloirs. A cette fin, E-Maj fournit un client spécifique qui se lance en ligne de commande. Celui-ci active les fonctions de rollback d'E-Maj au travers de plusieurs connexions à la base de données en parallèle.

4.8.1 Sessions

Pour paralléliser un rollback, E-Maj affecte les tables et séquences à traiter pour un ou plusieurs groupes de tables à un certain nombre de « *sessions* ». Chaque session est ensuite traitée dans un couloir propre.

Néanmoins, pour garantir l'intégrité de l'opération, le rollback de toutes les sessions s'exécute au sein d'une unique transaction.

Pour obtenir des sessions les plus équilibrées possibles, E-Maj tient compte :

- du nombre de sessions spécifiés par l'utilisateur dans sa commande,
- des statistiques des lignes à annuler, telles que la fonction *emaj_log_stat_group()* les restitue,
- des contraintes de clés étrangères qui relient plusieurs tables entre-elles, 2 tables mises à jour et reliées entre-elles par une clé étrangère étant affectées à une même session.

4.8.2 Préalables

La commande qui permet de lancer des rollbacks avec parallélisme est codée en php. En conséquence, le logiciel *php* et son interface PostgreSQL doivent être installés sur le serveur qui exécute cette commande (qui n'est pas nécessairement le même que celui qui héberge le cluster PostgreSQL).

Le rollback de chaque session au sein d'une unique transaction implique l'utilisation de commit à deux phases. En conséquence, le paramètre *max_prepared_transaction* du fichier *postgresql.conf* doit être ajusté. La valeur par défaut du paramètre est 0. Il faut donc la modifier en spécifiant une valeur au moins égale au nombre maximum de sessions qui seront utilisées.

4.8.3 Syntaxe

La syntaxe de la commande permettant un rollback avec parallélisme est :

emajParallelRollback.php -g <nom.du.ou.des.groupes> -m <marque> -s <nombre.de.sessions> [OPTIONS]...

Options générales :

-l	spécifie que le rollback demandé est de type « *logged rollback* » (voir §4.2.7)
-v	affiche davantage d'information sur le déroulement du traitement
--help	affiche uniquement une aide sur la commande
--version	affiche uniquement la version du logiciel

Options de connexion :

-d	base de données à atteindre
-h	hôte à atteindre
-p	port-ip à utiliser
-U	rôle de connexion
-W	mot de passe associé à l'utilisateur, si nécessaire

Pour remplacer tout ou partie des paramètres de connexion, les variables habituelles *PGDATABASE*, *PGPORT*, *PGHOST* et/ou *PGUSER* peuvent être également utilisées.

Pour spécifier une liste de groupes de tables dans le paramètre -g, séparer le nom de chaque groupe par une virgule.

Le rôle de connexion fourni doit être soit un super-utilisateur, soit un rôle ayant les droits *emaj_adm*.

Pour des raisons de sécurité, il n'est pas recommandé d'utiliser l'option -W pour fournir un mot de passe. Il est préférable d'utiliser le fichier *.pgpass* (voir la documentation de PostgreSQL).

Pour que l'opération de rollback puisse être exécutée, le ou les groupes de tables doivent être actifs. Si le rollback concerne plusieurs groupes, la marque demandée comme point de rollback doit correspondre à un même moment dans le temps, c'est à dire qu'elle doit avoir été créée par une unique commande *emaj_set_mark_groups()*.

Le mot clé '*EMAJ_LAST_MARK*' peut être utilisé pour référencer la dernière marque du ou des groupes de tables.

Pour tester la commande *emajParallelRollback.php*, E-Maj fournit un script, *prep-pr.sql*. Il prépare un environnement avec deux groupes de tables contenant quelques tables et séquences, sur lesquelles des mises à jour ont été effectuées, entrecoupées de marques. Suite à l'exécution de ce script sous *psql*, on peut lancer la commande telle qu'indiquée dans le message de fin d'exécution du script.

4.8.4 Exemples

La commande :

../php/emajParallelRollback.php -d mydb -g myGroup1 -m Mark1 -s 3

se connecte à la base de données mydb et exécute un rollback du groupe myGroup1 à la marque Mark1, avec 3 sessions en parallèle.

La commande :

../php/emajParallelRollback.php -d mydb -g "myGroup1,myGroup2" -m Mark1 -s 3 -l

se connecte à la base de données mydb et exécute un rollback annulable (« logged rollback ») des 2 groupes myGroup1 et myGroup2 à la marque Mark1, avec 3 sessions en parallèle.

5 CONSIDÉRATIONS DIVERSES

5.1 PARAMÉTRAGE

L'extension E-Maj fonctionne avec quelques paramètres. Ceux-ci sont stockés dans la table interne *emaj_param*.

La structure de la table *emaj_param* est la suivante :

Colonne	**Type**	**Description**
param_key	TEXT	mot-clé identifiant le paramètre
param_value_text	TEXT	valeur du paramètre, s'il est de type texte (sinon NULL)
param_value_int	INT	valeur du paramètre, s'il est de type entier (sinon NULL)
param_value_boolean	BOOLEAN	valeur du paramètre, s'il est de type booléen (sinon NULL)
param_value_interval	INTERVAL	valeur du paramètre, s'il est de type intervalle (sinon NULL)

La procédure d'installation de l'extension E-Maj ne crée qu'une seule ligne dans la table *emaj_param*. Cette ligne, qui ne doit pas être modifiée, décrit le paramètre :

- *version* (texte) version courante d'E-Maj

Mais l'administrateur d'E-Maj peut insérer d'autres lignes dans *emaj_param* pour modifier la valeur par défaut de certains paramètres.

Les valeurs de clé des paramètres sont, par ordre alphabétique :

- *avg_fkey_check_duration* (intervalle) valeur par défaut = 5 µs ; définit la durée moyenne du contrôle d'une clé étrangère ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données (voir §4.5.3).
- *avg_row_delete_log_duration* (intervalle) valeur par défaut = 10 µs ; définit la durée moyenne de suppression d'une ligne du log ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données (voir §4.5.3).
- *avg_row_rollback_duration* (intervalle) valeur par défaut = 100 µs ; définit la durée moyenne de rollback d'une ligne ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données (voir § 4.5.3).
- *fixed_table_rollback_duration* (intervalle) valeur par défaut = 5 ms ; définit un coût fixe de rollback de toute table ou séquence appartenant à un groupe ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données (voir § 4.5.3).

- *fixed_table_with_rollback_duration* (intervalle) valeur par défaut = 2,5 ms ; définit un coût fixe additionnel de rollback d'une table ayant effectivement des mises à jour à annuler ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données (voir §4.5.3).
- *history_retention* (intervalle) valeur par défaut = 1 mois ; elle peut être ajustée pour changer la durée de rétention des lignes dans la table historique d'E-Maj, *emaj_hist* (voir § 5.3),

Exemple de requête SQL permettant de spécifier une durée de rétention des lignes dans l'historique différente de la valeur par défaut (1 mois) :

```
INSERT INTO emaj.emaj_param (param_key, param_value_interval) VALUES
('history_retention','15 days'::interval);
```

Il est également possible de gérer la valeur des paramètres par des outils graphiques tels que PgAdmin ou phpPgAdmin.

5.2 CONTRÔLES INTERNES

Lors de l'exécution des fonctions de démarrage de groupe, de pose de marque et de rollback, E-Maj effectue un certain nombre de contrôles afin de vérifier l'intégrité des groupes de tables sur lesquels porte l'action.

Ces contrôles d'intégrité du groupe de tables vérifient que :

- la version de PostgreSQL avec laquelle le groupe a été créé est bien compatible avec la version actuelle,
- chaque séquence ou chaque table applicative du groupe existe toujours bien,
- chacune des tables d'un groupe a toujours sa table de log associée, ses deux fonctions de log et de rollback ainsi que ses triggers,
- la structure des tables de log est toujours en phase avec celle des tables applicatives associées.

5.3 TRAÇABILITÉ DES OPÉRATIONS

Toutes les opérations réalisées par E-Maj et qui modifient d'une manière ou d'une autre un groupe de tables sont tracées dans une table nommée *emaj_hist*.

La structure de la table *emaj_hist* est la suivante.

