



Derk van Veen

The things you can do with partitions you never dreamed of



Derk van Veen

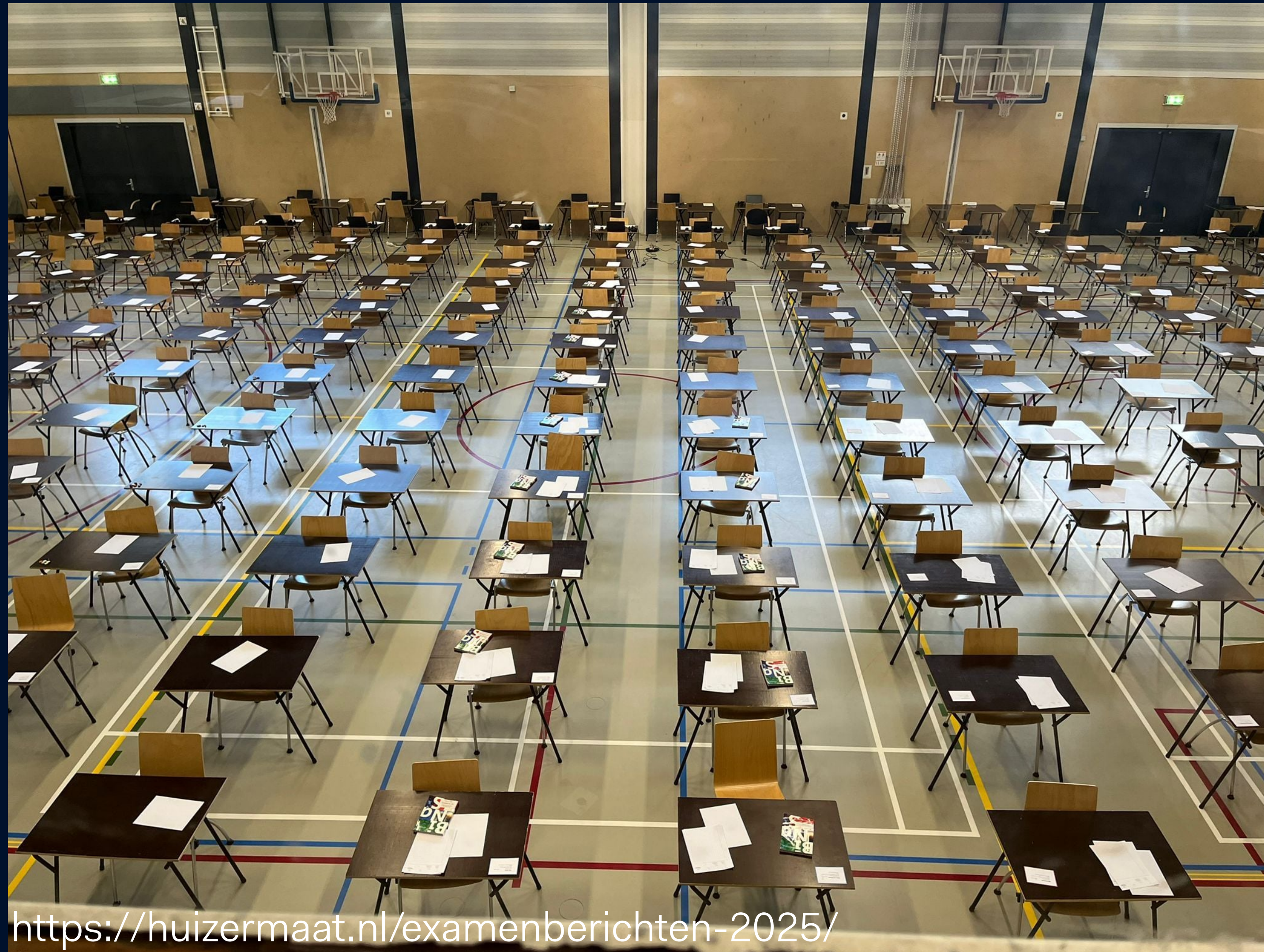
Partitioning incident creator #1

Adyen

adyen

engineered
for ambition





<https://huizermaat.nl/examenberichten-2025/>

adyen

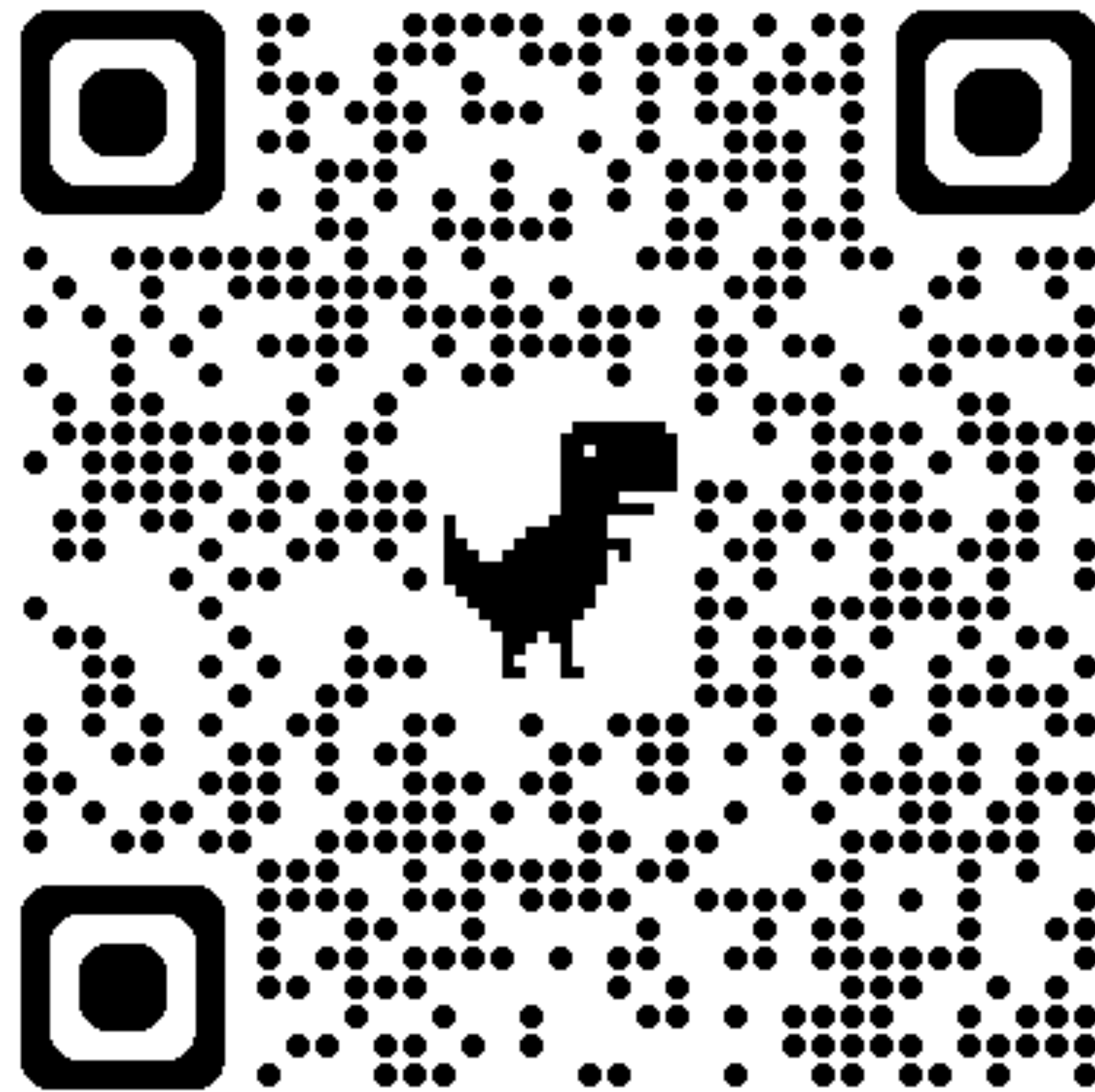
