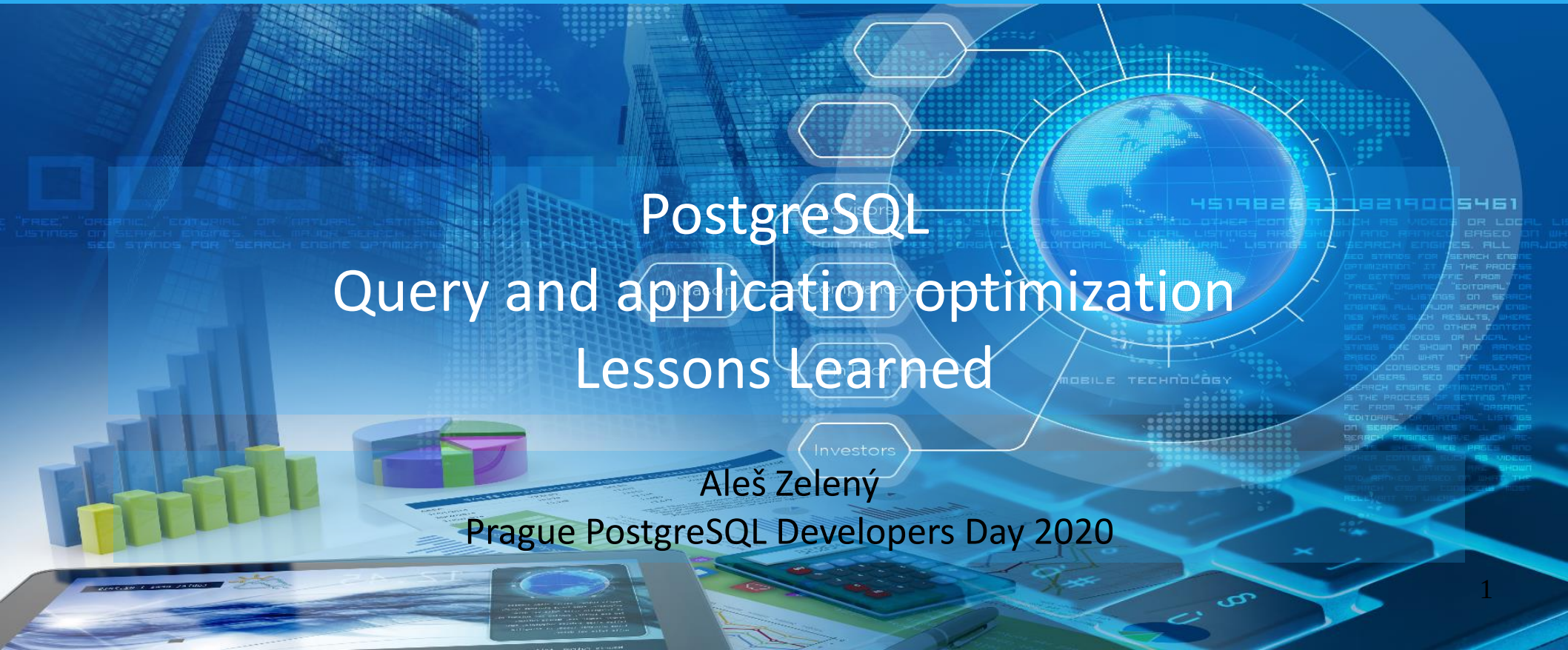
finmason



PostgreSQL
Query and application optimization
Lessons Learned

Aleš Zelený

Prague PostgreSQL Developers Day 2020

finmason

Who we are

Investment Analytics as a Service Platform

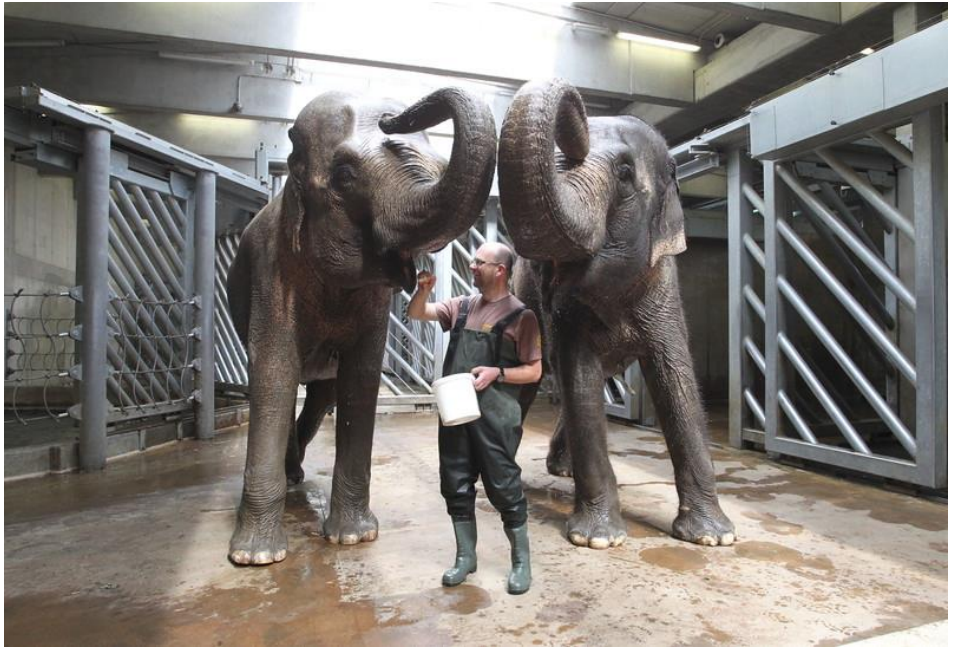
Who’s me?

InterBase / Firebird app developer, DBA (3 years)

Oracle DBA (17 years)

PostgreSQL DBA (since 2010)

Elephants enthusiast ...



Agenda

finmason

Distributed solution description

Requested feature

Query caused overloaded server CPU and network

View query evolution

Conclusion

Distributed

finmason

...around globe

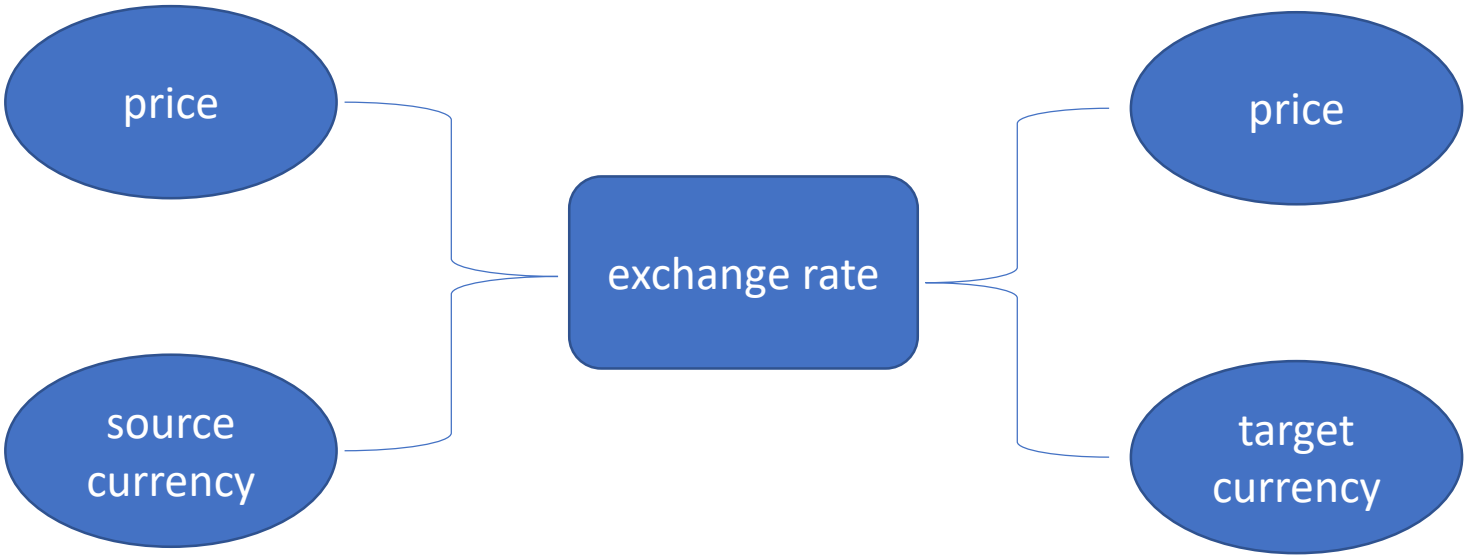
The feature request

- Write a view to convert prices
  - Source prices are provided in various currencies
  - There is **only one target** – base – currency in each geo locality
    - There is a configuration table defining target currency of each environment
  - **Code must be same** in each currency environment
  - An exchange rate table already exists

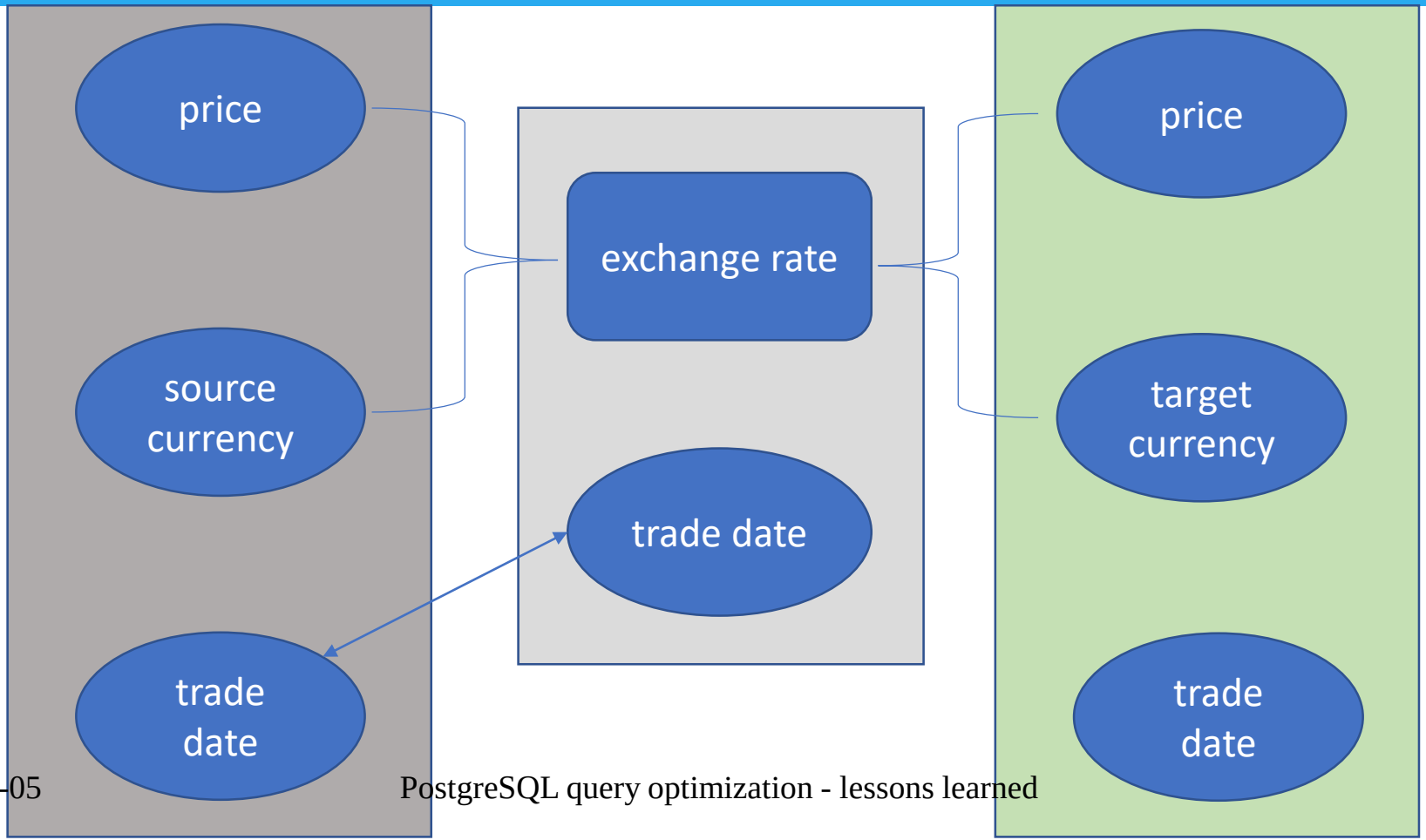
The view will replace an ETL used to convert the source process into base currency and storing them in a table → space saving, elimination of data fix logic used to convert updated values…

Analyze phase → easy task

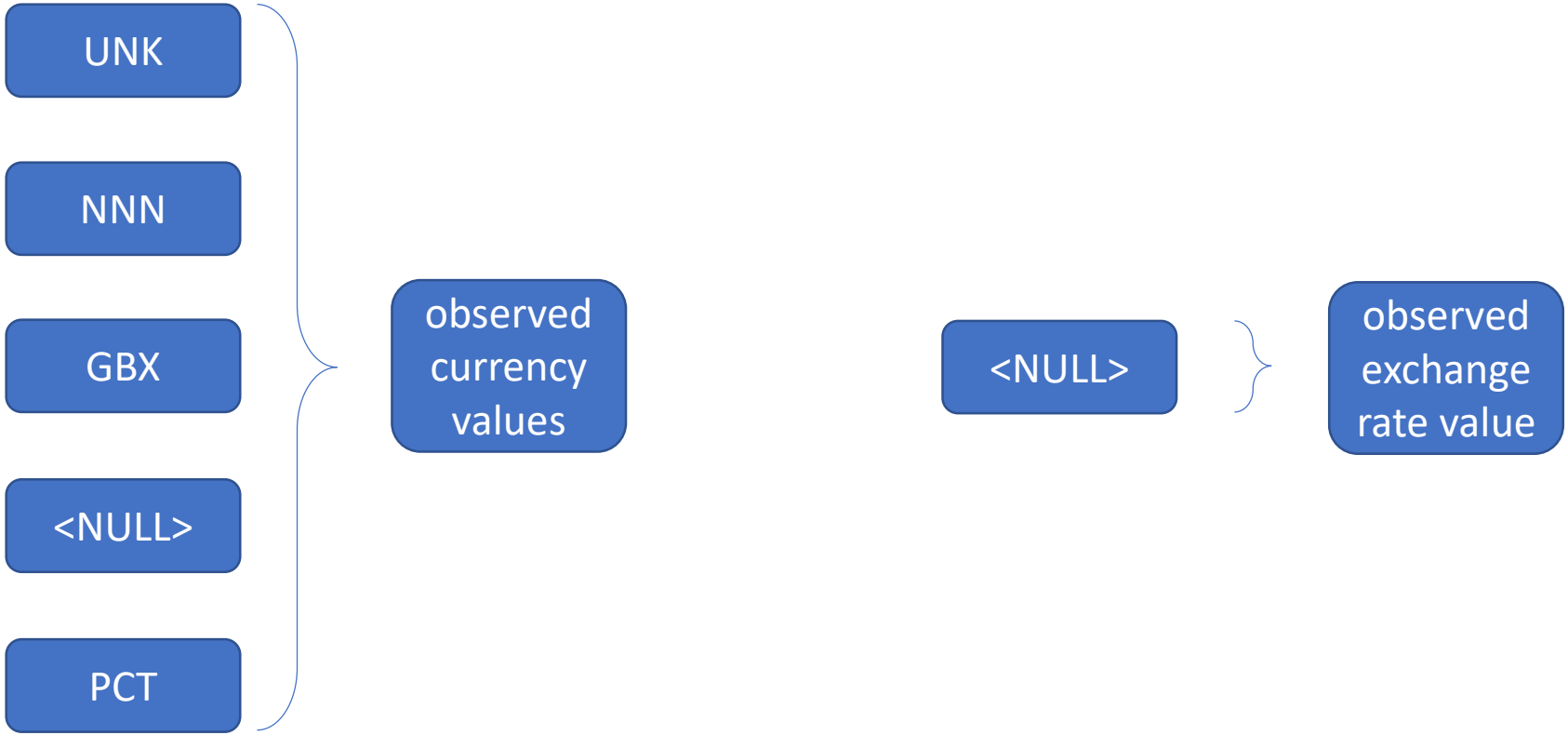
finmason



Analyze phase → oh, it is a time series…



Validate existing source data



Fix missing constraints

finmason

- Declarative data integrity is a must
  - Add missing lookup tables and foreign keys
  - Apply constraints – NOT NULL, unique, primary keys…
- Fix or discard data on load (ETL)
  - GBp/GBX are valid currencies despite not recognized by ISO 4217

Stocks are often traded in pence rather than pounds. Stock exchanges often use GBX (or GBp) to indicate that this is the case for the given stock rather than the ISO 4217 currency symbol GBP for pound sterling.

Implement prototype and test it

finmason

Sample result for USD prices converted into EUR.

```
 assetid | price | trade_date | p_currency | converted_value
---------+-------+------------+------------+------------------
  210235 | 11.72 | 1994-02-08 | EUR        | 12.9095804023743
  210235 | 11.29 | 1994-04-21 | EUR        |  12.934952750206
  210235 |  11.7 | 1994-01-14 | EUR        | 12.9682798504829
```

Something unexpected? Weekend prices...

finmason

```
=# SELECT date_trunc('mon', trade_date) AS month, count (*)
FROM public.asset_price_daily
WHERE extract(ISODOW FROM trade_date) >= 6
      AND trade_date < '1950-03-01'
GROUP BY ROLLUP(date_trunc('mon', trade_date))
ORDER BY month;
```

month	count
1929-01-01 00:00:00+00	1
1937-11-01 00:00:00+00	1
1939-12-01 00:00:00+00	1
1940-09-01 00:00:00+00	1
1949-05-01 00:00:00+00	1
1950-01-01 00:00:00+00	19
1950-02-01 00:00:00+00	16
«¤»	40

(8 rows)

Are exchange rates available from 1929? How convert to EUR prices from 1952 ?

```
=# SELECT trade_date, count (*)
FROM public.asset_price_daily
WHERE extract(ISODOW FROM trade_date) >= 6
      AND trade_date >= '2020-01-01'
GROUP BY ROLLUP(trade_date)
ORDER BY trade_date;
```

trade_date	count
2020-01-05	24
2020-01-11	4435
2020-01-12	4444
«¤»	8903

(4 rows)

Optimize…

- Decide how to handle corner cases
- Get details about intended view (feature) usage
  - Are there any “mandatory” predicates for given use case?
- Get details about underlying data nature
  - Most common values…
  - Special cases like weekend prices, weird currencies…
- Missing indexes?
  - Better indexes?

p_currency	count
USD	184844820
CAD	64228601
EUR	42215170
GBX	28855509
GBP	8170427

The feature request revised

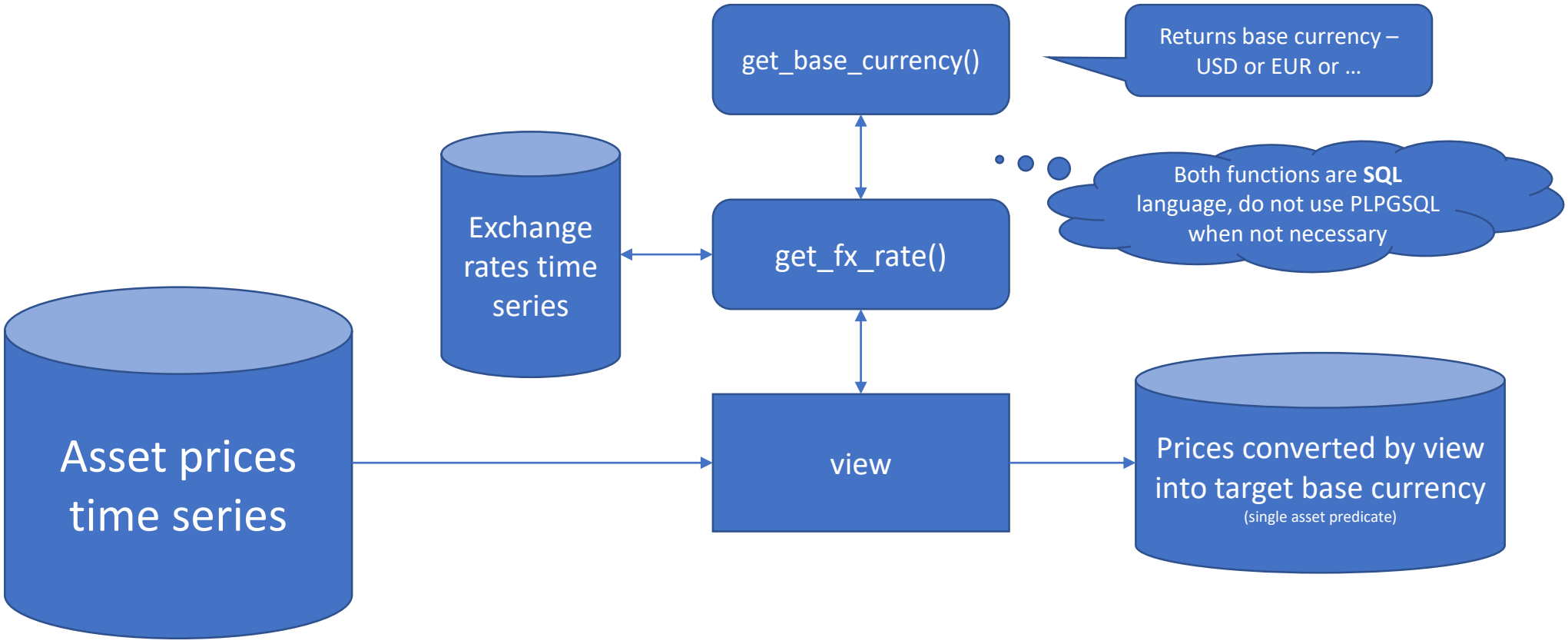
finmason

Write a view to convert prices from source daily prices of assets provided in various currencies to target – base – currency used in each geo/currency environment.

