

Time Related Range Types Revisited

Real World use cases from the KOF and SwissPUG daily business

Charles Clavadetscher

Swiss PostgreSQL Users Group

PGConf Europe 2018, 24.10.2018, Lisboa, Portugal

Outline

- 1 Introduction
- 2 Range Types
- 3 Use Cases
 - Manage Questionnaires Versions
 - History of Survey Participants
 - Dynamic Agenda Display
 - Publication Of Indicators
- 4 Conclusion

Introduction

Myself And The Company I work for

- Senior DB Engineer at KOF ETH Zurich
 - KOF is the Center of Economic Research of the
 - ETHZ the Swiss Institute of Technology in Zurich, Switzerland
 - Independent economic research on business cycle tendencies for almost all sectors
 - Maintenance of all databases at KOF: PostgreSQL, Oracle, MySQL and MSSQL Server. Focus on migrating to PostgreSQL
 - Support in business process re-engineering
- Co-founder and treasurer of the SwissPUG, the Swiss PostgreSQL Users Group
- Member of the board of the Swiss PGDay

Outline

1 Introduction

2 Range Types

3 Use Cases

- Manage Questionnaires Versions
- History of Survey Participants
- Dynamic Agenda Display
- Publication Of Indicators

4 Conclusion

Range Types

Characteristics

PostgreSQL has support for ranges of various data types. The common denominator is that the base data type has a clear and unique natural order and that the values being part of a range have no missing values. Natural candidates are date or time related types and numbers as well as its derivatives. Specifically PostgreSQL defines following range types:

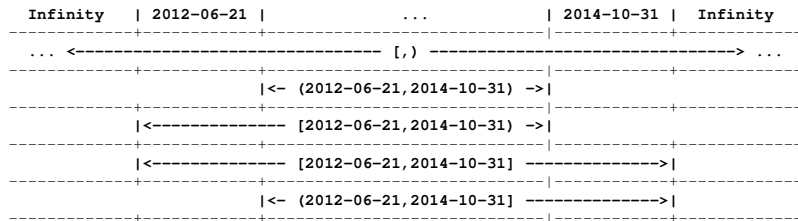
- `INT4RANGE`: Range of integer
- `INT8RANGE`: Range of bigint
- `NUMRANGE`: Range of numeric
- `TSRANGE`: Range of timestamp without time zone
- `TSTZRANGE`: Range of timestamp with time zone
- `DATERANGE`: Range of date

In this presentation we focus on the last two: `DATERANGE` and `TSTZRANGE`.

Range Types

Representation

Typically ranges are represented as a string containing the bounds and an indications if those bounds are included in the range or not. Example using `DATERANGE`



Analogue to the first example, you may have one of the bounds empty to create a range with only one open end.

Create Ranges 1/2

To create a range type you may type cast a text representation of it to the corresponding type or use one of the built in functions. In both cases the type creation will make sure that your entry is valid.

```
db=> SELECT '[2012-06-21,2014-10-31]'::DATERANGE;  
[2012-06-21,2014-10-31)
```

```
db=> SELECT daterange('2012-06-21','2014-10-31','[]');  
[2012-06-21,2014-10-31)
```

```
db=> SELECT '[2015-06-21,2014-10-31]'::DATERANGE;  
ERROR:  range lower bound must be less than or equal to range upper bound
```

In the documentation you will find the list of the range type creation functions. General format:

```
name_of_range_type(lower_bound,upper_bound,bounds);
```

lower and upper bound can be NULL for open ends. Bounds is a 2 char string containig the representation of the bounds. If not provided the default is "[)".

Create Ranges 2/2

A recommendation: Get used to operate with the default representation of ranges, unless you have a very good reason to do it differently. Consider the following:

```
db=> SELECT daterange('2012-06-21','2014-10-30','()') AS from_function,
           ' (2012-06-21,2014-10-30]'::DATERANGE AS from_typecast;
-[ RECORD 1 ]-+-----
from_function | [2012-06-22,2014-10-31)
from_typecast | [2012-06-22,2014-10-31)
```

The range creation process returns the default representation if the values are scaled ordinally.

```
db=> CREATE TABLE test (dr DATERANGE);
db=> INSERT INTO test VALUES (' (2012-06-21,2014-10-30]'::DATERANGE);
db=> SELECT * FROM test;
-[ RECORD 1 ]-----
dr | [2012-06-22,2014-10-31)
```

This can be confusing when you start retrieving the bounding values for whatever operation that you want to perform.

