Political Configurations Database

Documentation*

July 2015

Chair of Comparative Politics
Humboldt University of Berlin

* Last update: Tuesday 28th July, 2015

Contents

1 Introduction

The data compiled in the Political Configuration Database (PCDB) is programmed and organized using PostgreSQL, an open source object-relational database system.[1] Using *Structured Query Language* (SQL) is tought to gurantee for the integrity, reliability, and correctness of the data contained in the PCDB.

In fact, *integrety* of the data in the PCDB is imposed by

> **compiling primary data** (e.g., vote turnouts, seat results, election and configuration start dates), and
>
> **computing secondary data**, such as indicators (e.g., Effective Number of Parties in Parliament, Type A and B volatilities in seats and vote, etc.) and aggregates (e.g., total votes and seats at the level of the legislature, open veto points in a given configuration, etc.) from the primary data,

though there are also figures on aggregates recorded in the PCDB—mostly obtained from official election statistics—to allow for comparison between recorded and computed aggregate figures.

In addition, programming the computation of secondary data using PostgreSQL ensures the *reliability* and actuallity of the data contained in the PCDB, in that, for instance, recording new election figures (or updates, see Section **??**) requires no further action but indicies, aggregates, and changes in political configurations will be generated automatically.

Lastly, *correctness* of the data is improved by providing automatically generated consistency (see Section **??**) checks that users may query instantely, using the corresponding views.

These are few but nevertheless important features of PostgreSQL and the corresponding data administration and managment platform pgAdmin3, thought to improve the quality of data in the PCDB.

For comments and question the reader may contact to Hauke Licht or Matthias Orlowski, the administrator of the PCDB.

[1] http://www.postgresql.org/

2 Query for data in the PCDB

2.1 Access to the PCDB

The PCDB is accessed using the database managment and adminstration software pgAdmin3.

Users have to install pgAdmin3 on their computers and connect to the PCDB on the server of the Humboldt-University, which is hosted by the Computer and Media Service (CMS) (click 'add server' under 'File').

Enter the following properties of the PCDB in the corresponding lines

Databasesysteme	PostgreSQL
DNS-addresse	moodledb.cms.hu-berlin.de
Portnumber	5432
SSL-Port	5432

The PCDB is named polconfdb on the CMS' database server.

Contact the administrator to receive a username. User names are accounts that are defined as roles.

2.1.1 Roles in the PCDB

There exist three different roles with different sets of privileges to operate in the PCDB via pgAdmin3:

(1) **Administrator**: Having all privileges on both the public and the config_data schemes. This role is assumed by account polconfdb and polconfdb_1. Having all privileges includes to GRANT and REVOKE privileges to and from other the user roles.

(2) **Read-and-Write**: Having privileges SELCECT, INSERT, and UPDATE on both the public and the config_data schemes. This role is assumed by account polconfdb_2 and polconfdb_3. Note that the SELECT-privilege includes the operation COPY TO, which allows to extract data from queries to .csv-documents.

(3) **Read-Only**: Having privilege SELCECT on both the public and the config_data schemes. This role is assumed by account polconfdb_4 and polconfdb_5. The SELECT-privilege includes the operation COPY TO.

The roles in the PCDB are defined as follows:

```
-- Grant usage of schmea config_data to all accounts
GRANT usage ON SCHEMA public, config_data TO polconfdb_1 ;
GRANT usage ON SCHEMA public, config_data TO polconfdb_2 ;
GRANT usage ON SCHEMA public, config_data TO polconfdb_3 ;
GRANT usage ON SCHEMA public, config_data TO polconfdb_4 ;
GRANT usage ON SCHEMA public, config_data TO polconfdb_5 ;

-- create additional adminstrator role
GRANT ALL ON SCHEMA public, config_data TO polconfdb_1;

-- create two view-and-write accounts
GRANT select, insert, update ON ALL TABLES
  IN SCHEMA public, config_data TO polconfdb_2, polconfdb_3;
GRANT execute ON ALL FUNCTIONS
  IN SCHEMA public, config_data TO polconfdb_2, polconfdb_3;

-- creat etwo view-only accounts
GRANT select ON ALL TABLES
  IN SCHEMA public, config_data TO polconfdb_4, polconfdb_5;
```

2.2 Query for data from Tables and Views

3 Programming the PCDB

This chapter provides the code and corresponding explanatory descriptions of the data structre in the PCDB.

Four types of objects will be discussed in succession:

(1) **Tables**: The permanent data repositories that store information at different levels (e.g., parties, institutions, countries, etc.) and serve as priamry source for all computed indices and aggregate figures.

(1) **Views**: Virtual tables based on the result-sets of defined SQL-queries. Views serve two purposes in the PCDB:

   i. Compute aggregates and indices from the primary data contained in tables,

   ii. and create consistency checks that allow o control for the consistency of the data and to trace coding failures.

(3) **Materialized views**: Tables created from views that may be updated from the original base tables from time to time.

(3) **Triggers**: functions implemented on tables or materialized views to insert, update, or delete data as consequence of specific events. Triggers are mainly implemented to enable the automatic up-dating of the PCDB.

3.1 Tables in the config_data scheme

Tables store the primary data of the PCDB, that is used to compute aggregates and indices. This section provides a description of how tables in the PCDB are defined, and thus provides a comprehensiv overview of variable names, their types (i.e., storage format), and potential constraints.

Both types and constraints define the requirements that data thought to be inserted into a column needs to met. An overview of the types provided within `postgreSQL` can be found here; information on constraints in tables here.

In addition section 5.2 of the Appendix provides an overview of variables contained in the tables of the PCDB.

3.1.1 Country

This table contains the 34 countries covered in the PCDB as rows, attributing each country a unique identifier (`ctr_id`) and providing information on their accession date to specific international organizations.

Table `country` is defined as follows:

```
CREATE TABLE config_data.country (
    ctr_id    SMALLINT  PRIMARY KEY,
    ctr_n   NAME    UNIQUE,
    ctr_ccode VARCHAR(3)  UNIQUE,
    ctr_ccode2  VARCHAR(2)  UNIQUE,
    ctr_ccode_nr  NUMERIC(3)  UNIQUE,
    ctr_eu_date DATE    CONSTRAINT def_eu_date
      CHECK (ctr_eu_date >= '1951-04-18'::DATE OR ctr_eu_date IS NULL),
    ctr_oecd_date DATE    CONSTRAINT def_oecd_date
      CHECK (ctr_oecd_date >= '1961-04-10'::DATE OR ctr_oecd_date IS NULL),
    ctr_wto_date  DATE    CONSTRAINT def_wto_date
      CHECK (ctr_wto_date >= '1995-01-01'::DATE OR ctr_wto_date IS NULL),
    ctr_cmt   TEXT,
    ctr_src   TEXT
  );
```

3.1.2 Cabinet

Table `cabinet` contains information on cabinets. Rows are the different cabinet configurations, identified by variable `cab_id`. A new cabinet is enlisted if one of the following events took place:

a) Coalition composition changes at the party-level.

b) Head of government changes.

c) Government formation after general legislative elections (not in presidential systems).

Cabinet start date Variable cab_sdate refers to the date on which the cabinet, as proposed by the Head of Government, recieves a vote of confidence in the legislature. The variable cab_src regularly contains links to the websites or online repositories which are used as references. If available, data was compiled directly from information reported on government websites or other official sources.

Total number of cabinet portfolios In the present version of the database (!) the number of cabinet portolios is an integer counter equal to the number of parties in cabinet. Because it is an aggregate of data contained in the Cabinet Portfolios table (3.1.3), the total number of cabinet portfolios is cumputed in view_pty_cab_sts (3.3.4).

Table cabinet is defined as follows:

```
CREATE TABLE config_data.cabinet (
    cab_id    NUMERIC(5)  PRIMARY KEY,
    cab_prv_id  NUMERIC(5),
    ctr_id    SMALLINT
      REFERENCES config_data.country (ctr_id)
      ON UPDATE CASCADE,
    cab_sdate DATE,
    cab_hog_n VARCHAR(15) ,
    cab_sts_ttl NUMERIC(2,0)  ,
    cab_care  BOOLEAN   ,
    cab_cmt   TEXT,
    cab_src   TEXT
    );
```

3.1.3 Cabinet Portfolios

Table cabinet_portfolios provides information on parties in cabinets.

As cabinet portfolio we define the composition of a cabinet at the party-level. Thus, new portfolios are included whenever a new cabinet emerges. The changes that occur at the party-level regularly correspond to the events enumerated as criteria for recording a new cabinet configuration (cf. subsection 3.1.2):

a) Coalition composition changes.

b) Head of government changes.

c) Government formation after general legislative elections (not in presidential systems).

Obviously, combinations of cabinet and party identifier are unique in the cabinet portfolios table.

Table cabinet_portfolios is defined as follows:

```
CREATE TABLE config_data.cabinet_portfolios (
    ptf_id    NUMERIC(5)  PRIMARY KEY,
    cab_id    NUMERIC(5)
      REFERENCES config_data.cabinet(cab_id)
      ON UPDATE CASCADE,
    pty_id    NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    pty_cab   BOOLEAN ,
```

```
10      pty_cab_sts INTEGER ,
11      pty_cab_hog BOOLEAN ,
12      pty_cab_sup BOOLEAN ,
13      ptf_cmt   TEXT  ,
14      ptf_src   TEXT
15      );
```

3.1.4 Lower House

Table `lower_house` provides information on lower houses. Rows are compositions of lower houses, identified by `lh_id`.

A new lower house configuration is included when the seat composition is changed through legislative elections or through mergers or splits in factions during the legislature. When enlistment is due to the latter event, no lower house election identifier (`lhelc_id`) is recorded. Else, each lower house corresponds to a lower house election.

Lower house start date PCDB codes the date of the first meeting in the first legislative session of a new lower house as its start date (variable `lh_sdate`). Information on the sources is provided in variable `lh_src`. If no information on this event is available, the default is equal to the corresponding election date.

Total number of seats in lower house The figures on the total number of seats in the respective lower house are recorded in accordance with official electoral statistics (variable `lh_sts_ttl`). These figures do not necessarily equal the sum of all seats distributed between different parties of a legislature (as recorded in the lower house seat reuslts data, see subsection **??**).

Table `lower_house` is defined as follows:

```
1  CREATE TABLE config_data.cabinet (
2      cab_id    NUMERIC(5)  PRIMARY KEY,
3      cab_prv_id  NUMERIC(5),
4      ctr_id    SMALLINT
5        REFERENCES config_data.country (ctr_id)
6        ON UPDATE CASCADE,
7      cab_sdate DATE,
8      cab_hog_n VARCHAR(15) ,
9      cab_sts_ttl NUMERIC(2,0)  ,
10     cab_care  BOOLEAN   ,
11     cab_cmt   TEXT,
12     cab_src   TEXT
13     );
```

3.1.5 Lower House Election

Table `lh_election` provides information on lower house elections. Rows are lower house elections, identified by `lhelc_id`. It is noteworthy that each lower house election corresponds to a lower house configuration (cf. subsection 3.1.4).[1]

Elections, pluarality versus proportional voting, and seat allocation Lower house election dates (`lhelc_date`), and figures on registered voters (`lhelc_reg_vts*`), the number valid votes (`lhelc_vts_*`), and the number of seats elected (`lhelc_sts_*`) are recorded in accordance with official statistics, if available. Else, Nohlen (2001, 2005, 2010) is the primary source, complemented by individual-case research. Information on data sources is provided in variable `lhelc_src`.

Electoral system Key information on the electoral system to elect the lower house is provided for each tier disaggregatedly namely

- the electoral formular (`lhelc_fml_t*`), as defined by a customed type `elec_formula` ,
- the number of constituencies (`lhelc_ncst_t*`),
- the number of seats allocated(`lhelc_sts_t*`),
- the average district magnitude (`lhelc_mag_t*`),
- the national threshold (`lhelc_ntrsh_t*`), and
- the district threshold (`lhelc_dtrsh_t*`).

Type `elec_formula` is defined as follows:

```
CREATE TYPE elec_formula AS ENUM (
  '2RS', -- Two Round System
  'AV', -- Alternative Vote
  'DHondt',
  'Droop',
  'LR-Droop', -- Droop w/ Largest Remainders
  'Hare',
  'modified Hare',
  'LR-Hare', -- Hare w/ Largest Remainders
  'highest average remaining',
  'Imperiali',
  'MMD', -- Multi-Member District
  'mSainteLague',
  'Reinforced Imperiali',
  'SainteLague',
  'SMP', -- Single Member Plurality
  'SNTV', -- Single Non-Transferable Vote
  'STV' -- Single Transferable Vote
  );
```

In addition, variables `lhelc_dstr_mag` and `lhelc_dstr_mag_med` aggregate the average district magnitudes across the different tiers of the electoral system, reporting the mean and the median, respectively.

[1] While the opposite, that each lower house configuration corresponds to a lower house election, is not true.

Comments and information on the sources of data on the electoral system are provided in `lhelc_esys_cmt` and `lhelc_esys_src`, respectively.

Table `lh_election` is defined as follows:

```
CREATE TABLE config_data.lh_election (
    lhelc_id    NUMERIC(5)  PRIMARY KEY,
    lhelc_prv_id    NUMERIC(5),
    ctr_id      SMALLINT
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    lhelc_date    DATE    NOT NULL,
    lhelc_early   BOOLEAN,
    lhelc_reg_vts   NUMERIC,
    lhelc_reg_vts_pr  NUMERIC,
    lhelc_reg_vts_pl  NUMERIC,
    lhelc_vts_pr    NUMERIC   DEFAULT NULL,
    lhelc_vts_pl    NUMERIC   DEFAULT NULL,
    lhelc_sts_pr    NUMERIC   DEFAULT NULL,
    lhelc_sts_pl    NUMERIC   DEFAULT NULL,
    lhelc_sts_ttl   NUMERIC   DEFAULT NULL,

    lhelc_fml_t1    elec_formula,
    lhelc_ncst_t1   NUMERIC   DEFAULT NULL,
    lhelc_sts_t1    NUMERIC   DEFAULT NULL,
    lhelc_dstr_mag    NUMERIC   DEFAULT NULL,
    lhelc_dstr_mag_med  NUMERIC   DEFAULT NULL,
    lhelc_mag_t1    NUMERIC   DEFAULT NULL,
    lhelc_ntrsh_t1    NUMERIC   DEFAULT NULL,
    lhelc_dtrsh_t1    NUMERIC   DEFAULT NULL,

    lhelc_fml_t2    elec_formula,
    lhelc_ncst_t2   NUMERIC   DEFAULT NULL,
    lhelc_sts_t2    NUMERIC   DEFAULT NULL,
    lhelc_mag_t2    NUMERIC   DEFAULT NULL,
    lhelc_ntrsh_t2    NUMERIC   DEFAULT NULL,
    lhelc_dtrsh_t2    NUMERIC   DEFAULT NULL,

    lhelc_fml_t3    elec_formula,
    lhelc_ncst_t3   NUMERIC   DEFAULT NULL,
    lhelc_sts_t3    NUMERIC   DEFAULT NULL,
    lhelc_mag_t3    NUMERIC   DEFAULT NULL,
    lhelc_ntrsh_t3    NUMERIC   DEFAULT NULL,
    lhelc_dtrsh_t3    NUMERIC   DEFAULT NULL,

    lhelc_fml_t4    elec_formula,
    lhelc_ncst_t4   NUMERIC   DEFAULT NULL,
    lhelc_sts_t4    NUMERIC   DEFAULT NULL,
    lhelc_mag_t4    NUMERIC   DEFAULT NULL,
    lhelc_ntrsh_t4    NUMERIC   DEFAULT NULL,
    lhelc_dtrsh_t4    NUMERIC   DEFAULT NULL,

    lhelc_bon_sts   NUMERIC   DEFAULT NULL,
    lhelc_esys_cmt    TEXT,
    lhelc_cmt   TEXT,
    lhelc_esys_src    TEXT,
    lhelc_lsq   DOUBLE PRECISION,
    lhelc_vola_sts    DOUBLE PRECISION,
    lhelc_volb_sts    DOUBLE PRECISION,
    lhelc_vola_vts    DOUBLE PRECISION,
    lhelc_volb_vts    DOUBLE PRECISION,
    lhelc_src   TEXT  -
);
```

3.1.6 Lower House Vote Results

Table `lh_vote_results` contains data on the distribution of votes in the lower house at the party-level. Rows are the parties (identified by variable `pty_id`) and their respective vote results in a given lower house election (variable `lh_id`).

It is defined as follows:

```
CREATE TABLE config_data.lh_vote_results (
    lhvres_id NUMERIC(5)  PRIMARY KEY,
    lhelc_id  NUMERIC(5)
      REFERENCES config_data.lower_house(lh_id)
      ON UPDATE CASCADE,
    pty_id    NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    pty_lh_vts_pr INTEGER DEFAULT NULL,
    pty_lh_vts_pl INTEGER DEFAULT NULL,
    lhvres_cmt  TEXT,
    lhvres_src  TEXT
    );
```

3.1.7 Lower House Seat Results

Table `lh_seat_results` contains data on the distribution of seats in the lower house at the party-level. Rows are the parties (identified by variable `pty_id`) and their respective vote results in a given lower house election (variable `lh_id`).

It is defined as follows:

```
CREATE TABLE config_data.lh_seat_results (
    lhsres_id NUMERIC(5)  PRIMARY KEY,
    lhelc_id  NUMERIC(5)
      REFERENCES config_data.lower_house(lh_id)
      ON UPDATE CASCADE,
    pty_id    NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    pty_lh_sts_pr INTEGER DEFAULT NULL,
    pty_lh_sts_pl INTEGER DEFAULT NULL,
    pty_lh_sts  INTEGER,
    lhvres_cmt  TEXT,
    lhvres_src  TEXT
    );
```

3.1.8 Upper House

Table `upper_house` provides basic information on upper houses, including start date of legislature and the total number of seats. Rows are compositions of upper houses, identified by `uh_id` as well as unique combinations of `ctr_id` and `uh_sdate`.

A new upper house composition is included when

a) the composition changes through legislative elections, or

b) mergers or splits in factions occur during the legislature.

Obviously, information is only provided for countries with bicameral systems.

Upper house start date PCDB codes the date of the first meeting in the first legislative session of a new upper house as its start date. If no information on these events was available, the default is equal to the corresponding election date.

Table upper_house is defined as follows:

```
CREATE TABLE config_data.upper_house (
    uh_id   NUMERIC(5)  PRIMARY KEY,
    uh_prv_id NUMERIC(5),
    uhelc_id  NUMERIC(5)
      REFERENCES config_data.uh_election
      MATCH SIMPLE
      ON UPDATE CASCADE,
    ctr_id    SMALLINT
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    uh_sdate  DATE,
    uh_sts_ttl  INTEGER NOT NULL,
    uh_cmt    TEXT,
    uh_src    TEXT
    );
```

3.1.9 Upper House Election

Table uh_election includes information on upper house elections. Rows report elections to form the upper house and identified by uhelc_id as well as unique combinations of ctr_id and uhelc_date. Obviously, information is only provided on countries with bicameral systems.

It is defined as follows:

```
CREATE TABLE config_data.uh_election (
    uhelc_id  NUMERIC(5)  PRIMARY KEY,
    uhelc_prv_id  NUMERIC(5),
    ctr_id    SMALLINT
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    uhelc_date  DATE,
    uh_sts_ttl  INTEGER NOT NULL,
    uhelc_sts_elc INTEGER NOT NULL,
    uhelc_cmt TEXT,
    uhelc_src TEXT
    );
```

3.1.10 Upper House Seat Results

Table uh_seat_results compiles data on the seat composition in upper houses at the party-level. Rows are the parties, identified by variable pty_id, and their respective seat results in a given upper house (uh_id).

It is defined as follows:

```
CREATE TABLE config_data.uh_seat_results (
    uhsres_id NUMERIC(5)  PRIMARY KEY,
    uh_id NUMERIC(5)
      REFERENCES config_data.upper_house(uh_id)
      ON UPDATE CASCADE,
    pty_id    NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    pty_uh_sts_elc  NUMERIC,
    pty_uh_sts  NUMERIC NOT NULL,
    uhsres_cmt  TEXT,
    uhsres_src  TEXT
    );
```

3.1.11 Presidential Election

Table `presidential_election` contains contains information on the election date, the winner and the electoral system that was applied in an election. Rows are presidential elections, identified by variable `prselc_id` as well as unique combinations of `ctr_id` and `prselc_date`.[2]

In addition variable `prs_n`, `pty_id` and `prs_sdate`, respectively, report the name, the party affiliation and the date of investiture of the candidtate that won the election.

Table `presidential_election` is defined as follows:

```
CREATE TABLE config_data.presidential_election (
    prselc_id NUMERIC(5)  PRIMARY KEY,
    prselc_prv_id NUMERIC(5),
    ctr_id    SMALLINT
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    prselc_date   DATE,
    prselc_rnd_ttl    SMALLINT DEFAULT ('1'),
    prselc_vts_clg    NUMERIC,
    reg_vts_prselc_r1 NUMERIC,
    reg_vts_prselc_r2 NUMERIC DEFAULT NULL,
    prselc_vts_ppl_r1 NUMERIC,
    prselc_vts_ppl_r2 NUMERIC DEFAULT NULL,
    prselc_clg    BOOLEAN,
    prs_n   NAME,
    pty_id    NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    prs_sdate DATE,
    prselc_cmt  TEXT,
    prselc_src  TEXT
    );
```

3.2 Presidential Election Vote Results

Table `pres_elec_vres` provides data on vote results in presidential elections at the candidate-level. Rows are the candidates running in the (multiple rounds of) election(s) and their respec-

[2] Note that the direct elections of the Prime Minister in Israel between 1996 and 2001 are included in this table as well.

tive vote results, identified by prsvres_is as well as unique combinations of prselc_is, prselc_rnd and prselc_cnd_pty.

Table pres_elec_vres is defined as follows:

```
CREATE TABLE config_data.pres_elec_vres (
    prsvres_id  NUMERIC(5)  PRIMARY KEY,
    prselc_id NUMERIC(5)
      REFERENCES config_data.presidential_election(prselc_id)
      ON UPDATE CASCADE,
    prselc_rnd  SMALLINT,
    prs_cnd_pty NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    prs_cnd_n NAME,
    prs_cnd_vts_clg INTEGER,
    prs_cnd_vts_ppl INTEGER,
    prsvres_cmt TEXT,
    prsvres_src TEXT
);
```

3.2.1 Veto Points

Table veto_points contains information on the potential veto points in a country's political system, including the type of institution and the time period of its existence as a veto point. Rows are the different institutions in a country, identified by vto_id as well as unique combinations of ctr_id, vto_inst_typ and vto_inst_sdate.

Veto Institution Type Variable vto_inst_typ is defined as customed type labeled , which is defined as follows:

```
CREATE TYPE vto_type AS ENUM (
  'head of state',
  'head of government',
  'lower house',
  'upper house',
  'judicial',
  'electoral',
  'territorial');
```

Veto Potential Variable vto_pwr records the veto potential for each institution type in a country. It is ordinal and bound between 1 and 0.

- An institution's veto power is coded 0 if it is generally not entitled to a veto right;
- coded 1 if it enjoys unconditional veto potential;
- or may assume values in between 0.5 and 1, indicating conditionality of veto potential with regard to the required seats share of cabinet parties in lower or upper house, respectively, given a certain constitutional threshold.

Note that information on institutions' veto potential is essential to identify open institutional veto points in a given political configuration, for they depend on both constitutional entitlement of veto and the specific date (i.e., duration) of the present political configuartion, and—given some conditionality—on the size of political majorities or party allignment of the president.

Veto institution start and end date Variables `vto_inst_sdate` and `vto_inst_edate` report the start and end dates of the veto power status of respective institutions.

Though constitutional reforms are rare and in the vast majority of cases there is recorded only one veto power status per type of veto instution within countries, not every institution's veto power has remained unchanged throughout the PCDB's period of coverage. The Belgian Senaat (the upper house), for instance, lost its conditional, 50-percent counter-majoritarian threshold veto potential in 1995. The Veto Points table therefore records two rows for the Belgian upper house, one with start date 1st January, 1900, (the default start date) and May 20, 1995, as end date, and one row with start date May 21, 1995, and the default end date December 31, 2099, because no other change of veto power took place until the end of 2014.

Table `veto_points` is defined as follows:

```
CREATE TABLE config_data.veto_points (
    vto_id    NUMERIC(5)  PRIMARY KEY,
    ctr_id    SMALLINT
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    vto_inst_typ  VTO_TYPE,
    vto_inst_n  NAME,
    vto_inst_n_en NAME,
    vto_inst_sdate  DATE
      CONSTRAINT def_inst_sdate NOT NULL DEFAULT '1900-01-01'::date,
    vto_inst_edate  DATE
      CONSTRAINT def_inst_edate DEFAULT NULL,
    vto_pwr   NUMERIC(3,2),
    vto_cmt   TEXT,
    vto_src   TEXT
    );
```

3.2.2 Party

Table `party` provides basic information on parties, permitting to link them to other party-level databases or tables in the PCDB. Rows are parties within countries, identified by unique combinations of `ctr_id` and `pty_id`.

Party identifier The PCDB uses simple running counters to identify parties in a country's political system and history (variable `pty_id`). That is, in contrast to the coding schemes applied in the Manifesto Project (Volkens et al., 2013) or the ParlGov data (Döring and Manow, 2012), identifiers do not encode allignment with party-families or ideological leaning on a left-right scale.

Special suffix are assigned to independent candidates (##997), other parties with seats (##998), and other parties without seats in the legislature (##999).

Table `party` is defined as follows:

```
CREATE TABLE config_data.party (
    pty_id    NUMERIC(5)  PRIMARY KEY,
    pty_abr   VARCHAR(10) UNIQUE NOT NULL,
    pty_n   VARCHAR(45),
    pty_n_en  VARCHAR(45),
    cmp_id    NUMERIC(5),
```

```
    prlgv_id  INTEGER,
    pty_eal   INTEGER,
    pty_eal_id  NUMERIC(5),
    ctr_id    SMALLINT  UNIQUE
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    clea_id   VARCHAR(10),
    pty_cmt   TEXT,
    pty_src   TEXT
);
```

3.2.3 Electoral Alliances

Table `electoral_alliances` provides information on electoral alliances, attempting to identify the parties forming an electoral alliance. Parties listed in the Party table (3.2.2) that are recorded as electoral alliances are listed in with their respective `pty_ids`.

Variable `pty_eal_nbr` is a counter that enumerates parties that constitute an electoral alliance.[3] Accordingly, there occur as many rows for each electoral alliance in the table as variable `pty_eal` counts.

Variable `pty_eal_id`, in turn, records the party identifiers of the parties that form an electoral alliance. Combinations of `pty_id` (electoral alliance) and `pty_eal_nbr` (enumerator of party in electoral alliance) are therefore unique within countries.

Example: Composition of selected electoral alliances in Portugal.

Electoral Alliances			Party	
Identifier `pty_id`	Abbrevation `pty_abr`	Enumerator `pty_eal_n`	Identifier `pty_eal_id`	Abbrevation
8003	AP	1	8999	Other
8003	AP	2	8999	Other
8003	AP	3	8999	Other
8005	PSP.US	99	8058	PSP
8006	PDPC	1	8059	CDC
8006	PDPC	2	8999	Other
8006	PDPC	3	8999	Other
8006	PDPC	4	8999	Other

The example displays a selection from the recorded electoral alliances in Portugal, thought to illustrate the coding scheme and organization of data. Electoral alliance AP is formed by three parties, of which none is recorded in PCDB Party data (Table **??**) and thus ##999s are assigned. One party that forms electoral alliance PSP.US is identified as PSP; however it could not be validated how many parties form the alliance, and therefore the enumeraor is coded 99. PDPC

[3] The counter is also recorded in the Party table and equals one for all 'conventional' parties.

is knowingly formed by four parties of which only one (CDC) is recorded in the PCDB Party data.

Thought `pty_eal_id` often references ##999, it allows to link additional information on parties provided in Table **??** to the electoral-alliance information.

Table `electoral_alliances` is defined as follows:

```
CREATE TABLE config_data.electoral_alliances(
    ctr_id    SMALLINT
      REFERENCES config_data.country(ctr_id)
      ON UPDATE CASCADE,
    pty_id    NUMERIC(5)
      REFERENCES config_data.party(pty_id)
      ON UPDATE CASCADE,
    pty_abr   VARCHAR(50),
    pty_eal_nbr INTEGER,
    pty_eal_id  NUMERIC(5),
    pty_eal_cmt TEXT,
    pty_eal_src TEXT
);
```

3.3 Views in the config_data scheme

The views contained in the `config_data` scheme of the PCDB compute aggregates and indices from primary data (see section 3.1).

In the following subsections the views that exist in the `config_data` scheme will be dicussed with regard to the tables, views and materialized views they are based on, the level at which information is provided, how they are programmed, and sources of potential missings (i.e., NULL-values).

3.3.1 Cabinet's Seat Share in the Lower House

View `view_cab_lh_sts_shr` is based on tables Cabinet Portfolios and Lower House, and views Configuration Events (**??**) and Party's Seat Share in the Lower House (**??**), and provides information at the level of political configurations.

It computes the joint seat share of cabinet parties in the corresponding lower house (LH).

