Manage PostgreSQL databases and users

Document author: anna.hammais@tyks.fi

Create databases

Create database

```
CREATE DATABASE <database_name> WITH OWNER = <role_name>;
```

To simplify privilege management, revoke all rights from the public role in the new database, so that they cannot be inherited by new roles that will be created in the future

```
REVOKE ALL ON DATABASE <database_name> FROM PUBLIC;
REVOKE ALL ON SCHEMA <database_name>.public FROM PUBLIC;
```

Create and manage users/roles

Login as superuser. Create a user:

```
CREATE ROLE <username> WITH LOGIN ENCRYPTED PASSWORD <pwd_in_single_quotes>;
```

The following are equivalent

```
CREATE ROLE username WITH LOGIN;
CREATE USER username;
```

Check users, roles and privileges

```
psql=> \du -- roles and users
psql=> \dp [<schema_name>.* | <table_name>]     -- user privileges for a specific schema or table
```

In effect, the latter only shows the privileges for a table that are not in place by default. For example, the table owner has all privileges by default, and those are not shown here.

Similarly, as a query:

```
select * from pg_roles;
select * from information_schema.role_table_grants where grantee = '<role_name>';
```

Note that the last query only works on tables, not views or materialized views.

Grant privileges

Grant privileges on a specific schema. First, allow the user to list the objects in the schema, and then assign privileges

```
GRANT CONNECT ON DATABASE <database_name> TO <role_name>;
GRANT USAGE ON SCHEMA <schema_name> TO <role_name>;
GRANT SELECT ON ALL TABLES IN SCHEMA <schema_name> TO <role_name>;  -- all tables
GRANT SELECT ON TABLE <table_name> TO <role_name>;                  -- one table
```

It is also possible grant privileges on objects that will be created in the future. However, this applies to tables and views but NOT materialized views. For materialized views, privileges have to be granted explicitly after a new view is created.

```
ALTER DEFAULT PRIVILEGES IN SCHEMA <schema_name> GRANT SELECT ON TABLES TO <role_name>;
```

Altered default privileges can be checked with psql command `\ddp`. They can be revoked (reset to default) by:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA <schema_name> REVOKE SELECT ON TABLES FROM <role_name>;
```

According to PostgreSQL documentation: “If you wish to drop a role for which the default privileges have been altered, it is necessary to reverse the changes in its default privileges or use DROP OWNED BY to get rid of the default privileges entry for the role.”

Revoke privileges

```
revoke select on table <schema>.<table> from <user>;
```