Views & Materialized Views in PostgreSQL

CODEUS_

About me:

- Java Developer
- Work in Andersen Lab

Agenda

- What Are Database Views?
- What are Materialized Views?
- Comparison
- Decision Matrix
- Summary

Views

Database Views:

- Virtual tables based on the result of a SQL query
- Act as a layer of abstraction over the underlying tables
- Do not store data themselves

Database Views:

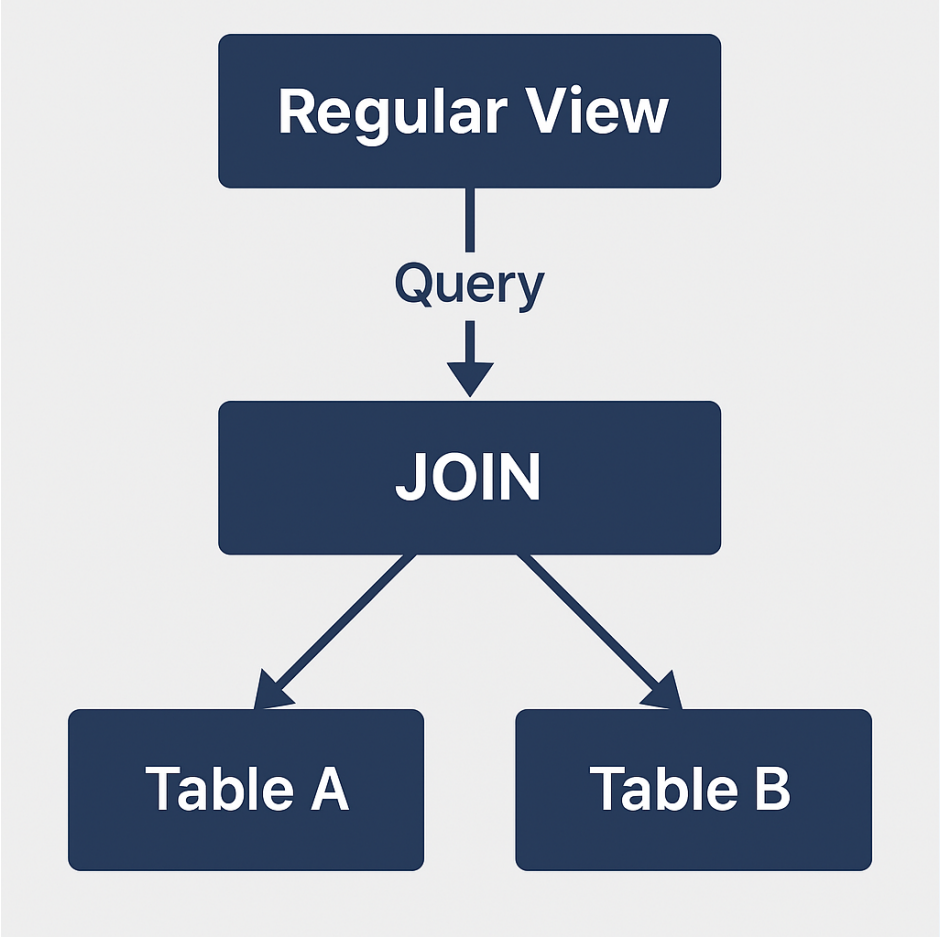
- Virtual tables based on the result of a SQL query
- Act as a layer of abstraction over the underlying tables
- Do not store data themselves

```
CREATE VIEW active_loans_view AS
SELECT
```

Key benefits:

- Simplify complex queries
- Provide a stable interface to data
- Can restrict access to sensitive data
- Enforce business rules consistently