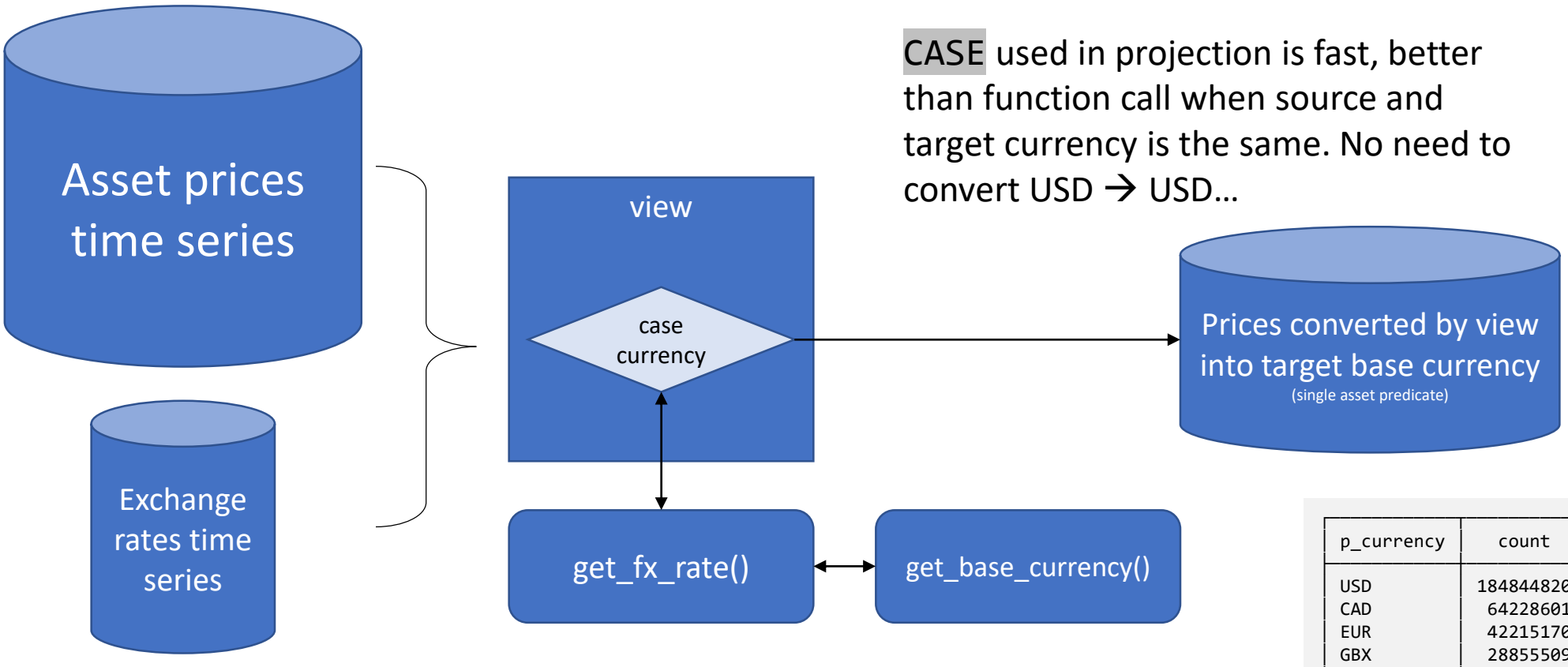
- Prices time series starts at first available exchange rate
  - Unless source and target price is identical (return all available prices)
- Use latest beforehand available exchange rate if exchange rate matching price date is not available (weekends, public holidays…)
- Application by its design always request full time series for one asset (data access pattern)

Conversion flow simplified

finmason



Convert only when needed – less function calls

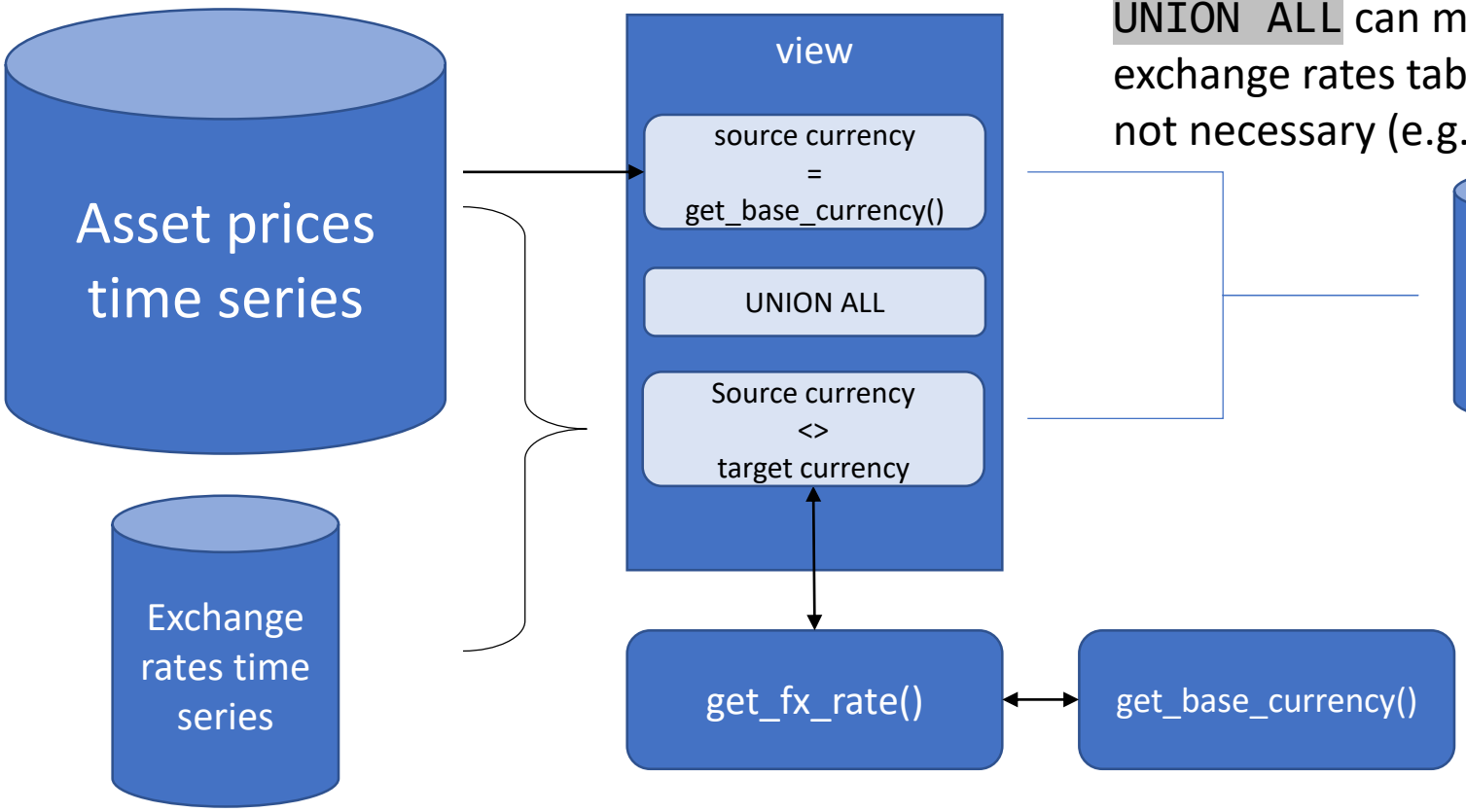


CASE used in projection is fast, better than function call when source and target currency is the same. No need to convert USD → USD…

p_currency	count
USD	184844820
CAD	64228601
EUR	42215170
GBX	28855509
GBP	8170427

Convert only when needed – less joins

finmason

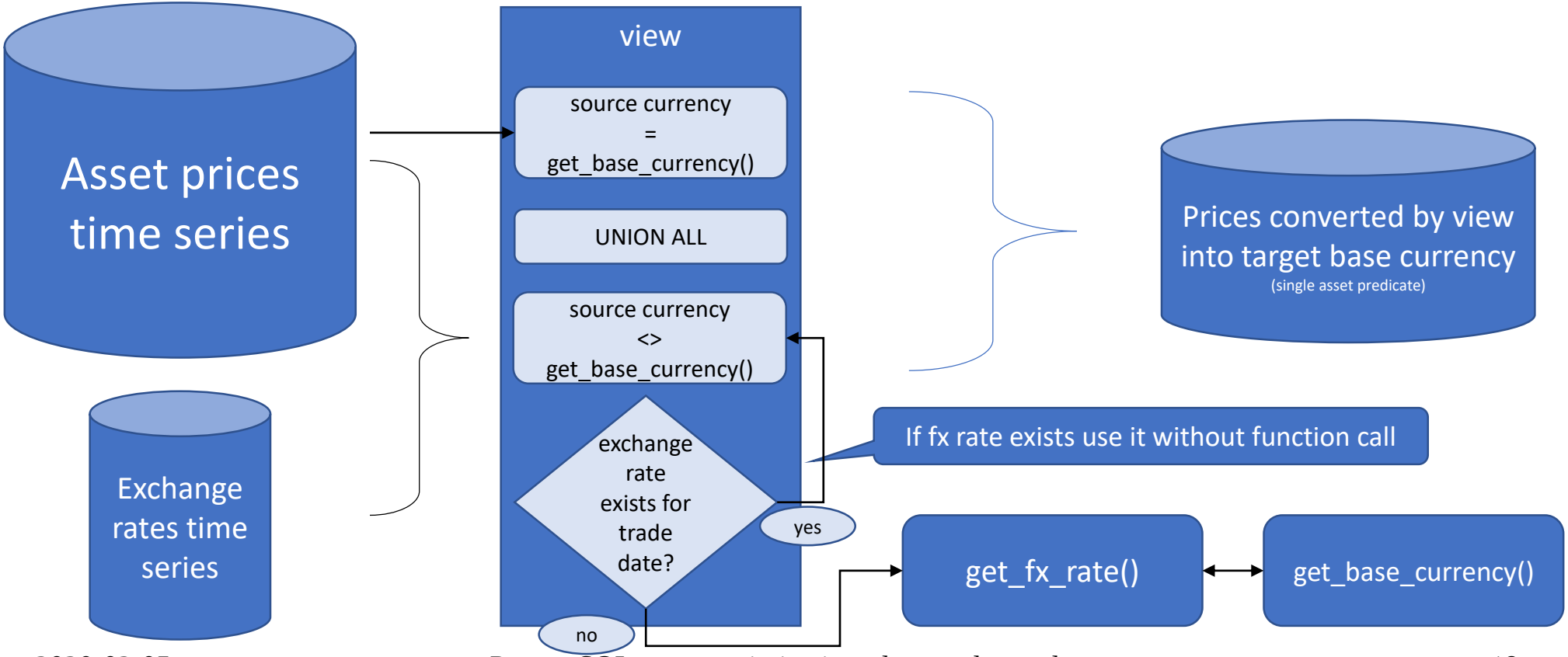


`UNION ALL` can mitigate join to exchange rates table when conversion is not necessary (e.g. CAD → CAD).

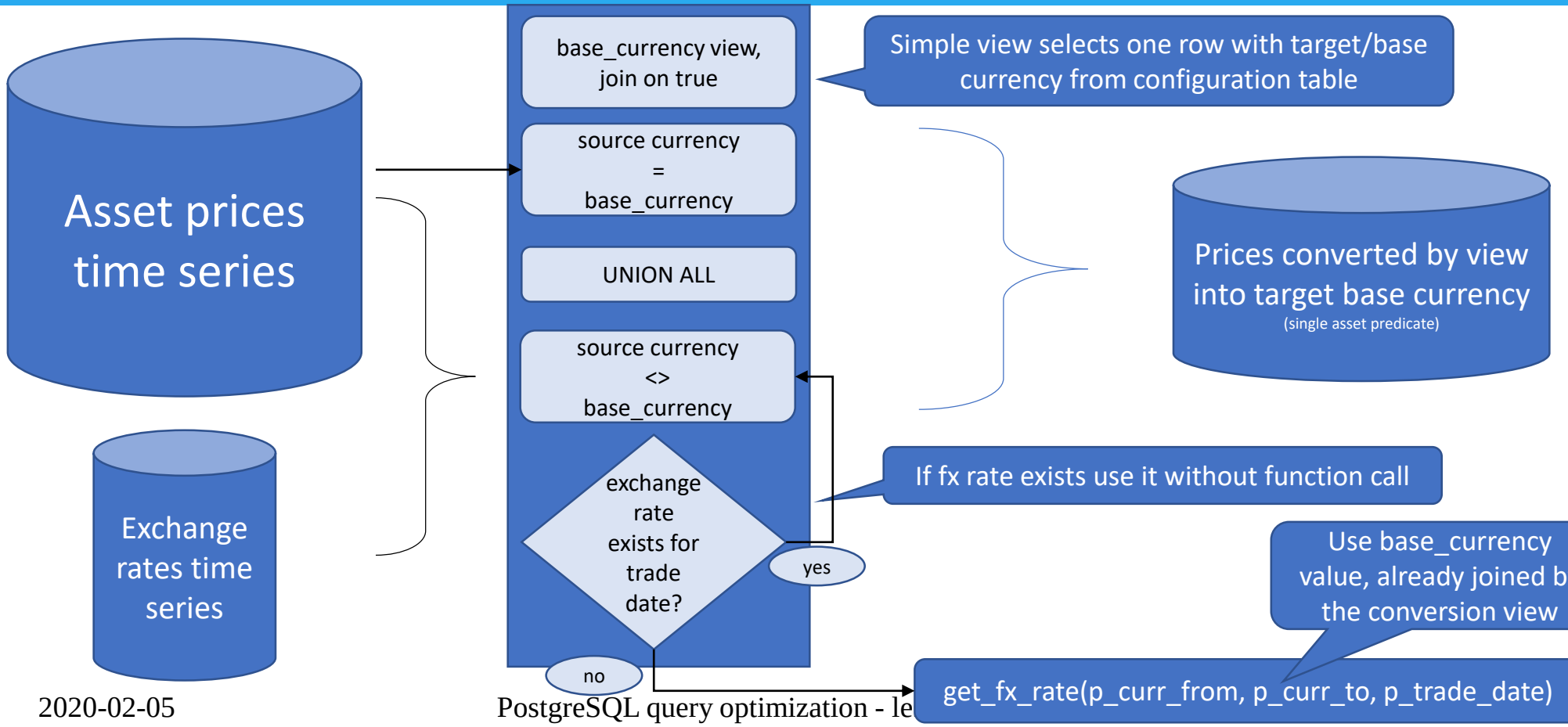
p_currency	count
USD	184844820
CAD	64228601
EUR	42215170
GBX	28855509
GBP	8170427

Call functions only when necessary

finmason



Simple join can be way faster than function call



Know your data helps optimizing

finmason

- Use `UNION ALL` instead of `UNION` if duplicates can't occur (or are allowed by application logic), it'll save one sort operation in execution plan
- Joins are usually faster than function calls
- Prefer SQL over PLPGSQL functions wherever possible
  - SQL functions can also have specified **volatility category**
  - If you can't find time spend in your execution plan, consider consult `pg_stat_xact_user_functions` view

Despite all SQL optimization…

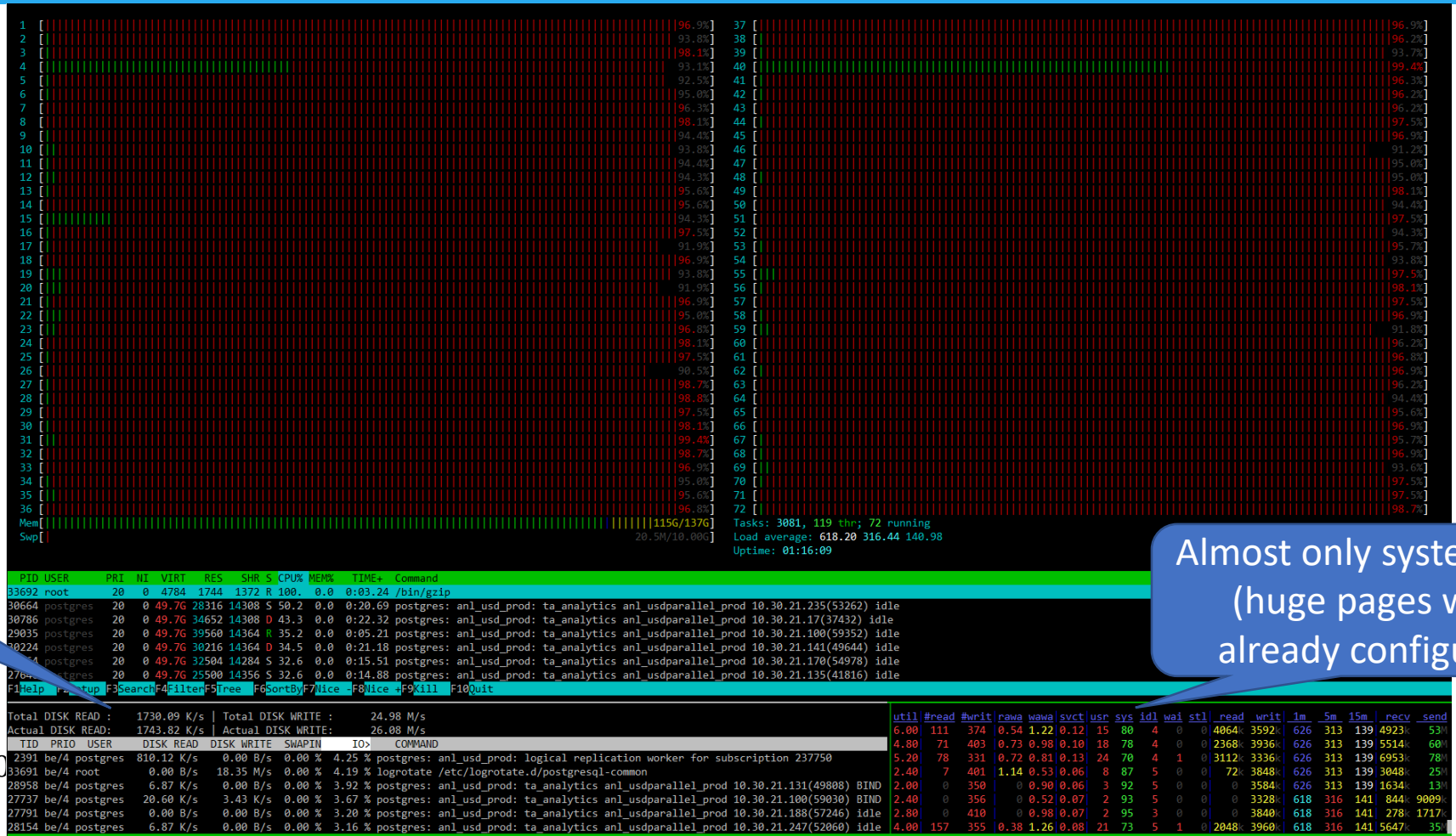
finmason

- SQL tuning – done (1200ms → 10-60ms)
- jMeter load tests – done
- Application tests... ☹
- Some benefits of open source will be demonstrated

Photo by Carlos Lincoln from Pixabay

You never want to see something like this...

finmason



Almost no IO

Almost only system time (huge pages were already configured)

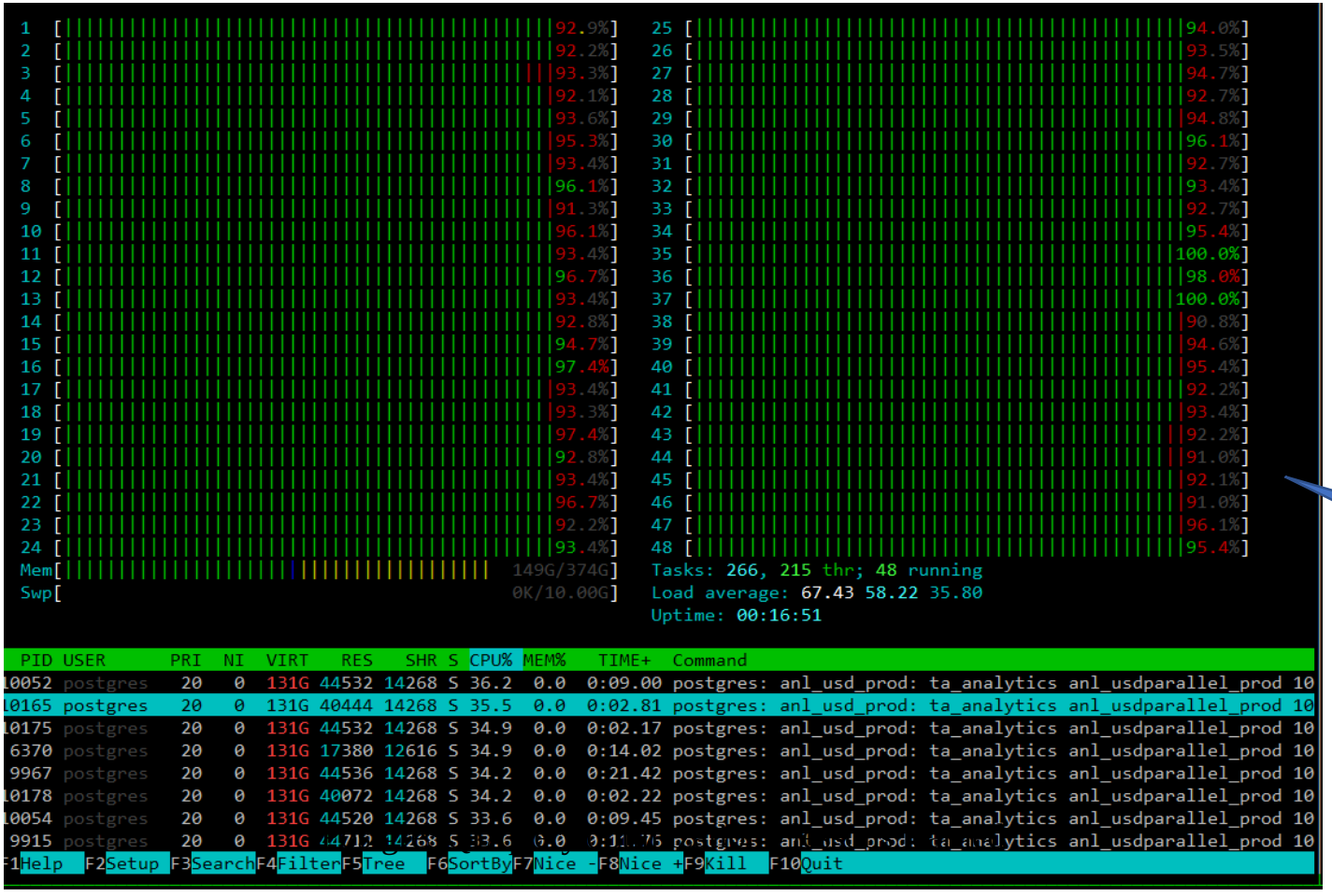
Connection pooling is necessary

Each time R code calls a query causes connect and disconnect on function exit.

```
… {
if(!exists('con')){ ReConnect() }
  on.exit(expr = dbDisconnect(con))
  …
}
```

Therefore **pgBouncer** was deployed.

