Межминистерская дирекция
по информационным системам
и средствам связи

Межминистерский центр
по свободному программному обеспечению

19 марта 2015

PostgreSQL:
помощь в принятии решения

Версия 1.01

Предыдущие версии документа

Версия	Дата	Комментарий
0.1	18/12/2013	Рабочая версия (черновик)
0.2	07/02/2014	Первое обсуждение
0.3	19/03/2014	Второе обсуждение
0.4	01/07/2014	Подготовка к опубликованию
0.5	04/08/2014	Включение замечаний
1.0	19/03/2015	Утверждение и последние изменения
1.01	26/06/2015	Уточнение о лицензии

Составители

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

Содержание

Введение

Это руководство было разработано в рамках осуществления предписаний циркуляра «Ayrault» от 19 сентября 2012 года, посвящённого использованию свободного программного обеспечения в административных учреждениях, который повлёк создание Межминистерского центра по свободному программному обеспечению (Socle Interministériel du Logiciel Libre – SILL). Это руководство посвящено использованию PostgreSQL вместо коммерческих программ. Его цель - ответить на вопросы пользователей о работе с PostgreSQL, а также, не входя в технические подробности, показать преимущества PostgreSQL, описывая механизмы обеспечения совместимости, безопасности и надёжности.

1 — Ответ на нужды пользователей

1.1 — Варианты использования

PostgreSQL — это система управления базами данных на языке SQL, который традиционно используется для работы с базами данных в том числе и в других коммерческих программах такого типа (Oracle, DB2, Informix, MS SQL, Sybase,...).

PostgreSQL уже готов ответить на нужды своих пользователей:

PostgreSQL предлагает расширение для геоинформатики (PostGIS) в соответствии со стандартами Open Geospatial Consortium (OGC).

PostgreSQL также может быть использован в сфере «business intelligence» как хранилище данных в связке с инструментами для их обработки (BusinessObjects, Pentaho,...).

Существуют многочисленные свободные программные продукты, которые изначально используют PostgreSQL (Gestion Electronique de Documents (GED), Moteurs de règles, GroupWare, Supervision,...).

PostgreSQL поддерживает также обработку данных в фоновом режиме, пакетную обработку и обработку с задержкой (batch,...).

1.2 — Доступ к данным

PostgreSQL разработан в соответствии с нормами SQL 2011, языка запросов в многочисленных системах управления базами данных (СУБД). Таким образом работа с теми СУБД, в которых соблюдаются эти стандарты, не представляет сложности.

1.3 — Требования безопасности

PostgreSQL отвечает требованиям безопасности в том, что касается доступности, целостности, конфиденциальности и отслеживаемости (DICT[1]).

1 Критерии безопасности DICT (Wikipedia): https://fr.wikipedia.org/wiki/DICT

Требования криптографии и надежной проверки подлинности осуществляются дополнительными модулями.

PostgreSQL располагает активной командой разработчиков, которая поставляет обновления к системе безопасности и следит за устареванием компонент. Основная версия поддерживается в течение 5 лет.

PostgreSQL гарантирует целостность обрабатываемых данных даже в случае сбоя благодаря своим свойствам атомарности, согласованности, изолированности и надёжности (ACID[2]).

PostgreSQL изначально предлагает механизмы, отвечающие требованиям конфиденциальности и управления правами. Свойства ACID также гарантируются механизмами, которые обеспечивают управление транзакциями.

1.4 — Сценарии перехода информационных систем (ИС) на PostgreSQL

Переход ИС влечёт существенные временны*е* затраты, которые складываются в основном из следующего:

- перенос данных, не смотря на то, что существуют многочисленные утилиты, позволяющие осуществить этот перенос в зависимости от исходной СУБД;

- исправления в исходном коде приложения. Важность этих исправлений связана с использованием специфических функциональностей в существующем коде (хранимые процедуры, триггеры, нестандартные функции) и использованием абстракции данных (Hibernate);

- тест, гарантирующий сохранение функциональности, т.е. что в результате миграции не возникло никаких ошибок (регрессионное тестирование).

Исходя из этого, рекомендуем приурочить на PostgreSQL только к обновлению функциональности приложения. В этом случае трудоёмкие тесты и проверка функциональности системы будут необходимы только один раз после осуществления совокупности мероприятий. (Именно так поступили в министерствах Экологии, Длительного Развития и Энергетики, а также в министерстве Экономики, Финансов и Промышленности).

Несмотря на вышесказанное, амбициозный план глобального перехода информационных систем может быть предпринят (как в случае министерства социальных дел, которое предприняло запланированный перевод своих информационных систем Informix на PostgreSQL).

2 Свойства ACID (Wikipedia)

2 – Интеграция PostgreSQL с техническими платформами

2.1 — Технические платформы

2.1.1 – Архитектуры процессоров

PostgreSQL работает на процессорах следующих архитектур: x86, x86_64, IA64, PowerPC, PowerPC 64, S/390, S/390x, Sparc, Sparc 64, Alpha, ARM, MIPS, MIPSEL, M68K et PA-RISC.[3]

2.1.2 — Операционная система

PostgreSQL работает на большинстве операционных систем. Бинарные пакеты имеются для семейства Red Hat (включая CentOS/Fedora/Scientific), семейства Debian GNU/Linux и его разновидностей, семейства Ubuntu Linux и его разновидностей, SuSE, OpenSuSE, Solaris, Windows, MacOSX, FreeBSD, OpenBSD. Исходные коды доступны.

2.1.3 - Совместимость с технологиями виртуализации

Наилучшая совместимость у PostgreSQL обеспечена с VMWare и KVM. И всё же уместность виртуализации баз данных необходимо проверять. Виртуализация улучшает свойства отказоустойчивости. Создатели инструмента виртуализации и сообщество разработчиков PostgreSQL поставляют рекомендации для оптимизации конфигурации PostgreSQL для этих окружений. В общем, в том, что касается виртуализации, PostgreSQL имеет те же ограничения, что и другие СУБД.

2.1.4 — Совместимость с технологиями хранения

PostgreSQL работает на дисковых массивах (SAN, NAS,...), но, как и для всякой СУБД, расположение блоков должно быть постоянным. Виртуализация хранения совершенно прозрачна для этой СУБД. Сообщества поставляют рекомендации о типах системы файлов, которые должны использоваться (EXT4, …). Не использовать NFS.

