Building a Postgres Data Warehouse

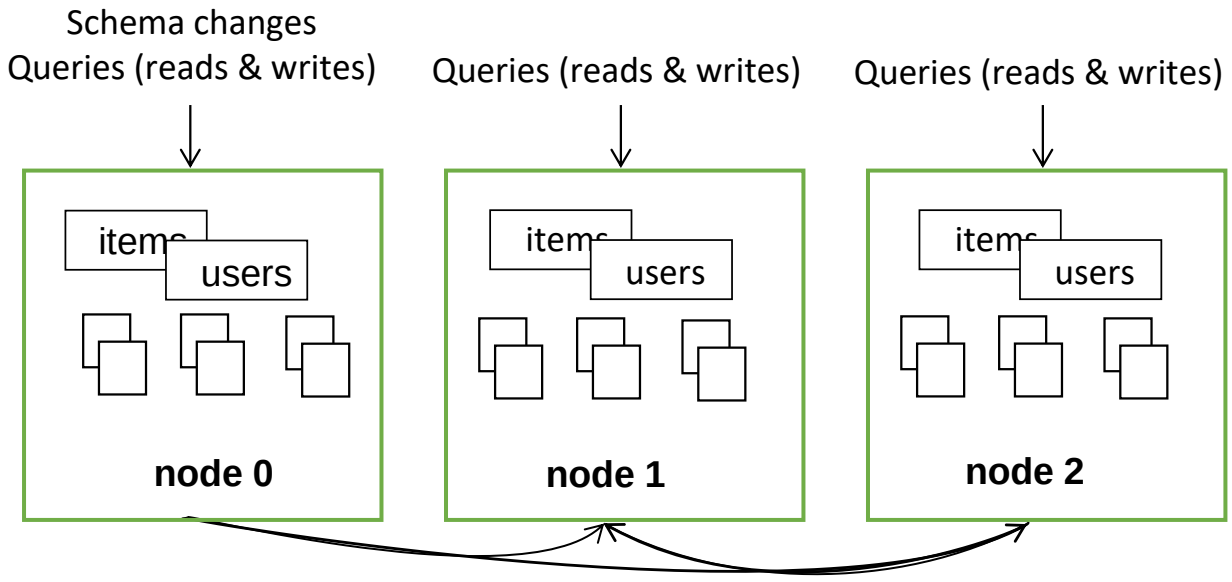
POSETTE ‘25

Marco Slot

crunchy data

Past project: Citus

Citus is a PostgreSQL *extension* that can distribute tables across a cluster of PostgreSQL servers.



Multi-tenant SaaS apps
Real-time analytics

OLTP

Operational system of record

SQL
Transactions

High query rate, small queries
Low response time
User-facing applications
Mission-critical, always on

OLAP

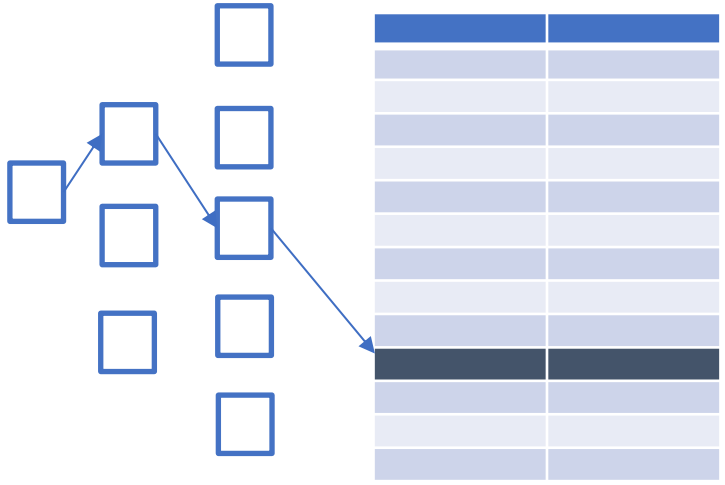
Analytics on collection of data sources

SQL
Transactions

Low query rate, big queries
High scan throughput
Business-facing dashboards
On demand, business hours

Row-oriented vs. columnar-oriented

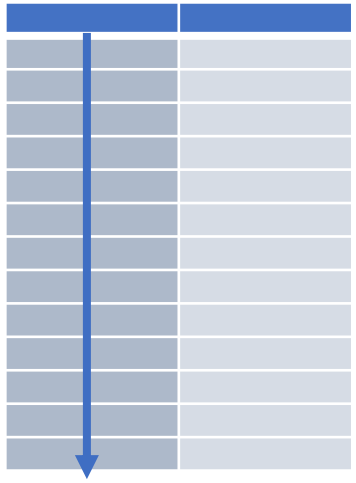
Row-oriented storage & execution



```
SELECT * FROM orders
WHERE orderid = $1
```

At scale: Fast on OLTP, Slow on OLAP

Column-oriented storage & execution



```
SELECT productid, count(*) FROM orders
GROUP BY 1 ORDER BY 2 DESC LIMIT 10
```

At scale: Slow on OLTP, Fast on OLAP

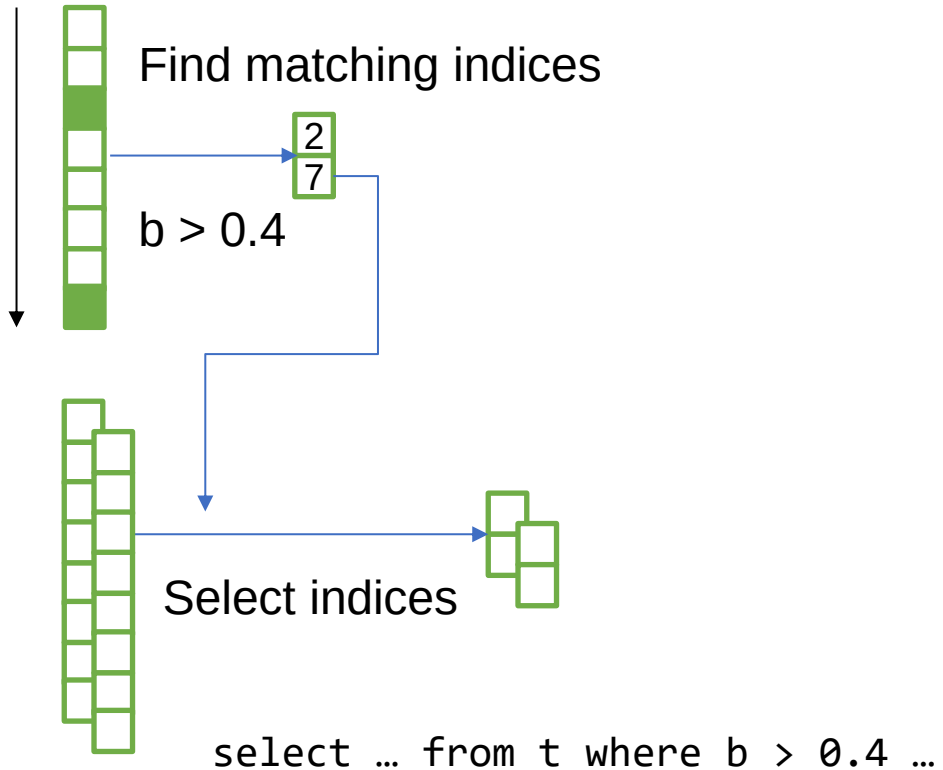
Column-oriented storage & vectorized execution

