

E-Maj 1.0.1

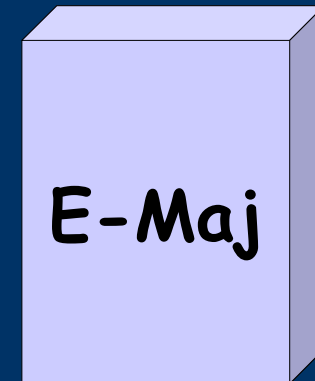
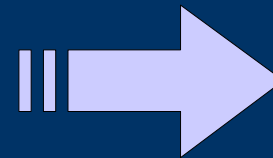
-

une extension PostgreSQL



De l'idée de restauration logique à ... E-Maj

- L'idée d'origine = contrib table_log d'Andreas Scherbaum
 - 1 trigger par table pour tracer toutes les mises à jour dans une table de log
 - 1 fonction pour annuler les mises à jour
- Développement de fonctions plpgsql étendant le concept, pour bâtir une solution plus « industrielle »



Acronyme de
« Enregistrement des Mises A Jour »

Composants

- E-Maj
 - Extension PostgreSQL
 - Open Source (licence GPL)
 - Disponible sur
 - pgFoundry.org
 - pgxn.org
 - [github](https://github.com)
- Plugin pour phpPgAdmin
 - Version avec phpPgAdmin 5.0.4 disponible sur demande



