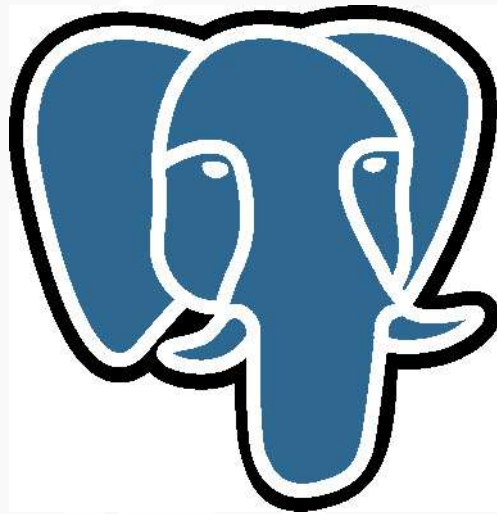


# PostgreSQL



Bienvenue

# PostgreSQL : SGBD libre/Menu

Historique

Rappel des caractéristiques d'un SGBDRO

Architecture

Fonctionnalités de postgresQL

Orientation objet

Catalogue système

Type de données

Structure d'une base

Gestion des utilisateurs

Transactions

Sauvegarde

Paramétrage

Outils de gestion

Interface de connexion

Réplication

Présentation de l' application ACS

Positionnement SGBD libre

Conclusion

# PostgreSQL/historique

## Le projet POSTGRES de Berkeley

mené par le Pr Stonebraker et sponsorisé par le DARPA, ARO, NSF

- ◆ 1986 : Reprise et amélioration du code d'INGRES (1977 – 1985)
- ◆ 1989 : Version 1
- ◆ 1993 : Version 4.2
- ◆ 1994 : Fin du projet POSTGRES de Berkeley, Illustra rachète Postgres et le transforme en Informix

## Postgres95

- ◆ 1994 : Repris par Andrew Yu et Jolly Chen, ajout du SQL et publication Open Source
- ◆ Nombreuses améliorations

## PostgreSQL

- ◆ 1996 : Changement de nom et reprise de la numérotation de version (6.0)
- ◆ 2000 : Version 7.0
- ◆ Fin 2003 : Version 7.3.4
- ◆ Août 2004 : Version 8.0 bêta

# PostgreSQL : Rappel des caractéristiques d'un SGBDR

## Définitions

- ◆ Base de données : ensemble physique de données
- ◆ SGBDRO : Système de gestion de base de données relationnelles objet

## Caractéristiques

- ◆ Gestion de gros volumes de données en consultation et mise à jour
- ◆ Disponibilité
- ◆ Fiabilité
- ◆ Confidentialité
- ◆ Accès concurrents
- ◆ Traçabilité
- ◆ Transactions
- ◆ Portabilité
- ◆ Administrabilité
- ◆ Export / Import
- ◆ Performances

# PostgreSQL : Rappel des caractéristiques d'un SGBDO

## Identité d'objets

Un objet est désigné de façon unique dans le système de gestion de bases de données par un identificateur (OID) qu'il garde pendant toute sa durée de vie, indépendamment de sa valeur.

## Types, classes et méthodes

Correspond à sa structure et aux opérations applicable à l'objet  
types atomiques ou structures complexes ou composition (OID)  
Les objets ayant les mêmes types sont regroupés dans des classes.  
Les opérations que l'on peut appliquer aux objets de la classe sont appelées méthodes.

## Encapsulation

Permet d'affranchir l'utilisateur d'une vision 'interne' de l'objet qu'il manipule.

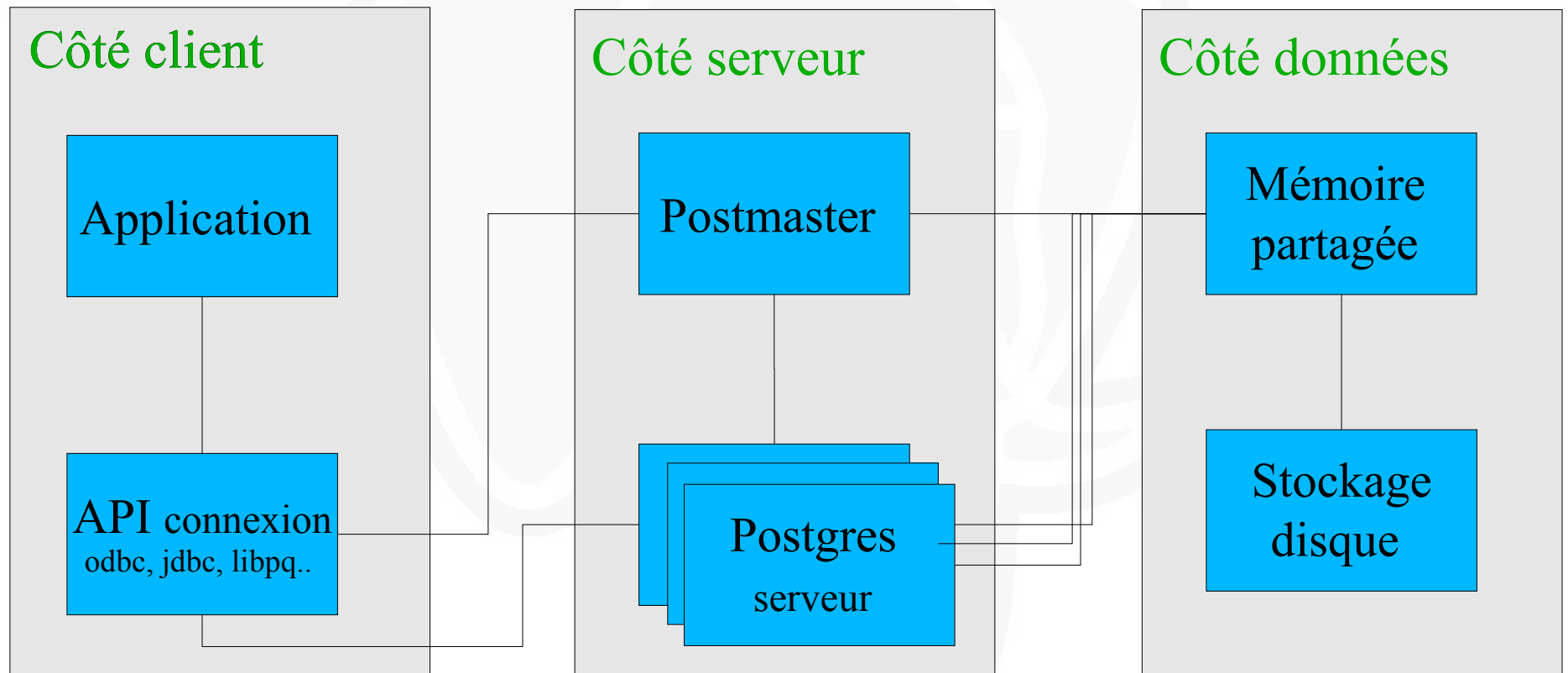
## Héritage

Possibilité de définir une sous-classe à partir d'une classe déjà définie en raffinant ses types.