View `view_cab_lh_sts_shr` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_cab_lh_sts_shr
AS
SELECT DISTINCT ctr_id, sdate, cab_id, lh_id, cab_lh_sts_shr
FROM
  (SELECT ctr_id, sdate, lh_id, cab_id, SUM(pty_lhelc_sts_shr) AS cab_lh_sts_shr
    FROM
      (SELECT cab_id, pty_id, pty_cab
        FROM config_data.cabinet_portfolios
      ) AS CAB_PORTFOLIOS
    JOIN
      (SELECT *
        FROM
          (SELECT ctr_id, sdate, lh_id, cab_id
            FROM config_data.mv_configuration_events
          ) AS CONFIGURATION_EVENTS
        LEFT OUTER JOIN
          (SELECT lh_id, pty_id, pty_lhelc_sts_shr
            FROM config_data.view_pty_lh_sts_shr
          ) AS PTY_LH_STS_SHR
        USING(lh_id)
      ) AS CAB_LH_CONFIGS
    USING(cab_id, pty_id)
    WHERE pty_cab IS TRUE
    GROUP BY ctr_id, sdate, lh_id, cab_id
    ORDER BY ctr_id, sdate
  ) AS CONFIGS_W_CAB_LH_STS_SHR
ORDER BY ctr_id, sdate, cab_id NULLS FIRST;
```

Varaible `cab_lh_sts_shr` is a column in the Configuration view and essential to determine whether the LH constitutes an open veto point vis-à-vis the government in a given political configuration.

NULL-values might stem from different missings, such as

- no lower house configuration correspoonds the given cabinet or vice-versa;

- no `lhelc_id` listed in table Lower House that would allow to compute joint seat share (check using `cc_no_lhelc_id_4_lh`, subsection 3.7.23);
- no seats recorded for cabinet parties in table Lower House Seat Results (check using `cc_missing_lhelc_pty_sts_records`, subsection **??**); or
- cabinet parties cannot be identified in table Lower House Seat Results (i.e., mismatch of `pty_ids`).

3.3.2 Cabinet's Seat Share in the Upper House

View `view_cab_uh_sts_shr` is based on table Cabinet Portfolios, and views Configuration Events (**??**) and Party's Seat Share in the Upper House (3.3.31), and provides information at the level of political configurations.

It computes the joint seat share of cabinet parties in the corresponding upper house (UH).

View `view_cab_uh_sts_shr` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_cab_uh_sts_shr
AS
SELECT DISTINCT ctr_id, sdate, cab_id, uh_id, cab_uh_sts_shr
FROM
  (SELECT ctr_id, sdate, uh_id, cab_id, SUM(pty_uh_sts_shr) AS cab_uh_sts_shr
    FROM
      (SELECT cab_id, pty_id, pty_cab
        FROM config_data.cabinet_portfolios
      ) AS CAB_PORTFOLIOS
    JOIN
      (SELECT *
      FROM
        (SELECT ctr_id, sdate, uh_id, cab_id
          FROM config_data.mv_configuration_events
        ) AS CONFIGURATION_EVENTS
      LEFT OUTER JOIN
        (SELECT uh_id, pty_id, pty_uh_sts_shr
          FROM config_data.view_pty_uh_sts_shr
          WHERE uh_id IS NOT NULL
        ) AS PTY_UH_STS_SHR
      USING(uh_id)
      ) AS CAB_UH_CONFIGS
    USING(cab_id, pty_id)
    WHERE pty_cab IS TRUE
    GROUP BY ctr_id, sdate, uh_id, cab_id
    ORDER BY ctr_id, sdate
  ) AS CONFIGS_W_CAB_UH_STS_SHR
ORDER BY ctr_id, sdate, cab_id NULLS FIRST;
```

Varaible `cab_uh_sts_shr` is a column in the Configuration view and essential to determine whether the UH constitutes an open veto point vis-à-vis the government in a given political configuration.

NULL-values might stem from different missings, such as

- no upper house configuration correspoonds the given cabinet or vice-versa; or
- cabinet parties cannot be identified in table Upper House Seat Results (i.e., mismatch of `pty_ids`).

3.3.3 Cabinet's Total Number of Seats

View `view_cab_sts_ttl` is based on tables Cabinet and Cabinet Portfolios, and provides information at the level of cabinets.

It computes the total number of cabinet parties in a given cabinet.

View `view_cab_sts_ttl` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_cab_sts_ttl
AS
SELECT cab_id, cab_sts_ttl_computed
  FROM
    (SELECT cab_id
      FROM config_data.cabinet
    ) CABINET
  LEFT JOIN
    (SELECT cab_id, COUNT(pty_cab) AS cab_sts_ttl_computed
      FROM config_data.cabinet_portfolios
    WHERE pty_cab IS TRUE
    GROUP BY cabinet_portfolios.cab_id
    ) CABINET_PORTFOLIOS
  USING (cab_id)
ORDER BY cabinet.cab_id;
```

Variable `cab_sts_ttl` is essential to determine the number of partisan veto players in a given political configuration.

3.3.4 Number of Cabinet Parties

View `view_pty_cab_sts` is based on tables Cabinet Portfolios and provides data at the level of cabinets.

It computes the number of parties in a given cabinet.

View `view_pty_cab_sts` is programmed as follows:

```
CREATE VIEW config_data.view_pty_cab_sts
AS
SELECT cab_id, COUNT(cab_id) AS pty_cab_sts
  FROM config_data.cabinet_portfolios
  WHERE pty_cab IS TRUE
  GROUP BY cab_id
ORDER BY cab_id;
```

Note: No difference to `view_cab_sts_ttl` (3.3.3).

3.3.5 Configuration Events

View `view_configuration_events` is based on tables Cabinet, Lower House, Upper, House, and Presidential Elections, and provides the primary information on political configurations, namely country identifiers, a configurations start date, and the identifiers of respective corresponding institutions.

Accordingly, every new row corresponds to a historically unique political configuration among a country's government, lower house, upper house and the position of the Head of State, and a configuration is uniquely identified by combinations of `ctr_id`, `cab_id`, `lh_id`, `uh_id` (if applies), and `prs_id` (if applies).

Yet, because configuration start dates are identical with the start date of the institution the most recent change occured, political configurations are also uniquely identified by combinations of `ctr_id` and `sdate`).

View `view_configuration_events` thus sequences changes in the political-institutional configurations of a country by date. A new political configuration is recorded when one of the following changes occurs at one point in time during the respective period of coverage of a given country:

- A change in cabinet composition (rows in table Cabinet, identified by `cab_id` or unique combinations of `cab_sdate` and `ctr_id`).
- A change in lower house composition (rows in rable Lower House, identified by `lh_id` or unique combinations of `lh_sdate` and `ctr_id`).
- If exists in the respective country, a change in upper house composition (rows in table Upper House, identified by `uh_id` or unique combination of `uh_sdate` and `ctr_id`).
- If exists in the respective country, a change in presidency (rows in table Presidential Election, identified by `prselc_id` or unique combination of `prs_sdate` and `ctr_id`).

Hence, changes in political configurations are either due to a change in the partisan composition of some institution, i.e., a change in the (veto-)power relations *within* the institution, and consquently reflect changes in the (veto-)power relations *between* the institutions.[4] Ot a new configuration is recorded due to party splits or merges, newly elected upper or lower houses, or new presidencies, that not necessarly affect the respective instituional veto potential visà-vis the government

View `view_configuration_events` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_configuration_events
2  AS
3  SELECT DISTINCT ctr_id, sdate, cab_id, lh_id, lhelc_id, uh_id, prselc_id,
4    DATE_PART('year', sdate)::NUMERIC AS year, NULL::DATE AS edate
5    FROM
6      (SELECT prselc_id, prs_sdate AS sdate, ctr_id
7        FROM config_data.presidential_election
8      ) AS PRES_ELEC
9    RIGHT OUTER JOIN
10     (SELECT *
11       FROM
12         (SELECT uh_id, uh_sdate AS sdate, ctr_id
13           FROM config_data.upper_house
14         ) AS UH
15       RIGHT OUTER JOIN
16         (SELECT *
17           FROM
18             (SELECT lh_id, lh_sdate AS sdate, lhelc_id, ctr_id
19               FROM config_data.lower_house
20             ) AS LH
21           RIGHT OUTER JOIN
22             (SELECT *
```

[4] Cases where ...constitute exceptions.

```
23                FROM
24                  (SELECT cab_id, cab_sdate AS sdate, ctr_id
25                    FROM config_data.cabinet
26                  ) AS CAB
27                RIGHT OUTER JOIN
28                  (
29                  SELECT cab_sdate AS sdate, ctr_id
30                    FROM config_data.cabinet
31                  UNION
32                  SELECT lh_sdate AS sdate, ctr_id
33                    FROM config_data.lower_house
34                  UNION
35                  SELECT uh_sdate AS sdate, ctr_id
36                    FROM config_data.upper_house
37                  UNION
38                  SELECT prs_sdate AS sdate, ctr_id
39                    FROM config_data.presidential_election
40                  ORDER BY ctr_id, sdate NULLS FIRST
41                  ) AS START_DATES
42                USING(ctr_id, sdate)
43              ) AS CAB_JOIN
44            USING(ctr_id, sdate)
45          ) AS LH_JOIN
46        USING(ctr_id, sdate)
47      ) AS UH_JOIN
48    USING(ctr_id, sdate)
49  ORDER BY ctr_id, sdate;
```

Note: Rows are reported for all temporally corresponding combinations of institutional-political configurations. Thus, no institution correspond to the very first institutional configuration that is recorded in the PCDB, resulting in rows with many non-trivial missings in countries' first configurations. Example 1 illustrates this for the Australian case.

Example 1: First Australian configurations with incomplete correspondence of institutions.

ctr_id	sdate	prselc_id	uh_id	lh_id	cab_id
1	1946-09-28		1001	1001	
1	1946-11-01				1001
1	1947-07-01		1002		

Apparently, no the first recorded Australian cabinet startef on November 1st, 1946; thus, no corresponding cabinet can be assigned to the first recorded lower house and upper house configuration (first row). This makes it impossible to determine veto constellations for the very first row, resulting in missing information.

From the conceptional point of view, these incomplete configurations generally provide no information on the institutional-political setting of legislation. However, to provide an overview on countries' political history these *incomplete configurations* are reported. It is up to the user to anticipate potential merging problems.

View `view_configuration_events` is used to create an identical *materialized* view (see section 3.4), which is, in turn, used to trigger-in configuration end dates (see subsection 3.6.3) and corresponding institution identifier (see subsection **??**).

3.3.6 Configuration Year Duplicates

View `view_configuration_year_duplicates` is based on materialized view Configuration Events, which matches temporally corresponding configurations of cabinets, lower houses, upper houses, and presidencies.

It compiles primary configuration information (country identifier and configuration start and end dates) for the year of a configuration's start as well as for the year of its end, provided they are not identical.

View `view_configuration_year_duplicates` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_configuration_year_duplicates
AS
SELECT DISTINCT ctr_id, in_year,
    sdate, edate,
    DATE_PART('year', sdate)::INT AS start_in_year,
    DATE_PART('year', edate)::INT AS end_in_year,
    NULL::INT AS config_duration_in_year
FROM
(SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT = DATE_PART('year', edate)::INT
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT = (DATE_PART('year', edate)::INT)-1
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-1
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+1 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT = (DATE_PART('year', edate)::INT)-1
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+1 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-1
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+2 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-2
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+3 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-3
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+4 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-4
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+5 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-5
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+6 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-6
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+7 AS in_year
  FROM config_data.mv_configuration_events
  WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-7
UNION
SELECT ctr_id, sdate, edate, DATE_PART('year', sdate)::INT+8 AS in_year
```

```
54    FROM config_data.mv_configuration_events
55    WHERE DATE_PART('year', sdate)::INT < (DATE_PART('year', edate)::INT)-8
56  UNION
57  SELECT ctr_id, sdate, edate, DATE_PART('year', edate)::INT AS in_year
58    FROM config_data.mv_configuration_events
59  ) AS CONFIG_YEAR_DUPLICATES
60  WHERE in_year IS NOT NULL
61  ORDER BY ctr_id, in_year, sdate;
62
63  -- the weakness of this procedure is that if a configuration durateed more than 10 years, only 10 a
```

View `view_configuration_year_duplicates` is essential to compute configruations' duration in a year (subsection 3.3.7).

3.3.7 Configuration Duration in Year

View `view_configuration_duration_in_year` is based on view Configuration Year Duplicates and provides information at the level of configuration-in-years.

It computes configurations' duration in the year of its start date and the year of its end date. Obviously, if a configuration started and ended in the same year, its duration in the given year is equal to the difference in days between both dates. If, however, a configuration started in another year than it ended, its duration in the year of its starty equals the count of days from its start date to the first day of the next year, and the count of days from the last day of the previous year to the end date for the year of its end for its duration in the year it ended.

In addition, it provides primary configuration information (country identifier and configuration start and end dates) for the year of a configuration's start as well as for the year of its end, given they are not identical.

View `view_configuration_duration_in_year` is programmed as follows:

```
1   CREATE OR REPLACE VIEW config_data.view_configuration_duration_in_year
2   AS
3   SELECT ctr_id, sdate, edate, in_year,
4     config_duration_in_year
5   FROM
6   (
7   SELECT ctr_id, sdate, edate, start_in_year AS in_year,
8     ((edate+1)-sdate)::INT AS config_duration_in_year
9     FROM config_data.view_configuration_year_duplicates
10    WHERE start_in_year = end_in_year
11  UNION
12  SELECT ctr_id, sdate, edate, start_in_year AS in_year,
13    (TO_TIMESTAMP(''|| start_in_year::INT+1
14    ||'-01-01', 'YYYY-MM-DD')::DATE-sdate)
15      AS config_duration_in_year
16    FROM config_data.view_configuration_year_duplicates
17    WHERE start_in_year < end_in_year
18  UNION
19  SELECT ctr_id, sdate, edate, end_in_year AS in_year,
20    (edate-TO_TIMESTAMP(''|| end_in_year::INT-1
21    ||'-12-31', 'YYYY-MM-DD')::DATE)
22      AS config_duration_in_year
23    FROM config_data.view_configuration_year_duplicates
24    WHERE start_in_year < end_in_year
25  UNION
26  SELECT ctr_id, sdate, edate, in_year,
27    (SELECT count(*)
```

```
28      FROM   generate_series(
29      TO_TIMESTAMP(''|| in_year::INT ||'-01-01', 'YYYY-MM-DD')::DATE,
30      TO_TIMESTAMP(''|| in_year::INT ||'-12-31', 'YYYY-MM-DD')::DATE,
31      '1 day') d(the_day)
32    ) AS config_duration_in_year
33    FROM config_data.view_configuration_year_duplicates
34    WHERE in_year != start_in_year
35    AND in_year != end_in_year
36  ORDER BY ctr_id, in_year, sdate
37  ) AS CONFIG_YEAR_DUPLICATES;
```

Note: Configurations end date is implemented by a trigger on materialized view Configuration Events (see subsection 3.6.3), which selects the start date of the next configuration within country and substracts one day from it.

3.3.8 Configuration Weight in Year

View `view_configuration_weight_in_year` is based on view Configuration Year Duplicates (3.3.6) and provides information at the level of configuration-in-years.

It computes the temporal weights of a configuration in both the year of its start and the year of its end date, if not identical). Weights are equal to the days of a configuration's duration in year relative to the days of duration of the year (regularly 365 days, except from leap years, and years of a country's first and last recorded configurations).

View `view_configuration_weight_in_year` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_config_weight_in_year
2  AS
3  SELECT ctr_id, sdate, year, config_duration_in_year, year_duration,
4    (config_duration_in_year::NUMERIC/year_duration::NUMERIC) AS config_weight_in_year
5    FROM
6      (SELECT ctr_id, sdate, config_duration_in_year, in_year AS year
7        FROM config_data.view_configuration_duration_in_year
8      ) AS CONFIGURATION_DURATION_IN_YEAR
9    LEFT OUTER JOIN
10     (SELECT ctr_id, SUM(config_duration_in_year) AS year_duration, in_year AS year
11       FROM config_data.view_configuration_duration_in_year
12       GROUP BY ctr_id, in_year
13     ) AS YEAR_DURATION
14   USING(ctr_id, year);
```

Temporal weights are essential to determine configurations relative dominance in a given year, for instance, when compiling a country-year dataset.

3.3.9 Configuration Cohabitation

View `view_configuration_cohabitation` is based on tables Presidential Elections and Cabinet Portfolios, and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether the president/Head of State (HoS) is in cohabitation with the government, that is, if he or she not affiliated with one of the cabinet parties (indicated by binary variable `cohabitation`).

View view_configuration_cohabitation is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_configuration_cohabitation
AS
SELECT ctr_id, sdate, least(in_cohabitation) AS cohabitation
  FROM
    (SELECT *, abs(sign(pty_id-pty_id_hos)) AS in_cohabitation
      FROM
        (SELECT *
          FROM
            (SELECT ctr_id, sdate, cab_id, prselc_id
              FROM config_data.mv_configuration_events
            ) AS CONFIG_EVENTS
          RIGHT OUTER JOIN
            (SELECT cab_id, pty_id
              FROM config_data.cabinet_portfolios
              WHERE pty_cab IS TRUE
            ) AS ALL_CAB_PARTIES
          USING(cab_id)
        ) AS CONFIG_EVENTS_w_ALL_CAB_PARTIES
      FULL OUTER JOIN
        (SELECT prselc_id, pty_id AS pty_id_hos
          FROM config_data.presidential_election
        ) AS PTY_ID_HOS
      USING(prselc_id)
      WHERE prselc_id IS NOT NULL
      ORDER BY ctr_id, sdate
    ) AS CAB_PTY_HOS_PTY
  GROUP BY ctr_id, sdate, in_cohabitation
  ORDER BY ctr_id, sdate;
```

Note: Variable cohabitation can only be computed if a configuration enlist both a cabinet identifier (providing for information on cabinet parties) and a corresponding presidential election identifier (providing for information of the president's/HoS' party affiliation).

view_configuration_cohabitation is essential to identify presidents/HoSs that constitute open veto points vis-à-vis the government (see view_configuration_vto_prs).

3.3.10 Partisan Veto Players

View view_configuration_vto_pts is based on view Cabinet's Seat Total (3.3.3) and materialized view Configuration Events, and provides information at the level of political configurations.

It computes the number of partisan veto players in a given configuration.

View view_configuration_vto_pts is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_configuration_vto_pts
AS
SELECT ctr_id, sdate, cab_id, (cab_sts_ttl_computed-1)::SMALLINT AS vto_pts
FROM
  (SELECT  ctr_id, sdate, cab_id
    FROM config_data.mv_configuration_events
  ) AS CONFIG_EVENTS
JOIN
  (SELECT cab_id, cab_sts_ttl_computed
    FROM config_data.view_cab_sts_ttl
  ) AS CAB_STS_TTL
USING(cab_id)
ORDER BY ctr_id, sdate NULLS FIRST;
```

3.3.11 Lower House Veto

View `view_configuration_vto_lh` is based on table Veto Points, view Cabinet's Seat Share in the Lower House (3.3.1), and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether the lower house constitutes an open veto point vis-à-vis the government in a given configuration by comparing cabinet's seat share in the temproally corresponding lower house with decisive threshold enlisted in table Veto Points.

To guarantee that the computation of the lower houses veto potential is sensitive to constitutional changes, joining poltitical configurations with veto inforamtion is proceeded by date and country. Computation of the lower house's veto power in a given configuration is therefore up-to-date according to the inforamtion recorded in the Veto Points table.

View `view_configuration_vto_lh` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_configuration_vto_lh
2  AS
3  SELECT VETO_INST.ctr_id, sdate,
4    cab_id, lh_id, cab_lh_sts_shr, vto_pwr AS vto_pwr_lh,
5    SIGN(SIGN(vto_pwr-(cab_lh_sts_shr+0.00001))+1)::SMALLINT AS vto_lh
6  FROM
7    (SELECT *
8      FROM
9        (SELECT  ctr_id, sdate, cab_id, lh_id
10         FROM config_data.mv_configuration_events
11       ) AS CONFIG_EVENTS
12     JOIN
13       (SELECT ctr_id, sdate, cab_id, lh_id, cab_lh_sts_shr
14         FROM config_data.view_cab_lh_sts_shr
15       ) AS CAB_LH_STS_SHR
16     USING(ctr_id, sdate, cab_id, lh_id)
17   ) AS CONFIG_EVENTS_w_CAB_LH_STS_SHR
18 ,
19   (SELECT ctr_id, vto_pwr, vto_inst_sdate, vto_inst_edate
20     FROM config_data.veto_points
21     WHERE vto_inst_typ = 'lower house'
22   ) AS VETO_INST
23 WHERE CONFIG_EVENTS_w_CAB_LH_STS_SHR.ctr_id = VETO_INST.ctr_id
24 AND CONFIG_EVENTS_w_CAB_LH_STS_SHR.sdate >= VETO_INST.vto_inst_sdate
25 AND CONFIG_EVENTS_w_CAB_LH_STS_SHR.sdate <= VETO_INST.vto_inst_edate
26 ORDER BY ctr_id, sdate NULLS FIRST;
```

Note: Substracting the total seat share of cabinet parties in the lower house from the respective veto power threshold of lower houses results in a positive value when the former is smaller than the latter, for instance, in the case of a minority government in a parliamentary system.

To guarantee that the binary variable `vto_lh` indicates a closed veto point even when the government holds a seat share equal to 50 percent in the lower house, and thus equals the veto power threshold (e.g. cab_lh_sts_shr = 50.0), the total seat share of cabinet parties in lower house is increased by an abitrarly small value ($1e^{-5}$) that does not effect the computation substantially.

Apparently, a lower house's veto potential in a given configuration can only be determined where full information on the veto institution's start and end date as well as on the respective veto power threshold exists in table Veto Points.

3.3.12 Upper House Veto

View `view_configuration_vto_uh` is based on table Veto Points, view Cabinet's Seat Share in the Upper House (3.3.2), and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether the upper house constitutes an open veto point vis-à-vis the government in a given configuration by comparing cabinet's seat share in the temproally corresponding lower house with decisive threshold enlisted in table Veto Points.

To guarantee that the computation of the upper houses veto potential is sensitive to constitutional changes, joining poltitical configurations with veto inforamtion is proceeded by date and country. Computation of the upper house's veto power in a given configuration is therefore up-to-date according to the inforamtion recorded in the Veto Points table.

View `view_configuration_vto_lh` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_configuration_vto_uh
2  AS
3  SELECT VETO_INST.ctr_id, sdate,
4    cab_id, uh_id, cab_uh_sts_shr, vto_pwr AS vto_pwr_uh,
5    SIGN(SIGN(vto_pwr-(cab_uh_sts_shr+0.00001))+1)::SMALLINT AS vto_uh
6  FROM
7    (SELECT *
8      FROM
9        (SELECT  ctr_id, sdate, cab_id, uh_id
10         FROM config_data.mv_configuration_events
11       ) AS CONFIG_EVENTS
12     JOIN
13       (SELECT ctr_id, sdate, cab_uh_sts_shr
14         FROM config_data.view_cab_uh_sts_shr
15       ) AS CAB_UH_STS_SHR
16     USING(ctr_id, sdate)
17   ) AS CONFIG_EVENTS_w_CAB_UH_STS_SHR
18 ,
19   (SELECT ctr_id, vto_pwr, vto_inst_sdate, vto_inst_edate
20     FROM config_data.veto_points
21     WHERE vto_inst_typ = 'upper house'
22   ) AS VETO_INST
23 WHERE CONFIG_EVENTS_w_CAB_UH_STS_SHR.ctr_id = VETO_INST.ctr_id
24 AND CONFIG_EVENTS_w_CAB_UH_STS_SHR.sdate >= VETO_INST.vto_inst_sdate
25 AND CONFIG_EVENTS_w_CAB_UH_STS_SHR.sdate <= VETO_INST.vto_inst_edate
26 ORDER BY ctr_id, sdate NULLS FIRST;
```

Note: Substracting the total seat share of cabinet parties in the upper house from the respective veto power threshold of upper houses results in a positive value when the former is smaller than the latter, for instance, in the case of a minority government in a parliamentary system.

To guarantee that the binary variable `vto_uh` indicates a closed veto point even when the government holds a seat share equal to 50 percent in the upper house, and thus equals the veto power threshold (e.g. cab_uh_sts_shr = 50.0), the total seat share of cabinet parties in upper house is increased by an abitrarly small value ($1e^{-5}$) that does not effect the computation substantially.

Apparently, a lower house's veto potential in a given configuration can only be determined where full information on the veto institution's start and end date as well as on the respective veto power threshold exists in table Veto Points.

3.3.13 Presidents' Veto

View `view_configuration_vto_prs` is based on tables Presidential Elections, Cabinet Portfolios, Veto Points, and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether the president/Head of State (HoS) constitutes an open veto point vis-à-vis the government in a given configuration by checking for cohabitation and whether the constitution assigns veto power to the president.

To guarantee that the computation of the lower houses veto potential is sensitive to constitutional changes, joining poltitical configurations with veto inforamtion is proceeded by date and country. Computation of the HoS's veto power in a given configuration is therefore up-to-date according to the inforamtion recorded in the Veto Points table.

View `view_configuration_vto_prs` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_configuration_vto_prs
AS
SELECT CONFIG_EVENTS_COHABITATION.ctr_id, sdate, cohabitation, vto_pwr,
  (cohabitation*vto_pwr)::SMALLINT AS vto_prs
  FROM
    (SELECT ctr_id, sdate, LEAST(in_cohabitation) AS cohabitation
      FROM
        (SELECT *, ABS(SIGN(pty_id-pty_id_hos)) AS in_cohabitation
          FROM
            (SELECT *
              FROM
                (SELECT ctr_id, sdate, cab_id, prselc_id
                  FROM config_data.mv_configuration_events
                ) AS CONFIG_EVENTS
              FULL OUTER JOIN
                (SELECT cab_id, pty_id
                  FROM config_data.cabinet_portfolios
                  WHERE pty_cab IS TRUE
                ) AS ALL_CAB_PARTIES
              USING(cab_id)
            ) AS CONFIG_EVENTS_w_ALL_CAB_PARTIES
          FULL OUTER JOIN
            (SELECT prselc_id, pty_id AS pty_id_hos
              FROM config_data.presidential_election
            ) AS PTY_ID_HOS
          USING(prselc_id)
          WHERE prselc_id IS NOT NULL
        ) AS CAB_PTY_HOS_PTY
      GROUP BY ctr_id, sdate, in_cohabitation
    ) AS CONFIG_EVENTS_COHABITATION
  ,
    (SELECT ctr_id, vto_pwr, vto_inst_sdate, vto_inst_edate
      FROM config_data.veto_points
      WHERE vto_inst_typ = 'head of state'
    ) AS VETO_INST
  WHERE CONFIG_EVENTS_COHABITATION.ctr_id = VETO_INST.ctr_id
  AND CONFIG_EVENTS_COHABITATION.sdate >= VETO_INST.vto_inst_sdate
  AND CONFIG_EVENTS_COHABITATION.sdate <= VETO_INST.vto_inst_edate
  ORDER BY ctr_id, sdate NULLS FIRST;
```

3.3.14 Judiciary's Veto

View `view_configuraion_vto_jud` is based on table Veto Points and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether the judiciary constitutes an open veto point vis-à-vis the government in a given configuration.

For veto power of the judiciary is dependent constitutional provision, joining poltitical configurations with veto inforamtion is proceeded by date and country. Computation of the judiciary's veto power in a given configuration is therefore up-to-date according to the inforamtion recorded in the Veto Points table.

View `view_configuraion_vto_jud` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_configuration_vto_jud
2  AS
3  SELECT CONFIG_EVENTS.ctr_id, sdate, ROUND(vto_pwr)::SMALLINT AS vto_jud
4  FROM
5    (SELECT ctr_id, sdate
6      FROM config_data.mv_configuration_events
7    ) AS CONFIG_EVENTS
8  ,
9    (SELECT ctr_id, vto_pwr, vto_inst_sdate, vto_inst_edate
10     FROM config_data.veto_points
11     WHERE vto_inst_typ = 'judicial'
12   ) AS VETO_INST
13 WHERE CONFIG_EVENTS.ctr_id = VETO_INST.ctr_id
14 AND CONFIG_EVENTS.sdate >= VETO_INST.vto_inst_sdate
15 AND CONFIG_EVENTS.sdate <= VETO_INST.vto_inst_edate
16 ORDER BY ctr_id, sdate NULLS FIRST;
```

3.3.15 Electorate's Veto

View `view_configuraion_vto_elec` is based on table Veto Points and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether the electorate constitutes an open veto point vis-à-vis the government in a given configuration.

For veto power of the electorate is dependent constitutional provision, joining poltitical configurations with veto inforamtion is proceeded by date and country. Computation of the electorate's veto power in a given configuration is therefore up-to-date according to the inforamtion recorded in the Veto Points table.