2.1.5 — Совместимость с технологиями резервного копирования

PostgreSQL поддерживает обычные способы резервного копирования:

- холодное копирование: копирование на уровне системы файлов — база должна быть остановлена;
- горячее копирование: копирование SQL. Первоначально не было возможности осуществлять резервное копирование только тех файлов, которые были изменены после предыдущего копирования. Но инструмент barman позволяет это делать, начиная с последней версии;
- непрерывное копирование: Point In Time Recovery (PITR).

PostgreSQL совместим с инструментами резервного копирования (TINA, TSM,...).

2.2 — Безопасность

2.2.1 — Управление идентификацией пользователей

PostgreSQL обеспечивает механизмы идентификации и управляет присвоением

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

привилегий (GRANT,...). Он также позволяет разделять схемы баз данных.

2.2.2 — Гарантия конфиденциальности и использование шифрования

Шифрование возможно с помощью дополнительного модуля pgCrypto, который позволяет шифрование столбцов с помощью публичного ключа и частного ключа. *(Этот модуль протестирован Министерством социальных дел)*.

2.2.3 — Обеспечение отслеживаемости и журналирование

Отслеживаемость действией в PostgreSQL обеспечивается журналированием:

- записи функционирования сервера (старт, остановка и т.д.);
- записи обращений к PostgreSQL (запросы, доступ пользователей, ошибки и т.д.).

3 — Масштабируемость и отказоустойчивость

3.1 — Механизмы кластеризации и репликации

Определения

Масштабируемость и Эластичность

Масштабируемость означает свойство PostgreSQL адаптироваться к изменению объема запросов (увеличение нагрузки или уменьшение).

Эластичность системы — это её способность автоматически расширять или сокращать свои ресурсы в зависимости от требований.

Отказоустойчивость и Робастность

Отказоустойчивость — это свойство системы или архитектуры сети продолжать функционировать в случае аварии (сбоя).

Робастность — это такое качество системы, когда она не прекращает работу и нормально функционирует даже в неблагоприятных условиях (хакерские атаки, DoS-атаки...) или в анормальном режиме (некорректные входы...).

Кластер

Кластер — это совокупность серверов (или «вычислительная ферма»), имеющая общее хранилище. Кластер обеспечивает высокую доступность и распределение нагрузки.

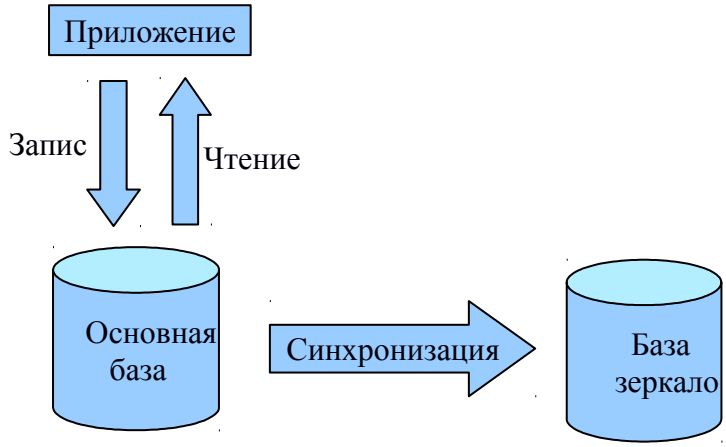
Репликация

Репликация — это процесс распределения информации с целью обеспечить синхронизацию данных на различных источниках, содержащих копии этих данных, для

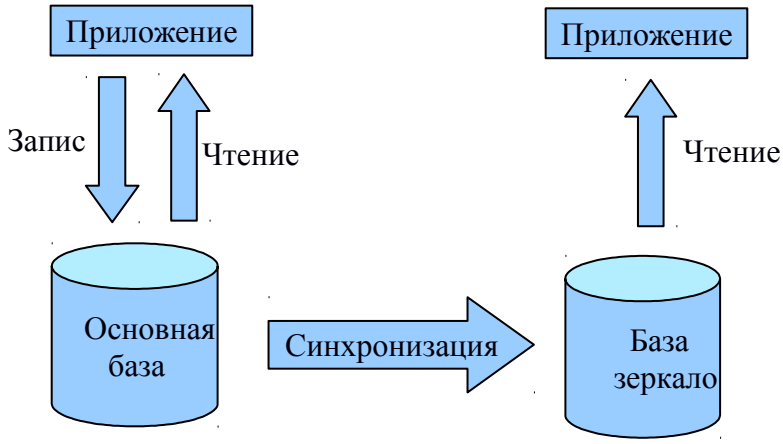
улучшения надёжности, устойчивости к сбоям или доступности системы.

PostgreSQL поддерживает различные способы кластеризации и репликации с преимуществами и недостатками, присущими каждому из этих способов. Эти замечания не зависят от выбора СУБД и касаются всех продуктов:

- Способ «Активный-Пассивный»: основная база в состоянии чтение/запись, тогда как другая база, называемая зеркалом, синхронизирована в фоновом режиме. Репликация может быть асинхронной (наилучшая производительность) или синхронной (наилучшая безопасность).



- Способ «Активный — частично активный» или «Read/Write - Read »: основная база в состоянии чтение/запись, базы-зеркала доступны только для чтения.



Как и другие СУБД, PostgreSQL не позволяет обеспечить непрерывность транзакций на нескольких серверах способом «Активный-Активный» или «Read/Write–Read/Write». Проект BDR (Bidirectional Replication) от 2nd Quadrant предоставляет такую возможность. Однако в этом случае речь идёт о модифицированной версии PostgreSQL.

Обычно переход в кластерный режим необратим. Однако существуют способы, делающие такую обратимость возможной.

3.2 — Отказоустойчивость сервера PostgreSQL

PostgreSQL обеспечивает механизмы репликации для установки кластера «активный-пассивный» или «активный-частично активный».

3.3 — Отказоустойчивость инфраструктуры хранения

PostgreSQL совместим с кластеризацией, осуществляемой платформой независимо от СУБД. Запись на узел осуществляется PostgreSQL, а репликация на второй узел обеспечивается с помощью подсистемы хранения данных

Чтобы запустить сервер на втором узле, необходимо прерывание услуг.

MEDDE-MLET [*это названия министерств]* использует PostgreSQL на единственном узле.

3.4 — Планы бесперебойности и возобновления работы системы

Проекты планов бесперебойности и возобновления должны содержаться в приложении. Рекомендации выбора сценария должны делаться в зависимости от нужд PRI/PCI независимо от СУБД.

PostgreSQL поставляет инструменты, позволяющие интеграцию в PRI/PCI.