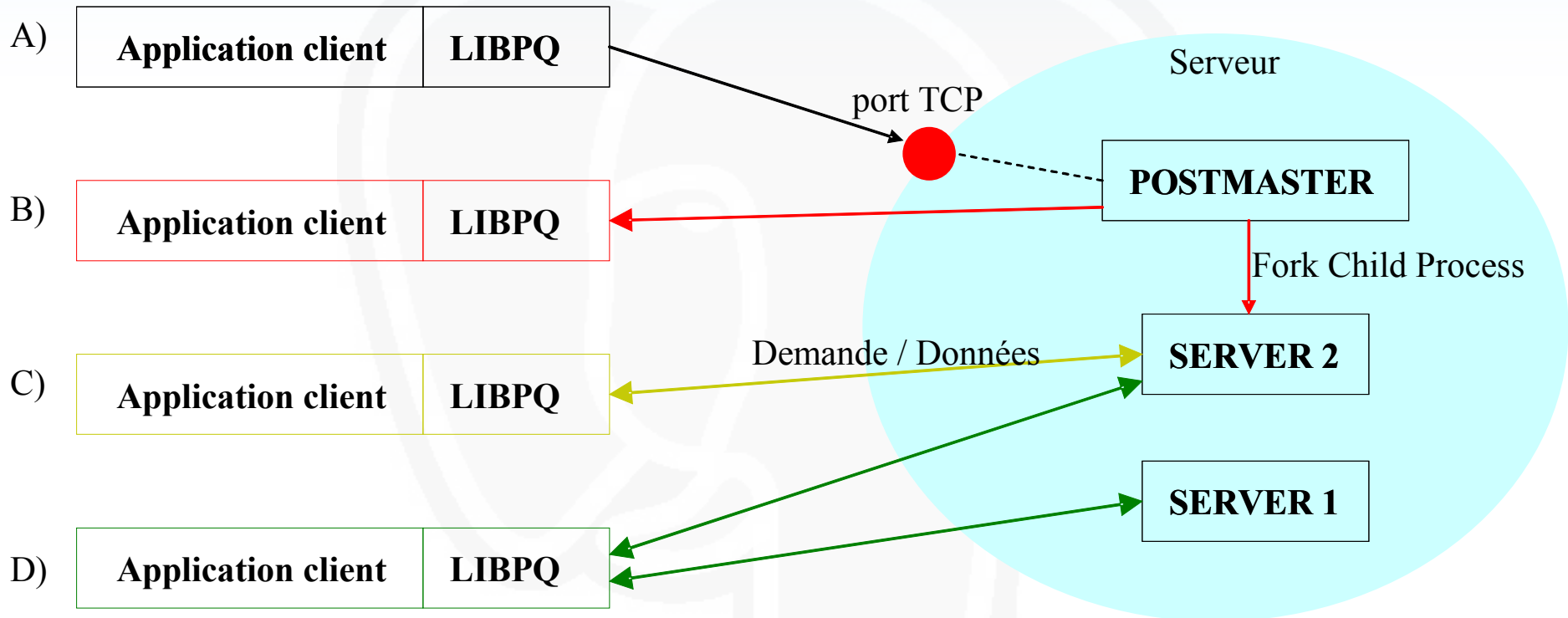
# PostgreSQL/Architecture

Trois sous systèmes :

Client / serveur / gestionnaire données



# PostgreSQL/Architecture



## Cheminement d'une requête :

- ◆ **parser**: vérification de syntaxe, transformation en arbre de requête
- ◆ **rewrite**: réécriture de la requête
  - ◆ vérification des règles (*rules in system catalogs*),
  - ◆ transformation de l'arbre (ex: vues, *virtual table*)
- ◆ **planner/optimizer**: meilleur plan d'exécution (ex: utilisation d'index)
- ◆ **executor**: traitement de l'arbre de requête établi (*plan tree*)
  - ◆ accède au système de stockage en parcourant les tables
  - ◆ exécute les tris et les jointures
  - ◆ évalue les qualifications
  - ◆ retourne les n-uplets demandés

Fonctions clés du modèle orienté objet de PostgreSQL :

