Le langage SQL (première partie)

© Olivier Caron

Polytech'Lille

Plan

- √ Le S.G.B.D. postgres
- √ Le langage SQL
  - ▶ Langage de manipulation de données
  - ▶ Langage de requêtes

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)
- √ Successeur du projet Ingres (d'ou son nom)

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)
- √ Successeur du projet Ingres (d'ou son nom)
- √ `http://www.postgresql.org/` (livre intégré au site)

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)
- √ Successeur du projet Ingres (d'ou son nom)
- √ `http://www.postgresql.org/` (livre intégré au site)
- √ Fonctionne sous Unix (linux, solaris, Mac OS X)
- √ Portage sous windows (version 8.0)

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)
- √ Successeur du projet Ingres (d'ou son nom)
- √ `http://www.postgresql.org/` (livre intégré au site)
- √ Fonctionne sous Unix (linux, solaris, Mac OS X)
- √ Portage sous windows (version 8.0)
- √ Gratuit, mais il existe des versions commerciales

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)
- √ Successeur du projet Ingres (d'ou son nom)
- √ `http://www.postgresql.org/` (livre intégré au site)
- √ Fonctionne sous Unix (linux, solaris, Mac OS X)
- √ Portage sous windows (version 8.0)
- √ Gratuit, mais il existe des versions commerciales
- √ S.G.B.D.O.R.

Quelques mots sur Postgres (1/2)

- √ Travaux de Stonebraker (Univ. Berkeley)
- √ Successeur du projet Ingres (d'ou son nom)
- √ `http://www.postgresql.org/` (livre intégré au site)
- √ Fonctionne sous Unix (linux, solaris, Mac OS X)
- √ Portage sous windows (version 8.0)
- √ Gratuit, mais il existe des versions commerciales
- √ S.G.B.D.O.R.
- √ Architecture client/serveur (processus, TCP-IP)

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)
- √ Plusieurs interfaces de programmation (C, python, php, tcl-tk, JDBC, ODBC,. . .)

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)
- √ Plusieurs interfaces de programmation (C, python, php, tcl-tk, JDBC, ODBC,. . .)
- √ *Gère plusieurs utilisateurs (sécurité)*

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)
- √ Plusieurs interfaces de programmation (C, python, php, tcl-tk, JDBC, ODBC,. . .)
- √ *Gère plusieurs utilisateurs (sécurité)*
- √ Définitions de contraintes d'intégrité

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)
- √ Plusieurs interfaces de programmation (C, python, php, tcl-tk, JDBC, ODBC,. . .)
- √ *Gère plusieurs utilisateurs (sécurité)*
- √ Définitions de contraintes d'intégrité
- √ *Gère les accès concurrents*

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)
- √ Plusieurs interfaces de programmation (C, python, php, tcl-tk, JDBC, ODBC,. . .)
- √ *Gère plusieurs utilisateurs (sécurité)*
- √ Définitions de contraintes d'intégrité
- √ *Gère les accès concurrents*
- √ *Procédures intégrées*

Quelques mots sur Postgres (2/2)

- √ Compatible SQL-92
- √ De nombreux éléments de SQL-99
- √ Langage QBE (pgaccess)
- √ Plusieurs interfaces de programmation (C, python, php, tcl-tk, JDBC, ODBC,. . .)
- √ *Gère plusieurs utilisateurs (sécurité)*
- √ Définitions de contraintes d'intégrité
- √ *Gère les accès concurrents*
- √ *Procédures intégrées*
- √ *Spécificités objet*

Les utilisateurs Postgres

√ L'utilisateur **postgres** dispose de tous les droits (root), création bases, utilisateurs, destruction,. . .

Les utilisateurs Postgres

- √ L'utilisateur **postgres** dispose de tous les droits (root), création bases, utilisateurs, destruction,. . .
- √ Les **utilisateurs-administrateurs** : peuvent créer des bases, peuvent autoriser d'autres utilisateurs à accèder à ces bases.
  Certains utilisateurs-administrateurs peuvent créer d'autres utilisateurs.

