SOLUTION BRIEF

Modernize Your MySQL Database With Redis Enterprise

Enable real-time performance at scale
on MySQL with Redis Enterprise

redis

Redis Enterprise brings the performance of your MySQL applications up to the standard required to power today's user expectations. As an in-memory real-time data platform, Redis Enterprise can be used alongside MySQL as an enterprise cache or database to make MySQL applications faster, more efficient, and more scalable.

Why MySQL with Redis Enterprise

MySQL is an open-source relational database designed around transactional data, not performance. It was also designed with rows and columns - not the way we access and visualize data as objects, documents, or time series. Using MySQL for real-time, modern applications that it was not designed for creates extreme database challenges.

Challenges with MySQL

- **Delivering rapid responses:** MySQL, fast for a relational database, at scale is saddled with a relatively high overhead and cannot deliver optimal speed. MySQL is unable to provide the real-time responses that are required for new applications that involve such things as session management, fraud detection, or real-time claims processing.
- **Struggling with high-velocity data:** When thousands of updates per second are written to a single database row (for example, flash online sales), it is crucial to maintain exact values every second. MySQL is designed around full transactional semantics with support for long transactions - transactions that will never be complete unless each operation within the group is successful. Long transactions are not suitable for high velocity data where the exact value needs to be constantly updated and instantly available. MySQL doesn't do well with data that is too big, moves too fast, or doesn't fit the structure of its architectures.
- **Scaling limitations:** MySQL was initially designed as a single-node system. Today's largest MySQL installations cannot scale by using MySQL as a single system and must rely on sharding or splitting a data set over multiple nodes or instances. However, most sharding solutions in MySQL are manual and make application code more complex. Any performance gain is lost when queries must access data across multiple shards.
- **Limited full-text searches at scale:** MySQL can handle basic full-text searches. However, because of its inability to manage parallel processing, searches do not scale well as data volumes increase. MySQL was not designed for running secondary indexed queries against massive data volumes, which requires crunching through data on a vast scale. A given MySQL query can neither scale among multiple CPU cores in a single system nor execute distributed queries across multiple nodes.
- **Limited real-time global distributions:** MySQL cannot distribute a unified dataset to power global access to your data and provide real-time responses. MySQL Cluster is strongly consistent, which does eventually provide up-to-date data, but the cost is that it has high latency.

Benefits of Redis Enterprise with MySQL

- **Enabling real-time responses:** Redis Enterprise as an enterprise cache is designed for sub-millisecond performance at scale and can guarantee high-performance responses. MySQL was built for transactions rather than speed at scale; Redis Enterprise provides the speed required for instant responses.
- **Ingesting highly velocity data:** To manage extreme data velocity and gain insights faster with MySQL, you need a data ingestion buffer, such as Redis Enterprise, to handle large concurrent writes to a relational database. Redis Enterprise offers a variety of data structures such as streams, lists, sets, sorted sets, and hashes that provide simple and versatile data processing to combine high-speed data ingestion and real-time analytics efficiently.
- **Easily scales:** Redis Enterprise automatically and linearly scales, optimizing the consumption of servers and DRAM. Unlike MySQL, with sharding or partitioning, Redis Enterprise is not restricted to storing data on a single computer's memory.
- **Secondary index searches at scale:** With its powerful search engine Redis Enterprise provides rapid secondary indexing against massive data volume across multiple nodes. MySQL, at scale, creates massive indexes that take forever to search and often deliver sub-optimal results.
- **Enables MySQL applications globally:** Active-Active Geo Distribution facilitates multiple Redis Enterprise clusters, distributed across geographies, to accept reads and writes simultaneously. The combination of real-time speed, distribution, and data consistency allows Redis Enterprise to easily distribute MySQL applications globally.

How to implement MySQL and Redis Enterprise

Redis Enterprise is regularly used as a cache with MySQL to enable sub-millisecond responses and reduce infrastructure costs. Redis Enterprise can also be used with MySQL to ensure all data can quickly be accessed by creating easily searchable secondary indexes that provide optimized results.

