Compléments au cours de BDR en vue du projet du S6

Mysql/Postgresql/Licence

- Deux logiciels libres, mais pas même type de licence
- Mysql : GPL pour mysql community edition
- Postgresql license : Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

Mysql/Norme SQL

- Mysql s'autorise à dévier de la norme SQL, pour la bonne cause dit il
- Exemples de petites différences
- UPDATE t1 SET col1 = col1 + 1, col2 = col1;
- -- espace pour les commentaires
- CONCAT et pas ||

- Ajout de fonctions non standard : exemple group_concat, etc

- Inconvénient : non respect de la norme
- Avantage : rapidité

- Postgresql n'implémente pas non plus totalement la norme, mais ne s'en écarte pas souvent

- Mysql : le meilleur choix si assez peu d'utilisateurs et la majorité des opérations sont des lectures

- Postgresql : le meilleur choix si le respect de la norme est obligatoire, si on ne veut absolument pas jouer avec l'intégrité des données, si très grand nombre d'utilisateurs. Mais a petite echelle moins rapide que mysql.

Postgresql : Authentification du client)

- Le contrôle se fait en premier lieu via un fichier de configuration usuellement nommé pg_hba.conf (répertoire des données de la base) [hba = host base authentification)

TYPE	DATABASE	USER	ADDRESS IP-ADRESS	IP_MASK	METHOD
local	all	all			trust
host	nomdebase	mike	192.168.54.1/32		md5
hostssl		all	0.0.0.0/0		ident
hostnossl					…..

Postgresql : rôle

- Postgresql gère les permissions d'accès en utilisant le concept de rôle. Un rôle est aussi bien un utilisateur qu'un groupe d'utilisateurs. Les rôles possède des objets (tables, fonctions) et peuvent donner des droits sur ces objets aux autre rôles.
- Il est possible de donner à un rôle un droit d'appartenance à un autre rôle (GRANT role1 to role2)
- Un rôle qui correspond a un utilisateur doit avoir le droit de LOGIN

Nom	Description	Syntaxe
LOGIN	Permet au rôle de se connecter à une base de données	CREATE ROLE maxim LOGIN
SUPERUSER	Permet au rôle de tout faire au niveau de la base de données	CREATE ROLE postgresbis SUPERUSER
CREATEDB	Permet au rôle de créer des bases de données	CREATE ROLE CREATEDB
CREATEROLE	Permet au rôle de créer d'autres rôles	CREATE ROLE maxim CREATEROLE
PASSWORD	Assigne un mot de passe à un rôle	CREATE ROLE maxim PASSWORD 'xxxxxx'
INHERIT/NOINHERIT	Permet, ou pas, à un rôle d'hériter des attributs d'autres rôles	CREATE ROLE maxim INHERIT

Postgresql : exemple d'organisation « classique »

Le principe d'organisation se présente en plusieurs points synthétiques

- un schéma par utilisateur pour stocker et partager le travail
- un schéma public non accessible en édition
- plusieurs schémas accueillant les données de références ou autres

ces deux derniers restant accessible en lecture.

- A l'installation, un super-utilisateur est créé qui possède tous les droits, par défaut il s'appelle postgres. Pour bien faire, cet utilisateur ne devrait être utilisé qu'en cas de nécessité.

- Un rôle d'administrateur peut être crée : C'est le rôle qui possédera tous les droits d'administration et de maintenance des bases sans être pour autant un super-utilisateur. Il lui faudra pour ce faire les droits suivants LOGIN, CREATEDB, CREATEROLE ,NOINHERIT

- CREATE ROLE admin NOSUPERUSER INHERIT CREATEDB CREATEROLE REPLICATION;

- En supposant que tous les utilisateurs doivent avoir les même droits: tous les droits sur son propre schema et droit de lecture sur le schéma des autres
- On va créer un rôle par utilisateur et de plus un rôle (grouê) «utilisateur». Celui-ci permet, par héritage de droits, la consultation de toutes les tables de tous les schémas par tous les membres du «rôle-groupe» «utilisateur».
- Chaque utilisateur possède un schéma à son nom où lui seul peut créer des tables

- CREATE ROLE utilisateur NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
- --Création du rôle nominatif et application des droits du groupe utilisateur
- CREATE ROLE <login> WITHPASSWORD '<password_choisi>‘ NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION LOGIN;
- GRANT utilisateur TO <login>;
- --Création d'un schéma pour chaque utilisateur
- CREATE SCHEMA u_<login> AUTHORIZATION <login>;

- --Chaque utilisateur donne accès en lecture à tous les autres utilisateurs aux tables de son schema
- GRANT ALL ON SCHEMA u_<login> TO <login>;
- GRANT USAGE ON SCHEMA u_<login> TO utilisateur;
- ALTER DEFAULT PRIVILEGES IN SCHEMA u_<login> GRANT SELECT ON TABLES TO utilisateur;