4 — Разработка в PostgreSQL

4.1— Типы данных

PostgreSQL предлагает стандартные типы данных (алфавитно-цифровые, date, data, index, blob,...), а также комплексные типы данных (геокосмические, объекты, hash, xml,...). Например, http://docs.postgresql.fr/9.3/datatype.html

Рекомендуется использовать формат ISO 8601 (http://fr.wikipedia.org/wiki/ISO_8601) для дат (YYYY-MM-DD).

Тип данных Binary Large OBject (BLOB) позволет хранить в базе данных различное содержимое в двоичной форме (офисные файлы, pdf, фото, аудио, видео, мультимедийные данные...).

PostgreSQL обеспечивает два способа хранения двоичных данных:

- непосредственно в таблице, используя тип bytea ;
- в отдельной таблице со специальным форматом, который хранит двоичные данные и ссылается на них с помощью значения типа oid в основной таблице, используя функцию Large Object.

Эти два способа поддерживают стриминг начиная с версии 7.2 драйвера JDBC PostgreSQL.

Тип данных bytea, который может содержать до 1 ГБ, не адаптирован для хранения очень больших объёмов двоичных данных.

Фнукция Large Object лучше подходит для хранения очень больших объёмов. Однако у неё есть свои ограничения:

- удаление записи не удаляет соответствующие двоичные данные. Для этого требуется произвести дополнительные действия по техническому обслуживанию базы;
- любой пользователь, соединившийся с базой, может получить доступ к двоичным данным, даже если у него нет прав на основной базе.

Использование полей BLOB не рекомендуется, если нет необходимости осуществлять поиск информации.[4]

4.2 — Схемы и структуры баз

Секционирование таблиц реализовано в стандарте (http://docs.postgresql.fr/9.1/ddlpartitioning.html) также как индексы типа bitmap.

4.3 — Особенности разработки

PostgreSQL соответствует стандарту SQL 2011.

На странице http://www.postgresql.org/docs/9.3/static/features.html приведен список функций, полностью соответствующих стандарту, и тех, которые ещё таковыми не являются : в целом, соответствие стандарту очень хорошее, хотя и удивительно, что инструкция MERGE не поддерживается.

Таким образом, PostgreSQL не требует длительного изучения синтаксиса запросов. Однако, привычки разработчиков, касающиеся использования специфичных функций и синтаксиса, для привязки их привычной СУБД потребуют изменения. Например, функции для работы с датами являются разными для разных СУБД (см. Приложения).

Существует возможность создавать библиотеки собственных функций, чтобы имитировать специфичные функции других СУБД и облегчить переход и начало работы.

Кодировка текста осуществляется в зависимости от применений. UTF-8 рекомендуется для всей цепочки. В случае необходимости кодирока ISO может быть также использована. Обратите внимание на опции по умолчанию при установке PostgreSQL: если никакая кодировка не указана, initdb и create database используют конфигурацию языка сервера, либо страницы кодов LATIN9 (ISO 8859-15), либо, за неимением, конфигурацию ASCII. http://www.postgresql.org/docs/9.3/static/app-

4 Скорее наоборот (прим. перев.)

initdb.html). Нужно будет указывать UTF-8 при создании базы данных.

PostgreSQL позволяет использовать материализированные представления.

4.4 — API и способы доступа

4.4.1 — Клиентские интерфейсы

PostgreSQL предоставляет несколько клиентских интерфейсов для доступа к данным:

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 — Абстракция данных

Использование абстракции данных (mapping objets/tables) типа ORM совместимо.

4.4.3 — Пул соединений

Начиная с нескольских сотен одновременных соединений, настоятельно рекомендуется администратор (менеджер) соединений. Pgbouncer стабилен и эффективен. Мы его и рекомендуем. PgPool, в свою очередь, скорее подходит для осуществления load-balancing.

4.5 — Хранимые процедуры, функции и триггеры

Триггер — это действие (хранимая процедура или запрос SQL), срабатывающее при каком-либо событии.

Хранимая процедура — это программа, сохранённая в базе данных.

Использование хранимых процедур и триггеров не рекомендуется, чтобы избегать привязки (зависимости между программами и модулями, которые начинают зависеть друг от друга) и обеспечить мобильность (базы данных с одного компьютера на другой). Действительно, бизнес-логика не должна быть встроена в базу данных. Функции могут использоваться, чтобы расширить использование уже существующих типов данных. PostgreSQL позволяет написание функций и процедур на языках, отличных от SQL и C. Эти другие языки обычно называют процедурными языками. В настоящее время в стандартном дистрибутиве PostgreSQL™ существует четыре процедурных языка :

- PL/pgSQL — процедурный язык SQL ;

- PL/Tcl - процедурный язык Tcl ;
- PL/Perl - процедурный язык Perl ;
- PL/Python - процедурный язык Python.

Существуют и другие процедурные языки, которые не включены в основной дистрибутив (PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh). Другие языки могут быть определены пользователями, но процедура может быть немного сложной.

4.6 — Foreign Data Wrapper (FDW)

Речь идёт о расширении, которое позволяет PostgreSQL общаться с другими источниками данных. Такими источниками могут быть СУБД SQL (PostgreSQL, MySQL, Oracle, …), СУБД не-SQL (CouchDB, MongoDB, ..), а также файлы CSV, каталоги LDAP. Тот факт, что данные взяты из других источников, прозрачен для финального пользователя.

Некоторые FDW обладают возможностями чтения/записи стиля Oracle, MySQL; другие - только чтения.

Для получения полного списка смотрите ссылку
https://wiki.postgresql.org/wiki/Foreign_data_wrappers

