Beyond Traditional Databases: Introducing the Type III Architecture

Javier Ramírez
@supercoco9
Fast Data Advocate

QuestDB

When was the last time you wished your database had a more proprietary data format?

"YOU SHALL NOT PASS"

20 years ago: No fast databases. No analytics

- Databases followed the OLTP pattern.
- Heavily biased for reads, not writes
- Designed for a few millions of rows (best case)
- Speed up queries using indexes

The database is the bottleneck

- Every developer in the 90s

Then there was OLAP and NoSQL

- NoSQL optimized for fast inserts and fast non-analytical queries.

- OLAP optimized for large batch inserts and fast analytical queries via complex indexes and materialization/ denormalization/ duplication.

Then there was OLAP separation of storage

- Following success of Map/Reduce and HDFS for data processing, many OLAP databases separated storage from computation, allowing for distributed queries.
- The data lake concept was created.
- Writes were still mostly batched.

OLAP = Immutable*

- File formats typically used in OLAP made it very costly to update individual records.

- Cloud-based object stores typically work with immutable files with no random-access updates.

* Until recently

Some ugly truths about streaming data

- It can get very big. It never stops. Always incomplete.
- It will burst, lag, and arrive out of order. It will get updated after you've already emitted results.
- Individual data points lose value over time, but long-term aggregations are priceless.
- Analysts prefer low latency and data freshness.

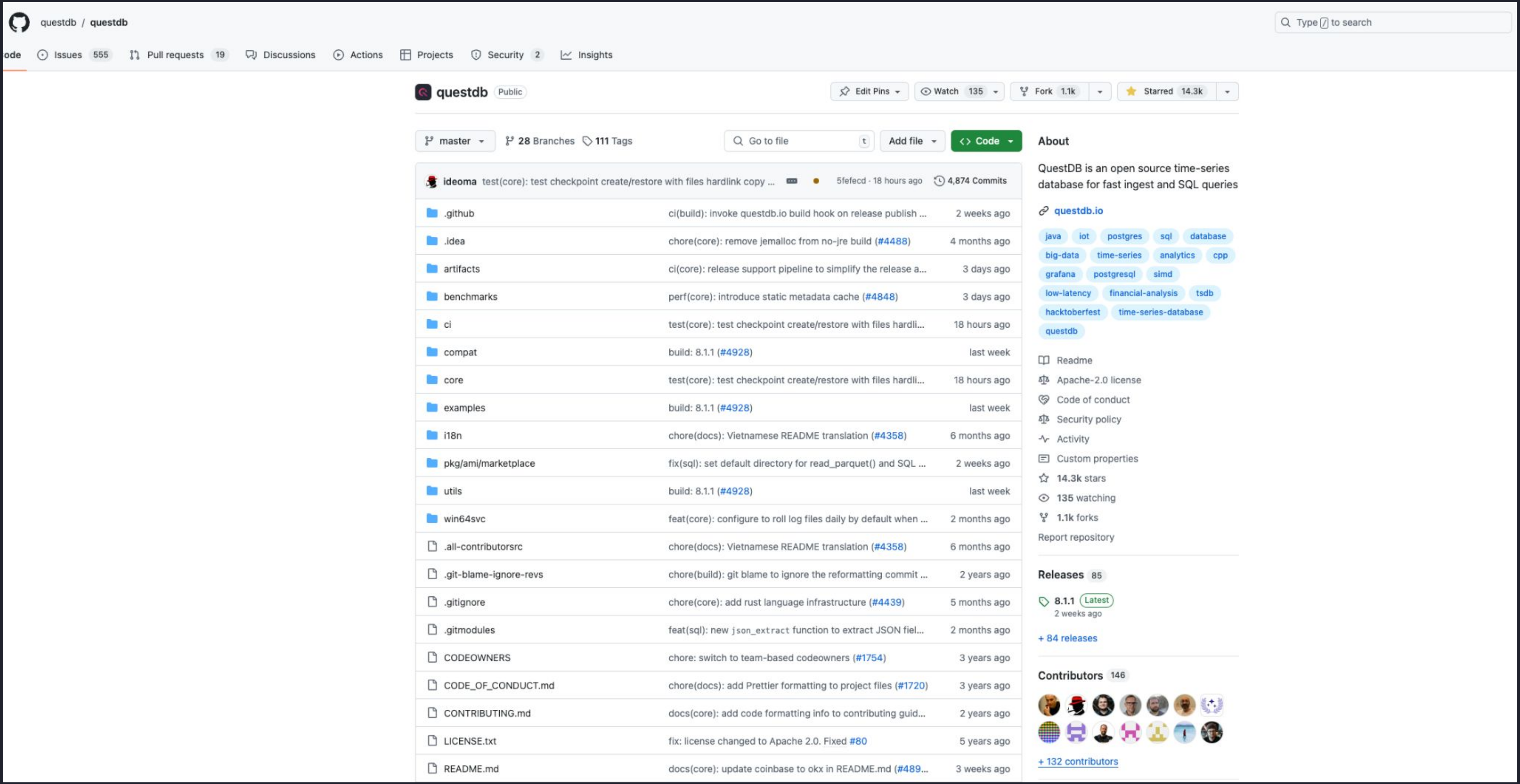
My fast database definition*:

Designed for performant frequent multi million record ingestion and performant frequent queries over multi billion record datasets.

* My own biased definition. But this is my talk, so just setting the context

Time Series Databases Enter the Scene

- Time Series Databases specialise in very fast ingestion, very fast queries over nascent data, and powerful time-based analytical queries.
- They focus on nascent data, deleting, downsampling, or slowing-down older data.



questdb Public

ideoma test(core): test checkpoint create/restore with files hardlink copy ... 5fefecd · 18 hours ago 4,874 Commits

.github	ci(build): invoke questdb.io build hook on release publish ...	2 weeks ago
.idea	chore(core): remove jemalloc from no-jre build (#4488)	4 months ago
artifacts	ci(core): release support pipeline to simplify the release a...	3 days ago
benchmarks	perf(core): introduce static metadata cache (#4848)	3 days ago
ci	test(core): test checkpoint create/restore with files hardli...	18 hours ago
compat	build: 8.1.1 (#4928)	last week
core	test(core): test checkpoint create/restore with files hardli...	18 hours ago
examples	build: 8.1.1 (#4928)	last week
i18n	chore(docs): Vietnamese README translation (#4358)	6 months ago
pkg/ami/marketplace	fix(sql): set default directory for read_parquet() and SQL ...	2 weeks ago
utils	build: 8.1.1 (#4928)	last week
win64svc	feat(core): configure to roll log files daily by default when ...	2 months ago
.all-contributorsrc	chore(docs): Vietnamese README translation (#4358)	6 months ago
.git-blame-ignore-revs	chore(build): git blame to ignore the reformatting commit ...	2 years ago
.gitignore	chore(core): add rust language infrastructure (#4439)	5 months ago
.gitmodules	feat(sql): new `json_extract` function to extract JSON fiel...	2 months ago
CODEOWNERS	chore: switch to team-based codeowners (#1754)	3 years ago
CODE_OF_CONDUCT.md	chore(docs): add Prettier formatting to project files (#1720)	3 years ago
CONTRIBUTING.md	docs(core): add code formatting info to contributing guid...	2 years ago
LICENSE.txt	fix: license changed to Apache 2.0. Fixed #80	5 years ago
README.md	docs(core): update coinbase to okx in README.md (#489...	3 weeks ago

About

QuestDB is an open source time-series database for fast ingest and SQL queries

questdb.io

java iot postgres sql database big-data time-series analytics cpp grafana postgresql simd low-latency financial-analysis tsdb hacktoberfest time-series-database questdb

Readme
Apache-2.0 license
Code of conduct
Security policy
Activity
Custom properties
14.3k stars
135 watching
1.1k forks
Report repository

Releases 85

8.1.1 Latest
2 weeks ago

+ 84 releases

Contributors 146

+ 132 contributors

https://github.com/questdb/questdb

QuestDB

- Column-first parallel SQL engine with JIT compiler
- Column-first, partitioned data store, sorted by timestamp.
- No indexes needed*. Data immediately available after write.
- Predictable ingestion rate, even under demanding workloads.
- Built-in event deduplication. Row updates and upserts.

