Partitioning,
... back to the core

DECATHLON RETAIL IT

Thomas BOUSSEKEY

2018 PgConf.eu LISBON

Once upon a time...https://goo.gl/UFF1iF

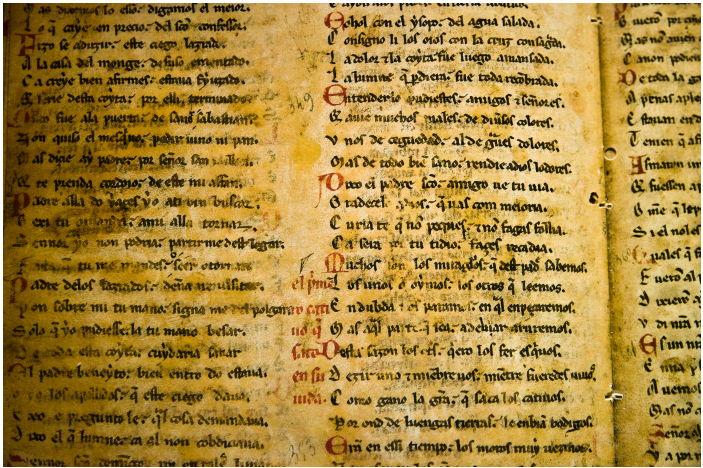


Image: @ Unsplash

Once upon a time...https://goo.gl/UFF1iF

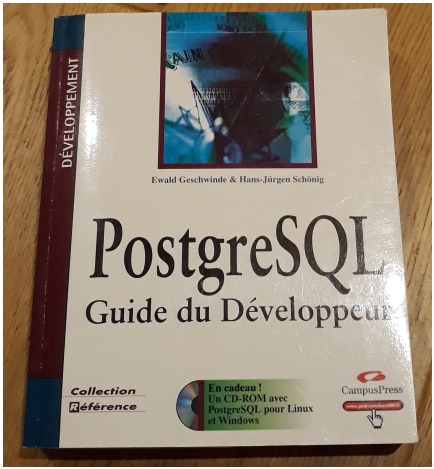
Image: @ Unsplash

DECATHLON

Once upon a time...https://goo.gl/UFF1iF

Image: @ Unsplash

Once upon a time...https://goo.gl/UFF1iF



... bought in 2002, but only tried 4 hours, to make PostgreSQL 7.1 work

DECATHLON

Agenda

1. Set the landscape
2. Dive into current settings
3. New technical target for PG 11

Next Section:

1. Set the landscape
2. Dive into current settings
3. New technical target for PG 11

Next Subsection:

1. Set the landscape
   - My introduction
   - Let's talk about DECATHLON
   - Describing POSDATA

My background

SELECT * FROM pro.me;

- DBA since 2001
  - MSSQL → ORACLE → PostgreSQL
  - Using PostgreSQL since 2012 using version **9.2**
- OPS @ Feature team
  - 12 years on CRM
  - Since 2016 on cashing system backoffice solutions

SELECT * FROM perso.me;

- docker-omnidb

My background

SELECT * FROM pro.me;

- DBA since 2001
  - MSSQL → ORACLE → PostgreSQL
  - Using PostgreSQL since 2012 using version **9.2**
- OPS @ Feature team
  - 12 years on CRM
  - Since 2016 on cashing system backoffice solutions

SELECT * FROM perso.me;

- docker-omnidb

DECATHLON

DECATHLON

Next Subsection:

1. Set the landscape
   - My introduction
   - Let's talk about DECATHLON
   - Describing POSDATA

DECATHLON – Activity

Slogan

"Make sports accessible for the many"

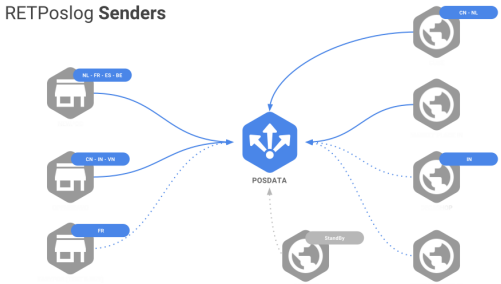
DECATHLON – some figures

- 94.000+ employees
  - **RETAIL:** 1412 stores in 43 countries
  - **RETAIL:** 66 warehouses + production in 45 countries
- 11 billion € of turnover in 2017
- IT Department
  - 850 employees
  - 1.100 external partners
  - 100+ open positions

Next Subsection:

1. Set the landscape
   - My introduction
   - Let's talk about DECATHLON
   - Describing POSDATA

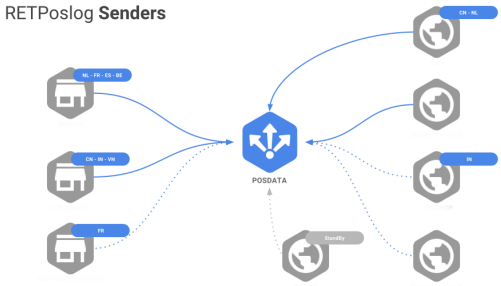
Application's aim



Usage

- Collect
  - 5 e-commerce CMS
  - 3 distinct Cashing systems
- Store
  - Into One application
  - ... on 3 continents
  - Using POSLOG XML format
- Distribute
  - Realtime (or so)
  - Offering API
  - ... with back in time solution

