E-Maj 2.0.1
-
une extension PostgreSQL

Enregistrement des Mises A Jour

Composants

- E-Maj
  - Extension PostgreSQL
  - Open Source (licence GPL)
  - Disponible sur
    - pgxn.org
    - github (https://github.com/beaud76/emaj)
- Plug-in pour phpPgAdmin 5.1+
  - Disponible sur github (https://github.com/beaud76/emaj_ppa_plugin)
- Source de la documentation disponible sur github (https://github.com/beaud76/emaj_doc)

Objectifs d'E-Maj

- Enregistrer les mises à jours sur des tables applicatives pour pouvoir :
  - les consulter (audit)
  - les annuler si nécessaire
- Utilisable avec :
  - des applications en test ou en production
  - des bases de données de toute taille

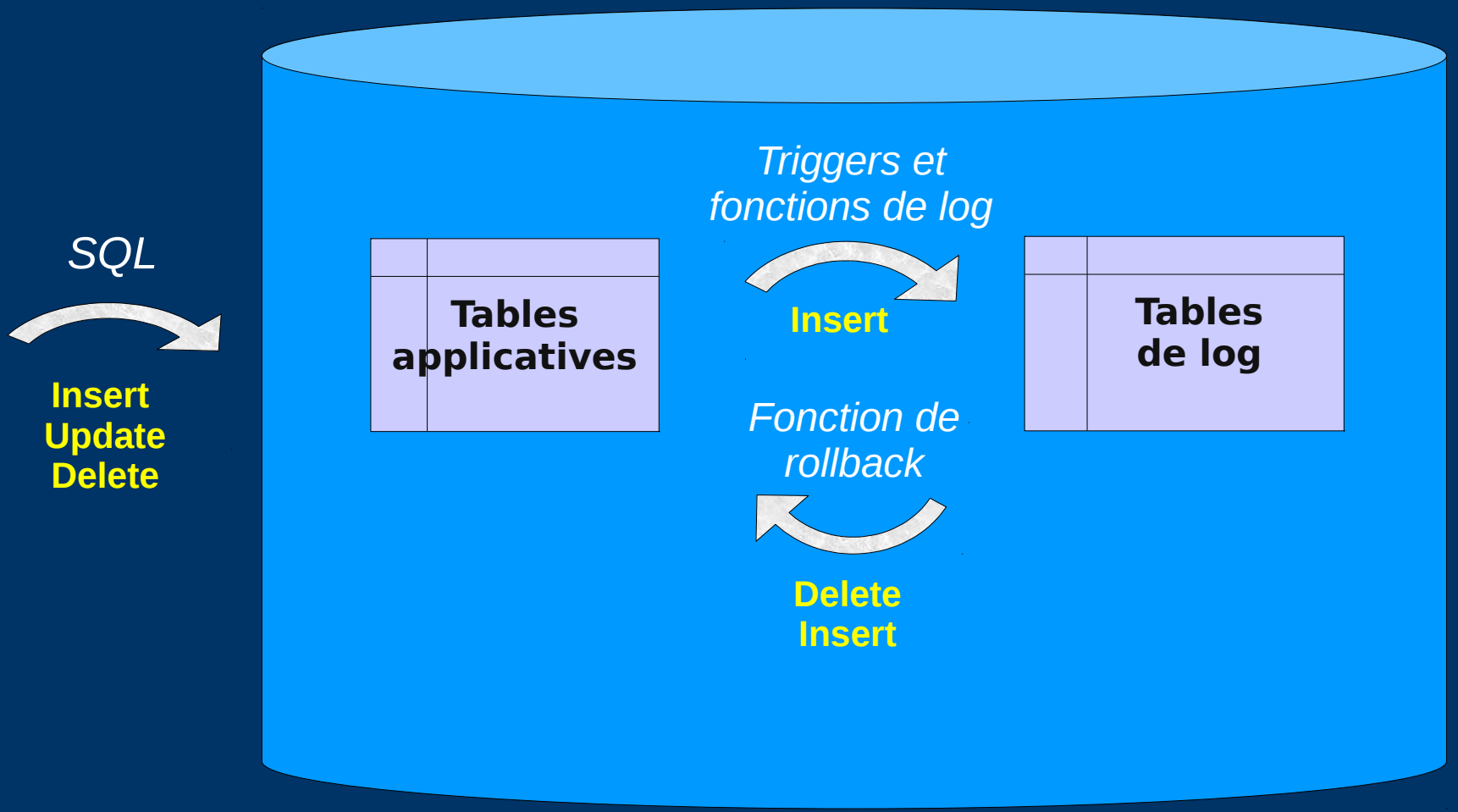
E-Maj : caractéristiques requises

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

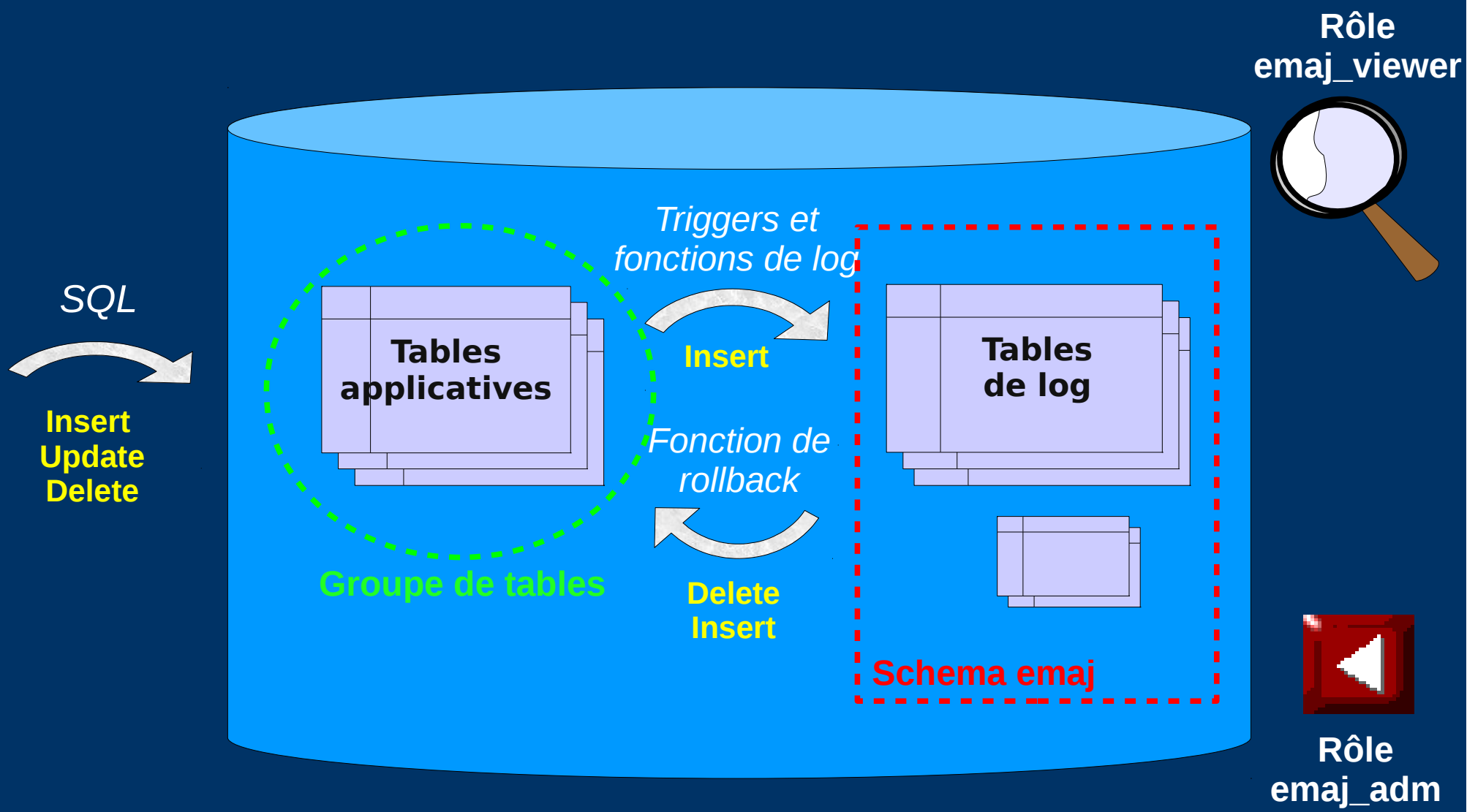
E-Maj : Concepts

- Groupe de tables = ensemble de tables et/ou séquences d'une base de données, appartenant à un ou plusieurs schémas, et ayant le même rythme de vie ; c'est le principal objet manipulé par l'utilisateur
- Marque = point stable dans la vie d'un « groupe de tables », et dont on peut retrouver l'état ; il est identifié par un nom
- Rollback = positionnement d'un « groupe de tables » à l'état dans lequel il se trouvait lors de la prise d'une « marque »

Le log des mises à jour



E-Maj : Principe général



E-Maj : Installation

- Télécharger et installer l'extension dans le répertoire *share/postgres/extension* du logiciel PostgreSQL
- Copier et adapter le fichier sql/emaj.control directement dans le répertoire *share/postgres/extension*
- Se connecter à la base ciblée en tant que super-utilisateur et exécuter les requêtes
  - CREATE EXTENSION IF NOT EXISTS DBLINK; (recommandé)
  - CREATE EXTENSION EMAJ;
- L'installation ajoute à la database :
  - 1 schema 'emaj' avec environ 90 fonctions, 12 tables techniques, 7 types, 1 vue et 1 séquence
  - 2 event triggers
  - 2 rôles

