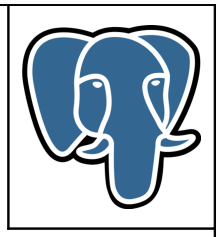


Extension PostgreSQL « E-Maj »

-

Manuel d'utilisation

Version 0.10.0

Mise à jour le jeu. 3 novembre 2011

Table des matières

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 en détail la mise en place et la façon d'utiliser E-Maj.

Le chapitre 4 apporte quelques compléments nécessaires à la bonne compréhension du fonctionnement de l'extension.

Enfin, le chapitre 5 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

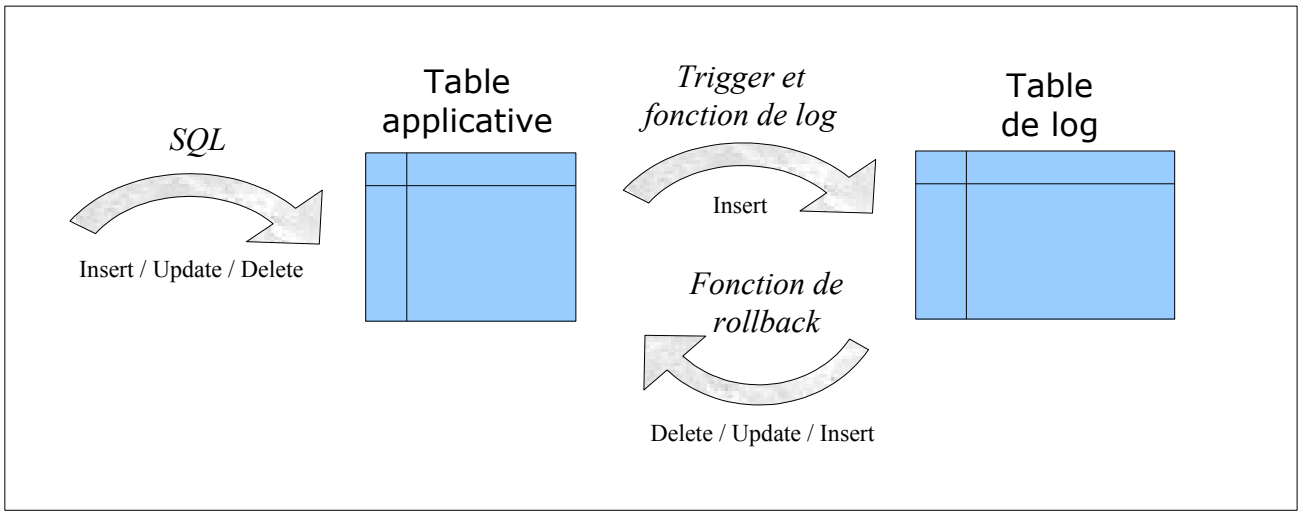
Pour mener à bien l'opération de rollback sans avoir au préalable conservé une image physique des fichiers du cluster PostgreSQL, il faut pouvoir enregistrer les mises à jour effectuées sur les tables applicatives de manière à pouvoir les annuler.

Avec E-Maj, cela prend la forme suivante.

2.2.1 Les 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*), 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 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,
- 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 (*ctid* 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,
- 2 types,
- 61 fonctions techniques, dont 29 directement appelables par les utilisateurs,
- 1 schéma spécifique, nommé *emaj*, qui contient tous ces objets ainsi que les tables de log et fonctions associées,
- 1 tablespace, *tspemaj*, supportant toutes les tables (tables techniques et tables de log),
- 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.

Certaines tables techniques, dont il est utile de connaître la structure, sont décrites dans les chapitres suivants.

2.2.2 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_emaj_id_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.

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 le site Internet pgFoundry.org à l'url 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, en exécutant, par exemple sous Unix/Linux, une commande du type :

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

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.control_base — fichier de contrôle de base pour l'extension
- sql/emaj--0.10.0.sql — script pour la requête CREATE EXTENSION
- sql/emaj--unpackaged--0.10.0.sql — script de transformation en extension d'une version installée d'E-Maj
- 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 de l'extension E-Maj
- php/emajParallelRollback.php — outil de rollback parallélisé

3.2 MISE EN PLACE DE L'EXTENSION E-MAJ

Si une version d'E-Maj est déjà installée dans la base de données, allez au chapitre §3.2.2.

3.2.1 Installation complète

Pour l'installation proprement dite, il faudra distinguer la version de PostgreSQL utilisée, les versions 9.1 et suivantes bénéficiant de l'architecture intégrée des extensions. Mais quelques opérations préliminaires sont requises.

3.2.1.1 Opérations préliminaires

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

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 consiste à créer un tablespace nommé *tspemaj* et dédié à E-Maj, sauf à ce que ce tablespace ait déjà été créé par une installation précédente.

Pour ce faire, 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.