72 vCPUs → 48 vCPUs



Almost only USER time

Why one might prefer open source?

Despite nice change from SYSTEM to USER time and less vCPU usage for same test…

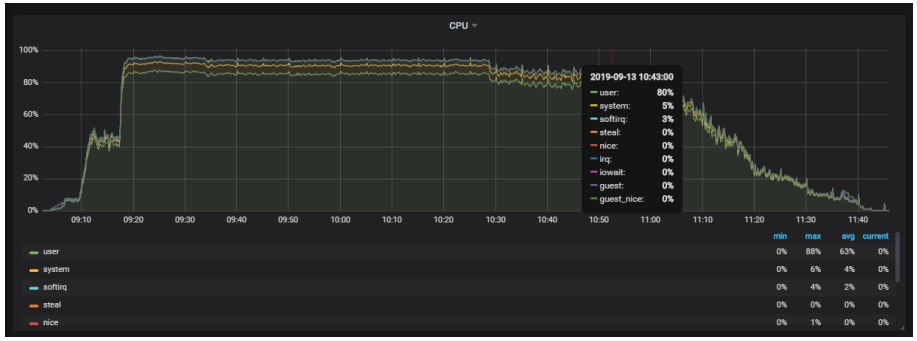
```
Failed to retrieve query result metadata: ERROR:  unnamed prepared
statement does not exist
```

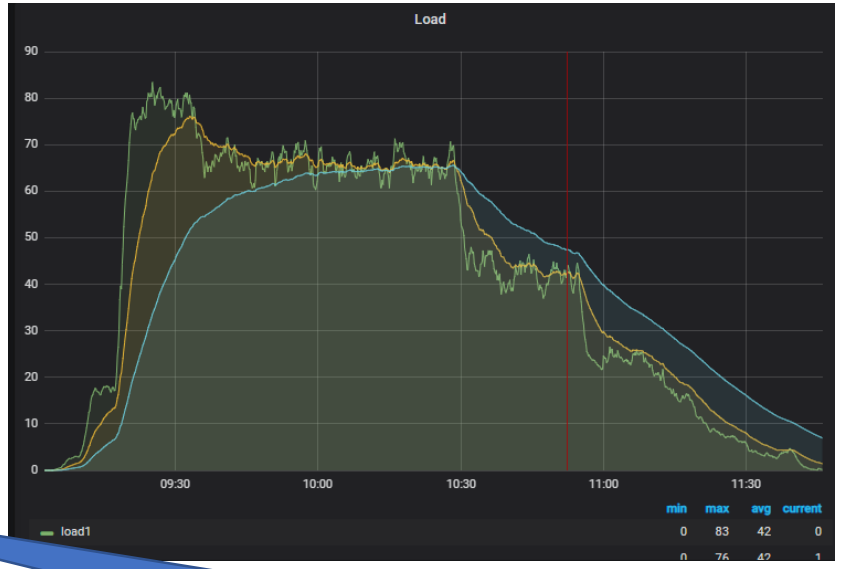
Database interface and 'PostgreSQL' driver for 'R'.

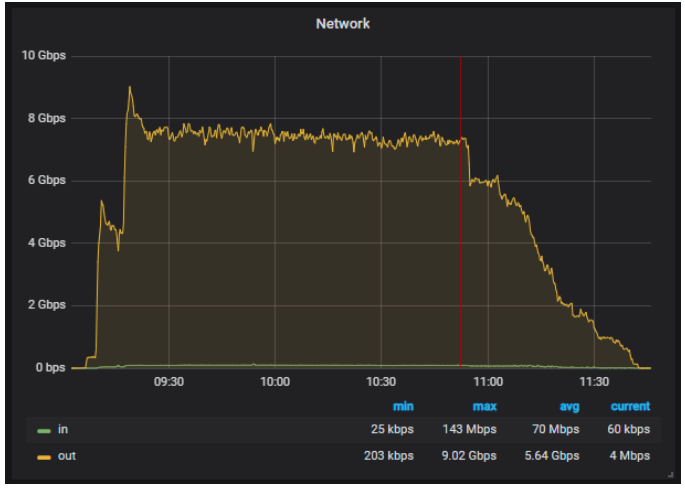
Github r-dbi issue 185 – **prepared statements are required** → Use pgBouncer in **session mode** instead of transaction mode.

c5d.9xlarge – 36vCPUs, 72 GiB RAM

finmason







pgBouncer pool in SESSION mode for R clients fixed issue with prepared statements, but Network throughput raises a lot.

pg_stat_statements is mandatory

finmason

Application query based on created view behaves as expected in terms of response times and amount of executions.

However, there were a blameless statement :

```
"SELECT oid, typname FROM pg_type"
```

Response time was OK, but the amount of executions was unbelievably high.

Searched application code – such statement was not found ☹

Open source allows one to search code…

finmason

RPostgres driver stores postgres types in cache ***fully populated on connect***. RPostgres PqConnection.R implementation:

```
conn@typnames <- dbGetQuery(conn, "SELECT oid, typname FROM pg_type")
```

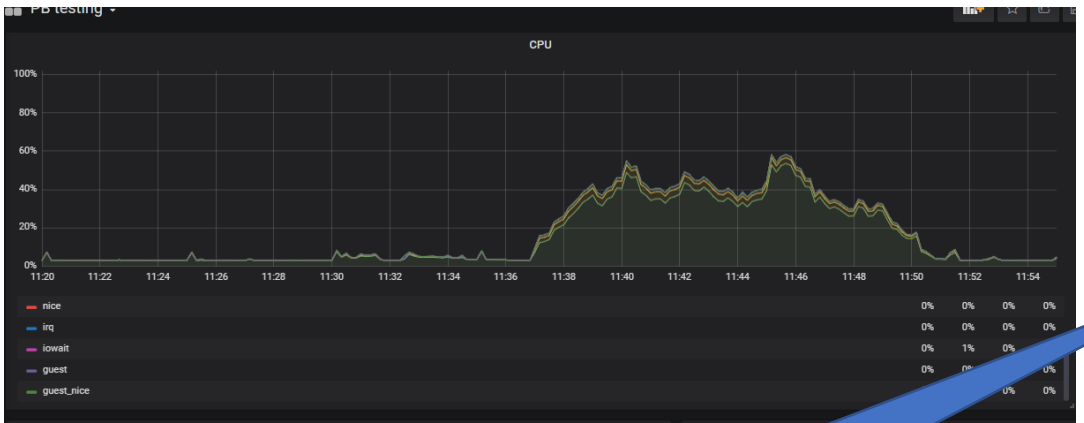
pg_type docs:The catalog pg_type stores information about data types. Base types and enum types (scalar types) are created with CREATE TYPE, and domains with CREATE DOMAIN. **A composite type is automatically created for each table in the database**, to represent the row structure of the table. It is also possible to create composite types with CREATE TYPE AS.

```
SELECT pg_size_pretty(sum(pg_column_size(row(q))))
FROM (SELECT oid, typname FROM pg_type)q;
```

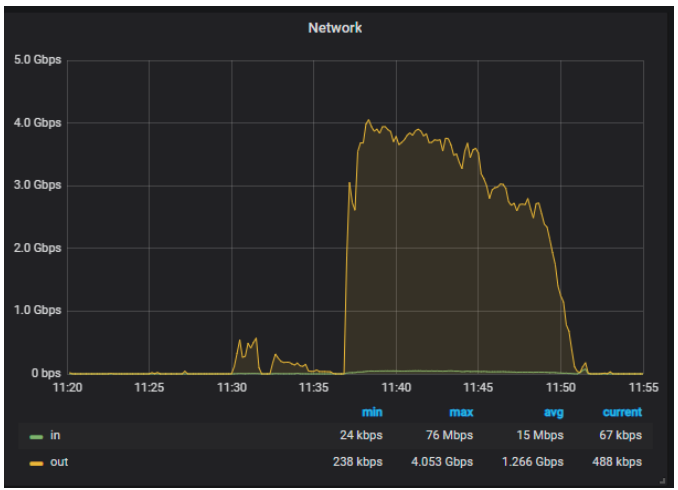
→ 1187 kB, 10757 rows

Application change

R code working with database must persist connection between queries.



Network utilization drops down significantly



Tedious part

finmason

Steps described above explained on examples…

Image by PublicDomainPictures from Pixabay

All sites shares same schema

finmason

local_config

variable	type	intvalue
currency	int	5

Geo/currency locality is defined in EAV like table …

currency

currencyid	shortname	Description
1	USD	US Dollar
2	CAD	Canadian Dollar
5	EUR	European euro

asset

assetid	assetname	ISIN
1234	Star Capitol America	FR0000991101

asset_price_daily

id	trade_date	assetid	p_currency	price
…	2019-09-12	1234	EUR	3.1415
…	2019-09-12	1235	USD	2.7182
	2019-09-12	1236	CAD	1.6605
	2019-09-12	1235	USD	6.6260

currency_exchange_rate

id	trade_date	curr_from	curr_to	fxrate
…	2019-09-12	USD	EUR	0.8912
…	2019-09-12	USD	CAD	1.2405
	2019-09-12	EUR	USD	1.1220
	2019-09-13	USD	EUR	0.8911

Column logical references

finmason

local_config

variable	type	intvalue
currency	int	5

currency

currencyid	shortname	Description
1	USD	US Dollar
2	CAD	Canadian Dollar
5	EUR	European euro

asset

assetid	assetname	ISIN
1234	Star Capitol America	FR0000991101

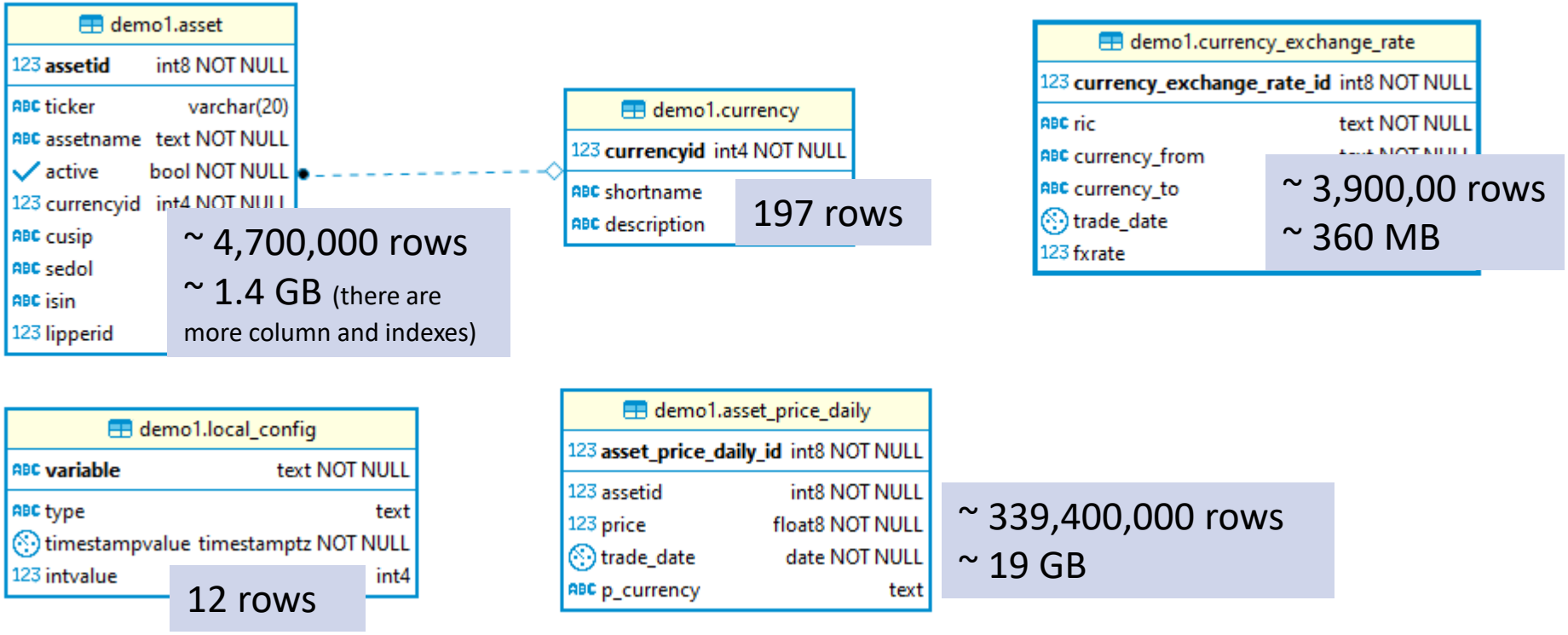
asset_price_daily

id	trade_date	assetid	p_currency	price
…	2019-09-12	1234	EUR	3.1415
…	2019-09-12	1235	USD	2.7182
	2019-09-12	1236	CAD	1.6605
	2019-09-12	1235	USD	6.6260

currency_exchange_rate

id	trade_date	curr_from	curr_to	fxrate
…	2019-09-12	USD	EUR	0.8912
…	2019-09-12	USD	CAD	1.2405
	2019-09-12	EUR	USD	1.1220
	2019-09-13	USD	EUR	0.8911

Initial model data volumes



The view creation

- Consider entities involved
- Check existing code, someone might already write such select
- Write the query prototype
- Close request ticket and relax ☺

On the other hand… things tends to be a bit more complicated.

Found suitable existing SQL?

finmason

- Check comments for suspicious complications
- Observe used predicates
- What was purpose of the already existing query?

Found suitable existing SQL?

finmason

- **Check comments for suspicious complications**
  - Data retrieval SQL should not sanitize inconsistent source data if they are supposed to be fast (enough)
  - Meaningful stuff with simple comment is usually easy to understand and locate in code
- Observe used predicates
- What was purpose of the already existing query?

Read code comments (if there are some…)

finmason

```
SELECT df.assetid, df.price as price, df.trade_date, cer.currency_to p_currency, df.price / fxrate as converted_value
/* Get Price For Asset when the currency is unknown or missing replace the p_currency
with the currency identified in asset table */
FROM  (
  SELECT apd.assetid, apd.trade_date,
    CASE WHEN apd.p_currency = 'GBX' THEN apd.price / 100 ELSE apd.price END as price,
    CASE WHEN apd.p_currency IS NULL OR apd.p_currency = 'UNK' THEN a_cur.shortname ELSE apd.p_currency END as p_currency
  FROM demo1.asset_price_daily apd
  /* Get the Asset currencyid in case a currency_code is NULL or UNKNOWN */
  JOIN asset a ON apd.assetid = a.assetid
  JOIN currency a_cur ON a_cur.currencyid = a.currencyid
) df
/* Get the Daily Currency Exchange Rates */
LEFT JOIN (
  SELECT trade_date, currency_from,  currency_to, fxrate FROM demo1.currency_exchange_rate WHERE fxrate > 0
) cer ON df.p_currency = cer.currency_from AND df.trade_date = cer.trade_date
/* Get the Currency Code for the Current Database */
JOIN demo1.currency cur ON cur.shortname = cer.currency_to
JOIN demo1.local_config lcfg ON lcfg.intvalue = cur.currencyid
WHERE lcfg.variable = 'targetcurrency';
```

Looks like sanity check

Looks like sanity check

Code comment

Code comment

Found suitable existing SQL?

finmason

- Check comments for suspicious complications

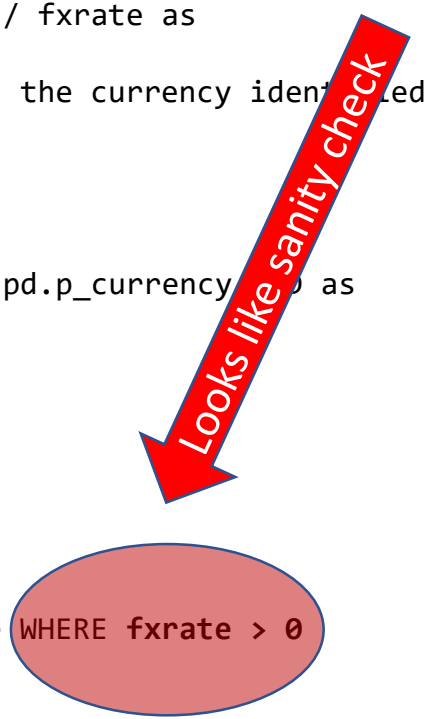
- **Observe used predicates**
  - Limiting query result set (like, today rows only…)?
    - No result set limit?
    - In our example, all prices for all assets are used by the query
  - Something weird ?

- What was purpose of the already existing query?

