E-Maj Documentation

Version 2.0.0

Philippe Beaudoin

déc. 09, 2017

Aperçu:

1 Introduction 1

2 Concepts 3

3 Architecture 5

4 Téléchargement et préparation de l'extension 9

5 Installation de l'extension E-Maj dans une base de données 11

6 Mise à jour d'une version E-Maj existante 15

7 Désinstallation d'E-Maj 19

8 Changement de version de PostgreSQL 21

9 Mise en place de la politique d'accès à E-Maj 23

10 Fonctions principales 25

11 Modification des groupes de tables 37

12 Autres fonctions de gestion des groupes 41

13 Fonctions de gestion des marques 45

14 Fonctions statistiques 49

15 Fonctions d'extraction de données 53

16 Autres fonctions 57

17 Fonctions multi-groupes 61

18 Client de rollback avec parallélisme 63

19 Client de suivi des rollbacks 67

20 Paramétrage 69

21 Fiabilisation du fonctionnement 71

22 Traçabilité des opérations 73

23 Impacts sur l'administration de l'instance et de la base de données 77

24 Sensibilité aux changements de date et heure système 83

25 Performances 85

26 Limites d'utilisation 87

27 Responsabilités de l'utilisateur 89

28 Présentation générale des clients web 91

29 Installation du plug-in phpPgAdmin 93

30 Installation du client Emaj_web 95

31 Utilisation des clients web 97

32 Liste des fonctions E-Maj 107

33 Index et tables 113

CHAPITRE 1

Introduction

1.1 Licence

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

1.2 Objectifs d'E-Maj

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

Il répond à deux objectifs principaux :

- E-Maj peut servir à **tracer les mises à jours** effectuées sur le contenu de tables par des traitements. La consultation de ces mises à jour enregistrées offre ainsi une réponse aux besoins d' « audit des mises à jour ».
- Utilisant ces mises à jour enregistrées, E-Maj est capable de **remettre le 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.

En d'autres termes, E-Maj est une extension PostgreSQL qui permet un enregistrement des mises à jour avec une fine granularité et un voyage dans le temps de sous-ensembles de bases de données.

Il constitue ainsi une bonne solution pour :

- positionner à des moments précis des points de sauvegarde sur un ensemble de tables,
- restaurer si nécessaire cet ensemble de tables dans un état stable, sans arrêt de l'instance,
- 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.

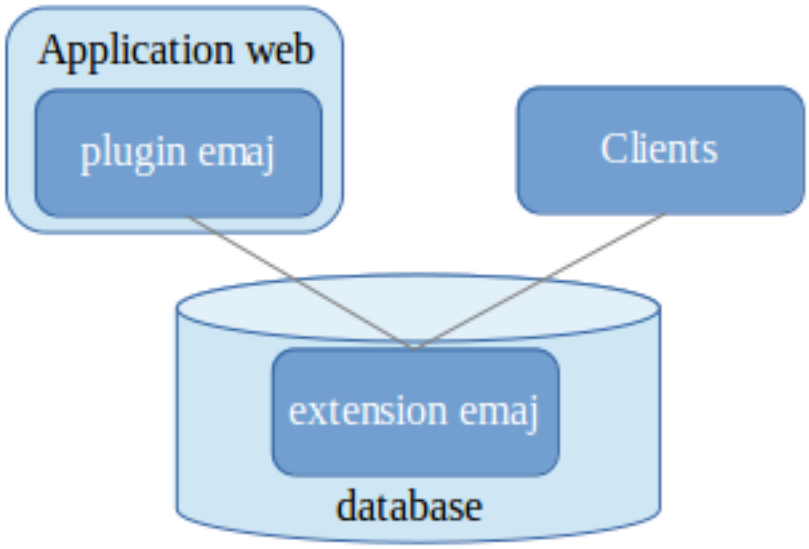
Dans un **environnement de test**, E-Maj permet également d'apporter de la souplesse dans les opérations. Il est ainsi possible de repositionner très facilement les bases de données dans des états stables prédéfinis afin de répéter autant de fois que nécessaire des tests de traitement.

1.3 Principaux composants

E-Maj regroupe en fait plusieurs composants :

- un objet PorstgreSQL **extension** créé dans chaque base de données, nommée emaj et contenant quelques tables, fonctions, séquences, ...
- un ensemble de **clients externes** appelables en ligne de commande,
- une **interface graphique web** sous la forme d'un plugin pour l'outil *phpPgAdmin*, ou d'une application indépendante, **Emaj_web**.

Les clients externes et l'interface graphique font appel aux fonctions de l'extension emaj.



Tous ces composants sont décrits dans cette documentation.

CHAPITRE 2

Concepts

E-Maj s'appuie sur trois concepts principaux.

2.1 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 d'une base de données 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 **partitions** de tables partitionnées et des **séquences**. Les tables (incluant les partitions) 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** » (*LOGGING*), soit dans un état « **inactif** » (*IDLE*). 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.

2.2 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.

2.3 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 deux types de rollback :

- avec un « **rollback non tracé** » (*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 « **rollback tracé** » (*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.

Notez que cette notion de « *Rollback E-Maj* » est distincte de celle du « *Rollback de transaction* » géré par PostgreSQL.

CHAPITRE 3

Architecture

Pour mener à bien l'opération de rollback sans avoir conservé au préalable une image physique des fichiers de l'instance 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.

3.1 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

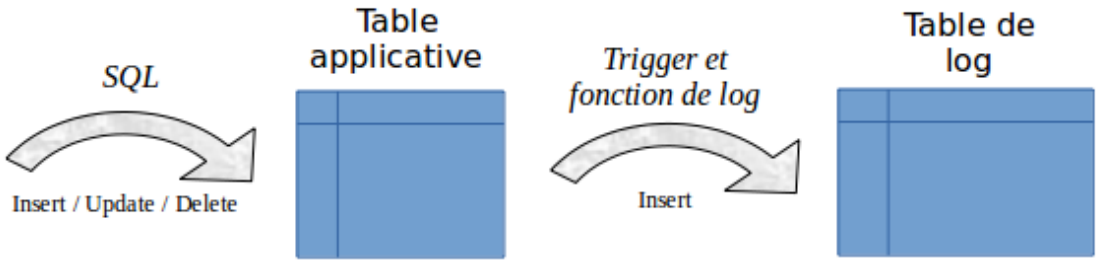
Pour les requêtes qui traitent plusieurs lignes, chaque création, modification ou suppression est enregistrée individuellement. Ainsi par exemple, une requête *DELETE FROM <table>* portant sur une table d'1 million de lignes générera l'enregistrement d'1 million de suppressions de ligne.

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 est interdite pour les groupes de tables de type « *ROLLBACKABLE* » à l'état « *actif* ». Son exécution est en revanche toujours autorisée pour les groupes de tables créés en mode « *AUTID_ONLY* ». Dans ce cas, seule l'exécution du verbe est enregistrée.

3.2 Les objects 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,
- un autre **trigger** permettant soit de bloquer toute exécution d'un verbe SQL *TRUNCATE* pour les groupes de type « *ROLLBACKABLE* », soit de tracer l'exécution des verbes SQL *TRUNCATE* pour les groupes de tables de type « *AUDIT_ONLY* »,
- 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 (*txid* 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,
- le port 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 :

- 13 tables,
- 5 types composites et 2 énumérations,
- 1 vue,
- plus de 90 fonctions, dont environ la moitié 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,
- 2 triggers sur événement avec PostgreSQL 9.3 et 9.4 et 3 triggers sur événement à partir de PostgreSQL 9.5.

Quelques tables techniques dont il peut être utile de connaître la structure sont décrites en détail : *emaj_group_def*, *emaj_param* et *emaj_hist*.

3.3 Norme de nommage des objets E-Maj

Les objets associés aux tables applicatives portent des noms construits en utilisant, par défaut, 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 de la séquence associée à la table de log est :** <nom.du.schema>_<nom.de.la.table>_log_seq

Mais il est possible de définir, pour chaque table applicative, le **préfixe** constituant le nom des objets E-Maj associés. Ceci permet notamment de gérer des tables avec des noms très longs.

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.

Les **triggers** créés sur les tables applicatives portent tous le même nom :

- *emaj_log_trg* pour les triggers de log,
- *emaj_trunc_trg* pour les triggers de contrôle des verbes *TRUNCATE*.

Le nom des **triggers sur événements** commence par *emaj_* et se termine par *_trg*.

3.4 Les schémas créés

Tous les objets techniques créés lors de l'installation de l'extension sont localisés dans le schéma **emaj**. Seule la fonction associée au trigger sur événement « emaj_protection_trg » appartient au schéma « public »

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 peut être défini dans le *paramétrage des groupes de tables*.

3.5 Les tablespaces utilisés

Lors de l'installation de l'extension, les tables techniques E-Maj sont stockées dans le tablespace par défaut, positionné au niveau de l'instance ou de la database ou explicitement défini pour la session courante.

Il en est de même pour les tables de log et leur index. Mais 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.

CHAPITRE 4

Téléchargement et préparation de l'extension

4.1 Téléchargement

E-Maj est disponible en téléchargement sur le site Internet **PGXN** (http://pgxn.org).

E-Maj et ses compléments sont également disponibles sur le site Internet **github.org** :

- Composants sources (https://github.com/beaud76/emaj)
- Documentation (https://github.com/beaud76/emaj_doc)
- Plugin pour phpPgAdmin (https://github.com/beaud76/emaj_ppa_plugin)
- Interface graphique Emaj_web (https://github.com/beaud76/emaj_web)

4.2 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 (Winzip, 7zip,...). Sous Unix/Linux, une commande du type

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

peut être utilisée pour un fichier .tar.gz ou

```
unzip e-maj-<version>.zip
```

pour un fichier zip.

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

Fichiers	Usage
sql/emaj–2.1.0.sql	script d'installation de l'extension (vers. 2.1.0)
sql/emaj–2.0.1–2.1.0.sql	script d'upgrade de l'extension de 2.0.1 vers 2.1.0
sql/emaj–2.0.0–2.0.1.sql	script d'upgrade de l'extension de 2.0.0 vers 2.0.1
sql/emaj–1.3.1–2.0.0.sql	script d'upgrade de l'extension de 1.3.1 vers 2.0.0
sql/emaj–unpackaged–1.3.1.sql	script de transformation en extension d'une version 1.3.1 existante
sql/emaj-1.3.0-to-1.3.1.sql	script de mise à jour d'une version E-Maj 1.3.0
sql/emaj-1.2.0-to-1.3.0.sql	script de mise à jour d'une version E-Maj 1.2.0
sql/emaj-1.1.0-to-1.2.0.sql	script de mise à jour d'une version E-Maj 1.1.0
sql/emaj-1.0.2-to-1.1.0.sql	script de mise à jour d'une version E-Maj 1.0.2
sql/emaj-1.0.1-to-1.0.2.sql	script de mise à jour d'une version E-Maj 1.0.1
sql/emaj-1.0.0-to-1.0.1.sql	script de mise à jour d'une version E-Maj 1.0.0
sql/emaj-0.11.1-to-1.0.0.sql	script de mise à jour d'une version E-Maj 0.11.1
sql/emaj-0.11.0-to-0.11.1.sql	script de mise à jour d'une version E-Maj 0.11.0
sql/emaj_demo.sql	script psql de démonstration d' E-Maj
sql/emaj_prepare_parallel_rollback_test.sql	script psql de test pour les rollbacks parallélisés
sql/emaj_uninstall.sql	script psql de désinstallation
README.md	documentation réduite de l'extension
CHANGES.md	notes de versions
LICENSE	information sur la licence utilisée pour E-Maj
AUTHORS	identification des auteurs
META.json	données techniques destinées à PGXN
emaj.control	fichier de contrôle utilisé par la gestion intégrée des extensions
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	client php pour les rollbacks parallélisés
php/emajRollbackMonitor.php	client php pour le suivi des rollbacks

4.3 Préparation du fichier emaj.control

A partir de la version 2.0.0, l'installation d'E-Maj dans les databases PostgreSQL s'effectue sous la forme d'une *EXTENSION*.

Mais pour qu'E-Maj soit connu du gestionnaire intégré des extensions, un fichier **emaj.control** doit être positionné dans le répertoire *SHAREDIR* de la version de PostgreSQL.

Pour ce faire :

— Identifier l'emplacement précis du répertoire *SHAREDIR* de votre installation en utilisant la commande shell

```
pg_config --sharedir
```

— Copier le fichier emaj.control fourni dans le répertoire racine de la version décompressée vers le répertoire *SHAREDIR*,

— Adapter la directive *directory* du fichier *emaj.control* pour spécifier le répertoire sql contenant les scripts d'installation d'E-Maj.

CHAPITRE 5

Installation de l'extension E-Maj dans une base de données

Si une version d'E-Maj est déjà installée dans la base de données, il faut la *mettre à jour*.

Quelques opérations préliminaires sont requises.

5.1 Opérations préliminaires

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

5.1.1 Extension DBLINK

L'extension **dblink**, fournie avec PostgreSQL, est utilisée lors des opérations de rollback E-Maj pour en permettre le suivi. Si elle n'est pas déjà présente dans la base de données, il est donc nécessaire de l'installer avant E-Maj. Pour ce faire, il suffit d'exécuter

```
CREATE EXTENSION dblink;
```

5.1.2 Tablespace

Optionnellement, si l'administrateur E-Maj veut stocker les tables techniques dans un **tablespace dédié**, il peut le créer si besoin et le définir comme tablespace par défaut pour la session :

```
SET default_tablespace = <nom.tablespace>;
```

5.2 Installation des composants E-Maj

Les composants E-Maj peuvent maintenant être installés dans la base de données, en exécutant la commande SQL :

```
CREATE EXTENSION emaj;
```

Le script commence par vérifier que la version de PostgreSQL est supérieure ou égale à la version 9.1, que le rôle qui exécute le script a bien l'attribut *superuser*.

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

Prudence : 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 de l'instance. Le cas échéant, il affiche un message concernant le paramètre *-max_prepared_statements*.

5.3 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 de l'instance. 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, il sera probablement nécessaire d'ajuster le paramètre **max_prepared_transaction**.

5.4 Paramétrage d'E-Maj

Un certain nombre de paramètres influence le fonctionnement d'E-Maj. Le détail des paramètres est présenté *ici*.

Cette étape de valorisation des paramètres est optionnelle. Leur valeur par défaut permet à E-Maj de fonctionner correctement.

Néanmoins, si l'administrateur E-Maj souhaite bénéficier du suivi des opérations de rollback, il est nécessaire de créer une ligne dans la table *emaj_param* pour définir la valeur du paramètre **dblink_user_password**.

5.5 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. Sous *psql*, il suffit d'exécuter le script *emaj_demo.sql* fourni avec l'extension

```
\i <répertoire_emaj>/sql/emaj_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!
```

L'examen des messages affichés par l'exécution du script permet de découvrir les principales fonctionnalités de l'extension. 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*.

CHAPITRE 6

Mise à jour d'une version E-Maj existante

6.1 Démarche générale

La procédure de mise à jour de la version d'E-Maj varie en fonction de la version E-Maj installée.

Pour les versions d'E-Maj antérieures à 0.11.0, il n'existe pas de procédure spécifique de mise à jour. On procédera donc à une simple désinstallation puis réinstallation de l'extension. 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 0.11.0 et suivantes, il est possible de procéder à une mise à jour sans désinstallation. Suivant la situation, il faut procéder en une ou en plusieurs étapes.