View `view_configuraion_vto_elec` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_configuration_vto_elct
2  AS
3  SELECT CONFIG_EVENTS.ctr_id, sdate, ROUND(vto_pwr)::SMALLINT AS vto_elct
4  FROM
5    (SELECT ctr_id, sdate
6      FROM config_data.mv_configuration_events
7    ) AS CONFIG_EVENTS
8  ,
9    (SELECT ctr_id, vto_pwr, vto_inst_sdate, vto_inst_edate
10     FROM config_data.veto_points
```

```
11      WHERE vto_inst_typ = 'electoral'
12    ) AS VETO_INST
13  WHERE CONFIG_EVENTS.ctr_id = VETO_INST.ctr_id
14  AND CONFIG_EVENTS.sdate >= VETO_INST.vto_inst_sdate
15  AND CONFIG_EVENTS.sdate <= VETO_INST.vto_inst_edate
16  ORDER BY ctr_id, sdate NULLS FIRST;
```

3.3.16 Territorial Units' Veto

View `view_configuraion_vto_terr` is based on table Veto Points and materialized view Configuration Events, and provides information at the level of political configurations.

It computes whether territorial units constitute an open veto point vis-à-vis the government in a given configuration.

For veto power of territorial units (e.g., in ...) is dependent constitutional provision, joining poltitical configurations with veto inforamtion is proceeded by date and country. Computation of the territrial units' veto power in a given configuration is therefore up-to-date according to the inforamtion recorded in the Veto Points table.

View `view_configuraion_vto_terr` is programmed as follows:

```
1   CREATE OR REPLACE VIEW config_data.view_configuration_vto_terr
2   AS
3   SELECT CONFIG_EVENTS.ctr_id, sdate, ROUND(vto_pwr)::SMALLINT AS vto_terr
4   FROM
5     (SELECT ctr_id, sdate
6       FROM config_data.mv_configuration_events
7     ) AS CONFIG_EVENTS
8   ,
9     (SELECT ctr_id, vto_pwr, vto_inst_sdate, vto_inst_edate
10      FROM config_data.veto_points
11      WHERE vto_inst_typ = 'territorial'
12    ) AS VETO_INST
13  WHERE CONFIG_EVENTS.ctr_id = VETO_INST.ctr_id
14  AND CONFIG_EVENTS.sdate >= VETO_INST.vto_inst_sdate
15  AND CONFIG_EVENTS.sdate <= VETO_INST.vto_inst_edate
16  ORDER BY ctr_id, sdate NULLS FIRST;
```

3.3.17 Lower House Total Seats

View `view_lh_sts_ttl_computed` is based on tables Lower House and Lower House Seat Results, and provides data at the level of lower houses.

It computes the total number of seats in a lower house as the sum of seats distributed in a corresponding lower house election.

View `view_lh_sts_ttl_computed` is programmed as follows:

```
1   CREATE OR REPLACE VIEW config_data.view_lh_sts_ttl_computed
2   AS
3   SELECT lh_id, SUM(COALESCE(pty_lh_sts_pl,0)+COALESCE(pty_lh_sts_pr,0))::NUMERIC
4         AS lh_sts_ttl_computed
5     FROM config_data.lh_seat_results
6     GROUP BY lh_id
7   ORDER BY lh_id;
```

Note: The computed figure might deviate from the recorded total of lower house seats (cf. **??**).

3.3.18 Effective Number of Parties in Parliament, Minimum Fragmentation

View `view_lh_enpp_minfrag` is based on table Lower House and Lower House Seat Results, and provides data at the level of lower houses.

The effective number of parties in parliament (ENPP) is a measure of party system fractionalization that takes into acount the relative size of parties present in a country's lower house.

Variable `lh_enpp_minfrag` is computed based on the formula originally proposed by Laakso and Taagepera (1979)

$$ENPP_{minfrag}(k) = 1/\sum_{j=1}^{J} s_{j,k}^2 \tag{3.1}$$

, where k denotes a country's lower house at a given point in time, J are parties in a given lower house k, and s is party j's seat share in the kth lower house.

The suffix `_minfrag` points to the fact that Laakso & Taagepera's original formular lumps small parties or single representatives in the parliement into single categories (here the categories 'Others with seats' [otherw] and 'Independents' [IND]). This is equivalent to assume minimum fragmentation, other parties and independents enter into the calculation as if it were a single party, and thus tend to increase the fractionalization indice only marignally. Apparantly, this likley results in an underestimate of fragmentation (Gallagher and Mitchell, 2005).

The PCDB provides for an alternative ENPP indice that adjusts for this tendency.

View `view_lh_enpp_minfrag` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_lh_enpp_minfrag
2  AS
3  SELECT lh_id, 1/SUM(pty_lh_sts_shr^2.0) AS lh_enpp_minfrag
4    FROM
5      (SELECT lh_id, pty_id,
6        (SEATS.pty_lh_sts/SEATS_TOTAL.lhelc_sts_ttl_computed)
7          AS pty_lh_sts_shr
8        FROM
9          (SELECT lh_id,
10           SUM(pty_lh_sts)::NUMERIC AS lhelc_sts_ttl_computed
11           FROM config_data.lh_seat_results
12           GROUP BY lh_id
13         ) AS SEATS_TOTAL
14       JOIN
15         (SELECT lh_id, pty_id, pty_lh_sts::NUMERIC
16           FROM config_data.lh_seat_results
17           WHERE pty_lh_sts <> 0
18         ) AS SEATS
19       USING(lh_id)
20       ORDER BY lh_id, pty_id
21     ) AS SEAT_SHR
22   GROUP BY lh_id
23   ORDER BY lh_id;
```

Note: Computation of the ENPPs is proceeded with the computed, not the recorded total number of Lower House seats (note that the computed sum of seats might deviate from the recorded figure in table Lower House; cf. `cc_lhelc_sts_ttl`).

3.3.19 Effective Number of Parties in Parliament, Maximum Fragmentation

View `view_lh_enpp_maxfrag` is based on tables Lower House and Lower House Seat Results, view ENPP Adjustment Parameter (m) (3.3.20), and provides data at the level of lower houses.

The effective number of parties in parliament (ENPP) is a measure of party system fractionalization that takes into acount the relative size of parties present in a country's lower house.

Variable `lh_enpp_maxfrag` adjusts for the tendency of underestmating fractionalization of lower houses that implicite in Laakso and Taagepera's original formular (Equ 3.1).

It employs what Gallagher and Mitchell (2005, pp. 600-602) refer to as 'Taagepera's least component approach': The seat share of the groups 'Others with seats' (otherw) and 'indpendents' (INDs) are split into m fractions each, resulting in m seat shares of size s_m.

The fromula to compute `lh_enpp_maxfrag` is

$$ENPP_{maxfrag}(k) = 1/\sum_{j=1}^{J} m \left(\frac{s_{j,k}}{m}\right)^2 \tag{3.2}$$

, where m is computed by dividing the number of seats of otherw or that of INDs by the number of seats of the smallest 'real'[5] party in the respective lower house, and upround to the next bigger integer value, to guarantee that the seat share of otherw and/or of INDs are smaller than that of the smallest 'real' party. In fact, this procedure implies assuming maximum fragmentation.

View `view_lh_enpp_maxfrag` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_lh_enpp_maxfrag
2  AS
3  SELECT lh_id,
4  1/SUM(COALESCE(lh_m_upround, 1)*
5    ((pty_lh_sts_shr/COALESCE(lh_m_upround, 1))^2))::NUMERIC
6    AS lh_enpp_maxfrag
7  FROM
8    (SELECT lh_id, pty_id,
9      (SEATS.pty_lh_sts/SEATS_TOTAL.lhelc_sts_ttl_computed)::NUMERIC
10       AS pty_lh_sts_shr
11     FROM
12       (SELECT lh_id, SUM(pty_lh_sts)::NUMERIC AS lhelc_sts_ttl_computed
13         FROM config_data.lh_seat_results
14         GROUP BY lh_id
15       ) AS SEATS_TOTAL
16     JOIN
17       (SELECT lh_id, pty_id, pty_lh_sts
18         FROM config_data.lh_seat_results
```

[5] 'Real' in the sense that the respective party is identified by a counter different from ##997 or ##998 (see table Party).

```
19          WHERE pty_lh_sts > 0
20        ) AS SEATS
21      USING(lh_id)
22      ORDER BY lh_id, pty_id
23    ) AS PTY_LH_STS_SHRs
24  LEFT OUTER JOIN
25    (SELECT lh_m_upround, lh_id, pty_id
26      FROM config_data.view_lh_m
27    ) AS MULTIPLIERS
28  USING (lh_id, pty_id)
29  GROUP BY lh_id;
```

Note: Computation of the ENPP is proceeded with the computed, not the recorded total number of Lower House seats (note that the computed sum of seats might deviate from the recorded figure in table Lower House; cf. `cc_lhelc_sts_ttl`).

3.3.20 ENPP Adjustment Parameter (m)

View `view_lh_m` is based on table Lower House Seat Results and provides data at the level of lower houses.

It computes the parameter m, which is used to account for the tendency to underestimate the effective number of parties in parliament (ENPP) in Laakso and Taagrepera's original formular (Equ 3.1) in cases where the number of lower house seats hold by others and indpendents exceeds the number of sears hold by the smallest 'real' [6] party.

Specifically, Gallagher and Mitchell (2005, pp. 600-602) suggest to devide the total share of seats of the groups others and independents, respectively, in m parts of equal size s_m, so that the s_ms are smaller than the share of the smallest party (referred to as 'Taagepera's least component approach').

Accordingly, m is computed by dividing the seats of (a) the groups 'Others with seat' (Otherw) and/or 'Independents' (IND) hold in the lower house by the number of seats (b) the smallest 'real' party holds in the respective lower house. When (a) > (b), then $m > 1$. To guarantee that the m seat shares s_m of Otherw and/or IND is/are smaller than that of the smallest party, m is upround to the next bigger integer value. Lower House elections in which m is bigger than one are enlisted in `view_lhelc_w_underestimated_ENPP` (**??**).

View `view_lh_m` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_lh_m
2  AS
3  SELECT
4    lh_id,
5    lh_pty_w_least_seats,
6    pty_id,
7    lh_sts_of_Oth_or_INDs,
8    (lh_sts_of_Oth_or_INDs/lh_pty_w_least_seats)::NUMERIC AS lh_m,
9    CEIL(lh_sts_of_Oth_or_INDs/lh_pty_w_least_seats)::NUMERIC AS lh_m_upround
10   FROM
11     (SELECT lh_id, MIN(pty_lh_sts)::NUMERIC AS lh_pty_w_least_seats
12       FROM config_data.lh_seat_results
13       WHERE pty_lh_sts > 0
```

[6] 'Real' in the sense that the respective party is identified by a counter different from ##997 or ##998 (see table Party).

```
14        GROUP BY lh_id
15        ORDER BY lh_id
16      ) AS LH_PTY_w_LEAST_SEATS
17    LEFT OUTER JOIN
18      (SELECT lh_id, pty_id, pty_lh_sts::NUMERIC AS lh_sts_of_Oth_or_INDs
19        FROM config_data.lh_seat_results
20        WHERE pty_lh_sts > 0
21        AND pty_id
22        IN (SELECT pty_id
23          FROM config_data.view_Others_and_INDs
24        )
25      ) AS SEATS_of_OTHERS_or_INDs
26    USING(lh_id)
27  WHERE (lh_sts_of_Oth_or_INDs/lh_pty_w_least_seats) > 1
28  OR NOT NULL;
```

Note: The WHERE-condition ensures that only lower house elections are selected in which the amount of seat of Others with seats or Independents exceedes the amount of seats the party with least seats gained, as 'Taagepera's least component approach' to prevent from underestimation of ENPPs only needs to be applied to these cases.

3.3.21 Others and Independents

View `view_Others_and_INDs` is based on table Party and provides data at the level of parties.

It enlists the party identifiers of categories 'Others without seat' (Other), Others with seat (Otherw), and Independents (IND) for all countries.

View `view_Others_and_INDs` is programmed as follows:

```
1  CREATE VIEW config_data.view_Others_and_INDs
2  AS
3  SELECT pty_id
4    FROM
5      (SELECT pty_id, pty_abr
6        FROM config_data.party
7        WHERE pty_abr LIKE 'Other%'
8        OR pty_abr LIKE 'IND'
9      ) AS Others_and_INDs;
```

3.3.22 Lower House Election Effective Thresholds

View `view_lhelc_eff_thrshlds` is based on table Lower House Election and provides data at the level of lower house elections.

It computes the effective threshold in a given lower house election.

Variable `lhelc_eff_thrshld_lijphart1994` computes the threshold according to the definition provided by Lijphart (1994):

$$EffT_{Lijphart} = \frac{0.5}{m+1} + \frac{0.5}{2m} \tag{3.3}$$

, where m is the district magnitude.

Variable `lhelc_eff_thrshld_taagepera2002`, in contrast, computes the threshold according to the definition provided by Taagepera (2002, p. 309):

$$EffT_{Taagepera} = \frac{0.75}{n^2 + (S/n^2)} \tag{3.4}$$

, where S is the size of the lower house (i.e., the total number of seats), and n is the number of seat winning parties.

In the PCDB it is assumed that $n \approx \sqrt[4]{m * S}$. This yields

$$EffT_{PCDB} = \frac{0.75}{(m+1) * \sqrt{S/m}} \tag{3.5}$$

to compute variable `lhelc_eff_thrshld_pcdb`. , which is in fact identical with Taagepera's formular, if $n = \sqrt[4]{m * S}$.

View `view_lhelc_eff_thrshlds` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_lhelc_eff_thrshlds
AS
SELECT lhelc_id, ctr_id, lhelc_date, lhelc_sts_ttl, lhelc_dstr_mag,
  ((0.5/(lhelc_dstr_mag+1))
  + (0.5/(2*lhelc_dstr_mag)))::NUMERIC(7,5)
    AS lhelc_eff_thrshld_lijphart1994,
  (0.75/(((lhelc_dstr_mag*lhelc_sts_ttl)^0.25)^2
  + (lhelc_sts_ttl/((lhelc_dstr_mag*lhelc_sts_ttl)^0.25)^2)))::NUMERIC(7,5)
    AS lhelc_eff_thrshld_taagepera2002,
  (0.75/((lhelc_dstr_mag+1)*
  (lhelc_sts_ttl/lhelc_dstr_mag)^0.5))::NUMERIC(7,5)
    AS lhelc_eff_thrshld_pcdb
  FROM config_data.lh_election
ORDER BY lhelc_id, ctr_id, lhelc_date NULLS FIRST;
```

3.3.23 Lower House Election Disproportionality, Gallagher's LSq

View `view_lhelc_lsq` is based on table Lower House Election and provides data at the level of lower house elections.

It computes the Gallagher's (1991) Least-square index (LSq), which measures the dispoportionality in the distribution seats in a lower house election:

$$LSq_{Gallagher} = \sqrt{\frac{1}{2}\sum_{j=1}^{J}(v_j - s_j)^2} \tag{3.6}$$

, where j denotes parties, v vote and s seat shares gained in an election to the lower house.

The LSq weighs the deviations by their own value, creating a responsive index, ranging from 0 to 100. The lower the index value the lower the disproportionality and vice versa.

View `view_lhelc_lsq` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_lhelc_lsq
AS
SELECT lhelc_id, lhelc_lsq_computed
FROM
  (SELECT lhelc_id FROM config_data.lh_election) AS LH_ELECTIONS
LEFT OUTER JOIN
  (SELECT lhelc_id, (1/(0.5*SUM(pty_vts_sts_square)))::DOUBLE PRECISION AS lhelc_lsq_computed
    FROM
      (SELECT lhelc_id, pty_id,
        (SUM(pty_lh_vts_shr - pty_lh_sts_shr))^2.0 AS pty_vts_sts_square
        FROM
          (SELECT lhelc_id, pty_id, pty_lh_sts_shr
            FROM
              (SELECT lhelc_id, lh_id FROM config_data.lower_house) AS LH_LHELC
            JOIN
              (SELECT lh_id, pty_id,
                100*MAX(SEATS.pty_lh_sts_computed
                /SEATS_TOTAL.lhelc_sts_ttl_computed) AS pty_lh_sts_shr
                FROM
                  (SELECT lh_id,
                    NULLIF(SUM(pty_lh_sts_computed),0)::NUMERIC
                      AS lhelc_sts_ttl_computed
                    FROM config_data.view_pty_lh_sts_computed
                    GROUP BY lh_id
                  ) AS SEATS_TOTAL
                JOIN
                  (SELECT lh_id, pty_id, pty_lh_sts_computed::NUMERIC
                    FROM config_data.view_pty_lh_sts_computed
                  WHERE pty_id
                  NOT IN
                    (SELECT DISTINCT pty_id
                      FROM config_data.party
                      WHERE pty_abr LIKE 'Other'
                    )
                  ) AS SEATS
                USING(lh_id)
                WHERE pty_lh_sts_computed > 0 AND pty_lh_sts_computed IS NOT NULL
                GROUP BY lh_id, pty_id
              ) AS SEAT_SHR_IN_LH
            USING(lh_id)
          ) AS SEAT_SHR
        JOIN
          (SELECT lhelc_id, pty_id,
            100*MAX(VOTES.pty_lh_vts
            /VOTES_TOTAL.lhelc_vts_ttl) AS pty_lh_vts_shr
            FROM
              (SELECT lhelc_id,
                SUM(COALESCE(pty_lh_vts_pr,0)
                + COALESCE(pty_lh_vts_pl,0))::NUMERIC
                  AS lhelc_vts_ttl
                FROM config_data.lh_vote_results
                GROUP BY lhelc_id
                ORDER BY lhelc_id
              ) AS VOTES_TOTAL
            JOIN
              (SELECT lhelc_id, pty_id,
                (COALESCE(pty_lh_vts_pr,0)
                + COALESCE(pty_lh_vts_pl,0))::NUMERIC
                  AS pty_lh_vts
                FROM config_data.lh_vote_results
              ) AS VOTES
            USING(lhelc_id)
            WHERE pty_lh_vts > 0 AND pty_lh_vts IS NOT NULL
            GROUP BY lhelc_id, pty_id
          ) AS VOTES_SHR
        USING(lhelc_id, pty_id)
        GROUP BY lhelc_id, pty_id
        ORDER BY lhelc_id, pty_id
```

```
69        ) AS VOTE_SEAT_SQARES
70      WHERE lhelc_id
71      NOT IN
72        (SELECT DISTINCT lhelc_id
73        FROM
74          (SELECT * FROM
75            (SELECT lhelc_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
76              FROM
77                (SELECT lhelc_id, lh_id
78                  FROM config_data.lower_house
79                ) AS LH_LHELC
80              JOIN
81                (SELECT lh_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
82                  FROM config_data.lh_seat_results
83                ) AS SEAT_RESULTS
84              USING(lh_id)
85            ) AS LH_SEATS
86          JOIN
87            (SELECT pty_id, pty_abr
88              FROM config_data.party
89            ) AS PARTIES
90          USING(pty_id)
91          ) AS SEATS
92        JOIN
93          (SELECT * FROM
94            (SELECT lhelc_id, pty_id, pty_lh_vts_pr, pty_lh_vts_pl
95              FROM config_data.lh_vote_results
96            ) AS LH_SEATS
97          JOIN
98            (SELECT pty_id, pty_abr
99              FROM config_data.party
100           ) AS PARTIES
101         USING(pty_id)
102         ) AS VOTES
103       USING(lhelc_id, pty_id)
104       WHERE pty_lh_vts_pr IS NULL
105         AND pty_lh_vts_pl IS NULL
106         AND VOTES.pty_abr NOT LIKE '%Other'
107       OR pty_lh_sts_pr IS NULL
108         AND pty_lh_sts_pl IS NULL
109         AND SEATS.pty_abr NOT LIKE '%Other'
110       )
111     GROUP BY lhelc_id
112   ) AS VALID_LSQs
113   USING(lhelc_id)
114 ORDER BY lhelc_id;
```

Note: Variable `lhelc_lsq_computed` is cannot be computed for lower house elections in which (a) for at least one party with seat(s) neither proportional nor plurality vote results are recorded (cf. 3.7.14), or (b) neither proportional nor plurality seats are recorded, even though party is not identified as 'Other without seat', i.e. `pty_id` is not ##999 (cf. **??**). Consistency check `cc_missing_lhelc_pty_sts_records` (**??**) enlists all party-election configurations to which one or boths applies.

The PCDB also includes the variable `lhelc_lsq_noothers_computed`, which excludes the vote and seat shares listed for the category 'Others with seats' from computing the LSq.[7]

View `view_lhelc_lsq_noothers` is programmed as follows:

```
1 CREATE OR REPLACE VIEW config_data.view_lhelc_lsq_noothers
```

[7] Essentially, this is achieved by extending the `WHERE`-condition, requiering not only that `SEATS.pty_abr NOT LIKE '%Other'` but also that `SEATS.pty_abr NOT LIKE '%Others'`.

```
2  AS
3  SELECT lhelc_id, lhelc_lsq_noothers_computed
4  FROM
5    (SELECT lhelc_id
6      FROM config_data.lh_election
7    ) AS LH_ELECTIONS
8  LEFT OUTER JOIN
9    (SELECT lhelc_id,
10     (1/(0.5*SUM(pty_vts_sts_square)))::DOUBLE PRECISION AS lhelc_lsq_noothers_computed
11     FROM
12       (SELECT lhelc_id, pty_id,
13         (SUM(pty_lh_vts_shr - pty_lh_sts_shr))^2.0
14           AS pty_vts_sts_square
15         FROM
16           (SELECT lhelc_id, pty_id, pty_lh_sts_shr
17             FROM
18               (SELECT lhelc_id, lh_id FROM config_data.lower_house) AS LH_LHELC
19             JOIN
20               (SELECT lh_id, pty_id,
21                 100*MAX(SEATS.pty_lh_sts_computed
22                 /SEATS_TOTAL.lhelc_sts_ttl_computed) AS pty_lh_sts_shr
23                 FROM
24                   (SELECT lh_id,
25                     NULLIF(SUM(pty_lh_sts_computed),0)::NUMERIC
26                       AS lhelc_sts_ttl_computed
27                     FROM config_data.view_pty_lh_sts_computed
28                     GROUP BY lh_id
29                   ) AS SEATS_TOTAL
30                 JOIN
31                   (SELECT lh_id, pty_id, pty_lh_sts_computed::NUMERIC
32                     FROM config_data.view_pty_lh_sts_computed
33                   WHERE pty_id
34                   NOT IN
35                     (SELECT DISTINCT pty_id
36                       FROM config_data.party
37                       WHERE pty_abr LIKE 'Other'
38                     )
39                   ) AS SEATS
40                 USING(lh_id)
41                 WHERE pty_lh_sts_computed > 0 AND pty_lh_sts_computed IS NOT NULL
42                 GROUP BY lh_id, pty_id
43               ) AS SEAT_SHR_IN_LH
44             USING(lh_id)
45           ) AS SEAT_SHR
46         JOIN
47           (SELECT lhelc_id, pty_id,
48             100*MAX(VOTES.pty_lh_vts
49             /VOTES_TOTAL.lhelc_vts_ttl) AS pty_lh_vts_shr
50             FROM
51               (SELECT lhelc_id,
52                 SUM(COALESCE(pty_lh_vts_pr,0)
53                 + COALESCE(pty_lh_vts_pl,0))::NUMERIC
54                   AS lhelc_vts_ttl
55                 FROM config_data.lh_vote_results
56                 GROUP BY lhelc_id
57                 ORDER BY lhelc_id
58               ) AS VOTES_TOTAL
59             JOIN
60               (SELECT lhelc_id, pty_id,
61                 (COALESCE(pty_lh_vts_pr,0)
62                 + COALESCE(pty_lh_vts_pl,0))::NUMERIC
63                   AS pty_lh_vts
64                 FROM config_data.lh_vote_results
65               ) AS VOTES
66             USING(lhelc_id)
67             WHERE pty_lh_vts > 0 AND pty_lh_vts IS NOT NULL
68             GROUP BY lhelc_id, pty_id
69           ) AS VOTES_SHR
```

```
70          USING(lhelc_id, pty_id)
71          GROUP BY lhelc_id, pty_id
72          ORDER BY lhelc_id, pty_id
73        ) AS VOTE_SEAT_SQARES
74      WHERE lhelc_id
75      NOT IN
76        (SELECT DISTINCT lhelc_id
77        FROM
78          (SELECT * FROM
79            (SELECT lhelc_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
80              FROM
81                (SELECT lhelc_id, lh_id
82                  FROM config_data.lower_house
83                ) AS LH_LHELC
84              JOIN
85                (SELECT lh_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
86                  FROM config_data.lh_seat_results
87                ) AS SEAT_RESULTS
88              USING(lh_id)
89            ) AS LH_SEATS
90          JOIN
91            (SELECT pty_id, pty_abr
92              FROM config_data.party
93            ) AS PARTIES
94          USING(pty_id)
95          ) AS SEATS
96        JOIN
97          (SELECT * FROM
98            (SELECT lhelc_id, pty_id, pty_lh_vts_pr, pty_lh_vts_pl
99              FROM config_data.lh_vote_results
100           ) AS LH_SEATS
101         JOIN
102           (SELECT pty_id, pty_abr
103             FROM config_data.party
104           ) AS PARTIES
105         USING(pty_id)
106         ) AS VOTES
107       USING(lhelc_id, pty_id)
108       WHERE pty_lh_vts_pr IS NULL
109         AND pty_lh_vts_pl IS NULL
110         AND VOTES.pty_abr NOT LIKE '%Other'
111         AND VOTES.pty_abr NOT LIKE '%Otherw'
112       OR pty_lh_sts_pr IS NULL
113         AND pty_lh_sts_pl IS NULL
114         AND SEATS.pty_abr NOT LIKE '%Other'
115         AND SEATS.pty_abr NOT LIKE '%Otherw'
116       )
117     GROUP BY lhelc_id
118     ORDER BY lhelc_id
119   ) AS VALID_LSQs
120 USING(lhelc_id)
121 ORDER BY lhelc_id;
```

3.3.24 Type A Volatility in Lower House Seat Shares

View `view_lh_vola_sts` is based on tables Lower House and Lower House Seat Results, and provides data at the level of lower houses.

Generally, type A volatility measures volatility from party entry and exit to the political system and is quantified by the change that occurs in the distribution of shares between parties due to parties newly entering respectively retiering from the electoral arena (Powell and Tucker, 2013), majorly the domestic party system or the lower house.

Type A volatitlity in seats in a given lower house is defined as volatility in the distribution of seats arising from new entering and retiering parties, given by the formular:

$$Seat\,A\,Volatility\,(k) = \frac{|\sum_{n=1}^{New} s_{n,k} + \sum_{o=1}^{Old} s_{o,k}|}{2} \tag{3.7}$$

, where o refers to retiering parties that contested only the election $k-1$ and n to new-entering parties that contested only election k, and generally s is party's seat share in the lower house (i.e., the number of seats gained by party, divided by the total number of seats distributed between all parties J that entered the lower house k in the corresponding election).

View `view_lh_vola_sts` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_lh_vola_sts
AS
SELECT lh_id,
  (ABS(COALESCE(old_pty_sum, 0)
  + COALESCE(new_pty_sum, 0))/2)::DOUBLE PRECISION
    AS lh_vola_sts_computed
  FROM
    (SELECT lh_id, old_pty_sum
      FROM
        (SELECT lh_id
          FROM config_data.lower_house
        ) AS LH_CONFIGS
      LEFT OUTER JOIN
        (SELECT lh_nxt_id AS lh_id,
          SUM(old_pty_lh_sts_shr)
            AS old_pty_sum
          FROM
            (SELECT lh_id, lh_nxt_id
              FROM config_data.lower_house
            ) AS LOWER_HOUSE
          JOIN
            (SELECT lh_id, pty_id,
              100*MAX(SEATS.pty_lh_sts
              /SEATS_TOTAL.lh_sts_ttl_computed)
                AS old_pty_lh_sts_shr
              FROM
                (SELECT lh_id, sum(pty_lh_sts)::NUMERIC
                  AS lh_sts_ttl_computed
                  FROM config_data.lh_seat_results
                  GROUP BY lh_id
                ) AS SEATS_TOTAL
              JOIN
                (SELECT lh_id, pty_id, pty_lh_sts::NUMERIC
                  FROM config_data.lh_seat_results
                  WHERE lh_id
                  NOT IN
                    (SELECT max(lh_id) AS lh_id
                      FROM config_data.lower_house
                      GROUP BY ctr_id
                    )
                  AND pty_lh_sts >= 1
                  EXCEPT
                    SELECT lh_id, CUR_LH.pty_id AS pty_id, pty_lh_sts
                      FROM
                        (SELECT lh_id, pty_id, pty_lh_sts
                          FROM config_data.lh_seat_results
                        ) AS PREV_LH
                      ,
                        (SELECT lh_prv_id, pty_id
                          FROM
```

```
51                              (SELECT lh_id, lh_prv_id
52                                FROM config_data.lower_house
53                              ) AS LOWER_HOUSE
54                            JOIN
55                              (SELECT lh_id, pty_id
56                                FROM config_data.lh_seat_results
57                              ) AS LH_SEAT_RESULTS
58                            USING(lh_id)
59                          ) AS CUR_LH
60                        WHERE CUR_LH.lh_prv_id = PREV_LH.lh_id
61                        AND CUR_LH.pty_id = PREV_LH.pty_id
62                  ) AS SEATS
63                USING(lh_id)
64                GROUP BY lh_id, pty_id
65              ) AS OLD_PTY_LH_STS_PCT
66            USING(lh_id)
67          GROUP BY lh_nxt_id
68          ) AS RETIERING_PARTIES
69        USING (lh_id)
70      ) AS LH_RETIERING_PARTIES
71    LEFT OUTER JOIN
72      (SELECT lh_id,
73        SUM(new_pty_lh_sts_shr)
74          AS new_pty_sum
75        FROM
76          (SELECT lh_id, pty_id,
77            100*MAX(SEATS.pty_lh_sts
78            /SEATS_TOTAL.lh_sts_ttl_computed)
79              AS new_pty_lh_sts_shr
80            FROM
81              (SELECT lh_id,
82                SUM(pty_lh_sts)::NUMERIC
83                  AS lh_sts_ttl_computed
84                FROM config_data.lh_seat_results
85                GROUP BY lh_id
86              ) AS SEATS_TOTAL
87            JOIN
88              (SELECT lh_id, pty_id, pty_lh_sts::numeric
89                FROM config_data.lh_seat_results
90                WHERE lh_id
91                NOT IN
92                  (SELECT max(lh_id) AS lh_id
93                    FROM config_data.lower_house
94                    GROUP BY ctr_id)
95              AND pty_lh_sts >= 1
96              EXCEPT
97                SELECT lh_id, CUR_LH.pty_id AS pty_id, pty_lh_sts
98                  FROM
99                    (SELECT lh_id, pty_id, pty_lh_sts
100                     FROM config_data.lh_seat_results
101                   ) AS PREV_LH
102                 ,
103                   (SELECT lh_nxt_id, pty_id
104                     FROM
105                       (SELECT lh_id, lh_nxt_id
106                         FROM config_data.lower_house
107                       ) AS LOWER_HOUSE
108                     JOIN
109                       (SELECT lh_id, pty_id
110                         FROM config_data.lh_seat_results
111                       ) AS LH_SEAT_RESULTS
112                     USING(lh_id)
113                   ) AS CUR_LH
114               WHERE CUR_LH.lh_nxt_id = PREV_LH.lh_id
115               AND CUR_LH.pty_id = PREV_LH.pty_id
116             ) AS SEATS
117           USING(lh_id)
118           GROUP BY lh_id, pty_id
```