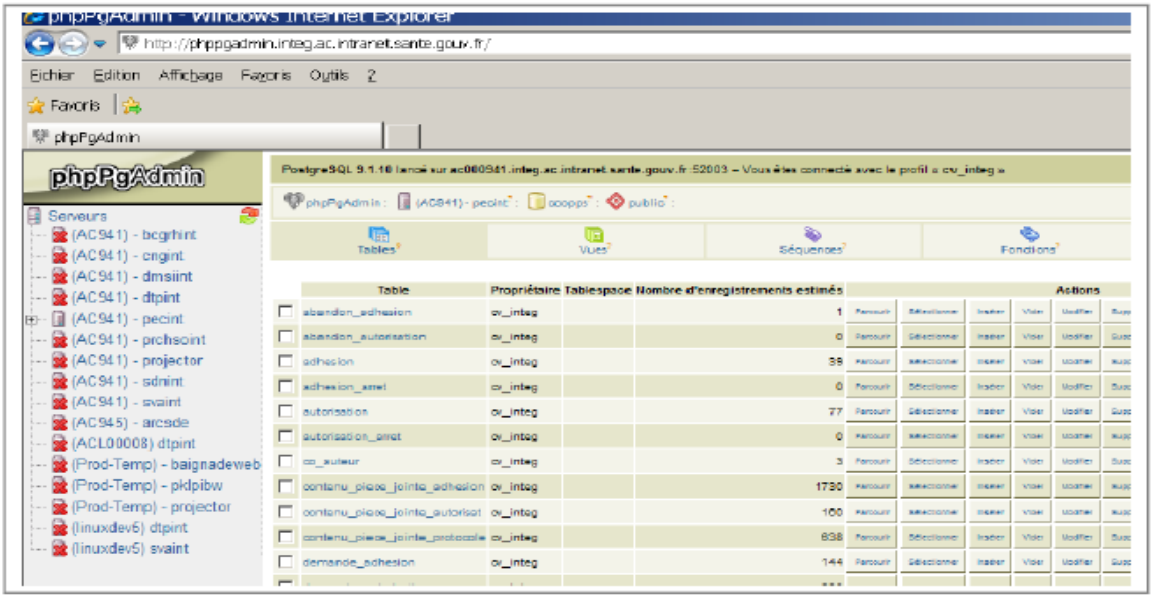
5 — Администрирование

5.1— Инструменты администрирования и эксплуатации PostgreSQL

5.1.1 — phpPgAdmin

phpPgAdmin — это веб-приложение администрирования, реализованное на языке PHP и предназначенное для облегчения управлением СУБД PostgreSQL. Оно позволяет осуществлять удалённый безопасный доступ через интернет. Это свободное программное обеспечение распространяется по лицензии GNU GPL. Оно доступно по следующему адресу:
http://phppgadmin.sourceforge.net/doku.php

Принцип следующий: пользователь соединяется по интернету с инстанцией apache phppgadmin, чтобы получить доступ ко всем инстанциям PostgreSQL, установленным на сервере. Он идентифицируется с помощью логина и пароля.



После соединения он может выполнять все классические операции:

- администрирование базы данных;
- доступ к схемам базы данных;
- осуществление запросов SQL;
- управлять табличными пространствами (Tablespace);
- экспортировать данные.

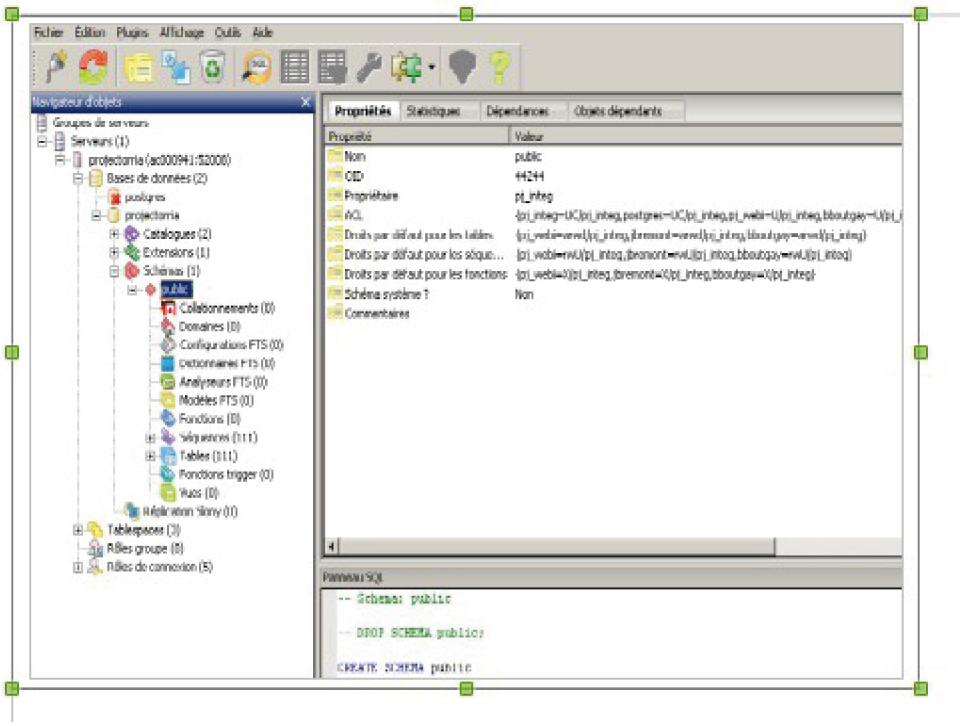
К сожалению, на сегодняшний день это приложение теряет скорость.

5.1.2 — PgAdminIII

PgAdminIII — это приложение для администрирования в варианте клиент-сервер. Речь идёт о свободном программном обеспечении, распространяемом по лицензии PostgreSQL. Это приложение может использоваться на всех платформах. При установке PostgreSQL на Windows оно входит в базовый установочный комплект. Оно так же доступно по следующему адресу:
http://www.pgadmin.org/download/source.php

Графический интерфейс делает его легким в использовании. Среди других достоинств можно отметить графический способ реализации запросов, персонализацию сообщений, добавление плагинов (например, PostGIS Shapefile и DBF loader) и присутствие скрипта.

Это программное обеспечение регулярно обновляется.



5.1.3 — psql

Инструмент psql позволяет выполнять администрирование в командной строке, ввод запросов SQL, визуализацию схемы базы, а также импорт и экспорт.

```
cv_integ@192.168.29.200's password:
Last login: Thu Jul 17 10:23:06 2014 from 10.100.70.69
AC000941-cv_integ> psql coopps
psql (9.1.10)
Saisissez Â« help Â» pour l'aide.

coopps=>
```

Для более подробной информации наберите «\help» в командной строке psql.

5.2 — Инструменты мониторинга PostgreSQL и анализа логов

Инструменты мониторинга (Nagios, Munin,...) предлагают дополнительные модули (плагины) для PostgreSQL, которые позволяют осуществлять мониторинг логов и таблиц системы. Скрипт perl check_postgres от Bucardo является основным плагином для этих инструментов. Он используется в MAE (Министерстве иностранных дел).
Другие модули позволяют отслеживать статистику изменений в базе (pg_stat_statements, pgtop,...) или анализировать логов (pgbadger). Последний модуль используется в MAE.

Отметим, что многочие модули (pgfouine, pgstatpack,…) перестают поддерживаться (обновляться) с течением времени.