Vectorized query engine takes vectors of column values and processes them in loops.

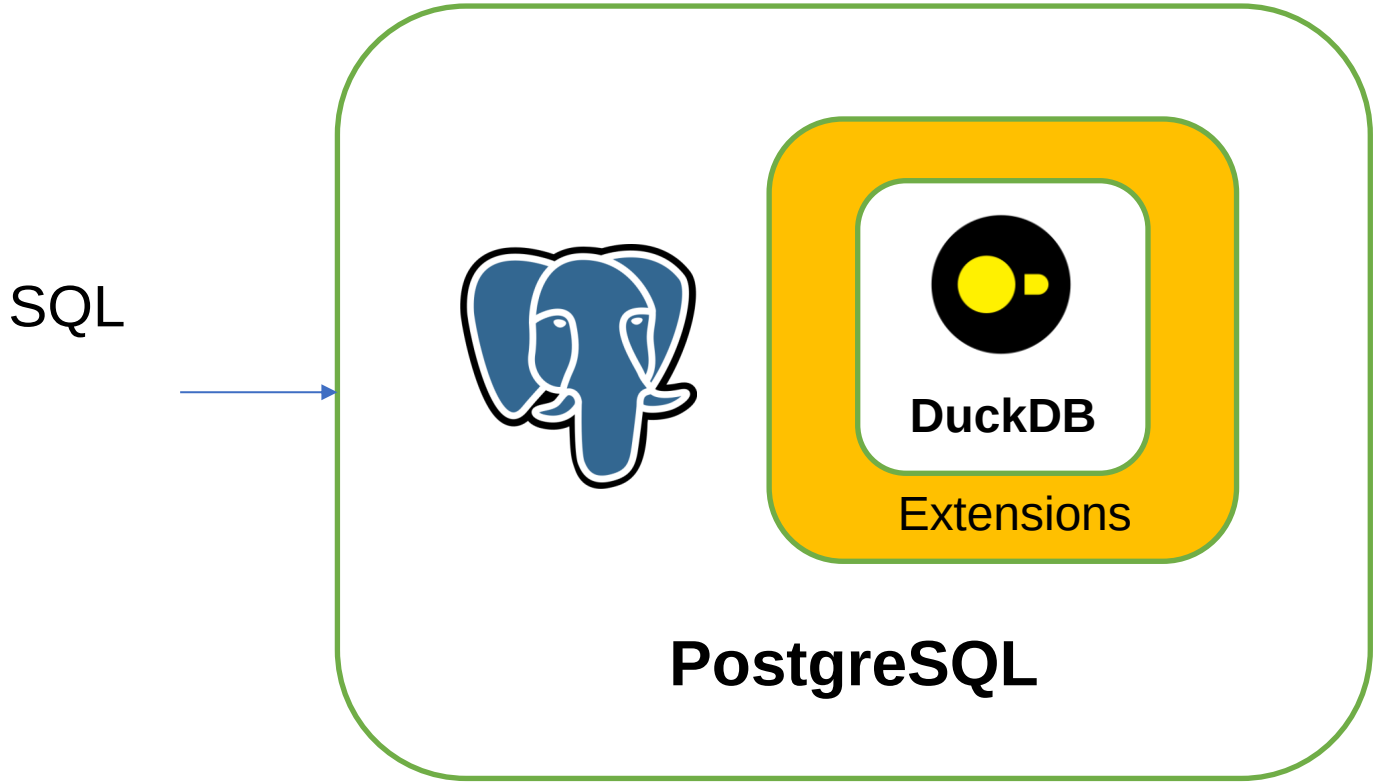
Several benefits:

- **Low function call overhead**
- **Better branch prediction**
- **Good CPU cache utilization**
- **SIMD**