3.2.1.2 Installation des composants E-Maj avec les versions de PostgreSQL 8.2 à 9.0

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

```
\i <emaj_directory>/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, et que le tablespace *tspemaj* est bien créé.

Le script crée alors le schéma *emaj* avec ses 10 tables techniques, ses 2 types et ses 61 fonctions.

S'ils n'existaient 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 §4.7).

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

>>> E-Maj objects successfully created

3.2.1.3 Installation des composants E-Maj avec les versions de PostgreSQL 9.1 et suivantes

A partir de PostgreSQL 9.1, il est possible d'installer E-Maj comme une *extension*, au sens de PostgreSQL.

Une première étape consiste à fournir à PostgreSQL quelques paramètres concernant l'extension E-Maj. Concrètement, il faut créer, dans le répertoire d'installation de PostgreSQL, un fichier :

share/postgresql/extension/emaj.control

en partant du fichier *sql/emaj.control_base* fourni avec E-Maj.

Dans ce fichier, il suffit de valoriser le paramètre *directory* avec le chemin du répertoire qui contient les scripts d'installation (typiquement le répertoire */sql* issu de la décompression d'E-Maj). Les autres paramètres doivent rester inchangés.

Ensuite, il ne reste plus qu'à créer l'*extension* avec la requête :

```
CREATE EXTENSION emaj;
```

Notons que l'extension E-Maj est toujours installée dans le schéma *emaj*. Il n'est pas possible de spécifier un autre nom de schéma dans la requête *CREATE EXTENSION*.

Si E-Maj 0.10.0 a été installé avec le script *emaj.sql*, il est possible par la suite de transformer l'installation en *extension* au sens PostgreSQL. Pour ce faire, il suffit d'exécuter la requête :

```
CREATE EXTENSION emaj FROM unpackaged;
```

Une fois installé sous forme d'*extension*, E-Maj peut être désinstallé par une requête du type *DROP EXTENSION*.

3.2.2 Mise à jour d'une version existante

Les différences entre la version 0.10.0 et les versions précédentes sont trop importantes pour qu'une simple migration soit intéressante. C'est pourquoi, pour migrer d'une version antérieure à la version 0.10.0, il faut supprimer puis réinstaller E-Maj. Néanmoins, il n'est pas nécessaire de procéder à une désinstallation complète, les tablespaces et les rôles pouvant rester en l'état.

3.2.2.1 Sauvegarde des données utilisateurs

Il peut 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.savegroupdef 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.saveparam AS SELECT * FROM emaj.emaj_param;
```

3.2.2.2 Réinstallation d'E-Maj

Avec les versions de PostgreSQL de 8.2 à 9.0, il suffit d'exécuter le script *emaj.sql* tel qu'indiqué au §3.2.1.2 pour installer la nouvelle version 0.10.0 d'E-Maj, le script supprimant la version précédente.

Avec les versions de PostgreSQL 9.1 et suivantes, il faut tout d'abord supprimer le schéma *emaj* :

```
DROP SCHEMA emaj CASCADE;
```

Ceci va supprimer l'*extension*. Suivre ensuite la précédure indiquée dans le §3.2.1.3.

3.2.2.3 Restauration des données utilisateurs

Les données sauvegardées au préalable peuvent alors être restaurées dans les tables E-Maj.

La requête d'insertion des données dans la table *emaj_group* doit tenir compte que la nouvelle table contient une colonne supplémentaire, qu'il est possible de valorisée avec des valeurs *NULL* (voir §4.2.2).

Par ailleurs, dans la nouvelle table *emaj_param*, les paramètres qui restent avec leur valeur par défaut n'ont plus de ligne correspondante. Il ne faut donc insérer dans cette table que les paramètres dont la valeur est différente de la valeur par défaut (voir §5.1).

3.2.3 Adaptation du fichier de configuration postgresql.conf

Les principales fonctions 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*. La valeur par défaut de ce paramètre est de 64. On peut le porter à une valeur au moins égale au nombre de tables composant le groupe le plus grand.

De plus, si l'utilisation de l'outil de rollback en parallèle est envisagée (voir § 4.7), 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, toujours sous psql, en exécutant le script de démonstration fourni avec l'extension *demo.sql*.

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

Si aucune erreur n'est rencontrée, le message final

"--- *End of E-Maj demo* ---"

est affiché.

3.3 DÉSINSTALLATION D'E-MAJ

Pour complètement désinstaller E-Maj, l'utilisateur doit se connecter avec psql à la base de données concernée en tant que super-utilisateur.

Si certains groupes de tables sont encore actifs, il faut les arrêter à l'aide de la fonction *emaj_stop_group()* (voir § 4.2.8).

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*.

Il suffit ensuite d'exécuter le script *uninstall.sql* fourni avec la version d'E-Maj installée.

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

Ce script supprime le schéma *emaj* et tout ce qu'il contient. Si E-Maj a été installée sous forme d'*EXTENSION*, au sens PostgreSQL, celle-ci se trouve supprimée,

Si les rôles *emaj_adm* et *emaj_viewer* ne sont plus associés à d'autres rôles et ne possèdent pas de droits sur d'autres tables, ils sont supprimés. Sinon, un simple message est affiché, invitant à une suppression manuelle, si on le désire.