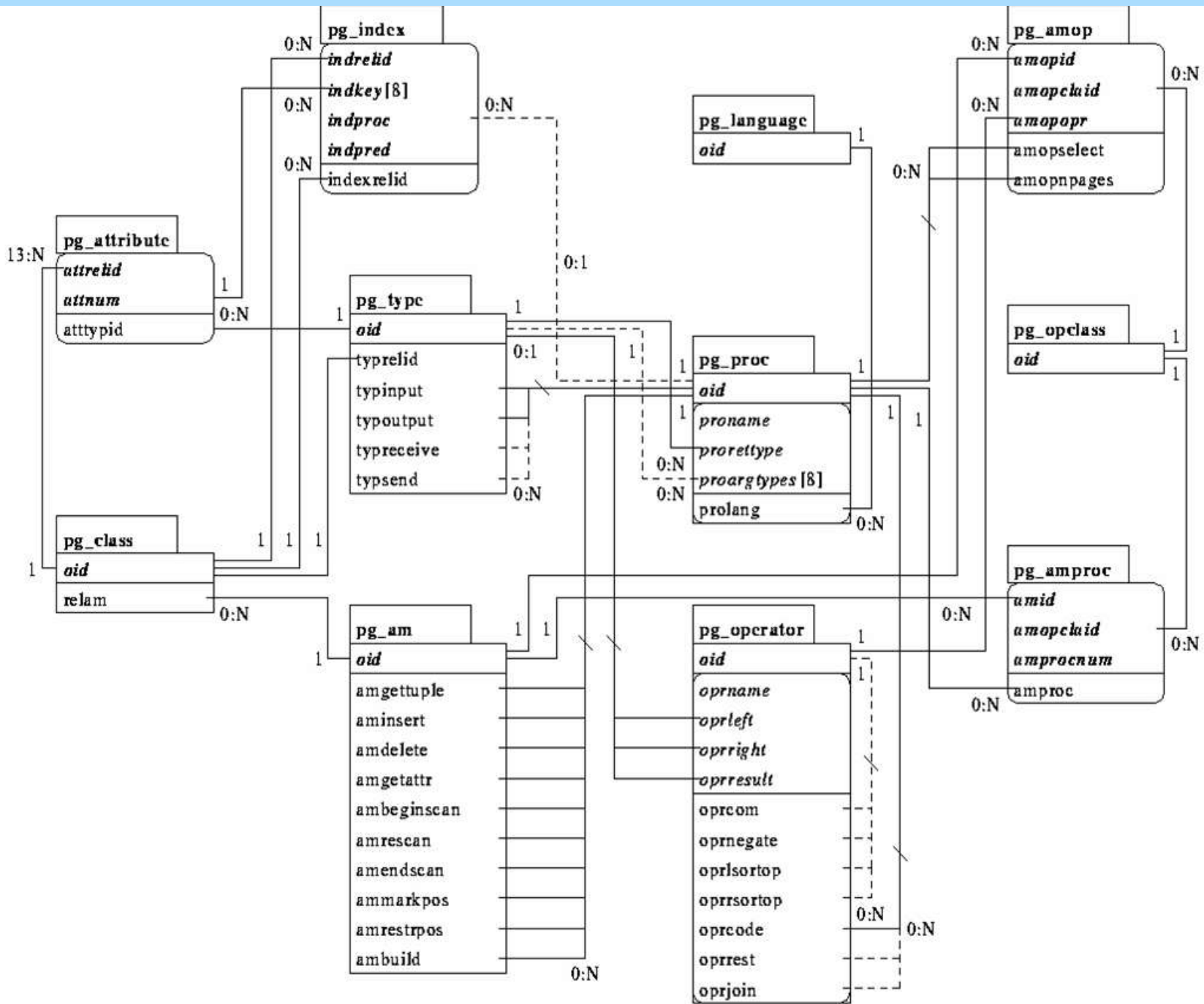
- ◆ **Classes** : Une classe correspond à un ensemble d'objets possédant un identificateur unique.
- ◆ **Héritage** : La notion d'héritage correspond à une organisation hiérarchique des tables. Par exemple, si deux tables se trouvent dans une relation parent/enfant, les informations contenues dans la table parent sont également disponible dans la table enfant.
- ◆ **Surcharge** : On parle de "surcharge de fonction" lorsqu'une fonction peut être définie plusieurs fois avec des paramètres différents.

## Catalogues systèmes (méta-schéma) extensibles

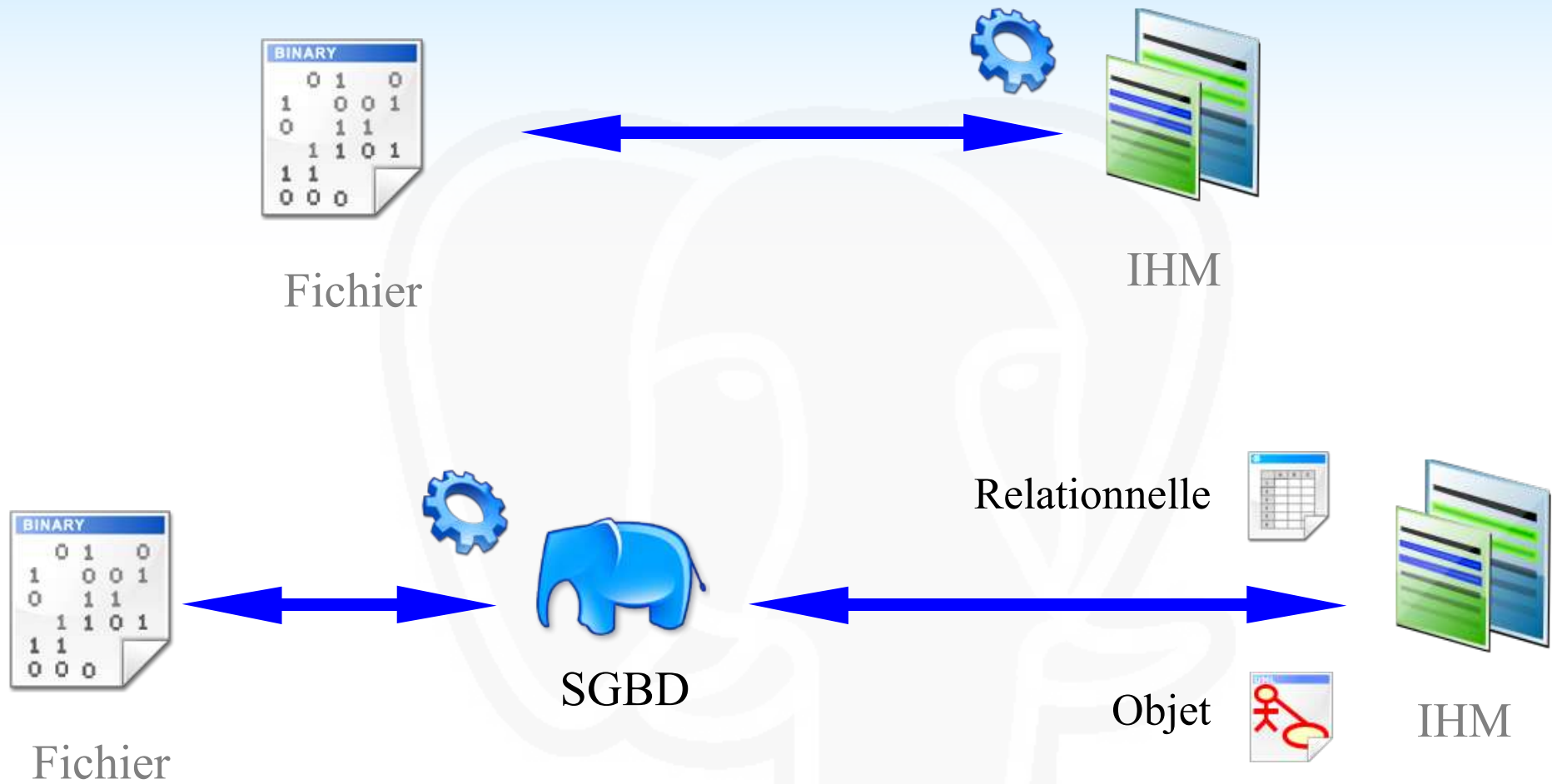
- ◆ pg\_database databases
- ◆ pg\_class classes
- ◆ pg\_attribute class attributes
- ◆ pg\_index secondary indices
- ◆ pg\_proc procedures (both C and SQL)
- ◆ pg\_type types (both base and complex)
- ◆ pg\_operator operators
- ◆ pg\_aggregate aggregates and aggregate functions
- ◆ pg\_am access methods
- ◆ pg\_amop access method operators
- ◆ pg\_amproc access method support functions
- ◆ pg\_opclass access method operator classes