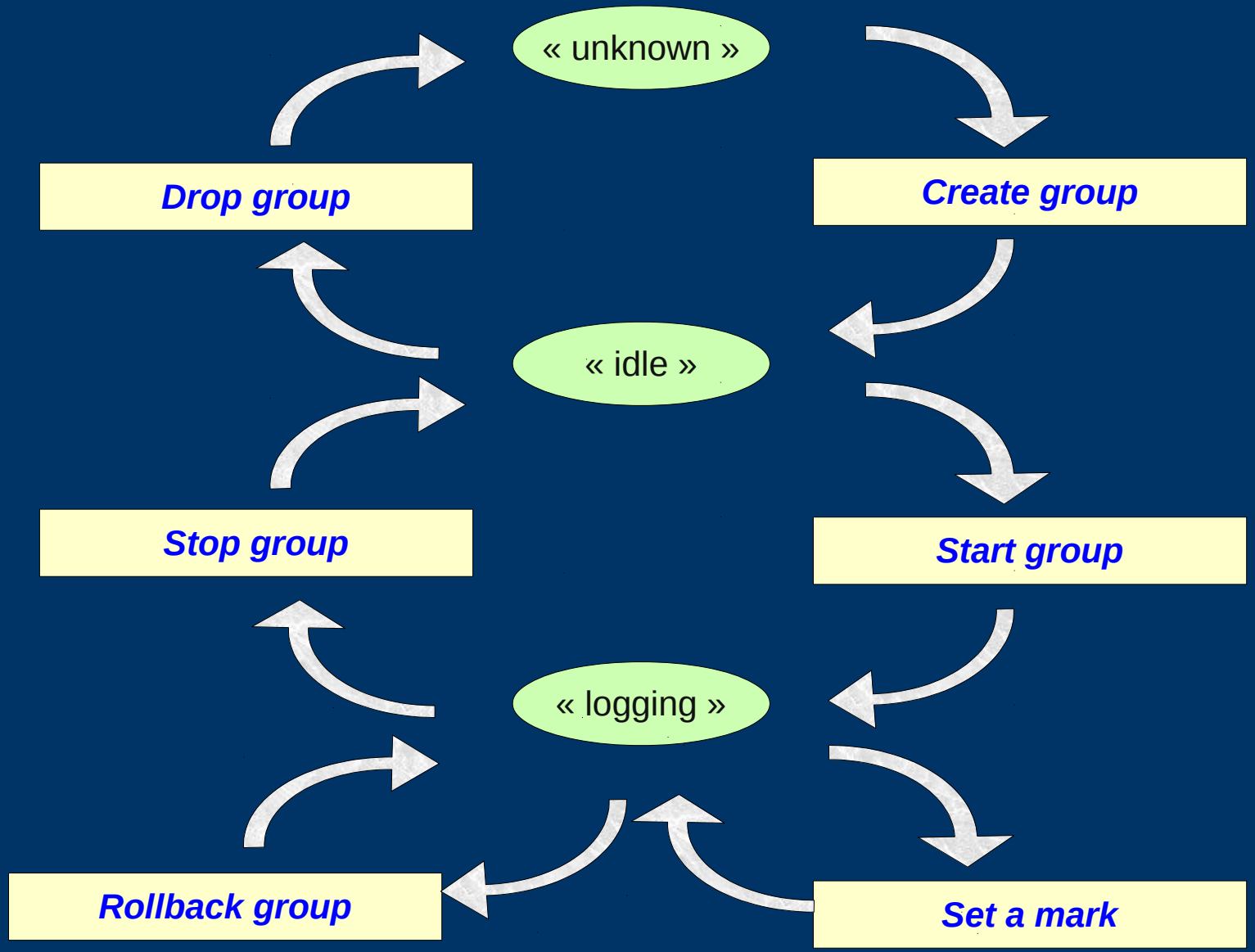
E-Maj : Initialisation

- 1) Alimentation de la table emaj_group_def pour définir le contenu des groupes de tables
- 2) Pour chaque groupe :
  - SELECT emaj_create_group (groupe, est_rollbackable);
    => crée pour chaque table applicative :
    - 1 table de log + 1 séquence dans un schéma emaj
    - 1 trigger + 1 fonction pour tracer les mises à jour
  - SELECT emaj_drop_group (groupe)
    … supprime un groupe créé auparavant

