Direction Interministérielle des
Systèmes d'Information et de
Communication

Socle Interministériel
du Logiciel Libre

19 Mars 2015

Guide de transition à PostgreSQL : aide à la décision

Version 1.0

Historique des versions du document

Version	Date	Commentaire
0.1	18/12/2013	Document de travail
0.2	07/02/2014	Première séance de travail
0.3	19/03/2014	Deuxième séance de travail
0.4	01/07/2014	Préparation pour diffusion
0.5	04/08/2014	Consolidation des contributions
1.0	19/03/2015	Validation et derniers correctifs
1.0.1	26/06/2015	Précision sur la licence

Contributeurs

Bruce BARDOU - DGFiP
Barek BOUTGAYOUT - MASS
Amina CHITOUR - MENMESR
Alain DELIGNY - MEDDE
Alain MERLE - MEDDE
Anthony NOWOCIEN - MAEDI
Loïc PESSONNIER - MINDEF
Marie-Claude QUIDOZ - CNRS
Laurent RAYMONDEAU - MENESR

SOMMAIRE

Introduction

Ce guide a été réalisé dans le cadre de la mise en œuvre de la circulaire « Ayrault » du 19 septembre 2012 relative à l'usage du logiciel libre dans l'administration, qui a induit la constitution d'un Socle Interministériel du Logiciel Libre (SILL). Comparativement aux solutions commerciales, ce guide présente la mise en œuvre de PostgreSQL. Il a pour objectif de répondre aux interrogations des maîtres d'ouvrage et maîtres d'œuvre pour la mise en œuvre de PostgreSQL en remplacement d'une solution commerciale. Ce guide a pour objectif, sans entrer dans le détail de l'implémentation technique, de démontrer l'intérêt de PostgreSQL en décrivant les mécanismes d'intégration, de sécurité et de robustesse.

1 - Satisfaire les besoins des maîtres d'ouvrage

1.1 - Les types d'usage

PostgreSQL est un système de gestion de bases de données au standard SQL utilisé classiquement dans les applications transactionnelles pour assurer la persistance des données comme tout produit commercial du même type (Oracle, DB2, Informix, MS SQL, Sybase,...).

PostgreSQL a déjà été mis en œuvre pour répondre aux besoins des maîtrises d'ouvrage.

PostgreSQL propose une extension pour la géomatique (PostGIS) conforme aux standards de l'Open Geospatial Consortium (OGC).

PostgreSQL est également utilisable dans le domaine de l'informatique décisionnelle comme entrepôt de données en liaison avec les outils de reporting (BusinessObjects, Pentaho,...).

De nombreux progiciels libres s'appuient nativement sur PostgreSQL (Gestion Électronique de Documents (GED), Moteurs de règles, GroupWare, Supervision,...).

PostgreSQL supporte également les traitements d'arrière plan comme les traitements par lots ou différés (batch,...).

1.2 - Accès aux données

PostgreSQL est conforme à la norme SQL 2011, le langage de requête commun à de nombreux SGBD. La migration est ainsi facilitée entre SGBDs respectant les standards.

1.3 - Les besoins de sécurité

PostgreSQL répond aux besoins de sécurité exprimé en terme de disponibilité, d'intégrité, de confidentialité et de traçabilité (DICT[1]).

Les besoins de cryptographie et d'authentification forte sont couverts par des modules complémentaires.

PostgreSQL dispose d'une communauté très réactive qui fournit les patchs correctifs de

1 Critères de sécurité DICT (Wikipédia)

sécurité et assure la gestion de l'obsolescence des composants. Une version majeure est supportée pendant 5 ans.

PostgreSQL garantit l'intégrité des données manipulées même en cas d'incident par ses propriétés d'atomicité, de cohérence, d'isolation et de durabilité (ACID[2]).

PostgreSQL offre nativement les mécanismes permettant de répondre aux besoins de confidentialité, de gestion des droits. Les propriétés ACID sont également garanties par des mécanismes assurant la gestion des transactions.

1.4 - Scénarios de migration des SI vers PostgreSQL

La migration d'un SI représente un coût non négligeable constitué principalement:

- de la migration des données, même s'il existe de nombreux outils permettant de réaliser cette migration en fonction des SGBD source ;
- des corrections à faire dans le code source de l'application. L'importance des adaptations est liée à l'usage des fonctionnalités spécifiques au produit existant (procédures stockées, triggers, fonctions non standards) et l'utilisation d'une couche d'abstraction (Hibernate) ;
- de la recette permettant de garantir à iso-fonctionnalités, qu'aucune erreur n'a été induite lors de la migration (tests de non régression).

De ce fait, les migrations vers PostgreSQL ne s'effectuent que lors d'une évolution fonctionnelle de l'application. Le coût de migration est intégré au projet dans son ensemble notamment en terme de recette technique et fonctionnelle. (Cas des MEDDE-MLET et DGFiP).

Cependant un plan ambitieux de migration globale des SI peut être programmé (cas du ministère des affaires sociales qui a entrepris une migration programmée de ses SI de Informix vers PostgreSQL).

2 - Intégration de PostgreSQL sur les plateformes techniques

2.1 - Plateformes techniques

2.1.1 - Architecture processeur

PostgreSQL fonctionne sur les architectures processeur suivantes[3] : x86, x86_64, IA64, PowerPC, PowerPC 64, S/390, S/390x, Sparc, Sparc 64, Alpha, ARM, MIPS, MIPSEL, M68K et PA-RISC.

2.1.2 - Système d'exploitation

PostgreSQL fonctionne sur la plupart des systèmes d'exploitation. Il existe des versions binaires pour la famille Red Hat (incluant CentOS/Fedora/Scientific), la famille Debian GNU/Linux et ses dérivées, la famille Ubuntu Linux et ses dérivées, SuSE, OpenSuSE,

2 Propriétés "ACID" (Wikipédia)

3 http://docs.postgresql.fr/9.1/supported-platforms.html

Solaris, Windows, MacOSX, FreeBSD, OpenBSD. Les codes sources sont disponibles.

2.1.3 - Compatibilité avec les technologies de virtualisation

PostgreSQL est compatible avec notamment VMWare et KVM. Cependant la pertinence de la virtualisation des bases de données est à vérifier. La virtualisation apporte des avantages en matière de résilience.

Les éditeurs d'outils de virtualisation et la communauté PostgreSQL fournissent des recommandations pour optimiser la configuration de PostgreSQL sur ces environnements. De manière générale, PostgreSQL a les mêmes limites que les autres SGBD en matière de virtualisation.

2.1.4 - Compatibilité avec les technologies de stockage

PostgreSQL fonctionne sur les baies de stockage (SAN, NAS,...) mais comme pour tout SGBD, l'attachement doit être permanent en mode blocs. La virtualisation du stockage est totalement transparente pour le SGBD.

