



houdata.

PRODUCT DOCUMENTATION

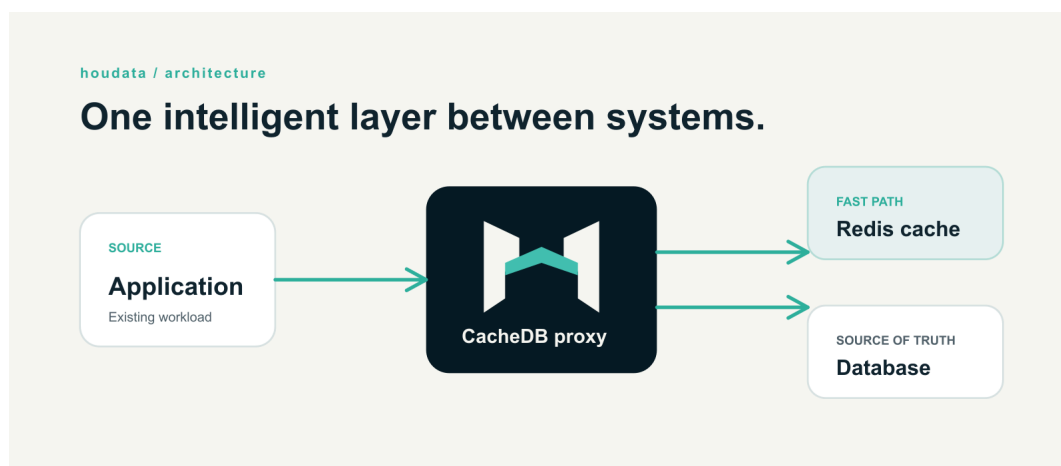
CacheDB Product Brief

Database performance and control

CacheDB is Houda's database performance and control product. It sits between an application and its database, observes SQL traffic, and can serve eligible reads from a shared cache. Teams define how cached data responds to writes, then use operational evidence to evaluate the result.

CacheDB is designed for critical workloads where rewriting application logic is expensive or risky. Applications keep their existing database driver and SQL and connect through the matching transparent CacheDB proxy.

Where CacheDB fits



An application connects through the transparent CacheDB proxy, which serves eligible reads from cache or forwards requests to the database.

The proxy sits between the application and database. It speaks the database's native protocol to the application and preserves that protocol on the database side. Eligible reads can use the cache; other requests continue to the database.

Database	Protocol spoken by the proxy	CacheDB proxy
PostgreSQL	PostgreSQL	PostgreSQL proxy
MySQL	MySQL	MySQL proxy
Microsoft SQL Server	TDS	SQL Server proxy
IBM Db2	DRDA	Db2 proxy
Oracle Database	TNS/TTC (Oracle Net)	Oracle proxy

Each proxy transparently relays its database's native protocol between application and database. Applications retain their database driver and SQL; CacheDB does not translate between SQL dialects. Compatibility depends on the client, database release, and configuration, so customer deployments should be qualified against the relevant compatibility guide.

Policy-driven caching

CacheDB focuses on eligible reads and policy-driven invalidation. Teams can begin in transparent mode, review workload behavior, then enable cache deliberately. Reads that are unsuitable for caching continue to the database. The configured policy determines how writes affect cached results.

Tune and clean the cache with scripts

Rhai scripts let teams tailor cache behavior to the workload. They can assign different time-to-live (TTL) values to different query patterns, bypass caching for selected reads, and choose how writes clean up cached results. Invalidation can range from clearing results associated with a table to removing only results that match a query or data conditions. This helps preserve unaffected entries while keeping related results current.

TTL policies apply as new results are stored; existing entries keep their current lifetime until they expire or are invalidated. Cache cleanup follows the configured script policy rather than a universal automatic rule.

The operations console brings together query activity, cache behavior, latency, and invalidation information. CacheDB supports Redis Community, Redis Enterprise, and Valkey.

The proxy can secure connections to the application and database using TLS. Deployment configuration depends on the selected database and client. CacheDB is designed for controlled operation in the customer's environment; workload performance and cache behavior should be evaluated with representative traffic.

A practical evaluation

Compatibility and results depend on the database, driver, workload, and chosen policy. A proof of concept with the customer's own application and database is the best way to confirm fit, observe cache opportunities, and establish expected outcomes.