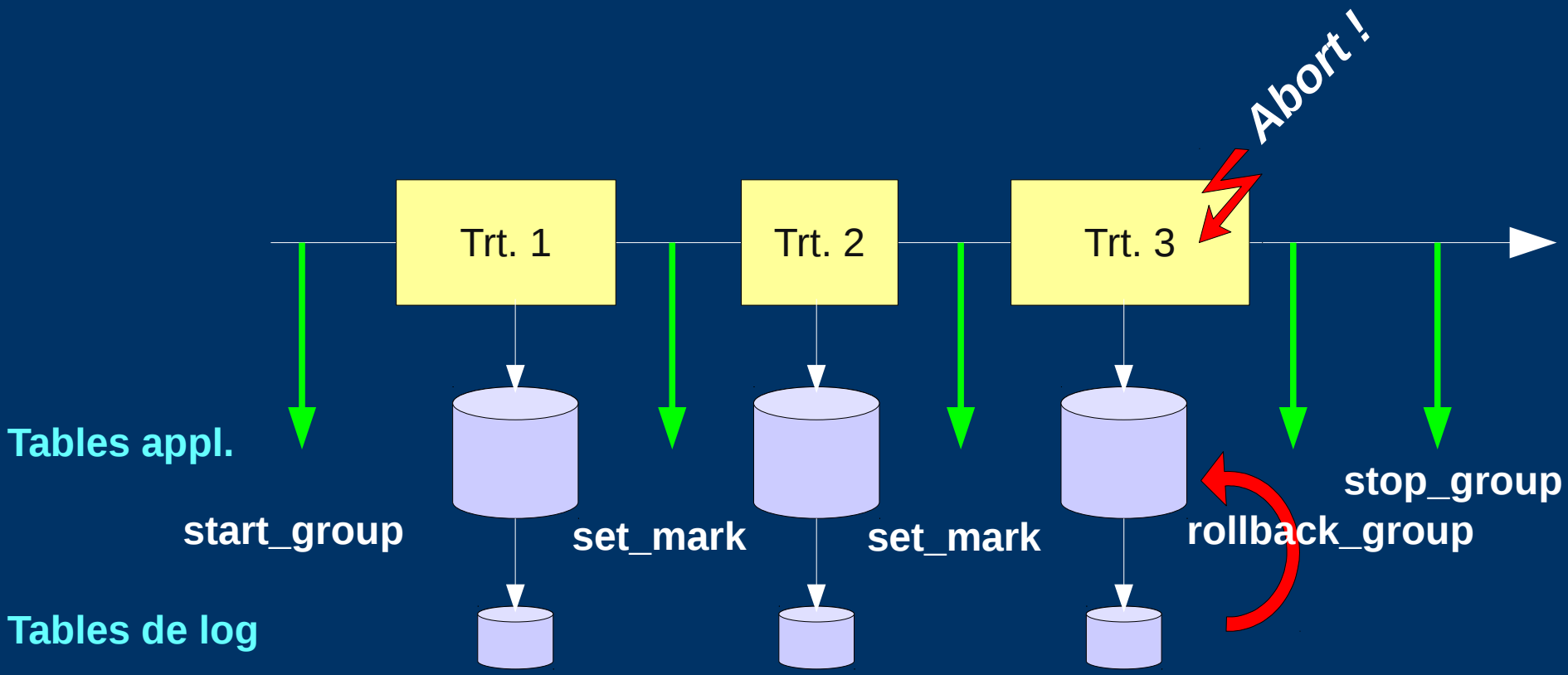
E-Maj : Fonctions principales

- emaj_start_group (groupe, marque)
  - Active les triggers de log et pose une marque initiale
- emaj_set_mark_group (groupe, marque)
  - Pose une marque intermédiaire
- emaj_rollback_group (groupe, marque)
  - Rollback les tables et séquences d'un groupe dans l'état correspondant à la marque
- emaj_logged_rollback_group (groupe, marque)
  - Idem emaj_rollback_group, mais le rollback peut être annulé ultérieurement (rollback rollbackable !)
- emaj_stop_group (groupe [,marque])
  - Désactive les triggers de log => rollback plus possible

Le cycle de vie d'un groupe de tables



Enchaînement E-Maj typique ...



