

# SERVICIOS DE ALTA DISPONIBILIDAD DE BASES DE DATOS CON POSTGRESQL

*Ernesto Quiñones A.*  
*ernestoq@apesol.org*

<http://www.apesol.org.pe>

 **APESOL**  
Asociación Peruana de Software Libre

# Conozcamos PostgreSQL

- Proyecto con mas de 15 años de vida.
- Se inicia en la Universidad de Berkeley en 1977 bajo el nombre Ingres como un ejercicio de aplicación de las teorías de las RDBMS.
- 1986, cambia su nombre a Postgres (nombre popular con el que se le conoce hasta hoy) con el objetivo de aplicar los conceptos de Objetos Relacionales.
- 1995, cambia su nombre a Postgres95 que luego derivaría a PostgreSQL
- 1996, el proyecto se integra al mundo del Open Source inicia en la versión 6.0
- 2000, se comienza a implementar el soporte de Ipv6
- 2004, PostgreSQL 8.0, adopción en el mundo comercial, se le califico como la 5ta DBMS mas popular en USA.
- 2005 Julio, PostgreSQL paso el test de Coverty Inspected encontrando solo 20 errores en 775,000 lineas de código.

# Conozcamos PostgreSQL



2000-2003-2004-2005  
Best Database



2004 Best Server  
Application Award



1999-2002-2004  
Best Database

# *Características de PostgreSQL*

- Corre en casi todos los principales sistemas operativos : Linux, Unix, Solaris, BSDs, Mac OS, Beos, Windows, etc. (34 OS soportados)
- Full ACID compliant
- Soporte de todas las características profesionales de una base de datos seria (objetos de uso común, orientación a objetos, eventos en los objetos de la base de datos, etc.).
- Soporte de ANSI SQL92 y 99
- Soporte para los lenguajes mas populares del medio : PHP, C, C++, Java, Perl, Python, Ruby, etc.
- Drivers : Odbc, Jdbc, .Net, etc.
- Soporte de protocolo de comunicación encriptado por SSL

# *Características de PostgreSQL*

- Proyectos de interfaces de administración WEB y por GUI convencional.
- Proyectos para datos geográficos/geométricos (PostGIS)
- Proyectos de uso de índices avanzados (OpenFTS)
- Proyectos para soportar diversos lenguajes de programación como lenguajes de funciones internas del motor (pl/Php, pl/Java, pl/Python, etc.)
- Licencia BSD, la mas permisiva de todas para los negocios
- Versiones comerciales derivadas del proyecto libre.
- Bajo TCO y alto ROI

# *Los Limites de PostgreSQL*

- Máximo de base de datos : ILIMITADO
- Máximo de tamaño de tabla : 32TB
- Máximo de tamaño de registro : 1.6TB
- Máximo de tamaño de campo : 1GB
- Máximo de registros por Tabla : ILIMITADO
- Máximo de campos por tabla : 250 a 1600 (depende de los tipos usados)
- Máximo de índices por tabla : ILIMITADO
- Número de lenguajes en los que se puede programar funciones : aproximadamente 10 (pl/pgsql, java, perl, python, tcl, php, C, C++, Ruby, etc.)
- Métodos de almacenamiento de índices : 4 (B-tree, R-tree, Hash y Gist)

# *Antes de crecer AFINAR*

- **Piensa bien tus índices, una tabla con demasiados índices demora en ABC de data.**
- **Utiliza Explain Analyze para analizar y optimizar tus queries.**
- **Hacer un Vacuum cada cierto tiempo es recomendable.**
- **La plataforma mas adecuada para trabajar con PostgreSQL es ..... Unix & BSD, motivo: mejor rendimiento en operaciones dinámicas de I/O.**
- **Siempre es bueno hacerle un tuning a tu OS, utiliza sistemas de archivos estables y robustos, con buena capacidad de recuperacin ante fallas.**

# Algunas Técnicas de optimización

- ¿Tipos de Datos grandes? : USA TOAST  
(The Oversized-Attribute Storage Technique).