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

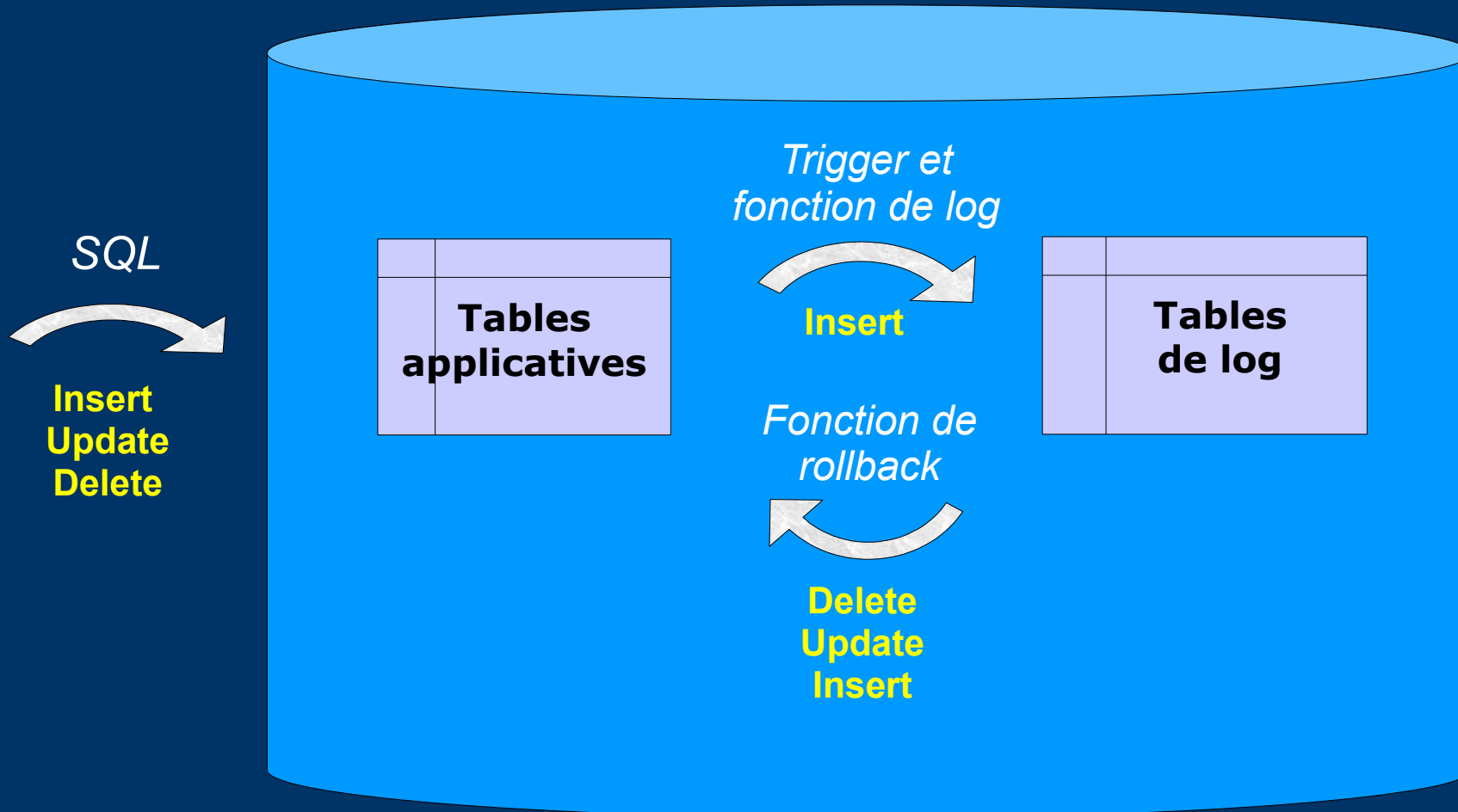
E-Maj : caractéristiques requises

- 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
-
-

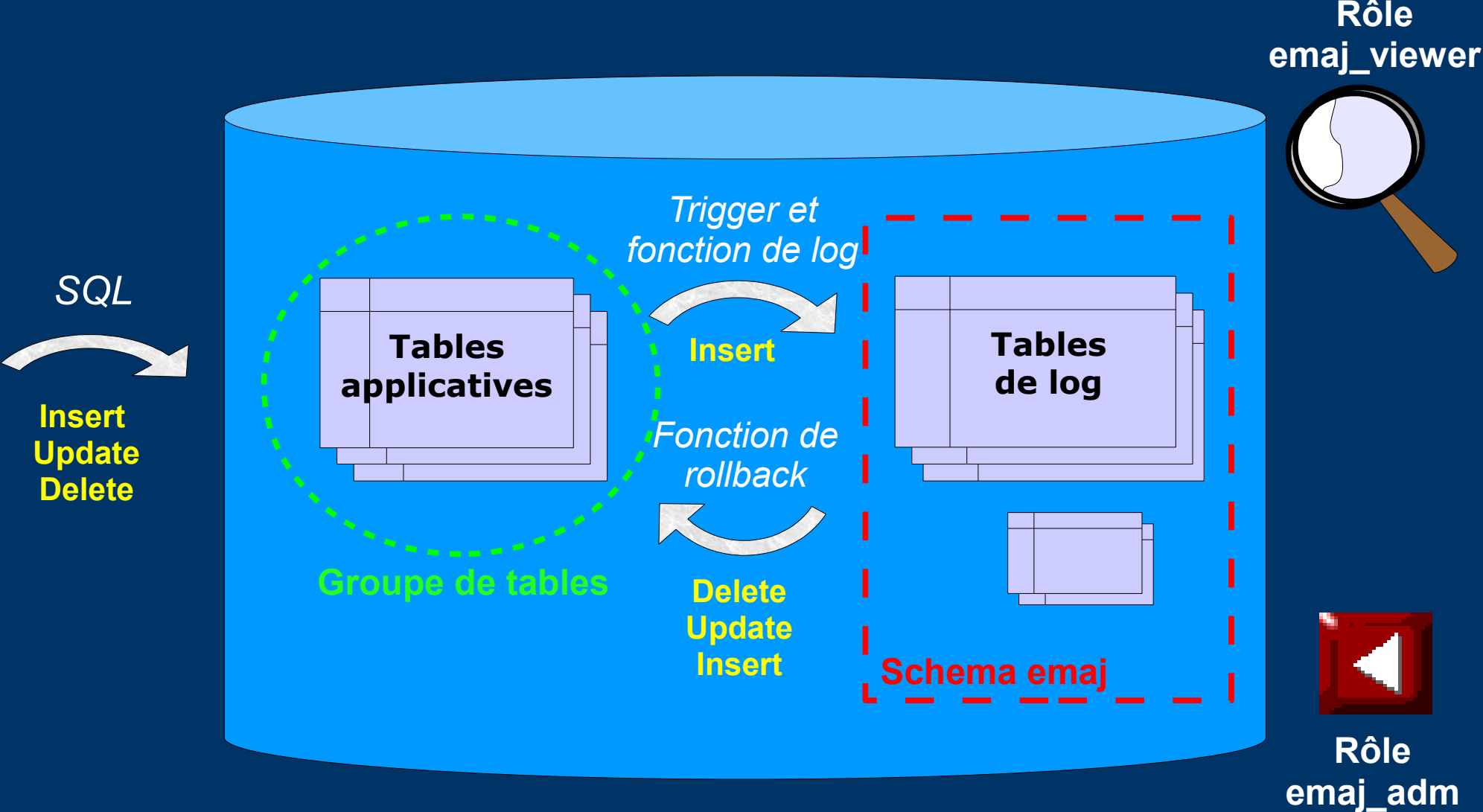
E-Maj : Concepts

- **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

- Opération préliminaire sur la database :
 - CREATE LANGUAGE plpgsql; (si pg < 9.0)
- Ensuite, en tant que super-utilisateur :
 - \i .../sql/emaj.sql
- L'installation ajoute à la database :
 - 1 schema 'emaj' avec 75 fonctions, 10 tables techniques et 3 types
 - 2 roles