Les tables de log

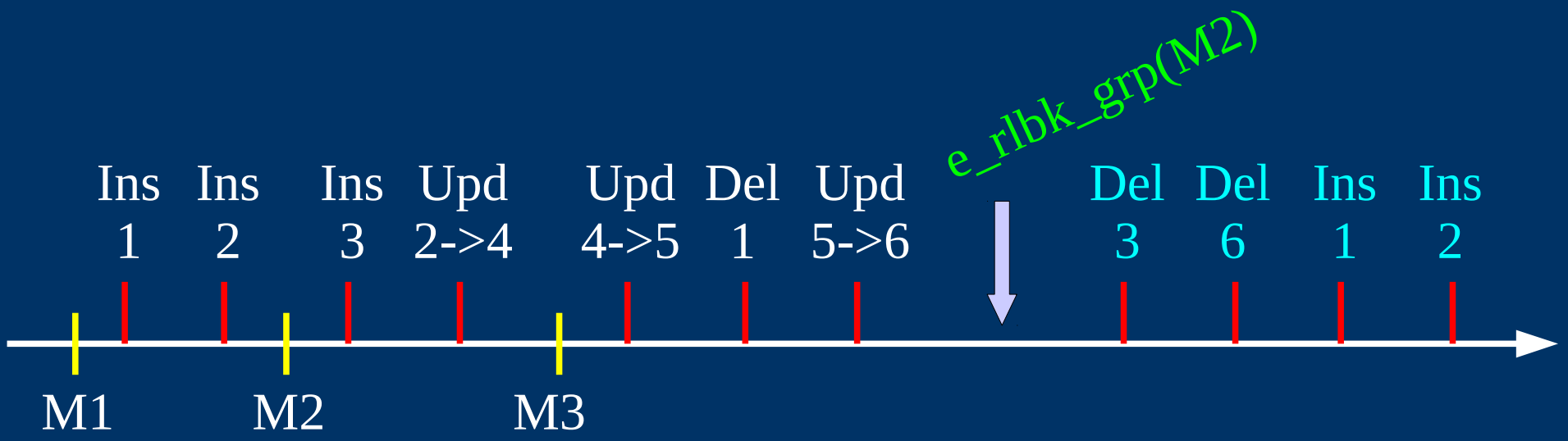
- L'examen des tables de log peut grandement aider le debuging des applications
- Une table de log contient
  - Les mêmes colonnes que la table applicative associée
  - Et quelques colonnes techniques
- Une ligne mise à jour dans une table applicative génère
  - 1 ligne de log pour un INSERT (nouvelle ligne)
  - 1 ligne de log pour un DELETE (ancienne ligne)
  - 2 lignes de log pour un UPDATE (ancienne et nouvelle lignes)
- Un TRUNCATE génère une ligne de log

Les colonnes techniques des tables de log

- 8 colonnes techniques en fin de chaque ligne de log
  - emaj_verb : type de mise à jour - INS/UPD/DEL/TRU
  - emaj_tuple : type de ligne - OLD/NEW
  - emaj_gid : numéro de séquence interne
  - emaj_changed : heure de la mise à jour - clock_timestamp()
  - emaj_txid : numéro de la transaction - txid_current()
  - emaj_user : rôle de connexion du client - session_user
  - emaj_user_ip : adresse ip du client - inet_client_addr()
  - emaj_user_port : port ip du client - inet_client_port()

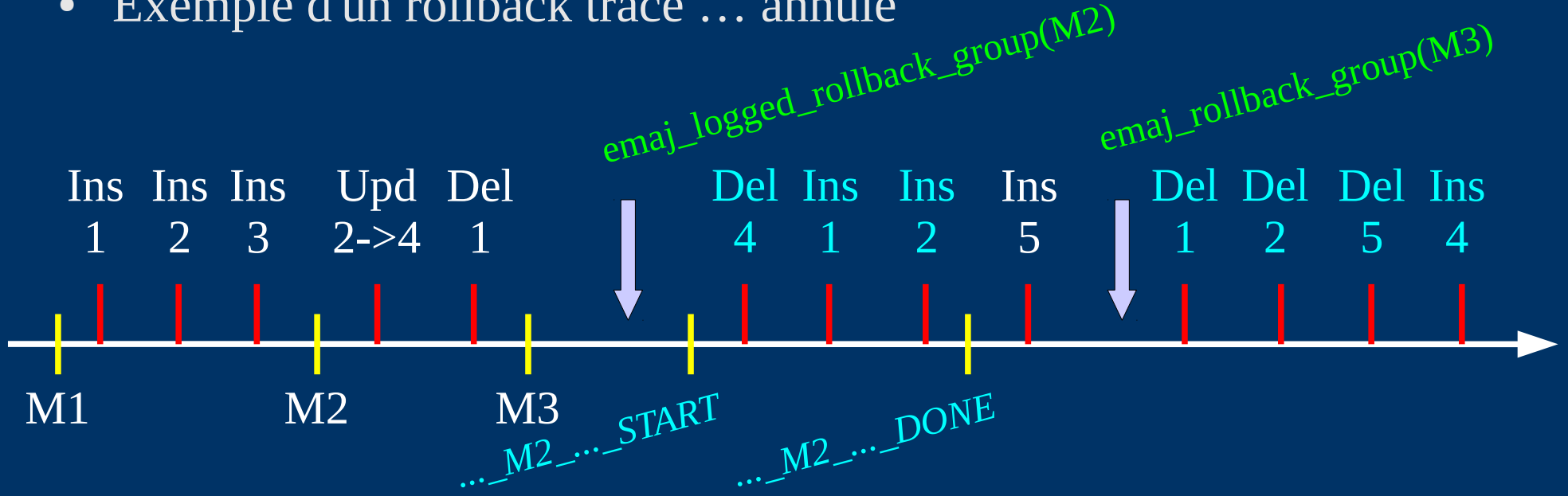
Le « Rollback simple »

