A practical (pragmatic) multi-tenant cluster

(Mental meandering)

by

Rod Anderson

raanders@acm.org
**Credits**

- Thomas Jacobs  
  http://wiki.postgresql.org/wiki/Shared_Database_Hosting  
  http://www.internet24.de/

- Thomas Jacobs  
  Scott Marlowe  
  Robert Treat  

- Tom Lane  
  For the gentle suggestion to just try it when I asked about using '@' in role names.
What is a multi-tenant cluster?

• A single PostgreSQL database cluster.

• One (or more) databases per customer.

• One chunk of hardware.

• Allowing many tenants to run the same applications in their own database.
Why a multi-tenant cluster?

- Single superuser (postgres?) for all databases, customers, clients.
- Sharing of memory and processor(s).
- Low impact applications or low usage needs, i.e. smaller businesses.
- Hardware goes further.
- Even a server inside a company/organization could benefit by having a database per department.
**Issues with a multi-tenant cluster**

- Missing - per database users/roles

  Actually there is a method in 8.4 but it doesn't play well with md5 requires the use of password. So if there is a possibility or concern of password "sniffing" this can't be used unless you use a SSL encrypted connection.

  See the following two links for the gory details.
  [http://www.postgresql.org/docs/current/static/runtime-config-connection.html#GUC-DB-USER-NAMESPACE](http://www.postgresql.org/docs/current/static/runtime-config-connection.html#GUC-DB-USER-NAMESPACE)
  [http://www.postgresql.org/docs/current/static/auth-methods.html](http://www.postgresql.org/docs/current/static/auth-methods.html)

- No method to add database roles when there is only a cluster superuser? That is allow create role but restrict those roles to a specific database.

- If `createuser` is granted to a database administrator, what controls and options are there on role naming.

- When deleting/removing users/roles how to clean up database spanning ACLs.
• Database resources controls.
  • Bad queries eating the database engine alive.
  • Disk and storage quotas
    • In PostgreSQL?
    • File system
    • OS?
  • CPU and memory quotas
    • PostgreSQL
    • OS
What can be done and what can we do to make it easier.

- Identify applications that can be modified to support or do support installation into a schema instead of a database.
- Get around the global user/roles
- Give each database it's own tablespace for data.
Other multi-tenant options

Though not what I'd consider true multi-tenant solutions there are

- Multiple PostgreSQL instances
- Linux-Vservers
- XEN
- BSD jails(?)
- Others?
**Steps to create a multi-tenant cluster.**

1. Come up with a database and role naming system.
2. Develop scripts to build the database and roles.
3. Method to control access to the cluster as painless as possible.
4. Modify application scripts to use a schema in a database instead of a database.
Here is what I came up with.

Databases and their controlling(?) role have short, almost mnemonic, names: acme, tsmg, snoi, etc.

Tenant user/role names are in the form dba@acme, rt_user@acme, or ledgersmb@acme.

pg_hba.conf has these four entries:

```
local   all             postgres                        md5
local   samerole        all                             md5
host    all             postgres        0/0             md5
host    samerole        all             0/0             md5
```
To install LedgerSMB I did a manual install and skipped/modified those steps that conflicted with the schema-based install.

That is: I created the database, role, and schema then ran the LedgerSMB INSTALL steps manually.

To create the database I use a shell script that makes the TABLESPACE directory and builds the SQL script. Here are some excerpts from the file.

```bash
TSDIR="/var/lib/pgsql/data/ts"
TENANT="$1"

pushd "$TSDIR"

[[ ! -d "$TENANT" ]] && mkdir "$TENANT"

chown postgres:postgres "$TENANT"

chmod 700 "$TENANT"

popd
```
The script creates and calls this SQL script. (Assuming a tenant called/named \texttt{acme}.)

\begin{verbatim}
CREATE ROLE acme NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT NOLOGIN;
CREATE ROLE "dba@acme" NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT LOGIN ENCRYPTED PASSWORD '07cc54da9f666';
GRANT acme TO "dba@acme";
CREATE TABLESPACE acme_ts OWNER "dba@acme" LOCATION '/var/lib/pgsql/data/ts/acme';
CREATE DATABASE acme WITH OWNER="dba@acme";
COMMENT ON DATABASE acme IS 'Database create 20090319';
\c acme
GRANT ALL ON SCHEMA public TO "dba@acme" WITH GRANT OPTION;
\end{verbatim}
To create a schema specifically for the application I use a script specific to that application. For LedgerSMB it generates the following SQL script.

```sql
CREATE ROLE "ledgersmb@acme" NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT LOGIN ENCRYPTED PASSWORD 'c3c20524a6621';
GRANT USAGE ON SCHEMA public TO "ledgersmb@acme";
GRANT CONNECT, TEMPORARY ON DATABASE acme TO "ledgersmb@acme";
GRANT acme TO "ledgersmb@acme";
\c acme
CREATE SCHEMA ledgersmb AUTHORIZATION "ledgersmb@acme";
COMMENT ON SCHEMA ledgersmb IS 'Schema create 20090319';
ALTER ROLE "ledgersmb@acme" SET search_path = ledgersmb, public
```