## Archivo de 36k en esquema convencional

Página 1

8k

Página 2

8k

Página 3

8k

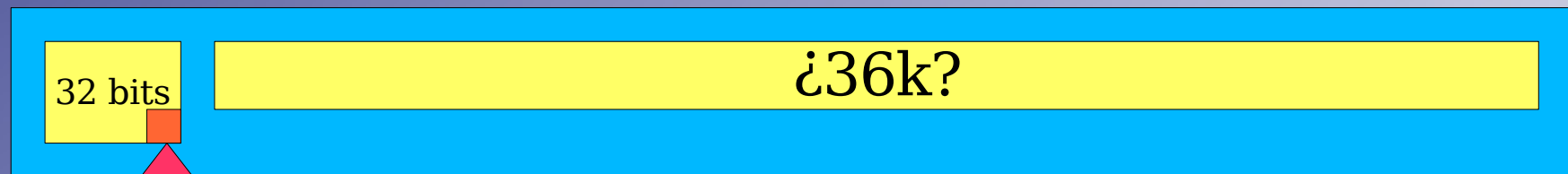
Página 4

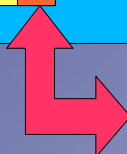
8k

Página 5

4k / 4k vacío

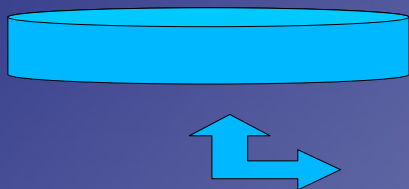
## Archivo de 36k usando TOAST



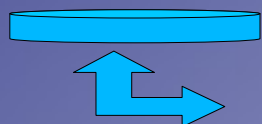
 Data comprimida

# Algunas Técnicas de optimización

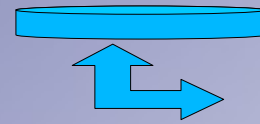
- **Compartir los índices en discos diferentes a los de la data es buena idea (tablespaces)**



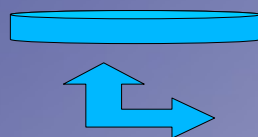
Un disco, una cabeza lee, ordena, pasa a cache y presenta la data



Disco normal con tablas maestras



Disco rápido con índices

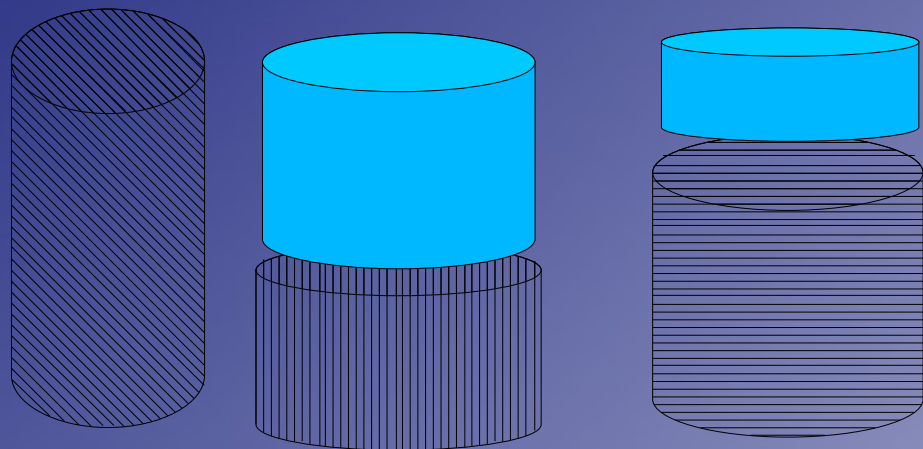


Disco ultra rápido con tablas de movimientos y principal

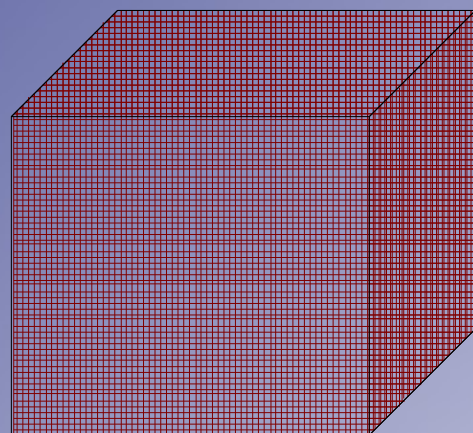
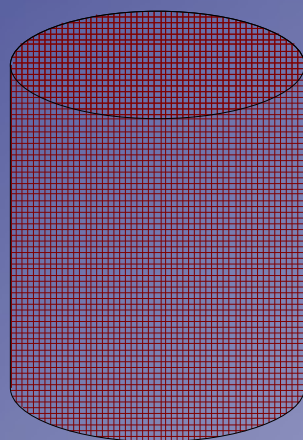
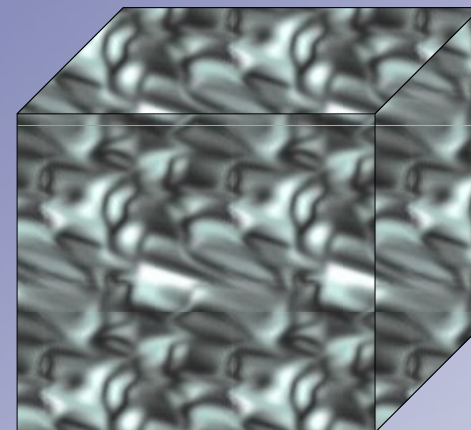
Un disco lee data, otro ordena y pasa a cache del primero

