CYBERTEC
DATA SCIENCE & POSTGRESQL

Fast faceting using roaring bitmaps

Ants Aasma

www.cybertec-postgresql.com

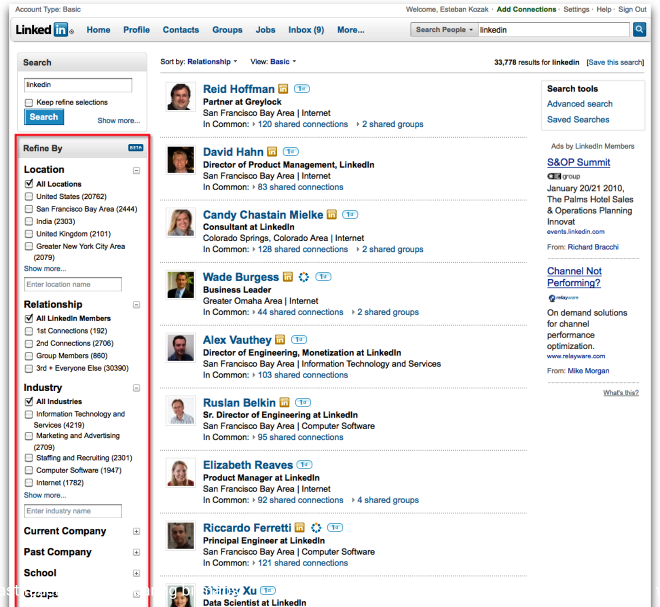
Hello

- Senior database engineer at Cybertec
- I focus on: Performance, high availability, core hacking
- We also do: support, HA setups, DBA-as-a-service, private cloud, general consulting

Problem statement

- Customer has hundreds of millions of documents.
- Wants to provide faceted search over these documents.
- Need accurate counts, or at worst slightly stale counts.
- Response time below 2 seconds.

What is faceted search



Account Type: Basic

Welcome, Esteban Kozak · **Add Connections** · Settings · Help · Sign Out

Search

linkedin

□ Keep refine selections

Refine By BETA

Location

☑ **All Locations**
□ United States (20762)
□ San Francisco Bay Area (2444)
□ India (2303)
□ United Kingdom (2101)
□ Greater New York City Area (2079)

Show more...

Enter location name

Relationship

☑ **All LinkedIn Members**
□ 1st Connections (192)
□ 2nd Connections (2706)
□ Group Members (860)
□ 3rd + Everyone Else (30390)

Industry

☑ **All Industries**
□ Information Technology and Services (4219)
□ Marketing and Advertising (2709)
□ Staffing and Recruiting (2301)
□ Computer Software (1947)
□ Internet (1782)

Show more...

Enter industry name

Current Company

Past Company

School

Groups

Sort by: Relationship ▾   View: Basic ▾

33,778 results for **linkedin**   [Save this search]

Reid Hoffman 1st
Partner at Greylock
San Francisco Bay Area | Internet
In Common: 120 shared connections   2 shared groups

David Hahn 1st
Director of Product Management, LinkedIn
San Francisco Bay Area | Internet
In Common: 83 shared connections

Candy Chastain Mielke 1st
Consultant at LinkedIn
Colorado Springs, Colorado Area | Internet
In Common: 128 shared connections   2 shared groups

Wade Burgess 1st
Business Leader
Greater Omaha Area | Internet
In Common: 44 shared connections   2 shared groups

Alex Vauthey 1st
Director of Engineering, Monetization at LinkedIn
San Francisco Bay Area | Information Technology and Services
In Common: 103 shared connections

Ruslan Belkin 1st
Sr. Director of Engineering at LinkedIn
San Francisco Bay Area | Computer Software
In Common: 95 shared connections

Elizabeth Reaves 1st
Product Manager at LinkedIn
San Francisco Bay Area | Internet
In Common: 92 shared connections   4 shared groups