Colonne	Type	Description
hist_id	BIGSERIAL	numéro de série identifiant une ligne dans cette table historique
hist_datetime	TIMESTAMPTZ	date et heure d'enregistrement de la ligne
hist_function	TEXT	fonction associée à l'événement
hist_event	TEXT	type d'événement
hist_object	TEXT	nom de l'objet sur lequel porte l'événement (groupe, table ou séquence)
hist_wording	TEXT	commentaires complémentaires
hist_user	TEXT	rôle à l'origine de l'événement
hist_txid	BIGINT	numéro de la transaction à l'origine de l'événement

La colonne *hist_function* peut prendre les valeurs suivantes :

- *EMAJ_INIT* initialisation E-Maj
- *CREATE_GROUP* création d'un groupe de tables
- *COMMENT_GROUP* positionnement d'un commentaire sur un groupe
- *DROP_GROUP* suppression d'un groupe de tables
- *ALTER_GROUP* modification d'un groupe de tables
- *FORCE_DROP_GROUP* suppression forcée d'un groupe de tables
- *START_GROUP* démarrage d'un groupe de tables
- *START_GROUPS* démarrage de plusieurs groupes de tables
- *STOP_GROUP* arrêt d'un groupe de tables
- *STOP_GROUPS* arrêt de plusieurs groupes de tables
- *FORCE_STOP_GROUP* arrêt forcé d'un groupe de tables
- *LOCK_GROUP* pose d'un verrou exclusif sur les tables d'un groupe
- *LOCK_GROUPS* pose d'un verrou exclusif sur les tables de plusieurs groupes
- *LOCK_SESSION* pose d'un verrou exclusif sur les tables d'une session de rollback
- *SET_MARK_GROUP* pose d'une marque pour un groupe de tables
- *SET_MARK_GROUPS* pose d'une marque pour plusieurs groupes de tables
- *COMMENT_MARK_GROUP* positionnement d'un commentaire sur une marque
- *DELETE_MARK_GROUP* suppression d'une marque pour un groupe de tables
- *RENAME_MARK_GROUP* renommage d'une marque pour un groupe de tables
- *ROLLBACK_GROUP* rollback des mises à jour pour un groupe de tables
- *ROLLBACK_GROUPS* rollback des mises à jour pour plusieurs groupes de tables
- *RESET_GROUP* réinitialisation du contenu des tables de log d'un groupe
- *ROLLBACK_TABLE* rollback des mises à jour d'une table
- *ROLLBACK_SEQUENCE* rollback d'une séquence

- *SNAP_GROUP* vidage des tables et séquences d'un groupe
- *SNAP_LOG_GROUP* vidage des tables de log d'un groupe
- *GENERATE_SQL* génération d'un script psql pour rejouer des mises à jour

La colonne *hist_event* prend les valeurs suivantes :

- *BEGIN* début
- *END* fin
- *MARK DELETED* marque supprimée
- *SCHEMA CREATED* schéma secondaire créé
- *SCHEMA DROPPED* schéma secondaire supprimé

Le contenu de la table *emaj_hist* peut être visualisé par quiconque dispose des autorisations suffisantes (rôles super-utilisateur, *emaj_adm* ou *emaj_viewer*)

A chaque démarrage de groupe (fonction *emaj_start_group()*) et suppression des marques les plus anciennes (fonction *emaj_delete_before_mark_group()*), les événements les plus anciens de la table *emaj_hist* sont supprimés. Les événements conservés sont ceux à la fois postérieurs à un délai de rétention paramétrable et postérieurs à la pose de la marque active la plus ancienne. Par défaut, la durée de rétention des événements est de 1 mois. Mais cette valeur peut être modifiée à tout moment en insérant par une requête SQL le paramètre *history_retention* dans la table *emaj_param* (voir § 5.1).

5.4 IMPACTS SUR L'ADMINISTRATION DU CLUSTER ET DE LA BASE DE DONNÉES

5.4.1 Arrêt/relance du cluster

L'utilisation d'E-Maj n'apporte aucune contrainte particulière sur l'arrêt et la relance des clusters PostgreSQL.

5.4.1.1 Règle générale

Au redémarrage du cluster, tous les objets d'E-Maj se retrouvent dans le même état que lors de l'arrêt du cluster : les triggers de logs des groupes de tables actifs restent activés et les tables de logs sont alimentées avec les mises à jours annulables déjà enregistrées.

Si une transaction avait des mises à jour en cours non validées lors de l'arrêt du cluster, celle-ci est annulée lors du redémarrage, les écritures dans les tables de logs se trouvant ainsi annulées en même temps que les modifications de tables.

Cette règle s'applique bien sûr aux transactions effectuant des opérations E-Maj telles que le démarrage ou l'arrêt d'un groupe, un rollback, une suppression de marque, etc.

5.4.1.2 Rollback des séquences

Lié à une contrainte de PostgreSQL, seul le rollback des séquences applicatives n'est pas protégé par les transactions. C'est la raison pour laquelle les séquences sont rollbackées en toute fin d'opération de rollback (voir §4.2.6). (Pour la même raison, lors de la pose d'une marque, les séquences applicatives sont traitées en début d'opération.)

Au cas où un rollback serait en cours au moment de l'arrêt du cluster, il est recommandé de procéder à nouveau à ce même rollback juste après le redémarrage du cluster, afin de s'assurer que les séquences et tables applicatives restent bien en phase.

5.4.2 Sauvegarde et restauration

E-Maj peut permettre de diminuer la fréquence avec laquelle les sauvegardes sont nécessaires. Mais E-Maj ne peut se substituer totalement aux sauvegardes habituelles, qui restent nécessaires pour conserver sur un support externe des images complètes des bases de données !

5.4.2.1 Sauvegarde et restauration au niveau fichier

Lors des sauvegardes ou des restauration des clusters au niveau fichier, il est essentiel de sauver ou restaurer TOUS les fichiers du cluster. Ceci inclut bien sûr les fichiers correspondant au tablespace *tspemaj*, s'il existe.

Après restauration des fichiers, les groupes de tables se retrouveront dans l'état dans lequel ils se trouvaient lors de la sauvegarde, et l'activité de la base de données peut reprendre sans opération E-Maj particulière.

5.4.2.2 Sauvegarde et restauration logique de base de données complète

Pour les groupes de tables arrêtés (en état *IDLE*), comme les triggers de logs sont inactifs et que le contenu des tables de log n'a pas d'importance, il n'y a aucune précaution particulière à prendre pour les retrouver dans le même état après une restauration.

Pour les groupes de tables en état *LOGGING* au moment de la sauvegarde, il faut s'assurer que les triggers de logs ne sont pas activés au moment de la reconstitution (restauration) des tables applicatives. Dans le cas contraire, pendant la reconstruction des tables, toutes les insertions de lignes seraient aussi enregistrées dans les tables de logs !

Lorsqu'on utilise les commandes pg_dump pour la sauvegarde et psql ou pg_restore pour la restauration et que l'on traite des bases complètes (schéma et données), ces outils font en sorte que les triggers, dont les triggers de log E-Maj, ne soient activés qu'en fin de restauration. Il n'y a donc pas de précautions particulières à prendre.

En revanche, dans le cas de sauvegarde et restauration des données seulement (sans schéma, avec les options *-a* ou *--data-only*), alors il faut spécifier l'option *--disable-triggers* :

- à la commande *pg_dump* (ou *pg_dumpall*) pour les sauvegardes au format *plain (psql* utilisé pour le rechargement),
- à la commande *pg_restore* pour les sauvegardes au format *tar* ou *custom*.

5.4.2.3 Sauvegarde et restauration logique de base de données partielle

Les outils *pg_dump* et *pg_restore* permettent de ne traiter qu'un sous-ensemble des schémas et/ou des tables d'une base de données.

Restaurer un sous-ensemble des tables applicatives et/ou des tables de log comporte un risque très élevé de corruption des données en cas de rollback E-Maj ultérieur sur le groupe de tables concerné. En effet, dans ce cas, il est impossible de garantir la cohérence entre les tables applicatives, les tables de log et les tables internes d'E-Maj, qui contiennent des données essentielles aux opérations de rollback.

S'il s'avère nécessaire de procéder à une restauration partielle de tables applicatives, il faut faire suivre cette restauration de la suppression puis recréation du ou des groupes de tables touchées par l'opération.

De la même manière il est fortement déconseillé de procéder à une restauration partielle des tables du schéma *emaj*.