# Algunas Técnicas de optimización

- Tienes data estática, ¿usas índices? : USA  
Indices clusterizados



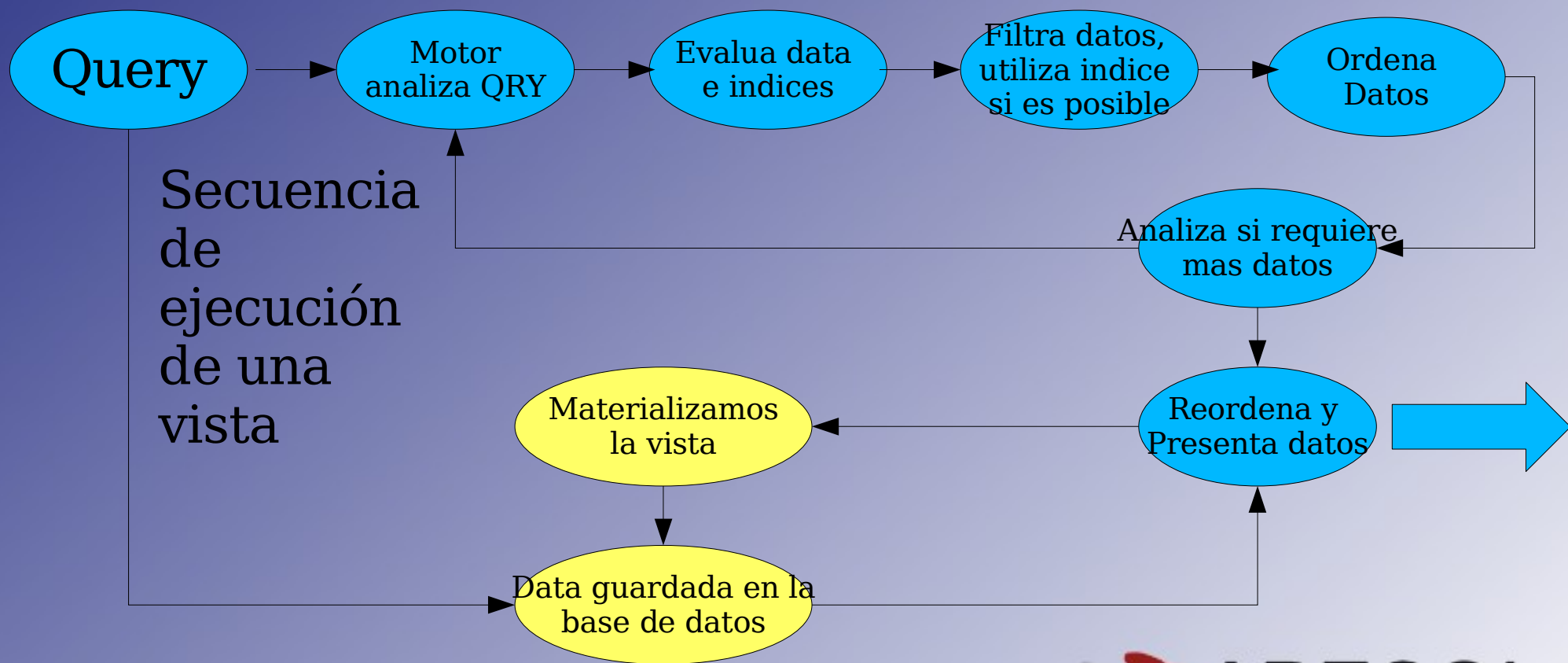
La data cambia, el índice cambia



Data estática,  
índice estático,  
mayor velocidad  
de acceso.