Prudence : A partir de la version 2.0.0, E-Maj ne supporte plus les versions de PostgreSQL antérieures à 9.1. Si une version antérieure de PostgreSQL est utilisée, il faut la faire évoluer avant de migrer E-Maj dans une version supérieure.

6.2 Mise à jour par désinstallation puis réinstallation

Pour ce type de mise à jour, il n'est pas nécessaire d'utiliser la procédure de *désinstallation complète*. 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.

6.2.1 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()* (ou de la fonction :ref :*emaj_force_stop_group() <emaj_force_stop_group>* si *emaj_stop_group()* retourne une erreur).

6.2.2 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';
```

6.2.3 Suppression et réinstallation d'E-Maj

Une fois connecté en tant que super-utilisateur, il suffit d'enchaîner le script de désinstallation *uninstall.sql* de la version en place puis la création de l'extension.

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

CREATE EXTENSION emaj;
```

NB : à partir de la version 2.0.0, le script de désinstallation se nomme *emaj_uninstall.sql*.

6.2.4 Restauration des données utilisateurs

Les données sauvegardées au préalable peuvent alors être restaurées dans les deux tables techniques d'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.

6.3 Mise à jour à partir d'une version E-Maj comprise entre 0.11.0 et 1.3.1

Pour les versions comprises entre 0.11.0 et 1.3.1, des **scripts psql de mise à jour** sont livrés. Ils permettent de passer d'une version à la suivante.

Chaque étape peut être réalisée sans toucher aux groupes de tables, ceux-ci pouvant même être actifs au moment du changement de version. Ceci signifie en particulier :

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

Version source	Version cible	script psql	Durée	Mises à jour concurrentes (1)
0.11.0	0.11.1	emaj-0.11.0-to-0.11.1.sql	Très rapide	Oui
0.11.1	1.0.0	emaj-0.11.1-to-1.0.0.sql	Très rapide	Oui
1.0.0	1.0.1	emaj-1.0.0-to-1.0.1.sql	Très rapide	Oui
1.0.1	1.0.2	emaj-1.0.1-to-1.0.2.sql	Très rapide	Oui
1.0.2	1.1.0	emaj-1.0.2-to-1.1.0.sql	Variable	Non (2)
1.1.0	1.2.0	emaj-1.1.0-to-1.2.0.sql	Très rapide	Oui
1.2.0	1.3.0	emaj-1.2.0-to-1.3.0.sql	Rapide	Oui (3)
1.3.0	1.3.1	emaj-1.3.0-to-1.3.1.sql	Très rapide	Oui

1. La dernière colonne indique si la mise à jour de la version E-Maj peut être effectuée alors que des tables couvertes par E-Maj sont accédées en mise à jour. Notons que durant la mise à jour, d'éventuelles autres actions E-Maj (pose de marque, rollback,. . .) sont mises en attentes.
2. Le passage en 1.1.0 nécessite la transformation des tables de log (ajout d'une colonne). Cela a pour conséquence que :

— même si les groupes de tables peuvent rester actifs, ce changement de version ne peut s'exécuter qu'à un moment où les tables ne sont pas mises à jour par des traitements,
— la durée de l'opération est très variable et dépend essentiellement du volume de données contenu dans les tables de log.

Notez également que les statistiques qu'E-Maj a collectées lors des précédentes opérations de rollback ne sont pas reprises (le fonctionnement des rollbacks est trop différent pour que ces anciennes statistiques soient pertinentes).

3. Il est recommandé de réaliser le passage en 1.3.0 dans une période de faible activité sur la base de données. En effet, le renommage des triggers E-Maj sur les tables applicatives entraîne la pose de verrous de type *Access Exclusive* qui peuvent entrer en conflit avec d'autres accès.

A la fin de chaque mise à jour le message suivant est affiché :

```
>>> E-Maj successfully upgraded to <nouvelle_version>
```

6.4 Passage d'E-Maj 1.3.1 à une version supérieure

La mise à jour de la version 1.3.1 est spécifique car elle doit gérer le passage d'une installation par script *psql* à une installation par *extension*.

Pour ce faire, il suffit d'exécuter la requête SQL

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

C'est le gestionnaire d'extension de PostgreSQL qui détermine le ou les scripts à exécuter en fonction de la version indiquée comme courante dans le fichier *emaj.control*.

Cette mise à jour ne peut néanmoins pas traiter le cas où au moins un groupe de tables a été créé avec une version de PostgreSQL antérieure à 8.4. Dans ce cas le ou les groupes de tables concernés doivent être supprimés au préalable puis recréés par la suite.

6.5 Mise à jour d'une version déjà installée comme extension

Une version existante installée comme une *extension* se met à jour par une simple requête :

```
ALTER EXTENSION emaj UPDATE;
```

C'est le gestionnaire d'extension de PostgreSQL qui détermine le ou les scripts à exécuter en fonction de la version installée et de la version indiquée comme courante dans le fichier *emaj.control*.

L'opération est très rapide et ne touche pas aux groupes de tables. Ceux-ci peuvent rester actifs au moment de la mise à jour. Ceci signifie en particulier :

- que des mises à jour de tables peuvent être enregistrées avant puis après le changement de version
- et donc qu'après le changement de version, un *rollback* à une marque posée avant ce changement de version est possible.

Spécificités liées aux versions :

- La procédure de mise à jour d'une version 2.0.1 en version 2.1.0 peut modifier la table *emaj_group_def* pour refléter le fait que le tablespace tspemaj n'est plus automatiquement considéré comme un tablespace par défaut. Si *tspemaj* est effectivement utilisé comme tablespace par défaut pour des groupes de tables créés, le contenu des colonnes *grpdef_log_dat_tsp* et *grpdef_log_idx_tsp* de la table *emaj_group_def* est automatiquement ajusté afin qu'une future opération de suppression puis recréation d'un groupe de tables puisse stocker les tables et index de log dans les mêmes tablespaces. L'administrateur peut revoir ces changements pour être sûr qu'ils correspondent bien à ses souhaits.

CHAPITRE 7

Désinstallation d'E-Maj

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*.

Bien qu'installé avec une requête *CREATE EXTENSION*, E-Maj ne peut se désinstaller par une simple requête *DROP EXTENSION*. Un trigger sur événement bloque d'ailleurs l'exécution d'une telle requête (à partir de PostgreSQL 9.3).

Pour désinstaller E-Maj, il faut simplement exécuter le **script fourni** *emaj_uninstall.sql*.

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

Ce script effectue les actions suivantes :

- iI exécute les éventuelles fonctions de nettoyage créées par l'exécution des scripts de démonstration ou de test fournis,
- il arrête les groupes de tables encore actifs, s'il y en a,
- il supprime les groupes de tables existants, supprimant en particulier les triggers sur les tables applicatives,
- il supprime l'extension et le schéma principal *emaj*,
- il supprime les rôles *emaj_adm* et *emaj_viewer* s'ils ne sont pas associés à d'autres rôles ou à d'autres bases de données de l'instance et ne possèdent pas de droits sur d'autres tables.

L'exécution du script de désinstallation affiche ceci :

```
$ psql ... -f sql/emaj_uninstall.sql
>>> Starting E-Maj uninstallation procedure...
SET
psql:sql/emaj_uninstall.sql:203: WARNING:  emaj_uninstall: emaj_adm and emaj_viewer␣
↪roles have been dropped.
DO
```

```
SET
>>> E-maj successfully uninstalled
```

CHAPITRE 8

Changement de version de PostgreSQL

Il est possible qu'un changement de version PostgreSQL impacte le contenu de l'extension E-Maj. Mais les principes suivants s'appliquent :

- il est possible de changer de version de PostgreSQL sans réinstallation d'E-Maj,
- les groupes de tables peuvent même rester actifs lors du changement de version PostgreSQL,
- s'il est nécessaire d'adapter le contenu de l'extension E-Maj, ceci doit être réalisé par un script.

Aussi, un script *psql*, à passer après chaque changement de version PostgreSQL, est fourni pour traiter d'éventuels impacts. Il faut l'exécuter en tant que super-utilisateur :

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

Dans les versions E-Maj 2.0.0 et suivantes, ce script ne fait que créer les éventuels triggers sur événement manquants :

- ceux qui apparaissent en version 9.3 et qui protègent contre la suppression de l'extension elle-même et contre la suppression d'objets E-Maj (tables de log, fonctions, . . .),
- celui qui apparaît en version 9.5 et qui protège contre les changements de structure des tables de log.

Le script peut-être lancé plusieurs fois de suite sur une même version, seule la première exécution modifiant l'environnement.

Si le changement de version PostgreSQL s'effectue avec un déchargement et rechargement des données et si les groupes de tables peuvent être arrêtés, une purge des tables de log grâce à l'exécution d'une fonction *emaj_reset_group()* peut permettre de diminuer la quantité de données à manipuler et donc d'accélérer l'opération.

CHAPITRE 9

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.

9.1 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*.

9.2 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>;
```

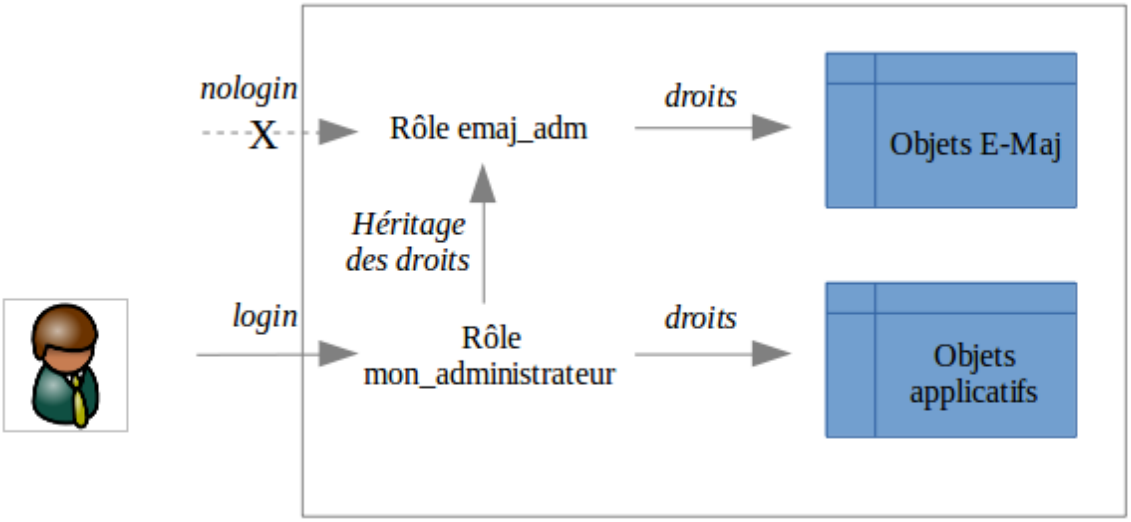
Naturellement, plusieurs rôles peuvent se voir attribuer les droits *emaj_adm* ou *emaj_viewer*.

9.3 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.

9.4 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*.

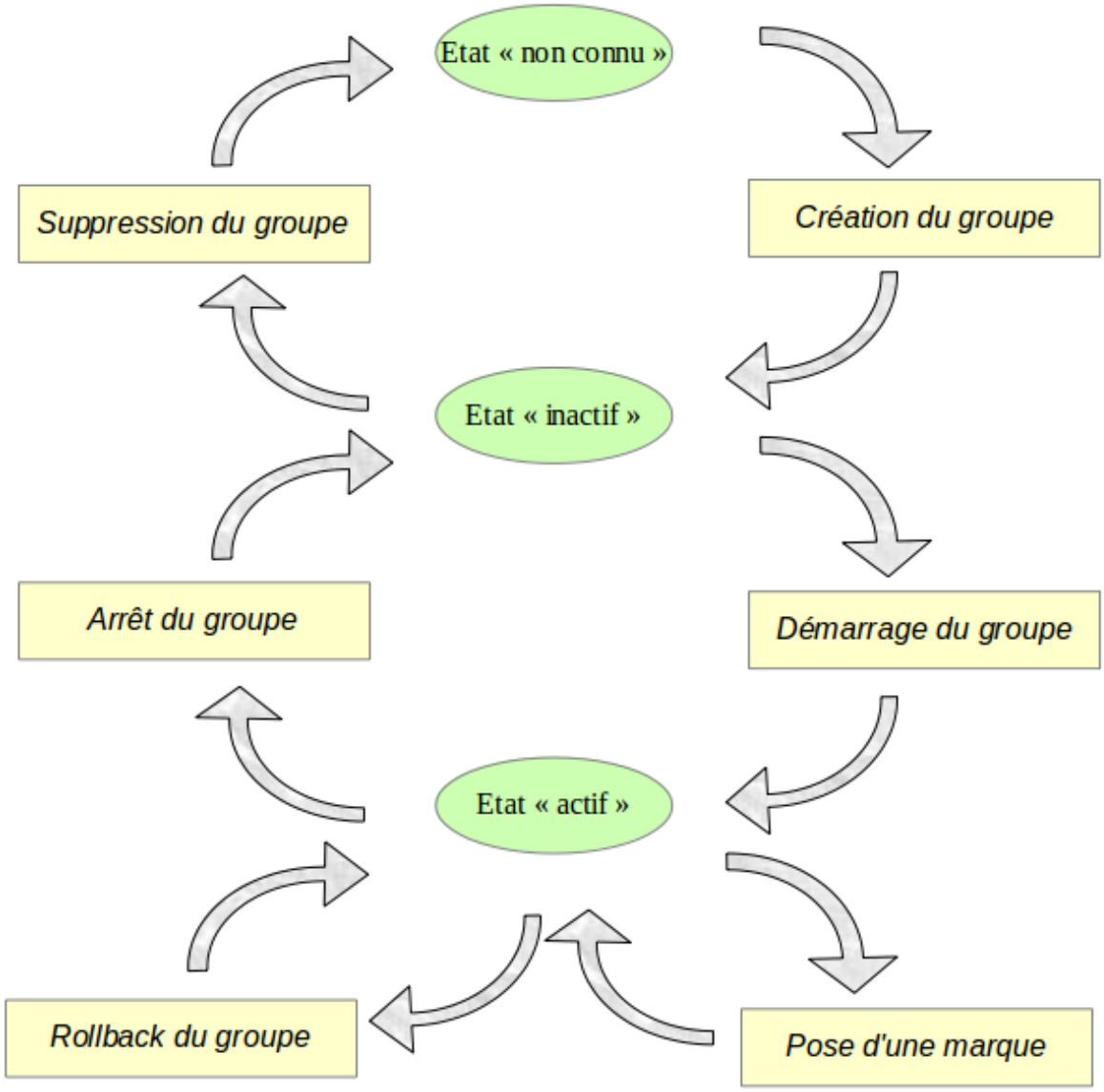
CHAPITRE 10

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.

10.1 Enchaînement des opérations

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



10.2 Définition des groupes de tables

10.2.1 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_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_emaj_names_prefix	TEXT	préfixe des noms d'objets E-Maj générés pour 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.

Prudence : 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*.

10.2.2 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*).

E-Maj gère les partitions élémentaires de tables partitionnées créées avec le DDL déclaratif (à partir de PostgreSQL 10). Elles sont gérées comme n'importe quelle autre table. En revanche, comme les tables mères restent toujours vides, E-Maj refuse qu'elles soient assignées à un groupe de tables. Toutes les partitions d'une même table partitionnée n'ont pas nécessairement besoin d'être couvertes par E-Maj. Des partitions d'une même table partitionnée peuvent être affectées à des groupes de tables différents.

De par leur nature, ni les tables temporaires (*TEMPORARY TABLE*), ni les tables non tracées (*UNLOGGED TABLE*) ne peuvent être supportées par E-Maj. Les tables doivent aussi être implicitement ou explicitement définies *WITHOUT OIDS*.

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 !

10.2.3 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 les tables dont le nom est très long, le préfixe par défaut des noms d'objet E-Maj générés peut s'avérer trop long pour respecter les limites de PostgreSQL. Mais un autre préfixe peut être défini pour chaque table, en alimentant la colonne **grpdef_emaj_names_prefix**.

Si cette colonne *grpdef_emaj_names_prefix* contient une valeur *NULL*, le préfixe par défaut *<nom_schéma>_<nom_table>* est utilisé.

Deux tables différentes ne peuvent pas, explicitement ou implicitement, partager le même préfixe.

Pour les séquences, la colonne *grpdef_emaj_names_prefix* 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 par défaut de la session courante au moment de la création du groupe est utilisé.

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

10.3 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*.

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 d'avertissement 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 soit de type *SERIAL* ou *BIGSERIAL* soit définie avec une clause *GENERATED AS IDENTITY*, et que sa table d'appartenance ne fait pas partie du groupe, la fonction génère également un message de type *WARNING*.

Une forme particulière de la fonction permet de créer un groupe de table vide, c'est à dire ne contenant à sa création aucune table ni séquence

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

Le troisième paramètre prend la valeur *faux* par défaut. Si le paramètre est valorisé à *vrai*, le groupe ne doit pas être référencé dans la table *emaj_group_def*. Une fois créé, un groupe vide peut ensuite être peuplé, à l'aide de la fonction *emaj_alter_group()*.

Toutes les actions enchaînées par la fonction *emaj_create_group()* sont 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()* ou supprimé par la fonction *emaj_drop_group()*.

10.4 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>' [, <effacer.
↪anciens.logs?>]]);
```

Le groupe de tables doit être au préalable à l'état inactif.