engineered
for ambition



<https://news.wine.co.za/>

adyen

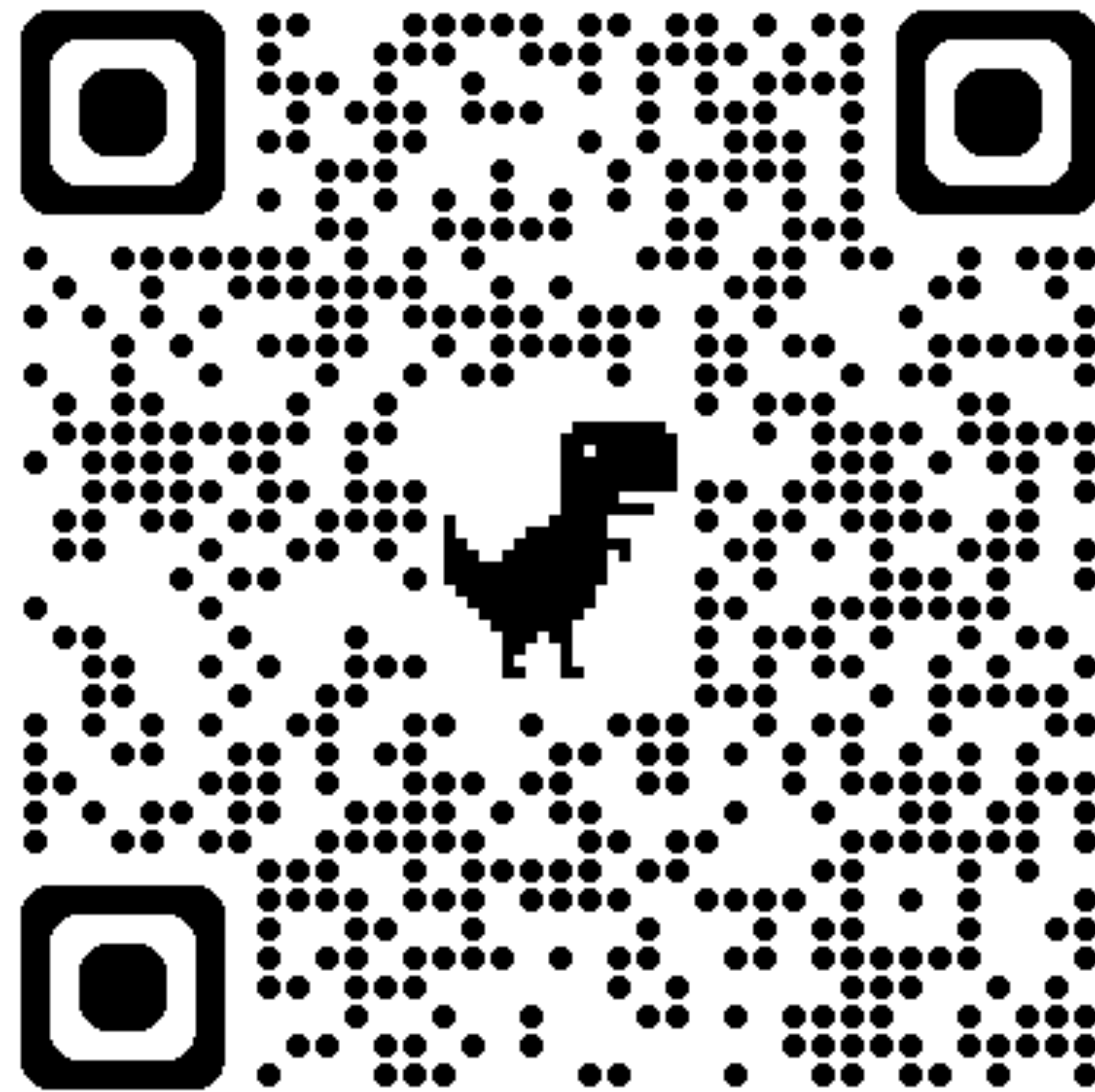
engineered
for ambition



adyen

engineered
for ambition

Partition Lifecycle Management



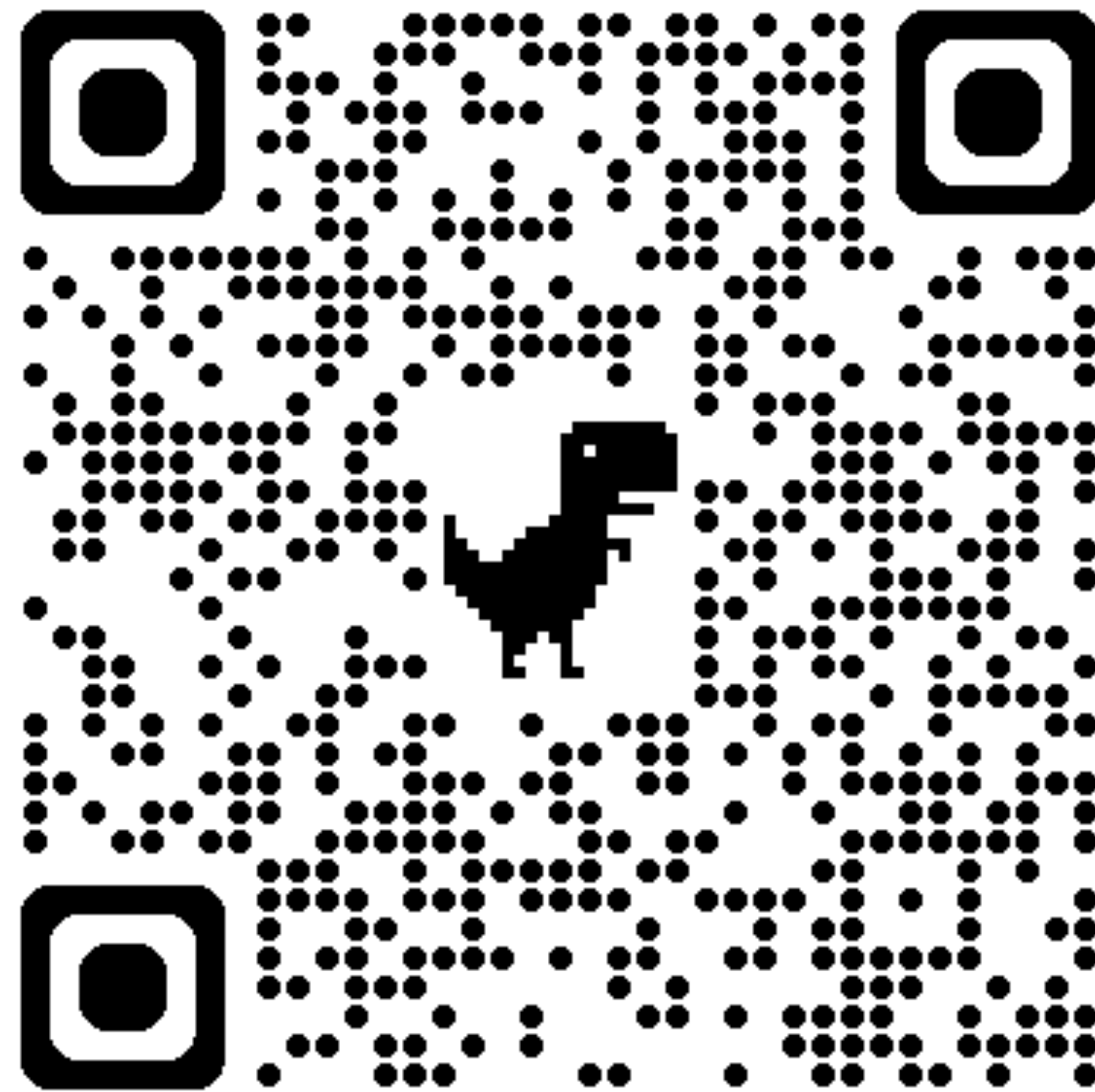
adyen

engineered
for ambition

Automatic (re)alignment



engineered