# Algunas Técnicas de optimización

- ¿Tienes data estática con consultas complejas y continuas? : USA VISTAS MATERIALIZADAS



# *Algunas Técnicas de optimización*

- **USA EL INDICE ADECUADO SEGUN TUS CONSULTAS**

- **HASH (=)**

- **BTREE (<, <=, =, >=, >)**

- **RTREE (data espacial)**

- << ... Is strictly left of?
- &< ... Does not extend to the right of?
- &> ... Does not extend to the left of?
- >> ... Is strictly right of?
- <<| ... Is strictly below?
- &<| ... Does not extend above?
- |&> ... Does not extend below?
- |>> ... Is strictly above?
- ~ ... Contains?
- @ ... Contained in or on?
- ~ = ... Same as?
- && ... Overlaps?

# *Analizando el problema*

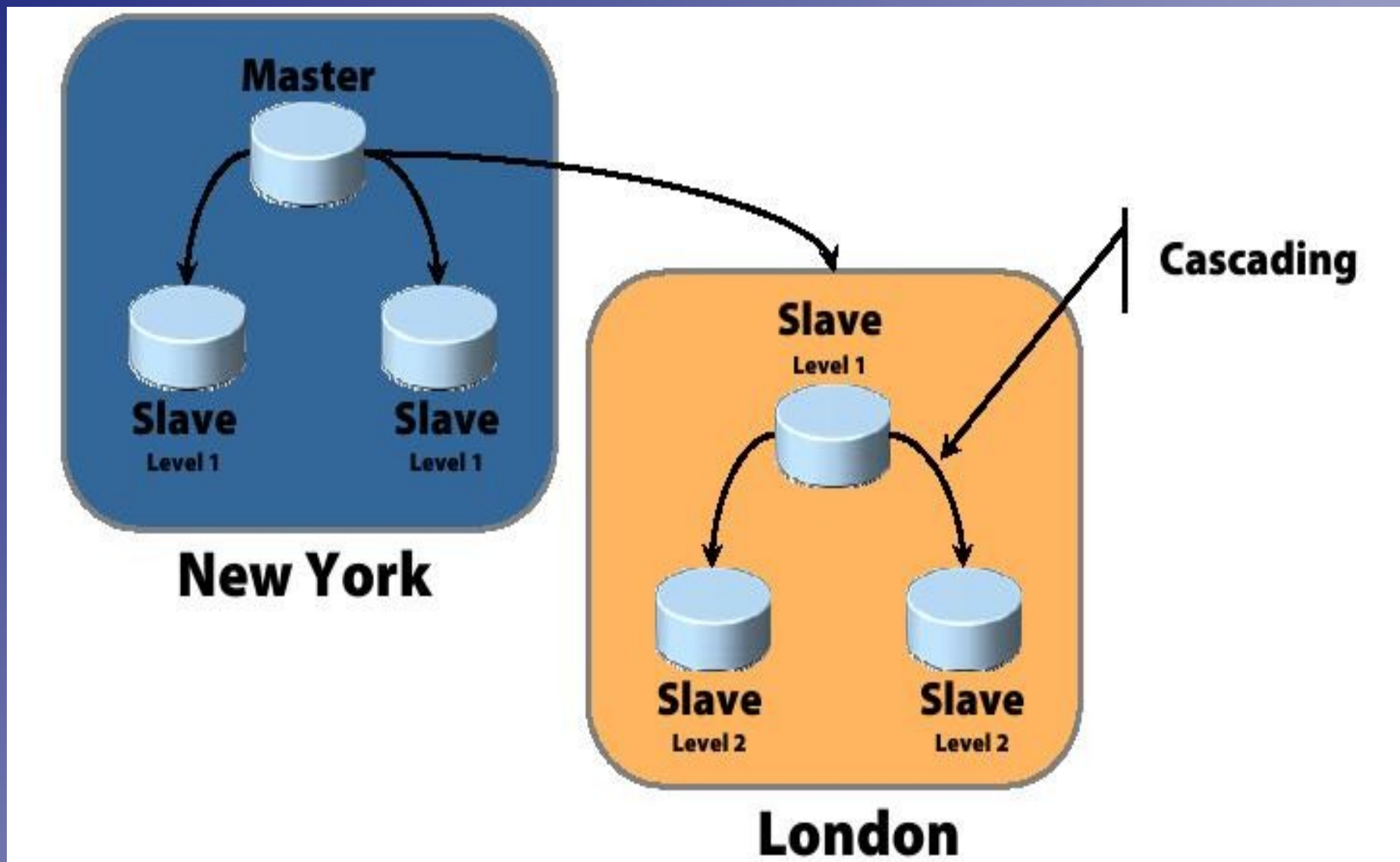
- **¿ Mi volumen de datos es muy grande ?**
- **¿ Tengo demasiada concurrencia de transacciones ?**
- **¿ Necesito un sistema tolerante a fallas ?**
- **¿ Necesito centralizar data de diversos lados ?**
- **¿ Necesito publicar data en diversos lados ?**