Read code comments (if there are some…)

finmason

```
SELECT df.assetid, df.price as price, df.trade_date, cer.currency_to p_currency, df.price / fxrate as
converted_value
/* Get Price For Asset when the currency is unknown or missing replace the p_currency with the currency iden ied
in asset table */
FROM  (
  SELECT apd.assetid, apd.trade_date,
    CASE WHEN apd.p_currency = 'GBX' THEN apd.price / 100 ELSE apd.price END as price,
    CASE WHEN apd.p_currency IS NULL OR apd.p_currency = 'UNK' THEN a_cur.shortname ELSE apd.p_currency  D as
p_currency
  FROM demo1.asset_price_daily apd
  /* Get the Asset currencyid in case a currency_code is NULL or UNKNOWN */
  JOIN asset a ON apd.assetid = a.assetid
  JOIN currency a_cur ON a_cur.currencyid = a.currencyid
) df
/* Get the Daily Currency Exchange Rates */
LEFT JOIN (
  SELECT trade_date, currency_from,  currency_to, fxrate FROM demo1.currency_exchange_rate WHERE fxrate > 0
) cer ON df.p_currency = cer.currency_from AND df.trade_date = cer.trade_date
/* Get the Currency Code for the Current Database */
JOIN demo1.currency cur ON cur.shortname = cer.currency_to
JOIN demo1.local_config lcfg ON lcfg.intvalue = cur.currencyid
WHERE lcfg.variable = 'targetcurrency';
```



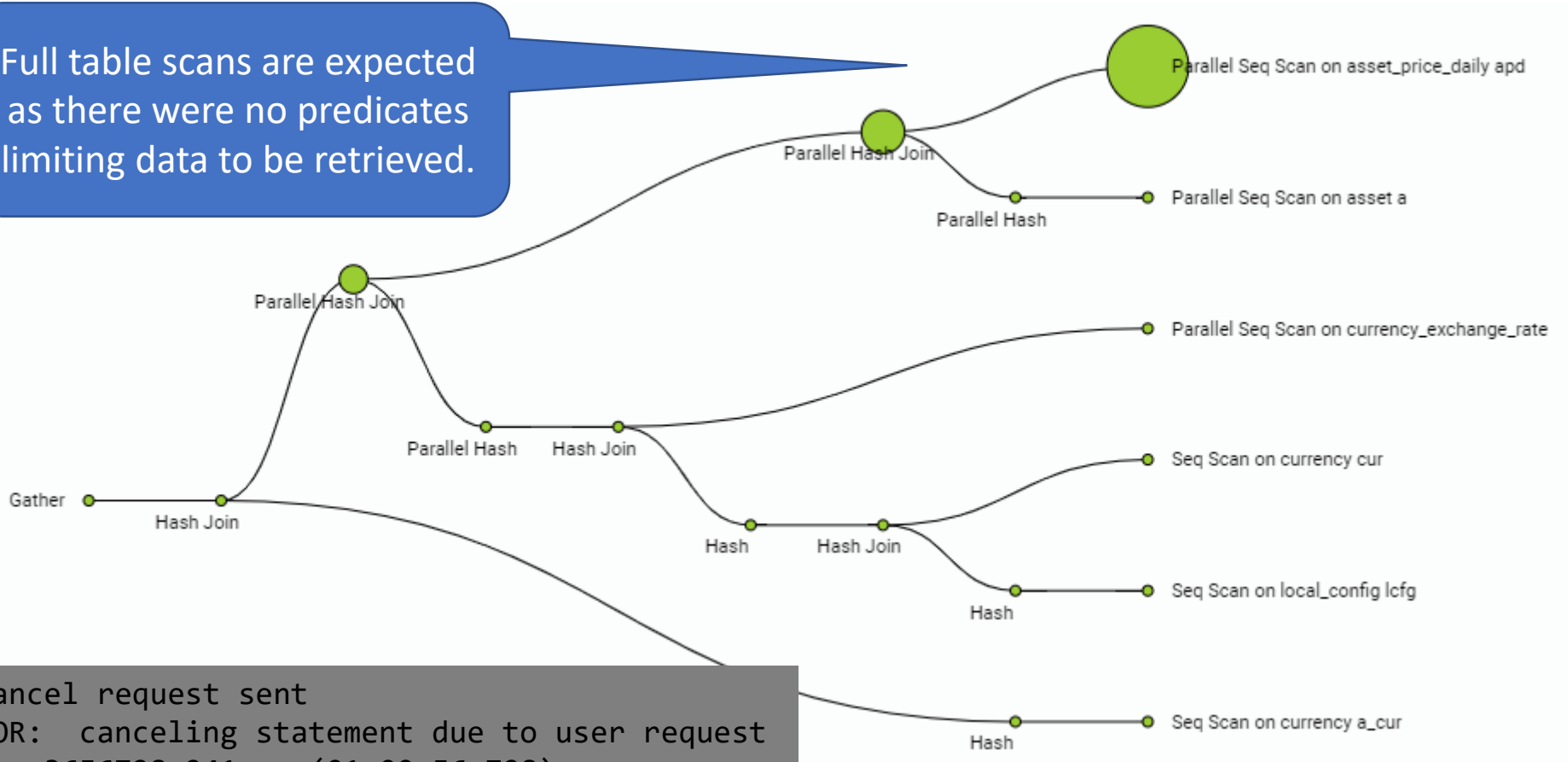
Found suitable existing SQL?

finmason

- Check comments for suspicious complications
- Observe used predicates
- **What was purpose of the already existing query?**
  - Ask for context, what was intended query usage?
    - It was intended to be part of **batch job for initial conversion** in new geo datacenter saving data into old legacy table with converted prices
    - BTW first implementation expect trade_date to be “leading” predicate
      - Inappropriate indexes were prepared for real use case in app

Check execution plan

Full table scans are expected as there were no predicates limiting data to be retrieved.




```
^CCancel request sent
ERROR:  canceling statement due to user request
Time: 3656728.241 ms (01:00:56.728)
```

Usage pattern

finmason

- Reach the person requesting a feature and get as much as possible details, how he/she will use it (code, API, human being user…)
  - Exactly one asset (asset_id) price history will be retrieved
  - trade_date upper limit will be provided (dividends might have future trade_date values, these can’t be converted to different currency)
  - Up to 5000 concurrent executions (bloody cloud capabilities… ☺)
  - OK, lets test it

Choose some test data

finmason

Choose an asset for testing and get amount of prices.

```
=> SELECT assetid, count(*) FROM demo1.asset_price_daily
 GROUP BY assetid ORDER BY 2 DESC LIMIT 1;
 assetid | count
---------+-------
  210235 | 24897

=> SELECT p_currency, count(*) FROM demo2.asset_price_daily
 WHERE assetid = 210235 GROUP BY p_currency;
 p_currency | count
------------+-------
 USD        | 24897
(1 row)

=> SELECT assetname FROM demo2.asset WHERE assetid = 210235;
           assetname
-------------------------------
 MASSACHUSETTS INVESTORS TRUST
(1 row)
```

Asset with highest amount of prices available

Amount of prices per currency for selected asset

Single asset test – add predicate

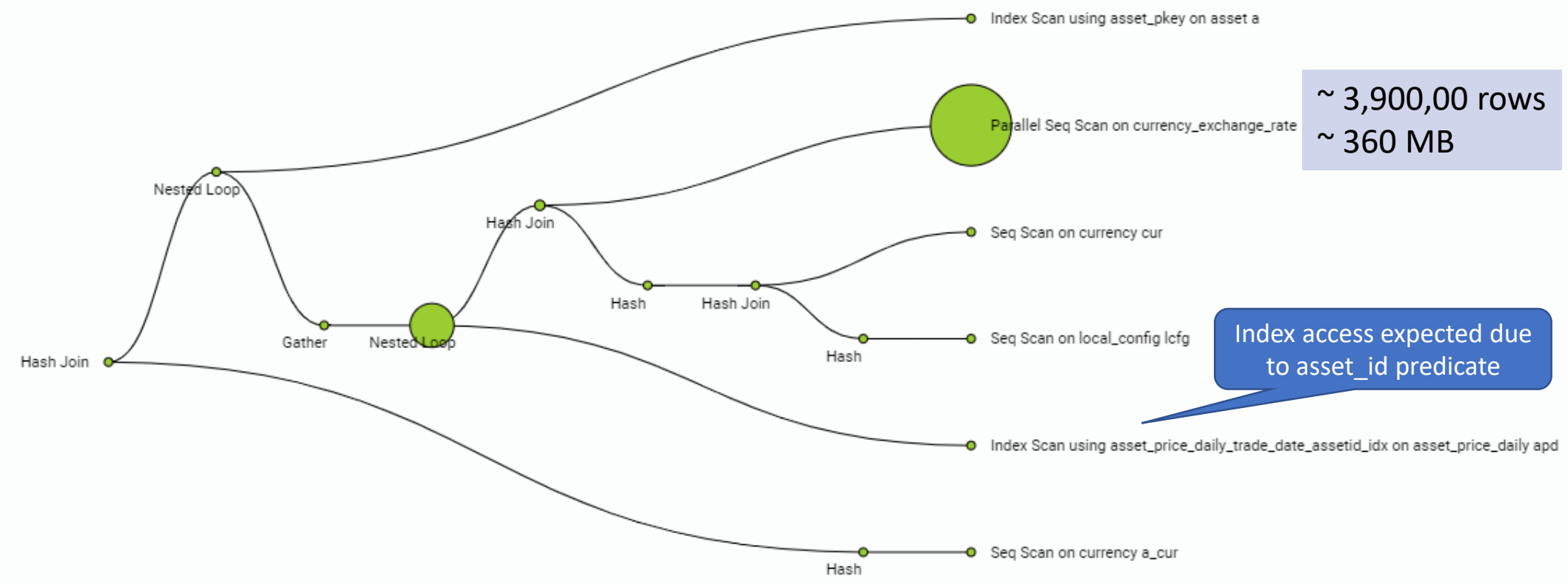
finmason

```
explain (analyze, buffers, costs off)
SELECT df.assetid, df.price as price, df.trade_date, cer.currency_to p_currency, df.price / fxrate as converted_value
/* Get Price For Asset when the currency is unknown or missing replace the p_currency with the currency identified in asset
table */
FROM  (
  SELECT apd.assetid, apd.trade_date,
    CASE WHEN apd.p_currency = 'GBX' THEN apd.price / 100 ELSE apd.price END as price,
    CASE WHEN apd.p_currency IS NULL OR apd.p_currency = 'UNK' THEN a_cur.shortname ELSE apd.p_currency END as p_currency
  FROM demo1.asset_price_daily apd
  /* Get the Asset currencyid in case a currency_code is NULL or UNKNOWN */
  JOIN asset a ON apd.assetid = a.assetid
  JOIN currency a_cur ON a_cur.currencyid = a.currencyid
) df
/* Get the Daily Currency Exchange Rates */
LEFT JOIN (
  SELECT trade_date, currency_from,  currency_to, fxrate FROM demo1.currency_exchange_rate WHERE fxrate > 0
) cer ON df.p_currency = cer.currency_from AND df.trade_date = cer.trade_date
/* Get the Currency Code for the Current Database */
JOIN demo1.currency cur ON cur.shortname = cer.currency_to
JOIN demo1.local_config lcfg ON lcfg.intvalue = cur.currencyid
WHERE lcfg.variable = 'targetcurrency'
and df.assetid = 210235;
```

Prices for one of 4.700.000 assets

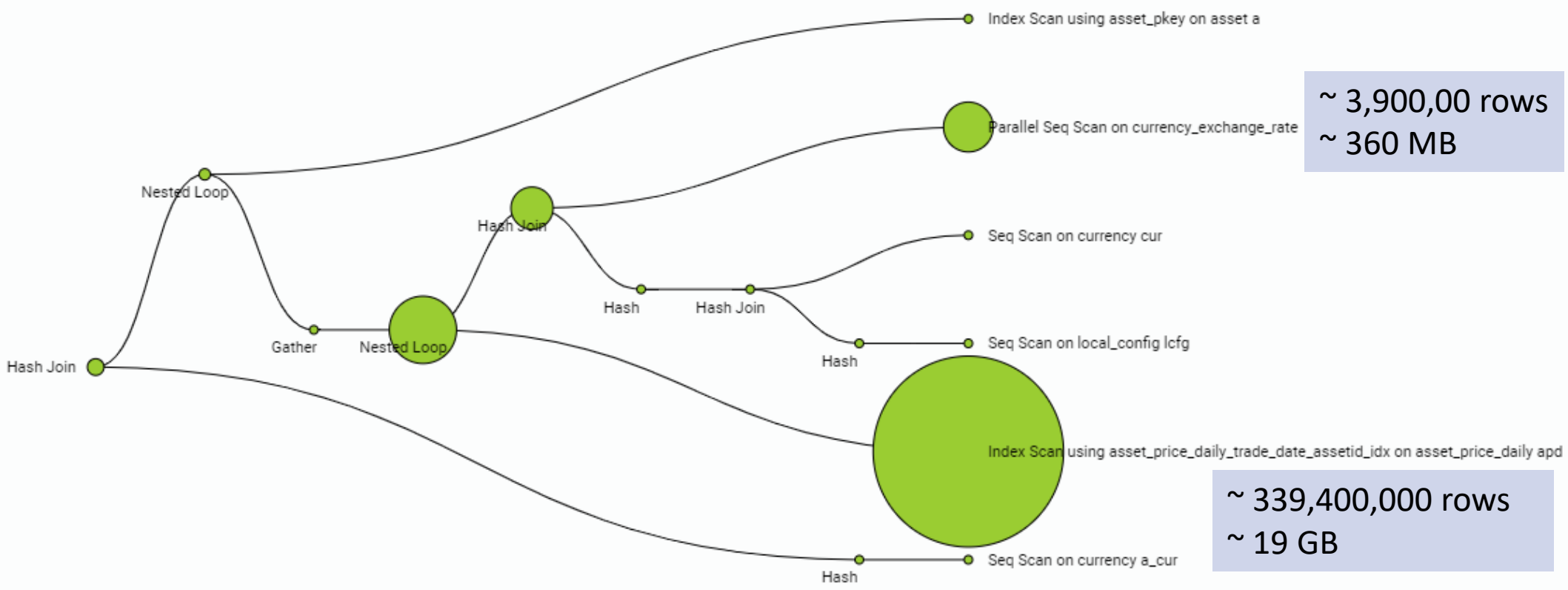
Single asset test – query plan

finmason



Single asset test – query plan - analyze

finmason



~ 3,900,00 rows
~ 360 MB

~ 339,400,000 rows
~ 19 GB

Single asset test – query plan - analyze

finmason

```
Hash Join (actual time=0.781..1298.360 rows=6754 loops=1)
  Hash Cond: (a.currencyid = a_cur.currencyid)
  Join Filter: (CASE WHEN ((apd.p_currency IS NULL) OR (apd.p_currency = 'UNK'::text)) THEN a_cur.shortname ELSE apd.p_currency
END = currency_exchange_rate.currency_from)
  Rows Removed by Join Filter: 831639
  Buffers: shared hit=4379443
  ->  Nested Loop (actual time=0.405..1141.843 rows=838115 loops=1)
      Buffers: shared hit=4379441
        -> Index Scan using asset_pkey on asset a (actual time=0.027..0.029 rows=1 loops=1)
        ->  Gather (actual time=0.376..1031.783 rows=838115 loops=1)
          ->  Nested Loop (actual time=0.164..1080.571 rows=279372 loops=3) # Buffers: shared hit=4379434
            ->  Hash Join (actual time=0.137..293.606 rows=292272 loops=3)
              Hash Cond: (currency_exchange_rate.currency_to = (cur.shortname)::text)
              Buffers: shared hit=28666
              ->  Parallel Seq Scan on currency_exchange_rate (actual time=0.008..153.417 rows=1292925 loops=3)
                  Filter: (fxrate > '0'::double precision)
                  Buffers: shared hit=28521
              ->  Hash (actual time=0.075..0.075 rows=1 loops=3)
                ->  Hash Join (actual time=0.042..0.073 rows=1 loops=3)
                  ->  Seq Scan on currency cur (actual time=0.010..0.024 rows=197 loops=3) # Buffers: shared hit=6
                  ->  Hash (actual time=0.009..0.009 rows=1 loops=3)
                      ->  Seq Scan on local_config lcfg (actual time=0.006..0.006 rows=1 loops=3)
        ->  Index Scan using asset_price_daily_trade_date_assetid_idx on asset_price_daily apd (actual time=0.002..0.002 rows=1 loops=876815)
              Index Cond: ((trade_date = currency_exchange_rate.trade_date) AND (assetid = 210235)) # Buffers: shared hit=4350768
  ->  Hash (actual time=0.051..0.051 rows=197 loops=1)
      ->  Seq Scan on currency a_cur (actual time=0.007..0.026 rows=197 loops=1)
Planning Time: 1.313 ms, Execution Time: 1301.637 ms
```

24897 rows expected

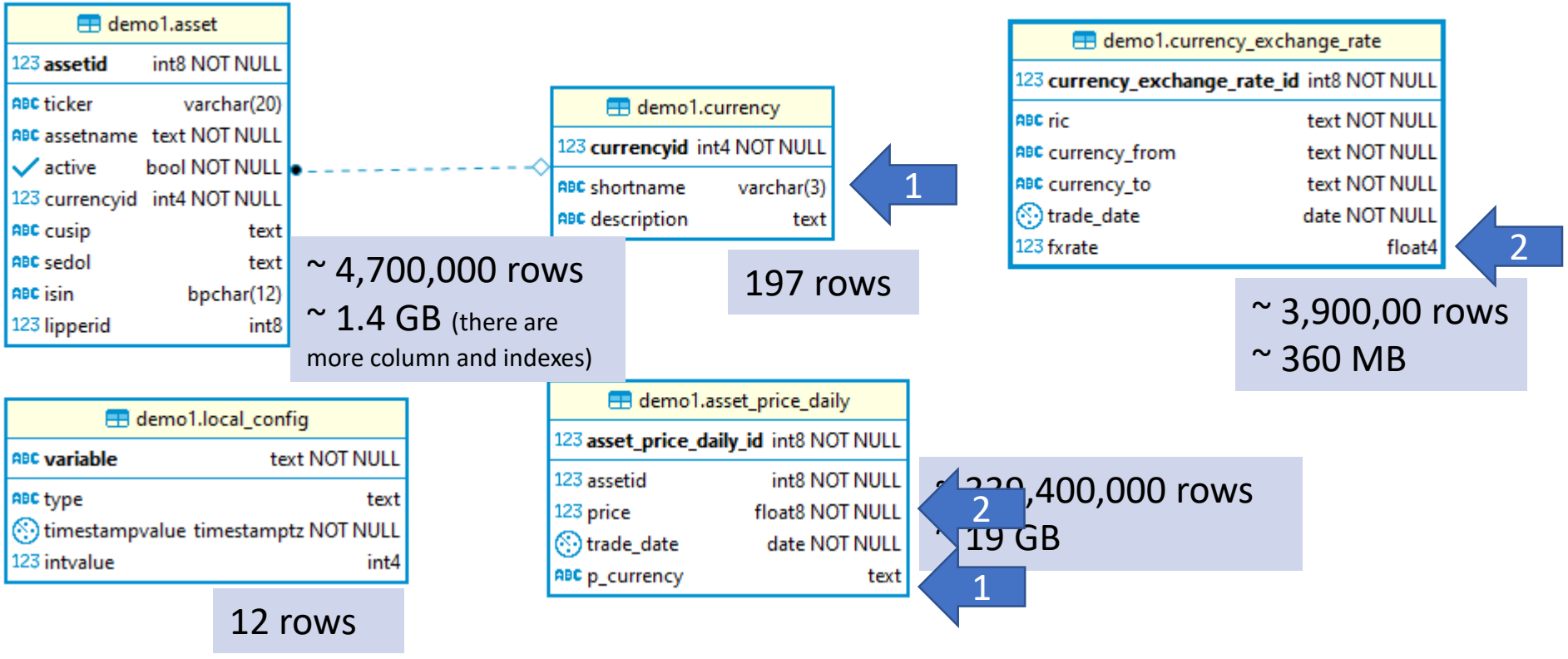
Sanitization, we are already aware of it…

Why typecast?

Seq scan…

Exclude null values?

Found not matching data types…



Data issue to be solved is still there…

The explain analyze result set was:

```
Hash Join (actual time=0.781..1298.360 rows=6754 loops=1)
```

```
 assetid | price | trade_date | p_currency | converted_value
---------+-------+------------+------------+------------------
  210235 | 11.72 | 1994-02-08 | EUR        | 12.9095804023743
  210235 | 11.29 | 1994-04-21 | EUR        |  12.934952750206
  210235 |  11.7 | 1994-01-14 | EUR        | 12.9682798504829
```

While expected count was:

```
 assetid |           assetname            | count
---------+--------------------------------+-------
  210235 | MASSACHUSETTS INVESTORS TRUST | 24897
```

Takeaway → never trust data, juts because sample looks ok…

Analyze source data

finmason

- Define test data carefully
- Looking just for longest price history is not enough (but already helped us to identify an issue)
- We have found UNK, NNN, PCT and *null* currencies in price table **some** of these values were in currencies table
- We have found null exchange rates (fxrate column)
- Review data model
    - Are all logical references covered by constraints?

Validate data with your expectations

finmason

```
 assetid | price | trade_date | p_currency | converted_value
---------+-------+------------+------------+------------------
  210235 | 11.72 | 1994-02-08 | EUR        | 12.9095804023743
  210235 | 11.29 | 1994-04-21 | EUR        |  12.934952750206
  210235 |  11.7 | 1994-01-14 | EUR        | 12.9682798504829
```

We do not have exchange rates for every price trade_date, also there are weekend prices and exchange rates over weekends are not expected to be available anyway

```
=> SELECT 'apd' as tbl, min(trade_date) FROM demo1.asset_price_daily apd WHERE apd.assetid = 248778
 UNION ALL
 SELECT 'cer' as tbl, min(trade_date) FROM demo1.currency_exchange_rate WHERE currency_to = 'EUR' and fxrate > 0;
 tbl |    min
-----+------------
 apd | 1924-07-15
 cer | 1994-01-03

=> SELECT shortname FROM demo1.local_config lcfg JOIN demo1.currency cur ON lcfg.intvalue = cur.currencyid WHERE
lcfg.variable = 'targetcurrency';
 shortname
-----------
 EUR
```

Check also less expected conditions, are we converting to intended currency?

Review data model and clean the data…

finmason

- Data types were inconsistent
- Nullable columns where it does not make any sense
  - Exchange rate without the fxrate value has no value and brings only additional issue
- Ensure suitable indexes are in place
- Logical relations are only nice, whereas on declarative data integrity constraints one can rely
  - Get rid of “UNK”, “NNN” currencies and since GBX is “standard” add it into currency lookup table and ensure exchange rates are populated for GBX/GBp even it is GBP/100

Decompose complex SQL for test

finmason

- SQL might look too complicated at first view
- SQL is composable, try to identify contributing pieces
  - Test them piece by piece
  - Compose back original intended statement
- Try to identify corner cases
  - Might they be mitigated
  - Might they be ignored (just be aware of)?