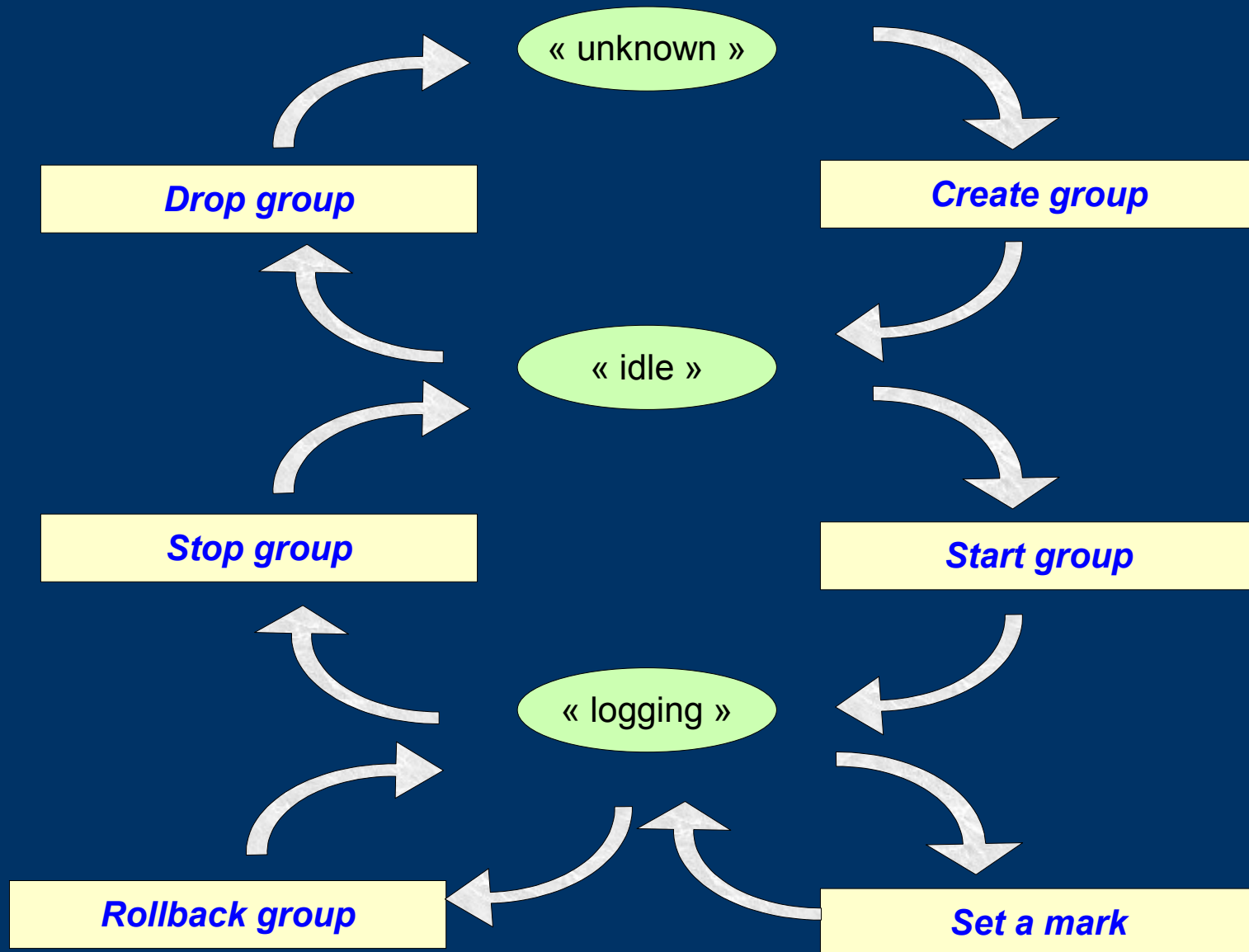
E-Maj : Initialisation

- 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 dans le schéma emaj et le tablespace tspemaj
 - 1 trigger + 1 fonction pour tracer les mises à jour
 - 1 fonction pour « rollbacker » les mises à jour sur la tables applicative (si groupe « rollbackable »)
 - SELECT emaj_drop_group (groupe)
... supprime un groupe créé auparavant
-
-