for ambition



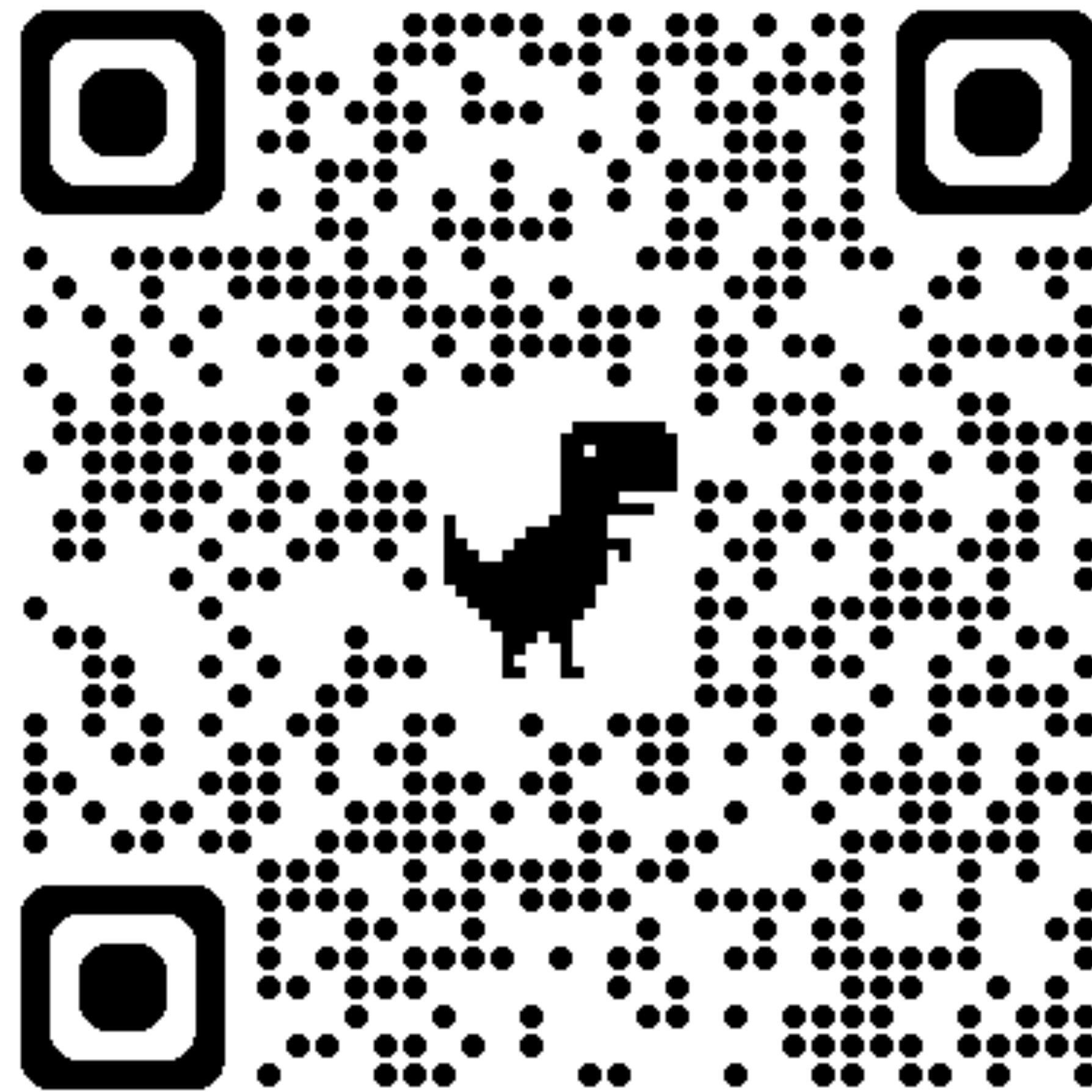
adyen

engineered
for ambition

Column tetris

adyen

engineered
for ambition



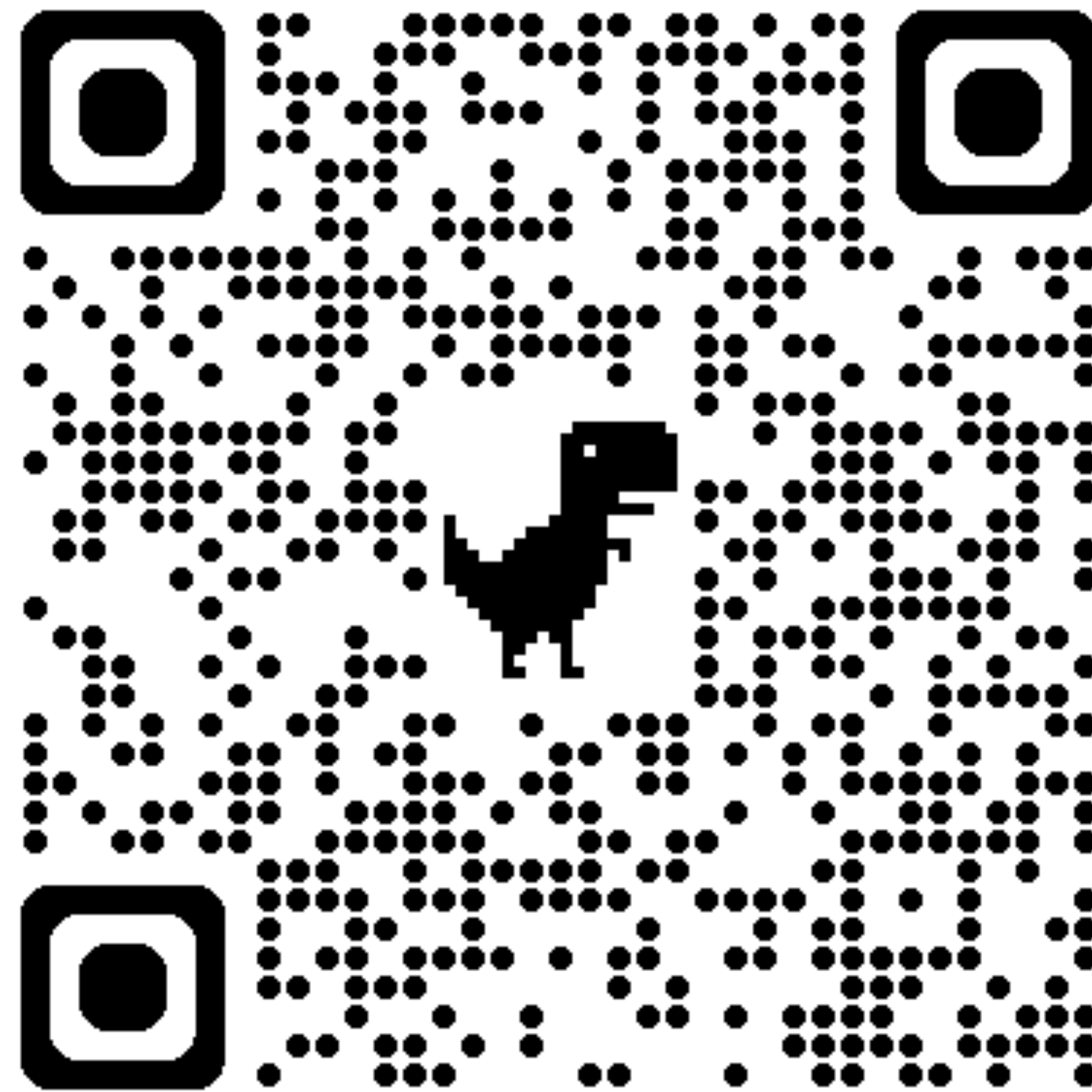
adyen

engineered
for ambition

Rollback



engineered
for ambition



adyen

engineered
for ambition

THANKS FOR LISTENING!



