Time Travel – Yes, But How?

How I am?

Hettie

DRW, Chicago

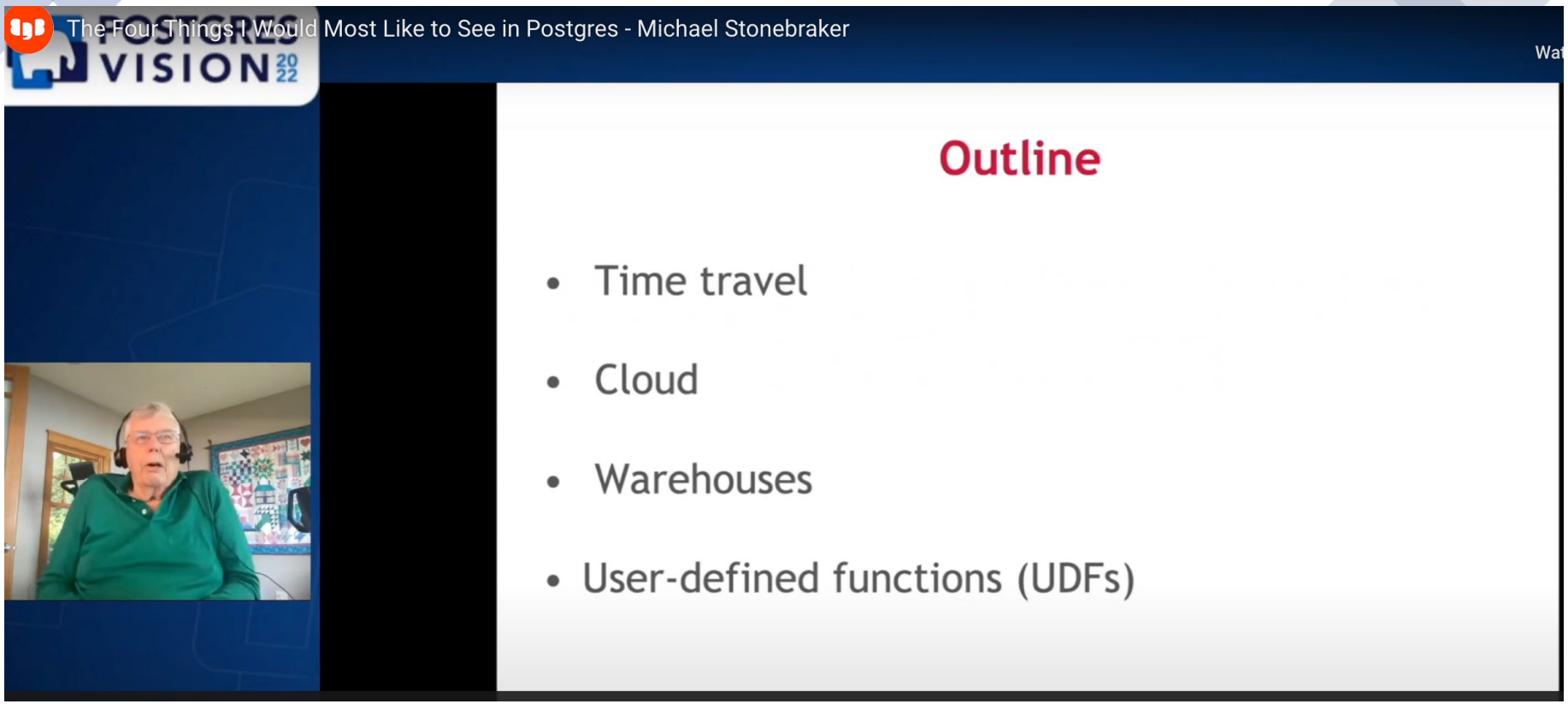
Chicago PUG

Changelog? Two-dimensional time? What works better?

Hettie Dombrovskaya

DRW

Postgres Vision 2022


The Four Things I Would Most Like to See in Postgres - Michael Stonebraker
POSTGRES VISION 2022
Outline
Time travel
Cloud
Warehouses
User-defined functions (UDFs)

Time Travel

Select (…)
from Table (T)
Where …

Allows you to query the database at a historical time (T)

DBg Database Group
MIT Computer Science and Artificial Intelligence Lab

Time Travel

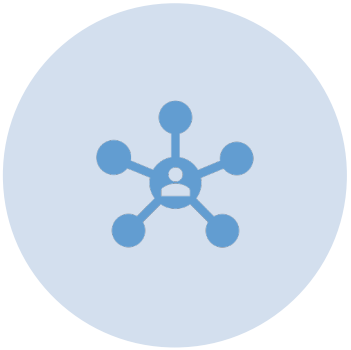
- My implementation was awful
  - Slow, slow, slow
- Rightfully deleted by the powers-that-be
- In my opinion time travel is a very good idea
  - User/application mistakes
  - Query the log
  - Security
  - Debugging
- New implementation needed (stay tuned)

How?

What exactly we need?

POINT-IN-TIME RECOVERY – YES!

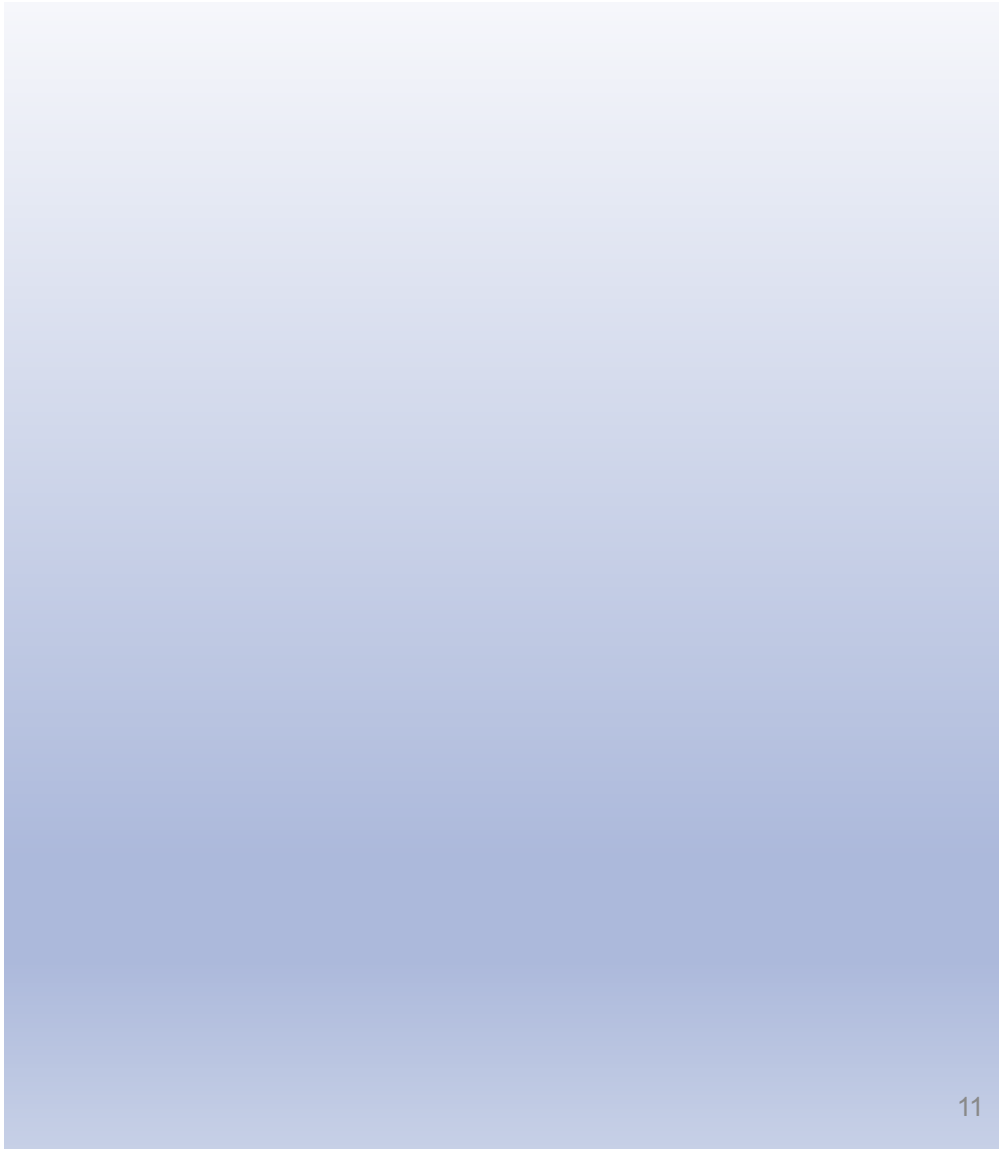
CHANGELOG – YES!