En revanche, le tablespace *tspemaj* n'est pas supprimé par le script. Si on souhaite le supprimer, il suffit, pour terminer, d'exécuter la requête SQL suivante :

```
DROP TABLESPACE tspemaj;
```

Si E-Maj est installé dans plusieurs bases de données du même cluster PostgreSQL, alors le tablespace *tspemaj* est utilisé par les objets E-Maj de toutes les bases. On ne pourra donc le supprimer qu'après la suppression des composants E-Maj de toutes les bases de données du cluster.

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 [1] 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*.

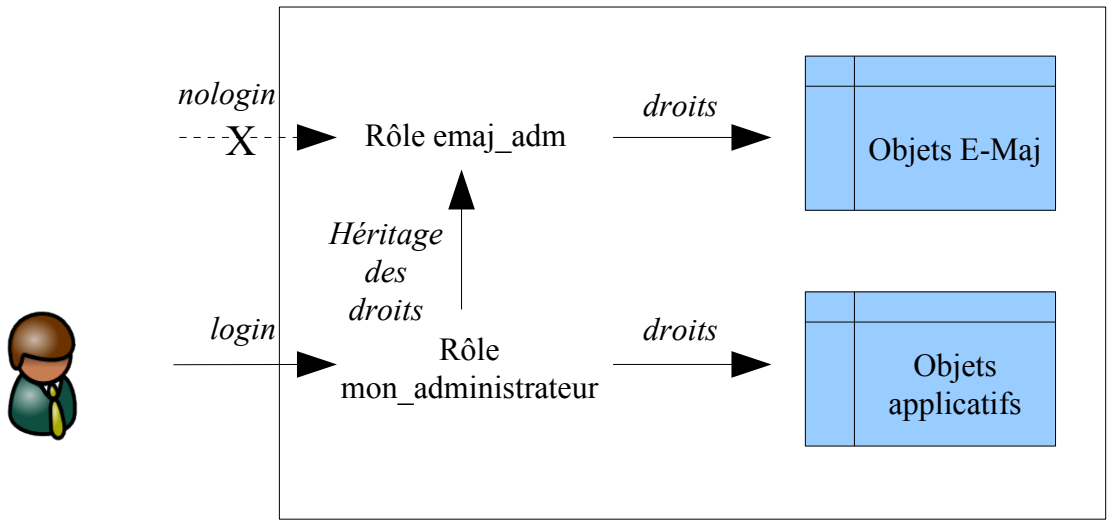
[1] à l'exception de la fonction *emaj_snap_group()* (voir § 4.6.6) qui nécessite d'être exécutée par un super-utilisateur.

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 röles *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*.

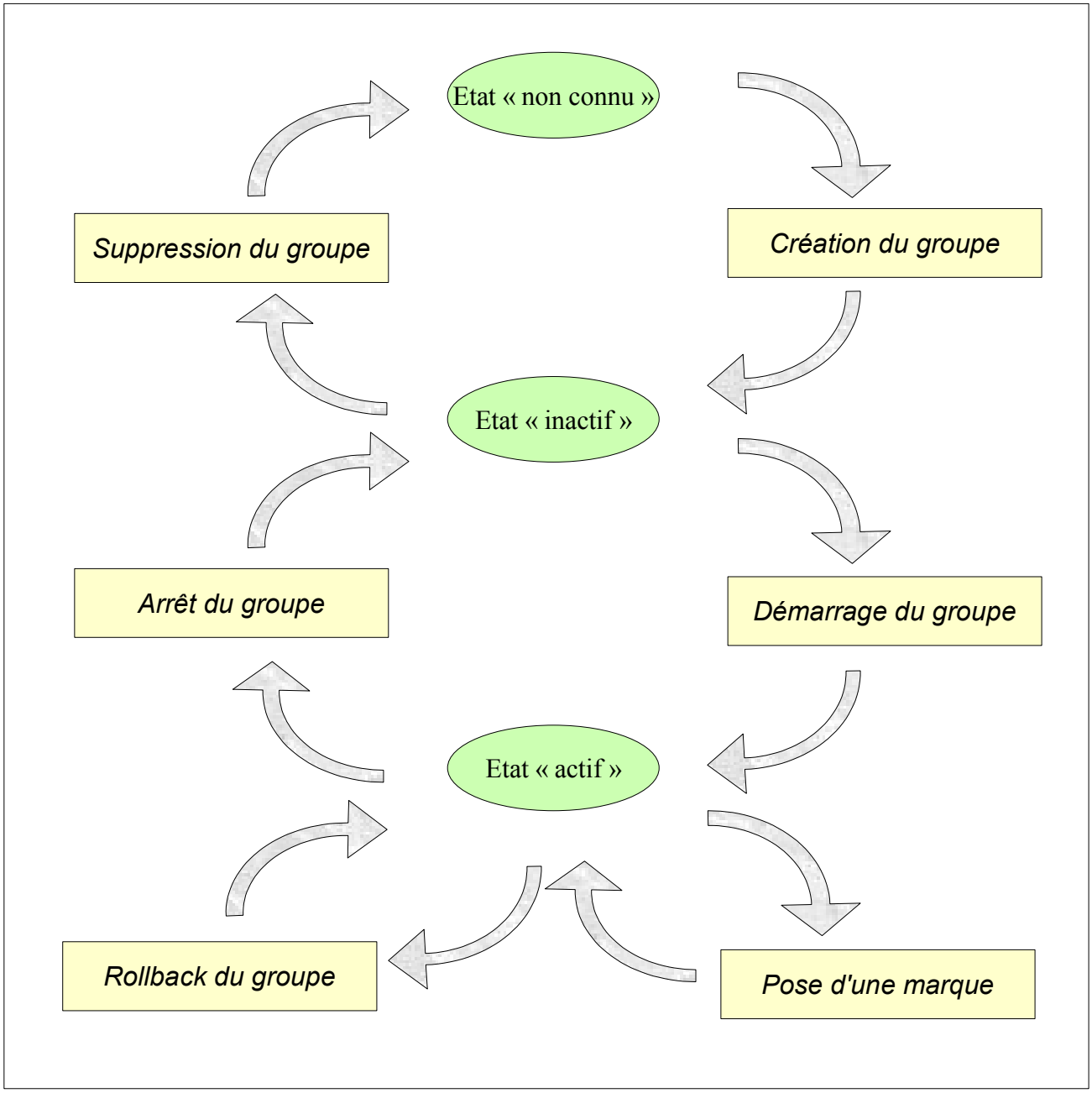
Sauf indication contraire, les opérations qui suivent vont pouvoir être exécutées indifféremment avec un rôle super-utilisateur ou un rôle du groupe *emaj_adm*.