Le seul cas de restauration partielle sans risque concerne la restauration du contenu complet du schéma emaj, ainsi que de toutes les tables et séquences appartenant à tous les groupes de tables créés dans la base de données.

5.4.3 Chargement de données

Au delà de l'utilisation de *pg_restore* ou de *psql* avec un fichier issu de *pg_dump* évoquée plus haut, il est possible de procéder à des chargements massifs de tables par la commande SQL *COPY* ou la méta-commande *psql \copy*. Dans les deux cas, le chargement des données provoque le déclenchement des triggers sur *INSERT*, dont bien sûr celui utilisé pour le log d'E-Maj. Il n'y a donc aucune contrainte à l'utilisation de *COPY* et *\copy* avec E-Maj.

Pour l'utilisation d'autres outils de chargement, il convient de vérifier que les triggers sont bien activés à chaque insertion de ligne.

5.4.4 Réorganisation des tables de la base de données

5.4.4.1 Réorganisation des tables applicatives

Les tables applicatives protégées par E-Maj peuvent être réorganisées par une commande SQL *CLUSTER*. Que les triggers de logs soient actifs ou non, le processus de réorganisation n'a pas d'impact pas le contenu des tables de log.

5.4.4.2 Réorganisation des tables E-Maj

Aucune table des schémas d'E-Maj ne possède d'index « cluster », que ce soit les tables de log ou les quelques tables internes. Aussi, l'installation d'E-Maj n'a aucun impact opérationnel sur l'exécution des commandes SQL *CLUSTER* au niveau de la base de données.

5.4.5 Utilisation d'E-Maj avec de la réplication

5.4.5.1 Réplication intégrée

E-Maj est parfaitement compatible avec le fonctionnement des différents mode de réplication intégrée de PostgreSQL (archivage des WAL et PITR, Streaming Replication asynchrone ou synchrone). Tous les objets E-Maj des bases hébergées sur le cluster sont en effet répliqués comme toutes les autres objets du cluster.

Néanmoins, compte tenu de la façon dont PostgreSQL gère les séquences, la valeur courante des séquences peut être un peu en avance sur les clusters esclave par rapport au cluster maître. Pour E-Maj, ceci induit des statistiques générales indiquant parfois un nombre de lignes de log un peu supérieur à la réalité. Mais il n'y a pas de conséquence sur l'intégrité des données.

5.4.5.2 Autres solutions de réplication

L'utilisation d'E-Maj avec des solutions de réplication externe basées sur des triggers, tels que Slony ou Londiste, nécessite réflexion... On évitera probablement de mettre sous réplication les tables de log et les tables techniques d'E-Maj.

5.4.6 Changement de version de PostgreSQL

5.4.6.1 Versions PostgreSQL 8.2 et 8.3

Les groupes de tables créés dans une version de PostgreSQL 8.2 ou 8.3 ne peuvent être gérés que dans leur version de création. En effet, dans ces versions de PostgreSQL, les fonctions E-Maj présentent quelques différences de fonctionnement.

C'est pourquoi, lors d'une migration de PostgreSQL 8.2 ou 8.3 vers une version supérieure, il est nécessaire de désinstaller et réinstaller complètement E-Maj (voir §3.4 et §3.2). Il est donc impossible de conserver des groupes de tables actifs pendant le changement de version PostgreSQL.

5.4.6.2 Versions PostgreSQL 8.4 et supérieures

Pour toutes les versions de PostgreSQL supérieures ou égales à 8.4, les objets et fonctions E-Maj sont identiques.

Aussi est-il possible de changer de version de PostgreSQL sans réinstallation d'E-Maj. Les groupes de tables peuvent même être actifs lors du changement de version.

Néanmoins, il est recommandé d'arrêter les groupes de tables avant un changement de version PostgreSQL, les bases de données étant alors en principe dans un état stable. De plus, si le changement de version s'effectue avec un déchargement et rechargement des données, l'exécution d'une fonction *emaj_reset_group()* peut permettre de diminuer la quantité de données à manipuler et donc d'accélérer l'opération.

5.5 SENSIBILITÉ AUX CHANGEMENTS DE DATE ET HEURE SYSTÈME

Pour garantir l'intégrité du contenu des tables gérées par E-Maj, il est important que le mécanisme de rollback soit insensible aux éventuels changements de date et heure du système qui héberge le cluster PostgreSQL.

Même si les date et heure de chaque mise à jour ou de chaque pose de marque sont enregistrées, ce sont les valeurs de séquences enregistrées lors des poses de marques qui servent à borner les opérations dans le temps. Ainsi, les rollbacks comme les suppressions de marques sont insensibles aux changements éventuels de date et heure du système.

Seules deux actions mineures peuvent être influencées par un changement de date et heure système :

- la suppression des événements les plus anciens dans la table emaj_hist (le délai de rétention est un intervalle de temps)
- la recherche de la marque immédiatement antérieure à une date et une heure données, telle que restituée par la fonction *emaj_get_previous_mark_group()*.

5.6 PERFORMANCES

5.6.1 Surcoût de l'enregistrement des mises à jour

Enregistrer toutes les mises à jour de tables dans les tables de log E-Maj a nécessairement un impact sur la durée d'exécution de ces mises à jour. L'impact global du log sur un traitement donné dépend de nombreux facteurs. Citons en particulier :

- la part que représente l'activité de mise à jour dans ce traitement,
- les performances intrinsèques du périphérique de stockage qui supporte les tables de log.

Néanmoins, le plus souvent, le surcoût du log E-Maj sur le temps global d'un traitement se limite à quelques pour-cents.

5.6.2 Durée d'un rollback E-Maj

La durée d'exécution d'une fonction de rollback E-Maj dépend elle aussi de nombreux facteurs, tels que :

- le nombre de mises à jour à annuler,
- les caractéristiques intrinsèques du serveur et de sa périphérie disque et la charge liée aux autres activités supportées par le serveur
- la présence de trigger ou de clés étrangères sur les tables traitées par le rollback
- les contentions sur les tables lors de la pose des verrous

Pour avoir un ordre de grandeur du temps que prendrait un rollback E-Maj, on peut utiliser la fonction *emaj_estimate_rollback_duration()* (Cf §4.5.3)

5.6.3 Optimiser le fonctionnement d'E-Maj

Voici quelques conseils pour optimiser les performances d'E-Maj.

5.6.3.1 Utiliser des tablespaces

Positionner des tables sur des tablespaces permet de mieux maîtriser leur implantation sur les disques et ainsi de mieux répartir la charge d'accès à ces tables, pour peu que ces tablespaces soient physiquement implantés sur des disques ou systèmes de fichiers dédiés. Pour minimiser les perturbations que les accès aux tables de log peuvent causer aux accès aux tables applicatives, l'administrateur E-Maj dispose de deux moyens d'utiliser des tablespaces pour stocker les tables et index de log.

En créant un tablespace nommé *tspemaj* avant la création des groupes de tables, les tables de log seront créées par défaut dans ce tablespace, sans autre paramétrage.

Mais, au travers de paramètres positionnés dans la table *emaj_group_def*, il est également possible de spécifier, pour chaque table et index de log, un tablespace à utiliser. (Voir le § 4.2.2.3.)

5.6.3.2 Déclarer les clés étrangères *DEFERRABLE*

Au moment de leur création, les clés étrangères (*foreign key*) peuvent être déclarées *DEFERRABLE*. Si une clé étrangère est déclarée *DEFERRABLE* et qu'aucune clause *ON DELETE* ou *ON UPDATE* n'est utilisée, elle ne sera pas supprimée en début et recréées en fin de rollback E-Maj. Les contrôles des clés étrangères pour les lignes modifiées seront simplement différés en fin de rollback, une fois toutes les tables de log traitées. En règle générale cela accélère sensiblement l'opération de rollback.

5.7 LIMITES D'UTILISATION

L'utilisation de l'extension E-Maj présente quelques limitations.