# *Replicando Datos*

## **Slony I**

- **<http://pgfoundry.org/projects/slony1/>**
- **Replicación en cascada Master & Slaves**
- **Asíncrono**
- **Tolerante a fallas, pero no es automático**
- **Basado en triggers**
- **Configuración sencilla**
- **En trabajo Slony II, disponible a inicios del próximo año**

# Replicando Datos



# *Mejorando Accesos a Data*

## **PgPool**

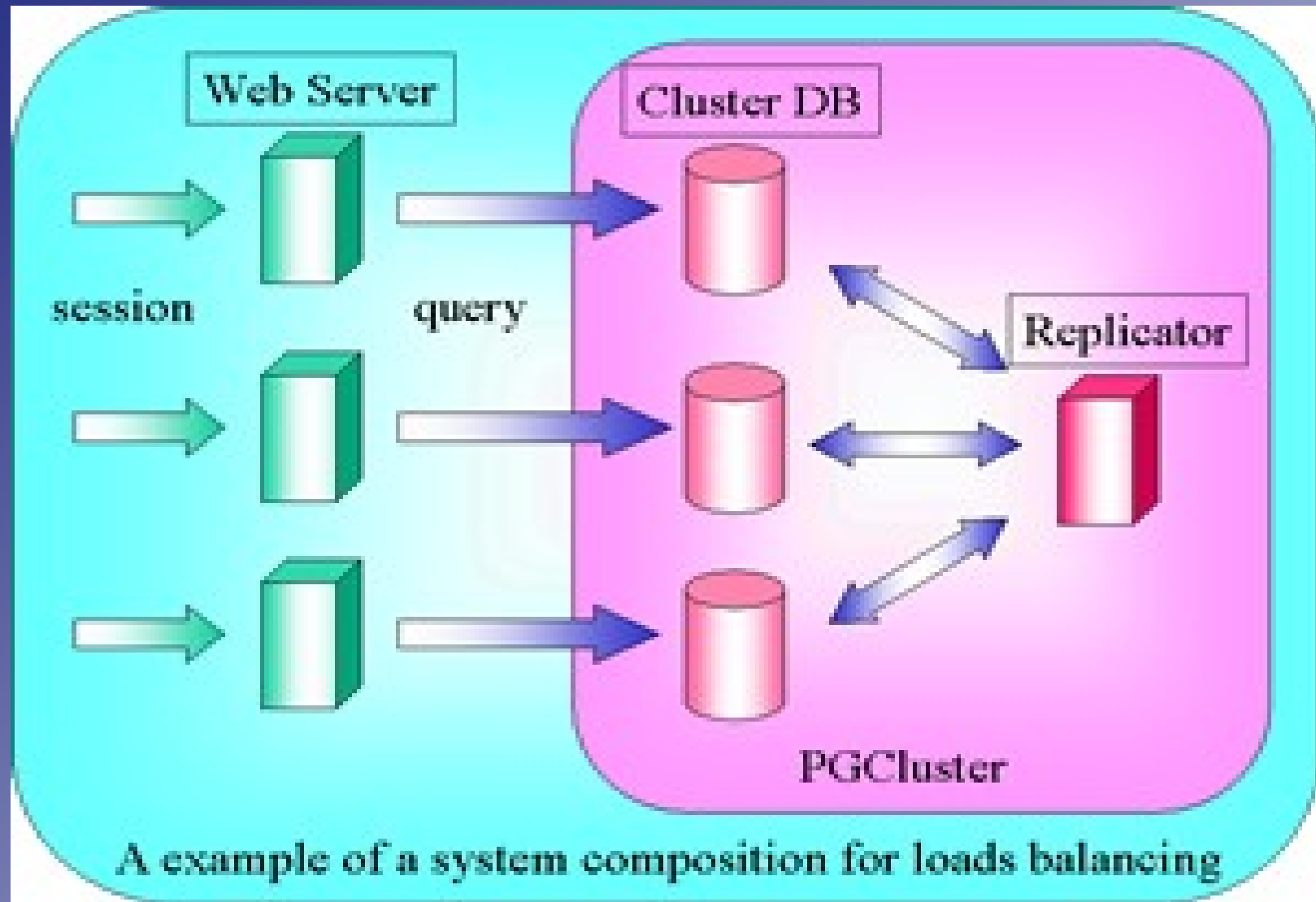
- <http://pgfoundry.org/projects/pgpool/>
- PgPool trabaja entre los servidores de datos y el cliente para reducir el overhead de establecimiento de conexión.
- Trabaja hasta con 2 servidores, si el principal cae automáticamente trabaja con el otro
- El complemento ideal de Slony para tolerancia a fallas.
- Incorpora un balanceador de carga
- Puede ser mas lento la conexión