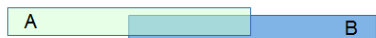
Operators 1/2

At the time of this writing (Version 10.5) range types are supplied with a set of 19 operators. From the official documentation:

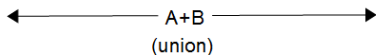
- The simple comparison operators `<`, `>`, `<=`, and `>=` compare the lower bounds first, and only if those are equal, compare the upper bounds. **These comparisons are not usually very useful for ranges, but are provided to allow B-tree indexes to be constructed on ranges.**
- The left-of/right-of/adjacent operators always return false when an empty range is involved; that is, **an empty range is not considered to be either before or after any other range.**
- The union and difference operators **will fail if the resulting range would need to contain two disjoint sub-ranges**, as such a range cannot be represented.

Operators 2/2

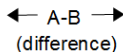
Let's focus on some range specific operators...



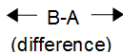
$\leftarrow A * B \rightarrow$ (intersection)



$A + B$
(union)

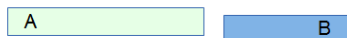


$A - B$
(difference)



$B - A$
(difference)

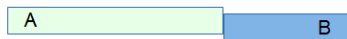
$A \&\& B = \text{true}$ (overlaps)



$A \ll B = \text{true}$ (A is strictly left of B)



$A @> B = \text{true}$ (A contains B)

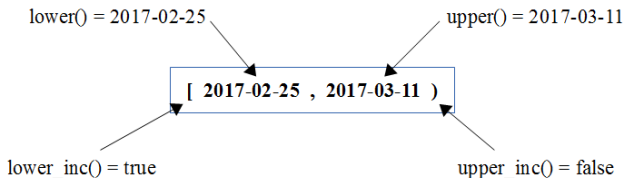


$A -|- B = \text{true}$ (A is adjacent to B)

The containment operator can also be used to check if a single element is included in a range.

Functions

The built in functions for range types are helpful to request specific information on the type's characteristics.



- `lower_inf(anyrange), upper_inf(anyrange)`: Return a boolean indicating if the requested bound is open ended.
- `isempty(anyrange)`: Return a boolean indicating if the range is empty. Notice that empty is not the same as NULL. In term of DATERANGE, for example, empty means as much as "never".
- `range_merge(anyrange, anyrange)`: The smallest range containing both range arguments. The difference to the union operator (+) it that the range parameters don't need to be at least contiguous.

Outline

1 Introduction

2 Range Types

3 Use Cases

- Manage Questionnaires Versions
- History of Survey Participants
- Dynamic Agenda Display
- Publication Of Indicators

4 Conclusion

Outline

- 1 Introduction
- 2 Range Types
- 3 Use Cases
 - Manage Questionnaires Versions
 - History of Survey Participants
 - Dynamic Agenda Display
 - Publication Of Indicators
- 4 Conclusion

Manage Questionnaires Versions

A KOF Questionnaire

ETH zürich KOF

KOF Inchieste

Inchiesta congiunturale Commercio al dettaglio 1.2017

DB Consulting Test (486992)

472 | Cd. di prodotti alim. bev. e tabacco

[Inchieste](#) [Informazioni](#) [Contatto](#) [Logout](#)

Domande mensili

Escludere le fluttuazioni stagionali

Nel corso dei prossimi tre mesi la cifra d'affari

Alimentari, bevande, tabacco e articoli per fumatori

- ☐ aumenterà
- ☐ rimarrà uguale
- ☐ diminuirà

Abbigliamento, calzature

- ☐ aumenterà

Manage Questionnaires Versions

... is a JSON document

```
db=> SELECT * FROM operations.get_form_by_language('DHU','it',0);
{
  "version": "1.0",
  "survey": "DHU",
  "items": [{
    "meta": {
      "frequency": "month"
    },
    "title": {
      "it": "Domande mensili"
    },
    "description": {
      "it": "<em>Escludere le fluttuazioni stagionali</em>"
    },
    "items": [{
      "description": {
        "it": "Nel corso dei prossimi tre mesi la cifra d'affari"
      },
      "items": [{
        "id": "q_ql_exp_turnover_food_n3m",
        "type": "single_choice",
        "question": {
          "it": "Alimentari, bevande, tabacco e articoli per fumatori"
        },
        "answers": {
          "values": [{
            [...]
```

