```
\c postgres postgres

\set on_error_stop

-- cleanup
drop database if exists my_pg_db;
drop tablespace if exists my_pg_tbs;
drop role if exists my_pg_usr;
drop role if exists my_pg_role;

-- create db owner, the db itself and a tablespace for the database
create role my_pg_usr login password 'pg';
create role my_pg_role;
create tablespace my_pg_tbs owner my_pg_usr location '/opt/PostgreSQL/9.1/data/pg_tblspc/my_pg_tbs';
create database my_pg_db tablespace = my_pg_tbs;

grant connect on database my_pg_db to my_pg_role;
grant create on database my_pg_db to my_pg_role;
grant my_pg_role to my_pg_usr;

\c my_pg_db my_pg_usr
create schema my_pg_schema;
-- although we can create it is not necessary beacuse we directly can define
-- a table's column as an array
create type my_pg_schema.my_pg_type as ( my_pg_type_attr VARCHAR(100)[100]);
create sequence my_pg_schema.my_pg_seq_1 cache 20;
create sequence my_pg_schema.my_pg_seq_2;
create table my_pg_schema.t_master ( master_id bigint not null default nextval
('my_pg_schema.my_pg_seq_1')
                                   , master_descr varchar(200) not null
                                   , master_date timestamp without time zone
                                   , master_int interval day to second
                                   , master_ts timestamp with time zone
                                   , master_ar varchar(100)[]
                                   , constraint t_master_pk primary key ( master_id )
                                   , constraint t_master_chk1 check ( length(master_descr) > 10 )
                                   , constraint t_master_uk1 unique ( master_date )
                                   ) tablespace my_pg_tbs;

create table my_pg_schema.t_detail ( detail_id bigint not null default nextval
( 'my_pg_schema.my_pg_seq_2' )
                                   , master_id bigint not null
                                   , detail_clob text
                                   , detail_blob bytea
                                   , change_date timestamp without time zone
                                   , constraint t_detail_pk primary key ( detail_id )
                                   , constraint t_detail_fk foreign key ( master_id ) references
my_pg_schema.t_master ( master_id )
                                   ) tablespace my_pg_tbs;

create index my_pg_idx1 on my_pg_schema.t_detail ( master_id ) tablespace my_pg_tbs;
-- no bitmap index in postgres, so creating a standard index
create index my_pg_idx2 on my_pg_schema.t_detail ( change_date ) tablespace my_pg_tbs;

-- in postgres we need a function of type trigger for using this in the actual trigger
create function my_pg_schema.last_changed () returns trigger AS '
    BEGIN
        new.change_date :=''now'';
        RETURN new;
    END;
' LANGUAGE 'plpgsql';
-- now create the trigger which uses the function
create trigger trg_t_detail before insert or update on my_pg_schema.t_detail
for each row
execute procedure my_pg_schema.last_changed(change_date);

-- prepare a binary file for loading
```

```
\! cp /opt/PostgreSQL/9.2/doc/enterprisedb.gif /tmp

-- no need to create a directory in postgres, if we have permissions, it's fine

-- generate data
do
  $$declare ln_master_key bigint;
begin
  for i in 1..10000
  loop
    ln_master_key := nextval ( 'my_pg_schema.my_pg_seq_1' );
    insert into my_pg_schema.t_master ( master_id
                                      , master_descr
                                      , master_date
                                      , master_int
                                      , master_ts
                                      , master_ar
                                      )
           values ( ln_master_key
                  , 'DESCRIPTION '||i
                  , clock_timestamp() + interval '1 day'
                  , interval '10 00:00:00' day to second
                  , clock_timestamp()
                  , '{"aaaaa","bbbbb"}'
                  );
    insert into my_pg_schema.t_detail ( detail_id
                                      , master_id
                                      , detail_clob
                                      , detail_blob
                                      )
           values ( nextval ( 'my_pg_schema.my_pg_seq_2' )
                  , ln_master_key
                  , 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
                  , '/tmp/enterprisedb.gif'
                  );
  end loop;
end$$;

select count(*)
  from my_pg_schema.t_master;
select count(*)
  from my_pg_schema.t_detail;

-- in postgresql a package does not exists, so we create schemas for this
create schema my_pg_pgk1;
create or replace function my_pg_pgk1.p1 ( pn_detail_id IN my_pg_schema.t_detail.detail_id%TYPE
                                         , pc_detail_clob IN text
                                         ) RETURNS VOID AS $$
declare
begin
  if ( pn_detail_id is not null )
     and
     ( pc_detail_clob::text != '' )
  then
    update my_pg_schema.t_detail
       set detail_clob = pc_detail_clob
     where detail_id = pn_detail_id
    ;
  end if;
  null;
end;
$$ LANGUAGE plpgsql;

create or replace function my_pg_pgk1.p2 ( pn_master_id IN int
                                         , pn_master_ar IN varchar(100)[]
                                         ) RETURNS VOID AS $$
declare
```

```
begin
  update my_pg_schema.t_master
     set master_ar = pn_master_ar
       , master_date = clock_timestamp()