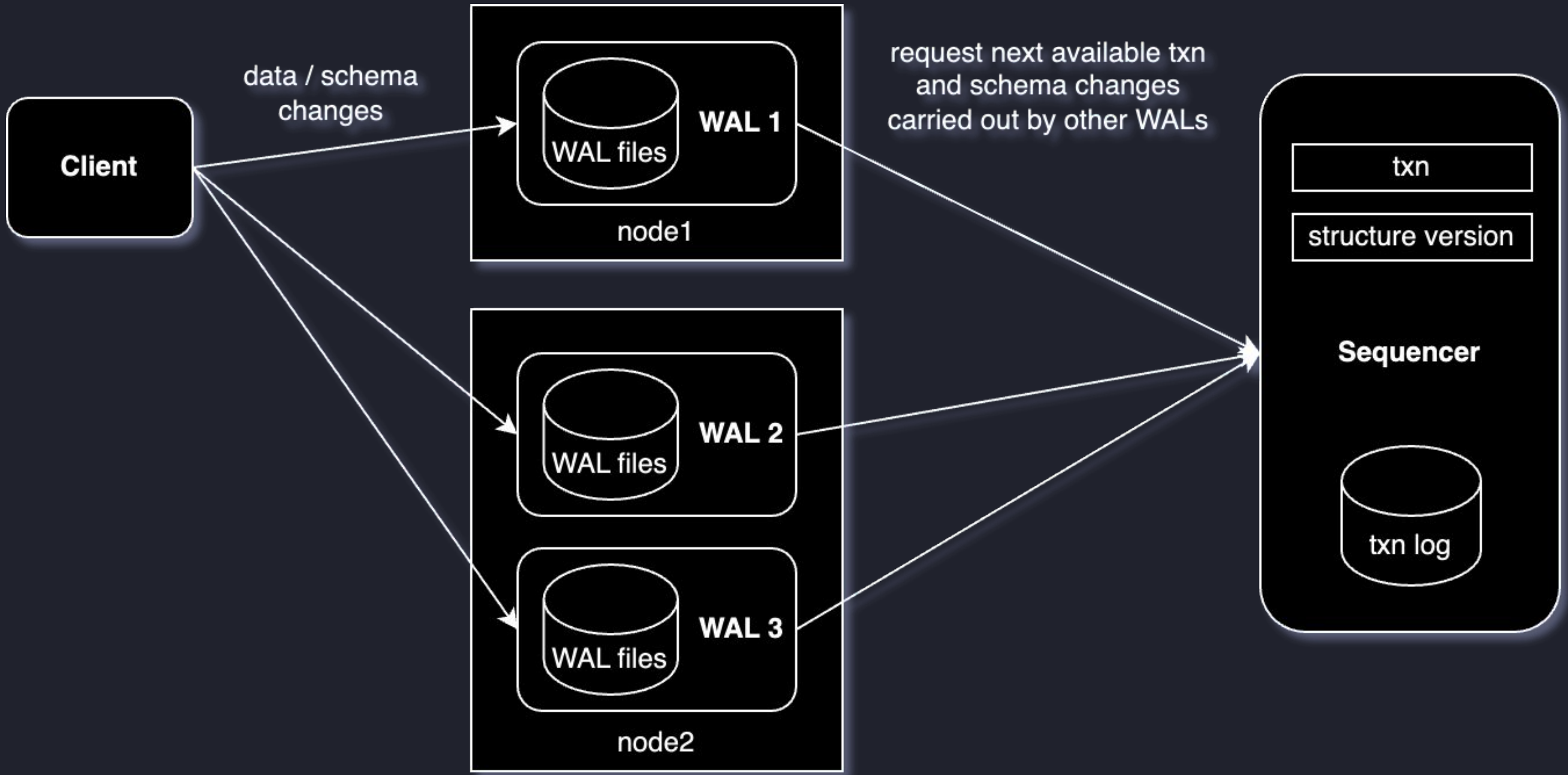
QuestDB in action: quick showcase

https://dashboard.demo.questdb.io/d/fb13b4ab-b1c9-4a54-a920-b60c5fb0363f/public-dashboard-questdb-io-use-cases-crypto?orgId=1&refresh=750ms