Les communautés fournissent des recommandations sur les types de système de fichiers à utiliser (EXT4, …). Ne pas utiliser NFS.

2.1.5 - Compatibilité avec les technologies de sauvegarde

PostgreSQL supporte les modes de sauvegarde habituels :

- sauvegarde à froid : sauvegarde de niveau système de fichier – la base doit être arrêtée ;
- sauvegarde à chaud : sauvegarde SQL. Pas de possibilité de réaliser une sauvegarde incrémentale de manière native. L'outil barman le permet depuis la dernière version ;
- sauvegarde permanente : Point In Time Recovery (PITR).

PostgreSQL est compatible avec les outils de sauvegarde (TINA, TSM,...)

2.2 - Sécurité

2.2.1 - Gestion de l'identification des utilisateurs

PostgreSQL fournit les mécanismes d'authentification et gère l'attribution des privilèges (GRANT,...). Il permet également la séparation des schémas de base de données.

2.2.2 - Garantir la confidentialité et utilisation du chiffrement

Le chiffrement est possible via le module complémentaire pgCrypto permettant le chiffrement de colonnes par clé publique et clé privée (*Le ministère des affaires sociales a testé ce module*).

2.2.3 - Assurer la traçabilité et la journalisation

La traçabilité des actions dans PostgreSQL est assurée par des journaux :

- Journalisation du fonctionnement du moteur (démarrage, arrêt,...)
- Journalisation d'accès à PostgreSQL (requêtes, accès utilisateurs, erreurs,...)

3 - Scalabilité et résilience

3.1 - Mécanismes de clustering et de réplication

Définitions

Scalabilité et Elasticité

La scalabilité désigne la capacité de PostgreSQL à s'adapter à un changement d'ordre de grandeur de la demande (montée en charge ou diminution).

L'élasticité d'un système est la faculté d'un système à étendre ou réduire automatiquement ses ressources en fonction des besoins.

Résilience et Robustesse (ou stabilité)

La résilience est la capacité d'un système ou d'une architecture réseau à continuer de fonctionner en cas de panne.

La robustesse est la qualité d'un système qui ne plante pas, qui fonctionne bien même dans un environnement hostile (pénétration, DoS...) ou anormal (entrées incorrectes...)

Cluster

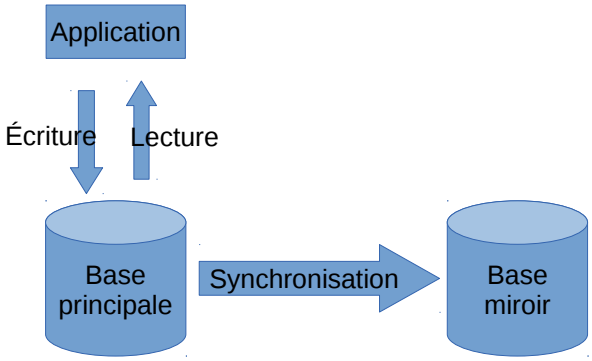
Un cluster est une grappe de serveurs (ou « ferme de calcul ») partageant un stockage commun. Un cluster fournit des fonctions de haute disponibilité et de répartition de charge.

Réplication

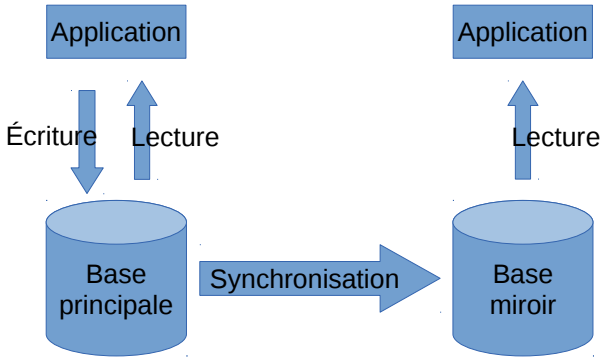
La réplication est un processus de partage d'informations pour assurer la cohérence de données entre plusieurs sources de données redondantes, pour améliorer la fiabilité, la tolérance aux pannes, ou la disponibilité.

Plusieurs modes de clustering et de réplication sont supportés par PostgreSQL avec les avantages et les inconvénients propres à chacun. Ces considérations sont indépendantes du choix du SGBD ; tous les produits sont concernés :

- mode « Actif-Passif » : une base principale est en lecture/écriture, l'autre base, dite « miroir » est synchronisée en arrière plan. La réplication peut être asynchrone (meilleures performances) ou synchrone (meilleure sécurité).



- mode « Actif-Actif partiel » ou «Read/Write - Read »: une base principale est en lecture/écriture, les bases « miroir » sont accessibles en lecture seule.



Tout comme d'autres SGBD, PostgreSQL ne permet pas d'assurer la continuité des transactions sur plusieurs serveurs en mode « Actif-Actif » ou « Read/Write – Read/Write ». Le projet BDR (Bidirectionnal Replication) de 2nd Quadrant propose cette fonctionnalité. Il s'agit toutefois d'une version modifiée de PostgreSQL.

Le passage en clustering est généralement irréversible. Il existe cependant des contributions permettant cette réversibilité.

3.2 - Résilience portée par le moteur PostgreSQL

PostgreSQL fournit les mécanismes de réplication pour la mise en place d'un cluster « actif-passif » ou « actif-actif partiel ».

3.3 - Résilience portée par les infrastructures de stockage

PostgreSQL est compatible avec le clustering, porté par la plateforme indépendamment du SGBD. L'écriture sur un nœud est réalisée par PostgreSQL et la réplication sur un second nœud est assurée par la baie de stockage.

Pour relancer le moteur sur le second nœud il y a obligatoirement une interruption de service.

Les MEDDE-MLET utilise PostgreSQL sur un seul nœud.

3.4 - Plans de continuité et de reprise informatiques

Les projets de plans de continuité et de reprises informatiques doivent être portés par l'application. Des recommandations de choix de scénarios sont faites suivant les besoins PRI/PCI indépendamment du SGDB.

PostgreSQL fournit les outils permettant une intégration dans les PRI/PCI.

4 - Développement sous PostgreSQL

4.1 - Typologie des données

PostgreSQL offre les types de données standards (alphanumérique, date, data, index, blob,...) ainsi que des types de données complexes (géospatiales, objets, hash, xml,...)

cf http://docs.postgresql.fr/9.3/datatype.html

Il est recommandé d'utiliser la norme ISO 8601 (http://fr.wikipedia.org/wiki/ISO_8601) pour les dates (YYYY-MM-DD).

Le type de donnée Binary Large OBject (BLOB) permet le stockage en base de données de contenus divers sous forme binaire (fichiers bureautiques, pdf, photos, audios, vidéos, multimédias,...).