5.3— Инструменты переноса данных в PostgreSQL

Инструмент Ora2pg позволяет анализировать базу данных на Oracle. Расширение Ora2pg DGFiP (министерства экономики, финансов и промышленности) позволяет оценить затраты на миграцию. Оно также позволяет реализовать переход.

Для переноса данных, использование инструмента ETL (Talend Open Studio,...) позволяет реализовать трансформацию данных в момент перевода (адаптацию типов данных,...).

Настоятельно рекомендуется проводить миграцию в условиях, наиболее близких к рабочим (рабочая база, логи для сессий приложений,...).

6 — Управление изменениями

6.1— Обучение сотрудников

Обучение будет касаться трёх типов сотрудников:

- разработчики: введение в курс дела и переход к PostgreSQL (2-3 дня);
- пользователи и архитекторы: установка, техническое обслуживание, резервное копирование, мониторинг,... (3 дня);
- администраторы базы данных (АБД): то же обучение, что для пользователей, + 3 дня.

6.2— Поддержка

Поддержка обеспечивается несколькими путями:

- межминистерским сообществом во главе с министерством внутренних дел;
- опираясь на межминистерскую сеть (MimPROD,...) ;
- сообществом PostgreSQL.

Профессиональная поддержка на французском языке также возможна по интернету:
http://www.postgresql.org/support/professional_support/europe/

6.3— План миграции

Переход требует более или менее значительной адаптации приложений, для которых должна быть соблюдена процедура, гарантирующая отсутствие регрессии. Это требует больших затрат.

Эти затраты могут быть минимизированы, если переход осуществляется по случаю значительных функциональных изменений, которые сами по себе требуют проведения технических и функциональных процедур вне зависимости от смены СУБД.

Должна быть запланирована поддержка руководителей проектов и сотрудников для повышения их квалификации в наилучших условиях.

7 — Снова о расходах

7.1— Затраты на миграцию базы

Переход к новой СУБД должен учитывать:

- начальные расходы (обучение сотрудников, повышение квалификации...);
- адаптацию приложений, включая стоимость процедур (технических, функциональных, регриссионных тестов).

7.2 — Стоимость владения

PostgreSQL охраняется лицензией open source, подобной лицензиям BSD или MIT. Таким образом, ценовая политика со стороны производителя отсутствует.

7.3 — Мастерство управления

PostgreSQL — это продукт с известной «дорожной картой» (roadmap). Экосистема PostgreSQL постоянно обогащается новыми дополнительными модулями и инструментами. Коммерческие программные продукты предлагают всё больше и больше интерфейсов, необходимых для использования PostgreSQL.

PostgreSQL обладает широким и очень активным сообществом: список рассылки http://www.postgresql.org/list/pgsql-fr-generale/
и форум http://forum.postgresql.fr/

8 — Приложения

8.1— На что обратить внимания во время перехода с некоторых СУБД

8.1.1 — Oracle

Вклад министерства внутренних дел:

Нужно обратить внимание на характеристики серверов-источников, чтобы определить сервера-цели, адекватные в том, что касается технического окружения, программного обеспечения, систем хранения, данных, критичности приложения и сохранения быстродействия.

8.1.1.1 — Техническое окружение

Характеристики серверов Oracle:
Проверить тип серверов: выделенный, общий или виртуализированный?
Выяснить следующие характеристики:

- типы процессоров;
- число процессоров;
- тактовую частоту процессоров;
- размер регистров;
- размер RAM.

8.1.1.2 – Программные стеки
Выяснить, какие программные стеки входит в состав приложения.
Действующая операционная система: для какого дистрибутива LINUX и какой версии?
Версия Oracle: к какой версии Postgres ?
Какие сервера приложений-источников ?
Тип JVM (Java Virtual Machine).
Система виртуализации.

8.1.1.3 — Система хранения данных
Какая система хранения? Обычно СУБД работают на SAN (Storage Area Network); но существет много и других решений.
Какой способ доступа? Наиболее часто VMDK (Virtual Machine Disk).

8.1.1.4 — Миграция базы данных
Инструменты перехода
Они многочислены : Ora2Pg, ETL или ELT, специально разработанные программы и т.д.

Перед загрузкой данных, обработайте сначала метаданные.
ПЕРЕНОС СТРУКТУР ORACLE К POSTGRESQL: под этим понимаются структуры таблиц и представлений с их приложениями, различные процедуры, функции и триггеры.

Затем переходите к проверке вероятного присутствия секционирования и материализированных представлений.

Секционирование — начиная с PostgreSQL 9.1, изначально оно было обходным путём, но начиная с версии 9.2 оно представляет собой полноценную функциональность.

Материализированные представления
С версии 9.0 до 9.2, их использование требовало изобретения приёмов (уловок), но начиная с версии 9.3, это встроенная функциональность.

Язык процедур SQL
Oracle использует PL/SQL, а PostgreSQL - PL/PGSQL; они достаточно похожи, но всё же адаптация потребуется.

Модель соответствия типов данных
Строковые переменные могут быть заменены на VARCHAR и TEXT.
Даты Oracle в 7 битах — на TIMESTAMP в 8 битах.
Длинные строковые переменные CLOB заменяются на TEXT.
Целые численные величины NUMBER в зависимости от точности (по восходящей) — на SMALLINT, INTEGER, BIGINT, NUMERIC.
Десятичные числовые значения — на NUMERIC.
Двоичные данные BLOB — на BYTEA (внимательно с экранированием символов с Ora2Pg).

Миграция обычных и хранимых процедур
Соберите все процедуры и функции PL/SQL и трансформируйте их в PL/PgSQL.

Генерация СКРИПТОВ МИГРАЦИИ БАЗЫ ДАННЫХ
Скрипт миграции структур должен быть отделён от данных; он будет служить для инициализации базы PostgreSQL.

Создание скрипта инициализации базы PostgreSQL
Он создается на основе структур данных, процедур и функций, адаптированных для PostgreSQL.

Перенос данных
Перенос может быть осуществлён с помощью SQL-скрипта миграции данных, которые должны быть помещены в новую базу PostgreSQL.
Так же возможно рассчитывать на прямую замену базы Oracle на базу PostgreSQL.

Код приложения
Изменение кода необходимо, чтобы интегрировать драйвер PostgreSQL для управления транзакциями, открытием и закрытием соединений, ключевых слов Oracle, которые нужно заменить ключевыми словами PostgreSQL, внешними соединениями, нумерацией страниц и т.д.

8.1.1.5 — Критичность резервного копирования
Принятие во внимания понятий PRA/PCA, RTO, RPO и REPLICATIONS.