## Chargement dynamique de code (shared library)

- ◆ Fonctions utilisateur
- ◆ Types utilisateur
  - ◆ Système de types de données extensible, permettant de personnaliser des types et de développer rapidement de nouveaux types.



# PostgreSQL/Types





## Types de base

- **Nombre** : smallint, integer, bigint, decimal, numeric, real, double précision, serial, bigserial
- **Texte** : character varying(n), varchar(n), character(n), char(n), text
- **Date** : timestamp without time zone, timestamp with time zone, interval , date, time without time zone, time with time zone



## Opérateurs

+	addition
-	subtraction
*	multiplication
/	division (integer division truncates results)
%	modulo (remainder)
^	exponentiation
/	square root
/	cube root
!	factorial
!!	factorial (prefix operator)
@	absolute value
&	bitwise AND
	bitwise OR
#	bitwise XOR
~	bitwise NOT
<<	bitwise shift left
>>	bitwise shift right



## Types de base

- **Nombre** : smallint, integer, bigint, decimal, numeric, real, double precision, serial, bigserial
- **Texte** : character varying(n), varchar(n), character(n), char(n), text
- **Date** : timestamp without time zone, timestamp with time zone, interval , date, time without time zone, time with time zone



## Fonctions

abs	cbrt
ceil	degrees
exp	floor
ln	log
log	mod
pi	pow
radians	random
round	round
setseed	sign
sqrt	trunc
acos	asin
atan	atan2
cos	cot
sin	tan



## Types de base

- **Nombre** : smallint, integer, bigint, decimal, numeric, real, double precision, serial, bigserial
- **Texte** : character varying(n), varchar(n), character(n), char(n), text
- **Date** : timestamp without time zone, timestamp with time zone, interval , date, time without time zone, time with time zone



## Opérateurs et fonctions

string    string	bit_length
char_length	character_length
convert	lower
octet_length	overlay
position	substring
trim	upper
ascii	btrim
chr	convert
decode	encode
initcap	length
lpad	ltrim
md5	pg_client_encoding
quote_ident	quote_literal
repeat	replace
rpad	rtrim
split_part	strpos
substr	to_ascii
to_hex	translate



## Types de base

- **Nombre** : smallint, integer, bigint, decimal, numeric, real, double precision, serial, bigserial
- **Texte** : character varying(n), varchar(n), character(n), char(n), text
- **Date** : timestamp without time zone, timestamp with time zone, interval , date, time without time zone, time with time zone



## Opérateurs et fonctions

+ - \* /

to\_char            to\_date  
to\_timestamp    to\_number

age  
current\_date  
current\_time  
current\_timestamp  
date\_part  
date\_trunc  
extract  
isfinite  
localtime  
localtimestamp  
now  
timeofday



## Types géométriques

- **Point** : (x,y)
- **Segment** : ((x1,y1),(x2,y2))
- **Rectangle** : ((x1,y1),(x2,y2))
- **Chemin fermé** : ((x1,y1),...)
- **Chemin ouvert** : [(x1,y1),...]
- **Polygone** : ((x1,y1),...)
- **Cercle** : <(x,y),r>



## Opérateurs

+	Translation	box
-	Translation	box
*	Scaling/rotation	
/	Scaling/rotation	
#	Point or box of intersection	
#	Number of points in path or polygon	
@-@	Length or circumference	
@@	Center	
##	Closest point to first operand on second operand	
<->	Distance between	
&&	Overlaps ?	
&<	Overlaps or is left of ?	
&>	Overlaps or is right of ?	



## Types géométriques

- **Point** : (x,y)
- **Segment** : ((x1,y1),(x2,y2))
- **Rectangle** : ((x1,y1),(x2,y2))
- **Chemin fermé** : ((x1,y1),...)
- **Chemin ouvert** : [(x1,y1),...]
- **Polygone** : ((x1,y1),...)
- **Cercle** : <(x,y),r>



## Opérateurs

<<	Is left of ?
>>	Is right of ?
<^	Is below ?
>^	Is above ?
?#	Intersects ?
?-	Is horizontal ?
?-	Are horizontally aligned ?
?	Is vertical ?
?	Are vertically aligned ?
?-	Is perpendicular ?
?	Are parallel ?
~	Contains ?
@	Contained in or on ?
~=	Same as ?



