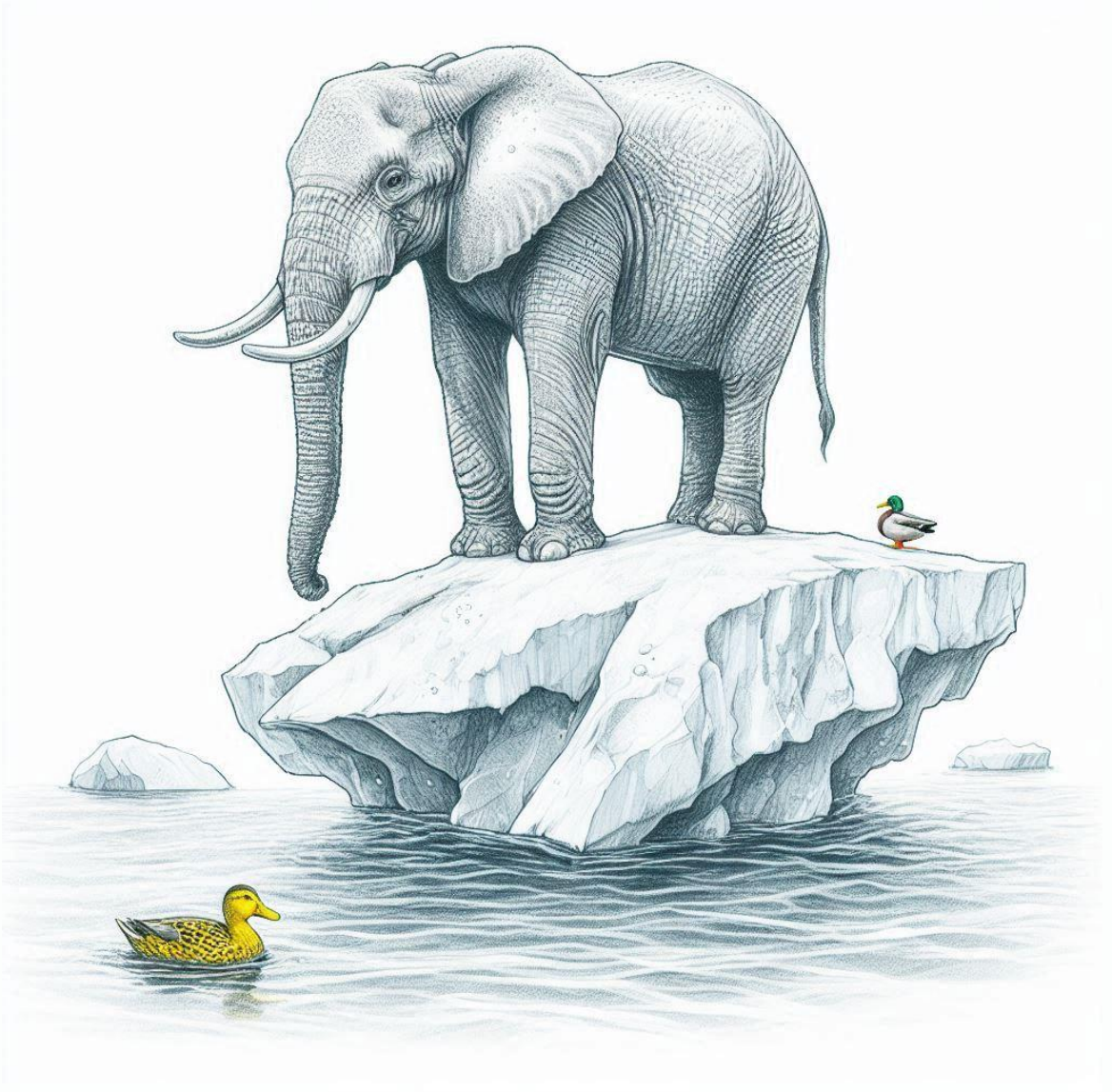
Building a Postgres Data Warehouse with Iceberg