MySQL and Redis Enterprise: How they work together

There are many ways MySQL and Redis Enterprise can work together to enable real-time responses. The architecture and functionality choices will depend on the specific use case you are trying to improve. These include:

1. Secondary indexing with real-time search
2. Cache prefetching/Caching using the CQRS pattern
3. Write-behind caching
4. Active-Active Geo-Replication

Secondary indexing with real-time search

Performing queries on secondary indexes can be incredibly difficult and time-consuming in MySQL due to the table structure. Redis Enterprise is commonly used for secondary indexing to build relationships between records and perform data queries (beyond primary keys) in real-time while keeping your raw data in MySQL. With Redis Enterprise, you can query your data at lightning speed, perform complex aggregations, and filter by properties, numeric ranges, and geographical distance.

Secondary indexing

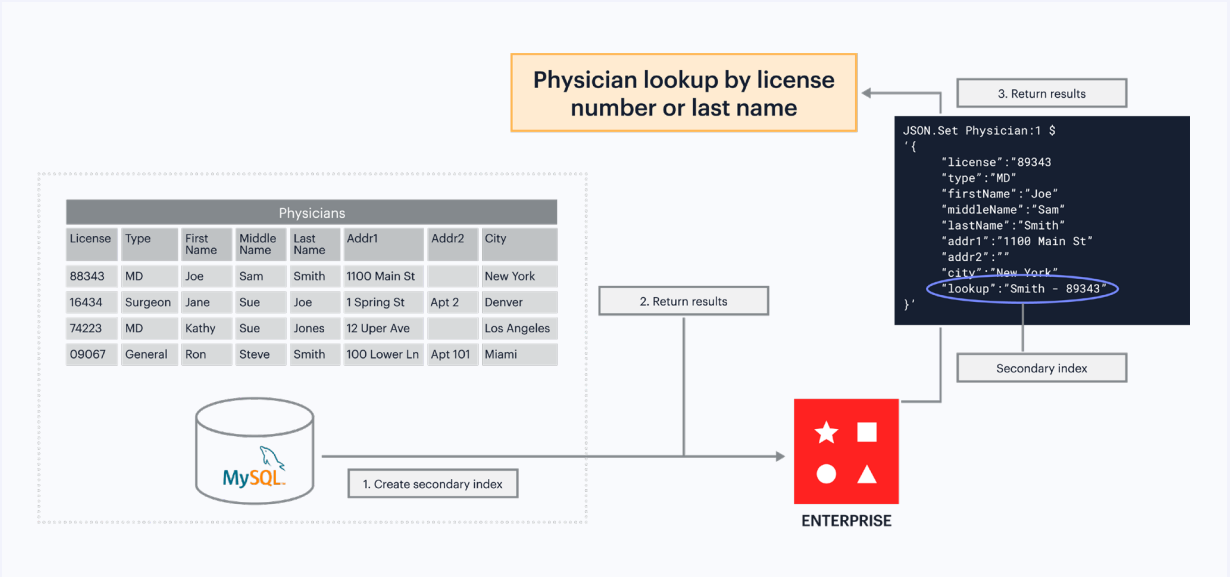
One of the biggest challenges with MySQL is the ability to perform quick data table look-ups, especially at scale. Redis Enterprise can be used to quickly generate secondary indexes to more easily query the critical data required. One of the key benefits of searching these secondary indexes on Redis Enterprise is that the results are provided in real-time.

Customer case study: Faster, richer query results with secondary indexing enabled by Redis Enterprise

This Redis Enterprise customer is a healthcare provider that is part of an extensive public health system. They have developed a homecare solution that manages 4M documents per year, 10K users per day, 600K users per year, and over 200 hospitals. The site contains, among other things, over $2B worth of billing information, lab reports, immunization data, and test results, so there is a lot of content.

The existing SQL database created extensive indexes, limiting how queries could be done, which were very slow. Secondary indexes via Redis Enterprise were the way they solved this problem. A secondary index is a data structure containing a subset of attributes from a table and an alternate key to support query operations. This is a way to efficiently access records in a database by using some piece of information other than the standard primary key.