## Types géométriques

- **Point** : (x,y)
- **Segment** : ((x1,y1),(x2,y2))
- **Rectangle** : ((x1,y1),(x2,y2))
- **Chemin fermé** : ((x1,y1),...)
- **Chemin ouvert** : [(x1,y1),...]
- **Polygone** : ((x1,y1),...)
- **Cercle** : <(x,y),r>



## Fonctions

area(object)	area
box_intersect(box, box)	intersection
center(object)	center
diameter(circle)	diameter
height(box)	vertical size
isclosed(path)	a closed path ?
isopen(path)	an open path ?
length(object)	length
npoints(path)	number of points
npoints(polygon)	number of points
pclose(path)	convert path to closed
popen(path)	convert path to open
radius(circle)	radius of circle
width(box)	



## Types Adresse réseau

- **cidr** : Réseau IPv4, IPv6
- **inet** : Hôte ou réseau IPv4, IPv6
- **macaddr** : Adresse MAC



## Opérateurs et Fonctions

< is less than  
<= is less than or equal  
= equals  
>= is greater or equal  
> is greater than  
<> is not equal  
<< is contained within  
<<= is contained within or equals  
>> contains  
>>= contains or equals

broadcast	host
masklen	set_masklen
netmask	hostmask
network	text
abbrev	

trunc



## Types Tableau

### Déclaration : []

```
CREATE TABLE agence
(
    -- Identifiant
    id_agence    serial,
    -- Nom de l'agence
    nom          varchar(64),
    -- Liste des jours de fermeture
    fermeture    date[],
    -- Liste des horaires d'ouverture
    horaire      time[][]
);
```



## Opérateurs et Fonctions

```
=          equal
<>        not equal
<         less than
>         greater than
<=        less than or equal
>=        greater than or equal
||         array-to-array concatenation
||         array-to-array concatenation
||         element-to-array concatenation
||         array-to-element concatenation

array_cat
array_append
array_prepend
array_dims
array_lower
array_upper
array_to_string
string_to_array
```



## Héritage

### Déclaration : **INHERITS**

```
CREATE TABLE agence_tele_prospect  
(  
    -- Nom de de télé-vendeurs  
    nb_vendeur      integer  
) INHERITS (agence);
```

```
CREATE TABLE agence_depot  
(  
    -- Nombre de quais  
    max_quai        integer  
) INHERITS (agence);
```



## Héritage

### Déclaration : **INHERITS**

```
CREATE TABLE agence_tele_prospect  
(  
    -- Identifiant propre  
    id_tele          serial,  
    -- Nom de de télé-vendeurs  
    nb_vendeur      integer  
) INHERITS (agence);
```

```
CREATE TABLE agence_depot  
(  
    -- Identifiant propre  
    id_depot        serial,  
    -- Nombre de quais  
    max_quai        integer  
) INHERITS (agence);
```



## Autres Types

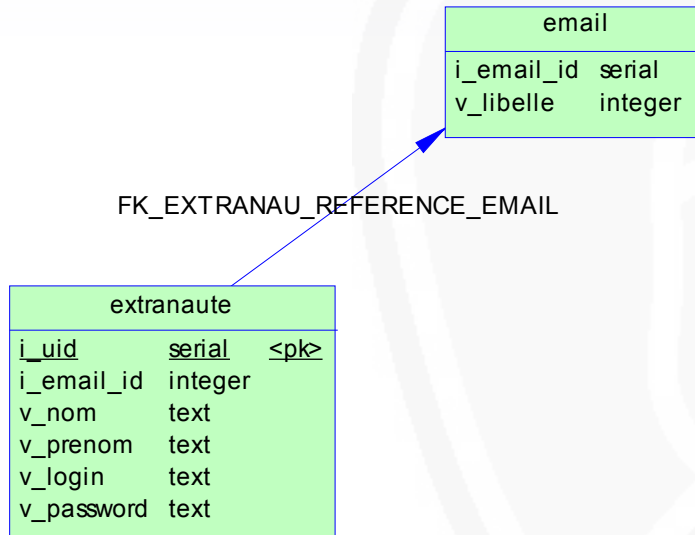
- **Chaîne de bits** : bit, bit varying
- **Identifiant d'objets** : oid
- **Pseudo-types** : any, anyarray, anyelement, cstring, internal, record, trigger, void, opaque

# PostgreSQL/Intégrité référentielle

Ensemble de règles liant des tables pour les opérations d'insertion, de mise à jour, de suppression, permettant d'assurer la cohérence des données. Les contraintes d'intégrité peuvent être :

- ◆ NULL/NO NULL
- ◆ UNIQUE
- ◆ PRIMARY KEY
- ◆ FOREIGN KEY
- ◆ CHECK

# PostgreSQL/Intégrité référentielle



```
CREATE TABLE "public"."extranaute" (  
  "i_uid" SERIAL,  
  "i_email_id" INTEGER,  
  "v_nom" TEXT,  
  "v_prenom" TEXT,  
  "v_login" TEXT NOT NULL,  
  "v_password" TEXT,  
  CONSTRAINT "pk_extranaute" PRIMARY KEY("i_uid"),  
  CONSTRAINT "extranaute_v_login_key" UNIQUE("v_login"),  
  CONSTRAINT "fk_email" FOREIGN KEY ("i_email_id")  
    REFERENCES "public"."email"("i_email_id")  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION  
    NOT DEFERRABLE  
) WITH OIDS;
```

# PostgreSQL/Triggers

- ◆ Opération qui sera appelée lors d'un accès à une table.
- ◆ Appelé dans le cadre d'un **SELECT** et/ou **INSERT**, et/ou **UPDATE**.
- ◆ Intervient avant et/ou après que la manipulation des données soit effective

```
BEGIN
```

```
  _the_timestamp:=to_char(current_timestamp,'YYYY-MM-DD HH24:MI:SS');
```

```
  If TG_OP='DELETE'
```

```
  Then
```

```
    _the_sql:='delete from extranaute '|| 'where i_uid='''||OLD.i_uid||''''|| ';;
```

```
    Insert into LOG_TABLE(v_script_sql,v_user) Values (_the_sql,current_user);
```

```
    Return OLD;
```

```
  End If;
```

```
  If TG_OP='INSERT' or TG_OP='UPDATE'
```

```
  Then
```

```
    NEW.d_date_synchro:=_the_timestamp;
```