- La version PostgreSQL minimum requise est la version 8.2.
- Toutes les tables appartenant à un groupe de tables de type « *rollbackable* » doivent avoir une clé primaire explicite (*PRIMARY KEY*).
- Pour une table déclarée dans un groupe, la somme des longueurs du nom du schéma et du nom de la table ne peut dépasser 52 caractères.
- Le schéma nommé "*emaj*" est créé lors de l'initialisation d'E-Maj. Si son nom devait être changé, le script *emaj.sql,* les scripts de test et la commande *emajParallelRollback.php* devraient être adaptés en conséquence.
- Si un verbe SQL *TRUNCATE* est exécuté sur une table applicative appartenant à un groupe de tables, il n'est pas possible pour E-Maj de remettre la table dans un état antérieur à cette requête. En effet, lors de l'exécution d'un *TRUNCATE*, aucun trigger n'est déclenché à chaque suppression de ligne. A partir des versions de PostgreSQL 8.4, un trigger, créé par E-Maj, empêche l'exécution d'une requête *TRUNCATE* sur toute table appartenant à groupe de tables en état démarré. Pour les version antérieures de PostgreSQL, cette détection n'est pas possible.
- L'utilisation d'une séquence globale pour une base de données induit une limite dans le nombre de mises à jour qu'E-Maj est capable de tracer tout au long de sa vie. Cette limite est égale à 2^63, soit environ 10^19 (mais seulement d'environ 10^10 sur de vieilles plate-formes). Cela permet tout de même d'enregistrer 10 millions de mises à jour par seconde (soit 100 fois les meilleurs performances des benchmarks en 2012) pendant … 30.000 ans (et dans le pire des cas, 100 mises à jour par seconde pendant 5 ans). S'il s'avérait nécessaire de réinitialiser cette séquence, il faudrait simplement désinstaller puis réinstaller l'extension E-Maj (voir §3.2).
- Si une opération de DDL est exécutée sur une table applicative appartenant à un groupe de tables, il n'est pas possible pour E-Maj de remettre la table dans un état antérieur.

Pour détailler ce dernier point, il peut être intéressant de comprendre les conséquences de l'exécution d'une requête SQL de type DDL sur le fonctionnement d'E-Maj, en fonction du type d'opération effectué.

- Si une nouvelle table est créée, elle ne pourra entrer dans la constitution d'un groupe qu'après l'arrêt, la suppression et la recréation du groupe.
- Si une table appartenant à un groupe en état actif était supprimée, il n'y aurait aucun moyen pour un rollback de retrouver le contenu de la table.
- Pour une table appartenant à un groupe en état actif, l'ajout ou la suppression d'une colonne provoquerait une erreur lors de l'*INSERT/UPDATE/DELETE* suivant.
- Pour une table appartenant à un groupe en état actif, le renommage d'une colonne ne provoquerait pas nécessairement d'erreur lors de l'enregistrement des mises à jour suivantes. En revanche, de par les contrôles propres à E-Maj, toute tentative de pose de marque ou de rollback échouerait ensuite.

- Pour une table appartenant à un groupe en état actif, le changement de type d'une colonne provoquerait une inconsistance entre les structures des tables applicative et de log. Mais, suivant le changement apporté au type de donnée, l'enregistrement dans la table de log pourrait échouer ou non. De plus, il pourrait y avoir une altération des données, par exemple en cas d'agrandissement de la longueur de la donnée. De toutes les façons, de par les contrôles propres à E-Maj, toute tentative de pose de mark ou de rollback échouerait ensuite.
- En revanche, il est possible de créer, modifier ou supprimer les index, les droits ou les contraintes d'une table appartenant à un groupe, alors que ce dernier se trouve dans un état actif. Mais un retour arrière sur ces évolutions ne pourrait bien sûr pas être assuré par E-Maj.

5.8 RESPONSABILITÉS DE L'UTILISATEUR

5.8.1 Constitution des groupes de tables

La constitution des groupes de tables est fondamentale pour garantir l'intégrité des bases de données. Il est de la responsabilité de l'administrateur d'E-Maj de s'assurer que toutes les tables qui sont mises à jour par un même traitement sont bien incluses dans le même groupe de tables.

5.8.2 Exécution appropriée des fonctions principales

Les fonctions de démarrage et d'arrêt de groupe, de pose de marque et de rollback positionnent des verrous sur les tables du groupe pour s'assurer que des transactions de mises à jour ne sont pas en cours lors de ces opérations. Mais il est de la responsabilité de l'utilisateur d'effectuer ces opérations au « bon moment », c'est à dire à des moments qui correspondent à des points vraiment stables dans la vie de la base.

5.8.3 Gestion des triggers applicatifs

Des triggers peuvent avoir été créés sur des tables applicatives. Il n'est pas rare que ces triggers génèrent une ou des mises à jour sur d'autres tables. Il est alors de la responsabilité de l'administrateur E-Maj de comprendre l'impact des opérations de rollback sur les tables concernées par des triggers et de prendre le cas échéant les mesures appropriées.

Si le trigger ajuste simplement le contenu de la ligne à insérer ou modifier, c'est la valeur finale des colonnes qui sera enregistrée dans la table de log. Le rollback permettra de repositionner les anciennes valeurs. Néanmoins, pour que le trigger ne se déclenche pas lors des rollbacks, il peut être nécessaire de le désactiver pour cette opération.

Si le trigger met à jour une autre table, deux cas sont à considérer :

- si la table modifiée par le trigger fait partie du même groupe de tables, il est nécessaire de désactiver le trigger avant l'opération de rollback et le réactiver après, de sorte que ce soit le rollback de la table modifiée qui procède à toutes les mises à jour,
- si la table modifiée par le trigger ne fait pas partie du même groupe de tables, il est essentiel d'analyser les conséquences du rollback de la table possédant le trigger sur la table modifiée par ce trigger, afin d'éviter que le rollback ne provoque un déphasage entre les 2 tables. Dans ce cas, la désactivation du trigger pendant l'opération de rollback peut ne pas être suffisante.

5.8.4 Modification des tables et séquences internes d'E-Maj

De par les droits qui leurs sont attribués, les super-utilisateurs et les rôles détenant les droits *emaj_adm* peuvent mettre à jour toutes les tables internes d'E-Maj.

Mais en pratique, seules les tables *emaj_group_def* et *emaj_param* ne doivent être modifiées par ces utilisateurs. Toute modification du contenu des autres tables ou des séquences internes peut induire des corruptions de données lors d'éventuelles opérations de rollback.

6 PLUGIN PHPPGADMIN

Pour faciliter l'utilisation d'E-Maj, un « plugin » pour l'outil d'administration phpPgAdmin 5 est également disponible.

6.1 PRÉSENTATION GÉNÉRALE

Pour les bases de données dans lesquelles l'extension E-Maj a été installée, et si l'utilisateur est connecté avec un rôle qui dispose des autorisations nécessaires, les objets E-Maj sont visibles et manipulables. Il est ainsi possible de :

- voir la liste des groupes de tables et effectuer toutes les actions possibles, en fonction de l'état du groupe (démarrage, arrêt, suppression, pose de marque, rollback),
- définir ou modifier la composition des groupes,
- voir la liste des marques posées pour un groupe de tables et effectuer toutes les actions possibles les concernant (suppression, renommage, rollback, ajout ou modification de commentaire),
- obtenir toutes les statistiques sur le contenu des tables de log.

6.2 UTILISATION

6.2.1 Comment accéder à E-Maj dans l'interface phpPgAdmin

Sur la figure 1 ci-dessous, on peut remarquer qu'après être connecté à une base de données dans laquelle l'extension E-Maj a été installée, et avec un rôle qui dispose des droits suffisants (super-utilisateur, *emaj_adm* ou *emaj_viewer*), une nouvelle icône apparaît dans la barre d'icônes horizontale de la base. Bien sûr, le schéma *emaj* apparaît dans la liste des schémas.

Dans l'arbre de gauche, un nouvel objet E-Maj apparaît également. Son ouverture permet de visualiser la liste des groupes de tables créés et d'accéder à l'un d'eux.