**PostgreSQL Conference
Europe 2026**

Valencia

Oct 20th - 23rd

WELCOME TO
VALENCIA



Daniel Phiri

Off the WAL: Postgres to ClickHouse Without
Logical Replication

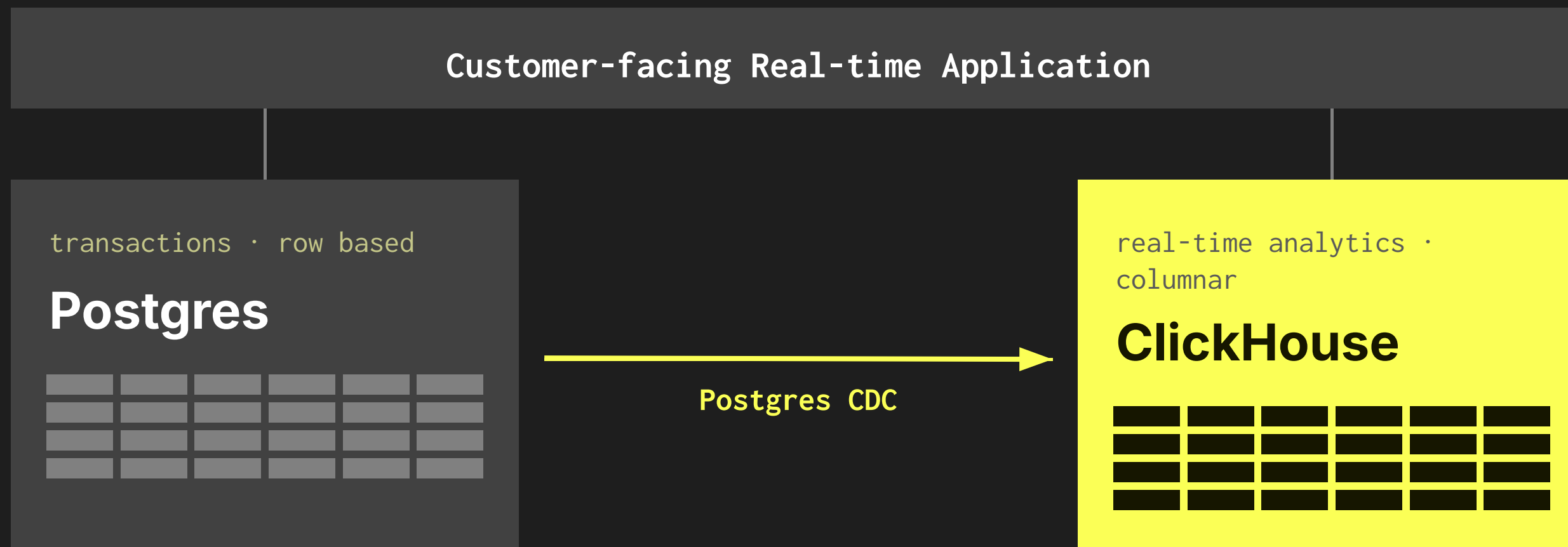
Off the WAL

Postgres to ClickHouse Without Logical Replication

October 2026

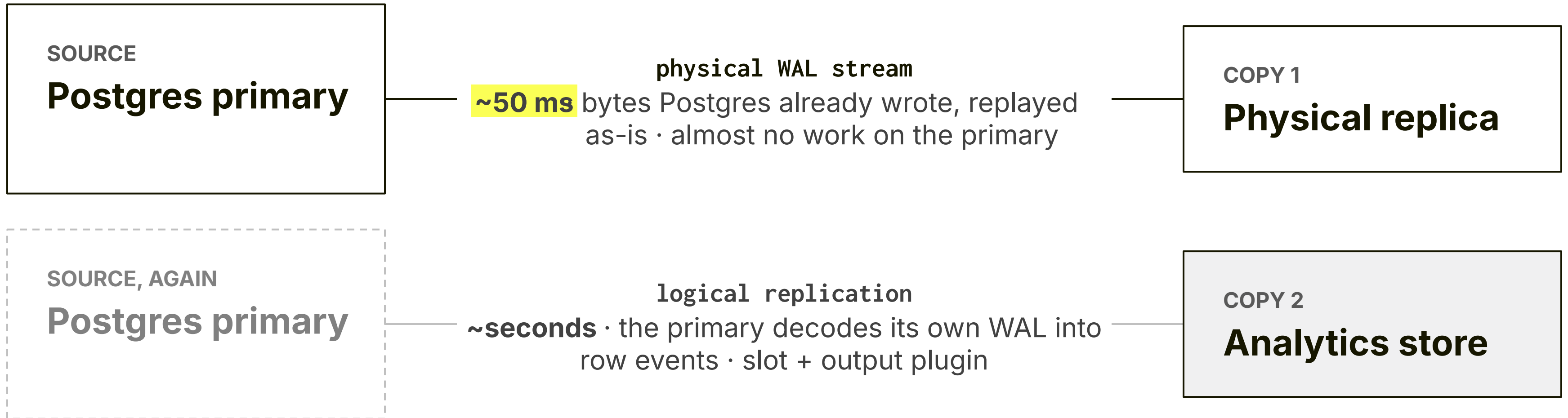
github.com/ClickHouse/walshadow

Postgres + ClickHouse is becoming **the default data stack**



The arrow in the middle is this talk. At scale it is the slowest and most fragile piece of the stack, and it was slower than it needed to be. So the Postgres team built WalShadow.

One WAL, read two ways

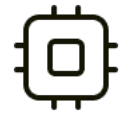


The logical replication tax



Slots you babysit

Forget one and WAL piles up on the primary until the disk fills.



Decoding in-process

The output plugin runs on the primary's CPU, right next to your OLTP traffic.



Fragile schema changes

DDL never makes it into the logical stream, so every ALTER TABLE is a chance to break the pipeline.



Seconds of lag

A committed row usually takes around 10 s to show up downstream.

To be fair, logical replication works, and it works at scale. PeerDB, which ClickHouse built, runs it for thousands of customers. The point here is narrower: the source should not have to do this work at all.

WalShadow: an analytical physical standby

An open-source engine that reads the same physical WAL stream a replica would, decodes it somewhere else, and writes ClickHouse native blocks. As far as the primary can tell, it is one more standby.

~~Replication slots~~

~~Output plugin~~

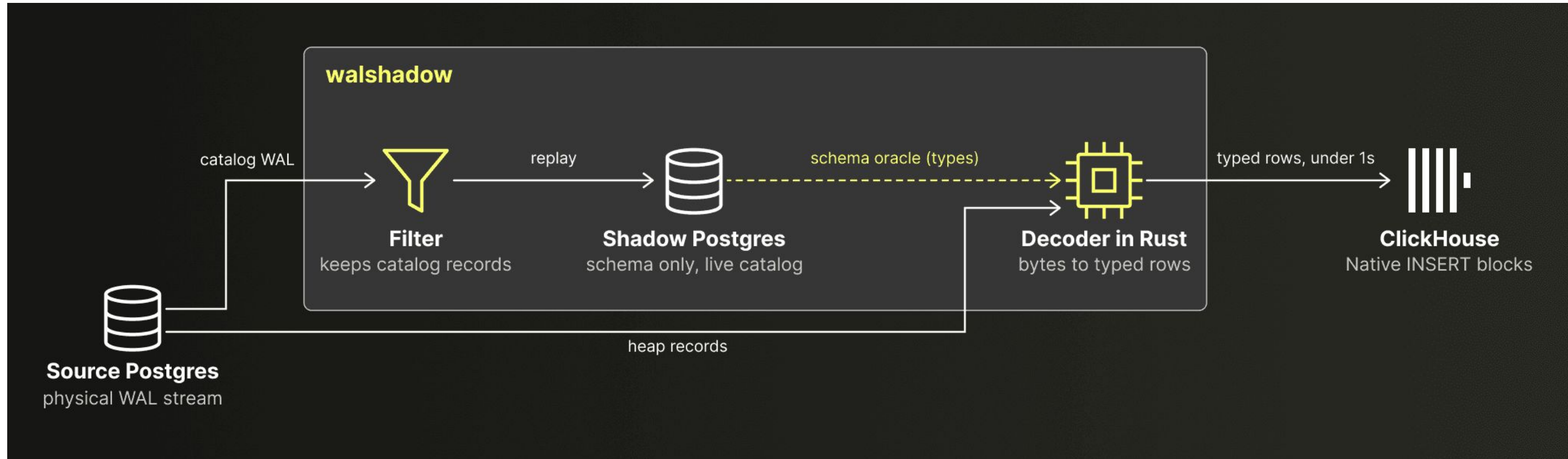
~~Kafka~~

~~JSON-pp~~

Full lifecycle: ADD/RENAME/DROP COLUMN and CREATE TABLE come through like any other change.

Still needs `wal_level = logical` on the source, so Postgres writes the extra row info into the WAL. That is all it does: no slot, no decoding on the primary.

How the data gets from WAL to ClickHouse



Catalog records replay

A schema-only shadow Postgres keeps types and DDL current.

Heap records decode in Rust

Page bytes become typed rows, off the source. Batched into native blocks.

Order comes from the WAL

Every row carries its *lsn*. DDL and truncates are the barriers.

Will the Fed cut rates this meeting?