STORING BITEMPORAL DATA - ????

What are bitemporal functions and why anybody would want to use them?

Quick refresher



Asserted Versioning Model

- *Bitemporal functions* operate on bitemporal tables
- *Bitemporal* – two-dimensional time:
  - effective time
  - asserted time
  - the combination of effective and asserted times: **time regions**

NOT SQL Standard

Bitemporal operations

- create bitemporal table
- bitemporal insert
- bitemporal update
- bitemporal correction
- Bitemporal inactivate
- bitemporal delete

Bitemporal CREATE TABLE

```
select * from bitemporal_internal.ll_create_bitemporal_table(
        'postgres_air_bitemp',
        'frequent_flyer_transaction',
        $$frequent_flyer_id int,
          level text,
          booking_leg_id int,
          award_points int,
          status_points int$$,
          'frequent_flyer_id');
```

```
CREATE TABLE IF NOT EXISTS postgres_air_bitemp.frequent_flyer_transaction(
  frequent_flyer_transaction_key integer NOT NULL DEFAULT
        nextval('postgres_air_bitemp.frequent_flyer_transaction_frequent_flyer_transaction_key_seq'::regclass),
    frequent_flyer_id integer NOT NULL,
    level text ,
    booking_leg_id integer,
    award_points integer,
    status_points integer,
    effective temporal_relationships.timeperiod NOT NULL,
    asserted temporal_relationships.timeperiod NOT NULL,
    row_created_at timestamp with time zone NOT NULL DEFAULT now(),
    CONSTRAINT frequent_flyer_transaction_pk PRIMARY KEY (frequent_flyer_transaction_key),
    CONSTRAINT frequent_flyer_transaction_frequent_flyer_id_assert_eff_excl EXCLUDE USING gist (
        effective WITH &&,
        asserted WITH &&,
        frequent_flyer_id WITH =)
)
```

2022

Why PostgreSQL works best for bitemporality?

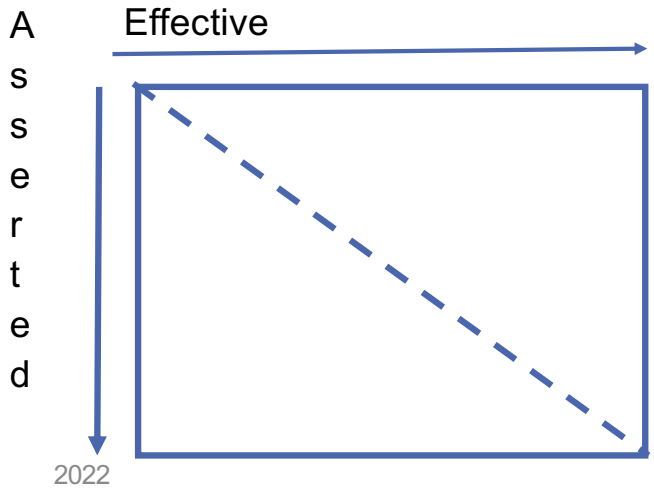
- Range types
- +/- infinity
- GIST/GIST with exclusion constraints

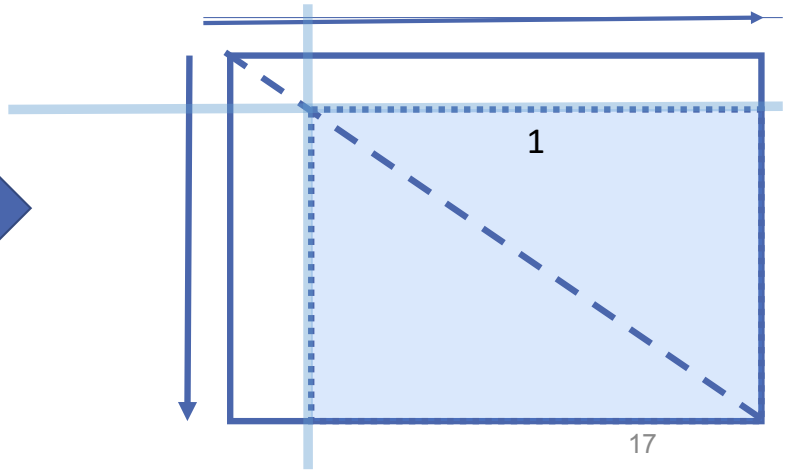
Bitemporal Insert

now = 2022-05-01

```
select ll_bitemporal_insert(
        'customers',
        $$'customer_no', name', 'type' $$,
        $$'C100','John Doe', 'Silver' $$,
        timeperiod('2022-06-01','infinity'),
        timeperiod('2022-05-01','infinity'))
```

#	Effective Interval	Assertive Interval	Customer No.	Name	Type
1	[2022-06-01, oo)	[2022-05-01 , oo)	C100	John Doe	Silver



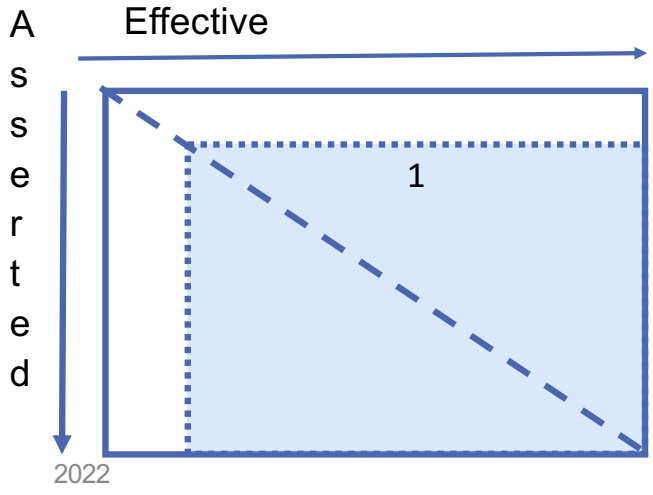


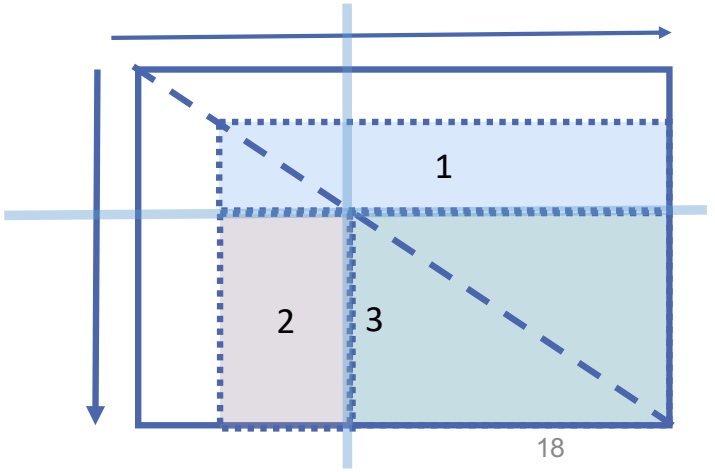
Bitemporal Update

now = 2022-09-15

#	Effective Interval	Assertive Interval	Customer No.	Name	Type
1	[2022-06-01, oo)	[2022-05-01,2022-09-15)	C100	John Doe	Silver
2	[2022-06-01,2022-09-15)	[2022-09-15, oo)	C100	John Doe	Silver
3	[2022-09-15, oo)	[2022-09-15, oo)	C100	John Doe	Gold

```
select
ll_bitemporal_update($$customers$$,
$$customer_no$$, $$100$$,
$$type$$, $$Gold$$,
timeperiod('2022-09-15', 'infinity'),
timeperiod('2022-09-15', 'infinity'))
```