Riccardo Ferretti 1st
Principal Engineer at LinkedIn
San Francisco Bay Area | Computer Software
In Common: 121 shared connections

Shirley Xu 1st
Data Scientist at LinkedIn

Search tools
Advanced search
Saved Searches

Example schema for testing

```
                         Table "test2.documents"
   Column    |           Type           | Collation | Nullable | Default
-------------+--------------------------+-----------+----------+---------
 id          | bigint                   |           | not null |
 created     | timestamp with time zone |           | not null |
 finished    | timestamp with time zone |           |          |
 category_id | bigint                   |           | not null |
 tags        | text[]                   |           |          |
 type        | public.mimetype          |           |          |
 size        | bigint                   |           |          |
 title       | text                     |           |          |
Indexes:
    "documents_category_id_idx" btree (category_id)

postgres=# SELECT category_id, ROUDN(COUNT(*)/1e6, 3) millions FROM documents GROUP BY ROLLUP (category_id) ORDER BY 2 DESC LIMIT 6;
 category_id | millions
-------------+----------
             | 100.000
          24 |  60.812
           9 |  15.214
          12 |   6.764
           8 |   3.811
          49 |   2.444
```

Facets and example query

- Facets:
  - Created month
  - Finished month
  - Category
  - Type
  - Size class
- Query: find all other facet counts when filtered by most popular category

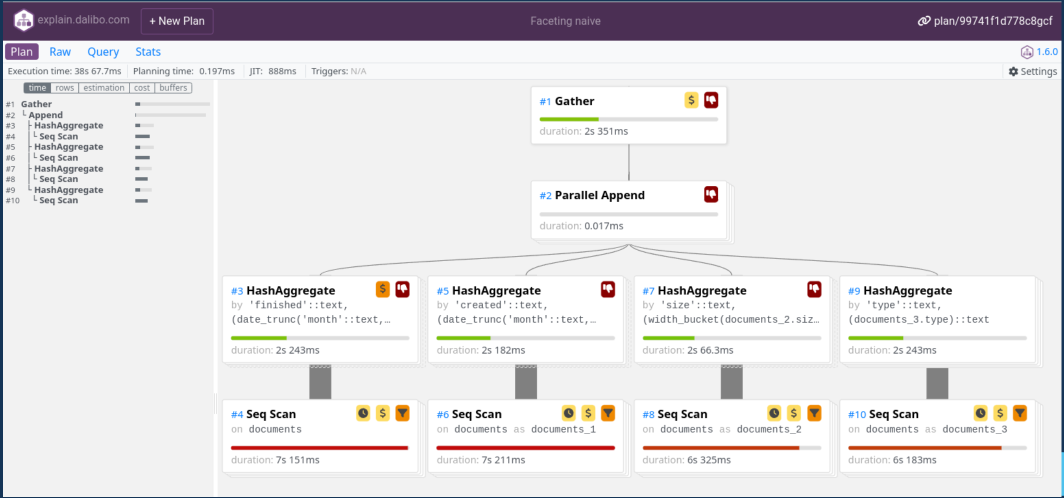
How to implement faceting, simple version

- Simple option, do a `count(*) group by` for every facet.

```
(SELECT 'created' AS facet_name, date_trunc('month', created)::text AS facet_value,
        COUNT(*) AS cardinality
 FROM documents WHERE category_id = 24 GROUP BY 1, 2)
    UNION ALL
(SELECT 'finished', date_trunc('month', finished)::text, COUNT(*)
 FROM documents  WHERE category_id = 24 GROUP BY 1, 2)
    UNION ALL
(SELECT 'type', type::text, COUNT(*)
 FROM documents WHERE category_id = 24 GROUP BY 1, 2)
    UNION ALL
(SELECT 'size', width_bucket(size, array[0,1000,5000,10000,50000,100000,500000])::text,
        COUNT(*)
 FROM documents WHERE category_id = 24 GROUP BY 1, 2);
```