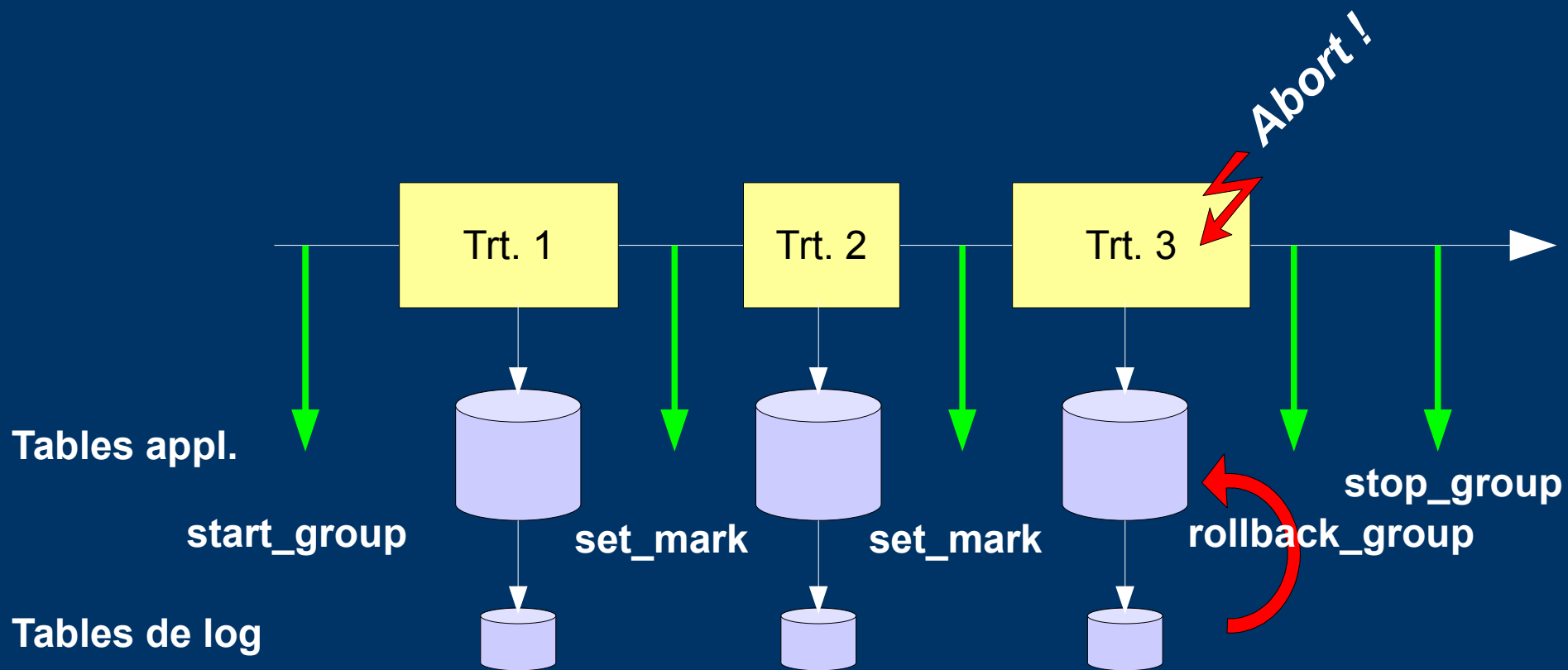
E-Maj : Fonctions principales

- **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

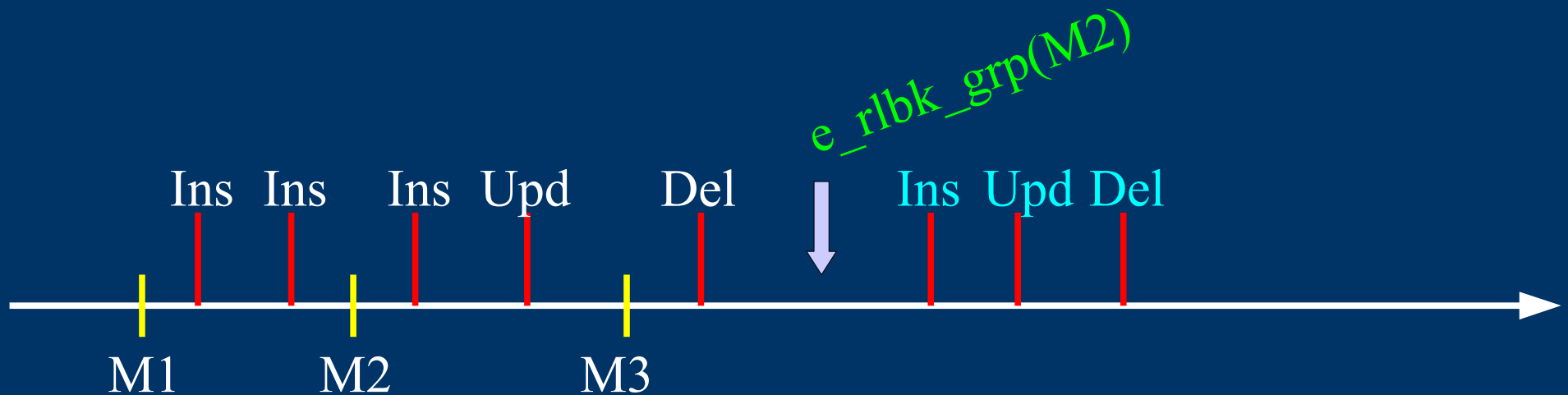


Enchaînement E-Maj typique ...



