INSIA
Bases de données
PostgreSQL - 2 - DDL
Bertrand LIAUDET

SOMMAIRE

SOMMAIRE	1
DDL	3
Structuration générale des données	3
Clusters : répertoires des données, groupes de BD	3
Espaces de tables	3
Principe	3
Situation par défaut	4
Bases de données	4
Principe	4
Situation par défaut	4
DDL d'une BD	4
Jeux de caractères	5
Schémas	5
Principe	5
Situation par défaut	5
Accès à un ou plusieurs schémas : search_path	6
Nom des tables	7
DDL d'un schéma	7
Tables	7
DDL des tables	7
Les contraintes	8
Les clés	8
CHECK et DEFAULT	8
Domaines de valeur	9
Séquence	9
Héritage	10
Types	11
Tous les types	11
Type composite	12

Les dates	13
Les tableaux	13
Les vues	14
Créer une table à partir d'une requête	14
Les tables temporaires	14
Principe	14
Création	14
Utilisation	14
Suppression	14
Import-Export de données : COPY	14
Exportation format Excel	14
Importation format Excel	14
Autres formats d'import-export	15
Les index	15
Principes	15
Create Index	15
SELECT	16
Opérateurs et Fonctions mathématiques	16
Opérations ensemblistes	16
IN et ROW	16
Fonctions de traitement de chaînes	16
TO_CHAR et EXTRACT	16
LIKE, ILIKE et SIMILAR	17
LIMIT et OFFSET	17
Jointure externe : droite, gauche et bilatérale	17
TP	18
1) PgAdmin III	18
Présentation de l'outil	18
2) EcolingScript n°3	19

Première édition : mars 2009

DDL

Structuration générale des données

- Clusters
- Espaces de tables
- Bases de données
- Schémas
- Tables et index

Clusters : répertoires des données, groupes de BD

Un **cluster** est un **répertoire des données** pour un **groupe de bases de données**.
Attention à la polysémie du mot cluster !
Un serveur peut avoir plusieurs instances.
Chaque instance possède un cluster unique et il y a une instance unique par cluster.
Quand on installe postgresQL, l'installation est faite pour une seule instance.
Par défaut le répertoire des données est :

```
/var/lib/postgres.data : sous unix  
C:/program files/postgres/n°Version/data : sous XP
```

L'installation Debian propose des scripts qui permettent de créer un cluster, supprimer un cluster, lister les cluster, démarrer ou arrêter une instance pour un cluster donné.

Espaces de tables

Principe

Un espace de tables est un répertoire du système d'exploitation qui contient les données des tables et des index.
Par défaut, une instance possède un espace de table situé dans le répertoire des données.
On peut créer d'autres espaces de tables, en général sur d'autres disques que celui du répertoire des données, ou sur d'autres partitions.
Un espace de tables contient des tables et des index.
Un espace de tables peut contenir des tables et des index de plusieurs schémas et de plusieurs bases de données.

Les tables et les index d'un même schéma peuvent être répartis dans plusieurs espaces de tables. On peut créer, modifier et supprimer les espaces de tables. On peut déplacer les tables et les index dans n'importe quel espace de tables.

La gestion des espaces de tables permet de :

- augmenter les ressources de stockage.
- améliorer les performances.
- sécuriser des accès aux données.

Situation par défaut

Il y a 2 espaces de table par défaut :

- pg_default : ce sera le tablespace par défaut des bases de données futures
- pg_global : il contient certains objets des bases de données systèmes.

Bases de données

Principe

Une base de données est contenue dans un cluster et un seul.

Un cluster contient plusieurs bases de données.

On ne peut travailler que sur une seule base à la fois.

Une base de données contient des schémas.

Elle accueille les connexions.

Situation par défaut

Il y a 3 BD par défaut :

- postgres
- template1
- template0

Template0 et Template1 sont des modèles (template) de BD.