Minimalistic rate conversion prototype

finmason

```
SELECT apd.asset_price_daily_id, apd.assetid, apd.price / cer.fxrate AS price,
       apd.trade_date, 'EUR' AS p_currency
FROM demo1.asset_price_daily apd
LEFT JOIN demo1.currency_exchange_rate cer
       ON cer.trade_date = apd.trade_date AND cer.currency_from = apd.p_currency
AND cer.currency_to = 'EUR'
WHERE apd.assetid = 210235;
```

Fxrate is not checked for “> 0” so all expected rows were returned even with null prices.

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------
 Gather (actual time=540.054..9798.100 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Hash Left Join (actual time=612.788..9763.884 rows=8299 loops=3)
         Hash Cond: ((apd.trade_date = cer.trade_date) AND (apd.p_currency = cer.currency_from))
         ->  Parallel Seq Scan on asset_price_daily apd (actual time=123.949..9267.408 rows=8299 loops=3)
               Filter: (assetid = 210235)
               Rows Removed by Filter: 113101986
         ->  Parallel Hash (actual time=485.575..485.575 rows=292340 loops=3)
               Buckets: 1048576  Batches: 1  Memory Usage: 49408kB
               ->  Parallel Seq Scan on currency_exchange_rate cer (actual time=247.370..383.635 rows=292340 loops=3)
                     Filter: (currency_to = 'EUR'::text)
                     Rows Removed by Filter: 1000585
 Planning Time: 0.329 ms
 Execution Time: 9801.198 ms
```

Usage pattern (again)

finmason

- Do not expect how the view will be used
  - I’ve created useless index, just because “expected” predicate on trade_date column, it was not the use case…

```
Indexes:
    "asset_price_daily_pkey" PRIMARY KEY, btree (asset_price_daily_id), tablespace "index01"
    "asset_price_daily_p_currency_idx" gin (p_currency), tablespace "index01"
    "asset_price_daily_trade_date_assetid_idx" btree (trade_date, assetid), tablespace "index01"
```

- Reach the person requesting a feature and get as much as possible details, how he/she will use it (code, API, human being user…)
  - **Exactly one asset (asset_id) price history will be retrieved**

Redefine/precise requirement if needed

finmason

- There probably will not be exchange rates from USD to EUR before 1 January 2002
- There is no guarantee that all prices for a given asset will be available in a specific currency forever (same asset might have prices in DEM and later in EUR)
- History of available exchange rates is limited
  - view will not show any data before oldest exchange rate available for a given currency pair
- Weekends (and other potential fx rate gaps) should be converted using latest available previous exchange rate

The prices table

```
 Table "demo1.asset_price_daily"
       Column        |       Type       | Collation | Nullable | Default
---------------------+------------------+-----------+----------+---------
 asset_price_daily_id | bigint           |           | not null |
 assetid              | bigint           |           | not null |
 price                | double precision |           | not null |
 trade_date           | date             |           | not null |
 p_currency           | text             |           | not null |
Indexes:
    "asset_price_daily_pkey" PRIMARY KEY, btree (asset_price_daily_id), tablespace
    "asset_price_daily_p_currency_idx" gin (p_currency), tablespace "index01"
    "asset_price_daily_assetid_trade_date_idx" btree (assetid, trade_date), tablespace
"index01"Check constraints:
    "p_currency_len" CHECK (length(p_currency) <= 3)
Foreign-key constraints:
    "asset_price_daily_currency_fkey" FOREIGN KEY (p_currency) REFERENCES demo2.currency(shortname)
Tablespace: "data01"
```

Price must have a currency

assetid is always used predicate

Foreign key added

Plan before changes on price table

finmason

```
 QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Gather (actual time=791.455..10475.893 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=30506 read=2493237
   I/O Timings: read=6833.744
   ->  Parallel Hash Left Join (actual time=720.518..10445.562 rows=8299 loops=3)
         Hash Cond: ((apd.trade_date = cer.trade_date) AND (apd.p_currency = cer.currency_from))
         Buffers: shared hit=30506 read=2493237
         I/O Timings: read=6833.744
         ->  Parallel Seq Scan on asset_price_daily apd (actual time=145.873..9862.848 rows=8299 loops=3)
               Filter: (assetid = 210235)
               Rows Removed by Filter: 113101986
               Buffers: shared hit=1843 read=2493237
               I/O Timings: read=6833.744
         ->  Parallel Hash (actual time=571.746..571.746 rows=292340 loops=3)
               Buckets: 1048576  Batches: 1  Memory Usage: 49376kB
               Buffers: shared hit=28521
               ->  Parallel Seq Scan on currency_exchange_rate cer (actual time=292.146..449.095 rows=292340 loops=3)
                     Filter: (currency_to = 'EUR'::text)
                     Rows Removed by Filter: 1000585
                     Buffers: shared hit=28521
 Planning Time: 0.522 ms
 Execution Time: 10480.743 ms
```

Plan after changes on price table

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Gather (actual time=373.096..394.177 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=29378
   ->  Parallel Hash Left Join (actual time=353.489..361.825 rows=8299 loops=3)
         Hash Cond: ((apd.trade_date = cer.trade_date) AND (apd.p_currency = cer.currency_from))
         Buffers: shared hit=29378
         ->  Parallel Bitmap Heap Scan on asset_price_daily apd (actual time=1.567..2.935 rows=8299 loops=3)
               Recheck Cond: (assetid = 210235)
               Heap Blocks: exact=156
               Buffers: shared hit=761
               ->  Bitmap Index Scan on asset_price_daily_assetid_trade_date_idx (actual time=1.438..1.438 rows=24897 loops=1)
                     Index Cond: (assetid = 210235)
                     Buffers: shared hit=175
         ->  Parallel Hash (actual time=349.534..349.534 rows=292340 loops=3)
               Buckets: 1048576  Batches: 1  Memory Usage: 49376kB
               Buffers: shared hit=28521
               ->  Parallel Seq Scan on currency_exchange_rate cer (actual time=17.082..227.964 rows=292340 loops=3)
                     Filter: (currency_to = 'EUR'::text)
                     Rows Removed by Filter: 1000585
                     Buffers: shared hit=28521
 Planning Time: 0.454 ms
 Execution Time: 398.030 ms
```

Index column order matters

Declarative Data Integrity – fx rates table

finmason

```
 Table "demo1.currency_exchange_rate"
          Column           |  Type   | Collation | Nullable |
---------------------------+---------+-----------+----------+
 currency_exchange_rate_id | bigint  |           | not null |
 ric                       | text    |           | not null |
 currency_from             | text    |           | not null |
 currency_to               | text    |           | not null |
 trade_date                | date    |           | not null |
 fxrate                    | real    |           |          |
Indexes:
    "currency_exchange_rate_pkey" PRIMARY KEY, btree (currency_exchange_rate_id), tablespace "index01"
    "currency_exchange_rate_key" UNIQUE CONSTRAINT, btree (ric, currency_from, currency_to, trade_date),
tablespace "index01"
    "currency_exchange_rate_currency_from_idx" btree (currency_from), tablespace "index01"
    "currency_exchange_rate_currency_to_idx" btree (currency_to), tablespace "index01"
    "currency_exchange_rate_ric_idx" btree (ric), tablespace "index01"
    "currency_exchange_rate_trade_date_idx" btree (trade_date), tablespace "index01"
Tablespace: "data01"
```

Float4, Nullable!

Weird column in UQ constraint and columns order

Are all these indexes used?

Declarative Data Integrity fixes

finmason

Add constraints and refactor unique constrain (column order, removed some technology code column).

```
-- fxrate is mandatory
ALTER TABLE demo2.currency_exchange_rate ALTER COLUMN fxrate SET NOT NULL;

-- replace UNIQUE constraint
ALTER TABLE demo2.currency_exchange_rate DROP CONSTRAINT currency_exchange_rate_key;
ALTER TABLE demo2.currency_exchange_rate ADD CONSTRAINT
    currency_exchange_rate_key UNIQUE (trade_date, currency_to, currency_from)
    USING INDEX TABLESPACE index01;

-- add foreign keys to ensure data validity
ALTER TABLE demo2.currency_exchange_rate
   ADD CONSTRAINT currency_exchange_rate_currency_from_fkey FOREIGN KEY (currency_from)
   REFERENCES demo2.currency(shortname);

ALTER TABLE demo2.currency_exchange_rate
   ADD CONSTRAINT currency_exchange_rate_currency_to_fkey FOREIGN KEY (currency_to)
   REFERENCES demo2.currency(shortname);
```

Plan after changes on fxrate table

finmason

```
 QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------
 Gather (actual time=2.477..126.995 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=75382 read=6843
   I/O Timings: read=69.700
   ->  Nested Loop Left Join (actual time=0.738..87.036 rows=8299 loops=3)
         Buffers: shared hit=75382 read=6843
         I/O Timings: read=69.700
         ->  Parallel Bitmap Heap Scan on asset_price_daily apd (actual time=0.701..2.396 rows=8299 loops=3)
               Recheck Cond: (assetid = 210235)
               Heap Blocks: exact=371
               Buffers: shared hit=757
               ->  Bitmap Index Scan on asset_price_daily_assetid_trade_date_idx (actual time=1.933..1.934 rows=24897 loops=1)
                     Index Cond: (assetid = 210235)
                     Buffers: shared hit=171
         ->  Index Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.009..0.009 rows=0 loops=24897)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
               Buffers: shared hit=74625 read=6843
               I/O Timings: read=69.700
 Planning Time: 0.734 ms
 Execution Time: 130.583 ms
```

Is that enough for 5k concurrent executions?

Include… PostgreSQL 11 feature

finmason

Using recent versions can be beneficial…

Daily exchange rates are loaded using batch process (no frequent updates, i.e. index maintenance costs are ok for us).

Index-Only Scans and Covering Indexes

```
ALTER TABLE demo2.currency_exchange_rate DROP CONSTRAINT currency_exchange_rate_key;

ALTER TABLE demo2.currency_exchange_rate ADD CONSTRAINT
    currency_exchange_rate_key UNIQUE (trade_date, currency_to, currency_from)
    INCLUDE (fxrate) USING INDEX TABLESPACE index01;

ANALYZE demo2.currency_exchange_rate;
```

Plan after with covering index on fxrates

finmason

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------
 Gather (actual time=1.541..68.175 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=68596 read=6916
   I/O Timings: read=24.630
   ->  Nested Loop Left Join (actual time=0.464..42.371 rows=8299 loops=3)
         Buffers: shared hit=68596 read=6916
         I/O Timings: read=24.630
         ->  Parallel Bitmap Heap Scan on asset_price_daily apd (actual time=0.435..1.858 rows=8299 loops=3)
               Recheck Cond: (assetid = 210235)
               Heap Blocks: exact=366
               Buffers: shared hit=757
               ->  Bitmap Index Scan on asset_price_daily_assetid_trade_date_idx (actual time=1.173..1.174 rows=24897 loops=1)
                     Index Cond: (assetid = 210235)
                     Buffers: shared hit=171
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.004..0.004 rows=0 loops=24897)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
               Buffers: shared hit=67839 read=6916
               I/O Timings: read=24.630
 Planning Time: 0.467 ms
 Execution Time: 69.804 ms
```

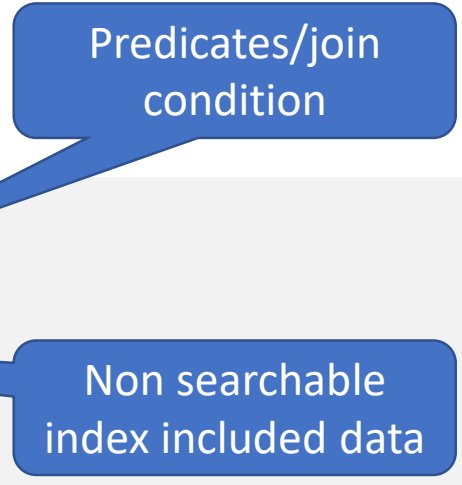
Covering index on prices table

finmason

Price table is also loaded by batch ETL but the reads are so frequent – typical request is whole price history for a given asset.

```
CREATE INDEX asset_price_daily_assetid_currency_idx_ext
    ON demo2.asset_price_daily (assetid, p_currency)
    INCLUDE (trade_date, price)
    TABLESPACE index01;

ANALYZE demo2.asset_price_daily;
```



Plan after with covering index on prices

finmason

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Gather (actual time=1.525..49.319 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=75467
   ->  Nested Loop Left Join (actual time=0.468..31.895 rows=8299 loops=3)
         Buffers: shared hit=75467
         ->  Parallel Bitmap Heap Scan on asset_price_daily apd (actual time=0.443..1.822 rows=8299 loops=3)
               Recheck Cond: (assetid = 210235)
               Heap Blocks: exact=289
               Buffers: shared hit=712
               ->  Bitmap Index Scan on asset_price_daily_assetid_currency_idx_ext (actual time=1.188..1.188 rows=24897 loops=1)
                     Index Cond: (assetid = 210235)
                     Buffers: shared hit=126
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.003..0.003 rows=0 loops=24897)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
               Buffers: shared hit=74755
 Planning Time: 0.353 ms
 Execution Time: 50.896 ms
(19 rows)
```

No benefit from included columns here

One more look at the test SQL…

finmason

```
SELECT apd.asset_price_daily_id, apd.assetid, apd.price / cer.fxrate AS price, apd.trade_date, 'EUR' AS p_currency
FROM demo2.asset_price_daily apd
LEFT JOIN demo2.currency_exchange_rate cer
       ON cer.trade_date = apd.trade_date AND cer.currency_from = apd.p_currency AND cer.currency_to = 'EUR'
WHERE apd.assetid = 210235;
```

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------
 Gather (actual time=1.525..49.319 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=75467
   ->  Nested Loop Left Join (actual time=0.468..31.895 rows=8299 loops=3)
         Buffers: shared hit=75467
         ->  Parallel Bitmap Heap Scan on asset_price_daily apd (actual time=0.443..1.822 rows=8299 loops=3)
               Recheck Cond: (assetid = 210235)
               Heap Blocks: exact=289
               Buffers: shared hit=712
               ->  Bitmap Index Scan on asset_price_daily_assetid_currency_idx_ext (actual time=1.188..1.188 rows=24897 loops=1)
                     Index Cond: (assetid = 210235)
                     Buffers: shared hit=126
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.003..0.003 rows=0 loops=24897)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
               Buffers: shared hit=74755
 Planning Time: 0.353 ms
 Execution Time: 50.896 ms
(19 rows)
```

Remove unnecessary columns

finmason

```
SELECT /*apd.asset_price_daily_id,*/ apd.assetid, apd.price / cer.fxrate AS price, apd.trade_date, 'EUR' AS
p_currency
FROM demo2.asset_price_daily apd
LEFT JOIN demo2.currency_exchange_rate cer
       ON cer.trade_date = apd.trade_date AND cer.currency_from = apd.p_currency AND cer.currency_to = 'EUR'
WHERE apd.assetid = 210235;
```

Index Only Scan on prices

```
QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
Nested Loop Left Join (actual time=0.029..56.476 rows=24897 loops=1)
  Buffers: shared hit=74928
  ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.017..4.111 rows=24897 loops=1)
        Index Cond: (assetid = 210235)
        Heap Fetches: 0
        Buffers: shared hit=177
  ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.002..0.002 rows=0 loops=24897)
        Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
        Heap Fetches: 0
        Buffers: shared hit=74751
Planning Time: 0.392 ms
Execution Time: 57.603 ms
```

No significant improvement

5k parallel executions…

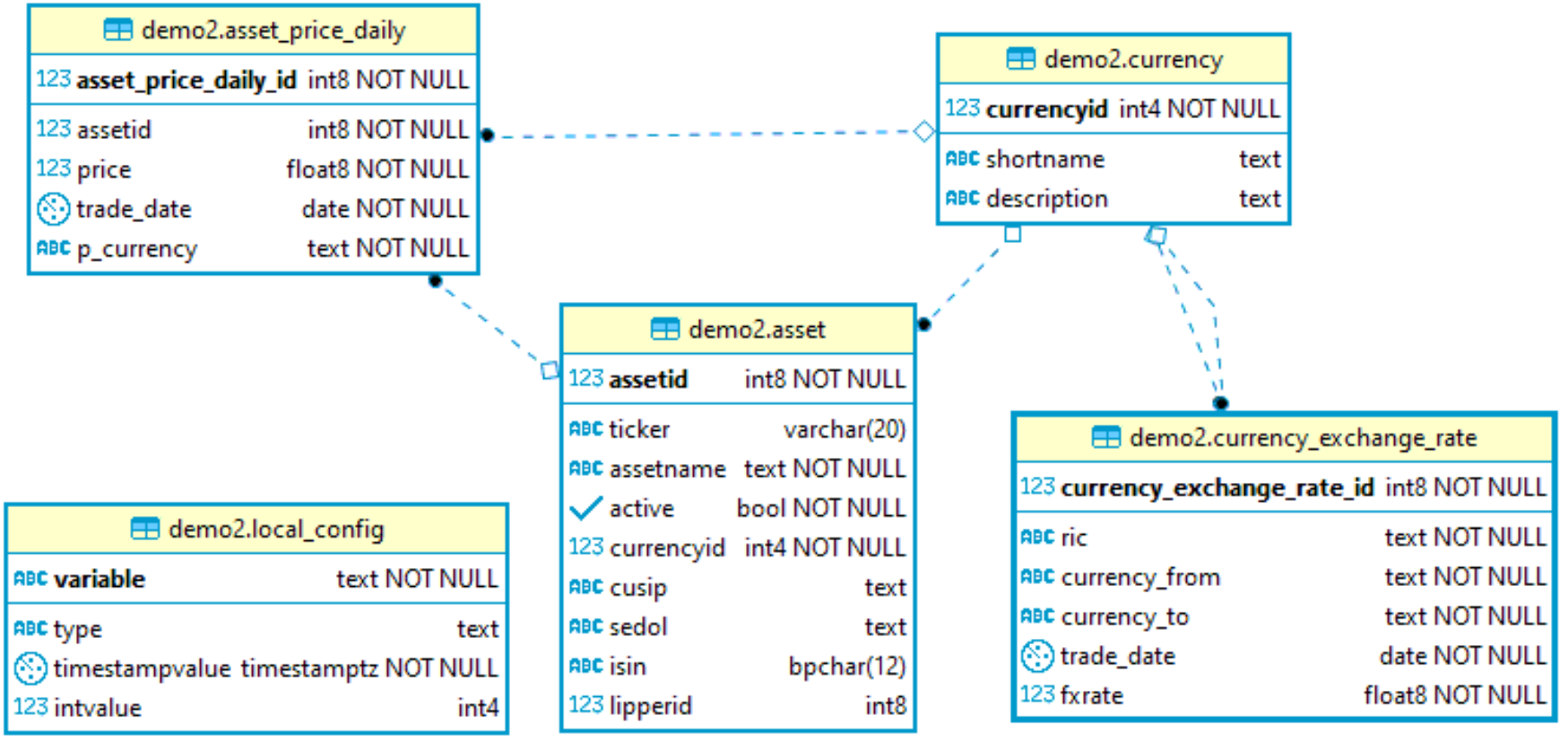
```
Gather (actual time=1.525..49.319 rows=24897 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   Buffers: shared hit=75467
   ->  Nested Loop Left Join (actual time=0.468..31.895 rows=8299 loops=3)
         Buffers: shared hit=75467
         ->  Parallel Bitmap Heap Scan on asset_price_daily apd (actual time=0.443..1.822 rows=8299 loops=3)
               Recheck Cond: (assetid = 210235)
               Heap Blocks: exact=289
               Buffers: shared hit=712
               ->  Bitmap Index Scan on asset_price_daily_assetid_currency_idx_ext (actual time=1.188..1.188 rows=24897 loops=1)
                     Index Cond: (assetid = 210235)
                     Buffers: shared hit=126
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.003..0.003 rows=0 loops=24897)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
               Buffers: shared hit=74755
 Planning Time: 0.353 ms
 Execution Time: 50.896 ms
```

Some CPU cycles was saved by reducing buffers visited, heap is not involved at all, despite slightly longer execution time