SOURCE MARKET ACTIVITY · COMMITTED POSTGRESQL TRADES

33¢

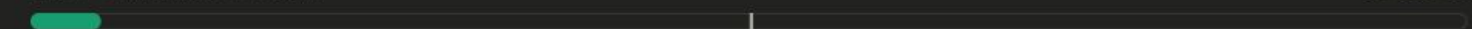
● Price (source) ● Live alert ● Delayed alert ● Burst window



WITH WALSHADOW

Unusual buying detected

BUY PRESSURE NOW

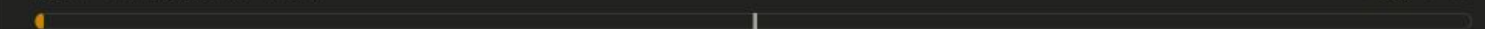


Alert in **309** ms · **during burst**

CLICKPIPES

Unusual buying detected

BUY PRESSURE NOW



Alert in **15,741** ms · **after burst ended**

COMMITTED TRADES / S

9,991

measured source commits · target 10,000

REPLICATION p95

256 ms

387 probes over 30s of a 30s window

LIVE ALERT TIME

309 ms

from first burst commit · measured in cloud

BUY 21@33¢ SELL 7@36¢ BUY 5@37¢ SELL 11@33¢ BUY 2@36¢ BUY 7@35¢ SELL 3@36¢ BUY 24@37¢ BUY 3@34¢ SELL

M101 · 325ms / 15,620ms

SIMULATE BURST

NEXT BURST —

D = details · F = fullscreen

Market: "Will the Fed cut rates this meeting?" The white line is the price of yes, taken from committed Postgres trades.

WalShadow alerts while the price is still climbing. Logical CDC alerts once the move is over.

WALSHADOW

309 ms During the burst

The alert fires inside the two-second window, with the price still rising.

CLICKPIPES (LOGICAL CDC)

15,741 ms After the burst ended

By then the price is nearly back to 40¢. The gap is mostly batch and flush interval, not decode speed.

The Numbers

~200_{ms}

Commit to visible

Physical replica baseline ~50 ms. Logical CDC ~10 s.

~50_x

Lower latency than logical CDC

What used to be seconds of lag is now a few hundred milliseconds.

289_k

Rows per second, sustained

The source was writing about 290K rows/s, so the pipeline is keeping up with it.

One table, 8-vCPU boxes on both sides. Your workload will look different, and you should measure it.

Try **WalShadow** today

github.com/ClickHouse/walshadow

Open source, Apache 2.0

On GitHub today. Point it at a Postgres that gives you physical WAL and go.

Private preview on ClickHouse Managed Postgres

Most managed Postgres services will not hand you physical WAL. Ours does, so WalShadow runs there fully managed.

THANKS FOR LISTENING!



**PostgreSQL Conference
Europe 2026**

Valencia

Oct 20th - 23rd

WELCOME TO
VALENCIA



Ayoub Kazar

Graph Queries: Reverted, Distributed, and Beyond

PostgreSQL as a Graph Database: Reverted, Distributed, and Oracle-compatible

One Hop, Two Hops, Three: SQL/PGQ, Citus, and IvorySQL

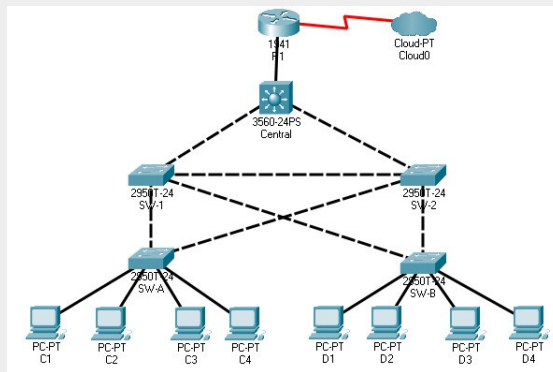
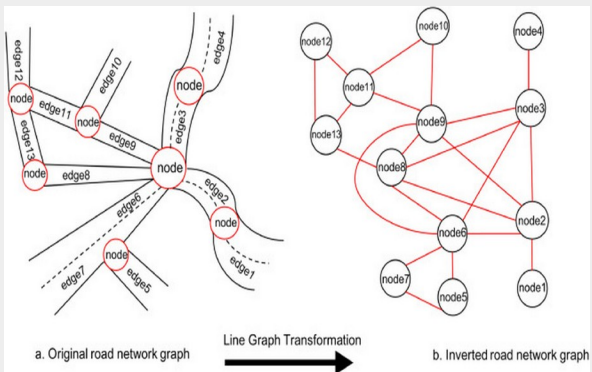
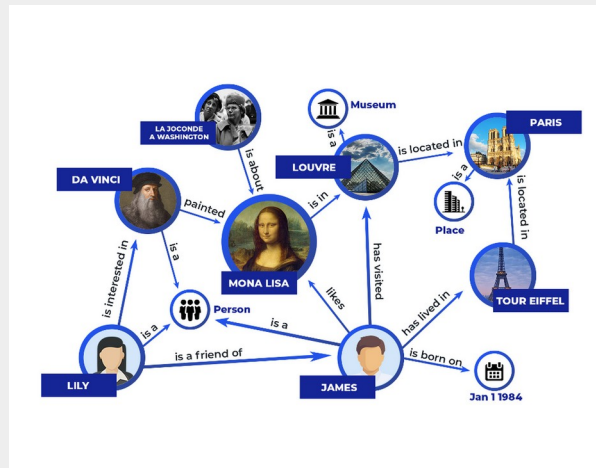
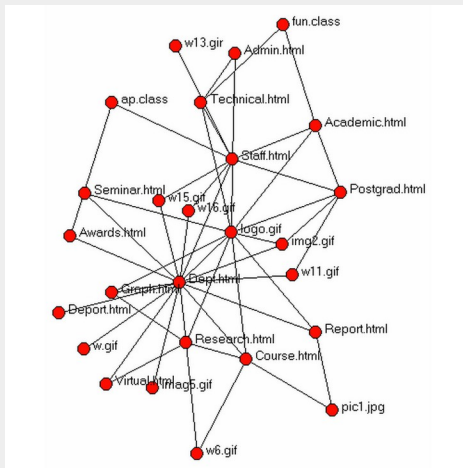
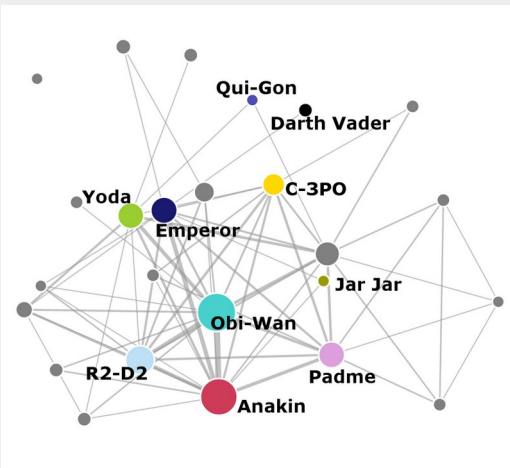
Ayoub Kazar - **Data Bene**
PGConf.EU 2026

Data Bene



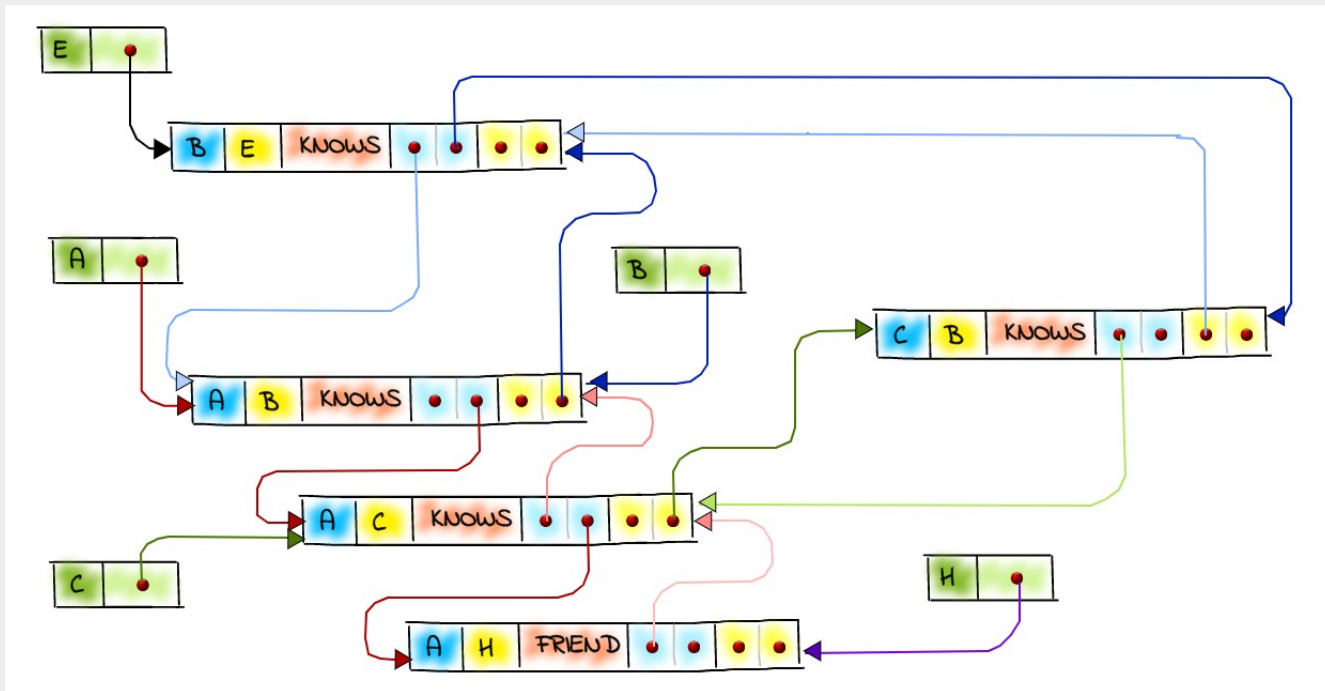
- PostgreSQL Expertise
- Technical Assistance & Support
- Training Center
- PostgreSQL R&D

Graphs are everywhere!