PostgreSQL fournit deux modes de stockage des données binaires :

- directement dans la table en utilisant le type bytea ;
- dans une table séparée avec un format spécial qui stocke les données binaires et s'y réfère par une valeur de type oid dans la table principale en utilisant la fonction Large Object.

Ces deux modes supportent le streaming depuis la version 7.2 du driver JDBC PostgreSQL.

Le type de données bytea qui peut contenir jusqu'à 1 Go, n'est pas adapté au stockage de très grandes quantités de données binaires.

La fonction Large Object est mieux adapté au stockage de très grands volume. Il a cependant ses propres limites :

- la suppression d'un enregistrement ne supprime pas les données binaires associées. Une action de maintenance doit être effectuée à cette fin sur la base.
- tout utilisateur connecté peut accéder aux données binaires même s'il n'a pas de droits sur la base principale.

L'utilisation des champs BLOB est déconseillée s'il n'y a pas de besoin de recherche dans les informations.

4.2 - Schémas et structures des bases

Le partitionnement de table est implémenté en standard (http://docs.postgresql.fr/9.1/ddl-partitioning.html) ainsi que les index de type bitmap.

4.3 - Spécificités de développement

PostgreSQL est conforme au standard SQL 2011.

La page http://www.postgresql.org/docs/9.3/static/features.html propose des liens permettant de voir les fonctions effectivement couvertes et celles qui ne le sont pas encore : globalement, la couverture de la norme est très bonne même s'il est surprenant de voir que l'instruction MERGE n'est pas supportée.

PostgreSQL ne nécessite donc pas un apprentissage lourd pour la syntaxe des requêtes. Toutefois, les habitudes des développeurs quant à l'utilisation des fonctions spécifiques et des syntaxes particulières pour les jointures de leur SGBD habituel nécessiteront un accompagnement. Par exemple, les fonctions sur les dates sont propres à chaque SGBD. (cf Annexes par SGBD).

Il est possible de créer des bibliothèques de fonctions personnalisées pour simuler les fonctions spécifiques des autres SGBD et faciliter la migration et la prise en main.

Le paramétrage des jeux de caractères se fait en fonction des applicatifs. L'UTF-8 est recommandé pour toute la chaîne. L'ISO peut être également utilisé si nécessaire.

Attention aux options par défaut à l'installation de PostgreSQL : si aucun paramétrage n'est fait, initdb et create database utilisent la configuration de langue du serveur soit la page de codes LATIN9 (ISO 8859-15) ou à défaut de l'ASCII (http://www.postgresql.org/docs/9.3/static/app-initdb.html). Il faudra préciser UTF-8 lors de la création des bases de données.

PostgreSQL permet l'utilisation de vues matérialisées.

4.4 - API et modes d'accès

4.4.1 - Interfaces client

PostgreSQL fournit plusieurs interfaces client pour l'accès aux données :

Name	Language	Comments	Website
DBD::Pg	Perl	Perl DBI driver	http://search.cpan.org/dist/DBD-Pg/
JDBC	Java	Type 4 JDBC driver	http://jdbc.postgresql.org/
libpqxx	C++	New-style C++ interface	http://pqxx.org/
Npgsql	.NET	.NET data provider	http://npgsql.projects.postgresql.org/
pgtclng	Tcl		http://sourceforge.net/projects/pgtclng/
psqlODBC	ODBC	ODBC driver	http://psqlodbc.projects.postgresql.org/
psycopg	Python	DB API 2.0-compliant	http://initd.org/psycopg/

4.4.2 - Couche d'abstraction

L'utilisation d'une couche d'abstraction (mapping objets/tables) de type « ORM » est compatible.

4.4.3 - Pool de connexions

Un gestionnaire de connexion est fortement recommandé à partir de plusieurs centaines de connexions simultanées. Pgbouncer est stable et performant. Nous le recommandons. PgPool, pour sa part, est plutôt à conseiller pour réaliser du load-balancing.

4.5 - Procédures stockées, fonctions et déclencheurs (triggers)

Un déclencheur ou « trigger » est une action (procédure stockée ou requête SQL) déclenchée sur un événement.

Une procédure stockée est un programme stocké dans la base de données.

L'utilisation des procédures stockées et des déclencheurs est déconseillée pour éviter les adhérences et faciliter la portabilité. En effet, la logique métier ne doit pas être embarquée dans la base de données.

Les fonctions pourront être utilisées pour étendre l'usage des types de données pré-existants.

PostgreSQL permet l'écriture de fonctions et de procédures dans des langages différents du SQL et du C. Ces autres langages sont appelés génériquement des langages de procédures. Il existe à ce jour quatre langages de procédures dans la distribution standard de PostgreSQL™ :

- PL/pgSQL - Langage de procédures SQL ;
- PL/Tcl - Langage de procédures Tcl) ;
- PL/Perl - Langage de procédures Perl
- PL/Python - Langage de procédures Python.

Il existe d'autres langages de procédures qui ne sont pas inclus dans la distribution principale (PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh). D'autres langages peuvent être définis par les utilisateurs, mais la procédure peut être un peu complexe.

4.6 - Foreign Data Wrapper

Il s'agit d'extension qui permettent à PostgreSQL de communiquer avec d'autres sources de données. Les sources de données peuvent être des SGBD SQL (PostgreSQL, MySQL, Oracle, …), SGBD NoSQL (CouchDB, MongoDB, ..), voire des fichiers CSV, des annuaires LDAP. Le fait que les données proviennent d'autres sources est transparent pour l'utilisateur final.

Certains FDW ont des possibilités de lecture/écriture style Oracle, MySQL ; d'autres uniquement de lecture.

Pour obtenir la liste complète https://wiki.postgresql.org/wiki/Foreign_data_wrappers