```
Nested Loop Left Join (actual time=0.029..56.476 rows=24897 loops=1)
   Buffers: shared hit=74928
   ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.017..4.111 rows=24897 loops=1)
         Index Cond: (assetid = 210235)
         Heap Fetches: 0
         Buffers: shared hit=177
   ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.002..0.002 rows=0 loops=24897)
         Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
         Heap Fetches: 0
         Buffers: shared hit=74751
 Planning Time: 0.392 ms
 Execution Time: 57.603 ms
```

Declarative integrity looks better

finmason



From prototype back to requested view

finmason

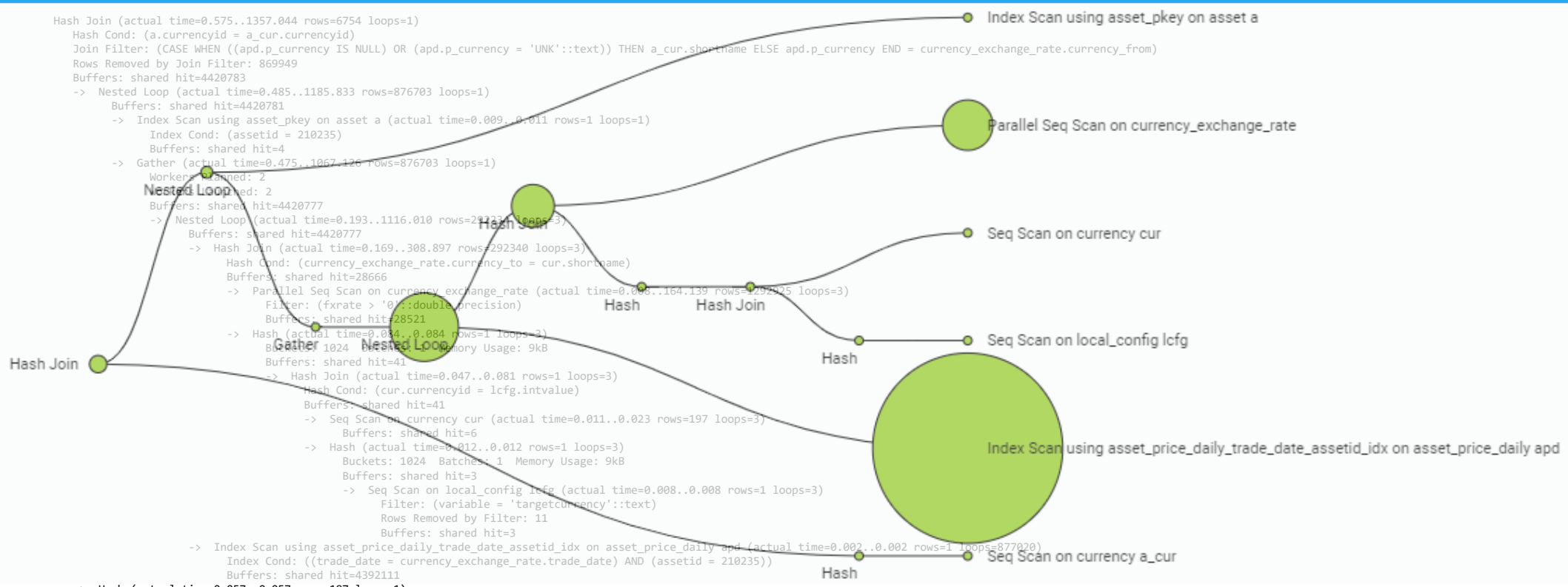
- Basic query seems to be optimized to benefit from indexes
- Hardcoded target currency needs to be replaced by configuration table data (same view DDL in all currency environments)
- For missing exchange rates latest previously available rate should be used (if any) – this is happening in real world, so we must deal with such fact in our view definition
- Data integrity – no need to fix potential issues in query for asset prices
  - Currency in price table and exchange rates table shares same lookup table of currencies
  - No exchange rate entries without fxrate value
  - No price entries without price value

Initial implementation (found in git repository, sometimes returning wrong data)

finmason

```
explain (analyze, buffers, costs off)
SELECT df.assetid, df.price as price, df.trade_date, cer.currency_to p_currency, df.price / fxrate as converted_value
/* Get Price For Asset when the currency is unknown or missing replace the p_currency with the currency identified in asset table */
FROM  (
  SELECT apd.assetid, apd.trade_date,
    CASE WHEN apd.p_currency = 'GBX' THEN apd.price / 100 ELSE apd.price END as price,
    CASE WHEN apd.p_currency IS NULL OR apd.p_currency = 'UNK' THEN a_cur.shortname ELSE apd.p_currency END as p_currency
  FROM demo1.asset_price_daily apd
  /* Get the Asset currencyid in case a currency_code is NULL or UNKNOWN */
  JOIN asset a ON apd.assetid = a.assetid
  JOIN currency a_cur ON a_cur.currencyid = a.currencyid
) df
/* Get the Daily Currency Exchange Rates */
LEFT JOIN (
  SELECT trade_date, currency_from,  currency_to, fxrate FROM demo1.currency_exchange_rate WHERE fxrate > 0
) cer ON df.p_currency = cer.currency_from AND df.trade_date = cer.trade_date
/* Get the Currency Code for the Current Database */
JOIN demo1.currency cur ON cur.shortname = cer.currency_to
JOIN demo1.local_config lcfg ON lcfg.intvalue = cur.currencyid
WHERE lcfg.variable = 'targetcurrency'
and df.assetid = 210235;
```

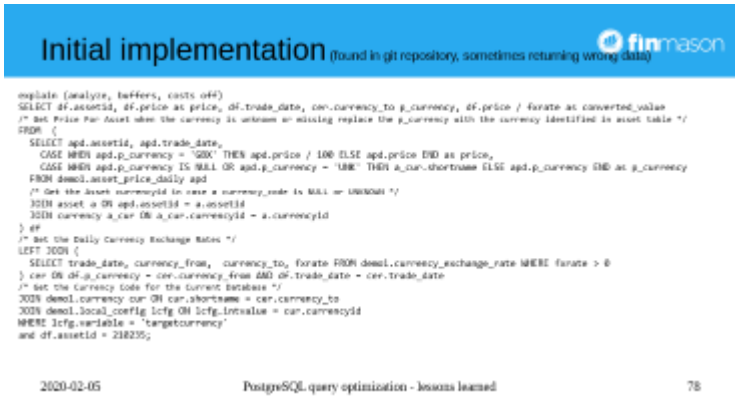
Plan use some of new indexes



Q1 – no hardcoded target currency

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       cer.currency_to p_currency,
       apd.price / cer.fxrate as converted_value
FROM demo2.asset_price_daily apd
 /* Get the Daily Currency Exchange Rates */
LEFT JOIN demo2.currency_exchange_rate cer
       ON apd.p_currency = cer.currency_from AND apd.trade_date = cer.trade_date
/* Get the Currency Code for the Current Database */
JOIN demo2.currency cur
       ON cur.shortname = cer.currency_to
JOIN demo2.local_config lcfg
       ON lcfg.intvalue = cur.currencyid
WHERE
       lcfg.variable = 'targetcurrency'
       AND apd.assetid = 210235;
```



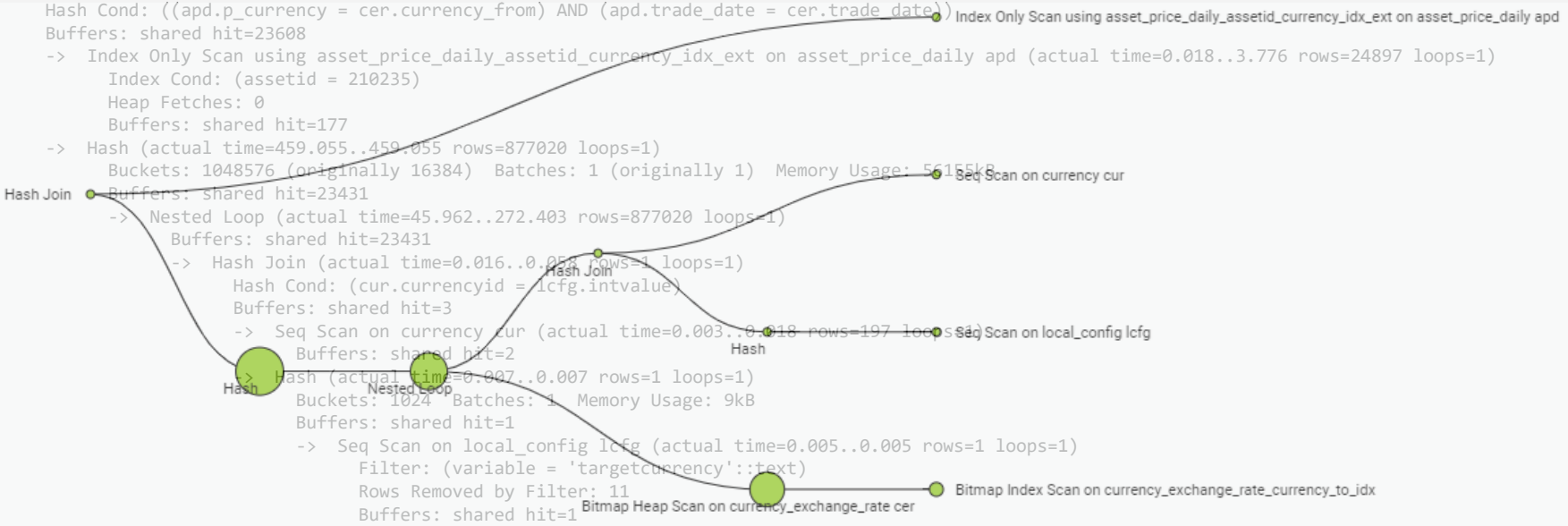
Get target currency from configuration table

Q1 – plan

finmason

```
QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------
Hash Join (actual time=459.105..470.803 rows=6754 loops=1)
  Hash Cond: ((apd.p_currency = cer.currency_from) AND (apd.trade_date = cer.trade_date))
  Buffers: shared hit=23608
  ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.018..3.776 rows=24897 loops=1)
        Index Cond: (assetid = 210235)
        Heap Fetches: 0
        Buffers: shared hit=177
  ->  Hash (actual time=459.055..459.055 rows=877020 loops=1)
        Buckets: 1048576 (originally 16384)  Batches: 1 (originally 1)  Memory Usage: 56165kB
        Buffers: shared hit=23431
        ->  Nested Loop (actual time=45.962..272.403 rows=877020 loops=1)
              Buffers: shared hit=23431
              ->  Hash Join (actual time=0.016..0.059 rows=1 loops=1)
                    Hash Cond: (cur.currencyid = lcfg.intvalue)
                    Buffers: shared hit=3
                    ->  Seq Scan on currency cur (actual time=0.003..0.018 rows=197 loops=1)
                          Buffers: shared hit=2
                    ->  Hash (actual time=0.007..0.007 rows=1 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 9kB
                          Buffers: shared hit=1
                          ->  Seq Scan on local_config lcfg (actual time=0.005..0.005 rows=1 loops=1)
                                Filter: (variable = 'targetcurrency'::text)
                                Rows Removed by Filter: 11
                                Buffers: shared hit=1
              ->  Bitmap Heap Scan on currency_exchange_rate cer (actual time=45.941..153.075 rows=877020 loops=1)
                    Recheck Cond: (currency_to = cur.shortname)
                    Heap Blocks: exact=20379
                    Buffers: shared hit=23428
                    ->  Bitmap Index Scan on currency_exchange_rate_currency_to_idx (actual time=42.794..42.794 rows=877020 loops=1)
                          Index Cond: (currency_to = cur.shortname)
                          Buffers: shared hit=3049
Planning Time: 0.700 ms
 Execution Time: 474.248 ms
(33 rows)
```

BUG, 24k expected



Exchange rate function – solve weekends

finmason

Returns exchange rate for given trade date or latest previous fxrate if not exists for trade_date requested – “fills” fxrate gaps.

```
CREATE OR REPLACE FUNCTION demo2.get_fx_rate(p_trade_date date, p_currency_from text, currency_to text)
 RETURNS double precision
 LANGUAGE SQL
AS $function$
    SELECT cer.fxrate
    FROM demo2.currency_exchange_rate cer
    WHERE cer.trade_date <= $1
        AND cer.currency_from = $2
        AND cer.currency_to = $3
    ORDER BY cer.trade_date DESC
    LIMIT 1;
$function$;
```

View with target environment currency

finmason

```
CREATE OR REPLACE VIEW demo2.base_currency AS
 SELECT cur.shortname AS base_currency
   FROM demo2.local_config lcfg
     JOIN demo2.currency cur ON lcfg.intvalue = cur.currencyid
  WHERE lcfg.variable = 'targetcurrency'::text;
```

Simple view returning environment local configured currency

```
explain (analyze, buffers, costs off) SELECT base_currency FROM demo2.base_currency;
                                    QUERY PLAN
-------------------------------------------------------------------------------------
 Hash Join (actual time=0.022..0.079 rows=1 loops=1)
   Hash Cond: (cur.currencyid = lcfg.intvalue)
   Buffers: shared hit=3
   ->  Seq Scan on currency cur (actual time=0.005..0.026 rows=197 loops=1)
         Buffers: shared hit=2
   ->  Hash (actual time=0.009..0.009 rows=1 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         Buffers: shared hit=1
         ->  Seq Scan on local_config lcfg (actual time=0.007..0.007 rows=1 loops=1)
               Filter: (variable = 'targetcurrency'::text)
               Rows Removed by Filter: 11
               Buffers: shared hit=1
 Planning Time: 0.159 ms
 Execution Time: 0.106 ms
```

Q2 – fix missing rows issue in Q1

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       bc.base_currency p_currency,
       apd.price / demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency) as converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.assetid = 210235;

 QUERY PLAN
-------------------------------------------------------------------------------------------
 Nested Loop Left Join (actual time=0.173..401.689 rows=24897 loops=1)
   Buffers: shared hit=156412
…
Execution Time: 575.120 ms
```

Even the plan looks use indexes again

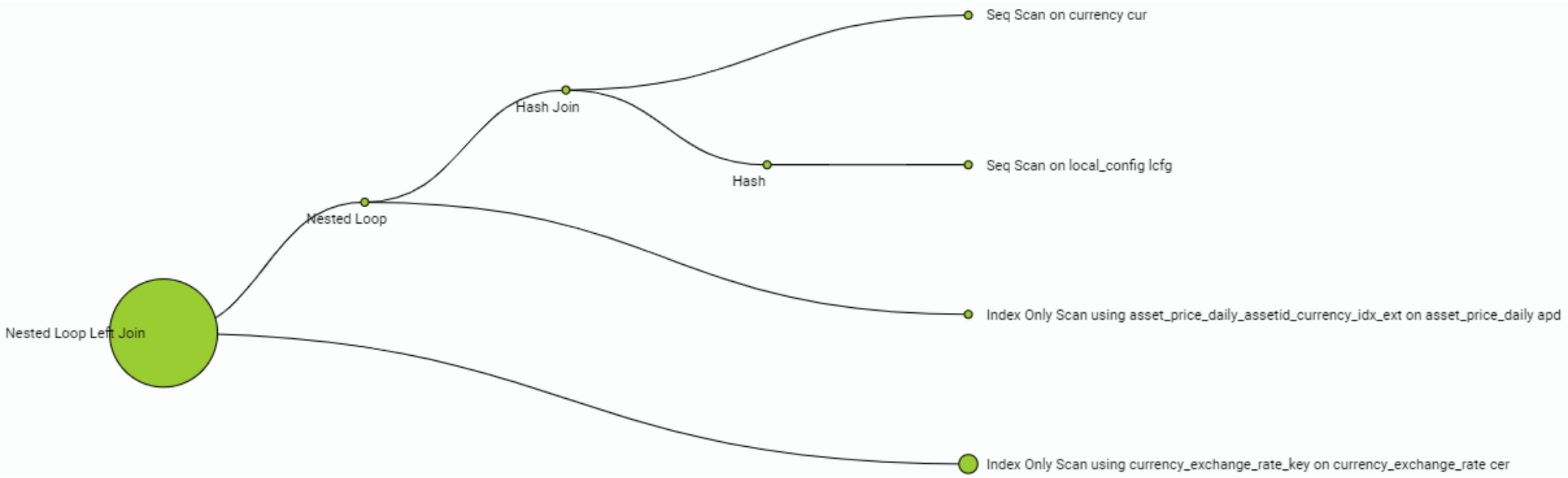
finmason

```
Nested Loop Left Join (actual time=0.173..401.689 rows=24897 loops=1)
   Buffers: shared hit=156412
   ->  Nested Loop (actual time=0.033..10.604 rows=24897 loops=1)
         Buffers: shared hit=180
        ->  Hash Join (actual time=0.016..0.079 rows=1 loops=1)
              Hash Cond: (cur.currencyid = lcfg.intvalue)
              Buffers: shared hit=3
              ->  Seq Scan on currency cur (actual time=0.003..0.026 rows=197 loops=1)
                    Buffers: shared hit=2
              ->  Hash (actual time=0.006..0.006 rows=1 loops=1)
                    Buffers: shared hit=1
                    ->  Seq Scan on local_config lcfg (actual time=0.005..0.005 rows=1 loops=1)
                          Filter: (variable = 'targetcurrency'::text)
                          Rows Removed by Filter: 11
                          Buffers: shared hit=1
         ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.017..6.565
rows=24897 loops=1)
               Index Cond: (assetid = 210235)
               Heap Fetches: 0
               Buffers: shared hit=177
   ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.002..0.002 rows=0 loops=24897)
         Index Cond: ((trade_date = apd.trade_date) AND (currency_to = cur.shortname) AND (currency_from = apd.p_currency))
         Heap Fetches: 0
         Buffers: shared hit=74751
 Planning Time: 0.550 ms
 Execution Time: 403.403 ms
(26 rows)
```

Execution time is not that nice…

Even the plan looks nice again

finmason



Functional tests / requirements

finmason

- Evaluate behavior when prices are converted
  - To different currency than source price currency
  - To same currency as source price currency
- Prices must be converted even for trade date without exchange rate available (weekends, public holidays…)
- Handle properly cases where a given asset price currency changes over time
- Exclude result rows with NULL values in converted price
  - No suitable exchange rate exists

Q2 – Test – USD prices converted to EUR

finmason

```
update demo2.local_config set intvalue = 5 where variable = 'targetcurrency'::text;
```

EUR currencyid value

```
SELECT count(*), count(converted_value) FROM (
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       bc.base_currency p_currency,
       apd.price / demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency) as converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.assetid = 210235 ) Q;
 count | count
-------+-------
 24897 |  6773
(1 row)

Time: 310.308 ms
```

Only 6.7k rows with converted prices out of 24.8k
No exchange rates available before 1994-01-03
see Validate data with your expectations

Q3 – USD prices converted to EUR

finmason

```
update demo2.local_config set intvalue = 5 where variable = 'targetcurrency'::text;
```

EUR currencyid value

```
anl_usdparallel_prod=# SELECT count(*), count(converted_value) FROM (
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, bc.base_currency p_currency,
       apd.price / demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency) as converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair */
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE
       /* Return only prices starting with first fxrate available */
       apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min
           FROM demo2.currency_exchange_rate cerdt
           WHERE cerdt.currency_to = bc.base_currency )
      AND apd.assetid = 210235 ) Q;
 count | count
-------+-------
  6773 |  6773
(1 row)

Time: 334.221 ms
```

Nice, rows with null prices are not returned

Q3 – Test – USD prices converted to USD

finmason

```
update demo2.local_config set intvalue = 1 where variable = 'targetcurrency'::text;
```

USD currencyid value

```
explain (analyze, buffers, costs off)
SELECT apd.assetid,
        apd.trade_date,
        apd.price,
        apd.p_currency,
        bc.base_currency p_currency,
       apd.price / demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency) as
converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair */
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
/* Return only prices starting with first fxrate available */
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min
           FROM demo2.currency_exchange_rate cerdt
           WHERE cerdt.currency_to = bc.base_currency )
   AND apd.assetid = 210235;
```

Q3 – Test – USD prices converted to USD

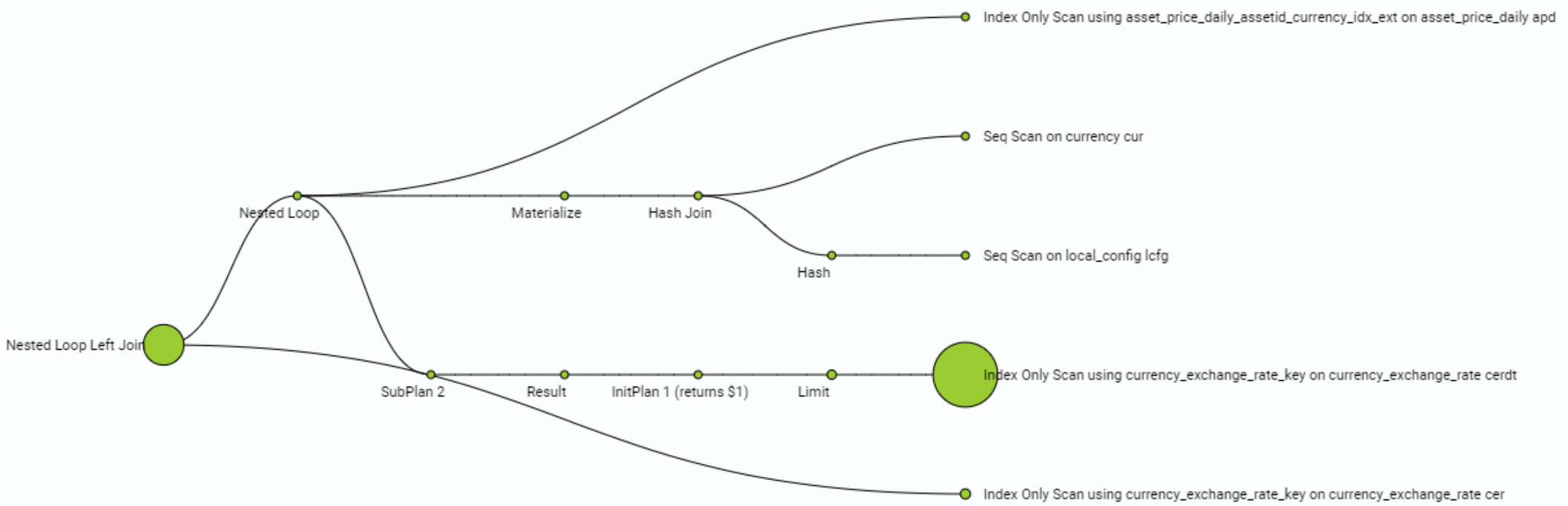
```
QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
Nested Loop Left Join (actual time=0.210..353.486 rows=6773 loops=1)
  Buffers: shared hit=122360
  ->  Nested Loop (actual time=0.056..215.977 rows=6773 loops=1)
        Join Filter: (apd.trade_date >= (SubPlan 2))
        Rows Removed by Join Filter: 18124
        Buffers: shared hit=74872
        ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.017..3.982 rows=24897 loops=1)
              Index Cond: (assetid = 210235)
        ->  Materialize (actual time=0.000..0.000 rows=1 loops=24897)
              Buffers: shared hit=3
              /* base_currency plan rows */
        SubPlan 2
          ->  Result (actual time=0.008..0.008 rows=1 loops=24897)
                Buffers: shared hit=74692
                InitPlan 1 (returns $1)
                  ->  Limit (actual time=0.007..0.008 rows=1 loops=24897)
                        Buffers: shared hit=74692
                        ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=0.007..0.007 rows=1 loops=24897)
                              Index Cond: ((trade_date IS NOT NULL) AND (currency_to = cur.shortname))
                              Heap Fetches: 0
                              Buffers: shared hit=74692
  ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.004..0.004 rows=1 loops=6773)
        Index Cond: ((trade_date = apd.trade_date) AND (currency_to = cur.shortname) AND (currency_from = apd.p_currency))
        Heap Fetches: 0
        Buffers: shared hit=20379
Planning Time: 0.665 ms
Execution Time: 354.010 ms
(40 rows)
```

Is it necessary to exclude USD prices converted to USD just because lack of exchange rates?

Like EUR conversion, can it be faster, please?

Is the nested loop for 7k rows that slow?

finmason



Track functions

Look for pg_stat_xact_user_functions view, it might help us to bring some light.

See also track_functions configuration parameter, default is none, for this test it needs to be changed at least in session context.

Q3 – Function calls are not for free

finmason