As you can see from the architecture diagram below, the secondary index includes the physician's last name, which is how many people search for a doctor. Creating these secondary indexes is fast and easy, significantly reducing the burden on the primary SQL database. Using Redis Enterprise for secondary indexing and as a cache in conjunction with your SQL database not only improves the usability of the data by making queries much more straightforward but also enables real-time responses.



- The customer's MySQL database acts as the system of record, storing large quantities of data about physicians in tables, with the physician license number as the primary key for queries.
- Redis Enterprise hashes the data and is used to create secondary indexes on this data, building relationships between records and allowing for queries on data held in the table beyond the primary key (license number) and storing it in Redis Enterprise for real-time query performance.
- When users need to look up a physician by other attributes in the table, like physicians with the last name "Smith," they can use Redis Enterprise to perform a secondary key query.
- Redis Enterprise queries a vast amount of physician data held in secondary keys and returns accurate results associated with that record in real-time.

Customer benefits:

1. The customer was able to get more value out of the data they held in MySQL, adding the ability to query the wealth of secondary data stored in their MySQL tables, making the data more useful to power better customer experiences.
2. Hashing the data into Redis Enterprise also greatly improved query performance. Queries were returned by Redis Enterprise in real-time rather than waiting seconds or even minutes for MySQL's disk-based database to produce results.
3. Querying Redis Enterprise reduced the burden on the customer's MySQL database, reducing MySQL infrastructure costs.

Caching

Using Redis Enterprise as a cache is one of the most effective ways to speed up MySQL applications. Redis Enterprise as a cache can also reduce the queries that MySQL needs to handle. Reducing the burden on MySQL enables you to minimize the infrastructure requried reducing costs.

Cache prefetching

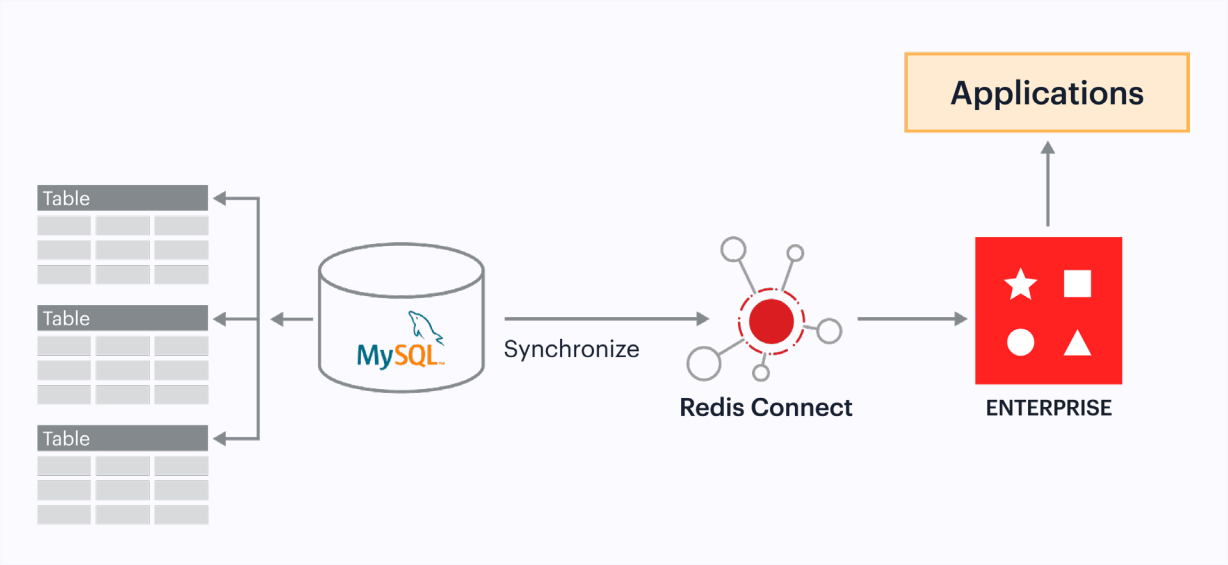
Cache prefetching is a technique used to boost performance. Data is read from its original storage in disk-based memory in MySQL. It is then written to a much faster in-memory database, Redis Enterprise before your application needs it. Accessing a cache is typically much faster than accessing main memory, so prefetching data and then accessing it from a cache is usually much faster than accessing it directly from main memory. This approach to offload reads to Redis Enterprise greatly enhances application speed and lowers the load on MySQL. Decreasing the load on MySQL will lower the required infrastructure, reducing the cost of running MySQL.

