SOLUTION BRIEF

Modernize Your Oracle Database With Redis Enterprise

Make Oracle faster and less expensive with Redis Enterprise

redis

You want your applications to run as fast as possible in order to provide the real-time user experiences expected by today's customers. Redis Enterprise is known for real-time speed and can greatly improve the performance of your Oracle applications. You can use Redis Enterprise alongside Oracle as an enterprise cache or database, as an in-memory real-time data platform, to make Oracle applications faster, more efficient, and more cost-effective.

Oracle challenges: slow, expensive, and complicated

Oracle originated nearly 50 years ago in an era that not only predated the cloud but during a time in which storage and RAM were incredibly expensive. Storage was the primary constraint at the time, so relational databases were built with disk-based storage as a design principle over performance or data access.

The consequence of this design was an impedance mismatch between the way we store data within relational databases as columns and rows and the way modern applications access and visualize it as objects, documents, as a time-series, etc. Most obviously, application performance was constrained by the physical limits of a spinning disk. In addition, the extra complexity required to overcome this impedance mismatch limits the latency and throughput performance that today's applications desperately need -- and customers expect.

Because of its aged design principles, Oracle customers often face challenges with:

- **Performance:** While relatively fast for a relational database, Oracle is still far too slow to power a slew of modern use-cases such as AI, session management, fraud detection, and real-time claims processing.
- **Cost:** Oracle is incredibly expensive, which makes cost among the most common challenges. Oracle licenses typically cost tens of thousands of dollars. And when more Oracle databases are used, more costly Oracle licenses are required.
- **Inflexibility:** Oracle only supports relational data. It cannot be used with multiple data models. Data in Oracle is subject to environment lock-in due to its difficulty in replicating data across deployment environments.

How Oracle and Redis Enterprise work together

There are many ways Redis Enterprise can make Oracle applications faster, less expensive, and capable of powering modern use-cases that process large volumes of data in real time. Redis Enterprise is commonly used with Oracle as an in-memory database or cache to perform sub-millisecond reads and writes, expand data usability and performance via secondary indexing and key queries, and to enable modern cloud and microservices applications.

- **Secondary key queries:** Use Redis Enterprise's search engine for secondary indexing to support queries of Oracle data held in secondary keys, and cache and index data into Redis Enterprise from your Oracle database. You get sub-millisecond performance which permits you to query more data, faster.
- **Enterprise caching:** Cache data from Oracle into Redis Enterprise to enhance the read and write application performance. Two common methods are:
    - **Write-behind caching for write-heavy workloads:** Use Redis Enterprise to perform high write-throughput tasks such as processing financial transactions. In this instance, Redis Enterprise interacts directly with an application to perform a transaction in real-time, while asynchronously updating tables in Oracle that act as the permanent system of record.
    - **Caching to offload read-heavy workloads:** Redis Enterprise is often prefetched with data held in Oracle so that applications can read data from Redis Enterprise with sub-millisecond latency. A Command Query Responsibility Segregation (CQRS) pattern is often used to decouple reads and writes by writing directly to Oracle, while performing reads from a prefetched Redis Enterprise cache.
- **Application modernization:** Redis Enterprise's performance and flexible data models mean that it frees data stored in Oracle from the limitations of disk-based storage and relational schemas allowing it to be used for modern real-time use-cases. Redis Enterprise can bridge the gap between legacy applications held on-premises and newer applications hosted in the cloud, by allowing data to be written and read in both environments simultaneously in real-time.

Secondary key queries

Due to the limitations of Oracle's performance, data table lookups on secondary keys can be complex and time consuming in Oracle. Redis Enterprise's real-time search engine is used to easily create a secondary index to support queries on secondary keys, enabling you to query more of your Oracle data, faster.