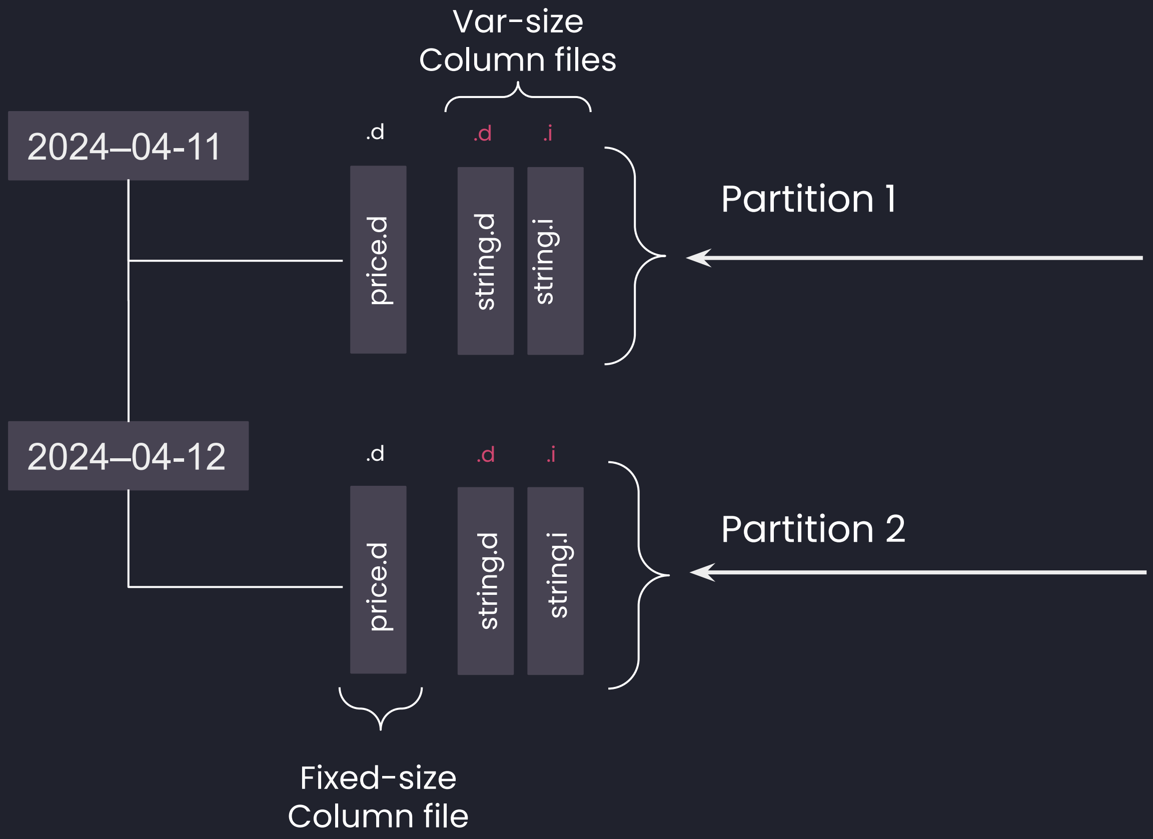
https://demo.questdb.io

https://github.com/questdb/time-series-streaming-analytics-template

Parallel Write Ahead Log (WAL)