Le démarrage du groupe de tables créé une première marque.

S'il est spécifié, le nom de la marque initiale 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 n'est pas spécifié, ou s'il est vide ou *NULL*, un nom est automatiquement généré : "*START_%*", où le caractère '*%*' représente l'heure de début de la transaction courante, au format « *hh.mn.ss.mmm* ».

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 sur chacune des tables du groupe un verrou de type *ACCESS EXCLUSIVE* si la version de PostgreSQL est antérieure à 9.5, ou *SHARE ROW EXCLUSIVE* dans le cas contraire. 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*.

A l'issue du démarrage d'un groupe, celui-ci devient actif ("*LOGGING*").

Plusieurs groupes de tables peuvent être démarrés en même temps, en utilisant la fonction *emaj_start_groups()*

```
SELECT emaj.emaj_start_groups('<tableau.des.groupes>'[, '<nom.de.marque>' [, <effacer.
↪anciens.logs?>]]);
```

Plus d'information sur les *fonctions multi-groupes*.

10.5 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 à l'état actif.

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 n'est pas spécifié ou s'il est vide ou *NULL*, un nom est automatiquement généré : « *MARK_%* », où le caractère '%' représente l'heure de début de la transaction courante, au format « *hh.mn.ss.mmm* ».

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.

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.

Une marque peut être posée sur plusieurs groupes de tables même temps, en utilisant la fonction *emaj_set_mark_groups()*

```
SELECT emaj.emaj_set_mark_groups('<tableau.des.groupes>'[, '<nom.de.marque>']);
```

Plus d'information sur les *fonctions multi-groupes*.

10.6 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* » ou « *non tracé* »), il suffit d'exécuter la requête SQL suivante

```
SELECT * FROM emaj.emaj_rollback_group('<nom.du.groupe>', '<nom.de.marque>', <est_
↪altération_groupe_permise>);
```

Le groupe de tables doit être à l'état actif et la marque indiquée doit être toujours « active », c'est à dire qu'elle ne doit pas être marquée comme logiquement supprimée.

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

Le 3ème paramètre est un booléen qui indique si l'opération de rollback peut cibler une marque posée antérieurement à une opération de *modification du groupe de tables*. Selon leur nature, les modifications de groupe de tables effectuées alors que ce dernier est en état *LOGGING* peuvent être ou non automatiquement annulées. Dans certains cas, cette annulation peut être partielle. Par défaut, ce paramètre prend la valeur *FAUX*.

La fonction retourne un ensemble de lignes comportant un niveau de sévérité pouvant prendre les valeurs « *Notice* » ou « *Warning* », et un texte de message. La fonction retourne une ligne de type « *Notice* » indiquant le nombre de tables et de séquences effectivement modifiées par l'opération de rollback. Des lignes de types « *Warning* » peuvent aussi être émises dans le cas où des opérations de modification du groupe de tables ont du être traitées par le 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. Lorsque la version de PostgreSQL est antérieure à 9.5, le verrou est même de type *ACCESS EXCLUSIVE* pour les tables ayant des mises à jour à annuler et dont le trigger de log doit donc être désactivé durant l'opération. 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 (plus de détails *ici*).

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.

Lorsque le volume de mises à jour à annuler est important et que l'opération de rollback est longue, il est possible de suivre l'avancement de l'opération à l'aide de la fonction *emaj_rollback_activity()* ou du client *emajRollbackMonitor.php*.

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.

Les opérations de rollback sont historisées dans la table *emaj_rlbk*. L'état final des opérations de rollback est accessible dans les colonnes *rlbk_status* et *rlbk_msg* de cette table *emaj_rlbk*.

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.

Prudence : 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.

Plusieurs groupes de tables peuvent être « rollbackés » en même temps, en utilisant la fonction *emaj_rollback_groups()*

```
SELECT * FROM emaj.emaj_rollback_groups('<tableau.des.groupes>', '<nom.de.marque>',
↪<est_altération_groupe_permise>);
```

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

Plus d'information sur les *fonctions multi-groupes*.

Une ancienne version de ces fonctions ne comportait pas de troisième paramètre et retournait un simple entier correspondant au nombre de tables et séquences effectivement traitées :

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

SELECT emaj.emaj_rollback_groups('<tableau.des.groupes>', '<nom.de.marque>');
```

Ces 2 fonctions sont dépréciées et sont amenées à être supprimées dans une prochaine version.

10.7 Rollback annulable d'un groupe de tables

Une autre fonction permet d'exécuter un rollback de type « *logged* », Dans ce cas, les triggers de log sur les tables applicatives ne sont pas désactivés 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 * FROM emaj.emaj_logged_rollback_group('<nom.du.groupe>', '<nom.de.marque>',
↪<est_altération_groupe_permise>);
```

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.

Le 3ème paramètre est un booléen qui indique si l'opération de rollback peut cibler une marque posée antérieurement à une opération de *modification du groupe de tables*. Selon leur nature, les modifications de groupe de tables effectuées alors que ce dernier est en état *LOGGING* peuvent être ou non automatiquement annulées. Dans certains cas, cette annulation peut être partielle. Par défaut, ce paramètre prend la valeur *FAUX*.

La fonction retourne un ensemble de lignes comportant un niveau de sévérité pouvant prendre les valeurs « *Notice* » ou « *Warning* », et un texte de message. La fonction retourne une ligne de type « *Notice* » indiquant le nombre de tables et de séquences effectivement modifiées par l'opération de rollback. Des lignes de types « *Warning* » peuvent aussi être émises dans le cas où des opérations de modification du groupe de tables ont du être traitées par le 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_logged_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 (plus de détails *ici*).

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* ».

Lorsque le volume de mises à jour à annuler est important et que l'opération de rollback est longue, il est possible de suivre l'avancement de l'opération à l'aide de la fonction *emaj_rollback_activity()* ou du client *emajRollbackMonitor.php*.

Les opérations de rollback sont historisées dans la table *emaj_rlbk*. L'état final des opérations de rollback est accessible dans les colonnes *rlbk_status* et *rlbk_msg* de cette table *emaj_rlbk*.

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 la marque automatiquement posée en début de rollback, pour annuler ce dernier, ou encore une ancienne marque postérieure à la marque utilisée pour le rollback. oDes rollbacks de différents types (*logged* / *unlogged*) peuvent être exécutés en séquence. 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)
```

Une *fonction de « consolidation »* de « *rollback tracé* » permet de transformer un rollback annulable en rollback simple.

Plusieurs groupes de tables peuvent être « rollbackés » en même temps, en utilisant la fonction *emaj_logged_rollback_groups()*

```
SELECT * FROM emaj.emaj_logged_rollback_groups ('<tableau.des.groupes>', '<nom.de.
↪marque>', <est_altération_groupe_permise>);
```

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

Plus d'information sur les *fonctions multi-groupes*.

Une ancienne version de ces fonctions ne comportait pas de troisième paramètre et retournait un simple entier correspondant au nombre de tables et séquences effectivement traitées :

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

SELECT emaj.emaj_logged_rollback_groups('<tableau.des.groupes>', '<nom.de.marque>');
```

Ces 2 fonctions sont dépréciées et sont amenées à être supprimées dans une prochaine version.

10.8 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'>]);
```

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 le paramètre représentant cette marque n'est pas spécifié ou s'il est vide ou *NULL*, un nom est automatiquement généré : « *STOP_%* », où le caractère '*%*' représente l'heure de début de la transaction courante, au format « *hh.mn.ss.mmm* ».

L'arrêt d'un groupe de table désactive simplement les triggers de log des tables applicatives du groupe. La pose de verrous de type *ACCESS EXCLUSIVE* pour les versions de PostgreSQL antérieure à 9.5, ou *SHARE ROW EXCLUSIVE* dans le cas contraire qu'entraîne 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 ». 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.

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.

Plusieurs groupes de tables peuvent être arrêtés en même temps, en utilisant la fonction *emaj_stop_groups()*

```
SELECT emaj.emaj_stop_groups('<tableau.des.groupes>'[, '<nom.de.marque'>]);
```

Plus d'information sur les *fonctions multi-groupes*.

10.9 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 d'abord utiliser la fonction *emaj_stop_group()*.

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, triggers de log.

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

La pose de verrous qu'entraîne 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.

CHAPITRE 11

Modification des groupes de tables

Plusieurs 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,
- un des paramètres liés à une table change dans la configuration E-Maj (priorité, schéma de log, 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,...).

11.1 Modification de groupes en état *IDLE*

Dans tous les cas, la démarche suivante peut être suivie :

- 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,
- 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 reste en 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()* sont 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()*.

Plusieurs groupes de tables peuvent être modifiés en même temps, en utilisant la fonction *emaj_alter_groups()*

```
SELECT emaj.emaj_alter_groups('<tableau.des.groupes>');
```

Cette fonction permet notamment de déplacer une table ou une séquence d'un groupe de tables à un autre dans une même opération.

Plus d'information sur les *fonctions multi-groupes*.

11.2 Modification de groupes en état *LOGGING*

Mais la méthode précédente présente plusieurs inconvénients :
- les logs antérieurs à l'opération sont perdus,
- il n'est plus possible de remettre le groupe de tables dans un état antérieur.

Néanmoins certaines actions sont possibles sur des groupes de tables maintenus en état *LOGGING*. Le tableau suivant liste ces actions possibles et leurs conditions de réalisation.

Action	Groupe LOGGING	Méthode
Changer le groupe d'appartenance	Non	
Changer le suffixe du schéma de log	Oui	Ajustement emaj_group_def
Changer le préfixe des noms emaj	Oui	Ajustement emaj_group_def
Changer le tablespace de log data	Oui	Ajustement emaj_group_def
Changer le tablespace de log index	Oui	Ajustement emaj_group_def
Changer la priorité E-Maj	Oui	Ajustement emaj_group_def
Oter une table d'un groupe	Non	
Oter une séquence d'un groupe	Non	
Ajouter une table à un groupe	Non	
Ajouter une séquence à un groupe	Non	
Réparer une table ou une séquence	Non	
Renommer une table	Non	
Renommer une séquence	Non	
Changer le schéma d'une table	Non	
Changer le schéma d'une séquence	Non	
Renommer une colonne d'une table	Non	
Changer la structure d'une table	Non	
Autres formes d'ALTER TABLE	Oui	Sans impact E-Maj
Autres formes d'ALTER SEQUENCE	Oui	Sans impact E-Maj

11.2.1 Méthode "Modification emaj_group_def"

La plupart des attributs de la table *emaj_group_def* décrivant les groupes de tables peuvent être modifiés et pris en compte en dynamique, sans que les groupes de tables ne soient arrêtés.

Pour ce faire, il suffit d'enchaîner les opérations :
- modifier la table *emaj_group_def*,
- appeler l'une des fonctions *emaj_alter_group()* ou *emaj_alter_groups()*.

Pour les groupes de tables en état *LOGGING*, ces fonctions posent un verrou de type *ROW EXCLUSIVE* sur chaque table applicative constituant les groupes de tables concernés.

Sur ces mêmes groupes, elles posent également une marque dont le nom peut être fourni en paramètre. La syntaxe de ces appels devient

```
SELECT emaj.emaj_alter_group('<nom.du.groupe>' [,'<marque>']);
```

ou

```
SELECT emaj.emaj_alter_groups('<tableau.des.groupes>' [,'<marque>']);
```

Si le paramètre représentant la marque n'est pas spécifié, ou s'il est vide ou *NULL*, un nom est automatiquement généré : « ALTER_% », où le caractère '%' représente l'heure de début de la transaction courante, au format « hh.mn.ss.mmm ».

Une opération de rollback E-Maj ciblant une marque antérieure à une modification de groupes de tables ne procède **PAS** automatiquement à une annulation de ces changements.

Néanmoins, l'administrateur a la possibilité d'appliquer cette même procédure pour revenir à un état antérieur.

CHAPITRE 12

Autres fonctions de gestion des groupes

12.1 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* »).

12.2 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.

12.3 Protection d'un groupe de tables contre les rollbacks

Il peut être utile à certains moments de se protéger contre des rollbacks intempestifs de groupes de tables, en particulier sur des bases de données de production. Deux fonctions répondent à ce besoin.

La fonction *emaj_protect_group()* pose une protection sur un groupe de tables.

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

La fonction retourne l'entier 1 si le groupe de tables n'était pas déjà protégé, ou 0 s'il était déjà protégé.

Une fois le groupe de tables protégé, toute tentative de rollback, tracé ou non, sera refusée.

Un groupe de tables de type « *audit-seul* » ou dans un état « inactif » ne peut être protégé.

Au démarrage d'un groupe de tables, ce dernier n'est pas protégé. Lorsqu'il est arrêté, un groupe de tables protégé contre les rollbacks perd automatiquement sa protection.

La fonction *emaj_unprotect_group()* ôte une protection existante sur un groupe de tables.

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

La fonction retourne l'entier 1 si le groupe de table était protégé au préalable, ou 0 s'il n'était pas déjà protégé.

Un groupe de tables de type « *audit-seul* » ne peut être déprotégé.

Une fois la protection d'un groupe de tables ôtée, il devient à nouveau possible d'effectuer tous types de rollback sur le groupe.

Un mécanisme de *protection au niveau des marques* complète ce dispositif.

12.4 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 par inadvertance 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 tables est en état « *IDLE* ». Il peut alors être supprimé ou modifié avec les fonctions *emaj_alter_group()* ou *emaj_drop_group()*.

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

12.5 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 un groupe de tables en état 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.

Note : 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.

12.6 « Consolidation » d'un rollback tracé

Suite à l'exécution d'un « *rollback tracé* », et une fois que l'enregistrement de l'opération de rollback devient inutile, il est possible de « consolider » ce rollback, c'est à dire, en quelque sorte, de le transformer en « *rollback non tracé* ». A l'issue de l'opération de consolidation, les logs entre la marque cible du rollback et la marque de fin de rollback sont supprimés. La fonction *emaj_consolidate_rollback_group()* répond à ce besoin. :