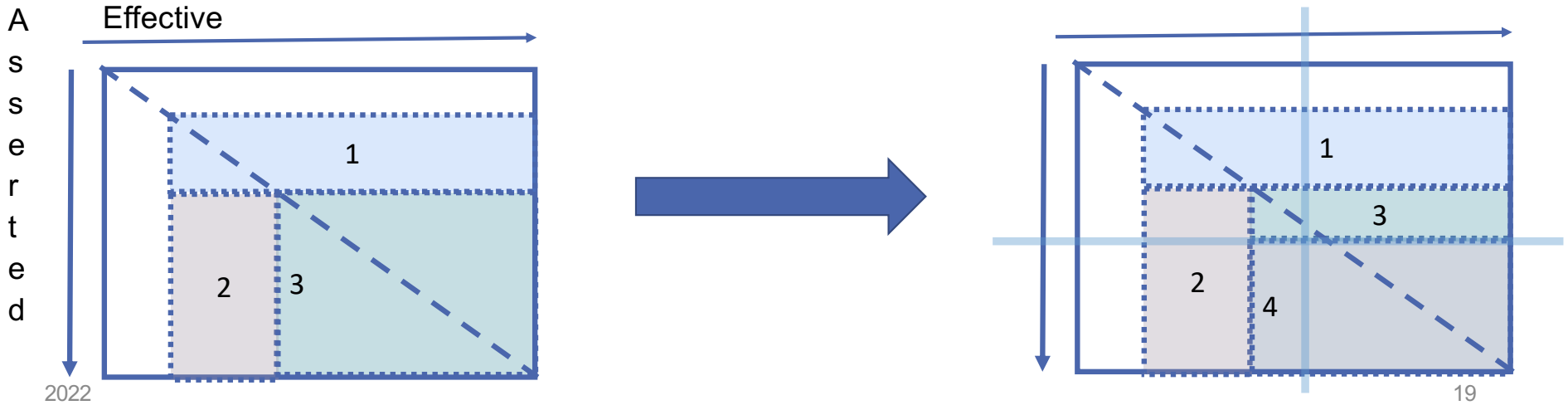


Bitemporal Correction

now = 2022-09-22

```
select ll_bitemporal_correction($$customers$$,
      $$type $$,
      $$ Platinum$$,
      $$ customer_no $$,
      $$ C100$$,
      timeperiod('2022-09-15','infinity'),
      now())
```

Effective Interval	Assertive Interval	Customer No.	Name	Type
[2022-06-01, oo)	[2022-05-01,2022-09-15)	C100	John Doe	Silver
[2022-06-01,2022-09-15)	[2022-09-15, oo)	C100	John Doe	Silver
[2022-09-15, oo)	[2022-09-15, 2022-09-22)	C100	John Doe	Gold
[2022-09-15, oo)	[2022-09-22, oo)	C100	John Doe	Platinum

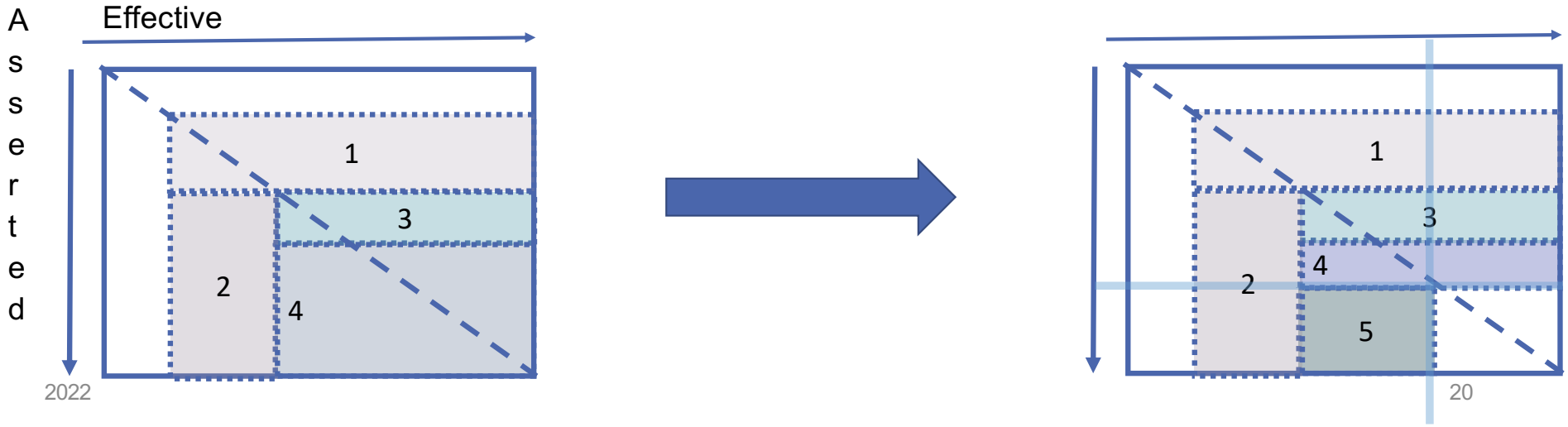


Bitemporal Inactivate

now = 2022-11-05

```
select ll_bitemporal_inactivate(
      $$customers$$,
      $$customer_no$$,
      $$C100$$,
      timeperiod('2022-12-31','infinity'),
      timeperiod('2022-11-05','infinity'),
```

#	Effective Interval	Assertive Interval	Customer No.	Name	Type
1	[2022-06-01, oo)	[2022-05-01,2022-09-15)	C100	John Doe	Silver
2	[2022-06-01,2022-09-15)	[2022-09-15, oo)	C100	John Doe	Silver
3	[2022-09-15, oo)	[2022-09-15, 2022-09-22)	C100	John Doe	Gold
4	[2022-09-15, oo)	[2022-09-22, 2022-11-05)	C100	John Doe	Platinum
5	[2022-09-15,2022-12-31)	[2022-11-05, oo)	C100	John Doe	Platinum