Storage Engine - file system layout



Physical layout of table storage

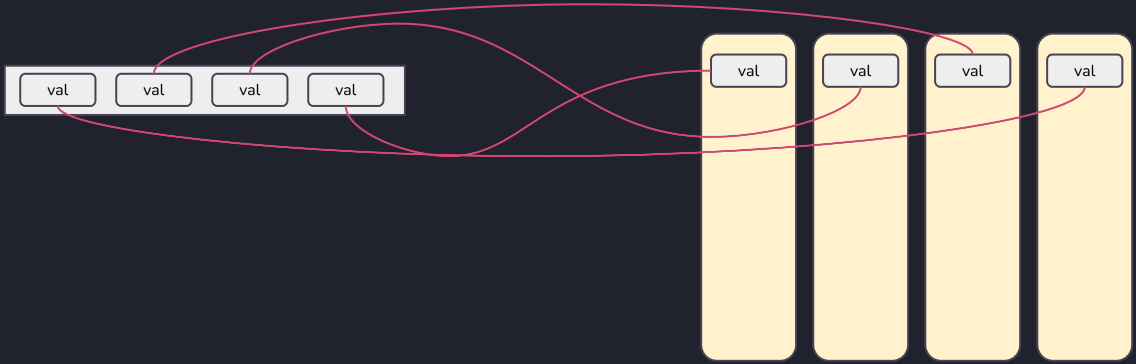
- One WALx subfolder per table and connection
- txn_seq folder to serialize transactions across parallel WAL folders
- _event file as transaction index for each WAL folder
- _meta files with schema version/data
- One* file per column, with the binary data
- _cv file for Commit Verification

```
├── db
│   ├── Table
│   │   │
│   │   ├── Partition 1
│   │   │   ├── _archive
│   │   │   ├── column1.d
│   │   │   ├── column2.d
│   │   │   ├── column2.k
│   │   │   └── ...
│   │   ├── Partition 2
│   │   │   ├── _archive
│   │   │   ├── column1.d
│   │   │   ├── column2.d
│   │   │   ├── column2.k
│   │   │   └── ...
│   │   ├── txn_seq
│   │   │   ├── _meta
│   │   │   ├── _txnlog
│   │   │   └── _wal_index.d
│   │   ├── wal1
│   │   │   └── 0
│   │   │       ├── _meta
│   │   │       ├── _event
│   │   │       ├── column1.d
│   │   │       ├── column2.d
│   │   │       └── ...
│   │   ├── wal2
│   │   │   └── 0
│   │   │   │   ├── _meta
│   │   │   │   ├── _event
│   │   │   │   ├── column1.d
│   │   │   │   ├── column2.d
│   │   │   │   └── ...
│   │   │   └── 1
│   │   │       ├── _meta
│   │   │       ├── _event
│   │   │       ├── column1.d
│   │   │       ├── column2.d
│   │   │       └── ...
│   │   │
│   │   ├── _meta
│   │   ├── _txn
│   │   └── _cv
```

Ingress/Egress paradox

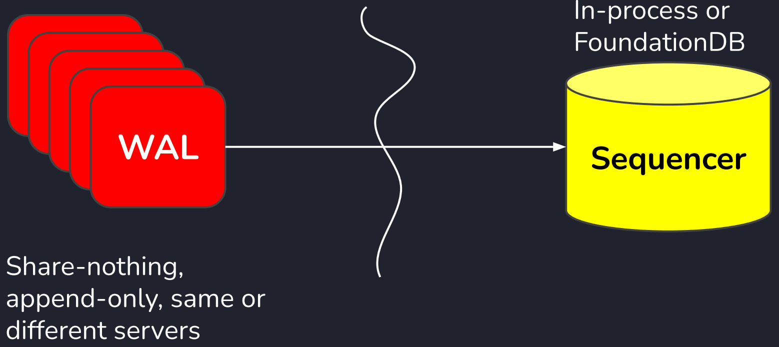
- Columnar (or column-first) data store favors egress.

- Most nascent data ingress is row-first.



Multi-primary ingestion

- Metadata and information about cluster members is coordinated via a sequencer backed by FoundationDB.
- Optimistic locking for conflict resolution.
- Client libraries transparently get the addresses of available primaries and replicas to send data and queries.


In-process or FoundationDB
WAL
Sequencer
Share-nothing, append-only, same or different servers

Our ability to look at data and see trends helps us to make better predictions about what comes next.

— Tim Berners-Lee, Inventor of the World Wide Web

New open file formats

- The hadoop ecosystem developed the Apache Hudi format, Netflix developed Apache Iceberg, and Databricks developed Delta Lake. The three of them are open formats and allow for mutable data, transactions, schema evolution, and streaming.

- They are open, so multiple data engines and applications can share the same datasets with no duplication.

Data Science and ML

- Dashboards and reports query over billions of records to produce a result with just a few filtered/aggregated rows.

- Might want to use the whole unaggregated dataset directly, or most of the dataset minus some outliers, or a subset of columns. On most databases that means a slow export, and expensive data duplication.

- Might want to use an aggregated downsampled version of the dataset, converting from multi billions of records to multi millions. Serializing and deserializing is slow and resource consuming.

What the present looks like

- Lakehouse Engine architecture offers the most flexibility, TSDBs generally not there yet.
- TSDBs double-down on the ingress performance, OLAP are on the backfoot.
- OLAP double down on storage cost and analytical workloads (queries).
- OLAP query engines are sophisticated, TSDBs are on the back foot.
- TSDB engines are simpler to operate. OLAP is more complex and sometimes cloud-only.