How Regular Views Work



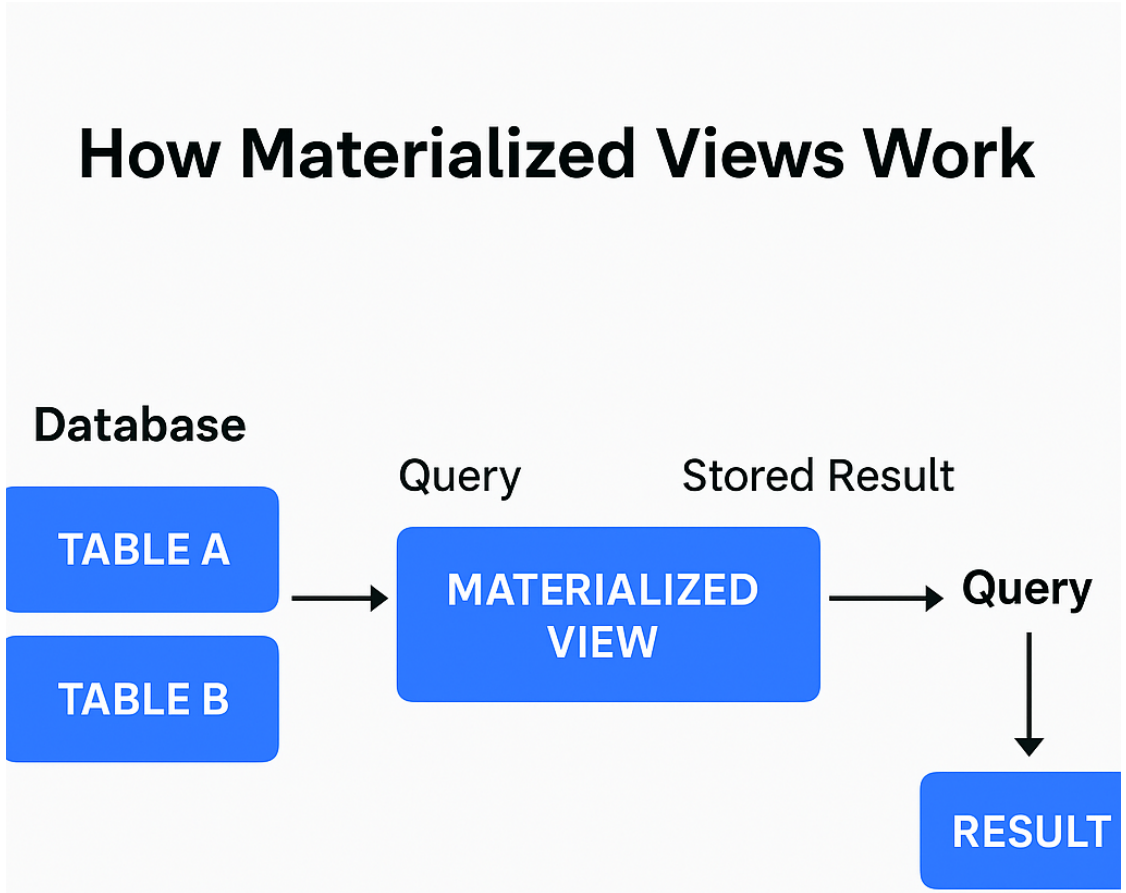
- User queries the view
- Database replaces view reference with the underlying query
- Database executes the combined query against base tables
- Results are returned to the user

Key Point:

Every query against a view re-executes the underlying query

Materialized Views

```
CREATE MATERIALIZED VIEW transaction_daily_summary AS
SELECT
```



- When created/refreshed:
  - Database executes the view query
  - Results are stored on disk
  - Indexes can be created on these results
- When queried:
  - Database reads directly from stored results
  - No re-execution of underlying query