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

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* comprend 3 colonnes de type textuel et 1 colonne numérique :

- *grpdef_group* : nom du groupe de tables
- *grpdef_schema* : nom du schéma contenant la table ou la séquence applicative
- *grpdef_tblseq* : nom de la table ou de la séquence applicative
- *grpdef_priority* : niveau de priorité de la table ou de la séquence dans les traitements E-Maj

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

Un nom de groupe de tables 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 ne peut pas être affecté 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 !

La colonne *grpdef_priority* est de type entière (*INTEGER*) et peut prendre la valeur nulle, 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.

Le contenu de la table *emaj_group_def* est sensible à la casse. Les noms de schéma, de tables et de séquences 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*.

4.2.3 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 *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.

Toutes les actions enchaînées par la fonction *emaj_create_group()* vont être 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 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>');
```

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.

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 *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.

Avant d'activer les triggers de log qui vont permettre ensuite l'enregistrement des mises à jour sur les tables du groupe, les tables de log sont purgées de toutes anciennes données.

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.

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 toujours être en état démarré (*LOGGING*).

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 impliquant une table du groupe n'est en cours ni qu'aucune mise à jour ne pourra ensuite être effectuée sur les tables 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 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.

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 §5.7.3).

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*).

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 impliquant une table du groupe n'est en cours ni qu'aucune mise à jour ne pourra ensuite être effectuée sur les tables du groupe pendant toute la durée du rollback, la fonction *emaj_logged_rollback_group()* pose explicitement un verrou de type *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.

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 §15.7.3).

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>');
```

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

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.

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 Changement de composition d'un groupe de tables

La composition d'un groupe de tables peut évoluer dans le temps. Il peut être en effet nécessaire d'ajouter ou de supprimer d'un groupe une table ou une séquence. Dans ce cas, il faut enchaîner les opérations suivantes :

- arrêter le groupe s'il est dans un état actif, avec la fonction *emaj_stop_group()*,
- supprimer le groupe avec la fonction *emaj_drop_group()*,
- adapter le contenu de la table *emaj_group_def* pour refléter l'évolution souhaitée,
- recréer le groupe avec la fonction *emaj_create_group()*.

Il est possible d'anticiper la mise à jour de la table *emaj_group_def*, même si le groupe de tables est encore actif.

4.2.11 Changement de structure d'une table applicative

Si une table applicative appartenant à un groupe de tables voit sa structure évoluer (ajout ou suppression de colonnes, changement de type de colonne), il est impératif de supprimer le groupe auquel elle appartient avec la fonction *emaj_drop_group()*, puis de le recréer avec la fonction *emaj_create_group()*. En effet, ces évolutions vont avoir un impact sur la structure de la table de log associée.

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 cinq fonctions suivantes traitent plusieurs groupes :

- ➢ *emaj.emaj_start_groups(<tableau.de.groupes>,<marque.de.début>)* démarre plusieurs groupes,
- ➢ *emaj.emaj_stop_groups(<tableau.de.groupes>)* 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).

