OpenAtlas - Snippets - # 67

Below are some unsorted database related code snippets used in development.

Exports

Database Structure

Export the database structure from the test database (used to avoid specialties in production databases) into install/1_structure.sql

```
pg_dump -sc --if-exists -n model -n gis -n log -n web -n import openatlas_test > install/1_structu
re.sql
```

Model Data

```
pg_dump openatlas --rows-per-insert 10000 -a -t model.cidoc_class -t model.cidoc_class_i18n -t mod
el.cidoc_class_inheritance -t model.property -t model.property_i18n -t model.property_inheritance
> install/2_data_model.sql
```

Single Schema

```
pg_dump -n web openatlas > /tmp/openatlas_web.sql
pg_dump -n model openatlas > /tmp/openatlas_web.sql
```

Add update trigger for modified field

Replace **schema.table** and execute:

```
CREATE TRIGGER update_modified BEFORE UPDATE ON schema.table FOR EACH ROW EXECUTE PROCEDURE model.
update_modified();
```

Export database for Windows

```
pg_dump -Fc openatlas > openatlas.dump
```

to ingest in windows

```
C:\Program Files\PostgreSQL\12\bin\pg_restore.exe --host "localhost" --port "5432" --username "pos
tgres" --no-password --dbname "openatlas" --clean --verbose "path to dump file"
```

Reset Demo

```
/var/lib/postgresql/reset_demo.sh
```

Find used but missing places

e.g. after a case study separation (not sure if this statement is now showing really missing or just needed locations)

```
SELECT e.id, e.name, e.system_type FROM model.entity e
WHERE e.class_code = 'E53' AND e.system_type = 'place location' AND e.id IN (
   SELECT r.id FROM model.link lr JOIN model.entity r ON lr.range_id = r.id AND lr.property_code
IN ('P74', 'OA8', 'OA9', 'P7'))
   AND e.id NOT IN (SELECT range_id FROM model.link WHERE property_code = 'P53');
```

Delete orphaned locations

```
DELETE FROM model.entity WHERE id IN (
    SELECT id FROM model.entity WHERE system_type = 'place location' AND id NOT IN (
        SELECT e.id FROM model.entity e JOIN model.link l ON e.id = l.range_id AND l.property_code
```

```
 = 'P53'
        WHERE e.class_code = 'E53' AND e.system_type = 'place location'));
```

Import database to Windows

Make a dump with inserts

```
pg_dump --attribute-inserts openatlas > openatlas.sql
```

In the pgadmin tool, select the corresponding database, run the *query tool* and select the *open file* option.
Load the provided dump file and execute (F5) it.
Login and adapt user as needed.

Recursive Events

To get all child events of a given event the SQL below (replace ROOT_EVENT_ID at bottom). It works but is slow and could be improved.

```
WITH RECURSIVE tree AS (
  SELECT e.id, ARRAY[]::INTEGER[] AS ancestors
  FROM model.entity e
  WHERE (SELECT s.id FROM model.entity s JOIN model.link l ON s.id = l.range_id AND l.domain_id =
e.id AND l.property_id = (SELECT id FROM model.property WHERE code = 'P117')) IS NULL
  UNION ALL
  SELECT e.id, tree.ancestors ||
    (SELECT s.id FROM model.entity s JOIN model.link l ON s.id = l.range_id AND l.domain_id = e.id
 AND l.property_id = (SELECT id FROM model.property WHERE code = 'P117'))
  FROM model.entity e, tree
  WHERE (SELECT s.id FROM model.entity s JOIN model.link l ON s.id = l.range_id AND l.domain_id =
e.id AND l.property_id = (SELECT id FROM model.property WHERE code = 'P117')) = tree.id
)
SELECT * FROM tree WHERE ROOT_EVENT_ID = ANY(tree.ancestors);
```