```
SELECT emaj.emaj_consolidate_rollback_group('<nom.du.groupe>', <marque.de.fin.de.
↪rollback>);
```

L'opération de rollback tracé concernée est identifiée par le nom de la marque de fin qui a été générée par le rollback. Cette marque doit toujours exister, mais elle peut avoir été renommée.

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

La fonction *emaj_get_consolidable_rollbacks()* peut aider à identifier les rollbacks susceptibles d'être consolidés.

A l'image des fonctions effectuant des rollbacks, cette fonction retourne le nombre de tables et de séquence effectivement concernées par la consolidation.

Le groupe de table peut être en état « actif » ou non.

La marque cible du rollback doit également toujours exister mais elle peut avoir été renommée. Néanmoins, des marques intermédiaires peuvent avoir été supprimées.

A l'issue de la consolidation, ne sont conservées que la marque cible du rollback et la marque de fin du rollback. Les marques intermédiaires sont supprimées.

La place occupée par les lignes supprimées redeviendra réutilisable une fois que ces tables de log auront été traitées par le *VACUUM*.

Bien évidemment, une fois consolidé, un « *rollback tracé* » ne peut plus être annulé, la marque de début de rollback et les logs couvrant ce rollback étant supprimés.

L'opération de consolidation est insensible aux éventuelles protections posées sur les groupes ou les marques.

Si une base n'a pas de contraintes d'espace disque trop fortes, il peut être intéressant de remplacer un « *rollback simple* » (non tracé) par un « *rollback tracé* » suivi d'une « *consolidation* » pour que les tables applicatives soient accessibles en lecture durant l'opération de rollback, en tirant profit du plus faible niveau de verrou posé lors des rollbacks tracés.

12.7 Liste des « rollbacks consolidables »

La fonction *emaj_get_consolidable_rollbacks()* permet d'identifier les rollbacks susceptibles d'être consolidés

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

La fonction retourne un ensemble de lignes comprenant les colonnes :

Colonne	Type	Description
cons_group	TEXT	groupe de tables rollbackés
cons_target_rlbk_mark_name	TEXT	nom de la marque cible du rollback
cons_target_rlbk_mark_id	BIGINT	identifiant interne de la marque cible
cons_end_rlbk_mark_name	TEXT	nom de la marque de fin de rollback
cons_end_rlbk_mark_id	BIGINT	identifiant interne de la marque de fin
cons_rows	BIGINT	nombre de mises à jour intermédiaires
cons_marks	INT	nombre de marques intermédiaires

A l'aide de cette fonction, il est ainsi facile de consolider tous les rollbacks possibles de tous les groupes de tables d'une base de données pour récupérer le maximum d'espace disque possible :

```
SELECT emaj.emaj_consolidate_rollback_group(cons_group, cons_end_rlbk_mark_name) FROM␣
↪emaj.emaj_get_consolidable_rollbacks();
```

La fonction *emaj_get_consolidable_rollbacks()* est utilisable par les rôles *emaj_adm* et *emaj_viewer*.

CHAPITRE 13

Fonctions de gestion des marques

13.1 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 des *clients web*. En effet, ces derniers les affichent systématiquement dans le tableau des marques d'un groupe.

13.2 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.

13.3 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.

13.4 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.

13.5 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 retourne le nombre de marques supprimées.

La fonction procède également à la purge des événements les plus anciens de la table technique *emaj_hist*.

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()*.

Associées, les fonctions *emaj_delete_before_mark_group()*, et *emaj_get_previous_mark_group()* permettent de supprimer les marques antérieures à un délai de rétention. 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));
```

13.6 Protection d'une marque contre les rollbacks

Pour compléter le mécanisme de *protection des groupes de tables* contre les rollbacks intempestifs, il est possible de positionner des protections au niveau des marques. Deux fonctions répondent à ce besoin.

La fonction *emaj_protect_mark_group()* pose une protection sur une marque d'un groupe de tables. :

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

La fonction retourne l'entier 1 si la marque n'était pas déjà protégée, ou 0 si elle était déjà protégée.

Une fois une marque protégée, toute tentative de rollback, tracé ou non, sera refusée si elle repositionne le groupe de tables à un état antérieur à cette marque protégée.

Une marque d'un groupe de tables de type « *audit-seul* » ou en état « *inactif* » ne peut être protégée.

Lorsqu'une marque est posée, elle n'est pas protégée. Les marques protégées d'un groupe de tables perdent automatiquement leur protection lorsque ce groupe de tables est arrêté. Attention, la suppression d'une marque protégée supprime de facto la protection. Elle ne reporte pas la protection sur une marque adjacente.

La fonction *emaj_unprotect_mark_group()* ôte une protection existante sur une marque d'un groupe de tables. :

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

La fonction retourne l'entier 1 si la marque était bien protégée au préalable, ou 0 si elle n'était déjà déjà protégée.

Une marque d'un groupe de tables de type « *audit-seul* » ne peut être déprotégée.

Une fois la protection d'une marque ôtée, il devient à nouveau possible d'effectuer tous types de rollback sur une marque antérieure.

CHAPITRE 14

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 2 fonctions, *emaj_estimate_rollback_group()* et *emaj_estimate_rollback_groups()*, qui permettent d'estimer la durée que prendrait un éventuel rollback d'un ou plusieurs groupes à une marque donnée.

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

14.1 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 :

Column	Type	Description
stat_group	TEXT	nom du groupe de tables
stat_schema	TEXT	nom du schéma
stat_table	TEXT	nom de la table
stat_rows	BIGINT	nombre de modifications de lignes enregistrées dans la table de log associée à la table

Une valeur *NULL* ou une chaîne 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, 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.

14.2 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 :

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

Une valeur *NULL* ou une chaîne 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, 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)
```

14.3 Estimation de la durée d'un rollback

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

```
SELECT emaj.emaj_estimate_rollback_group('<nom.du.groupe>', '<nom.de.marque>', <est␣
↪tracé>);
```

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

Le troisième paramètre, de type booléen, indique si le rollback à simuler est tracé ou non.

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
- 6 *paramètres* génériques 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 indication utile 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* génériques. 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.

La fonction *emaj_estimate_rollback_groups()* permet d'estimer la durée d'un rollback portant sur plusieurs groupes de tables

```
SELECT emaj.emaj_estimate_rollback_groups('<tableau.des.groupes>', '<nom.de.marque>',
↪<est tracé>);
```

Plus d'information sur les *fonctions multi-groupes*.

CHAPITRE 15

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.

15.1 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 l'instance 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.

15.2 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 l'instance 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)

15.3 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_gen_sql_group()*.

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_gen_sql_group('<nom.du.groupe>', '<marque.début>', '<marque.fin>', '
↪<fichier>'[,<liste.tables.séquences>]);
```

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 l'instance postgreSQL puisse y écrire. Si le fichier existe déjà, son contenu sera écrasé.

Le dernier paramètre, optionnel, permet de filtrer la liste des tables et séquences à traiter. Si le paramètre est omis ou a la valeur *NULL*, toutes les tables et séquences du groupe de tables sont traitées. S'il est spécifié, le paramètre doit être exprimé sous la forme d'un tableau non vide d'éléments texte, chacun d'eux représentant le nom d'une table ou d'une séquence préfixé par le nom de schéma. On peut utiliser indifféremment les syntaxes

```
ARRAY['sch1.tbl1','sch1.tbl2']
```

ou

```
'{ "sch1.tbl1" , "sch1.tbl2" }'
```

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.

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

Prudence : Si une liste de tables et séquences est spécifiée pour restreindre le champ d'application de la fonction *emaj_gen_sql_group()*, il est de la responsabilité de l'utilisateur de prendre en compte l'existence éventuelle de clés étrangères (*foreign keys*) pour la validité du script SQL généré par la fonction.

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.

La fonction *emaj_gen_sql_groups()* permet de générer des scripts SQL portant sur plusieurs groupes de tables

```
SELECT emaj.emaj_gen_sql_groups('<tableau.des.groupes>', '<marque.début>', '<marque.
↪fin>', '<fichier>'[,<liste.tables.séquences>]);
```

Plus d'information sur les *fonctions multi-groupes*.

CHAPITRE 16

Autres fonctions

16.1 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 (plus de détails *ici*).

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.

16.2 Suivi des opérations de rollback en cours

Lorsque le volume de mises à jour à annuler rend un rollback long, il peut être intéressant de suivre l'opération afin d'en apprécier l'avancement. Une fonction, *emaj_rollback_activity()*, et un client *emajRollbackMonitor.php* répondent à ce besoin.

16.2.1 Pré-requis

Pour permettre aux administrateurs E-Maj de suivre la progression d'une opération de rollback, les fonctions activées dans l'opération mettent à jour plusieurs tables techniques au fur et à mesure de son avancement. Pour que ces mises à jour soient visibles alors que la transaction dans laquelle le rollback s'effectue est encore en cours, ces mises à jour sont effectuées au travers d'une connexion *dblink*.

Le suivi des rollbacks nécessite donc d'une part l'*installation de l'extension dblink*, et d'autre part l'enregistrement dans la table des paramètres, *emaj_param*, d'un identifiant de connexion utilisable par *dblink*.

L'enregistrement de l'identifiant de connexion peut s'effectuer au travers d'une requête du type

```
INSERT INTO emaj.emaj_param (param_key, param_value_text)
VALUES ('dblink_user_password','user=<user> password=<password>');
```

Le rôle de connexion déclaré doit disposer des droits *emaj_adm* (ou être super-utilisateur).

Enfin, la transaction principale effectuant l'opération de rollback doit avoir un mode de concurrence « *read committed* » (la valeur par défaut).

16.2.2 Fonction de suivi

La fonction *emaj_rollback_activity()* permet de visualiser les opérations de rollback en cours.

Il suffit d'exécuter la requête

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

La fonction ne requiert aucun paramètre en entrée.

Elle retourne un ensemble de lignes de type *emaj.emaj_rollback_activity_type*. Chaque ligne représente une opération de rollback en cours, comprenant les colonnes suivantes :

Column	Type	Description
rlbk_id	INT	identifiant de rollback
rlbk_groups	TEXT[]	tableau des groupes de tables associés au rollback
rlbk_mark	TEXT	marque de rollback
rlbk_mark_datetime	TIMESTAMPTZ	date et heure de pose de la marque de rollback
rlbk_is_logged	BOOLEAN	booléen prenant la valeur « vrai » pour les rollbacks annulables
rlbk_nb_session	INT	nombre de sessions en parallèle
rlbk_nb_table	INT	nombre de tables contenues dans les groupes de tables traités
rlbk_nb_sequence	INT	nombre de séquences contenues dans les groupes de tables traités
rlbk_eff_nb_table	INT	nombre de tables ayant eu des mises à jour à annuler
rlbk_status	ENUM	état de l'opération de rollback
rlbk_start_datetime	TIMESTAMPTZ	date et heure de début de l'opération de rollback
rlbk_elapse	INTERVAL	durée écoulée depuis le début de l'opération de rollback
rlbk_remaining	INTERVAL	durée restante estimée
rlbk_completion_pct	SMALLINT	estimation du pourcentage effectué

Une opération de rollback en cours est dans l'un des états suivants :

- PLANNING : l'opération est dans sa phase initiale de planification,
- LOCKING : l'opération est dans sa phase de pose de verrou,
- EXECUTING : l'opération est dans sa phase d'exécution des différentes étapes planifiées

Si les fonctions impliquées dans les opérations de rollback ne peuvent utiliser de connexion *dblink*, (extension *dblink* non installée, paramétrage de la connexion absente ou incorrect,...), la fonction *emaj_rollback_activity()* ne retourne aucune ligne.

L'estimation de la durée restante est approximative. Son degré de précision est similaire à celui de la fonction *emaj_estimate_rollback_group()*.

16.3 Mise à jour de l'état des rollbacks

La table technique *emaj_rlbk*, et ses tables dérivées, contient l'historique des opérations de rollback E-Maj.

Lorsque les fonctions de rollback ne peuvent pas utiliser une connexion *dblink*, toutes les mises à jour de ces tables techniques s'effectuent dans le cadre d'une unique transaction. Dès lors :

- toute transaction de rollback E-Maj qui n'a pu aller à son terme est invisible dans les tables techniques,
- toute transaction de rollback E-Maj qui a été validé est visible dans les tables techniques avec un état « *COMMITTED* » (validé).

Lorsque les fonctions de rollback peuvent utiliser une connexion *dblink*, toutes les mises à jour de la table technique *emaj_rlbk* et de ses tables dérivées s'effectuent dans le cadre de transactions indépendantes. Dans ce mode de fonctionnement, les fonctions de rollback E-Maj positionnent l'opération de rollback dans un état « *COMPLETED* » (terminé) en fin de traitement. Une fonction interne est chargée de transformer les opérations en état « *COMPLETED* », soit en état « *COMMITTED* » (validé), soit en état « *ABORTED* » (annulé), selon que la transaction principale ayant effectuée l'opération a ou non été validée. Cette fonction est automatiquement appelée lors de la pose d'une marque ou du suivi des rollbacks en cours,

Si l'administrateur E-Maj souhaite de lui-même procéder à la mise à jour de l'état d'opérations de rollback récemment exécutées, il peut à tout moment utiliser la fonction *emaj_cleanup_rollback_state()*

```
SELECT emaj.emaj_cleanup_rollback_state();
```

La fonction retourne le nombre d'opérations de rollback dont l'état a été modifié.

16.4 Désactivation/réactivation des triggers sur événements

L'installation de l'extension E-Maj créé et active des *triggers sur événements* pour la protéger. En principe, ces triggers doivent rester en l'état. Mais si l'administrateur E-Maj a besoin de les désactiver puis les réactiver, il dispose de deux fonctions.

Pour désactiver les triggers sur événement existants :

```
SELECT emaj.emaj_disable_protection_by_event_triggers();
```

La fonction retourne le nombre de triggers désactivés (cette valeur dépend de la version de PostgreSQL installée).

Pour réactiver les triggers sur événement existants :

```
SELECT emaj.emaj_enable_protection_by_event_triggers();
```

La fonction retourne le nombre de triggers réactivés.

CHAPITRE 17

Fonctions multi-groupes

17.1 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.

17.2 Liste des fonctions multi-groupes

Le tableau suivant liste les fonctions multi-groupes existantes et leur fonction mono-groupe jumelle. Certaines des fonctions mono-groupes sont présentées plus loin.

Fonctions multi-groups	Fonctions mono-groupe jumelles
emaj.emaj_alter_groups()	*emaj.emaj_alter_group()*
emaj.emaj_start_groups()	*emaj.emaj_start_group()*
emaj.emaj_stop_groups()	*emaj.emaj_stop_group()*
emaj.emaj_set_mark_groups()	*emaj.emaj_set_mark_group()*
emaj.emaj_rollback_groups()	*emaj.emaj_rollback_group()*
emaj.emaj_logged_rollback_groups()	*emaj.emaj_logged_rollback_group*
emaj.emaj_estimate_rollback_groups()	*emaj.emaj_estimate_rollback_group()*
emaj.emaj_gen_sql_groups()	*emaj.emaj_gen_sql_group()*

Les paramètres des fonctions multi-groupes sont les mêmes que ceux de leurs fonctions mono-groupe associées, à l'exception du premier. Le paramètre groupe de tables de type *TEXT* est remplacé par une paramètre de type *tableau de TEXT* représentant la liste des groupes de tables.