5 - Administration

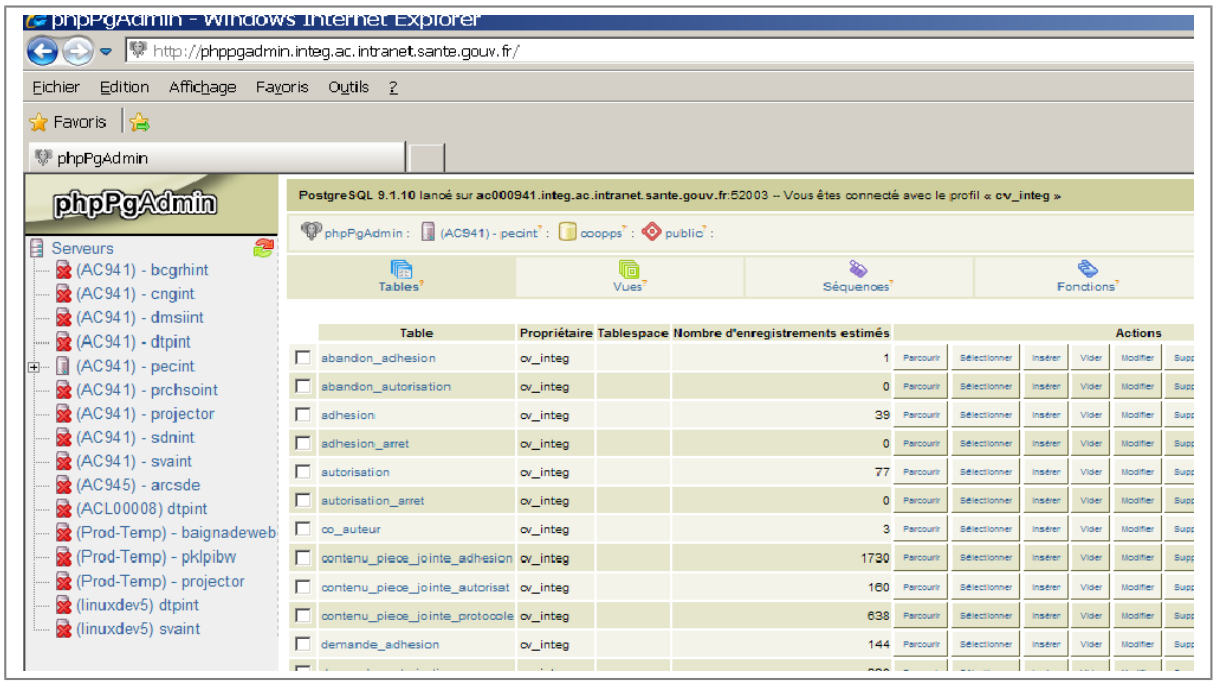
5.1 - Outils d'administration et d'exploitation de PostgreSQL

5.1.1 - phpPgAdmin

phpPgAdmin est une application Web d'administration réalisée en langage PHP destiné à faciliter la gestion du SGBD PostgreSQL. Elle permet l'accès distant sécurisé par le web.

Ce logiciel libre est distribué sous licence GNU GPL. Il est disponible à l'url suivante : http://phppgadmin.sourceforge.net/doku.php

Le principe est le suivant : l'utilisateur se connecte à l'url de l'instance apache de phppgadmin afin d'accéder à toutes les instances PostgreSQL installés sur le serveur. Il s'authentifie par le login et mot de passe.



Après connexion, il peut réaliser toutes les opérations classiques.

- Administrer la base de données
- Accéder au schéma de la base de données
- Faire des requêtes SQL
- Gérer les Tablespaces
- Exporter les données

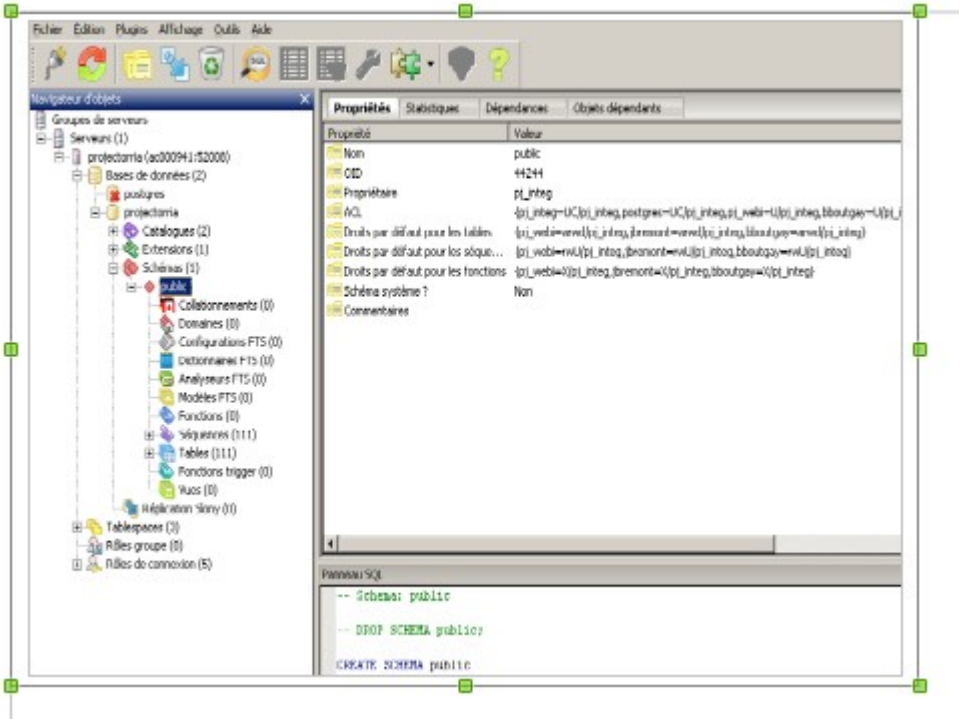
Ce logiciel est malheureusement aujourd'hui en perte de vitesse.

5.1.2 - PgAdminIII

PgAdminIII est une application d'administration en mode client-serveur. Il s'agit d'un logiciel libre diffusé sous license PostgreSQL. Cet outil est disponible sur toutes les plateformes. Il est installé de base lors de l'installation de PostgreSQL sous Windows. Mais il est disponible aussi à l'url suivante : http://www.pgadmin.org/download/source.php

Il possède une interface graphique, qui le rend utilisable par tous. Parmi les points forts, le mode graphique pour réaliser une requête, la personnalisation de l'affichage, l'ajout de plugins (par exemple PostGIS Shapefile and DBF loader) et la présence d'un langage de script.

Ce logiciel est mis à jour régulièrement.



5.1.3 - psql

L'outil psql permet une administration en ligne de commande, il permet la saisie de requêtes SQL, l'affichage du schéma de base, ainsi que l'import et l'export.

```
cv_integ@192.168.29.200's password:
Last login: Thu Jul 17 10:23:06 2014 from 10.100.70.69
ACOOO941-cv_integ> psql coopps
psql (9.1.10)
Saisissez Â« help Â» pour l'aide.

coopps=>
```

Pour plus d'information faire sous psql: \help.

5.2 - Outils de supervision de PostgreSQL et d'analyse des logs

Les outils de supervision (Nagios, Munin,...) proposent des modules complémentaires (plugins) pour PostgreSQL, permettant la surveillance des logs et tables systèmes. Le script perl check_postgres de Bucardo est le principal plugin pour ces outils. Il est utilisé au MAE.

D'autres modules permettent de suivre des statistiques d'évolution de la base (pg_stat_statements, pgtop,...) ou d'analyser les logs (pgbadger). Ce dernier est utilisé au MAE.

A noter que de nombreux modules (pgfouine, pgstatpack, …) ne sont pas maintenus sur la durée.

5.3 - Outils de migration des données vers PostgreSQL

L'outil « Ora2pg » permet d'analyser une base de données Oracle. L'extension Ora2pg de la DGFiP permet une évaluation des coûts de migration. L'outil permet également de réaliser la migration.