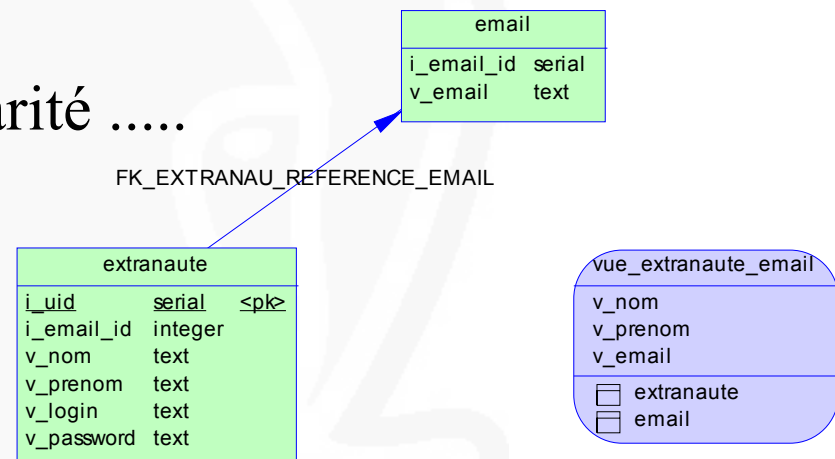
```
  End If;
```

```
.....
```

```
END
```

# PostgreSQL/Vue

- Objet de **définition logique**
- Seule la définition est stockée
- Construit sur une requête SQL
- Permet l'encapsulation, la modularité .....



```
CREATE OR REPLACE VIEW "vue_extranaute_email"
(v_nom, v_prenom, v_email)
AS
SELECT e.v_nom , e.v_prenom , m.v_email
from extranaute e , email m
where e.i_email_id = m.i_email_id
```

## ◆Création d'un utilisateur et affectation de droit

```
CREATE USER metaadmin  
WITH SYSID 115  
PASSWORD ENCRYPTED  
md5c7600fa65f1e272e7bb82c1f644554f9  
CREATEDB CREATEUSER;
```

## ◆Pour chaque objet, définition des droits

```
GRANT ALL ON TABLE public.extranaute TO postgres;  
GRANT SELECT, UPDATE, INSERT ON TABLE public.extranaute TO metaadmin;
```

**PL/pgSQL** « basé » sur **PL/SQL** : langage orienté bloc supportant

- ◆ la déclaration de variables
- ◆ les boucles
- ◆ des constructions logiques
- ◆ une gestion des erreurs avancée
- ◆ la surcharge de fonction

```
SELECT INTO clef i_email_id FROM EMAIL where i_email_id = row_data.i_person_num;
IF NOT FOUND THEN
  RAISE NOTICE "***** Enregistrement extranaute *****";
  RAISE NOTICE "1-Enregistrement email de la personne %." , row_data.i_person_num;
  -- enregistrement n existe pas : maj de la table

  INSERT INTO EMAIL (i_email_id,v_email,d_creat ) VALUES (
    row_data.i_person_num,"change_on_install@cohesion.net",row_data.d_creat);
END IF;
```

Langage moins riche que PL/SQL mais d'autres interfaces existent

- ◆ **Php, python, perl**

Les transactions sont un des concepts fondamentaux de tous les systèmes de base de données.

- ◆réunit plusieurs étapes en une unique opération "tout ou rien".
- ◆Etats intermédiaires entre les étapes ne sont pas visibles pour les autres transactions concurrentes
- ◆Abandon des étapes en cas d'erreur.
  
- ◆Données en cours de modification par la transaction sont marqués (pg\_log) et non visible
- ◆En cas de crash, le processus vérifie l'état des transactions en cours

Pas de support de transactions distribuées.

## Gestion des accès concurrents : système multi-version

- ▶ Chaque transaction voit la **version de la base** correspondant au début de l'exécution (notion de snapshot)
- ▶ **Lecteur** ne bloque pas **l'éditeur**, **l'éditeur** ne bloque pas le **lecteur**
- ▶ **L'éditeur** bloque **l'éditeur** sur le même tuple

## Exemple d'update concurrent

**transaction A** : en cours

*update extranaute set nbCon = nbCon + 1 where i\_ext\_id = 18*

**transaction B** : même requête avant commit de A

- ◆ B attends le résultat de la transaction A
- ◆ Si A abandonne, B s'exécute avec l'ancienne valeur
- ◆ Si A commit ....deux cas selon la valeur de x à prendre en compte

Deux solutions sont possibles : jouer sur les niveaux de transaction

## Read committed level

- ▶ transaction B utilise la nouvelle valeur
- ▶ Défini par défaut

## Serializable

- ▶ B abandonne : not serializable error.
- ▶ Nécessité de rejouer B

## Commande `pg_dump` et `pg_dumpall`

<code>--data-only</code>	dump only the data, not the schéma
<code>--blobs</code>	include large objects in dump
<code>--clean</code>	clean (drop) schema prior to create
<code>--create</code>	include commands to create database in dump
<code>--inserts</code>	dump data as INSERT, rather than COPY, commands
<code>--column-inserts</code>	dump data as INSERT commands with column names
<code>--oids</code>	include OIDs in dump
<code>--schema-only</code>	dump only the schema, no data
<code>--table=TABLE</code>	dump this table only

## Chargement en masse

```
COPY nomTable FROM 'nomfichier' WITH DELIMITER '|' WITH NULL  
'NULL'
```

# PostgreSQL/Paramétrage

PostgreSQL s'optimise de lui même, néanmoins il faut avoir connaissance de quelques paramètres :

- ♦ **shared\_buffers size** : taille de l'espace mémoire de travail
  - ♦ Grande pour stocker les objets
  - ♦ Petite pour éviter au système de swapper
- ♦ **sort\_mem size** : fichiers temporaires utilisés pour le tri
  - ♦ Diminue le nombre de fichier pour le merge final
  - ♦ Utilise plus de mémoire donc risque de swap
- ♦ Théorie du 25/4 : 25% cache, 4% sort de la RAM
- ♦ **checkpoint\_segments** : jouer sur la fréquence

Les données après un update sont marquées comme expirées et stockées. **Vacuum** permet de

- ▶ supprimer les données expirées
- ▶ compacter les fichiers de données
- ▶ calculer les statistiques pour l'accès aux données