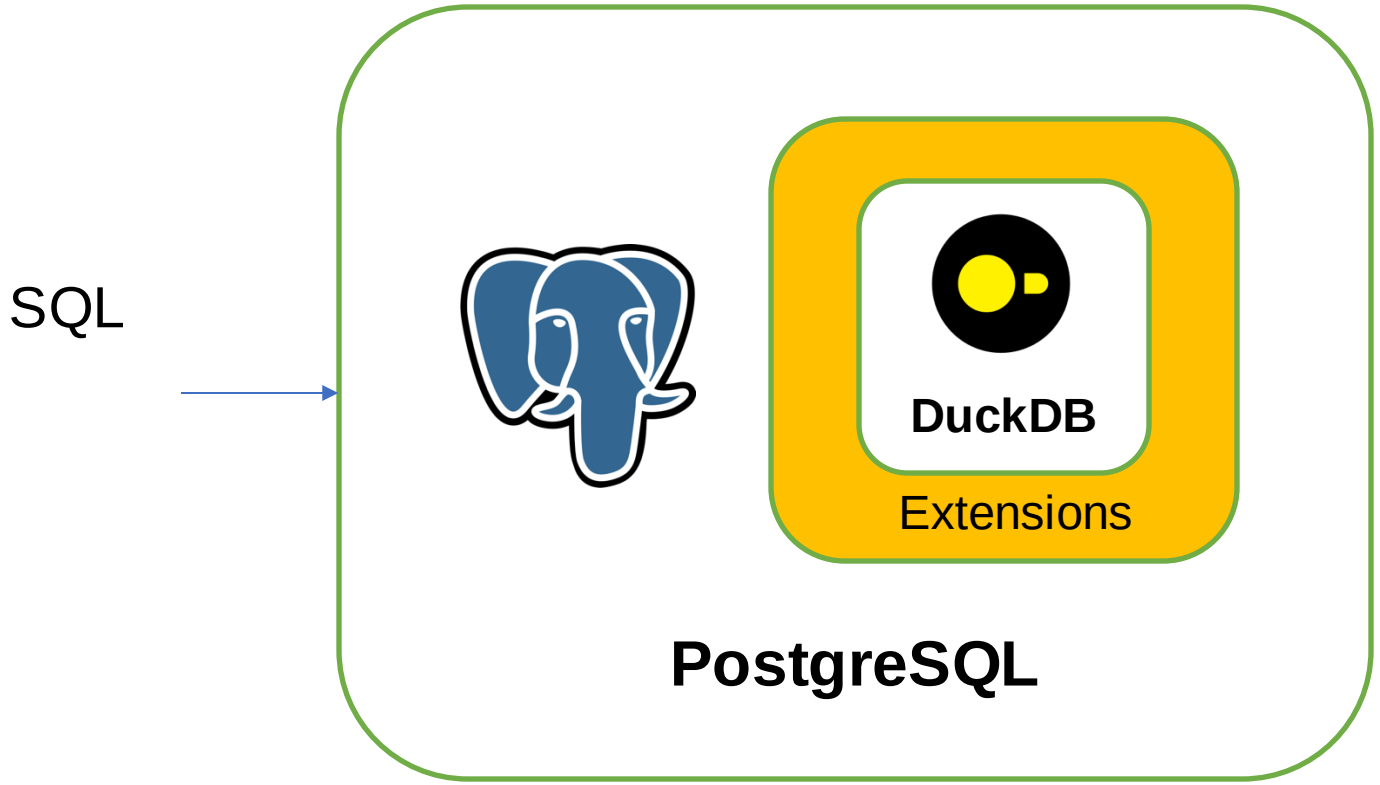
Marco Slot – marco.slot@crunchydata.com



crunchy data

OLTP	OLAP
Operational system of record	Analytics on collection of data sources
SQL	SQL
Transactions	Transactions
High query rate, small queries	Low query rate, big queries
Low response time	High scan throughput
User-facing applications	Business-facing dashboards
Mission-critical, always on	On demand, business hours

Hybrid OLTP/OLAP architecture



Postgres + Iceberg

PostgreSQL as an Iceberg query engine & catalog.