```
119           ) AS NEW_PTY_LH_STS_PCT
120         GROUP BY lh_id
121       ) AS NEWENTRY_PARTIES
122     USING(lh_id)
123   ORDER BY lh_id;
```

Because the SQL-syntax of `view_lhelc_vola_sts` is rather complex, some brief comments are instructive:

- The enumerator of Equ 3.7 consists of two summands; each is computed seperately (the parts embraced by paranthesis and labeled NEWENTRY_PARTIES and LH_RETIERING_PARTIES, respectively) and added only in the very first lines of the query.
- With respect to the subqueries, NEWENTRY_PARTIES aggregates the seat shares of parties that newly entered in the present lower house for the present lower house, and LH_RETIERING_PARTIES aggregates the seat shares of parties that entered the previous but not the current lower house.
- Exluding 'stable' parties (i.e., parties that entered the present as well as the previous lower house) within the subqueries is achieved by the EXCEPT-clauses, which pair parties recorded for the present and the previous lower house by party identifiers. If a party was only in the present lower house, or if it was in the previous but is not in present lower house, then it does not occur in the query that follows the EXCEPT-clauses. In consequence, only seats gained by new entering parties, and those lost by retiering parties enter the aggregation.
- Generally, joining parties' seat results with different combinations of the identifiers of the previous, the current, and the next lower house enables to easily identify new entering and retiering parties.

Note: No figures for first an last recorded elections in a given country are reported, because it is impossible to determin which parties are 'newcomers' in first and which parties will retier in last election, respectively. The exclusion of first lower house configurations is a consequence of the final left-outer join, as the first lower house reported by subquery NEWENTRY_PARTIES is always only the second for a given country. The excludsion of first lower house configurations, in turn, is achieved by selecting only parties seat results from lower house elections that have not the highest within country election identifier.[8]

3.3.25 Type B Volatility in Lower House Seat Shares

View `view_lhelc_volb_sts` is based on tables Lower House, Lower House Seat Results and Party, and provides data at the level of lower houses.

Type B volatility quantifies the change that occurs in the distribution of seat shares within parties in subseqent lower houses, comparing the results in the current to that of the previous

[8] specifically, that is quering ...`SELECT lhelc_id, pty_id, pty_lh_sts ...WHERE lhelc_id NOT IN (SELECT MAX(lhelc_id) ....` A feasible alternative would be programming the restriction based on selection of the election with the highest date within a country (would prevent from coding failures in the ordering of lower house election identifiers).

one. Accordingly, type B volatility considers only so-called stable parties and measures the volatility in the distribution of seats arising from gaines and losses of these stable parties.

The formular to compute `lh_volb_sts` is

$$Seat\,B\,Volatility\,(k) = \frac{\left|\sum_{j=1}^{Stable} s_{j,(k-1)} - s_{j,k}\right|}{2} \tag{3.8}$$

, where s are seat or vote shares that party j gained in the current lower house k or in the previous lower house $k-1$.

View `view_lh_volb_sts` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_lh_volb_sts
2  AS
3  SELECT lh_id, lh_volb_sts_computed
4    FROM
5      (SELECT lh_id
6        FROM config_data.lower_house
7      ) AS ALL_LOWER_HOUSES
8    LEFT OUTER JOIN
9      (SELECT lh_id,
10       (SUM(pty_lh_sts_pct_diff)/2)::DOUBLE PRECISION
11         AS lh_volb_sts_computed
12       FROM
13         (SELECT CUR_lh_STS_SHR.lh_id AS lh_id,
14           ABS(PREV_lh_STS_SHR.pty_prv_lh_sts_pct
15           - CUR_lh_STS_SHR.pty_cur_lh_sts_pct)
16             AS pty_lh_sts_pct_diff
17           FROM
18             (SELECT lh_id, pty_id,
19               100*(SEATS.pty_lh_sts
20               /SEATS_TOTAL.lh_sts_ttl_computed)
21                 AS pty_prv_lh_sts_pct
22               FROM
23                 (SELECT lh_id,
24                   SUM(pty_lh_sts)::NUMERIC
25                     AS lh_sts_ttl_computed
26                   FROM config_data.lh_seat_results
27                   GROUP BY lh_id
28                 ) AS SEATS_TOTAL
29               JOIN
30                 (SELECT lh_id, pty_id, pty_lh_sts::NUMERIC
31                   FROM config_data.lh_seat_results
32                   WHERE pty_lh_sts >= 1
33                 ) AS SEATS
34               USING(lh_id))
35             AS PREV_LH_STS_SHR
36           ,
37             (SELECT lh_id, lh_prv_id, pty_id,
38               100*(pty_lh_sts
39               /lh_sts_ttl_computed)
40                 AS pty_cur_lh_sts_pct
41               FROM
42                 (SELECT lh_id,
43                   SUM(pty_lh_sts)::NUMERIC
44                     AS lh_sts_ttl_computed
45                   FROM config_data.lh_seat_results
46                   GROUP BY lh_id
47                 ) AS SEATS_TOTAL
48               JOIN
49                 (SELECT lh_id, lh_prv_id, pty_id, pty_lh_sts
50                   FROM
51                     (SELECT lh_id, lh_prv_id
```

```
52                      FROM config_data.lower_house
53                    ) AS LOWER_HOUSE
54                  JOIN
55                    (SELECT lh_id, pty_id, pty_lh_sts::NUMERIC
56                      FROM config_data.lh_seat_results
57                      WHERE pty_lh_sts >= 1
58                    ) AS LH_SEAT_RESULTS
59                  USING(lh_id)
60                ) AS CUR_LH_STS
61              USING(lh_id)
62            ) AS CUR_LH_STS_SHR
63          WHERE CUR_LH_STS_SHR.lh_prv_id = PREV_LH_STS_SHR.lh_id
64          AND CUR_LH_STS_SHR.pty_id = PREV_LH_STS_SHR.pty_id
65        ) AS PTY_STS_SHR_DIFF
66      WHERE lh_id
67      NOT IN
68        (SELECT DISTINCT lh_id
69          FROM
70            (SELECT lh_id, pty_id, pty_lh_sts
71              FROM config_data.lh_seat_results
72            ) AS LH_SEATS
73          JOIN
74            (SELECT pty_id, pty_abr
75              FROM config_data.party
76            ) AS PARTIES
77          USING(pty_id)
78        WHERE pty_lh_sts IS NULL
79        AND pty_abr NOT LIKE '%Other'
80        )
81      GROUP BY lh_id
82    ) AS VALID_LH_VOLB_STS
83  USING(lh_id)
84 ORDER BY lh_id;
```

Stable parties are identified computationable by calculating the cross-product between rows in the subqueries CUR_LH_STS_SHR and PREV_LH_STS_SHR, and reporting only those for which a party identifier is enlisted in both the previous and the current election (cf. corresponding WHERE-clause).

Note: The concept of stable party makes no sense for first recorded lower houses, and hence B volaities are not computed. The measure is highly sensitive to missing data, as no aggregate value is computed for lower house elections in which at least one party except the group 'Others withour seat' has NULL records for total seat results (cf. consistency check `cc_missing_lh_pty_sts_records` [3.7.7]). A lack of reliable lower-level data thus causes severe lack of aggregate data.

Generally, consistency check `cc_lh_volb_sts` [3.7.12]) provides for a comparison of the computed and the recorded figures, though the recorded have been computed manually as well.

3.3.26 Type A Volatility in Lower House Vote Shares

View `view_lhelc_vola_vts` is based on tables Lower House Election, Lower House Vote Results and Party, and provides data at the level of lower house elections.

Generally, type A volatility measures volatility from party entry and exit to the political system and is quantified by the change that occurs in the distribution of shares between parties due

to parties newly entering respectively retiering from the electoral arena (Powell and Tucker, 2013), majorly the domestic party system or the lower house.

Type A volatitlity in votes in a given lower house election is defined as volatility in the distribution of votes arising from new entering and retiering parties, given by the formular:

$$Vote\,A\,Volatility\,(k) = \frac{|\sum_{n=1}^{New} v_{n,k} + \sum_{o=1}^{Old} v_{o,k}|}{2} \tag{3.9}$$

, where o refers to retiering parties that contested only the election $k-1$ and n to new-entering parties that contested only election k, and generally v is party's vote share in the lower house election (i.e., the number of votes gained by party, divided by the total number of votes distributed between all parties J that railed in the respective election k).

View `view_lhelc_vola_vts` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_lhelc_vola_vts
AS
SELECT lhelc_id,
  (ABS(COALESCE(old_pty_sum, 0)
  + COALESCE(new_pty_sum, 0))/2)::DOUBLE PRECISION
    AS lhelc_vola_vts_computed
  FROM
    (SELECT lhelc_id, old_pty_sum
      FROM
        (SELECT lhelc_id
          FROM config_data.lh_election
        ) AS LH_CONFIGS
      LEFT OUTER JOIN
        (SELECT lhelc_nxt_id AS lhelc_id,
          SUM(new_pty_lh_vts_shr) AS old_pty_sum
          FROM
            (SELECT lhelc_id, lhelc_nxt_id
              FROM config_data.lh_election
            ) AS LH_ELECTION
          JOIN
            (SELECT lhelc_id, pty_id,
              100*(VOTES.pty_lh_vts_computed
              /VOTES_TOTAL.lhelc_vts_ttl_computed)
                AS new_pty_lh_vts_shr
              FROM
                (SELECT lhelc_id,
                  SUM(COALESCE(pty_lh_vts_pl, 0)
                  + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
                    AS lhelc_vts_ttl_computed
                  FROM config_data.lh_vote_results
                GROUP BY lhelc_id
                ) AS VOTES_TOTAL
              JOIN
                (SELECT lhelc_id, pty_id,
                  (COALESCE(pty_lh_vts_pl, 0)
                  + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
                    AS pty_lh_vts_computed
                  FROM config_data.lh_vote_results
                  WHERE pty_id
                  NOT IN
                    (SELECT pty_id
                      FROM config_data.party
                      WHERE pty_abr LIKE 'Other'
                    )
                  EXCEPT
                    SELECT lhelc_id, CUR_LHELC.pty_id AS pty_id, pty_lh_vts_computed
```

```
                        FROM
                          (SELECT lhelc_id, pty_id,
                            (COALESCE(pty_lh_vts_pl, 0)
                            + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
                              AS pty_lh_vts_computed
                            FROM config_data.lh_vote_results
                          ) AS PREV_LHELC
                        ,
                          (SELECT lhelc_prv_id, pty_id
                            FROM
                              (SELECT lhelc_id, lhelc_prv_id
                                FROM config_data.lh_election
                              ) AS LH_ELECTION
                            JOIN
                              (SELECT lhelc_id, pty_id
                                FROM config_data.lh_vote_results
                              ) AS LH_VOTE_RESULTS
                            USING(lhelc_id)
                          ) AS CUR_LHELC
                      WHERE CUR_LHELC.lhelc_prv_id = PREV_LHELC.lhelc_id
                      AND CUR_LHELC.pty_id = PREV_LHELC.pty_id
                  ) AS VOTES
                USING(lhelc_id)
              ) AS OLD_PARTY_SHR_SUM
            USING(lhelc_id)
            GROUP BY lhelc_nxt_id
          ) AS OLD_PARTIES
        USING(lhelc_id)
      ) AS LH_RETIRING_PARTIES
    LEFT OUTER JOIN
      (SELECT lhelc_id, SUM(new_pty_lh_vts_shr) AS new_pty_sum
        FROM
          (SELECT lhelc_id, pty_id,
            100*(VOTES.pty_lh_vts_computed/VOTES_TOTAL.lhelc_vts_ttl_computed) AS new_pty_lh_vts_shr
            FROM
              (SELECT lhelc_id,
                SUM(COALESCE(pty_lh_vts_pl, 0)
                + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
                  AS lhelc_vts_ttl_computed
                FROM config_data.lh_vote_results
              GROUP BY lhelc_id
              ) AS VOTES_TOTAL
            JOIN
              (SELECT lhelc_id, pty_id,
                (COALESCE(pty_lh_vts_pl, 0)
                + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
                  AS pty_lh_vts_computed
                FROM config_data.lh_vote_results
              WHERE pty_id
              NOT IN
                (SELECT pty_id
                  FROM config_data.party
                  WHERE pty_abr LIKE 'Other'
                )
              AND lhelc_id
              NOT IN
                (SELECT MAX(lhelc_id) AS lhelc_id
                  FROM config_data.lh_election
                  GROUP BY ctr_id
                )
              EXCEPT
                SELECT lhelc_id, CUR_LHELC.pty_id AS pty_id, pty_lh_vts_computed
                  FROM
                    (SELECT lhelc_id, pty_id,
                      (COALESCE(pty_lh_vts_pl, 0)
                      + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
                        AS pty_lh_vts_computed
                      FROM config_data.lh_vote_results
```

```
115                     ) AS PREV_LHELC
116                   ,
117                     (SELECT lhelc_nxt_id, pty_id
118                       FROM
119                         (SELECT lhelc_id, lhelc_nxt_id
120                           FROM config_data.lh_election
121                         ) AS LH_ELECTION
122                       JOIN
123                         (SELECT lhelc_id, pty_id
124                           FROM config_data.lh_vote_results
125                         ) AS LH_VOTE_RESULTS
126                       USING(lhelc_id)
127                     ) AS CUR_LHELC
128                 WHERE CUR_LHELC.lhelc_nxt_id = PREV_LHELC.lhelc_id
129                 AND CUR_LHELC.pty_id = PREV_LHELC.pty_id
130               ) AS VOTES
131             USING(lhelc_id)
132           ) AS NEW_PARTY_VOTE_SHARES
133         GROUP BY lhelc_id
134       ) AS NEW_PARTIES
135     USING(lhelc_id)
136   ORDER BY lhelc_id;
```

Because the SQL-syntax of `view_lhelc_vola_vts` is rather complex, some brief comments are instructive:

- The enumerator of Equ 3.9 consists of two summands; each is computed seperately (the parts embraced by paranthesis and labeled NEW_PARTIES and LH_RETIERING_PARTIES, respectively) and added only in the very first lines of the query.
- With respect to the subqueries, NEW_PARTIES aggregates the vote shares of parties that contested in the present lower house election but not in the previous one, and LH_RETIERING_PARTIES aggregates the vote shares of parties that contested in the previous election but not in the current one.
- Exluding 'stable' parties (i.e., parties that entered the lower house in the present as well as the previous election) within the subqueries is achieved by the EXCEPT-clauses, which pair parties recorded for the present and the previous lower house by party identifiers. If a party contested only in the present election, or only in the previous elections, then it does not occur in the query that follows the EXCEPT-clauses. In consequence, only votes gained by new entering and retiering parties enter the aggregation.
- The category 'Others without seat' (`pty_id` is ##999) are excluded from the computation of individual parties' vote shares, because volatility in the lower house is of interest (not volatility in the party system more generally).
- Generally, joining parties' vote results with different combinations of the identifiers of the previous, the current, and the next lower house election enables to easily identify new entering and retiering parties.

Note: Figures for first an last recorded elections are unreliable, because it is impossible to determine which parties are 'newcomers' in first and which parties will retier in last election, respectively.

3.3.27 Type B Volatility in Lower House Vote Shares

View `view_lhelc_volb_sts` is based on tables Lower House, Lower House Vote Results and Party, and provides data at the level of lower house elections.

Type B volatility quantifies the change that occurs in the distribution of vote shares within parties in subseqent elections, comparing the results in the current election to that of the previous. Accordingly, type B volatility considers only so-called stable parties and measures the volatility in the distribution of votes arising from gains and losses of these stable parties

The formular to compute `lhelc_volb_vts` is

$$Seat\,B\,Volatility\,(k) = \frac{\left|\sum_{j=1}^{Stable} v_{j,(k-1)} - v_{j,k}\right|}{2} \tag{3.10}$$

, where v are vote shares that party j gained in the current lower house k or in the previous lower house $k-1$.

View `view_lhelc_volb_vts` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_lhelc_volb_vts
2  AS
3  SELECT lhelc_id, lhelc_volb_vts_computed
4    FROM
5      (SELECT lhelc_id
6        FROM config_data.lh_election
7      ) AS ALL_LH_ELECTIONS
8    LEFT OUTER JOIN
9      (SELECT lhelc_id,
10       (SUM(pty_lh_vts_shr_diff)/2)::DOUBLE PRECISION
11         AS lhelc_volb_vts_computed
12       FROM
13         (SELECT CUR_LHELC_VTS_SHR.lhelc_id AS lhelc_id,
14           ABS(PREV_LHELC_VTS_SHR.pty_prv_lh_vts_shr
15           - CUR_LHELC_VTS_SHR.pty_cur_lh_vts_shr)
16             AS pty_lh_vts_shr_diff
17           FROM
18             (SELECT lhelc_id, pty_id,
19             100*(VOTES.pty_lh_vts_computed
20             /VOTES_TOTAL.lhelc_vts_ttl_computed)
21               AS pty_prv_lh_vts_shr
22               FROM
23                 (SELECT lhelc_id,
24                   SUM(COALESCE(pty_lh_vts_pl, 0)
25                   + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
26                     AS lhelc_vts_ttl_computed
27                   FROM config_data.lh_vote_results
28                   GROUP BY lhelc_id
29                 ) AS VOTES_TOTAL
30               JOIN
31                 (SELECT lhelc_id, pty_id,
32                   (COALESCE(pty_lh_vts_pl, 0)
33                   + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
34                     AS pty_lh_vts_computed
35                   FROM config_data.lh_vote_results
36                   WHERE pty_id
37                   NOT IN
38                     (SELECT pty_id
39                       FROM config_data.party
40                       WHERE pty_abr LIKE 'Other'
41                     )
```

```
42                  ) AS VOTES
43                USING(lhelc_id)
44              ) AS PREV_LHELC_VTS_SHR
45            ,
46              (SELECT lhelc_id, lhelc_prv_id, pty_id,
47                100*(pty_lh_vts_computed
48                /lhelc_vts_ttl_computed)
49                  AS pty_cur_lh_vts_shr
50                FROM
51                  (SELECT lhelc_id,
52                    SUM(COALESCE(pty_lh_vts_pl, 0)
53                    + COALESCE(pty_lh_vts_pr, 0))::NUMERIC
54                      AS lhelc_vts_ttl_computed
55                    FROM config_data.lh_vote_results
56                    GROUP BY lhelc_id
57                  ) AS VOTES_TOTAL
58                JOIN
59                  (SELECT lhelc_id, lhelc_prv_id, pty_id, pty_lh_vts_computed
60                    FROM
61                      (SELECT lhelc_id, lhelc_prv_id
62                        FROM config_data.lh_election
63                      ) AS LH_ELECTION
64                    JOIN
65                      (SELECT lhelc_id, pty_id,
66                        (COALESCE(pty_lh_vts_pl, 0)
67                        + COALESCE(pty_lh_vts_pr, 0))::numeric
68                          AS pty_lh_vts_computed
69                        FROM config_data.lh_vote_results
70                        WHERE pty_id
71                        NOT IN
72                          (SELECT pty_id
73                            FROM config_data.party
74                            WHERE pty_abr LIKE 'Other'
75                          )
76                      ) AS LH_VOTE_RESULTS
77                    USING(lhelc_id)
78                  ) AS CUR_LHELC_VTS
79                USING(lhelc_id)
80              ) AS CUR_LHELC_VTS_SHR
81            WHERE CUR_LHELC_VTS_SHR.lhelc_prv_id = PREV_LHELC_VTS_SHR.lhelc_id
82            AND CUR_LHELC_VTS_SHR.pty_id = PREV_LHELC_VTS_SHR.pty_id
83          ) AS PTY_VTS_SHR_DIFF
84        WHERE lhelc_id
85        NOT IN
86          (SELECT DISTINCT lhelc_id
87            FROM
88              (SELECT lhelc_id, pty_id,
89                (COALESCE(pty_lh_vts_pr,0)
90                + COALESCE(pty_lh_vts_pl,0))::NUMERIC
91                  AS pty_lh_vts_computed
92                FROM config_data.lh_vote_results
93              ) AS LH_VOTES
94            JOIN
95              (SELECT pty_id, pty_abr
96                FROM config_data.party
97              ) AS PARTIES
98            USING(pty_id)
99            WHERE pty_lh_vts_computed = 0
100           AND pty_abr NOT LIKE 'Other'
101         )
102     GROUP BY lhelc_id
103     ) AS VALID_LHELC_VOLB_VTS
104   USING(lhelc_id)
105 ORDER BY lhelc_id;
```

Stable parties are identified computationable by calculating the cross-product between rows in the subqueries CUR_LHELC_VTS_SHR and PREV_LHELC_VTS_SHR, and reporting only those

for which a party identifier is enlisted in both the previous and the current election (cf. corresponding WHERE-clause).

Note: The concept of stable party makes no sense for first recorded lower house elections, and hence B volaities are not computed. The measure is highly sensitive to missing data, as no aggregate value is computed for lower house elections in which at least one party except the group 'Others withour seat' has NULL records for total vote results (cf. consistency check `cc_missing_lhelc_pty_sts_records` [**??**]). A lack of reliable lower-level data thus causes severe lack of aggregate data.

Generally, consistency check `cc_lhelc_volb_vts` [3.7.16]) provides for a comparison of the computed and the recorded figures, though the recorded have been computed manually as well.

3.3.28 Lower House Election Registered Voters

View `view_lhelc_reg_vts_computed` is based on table Lower House Election and provides data at the level of lower house elections.

It computes the total number of registered voters as sum of the recorded figures for registered proportional and plurality voters in a given lower house election. (The total number of registered voters is essential to compute vote turnout.)

View `view_lhelc_reg_vts_computed` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_lhelc_reg_vts_computed
2  AS
3  SELECT lhelc_id, ctr_id, lhelc_date, lhelc_reg_vts,
4  (COALESCE(lhelc_reg_vts_pr,0)+COALESCE(lhelc_reg_vts_pl,0)) AS lhelc_reg_vts_computed
5  FROM config_data.lh_election
6  ORDER BY ctr_id, lhelc_date;
```

3.3.29 Party's Total Seats in the Lower House

View `view_pty_lh_sts_computed` is based on tables Lower House Seat Results and provides data at the level of parties in lower house elections.

It computes the total number of seats a party holds in a given lower house as sum of seats gained through proportional and plurality vote.

View `view_pty_lh_sts_computed` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.view_pty_lh_sts_computed
2  AS
3  SELECT lhsres_id, lh_id, pty_id, pty_lh_sts,
4    (COALESCE(pty_lh_sts_pr,0)
5    + COALESCE(pty_lh_sts_pl,0))
6      AS pty_lh_sts_computed
7    FROM config_data.lh_seat_results
8  ORDER BY lh_id, pty_id;
```

3.3.30 Party's Seat Share in the Lower House

View `view_pty_lh_sts_shr` is based on table Lower House Seat Results and provides data at the level of parties in lower house elections.

It computes the seat share of a party in a given lower house election, and is used to calculate the seat share of cabinet parties in the lower house (cf. `view_cab_lh_sts_shr`

View `view_pty_lh_sts_shr` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_pty_lh_sts_shr
AS
SELECT lh_id, pty_id, pty_lh_sts, lh_sts_ttl_computed,
  (pty_lh_sts::NUMERIC / lh_sts_ttl_computed) AS pty_lhelc_sts_shr
    FROM
      (SELECT lh_id,
        SUM(pty_lh_sts::NUMERIC) AS lh_sts_ttl_computed
        FROM config_data.lh_seat_results
        GROUP BY lh_id
      ) SEATS_TOTAL
  JOIN
    (SELECT lh_id, pty_id, pty_lh_sts
      FROM config_data.lh_seat_results
      WHERE pty_lh_sts <> 0
    ) SEATS
  USING (lh_id)
ORDER BY lh_id, pty_id;
```

Note: The computed figure, not the reocorded total seats in the lower house are used for computation.

3.3.31 Party's Seat Share in the Upper House

View `view_pty_uh_sts_shr` is based on table Upper House Seat Results and provides data at the level of parties in upper house elections.

It computes the seat share of a party in a given upper house , and is used to calculate the seat share of cabinet parties in the upper house (cf. `view_cab_uh_sts_shr`).

View `view_pty_uh_sts_shr` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.view_pty_uh_sts_shr
AS
SELECT uh_id, pty_id, pty_uh_sts, uh_sts_ttl_computed,
  (pty_uh_sts::NUMERIC / uh_sts_ttl_computed) AS pty_uh_sts_shr
    FROM
    (SELECT uh_id, SUM(pty_uh_sts::NUMERIC) AS uh_sts_ttl_computed
      FROM config_data.uh_seat_results
      GROUP BY uh_id
    ) SEATS_TOTAL
  JOIN
    (SELECT uh_id, pty_id, pty_uh_sts
      FROM config_data.uh_seat_results
      WHERE pty_uh_sts <> 0
    ) SEATS
  USING (uh_id)
ORDER BY uh_id, pty_id;
```

Note: The computed figure, not the reocorded total seats in the upper house are used for computation.

3.4 Materialized view Configuration Events

Materialized view `mv_configuration_events` is an exact copy of the Configuration Events view (see subsection 3.3.5). Creating a materialization of the Configuration Events view is necessary to fill in corresponding institution identifiers and to compute configuration end dates, as described in subsections 3.6.4 and 3.6.3.

Generally, in database managment a view is a virtual table representing the result of a defined query on the database.While, a view complies the defined data whenever it is queried (and hence is always up-to-date), a materialized view caches the result of the defined query as a concrete table that may be updated from the original base tables from time to time. Due to materialization, this comes at the cost of being being potentially out-of-date.

To ensure that the Configuration Events materialized view is up-to-date, there exists a trigger structure that is described in subsection 3.6.5 in detail.

Materialized view `mv_configuration_events` is created by calling function `create_matview(mv_name, v_name)`, where `mv_name` is `'mv_configuration_events'` and `v_name` is `'config_data.view_configu`

Function `create_matview(mv_name, v_name)` is programmed as follows:[9]

```
1  CREATE OR REPLACE FUNCTION config_data.create_matview(NAME, NAME)
2  RETURNS VOID
3  SECURITY DEFINER
4  LANGUAGE plpgsql AS '
5  DECLARE
6      matview_name ALIAS FOR $1;
7      view_name ALIAS FOR $2;
8      entry config_data.matviews%ROWTYPE;
9  BEGIN
10     SELECT * INTO entry FROM config_data.matviews WHERE matviews.mv_name = matview_name;
11
12     IF FOUND THEN
13         RAISE EXCEPTION ''Materialized view ''''%'''' already exists.'',
14           matview_name;
15     END IF;
16
17     EXECUTE ''REVOKE ALL ON '' || view_name || '' FROM PUBLIC'';
18     EXECUTE ''GRANT SELECT ON '' || view_name || '' TO PUBLIC'';
19     EXECUTE ''CREATE TABLE '' || matview_name || '' AS SELECT * FROM '' || view_name;
20     EXECUTE ''REVOKE ALL ON '' || matview_name || '' FROM PUBLIC'';
21     EXECUTE ''GRANT SELECT ON '' || matview_name || '' TO PUBLIC'';
22
23     INSERT INTO config_data.matviews (mv_name, v_name, last_refresh)
24       VALUES (matview_name, view_name, CURRENT_TIMESTAMP);
25
26     RETURN;
27 END
28 ';
```

Function `refresh_matview(mv_name)` generally executes a refresh of materialized views.[10]

```
1  CREATE OR REPLACE FUNCTION config_data.refresh_matview(NAME) RETURNS VOID
2  SECURITY DEFINER
3  LANGUAGE plpgsql AS '
4  DECLARE
5      matview_name ALIAS FOR $1;
```

[9] Source is Listing 2 at http://www.varlena.com/GeneralBits/Tidbits/matviews.html.
[10] Source is Listing 3 at http://www.varlena.com/GeneralBits/Tidbits/matviews.html.

```
6       entry config_data.matviews%ROWTYPE;
7   BEGIN
8
9       SELECT mv_name, v_name INTO entry FROM config_data.matviews WHERE matviews.mv_name = matview_na
10
11      IF NOT FOUND THEN
12          RAISE EXCEPTION ''Materialized view % does not exist.'', matview_name;
13      END IF;
14
15      EXECUTE ''ALTER TABLE config_data.'' || matview_name || '' DISABLE TRIGGER USER'';
16      EXECUTE ''DELETE FROM config_data.'' || matview_name;
17      EXECUTE ''INSERT INTO config_data.'' || matview_name
18          || '' SELECT * FROM '' || entry.v_name;
19      EXECUTE ''ALTER TABLE config_data.'' || matview_name || '' ENABLE TRIGGER USER'';
20
21      UPDATE config_data.matviews
22          SET last_refresh=CURRENT_TIMESTAMP
23          WHERE matviews.mv_name = matview_name;
24
25      EXECUTE ''SELECT config_data.update_mv_config_events()'';
26
27      RETURN;
28  END';
```

Note: Alternatively, postgreSQL provides for an implemented command CREATE MATERIALIZED VIEW[11] and a corresponding REFRESH-command.[12] It is worth considering to re-implement materialized view Configuration Events with the built-in commands in a beta version, though the solution described above appeares to yield the same result.

[11] See http://www.postgresql.org/docs/9.3/static/sql-creatematerializedview.html
[12] See http://www.postgresql.org/docs/9.3/static/sql-refreshmaterializedview.html

3.5 Views in the public scheme

The views contained in the `public` scheme of the PCDB compile information on the time series contained in the PCDB at different levels of analysis and aggregation, and in different temporal formats. The Public Views are thought to provide for a user-friendly usage of the PCDB data.

3.5.1 Configuration

Public view `configuration` sequences changes in countries' political-institutional configurations by institutional start dates. The basic logic of poltiical configurations that applies in the PCDB is explained in subsection 3.3.5.

Accordingly, every new row corresponds to a historically unique political configuration among a country's government, lower house, upper house and the position of the Head of State, and a configuration is uniquely identified by combinations of `ctr_id`, `cab_id`, `lh_id`, `uh_id` (if applies), and `prs_id` (if applies).

Though the information provided with public view Configuration is based on the view Configuration Events and its materialization, Configuration adds the relevant information on veto constellations that correspond to temporal sequences. That is, public view Configuration generally draws on materialized view Configuration Events (described in subsection 3.4) and a variety of views that determine whether a variety of political isntitutions constitute open veto points vis-à-vis the government.

Configuration start dates, end dates, duration A configuration's start date corresponds to the start date of the institution the most recent change occured. End dates, in turn, equal the day before the start date of the next configuration in the given country. Obviously, variable `config_duration` simply counts the days from the first to the last day of a configuration. End dates are implemented by trigger on materialized view Configuration Events

Cabinet's seat share in the lower and the upper house Variable `cab_lh_sts_shr` quantifies the share of seats of the party/parties in the cabinet on the total seats in the corresponding lower house. Variable `cab_uh_sts_shr` quantifies the share of seats of the party/parties in the cabinet on the total seats in the corresponding upper house. Information is joined-in from the respectives views in the `config_data` scheme (see subsection 3.3.1 and 3.3.2).