Les utilisateurs Postgres

- √ L'utilisateur **postgres** dispose de tous les droits (root), création bases, utilisateurs, destruction,. . .
- √ Les **utilisateurs-administrateurs** : peuvent créer des bases, peuvent autoriser d'autres utilisateurs à accèder à ces bases.
  Certains utilisateurs-administrateurs peuvent créer d'autres utilisateurs.
- √ Les **utilisateurs** peuvent accéder à des bases (selon les droits)

Le réseau Polytech'Lille

- √ Un serveur postgres situé sur le serveur `weppes.studserv.deule.net`
- √ Chaque étudiant à un compte **utilisateur-administrateur**
- √ Accès disponible à distance sur toutes les machines
- √ Configuration initiale (`.bashrc`) :

```
export PGHOST=weppes.studserv.deule.net
```

- √ Par défaut : comptePostgres = compteUnix, password (postgres)
- √ Modifier le mot de passe (sous psql) :

```
ALTER USER comptePostgres WITH password 'motdepasse'
```

Les commandes de base

- √ Commandes sous unix
- √ Création d'une base ([] facultatif) :

```
createdb  nomBase [ -U comptePostgres ]
```

Les commandes de base

- √ Commandes sous unix
- √ Création d'une base ([] facultatif) :

```
createdb  nomBase [ -U comptePostgres ]
```

- √ Destruction d'une base :

```
dropdb nomBase [ -U comptePostgres ]
```

Les commandes de base

- √ Commandes sous unix
- √ Création d'une base ([] facultatif) :

```
createdb  nomBase [ -U comptePostgres ]
```

- √ Destruction d'une base :

```
dropdb nomBase [ -U comptePostgres ]
```

- √ Accès à une base :

```
psql nomBase [ -U comptePostgres ]
```

Environnement psql

√ Interface texte :-) ou :-(

Environnement psql

- √ Interface texte :-) ou :-(
- √ Reconnait toutes les requêtes SQL !

Environnement psql

- √ Interface texte :-) ou :-(
- √ Reconnait toutes les requêtes SQL !
- √ Et bien plus encore (aide en ligne,. . .)

Le langage SQL
Structured Query Language

√ Historique :

- En 70, Travaux de Codd
- De 72 à 75, IBM invente SEQUEL.

Le langage SQL
Structured Query Language

√ Historique :

- En 70, Travaux de Codd
- De 72 à 75, IBM invente SEQUEL.
- Puis SEQUEL/2 en 77 pour le prototype System-R de SGBD**R**.

Le langage SQL
Structured Query Language

√ Historique :

- En 70, Travaux de Codd
- De 72 à 75, IBM invente SEQUEL.
- Puis SEQUEL/2 en 77 pour le prototype System-R de SGBD**R**.
- Ce langage donne naissance à SQL
- Parallèlement, Ingres développe le langage QUEL en 76

Le langage SQL
Structured Query Language

√ Historique :

- En 70, Travaux de Codd
- De 72 à 75, IBM invente SEQUEL.
- Puis SEQUEL/2 en 77 pour le prototype System-R de SGBD**R**.
- Ce langage donne naissance à SQL
- Parallèlement, Ingres développe le langage QUEL en 76
- Dès 79, Oracle utilise SQL

Le langage SQL
Structured Query Language

√ Historique :

- En 70, Travaux de Codd
- De 72 à 75, IBM invente SEQUEL.
- Puis SEQUEL/2 en 77 pour le prototype System-R de SGBD**R**.
- Ce langage donne naissance à SQL
- Parallèlement, Ingres développe le langage QUEL en 76
- Dès 79, Oracle utilise SQL
- En 81, IBM sort SQL/DS (même année que le PC)
- En 83, IBM sort DB2 (héritier de System-R) exploitant SQL

Le langage SQL (2/2)

√ Standard de fait, puis véritable standard - Norme ISO

Le langage SQL (2/2)

- √ Standard de fait, puis véritable standard - Norme ISO
- √ Facile à utiliser car il se réfère toujours aux lignes ou colonnes d'une table

Le langage SQL (2/2)

- √ Standard de fait, puis véritable standard - Norme ISO
- √ Facile à utiliser car il se réfère toujours aux lignes ou colonnes d'une table
- √ Beaucoup plus facile que l'utilisation directe des opérateurs algébriques !
- √ Objectif des SGBD : applicable aux non-informaticiens ?