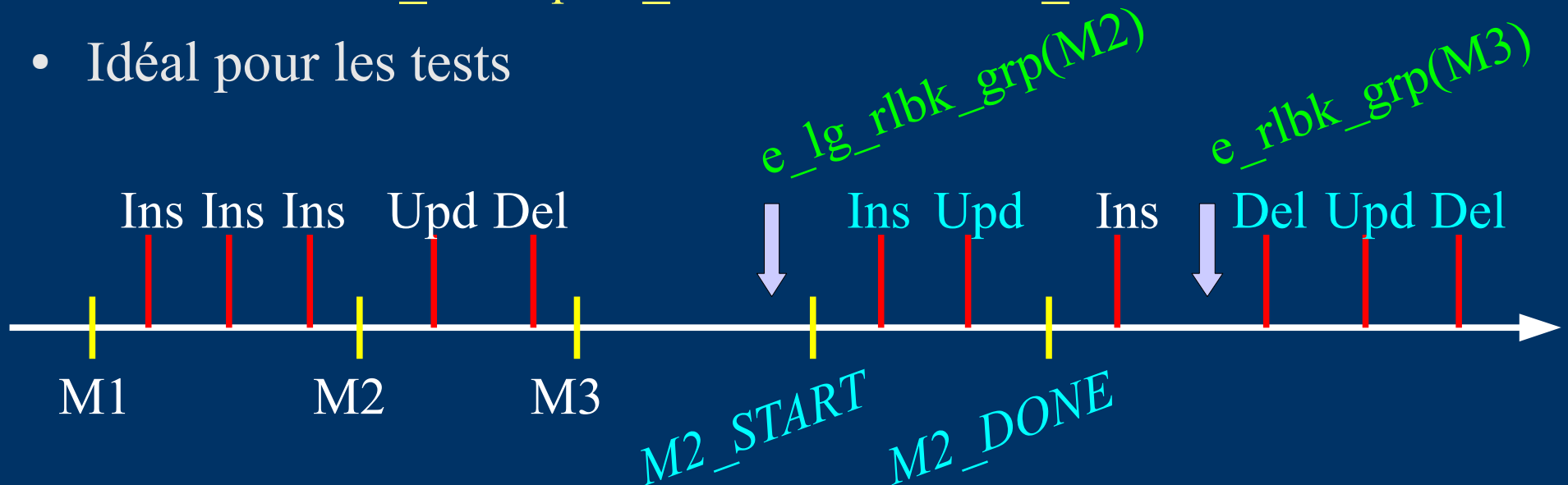
Le « *Rollback simple* »

- Les triggers de log sont désactivés
- Les INSERTs sont annulés par des DELETEDs, les DELETEDs par des INSERTS et les UPDATES par ... des UPDATES,
- Appliqués dans l'ordre inverse du log
- Les logs et les marques annulés sont supprimés



Le « *Rollback tracé* »

- Les triggers de log ne sont pas désactivés
- les logs et marques annulées sont conservées
- Pose automatique d'une marque avant et d'une marque après le rollback
 - `RLBK_<marque>_<HH.MI.SS.MS>_START`
 - `RLBK_<marque>_<HH.MI.SS.MS>_DONE`
- Idéal pour les tests



Utilisations possibles d'E-Maj

- Peut aider 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)

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

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



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_generate_sql** (groupe, marque_début, marque_fin, fichier)
 - Génère un script sql rejouant les mises à jour enregistrées entre 2 marques
-
-

Autres fonctions

- `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
- `emaj_verify_all` ()
 - Vérifie la consistance d'une installation E-Maj
- `emaj_estimate_rollback_duration` (groupe, marque)
 - Estime la durée nécessaire pour rollbacker un groupe à une 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 rollback parallélisé

- 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 <group_name or groups_list>
-m <mark> -s <#sessions> [-1]
 - Autres options : --help, -v, --version
 - Nécessite php avec l'extension PostgreSQL
-
-