Veto points Whether an existing institution constitutes a potential veto point vis-à-vis the government is determined by legal (i.e., constitutional) entitlement of veto power. Veto power is either non-existent, conditional, or unconditional. Information on a country's institutions veto powers is recorded in the Veto Points table in the `config_data` scheme, specifically variable `vto_pwr`.

Whether a potential veto institution constitute an *open veto point* vis-à-vis the government is only contingent if its veto power is conditional. Regularly, constitutional law specifies a

threshold that determines how large a counter-governmental faction needs to be to blockade government's legisaltive initiatives. The size of non-government factions in combination with the legal veto threshold thus determine whether an institution constitutes an open veto point vis-à-vis the government.

Technically, public view Configuration performs a join of materialized view Configuration Events and the respective Veto views (see subsection 3.3.10 to 3.3.16), using the respective institution identifiers and configuration start dates.

The code to compile public view `configuration` reads as follows:

```
CREATE OR REPLACE VIEW public.configuration
AS
SELECT ctr_id, sdate, edate, cab_id, lh_id, lhelc_id, uh_id, prselc_id,
  cab_sts_ttl::NUMERIC,
  cab_lh_sts_shr::NUMERIC(7,5), cab_uh_sts_shr::NUMERIC(7,5),
  vto_lh, vto_uh, vto_prs, vto_pts, vto_jud, vto_elct, vto_terr,
  (COALESCE(vto_lh,0)+COALESCE(vto_uh,0)+COALESCE(vto_prs,0)+
    COALESCE(vto_jud,0)+COALESCE(vto_elct,0)+COALESCE(vto_terr,0)
  )::NUMERIC AS vto_sum,
  year, (edate-sdate)::NUMERIC AS config_duration
FROM
  (SELECT cab_id, cab_sts_ttl_computed AS cab_sts_ttl
    FROM config_data.view_cab_sts_ttl
  ) AS CAB_STS_TTL
FULL OUTER JOIN
  (SELECT * FROM
    (SELECT ctr_id, sdate, cab_uh_sts_shr
      FROM config_data.view_cab_uh_sts_shr
    ) AS CAB_UH_STS_SHR
  FULL OUTER JOIN
    (SELECT * FROM
      (SELECT ctr_id, sdate, cab_lh_sts_shr
        FROM config_data.view_cab_lh_sts_shr
      ) AS CAB_LH_STS_SHR
    FULL OUTER JOIN
      (SELECT * FROM
        (SELECT ctr_id, sdate, vto_terr
          FROM config_data.view_configuration_vto_terr
        ) AS VTO_TERR
      FULL OUTER JOIN
        (SELECT *
        FROM
          (SELECT ctr_id, sdate, vto_elct
            FROM config_data.view_configuration_vto_elct
          ) AS VTO_ELCT
        FULL OUTER JOIN
          (SELECT *
          FROM
            (SELECT ctr_id, sdate, vto_jud
              FROM config_data.view_configuration_vto_jud
            ) AS VTO_JUD
          FULL OUTER JOIN
            (SELECT *
            FROM
              (SELECT ctr_id, sdate, vto_pts
                FROM config_data.view_configuration_vto_pts
              ) AS VTO_PTS
            FULL OUTER JOIN
              (SELECT *
              FROM
                (SELECT ctr_id, sdate, vto_prs
                  FROM config_data.view_configuration_vto_prs
                ) AS VTO_PRS
              FULL OUTER JOIN
```

```
55                  (SELECT *
56                  FROM
57                    (SELECT ctr_id, sdate, vto_uh
58                      FROM config_data.view_configuration_vto_uh
59                    ) AS VTO_UH
60                  FULL OUTER JOIN
61                    (SELECT *
62                    FROM
63                      (SELECT ctr_id, sdate, vto_lh
64                        FROM config_data.view_configuration_vto_lh
65                      ) AS VTO_LH
66                    FULL OUTER JOIN
67                      (SELECT ctr_id, sdate, edate, cab_id, lh_id, lhelc_id, uh_id, prselc_id, year
68                        FROM config_data.mv_configuration_events
69                      ) AS CONFIG_EVENTS
70                    USING(ctr_id, sdate)
71                    ) AS CONFIG_LH
72                  USING(ctr_id, sdate)
73                  ) AS CONFIG_LH_UH
74                USING(ctr_id, sdate)
75                ) AS CONFIG_LH_UH_PRS
76              USING(ctr_id, sdate)
77              ) AS CONFIG_LH_UH_PRS_PTS
78            USING(ctr_id, sdate)
79            ) AS CONFIG_LH_UH_PRS_PTS_JUD
80          USING(ctr_id, sdate)
81          ) AS CONFIG_LH_UH_PRS_PTS_JUD_ELCT
82        USING(ctr_id, sdate)
83        ) AS CONFIG_LH_UH_PRS_PTS_JUD_ELCT_CLH_SSHR
84      USING(ctr_id, sdate)
85      ) AS CONFIG_LH_UH_PRS_PTS_JUD_ELCT_CLH_SSHR_CUH_SSHR
86    USING(ctr_id, sdate)
87    ) AS CONFIG_LH_UH_PRS_PTS_JUD_ELCT_CLH_SSHR_CUH_SSHR_CAB_STTL
88  USING(cab_id)
89  ORDER BY ctr_id, sdate;
```

Note: Rows are reported for all temporally corresponding combinations of institutional-political configurations. Thus, no institution corresponds to the very first institutional configuration that is recorded in the PCDB, resulting in rows with many non-trivial missings in countries' first configurations. Refer to the note on subsection 3.3.5 for an example of this problematique

3.5.2 Configuration Country-Years

Public view `configuration_ctr_yr` provides information on political configurations in a country-year format. Thus it essentially draws on the configuration Events materilized view (3.4) and the basic logic of political configurations, descirbed in subsection 3.3.5, applies.

The configurations that are reported for country-years are *no* aggregates (e.g., averaging across all configurations in a given country-year, as it is often done when summarizing economic data), but public view Configuration Country-Years reports *representative configurations*, having the highest temporal weight in a given country-year.

Choosing representative configurations A configuration's temporal weight in a country-year is computed by dividing its duration in the given year[13] by the total recorded days of

[13] Not to be confused with variable `config_duration`, which reports a configuration's total duration from the day it started to its end.

that year (365 days, except from leap years, and years of a country's first and last recorded configurations). The configurations with the highest weight in a given country-year is selected as representative for this year.[14]

Example 2: Duration and temporal weight of configurations in Australia, 1946 to 1949.

Start date	End date	Year	Duration in year	Recorded days	Weight
1946-09-28	1946-10-31	1946	34	95	0.3579
1946-11-01	1947-06-30	1946	61	95	0.6421
1946-11-01	1947-06-30	1947	181	365	0.4959
1947-07-01	1949-12-09	1947	184	365	0.5041
1947-07-01	1949-12-09	1948	366	366	1.0000
1947-07-01	1949-12-09	1949	343	365	0.9397
1949-12-10	1949-12-18	1949	9	365	0.0247
1949-12-19	1950-06-30	1949	13	365	0.0356

Example 2 illustrates the procedure for choosing representative configurations of country-years. The first line lists the very first recorded Australian configuration, starting on September 28, 1946 and durating total 34 days. The second recorded configuraton started on the first November of the same year but prevailed until the next year, ending on June 30, 1947. Thus, the second configuration durated 61 days in 1946 and 181 days in 1947, having clearly the highest temporal weight in 1946.

The third configuration durated total 184 days in 1947 and lasted until December 9, 1949. Accordingly, it has slightly the highest temporal weight in 1947 and is therefore chosen as representative configuration for year 1947.[15]

In 1948 only one configuration is recorded, This is because the fourth configuration, starting on first July, 1947, lasted until 1949 and is obviously representative for the whole year of 1948.

The third configuration that started in 1947 and outlasted 1948 durated total 343 days in 1949. Apparently, it was temporally extremely dominant also in the year of its end, as the other to configurations recorded with a start date in 1949 only amounted to weights equal to 0.0247 and 0.0356, respectively.

The code to compile public view `configuration_ctr_yr` reads as follows:

```
1  CREATE OR REPLACE VIEW public.configuration_ctr_yr
2  AS
3  SELECT CONFIG.ctr_id, DURATION_W.year, CONFIG.sdate, CONFIG.edate,
4    cab_id, lh_id, uh_id, prselc_id,
5    cab_sts_ttl, cab_lh_sts_shr, cab_uh_sts_shr,
6    vto_lh, vto_uh, vto_prs, vto_pts, vto_jud, vto_elct, vto_terr, vto_sum,
```

[14] There occure no configurations between 1945 and 2014 where the weight of two or more configurations in a year equals each other.

[15] Obviously, choosing representative configurations based on such a slight difference in relative duration is not unproblematic.

```
7     config_duration_in_year,
8     COALESCE(config_weight_in_year,1)::NUMERIC(7,5) AS config_weight_in_year,
9     config_duration
10  FROM
11    (SELECT *
12      FROM
13        (SELECT DISTINCT ctr_id, sdate, in_year AS year, config_duration_in_year
14          FROM config_data.view_configuration_duration_in_year
15          ORDER BY ctr_id, in_year
16        ) AS CONFIGURATION_DURATION_IN_YEAR
17      LEFT OUTER JOIN
18        (SELECT ctr_id, sdate, config_weight_in_year, year
19          FROM config_data.view_config_weight_in_year
20        ) AS ALL_CONFIGS_WITH_WEIGHTS
21      USING(ctr_id, year, sdate)
22      ORDER BY ctr_id, year, sdate NULLS FIRST
23    ) AS DURATION_W
24  ,
25    (SELECT * FROM public.configuration) AS CONFIG
26
27  WHERE   DURATION_W.ctr_id = CONFIG.ctr_id
28  AND DURATION_W.sdate = CONFIG.sdate
29  AND (DURATION_W.year::numeric+0.1*COALESCE(config_weight_in_year,1))
30    IN
31    (SELECT (year::numeric+0.1*config_weight_in_year) AS ctr_yr_identifier
32      FROM
33        (SELECT ctr_id,
34          MAX(config_weight_in_year) AS config_weight_in_year,
35          MAX(year) AS year
36          FROM config_data.view_config_weight_in_year
37          GROUP BY ctr_id, year
38        ) AS CONFIGS_WITH_HIGHEST_WEIGHT_IN_YEAR
39    )
40  ORDER BY ctr_id, year, sdate;
```

Note: The WHERE-clause ensures that only the configurations that have the highest thempral weight within a country-year ar reported. Specifically, the IN-condition draws on a combination of year and temporal weights to uniquely identify configurations within country-years. Obviously, this procedure presupposes uniquness of temporal weights within country-years; a condtion that is met in the PCDB to date.[14]

3.6 Triggers

Triggers are functions executed on tables to insert, update, or delete data from specific columns or cells. Each function is 'triggered' by one or more specific events.

3.6.1 Identify previous institution-configurations within countries

A set of triggers (`trg_*_prv_id()`) is implemented on the base-tables Cabinet, Lower House, Upper House, and Presidential Election, and on table Lower House Election, respectively, to assign the identifiers of previous instituion-configurations into cells of column `*_prv_id`.

Specifically, functions `trg_*_prv_id()` selects the identifier of the previous configuration, as identified by the next lower date of all the configurations recorded for a country within a base-table. Schematically, it is programmed as follows:

```
CREATE OR REPLACE FUNCTION config_data.trg_*_prv_id()
RETURNS trigger AS $function$
  BEGIN
    NEW.*_prv_id :=
      (SELECT *_id FROM config_data.*table
      WHERE *_sdate < NEW.*_sdate
      AND ctr_id = NEW.ctr_id
      ORDER BY ctr_id, *_sdate DESC
      LIMIT 1);
  RETURN NEW;
  END;
$function$ LANGUAGE plpgsql;

CREATE TRIGGER trg_*_prv_id
  BEFORE INSERT OR UPDATE ON config_data.*table
  FOR EACH ROW
  EXECUTE PROCEDURE config_data.trg_*_prv_id();
```

Where generally `*` refers to either `cab`, `lh`, `lhelc`, `uh` or `prselc`, and `*table` to either `cabinet`, `lower_house`, `lh_election`, `upper_house` or `presidential_election`.

Trigger `trg_cab_nxt_id` is executed for each row before inserting or updating data to the base table.

Note: In the case of table Lower House Election `_sdate` is replaced by `_date`, as it refers to election date instead of institution-configuration start date.

A detailed description of the respective triggers and functions is provided in the appendix (5.1.1)

3.6.2 Identify next institution-configurations within countries

Another set of triggers (`trg_*_nxt_id()`) is implemented on the base-tables Cabinet, Lower House, and on table Lower House Election, respectively, to assign the identifiers of the next instituion-configurations into cells of column `*_prv_id`.

Specifically, functions `trg_*_nxt_id()` selects the identifier of the next configuration, as identified by the next higher date of all the configurations recorded for a country within a table. Schematically, it is programmed as follows:

```
CREATE OR REPLACE FUNCTION config_data.trg_*_nxt_id()
RETURNS trigger AS $function$
  BEGIN
    NEW.*_nxt_id :=
      (SELECT *_id FROM config_data.*table
      WHERE *_sdate > NEW.*_sdate
      AND ctr_id = NEW.ctr_id
      ORDER BY ctr_id, *_sdate ASC
      LIMIT 1);
  RETURN NEW;
  END;
$function$ LANGUAGE plpgsql;

CREATE TRIGGER trg_*_nxt_id
  BEFORE INSERT OR UPDATE ON config_data.*table
  FOR EACH ROW
  EXECUTE PROCEDURE config_data.trg_*_nxt_id();
```

Where generally `*` refers to either `cab`, `lh`, or `lhelc`, and `*table` to either `cabinet`, `lower_house`, or `lh_election`.

Trigger `trg_cab_nxt_id` is executed for each row before inserting or updating data to the base table.

Note: In the case of table Lower House Election `_sdate` is replaced by `_date`, as it refers to election date instead of institution-configuration start date.

A detailed description of the respective triggers and functions is provided in the appendix (5.1.2)

3.6.3 Identify end dates of political configurations

Trigger `trg_mv_config_ev_edate` is executed on materialized view Configuration Events and inserts data into cells of column `edate`. See the description of view Configuration Events (**??**) for an explanation of the concept and definition of political configurations in the PCDB.

Specifically, function `trg_mv_config_ev_edate()` selects the start date of the next recorded political configuration, as identified by the next bigger date of all recorded political configurations for a country, substracts one day from this date and assigns the resulting date as end date of the respective configuration

Function `trg_mv_config_ev_edate()` is programmed as follows:

```
CREATE OR REPLACE FUNCTION config_data.trg_mv_config_ev_edate()
RETURNS trigger AS $function$
  BEGIN
    NEW.edate :=
    (SELECT sdate-1 FROM config_data.mv_configuration_events
    WHERE sdate > NEW.sdate
    AND ctr_id = NEW.ctr_id
    ORDER BY ctr_id, sdate ASC
    LIMIT 1);
  RETURN NEW;
  END;
```

```
12 $function$ LANGUAGE plpgsql;
13
14 DROP TRIGGER IF EXISTS trg_it_mv_config_ev_edate
15 ON config_data.mv_configuration_events;
16 CREATE TRIGGER trg_it_mv_config_ev_edate
17   AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW
18   EXECUTE PROCEDURE config_data.trg_mv_config_ev_edate();
19
20 DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_edate
21 ON config_data.mv_configuration_events;
22 CREATE TRIGGER trg_dt_mv_config_ev_edate
23   AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW
24   EXECUTE PROCEDURE config_data.trg_mv_config_ev_edate();
25
26 DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_edate
27 ON config_data.mv_configuration_events;
28 CREATE TRIGGER trg_ut_mv_config_ev_edate
29   BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW
30   EXECUTE PROCEDURE config_data.trg_mv_config_ev_edate();
```

Trigger-function `trg_it_mv_config_ev_edate` is executed for each row of materialzied view Configuration Events after inserting new data, i.e., whenever a new configuration emerges; function `trg_dt_mv_config_ev_edate` is executed for each row of materialzied view Configuration Events after deleting data from it; and function `trg_ut_mv_config_ev_edate`, in turn, is executed for each row of materialzied view Configuration Events before its data is updated.

Note: The events insert, update or delete occur whenever data in the tables that underly view Configuration Events (and accordingly its materilization) is changed, that is, data is inserted to, updated in or deleted from tables Cabinet, Lower House, Upper House, or Presidential Elections.

The trigger structure that executes function `trg_mv_config_ev_edate()` is constituted on a chain of trigger functions, which *in toto* guarantee for the consistency and actuallity of the data that informs about countries' history of political configurations.

3.6.4 Selecting corresponding institution identifiers within political configurations

View Configuration Events (**??**) sequences changes in the political-institutional configurations of a country by date.

Each row corresponds to a historically unique political configuration of government, lower house, upper house and the position of the Head of State. Political configurations are also uniquely identified by combinations of `ctr_id` and `sdate`). The following excerpt illustrates what the structure of view Configuration Events looks like:[16]

Appraently, sequencing by start dates results in many empty cells. Yet, the second recorded president, who took office on December 23, 1995, was in charge throughout the subsequent five configurations. Thus, the presidential election identifier 25002 should also occur in this cells. Particularly, computation of veto point in a given politcal configuration requires to fill

[16] Poland has been chosen as an example because it is one of the few countries in the PCDB in which all political institutions of interest exist, as, besides lower and upper house, presidents are popularly elected since 1990.

Example 3a: Excerpt from view Configuration Events with empty cells for temporally corresponding institution-configuratins.

ctr_id	sdate	prselc_id	uh_id	lh_id	cab_id
25	1993-09-19			25002	
25	1993-10-15		25002		
25	1993-10-26				25005
25	1995-05-06				25006
25	1995-12-23	25002			
25	1996-02-07				25007
25	1997-09-21			25003	
25	1997-10-21		25003		
25	1997-11-11				25008
25	2000-06-29				25009
25	2000-12-23	25003			

the empty cells with the identifiers that refer to the cabinet, president, etc. that were in charge at a given point in time.

Because it is not possible to insert data into views, a materialized view that is identical with view Configuration Events is created: `mv_configuration_events`

To fill empty cells with temporally corresponding identifiers, a set of functions (`trg_mv_config_ev_prv_*_i` is created Schematically, they are defined as follows:

```
CREATE FUNCTION config_data.trg_mv_config_ev_prv_*_id()
RETURNS trigger AS $function$
  BEGIN
    IF
      OLD.*_id IS NOT NULL THEN NEW.*_id = OLD.*_id;
    ELSE
      NEW.*_id :=
      (SELECT *_id FROM config_data.mv_configuration_events
      WHERE sdate < NEW.sdate
      AND ctr_id = NEW.ctr_id
      ORDER BY ctr_id, sdate DESC
      LIMIT 1);
    END IF;
  RETURN NEW;
  END;
$function$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trg_it_mv_config_ev_prv_*_id
ON config_data.mv_configuration_events;
CREATE TRIGGER trg_it_mv_config_ev_prv_*_id
  AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW -- after insert
  EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_*_id();

DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_prv_*_id
ON config_data.mv_configuration_events;
CREATE TRIGGER trg_dt_mv_config_ev_prv_*_id
  AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW -- after delete
  EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_*_id();

DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_prv_*_id
ON config_data.mv_configuration_events;
CREATE TRIGGER trg_ut_mv_config_ev_prv_*_id
  BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW -- before update
```

```
34    EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_*_id();
```

The function inserts the identifier of the institution-configuration that is currently in charge into empty cells, by choosing that one which was in charge in the previous political configuration.

Functions `trg_mv_config_ev_prv_*_id()` are triggered by insert, update, or delete from materialized view Configuration Events; events that occur when data in the base-tables is changed (see subsection 3.6.5).

These procedures result in a structure that looks as follows:

Example 3b: Excerpt from materilaized view Configuration Events with cells of temporally corresponding institution-configuratins filled by triggers.

ctr_id	sdate	prselc_id	uh_id	lh_id	cab_id
25	1993-09-19	25001	25001	25002	25004
25	1993-10-15	25001	25002	25002	25004
25	1993-10-26	25001	25002	25002	25005
25	1995-05-06	25001	25002	25002	25006
25	1995-12-23	25002	25002	25002	25006
25	1996-02-07	25002	25002	25002	25007
25	1997-09-21	25002	25002	25003	25007
25	1997-10-21	25002	25003	25003	25007
25	1997-11-11	25002	25003	25003	25008
25	2000-06-29	25002	25003	25003	25009
25	2000-12-23	25003	25003	25003	25009

The empty cells have been filled and materialized view Configuration Events can be used to compute the respective veto-potential configurations, cabinet seat shares in the lower and upper houses, and so forth.

3.6.5 Integrity and consistency of materialized view Configuration Events

3.6.5.1 Defining tables and functions that underlie the trigger-structure

First, table Materialized Views[17] is defined as follows

```
1  CREATE TABLE config_data.matviews (
2    mv_name NAME NOT NULL PRIMARY KEY,
3    v_name NAME NOT NULL,
4    last_refresh TIMESTAMP WITH TIME ZONE);
```

It stores on which view a materialized view is based and the date time of its last refresh.

Second, the two following functions are defines:

[17] Source is Listing 1 at http://www.varlena.com/GeneralBits/Tidbits/matviews.html.

i) Function `mv_config_ev_refresh_row(#`$_{ctr}$`, #`$_{date}$`)`, which performs a refresh of rows in materialized view Configuration Events for a given combination of country identifier and start date:

```
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_refresh_row(SMALLINT, DATE)
RETURNS VOID
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
  DECLARE
  country ALIAS FOR $1;
  start_date ALIAS FOR $2;
  entry config_data.matviews%ROWTYPE;
    BEGIN
      ALTER TABLE config_data.mv_configuration_events DISABLE TRIGGER USER;

      DELETE FROM config_data.mv_configuration_events
        WHERE mv_configuration_events.ctr_id = country
        AND mv_configuration_events.sdate = start_date;

      INSERT INTO config_data.mv_configuration_events
      SELECT *
        FROM config_data.view_configuration_events
        WHERE view_configuration_events.ctr_id = country
        AND view_configuration_events.sdate = start_date;

      ALTER TABLE config_data.mv_configuration_events ENABLE TRIGGER USER;

      PERFORM config_data.update_mv_config_events();

      RETURN;
    END
';
```

The function performs the following procedures:

(1) disable all triggers implemented on materialized view Configuration Events;

(2) delete the row from materialized view Configuration Events that is identified by country identifier and start date;

(3) insert the respective configuration information (country identifier and start date) from view Configuration Events into materialized view Configuration Events;

(4) enable all triggers implemented on materialized view Configuration Events; and

(5) execute function `update_mv_config_events()`, which is defined as:

```
CREATE OR REPLACE FUNCTION config_data.update_mv_config_events()
RETURNS VOID
SECURITY DEFINER
LANGUAGE plpgsql AS '
BEGIN
  UPDATE config_data.mv_configuration_events
    SET cab_id = cab_id,
      lh_id = lh_id, lhelc_id = lhelc_id,
      uh_id = uh_id, prselc_id = prselc_id,
      edate = edate;
END';
```

and results in executing all functions that are implemented as triggers on materialized view Configuration Events (fill empty cells with identifiers of cabinets, lower house configurations, etc. in charge, and computing configuration end dates).

ii) Function `refresh_mv_config_events(#`$_{ctr}$`)`, defined as follows:

```
CREATE OR REPLACE FUNCTION config_data.refresh_mv_config_events(SMALLINT)
RETURNS VOID
SECURITY DEFINER
LANGUAGE plpgsql AS '
  BEGIN
    ALTER TABLE config_data.mv_configuration_events DISABLE TRIGGER USER;
    DELETE FROM config_data.mv_configuration_events WHERE ctr_id = $1;
    INSERT INTO config_data.mv_configuration_events
      SELECT * FROM config_data.view_configuration_events WHERE ctr_id = $1;
    ALTER TABLE config_data.mv_configuration_events ENABLE TRIGGER USER;

    UPDATE config_data.matviews
      SET last_refresh=(SELECT CURRENT_TIMESTAMP)
      WHERE matviews.mv_name LIKE ''config_data.mv_configuration_events'';

    EXECUTE ''SELECT config_data.update_mv_config_events()'';

    RETURN;
  END';
```

The function performs the following procedures:

(1) disable all triggers implemented on materialized view Configuration Events;

(2) delete all rows identified by country identifier #$_{ctr}$;

(3) insert (i.e., exact copy of) all rows from view Configuration Events that are identified by country identifier #$_{ctr}$;

(4) enable all triggers implemented on materialized view Configuration Events;

(5) update the date of the last refresh of materialized view Configuration events in table Materialized Views to current date and time (see page 68); and

(4) execute function `update_mv_config_events()`,

3.6.5.2 Implementing trigger-structure on base-tables

Function `mv_config_ev_refresh_row(#`$_{ctr}$`, #`$_{date}$`)`, in turn, is executed by three types of triggers that are each implemented on tables Cabinet, Lower House, Upper House, and Presidential Elections, respectively (the 'base-tables'). Because the definition is lengthy, here only the respective trigger-types are explained, while the full definition is provided in the appendix.

Insert function and trigger The first trigger-type that executes function `mv_config_ev_refresh_row(#`$_{ctr}$`, #`$_{date}$`)` is triggered by insert on the base-tables. Schematically, it is programmed as follows:

```
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_*table_it()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
  PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, *_sdate::DATE)
    FROM config_data.*table
    WHERE *table.*_id = NEW.*_id
    AND *table.*_sdate = NEW.*_sdate;
  RETURN NULL;
```

```
END';
DROP TRIGGER IF EXISTS mv_config_ev_insert ON config_data.*table;
CREATE TRIGGER mv_config_ev_insert
  AFTER INSERT ON config_data.*table
  FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_*table_it();
```

Where * refers to either cab, lh, uh or prselc, and *table to either cabinet, lower_house, upper_house or presidential_election.

Update function and trigger The second trigger-type that executes function mv_config_ev_refresh_r #$_{date}$) is triggered by update on the base-tables. Schematically, it is programmed as follows:

```
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_*table_ut()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
    PERFORM config_data.refresh_mv_config_events(ctr_id::SMALLINT)
      FROM config_data.*table
      WHERE *table.ctr_id = NEW.ctr_id
      AND *table.*_id = NEW.*_id
      AND *table.*_sdate = NEW.*_sdate;
  RETURN NULL;
END';
DROP TRIGGER IF EXISTS mv_config_ev_update ON config_data.*table;
CREATE TRIGGER mv_config_ev_update
  AFTER UPDATE ON config_data.*table
  FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_*table_ut();
```

Where * refers to either cab, lh, uh or prselc, and *table to either cabinet, lower_house, upper_house or presidential_election.

Delete function and trigger The third trigger-type that executes function mv_config_ev_refresh_row #$_{date}$) is triggered by delete from the base-tables. Schematically, it is programmed as follows:

```
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_*table_dt()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
  PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, *_sdate::DATE)
    FROM config_data.*table
    WHERE *table.*_id = OLD.*_id
    AND *table.*_sdate = OLD.*_sdate;
  RETURN NULL;
END';
DROP TRIGGER IF EXISTS mv_config_ev_delete ON config_data.*table;
CREATE TRIGGER mv_config_ev_delete
  AFTER DELETE ON config_data.*table
  FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_*table_dt();
```

3.6.5.3 Summarizing the trigger-structure

While view Configuration Events sequences changes in the politcal-instituional configurations of countries by date, and materialized view Configuration Events is only implemented to allow for perfomring permanent changes (fill empty cells, compute configuration end dates, etc.), the

trigger structure described above is thought to guarantee for the integrity and consistency of the data on politcal configurations.

In particular, when changes in the base-tables occur, triggering functions `mv_config_ev_refresh_row(#`$_{ctr}$ `#`$_{date}$`)` and `refresh_mv_config_events(#`$_{ctr}$`)`, respectively, results in corresponding changes in materialized view Configuration Events.

It is instructive to give three short examples to illustrate the functioning of the three trigger-types. For sake of convenience the examples elabrate on changes on table Cabinet only, but the working of the trigger structure is identical with regard to the other base-tables.

insert Assume we want to insert a new configuration into table Cabinet.[18] (Note that corrsponding entries in table Cabinet Portfolios need to be made manually.) Type:

```
1 INSERT INTO config_data.cabinet
2   (cab_id, ctr_id, cab_sdate, cab_hog_n, cab_care)
3   VALUES (1036, 1, '2014-01-01', 'Abbott', 'FALSE');
```

This action triggers `mv_config_ev_insert` implemented on table Cabinet, which in turn executes function `mv_config_ev_cabinet_it()`. The only thing the latter function does is executing a refresh of materialized view Configuration Events as defined by function `mv_config_ev_refresh_row(#`$_{ctr}$`, #`$_{date}$`)` (see page **??**), where `#`$_{ctr}$ is given by `NEW.ctr_id` and `#`$_{date}$ by `NEW.cab_sdate` as specified by the insert-command.
The result is that cabinet number 1036 also occurs in materialized view Configuration Events and all (public and non-public) views that are based on it (see Codebook and Section **??** in this manual).

update Assume we want to update the start date of an existing configuration in table Cabinet.[19] Type:

```
1 UPDATE config_data.cabinet
2   SET cab_sdate = '2014-03-15'::DATE
3   WHERE cab_id = 1036  \\
4   AND ctr_id = 1  \\
5   AND cab_sdate = '2014-01-01'::DATE;
```

This action triggers `mv_config_ev_update` implemented on table Cabinet, which in turn executes function `mv_config_ev_cabinet_ut()`. Again, this function executes a refresh of materialized view Configuration Events as defined by function `refresh_mv_config_events`
However, the results depend on whether the update on table Cabinet changes the start date of a cabinet configuration:

a) If after update the cabinet start date is unchanged (e.g. only the name of the Head of Government has been changed), no change occurs in materialized view Configuration Events occurs.

b) If, in contrast, the cabinet start date is changed after update, it follows a change in materialized view Configuration Events, executed by function`refresh_mv_config_events(#`$_{ctr}$ where `#`$_{ctr}$ is defined by the `ctr_id` that is recorded for the affected row in table Cabinet.