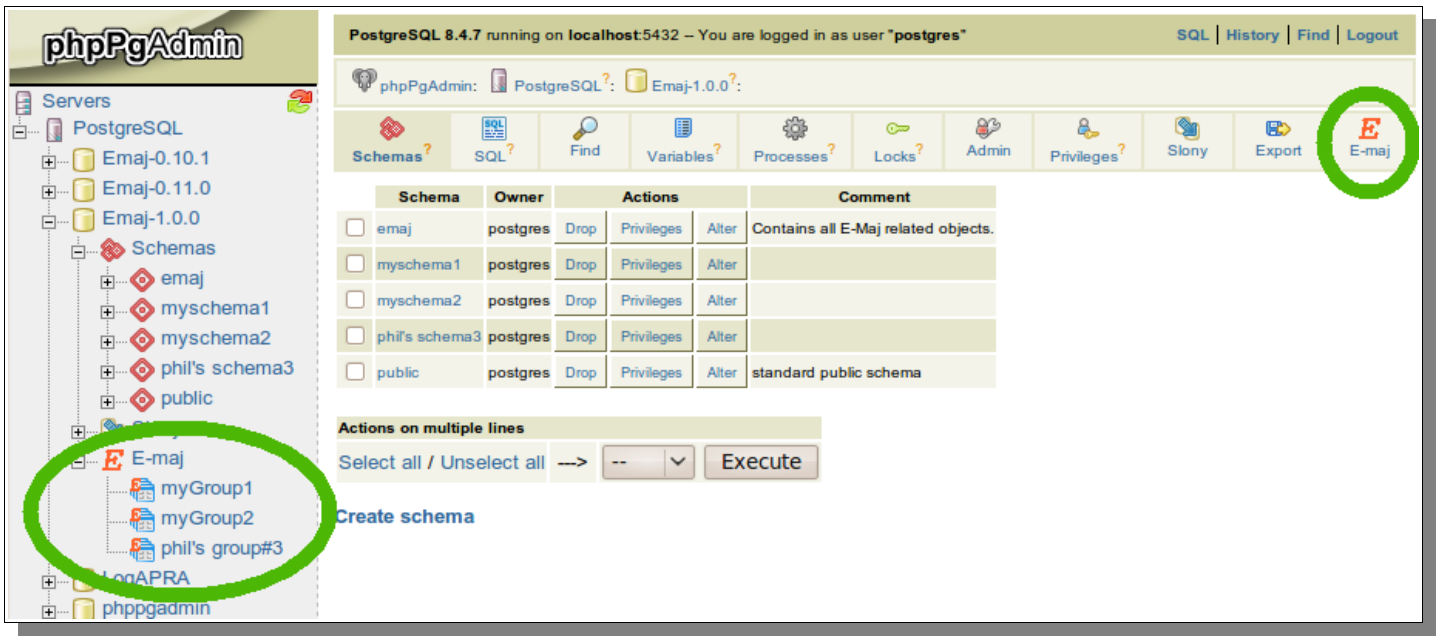


Figure 1 – Connexion à une base de données où E-Maj est installé.

6.2.2 Liste des groupes de tables

En cliquant sur l'une des icônes E-Maj, l'utilisateur accède à une page qui liste les groupes de tables créés sur cette base de données (voir la figure 2 ci-dessous).

En fait, deux listes sont affichées : la première présente les groupes de tables en état « *démarrés* » et la seconde les groupes de tables « *arrêtés* ».

En dessous, une liste déroulante présente les groupes de tables susceptibles d'être créés (ceuxréférencés dans la table *emaj_group_def* mais qui ne sont pas encore créés).

Enfin, un lien permet à l'utilisateur de définir ou de modifier la composition des groupes de tables.

Toutes les pages affichées par le plugin E-Maj ont une entête qui contiennent :

- un bouton pour rafraichir la page courante,
- l'heure d'affichage de la page courante,
- la version d'E-Maj installée dans la base de données, un clic sur ce champ vérifiant l'état de l'environnement E-Maj,
- la place disque occupée par le tablespace d'E-Maj et la part qu'il représente dans la place totale prise par la base de données,
- le titre de la page.

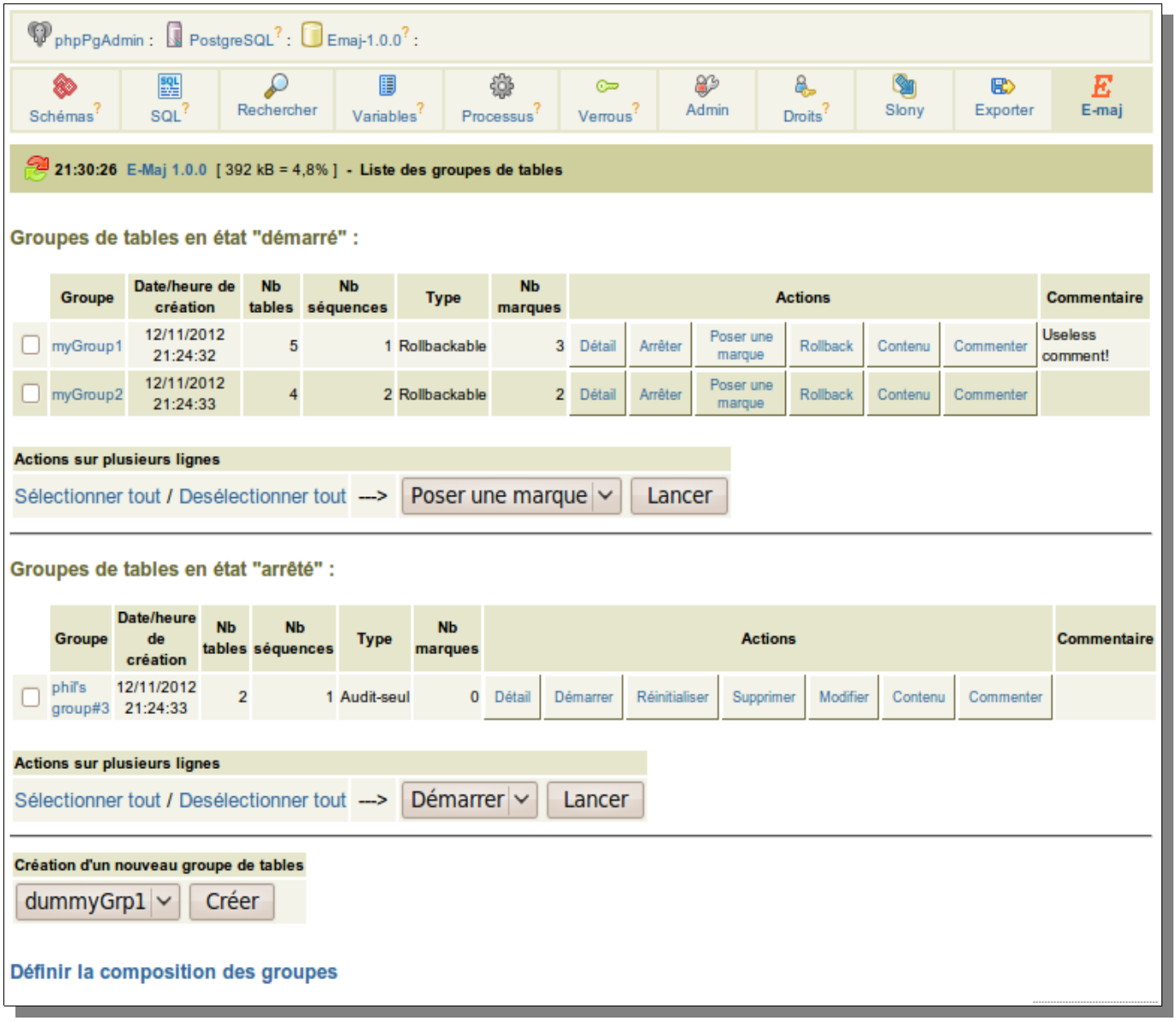


Figure 2 – Liste des groupes de tables créés sur la base de données, avec ici deux groupes en état « démarré » et un groupe en état « arrêté ».

Pour chaque groupe de tables, sont affichés les attributs suivants :

- sa date et son heure de création,
- le nombre de tables et de séquences applicatives qu'il contient,
- son type (« *ROLLBACKABLE* » ou « *AUDIT-SEUL* »),
- le nombre de marques qu'il possède
- son éventuel commentaire associé.

Plusieurs boutons sont proposés afin de pouvoir effectuer les actions que son état rend possible.

Sous chaque liste, une liste déroulante et un bouton permettent d'effectuer certaines actions sur plusieurs groupes simultanément.

6.2.3 Composition des groupes de tables

Par un clic sur le lien figurant en bas de la page des listes de groupes de tables, l'utilisateur atteint la fonction qui gère la composition des groupes de tables.

La partie supérieure de la page liste les schémas existants dans la base de données (à l'exception des schémas dédiés à E-Maj). En sélectionnant un schéma, la liste de ses tables et séquences apparaît.