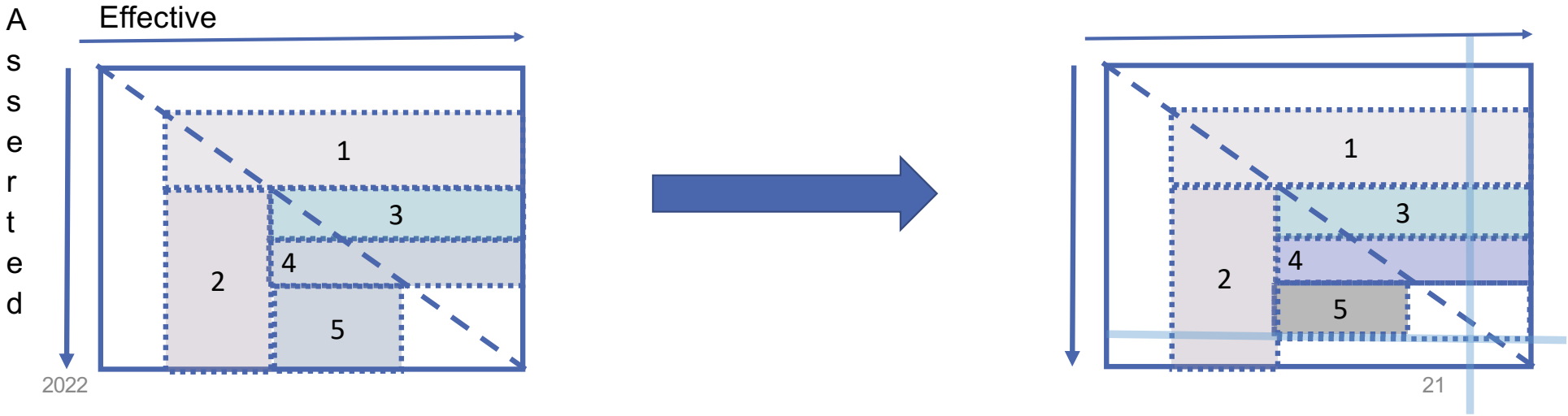


Bitemporal Delete

now = 2022-11-17

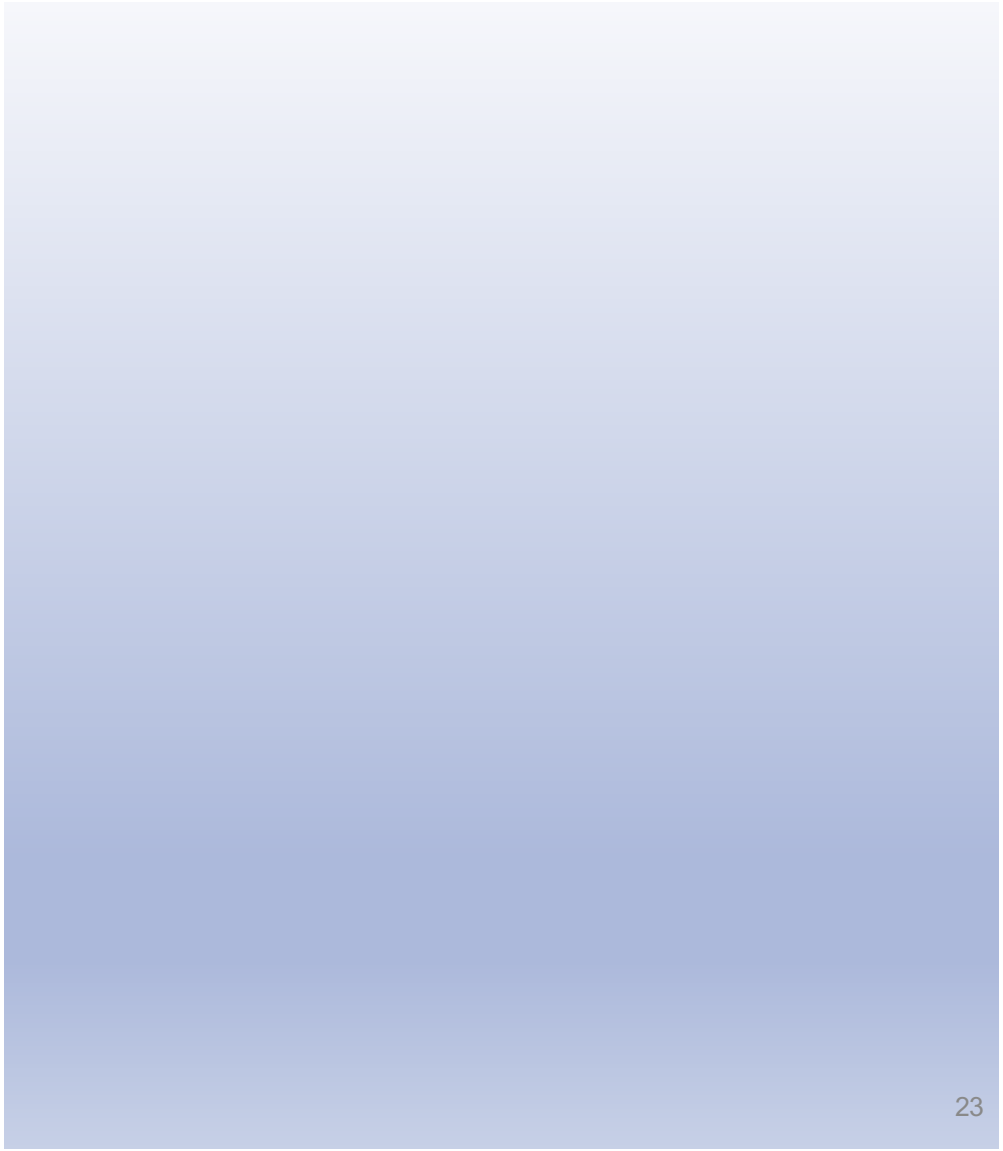
```
select ll_bitemporal_delete(
        'customers',
        $$ customer_no $$,
        $$ 'C100' $$,
        timeperiod('2022-11-17','infinity'))
```

#	Effective Interval	Assertive Interval	Customer No.	Name	Type
1	[2022-06-01, oo)	[2022-05-01,2022-09-15)	C100	John Doe	Silver
2	[2022-06-01,2022-09-15)	[2022-09-15, oo)	C100	John Doe	Silver
3	[2022-09-15, oo)	[2022-09-15, 2022-09-22)	C100	John Doe	Gold
4	[2022-09-15, oo)	[2022-09-22, 2022-11-05)	C100	John Doe	Platinum
5	[2022-09-15,2022-12-31)	[2022-11-05, 2022-11-17)	C100	John Doe	Platinum



Implementation: https://github.com/hettie-d/pg_bitemporal

Frequent Flyer Example

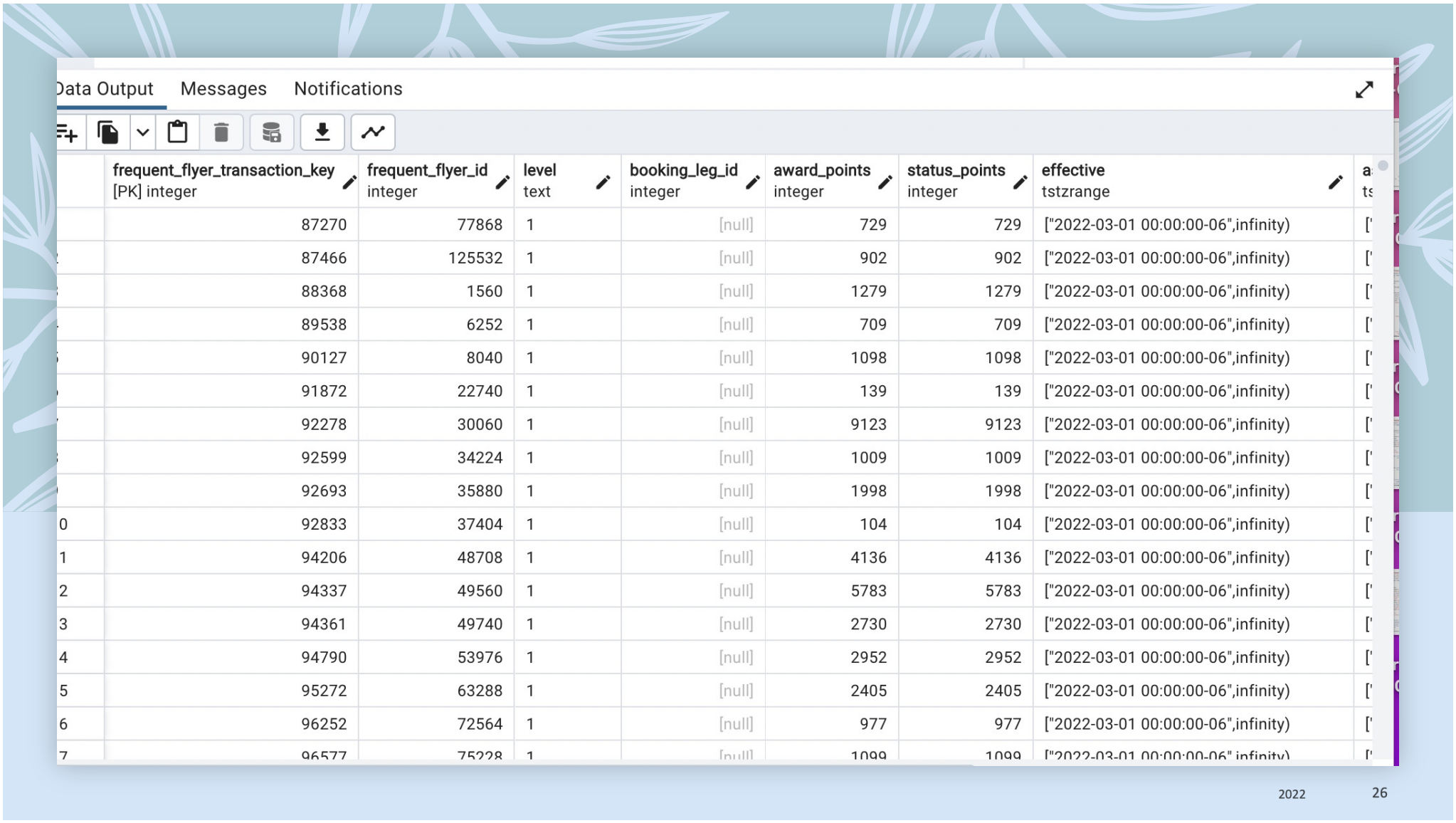


frequent_flyer_transaction(4 M rows)