Manage Questionnaires Versions

... with a History

```
db=> \d web_form_templates
```

```
Table "operations.web_form_templates"
```

Column	Type	Modifiers
survey_name	text	
survey_type	integer	
form_json	json	
validity	daterange	

```
Indexes:
```

```
"web_form_templates_survey_name_type_validity_excl"
```

```
EXCLUDE USING gist (survey_name WITH =, survey_type WITH =, validity WITH &&)
DEFERRABLE
```

```
db=> SELECT survey_name, survey_type, validity
FROM operations.web_form_templates
WHERE (survey_name,survey_type) = ('DHU',0)
ORDER BY validity;
```

survey_name	survey_type	validity
DHU	0	[2011-01-01,2015-02-01)
DHU	0	[2015-02-01,)

(2 rows)

Manage Questionnaires Versions

And some fancy utilities

```
db=> SELECT question_text, answers_codes
      FROM operations.get_questions_from_form('DHU','it',0,'2012-05-25');
```

question_text	answers_codes
---------------	---------------

question_text	answers_codes
Alimentari	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Bevande	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Tabacco e articoli per fumatori	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Abbigliamento, calzature	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Carburanti	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Altri gruppi di merci del commerc[...]	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
[...]	

```
db=> SELECT question_text, answers_codes
      FROM operations.get_questions_from_form('DHU','it',0,'2017-01-01');
```

question_text	answers_codes
---------------	---------------

question_text	answers_codes
Alimentari, bevande, tabacco e ar[...]	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Abbigliamento, calzature	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Carburanti	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
Altri gruppi di merci del commerci[...]	1: aumenterà, 0: rimarrà uguale, -1: diminuirà
[...]	

Outline

- 1 Introduction
- 2 Range Types
- 3 Use Cases
 - Manage Questionnaires Versions
 - **History of Survey Participants**
 - Dynamic Agenda Display
 - Publication Of Indicators
- 4 Conclusion

History of Survey Participants

Rationale

The entries in table `survey_participants` are used to keep track of the information that is required for

- Creating and sending questionnaires or mail invitations (survey name, contact information).
- Sectors and weights for the calculation of macro economic indicators (economic sector, weighting information).

By its nature, contact information is relevant at the moment when mails and questionnaires are sent out. When it comes to the information used for computation, it is clear that we must keep the information that was valid at the time the survey was conducted. We need to know, e.g. how big the company was for any specific day that it delivered survey data.

History of Survey Participants

Short digression: The difference between audit and history (in this context)

- When you are investigating problems in a database (or mostly in the application built on top of it), you usually want to know exactly who did what and when. In such cases every single change may be relevant to your analysis. This is what we call an audit, i.e. an exact record of all changes in one or more tables.
- The KOF surveys are conducted every month and always start at the beginning of a month. If the information about a company is modified many times during a month, e.g. because of mistyped information or follow ups from the company itself, it is not relevant for the survey itself and its analysis. What is important is the information that was valid at the moment when the company was called to attend the survey. To keep record of this specific information is what we call the history of the company.
- In short, an audit would record all changes made to a record, while an history keeps track of changes that have an impact on the processes they support. Notice that the history does not even need to be a copy of an existing table. Finally what needs to be kept in the history, again, depends on the processes. An address is relevant at the time of sending out a Questionnaire or an E-Mail. It is not required (at least for us) to keep track of these changes. On the contrary, data used to weigh responses must match the values valid at the time when these responses were delivered.