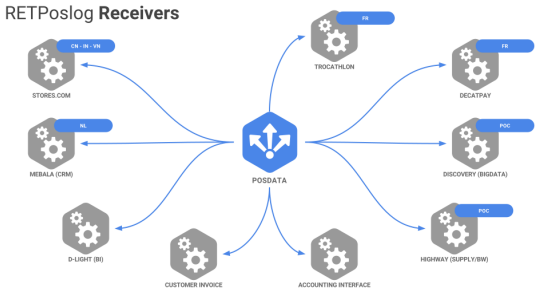
Application's aim



Usage

- Collect
  - 5 e-commerce CMS
  - 3 distinct Cashing systems
- **Store**
  - Into One application
  - ... on 3 continents
  - Using POSLOG XML format
- Distribute
  - Realtime (or so)
  - Offering API
  - ... with back in time solution

Application's aim



Usage

- Collect
  - 5 e-commerce CMS
  - 3 distinct Cashing systems
- **Store**
  - Into One application
  - ... on 3 continents
  - Using POSLOG XML format
- Distribute
  - Realtime (or so)
  - Offering API
  - ... with back in time solution

Next Section:

1. Set the landscape
2. Dive into current settings
   - Architecture – 2016's choice
   - Additionnal tooling
   - 2018 feedback
3. New technical target for PG 11

DECATHLON

Next Subsection:

DECATHLON

Start in production

- Using *PostgreSQL*9.5
  - with *pg_partman*2.6 partitioning extension
  - and some object to manage it!

User's autonomy

- Automation of partition creation
- Rundeck job using a PL/PgSQL function

Start in production

- Using *PostgreSQL*9.5
  - with *pg_partman*2.6 partitioning extension
  - and some object to manage it!

User's autonomy

- Automation of partition creation
- Rundeck job using a PL/PgSQL function

Start in production

- Using *PostgreSQL*9.5
  - with *pg_partman*2.6 partitioning extension
  - and some object to manage it!

User's autonomy

- Automation of partition creation
- Rundeck job using a PL/PgSQL function

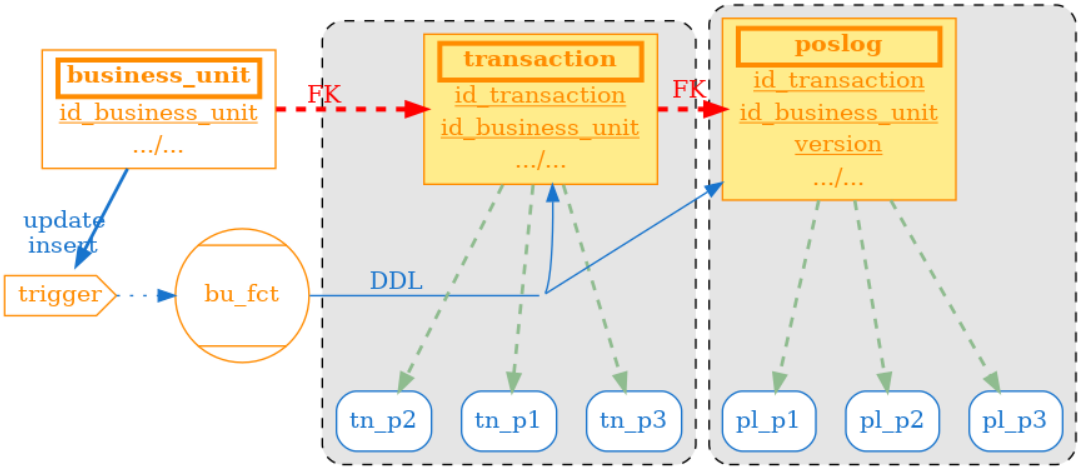
Define the partition column

- It needs to be unique
- Business units are defined with 3 distinct number

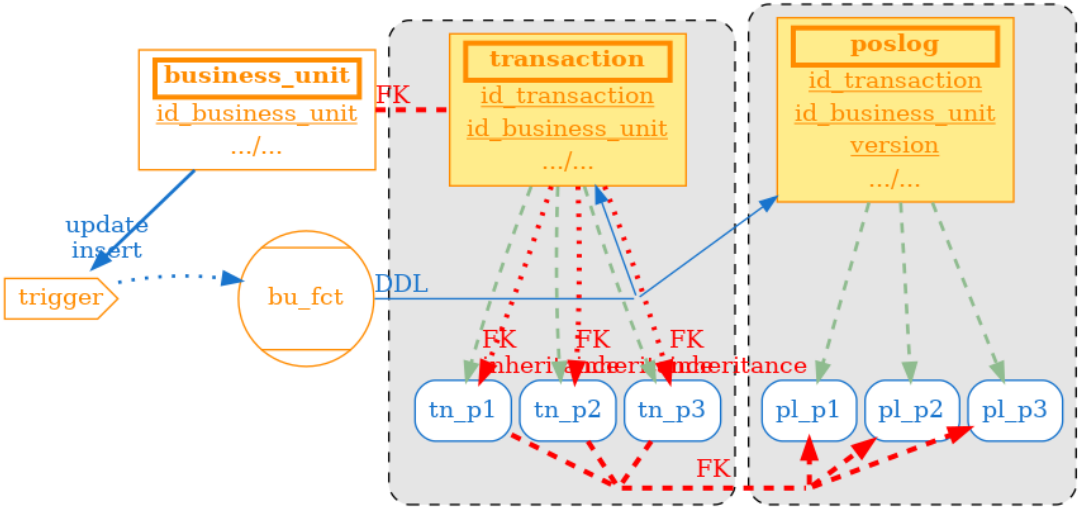
Solution found!!!

The 3 numbers were concatened by 5-digits
into a *BIGINT* column named *id_business_unit*

Data structure, ideallistic view



Data structure, realistic view



Next Subsection:

DECATHLON

Ops tooling

Index creation on all the partitions

- BRIN index, in order to identify the previous maximum

Massive data management

- Procedure to extract all flows received on a day
- GDPR dB-model migration (using Python psycopg2)

DECATHLON

Functionnal tooling

Store activation

Get a function that creates all the partitions for the business unit to implement.