Какое техническое решение использовано для обеспечения высокой доступности: Oracle RAC, Oracle DATAGUARD или другое ?

В PostgreSQL не существует эквивалента Oracle RAC; зато функция DATAGUARD обеспечена в PostgreSQL репликацией.

План Возобновления Работы предписывает задержку возобновления работы приложения. Он берёт в расчёт RTO (Recovery Time Objective или максимально допустимую длительность простоя) и RPO (Recovery Point Objective или допустимый объём возможных потерь данных в случае сбоя). Это обсуловливает также, в зависимости от объёма, технологию, частоту и тип резервного копирования, которые нужно осуществлять.

В Oracle резервным копированием и восстановлениями управляет RMAN; в PostgreSQL эквивалента не существует. Однако другой инструмент, называемый Barman, находится в процессе установки в MAE (министерстве иностранных дел), чтобы автоматизировать резервное копирование и восстановление. В то же время, вариант Archivelog поддерживается обеими СУБД практически одинаково.

Физическое восстановление не имеет более тонкой грануляции в PostgreSQL (восстанавливается всё).

8.1.1.6 — Сопровождение, повышение быстродействия и мониторинг
Аналог Grid Control, существуещего под Oracle, в PostgreSQL отсутствует. Однако возможно использовать Nagios с плагином check_postgres. Доступны и другие инструменты (powa, PGObserver, …).

Необходимо подумать об использовании и других доступных инструментов.

8.1.2 — DB2

Переход приводит к необходимости пересмотра DDL из db2: рекомендуется иметь инструмент трансформации (скрипт) из ddl, принимающий в расчёт детали, которые будут описаны ниже. Что касается DCL, предпочтительно не пытаться переносить из db2 в pg привелегии, определённые в db2, из-за слишком разных уровней авторизации. Не забудьте изменить процедуры технического обслуживания: резервного копирования, историзации, дефрагментации базы, сбора статистики, чистки устаревших файлов и т.д.

8.1.2.1 — Некоторые особенности DDL pg/db2

• Строковые переменные:

- в pg, строковые переменные определены в числе символов.
- В db2 строковые переменные, фиксированные или переменной длины, выражены в байтах с числом возможных символов, которое может быть различным в зависимости от кодировки базы: в utf-8 некоторые символы кодируются на нескольких байтах, в iso8859-15 каждый символ вмещается в один байт. В varchar(5) может содержаться меньше 5 символов в db2, если база в кодировке utf-8 и зона содержит символы с акцентами (для французского языка).

Это необходимо учитывать, если нужно перевести базу db2 utf-8 в базу pg.

- Создание таблицы в табличном пространстве:

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

- Создание таблицы с помощью ключевого слова LIKE :

В pg использование круглых скобочек () вокруг LIKE обязательно, в db2 в них нет необходимости.

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

- Значение по умолчанию в столбце таблицы:

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

- Создание таблицы:

Некоторые ключевые слова не признаются в pg, следовательно, их нужно убрать из ddl DB2 перед переходом. Например, *append*, *with restrict on drop*, *long in*, вся часть *table-*

partionning, которые в pg определяются иначе.

- Автоматическое приращение счётчика столбцов при создании таблицы:

Автоматическое приращение DB2 (*[generated always* | *generated by default] as identity*) не признаётся в pg, нужно использовать исключительно последовательности.

- Использование типа serial в pg позволяет легко создавать последовательность, но не нужно использовать ни тип integer (уже включённый в serial), ни not null, дабы избежать синтаксических ошибок.

В DB2 создание последовательности возможно только с помощью create sequence.
Следовательно, все автоинкременты в db2 нужно конвертировать.

- Таблица *not logged* :

Таблица в db2, задекларированная как *not logged initially* должна быть *unlogged* в pg.

- Временные таблицы

Синтаксис и функциональности временных таблиц отличаются.
Пример:
db2: declare global temporary table tabtemp like eqos.stats not logged
pg: create local temporary table tabtemp (like eqos.stats) unlogged (?)

Временные таблицы в db2 с общими данными между сессиями создаются с помощью *create global temporary table,* они не имеют эквивалента в pg, так как, хотя синтаксически ключевые слова *local* и *global* признаются в pg, но поведение временной таблицы остаётся локальным в обоих случаях (обмена временными данными нет).

- Столбец Timestamp при обновлении строки:

Эти столбцы, часто используемые в db2, не существуют в pg.
Пример в db2: столбец с именем TS_UPDATE автоматические обновляется, как только в строке сделаны изменения с помощью UPDATE/INSERT sql:

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

В случае вставки строки или обновления строки, db2 автоматически оповещает столбец ts_update. Эти столбцы систематически показываются в некоторых базах db2, **в pg это действие нужно будет делать дополнительно, то есть, как следствие, внести изменения в программы.**

- Формат timestamp ISO :

Не только точность timestamp отличается, но и разделитель между датой и временем также отличается. **Будьте внимательны при переводе базы данных из db2 к pg!**

pg	db2
2014-01-31 00:00:00	2014-01-31-00.00.00.000000

Заметим, что формат *NOT NULL DEFAULT CURRENT TIMESTAMP,* поддерживаемый DB2, не принимается в pg; нужно использовать *CURRENT_TIMESTAMP* (поддерживаемый также и в DB2).

- Материализованное представление :
  - pg : создаётся с помощью *CREATE MATERIALIZED VIEW* .
  - db2 : создаётся с помощью *create table* со специфическими опциями для материализованного представления (называемого MQT в db2). **Следовательно, ddl из db2 нуждается в исправлении.**

- Drop table и integrity constraints:
- pg : *drop table... CASCADE* уничтожает цintegrity constraints.
- db2 : *drop table* достаточно, чтобы уничтожить integrity constraints.

- Создание индекса, схемы :
  - pg : имя схемы не должно стоять перед именем индекса, иначе получим сообщение о синтаксической ошибке;
  - db2 : рекомендуется ставить имя схемы, и инструмент генрации ddl в db2 генерирует это имя схемы перед именем индекса. По умолчанию будет использовано то же имя, что у текущей схемы, или имя владельца соединения.

- Создание индекса, опции:

Некоторые ключевые слова не признаются в pg, следовательно, они должны быть удалены из ddl db2 перед переходом. Например, cluster, allow reverse scan, pctfree...

- Присваивание индексов соответствующему табличному пространству:
- pg: при создании таблицы, присваивание осуществляется в опции создания первоначального ключа или единства. Также можно присвоить индекс табличному пространству во время создания индекса.
- db2: можно прямо направить все индексы, связанные с таблицей, в табличное пространство с помощью опции *INDEX IN, например,*

create table.. IN <nom_ts_table> INDEX IN <nom_ts_index>.