```
=> begin transaction;
BEGIN
=> select * from pg_stat_xact_user_functions;
  funcid   | schemaname |  funcname   | calls | total_time | self_time
-----------+------------+-------------+-------+------------+-----------
 181415538 | demo2      | get_fx_rate |     0 |          0 |         0

=> explain (analyze, buffers, costs off)
SELECT apd.assetid,
…
 Planning Time: 0.713 ms
 Execution Time: 351.413 ms

=> select * from pg_stat_xact_user_functions;
  funcid   | schemaname |  funcname   | calls | total_time | self_time
-----------+------------+-------------+-------+------------+------------
 181415538 | demo2      | get_fx_rate |  6773 | 107.244669 | 107.244669

=> commit work;
COMMIT
```

Q4 – convert price only when needed

finmason

```
 explain (analyze, buffers, costs off)
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       bc.base_currency p_currency,
       CASE
           WHEN apd.p_currency = bc.base_currency
               THEN apd.price
           ELSE
               apd.price / demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency)
        END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min
           FROM demo2.currency_exchange_rate cerdt
           WHERE cerdt.currency_to = bc.base_currency
      )
      AND apd.assetid = 210235;
```

Q4 – USD → USD no function calls

```
=> begin transaction;
BEGIN
=> select * from pg_stat_xact_user_functions;
  funcid   | schemaname |  funcname   | calls | total_time | self_time
-----------+------------+-------------+-------+------------+-----------
 181415538 | demo2      | get_fx_rate |     0 |          0 |         0

=> explain (analyze, buffers, costs off)
SELECT apd.assetid,
…
 Nested Loop Left Join (actual time=0.063..225.638 rows=6773 loops=1)
…
 Planning Time: 0.668 ms
 Execution Time: 226.117 ms

=> select * from pg_stat_xact_user_functions;
  funcid   | schemaname |  funcname   | calls | total_time | self_time
-----------+------------+-------------+-------+------------+-----------
 181415538 | demo2      | get_fx_rate |     0 |          0 |         0

=> commit work;
COMMIT
```

USD → USD conversion might not rely on exchange rates availability… Function bug in case of no conversion, there is no need to limit price history by exchange rate availability.

Query response time is better, can we do more?

Q5 – return target currency rows as they are

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, bc.base_currency p_currency,
       CASE WHEN apd.p_currency = bc.base_currency THEN apd.price
           ELSE apd.price / demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency)
        END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE cerdt.currency_to = bc.base_currency)
       AND apd.p_currency != bc.base_currency AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       bc.base_currency p_currency,
       apd.price AS converted_value
FROM demo2.asset_price_daily apd
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
WHERE apd.p_currency = bc.base_currency AND apd.assetid = 210235;
```

Q5 – execution plan (USD → USD)

finmason

```
 QUERY PLAN
-------------------------------------------------------------------------------------------
 Append (actual time=10.876..19.426 rows=24897 loops=1)
   Buffers: shared hit=360
    ->  Nested Loop Left Join (actual time=10.832..10.832 rows=0 loops=1)
          Buffers: shared hit=180
          ->  Nested Loop (actual time=10.832..10.832 rows=0 loops=1)
                Join Filter: ((apd.p_currency <> cur.shortname) AND (apd.trade_date >= (SubPlan 2)))
                Rows Removed by Join Filter: 24897
                ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.018..3.680 rows=24897 loops=1)
                      Index Cond: (assetid = 210235)
                ->  Materialize (actual time=0.000..0.000 rows=1 loops=24897)
                      Buffers: shared hit=3
                      /* base currency plan rows */

                SubPlan 2
                  ->  Result (never executed)
                        /* min trade_date currency exchange rate plan rows */
          ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (never executed)
->  Nested Loop (actual time=0.043..6.671 rows=24897 loops=1)
          Buffers: shared hit=180
          ->  Hash Join (actual time=0.020..0.057 rows=1 loops=1)
                Hash Cond: (cur_1.currencyid = lcfg_1.intvalue)
                Buffers: shared hit=3
                /* base currency plan rows */
->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=0.020..3.977 rows=24897 loops=1)
                Index Cond: ((assetid = 210235) AND (p_currency = cur_1.shortname))
                Heap Fetches: 0
                Buffers: shared hit=177
 Planning Time: 0.980 ms
 Execution Time: 20.596 ms
(56 rows)
```

All expected rows returned – functional defect solved!

Nice improvement for “no conversion needed” prices.

Q5 – execution plan (USD → EUR)

finmason

EUR currencyid value

```
update demo2.local_config set intvalue = 5 where variable = 'targetcurrency'::text;
```

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Append (actual time=0.207..352.886 rows=6773 loops=1)
   Buffers: shared hit=122367
   ->  Nested Loop Left Join (actual time=0.206..352.078 rows=6773 loops=1)
...
 ->  Nested Loop (actual time=0.071..0.071 rows=0 loops=1)
         Buffers: shared hit=7
...
 Planning Time: 0.957 ms
 Execution Time: 353.377 ms
(60 rows)

Time: 354.916 ms
```

Expected row count

No significant degradation compared to query version 3 (Time: 334.221 ms)

Q6 – call get_fx_rate() only when needed

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, bc.base_currency p_currency,
  CASE WHEN apd.p_currency = bc.base_currency THEN apd.price
  ELSE
    apd.price / coalesce( cer.fxrate, demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency))
  END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= (
        SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt
        WHERE cerdt.currency_to = bc.base_currency
      )
      AND apd.p_currency != bc.base_currency AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       bc.base_currency p_currency,
       apd.price AS converted_value
FROM demo2.asset_price_daily apd
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
WHERE apd.p_currency = bc.base_currency AND apd.assetid = 210235;
```

Q6 – execution plan (USD → EUR)

```
 QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Append (actual time=0.068..244.922 rows=6773 loops=1)

 Buffers: shared hit=95351
   ->  Nested Loop Left Join (actual time=0.067..244.159 rows=6773 loops=1)
...
 ->  Nested Loop (actual time=0.078..0.078 rows=0 loops=1)
         Buffers: shared hit=7
...
 Planning Time: 0.934 ms
 Execution Time: 245.424 ms
```

query version 3: 334.221 ms
query version 5: 354.916 ms

```
=# select * from pg_stat_xact_user_functions;
  funcid   | schemaname |  funcname   | calls | total_time | self_time
-----------+------------+-------------+-------+------------+-----------
 181415538 | demo2      | get_fx_rate |    19 |   0.659637 |  0.659637
```

These call were the only necessary once (missing fx rate for a trade date

Know your data and optimize

finmason

- Prices in same currency as environment target are returned directly using index only scan
- Function calls were minimized by
  - Starting conversion from first available exchange rate ignoring older prices we are not able to convert anyway
  - Function is called only if exchange rate for given day is missing
- We do not have a box with 5k CPU cores available, extreme load is expected therefore not only response time, but also resource consumption is important for tuning for 5k concurrent sessions

Can we make it better?

finmason

Is there something to remove from our SQL?

The prototype SQL response time was `57.603 ms` to return 24k rows of converted prices (including nulls for missing exchange rates).

What is the significant difference?

Function calls were already minimized…

Image by Wokandapix from Pixbay.

Prototype and query version 6

finmason

```
SELECT apd.assetid, apd.price / cer.fxrate AS price, apd.trade_date, 'EUR' AS p_currency
FROM demo2.asset_price_daily apd
LEFT JOIN demo2.currency_exchange_rate cer
       ON cer.trade_date = apd.trade_date AND cer.currency_from = apd.p_currency AND cer.currency_to = 'EUR'
WHERE apd.assetid = 210235;
```

```
explain (analyze, buffers, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, bc.base_currency p_currency,
  CASE WHEN apd.p_currency = bc.base_currency THEN apd.price
  ELSE
    apd.price / coalesce( cer.fxrate, demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency))
  END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = bc.base_currency)
      AND apd.p_currency != bc.base_currency AND apd.assetid = 210235
UNION ALL
...
```

Hardcoded currency in query version 6

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, 'EUR' p_currency,
       CASE WHEN apd.p_currency = 'EUR' THEN apd.price
           ELSE apd.price / coalesce( cer.fxrate, demo2.get_fx_rate(apd.trade_date, apd.p_currency, 'EUR'))
        END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND 'EUR' = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = 'EUR')
      AND apd.p_currency != 'EUR' AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid,
       apd.trade_date,
       apd.price,
       apd.p_currency,
       'EUR' p_currency,
       apd.price AS converted_value
FROM demo2.asset_price_daily apd
WHERE apd.p_currency = 'EUR' AND apd.assetid = 210235;
```

Just for test…

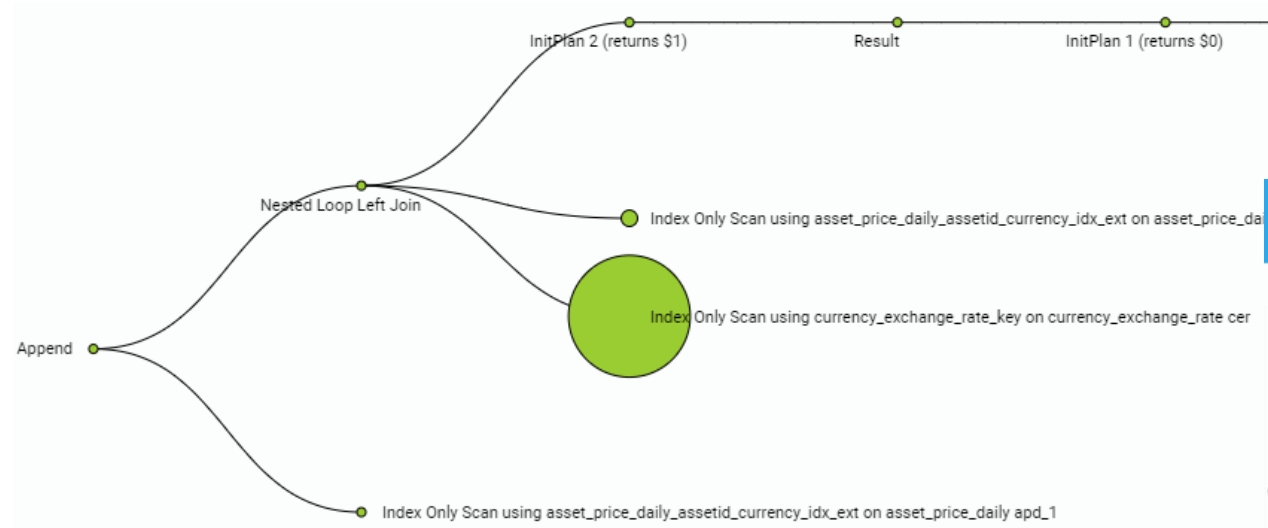
Hardcoded currency in query version 6

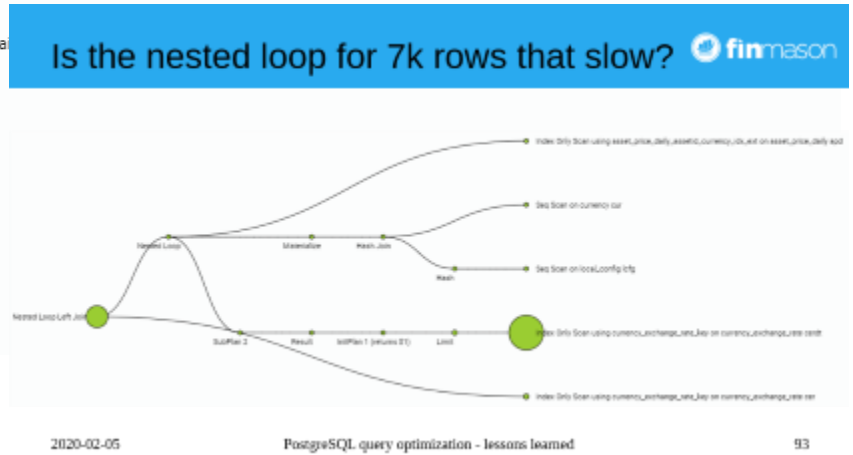
finmason

```
Append (actual time=0.046..33.719 rows=6773 loops=1)
   Buffers: shared hit=20657
   ->  Nested Loop Left Join (actual time=0.045..33.123 rows=6773 loops=1)
         Buffers: shared hit=20653
         InitPlan 2 (returns $1)
           ->  Result (actual time=0.014..0.014 rows=1 loops=1)
                 Buffers: shared hit=4
                 InitPlan 1 (returns $0)
                   ->  Limit (actual time=0.012..0.012 rows=1 loops=1)
                         Buffers: shared hit=4
                         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=0.010..0.011 rows=1 loops=1)
                               Index Cond: ((trade_date IS NOT NULL) AND (currency_to = 'EUR'::text))
                               Heap Fetches: 0
                               Buffers: shared hit=4
         ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.034..3.782 rows=6773 loops=1)
               Index Cond: (assetid = 210235)
               Filter: ((trade_date >= $1) AND (p_currency <> 'EUR'::text))
               Rows Removed by Filter: 18124
               Heap Fetches: 0
               Buffers: shared hit=181
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.004..0.004 rows=1 loops=6773)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'EUR'::text) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
               Buffers: shared hit=20379
   ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=0.014..0.014 rows=0 loops=1)
         Index Cond: ((assetid = 210235) AND (p_currency = 'EUR'::text))
         Heap Fetches: 0
         Buffers: shared hit=4
 Planning Time: 0.582 ms
 Execution Time: 34.089 ms
```

query version 6: 245.424 ms

Q6 – hardcoded target currency – plan





Never executed code is fastest one

finmason

What if we get rid of the join with `base_currency` view and replace hardcoded currency with function providing same result as the view? We'll safe the join operation cost.


```
CREATE OR REPLACE FUNCTION demo2.get_env_base_currency()
 RETURNS text
 LANGUAGE sql
AS $function$
    SELECT cur.shortname FROM demo2.local_config lcfg
    INNER JOIN demo2.currency cur
        ON  ( lcfg.intvalue = cur.currencyid )
    WHERE lcfg.variable = 'targetcurrency';
$function$;
```

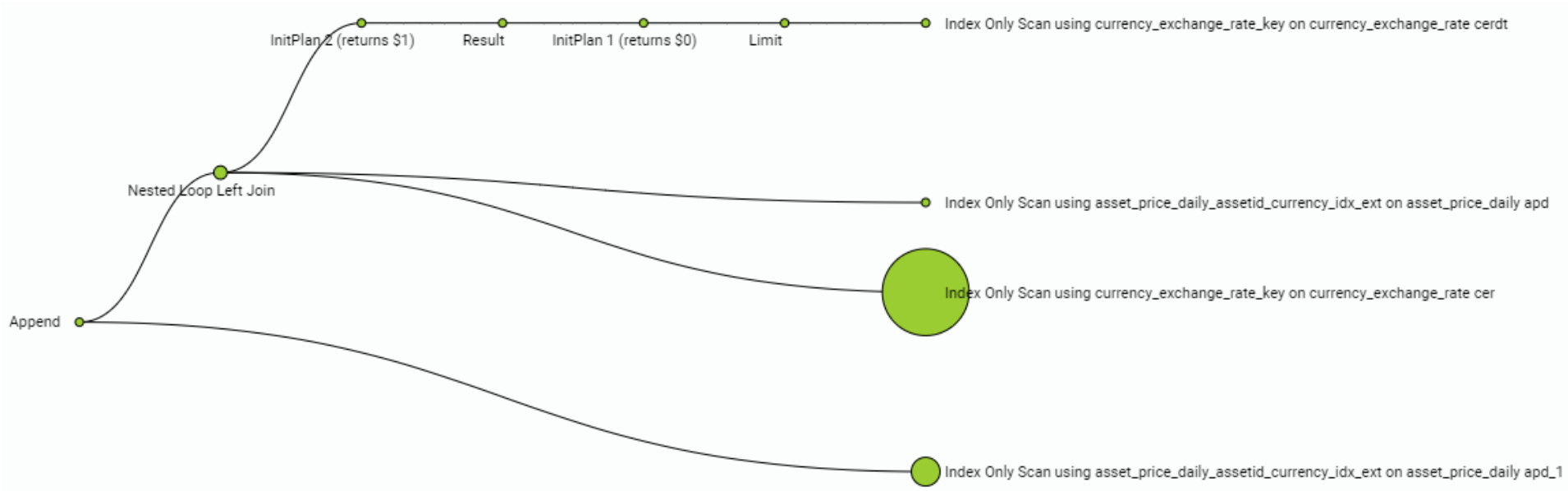
Simple SQL functions returns environment target currency based on local environment configuration table. SQL functions are preferred wherever PLPGSQL is not necessary.

Q7 – function instead of base_currency view

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, demo2.get_env_base_currency() p_currency,
  CASE WHEN apd.p_currency = demo2.get_env_base_currency() THEN apd.price
  ELSE apd.price / coalesce( cer.fxrate,
                             demo2.get_fx_rate(apd.trade_date, apd.p_currency, demo2.get_env_base_currency())
  ) END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
/* INNER JOIN demo2.base_currency bc ON true Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND demo2.get_env_base_currency() = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = demo2.get_env_base_currency())
      AND apd.p_currency != demo2.get_env_base_currency() AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, demo2.get_env_base_currency() p_currency,
       apd.price AS converted_value
FROM demo2.asset_price_daily apd
WHERE apd.p_currency = demo2.get_env_base_currency() AND apd.assetid = 210235;
```

Q7 – plan looks promising



Q7 – plan looks promising except…

```
Append (actual time=1.987..2656.410 rows=6773 loops=1)
   Buffers: shared hit=268578
   ->  Nested Loop Left Join (actual time=1.987..2234.247 rows=6773 loops=1)
         InitPlan 2 (returns $1)
           ->  Result (actual time=1.107..1.108 rows=1 loops=1)
                 InitPlan 1 (returns $0)
                   ->  Limit (actual time=1.105..1.105 rows=1 loops=1)
                         Buffers: shared hit=134
                         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=1.104..1.104 rows=1 loops=1)
                               Index Cond: (trade_date IS NOT NULL)
                               Filter: (currency_to = get_env_base_currency())
                               Rows Removed by Filter: 62
         ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=1.311..144.729 rows=6773 loops=1)
               Index Cond: (assetid = 210235)
               Filter: ((trade_date >= $1) AND (p_currency <> get_env_base_currency()))
               Rows Removed by Filter: 18124
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.134..0.269 rows=1 loops=6773)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_from = apd.p_currency))
               Filter: (get_env_base_currency() = currency_to)
               Rows Removed by Filter: 9
   ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=420.715..420.715 rows=0 loops=1)
         Index Cond: (assetid = 210235)
         Filter: (p_currency = get_env_base_currency())
         Rows Removed by Filter: 24897
 Planning Time: 1.065 ms
```

Execution Time: 2657.312 ms

Performance is far from expected…

finmason

Decompose SQL and do some tests…

USD → USD conversion - simple, expected to be fast

USD → USD

```
update demo2.local_config set intvalue = 1 where variable = 'targetcurrency';
```

```
explain (analyze, costs off, timing off)
SELECT apd.assetid, apd.trade_date,
       get_env_base_currency() p_currency,
       apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = get_env_base_currency() AND apd.assetid = 210235;
```