Le langage SQL (2/2)

- √ Standard de fait, puis véritable standard - Norme ISO
- √ Facile à utiliser car il se réfère toujours aux lignes ou colonnes d'une table
- √ Beaucoup plus facile que l'utilisation directe des opérateurs algébriques !
- √ Objectif des SGBD : applicable aux non-informaticiens ?
- √ Langage tout en un : consultation mais aussi création, modification, suppression,. . .
- √ Langage en constante évolution : SQL-86, SQL-89, SQL-92 (SQL 2), SQL-99 (SQL 3, non terminé)

Syntaxe des commandes

√ Les commandes commencent par un mot clé servant à nommer l'opération de base à exécuter. Chaque commande SQL doit remplir deux exigences :

- Indiquer les données sur lesquelles elle opère (un ensemble de lignes stockées dans une ou plusieurs classes)
- Indiquer l'opération à exécuter sur ces données

Distinction des commandes

- √ Les commandes d'administration (utilisateur-administrateur) : création, suppression de tables, d'index, de vues, *de droits d'accès, de fonctions ou procédures* (programme de GIS 2), modification de structure de tables.
- √ Les commandes d'utilisation : consultation, modification de lignes, suppression de lignes.

Les domaines par défaut

√ Les numériques : `integer, smallint, double, float,...`
exemple : 23, -122, 3.4, -6.7,. . .

Les domaines par défaut

- √ Les numériques : `integer, smallint, double, float,...`
  exemple : 23, -122, 3.4, -6.7,. . .
- √ Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : 'v', 'la vie est un long fleuve ...'

Les domaines par défaut

- √ Les numériques : `integer, smallint, double, float,...`
  exemple : 23, -122, 3.4, -6.7,. . .
- √ Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : 'v', 'la vie est un long fleuve ...'
- √ Le type date (format configurable)
  exemple : '2002-02-28'

Les domaines par défaut

- √ Les numériques : `integer, smallint, double, float,...`
  exemple : 23, -122, 3.4, -6.7,. . .
- √ Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : 'v', 'la vie est un long fleuve ...'
- √ Le type date (format configurable)
  exemple : '2002-02-28'
- √ Le type boolean : `bool`
  exemples :'f', 't'

Les domaines par défaut

- √ Les numériques : `integer, smallint, double, float,...`
  exemple : 23, -122, 3.4, -6.7,. . .
- √ Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : 'v', 'la vie est un long fleuve ...'
- √ Le type date (format configurable)
  exemple : '2002-02-28'
- √ Le type boolean : `bool`
  exemples :'f', 't'
- √ Et bien d'autres encore. . .

Création de tables

√ Syntaxe postgres (très simplifiée)
([] : 0 ou 1 occurrence, {} 0 ou plusieurs occurrences, | :alternative) :

```
CREATE TABLE table_name (
    { column_name type [ column_constraint [ ... ] ]
      | table_constraint } )
column_constraint ::
[ CONSTRAINT constraint_name ]
  PRIMARY KEY | REFERENCES table_name [ ( column [, ... ] ) ]
table_constraint::
[ CONSTRAINT constraint_name ]
  PRIMARY KEY ( column_name [, ... ] ) |
  FOREIGN KEY ( column_name [, ... ] )
     REFERENCES table_name [ ( column [, ... ] ) ]
```

Exemple Création (1/3)

```
CREATE TABLE utilisateur (
  num_u INTEGER PRIMARY KEY,
  nom VARCHAR(30), prenom VARCHAR(30)
) ;

CREATE TABLE auteur (
  num_a INTEGER,
  nom VARCHAR(30),
  CONSTRAINT cle_auteur PRIMARY KEY (num_a)
) ;
```

Exemple Création (2/3)

```
CREATE TABLE editeur (
  num_e INTEGER PRIMARY KEY,
  nom VARCHAR(30), adresse1 VARCHAR(30),
  adresse2 VARCHAR(30), code_postal integer, ville VARCHAR(30)
) ;

CREATE TABLE livre (
  num_l INTEGER,
  titre text,
  auteur INTEGER REFERENCES auteur,
  PRIMARY KEY (num_l)
) ;
```

Exemple Création (3/3)

```
CREATE TABLE reserve (
 num_l INTEGER REFERENCES livre, num_u INTEGER REFERENCES utilisateur,
 PRIMARY KEY (num_l, num_u)
) ;
CREATE TABLE emprunte (
 num_l INTEGER REFERENCES livre, num_u INTEGER REFERENCES utilisateur,
 PRIMARY KEY (num_l, num_u)
) ;
CREATE TABLE edite_par (
  num_l INTEGER references livre,
  num_e INTEGER references editeur,
  date_edition date,
  PRIMARY KEY (num_l,num_e, date_edition)
) ;
```