- Les triggers de log sont désactivés
- Chaque table est remise à niveau par un algorithme optimisé
  - Ne traite qu'une seule fois chaque valeur de clé primaire
  - Prend en compte les éventuelles clés étrangères
- Les logs et les marques annulés sont supprimés



Le « Rollback tracé » (1/2)

- Les triggers de log ne sont pas désactivés
- Les logs et marques annulées sont conservées
- Pose automatique d'une marque avant et après le rollback
  - RLBK_<marque cible>_<HH.MI.SS.MS>_START
  - RLBK_<marque cible>_<HH.MI.SS.MS>_DONE
- Exemple d'un rollback tracé … annulé



Le « Rollback tracé » (2/2)

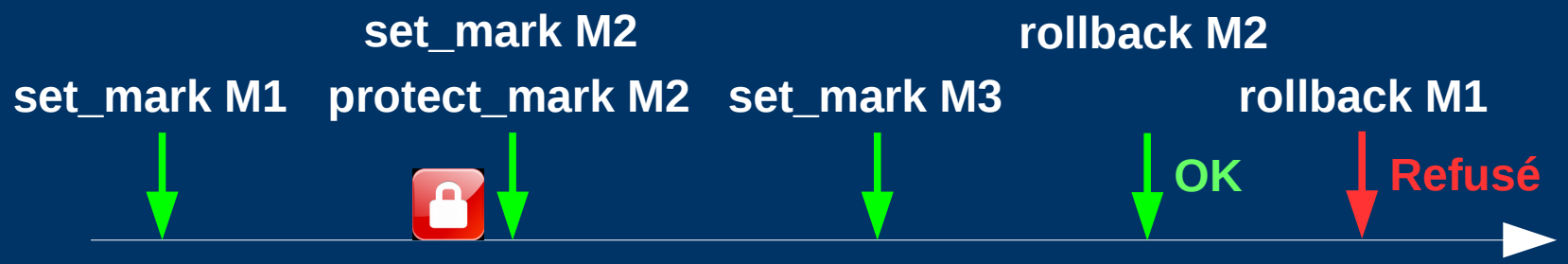
- Idéal pour les tests : évite d'avoir de nombreuses sauvegardes intermédiaires pour rejouer d'anciens tests
- Pendant le rollback les tables restent accessibles en lecture
- Un « rollback tracé » peut être transformé ultérieurement en « rollback simple » => « consolidation de rollback »
  - Les logs et marques intermédiaires sont supprimées, permettant de récupérer de la place dans les logs
  - emaj_consolidate_rollback_group(groupe, marque_fin_de_rollback)
  - Les tables peuvent être mises à jour des tables en parallèle

Suivre les rollbacks en exécution

- Nécessite dblink, et la valorisation du paramètre « dblink_user_password » dans la table emaj_param
- SELECT * FROM emaj.emaj_rollback_activity();
- Restitue
  - les caractéristiques du rollback (groupe, marque...)
  - état du rollback
  - durée écoulée
  - estimation de la durée restante et du % réalisé

Se protéger contre des rollbacks accidentels

- 2 fonctions pour gérer la protection d'un groupe de tables
  - emaj_protect_group (groupe)
  - emaj_unprotect_group (groupe)
- 2 fonctions pour gérer la protection d'une marque
  - emaj_protect_mark_group (groupe, marque)
  - emaj_unprotect_mark_group (groupe)



Utilisations possibles d'E-Maj

- Aide l'organisation des tests applicatifs en fournissant un moyen rapide d'annuler les mises à jour issues d'une exécution de programmes et de pouvoir ainsi répéter facilement des tests
- En production, fournit un moyen d'annuler des traitements sans devoir sauver et restaurer le cluster par pg_dump/restore ou copie physique
  - D'autant plus intéressant que les tables sont volumineuses et les mises à jour peu nombreuses

Stratégies d'utilisation des marques (1/2)

- « mono-marque », pour minimiser la place disque
  - répéter
    - start_group (groupe, marque)
    - traitement #i
    - stop_group (groupe)
- « multi-marques », pour rollbacks plus souples
  - start_group (groupe, marque1)
  - répéter
    - traitement #i
    - emaj_set_mark (groupe, marque #i+1)
  - stop_group (groupe)