Minimalistic test

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual rows=24897 loops=1)
   Index Cond: (assetid = 210235)
   Filter: (p_currency = get_env_base_currency())
   Heap Fetches: 0
 Planning Time: 0.138 ms
 Execution Time: 540.718 ms
```

And enjoy your frustration…

What is wrong?

finmason

Image by Steve Buissinne from Pixabay

There must be something wrong

Evaluate the function calls count (or read the query carefully)

```
begin transaction;
explain (analyze, costs off, timing off) SELECT apd.assetid, apd.trade_date, get_env_base_currency() p_currency,
       apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = get_env_base_currency() AND apd.assetid = 210235;

 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual rows=24897 loops=1)
   Index Cond: (assetid = 210235)
   Filter: (p_currency = get_env_base_currency())
   Heap Fetches: 0
 Planning Time: 0.138 ms
 Execution Time: 540.718 ms
(6 rows)
select * from pg_stat_xact_user_functions;
  funcid   | schemaname |       funcname        | calls | total_time | self_time
-----------+------------+-----------------------+-------+------------+------------
 181415537 | demo2      | get_env_base_currency | 49794 | 527.111195 | 527.111195
(2 rows)
```

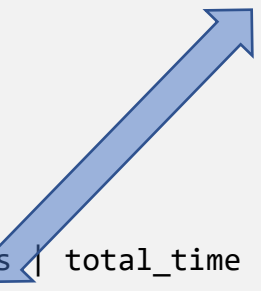
Unnecessary function calls in projection

finmason

```
begin transaction;
explain (analyze, costs off, timing off) SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency,
/*get_env_base_currency() p_currency,*/
       apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = get_env_base_currency() AND apd.assetid = 210235;

 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual rows=24897 loops=1)
   Index Cond: (assetid = 210235)
   Filter: (p_currency = get_env_base_currency())
   Heap Fetches: 0
 Planning Time: 0.123 ms
 Execution Time: 273.478 ms
(6 rows)

select * from pg_stat_xact_user_functions;
  funcid   | schemaname |       funcname        | calls | total_time | self_time
-----------+------------+-----------------------+-------+------------+-----------
 181415537 | demo2      | get_env_base_currency | 24897 |  263.82908 | 263.82908
(2 rows)
```



Still so slow, but reduced function call

Simple prices select is fast

finmason

```
explain (analyze, costs off, timing off) SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency,
/*get_env_base_currency() p_currency,*/
       apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = 'USD' AND apd.assetid = 210235;
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual rows=24897 loops=1)
   Index Cond: ((assetid = 210235) AND (p_currency = 'USD'::text))
   Heap Fetches: 0
 Planning Time: 0.085 ms
 Execution Time: 3.964 ms
(5 rows)
```

Currency hardcoded for test

That response is our friend, how to be fast but not hardcoded?

Never executed code is fastest one

Manual page for

CREATE FUNCTION

Function volatility status can be specified regardless of function language.

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | PARALLEL { UNSAFE | RESTRICTED | SAFE }
    | COST execution_cost
    | ROWS result_rows
    | SUPPORT support_function
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  }
```

```
CREATE OR REPLACE FUNCTION demo2.get_env_base_currency()
 RETURNS text
 STABLE -- we only read the database
 LANGUAGE sql
AS $function$
    SELECT cur.shortname FROM demo2.local_config lcfg
    INNER JOIN demo2.currency cur
        ON  ( lcfg.intvalue = cur.currencyid )
    WHERE lcfg.variable = 'targetcurrency';
$function$;
```

Same currency part work like a charm!

finmason

```
begin transaction;

explain (analyze, costs off, timing off) SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency,
/*get_env_base_currency() p_currency,*/ apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = get_env_base_currency() AND apd.assetid = 210235;

                                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual rows=24897 loops=1)
   Index Cond: ((assetid = 210235) AND (p_currency = get_env_base_currency()))
   Heap Fetches: 0
 Planning Time: 0.372 ms
 Execution Time: 4.041 ms
(5 rows)

select * from pg_stat_xact_user_functions;
  funcid   | schemaname |       funcname        | calls | total_time | self_time
-----------+------------+-----------------------+-------+------------+-----------
 181415537 | demo2      | get_env_base_currency |     3 |   0.350143 |  0.350143
```

This is what we were looking for…

Q7 – repeated test (USD → USD)

finmason

```
explain (analyze, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, /*apd.p_currency,*/ demo2.get_env_base_currency() p_currency,
  CASE WHEN apd.p_currency = demo2.get_env_base_currency() THEN apd.price
  ELSE apd.price / coalesce( cer.fxrate,
                             demo2.get_fx_rate(apd.trade_date, apd.p_currency, demo2.get_env_base_currency())
  ) END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
/* INNER JOIN demo2.base_currency bc ON true Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND demo2.get_env_base_currency() = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = demo2.get_env_base_currency())
      AND apd.p_currency != demo2.get_env_base_currency() AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, /*demo2.get_env_base_currency() p_currency,*/
       apd.price AS converted_value
FROM demo2.asset_price_daily apd
WHERE apd.p_currency = demo2.get_env_base_currency() AND apd.assetid = 210235;
```

Q7 – far from 4 ms response time

```
Append (actual time=75.090..81.307 rows=24897 loops=1)
   ->  Nested Loop Left Join (actual time=74.885..74.885 rows=0 loops=1)
         InitPlan 2 (returns $1)
           ->  Result (actual time=0.116..0.116 rows=1 loops=1)
                 InitPlan 1 (returns $0)
                   ->  Limit (actual time=0.115..0.115 rows=1 loops=1)
                         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=0.114..0.114 rows=1 loops=1)
                               Index Cond: ((trade_date IS NOT NULL) AND (currency_to = get_env_base_currency()))
                               Heap Fetches: 0
         ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=74.884..74.884 rows=0 loops=1)
               Index Cond: (assetid = 210235)
               Filter: ((trade_date >= $1) AND (p_currency <> get_env_base_currency()))
               Rows Removed by Filter: 24897
               Heap Fetches: 0
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (never executed)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = get_env_base_currency()) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
   ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=0.203..4.693 rows=24897 loops=1)
         Index Cond: ((assetid = 210235) AND (p_currency = get_env_base_currency()))
         Heap Fetches: 0
 Planning Time: 1.285 ms
 Execution Time: 82.289 ms
(22 rows)

select * from pg_stat_xact_user_functions;
  funcid   | schemaname |       funcname        | calls | total_time | self_time
-----------+------------+-----------------------+-------+------------+-----------
 181415537 | demo2      | get_env_base_currency |  6781 |  70.989737 | 70.989737
(2 rows)
```

Might it be, the join is not that bad…

finmason

```
explain (analyze, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, /*apd.p_currency,*/ bc.base_currency p_currency,
  CASE WHEN apd.p_currency = bc.base_currency THEN apd.price
  ELSE apd.price / coalesce( cer.fxrate,
                             demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency)
  ) END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
 INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair */
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = demo2.get_env_base_currency())
      AND apd.p_currency != bc.base_currency AND apd.assetid = 210235;
…
Execution Time: 3.976 ms
…
  funcid   | schemaname |       funcname        | calls | total_time | self_time
-----------+------------+-----------------------+-------+------------+-----------
 181415537 | demo2      | get_env_base_currency |     3 |   0.414949 |  0.414949
```

Once joined, use it in projection instead of function calls, we have seen the impact already

Efficient function call, using bc.base_currency raises response time to 10ms

This predicate drops response time to what looks much better

Query ver. 8 - put all together (USD → USD)

finmason

```
explain (analyze, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, /*apd.p_currency,*/ bc.base_currency p_currency,
  CASE WHEN apd.p_currency = bc.base_currency THEN apd.price
  ELSE apd.price / coalesce( cer.fxrate,
                             demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency)
  ) END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
 INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = demo2.get_env_base_currency())
      AND apd.p_currency != bc.base_currency AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, /*get_env_base_currency() p_currency,*/
apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = get_env_base_currency() AND apd.assetid = 210235;
```

Query ver. 8 - put all together (USD → USD)

```
Append (actual time=4.058..10.052 rows=24897 loops=1)
  ->  Nested Loop Left Join (actual time=3.895..3.895 rows=0 loops=1)
        InitPlan 2 (returns $1)
          ->  Result (actual time=0.112..0.113 rows=1 loops=1)
                InitPlan 1 (returns $0)
                  ->  Limit (actual time=0.111..0.111 rows=1 loops=1)
                        ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=0.110..0.110 rows=1 loops=1)
                              Index Cond: ((trade_date IS NOT NULL) AND (currency_to = get_env_base_currency()))
                              Heap Fetches: 0
        ->  Nested Loop (actual time=3.895..3.895 rows=0 loops=1)
              Join Filter: (apd.p_currency <> cur.shortname)
              Rows Removed by Join Filter: 6773
              ->  Hash Join (actual time=0.009..0.044 rows=1 loops=1)
                    Hash Cond: (cur.currencyid = lcfg.intvalue)
                    ->  Seq Scan on currency cur (actual time=0.002..0.015 rows=197 loops=1)
                    ->  Hash (actual time=0.003..0.003 rows=1 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 9kB
                          ->  Seq Scan on local_config lcfg (actual time=0.002..0.003 rows=1 loops=1)
                                Filter: (variable = 'targetcurrency'::text)
                                Rows Removed by Filter: 11
              ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.128..3.406 rows=6773 loops=1)
                    Index Cond: (assetid = 210235)
                    Filter: (trade_date >= $1)
                    Rows Removed by Filter: 18124
                    Heap Fetches: 0
        ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (never executed)
              Index Cond: ((trade_date = apd.trade_date) AND (currency_to = cur.shortname) AND (currency_from = apd.p_currency))
              Heap Fetches: 0
  ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=0.161..4.483 rows=24897 loops=1)
        Index Cond: ((assetid = 210235) AND (p_currency = get_env_base_currency()))
        Heap Fetches: 0
Planning Time: 1.193 ms
Execution Time: 11.001 ms
(33 rows)
```

```
select * from pg_stat_xact_user_functions;
  funcid   | schemaname |       funcname         | calls | total_time | self_time
-----------+------------+------------------------+-------+------------+-----------
 181415537 | demo2      | get_env_base_currency |     6 |   0.683619 |  0.683619
```

So Long, and Thanks for All the Fish

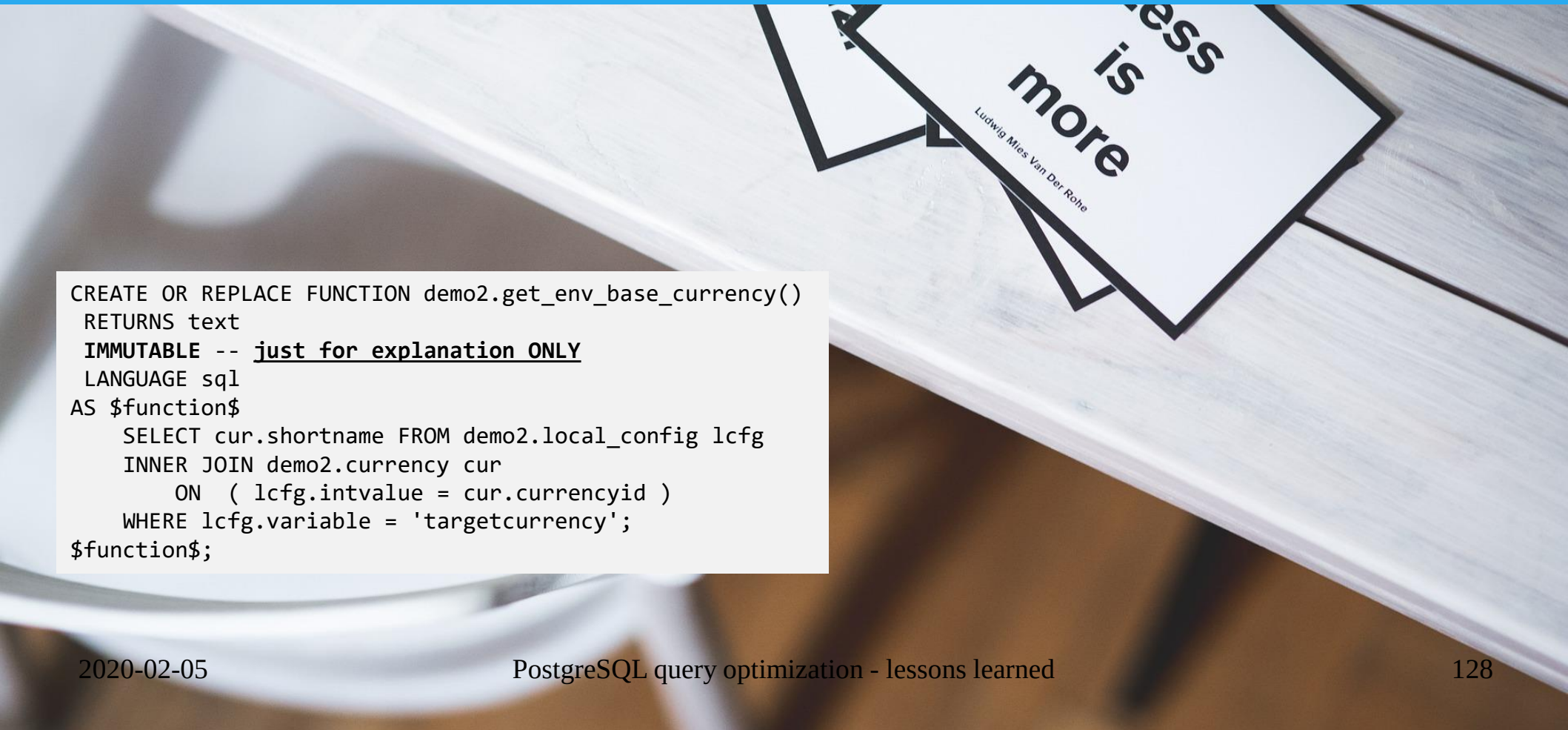
finmason

Questions ?

Thanks for your time

Fake test, evaluate only!

finmason




```
CREATE OR REPLACE FUNCTION demo2.get_env_base_currency()
 RETURNS text
 IMMUTABLE -- just for explanation ONLY
 LANGUAGE sql
AS $function$
    SELECT cur.shortname FROM demo2.local_config lcfg
    INNER JOIN demo2.currency cur
        ON  ( lcfg.intvalue = cur.currencyid )
    WHERE lcfg.variable = 'targetcurrency';
$function$;
```

The test SQL with all the function calls

finmason

```
explain (analyze, buffers, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, demo2.get_env_base_currency() p_currency,
  CASE WHEN apd.p_currency = demo2.get_env_base_currency() THEN apd.price
  ELSE apd.price / coalesce( cer.fxrate,
                             demo2.get_fx_rate(apd.trade_date, apd.p_currency, demo2.get_env_base_currency())
  ) END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
/* INNER JOIN demo2.base_currency bc ON true Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND demo2.get_env_base_currency() = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = demo2.get_env_base_currency())
      AND apd.p_currency != demo2.get_env_base_currency() AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, demo2.get_env_base_currency() p_currency,
       apd.price AS converted_value
FROM demo2.asset_price_daily apd
WHERE apd.p_currency = demo2.get_env_base_currency() AND apd.assetid = 210235;
```

Unfortunately, the function is *only* **stable**

finmason

```
 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Append (actual time=3.282..9.774 rows=24897 loops=1)
   Buffers: shared hit=358
   ->  Nested Loop Left Join (actual time=3.260..3.260 rows=0 loops=1)
         Buffers: shared hit=181
         InitPlan 2 (returns $1)
           ->  Result (actual time=0.045..0.045 rows=1 loops=1)
                 Buffers: shared hit=4
                 InitPlan 1 (returns $0)
                   ->  Limit (actual time=0.043..0.043 rows=1 loops=1)
                         Buffers: shared hit=4
                         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=0.042..0.042 rows=1 loops=1)
                               Index Cond: ((trade_date IS NOT NULL) AND (currency_to = 'USD'::text))
                               Heap Fetches: 0
                               Buffers: shared hit=4
         ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=3.259..3.259 rows=0 loops=1)
               Index Cond: (assetid = 210235)
               Filter: ((trade_date >= $1) AND (p_currency <> 'USD'::text))
               Rows Removed by Filter: 24897
               Heap Fetches: 0
               Buffers: shared hit=181
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (never executed)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = 'USD'::text) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
   ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=0.021..4.590 rows=24897 loops=1)
         Index Cond: ((assetid = 210235) AND (p_currency = 'USD'::text))
         Heap Fetches: 0
         Buffers: shared hit=177
 Planning Time: 1.379 ms
 Execution Time: 10.906 ms
(29 rows)
```

Immutable is optimized same as hardcoded target currency constants

At the end of the day, response time is same as longer SQL with STABLE function.

How fast we are converting prices now?

We should be able to perform currency conversion at reasonable speed.

```
update demo2.local_config set intvalue = 5 where variable = 'targetcurrency';
```

EUR

Query ver. 8 - put all together (USD → USD)

```
explain (analyze, costs off)
SELECT apd.assetid, apd.trade_date, apd.price, /*apd.p_currency,*/ bc.base_currency p_currency,
  CASE WHEN apd.p_currency = bc.base_currency THEN apd.price
  ELSE apd.price / coalesce( cer.fxrate,
                             demo2.get_fx_rate(apd.trade_date, apd.p_currency, bc.base_currency)
  ) END AS converted_value
FROM demo2.asset_price_daily apd /* Get the Daily Currency Exchange Rates */
 INNER JOIN demo2.base_currency bc ON true /* Get environment base currency */
LEFT JOIN demo2.currency_exchange_rate cer /* Get Exchange Rate if exists for trade_date and currency pair*/
    ON apd.p_currency = cer.currency_from
        AND bc.base_currency = cer.currency_to
        AND apd.trade_date = cer.trade_date
WHERE apd.trade_date >= ( SELECT min(cerdt.trade_date) AS min FROM demo2.currency_exchange_rate cerdt WHERE
cerdt.currency_to = demo2.get_env_base_currency())
      AND apd.p_currency != bc.base_currency AND apd.assetid = 210235
UNION ALL
SELECT apd.assetid, apd.trade_date, apd.price, apd.p_currency, /*get_env_base_currency() p_currency,*/
apd.price AS converted_value
FROM asset_price_daily apd
WHERE apd.p_currency = get_env_base_currency() AND apd.assetid = 210235;
```


2020-02-05 PostgreSQL query optimization - lessons learned 125

Test with fake immutable fn – (USD → EUR)

```
Append (actual time=0.153..36.272 rows=6773 loops=1)
   ->  Nested Loop Left Join (actual time=0.153..35.484 rows=6773 loops=1)
         InitPlan 2 (returns $1)
           ->  Result (actual time=0.114..0.114 rows=1 loops=1)
                 InitPlan 1 (returns $0)
                   ->  Limit (actual time=0.112..0.113 rows=1 loops=1)
                         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cerdt (actual time=0.111..0.112 rows=1 loops=1)
                               Index Cond: ((trade_date IS NOT NULL) AND (currency_to = get_env_base_currency()))
                               Heap Fetches: 0
         ->  Nested Loop (actual time=0.143..4.649 rows=6773 loops=1)
               Join Filter: (apd.p_currency <> cur.shortname)
               ->  Hash Join (actual time=0.010..0.053 rows=1 loops=1)
                     Hash Cond: (cur.currencyid = lcfg.intvalue)
                     ->  Seq Scan on currency cur (actual time=0.002..0.018 rows=197 loops=1)
                     ->  Hash (actual time=0.004..0.004 rows=1 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Seq Scan on local_config lcfg (actual time=0.002..0.003 rows=1 loops=1)
                                 Filter: (variable = 'targetcurrency'::text)
                                 Rows Removed by Filter: 11
               ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd (actual time=0.131..3.669 rows=6773 loops=1)
                     Index Cond: (assetid = 210235)
                     Filter: (trade_date >= $1)
                     Rows Removed by Filter: 18124
                     Heap Fetches: 0
         ->  Index Only Scan using currency_exchange_rate_key on currency_exchange_rate cer (actual time=0.004..0.004 rows=1 loops=6773)
               Index Cond: ((trade_date = apd.trade_date) AND (currency_to = cur.shortname) AND (currency_from = apd.p_currency))
               Heap Fetches: 0
   ->  Index Only Scan using asset_price_daily_assetid_currency_idx_ext on asset_price_daily apd_1 (actual time=0.265..0.265 rows=0 loops=1)
         Index Cond: ((assetid = 210235) AND (p_currency = get_env_base_currency()))
         Heap Fetches: 0
 Planning Time: 1.234 ms
 Execution Time: 36.589 ms
```

0.004*6773=27.1 [ms]

```
select * from pg_stat_xact_user_functions;
  funcid   | schemaname |       funcname        | calls | total_time | self_time
-----------+------------+-----------------------+-------+------------+-----------
 181415538 | demo2      | get_fx_rate           |    19 |   0.648934 |  0.648934
 181415537 | demo2      | get_env_base_currency |     6 |   0.826895 |  0.826895
```