Pour le transfert des données, l'utilisation d'un outil ETL (Talend Open Studio,...) permet de réaliser des transformations sur les données au moment du transfert (adaptation des types de données,…).

Il est fortement recommandé d'expérimenter la migration dans des conditions les plus proches de la production (base de production, logs des sessions applicatives,…).

6 - Conduite du changement

6.1 - Formation des acteurs

La formation ciblera trois publics :

- les développeurs initiés : prise en main et migration vers PostgreSQL (2 à 3 jours) ;
- les exploitants et architectes : installation, maintenance, sauvegarde, supervision,... (3 jours) ;
- les administrateurs (DBA) : (formation exploitants + 3 jours).

6.2 - Support

Le support est assuré de plusieurs manières :

- par le marché interministériel porté par le ministère de l'intérieur ;
- en s'appuyant sur le réseau interministériel (MimPROD,...) ;
- par la communauté PostgreSQL.

Un support professionnel français est aussi possible:
http://www.postgresql.org/support/professional_support/europe/

6.3 - Plan de migration

La migration nécessite des adaptations plus ou moins importantes des applications pour lesquelles une recette visant à garantir la non régression doit être menée. Son coût est significatif.

Ce coût pourra être minimisé si la migration s'effectue à l'occasion d'une évolution fonctionnelle significative, qui nécessitera des recettes fonctionnelles et techniques indépendamment du changement de SGDB.

L'accompagnement des MOA et MOE voire de la production, pour une montée en compétences dans les meilleures conditions, devra également être planifié.

7 - Retour sur investissement

7.1 - Coût de migration de la base

La migration vers un nouveau SGDB doit tenir compte :

- du coût d'entrée (formation des acteurs, montée en compétences,...)
- des adaptations des applicatifs incluant les coûts de recettes (technique, fonctionnelle et tests de non régression)

7.2 - Coût de possession

PostgreSQL est sous une licence open source similaire aux licences BSD ou MIT. Il n'y a donc pas de politique tarifaire subie de la part d'un éditeur.

7.3 - Maîtrise des trajectoires

PostgreSQL est un produit dont la feuille de route (roadmap) est bien connue et maîtrisée.

L'écosystème de PostgreSQL s'enrichit constamment de nouveaux modules complémentaires et d'outils. Les progiciels commerciaux proposent de plus en plus d'interfaces nécessaires pour utiliser PostgreSQL.

La communauté PostgreSQL est importante et très active : la liste de diffusion http://www.postgresql.org/list/pgsql-fr-generale/ et le forum http://forum.postgresql.fr/

8 - Annexes

8.1 - Points d'attention lors de la migration depuis certains SGBD.

8.1.1 - Oracle

Contribution du ministère de l'intérieur

Il faut prendre connaissance des éléments des serveurs sources pour pouvoir définir les serveurs cibles adéquats en matière d'environnement technique, de piles logicielles, de système de stockage, de données, de criticité de l'application et du suivi de ses performances.

8.1.1.1 - Environnement technique

Les caractéristiques des serveurs Oracle :

Vérifier le type de serveurs : dédiés , mutualisés ou virtualisés ?

S'assurer des éléments techniques suivants :

- Types de processeurs
- Nombre de processeurs
- Fréquences horloge des processeurs
- Taille des registres
- Taille de la RAM

8.1.1.2 - Les piles logicielles

Relever les différentes piles logicielles composant l'application.

L'OS actuel pour quelle distribution de LINUX et quelle version ?

La version Oracle vers quelle version Postgres ?

Quels sont les serveurs d'application source ?

Le type de JVM

Le système de virtualisation.

8.1.1.3 - Espace de stockage

Quel est le système de stockage ? en général les SGBD sont hébergés sur du SAN (Storage Area Network) ; des solutions alternatives ne manquent pas.

Quel est le mode d'accès ? le plus souvent VMDK (Virtual Machine Disk).

8.1.1.4 - Migration de la base de données

Les outils de Migration.

Ils sont divers Ora2Pg , les ETL ou ELT, le développement des Programmes spécifiques,

etc...

Traiter d'abord les métadonnées avant le chargement des données.

MIGRATION DES STRUCTURES ORACLE vers POSTGRESQL : sont regroupés sous ce vocable les structures des tables et des vues avec leurs annexes, les différentes procédures, les fonctions, les triggers.

Procéder à la vérification de la présence éventuelle du partitionnement et des vues matérialisées.

Le partitionnement est basique sous PostgreSQL 9.1 , ce fut avant tout une solution de contournement et depuis la 9.2 il est une fonctionnalité à part entière.

Les vues matérialisées

De la 9.0 à 9.2 , elles passaient par des artifices et depuis la 9.3 c'est une fonctionnalité intégrée.

Le langage des procédures SQL

Oracle utilise le langage PL/SQL et PostgreSQL le PL/PGSQL ; ils sont assez proches mais une adaptation est requise.

Modèle de correspondance des types de données

Les chaînes de caractères peuvent être remplacées par VARCHAR et TEXT.

Les dates Oracle sur 7 octets par TIMESTAMP en 8 octets.

Les longues chaînes de caractères CLOB sont remplacés par TEXT.

Les valeurs numériques entières NUMBER selon la précision ascendante par SMALLINT, INTEGER, BIGINT, NUMERIC.

Les valeurs numériques décimales par NUMERIC

Les données binaires BLOB par BYTEA (attention au caractère d'échappement avec Ora2Pg).

Migration des procédures normales et stockées

Recenser toutes les procédures et fonctions PL/SQL et les transformer en PL/PgSQL.

Génération des SCRIPTS DE MIGRATION DE LA BASE DE DONNEES

Le script de migration des structures doit être séparé des données ; il servira à initialiser la base PostgreSQL.

Création du script d'initialisation de la base PostgreSQL

Il est créé à partir des structures des données, des procédures et fonctions réadaptées pour PostgreSQL.

La migration des données

Elle peut se faire par un script SQL de données à insérer dans la nouvelle base PostgreSQL.

Il est aussi possible d'opter pour un chargement direct de la base Oracle vers la base PostgreSQL.

Le code applicatif

La modification du code est nécessaire pour intégrer le driver PostgreSQL pour la gestion des transactions, de l'ouverture et fermeture des connexions, des mots clés Oracle à remplacer par les mots clés PostgreSQL, des jointures externes, de la pagination, etc...

8.1.1.5 - Criticité et sauvegardes

Prise en compte des notions de PRA/PCA, RTO, RPO, et REPLICATIONS.

Quelle solution technique est utilisée pour assurer la haute disponibilité : Oracle RAC, Oracle DATAGUARD ou autre ?

Il n'y a pas d'équivalent Oracle RAC sous PostgreSQL ; par contre la fonction DATAGUARD est assurée sous PostgreSQL par la réplication.