Optimisation d'accès au tables

√ Mécanisme d'index pour favoriser l'accès

Optimisation d'accès au tables

- √ Mécanisme d'index pour favoriser l'accès
- √ Pour chaque clé, création automatique d'un index

Optimisation d'accès au tables

- √ Mécanisme d'index pour favoriser l'accès
- √ Pour chaque clé, création automatique d'un index
- √ Avantage : accès rapide lors d'une consultation

Optimisation d'accès au tables

- √ Mécanisme d'index pour favoriser l'accès
- √ Pour chaque clé, création automatique d'un index
- √ Avantage : accès rapide lors d'une consultation
- √ Inconvénient : modification, suppression de lignes

  $\longrightarrow$ mise à jour du ou des index
- √ Exemple :

```
create index livre_titre_index on livre (titre) ;
```

Suppression de tables, d'index

√ A chaque commande `CREATE`, une commande `DROP`

Suppression de tables, d'index

- √ A chaque commande `CREATE`, une commande `DROP`
- √ Syntaxes :

```
DROP TABLE name {, name}
DROP INDEX index_name [, ...]
```

Insertion de lignes

√ Parfois considéré comme une commande purement administrateur

Insertion de lignes

- √ Parfois considéré comme une commande purement administrateur
- √ Syntaxe (simplifiée) :

```
INSERT INTO table [ ( column {, ...} ) ]
    VALUES ( expression {, ...} ) | SELECT query
```

Exemples insertion

√ On insère une ligne complète :

```
INSERT INTO auteur VALUES (1, 'Uderzo');
```

Respecter l'ordre des colonnes définies à la création (et le type)

Exemples insertion

- √ On insère une ligne complète :

```
INSERT INTO auteur VALUES (1, 'Uderzo');
```

  Respecter l'ordre des colonnes définies à la création (et le type)

- √ On insère une ligne complète en spécifiant les colonnes :

```
INSERT INTO auteur (nom, num_a) VALUES ('Franquin', 2);
```

Exemples insertion

- √ On insère une ligne complète :

```
INSERT INTO auteur VALUES (1, 'Uderzo');
```

  Respecter l'ordre des colonnes définies à la création (et le type)

- √ On insère une ligne complète en spécifiant les colonnes :

```
INSERT INTO auteur (nom, num_a) VALUES ('Franquin', 2);
```

- √ On insère une ligne incomplète :

```
INSERT INTO livre(num_l, titre) VALUES (1, 'L \'Odyssée d\'Astérix')
```

- √ Attention aux incohérences de la base !

Suppression de lignes

- Syntaxe : `DELETE FROM nom_table [ WHERE condition ]`
- Exemples :

```
DELETE FROM utilisateurs ;
```

Suppression de lignes

- Syntaxe : `DELETE FROM nom_table [ WHERE condition ]`
- Exemples :

```
DELETE FROM utilisateurs ;

DELETE FROM livres where num_l=1 ;
```

Modification de lignes

√ Syntaxe :

```
UPDATE nom_table SET col = expression {, col = expression}
[ WHERE condition ]
```

Modification de lignes

√ Syntaxe :

```
UPDATE nom_table SET col = expression {, col = expression}
[ WHERE condition ]
```

√ Exemples :

- Modifier une colonne, pour une ligne :

```
UPDATE livre SET auteur=1 WHERE num_l=1
```

Modification de lignes

- Syntaxe :

```
UPDATE nom_table SET col = expression {, col = expression}
[ WHERE condition ]
```

- Exemples :
  - Modifier une colonne, pour une ligne :

```
UPDATE livre SET auteur=1 WHERE num_l=1
```

  - Modifier plusieurs colonnes pour une ligne :

```
UPDATE auteur SET nom='Victor Hugo', num_a=4 where num_a=3
```

  - Modifier toutes les lignes :

```
UPDATE personnel SET salaire=salaire+0.10*salaire
```

Modifier une table

- De plus en plus de possibilités au cours des versions (ex : supprimer une colonne)
- Syntaxes :

```
ALTER TABLE nom_table ADD [ COLUMN ] nom_colonne type

ALTER TABLE nom_table RENAME [ COLUMN ] nom_colonne TO nouveau_nom

ALTER TABLE nom_table RENAME TO nouveau_nom_table
ALTER TABLE nom_table DROP [ COLUMN] nom_colonne
ALTER TABLE nom_table OWNER to nouveau_proprietaire
...
```