[18] http://www.postgresql.org/docs/9.3/static/sql-insert.html
[19] http://www.postgresql.org/docs/9.2/static/sql-update.html

Thus, whether a change in materialized view Configuration Events occurs on update of table Cabinet depends on whether the start date of a recorded cabinet configuration is changed.

delete Assume we want to delete an existing cabinet configuration from table Cabinet.[20] Type:

```
1 DELETE FROM config_data.cabinet
2 WHERE cab_id = 1036
3 AND ctr_id = 1
4 AND cab_sdate = '2014-03-15'::DATE;
```

This action triggers `mv_config_ev_delete` implemented on table Cabinet, which in turn executes function `mv_config_ev_cabinet_dt()`. Again, this function executes a refresh of materialized view Configuration Events as defined by function `mv_config_ev_refresh_row(#`$_{date}$`)`. However, the results is that the row(s) that correspond(s) to the respective cabinet configuration is also deleted from materialized view Cabinet Configurations.

What to do in the worst case If despite (or because of) the trigger-structure no changes in materialized view Confguration Events follow from changes performed on the base-tables, simply use function `refresh_mv_config_events(#`$_{ctr}$`)`, where #$_{ctr}$ is the country identifier.

For instance, typing

```
1 SELECT config_data.refresh_mv_config_events(1::SMALLINT)
```

initiates the following changes:

(1) disable all triggers implemented on materialized view Configuration Events;

(2) delete all rows identified by country identifier 1 (Austria);

(3) insert (i.e., exact copy of) all rows from view Configuration Events that are identified by country identifier 1;

(4) enable all triggers implemented on materialized view Configuration Events;

(5) update the date of the last refresh of materialized view Confuguration events in table Materialized Views to current date and time (see page 68); and

(4) execute function `update_mv_config_events()`, which results in executing all function that are implemented by triggers on materialized view Confuguration Events (fill empty cells with identifiers of cabinets, lower house configurations, etc. in charge, and computing configuration end dates).

[20] http://www.postgresql.org/docs/9.0/static/sql-delete.html

3.7 Consistency Checks

Consistency checks (CCs) are provided as views in the `config_data` schema of the database. CCs are generally powerful to trace

i) inconsistencies in recorded figures, i.e., primary data;

ii) inconsistencies between recorded and computed aggregate data; or

iii) missing data.

In the following subsections the existing CCs will be dicussed with regard to the tables and views they are based on, the level at which information is provided, the potential inconsistencies they reveal, how they are programmed, and their proper usage.

3.7.1 CC Cabinet start date

Consistency check `cc_cab_sdate` is based on the Cabinet and Lower House Election tables and provides information at the level of cabinet configurations.

It compares a cabinet's start date (`cab_sdate`) and the date of the lower house election it originates from (`lhelc_date`). Variable `date_dif` measures the difference between both dates in days; variable `prob_corr_ddif` is zero when the recorded date of cabinet formation, i.e., its start date, and the election date are equal.

CC `cc_cab_sdate` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_cab_sdate
AS
SELECT 'Cabinet'::TEXT AS inst_name, MAX(ctr_id) AS ctr_id,
  cab_id, cab_sdate, MAX(lhelc_date) AS lhelc_date,
  (cab_sdate-MAX(lhelc_date))AS date_dif,
  SIGN(COALESCE(cab_sdate-MAX(lhelc_date), 0)) AS prob_corr_ddif
  FROM
    (SELECT CABINET.ctr_id AS ctr_id, cab_id, cab_sdate, lhelc_date
      FROM
        (SELECT ctr_id, cab_id, cab_sdate
          FROM config_data.cabinet
        ) AS CABINET
      ,
        (SELECT ctr_id, lhelc_date
          FROM config_data.lh_election
        ) AS LHELC_DATE
      WHERE lhelc_date <= cab_sdate
      AND CABINET.ctr_id = LHELC_DATE.ctr_id
    ) AS CAB_SDATE
  GROUP BY cab_id, cab_sdate
ORDER BY cab_id, cab_sdate, lhelc_date;
```

The explicit, and actually empirically reasonable assumption is that government formation regularly takes some days in the countries that are covered in the PCDB. Because in the coding process the date of the election a cabinet originates from has been used as default cabinet start date, if the difference is equal to zero, this strongly indicates that no proper start date has been recorded. Case-specific research is still required for these cabient configurations.

Note: Information on the sources of cabinet start dates is stored in variable `cab_src` of the Cabinet table and variable `valid_cab_sdate`, in turn, is an individually coded dummy that indicates whether individual case research has been properly conducted an the source of information appears reliable.

3.7.2 CC Lower House start date

Consistency check `cc_lh_sdate` is based on tables Lower House and Lower House Election and provides information at the level of lower house (LH) configurations.

It compares a LH's start date (`lh_sdate`) and the date of the LH election it emanates from (`lhelc_date`). Variable `date_dif` measures the difference between both dates in days; variable `prob_corr_ddif` is zero when the recorded date of lower house formation, i.e., its start date, and the election date are equal. Logically, this is not expecte to be the case, as a LH's start date should be later than the date of its election.

CC `cc_lh_sdate` is programmed as follows

```
CREATE OR REPLACE VIEW config_data.cc_lh_sdate
AS
SELECT 'Lower House'::TEXT AS inst_name, ctr_id,
  lh_id, lhelc_date, lh_sdate, (lh_sdate-lhelc_date) AS date_dif,
  SIGN(COALESCE(lh_sdate-lhelc_date, 0)) AS prob_corr_ddif
  FROM
    (SELECT ctr_id, lh_id, lhelc_id, lh_sdate
      FROM config_data.lower_house
    ) AS LOWER_HOUSE
  JOIN
    (SELECT lhelc_id, lhelc_date
      FROM config_data.lh_election
    ) AS LHELC_DATE
  USING (lhelc_id)
ORDER BY lh_id;
```

The explicit, and actually empirically reasonable assumption is the first meeting in the first session of a newly elected LH (coded as start date) is regularly not on the same day as the election but on a later date in the countries that are covered in the PCDB. Because in the coding process the date of the election a LH emanates from has been used as default LH start date, if the difference is equal to zero, this strongly indicates that no proper start date has been recorded. Case-specific research is still required for these LH configurations.

Note: Information on the sources of LH start dates is provided in variable `lh_src` of the Lower House table and variable `valid_lh_sdate`, in turn, is an individually coded dummy that indicates whether individual case research has been properly conducted an the source of information appears reliable.

3.7.3 CC President start date

Consistency check `cc_prs_sdate` is based on the table Presidential Election and provides information at the level of presidents.

It compares the start date of presidency (prs_sdate) and the date of the corresponding presidential election (prselc_date). Variable date_dif measures the difference between both dates in days; variable prob_corr_ddif is zero when the recorded date of cabinet formation, i.e., its start date, and the election date are equal.

CC cc_prs_sdate is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_prs_sdate
AS
SELECT 'President'::TEXT AS inst_name, ctr_id,
  prselc_id, prselc_date, prs_sdate,
  (prs_sdate-prselc_date) as date_dif,
  SIGN(COALESCE(prs_sdate-prselc_date, 0)) AS prob_corr_ddif
  FROM config_data.presidential_election
ORDER BY prselc_id;
```

The explicit, and actually empirically reasonable assumption is that a new presidency regularly starts only some days after elections in the countries that are covered in the PCDB. Because in the coding process the date of the election of president has been used as default presidency start date, if the difference is equal to zero, this strongly indicates that no proper start date has been recorded. Case-specific research is still required for these presidencies.

Note: Information on the sources of presidency start dates is stored in variable prs_src of the Presidential Election table and variable valid_prs_sdate, in turn, is an individually coded dummy that indicates whether individual case research has been properly conducted an the source of information appears reliable.

3.7.4 CC Upper House start date

Consistency check cc_uh_sdate is based on tables Upper House and Upper House Election and provides information at the level of upper house (LH) configurations.

It compares a UH's start date (lh_sdate) and the date of the UH election it emanates from (uhelc_date). Variable date_dif measures the difference between both dates in days; variable prob_corr_ddif is zero when the recorded date of lower house formation, i.e., its start date, and the election date are equal. Logically, this is not expecte to be the case, as a UH's start date should be later than the date of its election.

CC cc_uh_sdate is programmed as follows

```
CREATE OR REPLACE VIEW config_data.cc_uh_sdate
AS
SELECT 'Upper House'::TEXT AS inst_name, ctr_id,
  uh_id, uhelc_date, uh_sdate,
  (uh_sdate-uhelc_date) AS date_dif,
  SIGN(COALESCE(uh_sdate-uhelc_date, 0)) AS prob_corr_ddif
  FROM
    (SELECT ctr_id, uh_id, uhelc_id, uh_sdate
      FROM config_data.upper_house
    ) AS UPPER_HOUSE
  JOIN
    (SELECT uhelc_id, uhelc_date
      FROM config_data.uh_election
    ) AS UHELC_DATE
  USING (uhelc_id)
ORDER BY uh_id;
```

The explicit, and actually empirically reasonable assumption is the first meeting in the first session of a newly elected UH (coded as start date) is regularly not on the same day as the election but on a later date in the countries that are covered in the PCDB. Because in the coding process the date of the election a UH emanates from has been used as default UH start date, if the difference is equal to zero, this strongly indicates that no proper start date has been recorded. Case-specific research is still required for these UH configurations.

Note: Information on the sources of UH start dates is provided in variable `uh_src` of the Upper House table and variable `valid_uh_sdate`, in turn, is an individually coded dummy that indicates whether individual case research has been properly conducted an the source of information appears reliable.

3.7.5 CC Institution start and election dates summary statistics

Consistency check `cc_specification_date_differences` is based on CCs Cabinet Start Date, Lower House Start Date, Presidency Start Date, and Upper House Start Date, and provides mean and median date difference, the number of differences unequal zero, and the total number of respective recorded configurations (N).

The difference between total and non-zero numbers of date differences hints to the number of principally suspect records (the higher, the more configurations where start dates are coded as equal to election date).

CC `cc_specification_date_differences` is programmed as follows

```
CREATE OR REPLACE VIEW config_data.cc_specifaction_date_differences
AS
SELECT 'Cabinet'::TEXT AS inst_name,
  SUM(date_dif)/COUNT(cab_id) AS mean_date_dif,
  ROUND(median(date_dif)) AS median_date_dif,
  SUM(prob_corr_ddif) AS nr_date_difs_recorded,
  COUNT(cab_id) as N,
  'all measures of date difference (%date_dif%) in days; note that cabinets
  are assumed to formate from elections of the legislature'::TEXT AS comment
  FROM config_data.cc_cab_sdate
UNION
SELECT 'Lower House'::TEXT AS inst_name,
  SUM(date_dif)/COUNT(lh_id) AS mean_date_dif,
  ROUND(median(date_dif)) AS median_date_dif,
  SUM(prob_corr_ddif) AS nr_date_difs_recorded,
  COUNT(lh_id) as N,
  'all measures of date difference (%date_dif%) in days'::TEXT AS comment
  FROM config_data.cc_lh_sdate
UNION
SELECT 'Upper House'::TEXT AS inst_name,
  SUM(date_dif)/COUNT(uh_id) AS mean_date_dif,
  ROUND(median(date_dif)) AS median_date_dif,
  SUM(prob_corr_ddif) AS nr_date_difs_recorded,
  COUNT(uh_id) as N,
  'all measures of date difference (%date_dif%) in days'::TEXT AS comment
  FROM config_data.cc_uh_sdate
UNION
SELECT 'President'::TEXT AS inst_name,
  SUM(date_dif)/COUNT(prselc_id) AS mean_date_dif,
  ROUND(median(date_dif)) AS median_date_dif,
  SUM(prob_corr_ddif) AS nr_date_difs_recorded,
  COUNT(prselc_id) as N,
  'all measures of date difference (%date_dif%) in days'::TEXT AS comment
```

```
34    FROM config_data.cc_prs_sdate;
```

3.7.6 CC Country time-series

Consistency check `cc_ctry_time_series` is based on view Configuration Year Duplicates (3.3.6) and provides information at the level of countries.

It compares the sum of rows listed for a country in the Configurations Country-Year view (3.5.2) to the difference in years between the earliest and the last recorded year for that country. If the numbers equal, an indicator, labeled `mismatch`, assumes a value equal to zero. Apparently, ones in variable `mismatch` indicate that the procedure to compile country years results in a failure for that year. Into depth analysis is requiered in this case.

CC `cc_ctry_time_series` is programmed as follows

```
1  CREATE OR REPLACE VIEW config_data.cc_ctry_time_series
2  AS
3  SELECT ctr_id, year_diff_ctr, ctr_rows_in_time_series,
4    SIGN(year_diff_ctr - ctr_rows_in_time_series)::INT AS mismatch
5  FROM
6    (SELECT DISTINCT ctr_id, (MAX(end_in_year)+1 - MIN(start_in_year))::INT AS year_diff_ctr
7      FROM config_data.view_configuration_year_duplicates
8      GROUP BY ctr_id
9    ) YEAR_DIFF_CY
10 JOIN
11   (SELECT ctr_id, COUNT(in_year)::INT AS ctr_rows_in_time_series
12     FROM
13       (SELECT DISTINCT ctr_id, in_year
14         FROM config_data.view_configuration_year_duplicates
15       ) DISTINCT_CY_DUPLICATES
16     GROUP BY ctr_id
17   ) N_ROWS_IN_TIME_SERIES
18 USING(ctr_id)
19 ORDER BY ctr_id;
```

3.7.7 CC Lower House parties' seat records missing

Consistency check `cc_missing_lh_pty_sts_records` is based on tables Party and Lower House Seat Results, and provides information at the level of individual parties nested in lower house elections.

It enlists all parties individual seat results for any LH election in which for at least one party (excl. category 'Other without seat') no total seat result is recorded (i.e., `Null`-values).

CC `cc_missing_lh_pty_sts_records` is programmed as follow:

```
1  CREATE OR REPLACE VIEW config_data.cc_missing_lh_pty_sts_records
2  AS
3  SELECT lh_id, pty_id, pty_lh_sts
4    FROM config_data.lh_seat_results
5    WHERE lh_id
6    IN
7      (SELECT lh_id
8        FROM
9          (SELECT lh_id, pty_id, pty_lh_sts
10           FROM config_data.lh_seat_results
```

```
11          ) AS LH_SEATS
12        JOIN
13          (SELECT pty_id, pty_abr
14            FROM config_data.party
15          ) AS PARTIES
16        USING(pty_id)
17        WHERE pty_lh_sts IS NULL
18        AND pty_abr NOT LIKE 'Other'
19      )
20  ORDER BY lh_id, pty_id;
```

Note: Missing seat records at the party level of lower house elections are consequential for several aggregate figures, including cabinet parties seat share in the lower house of table Configurations (see CC `cc_no_cab_lh_sts_shr`, subsection **??**), as well as and indicators, including Type A and B Volatility in seats ...

3.7.8 CC Party seat results in Lower House elections

Consistency check `cc_pty_lh_sts` is based on table Lower House Seat Results and provides information at the level of individual parties nested in lower house elections.

It enlists all LH election for which the computed sum of plurality and proportional seat results does not equal the recorded total.

CC `cc_pty_lh_sts` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.cc_pty_lh_sts
2  AS
3  SELECT lh_id, pty_id,
4    pty_lh_sts_pr, pty_lh_sts_pl, pty_lh_sts,
5    (COALESCE(pty_lh_sts_pr,0)+COALESCE(pty_lh_sts_pl,0)) AS pty_lh_sts_computed
6    FROM config_data.lh_seat_results
7  WHERE pty_lh_sts != (COALESCE(pty_lh_sts_pr,0)+COALESCE(pty_lh_sts_pl,0));
```

3.7.9 CC Lower House total seats

Consistency check `cc_lh_sts_ttl` is based on tables Lower House and Lower House Seat Results and provides information at the level of lower house (LH) configurations.

It enlists all LHs for which the recorded number of total seats deviates from the computed total seats of the corresponding LH election results.

CC `cc_lh_sts_ttl` is programmed as follows:

```
1  CREATE OR REPLACE VIEW config_data.cc_lh_sts_ttl
2  AS
3  SELECT lh_id, lh_sts_ttl, lh_sts_ttl_computed
4  FROM
5    (SELECT lh_id, lh_sts_ttl
6      FROM config_data.lower_house
7    ) AS LOWER_HOUSE
8  JOIN
9    (SELECT lh_id,
10     SUM(COALESCE(pty_lh_sts_pl,0)
11     + COALESCE(pty_lh_sts_pr,0))::NUMERIC
12       AS lh_sts_ttl_computed
```

```
13      FROM config_data.lh_seat_results
14      GROUP BY lh_id
15    ) AS LH_STS_TTL_COMPUTED
16  USING(lh_id)
17  WHERE lh_sts_ttl <> lh_sts_ttl_computed
18  ORDER BY lh_id;
```

3.7.10 CC Lower House election total distributed seats

Consistency check `cc_lh_sts_ttl_distrib` is based on tables Lower House Election and Lower House Seat Results and provides information at the level of lower houses.

It enlists all lower houses for which the recorded number of total seats (as listed in the Lower House Election table) deviates from the computed total seats. Lower houses enlisted potentially represent lower houses in which not all available seats were distributed. Deviations can, however, also results from missing data on parties' seat results. Double-checking is recommended.

CC `cc_lh_sts_ttl_distrib` is programmed as follows:

```
1  CREATE VIEW config_data.cc_lh_sts_ttl_distrib
2  AS
3  SELECT *
4    FROM
5      (SELECT lh_id, lhelc_id, lhelc_sts_ttl
6        FROM
7          (SELECT lh_id, lhelc_id
8            FROM config_data.lower_house
9          ) AS LOWER_HOUSE
10       JOIN
11         (SELECT lhelc_id, lhelc_sts_ttl
12           FROM config_data.lh_election
13         ) AS LH_ELECTION
14       USING(lhelc_id)
15     ) AS RECORDED
16   JOIN
17     (SELECT lh_id,SUM(pty_lh_sts) AS lh_sts_ttl_computed
18       FROM config_data.lh_seat_results
19       GROUP BY lh_id
20     ) AS COMPUTED
21 USING(lh_id)
22 WHERE lh_sts_ttl_computed != lhelc_sts_ttl;
```

3.7.11 CC Lower House Seat A Volatility

Consistency check `cc_lh_vola_sts` is based on table Lower House Election and view Lower House Seat A Volatility, and provides information at the level of lower house elections.

It enlists all lower house elections for which the computed and recorded figures of Seat A Volatility deviate after the 7th decimal place.

CC `cc_lh_vola_sts` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_lh_vola_sts
AS
SELECT *
  FROM
    (SELECT lhelc_id, lh_vola_sts_computed
      FROM
        (SELECT lhelc_id, lh_id
          FROM config_data.lower_house
        ) AS LOWER_HOUSE
      JOIN
        (SELECT lh_id, lh_vola_sts_computed
          FROM config_data.view_lh_vola_sts
        ) AS LH_VOLA_STS
      USING (lh_id)
    ) AS COMPUTED
  FULL OUTER JOIN
    (SELECT lhelc_id, lhelc_vola_sts
      FROM config_data.lh_election
    ) AS RECORDED
  USING (lhelc_id)
  WHERE TRUNC(lh_vola_sts_computed::NUMERIC, 7) != TRUNC(lhelc_vola_sts::NUMERIC, 7);
```

Note: because seats destributions vary with lower houses (not only with elections, as, e.g., party splits alter the lower house configuration often substantially), the computed figure varies with lower houses, while the recorded figure varies with elections. However, in lower houses that derive directly from an election (a corrsponding election identifier is recorded) should have the same volatility value.

3.7.12 CC Lower House Seat B Volatility

Consistency check `cc_lh_volb_sts` is based on table Lower House Election and view Lower House Seat B Volatility, and provides information at the level of lower house elections.

It enlists all lower house elections for which the computed and recorded figures of Seat B Volatility deviate after the 7^{th} decimal place.

CC `cc_lh_volb_sts` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_lh_volb_sts
AS
SELECT *
  FROM
(   SELECT lhelc_id, lh_volb_sts_computed
      FROM
        (SELECT lhelc_id, lh_id
          FROM config_data.lower_house
        ) AS LOWER_HOUSE
      JOIN
        (SELECT lh_id, lh_volb_sts_computed
          FROM config_data.view_lh_volb_sts
        ) AS LH_VOLB_STS
      USING (lh_id)
    ) AS COMPUTED
  FULL OUTER JOIN
    (SELECT lhelc_id, lhelc_volb_sts
      FROM config_data.lh_election
    ) AS RECORDED
  USING (lhelc_id)
WHERE TRUNC(lh_volb_sts_computed::NUMERIC, 7) != TRUNC(lhelc_volb_sts::NUMERIC, 7);
```

Note: because seats destributions vary with lower houses (not only with elections, as, e.g., party splits alter the lower house configuration often substantially), the computed figure varies with lower houses, while the recorded figure varies with elections. However, in lower houses that derive directly from an election (a corrsponding election identifier is recorded) should have the same volatility value.

3.7.13 CC Lower House election vote records and seat results missing

Consistency check `cc_missing_lhelc_pty_vts_and_sts_records` is based on tables Party, Lower House Seat Results and Lower House Vote Results, and provides information at the level of individual parties nested in lower house elections.

It enlists all parties individual vote and seat results for any LH election in which for at least one party (incl. categories 'Others with seat' and 'Independents') neither plurality nor proportional vote result are recorded (i.e., `Null`-values).

CC `cc_missing_lhelc_pty_vts_and_sts_records` is programmed as follow:

```
CREATE VIEW config_data.cc_missing_lhelc_pty_vts_and_sts_records
AS
SELECT *
FROM
  (SELECT lhelc_id, pty_id, pty_lh_vts_pr, pty_lh_vts_pl
    FROM config_data.lh_vote_results
    WHERE lhelc_id
    IN
      (SELECT lhelc_id
        FROM
          (SELECT lhelc_id, pty_id, pty_lh_vts_pr, pty_lh_vts_pl F
            ROM config_data.lh_vote_results
          ) AS LH_VOTES
        JOIN
          (SELECT pty_id, pty_abr
            FROM config_data.party
          ) AS PARTIES
        USING(pty_id)
        WHERE pty_lh_vts_pr IS NULL AND pty_lh_vts_pl IS NULL
        AND pty_abr NOT LIKE '%Other')
  ) AS VOTES
JOIN
  (SELECT lhelc_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
    FROM config_data.lh_seat_results
  ) SEATS
USING(lhelc_id, pty_id);
```

Note: Missing vote records at the party level of lower house elections are consequential for several aggregate figures and indicators, including Gallagher's Least-Square Index of Disproportionality in vote and seat results ...

3.7.14 CC Lower House election vote records missing

Consistency check `cc_missing_lhelc_pty_vts_records` is based on tables Party and Lower House Vote Results, and provides information at the level of individual parties nested in lower house elections.

It enlists all parties individual vote results for any LH election in which for at least one party (incl. categories 'Others with seat' and 'Independents') neither plurality nor proportional vote result are recorded (i.e., `Null`-values).

CC `cc_missing_lhelc_pty_vts_records` is programmed as follow:

```
CREATE OR REPLACE VIEW config_data.cc_missing_lhelc_pty_vts_records
AS
SELECT lhelc_id, pty_id,
  (COALESCE(pty_lh_vts_pr,0)
  + COALESCE(pty_lh_vts_pl,0))
    AS pty_lh_vts_computed
  FROM config_data.lh_vote_results
  WHERE lhelc_id
  IN
    (SELECT lhelc_id
      FROM
        (SELECT lhelc_id, pty_id,
          (COALESCE(pty_lh_vts_pr,0)
          + COALESCE(pty_lh_vts_pl,0))
            AS pty_lh_vts_computed
          FROM config_data.lh_vote_results
        ) AS LH_VOTES
      JOIN
        (SELECT pty_id, pty_abr
          FROM config_data.party
        ) AS PARTIES
      USING(pty_id)
    WHERE pty_lh_vts_computed = 0
    AND pty_abr NOT LIKE '%Other'
    )
ORDER BY lhelc_id, pty_id;
```

Note: Missing vote records at the party level of lower house elections are consequential for several aggregate figures and indicators, including the Type A and B Volatility in votes ...

3.7.15 CC Lower House Election Vote A Volatility

Consistency check `cc_lhelc_vola_vts` is based on table Lower House Election and view Lower House Election Vote A Volatility, and provides information at the level of lower house (LH) elections.

It enlists all LH elections for which the computed and recorded figures of Vote A Volatility deviate after the 7th decimal place.

CC `cc_lhelc_vola_vts` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_lhelc_vola_vts
AS
SELECT *
  FROM
    (SELECT lhelc_id, lhelc_vola_vts_computed
      FROM config_data.view_lhelc_vola_vts
    ) AS COMPUTED
  FULL OUTER JOIN
    (SELECT lhelc_id, lhelc_vola_vts
      FROM config_data.lh_election
    ) AS RECORDED
  USING (lhelc_id)
WHERE TRUNC(lhelc_vola_vts_computed::NUMERIC, 7) != TRUNC(lhelc_vola_vts::NUMERIC, 7);
```

3.7.16 CC Lower House Election Vote B Volatility

Consistency check `cc_lhelc_volb_vts` is based on table Lower House Election and view Lower House Election Vote B Volatility, and provides information at the level of lower house (LH) elections.

It enlists all LH elections for which the computed and recorded figures of Vote B Volatility deviate after the 7th decimal place.

CC `cc_lhelc_volb_vts` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_lhelc_volb_vts
AS
SELECT *
  FROM
    (SELECT lhelc_id, lhelc_volb_vts_computed
      FROM config_data.view_lhelc_volb_vts
    ) AS COMPUTED
  FULL OUTER JOIN
    (SELECT lhelc_id, lhelc_volb_vts
      FROM config_data.lh_election
    ) AS RECORDED
  USING (lhelc_id)
WHERE TRUNC(lhelc_volb_vts_computed::NUMERIC, 7) != TRUNC(lhelc_volb_vts::NUMERIC, 7);
```

3.7.17 CC Lower House Election Vote Results

Consistency check `cc_lhelc_vote_results` is based on tables Lower House Elections and Lower House Vote Results, and provides information at the level of lower house elections.

It enlists all LH elections for which the recorded number of proportional votes and/or of plurality votes do not equal the computed aggregates.

CC `cc_lhelc_vote_results` is programmed as follows:

```
CREATE VIEW config_data.cc_lhelc_vote_results
AS
SELECT lhelc_id,
  lhelc_vts_pr, lhelc_vts_pr_computed,
  lhelc_vts_pl, lhelc_vts_pl_computed
  FROM
    (SELECT lhelc_id,
      COALESCE(lhelc_vts_pr,0) AS lhelc_vts_pr,
      COALESCE(lhelc_vts_pl,0) AS lhelc_vts_pl
      FROM config_data.lh_election
    ) AS LH_ELECTION
  JOIN
    (SELECT lhelc_id,
      SUM(COALESCE(pty_lh_vts_pr,0)) AS lhelc_vts_pr_computed,
      SUM(COALESCE(pty_lh_vts_pl,0)) AS lhelc_vts_pl_computed
      FROM config_data.lh_vote_results
      GROUP BY lhelc_id
    ) AS COMPUTED
  USING(lhelc_id)
  WHERE lhelc_vts_pr != lhelc_vts_pr_computed
  OR lhelc_vts_pl != lhelc_vts_pl_computed
ORDER BY lhelc_id;
```

3.7.18 CC Cabinet Head of Government

Consistency check `cc_cabinet_hog_info` is based on the Cabinet Portfolios table and provides information at the level of cabinet configurations.

It enlists all cabinet configurations for which more than one party is recorded as producing the Head of Government (HOG) (variable `pty_cab_hog`).

The view is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_cabinet_hog_info
AS
SELECT cab_id, COUNT(pty_id)
  FROM config_data.cabinet_portfolios
  WHERE pty_cab_hog IS TRUE
  GROUP BY cab_id
  HAVING COUNT(pty_id) <> 1
ORDER BY cab_id;
```

It groups rows in cabinet portfolios (i.e., parties) by `cab_id`, the identifier of cabinet configurations (equivalent to unique combinations of country and cabinet start date), and counts parties within a cabinet which are recorded to produce the HOG. If this count is *not* equal to one, this indicates an inconsistency in the data, because logically only one party can produce the HOG. Accordingly, no cabinet configuration should be enlisted in this CC.

3.7.19 CC Head of State and cabinet in cohabitation

Consistency check `cc_hos_and_cab_cohabitation` is based on tables Cabinet Portfolios and Presidential Election and on view Configuration Events, and provides information at the level of political configurations (rows uniquely identified by combinations of `ctr_id` and `sdate`).

It enlists all configurations of cabinet and the Head of State (HOS) and provides information on respective party affiliations. If the HOS is affiliated to another party than the respective party/-parties that form the cabinet, variable`in_cohabitation` equals one, indicating cohabitation.

CC `cc_hos_and_cab_cohabitation` is programmed as follows:

```
CREATE OR REPLACE VIEW config_data.cc_hos_and_cab_cohabitation
AS
SELECT *, ABS(SIGN(pty_id-pty_id_hos)) AS in_cohabitation
  FROM
    (SELECT *
      FROM
        (SELECT ctr_id, sdate, cab_id, prselc_id
          FROM config_data.mv_configuration_events
        ) AS CONFIG_EVENTS
      FULL OUTER JOIN
        (SELECT cab_id, pty_id
          FROM config_data.cabinet_portfolios
          WHERE pty_cab IS TRUE
        ) AS ALL_CAB_PARTIES
      USING(cab_id)
    ) AS CONFIG_EVENTS_w_ALL_CAB_PARTIES
  FULL OUTER JOIN
    (SELECT prselc_id, pty_id AS pty_id_hos
      FROM config_data.presidential_election
    ) AS PTY_ID_HOS
```

```
21    USING(prselc_id)
22    WHERE prselc_id IS NOT NULL
23  ORDER BY ctr_id, sdate;
```

This CC enables to

i) compute variable `vto_prs`, which indicates whether the president (i.e., HOS) constitutes an open veto point vis-à-vis the government at the level of political configurations (see. subsection ??), and

ii) investigate what causes `NULL`-values in variable `vto_prs` of view Configurations, as it allows to check whether `NULL`-values in variable `in_cohabitation` are due to an lack of information on party affilation of cabinet parties or of the HOS.