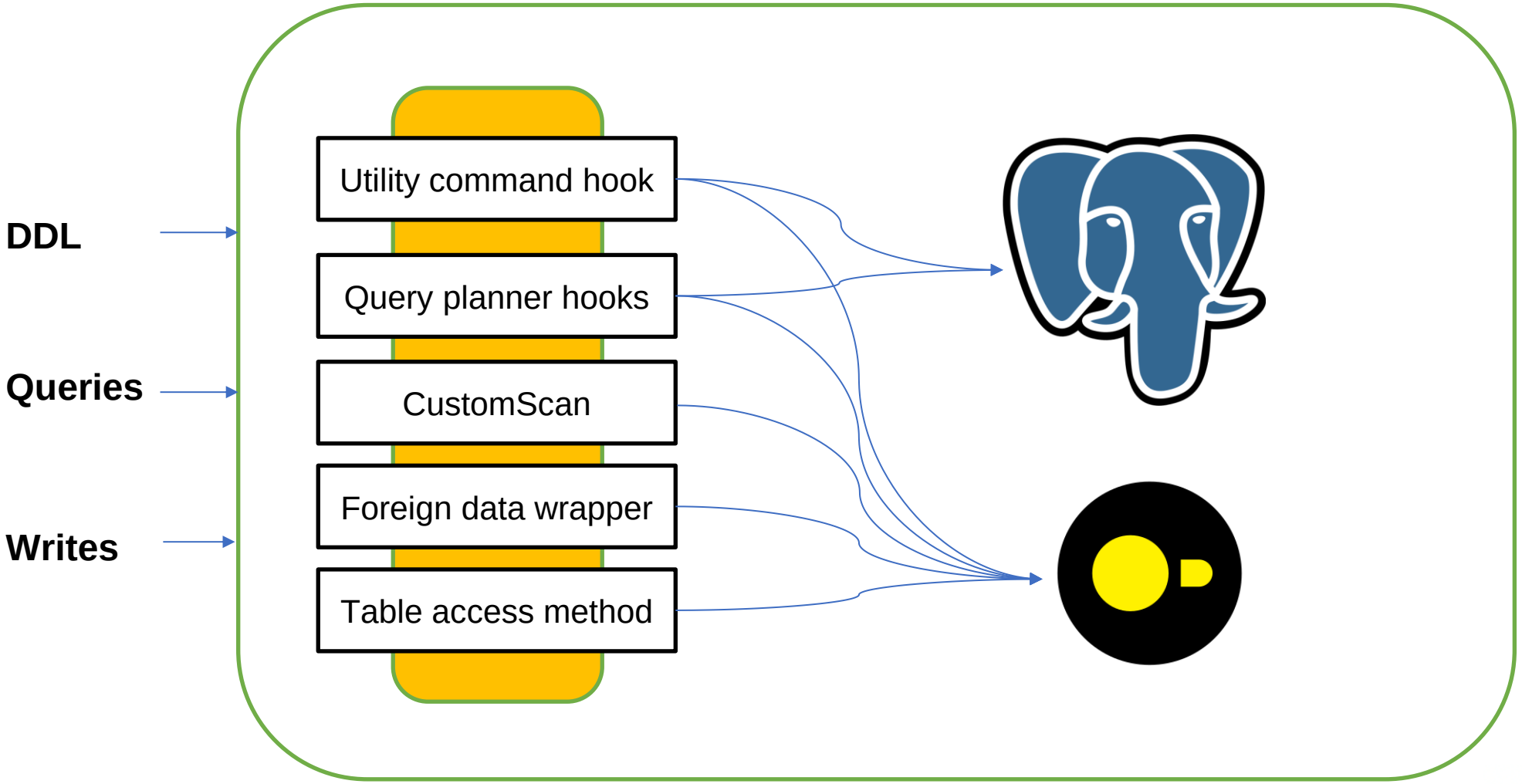
Many vectors processed in parallel.



Hybrid OLTP/OLAP architecture

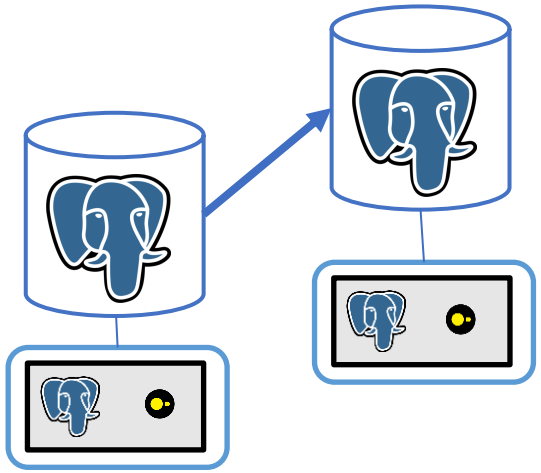


Extending PostgreSQL

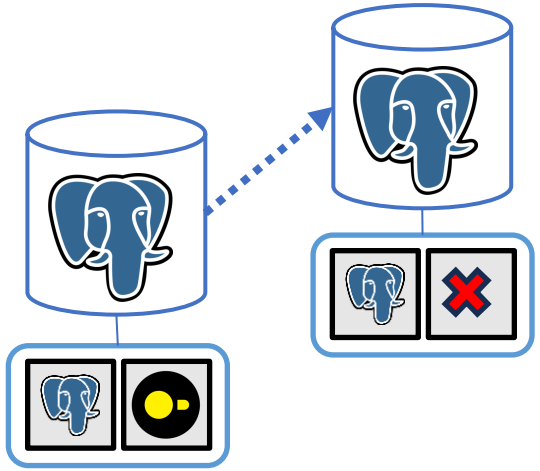


Hybrid table storage

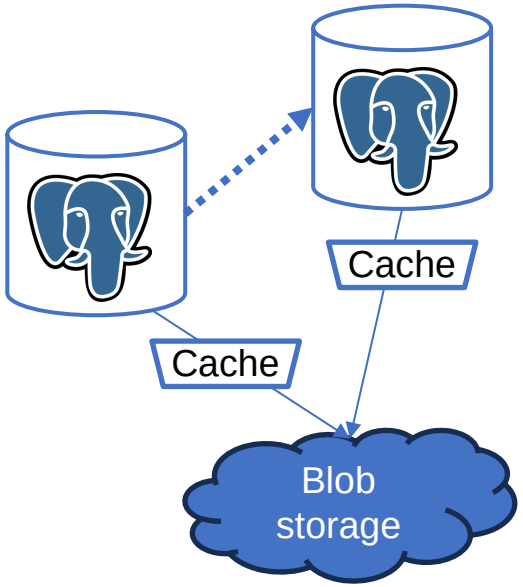
Different methods of storing analytics tables



PostgreSQL Buffer Manager & WAL?



DuckDB tables on disk?



Files in object storage?

Postgres + Apache Iceberg

PostgreSQL as an Iceberg query engine & catalog.