FREQUENT_FLYER_TRANSACTION	
frequent_flyer_transaction_key	serial
frequent_flyer_id	int
level	text
booking_leg_id	int
award_points	int
status_points	int
effective	tstzrange
asserted	tstzrange
row_created_at	timestamptz

```
select *
from
   postgres_air_bitemp.frequent_flyer_transaction t
   where  now()<@asserted
          and now()<@effective

--700 ms
```



Data Output Messages Notifications

	frequent_flyer_transaction_key [PK] integer	frequent_flyer_id integer	level text	booking_leg_id integer	award_points integer	status_points integer	effective tstzrange	a ts
	87270	77868	1	[null]	729	729	["2022-03-01 00:00:00-06",infinity)	["
1	87466	125532	1	[null]	902	902	["2022-03-01 00:00:00-06",infinity)	["
	88368	1560	1	[null]	1279	1279	["2022-03-01 00:00:00-06",infinity)	["
	89538	6252	1	[null]	709	709	["2022-03-01 00:00:00-06",infinity)	["
	90127	8040	1	[null]	1098	1098	["2022-03-01 00:00:00-06",infinity)	["
	91872	22740	1	[null]	139	139	["2022-03-01 00:00:00-06",infinity)	["
	92278	30060	1	[null]	9123	9123	["2022-03-01 00:00:00-06",infinity)	["
	92599	34224	1	[null]	1009	1009	["2022-03-01 00:00:00-06",infinity)	["
	92693	35880	1	[null]	1998	1998	["2022-03-01 00:00:00-06",infinity)	["
10	92833	37404	1	[null]	104	104	["2022-03-01 00:00:00-06",infinity)	["
11	94206	48708	1	[null]	4136	4136	["2022-03-01 00:00:00-06",infinity)	["
12	94337	49560	1	[null]	5783	5783	["2022-03-01 00:00:00-06",infinity)	["
13	94361	49740	1	[null]	2730	2730	["2022-03-01 00:00:00-06",infinity)	["
14	94790	53976	1	[null]	2952	2952	["2022-03-01 00:00:00-06",infinity)	["
15	95272	63288	1	[null]	2405	2405	["2022-03-01 00:00:00-06",infinity)	["
16	96252	72564	1	[null]	977	977	["2022-03-01 00:00:00-06",infinity)	["
17	96577	75228	1	[null]	1099	1099	["2022-03-01 00:00:00-06",infinity)	["

Individual History

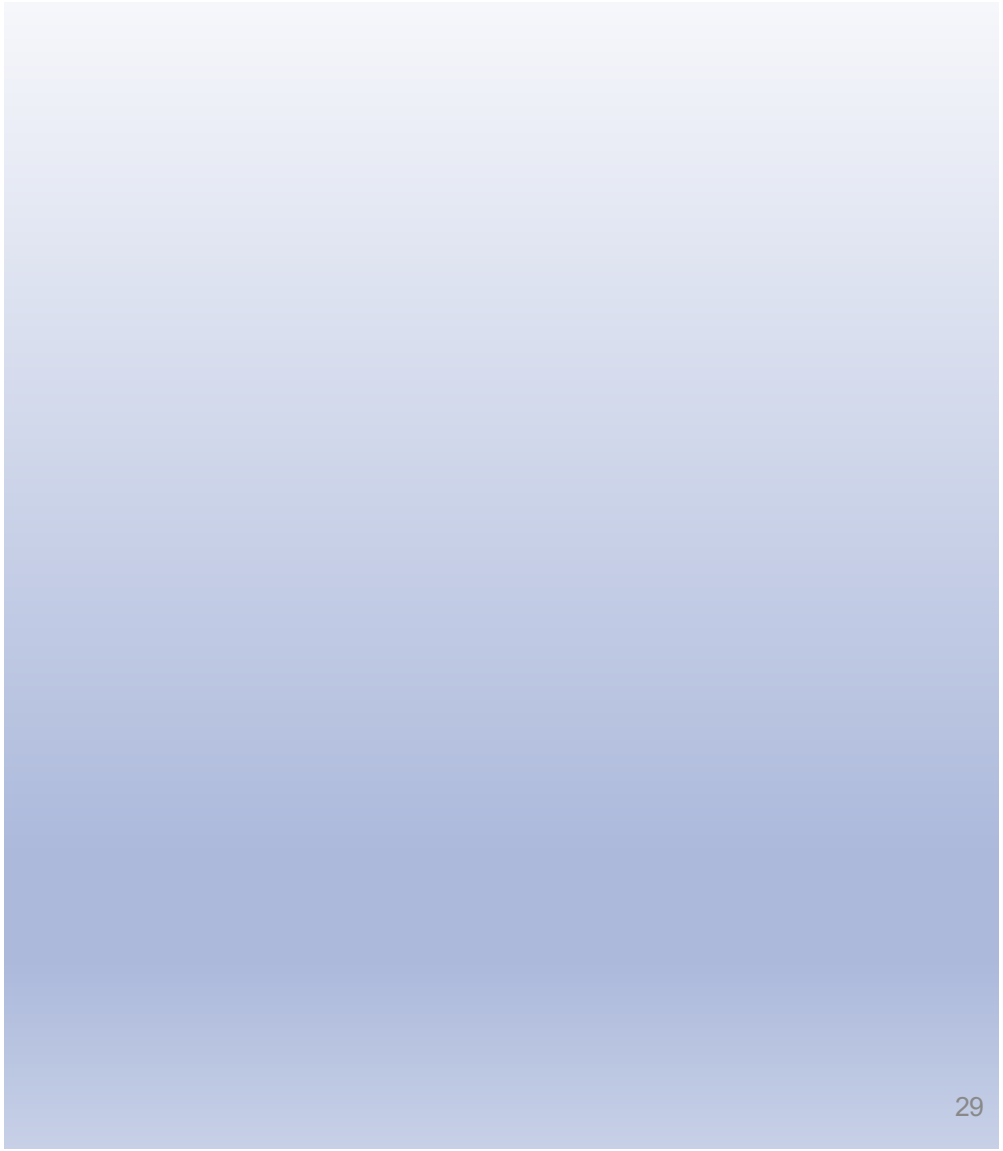
14 ms

```
select frequent_flyer_id,
lower(effective) as start_time,
booking_leg_id,
departure_airport,
arrival_airport,
award_points,
status_points,
level
from postgres_air_bitemp.frequent_flyer_transaction t
join postgres_air.booking_leg bl using (booking_leg_id)
join postgres_air.flight f
using (flight_id)
where frequent_flyer_id=39189
and now()<@asserted
order by lower(effective)
```

Data Output | Messages | Notifications

	frequent_flyer_id integer	start_time timestamp with time zone	booking_leg_id integer	departure_airport character (3)	arrival_airport character (3)	award_points integer	status_points integer	level text
1	39189	2022-03-12 21:43:06.483…	17463738	CDG	BOS	47064	63864	1
2	39189	2022-04-03 05:23:31.982…	2318694	NRT	KUL	30564	63864	1
3	39189	2022-04-03 05:23:34.982…	2318697	CPH	FRA	28464	63864	1
4	39189	2022-04-30 19:08:16.484…	6204495	KBP	GME	27564	63864	1
5	39189	2022-05-31 06:14:53.04-05	17463737	TNR	CDG	36364	72664	1
6	39189	2022-06-08 12:12:58.08-05	17463739	BOS	CDG	41964	78264	1
7	39189	2022-06-09 11:23:43.08-05	17463740	CDG	TNR	50764	87064	1
8	39189	2022-06-23 08:57:37.68-05	2318695	KUL	FRA	60764	97064	1
9	39189	2022-06-24 08:18:00.72-05	2318696	FRA	CPH	61464	97764	1
10	39189	2022-07-01 04:09:42.981…	10874409	EWR	MSY	55764	97764	1
11	39189	2022-07-04 11:40:00-05	2318698	FRA	KUL	65764	7764	2
12	39189	2022-07-05 07:25:00-05	2318699	KUL	NRT	71264	13264	2
13	39189	2022-07-13 09:07:08.233…	12743928	ORD	DEN	66764	13264	2
14	39189	2022-07-20 01:40:00-05	6204494	GME	KBP	67064	13564	2