History of Survey Participants

Implementation

```
FROM operations.company_weights:
```

survey_name	company_id	contact_id	sector_class	sector_code	fte
BAU	550611	486992	NA608	4213	9

(1 row)

```
FROM operations.company_weights_history:
```

```
-[ RECORD 1 ]+-----
survey_name | BAU
company_id  | 550611
contact_id  | 486992
sector_class| NA608
sector_code | 4213
fte         | 5
validity    | [2012-03-01,2016-06-01)
-[ RECORD 2 ]+-----
survey_name | BAU
company_id  | 550611
contact_id  | 486992
sector_class| NA608
sector_code | 4213
fte         | 9
validity    | [2016-06-01,)
```

Outline

- 1 Introduction
- 2 Range Types
- 3 Use Cases
 - Manage Questionnaires Versions
 - History of Survey Participants
 - **Dynamic Agenda Display**
 - Publication Of Indicators
- 4 Conclusion

Dynamic Agenda Display

The image displays two overlapping browser windows showing the Swiss PostgreSQL Users Group (SwissPUG) website. The top window shows the 'Discussion' page, and the bottom window shows the 'Agenda' page.

Top Window (Discussion Page):

- URL: <https://www.swisspug.org/wiki/>
- Page: Discussion
- Search:
- Log in

Bottom Window (Agenda Page):

- URL: <https://www.swisspug.org/wiki/index.php/Agenda>
- Page: Agenda
- Search:
- Log in

Agenda Contents:

1 Current Events

- 1.1 Presentation: PostgreSQL und MediaWiki - Dokumentieren mit Minimalaufwand
- 1.2 Assembly: General Assembly 2017
- 1.3 Conference: Swiss PGDay 2017
- 1.4 External: Events published on PostgreSQL.org

2 Past Events

- 2.1 Presentation: PostgreSQL aus Sysadmin-Sicht
- 2.2 Presentation: Datawarehouse und Open Source

Next Scheduled Event:

PostgreSQL und MediaWiki - Dokumentieren mit Minimalaufwand

Donnerstag, 19.01.2017 19:00
 Punto d'incontro, Josefstrasse 102, 8005 Zürich
 (Karte [↗](#))

Dokumentieren ist langwierig, langweilig, mühsam. Jedoch ist die Dokumentation der Objekte in der Datenbank, ob man es wahrhaben will oder nicht, eine wichtige Zusatzaufgabe von DBAs und DB Entwicklern. [\(read more\)](#)

Consider subscribing to our [discussion](#) and announce [mailing lists](#).

Dynamic Agenda Display

Agenda - Swiss PostgreSQL Management

https://www.swisspug.org/operation

Most Visited Getting Started ETHZ Services PostgreSQL SwissPUG

Members Board Events Scheduler

SwissPUG Management Events

Current date: 2017-01-03 - Logged in as: swisspug - [Logout](#)

current Change scope

New event

ID	Speaker/Moderator	Title	Date	Published
12	[[User:Charles Charles Clavadetscher]]	PostgreSQL und MediaWiki - Dokumentieren mit Minimalaufwand	2017-01-19	x
21	[[User:Markus Markus Wanner]] (SwissPUG President)	General Assembly 2017	2017-04-27	x
20		Swiss PGDay 2017	2017-06-30	x

Message: No messages

Agenda - Swiss PostgreSQL Management

https://www.swisspug.org/operation

Most Visited Getting Started ETHZ Services PostgreSQL SwissPUG Charles Surveyweb KOF Surveys (old)

Back to list Insert

Event information

Event ID	
Event type*	Presentation
Moderator	
Speaker	Pinco Pallino
Title*	How To Use Dateranges For a Dynamic Calendar
Date display*	Saturday, January 7th, 2017
Location	Plaza de la Revolución, La Habana, Cuba
Language	English / Español
Fees	Free
Level	
Registration	
Link(s)	
Abstract	This is a fake event to showcase the usage of date range types in the automatic display of an online agenda.
Event duration*	2017-01-07 Start 2017-01-07 End
Event publication	2017-01-03 Start End

Message: No messages

Dynamic Agenda Display

The top screenshot shows the main page of the Swiss PostgreSQL Users Group (SwissPUG) website. The title is "Swiss PostgreSQL Users Group". The navigation bar includes links for "Most Visited", "Getting Started", "ETHZ", "Services", "PostgreSQL", "SwissPUG", "Charles", "Surveyweb", and "KOF Surveys (old)". The page content includes a welcome message and a list of upcoming events.