On ne peut rien ajouter à template0 : on ne peut pas se connecter à template0

On peut ajouter tout ce qu'on veut à template1 : on peut se connecter à template1.

N'importe quelle BD peut servir de modèle.

Créer une BD sans aucuns paramètres revient à créer une BD avec template0 comme modèle.

DDL d'une BD

CREATE DATABASE nomBase

\h create database

Les bases ont : un propriétaire, un tablespace, un encodage (jeu de caractères).

Les bases peuvent avoir : un modèle, une limite du nombre de connexion.

ALTER DATABASE nomBase

\h alter database

On peut modifier le nom, le propriétaire, la limite du nombre de connexion.

On peut modifier des variables d'environnement (par exemple : search_path).

On ne peut modifier ni le tablespace, ni l'encodage.

➤ **Exemple : mise à jour du search_path dans une BD mono-schéma**

```
Alter database maBase set search_path='monSchéma' ;
```

DROP DATABASE [if exists] nomBase

\h drop database

Jeux de caractères

Les principaux jeux de caractères sont les suivants :

UTF8 = Unicode : pour tous les alphabets de toutes les langues

LATIN1 = ISO88591 : alphabets d'Europe de l'ouest.

LATIN9 = ISO88915 : alphabets d'Europe de l'ouest.

WIN1252 : alphabets d'Europe de l'ouest.

```
\dc : pour voir les conversions entre jeux de caractères.
```

Exemple : création d'une BD avec un jeu particulier

```
Create database maBase ENCODING 'UTF8' ;
```

Schémas

Principe

Un schéma est contenu par une BD et une seule.

Une BD contient plusieurs schémas.

Un schéma contient des tables, des vues, des index, des procédures stockées, etc : tous les objets possibles d'une BD.

On peut travailler sur des objets de plusieurs schémas en même temps.

Situation par défaut

5 schémas par défaut

Il y a 5 schémas par défaut dans la BD postgres.

- information_schema
- pg_catalog
- pg_toast

- pg_toast_temp_1
- public

On retrouve ces 5 schémas à chaque création de BD ou quand on crée une BD avec le template0.

Information schema

C'est le dictionnaire des données. Il contient essentiellement des vues. Elles sont visibles directement avec un \d.

Les schémas « pg »

Ce sont les schémas systèmes. Ils contiennent des tables systèmes (74) qu'il faut éviter de manipuler. Ces tables sont visibles avec un \dS.

On retrouve les tables système dans tous les schémas.

Ces tables sont utilisées pour construire des vues du schéma «information_schema ». On peut voir le code de ces vues avec un \d.

Le schéma public

C'est un schéma public dans lequel on peut mettre ce qu'on veut. Par défaut, il ne contient rien, à part les tables systèmes.

Accès à un ou plusieurs schémas : search_path

présentation

La variable « search_path » permet de sélectionner les schémas de travail.

Par défaut cette variable vaut : le schéma du nom de l'utilisateur et le schéma public.

```
postgres=# show search_path;
search_path
-----
"$user",public
```

avec :

```
postgres=# select user;
current_user
-----
postgres
```

Si on crée une table, elle ira d'abord dans le schéma du nom de l'utilisateur, si elle existe, sinon elle ira dans le schéma public.

Modification de la variable « search_path »

➤ *Au niveau de la session*

```
postgres=# set search_path='information_schema', 'public';
```

ou

```
postgres=# set search_path to 'information_schema', 'public';
```

➤ **Au niveau de la BD**

```
Alter database maBase set search_path='monSchéma' ;
```

➤ **Au niveau d'un utilisateur**

```
Alter user nomUser set search_path='monSchéma' ;
```

Alter user nomUser Set search_path to schéma1, schéma2, etc; // modification définitive.

Nom des tables

On peut préfixer les tables du nom de leur schema : nomSchéma.nomTable

DDL d'un schéma

CREATE SCHEMA nomSchema

\h create schema

Un schéma a un nom et un propriétaire (AUTHORIZATION).