Results

38s: https://explain.dalibo.com/plan/99741f1d778c8gcf



explain.dalibo.com + New Plan Faceting naive plan/99741f1d778c8gcf

Plan Raw Query Stats 1.6.0

Execution time: 38s 67.7ms | Planning time: 0.197ms | JIT: 888ms | Triggers: N/A | Settings

time | rows | estimation | cost | buffers

```
#1  Gather
#2  └ Append
#3    ├ HashAggregate
#4    │ └ Seq Scan
#5    ├ HashAggregate
#6    │ └ Seq Scan
#7    ├ HashAggregate
#8    │ └ Seq Scan
#9    └ HashAggregate
#10     └ Seq Scan
```

#1 Gather
duration: 2s 351ms

#2 Parallel Append
duration: 0.017ms

#3 HashAggregate
by 'finished'::text, (date_trunc('month'::text,…
duration: 2s 243ms

#5 HashAggregate
by 'created'::text, (date_trunc('month'::text,…
duration: 2s 182ms

#7 HashAggregate
by 'size'::text, (width_bucket(documents_2.siz…
duration: 2s 66.3ms

#9 HashAggregate
by 'type'::text, (documents_3.type)::text
duration: 2s 243ms

#4 Seq Scan
on documents
duration: 7s 151ms

#6 Seq Scan
on documents as documents_1
duration: 7s 211ms

#8 Seq Scan
on documents as documents_2
duration: 6s 325ms

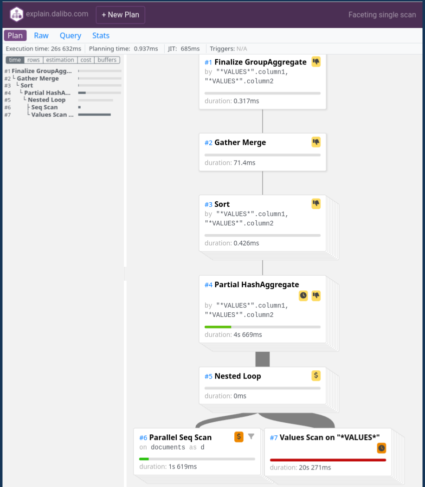
#10 Seq Scan
on documents as documents_3
duration: 6s 183ms

Getting rid of multiple scans

```
SELECT facet_name, facet_value, COUNT(*) cardinality
FROM documents d, LATERAL (VALUES
    ('created', date_trunc('month', created)::text),
    ('finished', date_trunc('month', finished)::text),
    ('type', type::text),
    ('size', width_bucket(size, array[0,1000,5000,10000,50000,100000,500000
) t(facet_name, facet_value)
WHERE category_id = 24 GROUP BY 1, 2
```

Results

27s: https://explain.dalibo.com/plan/befh3292ad7cdb12



Execution time: 26s 632ms | Planning time: 0.937ms | JIT: 685ms | Triggers: N/A

```
#1 Finalize GroupAgg...
#2 └ Gather Merge
#3   └ Sort
#4     └ Partial HashA...
#5       └ Nested Loop
#6         ├ Seq Scan
#7         └ Values Scan ...
```

```
#1 Finalize GroupAggregate
by "*VALUES*".column1,
"*VALUES*".column2
duration: 0.317ms

#2 Gather Merge
duration: 71.4ms

#3 Sort
by "*VALUES*".column1,
"*VALUES*".column2
duration: 0.426ms

#4 Partial HashAggregate
by "*VALUES*".column1,
"*VALUES*".column2
duration: 4s 669ms

#5 Nested Loop
duration: 0ms

#6 Parallel Seq Scan
on documents as d
duration: 1s 619ms

#7 Values Scan on "*VALUES*"
duration: 20s 271ms
```

More parallelism?

```
ALTER TABLE documents SET ( parallel_workers = 23 );
```

More parallelism?

```
ALTER TABLE documents SET ( parallel_workers = 23 );
```

- Time: 18261.594 ms (00:18.262)

Is this even possible?