The type III database

- Distributed computing, with separate storage.
- Data is stored in open formats, for easier collaboration, no deserialization, and no duplication. With file compression/low cost storage.
- Allow data consumers to bypass the database on egress and ingress.
- Support for structured and semi-structured data (JSON…).
- Data egress is as performant as data ingress. Multi-million records per second (aggregated or not) can be streamed out of the database.

Avoiding streaming data deserialization: Apache Arrow

- Open Memory format, open database API, and open SQL dialect.
- Initially developed by Dremio, but widely adopted by many projects.
- Adopted by tools like Apache Spark, Pandas, Dask…
- Provides libraries for multiple programming languages (e.g., C++, Java, Python, R, Go…).

ADBC, like JDBC/ODBC, but with Arrow

- A set of abstract APIs in different languages for working with databases and Arrow data.

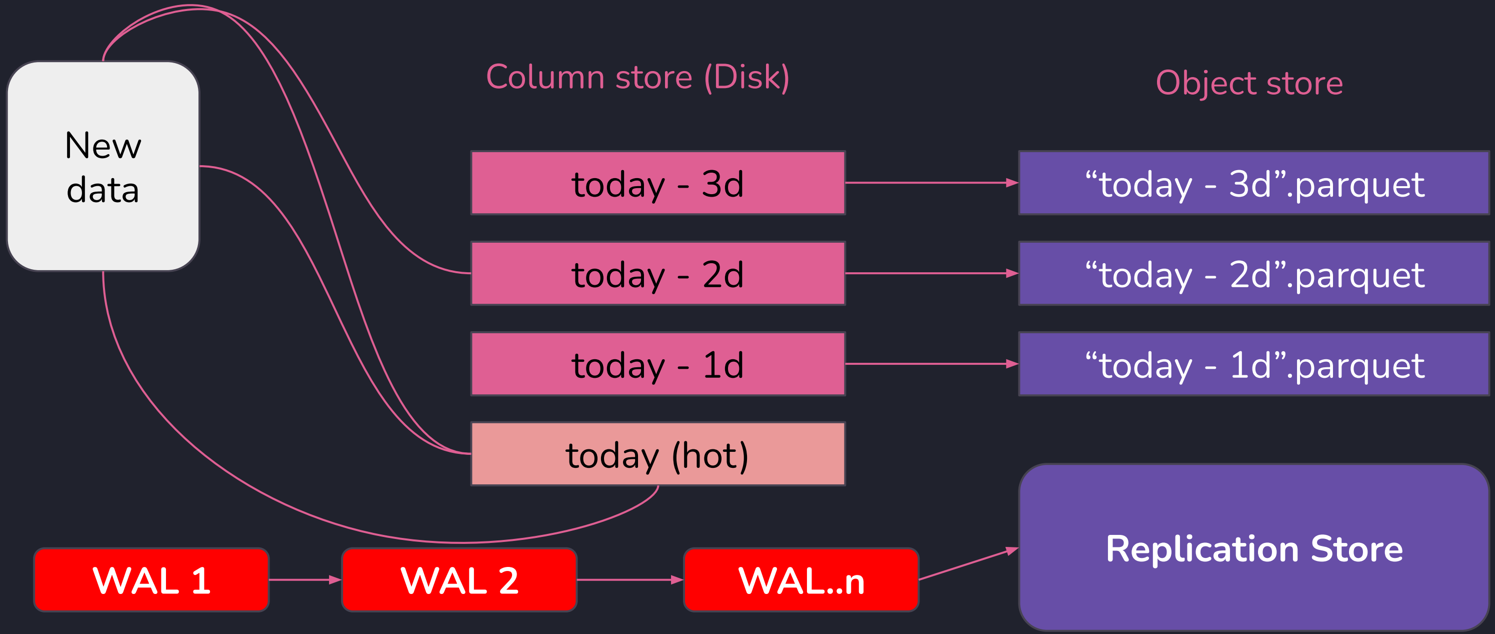
- Result sets of queries in ADBC are all returned as streams of Arrow data, not row-by-row. Client app does not need to convert rows to columns.

- Zero-copy. Client can use directly the values sent over the wire.