# *Todo en UNO*

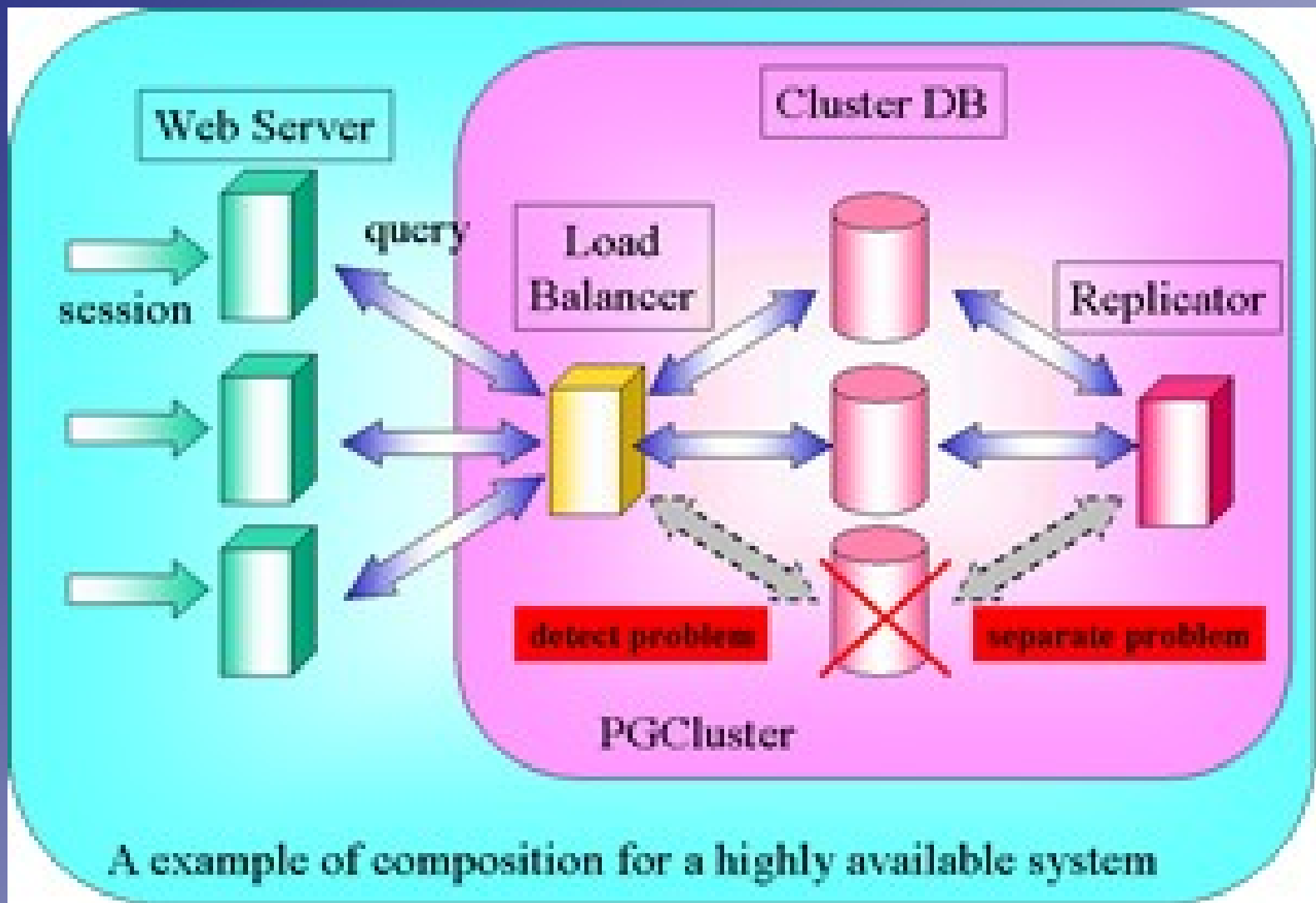
## **PgCluster**

- <http://pgcluster.projects.postgresql.org/>
- **Incorpora : Balanceo de Carga, Replicación de Datos, Tolerancia a fallos.**
- **Es síncrono, sincroniza el árbol de directorio del Pgsq.**
- **Cuando un servidor cae y se vuelve a poner en funcionamiento automáticamente inicia la sincronización de datos.**
- **Replicación de datos por querys**
- **Se replican también objetos adicionales a la data como las funciones definidas por el usuario.**
- **Funciona dentro de la misma base de datos.**