ALTER SCHEMA nomSchema

\h alter schema

On peut changer le nom (RENAME TO) et le propriétaire (OWNER TO) d'un schéma.

DROP SCHEMA [if exists] nomBase

\h drop schema

On ne peut supprimer que les schémas vides, sauf avec l'option CASCADE

Tables

DDL des tables

Le DDL des tables retrouve la syntaxe standard SQL.

CREATE TABLE nomTable

\h create table

WITH OIDS : permet d'ajouter à la table une colonne d'identifiants de ligne.

WITH TABLESPACE nomSpace : permet d'indiquer le tablespace de la table

WITH FILLFACTOR = valeur entre 0 et 100, 100 par défaut. Pour optimiser les écritures.

ALTER TABLE nomTable

\h alter table

ALTER TABLE nomTable SET TABLE SPACE nomTablespace;

ALTER TABLE nomTable SET SCHEMA nomSchema;

ALTER TABLE nomTable OWNER TO nomUser;

ALTER TABLE nomTable RENAME TO newNomTable;
ALTER TABLE nomTable RENAME colonne TO newNomColonne;
ALTER TABLE nomTable ADD colonne type [contraintes];
ALTER TABLE nomTable DROP colonne [CASCADE]
ALTER TABLE nomTable ALTER colonne TYPE newType ;
ALTER TABLE nomTable ALTER colonne SET DEFAULT newDefault;
ALTER TABLE nomTable ALTER colonne DROP DEFAULT;
ALTER TABLE nomTable ALTER colonne {SET | DROP} NOT NULL;
ALTER TABLE nomTable ADD contrainteIntégrité;
ALTER TABLE nomTable DROP CONSTRAINT nomContrainte [CASCADE];
ALTER TABLE nomTable SET FILLFACTOR=valeur;

DROP TABLE [if exists] table [CASCADE]

\h drop table

Les contraintes

Les clés

Clé primaire, clé étrangère, unique, non null.

Une clé étrangère fait référence à une clé primaire ou a un unique.

Les clés étrangères : on delete, update : no action (par défaut), restrict, cascade, set null, set default.

Contrainte d'intégrité référentiel concaténée

Il y a 3 façons de traiter une contrainte d'intégrité concaténée

- 1) MATCH FULL : la contrainte s'applique pour toutes les valeurs possibles, sauf si tous les attributs valent NULL : c'est la situation par défaut.
- 2) MATCH PARTIAL : la contrainte s'applique sur les colonnes non NULL
- 3) MATCH SIMPLE : la contrainte ne s'applique pas dès qu'une des colonnes vaut NULL.

Contrainte d'intégrité référentiel et transaction

ON {DELETE | UPDATE } RESTRICT : est equivalent à NO ACTION mais la contrainte est reportée à la fin de la transaction.

CHECK et DEFAULT

DEFAULT expression : pas de variables ni de requêtes dans l'expression.

Le CHECK est suivi d'une expression booléenne dont l'évaluation à vrai ou indéfini conditionne les insertions et les mises à jour.

CHECK (attribut > 0)

Les expressions booléennes peuvent utiliser les expressions régulières POSIX :

CHECK (VALUE ~ 'expression régulière')

<http://www.postgresql.org/docs/8.3/static/functions-matching.html>

Domaines de valeur

Principe

Un domaine de valeur est un type associé à certaines contraintes.

Le domaine étant créé, la définition des attributs des tables peut utiliser l'utiliser.

Création d'un domaine

```
CREATE DOMAINE nomDomain AS type [DEFAULT expression] [NOT NULL] [CHECK expression] ;
```

Suppression d'un domaine

```
DROP DOMAINE nomDomain;
```

Séquence

Principe et codage

Les séquences sont des compteurs.

On les manipule avec la fonction nextval().

```
CREATE SEQUENCE no_seq;
CREATE TABLE maTable(
    NT      integer default nextval ('no_seq') primary key,
    ...
);
```

Les types « serial » et « bigserial » permettent la création automatique d'un compteurs.