Il est également autorisé que la liste comporte des doublons, des valeurs NULL ou des chaînes vides. Dans tous ces cas, ces valeurs sont ignorées et un message d'avertissement est généré, comme dans le cas où la liste est vide.

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>');
```

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 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.

4.4.3 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.

4.4.4 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>');
```

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.

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 tablespace tspemaj.

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.

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 ou NULL>', '<marque.fin ou NULL>');
```

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 ou NULL>', '<marque.fin ou NULL>');
```

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*).

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

- 4 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.

4.5.4 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 une date et une heure donnée.

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

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 l'heure d'une marque existante, la marque retournée sera la marque précédente.

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));
```

4.6 AUTRES FONCTIONS

4.6.1 Rollback et arrêt simultanés d'un groupe de tables

Si l'administrateur E-Maj veut effectuer un rollback sur un groupe de table puis arrêter ce même groupe immédiatement après, il est possible de regrouper les actions effectuées par les deux fonctions *emaj_rollback_group()* et *emaj_stop_group()* en une seule opération.

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

La fonction retourne le nombre de tables et de séquences effectivement modifiées par l'opération de rollback, à l'instar de la fonction *emaj_rollback_group()*.

L'utilisation de cette fonction *emaj_rollback_and_stop_group()* à la place des deux fonctions habituelles permet de gagner un peu de temps. En effet, le groupe étant arrêté, aucun rollback ne pourra être effectué à l'issue de la commande. En conséquence, la fonction *emaj_rollback_and_stop_group()* n'efface pas le contenu des tables de log correspondant aux mises à jour annulées. Ceci permet donc un gain de temps qui pourra être d'autant plus significatif que le rollback annule un grand nombre de mises à jour.

4.6.2 Réinitialisation des tables de log d'un groupe

En standard, les tables de log sont vidées lors du démarrage du groupe de tables concerné. 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 dans un état inactif.

4.6.3 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.

4.6.4 Vérification de la consistance des objets E-Maj