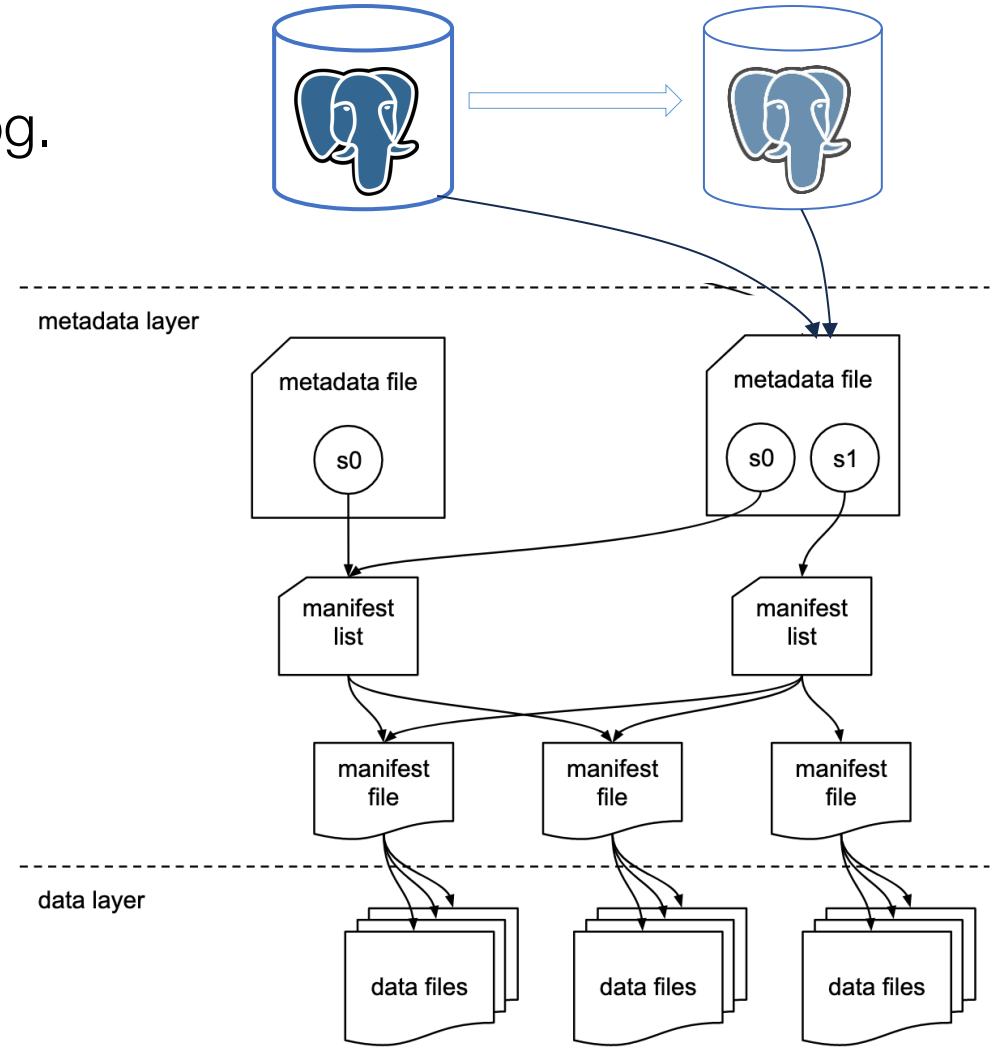
Extensions add an Iceberg table type.

Writes generate Parquet & Iceberg metadata.

Reads query Parquet using DuckDB.

Iceberg files will still be available to replicas.

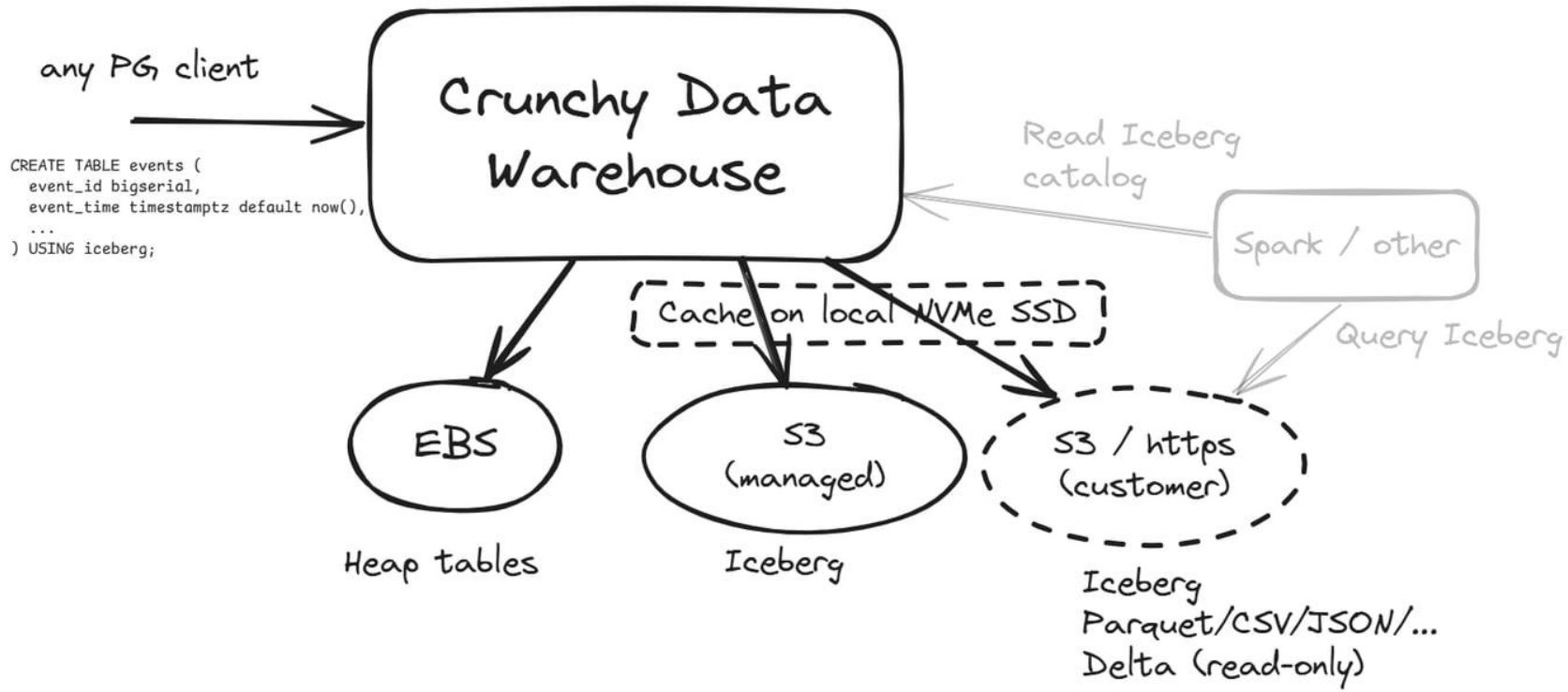
Iceberg catalog is in WAL, replicated.