Post-treatment management

New functionnal need, generate a batch to extract the value for a new column, awaiting that the integration process manages it.

Reporting

- Report of last received flows for activated business units
- Flows received per month on a specific cashing solution
- Flows received per hour and per country for TICK monitoring solution

DECATHLON

Next Subsection:

DECATHLON

Feedback after 2 years in production

Good points

- Robust solution
- Good data spreading
- Nice inheritance features
  - Column adding

To improve

- Need better volumetry dispatch
- Tricky maintenance operation
- 3 major releases of PostgreSQL released since 2016 & Pg9.5
- Need better inheritance features
  - index inheritance

Feedback after 2 years in production

Good points

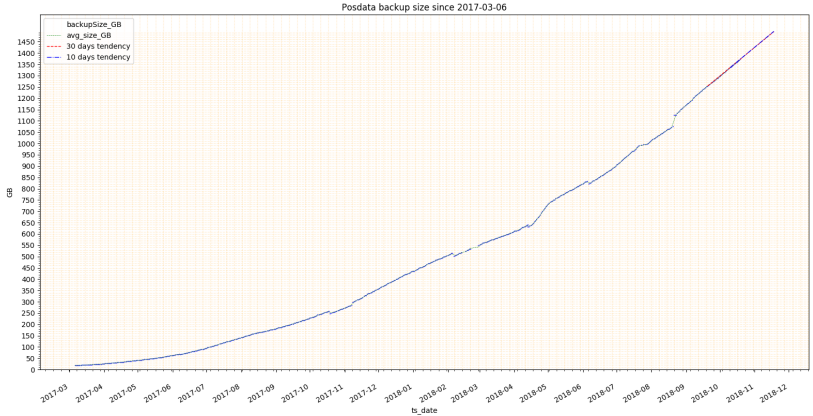
- Robust solution
- Good data spreading
- Nice inheritance features
  - Column adding

To improve

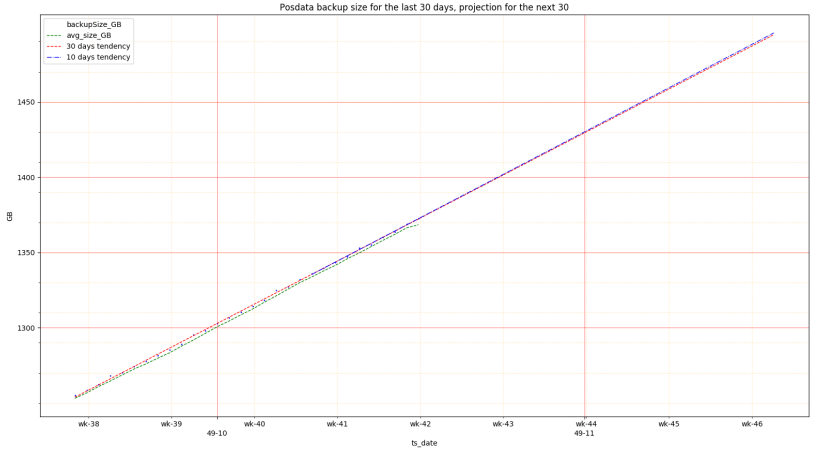
- Need better volumetry dispatch
- Tricky maintenance operation
- 3 major releases of PostgreSQL released since 2016 & Pg9.5
- Need better inheritance features
  - index inheritance

DECATHLON

Size evolution last 18 months

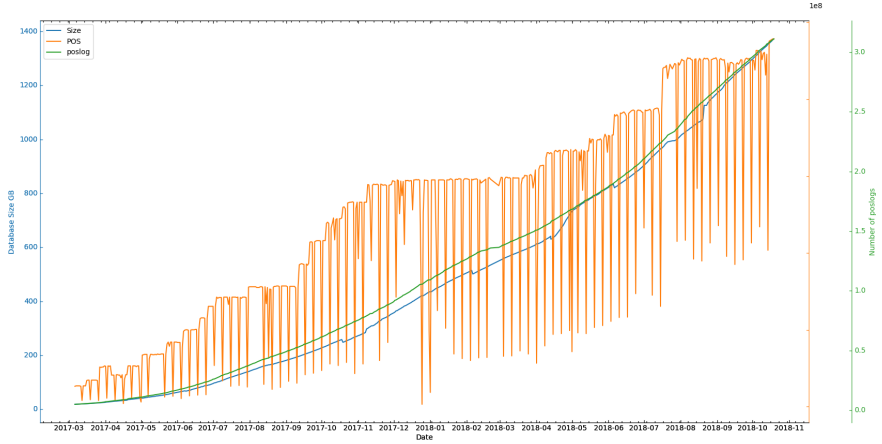


Capacity planning 30 days



DECATHLON

Still increasing



DECATHLON

Still increasing

- Still
  - 100+ european stores
  - some e-commerce web sites
- to deploy on the solution

Next Section:

1. Set the landscape
2. Dive into current settings
3. New technical target for PG 11
   - Partitioning possibilities
   - Subpartitioning
   - Tests configuration
   - Results obtained

DECATHLON

Next Subsection:

DECATHLON

Partitioning, yes but...

Overview

Positive impact

- When most of the heavily accessed rows of the table are in a single partition or a small number of partitions.
- When queries or updates access a large percentage of a single partition.
- Bulk loads and deletes can be accomplished by adding or removing partitions. These commands also entirely avoid the VACUUM overhead caused by a bulk DELETE.
- Seldom-used data can be migrated to cheaper and slower storage media.

Partitioning, but when...

When should I consider partitioning a table?