3.7.20 CC Upper House seat records missing

Consistency check `cc_missing_uh_pty_sts_records` is based on tables Party and Upper House Seat Results, and provides information at the level of individual parties nested in upper houses (UH).

It enlists all parties individual seat results for any UH in which for at least one party (excl. category 'Other without seat') no seat result is recorded (i.e., `Null`-values).

CC `cc_missing_uh_pty_sts_records` is programmed as follow:

```
1  CREATE OR REPLACE VIEW config_data.cc_missing_uh_pty_sts_records
2  AS
3  SELECT uh_id, pty_id, pty_uh_sts
4    FROM config_data.uh_seat_results
5    WHERE uh_id
6    IN
7      (SELECT uh_id
8        FROM
9          (SELECT uh_id, pty_id, pty_uh_sts
10           FROM config_data.uh_seat_results
11         ) AS UH_SEATS
12       JOIN
13         (SELECT pty_id, pty_abr
14           FROM config_data.party
15         ) AS PARTIES
16       USING(pty_id)
17       WHERE pty_uh_sts IS NULL
18       AND pty_abr NOT LIKE 'Other'
19     )
20 ORDER BY uh_id, pty_id;
```

Note: Missing seat records at the party level of UHs are consequential for several aggregate figures and indicators, including cabinet parties seat share in the upper house of table Configurations (see CC `cc_no_cab_uh_sts_shr`, subsection 3.7.22).

3.7.21 CC Cabinet's Lower House seat share

Consistency check cc_no_cab_lh_sts_shr is based on views Cabinet's Lower House Seat Share and Configuration Events, and provides information at the level of political configurations.

It enlists all political configurations for which cabinet parties total seat share in the corresponding lower house cannot be computed (i.e., Null-value displayed). Null-values possibly stem from missing LH election seat result records (see CC cc_missing_lhelc_pty_sts_records, subsection ??), or configurations in which the given cabinet party identifiers do not match party identifiers in the corresponding LH election.

CC cc_no_cab_lh_sts_shr is programmed as follow:

```
1  CREATE OR REPLACE VIEW config_data.cc_no_cab_lh_sts_shr
2  AS
3  SELECT ctr_id, sdate,
4    CONFIGS.cab_id, CONFIGS.lh_id, cab_lh_sts_shr
5    FROM
6      (SELECT ctr_id, sdate, cab_id, lh_id, cab_lh_sts_shr
7        FROM config_data.view_cab_lh_sts_shr
8      ) CAB_LH_STS_SHR
9    FULL JOIN
10     (SELECT ctr_id, sdate, prselc_id, uh_id, lh_id, cab_id, year, edate
11       FROM config_data.mv_configuration_events
12     ) CONFIGS
13   USING (ctr_id, sdate)
14   WHERE cab_lh_sts_shr IS NULL
15   AND CONFIGS.cab_id IS NOT NULL
16   AND CONFIGS.lh_id IS NOT NULL
17 ORDER BY ctr_id, sdate NULLS FIRST;
```

3.7.22 CC Cabinet's Upper House seat share

Consistency check cc_no_cab_uh_sts_shr is based on views Cabinet's Upper House Seat Share and Configuration Events, and provides information at the level of political configurations.

It enlists all political configurations for which cabinet parties total seat share in the corresponding upper house (UH) cannot be computed (i.e., Null-value displayed). Null-values possibly stem from missing UH election seat result records (see CC cc_missing_uh_pty_sts_records, subsection ??), or configurations in which the given cabinet party identifiers do not match party identifiers in the corresponding UH.

CC cc_no_cab_uh_sts_shr is programmed as follow:

```
1  CREATE OR REPLACE VIEW config_data.cc_no_cab_uh_sts_shr
2  AS
3  SELECT ctr_id, sdate, CONFIGS.cab_id,
4    CONFIGS.uh_id, cab_uh_sts_shr
5    FROM
6      (SELECT ctr_id, sdate, cab_id, uh_id, cab_uh_sts_shr
7        FROM config_data.view_cab_uh_sts_shr
8      ) CAB_UH_STS_SHR
9    FULL JOIN
10     (SELECT ctr_id, sdate, prselc_id, uh_id, lh_id, cab_id, year, edate
11       FROM config_data.mv_configuration_events
```

```
12      ) CONFIGS
13    USING (ctr_id, sdate)
14    WHERE cab_uh_sts_shr IS NULL
15    AND CONFIGS.cab_id IS NOT NULL
16    AND CONFIGS.uh_id IS NOT NULL
17  ORDER BY ctr_id, sdate NULLS FIRST;
```

Note: For Germany (country identifier = 6) generally no seat share of cabinet parties in the UH can be computed. Consider manual computation!

3.7.23 CC Lower House and corresponding election

Consistency check `cc_no_lhelc_id_4_lh` is based on table Lower House and provides information at the level of lower houses (LH).

It enlists all LHs for which no corresponding LH election identifier is recorded (i.e., `Null`-value displayed).

CC `cc_no_cab_lh_sts_shr` is programmed as follow:

```
1  CREATE OR REPLACE VIEW config_data.cc_no_lhelc_id_4_lh
2  AS
3  SELECT lh_id, lhelc_id
4    FROM config_data.lower_house
5    WHERE lhelc_id IS NULL;
```

Note: `Null`-values cause missing in computation of `cab_lh_sts_shr` (see CC `cc_no_cab_lh_sts_shr`, subsection **??**).

3.7.24 CC LSq missing

Consistency check `cc_no_lsq` is based on tables Party, Lower House Seat Results, and Lower House Vote Results, and provides information at the level of lower houses (LH) elections.

It enlists all LH elctions for which no LSq (Gallagers Least-Squar index) can be computeted because of `Null`-values in one or more party's vote results, seat results, or both (excl. category 'Others without seat').

CC `cc_no_lsq` is programmed as follow:

```
1  CREATE OR REPLACE VIEW config_data.cc_no_lsq
2  AS
3  SELECT DISTINCT lhelc_id
4    FROM
5      (SELECT *
6        FROM
7          (SELECT lhelc_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
8            FROM
9              (SELECT lh_id, lhelc_id FROM config_data.lower_house) AS LH
10           JOIN
11             (SELECT lh_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
12               FROM config_data.lh_seat_results
13             ) AS LH_SEATS
14           USING(lh_id)
15         ) AS LH_SEATS
16       JOIN
```

```
17          (SELECT pty_id, pty_abr
18            FROM config_data.party
19          ) AS PARTIES
20        USING(pty_id)
21      ) AS SEATS
22    JOIN
23      (SELECT *
24        FROM
25          (SELECT lhelc_id, pty_id, pty_lh_vts_pr, pty_lh_vts_pl
26            FROM config_data.lh_vote_results
27          ) AS LH_SEATS
28        JOIN
29          (SELECT pty_id, pty_abr
30            FROM config_data.party
31          ) AS PARTIES
32        USING(pty_id)
33      ) AS VOTES
34    USING(lhelc_id, pty_id)
35    WHERE pty_lh_vts_pr IS NULL
36      AND pty_lh_vts_pl IS NULL
37      AND VOTES.pty_abr NOT LIKE '%Other'
38    OR pty_lh_sts_pr IS NULL
39      AND pty_lh_sts_pl IS NULL
40      AND SEATS.pty_abr NOT LIKE '%Other'
41  ORDER BY lhelc_id;
```

3.7.25 CC 'Othersw'-excluding LSq missing

Consistency check `cc_no_lsq_noothersw` is based on tables Party, Lower House Seat Results, and Lower House Vote Results, and provides information at the level of lower houses (LH) elections.

It enlists all LH elctions for which no LSq (Gallagers Least-Squar index) can be computeted because of `Null`-values in one or more party's vote results, seat results, or both (excl. categories 'Others without seat' and 'Others with seat').

CC `cc_no_lsq_noothersw` is programmed as follow:

```
1   CREATE OR REPLACE VIEW config_data.cc_no_lsq_noothersw
2   AS
3   SELECT DISTINCT lhelc_id
4     FROM
5       (SELECT *
6         FROM
7           (SELECT lhelc_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
8             FROM
9               (SELECT lh_id, lhelc_id FROM config_data.lower_house) AS LH
10            JOIN
11              (SELECT lh_id, pty_id, pty_lh_sts_pr, pty_lh_sts_pl
12                FROM config_data.lh_seat_results
13              ) AS LH_SEATS
14            USING(lh_id)
15          ) AS LH_SEATS
16        JOIN
17          (SELECT pty_id, pty_abr
18            FROM config_data.party
19          ) AS PARTIES
20        USING(pty_id)
21      ) AS SEATS
22    JOIN
23      (SELECT *
24        FROM
```

```
25          (SELECT lhelc_id, pty_id, pty_lh_vts_pr, pty_lh_vts_pl
26            FROM config_data.lh_vote_results
27          ) AS LH_SEATS
28        JOIN
29          (SELECT pty_id, pty_abr
30            FROM config_data.party
31          ) AS PARTIES
32        USING(pty_id)
33      ) AS VOTES
34    USING(lhelc_id, pty_id)
35    WHERE pty_lh_vts_pr IS NULL
36      AND pty_lh_vts_pl IS NULL
37      AND VOTES.pty_abr NOT LIKE '%Other'
38      AND VOTES.pty_abr NOT LIKE '%Otherw'
39    OR pty_lh_sts_pr IS NULL
40      AND pty_lh_sts_pl IS NULL
41      AND SEATS.pty_abr NOT LIKE '%Other'
42      AND SEATS.pty_abr NOT LIKE '%Otherw'
43  ORDER BY lhelc_id;
```

3.7.26 CC President candidates' electoral collage votes

Consistency check `cc_pres_elec_collage_vts` is based on table Presidential Election Vote Results and provides information at the level of presidential election candidates.

It enlists all candidates presidential election vote results for which electoral collage votes are recorded (variable `prselc_vts_clg` is *not* NULL) but PCDB records indicate that no electoral collage should have been involved in the election of the president (variable `prselc_clg` is FALSE). Thus one of both records is wrong.

CC `cc_pres_elec_collage_vts` is programmed as follow:

```
1   CREATE VIEW config_data.cc_pres_elec_collage_vts
2   AS
3   SELECT *
4     FROM
5       (SELECT prselc_id, prs_cnd_pty, prselc_rnd, prs_cnd_vts_clg, prs_cnd_vts_ppl
6         FROM config_data.pres_elec_vres
7       ) AS PRES_ELEC_VRES
8     JOIN
9       (SELECT prselc_id, prselc_rnd_ttl, prselc_vts_clg, prselc_clg
10        FROM config_data.presidential_election
11      ) AS PRES_ELEC
12    USING(prselc_id)
13    WHERE prselc_clg IS FALSE
14    AND prselc_vts_clg IS NOT NULL;
```

3.8 Keeping the PCDB updated

I may also provide a guide how to insert, update, and delete data from the tables contained in the PCDB. I have not yet developed any tool to insert data, e.g., from exel tables. Inserting a mass of data is thus far proceded manually, using SQL, and often painstacking.

The following paragraphs will use table Cabonet as an example to introdcue some minimal working examples (MWE) thought illustrate how data is inserted inot and deleted from the tables in the confgidata scheme, and how recorded date can be updated

THE MWEs can easily be transfered to the other base tables in the PCDB. However, it is imperative to stress that no data should be cahnged without having a clear idea of

(a) what is the primary key of a given table or the columns that uniquely identify rows;

(b) which referntial dependencies are implied by the structure of the PCDB; and accordingly,

(c) how incomplete insertation or updating, or thoughless deletion affects the integrity and constistency of the PCDB.

With respect to the MWE, (a) `cab_id` is primary key of table Cabinet while additionally `cab_sdate` in combination with `ctr_id` uniquely identify observations, i.e., rows.

With reespect to (b), `cab_id` is referenced as foreign key in table Cabinet Portfolios and, in combination with `pty_id`, uniquely identifies cabinet portfolios; table Cabinet being a base-table, `cab_ids` are sequenced in the Configuration view and thus are essential to compute configuration-specific indicators, such as veto constellations, and cabinet-parties seat share in the lower and upper houses; and `cab_ids` are selected by several triggers to identify previous or subsequent cabinets for any given cabinet (subsections **??** and **??**).

Lastly, in view of (c), though it is possible to insert a new observation to table Cabinet without providing, for instance, its start date, this would cause non-trivial problems in compiling view Configurations and selecting it as next cabinet for the preceeding cabinet configuration, to name but few. Users are thus strongly inclined to pay attention to the key and uniquness characteristics of a given table when inserting, updating or deleting data from it-

3.8.1 Insert

Adding new row (i.e., an observation) to a table is proceeded with the `INSERT INTO`- command, specifying first the table, second the columns, and third the values to insert. Though insertation does not requiere to specify the destination-columns when using the original order of columns of a table as default, specification is best-practice, as it guarantees for correctness of the procedure. A MWE reads as follows:

```
1 INSERT INTO config_data.cabinet
2   (cab_id, ctr_id, cab_sdate, cab_hog_n, cab_care)
3   VALUES (1036, 1, '2014-01-01', 'Abbott', 'FALSE');
```

Note that the values thought to insert need to match the specified types of the destination-columns. Typing instead

```
INSERT INTO config_data.cabinet
  (cab_id, ctr_id, cab_sdate, cab_hog_n, cab_care)
  VALUES (1036::NUMERIC(5,0), 1::SMALLINT, '2014-01-01'::DATE, 'Abbott'::NAME, 'FALSE'::BOOLEAN);
```

would thus avoid any surprises.

If one attempts to insert a value that does not match the type of the respective column, pgAdmin3 notes the error and states To recall the type of a given column, refer to the Codebook or browse the properties of the given table in pgAdmin3 (right click on table in menu bar, querying "")

It is generally recomended to refer to either the Codebook or Section X of the Manual, before inserting data into tables, as there are set constraints (e.g., NOT NULL, PRIMARY KEY, or UNIQUE) on some of the columns that make insertation of a value obligatory when adding a new row to the table.

In addition, it is best-practice to assign ascending integer counters to subsequent instituion configurations withn countries, thought the trigger structure that assigns identifiers of previous and next configurations to a current configuration does not require this order (see subsections 3.6.2 and **??**).

Finally, remember that the primary key of the cabinet table, cab_id, contributes to the unambigous identification of observations in the Cabinet Portfolio table. Following the tree of dependencies, inserting a new cabinet should be followed by specifying the correspoonding cabinet portfolio. Also, information on the on the newly inserted cabinet's portfolio is requiered to obtained meaningful information on the political configuration (i.e., the lower house, upper house, and/or presidency cabient parties face) in which it is embedded.

3.8.2 Update

Altering the values of an existing row in a table is proceeded with the UPDATE-command, specifying the table and the column of the values that is thought to be updated. Updating is achieved by SETting a column equal to some value that mathces the type of the respective column. A WHERE-clause is requiered to identify the row(s) which are ment to be updated. A MWE reads as follows:

```
UPDATE config_data.cabinet
  SET cab_sdate = '2014-03-15'::DATE
  WHERE cab_id = 1036
  AND ctr_id = 1
  AND cab_sdate = '2014-01-01'::DATE;
```

Here, the value of the column that reports the cabinet's start date is updated in only one observation, as the attributes cab_id, and cab_id and cab_id, respectively, uniquely identify rows in the Cabinet table.

It is possible to update information of more than one row.

4 Bibliography

Abou-Chadi, Tarik. 2014. "Niche Party Success and Mainstream Party policy Shifts: How Green and radical Right Parties Differ in Their Impact." *British Journal of Political Science* FirstView.
URL: *http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=9290332&fileId=S0007123414000155* 6

Carey, John M. and Simon Hix. 2008. *Maximizing Representation and Accountability in the Design of Electoral systems: Data Codebook Version 2.0.* 8

Carey, John M. and Simon Hix. 2011. "The Electoral Sweet Spot: Lsow-Magnitude Proportional Electoral Systems." *American Journal of Political Science* 55. 7

Döring, Holger and Philip Manow. 2012. "Parliament and government composition database (ParlGov): An infrastructure for empirical information on parties, elections and governments in modern democracies. Version 12/10 – 15 October 2012.".
URL: *http://Parlgov.org/stable/data.html* 3.2.2, 15

EU, European Union. 2015. "EU member mountries.".
URL: *http://europa.eu/about-eu/countries/member-countries/index_en.htm* 2

Gallagher, M. and P. Mitchell. 2005. *The Politics of Electoral Systems.* Oxford University Press. 3.3.18, 3.3.19, 7

Gallagher, Michael. 1991. "Proportionality, Disproportionality and Electoral Systems." *Electoral Studies* 10:33–51. 3.3.23, 9

Gallagher, Michael. 1992. "Comparing Proportional Representation Electoral Systems: Quotas, Thresholds, Paradoxes and Majorities." *British Journal of Political Science* 22:469–496. 9

ISO, International Organization for Standardization. 2015. "Country Codes – ISO 3166.".
URL: *http://www.iso.org/iso/home/standards/country_codes.htm* 1

Kollman, K., A. Hicken, D. Caramani, D. Backer and D. Lublin. 2014. "Constituency-level elections archive [data file and codebook].".
URL: *http://www.electiondataarchive.org/cite.html* 16

Laakso, Markku and Rein Taagepera. 1979. "The 'Effective' Number of Parties: A Measure with Application to West Europe." *Comparative Political Studies* 12(1):3–27. 3.3.18, 5

Lijphart, Arend. 1994. *Electoral Systems and Party Systems: A Study of Twenty-Seven Democracies, 1945-1990.* Comparative European Politics Series Oxford: Oxford University Press. 3.3.22

Nohlen, Dieter. 2001. *Elections in Asia and the Pacific. A Data Handbook*. Oxford and others: Oxford Universtiy Press. 2

Nohlen, Dieter. 2005. *Elections in the Americas. A Data Handbook*. Oxford and others: Oxford Universtiy Press. 2

Nohlen, Dieter. 2010. *Elections in Europe. A Data Handbook*. Baden-Baden: Nomos Verlags-Gesellschaft. 2

OECD, The Organisation for Economic Co-operation and Development. 2015. "Members and Partners: List of OECD Member countries – Ratification of the Convention on the OECD.". **URL:** *http://www.oecd.org/about/membersandpartners/list-oecd-member-countries.htm* 3

Powell, Eleanor Neff and Joshua A. Tucker. 2013. "Revisiting electoral Volatility in Post-Communist Sountries: New Data, New Results and New Approaches." *British Journal of Political Science* 44:123–147. 3.3.24, 3.3.26, 10, 11

Taagepera, Rein. 2002. "Nationwide threshold of representation." *Electoral Studies* 21(3):383–401. 3.3.22

Volkens, Andrea, Pola Lehmann, Nicolas Merz, Sven Regel, Annika Werner, Onawa Promise Lacewell and Henrike Schultze. 2013. "The Manifesto Data Collection (version 2013b).". **URL:** *https://manifesto-project.wzb.eu/* 3.2.2, 14

WTO, World Trade Organization. 2015. "Members and Observers.". **URL:** *https://www.wto.org/english/thewto_e/whatis_e/tif_e/org6_e.htm* 4

5 Appendix

5.1 Details on Trigger functions

5.1.1 Description of Triggers to Identify Previous Instituion-Configrations

Identify Previous Cabinet within Country Trigger `trg_cab_prv_id` is implemented on table Cabinet and inserts data into cells of column `cab_prv_id`.

Specifically, function `trg_cab_nxt_id()` selects the identifier of the previous cabinet configuration, as identified by the next lower date of all cabinets recorded for a country. It is programmed as follows:

```
1  CREATE OR REPLACE FUNCTION config_data.trg_cab_prv_id()
2  RETURNS trigger AS $function$
3    BEGIN
4      NEW.cab_prv_id :=
5        (SELECT cab_id FROM config_data.cabinet
6        WHERE cab_sdate < NEW.cab_sdate
7        AND ctr_id = NEW.ctr_id
8        ORDER BY ctr_id, cab_sdate DESC
9        LIMIT 1);
10   RETURN NEW;
11   END;
12 $function$ LANGUAGE plpgsql;
13
14 CREATE TRIGGER trg_cab_prv_id
15   BEFORE INSERT OR UPDATE ON config_data.cabinet
16   FOR EACH ROW
17   EXECUTE PROCEDURE config_data.trg_cab_prv_id();
```

Trigger `trg_cab_prv_id` is executed for each row before inserting or updating of data in table Cabinet is performed.

Identify Previous Lower House within Country Trigger `trg_lh_prv_id` is implemented on table Lower House and inserts data into cells of column `lh_prv_id`.

Specifically, function `trg_lh_prv_id()` selects the identifier of the next recorded lower house, as identified by the next bigger date of all lower houses recorded for a country. It is programmed as follows:

```
1  -- Trigger selects previous LH id
2  CREATE FUNCTION config_data.trg_lh_prv_id() RETURNS trigger AS $function$
3    BEGIN
4      NEW.lh_prv_id :=
5        (SELECT lh_id FROM config_data.lower_house
```

```
6         WHERE lh_sdate < NEW.lh_sdate
7         AND ctr_id = NEW.ctr_id
8         ORDER BY ctr_id, lh_sdate DESC
9         LIMIT 1); -- selects next lowest LH start date
10    RETURN NEW;
11    END;
12  $function$ LANGUAGE plpgsql;
13
14  CREATE TRIGGER trg_lh_prv_id
15    BEFORE INSERT OR UPDATE ON config_data.lower_house
16    FOR EACH ROW
17    EXECUTE PROCEDURE config_data.trg_lh_prv_id();
```

Trigger `trg_lh_prv_id` is executed for each row before inserting or updating of data in table Lower House is performed.

Identify Previous Lower House Election within Country Trigger `trg_lhelc_prv_id` is implemented on table Lower House Election and inserts data into cells of column `lhelc_nxt_id`.

Specifically, function `trg_lhelc_prv_id()` selects the identifier of the previous lower house election, as identified by the next smaller date of all recorded lower houses election dates for a country. It is programmed as follows:

```
1   CREATE FUNCTION config_data.trg_lhelc_prv_id() RETURNS trigger AS $function$
2     BEGIN
3       NEW.lhelc_prv_id :=
4         (SELECT lhelc_id FROM config_data.lh_election
5         WHERE lhelc_date < NEW.lhelc_date
6         AND ctr_id = NEW.ctr_id
7         ORDER BY ctr_id, lhelc_date DESC
8         LIMIT 1);
9     RETURN NEW;
10    END;
11  $function$ LANGUAGE plpgsql;
12
13  CREATE TRIGGER trg_lhelc_prv_id
14    BEFORE INSERT OR UPDATE ON config_data.lh_election
15    FOR EACH ROW
16    EXECUTE PROCEDURE config_data.trg_lhelc_prv_id();
```

Trigger `trg_lhelc_prv_id` is executed for each row before inserting or updating of data in table Lower House Election is performed.

Identify Previous Upper House within Country Trigger `trg_uh_prv_id` is implemented on table Upper House and inserts data into cells of column `uh_prv_id`.

Specifically, function `trg_uh_prv_id()` selects the identifier of the next recorded upper house configuration, as identified by the next bigger date of all upper houses recorded for a country. It is programmed as follows:

```
1   CREATE OR REPLACE FUNCTION config_data.trg_uh_prv_id()
2   RETURNS trigger AS $function$
3     BEGIN
4       NEW.uh_prv_id :=
5         (SELECT uh_id FROM config_data.upper_house
6         WHERE uh_sdate < NEW.uh_sdate
7         AND ctr_id = NEW.ctr_id
8         ORDER BY ctr_id, uh_sdate DESC
9         LIMIT 1);
```

```
10    RETURN NEW;
11    END;
12  $function$ LANGUAGE plpgsql;
13
14  CREATE TRIGGER trg_uh_prv_id
15    BEFORE INSERT OR UPDATE ON config_data.upper_house
16    FOR EACH ROW
17    EXECUTE PROCEDURE config_data.trg_uh_prv_id();
```

Trigger `trg_uh_prv_id` is executed for each row before inserting or updating of data in table Upper House is performed.

Identify Previous Presidential Election within Country Trigger `trg_prselc_prv_id` is implemented on table Presidential Election and inserts data into cells of column `prselc_prv_id`.

Specifically, function `trg_prselc_prv_id()` selects the identifier of the previous presidential election, as identified by the next lower date of all presidential elections recorded for a country. It is programmed as follows:

```
1  CREATE FUNCTION config_data.trg_prselc_prv_id()
2  RETURNS trigger AS $function$
3    BEGIN
4      NEW.prselc_prv_id :=
5        (SELECT prselc_prv_id FROM config_data.presidential_election
6        WHERE prselc_date < NEW.prselc_date
7        AND ctr_id = NEW.ctr_id
8        ORDER BY ctr_id, prselc_date DESC
9        LIMIT 1);
10   RETURN NEW;
11   END;
12 $function$ LANGUAGE plpgsql;
13
14 CREATE TRIGGER trg_prselc_prv_id
15   BEFORE INSERT OR UPDATE ON config_data.presidential_election
16   FOR EACH ROW
17   EXECUTE PROCEDURE config_data.trg_prselc_prv_id();
```

Trigger `trg_prselc_prv_id` is executed for each row before inserting or updating of data in table Presidential Election is performed.

5.1.2 Description of Triggers to Identify Next Instituion-Configrations

Identify Next Cabinet within Country Trigger `trg_cab_nxt_id` is implemented on table Cabinet and inserts data into cells of column `cab_nxt_id`.

Specifically, function `trg_cab_nxt_id()` selects the identifier of the next recorded cabinet configuration, as identified by the next bigger date of all cabinets recorded for a country. It is programmed as follows:

```
1  CREATE OR REPLACE FUNCTION config_data.trg_cab_nxt_id() RETURNS trigger AS $function$
2    BEGIN
3      NEW.cab_nxt_id :=
4        (SELECT cab_id FROM config_data.cabinet
5        WHERE cab_sdate > NEW.cab_sdate
6        AND ctr_id = NEW.ctr_id
7        ORDER BY ctr_id, cab_sdate ASC
8        LIMIT 1);
```

```
9     RETURN NEW;
10    END;
11  $function$ LANGUAGE plpgsql;
12
13  CREATE TRIGGER trg_cab_nxt_id
14    BEFORE INSERT OR UPDATE ON config_data.cabinet
15    FOR EACH ROW
16    EXECUTE PROCEDURE config_data.trg_cab_nxt_id();
```

Trigger `trg_cab_nxt_id` is executed for each row before inserting or updating data of table Cabinet.

Identify Next Lower House within Country Trigger `trg_lh_nxt_id` is implemented on table Lower House and inserts data into cells of column `lh_nxt_id`.

Specifically, function `trg_lh_nxt_id()` selects the identifier of the next recorded lower house, as identified by the next bigger date of all lower houses recorded for a country. It is programmed as follows:

```
1   CREATE FUNCTION config_data.trg_lh_nxt_id() RETURNS trigger AS $function$
2     BEGIN
3       NEW.lh_nxt_id :=
4         (SELECT lh_id FROM config_data.lower_house
5         WHERE lh_sdate > NEW.lh_sdate
6         AND ctr_id = NEW.ctr_id
7         ORDER BY ctr_id, lh_sdate ASC -- ascending
8         LIMIT 1);
9     RETURN NEW;
10    END;
11  $function$ LANGUAGE plpgsql;
12
13  CREATE TRIGGER trg_lh_nxt_id
14    BEFORE INSERT OR UPDATE ON config_data.lower_house
15    FOR EACH ROW
16    EXECUTE PROCEDURE config_data.trg_lh_nxt_id();
```

Trigger `trg_lh_nxt_id` is executed for each row before inserting or updating data of table Lower House.

Identify Next Lower House Election within Country Trigger `trg_lhelc_nxt_id` is implemented on table Lower House Election and inserts data into cells of column `lhelc_nxt_id`.

Specifically, function `trg_lhelc_nxt_id()` selects the identifier of the next recorded lower house election, as identified by the next bigger date of all recorded lower houses election dates for a country. It is programmed as follows:

```
1   CREATE OR REPLACE FUNCTION config_data.trg_lhelc_nxt_id() RETURNS trigger AS $function$
2     BEGIN
3       NEW.lhelc_nxt_id :=
4         (SELECT lhelc_id FROM config_data.lh_election
5         WHERE lhelc_date > NEW.lhelc_date
6         AND ctr_id = NEW.ctr_id
7         ORDER BY ctr_id, lhelc_date ASC
8         LIMIT 1);
9     RETURN NEW;
10    END;
11  $function$ LANGUAGE plpgsql;
12
13  CREATE TRIGGER trg_lhelc_nxt_id
```

```
14    BEFORE INSERT OR UPDATE ON config_data.lh_election
15    FOR EACH ROW
16    EXECUTE PROCEDURE config_data.trg_lhelc_nxt_id();
```

Trigger `trg_lhelc_nxt_id` is executed for each row before inserting or updating data of table Lower House Election.

