PERIODS AND SYSTEM-VERSIONED TABLES

Vik Fearing

February 1, 2019

FOSDEM PGDay, Brussels

VIK FEARING

https://www.2ndQuadrant.fr/

PERIODS

WHAT IS A PERIOD?

Wait.

Please do not interrupt me to talk about range types.

WHAT IS A PERIOD?

SQL:2011

A starting value: not null, inclusive

An ending value: not null, exclusive

A constraint: start < end

Same namespace as columns

SCHEDULING

```
CREATE TABLE reservations (
    guest       text    NOT NULL,
    room_number integer NOT NULL,
    checkin     date    NOT NULL,
    checkout    date    NOT NULL,

    PERIOD FOR stay (checkin, checkout)
);
```

```
ALTER TABLE reservations
    ADD PRIMARY KEY (room_number, stay WITHOUT OVERLAPS);
```

SCHEDULING

Guest	Room	Check In	Check Out
Alice	112	2018-10-08	2018-10-11

```
INSERT INTO reservations
VALUES ('Bob', 112, '2018-10-10', '2018-10-12');
```

ERROR

PERIOD PREDICATES

PERIOD PREDICATES

OVERLAPS	p1 OVERLAPS p2
EQUALS	p1 EQUALS p2
CONTAINS	p1 CONTAINS p2 p1 CONTAINS value

PERIOD PREDICATES

PRECEDES	p1 PRECEDES p2
SUCCEEDS	p1 SUCCEEDS p2
IMMEDIATELY PRECEDES	p1 IMMEDIATELY PRECEDES p2
IMMEDIATELY SUCCEEDS	p1 IMMEDIATELY SUCCEEDS p2

SALES TAX RATES

```
CREATE TABLE vat (
    start_date  date        NOT NULL,
    end_date    date        NOT NULL,
    rate        percentage  NOT NULL,

    PERIOD FOR validity (start_date, end_date),
    PRIMARY KEY (validity WITHOUT OVERLAPS)
);
```

SALES TAX RATES

for Switzerland

Start	End	Rate
-infinity	2011	7.6%
2011	2018	8%
2018	*infinity*	7.7%

```
SELECT *
FROM invoices
JOIN vat ON vat.validity CONTAINS invoices.invoice_date;
```

NON-TEMPORAL PERIODS

(not in the SQL standard)

PRICING STRATEGIES

```
CREATE TABLE pricing (
    product_name  text    NOT NULL,
    unit_price    numeric NOT NULL,
    min_quantity  integer NOT NULL,
    max_quantity  integer NOT NULL,

    PERIOD FOR quantity_range (min_quantity, max_quantity),
    CHECK (min_quantity > 0),
    PRIMARY KEY (product_name, quantity_range WITHOUT OVERLAPS)
);
```

PRICING STRATEGIES

Product	Unit Price	Min	Max
Trinket	€1000	1	10
Trinket	€800	10	500
Trinket	€600	500	1000
Trinket	€250	1000	10000

```
SELECT * FROM pricing WHERE quantity_range CONTAINS 42;
```

PORTIONS

DELETE

```
CREATE TABLE vacation (
    employee   text      NOT NULL,
    start_date date      NOT NULL,
    end_date   date      NOT NULL,

    PERIOD FOR dates (start_date, end_date),
    PRIMARY KEY (employee, dates WITHOUT OVERLAPS)
);
```

DELETE

Employee	Start	End
Charlotte	2018-08-01	2018-09-01

```
DELETE FROM vacation
FOR PORTION OF dates FROM '2018-08-10' TO '2018-08-11'
WHERE employee = 'Charlotte';
```

Employee	Start	End
Charlotte	2018-08-01	2018-08-10
Charlotte	2018-08-11	2018-09-01

UPDATE

```
CREATE TABLE products (
    name       text       NOT NULL,
    opening    date       NOT NULL,
    closing    date       NOT NULL,
    price      numeric    NOT NULL,

    PERIOD FOR season (opening, closing),
    PRIMARY KEY (name, season WITHOUT OVERLAPS)
);
```

UPDATE

Product	Open	Close	Price
Trinket	2018-01-01	2019-01-01	€894.85

```
UPDATE products
FOR PORTION OF season FROM '2018-12-01' TO '2018-12-23'
SET price = 100;
```

UPDATE

Product	Open	Close	Price
Trinket	2018-01-01	2018-12-01	€894.85
Trinket	2018-12-01	2019-12-23	€100.00
Trinket	2018-12-23	2019-01-01	€894.85

SYSTEM-VERSIONED TABLES

(must be timestamp with time zone*)*

SYSTEM-VERSIONED TABLES

```
CREATE TABLE clients (
    id         bigint         PRIMARY KEY,
    name       text           NOT NULL,
    email      text           NOT NULL,
    sys_start timestamptz,
    sys_end   timestamptz,
    PERIOD FOR SYSTEM_TIME (sys_start, sys_end)
);
```

SYSTEM-VERSIONED TABLES

```
CREATE TABLE clients (
    id         bigint         PRIMARY KEY,
    name       text           NOT NULL,
    email      text           NOT NULL,
    sys_start timestamptz   GENERATED ALWAYS AS ROW START,
    sys_end   timestamptz   GENERATED ALWAYS AS ROW END,
    PERIOD FOR SYSTEM_TIME (sys_start, sys_end)
);
```

NOTHING CHANGES!