- The benefits will normally be worthwhile only when a table would otherwise be very large.
- The exact point at which a table will benefit from partitioning depends on the application,
- ...although a rule of thumb is that the size of the table should exceed the physical memory of the database server.

DECATHLON

Partitioning, but when...

When should I consider partitioning a table?

- The benefits will normally be worthwhile only when a table would otherwise be very large.
- The exact point at which a table will benefit from partitioning depends on the application,
- ...although a rule of thumb is that the size of the table should exceed the physical memory of the database server.

DECATHLON

Pg_partman from 2.6 to 4.0

- Implementation of index and FK inheritance
- Support for native partitioning
- V4 released on October 11, for PG 11 support
  - Change python script to PROCEDURE
  - Security changes for functions

Release notes:
https://raw.githubusercontent.com/pgpartman/pg_partman/master/CHANGELOG.txt

DECATHLON

Pg_partman implementation

1. Activate extension pg_partman

```
CREATE EXTENSION IF NOT EXISTS pg_partman
        WITH SCHEMA partman;
```

2. Create root table

```
CREATE TABLE posdata.transaction (column_name column_def...);
```

DECATHLON

Pg_partman implementation

3. Activate partitioning for root table

```
SELECT partman.create_parent(
    p_parent_table := 'posdata.transaction',
    p_type := 'partman',
    p_control := 'id_business_unit',
    p_interval := '10',
    p_premake := 1,
    p_start_partition := '1');
```

4. Create level-1 table

```
SELECT partman.create_partition_id(
    p_parent_table := 'posdata.transaction'::text,
    p_partition := array[(id_business_unit)::bigint]);
```

DECATHLON

Pg_pathman from 0.4 to 1.5.2

- Partial support of native partitioning
- multilevel-partitioning
- Support for PG 11
- Integration with pg_shardman
- Partitioning by:
  - expression
  - composite key

Release notes: https://github.com/postgrespro/pg_pathman/releases

Pg_pathman implementation

1. Activate extension pg_pathman

```
CREATE EXTENSION IF NOT EXISTS pg_pathman
        WITH SCHEMA pathman;
```

2. Create root table

```
CREATE TABLE posdata.transaction (column_name column_def...);
```

Pg_pathman implementation

3. Activate partitioning for root table

```
SELECT pathman.create_range_partitions(
    parent_relid := 'posdata.transaction'::regclass,
    expression := 'id_business_unit',
    start_value := 1,
    p_interval := 10,
    p_count := 10);
```

4. Create level-1 table

```
SELECT pathman.add_range_partition(
    parent_relid := 'posdata.transaction'::text,
    start_value := id_business_unit::bigint,
    end_value := id_business_unit::bigint + 1,
    partition_name := 'posdata.transaction_p' || id_business_unit::varc
```

DECATHLON

Native partitioning, birth in PG10

- First version of native partitioning
  - ID-partitioning
  - TIME-partitioning
- Prevent inheritance + trigger tooling
- Better performance due to low-level procedures

Native partitioning, evolutions in PG11

- indexing, PK, FK
- Hash partitioning
- default partition
- on conflict
- update keys

Native partitioning implementation

1. Create root table

```
CREATE TABLE posdata.transaction (column_name column_def...)
        PARTITION BY LIST (id_business_unit);
```

2. Create level-1 table

```
CREATE TABLE posdata.transaction_p700001000010
    PARTITION OF posdata.transaction FOR VALUES in (700001000010);
```

Native partitioning, partmgr tooling

- Tooling for time-partioning management.
- Offer packaging for standard operations that will generate SQL code for you.

Sum up, partitioning solutions / keys

PG version	Native PG 10	Native PG 11	Partman	Pathman
range	✓	✓	✓	✓
list	✓	✓	✗	✗
hash	✗	✓	✗	✓
composite	✗	✓	✓	✓
expression	✓	✓	✓	✓

Table: Table to sum up partitioning techniques on keys

Sum up, partitioning solutions / related objects

PG version	Native PG 10	Native PG 11	Partman	Pathman
PK	✗	✓	✓	✓
FK	✗	✗	✗	☐
index	✗	✓	✗	☐
upsert	✗	✓	✗	☐

Table: Table to sum up partitioning techniques

Nice article to share with you

Feature	PG9.6	PG10	PG11
Declarative Partitioning	✗	✓	✓
Auto Tuple Routing – INSERT	✗	✓	✓
Auto Tuple Routing – UPDATE	✗	✗	✓
Optimizer Partition Elimination	✓[1]	✓[1]	✓
Executor Partition Elimination	✗	✗	✓[2]
Foreign keys	✗	✗	✓[3]
Unique indexes	✗	✗	✓[4]
Default Partitions	✗	✗	✓
Hash Partitions	✗	✗	✓
FOR EACH ROW triggers	✗	✗	✓
Partition-level joins	✗	✗	✓[5]
Partition-level aggregation	✗	✗	✓
Foreign partitions	✗	✗	✓
Parallel Partition Scans	✗	✗	✓

1. Using constraint exclusion
2. Append nodes only
3. On partitioned table referencing non-partitioned table only
4. Indexes must contain all partition key columns
5. Partition constraint on both sides must match exactly

https://blog.2ndquadrant.com/partitioning-evolution-postgresql-11/

Next Subsection:

DECATHLON

Subpartitioning – introduction

Disclaimer

- Performed on PostgreSQL 11 beta4
- Only implemented on native partitioning
- ... but also exists on other partitioning solutions

Why subpartitioning?

Why subpartitioning?