- --Retrait des droits sur le schema public
- --Acces en lecture au groupe utilisateur
- REVOKE ALL ON SCHEMA public FROM public;
- GRANT USAGE ON SCHEMA public TO utilisateur;
- ALTER DEFAULT PRIVILEGES IN SCHEMA public

GRANT SELECT ON TABLES TO utilisateur;

-- si un nouvel utilisateur membre de utilisateurs est créé il aura les bons droits sur le schéma public

- Les membres d’un rôle peuvent utiliser les privilèges de ce rôle de deux manières différentes.
- Chaque membre du groupe peut explicitement faire SET ROLE pour devenir explicitement ce rôle. Dans cet état la session a accès à tous les privilèges du groupe rôle plutot qu’aux privilèges du rôle de login original et tout objet créé est la propriété du role groupe
- D’autre part les membres d’un groupe s’ils ont l’attribut INHERIT récupèrent systématiquement les privilèges des rôles auxquels ils appartiennent, y compris les privilèges hérités par ceux-ci.

- CREATE ROLE joe LOGIN INHERIT;
- CREATE ROLE admin NOINHERIT;
- CREATE ROLE wheel NOINHERIT;
- GRANT admin TO joe;
- GRANT wheel TO admin;
- Immédiatement après s'être connecté sous le rôle joe, une session aura les privilèges de joe plus les privilèges de admin, car joe hérite des privilèges , mais ils n'aura pas les privilèges de wheel, car joe est un membre indirect de wheel (via admin) et admin a l'attribut noinherit
- Après SET ROLE admin; the session aura les privilèges de admin seulement
- After SET ROLE wheel; the session aura les privilèges de wheel uniquement.

Suppression d’un rôle

- On ne peut supprimer un rôle que s’il ne possède aucun objet et n’a aucune permission
  - ALTER TABLE bobs_table OWNER TO alice;
  - REASSIGN OWNED
  - DROP OWNED

- Il faut un droit de CONNECT sur la base de donnée
- Il faut un droit de USAGE sur le schéma (par défaut sur le schéma public tout le monde a ce droit)
- Il faut tous les droits nécessaires sur les objets du schéma (table, vue, fonctions, séquences…)

Remarque

- Pour se connecter à une base de donnée, un utilisateur doit passer le contrôle de pg_hba.conf et doit aussi avoir le privilège de CONNECTion à la base de données.
- Il est usuellement plus facile de contrôler qui a doit de se connecter à quelle base de donnée via granting/revoking CONNECT privilège plutôt que via des lignes du fichier ph_hba.conf

La table pg_authid contient les infos sur les roles

Name	Type	Description
oid	oid	Row identifier (hidden attribute; must be explicitly selected)
rolname	name	Role name
rolsuper	bool	Role has superuser privileges
rolinherit	bool	Role automatically inherits privileges of roles it is a member of
rolcreaterole	bool	Role can create more roles
rolcreatedb	bool	Role can create databases
rolcanlogin	bool	Role can log in. That is, this role can be given as the initial session authorization identifier
rolreplication	bool	Role is a replication role. A replication role can initiate replication connections and create and drop replication slots.
rolbypassrls	bool	Role bypasses every row level security policy, see Section 5.7 for more information.
rolconnlimit	int4	For roles that can log in, this sets maximum number of concurrent connections this role can make. -1 means no limit.
rolpassword	text	Password (possibly encrypted); null if none. The format depends on the form of encryption used.
rolvaliduntil	timestamptz	Password expiry time (only used for password authentication); null if no expiration

ATTENTION

- Les privilèges sur les diffèrents objets sont indépendants. Un privilège GRANT CONNECT sur une base de données ne donne pas les droits sur les schémas qu'elle contient. De même, un privilège GRANT USAGE sur un schéma ne donne pas les droits sur les tables du schéma

- Si vous avez un privilege SELECT sur une table, mais pas les droits d'usage sur le schema qui la contient, vous ne pouvez pas acceder à la table !

MySQL premier niveau de contrôle

- Quand un client se connecte au serveur mysql, le serveur utilise le nom donné par le client et l'hôte du client pour sélectionner la bonne ligne (compte) de la table mysql.user.
- Le serveur va alors authentifier le client, en utilisant la ligne de la table pour savoir quel plugin d'authentification appliquer.

user and db Table Columns

Table Name	user	db
Scope columns	Host	Host
	User	Db
		User
Privilege columns	Select_priv	Select_priv
	Insert_priv	Insert_priv
	Update_priv	Update_priv
	Delete_priv	Delete_priv
	Index_priv	Index_priv
	Alter_priv	Alter_priv
	Create_priv	Create_priv
	Drop_priv	Drop_priv
	Grant_priv	Grant_priv
	Create_view_priv	Create_view_priv
	Show_view_priv	Show_view_priv
	Create_routine_priv	Create_routine_priv
	Alter_routine_priv	Alter_routine_priv
	Execute_priv	Execute_priv
	Trigger_priv	Trigger_priv
	Event_priv	Event_priv
	Create_tmp_table_priv	Create_tmp_table_priv
	Lock_tables_priv	Lock_tables_priv
	References_priv	References_priv
	Reload_priv	
	Shutdown_priv	
	Process_priv	