Exécution de la commande

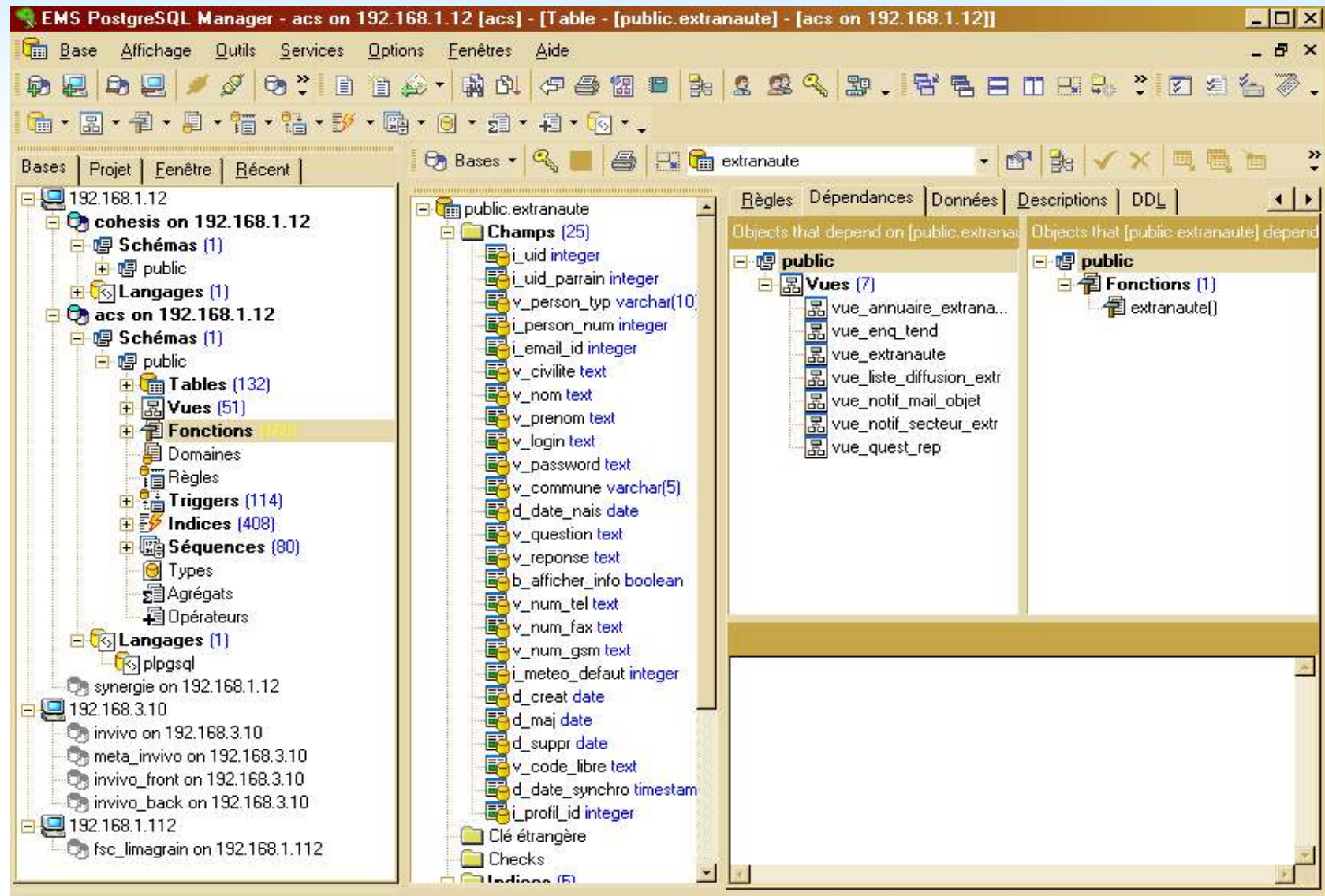
- ▶ par table ou sur la base
- ▶ pas de génération de verrou
- ▶ récurrente, à partir 5% des données modifiées

The screenshot displays the PgAdmin III interface. On the left, a tree view shows the database structure: Langages (1), Schémas (1), and the 'public' schema containing various objects like Aggrégats, Fonctions, and Tables. The 'apporg' table is selected. The main pane shows the 'Propriétés' (Properties) tab for this table, listing attributes such as Nom, OID, Propriétaire, ACL, Clé primaire, and Lignes. Below this, the 'Statistiques' (Statistics) tab is active, showing the SQL code to create the table.

Propriétés	Valeur
Nom	apporg
OID	199070
Propriétaire	postgres
ACL	{=,postgres=arwdRxt,javauser=arwd,acscsconsult=r}
Clé primaire (Primary Key)	aog_nao
Lignes (estimées)	1000
Lignes (comptées)	32
Hérite de tables	Non
Nb. tables héritées	0
Avec OIDs ?	Oui

```
-- Table: public.apporg
-- DROP TABLE public.apporg;

CREATE TABLE public.apporg
(
  aog_nao varchar(30) NOT NULL,
  aog_unao int4 NOT NULL,
  aog_azoa float8,
  aog_k2oa float8,
  aog_p2o5a float8,
  aog_typeao int4,
```



## Connectivité importante

- ◆ ODBC : plateforme .NET
- ◆ JDBC : plateforme J2EE
- ◆ libpq : application C, C++
- ◆ Module de processeur de texte : PHP, Perl, TCL, Python

## Type de réplication

- ♦ **Maitre/esclave** : mise à jour maître et copie des données sur l'esclave
- ♦ **Point à point** : mise à jour sur tous les serveurs