What You're Sold

Sounds specialized. Is it any better/different ?



The empire (NoSQL) strikes back ?

Worse than that, each relational JOIN operation is a search that takes twice as long as the database doubles in size, meaning it results in $O(\log(n))$ binary search operation. The more JOINS there are, the slower the search is. Relational databases excel at queries on small datasets with a uniform flat structure.

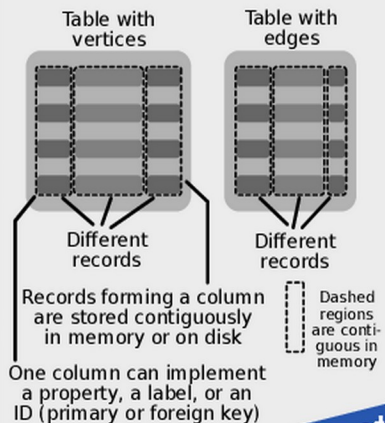


Achieve up to 1000x faster queries versus relational and non-native graph databases. Neo4j's native graph architecture eliminates complex joins and optimizes execution for rapid results.

You can do it too!

Column RDBMS

Vertices and edges are stored in rows of two column-oriented tables

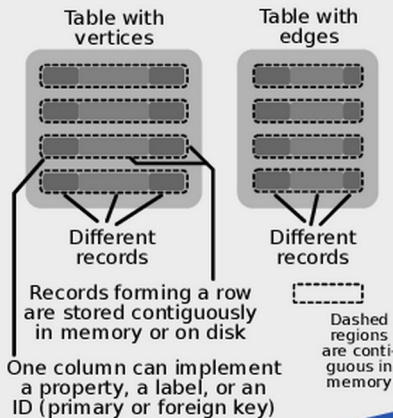


Example: SAP HANA

Model used: tables (implementing relations)

Row RDBMS

Vertices and edges are stored in rows of two row-oriented tables

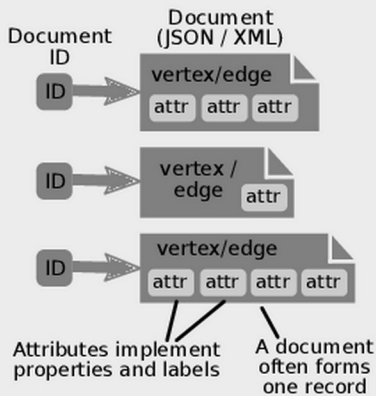


Example: Oracle Spatial and Graph

Model used: tables (implementing relations)

Document Store

Vertices and edges are encoded in documents (e.g. JSON) and linked via pointers or document IDs

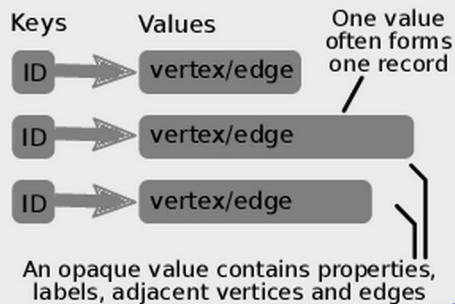


Examples: OrientDB, ArangoDB, Azure Cosmos DB, FaunaDB

Model used: documents

Key-Value Store

Vertices and edges are encoded in values and indexed by keys (IDs)

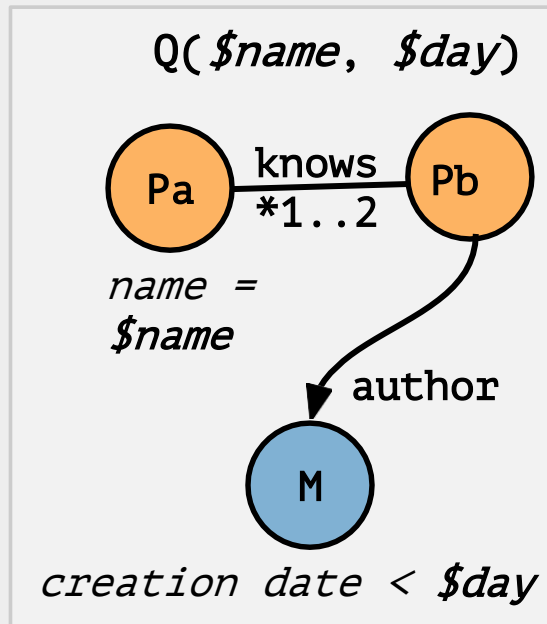


Examples: Dgraph, HyperGraphDB, MS Graph Engine

Model used: pairs of keys and values



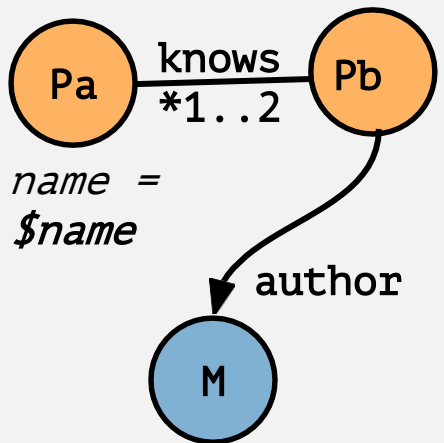
SQL/PGQ (ISO/IEC 9075-16:2023) : A new hope



```
person(id, name)
knows(p1id, p2id)
message(id, authorId,
        creationDate)
```

SQL/PGQ (ISO/IEC 9075-16:2023) : A new hope

$Q(\$name, \$day)$



$creation\ date < \$day$

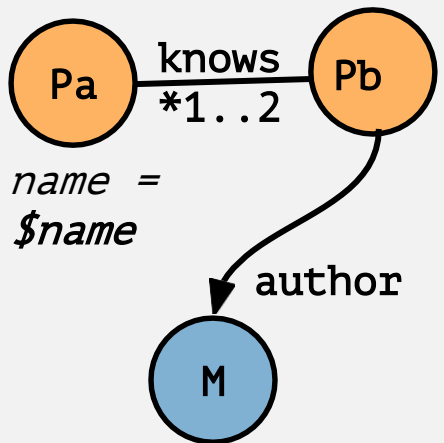
```
person(id, name)
knows(p1id, p2id)
message(id, authorId,
        creationDate)
```

SQL:1992

```
SELECT DISTINCT m.id
FROM (
  SELECT k.p2id
  FROM person Pa, knows k
  WHERE Pa.name = $name
  AND ...
  UNION
  SELECT k2.p2id AS id
  FROM person Pa, knows k1,
  knows k2
  WHERE Pa.name = $name
  AND ...
) Pb, Message m
WHERE m.creationDate <
      $day AND ...
```

SQL/PGQ (ISO/IEC 9075-16:2023) : A new hope

$Q(\$name, \$day)$



$creation\ date < \$day$

```
person(id, name)
knows(p1id, p2id)
message(id, authorId,
        creationDate)
```

SQL:1992

```
SELECT DISTINCT m.id
FROM (
  SELECT k.p2id
  FROM person Pa, knows k
  WHERE Pa.name = $name
    AND ...
  UNION
  SELECT k2.p2id AS id
  FROM person Pa, knows k1,
  knows k2
  WHERE Pa.name = $name
    AND ...
) Pb, Message m
WHERE m.creationDate <
      $day AND ...
```