17.3 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.

17.4 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, en utilisant par exemple la fonction *array_agg()*.

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_group()*.

CHAPITRE 18

Client de 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.

18.1 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*.

18.2 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 l'instance 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.

18.3 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*
- -a spécifie que le rollback demandé est *autorisé à remonter à une marque antérieure à une modification de groupe de tables*
- -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_group()*.

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

Il est possible de suivre l'avancement des opérations de rollback multi-sessions de la même manière que celui des opérations de rollbacks mono-session.

Pour tester la commande emajParallelRollback.php, E-Maj fournit un script, *emaj_prepare_parallel_rollback_test.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.

18.4 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.

CHAPITRE 19

Client de suivi des rollbacks

E-Maj fournit un client externe qui se lance en ligne de commande et qui permet de suivre l'avancement des opérations de rollback en cours.

19.1 Préalables

La commande qui permet de suivre l'exécution des opérations de rollbacks 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 l'instance PostgreSQL).

19.2 Syntaxe

La syntaxe de la commande permettant le suivi des opérations de rollback est

```
emajRollbackMonitor.php [OPTIONS]...
```

Options générales :
- -i <intervalle de temps entre 2 affichages> (en secondes, défaut = 5s)
- -n <nombre d'affichages> (défaut = 1)
- -a <intervalle de temps maximum pour les opérations de rollback terminés à afficher> (en heures, défaut = 24h)
- -l <nombre maximum d'opérations de rollback terminés à afficher> (défaut = 3)
- –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>

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

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

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).

19.3 Exemples

La commande

```
./php/emajRollbackMonitor.php -i 3 -n 10
```

affiche 10 fois la liste des opérations de rollback en cours et celles des au plus 3 dernières opérations terminés depuis 24 heures, avec 3 secondes entre chaque affichage.

La commande

```
./php/emajRollbackMonitor.php -a 12 -l 10
```

affichera une seule fois la liste des opérations de rollback en cours et celle des au plus 10 opérations terminées dans les 12 dernières heures.

Exemple d'affichage de l'outil

```
 E-Maj (version 1.1.0) - Monitoring rollbacks activity
------------------------------------------------------------------
04/07/2013 - 12:07:17
** rollback 34 started at 2013-07-04 12:06:20.350962+02 for groups {myGroup1,myGroup2}
   status: COMMITTED ; ended at 2013-07-04 12:06:21.149111+02
** rollback 35 started at 2013-07-04 12:06:21.474217+02 for groups {myGroup1}
   status: COMMITTED ; ended at 2013-07-04 12:06:21.787615+02
-> rollback 36 started at 2013-07-04 12:04:31.769992+02 for groups {group1232}
   status: EXECUTING ; completion 89 % ; 00:00:20 remaining
-> rollback 37 started at 2013-07-04 12:04:21.894546+02 for groups {group1233}
   status: LOCKING ; completion 0 % ; 00:22:20 remaining
-> rollback 38 started at 2013-07-04 12:05:21.900311+02 for groups {group1234}
   status: PLANNING ; completion 0 %
```

CHAPITRE 20

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 = 20 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 à l'exécution d'une fonction *emaj_estimate_rollback_group()*.
- **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 à l'exécution d'une fonction *emaj_estimate_rollback_group()*.
- **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 à l'exécution d'une fonction *emaj_estimate_rollback_group()*.
- **fixed_dblink_rollback_duration** : (intervalle) valeur par défaut = 4 ms ; définit un coût additionnel pour chaque étape de rollback quand une connexion dblink est utilisée ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données à l'exécution d'une fonction *emaj_estimate_rollback_group()*.
- **fixed_table_rollback_duration** : (intervalle) valeur par défaut = 1 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 à l'exécution d'une fonction *emaj_estimate_rollback_group()*.

— **fixed_step_rollback_duration** : (intervalle) valeur par défaut = 2,5 ms ; définit un coût fixe pour chaque étape de rollback ; peut être modifiée pour mieux représenter la performance du serveur qui héberge la base de données à l'exécution d'une fonction *emaj_estimate_rollback_group()*.
— **history_retention** (intervalle) valeur par défaut = 1 an ; elle peut être ajustée pour changer la durée de rétention des lignes dans la table historique d'E-Maj, *emaj_hist*,

Exemple de requête SQL permettant de spécifier une durée de rétention des lignes dans l'historique de 3 mois :

```
INSERT INTO emaj.emaj_param (param_key, param_value_interval) VALUES ('history_
↪retention','3 months'::interval);
```

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

Seuls les super-utilisateurs et les utilisateurs ayant acquis les droits *emaj_adm* ont accès à la table *emaj_param*.

Les utilisateurs ayant acquis les droits *emaj_viewer* n'ont accès qu'à une partie de la table *emaj_param*. au travers de la vue *emaj.emaj_visible_param*. Cette vue masque simplement le contenu réel de la colonne *param_value_text* pour la clé *'dblink_user_password'*.

CHAPITRE 21

Fiabilisation du fonctionnement

Deux éléments complémentaires concourent à la fiabilité de fonctionnement d'E-Maj : des contrôles internes effectués à certains moments clé de la vie des groupes de tables, et l'activation de triggers sur événement bloquant certaines opérations risquées.

21.1 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, sa fonction de log ainsi que ses triggers,
- la structure des tables de log est toujours en phase avec celle des tables applicatives associées,
- aucune table n'a été transformée en table *UNLOGGED* ou *WITH OIDS*,
- pour les groupes de tables rollbackable, les tables applicatives ont toujours leur clé primaire.

En utilisant la fonction *emaj_verify_all()*, l'administration peut effectuer à la demande ces mêmes contrôles sur l'ensemble des groupes de tables.

21.2 Triggers sur événements

L'installation d'E-Maj dans des instances PostgreSQL de version 9.3 et suivantes inclut la création de 2 triggers sur événements de type « *sql_drop* » :

- *emaj_sql_drop_trg* bloque la suppression :
  - de tout objet E-Maj (table de logs, séquence de log, fonction de log, trigger de log, schéma secondaire),
  - de toute table ou séquence applicatives appartenant à un groupe de tables en état « *LOGGING* »,
  - de tout schéma contenant au moins une table ou séquence appartenant à un groupe de tables en état « *LOGGING* ».
- *emaj_protection_trg* bloque la suppression de l'extension *emaj* elle-même et du schéma principal *emaj*.

L'installation d'E-Maj dans des instances PostgreSQL de version 9.5 et suivantes inclut la création d'un 3ème trigger sur événements, de type « *table_rewrite* ».

— *emaj_table_rewrite_trg* bloque tout changement de structure de table applicative ou de table de log.

Il est possible de désactiver/réactiver ces triggers grâce aux deux fonctions : *emaj_disable_protection_by_event_triggers()* et *emaj_enable_protection_by_event_triggers()*.

Les protections mises en place ne couvrent néanmoins pas tous les risques. En particulier, le renommage de tables ou de séquences ou leur changement de schéma d'appartenance ne sont pas couverts ; et certaines requêtes changeant la structure d'une table ne déclenchent aucun trigger.

CHAPITRE 22

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, 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.

Valeur	Signification
ALTER_GROUP	modification d'un groupe de tables
ALTER_GROUPS	modification de plusieurs groupes de tables
CLEANUP_RLBK_STATE	nettoyage du code état des opérations de rollback récemment terminées
COMMENT_GROUP	positionnement d'un commentaire sur un groupe
COMMENT_MARK_GROUP	positionnement d'un commentaire sur une marque
CONSOLIDATE_RLBK_GROUP	consolide une opération de rollback tracé
CREATE_GROUP	création d'un groupe de tables
DBLINK_OPEN_CNX	ouverture d'une connexion dblink pour un rollback
DBLINK_CLOSE_CNX	fermeture d'une connexion dblink pour un rollback
DELETE_MARK_GROUP	suppression d'une marque pour un groupe de tables
DISABLE_EVENT_TRIGGERS	désactivation des triggers sur événements
DROP_GROUP	suppression d'un groupe de tables
EMAJ_INSTALL	installation ou mise à jour de la version d'E-Maj
ENABLE_EVENT_TRIGGERS	activation des triggers sur événements

Suite sur la page suivante

Tableau 22.1 – Suite de la page précédente

Valeur	Signification
FORCE_DROP_GROUP	suppression forcée d'un groupe de tables
FORCE_STOP_GROUP	arrêt forcé d'un groupe de tables
GEN_SQL_GROUP	génération d'un script psql pour un groupe de tables
GEN_SQL_GROUPS	génération d'un script psql pour plusieurs groupes de tables
LOCK_GROUP	pose d'un verrou sur les tables d'un groupe
LOCK_GROUPS	pose d'un verrou sur les tables de plusieurs groupes
LOCK_SESSION	pose d'un verrou sur les tables d'une session de rollback
PROTECT_GROUP	pose d'une protection contre les rollbacks sur un groupe
PROTECT_MARK_GROUP	pose d'une protection contre les rollbacks sur une marque d'un groupe
PURGE_HISTORY	suppression dans la table *emaj_hist* des événements antérieurs au délai de rétention
RENAME_MARK_GROUP	renommage d'une marque pour un groupe de tables
RESET_GROUP	réinitialisation du contenu des tables de log d'un groupe
ROLLBACK_GROUP	rollback des mises à jour pour un groupe de tables
ROLLBACK_GROUPS	rollback des mises à jour pour plusieurs groupes de tables
ROLLBACK_SEQUENCE	rollback d'une séquence
ROLLBACK_TABLE	rollback des mises à jour d'une table
SET_MARK_GROUP	pose d'une marque pour un groupe de tables
SET_MARK_GROUPS	pose d'une marque pour plusieurs groupes de tables
SNAP_GROUP	vidage des tables et séquences d'un groupe
SNAP_LOG_GROUP	vidage des tables de log d'un groupe
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
UNPROTECT_GROUP	suppression d'une protection contre les rollbacks sur un groupe
UNPROTECT_MARK_GROUP	suppression d'une protection contre les rollbacks sur une marque d'un groupe

La colonne *hist_event* peut prendre les valeurs suivantes.

Valeur	Signification
BEGIN	début
END	fin
EVENT TRIGGERS DISABLED	triggers sur événements désactivés
EVENT TRIGGERS ENABLED	triggers sur événements activés
MARK DELETED	marque supprimée
SCHEMA CREATED	schéma secondaire créé
SCHEMA DROPPED	schéma secondaire supprimé
TABLE ATTR CHANGED	attributs E-Maj pour la table modifiés
LOG SCHEMA CHANGED	Schéma de log modifié
NAMES PREFIX CHANGED	Préfixe des noms E-Maj modifié
LOG DATA TABLESPACE CHANGED	Tablespace pour la table de log modifié
LOG INDEX TABLESPACE CHANGED	Tablespace pour l'index de log modifié

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

Deux autres tables internes conservent également des traces des opérations effectuées :

— *emaj_alter_plan* liste les actions élémentaires réalisées lors de l'exécution d'opérations de *modification de groupes de tables*,

— *emaj_rlbk_plan* liste les actions élémentaires réalisées lors de l'exécution d'opérations de *rollback E-Maj*.

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 événe-

ments conservés sont ceux à la fois postérieurs à un délai de rétention paramétrable, postérieurs à la pose de la plus ancienne marque active et postérieurs à la plus ancienne opération de rollback non terminée. Par défaut, la durée de rétention des événements est de 1 an. Mais cette valeur peut être modifiée à tout moment en insérant par une requête SQL le paramètre *history_retention* dans la table *emaj_param*. La même rétention s'applique aux contenus des tables qui historisent les actions élémentaires des opérations de modification ou de rollback de groupes de tables.

CHAPITRE 23

Impacts sur l'administration de l'instance et de la base de données

23.1 Arrêt/relance de l'instance

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

23.1.1 Règle générale

Au redémarrage de l'instance, tous les objets d'E-Maj se retrouvent dans le même état que lors de l'arrêt de l'instance : 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 de l'instance, 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.

23.1.2 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*. (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 de l'instance, il est recommandé de procéder à nouveau à ce même rollback juste après le redémarrage de l'instance, afin de s'assurer que les séquences et tables applicatives restent bien en phase.

23.2 Sauvegarde et restauration

Prudence : 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 !

23.2.1 Sauvegarde et restauration au niveau fichier

Lors des sauvegardes ou des restauration des instances au niveau fichier, il est essentiel de sauver ou restaurer **TOUS** les fichiers e l'instance, y compris ceux stockés sur des tablespaces dédiés.

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.

23.2.2 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*.

23.2.3 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.

23.3 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.

23.4 Réorganisation des tables de la base de données

23.4.1 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.

23.4.2 Réorganisation des tables E-Maj

L'index correspondant à la clé primaire de chaque table des schémas d'E-Maj est déclaré « *cluster* », que ce soit les tables de log ou les quelques tables internes.

Prudence : Aussi, l'installation d'E-Maj peut avoir un impact opérationnel sur l'exécution des commandes SQL *CLUSTER* au niveau de la base de données.

Dans le cas d'une utilisation en mode continu d'E-Maj, c'est à dire sans arrêt et relance réguliers des groupes de tables, mais avec suppression des marquess les plus anciennes, il est recommandé de procéder régulièrement à des réorganisations des tables de log E-Maj. Ceci permet ainsi de récupérer de l'espace disque inutilisé suite aux suppressions des marques.

23.5 Utilisation d'E-Maj avec de la réplication

23.5.1 Réplication physique intégrée

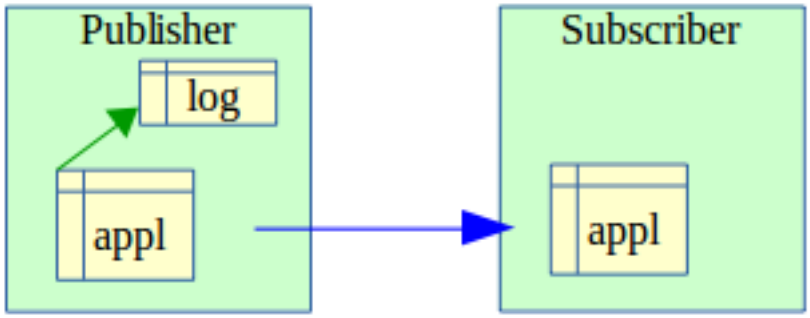
E-Maj est parfaitement compatible avec le fonctionnement des différents modes de réplication physique intégrée de PostgreSQL (archivage des *WAL* et *PITR*, *Streaming Replication* asynchrone ou synchrone). Tous les objets E-Maj des bases hébergées sur l'instance sont en effet répliqués comme toutes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

Néanmoins, compte tenu de la façon dont PostgreSQL gère les séquences, la valeur courante des séquences peut être un peu en avance sur les instances secondaires par rapport à l'instance 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.