Stratégies d'utilisation des marques (2/2)

- Logging permanent et suppression régulière des marques les plus anciennes (« rolling log »)
  - répéter
    - traitement #i
    - emaj_set_mark (groupe, marque #i+1)
    - emaj_delete_before_mark (groupe, marque #j)

(attention, la suppression des marques peut être coûteuse si le volume de log à effacer est important)

Fonctions multi-groupes

- Pour traiter plusieurs groupes dans une même transaction
  - emaj_start_groups (tableau de groupes, marque)
  - emaj_stop_groups (tableau de groupes)
  - emaj_set_mark_groups (tableau de groupes, marque)
  - emaj_rollback_groups (tableau de groupes, marque)
  - emaj_logged_rollback_groups (tableau de groupes, marque)
- 2 syntaxes pour un tableau de groupes
  - ARRAY['groupe 1','groupe 2',...]
  - '{"groupe 1", "groupe 2",…}'

Fonctions de gestion des marques

- emaj_comment_mark_group (groupe, marque)
  - Ajoute, modifie ou supprime un commentaire sur un groupe
- emaj_rename_mark_group (groupe, old mark, new mark)
  - Renomme une marque
- emaj_delete_mark_group (groupe, marque)
  - Supprime une marque
- emaj_delete_before_mark_group (groupe, marque)
  - Supprime toutes les marques antérieures à une marque donnée

Autres fonctions de gestion des groupes

- emaj_comment_group (groupe, commentaire)
  - Ajoute, modifie ou supprime un commentaire sur un groupe
- emaj_reset_group (groupe)
  - Purge les tables de log avant le prochain démarrage
- emaj_force_stop_group (group)
  - Force l'arrêt d'un groupe

Autres fonctions de gestion des rollbacks

- emaj_estimate_rollback_group (groupe, marque)
  - Estime la durée nécessaire pour rollbacker un groupe à une marque
- emaj_consolidate_rollback_group (groupe, marque)
  - Consolide un rollback tracé identifié par le groupe de tables et la marque de fin de rollback générée. Transforme un rollback tracé en rollback non tracé en supprimant toutes les marques et log entre la marque cible du rollback et la marque de fin de rollback
- emaj_get_consolidable_rollbacks ()
  - Liste les rollbacks consolidables

Fonctions statistiques

- emaj_log_stat_group (groupe, marque_début, marque_fin)
  - Retourne rapidement des statistiques sur le nombre de lignes présentes dans chaque table de log, entre 2 marques ou entre 1 marque et la situation courante
- emaj_detailed_log_stat_group (groupe, marque_début, marque_fin)
  - Retourne des statistiques sur le contenu des tables de logs, entre 2 marques
  - Par table, par type de requête (INSERT / UPDATE / DELETE) et par ROLE à l'origine des mises à jour

Fonctions d'export

- emaj_snap_group (groupe, directory, options_copy)
  - Vide toutes les tables et séquences d'un groupe sur des fichiers dans une directory
- emaj_snap_log_group (groupe, marque_début, marque_fin, directory, options_copy)
  - Vide une partie des tables de log et des séquences d'un groupe sur des fichiers dans une directory
- emaj_gen_sql_group (groupe, marque_début, marque_fin, fichier [,liste_tables/seq])
  - Génère un script sql rejouant les mises à jour enregistrées entre 2 marques pour toutes ou partie des tables et séquences d'un groupe

Autres fonctions

- emaj_verify_all ( )
  - Vérifie la consistence d'une installation E-Maj
- emaj_find_previous_mark_group (groupe, date-heure) ou emaj_find_previous_mark_group (groupe, marque)
  - Retourne le nom de la marque qui précède immédiatement la date et heure donnée ou une autre marque

Pour les grosses bases de données...

- Possibilité de stocker les tables de log et leur index dans des tablespaces
  - Tablespace tspemaj utilisé par défaut s'il existe
  - Pour utiliser d'autres tablespaces :
    - Les créer
    - Configurer leur utilisation dans la table emaj_group_def
- Possibilité de mettre les objets de log dans des schémas emaj secondaires dédiés
  - Configurable pour chaque table dans emaj_group_def
  - Schémas créés et supprimés par E-Maj

Client pour des rollbacks parallélisés

- Un module php effectue des rollbacks en parallèle
- Client de la base de données
- Répartir automatiquement des tables du(des) groupe(s) à rollbacker dans un nombre donné de sessions
- Toutes les sessions appartiennent à une transaction (2PC) (➔ max_prepared_transaction >= #sessions)
- emajParallelRollback.php -d <database> -h <host> -p <port> -U <user> -W <password> -g <nom_groupe ou listes_groupes> -m <marque> -s <nb_sessions> [-l]
- Autres options : --help, -v, --version
- Nécessite php avec l'extension PostgreSQL