# Todo en UNO



# Todo en UNO



# *Otras herramientas de replicación*

- **ERServer :**

<http://gborg.postgresql.org/project/erserver/projectdisplay.php>

- **RServ :**

<http://gborg.postgresql.org/project/rservimp/projectdisplay.php>

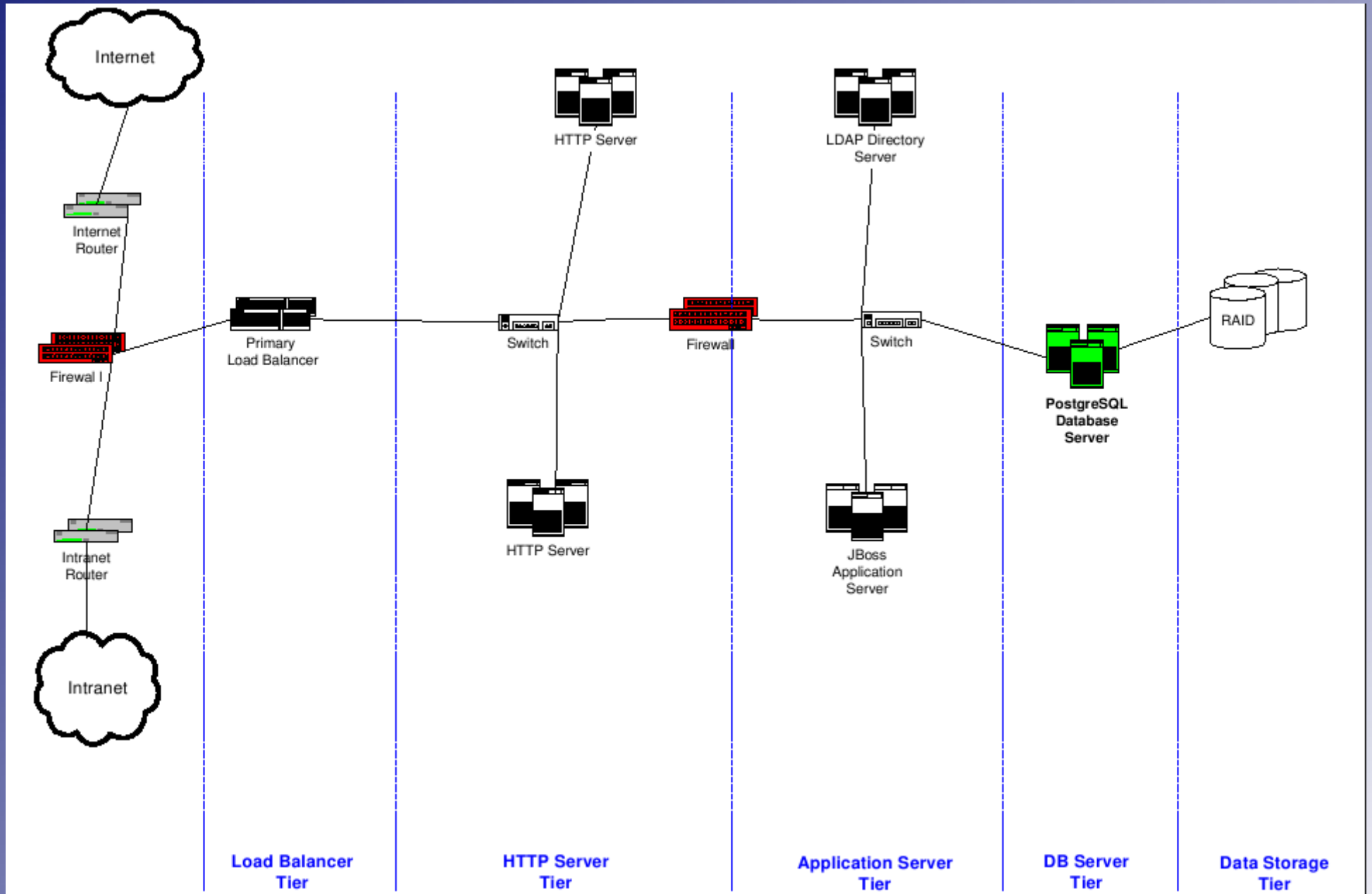
- **Pgreplication :**

<http://gborg.postgresql.org/project/pgreplication/projectdisplay.php>

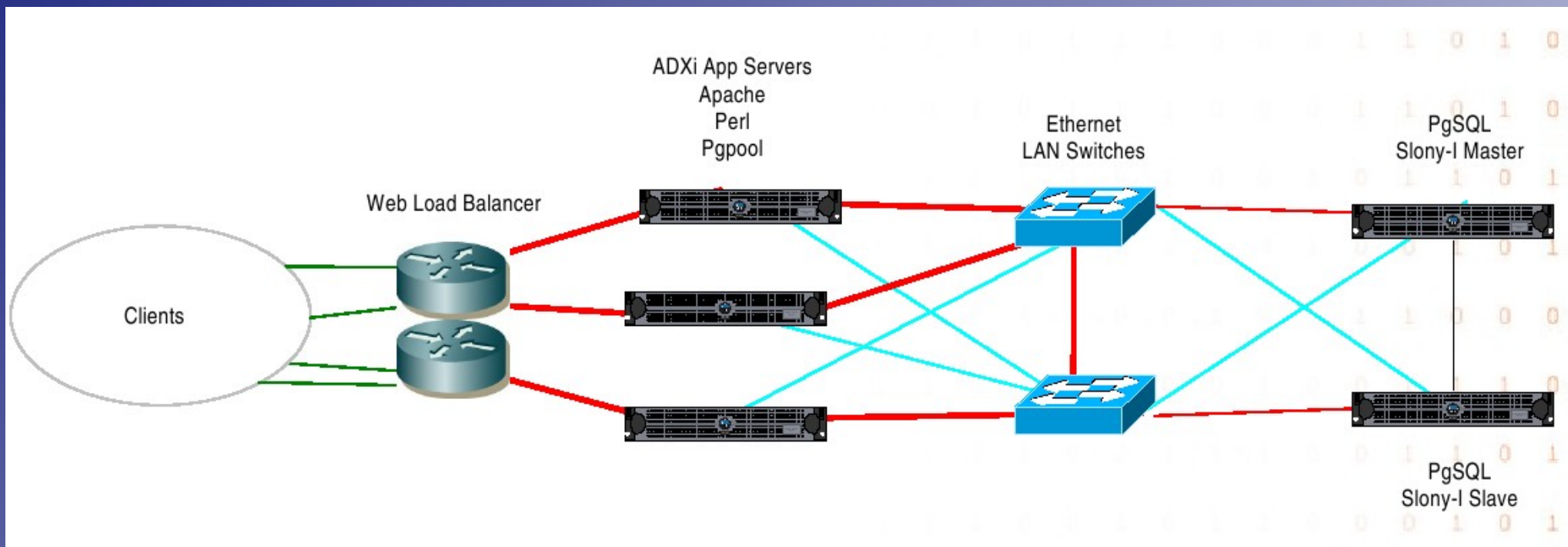
<http://gborg.postgresql.org>

<http://pgfoundry.org/>

# Caso de BASF



# eXo2.net Corp



This architecture has transferred :

- 522,000 Transactions
- 815,000 Files
- To 585,000 Receivers

# *Algunas otras características*

- **Probado con torres de discos duros de alta disponibilidad.**
- **Pg\_buffercache, permite ejecutar queries en cache de Pgsq**
- **Aplicativos específicos para hacer tuning del servidor**
- **Aplicativos de comparación de bases de datos para testar la replicación**

<http://gborg.postgresql.org>

<http://pgfoundry.org/>

*Muchas Gracias*

**Ernesto Quiñones A.**  
**[ernestoq@apesol.org](mailto:ernestoq@apesol.org)**

**Web Site**  
**<http://www.apesol.org>**

**IRC**  
**server: irc.freenode.net    sala:**  
**#apesol #postgresql-es**

**Incribanse en las listas!!!**