23.5.2 Réplication logique intégrée

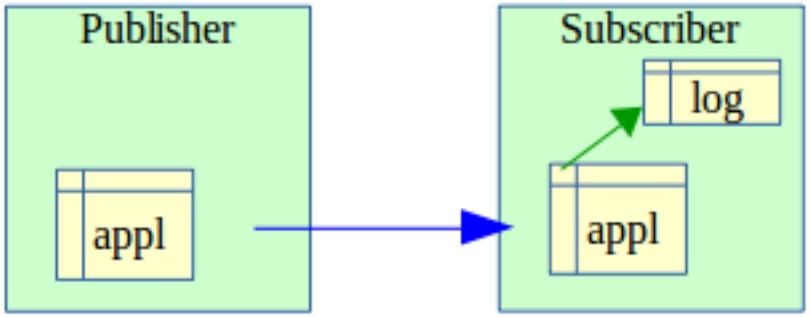
Les versions 10 et suivantes de PostgreSQL intègrent des mécanismes de réplication logique, La granularité de réplication est ici la table. L'objet de publication utilisé dans la réplication logique est assez proche du concept de groupes de tables E-Maj, à ceci près qu'une publication ne peut contenir de séquences.

Réplication de tables applicatives gérées par E-Maj



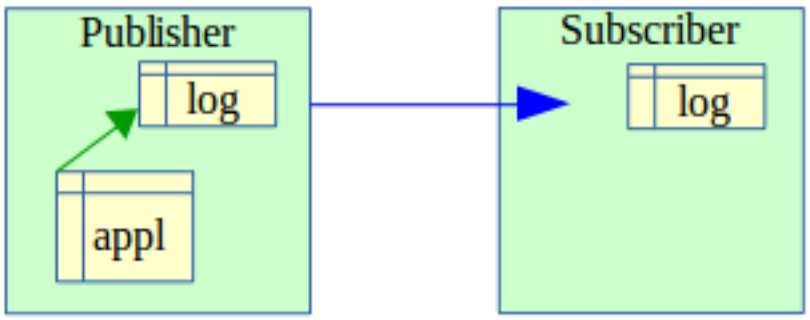
Une table applicative appartenant à un groupe de tables E-Maj peut être également mise en réplication, sans condition particulière. Les éventuels rollbacks E-Maj se répliqueront naturellement côté *subscriber*.

Réplication de tables applicatives avec gestion par E-Maj côté subscriber



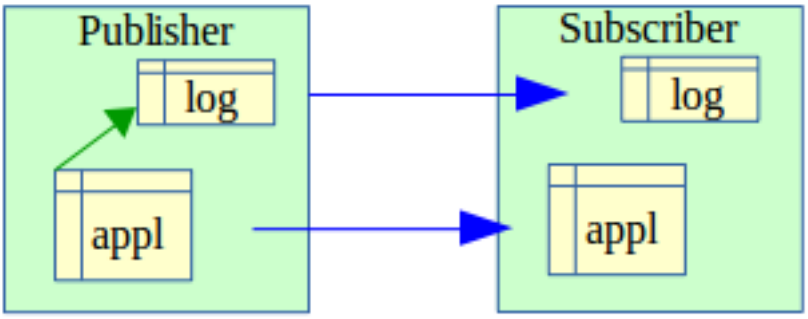
Il est possible d'insérer une table applicative dans un groupe de tables E-Maj avec des mises à jour en provenance d'un flux de réplication. Mais toutes les opérations E-Maj sont bien sûr exécutées côté *subscriber* (démarrage/arrêt du groupe, pose de marque,...). On peut effectuer un rollback E-Maj de ce groupe de tables, une fois stoppée la réplication (pour éviter des conflits dans les mises à jour). Mais à l'issue du rollback, les tables du *publisher* et du *subscriber* ne seront plus en cohérence.

Réplication de tables de log E-Maj



Il est techniquement possible de mettre une table de log E-Maj en réplication (en trouvant un moyen de construire le DDL de création – par *pg_dump* par exemple). Ceci peut permettre de dupliquer ou concentrer les données de log sur un autre serveur. Mais la table de log répliquée ne peut être utilisée qu'en **consultation**. En effet, les séquences de log et les verbes de *TRUNCATE* n'étant pas répliqués, ces logs ne peuvent pas être utilisés à d'autres fins.

Réplication de tables applicatives et de tables de log E-Maj



Tables applicatives et tables de log peuvent être répliquées simultanément. Mais comme dans le cas précédent, ces logs ne sont utilisables qu'à des fins de **consultation**. Les éventuelles opérations de rollback E-Maj ne peuvent s'effectuer que côté *publisher*.

23.5.3 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.

CHAPITRE 24

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 l'instance 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()*.

CHAPITRE 25

Performances

25.1 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 pourcents. Mais ce surcoût est à mettre en relation avec la durée des éventuelles sauvegardes intermédiaires de base de données évitées.

25.2 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 les fonctions *emaj_estimate_rollback_group()* et *emaj_estimate_rollback_groups()*.

25.3 Optimiser le fonctionnement d'E-Maj

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

25.3.1 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 positionnant un tablespace par défaut pour sa session courante 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.

25.3.2 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.

CHAPITRE 26

Limites d'utilisation

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

- La **version PostgreSQL** minimum requise est la version 9.1.
- Toutes les tables appartenant à un groupe de tables de type « *rollbackable* » doivent avoir une **clé primaire** explicite (*PRIMARY KEY*).
- Les tables *TEMPORARY*, *UNLOGGED* ou *WITH OIDS* ne sont pas gérées par E-Maj
- 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. Un trigger, créé par E-Maj, empêche l'exécution d'une requête *TRUNCATE* sur toute table appartenant à groupe de tables en état démarré.
- 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.
- 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 marque 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.

CHAPITRE 27

Responsabilités de l'utilisateur

27.1 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.

27.2 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. Il doit également apporter une attention particulière aux éventuelles messages d'avertissement rapportés par les fonctions de rollback.

27.3 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.

27.4 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.

Prudence : 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.

CHAPITRE 28

Présentation générale des clients web

Pour faciliter l'utilisation d'E-Maj, deux applications web sont disponibles :
- un « **plug-in** » pour l'outil d'administration **phpPgAdmin**, dans ses versions 5.1 et suivantes,
- une application web indépendante, **Emaj_web**.

Les deux clients web offrent les mêmes fonctionnalités autour d'E-Maj, avec une interface utilisateur similaire.

Emaj_web empreinte à *phpPgAdmin* son infrastructure (browser, barre d'icones, connexion aux bases de données,...) et quelques fonctions utiles telles que la consultation du contenu de tables ou la saisie de requêtes SQL.

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, tous les objets E-Maj sont visibles et manipulables.

Il est ainsi possible de :
- définir ou modifier la composition des groupes,
- voir la liste des groupes de tables et effectuer toutes les actions possibles, en fonction de l'état du groupe (création, suppression, démarrage, arrêt, pose de marque, rollback, ajout ou modification de commentaire),
- 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 et en visualiser le contenu,
- suivre les opérations de rollbacks en cours d'exécution.

CHAPITRE 29

Installation du plug-in phpPgAdmin

29.1 Pré-requis

Une version de *phpPgAdmin*, de version au moins égale à 5.1, doit être installée et opérationnelle dans un serveur web.

29.2 Téléchargement du plug-in

Le plug-in E-Maj pour *phpPgAdmin* peut être téléchargé depuis le dépôt git suivant : https://github.com/beaud76/emaj_ppa_plugin

Le répertoire *Emaj* téléchargé doit être copié dans le sous-répertoire *plugins* de la version *phpPgAdmin* utilisée.

29.3 Activation du plug-in

Pour activer le plug-in, il suffit d'ouvrir le fichier *conf/config.inc.php* de l'arborescence de *phpPgAdmin* et d'ajouter la chaîne de caractères *'Emaj'* à la variable *$conf['plugins']*.

On peut ainsi avoir par exemple

```
$conf['plugins'] = array('Emaj');
```

ou encore, si un autre plug-in est déjà activé

```
$conf['plugins'] = array('Report','Emaj');
```

29.4 Paramétrage du plug-in

Pour pouvoir soumettre des rollbacks en tâche de fonds (c'est à dire sans mobiliser le navigateur durant le déroulement des rollbacks), il est nécessaire de valoriser deux paramètres de configuration contenus dans le fichier

Emaj/conf/config.inc.php :

- *$plugin_conf['psql_path']* définit le chemin de l'exécutable *psql*,
- *$plugin_conf['temp_dir']* définit un répertoire temporaire utilisable lors des rollbacks en tâche de fonds.

Le fichier *config.inc.php-dist* fourni peut-être utilisé comme modèle de fichier de configuration.

CHAPITRE 30

Installation du client Emaj_web

30.1 Pré-requis

Emaj_web nécessite un serveur web avec un interpréteur php.

30.2 Téléchargement du plug-in

L'application *Emaj_web* peut être téléchargée depuis le dépôt git suivant :

https://github.com/beaud76/emaj_web

30.3 Configuration de l'application

Deux fichiers de configuration sont à renseigner.

30.3.1 Paramétrage générale

Le fichier *emaj_web/conf/config.inc.php* contient les paramètres généraux de l'application, incluant notamment la description des connexions aux instances PostgreSQL.

Le fichier *emaj_web/conf/config.inc.php-dist* peut servir de base pour la configuration.

30.3.2 Paramétrage du plug-in

Comme *Emaj_web* utilise lui même le plugin pour *phpPgAdmin*, les paramètres spécifiques à ce plugin sont décrits dans un fichier de configuration séparé : *emaj_web/plugins/Emaj/conf/config.inc.php*

Pour pouvoir soumettre des rollbacks en tâche de fonds (c'est à dire sans mobiliser le navigateur durant le déroulement des rollbacks), il est nécessaire de valoriser deux paramètres de configuration contenus dans le fichier *Emaj/conf/config.inc.php* :

— *$plugin_conf['psql_path']* définit le chemin de l'exécutable *psql*,

— *$plugin_conf['temp_dir']* définit un répertoire temporaire utilisable lors des rollbacks en tâche de fonds.

Le fichier *config.inc.php-dist* fourni peut-être utilisé comme modèle de fichier de configuration.

CHAPITRE 31

Utilisation des clients web

31.1 Accès à E-Maj dans l'interface phpPgAdmin

Une fois 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 rouge apparaît à droite dans la barre d'icônes horizontale de la base. Bien sûr, le schéma *emaj* figure dans la liste des schémas.

Dans l'arborescence 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.

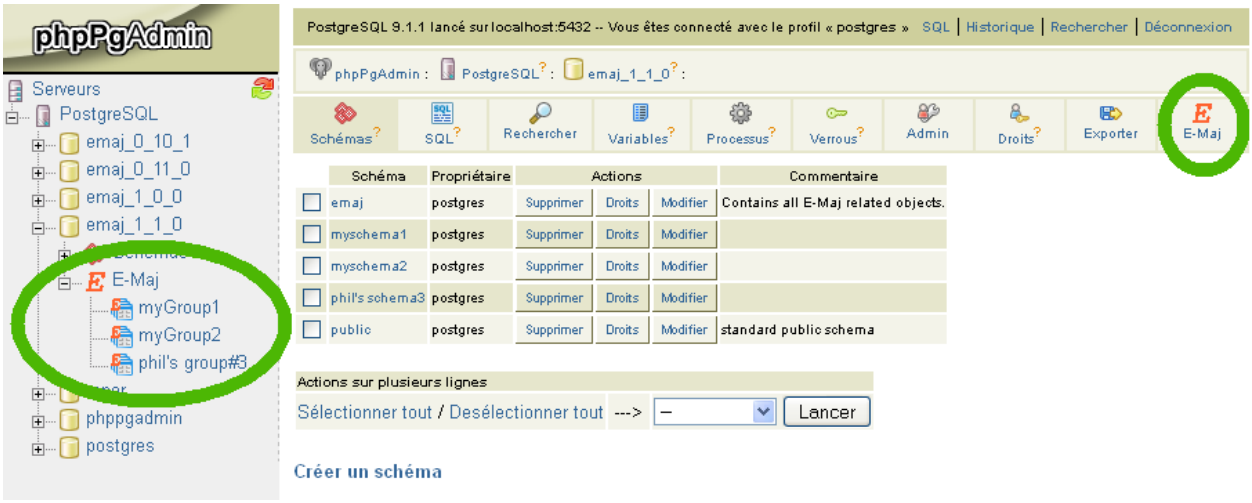


Fig. 31.1 – Figure 1a – *phpPgAdmin* : connexion à une base de données où E-Maj est installé.

31.2 Accès à Emaj_web

La connexion à une base de données est similaire à *phpPgAdmin*.

Une fois 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*), l'icône rouge à droite dans la barre d'icônes horizontale de la base permet d'accéder aux fonctions spécifiques d'E-Maj.

Dans l'arborescence de gauche, l'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.

Fig. 31.2 – Figure 1b – *Emaj_web* : connexion à une base de données où E-Maj est installé.

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

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* », protégé contre les rollbacks ou non),
- le nombre de marques qu'il possède,
- son éventuel commentaire associé.

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

Sous chacune des deux listes, une liste déroulante et un bouton permettent d'effectuer certaines actions sur plusieurs groupes simultanément.

Au bas de la page, une liste déroulante présente les groupes de tables susceptibles d'être créés (ceux référencés dans la table *emaj_group_def* mais qui ne sont pas encore créés).

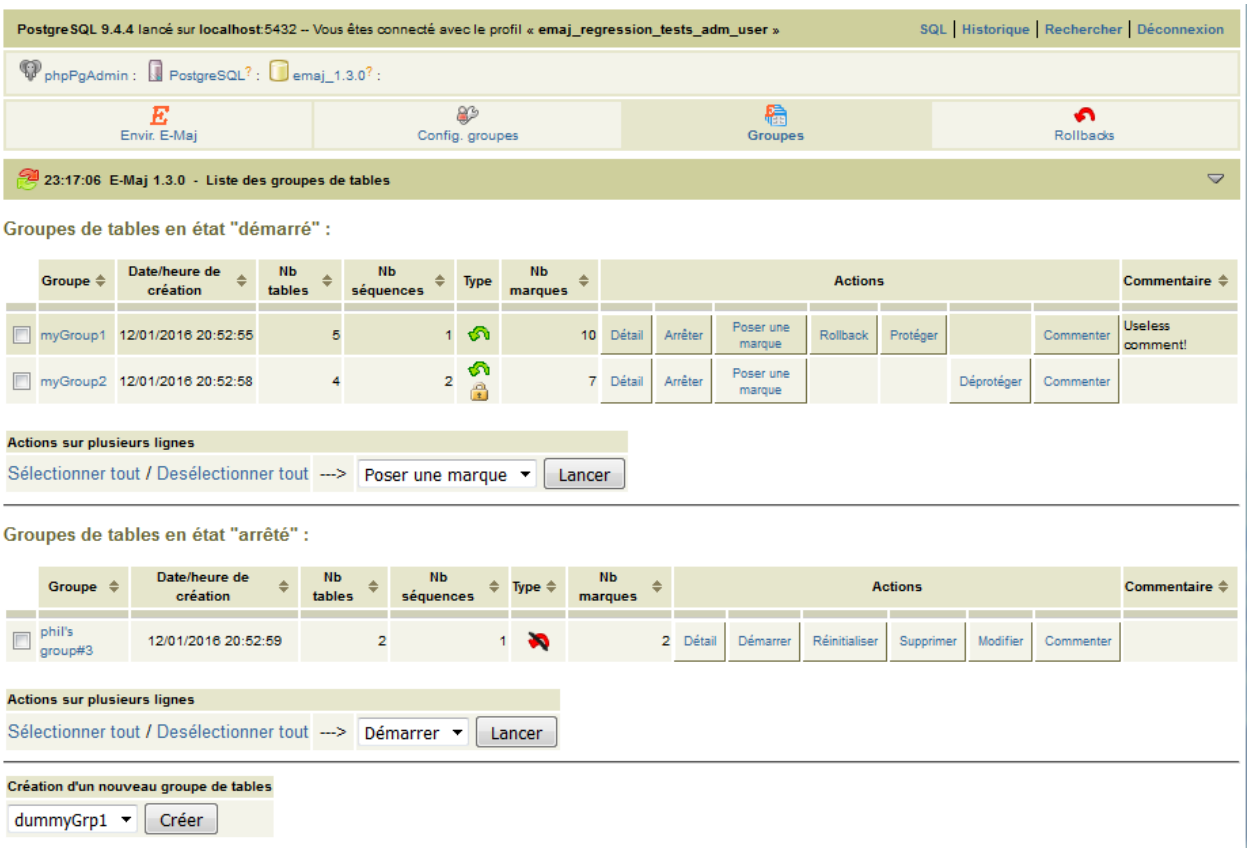


Fig. 31.3 – Figure 2 – Liste des groupes de tables créés sur la base de données.