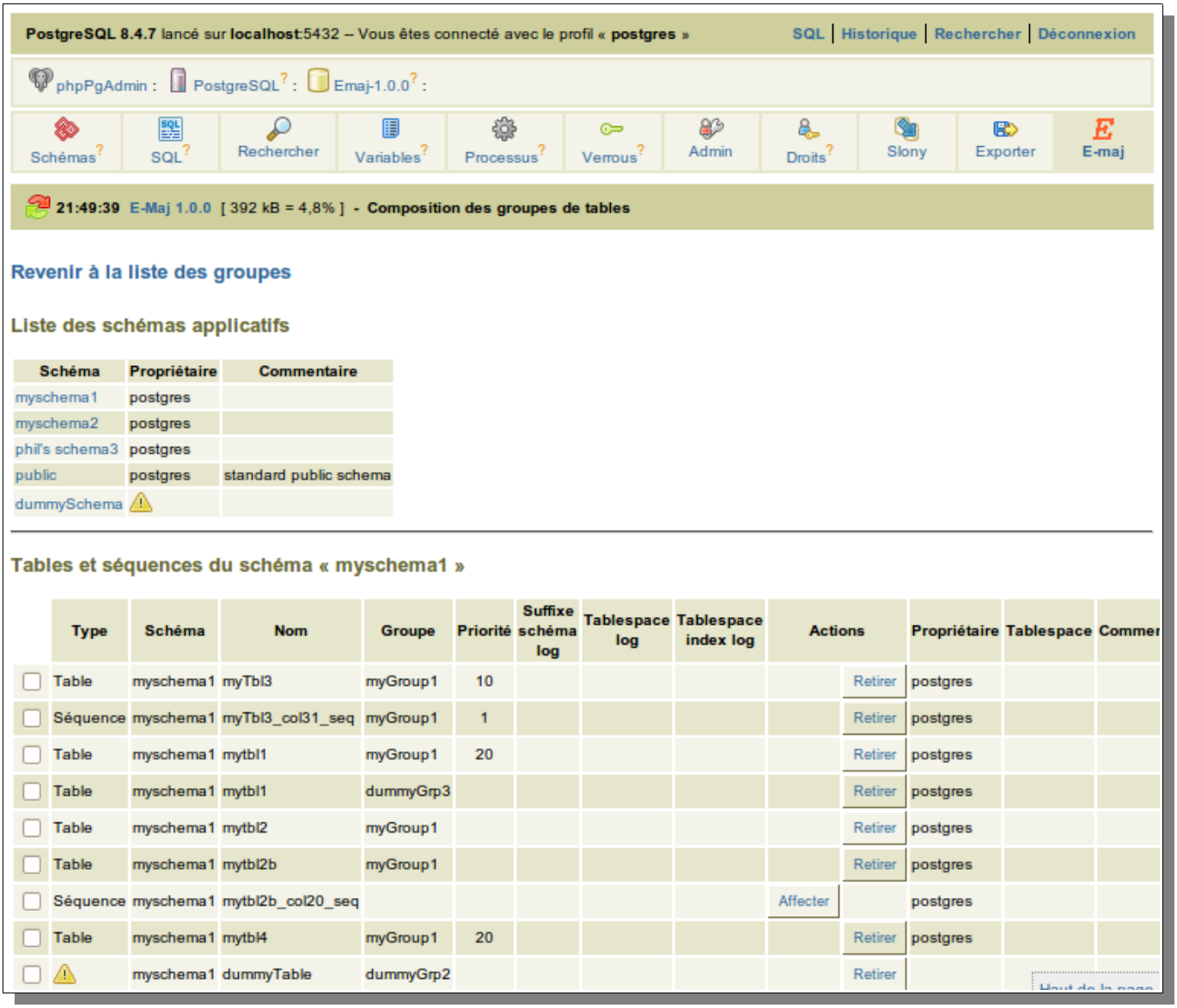


Figure 3 – Composition des groupes de tables.

Il est alors possible de voir ou de modifier le contenu de la table *emaj_group_def* utilisée pour la création du groupe de tables (fonction *emaj_create_group()*).

Sont listés pour chaque table ou séquence :

- le groupe de table auquel il appartient, s'il y en a un,

- les attributs de la table ou séquence dans *emaj_group_def*, si elle est déjà affectée à un groupe (voir §4.2.2) :
  - le niveau de priorité affecté dans le groupe,
  - le suffixe définissant le schéma de log
  - le nom du tablespace éventuel supportant la table de log
  - le nom du tablespace éventuel supportant l'index de la table de log
- l'identité du propriétaire de la table ou de la séquence,
- le tablespace auquel elle est rattaché, s'il y en a un,
- son commentaire enregistré dans la base de données.

Les deux listes de schémas et de tables et séquences référencent également les objets déjà référencés dans la table emaj_group_def mais qui n'existe pas dans la base de données. Ces objets sont identifiés par une icône « ! » dans la colonne « *propriétaire* ».

A l'aide d'un bouton, il est possible soit d'assigner la table ou la séquence à un groupe de tables nouveau ou existant, soit de la détacher du groupe de tables auquel elle est déjà assignée.

Notons que les modifications apportées au contenu de la table *emaj_group_def* ne prendront effet que que lorsque les groupes de tables concernés seront soit modifiés, soit supprimés puis recréés.

6.2.4 Détail d'un groupe de tables

Depuis la page listant les groupes de tables, il est possible d'en savoir davantage sur un groupe de tables particulier en cliquant sur son nom ou sur son bouton « Détail ».