Building Change Logs



```
with award_points as (select
           frequent_flyer_id,
           time_of_change,
           booking_leg_id,
           current_points_balance,
           points_balance_change,
           activity
from (
           select frequent_flyer_id,
           lower(effective) as time_of_change,
           booking_leg_id,
    award_points as current_points_balance,
    coalesce(lag(award_points) over (w),0) as previous_points_balance,
    award_points - coalesce(lag(award_points) over(w),0) as points_balance_change,
case when (award_points -  coalesce(lag(award_points)over(w), 0))<0 then 'points used'
     when (award_points - coalesce(lag(award_points)over(w),0))>0 then 'points earned'
else 'no_change'
end as activity
from postgres_air_bitemp.frequent_flyer_transaction
where now()<@asserted
window w as (partition by frequent_flyer_id order by lower(effective))
) a where points_balance_change !=0 )
```

1.7 sec

```
select
    frequent_flyer_id,
     time_of_change,
     departure_airport,
    arrival_airport,
     current_points_balance,
     points_balance_change,
     activity
from award_points a
join postgres_air.booking_leg bl using (booking_leg_id)
join postgres_air.flight f
using (flight_id)
order by 1,2;
```

Data Output Messages Notifications

	frequent_flyer_id integer	time_of_change timestamp with time zone	departure_airport character (3)	arrival_airport character (3)	current_point integer	points_balance_ integer	activity text
1	2	2022-03-29 06:36:39.236…	BRU	LAD	15334	-20100	points used
2	2	2022-04-24 02:43:36.794…	JFK	BNA	11434	-3900	points used
3	2	2022-06-17 18:20:00-05	LAD	BRU	18134	6700	points earned
4	2	2022-07-05 23:04:22.147…	ORD	ANC	4334	-13800	points used
5	2	2022-07-21 08:35:00-05	BNA	JFK	5634	1300	points earned
6	2	2022-07-24 21:05:02.030…	HRB	TAO	2034	-3600	points used
7	2	2022-08-02 05:17:02.396…	CMH	ORD	534	-1500	points used
8	3	2022-03-12 02:31:03.270…	SJC	PDX	17819	-3000	points used
9	3	2022-03-12 02:31:06.270…	PDX	SJC	14819	-3000	points used
10	3	2022-04-06 13:50:58.031…	ATL	PHL	11519	-3300	points used
11	3	2022-04-18 01:47:01.833…	MSP	DTW	8819	-2700	points used
12	3	2022-04-20 16:14:31.385…	FLL	BWI	4319	-4500	points used
13	3	2022-04-30 12:53:07.529…	GCJ	PLZ	1319	-3000	points used
14	3	2022-05-30 16:55:00-05	PDX	MCI	3719	2400	points earned
15	3	2022-06-10 19:50:54.96-05	MCI	PDX	6119	2400	points earned
16	3	2022-06-13 06:11:21-05	ZRH	IAD	12819	6700	points earned
17	3	2022-06-13 16:57:49.590…	NGO	KMJ	10719	-2100	points used

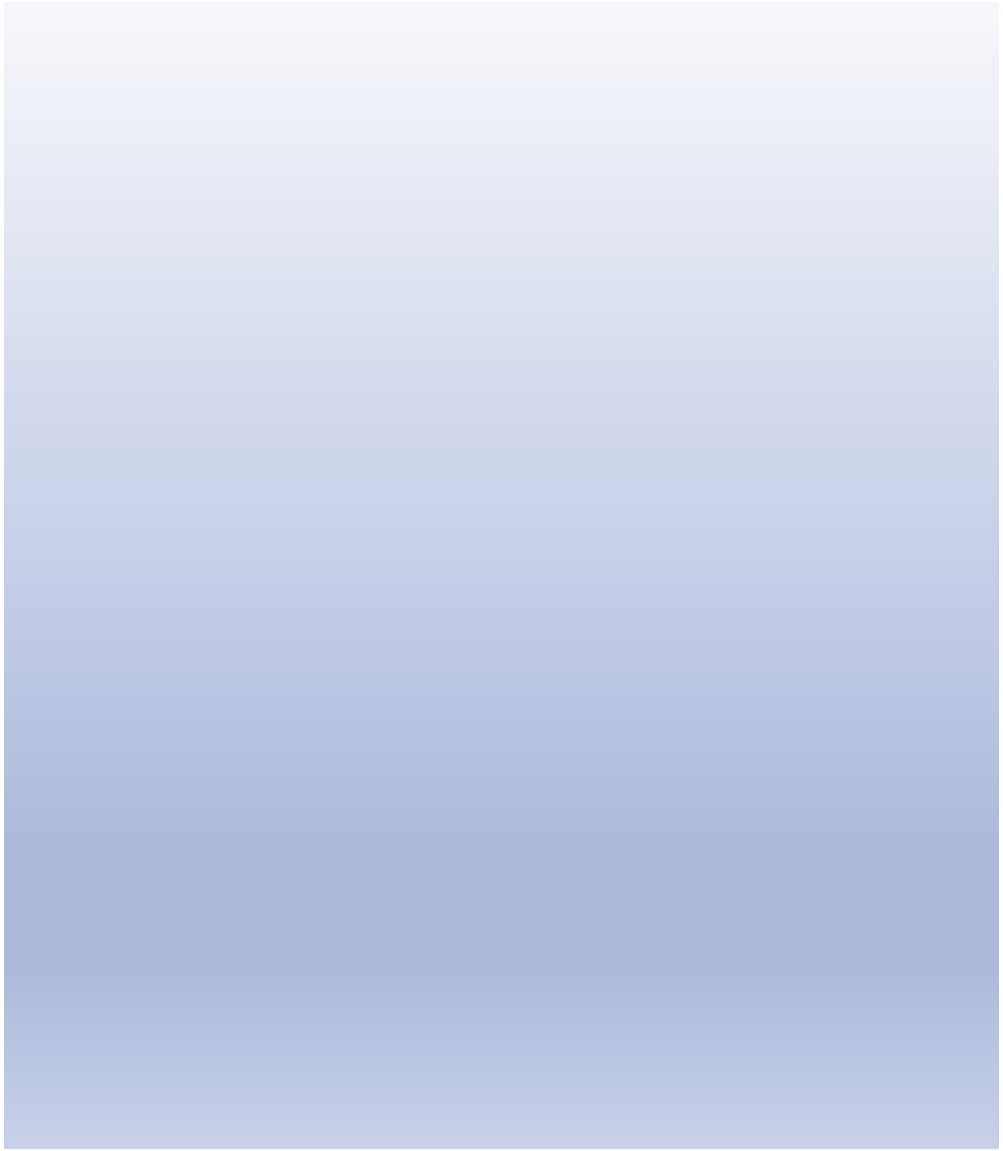
```
select                                                  100 ms
    frequent_flyer_id,
     time_of_change,
     departure_airport,
    arrival_airport,
     current_points_balance,
     points_balance_change,
     activity
from award_points a
join postgres_air.booking_leg bl using (booking_leg_id)
join postgres_air.flight f
using (flight_id)
where a.frequent_flyer_id=36966
order by 1,2;
```

Data Output | Messages | Notifications

	frequent_flyer_id integer	time_of_change timestamp with time zone	departure_airport character (3)	arrival_airport character (3)	current_points_balance integer	points_balance_change integer	activity text
	36966	2022-03-13 05:03:36.0258-05	DUB	ORD	7803	-17700	points used
2	36966	2022-04-19 16:58:19.942274-...	IAD	DEN	603	-7200	points used
3	36966	2022-06-02 16:40:00-05	ORD	ROC	1503	900	points earn...
4	36966	2022-06-09 11:17:20.447413-...	ORD	GRR	603	-900	points used
5	36966	2022-06-11 13:15:56.76-05	ROC	ORD	1503	900	points earn...
6	36966	2022-06-11 23:45:00-05	ORD	DUB	7403	5900	points earn...
7	36966	2022-07-07 19:00:00-05	DEN	IAD	9803	2400	points earn...
8	36966	2022-07-14 04:11:18.7315-05	AYT	DUS	2303	-7500	points used

And you can do it for any attribute history!