Crunchy Data Warehouse

PostgreSQL with Iceberg and external data lake tables.

10-100x faster for analytics by integrating DuckDB and write-through file caching.



crunchy data

Constellation of Postgres Extensions

Add new Postgres user experiences through many small, composable extensions.

List of installed extensions

Name	Version	Schema	Description
btree_gist	1.7	public	support for indexing common datatypes in GiST
crunchy_base	1.4	pg_catalog	Crunchy Data base extension
crunchy_copy	2.2	pg_catalog	Data lake copy extension
crunchy_data_warehouse	2.2	pg_catalog	Crunchy Data Warehouse
crunchy_extension_updater	1.0	pg_catalog	Crunchy Data Extension Updater
crunchy_iceberg	2.2	pg_catalog	Iceberg in PostreSQL
crunchy_lake_analytics	2.3	pg_catalog	Crunchy lake analytics and Iceberg tables
crunchy_map	1.1	pg_catalog	Associate array, dict / map type.
crunchy_query_engine	2.2-1	pg_catalog	Crunchy query engine common library
crunchy_spatial_analytics	2.2	pg_catalog	Geospatial analytics on data lakes
pg_cron	1.6	pg_catalog	Job scheduler for PostgreSQL
pg_incremental	1.3	pg_catalog	Incremental Processing by Crunchy Data
pg_parquet	0.3.1	public	copy data between Postgres and Parquet

Iceberg as a Postgres table format

Capture queries, writes, & schema changes to provide a transactional table experience for Iceberg in S3.

```
create table chats (
  message_id bigserial not null,
  thread_id bigint not null,
  …
) using iceberg;
```

→ Write metadata (avro, json) files to blob storage, insert to catalog

```
…
update chats set answer = '42'
where question is null;
```

```
SELECT * FROM read_parquet(…, filename=…, file_row_number=…)
WHERE question IS NULL;
```

→ Write updated rows into new Parquet file.
Write deleted rows into position delete Parquet file.
Write metadata (avro, json) files to blob storage, update catalog

```
select count(*) from chats;
```

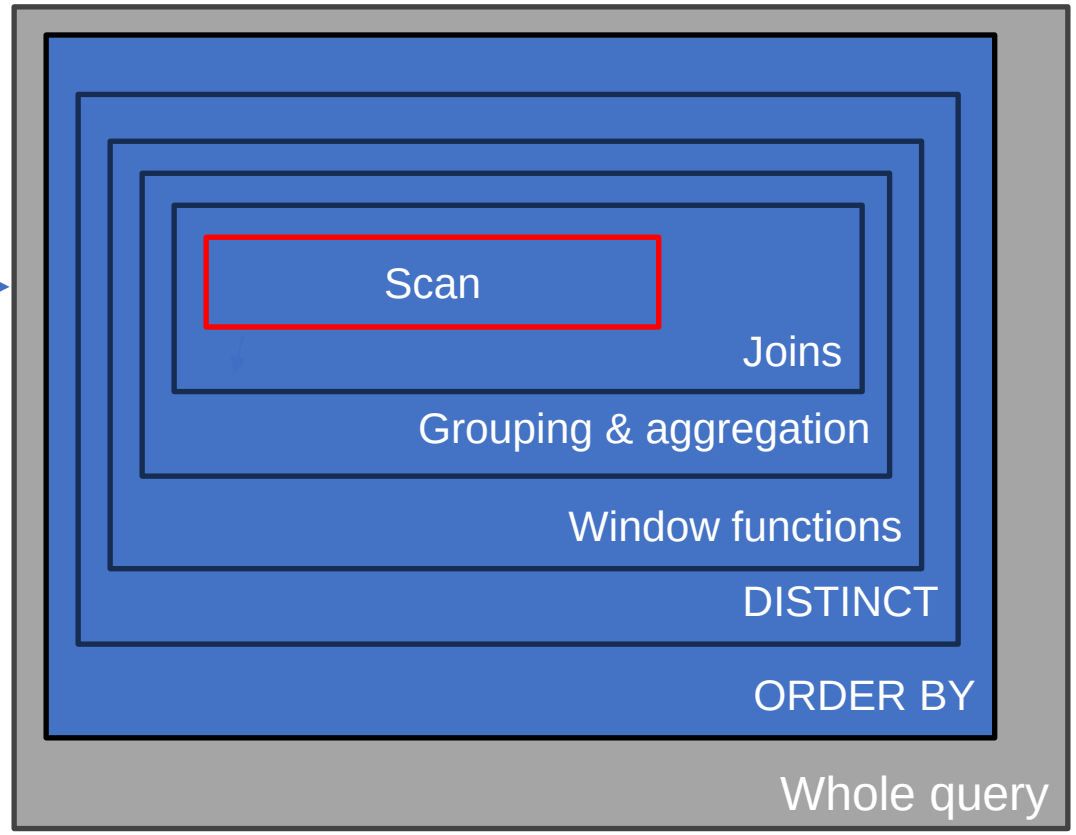
```
SELECT count(*)
FROM read_parquet(…, schema=…, filename=…, file_row_number=…))
WHERE (filename, file_row_number)
NOT IN (SELECT (file_path, pos) FROM read_parquet(…));
```

crunchy data

Extending the Postgres query planner

Extensions can propose or enforce alternative plans for whole query or fragments.