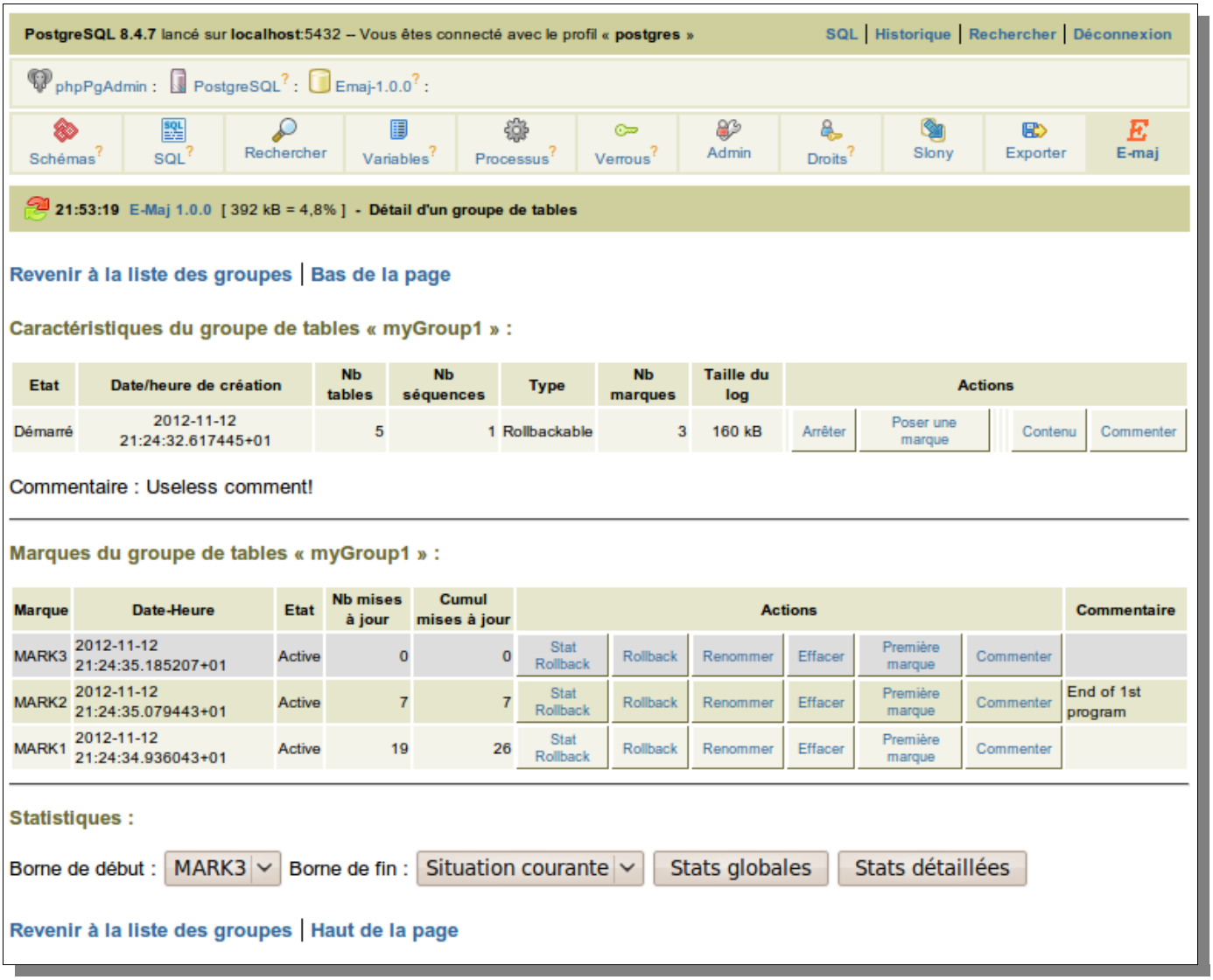


Figure 4 – Détail d'un groupe de tables

Une première ligne reprend des informations déjà affichées sur le tableau des groupes (nombre de tables et de séquences, type et nombre de marques), complété par l' espace disque utilisé par les tables de log et un ensemble de boutons permettant de réaliser les actions que l'état du groupe permet.

Cette ligne est suivie par l'éventuel commentaire associé au groupe.

L'utilisateur trouve ensuite un tableau des marques positionnées pour le groupe. Pour chacune d'elles, on trouve :

- son nom,
- sa date et son heure de pose,
- son état,
- le nombre de lignes de log enregistrées entre cette marque et la suivante (ou la situation courante s'il s'agit de la dernière marque),
- le nombre total de lignes de log enregistrées depuis que la marque a été posée,
- l'éventuel commentaire associé à la marque.

Plusieurs boutons permettent d'exécuter toute action que son état permet.

Enfin une dernière ligne offre la possibilité d'obtenir des statistiques globales ou détaillées entre deux marques ou bien entre une marque et la situation courante.

6.2.5 Statistiques pour les rollbacks

La figure suivante montre la page obtenue en cliquant sur le bouton « Stat Rollback » d'une marque.

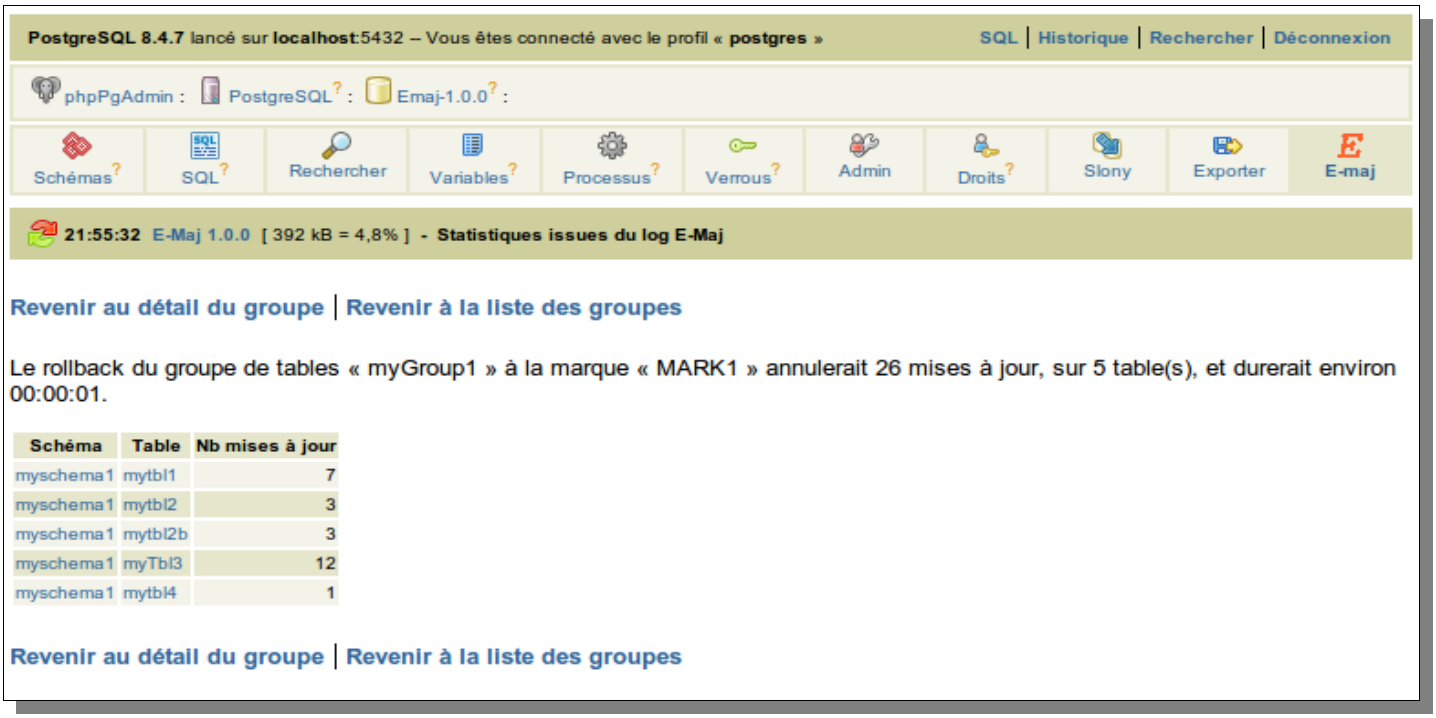


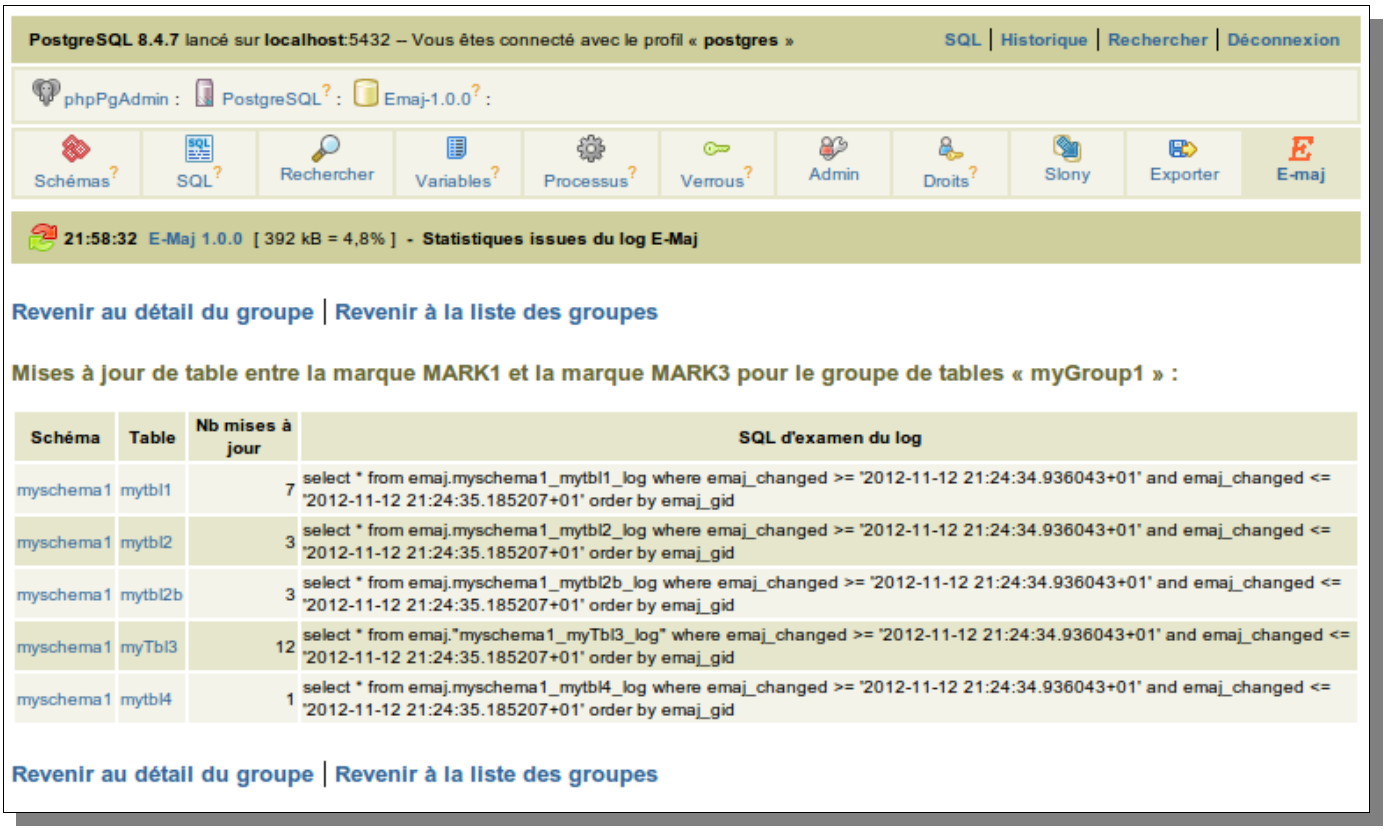
Figure 5 – Statistiques avant rollback

La page restituée contient une première ligne indiquant le nombre de lignes de log concernées par un éventuel rollback à cette marque et une estimation du temps nécessaire à ce rollback.