Fiabilité

- 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
-
-

Sécurité

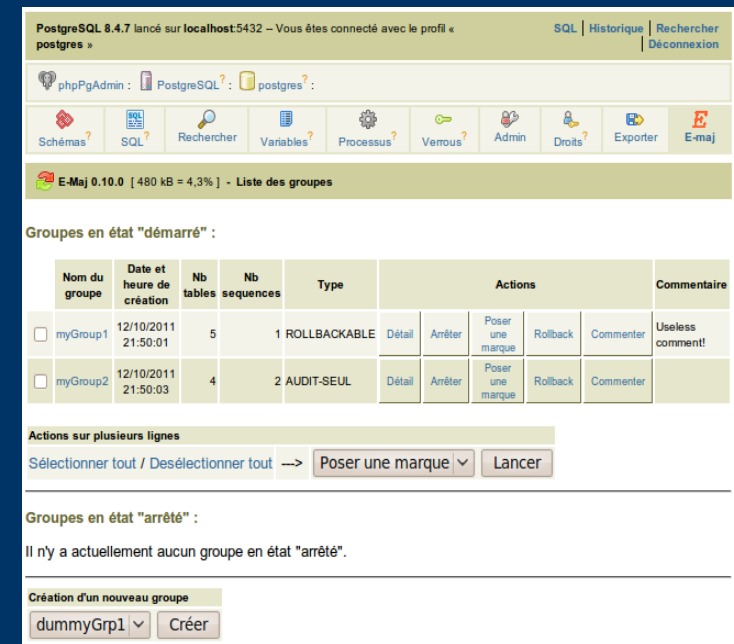
- 2 rôles dont les droits peuvent être donnés :
 - emaj_adm pour ... l'administration E-Maj
 - emaj_viewer pour uniquement voir les tables de log
 - Les objets E-Maj ne sont créés et manipulés que par un super-utilisateur ou un membre de emaj_adm
 - Aucun autre droit donné sur les schémas, les tables et les fonctions d'E-Maj
 - Triggers de log 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...)
 - Mesuré sur du matériel récent avec une application réelle : environ 10Gb de log en 1 heure
-
-

Plugin PhpPgAdmin

- Un plugin pour phpPgAdmin 5 est disponible pour aider l'administrateur ou les « viewers »
 - Montre tous les objets E-Maj et leurs attributs
 - Permet toutes les actions possibles sur les objets E-Maj
- Le demander, si vous voulez l'essayer...



PostgreSQL 8.4.7 lancé sur localhost:5432 – Vous êtes connecté avec le profil « postgres »

phpPgAdmin : PostgreSQL : postgres :

E-Maj 0.10.0 [480 kB = 4,3%] - Liste des groupes

Groupes en état "démarré" :

	Nom du groupe	Date et heure de création	Nb tables	Nb sequences	Type	Actions					Commentaire
<input type="checkbox"/>	myGroup1	12/10/2011 21:50:01	5	1	ROLLBACKABLE	Détail	Arrêter	Poser une marque	Rollback	Commenter	Useless comment!
<input type="checkbox"/>	myGroup2	12/10/2011 21:50:03	4	2	AUDIT-SEUL	Détail	Arrêter	Poser une marque	Rollback	Commenter	

Actions sur plusieurs lignes

Sélectionner tout / Désélectionner tout → Poser une marque ▼ Lancer

Groupes en état "arrêté" :

Il n'y a actuellement aucun groupe en état "arrêté".

Création d'un nouveau groupe

dummyGrp1 ▼ Créer

Limitations actuelles

- Version PostgreSQL minimum : 8.2
- Les tables applicatives appartenant à un groupe « rollbackable » doivent avoir une **PRIMARY KEY**
- Nom du schéma + nom de la table applicative ≤ 52 caractères
- **DDL** et **TRUNCATE** ne peuvent pas être gérés par E-Maj
 - Mais les TRUNCATEs sont bloqués avec les versions de PostgreSQL ≥ 8.4

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
 - Tous ceux qui m'ont contacté pour m'adresser leur commentaires ou doléances...
- email : [phb<dot>emaj<at>free<dot>fr](mailto:phb.emaj@free.fr)
N'hésitez pas