SYSTEM-VERSIONED TABLES

```
SELECT * FROM clients;
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	alice@gadgets.com	2018-10-01 09:53+02	'infinity'

SYSTEM-VERSIONED TABLES

```
UPDATE clients SET email = 'bob@gadgets.com' WHERE id = 3784;
```

```
SELECT * FROM clients;
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	bob@gadgets.com	2018-10-09 15:47+02	'infinity'

SYSTEM-VERSIONED TABLES

```
UPDATE clients SET email = 'carla@gadgets.com' WHERE id = 3784;
```

```
SELECT * FROM clients;
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	carla@gadgets.com	2018-10-11 12:04+02	'infinity'

SYSTEM-VERSIONED TABLES

```
SELECT *
FROM clients FOR SYSTEM_TIME FROM '-infinity' TO 'infinity';
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	alice@gadgets.com	2018-10-01 09:53+02	2018-10-09 15:47+02
3784	Gadgets Inc.	bob@gadgets.com	2018-10-09 15:47+02	2018-10-11 12:04+02
3784	Gadgets Inc.	carla@gadgets.com	2018-10-11 12:04+02	'infinity'

SYSTEM-VERSIONED TABLES

```
SELECT *
FROM clients FOR SYSTEM_TIME AS OF '2018-10-07 12:00+02';
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	alice@gadgets.com	2018-10-01 09:53+02	2018-10-09 15:47+02

SYSTEM-VERSIONED TABLES

- AS OF ts
- FROM ts1 TO ts2
- BETWEEN ts1 AND ts2
- BETWEEN SYMMETRIC ts1 AND ts2

IMPLEMENTATION IN POSTGRESQL

WHAT IS A PERIOD?

A starting value: not null, inclusive

An ending value: not null, exclusive

A constraint: start < end

Same namespace as columns

WHAT IS A RANGE TYPE?

A starting value

An ending value

A constraint: start <= end

BUT!

Bounds can be either inclusive or exclusive

Bounds can be null

Ranges can be empty

SCHEDULING

```
CREATE TABLE reservations (
    guest       text    NOT NULL,
    room_number integer NOT NULL,
    checkin     date    NOT NULL,
    checkout    date    NOT NULL,

    PERIOD FOR stay (checkin, checkout)
);
```

```
ALTER TABLE reservations
    ADD PRIMARY KEY (room_number, stay WITHOUT OVERLAPS);
```

SCHEDULING

```
CREATE TABLE reservations (
    guest       text      NOT NULL,
    room_number integer   NOT NULL,
    stay        daterange NOT NULL
);
```

```
ALTER TABLE reservations ADD PRIMARY KEY (room_number, stay);
```

```
CREATE EXTENSION btree_gist;
```

```
ALTER TABLE reservations
    ADD EXCLUDE USING gist (room_number WITH =, stay WITH &&);
```

SCHEDULING

Guest	Room	Check In	Check Out
Alice	112	2018-10-08	2018-10-11

```
INSERT INTO reservations
VALUES ('Bob', 112, '2018-10-10', '2018-10-12');
```

ERROR

SCHEDULING

Guest	Room	Stay
Alice	112	[2018-10-08,2018-10-11)

```
INSERT INTO reservations
VALUES ('Bob', 112, '[2018-10-10,2018-10-12)');
```

ERROR

PERIOD PREDICATES

OVERLAPS	p1 OVERLAPS p2	p1 && p2
EQUALS	p1 EQUALS p2	p1 = p2
CONTAINS	p1 CONTAINS p2	p1 @> p2
	p1 CONTAINS value	p1 @> value

PERIOD PREDICATES

PRECEDES	p1 PRECEDES p2	p1 << p2
SUCCEEDS	p1 SUCCEEDS p2	p1 >> p2

PERIOD PREDICATES

IMMEDIATELY PRECEDES	p1 IMMEDIATELY PRECEDES p2	p1 << p2 and p1 -|- p2
IMMEDIATELY SUCCEEDS	p1 IMMEDIATELY SUCCEEDS p2	p1 >> p2 and p1 -|- p2

Two new operators, <| and |>, would be useful outside of this.

PRICING STRATEGIES

```
CREATE TABLE pricing (
    product_name  text    NOT NULL,
    unit_price    numeric NOT NULL,
    min_quantity  integer NOT NULL,
    max_quantity  integer NOT NULL,

    PERIOD FOR quantity_range (min_quantity, max_quantity),
    CHECK (min_quantity > 0),
    PRIMARY KEY (product_name, quantity_range WITHOUT OVERLAPS)
);
```

PRICING STRATEGIES