- Limit size per object
- Improve performance
- Largest partition is over 23 GB and largest are coming

Subpartitioning implementation

1. Create root table

```
CREATE TABLE posdata.transaction (column_name column_def...);
```

2. Create level-1 table

```
CREATE TABLE posdata.transaction_p700001000010
    PARTITION OF posdata.transaction FOR VALUES in (700001000010)
    PARTITION BY range (transaction_date);
```

3. Create level-2 table

```
CREATE TABLE posdata.transaction_p700001000010_2018Q4
    PARTITION OF posdata.transaction_p700001000010
        FOR VALUES from ('2018-10-01') to  ('2019-01-01');
```

DECATHLON

First impressions

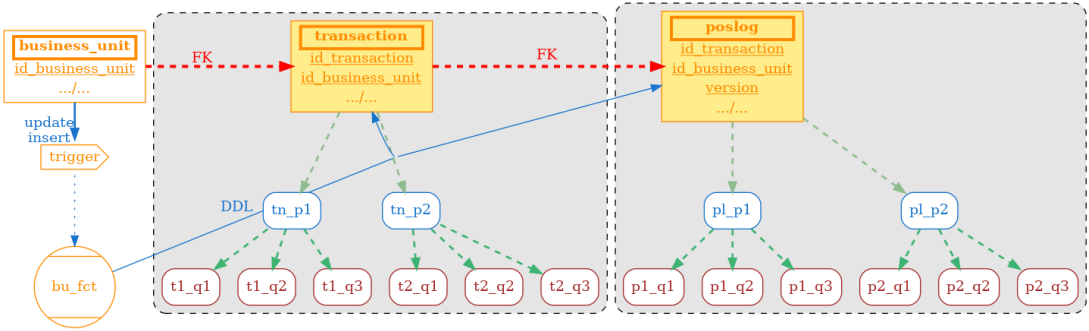
- From 1407 to 7770 partitions +500% number of objects
- Not enough locks!!

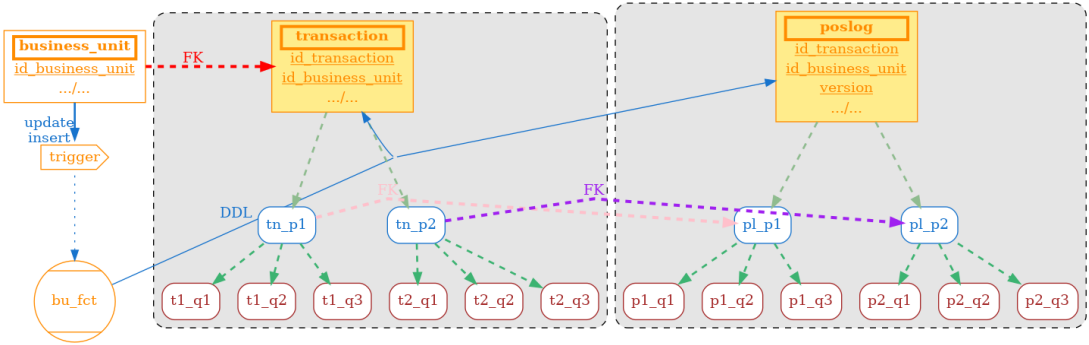
```
sed -i \
's/^max_locks_per_transaction= 512/max_locks_per_transaction = 1024/g'
*/inst_5*2/data/postgresql.conf
```

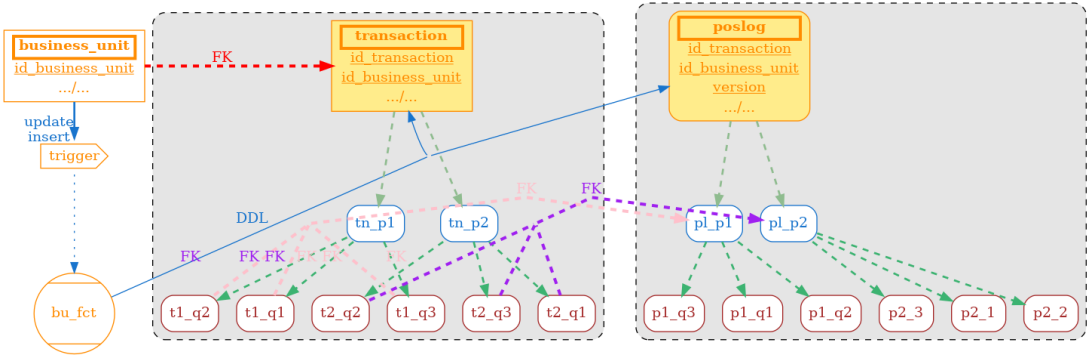
First impressions

- From 1407 to 7770 partitions +500% number of objects
- Not enough locks!!

```
sed -i \
's/^max_locks_per_transaction= 512/max_locks_per_transaction = 1024/g'
*/inst_5*2/data/postgresql.conf
```


business_unit
id_business_unit
.../...
update
insert
trigger
bu_fct
FK
transaction
id_transaction
id_business_unit
.../...
FK
poslog
id_transaction
id_business_unit
version
.../...
DDL
tn_p1
tn_p2
t1_q1
t1_q2
t1_q3
t2_q1
t2_q2
t2_q3
pl_p1
pl_p2
p1_q1
p1_q2
p1_q3
p2_q1
p2_q2
p2_q3


business_unit
id_business_unit
.../...
FK
transaction
id_transaction
id_business_unit
.../...
poslog
id_transaction
id_business_unit
version
.../...
update
insert
trigger
DDL
FK
FK
tn_p1
tn_p2
pl_p1
pl_p2
bu_fct
t1_q1
t1_q2
t1_q3
t2_q1
t2_q2
t2_q3
p1_q1
p1_q2
p1_q3
p2_q1
p2_q2
p2_q3


business_unit
id_business_unit
.../...
update
insert
trigger
bu_fct
FK
transaction
id_transaction
id_business_unit
.../...
DDL
FK
FK FK
FK
FK
tn_p1
tn_p2
t1_q2
t1_q1
t2_q2
t1_q3
t2_q3
t2_q1
poslog
id_transaction
id_business_unit
version
.../...
pl_p1
pl_p2
p1_q3
p1_q1
p1_q2
p2_3
p2_1
p2_2

Next Subsection:

DECATHLON

preamble

Disclaimer

Performed on PostgreSQL 11 beta4

BREAKING NEWS!!!