   where master_id = pn_master_id
  ;
end;
$$ LANGUAGE plpgsql;

create or replace function my_pg_pgk1.f1() RETURNS bigint AS $$
declare
begin
  return mod(99,4);
end;
$$ LANGUAGE plpgsql;

-- check if the functions work
select my_pg_pgk1.p1(1,'aaaaaaaaaaaaaaa');
select detail_clob
  from my_pg_schema.t_detail
 where detail_id = 1
;
select my_pg_pgk1.p2(1,'{"eeeee","ffffff"}');
select master_ar
  from my_pg_schema.t_master
 where master_id = 1
;
select my_pg_pgk1.f1();

-- currently there are no materialized views in postgres, so this is a bit more work to to
--    check: http://tech.jonathangardner.net/wiki/PostgreSQL/
Materialized Views#Snapshot Materialized Views

-- partitioning

create table my_pg_schema.my_ptab1 ( id bigint
                                   , mywhen timestamp
                                   ) tablespace my_pg_tbs;

create table my_pg_schema.my_ptab1_p1 (
   check ( mywhen < DATE '2012-01-01' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p2 (
   check ( mywhen >= DATE '2012-01-01' and mywhen <= DATE '2012-01-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p3 (
   check ( mywhen >= DATE '2012-02-01' and mywhen <= DATE '2012-02-29' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p4 (
   check ( mywhen >= DATE '2012-03-01' and mywhen <= DATE '2012-03-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p5 (
   check ( mywhen >= DATE '2012-04-01' and mywhen <= DATE '2012-04-30' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p6 (
   check ( mywhen >= DATE '2012-05-01' and mywhen <= DATE '2012-05-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p7 (
   check ( mywhen >= DATE '2012-06-01' and mywhen <= DATE '2012-06-30' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p8 (
   check ( mywhen >= DATE '2012-07-01' and mywhen <= DATE '2012-07-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p9 (
   check ( mywhen >= DATE '2012-08-01' and mywhen <= DATE '2012-08-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p10 (
```

```
   check ( mywhen >= DATE '2012-09-01' and mywhen <= DATE '2012-09-30' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p11 (
   check ( mywhen >= DATE '2012-10-01' and mywhen <= DATE '2012-10-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p12 (
   check ( mywhen >= DATE '2012-11-01' and mywhen <= DATE '2012-11-30' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p13 (
   check ( mywhen >= DATE '2012-12-01' and mywhen <= DATE '2012-12-31' )
   ) inherits ( my_pg_schema.my_ptab1 );
create table my_pg_schema.my_ptab1_p14 (
   check ( mywhen >= DATE '2013-01-01' )
   ) inherits ( my_pg_schema.my_ptab1 );

CREATE OR REPLACE FUNCTION my_pg_schema.my_ptab1_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF ( NEW.mywhen < DATE '2012-01-01' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p1 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-01-01' and NEW.mywhen <= DATE '2012-01-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p2 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-02-01' and NEW.mywhen <= DATE '2012-02-29' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p3 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-03-01' and NEW.mywhen <= DATE '2012-03-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p4 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-04-01' and NEW.mywhen <= DATE '2012-04-30' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p5 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-05-01' and NEW.mywhen <= DATE '2012-05-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p6 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-06-01' and NEW.mywhen <= DATE '2012-06-30' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p7 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-07-01' and NEW.mywhen <= DATE '2012-07-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p8 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-08-01' and NEW.mywhen <= DATE '2012-08-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p9 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-09-01' and NEW.mywhen <= DATE '2012-09-30' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p10 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-10-01' and NEW.mywhen <= DATE '2012-10-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p11 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-11-01' and NEW.mywhen <= DATE '2012-11-30' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p12 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2012-12-01' and NEW.mywhen <= DATE '2012-12-31' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p13 VALUES (NEW.*);
    ELSIF ( NEW.mywhen >= DATE '2013-01-01' ) THEN
        INSERT INTO my_pg_schema.my_ptab1_p14 VALUES (NEW.*);
    ELSE
        RAISE EXCEPTION 'Date out of range.  Fix the my_ptab1_insert_trigger() function!';
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER my_ptab1_insert_trigger
    BEFORE INSERT ON my_pg_schema.my_ptab1
    FOR EACH ROW EXECUTE PROCEDURE my_pg_schema.my_ptab1_insert_trigger();

do
  $$declare ld_begin timestamp := to_date ( '01.01.2012','DD.MM.YYYY');
begin
  for i in 1..465
  loop
    insert into my_pg_schema.my_ptab1 ( id, mywhen )
           values ( i, ld_begin );
    ld_begin := ld_begin + interval '1 day';
  end loop;
```

```
end$$;

SELECT min(mywhen)
     , max(mywhen)
  FROM my_pg_schema.my_ptab1;

SELECT tablename
     , tablespace
  FROM pg_tables
 WHERE tablename LIKE 'my_ptab%'
 ORDER BY 1;
```