﻿-- Function: clone\_schema(text, text)

-- DROP FUNCTION clone\_schema(text, text);

CREATE OR REPLACE FUNCTION clone\_schema(

source\_schema text,

dest\_schema text,

include\_recs boolean)

RETURNS void AS

$BODY$

-- This function will clone all sequences, tables, data, views & functions from any existing schema to a new one

-- SAMPLE CALL:

-- SELECT clone\_schema('public', 'new\_schema', TRUE);

DECLARE

src\_oid oid;

tbl\_oid oid;

func\_oid oid;

object text;

buffer text;

srctbl text;

default\_ text;

column\_ text;

qry text;

dest\_qry text;

v\_def text;

seqval bigint;

sq\_last\_value bigint;

sq\_max\_value bigint;

sq\_start\_value bigint;

sq\_increment\_by bigint;

sq\_min\_value bigint;

sq\_cache\_value bigint;

sq\_log\_cnt bigint;

sq\_is\_called boolean;

sq\_is\_cycled boolean;

sq\_cycled char(10);

BEGIN

-- Check that source\_schema exists

SELECT oid INTO src\_oid

FROM pg\_namespace

WHERE nspname = quote\_ident(source\_schema);

IF NOT FOUND

THEN

RAISE NOTICE 'source schema % does not exist!', source\_schema;

RETURN ;

END IF;

-- Check that dest\_schema does not yet exist

PERFORM nspname

FROM pg\_namespace

WHERE nspname = quote\_ident(dest\_schema);

IF FOUND

THEN

RAISE NOTICE 'dest schema % already exists!', dest\_schema;

RETURN ;

END IF;

EXECUTE 'CREATE SCHEMA ' || quote\_ident(dest\_schema) ;

-- Create sequences

-- TODO: Find a way to make this sequence's owner is the correct table.

FOR object IN

SELECT sequence\_name::text

FROM information\_schema.sequences

WHERE sequence\_schema = quote\_ident(source\_schema)

LOOP

EXECUTE 'CREATE SEQUENCE ' || quote\_ident(dest\_schema) || '.' || quote\_ident(object);

srctbl := quote\_ident(source\_schema) || '.' || quote\_ident(object);

EXECUTE 'SELECT last\_value, max\_value, start\_value, increment\_by, min\_value, cache\_value, log\_cnt, is\_cycled, is\_called

FROM ' || quote\_ident(source\_schema) || '.' || quote\_ident(object) || ';'

INTO sq\_last\_value, sq\_max\_value, sq\_start\_value, sq\_increment\_by, sq\_min\_value, sq\_cache\_value, sq\_log\_cnt, sq\_is\_cycled, sq\_is\_called ;

IF sq\_is\_cycled

THEN

sq\_cycled := 'CYCLE';

ELSE

sq\_cycled := 'NO CYCLE';

END IF;

EXECUTE 'ALTER SEQUENCE ' || quote\_ident(dest\_schema) || '.' || quote\_ident(object)

|| ' INCREMENT BY ' || sq\_increment\_by

|| ' MINVALUE ' || sq\_min\_value

|| ' MAXVALUE ' || sq\_max\_value

|| ' START WITH ' || sq\_start\_value

|| ' RESTART ' || sq\_min\_value

|| ' CACHE ' || sq\_cache\_value

|| sq\_cycled || ' ;' ;

buffer := quote\_ident(dest\_schema) || '.' || quote\_ident(object);

IF include\_recs

THEN

EXECUTE 'SELECT setval( ''' || buffer || ''', ' || sq\_last\_value || ', ' || sq\_is\_called || ');' ;

ELSE

EXECUTE 'SELECT setval( ''' || buffer || ''', ' || sq\_start\_value || ', ' || sq\_is\_called || ');' ;

END IF;

END LOOP;

-- Create tables

FOR object IN

SELECT TABLE\_NAME::text

FROM information\_schema.tables

WHERE table\_schema = quote\_ident(source\_schema)

AND table\_type = 'BASE TABLE'

LOOP

buffer := dest\_schema || '.' || quote\_ident(object);

EXECUTE 'CREATE TABLE ' || buffer || ' (LIKE ' || quote\_ident(source\_schema) || '.' || quote\_ident(object)

|| ' INCLUDING ALL)';

IF include\_recs

THEN

-- Insert records from source table

EXECUTE 'INSERT INTO ' || buffer || ' SELECT \* FROM ' || quote\_ident(source\_schema) || '.' || quote\_ident(object) || ';';

END IF;

FOR column\_, default\_ IN

SELECT column\_name::text,

REPLACE(column\_default::text, source\_schema, dest\_schema)

FROM information\_schema.COLUMNS

WHERE table\_schema = dest\_schema

AND TABLE\_NAME = object

AND column\_default LIKE 'nextval(%' || quote\_ident(source\_schema) || '%::regclass)'

LOOP

EXECUTE 'ALTER TABLE ' || buffer || ' ALTER COLUMN ' || column\_ || ' SET DEFAULT ' || default\_;

END LOOP;

END LOOP;

-- add FK constraint

FOR qry IN

SELECT 'ALTER TABLE ' || quote\_ident(dest\_schema) || '.' || quote\_ident(rn.relname)

|| ' ADD CONSTRAINT ' || quote\_ident(ct.conname) || ' ' || pg\_get\_constraintdef(ct.oid) || ';'

FROM pg\_constraint ct

JOIN pg\_class rn ON rn.oid = ct.conrelid

WHERE connamespace = src\_oid

AND rn.relkind = 'r'

AND ct.contype = 'f'

LOOP

EXECUTE qry;

END LOOP;

-- Create views

FOR object IN

SELECT table\_name::text,

view\_definition

FROM information\_schema.views

WHERE table\_schema = quote\_ident(source\_schema)

LOOP

buffer := dest\_schema || '.' || quote\_ident(object);

SELECT view\_definition INTO v\_def

FROM information\_schema.views

WHERE table\_schema = quote\_ident(source\_schema)

AND table\_name = quote\_ident(object);

EXECUTE 'CREATE OR REPLACE VIEW ' || buffer || ' AS ' || v\_def || ';' ;

END LOOP;

-- Create functions

FOR func\_oid IN

SELECT oid

FROM pg\_proc

WHERE pronamespace = src\_oid

LOOP

SELECT pg\_get\_functiondef(func\_oid) INTO qry;

SELECT replace(qry, source\_schema, dest\_schema) INTO dest\_qry;

EXECUTE dest\_qry;

END LOOP;

RETURN;

END;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION clone\_schema(text, text, boolean)

OWNER TO enterprisedb;