Le code précédent est équivalent au suivant :

```
CREATE TABLE maTable(
    NT      serial primary key,
    ...
);
```

Détail du contenu

\d : affiche toutes les relations dont les séquences

\ds : affiche toutes les séquences

\d nomSéquence : affiche le détail des attributs d'une séquence

select * from nomSéquence : affiche l'état actuel de la séquence : particulièrement la dernière valeur entrée.

Limite des séquences

On peut imposer une valeur indépendamment de l'auto-incrément. Dans ce cas, l'auto-incrément n'est pas mis à jour !

Exemple : si on entre les 5 premiers en auto-incrément, la dernière valeur entrée vaut 5. Si on entre un 6^{ème} « en dur » en passant la valeur 6, la dernière valeur entrée dans le compteur est toujours 5. Si on relance les entrées avec l'auto-incrément, rien ne passe car la première valeur de l'auto-incrément est désormais 6... qui est déjà dans la BD !

Modification d'une séquence

On peut modifier une séquence et particulièrement changer la valeur de la dernière valeur entrée.

On ne peut pas faire d'update d'une séquence.

Mais on peut faire un ALTER d'une séquence.

```
ALTER SEQUENCE nomSequence RESTART WITH valeurDeDépart;
```

On peut aussi utiliser la fonction « setval » (cf ci-dessous).

Paramétrage d'une séquence

Dans une séquence, on peut paramétrer : l'incrément, la valeur de départ, la valeur minimum, la valeur maximum, la possibilité de reprendre au minimum quand on a atteint le maximum.

Suppression

```
DROP [if exists] SEQUENCE nomSequence
```

Fonctions de manipulation des séquences

bigint nextval (sequence) : incrémente la séquence et renvoie la nouvelle valeur.

bigint currval (sequence) : renvoie le last_val de la séquence

bigint lastval () : renvoie le lastval faisant suite au dernier nextval (équivalent: last insert id)

bigint setval (sequence, bigint [, Boolean]) : impose une valeur au last_val de la séquence. Le booléen à faux indique que le prochain appel (1 seul) à nextval n'avancera pas.

```
DROP [if exists] SEQUENCE nomSequence
```

Héritage

Principes

Les tables espèces sont codées comme des tables classiques.

Mais elles ne contiennent pas de clé primaire.

Elles héritent de leurs tables genres.

```
Create table nomEspèce (  
...  
) inherits (nomTableGenre);
```

La table espèce hérite des attributs de la table genre.

```
\d tableEspèce : décrit les attributs de l'espèce et du genre.
```

Quand on insère un tuple dans la table espèce, le SGBD répartit l'information dans les tables espèce et genre et crée donc 2 tuples.

Quand on supprime un tuple de la table espèce, le SGBD supprime aussi le tuple correspondant dans la table genre.

La table espèce hérite des attributs de la table genre.

Limites

La contrainte d'unicité de la clé primaire de la table genre n'est pas gérée pour les tuples provenant de la table espèce, dans tous les cas de figure (situation sur PostgreSQL 8.3.6 sous XP) :

- On peut mettre dans la table espèce des tuples dont l'identifiant est déjà dans la table genre.
- On peut mettre dans la table espèce des tuples dont l'identifiant est déjà dans la table espèce.
- On peut mettre dans la table genre des tuples dont l'identifiant est déjà dans la table espèce.
- On ne peut pas mettre dans la table genre des tuples dont l'identifiant est déjà dans la table genre et pas dans la table espèce.