Une fonction permet de vérifier la consistance entre d'une part tous les objets E-Maj présents dans le schéma *emaj* et d'autre part les tables et séquences référencées dans les groupes de tables créés. Cette fonction s'exécute par la requête SQL suivante :

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

La fonction vérifie que :

- chaque table de log existante dans le schéma *emaj* correspond bien à une table référencée dans *emaj_group*, la table interne qui décrit la composition des groupes créés,
- chaque fonction de log ou de rollback existante dans le schéma *emaj* correspondant bien à une table référencée dans cette même table *emaj_group*.

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

'*No error encountered*'

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.

4.6.5 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.

4.6.6 Vidage des tables d'un groupe

Pour tester l'extension E-Maj, et en particulier les fonctions de rollback, on peut avoir besoin de prendre des images de toutes les tables appartenant à un groupe, afin de pouvoir les comparer. Une fonction permet d'obtenir le vidage des tables d'un groupe :

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

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.

Le nom du répertoire fourni doit être un chemin absolu. Ce répertoire doit avoir les permissions adéquates pour que le cluster postgreSQL puisse y écrire.

Le nom des fichiers créés est du type :
<nom.du.schema>_<nom.de.table/séquence>.snap

Comme les fichiers sont créés par des commandes *COPY*, il est impératif que l'utilisateur qui utilise la fonction *emaj_snap_group()* soit connecté en tant que super-utilisateur. Un rôle du groupe *emaj_adm* ne peut pas exécuter cette fonction.

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 par défaut utilisé par la commande *COPY*. Ces enregistrements sont triés dans l'ordre croissant de la clé primaire.

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.

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.

4.7 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.

4.7.1 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.

4.7.2 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.

4.7.3 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.

4.7.4 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 :

- *param_key* mot-clé identifiant le paramètre
- *param_value_text* valeur du paramètre, s'il est de type texte (sinon NULL)
- *param_value_int* valeur du paramètre, s'il est de type entier (sinon NULL)
- *param_value_boolean* valeur du paramètre, s'il est de type booléen (sinon NULL)
- *param_value_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_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

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