- Postgresql 11 RC1 released on October, 11 th
- Postgresql 11 GA released on October, 18 th
- Fix some problems and improve performance
  - Assign constraint name when cloning FK definition for partitions
  - Fix duplicate primary keys in partitions

... and for your own database??

Rely on your own tests!

DECATHLON

preamble

Disclaimer

Performed on PostgreSQL 11 beta4

BREAKING NEWS!!!

- Postgresql 11 RC1 released on October, 11 th
- Postgresql 11 GA released on October, 18 th
- Fix some problems and improve performance
  - Assign constraint name when cloning FK definition for partitions
  - Fix duplicate primary keys in partitions

... and for your own database??

Rely on your own tests!

DECATHLON

preamble

Disclaimer

Performed on PostgreSQL 11 beta4

BREAKING NEWS!!!

- Postgresql 11 RC1 released on October, 11 th
- Postgresql 11 GA released on October, 18 th
- Fix some problems and improve performance
  - Assign constraint name when cloning FK definition for partitions
  - Fix duplicate primary keys in partitions

... and for your own database??

Rely on your own tests!

DECATHLON

Test instances

PG version	Native	Partman	Pathman
9.5	✗	✓	✗
10	✓	✓	✓
11	✓	☐	✗

Table: Table to list tested configurations

PostgreSQL configuration - systemd

```
# Note: do not use a PGDATA pathname containing spaces, or you will break postgresql-setup.
[Unit]
Description=PostgreSQL 11 database server
Documentation=https://www.postgresql.org/docs/11/static/
After=syslog.target
After=network.target
[Service]
Type=notify
User=postgres
Group=postgres
# Note: avoid inserting whitespace in these Environment= lines, or you may
# break postgresql-setup.
# Location of database directory
Environment=PGDATA=/home/pgsql/11/inst_5432/data

# Where to send early-startup messages from the server (before the logging
# options of postgresql.conf take effect)
# This is normally controlled by the global default set by systemd
# StandardOutput=syslog

# Disable OOM kill on the postmaster
```

DECATHLON

PostgreSQL configuration - postgresql.conf

```
port = 54xx
shared_buffers = 512M
effective_cache_size = 1024M
maintenance_work_mem = 128MB
max_locks_per_transaction = 512
max_pred_locks_per_transaction = 256
# If needed
shared_preload_library = '...'
```

Which data have been used into the test?

- Get a realistic representation of production data
- ... but light enough to get 6 copies on my laptop
- Use of the *modulo* operator

```
create schema miniexport;
create table miniexport.transaction
    as select * from v2.transaction where id_transaction %20=1;
create table miniexport.poslog
    as select * from v2.poslog where id_transaction %20=1;
```

DECATHLON

What about the data?

	transaction	poslog
Insert partition	89.304	90.618
Insert main table	103.484	105.121
Insert many BU	13.591.900	14.361.054

DECATHLON

What about the select tests?

- Get transaction count for a BU

```
SET constraint_exclusion = :PAR_CONST_EXCL;
explain (costs, analyse, buffers, timing)
select count (*)
from :PAR_SCHEMA.transaction
where id_business_unit = 700118001180;
```

What about the select tests?

- Get transaction count for a BU over a week

```
SET constraint_exclusion = :PAR_CONST_EXCL;
explain (costs, analyse, buffers, timing)
select count (*)
from :PAR_SCHEMA.transaction
where id_business_unit = 700118001180
and transaction_date between '2018-08-22' and '2018-08-30';
```

What about the select tests?

- Get one precise transaction + join poslog

```
SET constraint_exclusion = :PAR_CONST_EXCL;
explain (costs, analyse, buffers, timing)
select count (*)
from :PAR_SCHEMA.transaction t
    inner join :PAR_SCHEMA.poslog p
        on t.id_business_unit = p.id_business_unit
        and t.id_transaction = p.id_transaction
where p.id_business_unit = 700118001180
    and p.id_transaction = 4681;
```

What about the select tests?

- Copy into main table (240+ k rows)

```
COPY posdata.poslog (id_poslog, id_transaction, id_business_unit, docum
315954394    315954462    700400004000    7-400-400-20181003114331-10-1
    2018-10-03 11:43:46+00    66070040000010000001822018100311415 1
    2018-10-03 11:43:49.268+00    Flow    naoxz28gl2LglXFwusyKFa8iNl.jA
315954404    315954473    700400004000    7-400-400-20181003114340-9-80
    2018-10-03 11:43:52+00    66070040000009000080062018100311423 0
    2018-10-03 11:43:52.471+00    Flow    ZfqNEY8V6UMzOQW8wRAEWh7KlODPa
```

What about the documentation?

QUIZZ: Do you know the constraint exclusion setting?

https://www.postgresql.org/docs/current/static/ddl-partitioning.html#DDL-PARTITIONING-CONSTRAINT-EXCLUSION

Possible settings

The default (and recommended) setting of constraint_exclusion is neither on nor off, but an intermediate setting called partition, which causes the technique to be applied only to queries that are likely to be working on inheritance partitioned tables. The on setting causes the planner to examine CHECK constraints in all queries, even simple ones that are unlikely to benefit.

What about the documentation?

QUIZZ: Do you know the constraint exclusion setting?