```
CREATE TABLE pricing (
    product_name   text      NOT NULL,
    unit_price     numeric   NOT NULL,
    quantity_range int4range NOT NULL,

    CHECK (lower(quantity_range) > 0),
    PRIMARY KEY (product_name, quantity_range),
    EXCLUDE USING gist (product_name   WITH =,
                        quantity_range WITH &&)
);
```

PRICING STRATEGIES

Product	Unit Price	Min	Max
Trinket	€1000	1	10
Trinket	€800	10	500
Trinket	€600	500	1000
Trinket	€250	1000	10000

```
SELECT * FROM pricing WHERE quantity_range CONTAINS 42;
```

PRICING STRATEGIES

Product	Unit Price	Quantities
Trinket	€1000	[1,10)
Trinket	€800	[10,500)
Trinket	€600	[500,1000)
Trinket	€250	[1000,10000)

```
SELECT * FROM pricing WHERE quantity_range @> 42;
```

NON-TEMPORAL PERIODS

(not in the SQL standard)

(range types aren't either so ¯_(ツ)_/¯)

SALES TAX RATES

Start	End	Rate
-infinity	2011	7.6%
2011	2018	8%
2018	*infinity*	7.7%

```
SELECT *
FROM invoices
JOIN vat ON vat.validity CONTAINS invoices.invoice_date;
```

SALES TAX RATES

Validity	Rate
[-*infinity*,2011)	7.6%
[2011,2018)	8%
[2018,*infinity*)	7.7%

```
SELECT *
FROM invoices
JOIN vat ON vat.validity @> invoices.invoice_date;
```

This can't be done with an extension.

```
DELETE FROM vacation
FOR PORTION OF dates FROM '2018-08-10' TO '2018-08-11'
WHERE employee = 'Charlotte';
```

```
UPDATE products
FOR PORTION OF season FROM '2018-12-01' TO '2018-12-23'
SET price = 100;
```

SYSTEM-VERSIONED TABLES

```
CREATE EXTENSION sysver;
```

https://github.com/xocolatl/sysver

SYSTEM-VERSIONED TABLES

```
CREATE TABLE clients (
    id          bigint          PRIMARY KEY,
    name        text            NOT NULL,
    email       text            NOT NULL,
);
```

SYSTEM-VERSIONED TABLES

```
SELECT sysver.sysver_register('clients');
```

- Two columns: sysver_start and sysver_end
- NOT NULL constraints
- CHECK (sysver_start < sysver_end)
- BEFORE trigger for GENERATED ALWAYS
- AFTER trigger for historization

SYSTEM-VERSIONED TABLES

```
SELECT sysver.sysver_register('clients');
```

- `clients_history` table, same structure
- `clients_with_history` view to combine them
- Several functions

SYSTEM-VERSIONED TABLES

```
SELECT * FROM clients;
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	carla@gadgets.com	2018-10-11 12:04+02	'infinity'

SYSTEM-VERSIONED TABLES

```
SELECT *
FROM clients FOR SYSTEM_TIME FROM '-infinity' TO 'infinity';
```

```
SELECT *
FROM clients__from_to('-infinity', 'infinity');
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	alice@gadgets.com	2018-10-01 09:53+02	2018-10-09 15:47+02
3784	Gadgets Inc.	bob@gadgets.com	2018-10-09 15:47+02	2018-10-11 12:04+02
3784	Gadgets Inc.	carla@gadgets.com	2018-10-11 12:04+02	'infinity'

SYSTEM-VERSIONED TABLES

```
SELECT *
FROM clients__as_of('2018-10-07 12:00+02');
```

ID	Client	Email	SysStart	SysEnd
3784	Gadgets Inc.	alice@gadgets.com	2018-10-01 09:53+02	2018-10-09 15:47+02

SYSTEM-VERSIONED TABLES

AS OF ts	table__as_of(ts)
FROM ts1 TO ts2	table__from_to(ts1, ts2)
BETWEEN ts1 AND ts2	table__between(ts1, ts2)
BETWEEN SYMMETRIC ts1 AND ts2	table__between_symmetric(ts1, ts2)

BETWEEN SYMMETRIC

```
commit 6f19a8c41f976236310a272bb646d3411759e18d
Author: Tom Lane
Date:   Sun Dec 30 13:42:04 2018 -0500

    Teach eval_const_expressions to constant-fold LEAST/GREATEST expressions.

    Doing this requires an assumption that the invoked btree comparison
    function is immutable.  We could check that explicitly, but in other
    places such as contain_mutable_functions we just assume that it's true,
    so we may as well do likewise here.  (If the comparison function's
    behavior isn't immutable, the sort order in indexes built with it would
    be unstable, so it seems certainly wrong for it not to be so.)

    Vik Fearing
```

IMPLEMENTATION IN POSTGRESQL

(for real)

IMPLEMENTATION IN POSTGRESQL

WIP Patch posted to -hackers on May 26, 2018

Almost complete infrastructure for periods

Incomplete support for table inheritance

Incomplete support for pg_dump

None of the other features mentioned in this presentation

QUESTIONS?