Или во время создания индекса, как в pg.

- Размер страниц :
- pg: разрешены страницы только по 8 КБ.
- db2: работает со страницами размеров 4, 8, 16 или 32 КБ. **Нужно изменить ddl db2 так, чтобы размер страниц был чётко указан.**

- Bufferpool :

Понятие bufferpool, существуещее в db2, не существует в pg. Следовательно, такие ресурсы создавать не будем, а ddl db2 должен быть адаптирован, из него должны быть удалены все инструкции создания, а также ссылки на bufferpool в табличном пространстве.

- Табличные пространства (Tablespaces):

Опции создания табличных пространств различны в db2 и pg, требуются исправления.

8.1.2.2 — DCL

- Роли по отношению к группам в Unix:
  - pg : обязательно требуется участие в привилегиях, выданных ролям.
  - db2 : привилегии выдаются либо ролям, либо непосредственно группам Unix, что является наиболее частым случаем в MEN.

- Права PUBLIC :

Понятие RESTRICTIVE для базы, которое существует в db2 и уничтожает привилегии PUBLIC, не существует в pg.

- GRANT ALTER TABLE:

Используемое в db2, это понятие не существует как таковое в pg. В pg нужно, чтобы пользователь, желающий изменить таблицу, принадлежал роли, которая является OWNER таблицы.

- TRUNCATE TABLE:

- pg: чтобы иметь право на осуществление truncate, нужно сделать *grant truncate.*
- db2: эту привилегию даёт *grant delete* (или *conrol*).

- GRANT CONNECT:

Чтобы получить разрешение соединиться с базой, нужно сделать следующее:

- pg: *grant connect on database <nom_base>…*
- db2: *grant connect on database.* Не нужно уточнять имя базы, так как эта привилегия включает соединение с базой, то есть это текущая база. Если уточним имя базы, получим сообщение о синтаксической ошибке.

Могут появиться и другие отличия в синтаксисе.

- Привилегии без эквивалентов в db2/pg :

Часть привилегий в DB2, которые связаны с функциональностями, несуществующими или отличными в pg, не могут быть перенесены в pg. Это, например, случай *grant load*, *grant use of tablespace*, *grant usage on workflow* и др.

- Системный каталог

Информация хранится в виде прописных (больших) букв в DB2 и строчных (маленьких) в pg. Это нужно учитывать при поиске информации и, особенно, в скриптах, которые используют каталог, чтобы собирать там информацию.

Например: нужно найти информацию в таблицах, принадлежащих схеме с именем IDENTITE
pg : select * from pg_tables where schemaname = '**identite**'
db2 : select * from syscat.tables where tabschema = '**IDENTITE**'

8.1.2.3 — Прочие замечания

- Утилита загрузки:
  - db2: таблица в db2 может быть загружена с помощью *import, load, ingest* или

db2move.

- pg использует исключительно утилиту *copy*, которая гораздо менее богата возможностями, чем утилиты db2. Кажется, что с помощью *copy* невозможно загрузить таблицу частично, как в db2.

- Другие утилиты db2/pg:

reorg становится *vacuum*
runstats становится *analyse*
backup становится *pg_dump, pg_dumpall ou pg_basebackup*

- Двойные кавычки ":

Во время перехода обратите внимание на двойные кавычки: в db2 при экспорте они являются ограничителем по умолчанию и, следовательно, находятся в загруженном файле (export), обрамляя каждую строковую величину. При загрузке к pg эти двойные кавычки оказываются по умолчанию в таблице, даже если они присутствуют в файле с данными. Чтобы избежать этой нежелаемой загрузки, файлы, экспортируемые с db2, должны быть в моде *del* (ascii) и импортируемые в pg, как csv с помощью опции *WITH CSV delimiter ',' QUOTE '"'* в *copy.*

- Журнал транзакций:
  - db2 : каждая база обладает своими собственными журналами.
  - pg: это не так, журналы являются общими для нескольких баз во время одного процесса, что может вызвать проблемы (журналы, создаваемые одной базой, влияние многих баз в случае ошибки или исчезновения журналов...)

- LOBs:
  - BLOB db2 должны быть заменены столбцами в формате BYTEA;
  - CLOB db2 должны быть заменены форматом TEXT;
  - в отличие от DB2, который позволяет загрузить LOBs во время резервного копирования (backup), pg не сохраняет LOBs с помощью pg_dumpall (pg сохраняет LOBs с помощью pg_dump).
  - les options db2, связанные с LOBs, более не применяются в pg (*logged/not logged*, *compact/not compact*, *inline length*...).

- Сохранённые процедуры :

Написанные в db2 в SQL/PL, они должны быть адаптированы к pg (PL/PgSQL).

- Исходный код (programmes) :
  - изменить методы вызова драйвера и использования его свойств.
  - изменить код SQL, который может иметь свои особенности в db2/pg.

8.1.3 — Informix

Вклад министерства социальных дел

8.1.3.1 — Структура

Схема баз данных (таблицы, индексы, принуждения...).

Для создание различных баз данных в PostgreSQL понадобятся скрипты.

Все следующие создаваемые объекты должны быть созданы в соответствии с синтаксисом **PostgreSQL 9.1** и существующими нормами:

- таблицы,
- представления,
- триггеры,
- constraint'ы,
- индексы

Следующая таблица описывает некоторые различия для объектов СУБД Informix и PostgreSQL :

Объект / СУБД	Informix	PostgreSQL
Тип данных	BLOB	BYTEA
	DATETIME	TIMESTAMP
constraints (внешний ключ)	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);
Индекс	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 — План действий во время миграции баз данных

Во время миграции должны быть выполнены следующие этапы:

i. В исходной базе данных (Informix) уничтожение хранимых процедур, которые не используются приложениями java, для того, чтобы избежать переноса к PostgreSQL процедур, которые никогда не будут выполнены.

ii. Создание баз данных:
   a) Создание таблиц,
   b) Создание триггеров,
   c) Создание хранимых процедур

iii. Загрузка данных.

iv. Создание индексов и constraint'ов. Индексы и constraint'ы будут созданы после загрузки данных, чтобы оптимизировать длительность процедуры миграции.

8.1.3.3 — Интеграция фреймворка Hibernate

Если никакой фреймворк не используется в приложении, чтобы управлять хранением объектов в реляционной базе данных, решением могло бы служить использование фреймворка Hibernate.