Types

Tous les types

Les numériques

A chaque type, correspond un autre nom avec le nombre d'octets

smallint, int2

integer, int, int4

bigint, int8

serial, serial4

bigserial, bigserial8

real, float4

double precision, float8

numeric[(p,s)], decimal[(p,s)], avec p, nombre total de chiffres, s, nombre de décimales

Les caractères

char [(n)], character[(n)] : longueur fixe

varchar[(n)], character varying[(n)] : longueur variable

text : longueur illimitée

Booléen

boolean, bool

Date

date : de –beaucoup avant JC à + beaucoup après JC !

time : jusqu'à la microseconde

timestamp : date et heure, jusqu'à la microseconde.

interval : interval de temps, entre – et + 178 millions d'année, précision jusqu'à la microseconde.

Type spéciaux

Adresses d'ordinateur

Suite de bits fixe, variable

Tableau d'octets.

<u>Type composite</u>

Création

```
CREATE TYPE nomType AS(  
    nomAttribut1 type,  
    nomAttribut2 type,  
    Etc.  
)
```

Contraintes

On ne peut pas créer de contraintes à la création du type.

On peut ajouter des contraintes dans les tables qui l'utilisent.

On peut créer des domaines avec un type composite.

Consultation

```
\d nomType
```

Suppression

```
DROP TYPE nomType
```

Insert

A la place d'une simple valeur on écrit :

```
ROW(valeur att1, valeur att2, etc.)
```

Update / Select

On préfixe l'attribut avec le nom du type composé entre parenthèses.

```
(typeComposé).attribut
```

Exemple

```
CREATE TYPE adresse AS(  
    numero        smallint,
```

```
voie      varchar(10)
rue       varchar(40),
cp        smallint,
ville     varchar(30),
)
```

Les dates

La variable « datestyle »

show datestyle : SQL, DMY

« Datestyle » prend deux valeurs parmi (ISO, SQL, POSTGRES, GERMAN) et (DMY, MDY)

Mise à jour de la variable « datestyle »

SET datestyle TO valeur

Les tableaux

Principe

POSTGRES permet de déclarer des attributs de type « tableau », ce qui est une violation de la première forme normale !

Création

```
CREATE TABLE nomTable(
...
attributN    type[ ]
attributM    type[5]
attributP    type[2][3]
...
)
```

Insert

A la place d'une simple valeur on écrit :

```
ARRAY[valeur1, valeur att2, etc.]
```

Ou bien :

```
'{valeur1, valeur2, etc.}'
```

Pour un tableau à deux dimensions :

```
ARRAY[ [4,3], [5,2], ...]
```

Ou bien :

```
'{{4,3}, {5,2}, ...}'
```

Les vues

Fonctionnement standard.

```
CREATE [ OR REPLACE ] VIEW nomVue [ (nomColonne, ...) ] AS requête ;
```

```
DROP VIEW nomVue ;
```

Créer une table à partir d'une requête

```
CREATE TABLE nomTable AS requête;
```

Les tables temporaires

Principe

La durée de vie d'une table temporaire est celle de la session qui l'a créée.

Les tables temporaires sont rangées dans des schémas temporaires nommées pg_temp_«n», n étant un numéro.

Création

```
CREATE TEMPORARY TABLE nomTable AS requête ;
```

Utilisation

Les schémas temporaires sont inclus automatiquement aux search_path. Les tables des schémas temporaires sont parcourues avant les objets persistants. Ainsi, les objets que ces schémas contiennent masquent automatiquement les objets persistants ayant le même nom.

Suppression

```
DROP TEMPORARY TABLE nomTable AS requête ;
```

Import-Export de données : COPY

Exportation format Excel

```
COPY nomtable TO 'C:/.../maTable.csv' WITH CSV;
```

Ou

```
COPY (select ...) TO 'C:/.../maTable.csv' WITH CSV;
```

Importation format Excel

COPY nomtable FROM 'C:/.../maTable.csv' WITH CSV;

La table doit déjà exister.

Autres formats d'import-export

<http://www.postgresql.org/docs/8.3/static/sql-copy.html>

Les index

Principes

Les index sont créés automatiquement sur les clés primaires et les attributs uniques.