31.4 Quelques détails de l'interface utilisateur

Deux barres d'icônes permettent de naviguer dans les différentes fonctions d'E-Maj : l'une regroupe les fonctions globales de l'interface, et l'autre les fonctions associées à un groupe de tables particulier.

Fig. 31.4 – Figure 3 – Barre d'icônes principale.

Fig. 31.5 – Figure 4 – Barre d'icônes des groupes de tables.

Pour les rôles de type *emaj_viewer* certaines icônes ne sont pas visibles.

Toutes les pages affichées par le plug-in E-Maj ont une entête qui contient :
- un bouton pour rafraîchir la page courante,
- l'heure d'affichage de la page courante,
- la version d'E-Maj installée dans la base de données,
- le titre de la page,
- un bouton permettant d'atteindre le bas de la page, à l'extrême droite de l'entête.

Sur certains tableaux, il est possible de trier en dynamique les lignes affichées à l'aide de petites flèches verticales situées à droite des titres de colonnes. Sur certains tableaux également, le passage de la souris sur la ligne grise située juste au dessous de la ligne de titre laisse apparaître des champs de saisie permettant le filtrage des lignes affichées.

Fig. 31.6 – Figure 5 – Filtrage des groupes démarrés. Ne sont affichés ici que les groupes de tables dont le nom comprend « *my* » et contenant plus de 2 marques, cette liste étant triée par ordre décroissant du nombre de tables.

31.5 État de l'environnement E-Maj

En cliquant sur l'icône « *Envir. E-Maj* » de la barre principale, l'utilisateur accède à une synthèse de l'état de l'environnement E-Maj.

Sont d'abord restitués :
- la version d'E-Maj installée,
- la place disque occupée par E-Maj (tables de log, tables techniques et index associés) et la part que cela représente dans la taille globale de la base de données.

Puis l'intégrité de l'environnement est testé ; le résultat de l'exécution de la fonction *emaj_verify_all()* est affiché.

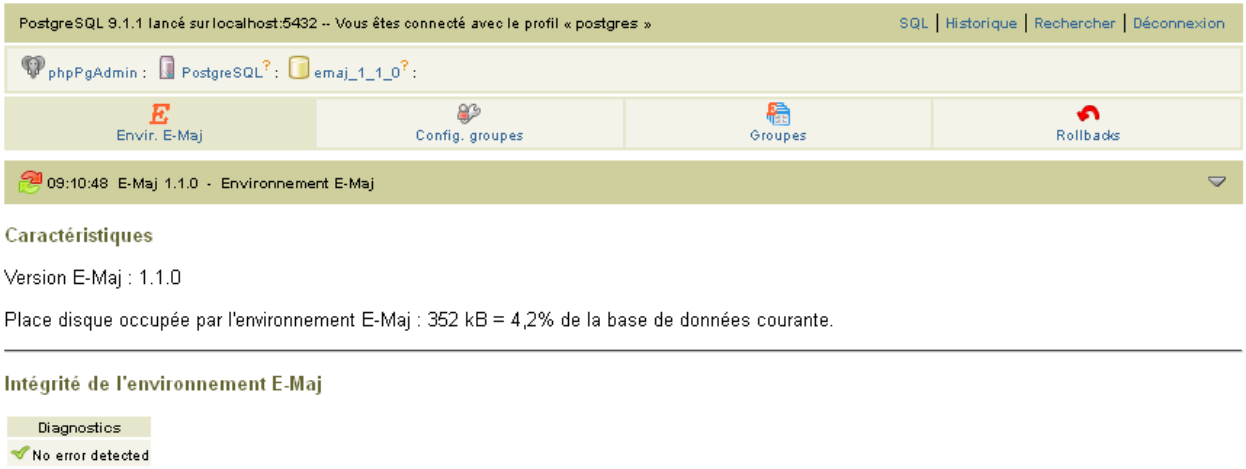


Fig. 31.7 – Figure 6 – État de l'environnement E-Maj

31.6 Composition des groupes de tables

Grâce à l'icône « *Config. Groupes* » de la barre principale, 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.

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.

Sont listés pour chaque table ou séquence :
- son type,
- le groupe de table auquel il appartient, s'il y en a un,
- les attributs de la table ou de la séquence dans *emaj_group_def*, si elle est déjà affectée à un groupe :
  - le niveau de priorité affecté dans le groupe,
  - le suffixe définissant le schéma de log,
  - le préfixe éventuel des noms des objets E-Maj associés à la table,
  - le nom du tablespace éventuel supportant la table de log,
  - le nom du tablespace éventuel supportant l'index de la table de log,
- son propriétaire,
- le tablespace auquel elle est rattachée, s'il y en a un,
- son commentaire enregistré dans la base de données.

Les deux listes de schémas et de tables et séquences affichent 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 première colonne de chaque tableau.

A l'aide de boutons, il est possible :
- d'assigner une table ou une séquence à un groupe de tables nouveau ou existant,
- de modifier les propriétés de la table ou de la séquence dans son groupe de tables,
- de détacher une table ou une séquence de son groupe de tables.

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

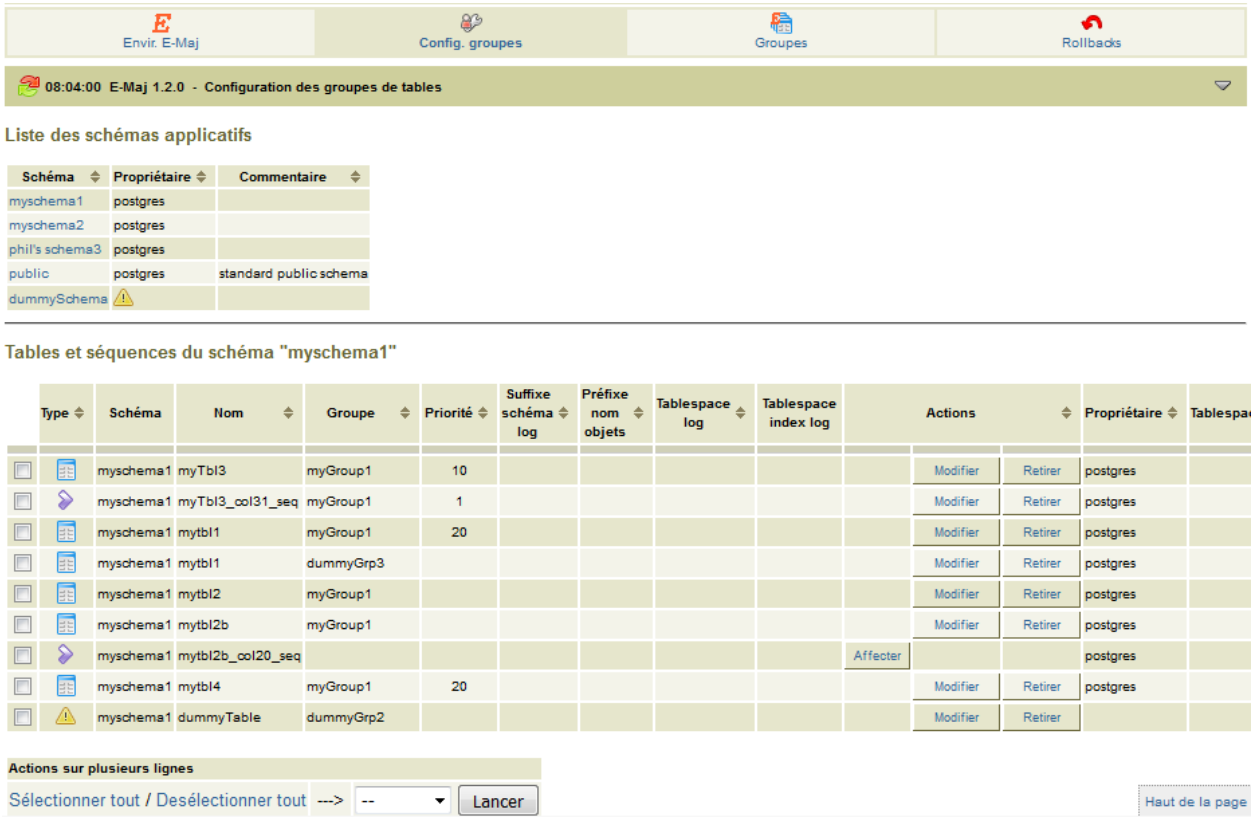


Fig. 31.8 – Figure 7 – Composition des groupes de tables.

31.7 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* ». Cette page est aussi accessible par l'icône « *Propriétés* » de la barre des groupes ou par l'arborescence de gauche.

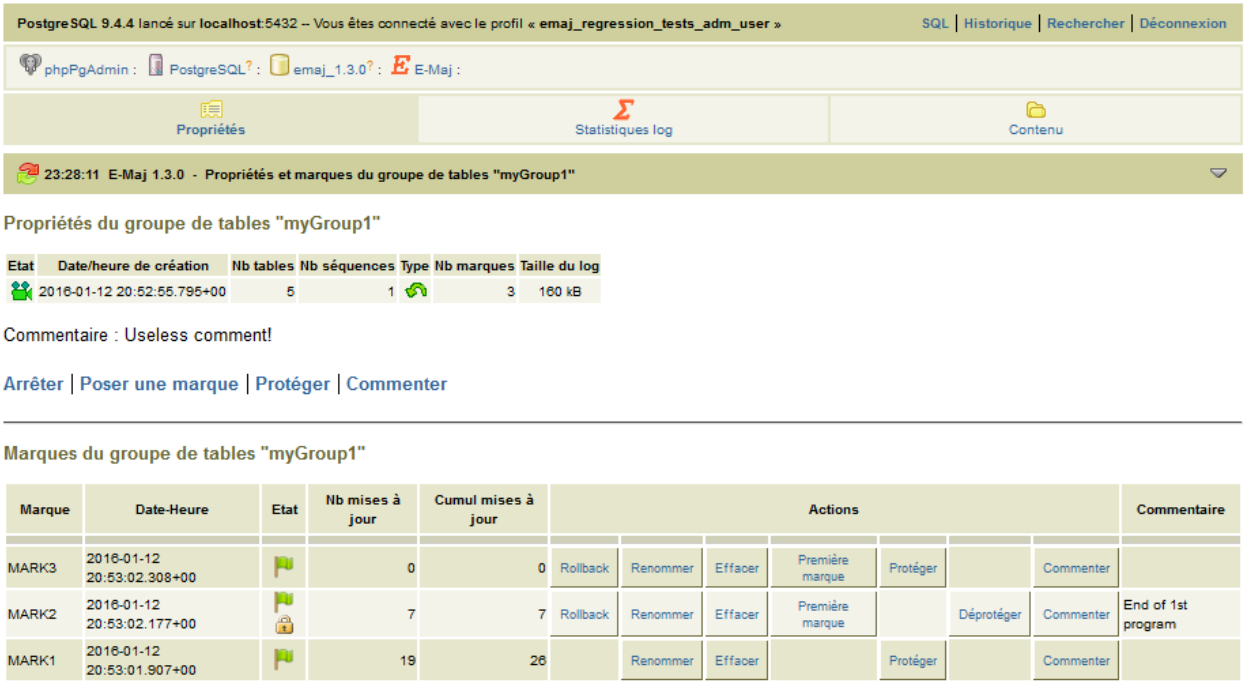


Fig. 31.9 – Figure 8 – Détail d'un groupe de tables

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 du groupe.

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

Puis une liste de liens permet de réaliser les actions que l'état du groupe permet.

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 (actif ou non, protégé contre les rollbacks ou non),
- 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.

31.8 Statistiques

L'icône « *Statistiques log* » de la barre des groupes permet d'obtenir des statistiques sur le contenu des mises à jour enregistrées dans les tables de log pour le groupe de tables.

Deux types de statistiques peuvent être obtenues :

— des estimations du nombre de mises à jour par table, enregistrées entre 2 marques ou entre une marque et la situation présente,
— un dénombrement précis du nombre de mises à jour par table, type de requête (*INSERT/UPDATE/DELETE/TRUNCATE*) et rôle.

Si la borne de fin correspond à la situation courante, une case à cocher permet de demander en même temps une simulation de rollback à la première marque sélectionnée afin d'obtenir rapidement une durée approximative d'exécution de cet éventuel rollback.

La figure suivante montre un exemple de statistiques détaillées.

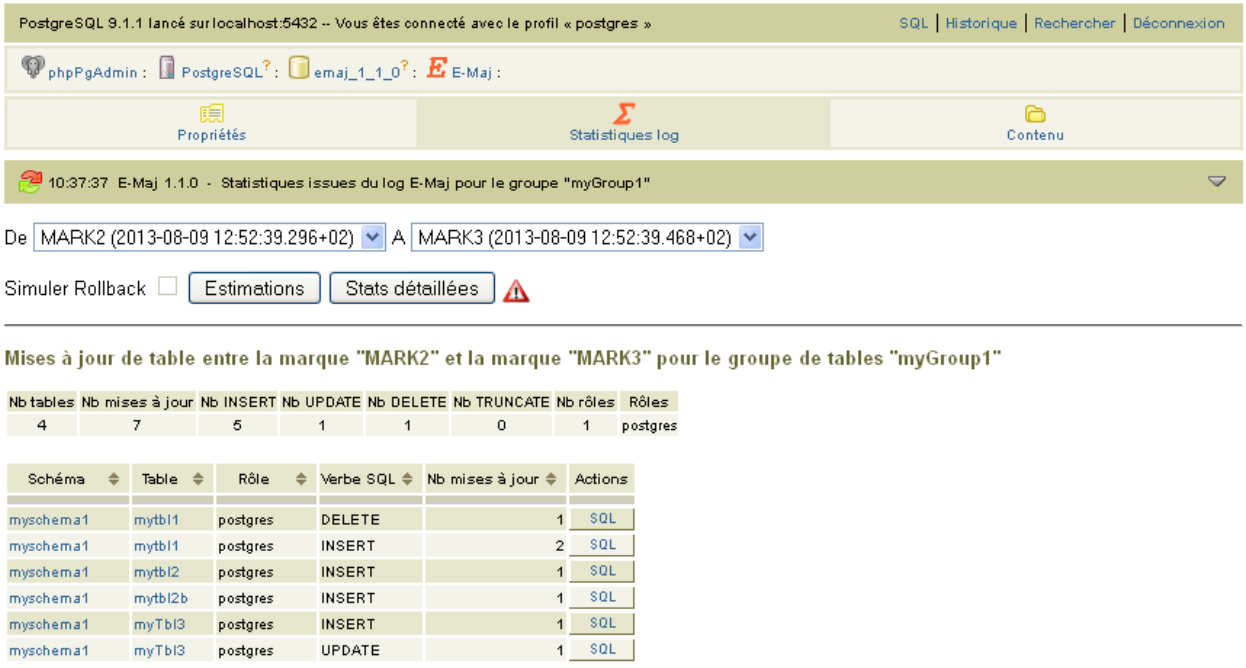


Fig. 31.10 – Figure 9 – Statistiques détaillées des mises à jour enregistrées entre 2 marques

La page restituée contient une première ligne contenant des compteurs globaux.