- https://www.postgresql.org/docs/current/static/ddl-partitioning.html#DDL-PARTITIONING-CONSTRAINT-EXCLUSION

Possible settings

The default (and recommended) setting of constraint_exclusion is neither on nor off, but an intermediate setting called partition, which causes the technique to be applied only to queries that are likely to be working on inheritance partitioned tables. The on setting causes the planner to examine CHECK constraints in all queries, even simple ones that are unlikely to benefit.

DECATHLON

Automate with PSQL

- Shell *PSQL* command with parameters

```
psql -f ${_sqlFiles} -v PAR_SCHEMA="${_execSchema}" \
    -v PAR_CONST_EXCL="${_execConstExcl}" \
    -v PAR_INT_FK="${_execIntFK}" \
    -v PAR_INT_MULTIPART="${_execIntMultipart}" \
    --output=${_logFile} --port=${_pgport} \
    --username=posdata --dbname=posdata --port=${_pgport}
```

- Variable usage into *SQL* script

```
\set quoted_myvariable '\'' :PAR_SCHEMA '\''
SELECT posdata.tby_applicative_keys(
    :quoted_myvariable,
    :PAR_INT_FK,
    :PAR_INT_MULTIPART
    );
```

DECATHLON

Next Subsection:

DECATHLON

Funny thing encountered with EXPLAIN on Pg11

Pg10 version

```
                                QUERY PLAN
-----------------------------------------------------------------------
Insert on transaction_p700539005390
Buffers: shared hit=282955 read=2451
->  Index Scan using minixp_trans_pk
Index Cond: (id_business_unit = '7005
Buffers: shared read=1268
Planning time: 0.261 ms
Execution time: 151.118 ms
(7 rows)
```

Pg11 version

```
QUERY PLAN
-----------------------------------
.../...
Buffers: shared hit=891 read=377
->  Bitmap Index Scan on minixp_tra
Index Cond: (id_business_unit = '70
Buffers: shared hit=346
Planning Time: 0.305 ms
Trigger for constraint fk_transacti
Execution Time: 669.566 ms
(12 rows)
```

Average - Time (ms)		PgVersion	Extension	Data				
		9	10				11	
		partman	native		partman	pathman	native	
Test Name	constExcl	1	1	2	1	1	1	2
InsertPoslog_direct1BU	Off	636	1,362		818	1,371	1,324	
	On	426	623	1,803	99	507	542	4,036
	Part	327	515		128	437	665	
InsertPoslog_direct1BU_v2	Off	5,712	1,292		4,927	1,851	1,407	
	On	5,492	1,309	1,224	118	2,041	1,056	1,571
	Part	5,432	1,735		447	957	924	
InsertPoslog_global1BU	Off	6,688	1,537	0	6,077	1,279	1,129	0
	On	6,276	774	5,078	196	1,031	698	2,754
	Part	6,107	693	0	703	592	735	0
InsertPoslog_global1BU_v2	Off	6,826	1,456	0	5,421	1,469	2,660	0
	On	6,332	1,080	5,366	150	2,196	1,054	3,544
	Part	6,622	1,792	0	1,047	787	1,166	0
InsertPoslog_manyBU	Off	916,545	280,921	0	867,661	217,055	318,901	0
	On	843,657	205,952	461,726	76,701	177,334	198,201	566,561
	Part	814,112	193,037	0	95,416	179,415	217,941	0
InsertPoslog_manyBU_v2	Off	899,220	231,287	0	772,757	213,188	263,828	0
	On	832,085	197,974	321,203	91,125	187,381	233,795	928,082
	Part	854,060	181,411	0	101,212	192,867	231,689	0
InsertTransaction_direct1BU	Off	526	162		757	682	739	
	On	398	237	839	73	406	388	1,924
	Part	376	150		75	393	380	

DECATHLON

Average - Global cost		PgVersion	Extension	Data				
		9	10				11	
		partman	native		partman	pathman	native	
Test Name	constExcl	1	1	2	1	1	1	2
InsertPoslog_direct1BU	Off	146,042	151,711		138,145	144,626	137,166	
	On	148,292	146,544	1,357	152,195	145,852	138,246	1,358
	Part	146,345	146,544		152,195	145,852	138,246	
InsertPoslog_direct1BU_v2	Off	150,645	149,005		142,008	145,580	137,472	
	On	149,217	147,456	104,930	153,146	146,759	139,101	95,320
	Part	147,255	147,456		153,146	146,759	139,101	
InsertPoslog_global1BU	Off	171,909	175,283	0	165,135	165,077	175,518	0
	On	183,540	172,383	1,587	156,313	167,714	167,828	1,587
	Part	179,857	172,383	0	156,313	167,714	167,828	0
InsertPoslog_global1BU_v2	Off	177,798	176,496	0	169,987	167,137	172,182	0
	On	184,719	173,480	126,629	157,293	168,777	168,890	124,243
	Part	181,009	173,480	0	157,293	168,777	168,890	0
InsertPoslog_manyBU	Off	543,283	543,276	0	543,261	543,265	543,296	0
	On	543,282	543,236	4,553,759	543,261	543,204	543,308	4,563,319
	Part	543,288	543,236	0	543,261	543,204	543,308	0
InsertPoslog_manyBU_v2	Off	1,523,103	1,522,930	0	1,523,356	1,523,164	1,523,262	0
	On	1,522,970	1,522,915	6,445,044	1,523,376	1,522,985	1,523,358	6,001,209
	Part	1,523,248	1,522,915	0	1,523,376	1,522,985	1,523,358	0
InsertTransaction_direct1BU	Off	124,113	126,162		126,728	121,749	119,695	
	On	126,083	121,754	127,044	114,199	106,457	126,224	119,414
	Part	124,494	121,754		114,199	106,457	126,224	
InsertTransaction_global1BU	Off	135,279	137,561		137,388	129,057	127,748	
	On	134,856	126,138	139,846	142,307	122,759	124,773	129,077
	Part	135,724	126,138		142,307	122,759	124,773	
InsertTransaction_manyBU	Off	918,508	918,410		918,358	998,649	958,623	
	On	918,459	918,597	918,372	918,438	918,779	918,694	918,593
	Part	918,495	918,597		918,438	918,779	918,694	