Le PRA (Plan de Reprise d'Activité) nous impose un délai de reprise d'activité de l'application. Il prend en compte le RTO (Recovery Time Objective ou la durée maximale d'interruption admissible) et le RPO (Recovery Point Objective ou la durée maximum d'enregistrement des données qu'il serait tolérable de perdre lors d'une panne). Cela conditionnera aussi en fonction de la volumétrie la technologie , la fréquence et le type de sauvegarde à mettre en place.

RMAN sous Oracle gère les sauvegardes et restaurations ; il n'y a pas d'équivalent de base sous PostgreSQL. Par contre, un outil tiers nommé Barman est en cours d'installation au MAE pour automatiser les sauvegardes et restaurations.. Par contre le mode Archivelog est bien pourvu dans les deux SGBD de façon quasiment identique.

Les restaurations physiques n'ont pas une granularité plus fine sous PostgreSQL (on restaure tout).

8.1.1.6 - Suivi, amélioration des performances ,et supervision

Il n'existe pas de monitoring global comme le Grid Control sous Oracle. Cependant il est possible d'utiliser Nagios avec le plugin check_postgres. D'autres outils sont également disponibles (powa, PGObserver, …).

Il faut mener une réflexion sur les différents outils disponibles.

8.1.2 - DB2

La migration amène à revoir le DDL issu de db2 : il est recommandé de prévoir un outil de transformation (script) du ddl prenant en compte les éléments décrits ci-dessous.

Concernant le DCL, il est préférable de ne pas essayer de transposer de db2 à pg les grants définis dans db2, les niveaux d'autorisation étant trop différents.

Ne pas oublier de modifier les procédures de maintenance : sauvegardes, historisation, défragmentation de la base, collecte de statistiques, nettoyage des fichiers obsolètes, etc.

8.1.2.1 - Quelques spécificités du DDL pg/db2

- Chaînes de caractères :

  - dans pg, une chaîne de caractères est définie en nombre de caractères.

  - dans db2, les chaînes de caractères fixes ou variables sont exprimées en octets avec pour conséquence un nombre de caractères possibles différent suivant l'encodage de la base : en utf-8 certains caractères sont codés sur plusieurs octets, en iso8859-15 chaque caractère vaut un octet. Une chaîne varchar(5) peut donc contenir moins de 5 caractère en db2 si la base est en utf-8 et la zone alimentée par des caractères accentués.

  En tenir compte si on doit migrer une base db2 utf-8 vers une base pg.

- Création d'une table dans un tablespace :

pg	db2
`create table... tablespace <nom_ts>`	`create table...IN <nom_ts>`

- Création d'une table via le mot clé LIKE :
  dans pg, l'utilisation des ( ) est obligatoire autour de la clause LIKE, dans db2 il n'en faut pas.

pg	db2
`create table eqos.statssov (LIKE eqos.stats INCLUDING ALL)`	`create table eqos.statssov LIKE eqos.stats`

- Valeur par défaut d'une colonne de table :

pg	db2
`create table...DEFAULT <valeur>`	`create table ...WITH DEFAULT <valeur>`

- Création d'une table :

  Certains mots-clés ne sont pas reconnus par pg donc à supprimer du ddl DB2 avant migration. Exemple : *append*, *with restrict on drop*, *long in*, toute la partie sur le *table-partionning* qui se défini différemment sous pg.

- Incrément automatique d'un compteur colonne sur création de table:

L'autoincrément de DB2 (*[generated always | generated by default] as identity*) n'est pas connu de pg, il faut utiliser les séquences exclusivement.

- L'utilisation du type serial de pg permet de créer une séquence facilement mais il ne faut pas mettre le type integer (déjà porté par serial) ni « not null » sous peine d'erreur de syntaxe.

  Dans DB2, la création de séquence est possible en passant obligatoirement par un « create sequence ». **Tous les autoincréments db2 sont donc à convertir.**

- Table non logguée :
  Une table db2 déclarée en « *not logged initially* » devra être « *unlogged* » sous pg.

- Tables temporaires :
  La syntaxe et les fonctionnalités des tables temporaires diffèrent.
  Exemple :
  db2 : declare global temporary table tabtemp like eqos.stats not logged
  pg : create local temporary table tabtemp (like eqos.stats) unlogged (?)

  Les tables temporaires db2 avec du partage de données inter-sessions se font via un « *create global temporary table* », elles n'ont pas d'équivalence dans pg car dans pg bien que syntaxiquement les mots-clés « *local* » et « *global* » sont acceptés, le comportement de la table temporaire reste local dans les 2 cas (pas de partage des données temporaires).