CQRS (Command Query Responsibility Segregation) is an application architecture pattern often used in cache prefetching solutions. CQRS, in essence, treats retrieving data and changing data differently. CQRS uses command handlers to simplify the query process and hide complex multi-system changes. It is a pattern that separates read and update operations for a data store. CQRS is a critical pattern within microservice architectures that decouple reads and writes. With MySQL as the system of record and Redis Enterprise as an in-memory cache read database, you can avoid slow queries and scale more easily.

Customer case study: Ensure customer's requirements are met with cache prefetching from MySQL to Redis Enterprise

This company is a global leader in customer engagement software. Its cloud-based business software suite is used by more than 150,000 organizations worldwide. They have seen upwards of 50% year-over-year growth for the past six years and topped $100 million in annual recurring revenue. This extraordinary growth, spurred by the rapid adoption of its products, was straining the capabilities of application architecture and development operations. As the company's database load grew, it faced trouble scaling performance. They needed to reduce the burden on its primary MySQL database.

They added Redis Enterprise as a write-behind cache to limit the number of queries going to MySQL from their customer when they want to access their customer engagement applications. Using Redis Enterprise as a cache significantly improves the response time of the customer profile and usage information and minimizes the impact and burden on their MySQL database.



- The customer's MySQL database is the source for all customer profiles and usage information.
- Redis Enterprise is used as a write-behind cache that maintains all customer interactions and queries in real-time, enhancing customer satisfaction.
- Redis Enterprise asynchronously writes any updates required to the MySQL database, limiting the burden on the database but keeping MySQL, the system of record, up to date.

Customer benefits:

1. The customer could more easily access the data they held in MySQL, ensuring customers received their information in real-time.
2. Caching the data into Redis Enterprise greatly improved query performance. Queries were returned by Redis Enterprise in real-time rather than waiting seconds or even minutes for MySQL's disk-based database to return results.
3. Querying Redis Enterprise reduced the burden on the customer's MySQL database, reducing MySQL usage and infrastructure costs.

Write-behind caching

Write behind cache is a caching strategy in which your application communicates directly with the cache, Redis Enterprise, and the data is later updated to MySQL. This means that your applications only need to link to your cache layer, and the cache then reads from or updates the back-end database as required.

Data is first written to Redis Enterprise and is then asynchronously updated in MySQL. This approach improves write performance and eases application development since the developer writes in only one place. Session Management is one of the use cases where write-behind is used.

Session management

Session management captures the current status of user interaction with applications such as a website or video games. A typical web application keeps a session for each connected user for as long as the user is logged in. Session state is how apps remember user identity, login credentials, personalization information, recent actions, shopping cart, and more.