The bottom screenshot shows the "Agenda" page. The title is "Agenda". The page content includes a list of upcoming events, a "Contents" table of contents, and a "Next Scheduled Event" section. The sidebar on the left contains links for "Welcome", "Agenda", "Bylaws", "Board", "Members", "Contact", "Related Organizations", "Apply for Membership", "Additional", "Promotion", "Recent changes", "Random page", "Help", "Tools", "What links here", "Related changes", and "Upload file".

Dynamic Agenda Display

The entry in the DB

```

swisspug@swisspug=> SELECT * FROM events WHERE event_id = 22;
-[ RECORD 1 ]+-----
event_id      | 22
event_type    | Presentation
moderator     |
speaker       | Pinco Pallino
title         | How To Use Dateranges For a Dynamic Calendar
date_display  | Saturday, January 7th, 2017
location      |
language      |
fees          |
level         |
registration  |
links         |
abstract      | This is a fake event to showcase the usage of date range types
               | in the automatic display of an online agenda.
publication | [2017-01-03,)
duration   | [2017-01-07,2017-01-08)

```

Dynamic Agenda Display

The display function code

```
CREATE OR REPLACE FUNCTION public.mw_next_event()
RETURNS TEXT
AS $$
DECLARE
    v_event operations.events;
    v_wiki_text TEXT := '';
BEGIN
    SELECT * FROM operations.events
    INTO v_event
    WHERE CURRENT_DATE <@ publication      -- is open for publication
    AND CURRENT_DATE <= upper(duration)    -- has not finished yet
    ORDER BY lower(duration) ASC           -- put the newest on top of list
    LIMIT 1;                               -- take only the most recent
    IF v_event.event_id IS NOT NULL THEN
        -- Format wiki text for output.
    END IF;
    RETURN v_wiki_text;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER;
```

Dynamic Agenda Display

The display function in action

```
swisspug@swisspug=> SELECT * FROM public.mw_next_event();

{| class="wikitable" style="float: right; clear: right; font-size: 88%; width: 300px"
|-
! Next Scheduled Event
|- style="text-align: center"
| <h3>[[Agenda#Presentation: How To Use Dateranges For a Dynamic Calendar|
   How To Use Dateranges For a Dynamic Calendar]]</h3><br/>
   Saturday, January 7th, 2017<br/>
   <br/><br/>This is a fake event to showcase the usage of date range types
   in the automatic display of an online agenda.
|}
```

Outline

- 1 Introduction
- 2 Range Types
- 3 Use Cases
 - Manage Questionnaires Versions
 - History of Survey Participants
 - Dynamic Agenda Display
 - Publication Of Indicators
- 4 Conclusion

Publication Of Indicators

The KOF Barometer

[Institut](#)
[Forschung](#)
[Umfragen](#)
[Prognosen & Indikatoren](#)
[Publikationen](#)
[Daten & Be](#)

Medienagenda

Datum	Zeit	Thema
20. Dez. 2016	09:00	KOF Consensus Forecast
23. Dez. 2016	09:00	KOF Konjunkturbarometer
19. Jan. 2017	17:30	KOF Monetary Policy Communicator (MPC)
30. Jan. 2017	09:00	KOF Konjunkturbarometer
06. Febr. 2017	09:00	KOF Beschäftigungsindikator

[News & Veranstaltungen](#)
[Das Institut](#)
[Forschung](#)
[Umfragen](#)
[Prognosen & Indikatoren](#)
[Publikationen](#)

ETH Zürich → D-MTEC → KOF →

Prognosen

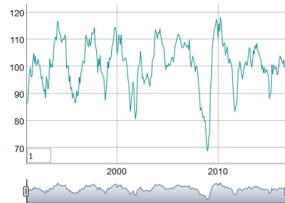
Indikatoren

KOF Konjunkturbarometer

KOF Beschäftigungsindikator
 KOF Monetary Policy Communicator
 KOF Geschäftslageindikator
 KOF Globalisierungsindex
 KOF Surprise Indicator
 KOF Baubiatt Indikator
 KOF Jugendarbeitsmarkindex
 KOF Unsicherheitsindikatoren

KOF Konjunkturbarometer

— Konjunkturbarometer



Publication Of Indicators

Requirements on Publication

- Publication date and time are published on the KOF website.
- Everybody should get access to the data at the same point in time.