Client pour suivre les rollbacks

- Un autre module php pour suivre les rollbacks en cours ou récemment terminés
- emajRollbackMonitor.php -d <database> -h <host> -p <port> -U <user> -W <password> -n <nb_itérations> -i <rafraichissement_en_secondes> -l <nb_rollbacks_terminés> -a <historique_rollbacks_terminés_en_heures>
- Autres options : --help, -v, --version

```
 E-Maj (version 1.1.0) - Monitoring rollbacks activity
------------------------------------------------------------

04/07/2013 - 12:07:17
** 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
```

Fiabilité (1/2)

- Nombreux contrôles, en particulier aux start_group set_mark_group et rollback_group :
  - Existence de toutes les tables, séquences, fonctions et triggers ?
  - Cohérence des colonnes entre les tables applicatives et les tables de log (existence, type) ?
- Verrous forts sur les tables lors des start_group, set_mark_group et rollback_group, pour être sûr qu'aucune transaction n'est en train de mettre à jour les tables applicatives
- Rollback de toutes les tables et séquences dans une seule transaction

Fiabilité (2/2)

- Les requêtes TRUNCATEs sont bloquées pour les groupes « rollbackables » actifs
- Pour les versions de PostgreSQL les plus récentes (9.3+), des « event triggers » bloquent la suppression intempestive ou certaines modifications de composants (tables, séquences, fonctions...)
  - 2 fonctions pour désactiver/ré-activer le blocage
    - emaj_disable_protection_by_event_triggers()
    - emaj_enable_protection_by_event_triggers()

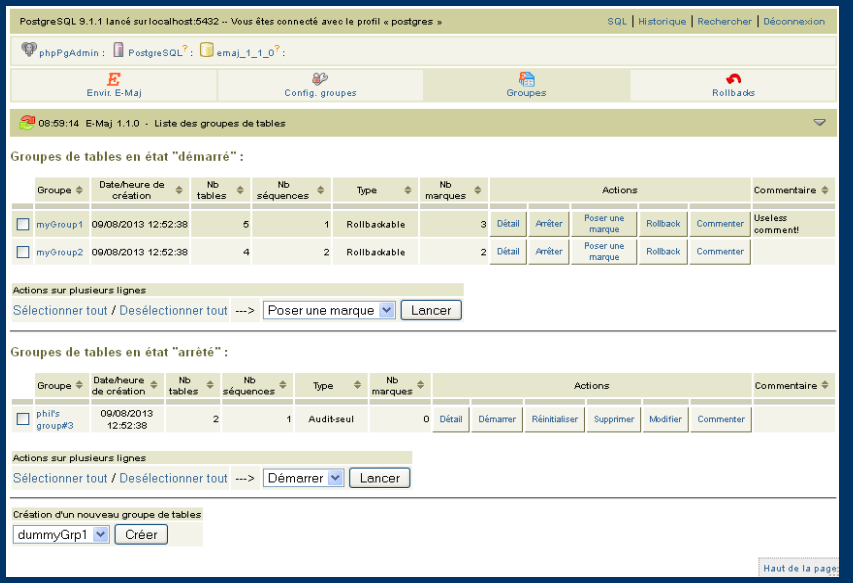
Sécurité

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

Performances

- Surcoût du log
  - Dépend largement du matériel et du ratio lecture/écriture SQL des traitements
  - Typiquement quelques % sur les temps elapse
- Durée de rollback
  - Dépend largement du matériel, de la structure des tables (taille des lignes, index, contraintes...)

Plug-in PhpPgAdmin



- Complètement intégré dans phpPgAdmin 5.1+
- Aide les administrateurs et les « viewers »
- Montre tous les objets E-Maj (groupes, marques...) et leurs attributs
- Permet toutes les actions possibles sur les objets E-Maj
- Justifie à lui seul l'installation de phpPgAdmin 

Limitations actuelles

- Depuis E-Maj 2.0, la version PostgreSQL minimum requise est la 9.1
- Les tables applicatives appartenant à un groupe « rollbackable » doivent avoir une PRIMARY KEY
- Les requêtes de DDL ne peuvent pas être gérés par E-Maj

Pour conclure...

- Beaucoup plus d'information dans la documentation et dans les fichiers README et CHANGES
- Grand merci pour leur aide à :
  - Andreas Scherbaum, Jean-Paul Argudo et l'équipe Dalibo, les DBA de la CNAF, Ronan Dunklau, Don Levine
  - Tous ceux qui m'ont contacté pour m'adresser leur commentaires ou doléances...
- email : phb<dot>emaj<at>free<dot>fr
  N'hésitez pas