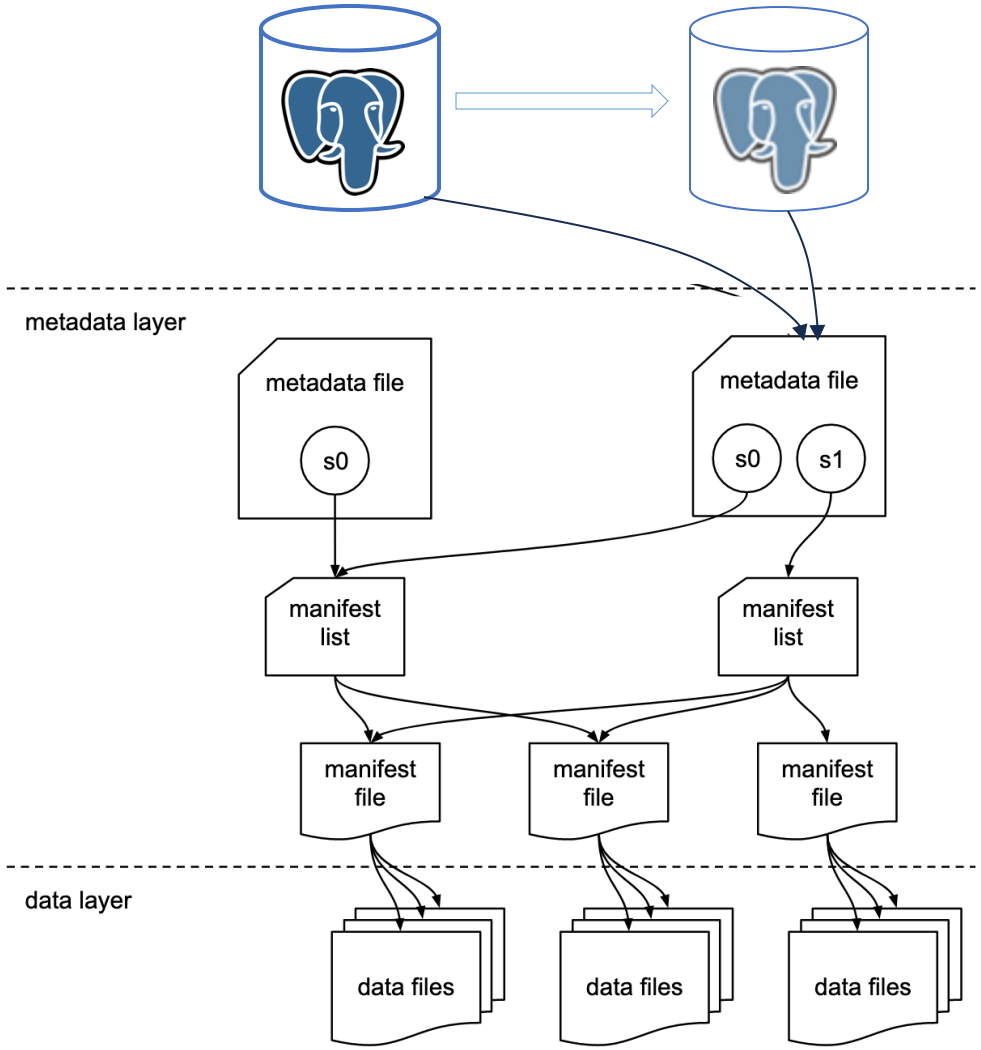
Extensions add an Iceberg table format.

Writes generate Parquet & Iceberg metadata.

Reads query Parquet using DuckDB.

Iceberg files (in S3) will still be available to replicas.

Iceberg catalog is in WAL, replicated to replicas.