SQL/PGQ

```
SELECT id
FROM GRAPH_TABLE (socialNetwork
  MATCH ANY ACYCLIC

  (Pa IS Person WHERE Pa.name = $name)
  -[k IS knows]-{1,2}(Pb IS Person)
  -[a IS author]->(m IS Message)

WHERE m.creationDate < $day
COLUMNS (m.id))
```

SQL/PGQ (ISO/IEC 9075-16:2023) : A new hope

Reverted.

```
Commit b1f106c
petere committed 3 weeks ago · ✓ 10 / 11

Revert SQL Property Graph Queries (SQL/PGQ)
List of commits reverted:

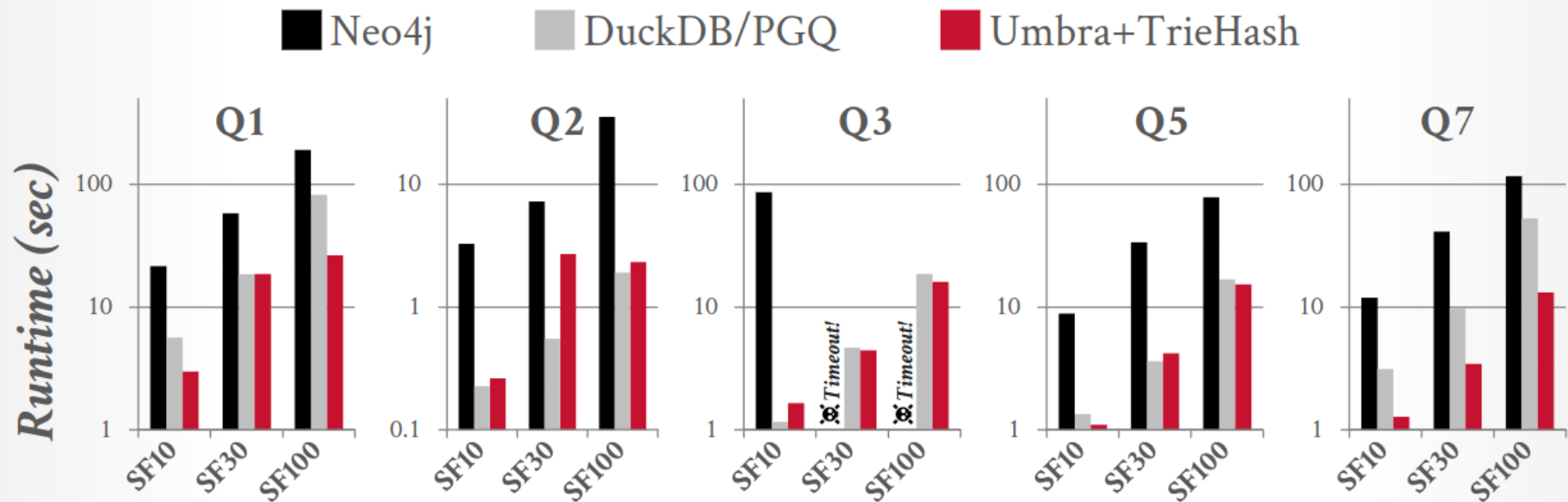
2f094e7 SQL Property Graph Queries (SQL/PGQ)
cd8844e Make some tests more stable by adding more explicit ordering
182cdf5 pg_dump: Add appropriate version check
c9babbc Dump labels in reproducible order
040a56b Cleanup users and roles in graph_table_rls test
```

The future:

- Getting it back ...
- Variable length edge (paths of any length)
- Worst Case Optimal Joins
- Resurrect parser hook discussion!

and more ...

Relational FTW !



Labeled Subgraph Query Benchmark (LSQB)

Source: [DuckPGQ: Efficient Property Graph Queries in an analytical RDBMS, CIDR 2023](#)



Hop 2: Distributed ?

Why?

- Graphs became too big in different domains.
- The need for high-throughput OLTP.
- The need for very large scale OLAP work.

Two main goals:

- Fault Tolerance & High Throughput
- Large scale analytics

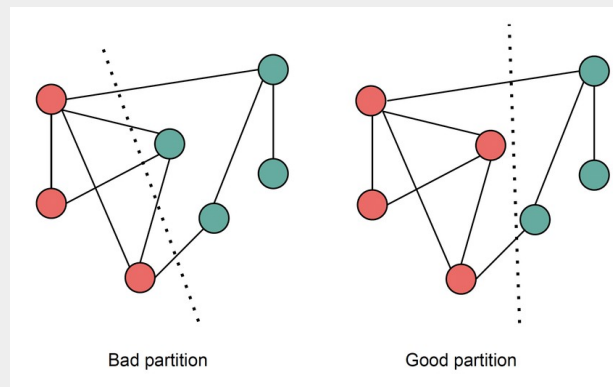
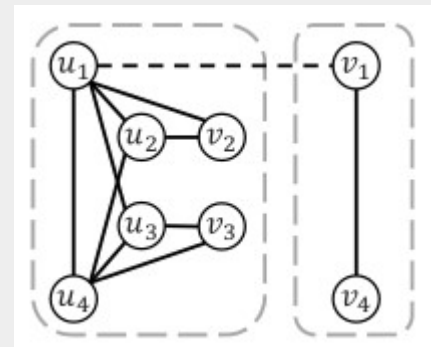
Thankfully, lots of work was done before for non-graph distributed databases to learn from.

Sadly, distributed graph databases have some special difficulties.

Hop 2: Distributed ?

Issues

- Parallelism: Graph algorithms are not easily parallelizable.
- Load Balancing: e.g: Cristiano Ronaldo account Node followed by 679M account nodes. **NP-Hard problem**.
- Communication overhead (cross network)

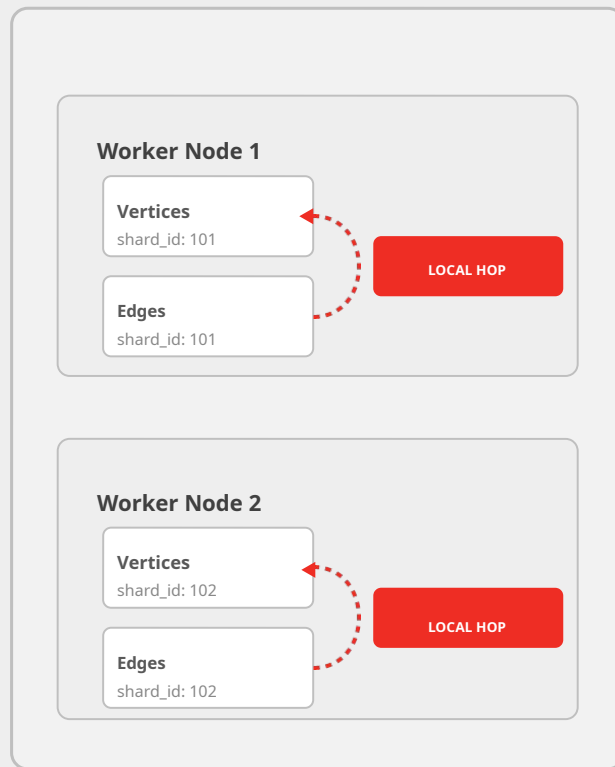


Hop 2: Citus to save us

Citus

- Open-source extension
- Sharding
- Distributed query engine
- Query from any node
- **Table Colocation**

The goal: Distributed Graph Database on PostgreSQL

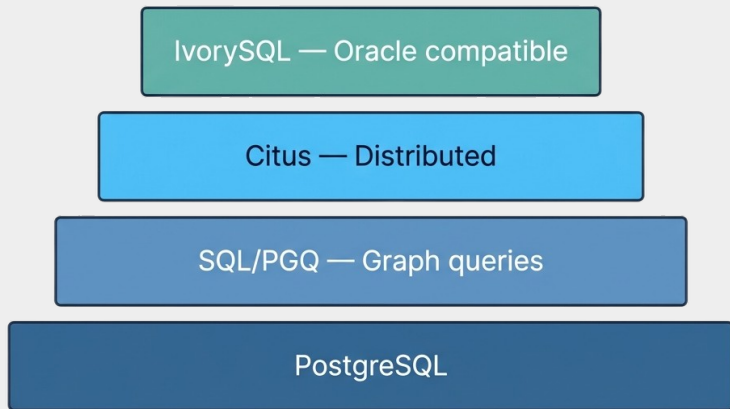


Hop 3: IvorySQL

- Oracle-compatible PL/iSQL
- 21 new Oracle features in v5.0
- Built on PostgreSQL 18
- Drop-in replacement for PostgreSQL
- Real migration path

The future:

- Oracle SQL/PGQ in IvorySQL



Verdict: Build on the foundations and don't ignore your predecessors' work.