Change Log Solution



Change Log Tables

AWARD_EVENT_LOG – 1.2M	
frequent_flyer_id	int
time_of_change	timestamptz
booking_leg_id	int
next_award_points_balance	int
prev_award_points_balance	int
activity	text

STATUS_EVENT_LOG - 1.2 M	
frequent_flyer_id	int
time_of_change	timestamptz
booking_leg_id	int
next_status_points_balance	int
prev_status_points_balance	int
activity	text

LEVEL_EVENT_LOG – 9K	
frequent_flyer_id	int
time_of_change	timestamptz
booking_leg_id	int
next_level	text
previous_level	text
activity	text

```
with
award_point_period as (
select * from (select
   frequent_flyer_id,
   time_of_change,
   booking_leg_id,
   next_award_points_balance,
   prev_award_points_balance,
   award_points_balance_change,
   activity, ---'used' or 'booked'
   time_of_change as expi_ts,
   lag (   time_of_change,1, '-infinity'::timestamptz) over
   (partition by frequent_flyer_id order by time_of_change) as effe_ts
from award_event_log) a
      where  '07-01-2022' >= effe_ts
   and  '07-01-2022' <   expi_ts
),
```

```
status_point_period as (
select * from (select
   frequent_flyer_id,
   time_of_change,
   booking_leg_id,
   next_status_points_balance,
   prev_status_points_balance,
   status_points_balance_change,
   activity, ---'used', 'reset'
   time_of_change as expi_ts,
   lag (   time_of_change,1, '-infinity') over (partition by
frequent_flyer_id order by time_of_change) as effe_ts
from status_event_log) a
      where  '07-01-2022' >= effe_ts
   and  '07-01-2022' <   expi_ts
)
```

```
level_period as (
select * from (select
   frequent_flyer_id,
   time_of_change,
   booking_leg_id,
   next_level,
   previous_level,
   activity, ---'used', 'reset'
   time_of_change as expi_ts,
   lag (   time_of_change,1, '-infinity') over
 (partition by frequent_flyer_id order by time_of_change)
       as effe_ts
from level_event_log)a
      where  '07-01-2022' >= effe_ts
   and  '07-01-2022' <   expi_ts
)
```

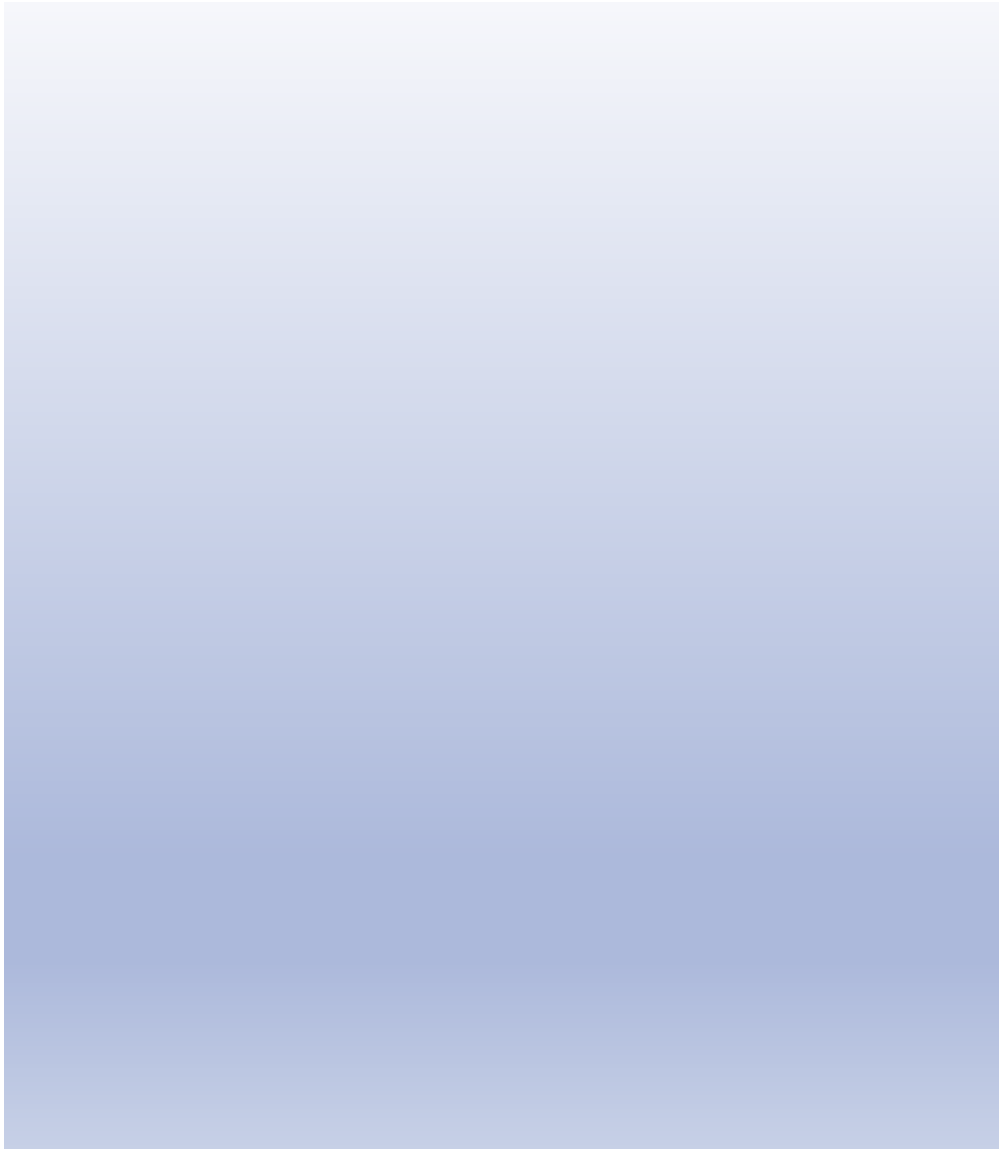
```
select f.frequent_flyer_id ,
coalesce(a.booking_leg_id, s.booking_leg_id, l.booking_leg_id,f.booking_leg_id)
 as booking_leg_id,
coalesce(a.time_of_change, s.time_of_change, l.time_of_change,f.time_of_change)
 as booking_leg_id,
coalesce(a.activity, s.activity, l.activity,'initial') as activity,
coalesce   (a.prev_award_points_balance, f.award_points) as award_points_balance,
coalesce   (s.prev_status_points_balance, f.status_points) as
 status_points_balance,
coalesce (l.previous_level, f.level) as level
from frequent_flyer f
 left join award_point_period a
    on f.frequent_flyer_id = a.frequent_flyer_id
left join status_point_period s
on f.frequent_flyer_id = s.frequent_flyer_id
left join level_period l
on f.frequent_flyer_id = l.frequent_flyer_id
```

1.8 sec

Data Output Messages Notifications

	frequent_flyer_id integer	booking_leg_id integer	booking_leg_id timestamp with time zone	activity text	award_points_balance integer	status_points_balance integer	level text
1	2	11291181	2022-07-05 23:04:22.147…	points us…	18134	42134	1
2	3	2800201	2022-07-02 20:50:00-05	points ear…	22219	43819	2
3	5	3303535	2022-07-06 13:00:06.12-05	points ear…	9581	103781	2
4	6	10762296	2022-07-01 20:21:16.251…	points us…	22133	0	2
5	7	3624020	2022-07-01 15:00:00-05	points ear…	30375	69675	1
6	10	3350101	2022-07-01 10:30:00-05	points ear…	12003	99003	2
7	14	11762691	2022-07-06 20:24:10.467…	points us…	25674	92274	1
8	15	3288269	2022-07-05 22:10:07.8-05	points ear…	5770	78670	2
9	16	7470070	2022-07-27 20:00:00-05	points ear…	3508	9208	1
10	17	3234493	2022-07-03 16:35:43.08-05	points ear…	15577	62977	2
11	18	10981458	2022-07-03 15:33:14.722…	points us…	28585	0	2
12	19	11061351	2022-07-04 12:27:31.442…	points us…	60587	129287	2

And if you want to build a generic case…



```
with
award_point_peri as (
select
   frequent_flyer_id,
   booking_leg_id,
   next_award_points_balance,
   prev_award_points_balance,
   award_points_balance_change,
   activity, ---'used' or 'booked'
   time_of_change as expi_ts,
   lag (   time_of_change,1, '-infinity'::timestamptz) over (partition by
frequent_flyer_id order by time_of_change) as effe_ts
from award_event_log
),
```

