Changing PostgreSQL views

When you want to change a view in Postgres, for instance to add an extra field, it can happen that the view is used elsewhere (perhaps in another view) and that it can't be changed. Instead, the depending views first have to be dropped. This can easily cascade, and you might even end up having to drop all views and recreate them. This is a lot of tricky work and very error prone.

I've come up with a bare bones way around this, and basically comes down to:

1. dump the database
2. change the view definition in the SQL file
3. drop the database
4. recreate the database from the altered SQL file

This might sound draconic, but it is trivial to do and not error prone at all. The only downside is that the database in unavailable for a small amount of time. In our case this was not a problem.

Follow these steps:

- Dump the database to an SQL file, and create a copy too:

```
pg_dump terena > terena.sql
cp terena.sql backup.sql
```

- Now edit the SQL file, by locating the view definition, and change it to what it should be.
- Create a database patch from the changes:

```
diff -Nru backup.sql terena.sql > terena.diff
```

An example of such a patch, that adds an extra field to the `vw_events` view:

```
--- terena.sql    2012-08-20 12:52:47.000000000 +0200
+++ backup.sql 2012-08-20 12:58:39.000000000 +0200
@@ -1935,7 +1935,7 @@
 --
 CREATE VIEW vw_events AS
-    SELECT event_status.status, event_types.name AS event_type, events.event_id, events.gcal_event_id,
events.owner_email, events.event_type_id, events.name, events.description, events.starts, events.ends,
date_part('day'::text, events.starts) AS start_day, date_part('month'::text, events.starts) AS
start_month, date_part('year'::text, events.starts) AS start_year, date_part('day'::text, events.ends)
AS end_day, date_part('month'::text, events.ends) AS end_month, date_part('year'::text, events.ends) AS
end_year, events.venue_name, events.address, events.city, events.country, events.country_code, events.
project_url, events.direct_link, events.agenda_url, events.show, events.show_homepage, events.
event_status_id, events.reg_deadline, events.capacity, events.customfield1_label, events.inserted,
events.modified, events.group_id FROM ((event_types JOIN events ON ((event_types.event_type_id = events.
event_type_id))) JOIN event_status ON ((events.event_status_id = event_status.event_status_id)));
+    SELECT event_status.status, event_types.name AS event_type, events.event_id, events.gcal_event_id,
events.owner_email, events.event_type_id, events.name, events.description, events.starts, events.ends,
date_part('day'::text, events.starts) AS start_day, date_part('month'::text, events.starts) AS
start_month, date_part('year'::text, events.starts) AS start_year, date_part('day'::text, events.ends)
AS end_day, date_part('month'::text, events.ends) AS end_month, date_part('year'::text, events.ends) AS
end_year, events.venue_name, events.address, events.city, events.country, events.country_code, events.
project_url, events.direct_link, events.agenda_url, events.show, events.show_homepage, events.
event_status_id, events.reg_deadline, events.capacity, events.customfield1_label, events.inserted,
events.modified, events.group_id, events.show_reglink FROM ((event_types JOIN events ON ((event_types.
event_type_id = events.event_type_id))) JOIN event_status ON ((events.event_status_id = event_status.
event_status_id)));

 ALTER TABLE public.vw_events OWNER TO postgres;
```

- Now make sure the database has no users. In our case this means shutting down the apache web server. Then dump the database, drop it, patch the dump, and restore the patched dump:

```
sudo apache2ctl stop
pg_dump terena > terena.latest.sql
patch terena.latest.sql < terena.diff
dropdb terena
createdb -T template0 terena
psql terena < terena.latest.sql
sudo apache2ctl start
```

You should test this of course by restoring to another database to make sure the patch is working OK. If you feel comfortable you could script everything together to minimise downtime.