- Colonne d'horodatage sur mise à jour de ligne :
  Ces colonnes fréquemment utilisées dans db2 n'existent pas dans pg.
  Exemple dans db2 : une colonne de nom TS_UPDATE est mise à jour automatiquement dès que la ligne est modifiée par UPDATE/INSERT sql :

```
create table...
TS_UPDATE TIMESTAMP NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

  Lors d'une insertion de ligne ou un update sur la ligne, db2 renseigne automatiquement la colonne ts_update. Ces colonnes sont systématiquement déployées dans certaines bases db2, **sous pg il faudra faire cette action applicativement donc modifier les programmes en conséquence.**

- Format timestamp ISO :

  Non seulement la précision du timestamp est différente mais le séparateur entre date et time diffère également. **Attention lors de la migration des données de db2 vers pg** !

pg	db2
2014-01-31 00:00:00	2014-01-31**-**00.00.00**.000000**

A noter que le format « *NOT NULL DEFAULT CURRENT TIMESTAMP* » toléré dans DB2 n'est pas admis dans pg, il faut utiliser *CURRENT_TIMESTAMP*

(reconnu par DB2 également).

- Vue matérialisée :
  - pg : création via un « *CREATE MATERIALIZED VIEW* » .
  - db2 : création via un « *create table* » suivi d'options spécifiques à une vue matérialisée (appelée MQT dans db2). **Le ddl issu de db2 est donc à corriger.**

- Drop table et contraintes d'intégrité (CI):
  - pg : un « *drop table... CASCADE* » supprime les CI.
  - db2 : un « *drop table* » suffit à supprimer les CI.

- Création d'un index, schéma :

  - pg : un nom d'index ne doit pas être précédé d'un nom de schéma sinon on a une erreur de syntaxe

  - db2 : il est recommandé de mettre le nom de schéma et l'outil db2 de génération de ddl génère ce nom de schéma devant le nom d'ndex. S'il est omis dans db2, ce sera un schéma de même nom que le schéma courant ou que l'autorité de connexion qui sera utilisé.

- Création d'un index, options :

  Certains mots-clés ne sont pas reconnus par pg donc à supprimer du ddl db2 avant migration. Exemple : cluster, allow reverse scan, pctfree...

- Affectation des index à un tablespace dédié :
  - pg : sur la création de la table, l'affectation se fait dans la clause de création de la clé primaire ou de l'unicité. On peut mettre l'index dans un tablespace dédié lors de la création de l'index.
  - db2 : on peut directement diriger tous les index liés à une table dans un tablespace avec la clause *INDEX IN*, exemple :
    create table.. IN <nom_ts_table> INDEX IN <nom_ts_index>.
    Ou lors de la création de l'index comme sous pg.

- Taille des pages :
  - pg : seules les pages de 8ko sont permises.
  - db2 : on travaille avec des tailles de pages de 4, 8, 16 ou 32ko. **Modifier le ddl db2 qui préciserait explicitement des tailles de pages.**

- Bufferpool :
  La notion db2 de bufferpool n'existe pas dans pg, on ne crée pas ces ressources donc le ddl db2 doit être adapté en conséquence en retirant toutes les instructions de création mais aussi les références aux bufferpools dans les tablespaces.

- Tablespaces :
  les options de création des ts diffèrent entre db2 et pg, à corriger.

8.1.2.2 - DCL

- Rôles vs groupes Unix :

- pg : il faut impérativement passer par des grants donnés à des rôles.

- db2 : les grants sont donnés soit à des rôles soit directement aux groupes Unix ce qui est le cas le plus fréquent au MEN.

- Droits PUBLIC :

  La notion db2 de base RESTRICTIVE qui supprime les grants dits PUBLIC n'existe pas sous pg.

- GRANT ALTER TABLE :

  Utilisé sous db2, il n'existe pas tel quel sous pg. Il faut sous pg faire appartenir le compte souhaitant modifier la table à un rôle qui est OWNER de la table.

- TRUNCATE TABLE :
  - pg : il faut faire un « *grant truncate* » pour être autorisé à faire un truncate.
  - db2 : un « grant delete » (ou control) donne ce privilège.

- GRANT CONNECT :

  Pour être autorisé à se connecter à une database, il faut faire :

  - pg : *grant connect on database <nom_base>…*

  - db2 : *grant connect on database* il ne faut pas préciser le nom de la base car le grant est fait connexion déjà prise sur la base, donc c'est la base courante. Si on précise le nom de la base on a une erreur de syntaxe.

  D'autres différences de syntaxe peuvent apparaître.

- GRANTS sans équivalence db2/pg :

  Une partie des grants DB2, ceux liés à des fonctionnalités non présentes ou différentes sous pg, ne sont pas reportés sous pg. C'est le cas des *grant load*, *grant use of tablespace*, *grant usage on workflow*, etc.

- **Catalogue système**

  Les informations sont stockées en majuscules dans DB2, en minuscules dans pg. En tenir compte dans la recherche d'information et notamment dans les scripts qui utilisent le catalogue pour y collecter des informations.

  Exemple : rechercher les informations des tables appartenant à un schéma de nom IDENTITE

pg : `select * from pg_tables where schemaname = 'identite'`
db2 : `select * from syscat.tables where tabschema = 'IDENTITE'`

8.1.2.3 - Autres considérations

- Utilitaire de chargement :

- db2 : une table db2 peut être chargée par import, load, ingest ou db2move.
- pg utilise exclusivement l'utilitaire copy qui est bien moins riche que les utilitaires db2. Il ne semble pas possible par copy de charger partiellement une table comme dans db2.

- Autres utilitaires db2/pg:
  reorg devient *vacuum*
  runstats devient *analyse*
  backup devient *pg_dump, pg_dumpall ou pg_basebackup*

- Double quote ":
  Lors de la migration, prendre garde à la double quote " : elle est le délimiteur de chaîne par défaut dans db2 quand on exporte les données donc elle se retrouve dans le fichier déchargé (export) entourant chaque chaîne de caractère. Au chargement vers pg, cette double quote se retrouve par défaut dans la table si elle est présente dans le fichier de données. Pour éviter ce chargement non souhaité, les fichiers exportés depuis db2 doivent être en mode « del » (ascii) et importés dans pg comme du csv via les options *WITH CSV delimiter ',' QUOTE '"'* sur le copy.

- Journaux de transaction :
  - db2 : chaque base possède ses propres journaux.
  - pg : ce n'est pas le cas, les journaux sont communs à plusieurs bases au sein d'une même instance ce qui peut poser des problèmes (journaux mobilisés par une base, impact multi-bases en cas de corruption ou de disparition des journaux....).

- LOBs :
  - les BLOB db2 sont à remplacer par des colonnes au format BYTEA
  - les CLOB db2 sont à remplacer par le format TEXT
  - contrairement à DB2 qui permet d'embarquer les LOBs dans sa sauvegarde (backup), pg ne sauvegarde pas les LOBs via pg_dumpall (pg sauvegarde les LOBs vi pg_dump).
  - les options db2 liées aux LOBs ne s'appliquent plus sous pg (*logged/not logged*, *compact/not compact*, *inline length*...).

- Procédures stockées :
  Ecrites dans db2 en SQL/PL, elles devront être adaptées à pg (PL/PgSQL).

- Code source (programmes) :
  - modifier les méthodes d'appel du driver et l'utilisation de ses properties.
  - changer le code SQL qui pourrait être spécifique à db2/pg

8.1.3 - Informix

Contribution du ministère des affaires sociales

8.1.3.1 - Structure

Schéma des bases de données (tables, indexes, contraintes, …).

Des scripts devront permettre la création des différentes bases de données sous PostgreSQL.

Il faudra s'assurer que l'ensemble des objets soient créés en respectant la syntaxe de **PostgreSQL 9.1**, ainsi que les normes préconisées :

- Tables,
- Vues,
- Triggers,
- Contraintes,
- Indexes