Publication Of Indicators

The timeseries data table

```

Table "public.timeseries_main"
  Column      |      Type      | Modifiers
-----+-----+-----
ts_key        | character varying | not null
ts_data       | hstore          |
ts_frequency  | integer         |
validity     | tstzrange      | not null

Indexes:
    "timeseries_main_pkey" PRIMARY KEY, btree (ts_key, validity)
    "timeseries_main_ts_key_validity_excl"
        EXCLUDE USING gist (ts_key WITH =, validity WITH &&)

charles@charles=# SELECT * FROM public.timeseries_main WHERE ts_key = 'kofbarometer';
-[ RECORD 1 ]+-----
ts_key        | kofbarometer
ts_data       | "2016-11-01"=>"102.162463328593", "2016-12-01"=>"102.161371136933"
ts_frequency  | 12
validity     | ["2016-12-01 09:00:00+01",)

charles@charles=# SELECT (each(ts_data)).* FROM public.timeseries_main
                    WHERE ts_key = 'kofbarometer';
   key   |      value
-----+-----
2016-11-01 | 102.162463328593
2016-12-01 | 102.161371136933

```


Publication Of Indicators

The Publication Process 1/2

Modify the current valid entry to end at the time of publication.
Let's assume for this example that the new data should be accessible publicly from the 1st of January 2017 at 09:00 CET.

```
BEGIN;
```

```
UPDATE public.timeseries_main  
SET validity = tstzrange(lower(validity), '2017-01-01 09:00+01')  
WHERE ts_key = 'kofbarometer'  
AND validity @> now();
```

Create a new current entry accessible from the time of publication.

```
INSERT INTO public.timeseries_main  
SELECT ts_key,  
       ts_data || '2017-01-01=>100.00'::HSTORE, -- Add the new timeseries point  
       ts_frequency,  
       tstzrange('2017-01-01 09:00+01', NULL) -- Make available from this point in time  
FROM public.timeseries_main  
WHERE ts_key = 'kofbarometer'  
AND upper(validity) = '2017-01-01 09:00+01';  
  
COMMIT;
```

Publication Of Indicators

The Publication Process 2/2

We have 2 entries with the same key and and order in time.

```
SELECT ts_key, ts_data, validity
FROM public.timeseries_main
WHERE ts_key = 'kofbarometer';
-[ RECORD 1 ]-----
ts_key   | kofbarometer
ts_data  | "2016-11-01"=>"102.162463328593", "2016-12-01"=>"102.161371136933"
validity | ["2016-12-01 09:00:00+01", "'2017-01-01 09:00+01'")
-[ RECORD 2 ]-----
ts_key   | kofbarometer
ts_data  | "2016-11-01"=>"102.162463328593", "2016-12-01"=>"102.161371136933",
          | "2017-01-01"=>"100.00"
validity | ["2017-01-01 09:00+01",)
```

Querying with a time filter only delivers the current entry.

```
SELECT (each(ts_data)).*
FROM public.timeseries_main
WHERE ts_key = 'kofbarometer'
AND validity @> clock_timestamp();
```

key	value
2016-11-01	102.162463328593
2016-12-01	102.161371136933
2017-01-01	100.00

(3 rows)

Publication Of Indicators

Using Row Level Security

We have seen that filtering is done in the WHERE clause. We can achieve the same result using RLS.

```
CREATE POLICY public_access
ON public.timeseries_main
FOR SELECT
TO public
USING (validity @> clock_timestamp());
```

```
ALTER TABLE timeseries_main ENABLE ROW LEVEL SECURITY;
```

```
Table "public.timeseries_main"
Column      |      Type      | Modifiers
-----+-----+-----
ts_key       | character varying | not null
ts_data      | hstore          |
ts_frequency | integer         |
validity     | tstzrange       | not null
```