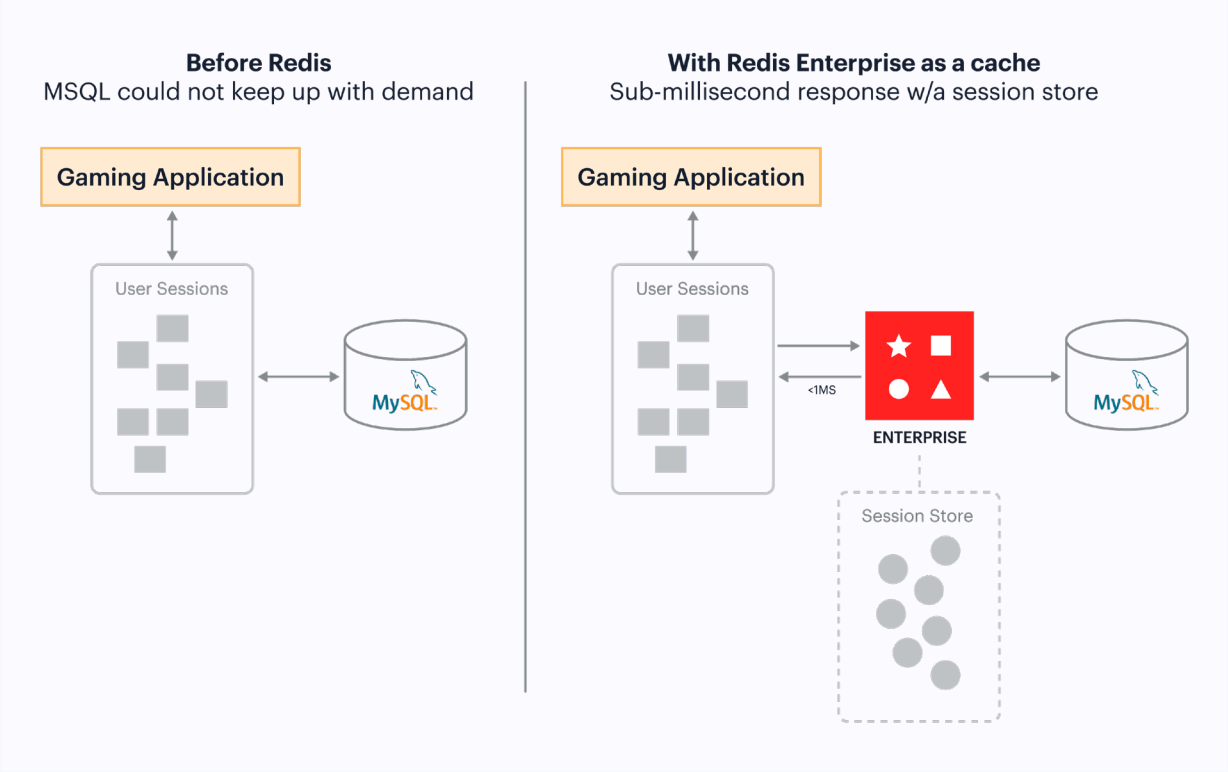
Reading and writing session data at each user interaction must be done without hurting the user experience. Behind the scenes, the session state is cached for a specific user or application, which allows a fast response to user actions. Therefore, while the user session is live, no round-trip to the central database should be needed.

In MySQL, storing session data, which requires frequent reads and writes, is very slow and inefficient. Because every user interaction involves access to the session's data, keeping that data in Redis Enterprise increases the response time to the application user.

Customer case study: Millions create their fantasy teams with Redis Enterprise as a session store

MyTeam11 is a fantasy sports gaming platform with unpredictable data loads for cricket, football (soccer), kabaddi, hockey, basketball, handball, volleyball, rugby, and baseball. It delivers 15 million+ MyTeam11 users with over 25,0000 operations per second. MyTeam11 needs to handle the peak data loads that hit in the 30 minutes leading up to a cricket match as users rush to the platform to set their fantasy rosters as soon as the starters are announced. MyTeam11's MySQL database was unable to handle the load.

MyTeam11 added Redis Enterprise as a cache in front of their MySQL database. Doing this provided sub-millisecond responses to all of their users, even during peak data loads.



- MyTeam11 player information is easily and quickly loaded into Redis Enterprise and always in sync. The MySQL database is the system of record.
- Users log in to set up fantasy teams resulting in a session store on Redis Enterprise, ensuring real-time interaction and customer satisfaction.
- All session management data is maintained in Redis Enterprise until the session ends, enabling millions of people to interact with the platform simultaneously.
- Once the session ends, the data required to be maintained is asynchronously written to MySQL, limiting the burden and infrastructure resources needed.

Customer benefits:

1. The customer could easily handle the volume of users that want to access the data.
2. Caching the data into Redis Enterprise and using a session store eliminated unnecessary queries and writes to the MySQL database, ensuring a real-time response.
3. Querying Redis Enterprise reduced the burden on the customer's MySQL database, reducing MySQL usage and infrastructure costs.