Thank you for listening



Blog: PostgreSQL as a Graph Database

Ayoub Kazar - **Data Bene**
PGConf.EU 2026

THANKS FOR LISTENING!



**PostgreSQL Conference
Europe 2026**

Valencia

Oct 20th - 23rd



Hervé Schweitzer

OMrun the Data Quality Platform That Keeps
Your Data Trustworthy !

OMrun New Positioning

The **Data Quality Platform** that keeps your data trustworthy

- Catch data issues before they become business problems
- Continuously tests, monitor and validate your data continuously



Data Testing



**Data
Observability
& Consistency**



**Data
Validation**



OMrun New Positioning

Data Testing

Automatically test data throughout its lifecycle

- Simulate application activity and validate outputs
- Detect regression failures before release

Result

- Fewer incidents
- Safer & faster releases

Our use cases

ERP

Version upgrade regression testing

Compare the outputs before and after the upgrade

TAX SYSTEM

Code migration

Run the old and the new code, then compare the results

Data Observability & Consistency

Ensure analytics are consistent with source systems

- Monitor data freshness, volumes, trends and anomalies
- Monitor interfaces (APIs) in heterogeneous environments
- Detect drift, missing data and abnormal behaviour

Result

- Build trust in dashboards and KPIs
- Prove completeness, correctness, consistency

Our use cases

AUTO LEASING

Integration of new car models

Check the car model data that comes in over the interface

TAX SYSTEM

Billing consistency check

Find double or forgotten bills

OMrun New Positioning

Data Validation

Turn data validation into a controlled process

- Generate comparison scripts automatically
- Accept expected transformations, detect unintended changes
- Compare 100% of the source vs target data automatically

Result

- Provide objective proof before go-live

Our use cases

DATABASE MIGRATION

Oracle to PostgreSQL or MariaDB

Our reference projects

ANY DATABASE

Any RDBMS to any RDBMS

The same approach for every source and target

OMrun New Positioning

NEW - AI Integration



Database-aware context

Uses schema metadata to guide AI suggestions

- Reuses the active OMrun database connection
- Reads metadata only (tables, views, columns, and data types)
- SQL Server · Oracle · PostgreSQL · ODBC

Security

DB permissions come from the configured connection.
OMrun does not elevate privileges



From intent to full comparison

Build compare logic with no SQL expertise required

- Describe what should be compared in plain language
- AI drafts Query A / B using the active database context
- Supports joins and more complex comparison logic
- Generate the complete Business Mapping
- Compare automatically

AI-assisted
workflow

Describe



Generate SQL



Review & Apply



Generate Mapping



Compare

THANKS FOR LISTENING!



**PostgreSQL Conference
Europe 2026**

Valencia

Oct 20th - 23rd

WELCOME TO
VALENCIA



David Roberts

Distributed SQL as the storage layer for agentic,
multi-tenant Postgres

Introducing YugabyteDB AMP

YugabyteDB

Open source distributed SQL built for mission-critical apps



Single SQL database replicated
across nodes, regions or clouds for
massive scale and ultra resilience



Private, public, hybrid or multi cloud



PostgreSQL-compatible for familiar,
flexible development



Geo-placement for local access to
global data and data sovereignty

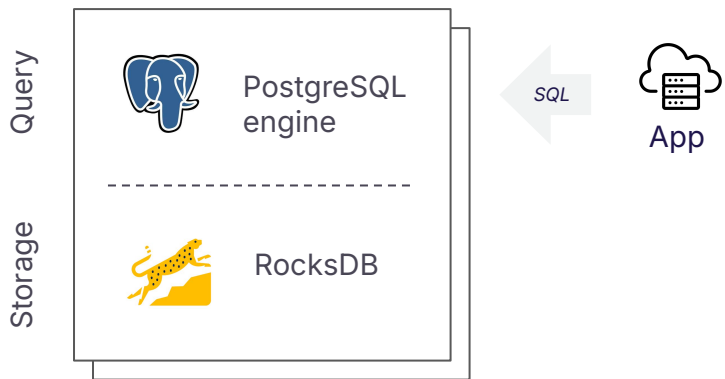


How YugabyteDB works



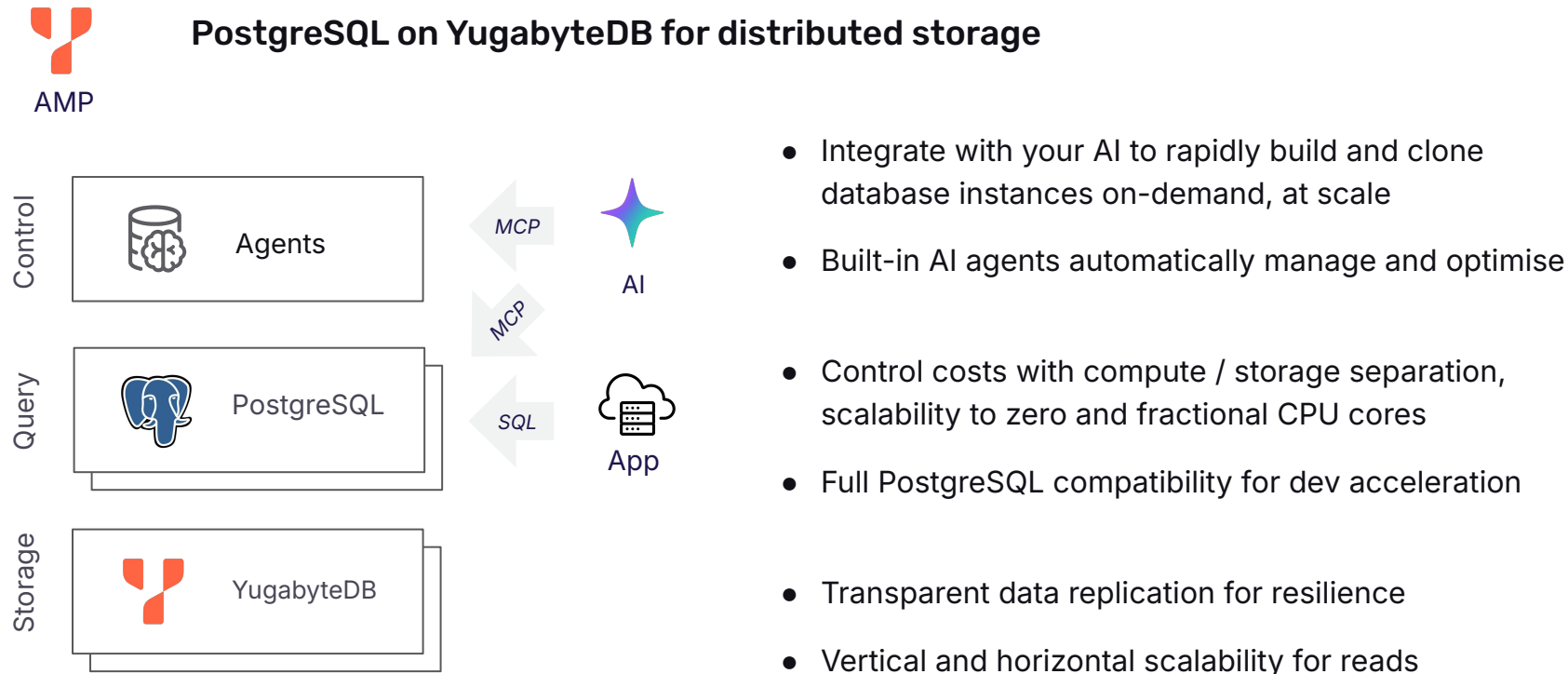
YugabyteDB

PostgreSQL for query layer on RocksDB for distributed storage



- Transactional, ACID-compliant and strongly consistent for simple application development
- PostgreSQL compatibility for dev velocity
- Transparent data replication for resilience
- Vertical and horizontal scalability for reads and writes

How YugabyteDB Agentic Multi-tenant Postgres works



THANKS FOR LISTENING!



**PostgreSQL Conference
Europe 2026**

Valencia

Oct 20th - 23rd
