Azure Cosmos DB for PostgreSQL
Distributed SQL service built on open-source Postgres & Citus

Marco Slot
Principal Software Engineer at Microsoft
Azure Cosmos DB for PostgreSQL

Distributed PostgreSQL for modern cloud-native applications

True PostgreSQL
Not a fork
Latest PostgreSQL version within 1 week
50+ PostgreSQL extensions

Scale out from 1 to 1000s of cores
Distributed query execution
Online scaling with zero downtime
Local and Global replication of data

Built for relational workloads
Transactions
Primary/Foreign Keys, Joins, and Constraints
Custom types, stored procedures

Cosmos DB managed service platform
Global Distribution – Seamless Elasticity – High Availability – Point in time recovery – Azure integrations

Available Open Source as the Citus extension to PostgreSQL
Citus: Distributed PostgreSQL as an Extension

Citus is a PostgreSQL extension that uses planner, executor, and utility command hooks to transparently distribute and replicate PostgreSQL tables across a shared-nothing PostgreSQL cluster.

Fully open source:  
https://github.com/citusdata/citus

SIGMOD ’21:  
“Citus: Distributed PostgreSQL for Data-Intensive Applications”
## Citus Features & Gaps

<table>
<thead>
<tr>
<th>Most PostgreSQL features just work on Citus tables</th>
<th>Distributed database superpowers with PostgreSQL-level efficiency</th>
<th>Some gaps remain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joins</td>
<td>Distributed &amp; reference tables</td>
<td>Schema-level sharding</td>
</tr>
<tr>
<td>Transaction blocks</td>
<td>Co-location</td>
<td>DDL from any node</td>
</tr>
<tr>
<td>Subqueries &amp; CTEs</td>
<td>Scale OLTP throughput</td>
<td>Automatic shard splits</td>
</tr>
<tr>
<td>Sequences</td>
<td>Fast co-located joins, foreign keys, ..</td>
<td>Non-co-located foreign keys, triggers</td>
</tr>
<tr>
<td>Expression indexes</td>
<td>Parallel, distributed queries</td>
<td>Unique constraints on non-dist. column</td>
</tr>
<tr>
<td>Partial indexes</td>
<td>Transactional ETL (INSERT..SELECT)</td>
<td>Cross-node snapshot isolation</td>
</tr>
<tr>
<td>Custom types</td>
<td>Fast data loading (COPY)</td>
<td>Geo-partitioning</td>
</tr>
<tr>
<td>Prepared statements</td>
<td>Online rebalancing</td>
<td>Database-level sharding</td>
</tr>
<tr>
<td>Stored procedures</td>
<td>Stored procedure call routing</td>
<td>Non-co-located correlated subqueries</td>
</tr>
<tr>
<td>Time-partitioning</td>
<td>Columnar compression</td>
<td>Vectorized execution</td>
</tr>
</tbody>
</table>

...
Common workload patterns

Multi-tenant OLTP (e.g. Software-as-a-service)

High throughput CRUD (e.g. IoT)

Real-time analytics (e.g. customer dashboards)
Hybrid local-distributed databases
Lessons learned in 10 years of Citus development

Relational database workloads are highly latency-sensitive due to the need to evaluate relationships, interactive protocols, ORMs
→ Pack related data together using co-location, reference tables, local tables, ...

At scale, efficiency is too important to trade efficiency for scale
→ Lean on existing RDBMS functionality to inherit price-performance characteristics

Distributed PostgreSQL only makes sense for specific workload patterns
→ Target multi-tenant (SaaS), real-time analytics (IoT, time series), CRUD, or hybrid.

...
Lessons learned in 10 years of Citus development

**Scalability is not (just) about transaction throughput**
→ Real workloads are complex. Infrequent $O(N)$ operations often dominate at scale.

**PostgreSQL development never stops**
→ Contribute to PostgreSQL, build extensions, do not fork

**Developing a complex mission-critical distributed database in which all features are related is hard**
→ Do small, independently useful projects with long-term goals in mind
Thank you!

We will at some point be hiring again 😊
(in Amsterdam, Istanbul, Redmond, San Francisco, or remote)

marco.slot@microsoft.com

Cosmos DB for PostgreSQL: https://aka.ms/AzureCosmosDBPGBlog
Citus on GitHub: https://github.com/citusdata/citus