Il y a 4 grandes catégories d'index :

- les B-tree, qui correspondent à des arbres balancés. C'est l'index par défaut.
- Les Hash, index hachés, qui ne supporte que la recherche sur une égalité
- L'indexation full text : Gist et GIN
- L'indexation spatiale : R-tree (tableau à deux dimensions).

Les index peuvent s'appliquer à plusieurs colonnes à la fois.

L'indexation peut se faire sur une colonne traitée : par exemple sur la mise en minuscule de la colonne.

Les index peuvent être associés à des tablespaces : `USING INDEX TABLESPACE nomTablespace`.

Create Index

```
CREATE INDEX nomIndex ON nomTable
  [ USING { btree | hash | gist | gin } ]
  ( { column | ( expression ) } [, ...] )
  [ TABLESPACE tablespace ]
  [ WHERE predicat ]
```

L'expression peut être: `lower(column)`

Le prédicat consiste à définir un sous-ensemble indexé dans la table d'origine.

SELECT

Opérateurs et Fonctions mathématiques

<http://www.postgresql.org/docs/current/static/functions-math.html>

=, !=, <>

abs, ceil, floor, mod, power, random, round, sqrt, trunc

Opérations ensemblistes

UNION : union ensembliste

INTERSECT : intersection ensembliste

EXCEPT : soustraction ensembliste

IN et ROW

En cas de comparaison avec plusieurs attributs, on peut ajouter ROW devant le n-uplet d'attributs à comparer :

... ROW(attribut1, attribut2) IN (select attribut1, attribut2 FROM...

On peut aussi se passer du ROW !

Fonctions de traitement de chaînes

<http://www.postgresql.org/docs/8.1/static/functions-string.html>

upper, lower, initcap, substring, substr, rephrase, chr, ascii, length, char_length, to_ascii, etc.

TO_CHAR et EXTRACT

<http://www.postgresql.org/docs/8.1/static/functions-formatting.html>

```
select to_char(now( ),'Day') ; // renvoie "jeudi"
```

```
select to_char(now( ),'Month') ; // renvoie "mars"
```

```
select to_char(now( ),'YYYY') ; // renvoie "2009"
```

```
select extract( dow FROM now() ); // renvoie 4 pour jeudi
```

```
select extract( day FROM now() ); // renvoie 14 pour le 14 mars
```

```
select extract( month FROM now() ); // renvoie 3 pour mars
```

LIKE, ILIKE et SIMILAR

<http://www.postgresql.org/docs/7.4/static/functions-matching.html>

LIKE, not LIKE

iLIKE : insensible à la casse

SIMILAR TO : pattern de comparaison avec des expressions régulières SQL 99

~ : pattern de comparaison avec des expressions régulières POSIX (les mêmes que pour les CHECK).

~* : pour insensible à la casse.

!~, !~* : pour la négation

LIMIT et OFFSET

LIMIT 20 OFFSET 10 : 20 tuples à partir du 10^{ème}

Jointure externe : droite, gauche et bilatérale

RIGHT OUTER JOIN : rajoute les tuples de la table de gauche

LEFT OUTER JOIN: rajoute les tuples de la table de droite

FULL OUTER JOIN: rajoute les tuples des deux tables

TP

1) PgAdmin III

Présentation de l'outil

La fenêtre comprend 4 éléments :

- Une arborescence à gauche : c'est l'arborescence des serveurs (c'est-à-dire des clusters, des groupes de bases de données. En général, il n'y a qu'un serveur.
- Une fenêtre de propriétés : elle décrit les propriétés de l'objet sélectionné dans l'arborescence.
- Un panneau SQL : il décrit le code SQL de l'objet sélectionné dans l'arborescence.
- Un menu horizontal et des icônes de raccourcis en haut : c'est un menu contextuel qui varie selon l'objet sélectionné dans l'arborescence

Démarre le serveur

Double cliquer sur le serveur ou, serveur sélectionné : « menu / démarrer le service »

Circulation dans l'arborescence

Par défaut on trouve :

- Les bases de données : une seule à l'installation : postgres.
- Les espaces de tables : 2 à l'installation
- Les rôles groupe : ce sont des rôles (utilisateurs) type
- Les rôles de connexion : les utilisateurs du système

On peut alors pénétrer dans l'arborescence de façon classique et accéder à tous les objets de la BD.