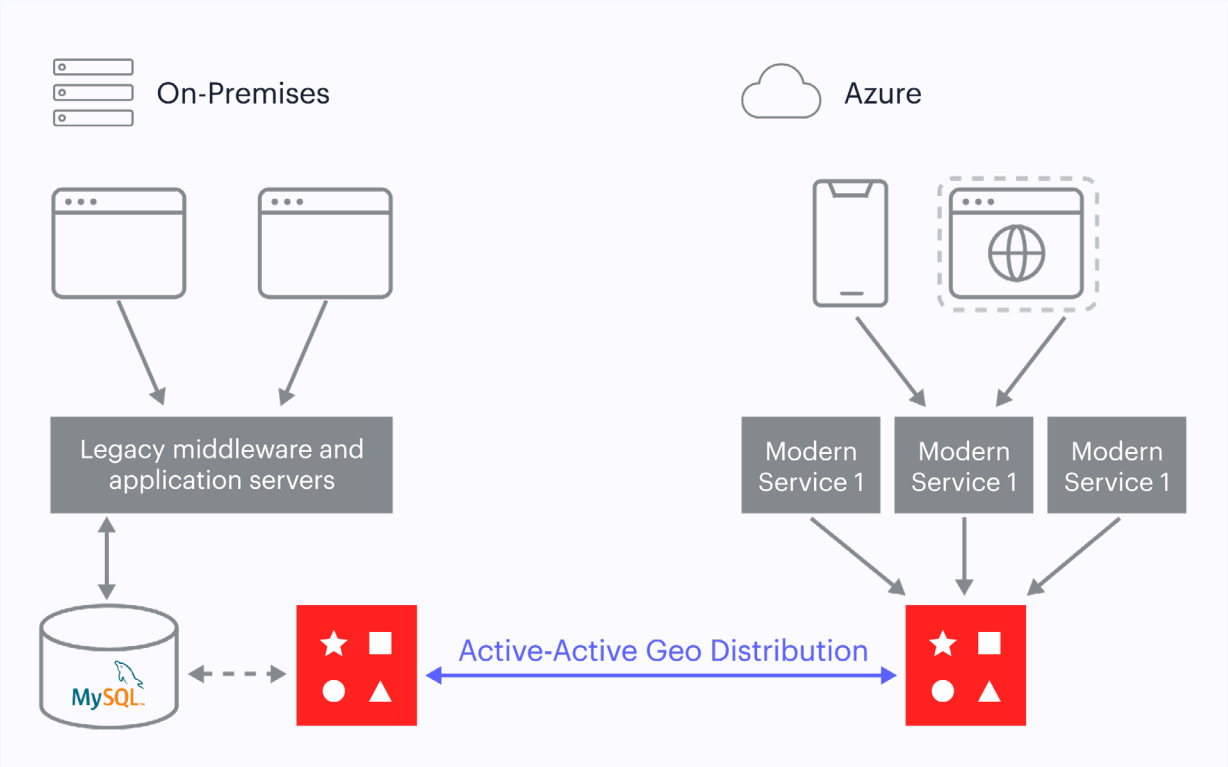
Active-Active Geo-Replication

Redis Enterprise enables the global distribution of MySQL applications with synchronized data. Redis Enterprise is used to move to hybrid/multicloud architectures or support application modernization with a cloud-agnostic data layer that unifies data across all your environments.

Customer case study: Easily create globally distributed applications with Redis Enterprise

A Redis Enterprise customer had an on-premises environment that hosted legacy applications, their middleware, and a MySQL database that stored their application data. They were phasing workloads and applications out of the on-premises environment and re-platforming into cloud-native microservices applications hosted in Azure.

Replicating data in MySQL into Redis Enterprise in their cloud environment was critical to operating efficiently in both environments while the customer globally distributed the application. It allowed data held in MySQL to be cached into Redis Enterprise and replicated into a cluster in their new cloud environment. Redis Enterprise's Active-Active Geo Distribution synchronized data between the on-premises and cloud environment in real-time, enabling reads and writes in both environments with data consistency.