DECATHLON

Average - Time (ms)		PgVersion	Extension	Data				
		9	10				11	
		partman	native		partman	pathman	native	
Test Name	constExcl	1	1	2	1	1	1	2
CountTransaction_1BU	Off	2,696	3,954	1,508	1,168	15	13	11
	On	16	14	19	92	14	13	14
	Part	16	15	15	102	14	14	12
CountTransaction_1BU_1transaction	Off	4,915	6,668	4,561	5,272	203	6	2
	On	14	11	116	1,525	8	1	5
	Part	10	241	94	1,470	8	5	2
CountTransaction_1BU_1week	Off	11	108	4	56	10	6	1
	On	8	9	6	104	7	6	1
	Part	7	26	4	102	6	113	1

Average - Global cost		PgVersion	Extension	Data				
		9	10				11	
		partman	native		partman	pathman	native	
Test Name	constExcl	1	1	2	1	1	1	2
CountTransaction_1BU	Off	306,701	228,659	200,629	233,770	1,698	2,169	1,468
	On	1,800	1,698	1,406	41,535	1,698	2,031	1,905
	Part	1,800	1,698	1,247	36,417	1,800	2,031	1,468
CountTransaction_1BU_1transaction	Off	403,708	389,740	1,108,755	417,641	1,821	17	120
	On	1,874	1,821	2,047	113,444	1,821	17	150
	Part	1,874	2,687	1,843	109,180	1,821	17	120
CountTransaction_1BU_1week	Off	1,986	1,870	162	23,355	1,870	1,990	165
	On	1,986	1,870	182	47,376	1,870	1,874	213
	Part	1,986	1,870	162	40,219	1,986	1,874	165

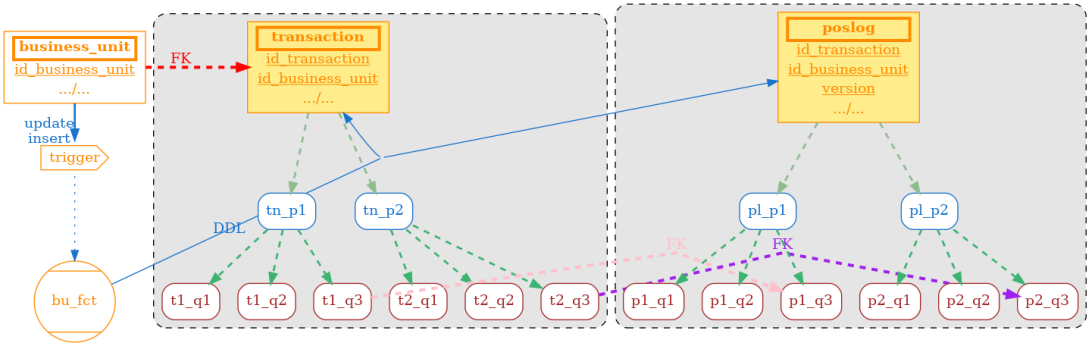
Next tests

- *PostgreSQL11GA*
- 100% of the volumetry
- Autovacuum impact on the tests
- parallel mode RealLife
- ATTACH/DETACH feat

What about subpartitioning?

- Heavy constraint on FK, target configuration:

Implementation of FK



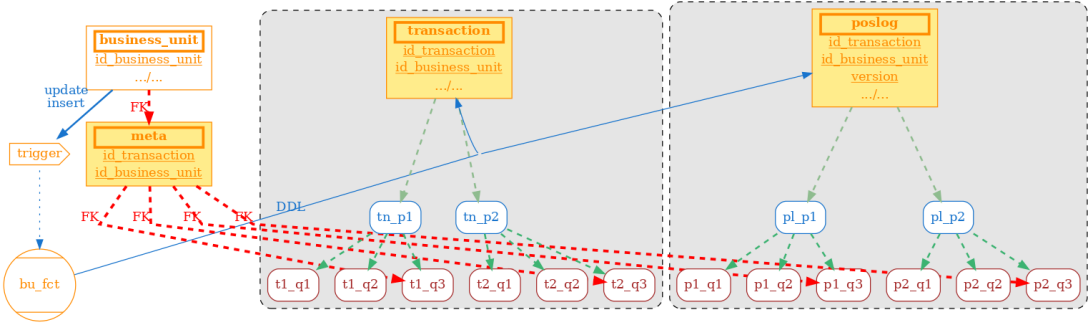
But a functionnal problem happened...

- Poslog related to a transaction that began on previous quarter

But a functionnal problem happened...

- Poslog related to a transaction that began on previous quarter

Possible FK workaround



Conclusion

- Nice improvements on partitioning (and many other features) since PostgreSQL 9.5
- I will prioritize this migration in best possible delays
- and ....
- Should I use subpartitioning??
  - I need to deal with the foreign key issue.
  - Evaluation the impact, if 1400+ subpartitions are created each quarter

DECATHLON

Conclusion

- Nice improvements on partitioning (and many other features) since PostgreSQL 9.5
- I will prioritize this migration in best possible delays
- and ....
- Should I use subpartitioning??
  - I need to deal with the foreign key issue.
  - Evaluation the impact, if 1400+ subpartitions are created each quarter

DECATHLON

Thanks for your attention

Any question?