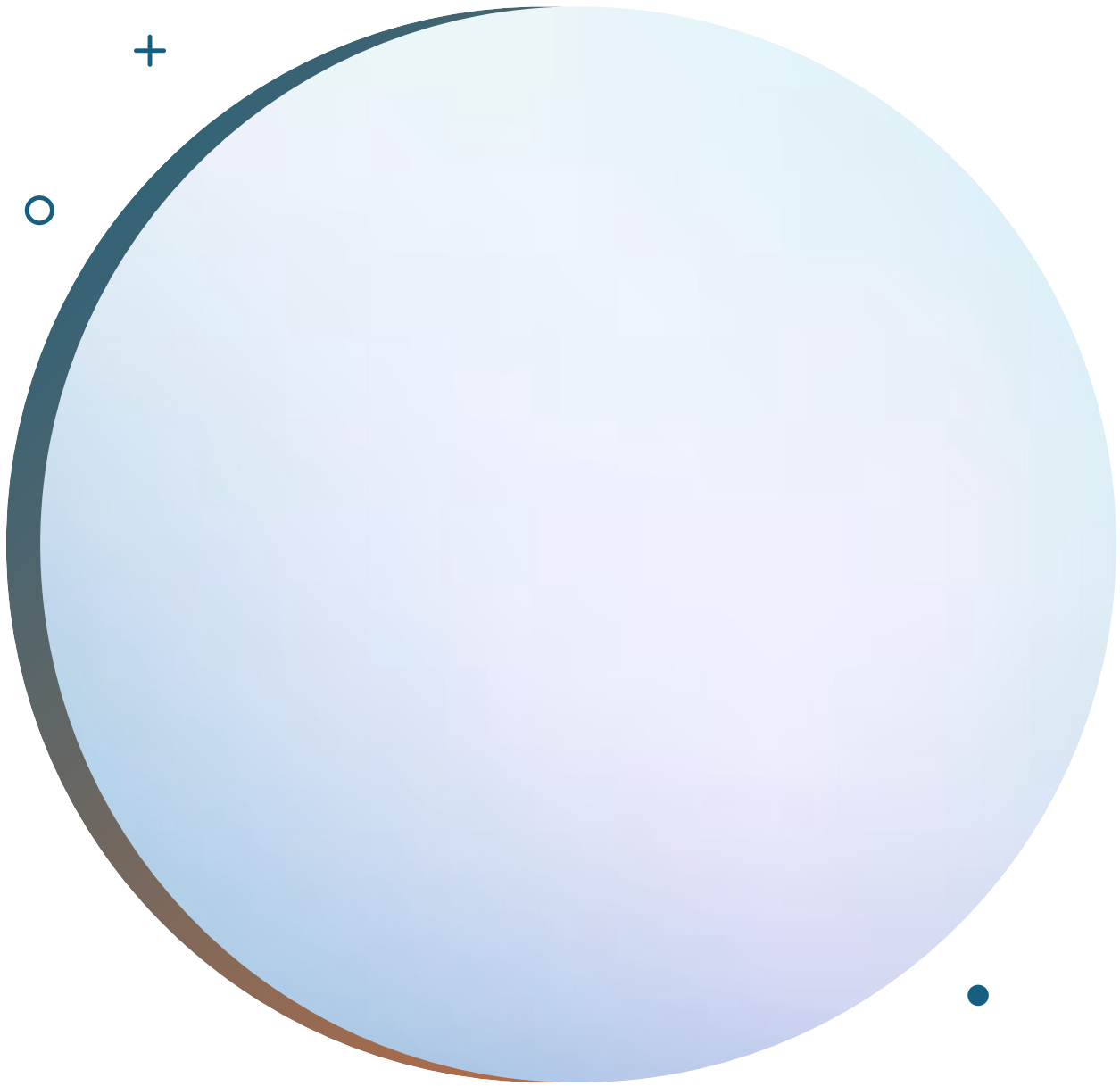
Key Point:

Query performance is dramatically improved, at the cost of data freshness

CODE

Regular Views vs. Materialized Views

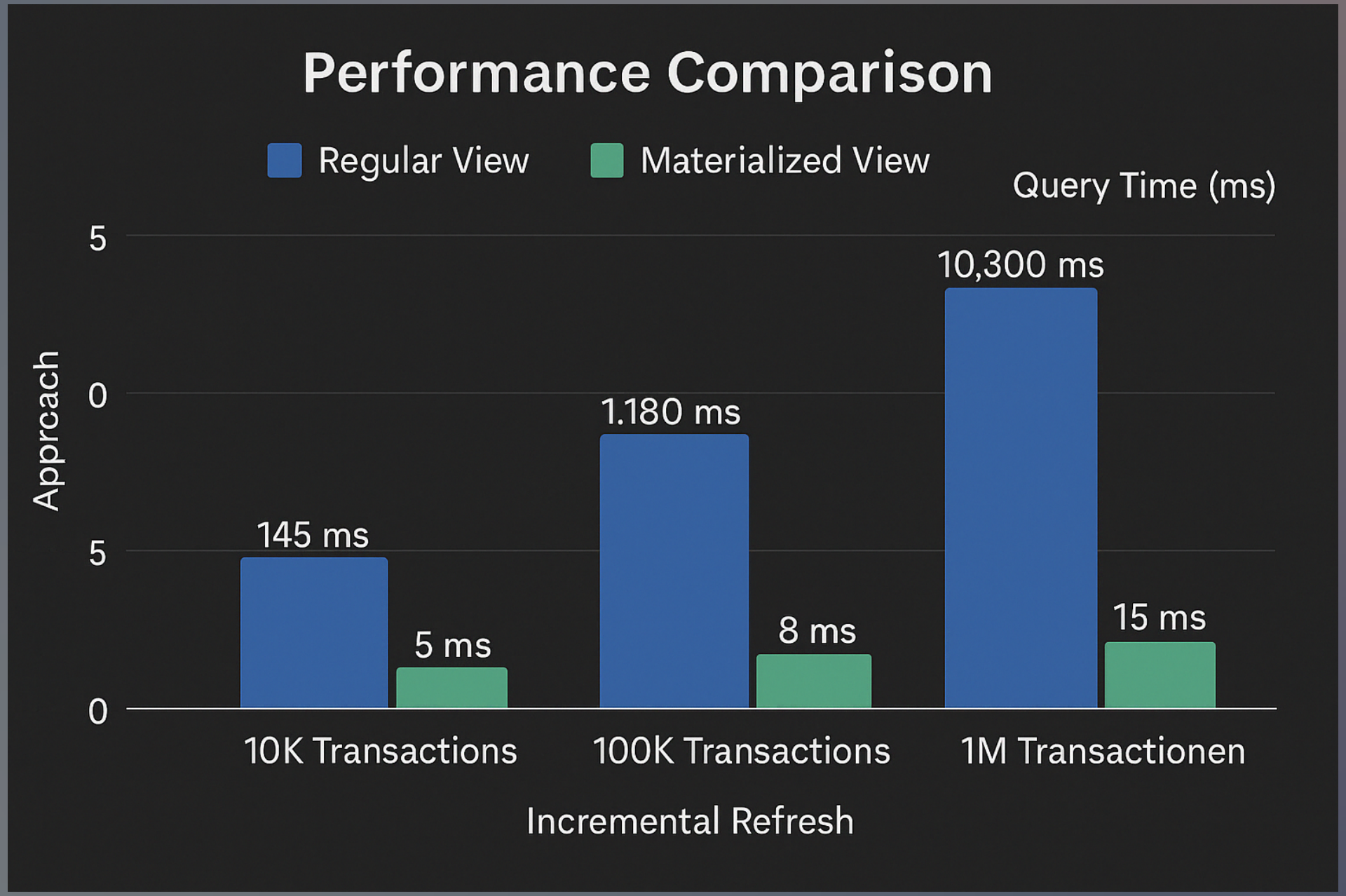
Aspect	Regular Views	Materialized Views
Storage	Virtual (no storage)	Physical (stored on disk)
Data Freshness	Real-time	Point-in-time snapshot
Query Performance	Computed each time	Pre-computed
Maintenance	None required	Refresh operations needed
Use Case	Up-to-date data access	Performance-critical reporting



Title

- Some text

CODEUS_


Performance Comparison
Regular View
Materialized View
Query Time (ms)
Apprcach
5
0
5
0
145 ms
5 ms
1.180 ms
8 ms
10,300 ms
15 ms
10K Transactions
100K Transactions
1M Transactionen
Incremental Refresh

CODEUS

Factor	Regular View	Materialized View
Query Complexity	Simple	Complex
Data Freshness	Critical	Flexible
Query Frequency	Low	High
Source Data Change Rate	High	Low
Resource Constraint	CPU	Disk

Thank you

- Author: Dmytro Shpak
- My LinkedIn: https://www.linkedin.com/in/shpak-dmytro/
- Date: April 2025
- Join Codeus community in Discord
- Join Codeus community in LinkedIn