- Need to process 60M rows in 2s.
- For each matching row and each facet:
  - Calculate the facet value from row
  - Look up the counter for this facet value and add 1 to it.
- We have 2s/60e6/4 = 8ns of time to do this
- RAM access takes ~ 80ns, L3 cache access is 15ns

What if we invert the problem

- Random access is really bad for CPUs. Can we turn the problem into a sequential one?
- What if for each facet and value we store a bitmap of matching documents.
- Counting matches would then be AND'ing together of two bitmaps and counting the number of bits set.
- CPU's can do this at 128bits*4GHz = 512 Gbit/s/core
- Could calculate number of matches for 10'000 facet values per second per core.

Not a new idea

- Concept is also known as bitmap indexes.
- Ideally we would want to use different approaches for storing 60M values and 60 values.
- A popular and fast implementation is Roaring bitmaps. (roaringbitmap.org)
  - Used by others too including Solr/Elasticsearch, Pinot, Hive, ClickHouse.
- Good news, there is a postgres extension wrapping it:
  - github.com/ChenHuajun/pg_roaringbitmap

What are roaring bitmaps

- Compressed storage of a set of integers (e.g. document id)
- Provides fast SIMD optimized operations for intersection (AND), union (OR) and cardinality (COUNT).
- Divides values into 16bit ranges (65’536 values per range) called containers.
- First layer is a sorted list of containers that have values, with starting value, type and pointer for each one.
- Three types of containers:
  - Array of 16bit values (up to 4’096 values)
  - 8KB bitmap
  - Run length encoded bitmap

pg_roaringbitmap extension

- Introduces a new roaringbitmap datatype for storing bitmaps in the database
- Some useful functions for us:
  - rb_build_agg(int) -> roaringbitmap
  - rb_and(roaringbitmap, roaringbitmap) -> roaringbitmap
  - rb_or(roaringbitmap, roaringbitmap) -> roaringbitmap
  - rb_and_agg(roaringbitmap) -> roaringbitmap
  - rb_cardinality(roaringbitmap) -> int

The concept

- Lets build and maintain a second datastructure of roaring bitmaps for each facet value.
- Conceptually an index, but we are doing it in “userspace”
- We plan to be modifying data, but mostly only recent rows.
  - Store bitmaps as chunks so we can update only modified chunks.
  - Capture modifications into a “delta” table and batch apply changes.
- Add some functions to make querying easier

Building the index

```
CREATE TABLE documents_facets AS
SELECT facet_name, facet_value, id >> 20 chunk_id, rb_build_agg((id & ((1<<20)-1))::int4)
FROM documents d, LATERAL (VALUES
    ('created', date_trunc('month', created)::text),
    ('finished', date_trunc('month', finished)::text),
    ('category_id', category_id::text),
    ('type', type::text),
    ('size', width_bucket(size, array[0,1000,5000,10000,50000,100000,500000])::text)
) t(facet_name, facet_value) GROUP BY 1, 2, 3;

Time: 37334.313 ms (00:37.334)
```

