Microsoft

Hyperscale on Azure Database for PostgreSQL

Build applications that scale simply from one tenant to 100,000s

Azure Database for PostgreSQL is fully-managed, community PostgreSQL in the Azure cloud. Focus on app innovation, not database administration, with valuable Postgres features, intelligent performance recommendations, and built-in security.

Build massively scalable applications with Hyperscale

Azure Database for PostgreSQL - Hyperscale scales out data across multiple physical nodes to give applications more memory, compute, and disk storage by automatically sharding underlying data.

- Ensure performance and flexibility with unmatched scalability
- Query events in real time with sub-second responses
- Run everything in one database to save resources
- Stay current thanks to compatibility with the latest versions and tools

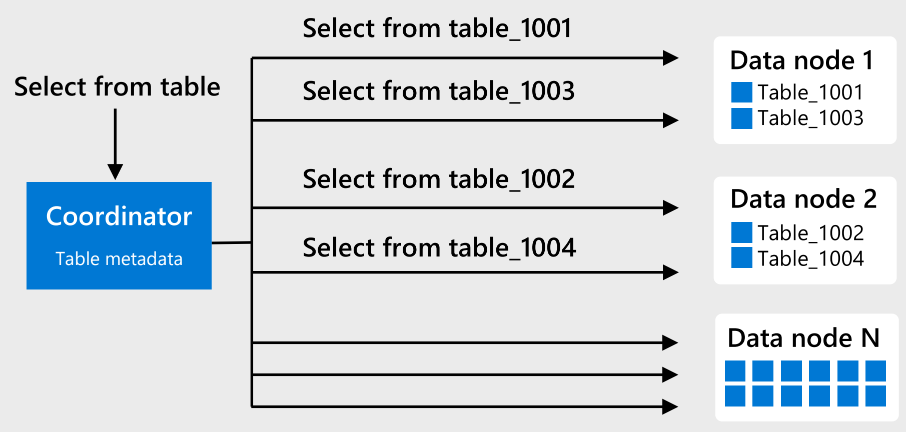
How Hyperscale works

All queries to a database cluster are managed by a coordinator node, which looks just like single-node Postgres to the application.

The coordinator is equipped with a distributed engine that transforms SQL queries and routes them to the correct shards on worker nodes.

Deployments include one coordinator node and at least two worker nodes. Scale horizontally by adding worker nodes.

Sharding data across multiple nodes



Two ways to scale

Hyperscale provides self-service scaling by adding new worker nodes or increasing the memory and CPU of existing nodes in one simple user interface. Choosing the right approach is dependent on your specifics needs.

Scaling up – increasing node size

If there's a lot of headroom on disk but performance is suffering, scale node RAM and processing power. Resizing nodes works by creating a PostgreSQL follower for each node with the desired amount of RAM and CPU vCores.

RAM / vCore per node

32 GB RAM 4 vCores	**64 GB** RAM 8 vCores	**128 GB** RAM 16 vCores	**256 GB** RAM 32 vCores	**432 GB** RAM 64 vCores

Scaling out – adding nodes

If the cluster is reaching its disk limit, then adding more nodes is the best choice. Once new nodes are created, you must manually move shards from existing nodes to new ones – called rebalancing.

Total Nodes
2

Key use cases for Hyperscale

Multi-tenant & SaaS apps
Support fast-growing SaaS apps and isolate large tenants on their own hardware

Real-time operational analytics
Ingest billions of transactions and enable sub-second query responses

High throughput transactional apps
Distribute transactions across multiple nodes and avoid single points of failure

Some workloads do not need a powerful distributed database. Single node PostgreSQL is appropriate in scenarios where there are few concurrent users, when analytics are offline, and when scalability isn't required.

Tips and tricks

Handling worker node failure
Hyperscale supports two modes of replication – streaming replication on the entire worker node and specific shard replication

Cluster node failover
Regular PostgreSQL synchronous replication and failover can be used for higher availability and resilience

Joining distributed and non-distributed tables
To do joins between small dimension tables and large tables, wrap the local table in a subquery

Ingesting a query into a distributed table
Hyperscale supports INSERT/SELECT syntax for copying the results of a query into a distributed, co-located table

Get started

View the Quickstart guide to learn how to create an Azure Database for PostgreSQL – Hyperscale server group using the Azure Portal.

If you don't have an Azure subscription, create a free account before you begin

Learn more

Explore tutorials and other key information in the Azure Database for PostgreSQL documentation page

Hyperscale your PostgreSQL database today!