Ces contrôles d'intégrité du groupe de tables portent sur la vérification 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 et 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 :

- *hist_id* numéro de série identifiant une ligne dans cette table historique
- *hist_datetime* date et heure d'enregistrement de la ligne
- *hist_function* fonction associée à l'événement
- *hist_event* type d'événement
- *hist_object* nom de l'objet sur lequel porte l'événement (groupe, table ou séquence)
- *hist_wording* commentaires complémentaires
- *hist_user* rôle à l'origine de l'événement
- hist_txid 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
- *FORCE_DROP_GROUP* suppression forcée d'un groupe de tables
- *START_GROUP(S)* démarrage d'un groupe de tables
- *STOP_GROUP(S)* arrêt d'un groupe de tables
- *LOCK_GROUP(S)* pose d'un verrou exclusif sur les tables d'un groupe
- *LOCK_SESSION* pose d'un verrou exclusif sur les tables d'une session
- *SET_MARK_GROUP(S)* pose d'une marque pour un groupe 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(S)* rollback des mises à jour pour un groupe 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

La colonne *hist_event* prend les valeurs suivantes :
- *BEGIN* début
- *END* fin
- *MARK DELETED* marque supprimée

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()*), les lignes les plus anciennes de la table *emaj_hist* sont supprimées. Par défaut, la durée de rétention des lignes dans la table 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).

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 fichiers des clusters PostgreSQL !

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*.

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 Réorganisation des tables de la base de données

Aucune table du schéma *emaj* ne possède d'index « cluster ». 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.

5.4.4 Utilisation d'E-Maj avec de la réplication

E-Maj est parfaitement compatible avec le fonctionnement des différents mode de réplication interne 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.

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.

5.4.5 Changement de version de PostgreSQL

5.4.5.1 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 §3.3 et §3.2). Il est donc impossible de conserver des groupes de tables actifs pendant le changement de version PostgreSQL.

5.4.5.2 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 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 nombre maximum de lignes qu'une table de log peut contenir n'est pas totalement illimité, bien que cette limite soit très grande (4.611.686.018.427.387.903 !).
- 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.
- Le tablespace nommé "*tspemaj*" et contenant toutes les tables du schéma *emaj* doit être créé avant l'installation de l'extension. Si son nom devait être changé, le script *emaj.sql* devrait être adapté 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.