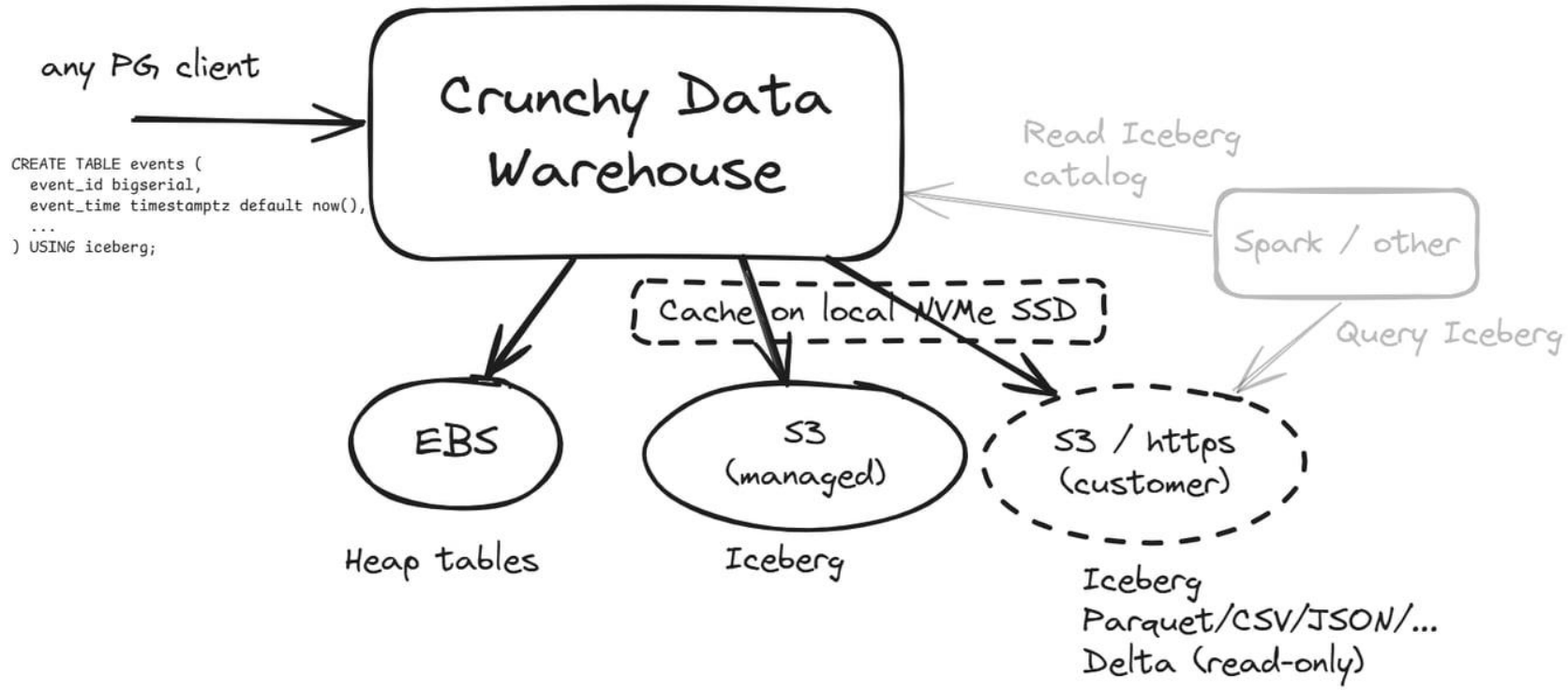


crunchy data

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

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 S3, 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 S3, 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(…));
```

Why Hybrid OLTP/OLAP architecture?

Bad idea:

- Run analytical queries on operational system of record

Good ideas:

- Speed up “accidental Postgres data warehouse”
- Analytics Database +: high throughput insert queues, materialized views
- Operational database +: Archive to Parquet, Iceberg
- Operational ~ Analytical data model consistency
- …

Stack simplification!

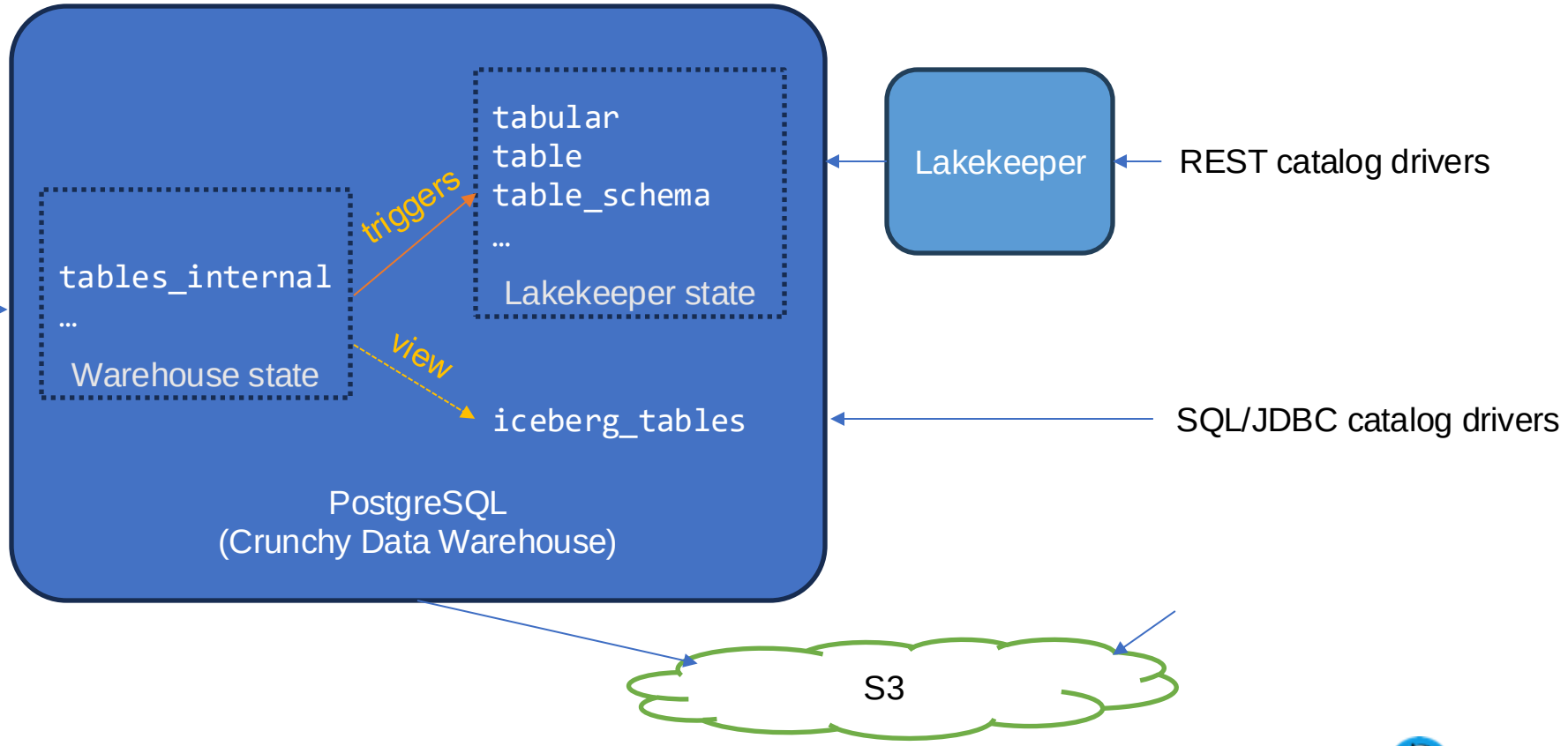
```
postgres=# /*
postgres*#  Use Iceberg with an insert queue for fast inserts
postgres*#  in Crunchy Data Warehouse.
postgres*# */
postgres-#
postgres-#
```

PoC: Lakekeeper + Crunchy Data Warehouse

Can synchronize state into Lakekeeper internally and offer REST API.

```
create table logs (
  event_id bigserial,
  server_id bigint,
  …
) using iceberg;

copy logs
from 's3://buck/003.csv';
```



crunchy data

Summary

PostgreSQL can offer a simple experience for Iceberg with full transaction support and no need for an external catalog.

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

SQL catalog protocol is supported for external query engines (or REST with Lakekeeper)

Crunchy Data Warehouse is the first production-ready solution.

Let’s see how it goes ☺

crunchy data

Questions?