Это решение позволило бы не писать запросы SQL в исходном коде Java, а генерировать их с помощью Hibernate.

Приложения были бы, следовательно, менее зависимы от СУБД (кроме хранимых процедур).

Приложения, которые используют фреймворк Hibernate, управляющий хранением объектов в реляционных базах данных. Запросы SQL не пишутся на языке Java, в противоположность приложениям без Hibernate, а генерируются с помощью Hibernate.

В этом случае переход от Informix к PostgreSQL будет иметь ограниченное воздействие на исходные коды этих приложений.

Но в любом случае регрессионные тесты будут необходимы.

8.1.3.4 — Риски
Эти риски должны быть проанализированы:

- Прерывание использования приложений во время фазы загрузки данных баз Informix к базам PostgreSQL.
- Решением, позволяющим уменьшить длительность проведения перехода, состояло бы в том, чтобы загрузить сначала статические данные (как таблицы справочников н), а во вторую очередь загружать только данные, которые могут меняться (запросы, досье и т.п.).
- Производительность в день перехода. Выбор дня - это решение было бы подходящим в случае, когда длительность загрузки ожидается слишком долгой и если выигрыш во времени большой (это зависит от количества данных, могущих быть рассмотренными как статические данные).
- Замедление некоторых обработок, которые были оптимизированы под СУБД Informix IDS.

8.1.4 — MS SQL
Несколько замечаний:
Инструменты, могущие облегчить перенос данных

- https://github.com/dalibo/sqlserver2pgsql
- http://pgloader.io/howto/quickstart.html (более новый продукт)

Что касается переноса процедур: код очень разный и должен быть переписан.

8.2- Несколько ссылок

8.2.1 — Несколько сайтов, использующих PostgreSQL
Meteo France (http://www.postgresql.fr/temoignages:meteo_france)
Объём данных : 3.5 ТБ

Le Bon Coin (http://www.postgresql.fr/temoignages:le_bon_coin)
Объём данных : > 6 ТБ

IGN (http://www.postgresql.fr/temoignages:ign)
Возможность обрабатывать более 100 миллионов геометрических объектов ;

Mappy (http://www.oslandia.com/oslandia-et-mappy-vers-lopensource.html)
75 ГБ для базы PostGIS

8.2.2 — Несколько ссылок на MINEFI-MEDDE (Министерство экологии, длительного развития и энергетики)

MEDDE использует PostgreSQL в течение 10 лет. На сегодняшний день более 250 приложений работают на PostgreSQL.

Право на землевладение (ADS)

Объём данных : 95 ГБ

Разрешение на строительство (Sitadel)

Объём данных : 60 ГБ

Управление информацией об автомобильном движении в районе Ile de France (Sytadin)

Объём данных : 3 ГБ

8.2.3 — Несколько ссылок на MINEFI-DGFiP (Министерство экономики, финансов и промышленности)

FIBANC

Приложение позволяет B.R.S (DNEF) использовать своё право на связь в исполнение
закона BOI 13 K-2-88.

Объём данных : 2 ГБ

PROCOLMAS

Приложение коллективных процедур. Это приложение предназначено для получения и хранения коллективных процедур, вышедших из BODACC-A и их использование SI
Copernic.

Объём данных : 2 ГБ

PATRIM Usagers

Приложение позволяет частным лицам и профессионалам новую услугу поиска и сравнения величин земельной ренты их недвижимости благодаря порталу пользователей и порталу профессионалов.

Базы офф-лайн для обработки batchs : 330 ГБ в целом, в том числе одна база 280 ГБ.

Базы он-лайн (оптимизированные для запросов в моде «только чтение» типа datamart): 70 ГБ в целом, в том числе одна база 30 ГБ.

Массивное использование расширения postgis.

PATRIM Colloc

Автоматизированная обработка запросов географической информации Patrimoine
Immobilier.

База batch 13 ГБ

TREVI

Декларация в удалённом доступе профессионального недвижимого имущества в рамках ревизии величин земельной ренты. Приложение ориентировано на профессиональное использование.

База 60 ГБ

FNTD

Национальный файл доверителей (проверка задекларированных обязательств).

База 20 ГБ

8.2.4 — Несколько ссылок на MINEFI-DGDDI (Генеральную дирекцию таможни)

BANACO

- Внутреннее приложение, которое предоставляет инструменты, позволяющие собирать необходимую информацию о политике таможенного контроля.
- База 193 ГБ

EDDI

- Отчёты ежедневные, 10-дневные и ежемесячные. Существует два способа доступа к отчётам: с помощью приложения Aladin для таможенных служб и с помощью Prodouane для операторов.
- 130 пользователей-таможенников
- 2500 пользователей-операторов
- 22 ГБ

TSAR

- Удалённая услуга (Обработка информации и поддержка пользователей в сети), позволяющая поддержку пользователей информационной таможенной системы.
- Общее число пользователей: 60000, максимальное число одновременных пользователей: 100
- 242 МГ для цели 12 ГБ

8.2.5 — Несколько ссылок на MENESR (DNE) (Министерство национального образования, высшей школы и научных исследований)

DEMACT

- Приложение позволяет управлять административными и финансовыми делами в колледжах и лицеях.
- Для академии и выборки из 20 учреждений : 150 МБ

Сервер исполнения бизнес-правил (BRMS)

- Объём данных: 120 ГБ

GED SIRHEN

- Для численности 4000 служащих : 10 ГБ

8.2.6 — Несколько ссылок на MAE (Министерство иностранных дел)

Обучение во Франции

- Ожидаемый объём 1 ТБ

Межминистерский портал дипломатической корреспонденции (Diplomatie)

- Объём данных: проектирование: 20 ГБ ежегодного увеличения объёма

8.3 — Расширения и плагины для PostgreSQL

В таблице ниже собраны некоторые плагины, о которых говорится в этом документе:

PostGIS	Географические объекты
pgstatpack	Статистика
pgCrypto	Криптография

8.4— Прочие инструменты для PostgreSQL

В таблице ниже представлены прочие инструменты для PostgreSQL:

barman	Резервное копирование/ восстановление
pgtop	Статистика
pgfouine	Анализ логов
pgbadger	Анализ логов
pgPool	Управление пулами соединений

9 — Справочные документы

Технические заметки Dalibo (эти документы доступны клиентам Dalibo)

- Обзор о резервном копировании, восстановлении и переходе PostgreSQL ;
- Обзор об установке PCA/PRA в PostgreSQL ;
- Масштабируемость и производительность с PostgreSQL.

Документация PostgreSQL: http://docs.postgresql.fr/9.3/