Le tableau suivant décrits quelques différences pour les objets précités entre les SGBD Informix et PostgreSQL :

Objets / SGBD	Informix	PostgreSQL
Type de données	BLOB	BYTEA
	DATETIME	TIMESTAMP
Contraintes (clé étrangère)	alter table nom_table add constraint (foreign key (nom_colonne) references adr constraint nom_contrainte) ;	alter table nom_table add constraint nom_contrainte foreign key (nom_colonne) references adr (nom_colonne);
Index	create index nom_index on nom_table (nom_colonne) using btree ;	create index nom_index on nom_table using btree(nom_colonne) ;

8.1.3.2 - Plan à suivre lors de la migration des bases de données

Les étapes suivantes seront suivies lors de la migration :

i. Dans les bases de données sources (Informix), suppression des procédures stockées n'étant pas utilisées par les applications java, cela afin d'éviter de migrer vers PostgreSQL des traitements qui ne seront jamais exécutés.

ii. Création des bases de données :

   a Création des tables

   b Création des triggers

   c Création des procédures stockées

iii. Chargement des données

iv. Création des indexes et des contraintes. Les indexes et contraintes seront créées après le

chargement des données afin d'optimiser la durée d'exécution du traitement.

8.1.3.3 - Intégration du framework hibernate

Si aucun framework n'est utilisé dans une application pour gérer la persistance des objets en base de données relationnelle, la solution serait l'utilisation du framework Hibernate.

Cette solution permettrait de ne pas écrire en dur les requêtes SQL dans le code source Java, les requêtes étant générées par Hibernate.

Les applications seraient donc moins dépendantes d'un **SGBD** (il resterait les procédures stockées).

Les applications qui utilisent le framework « Hibernate » gérant la persistance des objets en base de données relationnelle. Les requêtes SQL ne sont donc pas écrites dans le code Java, contrairement aux applications sans Hibernate, mais générées par Hibernate.

La migration d'Informix vers PostgreSQL aura donc un impact limité sur les sources de ces applications.

Des tests seront toutefois nécessaires afin de s'assurer qu'il n'y ait pas de régression.

8.1.3.4 - Risques

Ce type de risques peut être identifié :

- Interruption de l'utilisation des applications durant la migration pendant la phase de chargement des données des bases Informix vers les bases PostgreSQL.

- Une solution permettant de réduire la durée d'exécution consisterait à charger dans un premier temps les données statiques (comme les tables de nomenclature), pour ne charger que les données susceptibles d'évoluer (demandes, dossiers, …).

- le jour de la migration en production. Cette solution serait à envisager dans le cas où la durée du chargement dans l'environnement du Ministère serait trop longue et si un gain de temps est réalisé (cela dépend de la proportion des données pouvant être qualifiées comme données statiques).

- Ralentissement de certains traitements qui avaient été optimisés pour le SGBD Informix IDS.

8.1.4 - MS SQL

Quelques éléments :

Les outils pouvant faciliter la migration des données sont

- https://github.com/dalibo/sqlserver2pgsql
- http://pgloader.io/howto/quickstart.html (produit plus récent)

Pour la migration des procédures, le code est très différent et doit être réécrit.

8.2 - Quelques références

8.2.1 - Quelques sites utilisant PostgreSQL

Meteo France (http://www.postgresql.fr/temoignages:meteo_france)
Volume de données : 3.5 To

Le Bon Coin (http://www.postgresql.fr/temoignages:le_bon_coin)
Volume de données : > 6 To

IGN (http://www.postgresql.fr/temoignages:ign)
Capacité de traiter plus de 100 millions d'objets géométriques ;

Mappy (http://www.oslandia.com/oslandia-et-mappy-vers-lopensource.html)
75 Go pour la base PostGIS

8.2.2 - Quelques références au MINEFI-MEDDE

Le MEDDE utilise PostgreSQL depuis 10 ans. A ce jour plus de 250 applications fonctionnent sous PostgreSQL.

Droit des sols (ADS)
- Volume de données : 95 Go

Permis de construire (Sitadel)
- Volume de données : 60 Go

Gestion de l'info trafic Ile de France (Sytadin)
- Volume de données : 3 Go

8.2.3 - Quelques références au MINEFI-DGFiP

FIBANC
- L'application permet à la B.R.S (DNEF) d'exercer son droit de communication en vertu du BOI 13 K-2-88.
- Volume de données : 2 Go

PROCOLMAS
- Silo des procédures collectives.Ce silo est destiné à réceptionner et à stocker les procédures collectives issus du BODACC-A et les mettre à disposition du SI Copernic.
- Volume de données : 2 Go

PATRIM Usagers
- Offre aux usagers et aux agents, via le portail usager ou le portail métier, un nouveau service de recherche de terme de comparaison concernant les valeurs foncières de leurs biens immobiliers.

- Bases hors ligne pour traitements batchs : 330 Go au total dont une de 280 Go
- Bases en ligne (optimisées pour des sollicitations en lecture seule de type datamart) : 70 Go au total dont deux de 30 Go
- Utilise massivement la cartouche spatiale postgis.

PATRIM Colloc

- Traitement automatisé des demandes d'informations géographique de Patrimoine Immobilier issues des collectivités locales.
- Une base batch de 13 Go

TREVI

- Télédéclaration des locaux professionnels dans le cadre de la révision des valeurs locatives foncières. Application orientée usagers professionnels.
- Base de gestion de 60 Go

FNTD

- Fichier National des Tiers Déclarants (vérification des obligations déclaratives).
- Base de gestion de 20 Go

8.2.4 - Quelques références au MINEFI-DGDDI

BANACO

- Application interne qui fournit les outils permettant de collecter les informations nécessaires au suivi de la politique des contrôles douaniers.
- Base de 193 Go

EDDI

- Edition des états déclarants journaliers, décadaires et mensuels. Offre deux points d'accès aux états : sur Aladin pour les services douaniers et sur Prodouane pour les opérateurs.
- 130 utilisateurs douaniers
- 2500 utilisateurs opérateurs
- 22 Go

TSAR

- Téléservice (Traitement et suivi de l'assistance en réseau) permettant l'assistance aux utilisateurs du système d'information de la douane.
- Nombre total d'utilisateurs : 60 000, nombre maximum d'utilisateurs concurrents : 100
- 242 Mo pour une cible de 12 Go

8.2.5 - Quelques références au MENESR (DNE)

DEMACT

- L'application permet la gestion des actes administratifs et financiers dans les collèges et les lycées.
- Pour un académie et un échantillon de 20 établissements : 150 Mo

Moteur de règles (BRMS)

- Volume de données : 120 Go

GED SIRHEN

- Pour une population de 4000 agents : 10 Go

8.2.6 - Quelques références au MAE

Etudes en France

- Volumétrie attendue de 1 To

Portail interministériel de la correspondance diplomatique (Diplomatie)

- Volume de données : Projection : 20 Go de croissance annuelle

8.3 - Extensions et plugins pour PostgreSQL

Le tableau ci-dessous présente quelques plugins évoqués dans le document :

PostGIS	Cartouche spatiale
pgstatpack	Statistiques
pgCrypto	Cryptographie

8.4 - Outils tiers pour PostgreSQL

Le tableau ci-dessous présente quelques outils tiers évoqués dans le document :

barman	Sauvegarde / restauration
pgtop	Statistiques
pgfouine	Analyse des logs
pgbadger	Analyse des logs
pgPool	Gestion des pools de connexions

9 - Documents de référence

Notes techniques Dalibo (ces documents sont accessibles aux clients de Dalibo)

- Étude sur la sauvegarde et la restauration et la migration de PostgreSQL ;
- Étude sur la mise en place d'un PCA/PRA sous PostgreSQL ;
- Scalabilité et performances avec PostgreSQL.

Documentation PostgreSQL

- http://docs.postgresql.fr/9.3/