Ensuite, on trouve un tableau qui présente le détail table par table du nombre de lignes de log à traiter.

6.2.6 Statistiques sur le contenu des tables de log

Les statistiques globales ou détaillées sur le contenu des tables de log sont obtenues à partir de la page détail d'un groupe de tables.



PostgreSQL 8.4.7 lancé sur localhost:5432 – Vous êtes connecté avec le profil « postgres » SQL | Historique | Rechercher | Déconnexion

phpPgAdmin : PostgreSQL : Emaj-1.0.0 :

Schémas | SQL | Rechercher | Variables | Processus | Verrous | Admin | Droits | Slony | Exporter | E-maj

21:58:32 E-Maj 1.0.0 [392 kB = 4,8%] - Statistiques issues du log E-Maj

Revenir au détail du groupe | Revenir à la liste des groupes

Mises à jour de table entre la marque MARK1 et la marque MARK3 pour le groupe de tables « myGroup1 » :

Schéma	Table	Nb mises à jour	SQL d'examen du log
myschema1	mytbl1	7	select * from emaj.myschema1_mytbl1_log where emaj_changed >= '2012-11-12 21:24:34.936043+01' and emaj_changed <= '2012-11-12 21:24:35.185207+01' order by emaj_gid
myschema1	mytbl2	3	select * from emaj.myschema1_mytbl2_log where emaj_changed >= '2012-11-12 21:24:34.936043+01' and emaj_changed <= '2012-11-12 21:24:35.185207+01' order by emaj_gid
myschema1	mytbl2b	3	select * from emaj.myschema1_mytbl2b_log where emaj_changed >= '2012-11-12 21:24:34.936043+01' and emaj_changed <= '2012-11-12 21:24:35.185207+01' order by emaj_gid
myschema1	myTbl3	12	select * from emaj."myschema1_myTbl3_log" where emaj_changed >= '2012-11-12 21:24:34.936043+01' and emaj_changed <= '2012-11-12 21:24:35.185207+01' order by emaj_gid
myschema1	mytbl4	1	select * from emaj.myschema1_mytbl4_log where emaj_changed >= '2012-11-12 21:24:34.936043+01' and emaj_changed <= '2012-11-12 21:24:35.185207+01' order by emaj_gid

Revenir au détail du groupe | Revenir à la liste des groupes

Figure 6 – Statistiques globales avec des mises à jour concernant une table applicative.

Les statistiques affichées comprennent :

- l'identité de la table (noms du schéma et de la table),
- le nombre de mises à jour enregistrées entre les deux marques spécifiées,
- une requête SQL qui, par copier/coller, permet d'afficher le contenu de la table de log correspondant à cet intervalle de temps.

Pour les statistiques détaillées, deux colonnes supplémentaires sont affichées :

- le rôle de connexion qui a effectué les mises à jour,
- le type de verbe SQL (*insert*/*update*/*delete*).

7 ANNEXES

7.1 LISTE DES FONCTIONS E-MAJ

Les fonctions E-Maj disponibles pour les utilisateurs sont listées ci-dessous par ordre alphabétique. Toutes ces fonctions sont appelables par les rôles disposant des privilèges *emaj_adm*. Le tableau précise celles qui sont également appelables par les rôles *emaj_viewer*.

Fonctions	Paramètres	Retour	Exécutable par emaj_viewer	Réf.
emaj_alter_group	groupe TEXT	nb.tables.et.séq INT		§ 4.2.10
emaj_comment_group	groupe TEXT commentaire TEXT	-		§ 4.7.2
emaj_comment_mark_group	groupe TEXT marque TEXT commentaire TEXT	-		§ 4.4.1
emaj_create_group	groupe TEXT [est.rollbackable BOOLEAN]	nb.tables.et.séq INT		§ 4.2.3
emaj_delete_before_mark_group	groupe TEXT marque TEXT	nb.marques.suppr INT		§ 4.4.5
emaj_delete_mark_group	groupe TEXT marque TEXT	1 INT		§ 4.4.4
emaj_detailed_log_stat_group	groupe TEXT marque.début TEXT marque.fin TEXT	SETOF emaj_detailed_log_stat_type	Oui	§ 4.5.2
emaj_drop_group	groupe TEXT	nb.tables.et.séq INT		§ 4.2.9
emaj_estimate_rollback_duration	groupe TEXT marque TEXT	durée INTERVAL	Oui	§ 4.5.3
emaj_force_drop_group	groupe TEXT	nb.tables.et.séq INT		§ 4.7.5
emaj_force_stop_group	groupe TEXT	nb.tables.et.séq INT		§ 4.7.4
emaj_generate_sql	groupe TEXT marque.début TEXT marque.fin TEXT chemin.fichier.sortie TEXT	nb.requêtes.gén. INT		§ 4.6.3
emaj_get_previous_mark_group	groupe TEXT date.heure TIMESTAMPTZ	marque TEXT	Oui	§ 4.4.2
emaj_get_previous_mark_group	groupe TEXT marque TEXT	marque TEXT	Oui	§ 4.4.2

Fonctions	Paramètres	Retour	Exécutable par emaj_viewer	Réf.
emaj_log_stat_group	groupe TEXT marque.début TEXT marque.fin TEXT	SETOF emaj_log_stat_type	Oui	§ 4.5.1
emaj_logged_rollback_group	groupe TEXT marque TEXT	nb.tables.et.séq.traitées INT		§ 4.2.7
emaj_logged_rollback_groups	tableau.groupes TEXT[] marque TEXT	nb.tables.et.séq.traitées INT		§ 4.3.2
emaj_rename_mark_group	groupe TEXT marque TEXT nouveau.nom TEXT	-		§ 4.4.3
emaj_reset_group	groupe TEXT	nb.tables.et.séq INT		§ 4.7.1
emaj_rollback_group	groupe TEXT marque TEXT	nb.tables.et.séq.traitées INT		§ 4.2.6
emaj_rollback_groups	tableau.groupes TEXT[] marque TEXT	nb.tables.et.séq.traitées INT		§ 4.3.2
emaj_set_mark_group	groupe TEXT marque TEXT	nb.tables.et.séq INT		§ 4.2.5
emaj_set_mark_groups	tableau.groupes TEXT[] marque TEXT	nb.tables.et.séq INT		§ 4.3.2
emaj_snap_group	groupe TEXT répertoire TEXT option.copy TEXT	nb.tables.et.séq INT		§ 4.6.1
emaj_snap_log_group	groupe TEXT marque.début TEXT marque.fin TEXT répertoire TEXT option.copy TEXT	nb.tables.et.séq INT		§ 4.6.2
emaj_start_group	groupe TEXT marque TEXT [reset.loge BOOLEAN]	nb.tables.et.séq INT		§ 4.2.4
emaj_start_groups	tableau.groupes TEXT[] marque TEXT [reset.loge BOOLEAN]	nb.tables.et.séq INT		§ 4.3.2
emaj_stop_group	groupe TEXT [marque TEXT]	nb.tables.et.séq INT		§ 4.2.8
emaj_stop_groups	tableau.groupes TEXT[] [marque TEXT]	nb.tables.et.séq INT		§ 4.3.2
emaj.emaj_verify_all	-	Setof TEXT	Oui	§ 4.7.3