Création d’utilisateur en mysql

- -- thibault peut se connecter à partir de n'importe quel hôte dont l'adresse IP commence par 194.28.12.
- CREATE USER 'thibault'@'194.28.12.%' IDENTIFIED BY 'basketball8';

- -- joelle peut se connecter à partir de n'importe quel hôte du domaine brab.net
- CREATE USER 'joelle'@'%.brab.net' IDENTIFIED BY 'singingisfun';

- -- hannah peut se connecter à partir de n'importe quel hôte
- CREATE USER 'hannah'@'%' IDENTIFIED BY 'looking4sun';

Les rôles mysql

- Un rôle mysql est un ensemble de privilèges a qui l'on donne un nom. On peut accorder ou revoquer des privilèges à un rôle.
- Un compte utilisateur peut se voir accorder (grant) des rôles
- Les commandes mysql
- CREATE ROLE DROP ROLE.
- GRANT REVOKE s'appliquent aux users et aux rôles.
- SHOW GRANTS
- SET DEFAULT ROLE spécifie quels sont les rôles actifs par défault.
- SET ROLE change les rôles actifs de la session.

Exemple d’utilisation des rôles en mysql

- CREATE ROLE 'app_developer', 'app_read', 'app_write';
- GRANT ALL ON app_db.* TO 'app_developer';
- GRANT SELECT ON app_db.* TO 'app_read';
- GRANT INSERT, UPDATE, DELETE ON app_db.* TO 'app_write';
    - CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1pass';
    - CREATE USER 'read_user1'@'localhost' IDENTIFIED BY 'read_user1pass';
    - CREATE USER 'read_user2'@'localhost' IDENTIFIED BY 'read_user2pass';
    - CREATE USER 'rw_user1'@'localhost' IDENTIFIED BY 'rw_user1pass';
- GRANT 'app_developer' TO 'dev1'@'localhost';
- GRANT 'app_read' TO 'read_user1'@'localhost', 'read_user2'@'localhost';
- GRANT 'app_read', 'app_write' TO 'rw_user1'@'localhost';

Gestion fine des privilèges

- GRANT ALL ON mydb.* TO 'someuser'@'somehost';
- GRANT SELECT, INSERT ON mydb.* TO 'someuser'@'somehost';

- GRANT ALL ON mydb.mytbl TO 'someuser'@'somehost';
- GRANT SELECT, INSERT ON mydb.mytbl TO 'someuser'@'somehost';

- GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO 'someuser'@'somehost‘;
- -- insertion sur seulement des colonnes pas standard !!

Etablir une connection à une base de données

- Il vous faut un serveur de bases de données .
- Pour mysql vous pouvez l'installer seul, ou installer un wamp, lamp, xamp qui installera d'un coup un serveur apache, un serveur mysql et php [mais vous n'avez besoin que de mysql]

- Pensez a tester inependament les installations !!

Créer un serveur http avec node.js

```
var http = require('http');

var serv = http.createServer( //création d'un serveur web

  function (req, res) { //callback sur les requêtes HTTP
     //construction d'une réponse HTTP
     res.writeHead(200, {'Content-Type': 'text/plain'});
     res.write('Hello world !');

     res.end(); //envoi de la réponse
   }

);

serv.listen(8000); //commence à accepter les requêtes

console.log("Server running at http://localhost:8000");
```

Se connecter une base de donnée mysql avec node.js

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'claudine',
  password : 'claudine',
  database : 'claudine'
});

connection.connect();

connection.query('SELECT * from test', function(err, rows, fields) {
  if (!err)
    console.log('The solution is: ', rows);
  else
    console.log('Error while performing Query.');
});

connection.end();
```

Exécuter une requête suite

```
connection.query('SELECT * from test', function(err, rows, fields) {
  if (!err)
    console.log(‘La premiere ligne est: ', rows[0]);

  else
    console.log('Error while performing Query.');
});

connection.query('SELECT * from test', function(err, rows, fields) {
  if (!err)
    console.log(La cle de la premiere ligne est ', rows[0].cle);

  else
    console.log('Error while performing Query.');
});
```

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "claudine",
  password: "claudine",
  database: "claudine"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  //Insert a record in the "customers" table:
  var sql1 ="CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))";
  con.query(sql1, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
  var sql2 = "INSERT INTO customers (name, address) VALUES ('Company Inc', 'Highway 37')";
  con.query(sql2, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```