Indexes:

```
"timeseries_main_pkey" PRIMARY KEY, btree (ts_key, validity)
"timeseries_main_ts_key_validity_excl"
EXCLUDE USING gist (ts_key WITH =, validity WITH &&)
```

Policies:

```
POLICY "public_access" FOR SELECT
USING ((validity @> clock_timestamp()))
```

Publication Of Indicators

Testing RLS Settings

Table owner sees all entries.

```
SELECT CURRENT_USER;
```

```
-[ RECORD 1 ]+-----
```

```
current_user | charles
```

```
SELECT ts_key, ts_data, validity
```

```
FROM public.timeseries_main
```

```
WHERE ts_key = 'kofbarometer';
```

```
-[ RECORD 1 ]-----
```

```
ts_key      | kofbarometer
```

```
ts_data     | "2016-11-01"=>"102.162463328593", "2016-12-01"=>"102.161371136933"
```

```
validity    | ["2016-12-01 09:00:00+01", "2017-01-01 09:00+01")
```

```
-[ RECORD 2 ]-----
```

```
ts_key      | kofbarometer
```

```
ts_data     | "2016-11-01"=>"102.162463328593", "2016-12-01"=>"102.161371136933",
```

```
            | "2017-01-01"=>"100.00"
```

```
validity    | ["2017-01-01 09:00+01",)
```

Publication Of Indicators

Testing RLS Settings

Any other user without any specific policy defined is public and only sees the currently valid row.

```
SET ROLE public_reader;
```

```
SELECT CURRENT_USER;
-[ RECORD 1 ]+-----
current_user | public_reader
```

```
SELECT ts_key, ts_data, validity
FROM public.timeseries_main
WHERE ts_key = 'kofbarometer';
```

```
-[ RECORD 1 ]-----
ts_key      | kofbarometer
ts_data     | "2016-11-01"=>"102.162463328593", "2016-12-01"=>"102.161371136933",
           | "2017-01-01"=>"100.00"
validity    | ["2017-01-01 09:00+01",)
```

Outline

- 1 Introduction
- 2 Range Types
- 3 Use Cases
 - Manage Questionnaires Versions
 - History of Survey Participants
 - Dynamic Agenda Display
 - Publication Of Indicators
- 4 Conclusion

Conclusions

Date and time based range types offer a solid and flexible solution for a wide variety of needs, including but not limited to:

- Historicize entries for archiving or documentation.
- Create time responsive entries, i.e. decouple database interactions of delivery and consumption.
- Can easily be integrated with capabilities of the database such as indexing.
- They integrate also very well with advanced features like Row Level Security.

Resources

Documentation

- These slides: http://www.artesano.ch/documents/04-publications/time_related_range_types_revisited_pdfa.pdf
- Description: <https://www.postgresql.org/docs/current/static/rangetypes.html>
- Functions/Operators: <https://www.postgresql.org/docs/current/static/functions-range.html>

Live examples

- Show Next Event: <http://www.swispug.org>
- Dynamic Agenda: <https://www.swisspug.org/wiki/index.php/Agenda>
- Just In Time Publishing: <https://www.kof.ethz.ch/prognosen-indikatoren/indikatoren/kof-konjunkturbarmeter.html>

Related presentations

- Hubert "depesz" Lubaczewski (2010): <https://www.depesz.com/2010/10/22/grouping-data-into-time-ranges/>
- Jonathan Katz (2012): <https://wiki.postgresql.org/images/7/73/Range-types-pgopen-2012.pdf>
- Magnus Hagander (2015): https://www.hagander.net/talks/tardis_orm.pdf
- Miroslav Šedivý (2017): http://www.pgday.ch/common/slides/2018_swisspgday_miroslav_sedivy_-_tracking_reproducible_revisions_in_postgresql_with_psyncpg2.pdf

Contact

- Work: clavadetscher@kof.ethz.ch
<http://www.kof.ethz.ch>
- SwissPUG: clavadetscher@swisspug.org
<http://www.swisspug.org>
- Private: charles@artesanoch.ch
<http://www.artesanoch.ch>

Thank you

Thank you very much for your attention !

Feedback on talk and conference: <https://2018.pgconf.eu/f>

Q&A