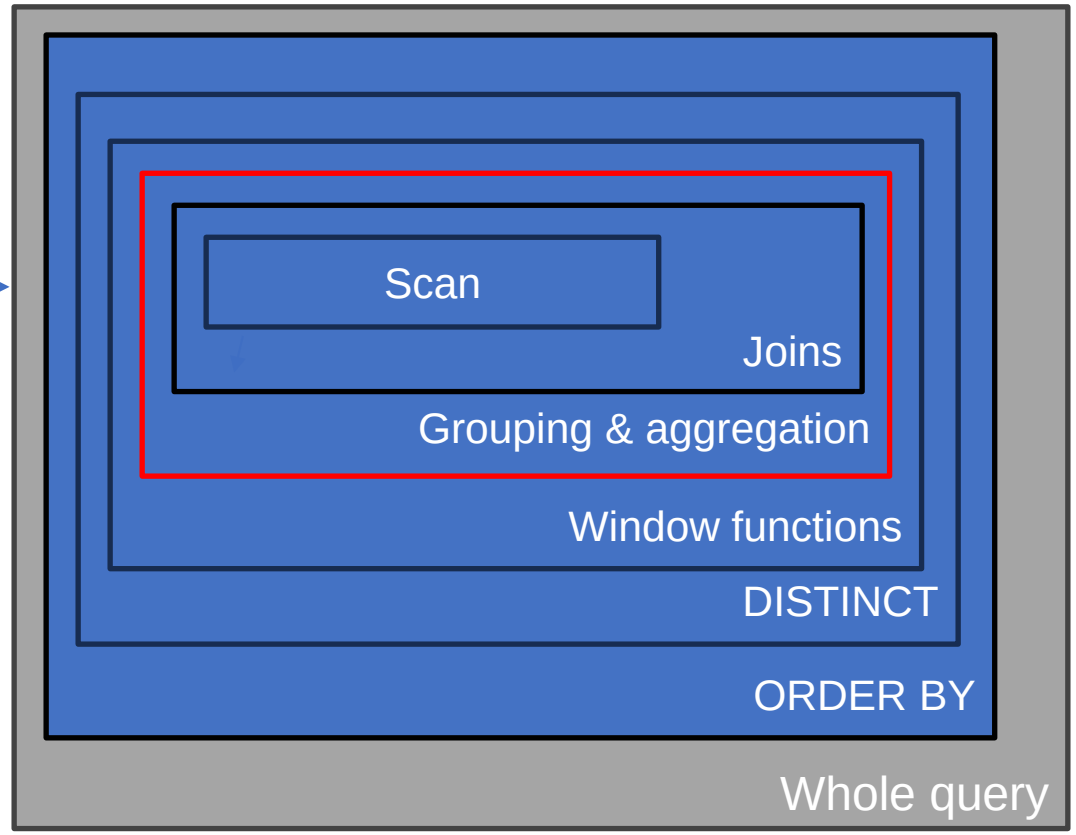
```
with top10 as (
  select cust_id, sum(…)
  from sales
  where …
  group by 1
  order by 2 desc limit 10
)
select pg_func(cust_id)
from top10;
```



Extending the Postgres query planner

Extensions can propose or enforce alternative plans for whole query or fragments.

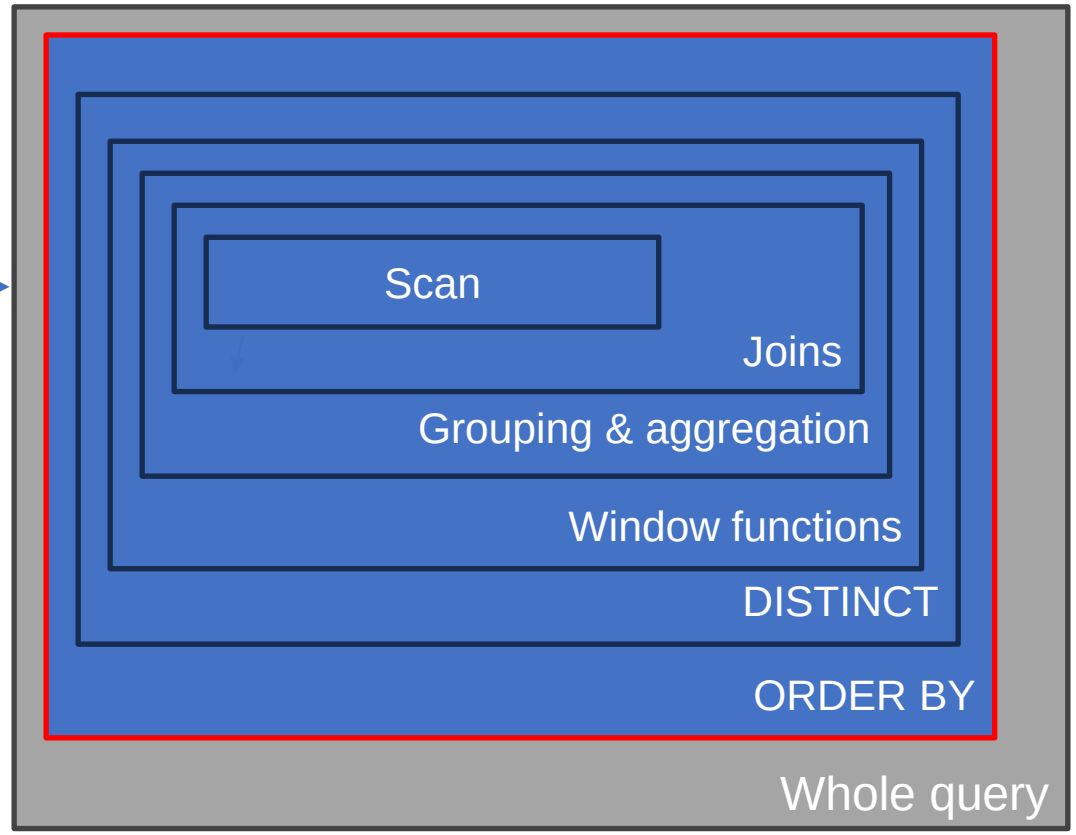
```
with top10 as (
  select cust_id, sum(…)
  from sales
  where …
  group by 1
  order by 2 desc limit 10
)
select pg_func(cust_id)
from top10;
```



Extending the Postgres query planner

Extensions can propose or enforce alternative plans for whole query or fragments.

```
with top10 as (
  select cust_id, sum(…)
  from sales
  where …
  group by 1
  order by 2 desc limit 10
)
select pg_func(cust_id)
from top10;
```