Building the index

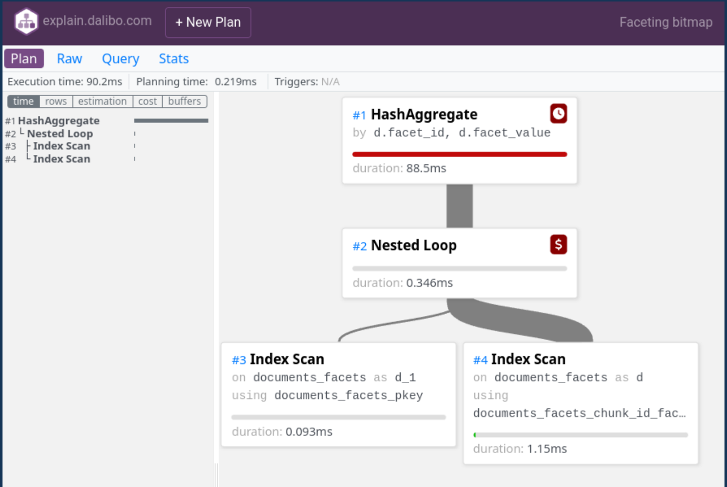
```
CREATE TABLE documents_facets AS
SELECT facet_name, facet_value, id >> 20 chunk_id, rb_build_agg((id & ((1<<20)-1))::int4)
FROM documents d, LATERAL (VALUES
    ('created', date_trunc('month', created)::text),
    ('finished', date_trunc('month', finished)::text),
    ('category_id', category_id::text),
    ('type', type::text),
    ('size', width_bucket(size, array[0,1000,5000,10000,50000,100000,500000])::text)
) t(facet_name, facet_value) GROUP BY 1, 2, 3;

Time: 37334.313 ms (00:37.334)

      table       | table_size | indexes_size
------------------+------------+--------------
 documents        | 20 GB      | 661 MB
 documents_facets | 216 MB     | 848 kB
```

Querying

```
WITH lookup AS (
    SELECT chunk_id, postinglist FROM documents_facets d
    WHERE facet_name = 'category_id' AND facet_value = '24'
)
SELECT facet_id, facet_value,
       sum(rb_and_cardinality(lookup.postinglist, d.postinglist)) count
FROM lookup JOIN documents_facets d USING (chunk_id)
WHERE facet_id IN (1,2,4,5)
GROUP BY 1,2;
```

Results

90ms: https://explain.dalibo.com/plan/71914e60dc7ab324



Execution time: 90.2ms | Planning time: 0.219ms | Triggers: N/A

```
#1 HashAggregate
#2 └ Nested Loop
#3   ├ Index Scan
#4   └ Index Scan
```

#1 HashAggregate
by d.facet_id, d.facet_value
duration: 88.5ms

#2 Nested Loop
duration: 0.346ms

#3 Index Scan
on documents_facets as d_1
using documents_facets_pkey
duration: 0.093ms

#4 Index Scan
on documents_facets as d
using
documents_facets_chunk_id_fac…
duration: 1.15ms

Updating the index

- Don’t want to rewrite multiple big bitmaps for each inserted row.
- Create a delta table with structure `(facet_id, facet_value, id, delta)`
- A trigger on main table that inserts into delta table +1 or -1 for each changed facet.
- Periodically aggregate deltas and merge into main index.

The extension

- Started an extension called pgfaceting
  - github.com/cybertec-postgresql/pgfaceting
- SQL only, but requires pg_roaringbitmap
- Open source, help appreciated
- API is still WIP

What is there

- Automatic creation and population of facet and delta tables.
- Facet types:
  - plain
  - date_trunc
  - bucket_facet
- Delta trigger generation.
- Query generation facility

Example usage

```
SELECT faceting.add_faceting_to_table(
    'documents',
    key => 'id',
    facets => array[
        faceting.datetrunc_facet('created', 'month'),
        faceting.datetrunc_facet('finished', 'month'),
        faceting.plain_facet('category_id'),
        faceting.plain_facet('type'),
        faceting.bucket_facet('size',
             buckets => array[0,1000,5000,10000,50000,100000,500000])
    ]
);
-- Add a cron job that does:
CALL faceting.run_maintenance();
```

Example usage

```
SELECT * FROM faceting.count_results('documents'::regclass,
    filters => array[row('category_id', '24'),
                     row('type', 'image/jpeg')
                    ]::faceting.facet_filter[]);
```

Future ideas

- Support id’s larger than int4
- New facet types
  - array (e.g. unnest(tags)
  - Tree structured facets
  - Hierarchical date and time
- Automatic range queries
- Use delta tables in queries
- Full text search

When to not use it

- Doesn’t work well with sparse id spaces like uuid, snowflake.

  Also rules out using ctid
- Tons of unique values (tens to hundreds of thousands)
- Heavily updated facets

Thanks

- Q & A