Mécanique générale

Pour chaque classificateurs (bases de données, schéma), on peut

- Créer un nouvel objet : bouton droit. La fenêtre permet de paramétrer tous les éléments de définition de l'objet. On peut vérifier le code SQL généré au fur et à mesure.

Pour chaque objet existant, on peut

- Accéder au script SQL-DDL et le modifier : bouton droit

A partir des tables, on peut

- Accéder au script SQL-Select et le modifier : bouton droit.

2) EcolingScript n°3

Modèle relationnel

Etudiants (**NET**, nom, prenom, email)

Examens (**NEX**, matiere, prof, session, annee, niveau, type, duree)

Groupe (**NGR**, (promo, annee, groupe))

Epreuves (**NEP**, (#NGR, dateheure), salle, #NEX)

Evaluer (**#NEP, #NET**, note)

Participer (**#NET, #NGR**)

Dans la table Epreuves, (**#NGR, dateheure**) est une clé secondaire.

Dans la table Groupes, (promo, annee, groupe) est une clé secondaire.

Les Epreuves sont des instances d'un Examen

Les étudiants sont évalués dans une épreuve (qui correspond à un examen): il y a une note. En cas d'absence, ils ne sont pas dans la table.

Exercices

Connectez-vous en tant que super-utilisateur.

Vous allez utiliser le script : **EcolingTPpg03.txt**. **Il faut faire des mises à jour :**

Ajouter au script la création d'une BD appelée INSIA.

Choisissez un nom de schéma.

Mettez à jour le search_path pour que vous puissiez accéder aux tables.

C'est un script mysql. Faites les mises à jour.

Ajouter toutes les contraintes d'intégrité possibles.

Créer un domaine qui gère le format des adresses mail. Au minimum avec des like (ou similar to : chercher dans la doc), au mieux avec une expression régulière (cf expression régulière MySQL et la doc POSTGRES).

Consultation de la BD

1. Pour chaque table verbes, afficher son contenu en sur-informant au maximum la table. Pour faciliter la lecture, on projettera uniquement les clés primaires et significatives.
2. Quel est le parcours d'un étudiant donné : quelles années, quelles promos, quels groupes ?
3. Enregistrer le résultat précédent dans un fichier.

Questions statistiques

4. Quelle sont les moyennes générales de l'étudiant n°1624 par année ?
5. Quelle est la moyenne des groupes pour les examens 4, 5 et 6
6. Quels sont les étudiants ayant plus que la moyenne des étudiants pour l'épreuve n°1 ?

Questions à jointures externes ou opérations ensemblistes

7. Quelles sont les épreuves qui n'ont pas donné lieu à des évaluations ?
8. Quels sont les étudiants qui n'ont pas été évalués pour l'épreuve n°2 et qui auraient du l'être (autrement dit qui participent au groupe correspondant à l'épreuve.

On aura intérêt à commencer par traiter les requêtes :

- Tous les étudiants ayant passé l'épreuve 2
 - Tous les étudiants d'un groupe donné
 - Tous les étudiants qui n'ont pas été évalués pour telle épreuves (même ceux qui ne sont pas concernés par l'épreuve)
9. Quels sont les étudiants de SIGL 2, promo 2009 n'ayant pas passé toutes les épreuves de leur groupe ?
 10. Quels sont les numéros des étudiants n'ayant pas passé les épreuves de leur groupe ?

On commencera par traiter les requêtes :

Tous les étudiants évalués par groupe et par épreuve (le groupe étant celui de l'épreuve)

Tous les étudiants prévus par épreuves, par groupe et par épreuve.