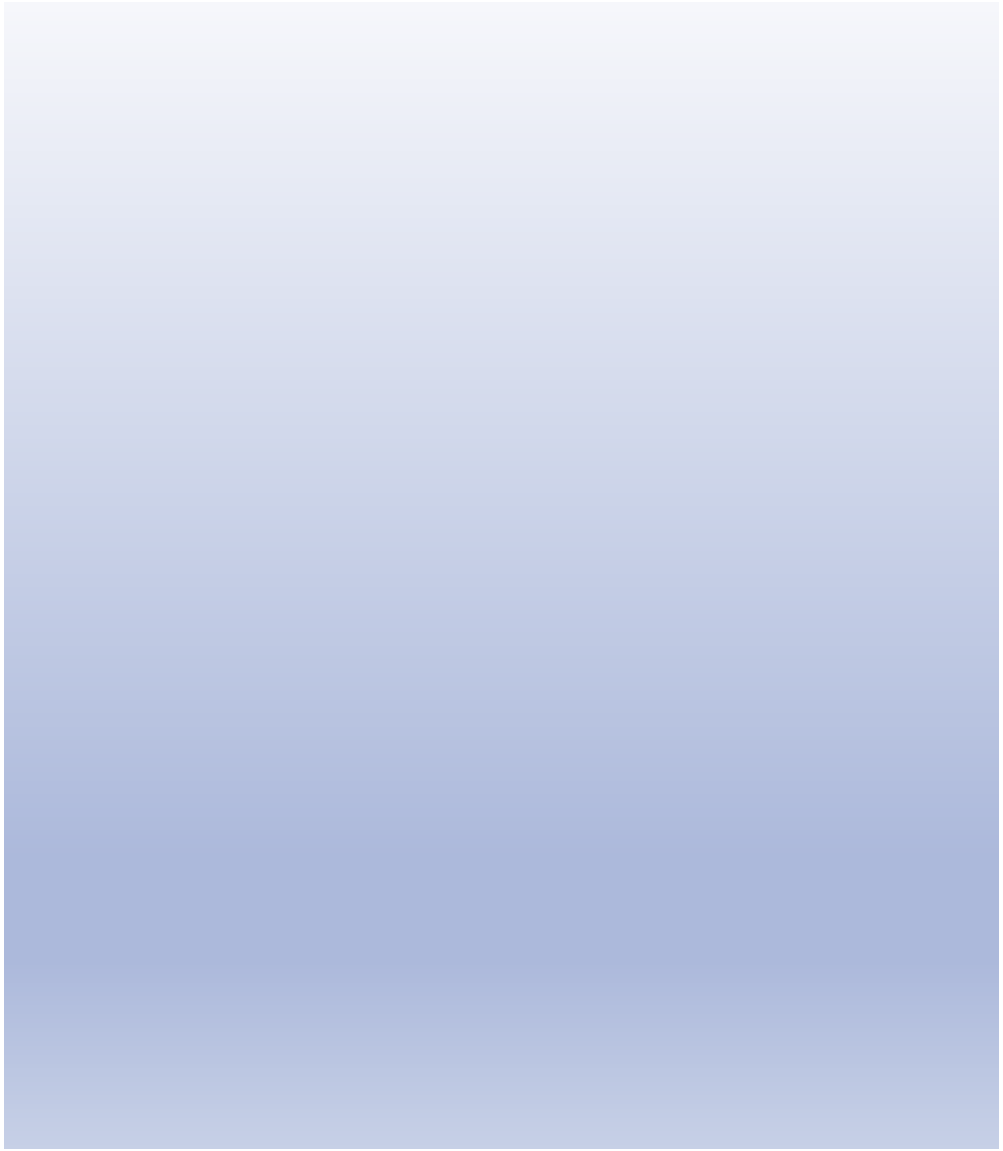
```
award_point_period as (
select * from  award_point_peri
union all
select
   frequent_flyer_id,
   null,
   null,
   null,
   null,
   null, ---'used' or 'booked'
   'infinity'::timestamptz  as expi_ts,
   max(expi_ts) as effe_ts
from  award_point_peri  group by frequent_flyer_id
),
```

```
frequent_flyer_asl_period as(
select
 f.frequent_flyer_id ,
coalesce(l.booking_leg_id, f.booking_leg_id) as booking_leg_id,
coalesce(l.activity, f.activity) as activity,
f.award_points_balance,
f.status_points_balance as status_points_balance,
coalesce(l.next_level, f.level) as level,
greatest (f.effe_ts, coalesce(l.effe_ts,'-infinity')) as effe_ts,
least (f.expi_ts,  coalesce(l.expi_ts, 'infinity')) as expi_ts
from frequ_flyer_as_period f
left join level_period l
on f.frequent_flyer_id = l.frequent_flyer_id
and (f.effe_ts,f.expi_ts) overlaps (l.effe_ts, l.expi_ts)
)
```

```
select *
from frequent_flyer_asl_period
where  '07-01-2022' >= effe_ts
   and  '07-01-2022' <   expi_ts
;
```

8 sec

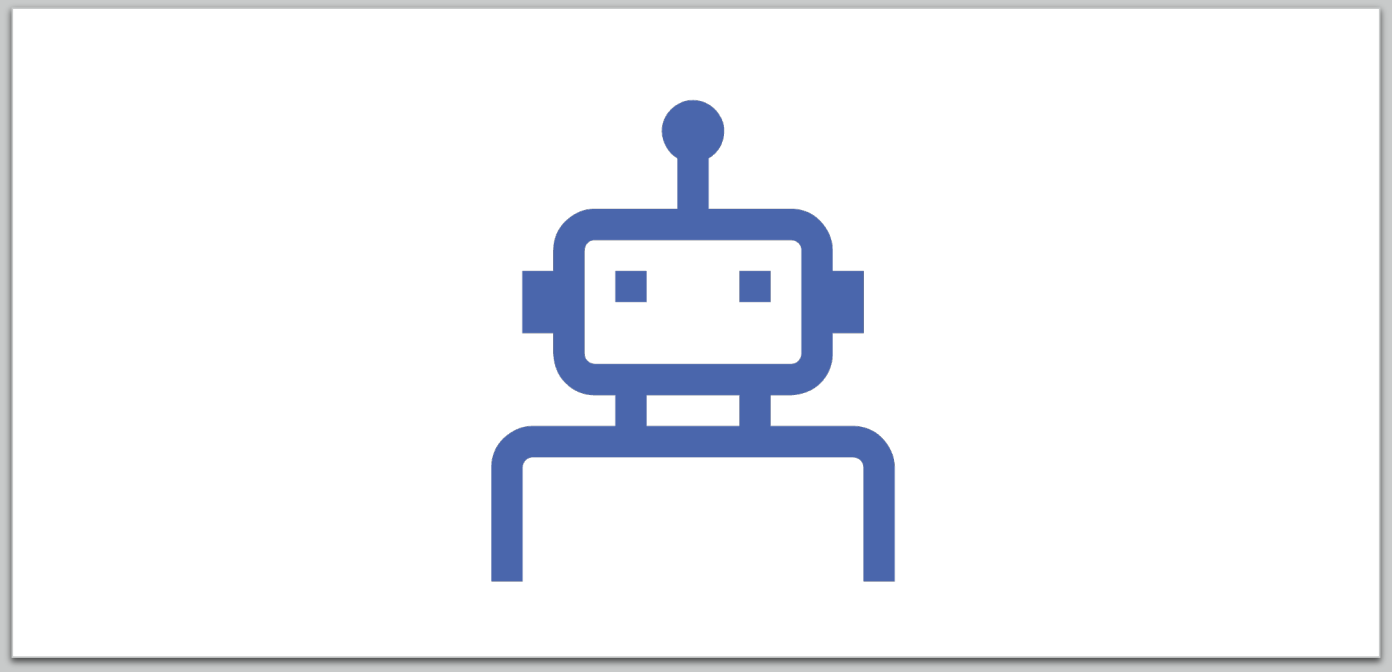
Conclusion



Comparison

Bitemporal model:

- Less space (more attributes => more space saving)
- Easier querying at point-in time (Stonebreaker)
- No extra costs for building individual change log
- Significantly less cost of building relative time-dependent queries
- Support of correction



Give it a try!
https://github.com/hettie-d/pg_bitemporal

Thank you!

PG Day Chicago Apr 20 2023!