Sur chacune des lignes du tableau de statistiques, un bouton « *SQL* » permet à l'utilisateur de visualiser facilement le contenu des mises à jour enregistrées dans les tables de log. Un clic sur ce bouton ouvre l'éditeur de requêtes SQL et propose la requête visualisant le contenu de la table de log correspondant à la sélection (table, tranche de temps, rôle, type de requête). L'utilisateur peut la modifier à sa convenance avant de l'exécuter, afin, par exemple, de cibler davantage les lignes qui l'intéressent.

La page restituée contient une première partie indiquant le nombre de tables et de mises à jour concernées par un éventuel rollback à cette marque et une estimation du temps nécessaire à ce rollback.

31.9 Contenu d'un groupe de tables

L'icône « *Contenu* » de la barre des groupes permet d'obtenir une vision synthétique du contenu d'un groupe de tables.

Le tableau affiché reprend, pour chaque table et séquence du groupe, les caractéristiques configurées dans la table *emaj_group_def*, ainsi que la place prise par la table de log et son index.

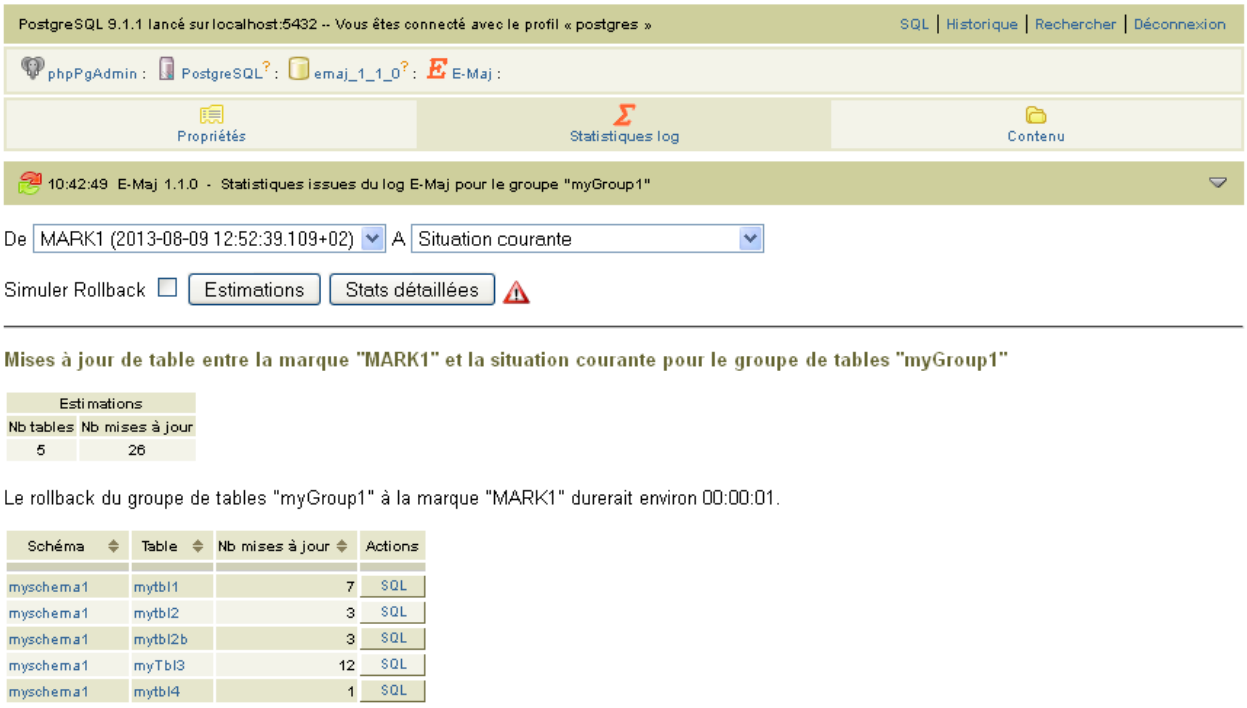


Fig. 31.11 – Figure 10 – Résultat de la simulation d'un rollback avec estimation du nombre de mises à jour par table.

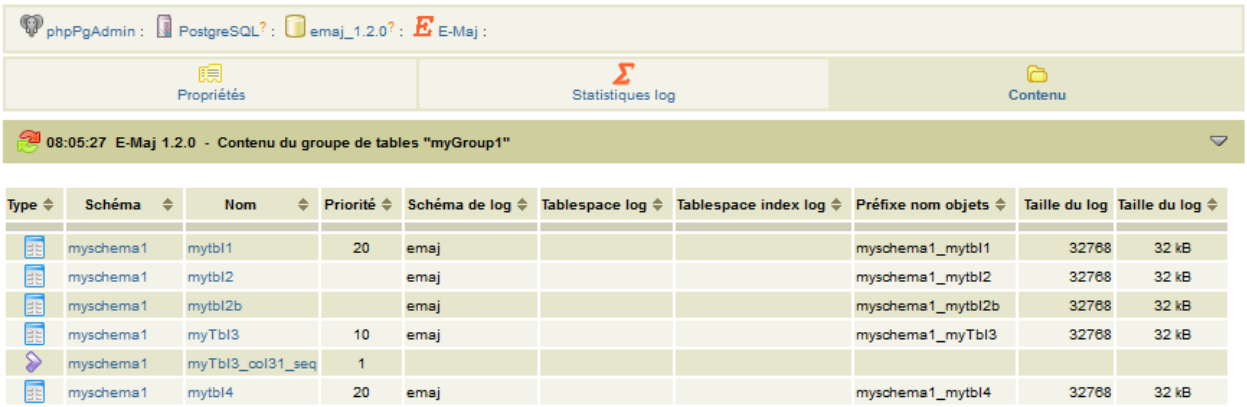


Fig. 31.12 – Figure 11 – Contenu d'un groupe de tables.

31.10 Suivi des opérations de rollback

Une page, accessible par l'icône « *Rollbacks* » de la barre globale, permet de suivre les opérations de rollback. Trois listes distinctes sont affichées :

- les opérations de rollback en cours, avec le rappel des caractéristiques de l'opération et une estimation de la part de l'opération déjà effectuée et de la durée restante,
- les dernières opérations de rollback terminées,
- les opérations de rollback tracés susceptibles d'être consolidées.

L'utilisateur peut filtrer la liste des rollbacks terminés sur une profondeur d'historique plus ou moins grande.

Pour chaque rollback tracé consolidable listé, un bouton permet d'exécuter la consolidation.

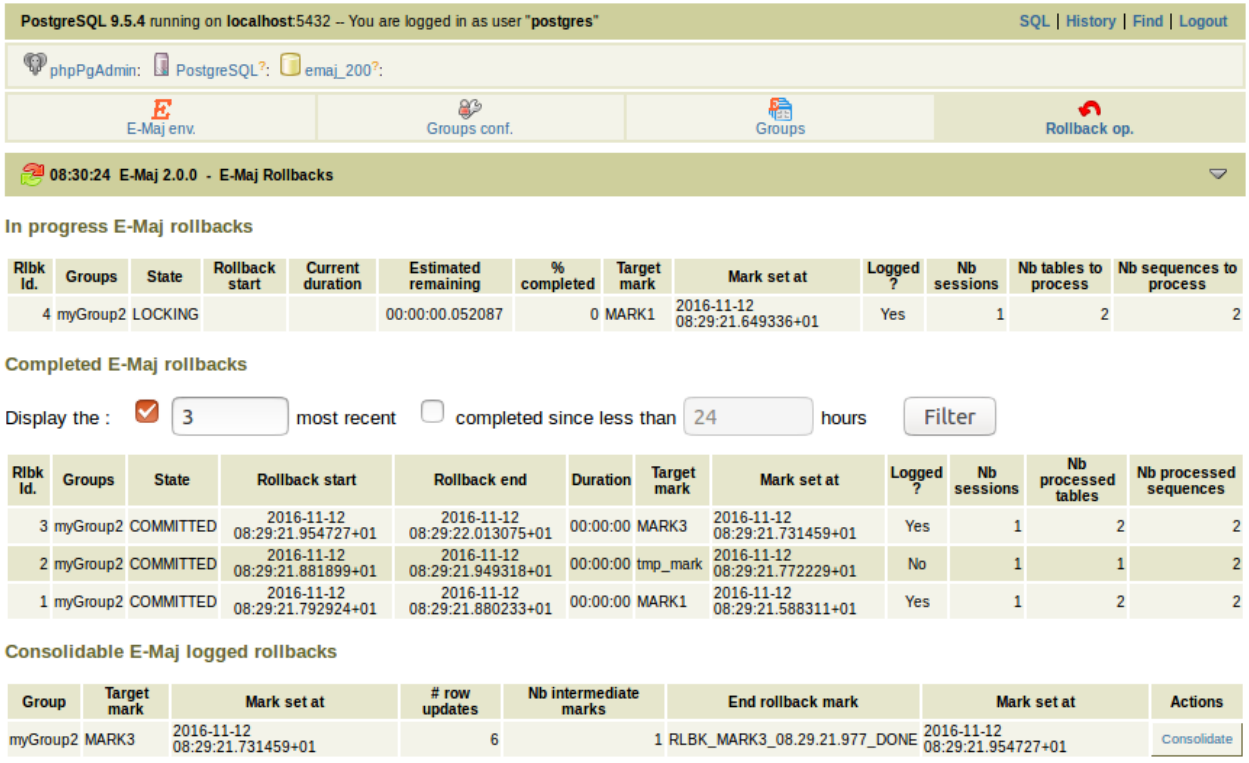


Fig. 31.13 – Figure 12 – Suivi des opérations de rollback.

CHAPITRE 32

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* (marque *(V)* derrière le nom de la fonction).

Fonctions	Paramètres en entrée	Données restituées
emaj_alter_group	groupe TEXT	nb.tables.et.seq INT
emaj_alter_groups	tableau.groupes TEXT[]	nb.tables.et.seq INT
emaj_cleanup_rollback_state		nb.rollback INT
emaj_comment_group	groupe TEXT, commentaire TEXT	
emaj_comment_mark_group	groupe TEXT, marque TEXT, commentaire TEXT	
emaj_consolidate_rollback_group	groupe TEXT, marque.fin.rollback TEXT	nb.tables.et.seq INT
emaj_create_group	groupe TEXT, [est.rollbackable BOOLEAN] [est.vide BOOLEAN]	nb.tables.et.seq INT

Suite sur la page suivante

Tableau 32.1 – Suite de la page précédente

Fonctions	Paramètres en entrée	Données restituées
emaj_delete_before_mark_group	groupe TEXT, marque TEXT	nb.marques.supprimées INT
emaj_delete_mark_group	groupe TEXT, marque TEXT	1 INT
emaj_detailed_log_stat_group (V)	groupe TEXT, marque.début TEXT, marque.fin TEXT	SETOF emaj_detailed_log_stat_type
emaj_disable_protection_by_event_triggers		nb.triggers INT
emaj_drop_group	groupe TEXT	nb.tables.et.seq INT
emaj_enable_protection_by_event_triggers		nb.triggers INT
emaj_estimate_rollback_group (V)	groupe TEXT, marque TEXT	durée INTERVAL
emaj_estimate_rollback_groups (V)	tableau.groupes TEXT[], marque TEXT	durée INTERVAL
emaj_force_drop_group	groupe TEXT	nb.tables.et.seq INT
emaj_force_stop_group	groupe TEXT	nb.tables.et.seq INT
emaj_gen_sql_group	groupe TEXT, marque.début TEXT, marque.fin TEXT, fichier.sortie TEXT, [tableau.tables.seq TEXT[]]	nb.req.générées BIGINT
emaj_gen_sql_groups	tableau.groupes TEXT[], marque.début TEXT, marque.fin TEXT, fichier.sortie TEXT, [tableau.tables.seq TEXT[]]	nb.req.générées BIGINT
emaj_get_consolidable_rollbacks (V)		SETOF emaj_consolidable_rollback_type

Suite sur la page suivante

Tableau 32.1 – Suite de la page précédente

Fonctions	Paramètres en entrée	Données restituées
emaj_get_previous_mark_group (V)	groupe TEXT, date.heure TIMESTAMPTZ	marque TEXT
emaj_get_previous_mark_group (V)	groupe TEXT, marque TEXT	marque TEXT
emaj_log_stat_group (V)	groupe TEXT, marque.début TEXT, marque.fin TEXT	SETOF emaj_log_stat_type
emaj_logged_rollback_group	groupe TEXT, marque TEXT, est_modif_groupe_autorisé BOOLEAN	SETOF (sévérité TEXT, message TEXT)
emaj_logged_rollback_groups	tableau.groupes TEXT[], marque TEXT, est_modif_groupe_autorisé BOOLEAN	SETOF (sévérité TEXT, message TEXT)
emaj_protect_group	groupe TEXT	0/1 INT
emaj_protect_mark_group	groupe TEXT, marque TEXT	0/1 INT
emaj_rename_mark_group	groupe TEXT, marque TEXT, nouveau.nom TEXT	
emaj_reset_group	groupe TEXT	nb.tables.et.seq INT
emaj_rollback_activity (V)		SETOF emaj_rollback_activity_type
emaj_rollback_group	groupe TEXT, marque TEXT, est_modif_groupe_autorisé BOOLEAN	SETOF (sévérité TEXT, message TEXT)

Suite sur la page suivante

Tableau 32.1 – Suite de la page précédente

Fonctions	Paramètres en entrée	Données restituées
emaj_rollback_groups	tableau.groupes TEXT[], marque TEXT, est_modif_groupe_autorisé BOOLEAN	SETOF (sévérité TEXT, message TEXT)
emaj_set_mark_group	groupe TEXT, [marque TEXT]	nb.tables.et.seq INT
emaj_set_mark_groups	tableau.groupes TEXT[], [marque TEXT]	nb.tables.et.seq INT
emaj_snap_group	groupe TEXT, répertoire TEXT, options.copy TEXT	nb.tables.et.seq INT
emaj_snap_log_group	groupe TEXT, marque.début TEXT, marque.fin TEXT, répertoire TEXT, options.copy TEXT	nb.tables.et.seq INT
emaj_start_group	groupe TEXT, [marque TEXT], [reset.log BOOLEAN]	nb.tables.et.seq INT
emaj_start_groups	tableau.groupes TEXT[], [marque TEXT], [reset.log BOOLEAN]	nb.tables.et.seq INT
emaj_stop_group	groupe TEXT, [marque TEXT]	nb.tables.et.seq INT
emaj_stop_groups	tableau.groupes TEXT[], [marque TEXT]	nb.tables.et.seq INT
emaj_unprotect_group	groupe TEXT	0/1 INT

Suite sur la page suivante

Tableau 32.1 – Suite de la page précédente

Fonctions	Paramètres en entrée	Données restituées
emaj_unprotect_mark_group	groupe TEXT, marque TEXT	0/1 INT
emaj_verify_all (V)		Setof TEXT

CHAPITRE 33

Index et tables

— genindex
— modindex
— search