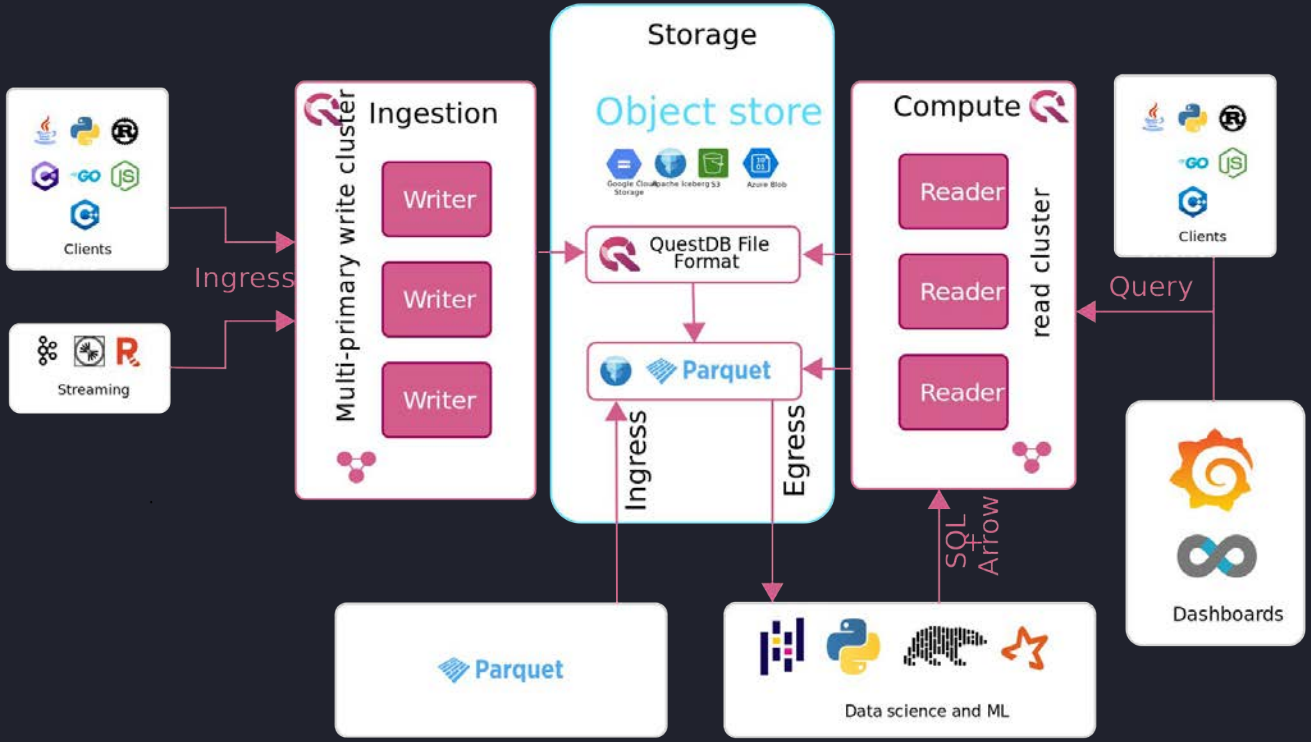
The (near) future of QuestDB

- Address the gap between time series and OLAP queries.
- Distributed Query Engine, decoupled from storage.
- High performance ingress (via streaming protocol) and egress via ADBC.
- Pgwire still supported for compatibility with the ecosystem.
- Data is stored in compressed parquet.
- The database engine can read parquet data produced externally.

Balancing hot and cold data: the data first mile



QuestDB Type III Architecture



QuestDB and Parquet Quick Demo

QuestDB OSS

Open Source. Self-managed. Suitable for demanding production workloads.
https://github.com/questdb/questdb

QuestDB Enterprise

Licensed. Self-managed or BYOC. Enterprise features like RBAC, replication, TLS on all protocols, cold storage, K8s operator…
https://questdb.io/enterprise/

We 💕 contributions and GitHub ⭐ stars

- https://github.com/questdb/questdb
- https://questdb.io
- https://demo.questdb.io
- https://slack.questdb.io/
- https://github.com/questdb/time-series-streaming-analytics-template

Javier Ramírez
@supercoco9
Fast Data Advocate

THANK YOU!