5.1.3 Description of Triggers that Insert Corresponding Identifiers in Political Configurations

The following functions select corresponding institution-identifiers and triggers insert them into the respective empty cells that result from the sequencing procedure to identify countries' politcal configurations (cf. subsection 3.6.4).

```
1  -- Select corresponding cab_id
2  CREATE FUNCTION config_data.trg_mv_config_ev_prv_cab_id()
3  RETURNS trigger AS $function$
4    BEGIN
5      IF
6        OLD.cab_id IS NOT NULL THEN NEW.cab_id = OLD.cab_id;
7      ELSE
8        NEW.cab_id :=
9        (SELECT cab_id FROM config_data.mv_configuration_events
10       WHERE sdate < NEW.sdate
11       AND ctr_id = NEW.ctr_id
12       ORDER BY ctr_id, sdate DESC
13       LIMIT 1);
14     END IF;
15   RETURN NEW;
16   END;
17 $function$ LANGUAGE plpgsql;
18
19 DROP TRIGGER IF EXISTS trg_it_mv_config_ev_prv_cab_id
20 ON config_data.mv_configuration_events;
21 CREATE TRIGGER trg_it_mv_config_ev_prv_cab_id
22   AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW -- after insert
23   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_cab_id();
24
25 DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_prv_cab_id
26 ON config_data.mv_configuration_events;
27 CREATE TRIGGER trg_dt_mv_config_ev_prv_cab_id
28   AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW -- after delete
29   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_cab_id();
30
31 DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_prv_cab_id
32 ON config_data.mv_configuration_events;
33 CREATE TRIGGER trg_ut_mv_config_ev_prv_cab_id
34   BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW -- before update
35   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_cab_id();
36
37 -- Select corresponding lh_id
38 CREATE FUNCTION config_data.trg_mv_config_ev_prv_lh_id()
39 RETURNS trigger AS $function$
40   BEGIN
41     IF
42       OLD.lh_id IS NOT NULL THEN NEW.lh_id = OLD.lh_id;
43     ELSE
44       NEW.lh_id :=
45       (SELECT lh_id FROM config_data.mv_configuration_events
46       WHERE sdate < NEW.sdate
47       AND ctr_id = NEW.ctr_id
48       ORDER BY ctr_id, sdate DESC
```

```
49        LIMIT 1);
50      END IF;
51    RETURN NEW;
52    END;
53  $function$ LANGUAGE plpgsql;
54
55  DROP TRIGGER IF EXISTS trg_it_mv_config_ev_prv_lh_id
56  ON config_data.mv_configuration_events;
57  CREATE TRIGGER trg_it_mv_config_ev_prv_lh_id
58    AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW -- after insert
59    EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_lh_id();
60
61  DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_prv_lh_id
62  ON config_data.mv_configuration_events;
63  CREATE TRIGGER trg_dt_mv_config_ev_prv_lh_id
64    AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW -- after delete
65    EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_lh_id();
66
67  DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_prv_lh_id
68  ON config_data.mv_configuration_events;
69  CREATE TRIGGER trg_ut_mv_config_ev_prv_lh_id
70    BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW -- before update
71    EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_lh_id();
72
73
74  -- Select corresponding lhelc_id
75  CREATE FUNCTION config_data.trg_mv_config_ev_prv_lhelc_id()
76  RETURNS trigger AS $function$
77    BEGIN
78      IF
79        OLD.lhelc_id IS NOT NULL THEN NEW.lhelc_id = OLD.lhelc_id;
80      ELSE
81        NEW.lhelc_id :=
82        (SELECT lhelc_id FROM config_data.mv_configuration_events
83        WHERE sdate < NEW.sdate
84        AND ctr_id = NEW.ctr_id
85        ORDER BY ctr_id, sdate DESC
86        LIMIT 1);
87      END IF;
88    RETURN NEW;
89    END;
90  $function$ LANGUAGE plpgsql;
91
92  DROP TRIGGER IF EXISTS trg_it_mv_config_ev_prv_lhelc_id
93  ON config_data.mv_configuration_events;
94  CREATE TRIGGER trg_it_mv_config_ev_prv_lhelc_id
95    AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW -- after insert
96    EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_lhelc_id();
97
98  DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_prv_lhelc_id
99  ON config_data.mv_configuration_events;
100 CREATE TRIGGER trg_dt_mv_config_ev_prv_lhelc_id
101   AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW -- after delete
102   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_lhelc_id();
103
104 DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_prv_lhelc_id
105 ON config_data.mv_configuration_events;
106 CREATE TRIGGER trg_ut_mv_config_ev_prv_lhelc_id
107   BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW -- before update
108   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_lhelc_id();
109
110 -- Select corresponding uh_id
111 CREATE FUNCTION config_data.trg_mv_config_ev_prv_uh_id()
112 RETURNS trigger AS $function$
113   BEGIN
114     IF
115       OLD.uh_id IS NOT NULL THEN NEW.uh_id= OLD.uh_id;
116     ELSE
```

```
117         NEW.uh_id :=
118         (SELECT uh_id FROM config_data.mv_configuration_events
119         WHERE sdate < NEW.sdate
120         AND ctr_id = NEW.ctr_id
121         ORDER BY ctr_id, sdate DESC
122         LIMIT 1);
123       END IF;
124     RETURN NEW;
125     END;
126 $function$ LANGUAGE plpgsql;
127
128 DROP TRIGGER IF EXISTS trg_it_mv_config_ev_prv_uh_id
129 ON config_data.mv_configuration_events;
130 CREATE TRIGGER trg_it_mv_config_ev_prv_uh_id
131   AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW -- after insert
132   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_uh_id();
133
134 DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_prv_uh_id
135 ON config_data.mv_configuration_events;
136 CREATE TRIGGER trg_dt_mv_config_ev_prv_uh_id
137   AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW -- after delete
138   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_uh_id();
139
140 DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_prv_uh_id
141 ON config_data.mv_configuration_events;
142 CREATE TRIGGER trg_ut_mv_config_ev_prv_uh_id
143   BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW -- before update
144   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_uh_id();
145
146 -- Select corresponding prselc_id
147 CREATE FUNCTION config_data.trg_mv_config_ev_prv_prselc_id()
148 RETURNS trigger AS $function$
149   BEGIN
150     IF
151       OLD.prselc_id IS NOT NULL THEN NEW.prselc_id= OLD.prselc_id;
152     ELSE
153       NEW.prselc_id :=
154       (SELECT prselc_id FROM config_data.mv_configuration_events
155       WHERE sdate < NEW.sdate
156       AND ctr_id = NEW.ctr_id
157       ORDER BY ctr_id, sdate DESC
158       LIMIT 1);
159     END IF;
160   RETURN NEW;
161   END;
162 $function$ LANGUAGE plpgsql;
163
164 DROP TRIGGER IF EXISTS trg_it_mv_config_ev_prv_prselc_id
165  ON config_data.mv_configuration_events;
166  CREATE TRIGGER trg_it_mv_config_ev_prv_prselc_id
167   AFTER INSERT ON config_data.mv_configuration_events FOR EACH ROW -- after insert
168   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_prselc_id();
169
170 DROP TRIGGER IF EXISTS trg_dt_mv_config_ev_prv_prselc_id
171 ON config_data.mv_configuration_events;
172 CREATE TRIGGER trg_dt_mv_config_ev_prv_prselc_id
173   AFTER DELETE ON config_data.mv_configuration_events FOR EACH ROW -- after delete
174   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_prselc_id();
175
176 DROP TRIGGER IF EXISTS trg_ut_mv_config_ev_prv_prselc_id
177 ON config_data.mv_configuration_events;
178 CREATE TRIGGER trg_ut_mv_config_ev_prv_prselc_id
179   BEFORE UPDATE ON config_data.mv_configuration_events FOR EACH ROW -- before update
180   EXECUTE PROCEDURE config_data.trg_mv_config_ev_prv_prselc_id();
```

5.1.4 Description of Triggers that Execute Refresh of Materialized View Configuration Events

```
-- cabinet triggers
  -- update
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_cabinet_ut()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
    PERFORM config_data.refresh_mv_config_events(ctr_id::SMALLINT)
      FROM config_data.cabinet
      WHERE cabinet.ctr_id = NEW.ctr_id
      AND cabinet.cab_id = NEW.cab_id
      AND cabinet.cab_sdate = NEW.cab_sdate;
  RETURN NULL;
END';
DROP TRIGGER IF EXISTS mv_config_ev_update ON config_data.cabinet;
CREATE TRIGGER mv_config_ev_update
  AFTER UPDATE ON config_data.cabinet
  FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_cabinet_ut();

  -- delet
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_cabinet_dt()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
  PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, cab_sdate::DATE)
    FROM config_data.cabinet
    WHERE cabinet.cab_id = OLD.cab_id
    AND cabinet.cab_sdate = OLD.cab_sdate;
  RETURN NULL;
END';
DROP TRIGGER IF EXISTS mv_config_ev_delete ON config_data.cabinet;
CREATE TRIGGER mv_config_ev_delete
  AFTER DELETE ON config_data.cabinet
  FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_cabinet_dt();

  -- insert
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_cabinet_it()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
  PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, cab_sdate::DATE)
    FROM config_data.cabinet
    WHERE cabinet.cab_id = NEW.cab_id
    AND cabinet.cab_sdate = NEW.cab_sdate;
  RETURN NULL;
END';
DROP TRIGGER IF EXISTS mv_config_ev_insert ON config_data.cabinet;
CREATE TRIGGER mv_config_ev_insert
  AFTER INSERT ON config_data.cabinet
  FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_cabinet_it();

-- lower house triggers
  -- update
CREATE OR REPLACE FUNCTION config_data.mv_config_ev_lower_house_ut()
RETURNS TRIGGER
SECURITY DEFINER
LANGUAGE 'plpgsql' AS '
BEGIN
    PERFORM config_data.refresh_mv_config_events(ctr_id::SMALLINT)
      FROM config_data.lower_house
      WHERE lower_house.ctr_id = NEW.ctr_id
      AND lower_house.lh_id = NEW.lh_id
```

```
65        AND lower_house.lh_sdate = NEW.lh_sdate;
66    RETURN NULL;
67  END';
68  DROP TRIGGER IF EXISTS mv_config_ev_update ON config_data.lower_house;
69  CREATE TRIGGER mv_config_ev_update
70    AFTER UPDATE ON config_data.lower_house
71    FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_lower_house_ut();
72
73    -- delet
74  CREATE OR REPLACE FUNCTION config_data.mv_config_ev_lower_house_dt()
75  RETURNS TRIGGER
76  SECURITY DEFINER
77  LANGUAGE 'plpgsql' AS '
78  BEGIN
79    PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, lh_sdate::DATE)
80      FROM config_data.lower_house
81      WHERE lower_house.lh_id = OLD.lh_id
82      AND lower_house.lh_sdate = OLD.lh_sdate;
83    RETURN NULL;
84  END';
85  DROP TRIGGER IF EXISTS mv_config_ev_delete ON config_data.lower_house;
86  CREATE TRIGGER mv_config_ev_delete
87    AFTER DELETE ON config_data.lower_house
88    FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_lower_house_dt();
89
90    -- insert
91  CREATE OR REPLACE FUNCTION config_data.mv_config_ev_lower_house_it()
92  RETURNS TRIGGER
93  SECURITY DEFINER
94  LANGUAGE 'plpgsql' AS '
95  BEGIN
96    PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, lh_sdate::DATE)
97      FROM config_data.lower_house
98      WHERE lower_house.lh_id = NEW.lh_id
99      AND lower_house.lh_sdate = NEW.lh_sdate;
100   RETURN NULL;
101 END';
102 DROP TRIGGER IF EXISTS mv_config_ev_insert ON config_data.lower_house;
103 CREATE TRIGGER mv_config_ev_insert
104   AFTER INSERT ON config_data.lower_house
105   FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_lower_house_it();
106
107 -- upper house triggers
108   -- update
109 CREATE OR REPLACE FUNCTION config_data.mv_config_ev_upper_house_ut()
110 RETURNS TRIGGER
111 SECURITY DEFINER
112 LANGUAGE 'plpgsql' AS '
113 BEGIN
114     PERFORM config_data.refresh_mv_config_events(ctr_id::SMALLINT)
115       FROM config_data.upper_house
116       WHERE upper_house.ctr_id = NEW.ctr_id
117       AND upper_house.uh_id = NEW.uh_id
118       AND upper_house.uh_sdate = NEW.uh_sdate;
119   RETURN NULL;
120 END';
121 DROP TRIGGER IF EXISTS mv_config_ev_update ON config_data.upper_house;
122 CREATE TRIGGER mv_config_ev_update
123   AFTER UPDATE ON config_data.upper_house
124   FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_upper_house_ut();
125
126   -- delet
127 CREATE OR REPLACE FUNCTION config_data.mv_config_ev_upper_house_dt()
128 RETURNS TRIGGER
129 SECURITY DEFINER
130 LANGUAGE 'plpgsql' AS '
131 BEGIN
132   PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, uh_sdate)
```

```
133       FROM config_data.upper_house
134       WHERE upper_house.uh_id = OLD.uh_id
135       AND upper_house.uh_sdate = OLD.uh_sdate;
136     RETURN NULL;
137   END';
138   DROP TRIGGER IF EXISTS mv_config_ev_delete ON config_data.upper_house;
139   CREATE TRIGGER mv_config_ev_delete
140     AFTER DELETE ON config_data.upper_house
141     FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_upper_house_dt();
142
143     -- insert
144   CREATE OR REPLACE FUNCTION config_data.mv_config_ev_upper_house_it()
145   RETURNS TRIGGER
146   SECURITY DEFINER
147   LANGUAGE 'plpgsql' AS '
148   BEGIN
149     PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, uh_sdate::DATE)
150       FROM config_data.upper_house
151       WHERE upper_house.uh_id = NEW.uh_id
152       AND upper_house.uh_sdate = NEW.uh_sdate;
153     RETURN NULL;
154   END';
155   DROP TRIGGER IF EXISTS mv_config_ev_insert ON config_data.upper_house;
156   CREATE TRIGGER mv_config_ev_insert
157     AFTER INSERT ON config_data.upper_house
158     FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_upper_house_it();
159
160   -- presidential election triggers
161       -- update
162   CREATE OR REPLACE FUNCTION config_data.mv_config_ev_presidential_election_ut()
163   RETURNS TRIGGER
164   SECURITY DEFINER
165   LANGUAGE 'plpgsql' AS '
166   BEGIN
167       PERFORM config_data.refresh_mv_config_events(ctr_id::SMALLINT)
168         FROM config_data.presidential_election
169         WHERE presidential_election.ctr_id = NEW.ctr_id
170         AND presidential_election.prselc_id = NEW.prselc_id
171         AND presidential_election.prselc_sdate = NEW.prselc_sdate;
172     RETURN NULL;
173     END';
174   DROP TRIGGER IF EXISTS mv_config_ev_update ON config_data.presidential_election;
175   CREATE TRIGGER mv_config_ev_update
176     AFTER UPDATE ON config_data.presidential_election
177     FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_presidential_election_ut();
178
179     -- delet
180   CREATE OR REPLACE FUNCTION config_data.mv_config_ev_presidential_election_dt()
181   RETURNS TRIGGER
182   SECURITY DEFINER
183   LANGUAGE 'plpgsql' AS '
184   BEGIN
185     PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, prselc_date::DATE)
186       FROM config_data.presidential_election
187       WHERE presidential_election.prselc_id = OLD.prselc_id
188       AND presidential_election.prselc_date = OLD.prselc_date;
189     RETURN NULL;
190   END';
191   DROP TRIGGER IF EXISTS mv_config_ev_delete ON config_data.presidential_election;
192   CREATE TRIGGER mv_config_ev_delete
193     AFTER DELETE ON config_data.presidential_election
194     FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_presidential_election_dt();
195
196     -- insert
197   CREATE OR REPLACE FUNCTION config_data.mv_config_ev_presidential_election_it()
198   RETURNS TRIGGER
199   SECURITY DEFINER
200   LANGUAGE 'plpgsql' AS '
```

```
201 BEGIN
202   PERFORM config_data.mv_config_ev_refresh_row(ctr_id::SMALLINT, prselc_date::DATE)
203     FROM config_data.presidential_election
204     WHERE presidential_election.prselc_id = NEW.prselc_id
205     AND presidential_election.prselc_date = NEW.prselc_date;
206   RETURN NULL;
207 END';
208 DROP TRIGGER IF EXISTS mv_config_ev_insert ON config_data.presidential_election;
209 CREATE TRIGGER mv_config_ev_insert
210   AFTER INSERT ON config_data.presidential_election
211   FOR EACH ROW EXECUTE PROCEDURE config_data.mv_config_ev_presidential_election_it();
```

5.2 Overview of variables in Tables

Table 5.1: Variables in Country Table

Variable	*Description*	*Format*
ctr_id	Country identifier	Integer
ctr_n	Country name	Character
ctr_ccode	ISO3 country code[1]	Character
ctr_ccode2	ISO2 country code[1]	Character
ctr_ccode_nr	ISO3 country code[1]	Numeric
ctr_eu_date	Date of EU accession[2]	YYYY-MM-DD
ctr_oecd_date	Date of OECD accession[3]	YYYY-MM-DD
ctr_wto_date	Date of WTO accession[4]	YYYY-MM-DD
ctr_cmt	Comments	Text
ctr_src	Data sources	Text

[1] ISO (2015), http://www.iso.org/iso/home/standards/country_codes.htm
[2] EU (2015), http://europa.eu/about-eu/countries/member-countries/index_en.htm
[3] OECD (2015), http://www.oecd.org/about/membersandpartners/list-oecd-member-countries.htm
[4] WTO (2015), https://www.wto.org/english/thewto_e/whatis_e/tif_e/org6_e.htm

Table 5.2: Variables in Cabinet Table

Variable	***Description***	***Format***
cab_id	Cabinet identifier	Numeric(5,0)
cab_prv_id	Cabinet identifier of the previous cabinet	Numeric(5,0)
ctr_id	Country identifier	Integer
cab_sdate	Cabinet start date	YYYY-MM-DD
cab_hog_n	Name of the Head of Government	Character
cab_sts_ttl	Total number of cabinet portfolios	Numeric
cab_care	Indicates if cabinet is a caretaker cabinet	Boolean
cab_cmt	Comments	Text
cab_src	Data sources	Text
cab_valid_sdate	Indicates whether cabinet start date has been double-checked	Boolean

Table 5.3: Variables in Cabinet Portfolios Table

Variable	***Description***	***Format***
ptf_id	Portfolio identifier	Numeric(5,0)
cab_id	Cabinet identifier	Numeric(5,0)
pty_id	Party identifier	Numeric(5,0)
pty_cab	Indicates if party is in cabinet	Boolean
pty_cab_sts	A party's number of portfolios/ministries in a cabinet	Numeric
pty_cab_hog	Indicates if party fills the position of the Head of Government	Boolean
pty_cab_sup	Indicates if party is supporting the cabinet but is not part of it	Boolean
ptf_cmt	Comments	Text
ptf_src	Data sources	Text

Table 5.4: Variables in Lower House Table

Variable	*Description*	*Format*
lh_id	Lower house identifier	Numeric(5,0)
lh_prv_id	Identifier of the previous lower house	Numeric(5,0)
lh_nxt_id	Identifier of the next lower house	Numeric(5,0)
lhelc_id	Lower house election identifier	Numeric(5,0)
ctr_id	Country identifier	Integer
lh_sdate	Lower house start date	YYYY-MM-DD
lh_sts_ttl	Total number of seats in lower house	Numeric
lh_enpp	Effective number of parties in parliament[5]	Numeric
lh_cmt	Comments	Text
lh_src	Sources of information on lower house	Text
pty_lh_rght	Indicates whether there was a right-winged party in the lower house[6]	Boolean
lh_valid_sdate	Indicates whether lower house start date has been double-checked	Boolean

[5] Recorded figures only; computed as proposed by Laakso and Taagepera (1979).
[6] Abou-Chadi (2014)

Table 5.5: Variables in Lower House Election Table

Variable	*Description*	*Format*
lhelc_id	Lower house election identifier	Numeric(5,0)
lhelc_prv_id	Previous lower house election identifier	Numeric(5,0)
ctr_id	Country identifier	Integer
lhelc_date	Lower house election date	YYYY-MM-DD
lhelc_early	Indicates an early election	Boolean
lhelc_reg_vts	Number of registered voters	Numeric
lhelc_reg_vts_pr	Number of registered voters, PR system	Numeric
lhelc_reg_vts_pl	Number of registered voters, plurality system	Numeric
lhelc_vts_pr	Valid votes for lower house elected with proportional representation system	Numeric
lhelc_vts_pl	Valid votes for lower house elected with plurality system	Numeric
lhelc_sts_pr	Number of lower house seats elected with proportional representation system	Numeric
lhelc_sts_pl	Number of lower house seats elected with plurality system	Numeric
lhelc_sts_ttl	Total number of lower house seats elected in the election	Numeric
lhelc_fml_t1	Electoral formula used for allocation of lower house seats on the first tier	Character
lhelc_ncst_t1	Number of lower house constituencies at the first tier	Numeric
lhelc_sts_t1	Number of lower house seats allocated at the first tier	Numeric
lhelc_dstr_mag	Mean average lower house district magnitude[7]	Numeric

continued on next page ...

[7] Data obtained from Carey and Hix (2011).

Table 5.5: *... continued*

Variable	***Description***	***Format***
lhelc_dstr_mag_med	Median average lower house district magnitude[8]	Numeric
lhelc_ mag_t1	Average lower house district magnitude on first tier	Numeric
lhelc_ntrsh _t1	National threshold for lower house on the first tier	Numeric
lhelc_dtrsh_t1	District threshold for lower house on first tier	Numeric
lhelc_fml_t2	Electoral formula used for allocation of lower house seats on the second tier	Character
lhelc_ncst_t2	Number of lower house constituencies at the second tier	Numeric
lhelc_sts_t2	Number of lower house seats allocated at the second tier	Numeric
lhelc_ mag_t2	Average lower house district magnitude on second tier	Numeric
lhelc_ntrsh _t2	National threshold for lower house on the second tier	Numeric
lhelc_dtrsh_t2	District threshold for lower house on second tier	Numeric
lhelc_fml_t3	Electoral formula used for allocation of lower house seats on the third tier	Character
lhelc_ncst_t3	Number of lower house constituencies at the third tier	Numeric
lhelc_sts_t3	Number of lower house seats allocated at the third tier	Numeric
lhelc_ mag_t3	Average lower house district magnitude on third tier	Numeric
lhelc_ntrsh _t3	national threshold for lower house on the third tier	Numeric

continued on next page ...

[8] Data and definition provided byCarey and Hix (2008).

Table 5.5: *... continued*

Variable	*Description*	*Format*
lhelc_dtrsh_t3	District threshold for lower house on third tier	Numeric
lhelc_fml_t4	Electoral formula used for allocation of lower house seats on the fourth tier	Character
lhelc_ncst_t4	Number of lower house constituencies at the fourth tier	Numeric
lhelc_sts_t4	Number of lower house seats allocated at the fourth tier	Numeric
lhelc_mag_t4	Average lower house district magnitude on fourth tier	Numeric
lhelc_ntrsh_t4	National threshold for lower house on the fourth tier	Numeric
lhelc_dtrsh_t4	District threshold for lower house on fourth tier	Numeric
lhelc_bon_sts	Majority seat bonus	Numeric
lhelc_esys_cmt	Comment on electoral system	Text
lhelc_cmt	Comments on lower house elections	Text
lhelc_esys_src	Source of inforamtion on electoral system	Text
lhelc_lsq	Gallagher's Least-square index (LSq) of disproportionality[9]	Numeric
lhelc_vola_sts	Seat A volatility[10]	Numeric
lhelc_volb_sts	Seat B volatility[11]	Numeric
lhelc_vola_vts	Vote A volatility[10]	Numeric
lhelc_volb_vts	Vote B volatility[11]	Numeric
lhelc_src	Sources of information on lower house elections	Text

continued on next page ...

[9] Gallagher (1991, 1992)
[10] Volatility arising from new entering and retiering parties, respectively (Powell and Tucker, 2013).
[11] Volatility arising from gaines and losses of stable parties (Powell and Tucker, 2013).

Table 5.5: *... continued*

Variable	***Description***	***Format***
lhelc_valid_date	Indicates whether lower house election date has been double-checked	Boolean

Table 5.6: Variables in Lower House Vote Results Table

Variable	***Description***	***Format***
lhvres_id	Lower house vote result identifier	Numeric(5,0)
lhelc_id	Lower house election identifier	Numeric(5,0)
pty_id	Party identifier	Numeric(5,0)
pty_lh_vts_pr	A party's valid votes in lower house elected with proportional representation system	Numeric
pty_lh_vts_pl	A party's valid votes in lower house elected with plurality system	Numeric
lhvres_cmt	Comments	Text
lhvres_src	Sources of information on lower house vote results	Text

Table 5.7: Variables in Lower House Seat Results Table

Variable	*Description*	*Format*
lhsres_id	Lower house seats results identifier	Numeric(5,0)
lh_id	Lower house identifier	Numeric(5,0)
pty_id	Party identifier	Numeric(5,0)
pty_lh_sts_pr	A party's number of seats in lower house elected with proportional representation system	Numeric
pty_lh_sts_pl	A party's number of seats in lower house elected with plurality system	Numeric
pty_lh_sts	A party's total number of seats in lower house	Numeric
lhsres_cmt	Comments	Text
lhsres_src	Sources of information on lower house seat results	Text

Table 5.8: Variables in Upper House Table

Variable	*Description*	*Format*
uh_id	Upper house identifier	Numeric(5,0)
uh_prv_id	Identifier of previous upper house	Numeric(5,0)
uhelc_id	Upper house election identifier	Numeric(5,0)
ctr_id	Country identifier	Integer
uh_sdate	Upper house start date	YYYY-MM-DD
uh_sts_ttl	Total number of seats in the upper house	Numeric
uh_cmt	Comments	Text
uh_src	Sources of information on upper house	Text
uh_valid_sdate	Indicates whether upper house start date has been double-checked	Boolean

Table 5.9: Variables in Upper House Election Table

Variable	*Description*	*Format*
uhelc_id	Upper house election identifier	Numeric(5,0)
uhelc_prv_id	Previous upper house election identifier	Numeric(5,0)
ctr_id	Country identifier	Integer
uhelc_date	Upper house election date	YYYY-MM-DD
uh_sts_ttl	Total number of seats	Numeric
uhelc_sts_elc	Total number of seats elected in the election	Numeric
uhelc_cmt	Comments	Text
uhelc_src	Sources of information on upper house election	Text
uhelc_valid_date	Indicates whether upper house election date has been double-checked	Boolean

Table 5.10: Variables in Upper House Seat Results Table

Variable	*Description*	*Format*
uhsres_id	Upper house seats result identifier	Numeric(5,0)
uh_id	Upper house identifier	Numeric(5,0)
pty_id	Party identifier	Numeric(5,0)
pty_uh_sts_elc	A party's number of seats in upper house gained through election	Numeric
pty_uh_sts	A party's total number of seats in upper house (including seats allocated through appointment)	Numeric
uhsres_cmt	Comments	Text
uhsres_src	Sources of information on upper house seats results	Text

Table 5.11: Variables in Presidential Election Table

Variable	*Description*	*Format*
prselc_id	Presidential election identifier	Numeric(5,0)
prselc_prv_id	Previous presidential election identifier	Numeric(5,0)
ctr_id	Country identifier	Integer
prselc_date	Presidential election date	YYYY-MM-DD
prselc_rnd_ttl	Number of rounds in the presidential election	Integer
prselc_vts_clg	Number of total votes in electoral college	Numeric
reg_vts_prselc_r1	Registered voters for presidential elections first round	Numeric
reg_vts_prselc_r2	Registeres voters for presidential elections second round	Numeric
prselc_vts_ppl_r1	Number of total valid votes in presidential election in round 1	Numeric

continued on next page ...

Table 5.11: *... continued*

Variable	***Description***	***Format***
prselc_vts_ppl_r2	Number of total valid votes in presidential election in round 2	Numeric(5,0)
prselc_clg	Indicates if president is elected through an electoral college (coded 1 if yes, 0 if no)	Boolean
prs_n	Name of president	Name
pty_prs	Party identifier of President's party	Numeric(5,0)
prs_sdate	Start date of presidency	YYYY-MM-DD
prselc_cmt	Comments	Text
prselc_src	Sources of information on presidential election	Text
prselc_valid_date	Indicates whether Presidency start date has been double-checked	Boolean
prs_valid_sdate	Indicates whether Presidential election date has been double-checked	Boolean

Table 5.12: Variables in Presidential Election Vote Results Table

Variable	***Description***	***Format***
prsvres_id	Presidential election vote results identifier	Numeric(5,0)
prselc_id	Presidential election identifier	Numeric(5,0)
prselc_rnd	Enumerates the round of a presidential election	Integer
prs_cnd_pty	Party identifier of candidate's party	Numeric(5,0)
prs_cnd_n	Name of candidate	Name
prs_cnd_vts_clg	Number of electoral college votes for candidate	Numeric
prs_cnd_vts_ppl	Number of popular votes for candidate	Numeric
prsvres_cmt	Comments	Text
prsvres_src	Sources of information on presidential election vote results	Text

Table 5.13: Variables in Veto Points Table

Variable	***Description***	***Format***
vto_id	Veto point identifier	Numeric(5,0)
ctr_id	Country identifier	Integer
vto_inst_typ	One of the following types of veto institutions: 1. Head of State 2. Head of Government 3. Lower House 4. Upper House 5. Judicial 6. Electoral 7. Territorial	Character
vto_inst_n	Original name of institution	Character
vto_inst_n_en	Name of institution in English	Character
vto_inst_sdate	Date since which this institution exists[12]	YYYY-MM-DD
vto_inst_edate	Date on which the institution was abolished[13]	YYYY-MM-DD
vto_pwr	Instituional veto potential	Numeric
vto_cmt	Comments	Text
vto_src	Data sources	Text

[12] Coded 1900-01-01 if institutionalized before time period covered by PCDB
[13] Coded 2099-12-31 if still existent at the end of time period covered by PCDB

Table 5.14: Variables in Party Table

Variable	*Description*	*Format*
pty_id	Party identifier	Numeric(5,0)
pty_abr	Abbreviation of party name	Character
pty_n	Full party name in country's official language	Character
pty_n_en	Full party name in English	Character
cmp_id	Party identifier in Manifesto Project Database[14]	Numeric(6,0)
prlgv_id	Party identifier in Parlgov database[15]	Integer
pty_eal	Indicates the number of parties participating in an electoral alliance	Integer
pty_eal_id	Lists party IDs of parties participating in an alliance	Text
ctr_id	Country identifier	Integer
clea_id	Party identifier in Constituency-Level Elections Archive (CLEA)[16]	Character
pty_cmt	Comments	Text
pty_src	Sources of information on party	Text

[14] Volkens et al. (2013)
[15] Döring and Manow (2012)
[16] Kollman et al. (2014)

Table 5.15: Variables in Electoral Alliances Table

Variable	***Description***	***Format***
ctr_id	Country identifier	Integer
pty_id	Party identifier	Numeric(5,0)
pty_abr	Party abbrevation	Character
pty_eal _nbr	Indicates the number of parties participating in an electoral alliance	Integer
pty_eal_id	Electoral alliance party identifier	Numeric(5,0)
pty_eal_cmt	Comment	Text
pty_eal_src	Source of inforamtion on party's participation in electoral alliance	Text