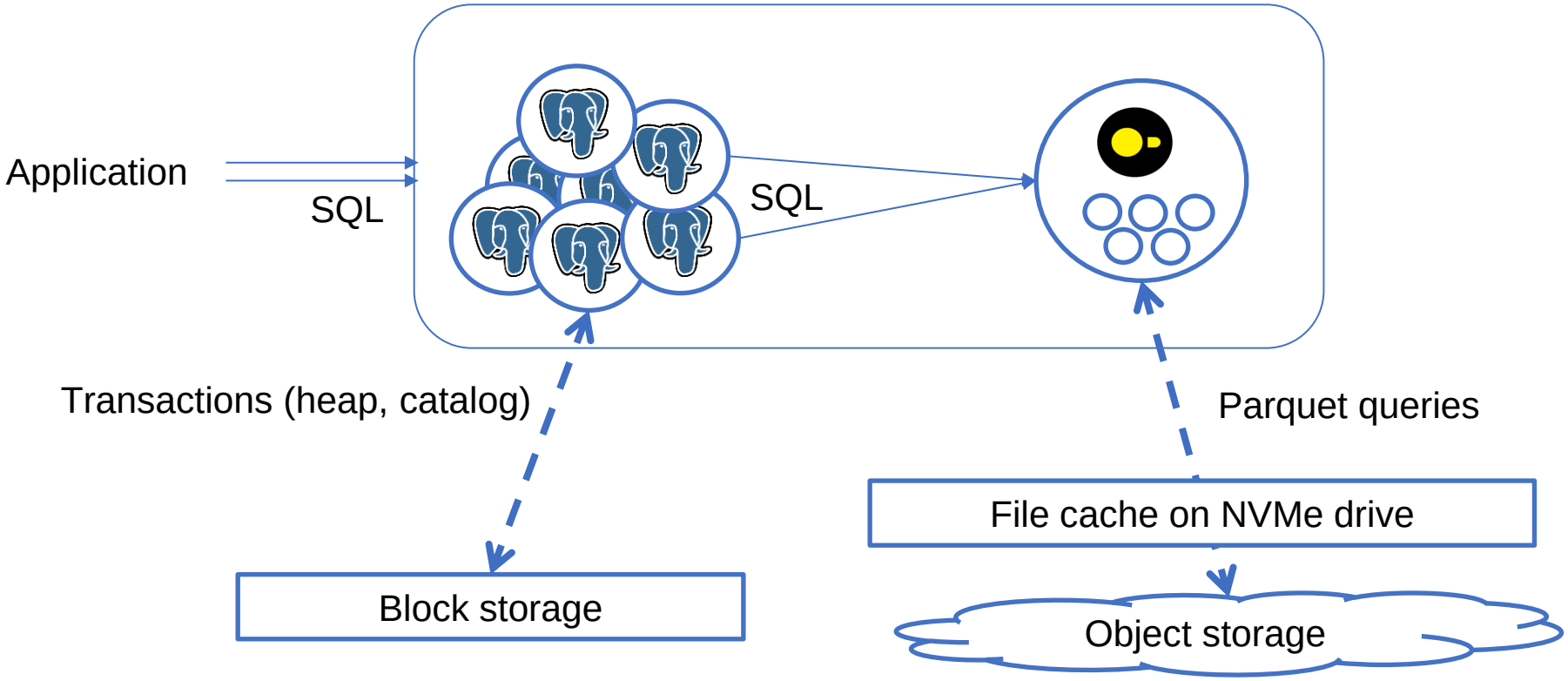


CteScan
pg_func(cust_id)

CustomScan (DuckDB)

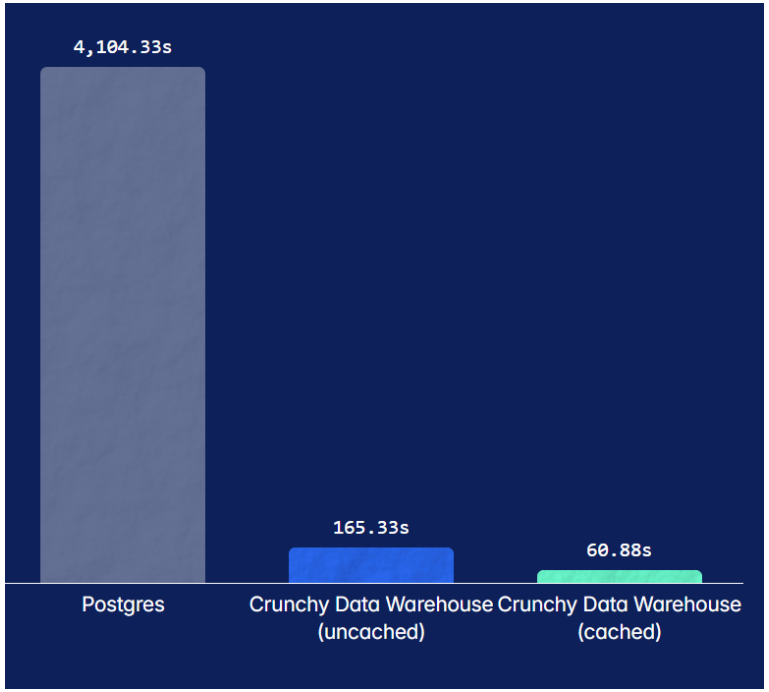
```
select …
from read_parquet(…)
where …
group by …
order by … desc limit 10
```