- 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 d'une opération de 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 d'erreur lors de l'enregistrement des mises à jour suivantes. En revanche, de par les contrôles propres à E-Maj, toute tentative 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 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.

5.7 RESPONSABILITÉS DE L'UTILISATEUR

5.7.1 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.

5.7.2 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.

5.7.3 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.

5.7.4 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

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),
- voir la liste des marques posées pour un groupe de tables et effectuer toutes les actions possibles (suppression, renommage),
- 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.

En fait, 2 listes sont affichées : la première présente les groupes de tables en état « démarrés » et la seconde les groupes de tables « arrêtés ».

Si des groupes sont référencés dans la table *emaj_group_def* mais ne sont pas encore créés, la dernière partie de la page permet à l'utilisateur de les créer.

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

- un bouton pour rafraichir la page courante,
- la version d'E-Maj installée dans la base de données,
- 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.

La figure 2 ci-dessous montre une liste comprenant 2 groupes en état « démarré ».

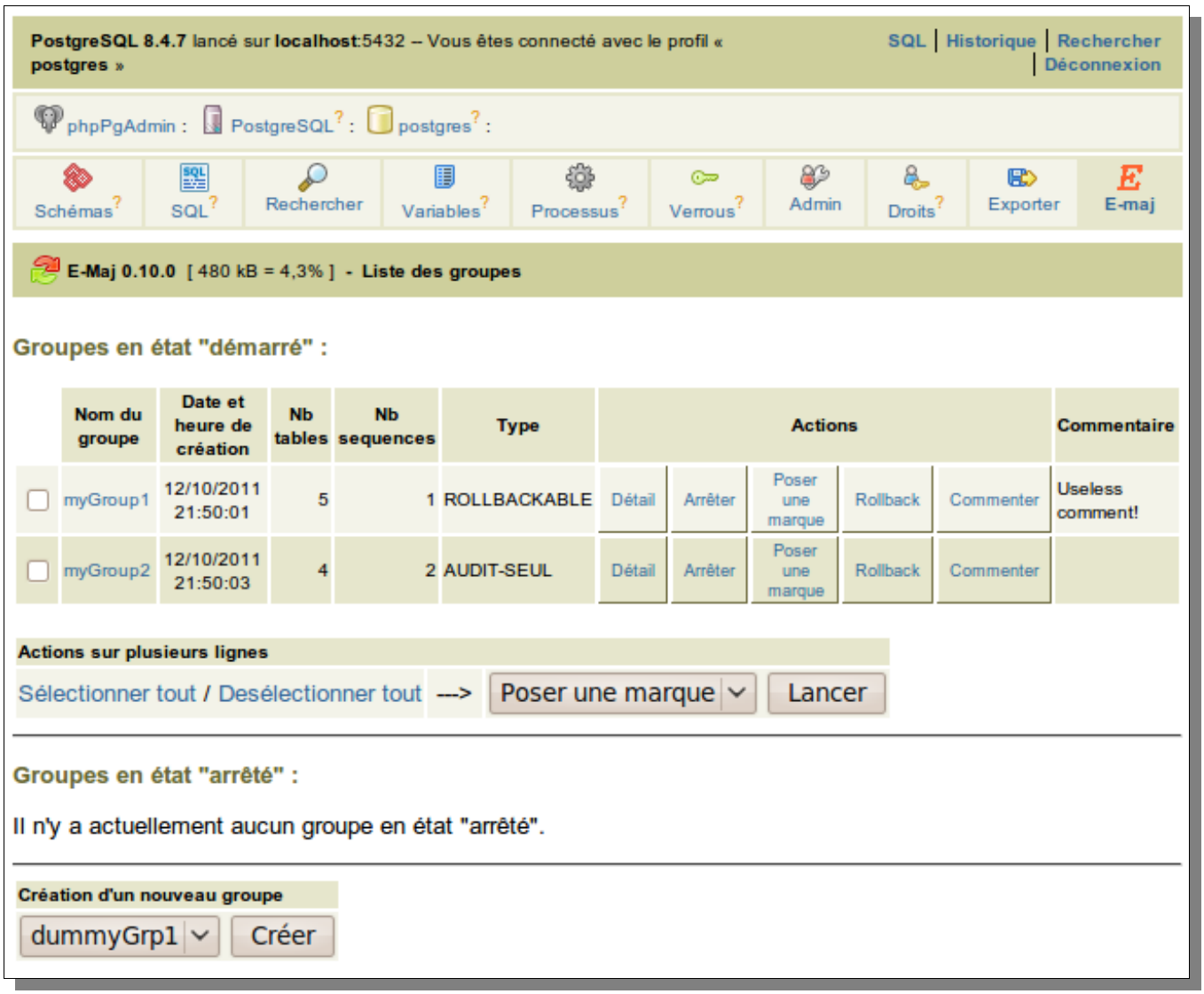


Figure 2 – Liste des groupes de tables créés sur la base de données

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 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 Détail d'un groupe de tables

En cliquant sur le bouton « Détail », on peut en savoir davantage sur un groupe de table particulier.

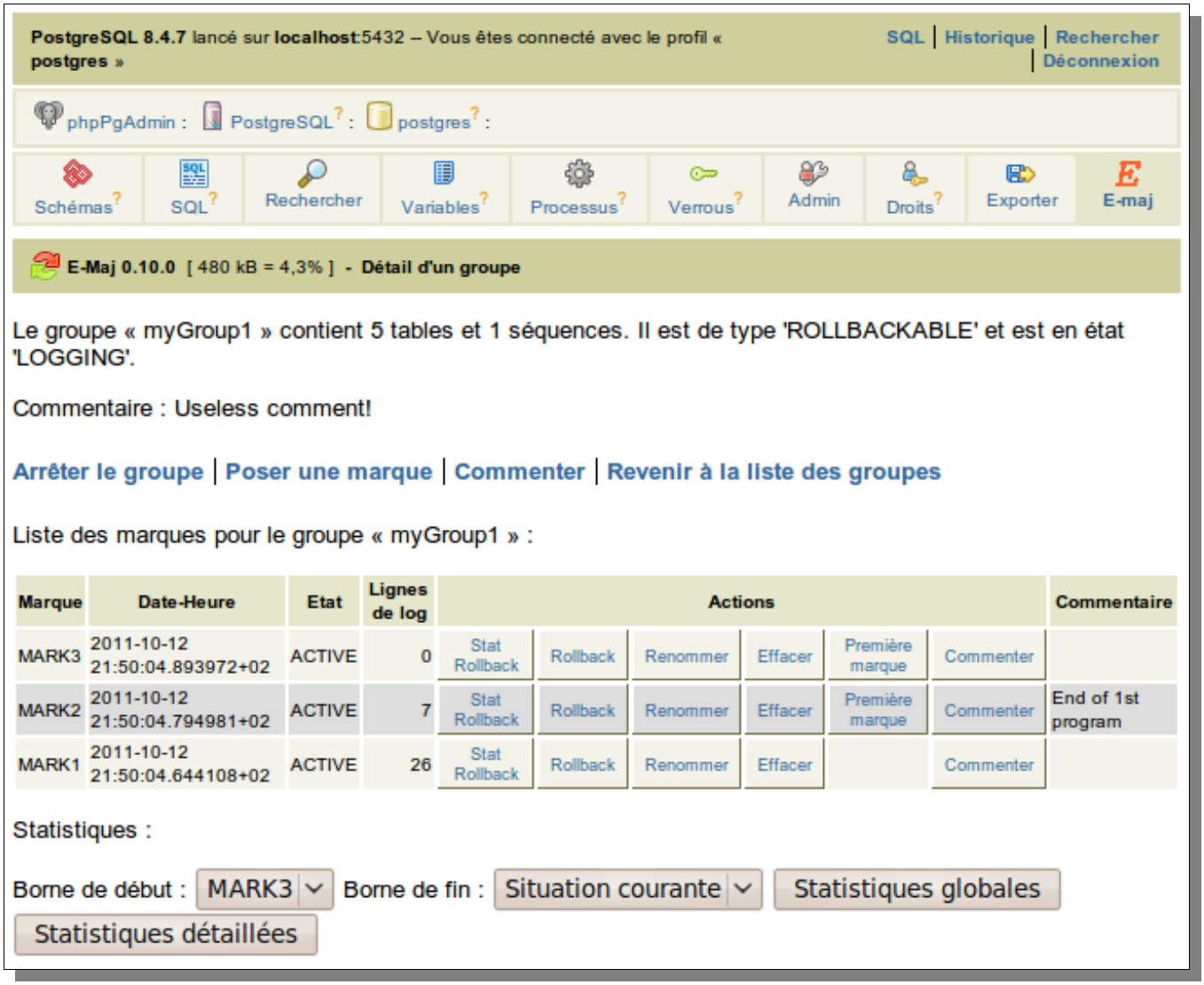


Figure 3 – Détail d'un groupe de tables

Une première ligne reprend des informations déjà affichées sur le tableau des groupes : nombre de tables et de séquences, type et état.

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

Une troisième ligne présente les actions possibles sur le groupe, en fonction de son état.

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 depuis cette marque ou entre cette marque et la suivante si une autre marque la suit,
- 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.

6.2.4 Statistiques pour les rollback

La figure 4 montre la page obtenue en cliquant sur le bouton « Stat Rollback » d'une marque.

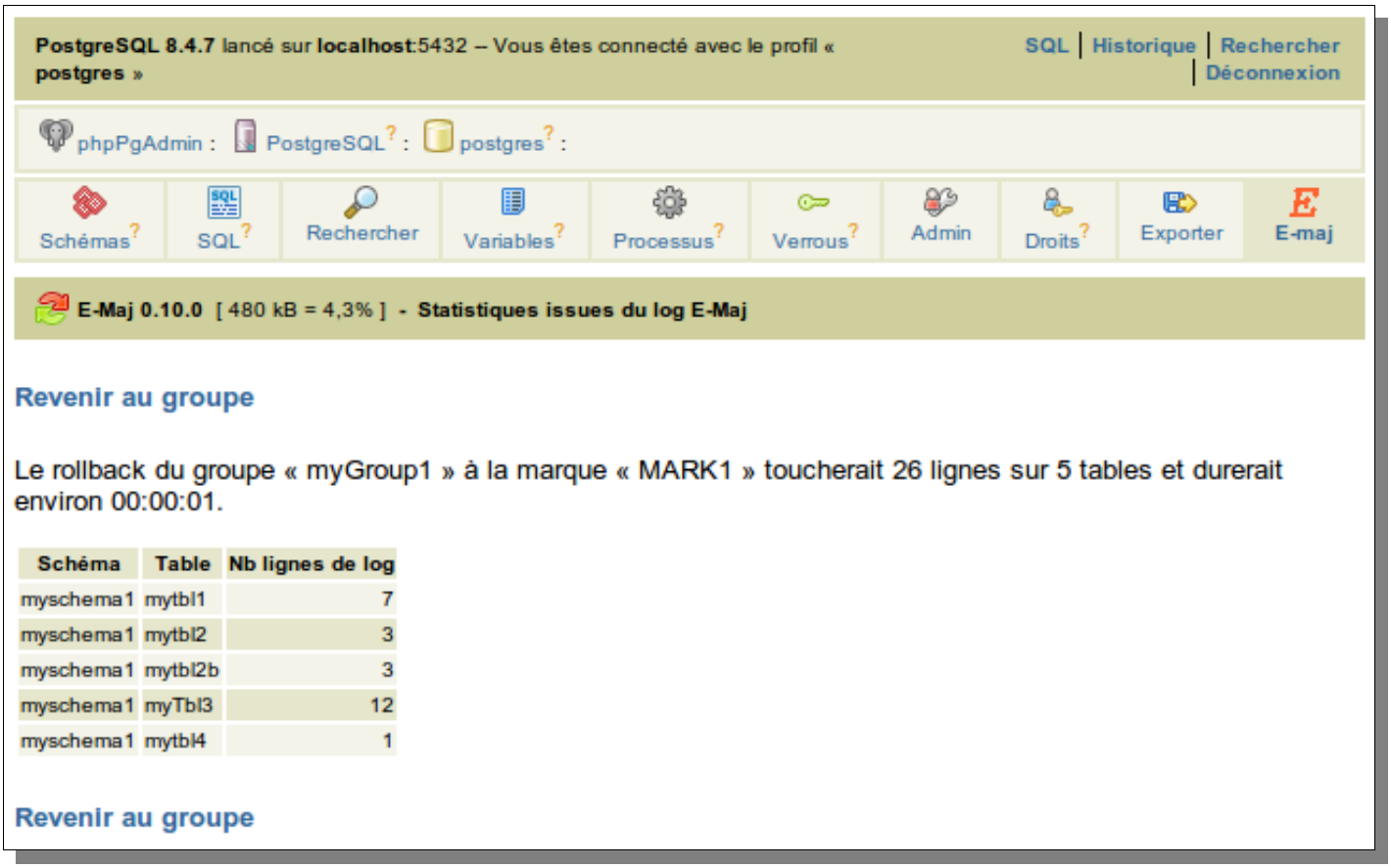


Figure 4 – Statistiques avant rollback

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 cet éventuel rollback.

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