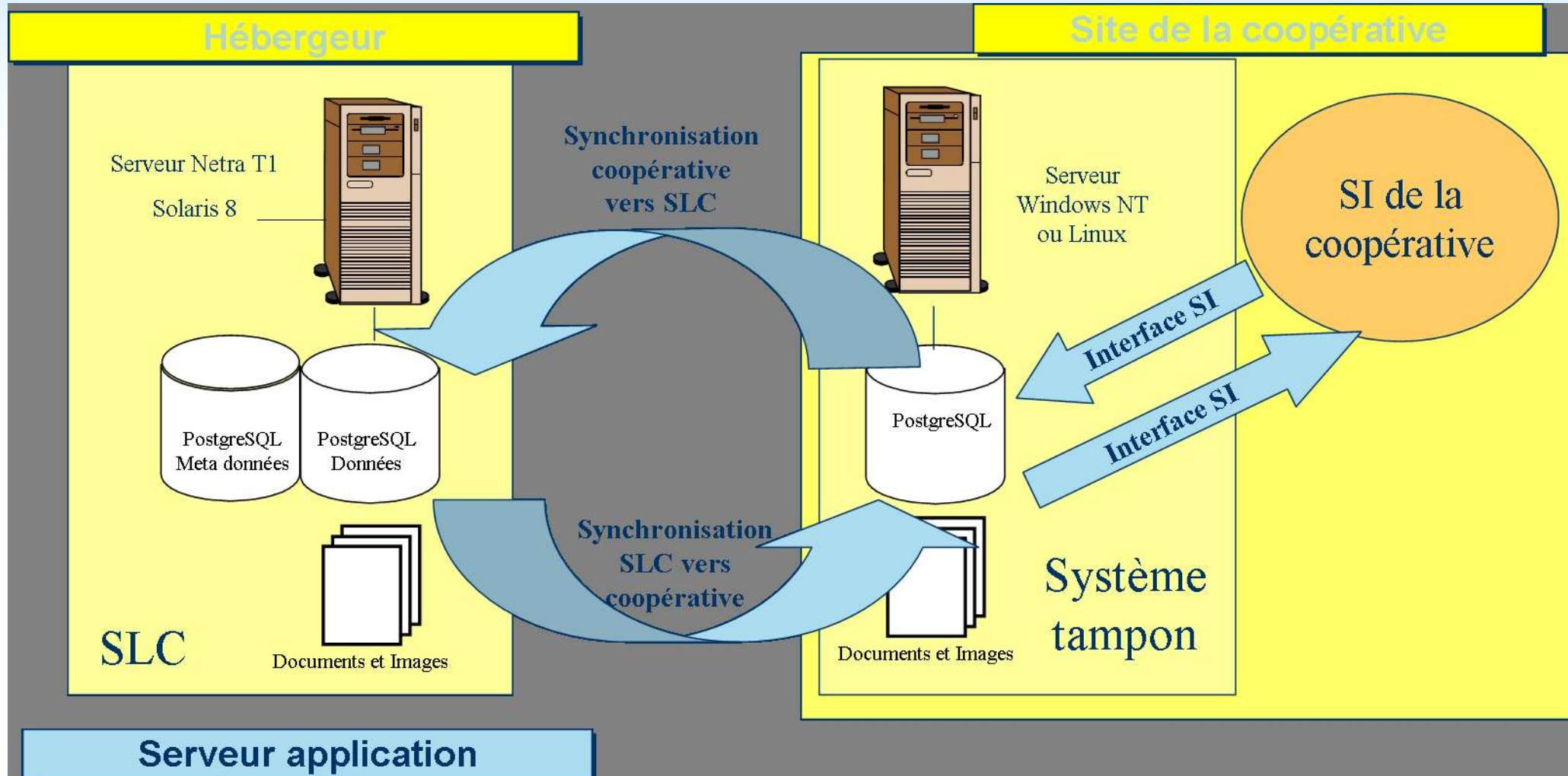
## Type de propagation

- ♦ **Asynchrone** : données validées envoyées selon ..
- ♦ **Synchrone** : phase de pre-commit envoie les données aux abonnés et attend les résultats

Pas de solution native : Postgres-R, Replicator, eRServer

The screenshot shows a Mozilla browser window displaying the 'Extranet de démonstration' for 'COOP DE PARIS'. The page features a navigation menu with links like 'COOP', 'SERVICES', 'ACTUALITES', etc. The main content area is titled 'SERVICES' and includes a 'Questionnaire' section. Below this, there is a table titled 'Listes des Questionnaires' with the following data:

Tous les Questionnaires		Etat	N°	Auteur	Modifié le	Par
<a href="#">Demande inscription extranet</a>	Questionnaire utilisé par la demande d'authentification	1 rép.	56	Désangles Stéphane	08/07/2004	Désangles Stéphane
<a href="#">Questionnaire de juillet</a>	Résumé...	Inactif	60	Désangles Stéphane	07/07/2004	Désangles Stéphane
<a href="#">test questionnaire</a>		Inactif	61	Désangles Stéphane	07/07/2004	Désangles Stéphane
<a href="#">Copie de Promo choc de l'automne</a>	Questionnaire utilisé par une offre	Inactif	19	Duquesne laurence	19/01/2004	Désangles Stéphane
<a href="#">Questionnaire n°22 - A utiliser pour la recette LOT3</a>	Ligne 1 : Exemple de résumé	8 rép.	22	Laviale Frederic	29/01/2003	DURLACH Martine



Par exemple passage de 7.3 à 7.4

- ◆ Améliorations dans le traitement des requêtes imbriquées par l'optimiseur de requêtes, permettant une accélération jusqu'à 400 % de certaines requêtes complexes
- ◆ Nouveau protocole de communication réseau (version 3), qui améliore la vitesse des transferts de données
- ◆ Amélioration des index basés sur des fonctions, qui permet une meilleure indexation des types de données personnalisés et des champs composites

A venir

- ◆ Version windows native en cours
- ◆ Gestion des tablespaces
- ◆ Support XML

## Cinq critères essentiels de classement des SGBD

### ◆ Performance

accès en lecture, écriture, montée en charge

### ◆ Ouverture et respect des standards

connectivité, SQL99, support XML

### ◆ Disponibilité et sécurité

réplication, sauvegarde à froid, chaud, reprise sur incident

### ◆ Couverture fonctionnelle

fonctions SQL avancées, OLAP, web services

### ◆ Installation et administration

prise en main, outils, documentation, support technique

## Cinq critères essentiels de classement des SGBD

### ◆ Performance

très bonnes performances depuis 7.4

### ◆ Ouverture et respect des standards

SQL99 et partiellement SQL2003, nombreux langages

### ◆ Disponibilité et sécurité

faible support (pas de réplication, de répartition)

### ◆ Couverture fonctionnelle

Bonne couverture mais manque XML, OLAP

### ◆ Installation et administration

Nécessite des compétences « pointues », assistance technique aléatoire et documentation moyenne. Pas d'outils.

Communauté importante.

- ♦ Base performante et stable
- ♦ Couverture fonctionnelle « rivalise » les SGBD commerciaux
- ♦ Administration simple ...
- ♦ Support GIS
  
- ♦ Nécessite une bonne connaissance du fonctionnement du SGBD et de l'OS serveur
- ♦ Support XML
- ♦ ...administration limitée, peu d'outils d'administration