Hybrid architecture

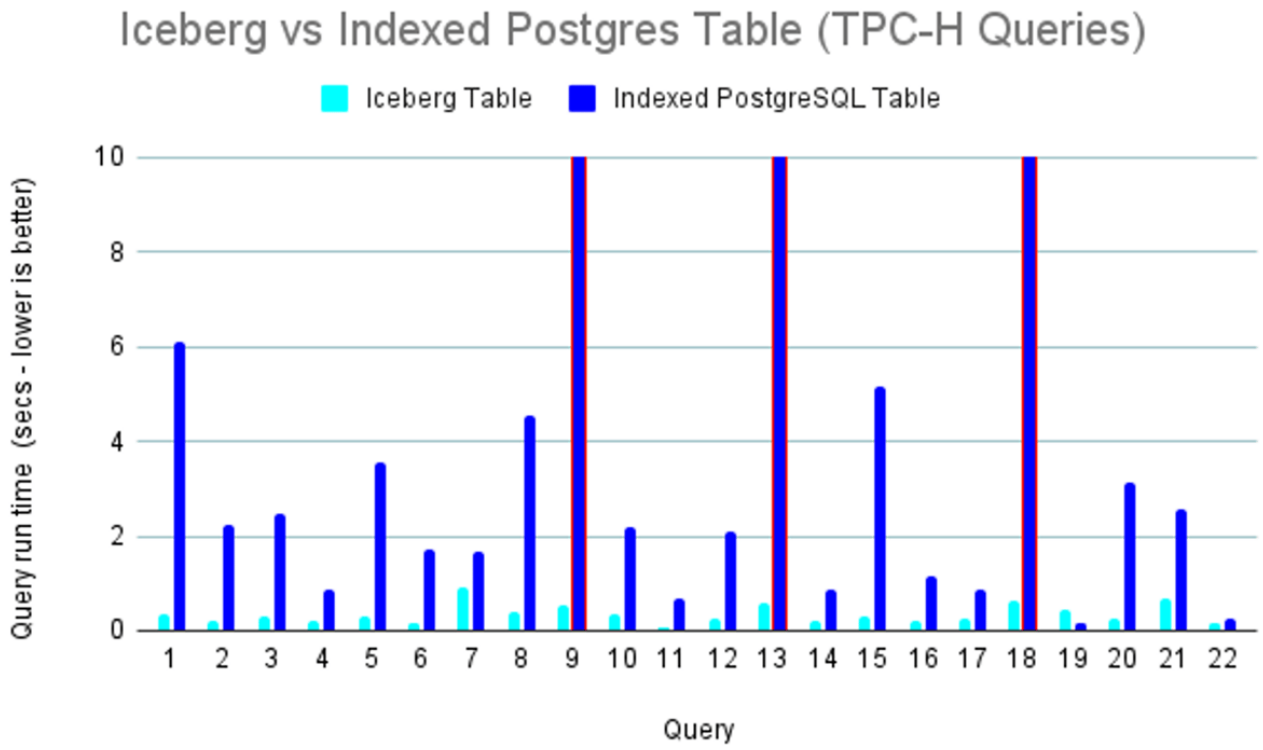


Iceberg to heap performance comparison

>50x faster on ClickBench



>10x faster on TPC-H

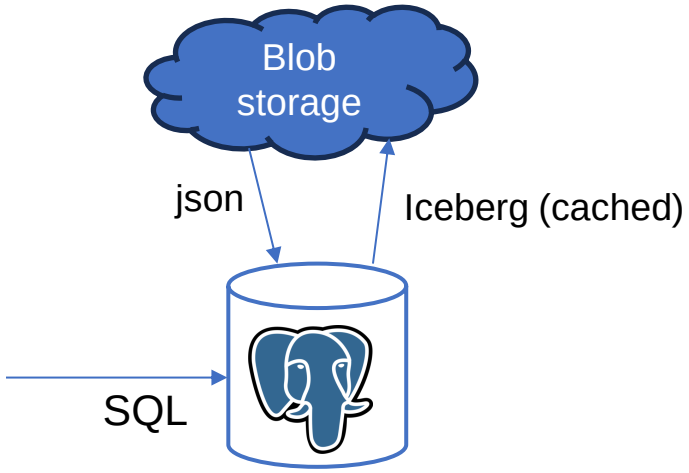


crunchy data

Zero-ETL log management

Incrementally & transactionally load new files into Iceberg

Query Iceberg using SQL




```
-- Create a table to query JSON logs (infer columns)
create foreign table logs ()
server crunchy_lake_analytics
options (path 'az://crunchy-bridge/logs/*.json.gz', filename 'true');

-- Create an Iceberg table with the same schema
create table logs_iceberg (like logs)
using iceberg;

-- Set up a pg_incremental job to process new & existing files

select incremental.create_file_list_pipeline('process-logs',
   file_pattern := 'az://crunchy-bridge/logs/*.json.gz',
   batched := true,
   command := $$
       insert into logs_iceberg
       select * from logs where _filename = any($1)
   $$);
```

crunchy data

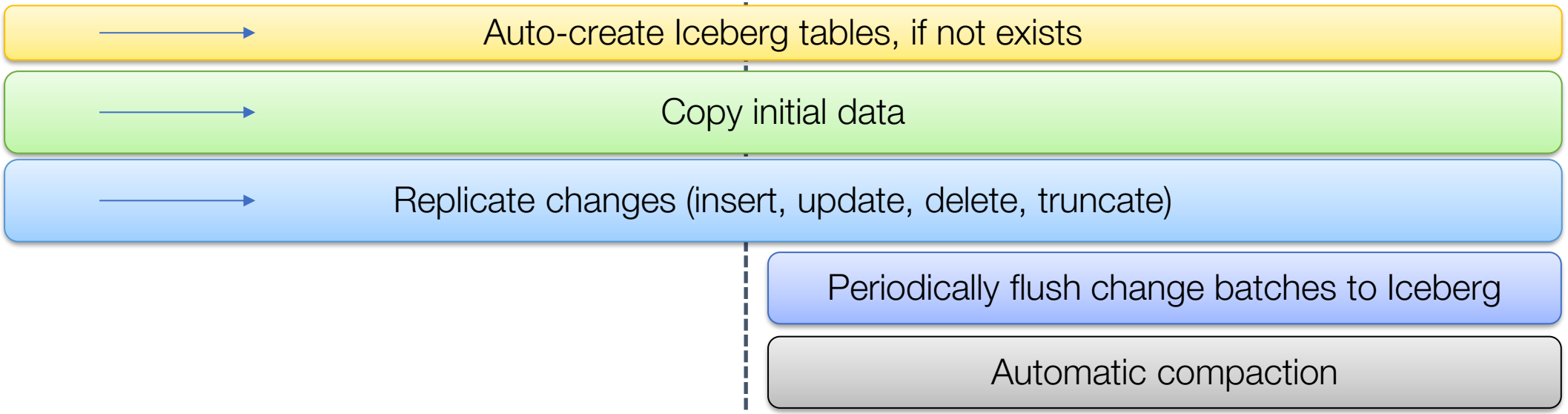
Logical replication from Postgres to Iceberg

Source (any postgres):

```
CREATE PUBLICATION pub
FOR TABLE orders, customers;
```

Destination (Crunchy Data Warehouse):

```
CREATE SUBSCRIPTION sub
CONNECTION 'host=... '
PUBLICATION pub
WITH (create_tables_using 'iceberg');
```



$

Summary

PostgreSQL can offer a comprehensive data warehouse/lakehouse experience with Iceberg and full transaction support.

Fast analytical queries by integrating DuckDB and write-through caching.

Iceberg SQL catalog protocol is supported for external query engines.

Crunchy Data Warehouse is the first production-ready solution.

Let’s see how it goes ☺

Questions?

marco.slot@crunchydata.com

Crunchy Data blog