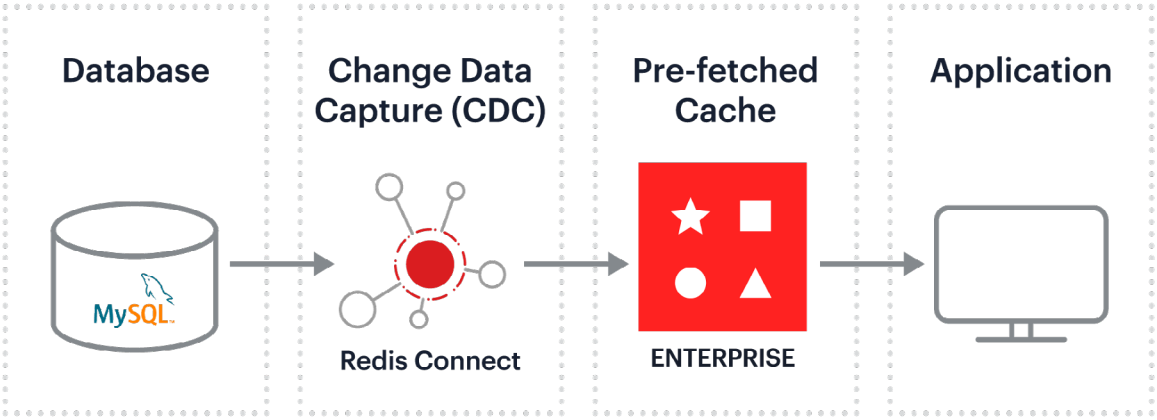
- The on-premises MySQL environment contains several critical applications that still serve customers and are the system of record.
- The Azure environment is the destination for workloads that have been moved to the cloud using a phased migration approach.
- Data is cached from the MySQL database into a local Redis Enterprise cluster hosted on-premises.
- A Redis Enterprise cluster is hosted in Azure to provide real-time data with local latency to the new cloud-based applications.
- Active-Active Geo Distribution synchronizes data between the on-premises and cloud environments, enabling real-time reads and writes in both environments with data consistency.

Customer benefits:

1. Using Redis Enterprise enabled the customer to globally distribute their applications stack without disrupting their application or data held in MySQL. Because of Active-Active Geo Distribution, applications hosted on-premises and new modern cloud applications were both able to access and process data without impacting the user experiences.
2. Using Redis Enterprise allowed for sub-millisecond latency data cached in-memory close to users for real-time user experiences regardless of the hosting environment.

Connecting MySQL and Redis Enterprise

Redis Connect makes it easy for you to use Redis Enterprise as a cache with MySQL. The capture, transform, and load process from MySQL into Redis Enterprise is easy and automated. Redis Connect propagates exact copies and performs data synchronization of your MySQL database data without disrupting ongoing operations.



Get the most out of your MySQL database

Use Redis Enterprise to get the most out of and extend the life of your MySQL database. Redis Enterprise with MySQL applications will enable sub-millisecond responses, increase customer satisfaction and reduce your infrastructure cost. Some of the main benefits of using Redis Enterprise alongside MySQL are:

1. **Redis Enterprise brings real-time speed to our real-time world:** Used in conjunction with MySQL to add sub-millisecond performance for real-time application experiences, not only providing better user experiences.
2. **Redis Enterprise saves you money by minimizing the infrastructure you need for MySQL:** Used to offload reads and writes from MySQL, allowing you to only use MySQL when you need to, which means less MySQL infrastructure while boosting performance.
3. **Redis Enterprise supports globally distributed applications:** Freeing your MySQL data to move to hybrid or multicloud architectures and globally distributed synchronized data, you can provide real-time responses to your customers regardless of location.

Want to learn more about MySQL and caching with Redis Enterprise?

Caching at Scale with Redis – The only primer you need to understand what application caching is, why and when it's needed, and how to get the best performance from your applications. **Download Now.**

Or start your journey toward faster, more powerful, more cost-efficient MySQL applications today. **Book a meeting with Redis and get started.**