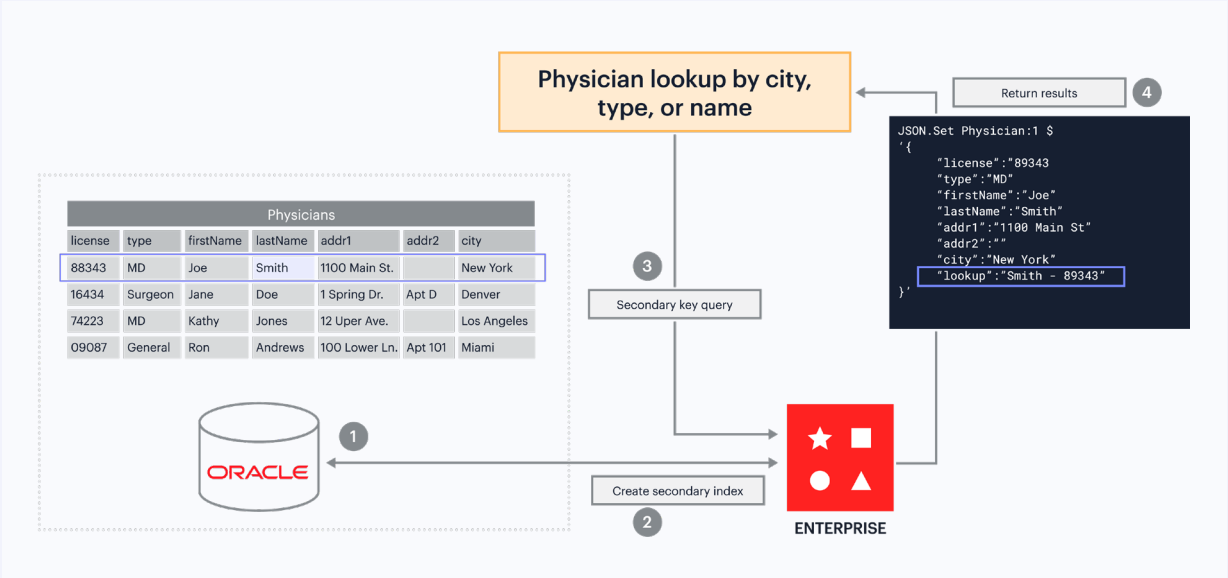
To do this, Redis Enterprise can easily be used to store Oracle data in hashes, create an index on that data, and then query data at the speed of Redis.

Customer case study: Faster, more efficient Oracle data table lookups with Redis Enterprise

A Redis Enterprise customer is a healthcare provider that manages over 600,000 patients per year at over 200 hospitals, along with all the data patient, medical, and billing records that they produce.

They previously stored this vast quantity of data in an Oracle database that created incredibly large indexes, which limited the ability to query the full dataset and was incredibly slow.

The customer turned to Redis Enterprise in order to address these challenges. They hashed data into Redis Enterprise and created a highly-performant secondary index of their Oracle data. In this customer's case, they had a database on physicians with the license number as the primary key:value, however they wanted to open this data up to additional search by attributes like last name or location. They used Redis Enterprise to index and perform real-time queries on secondary key attributes.



Explore a real customer scenario where a healthcare organization uses Redis Enterprise for secondary indexing and real-time queries on secondary key data in their large physician database:

1. The customer's Oracle 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.
2. 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 stores it in Redis Enterprise for real-time query performance.
3. When a user needs to 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.
4. 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:

- The customer was able to get more value out of the data that they held in Oracle, adding the ability to query the wealth of secondary data held in their Oracle tables, making the data more useful to power better customer experiences.
- 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 Oracle's disk-based database to return results.
- Querying Redis Enterprise reduced the burden on the customer's Oracle database, reducing Oracle usage and overall costs.

Caching

Redis Enterprise can be used with Oracle as a cache to speed up Oracle application performance and reduce Oracle usage and cost. Two common Redis Enterprise and Oracle caching patterns are: write-behind caching for high write throughput workloads, and cache prefetching for read-heavy workloads.

Write-behind caching

Oracle typically struggles in high write throughput scenarios when applications have a large volume of transactions that need to be processed in real time and updated in multiple Oracle tables.

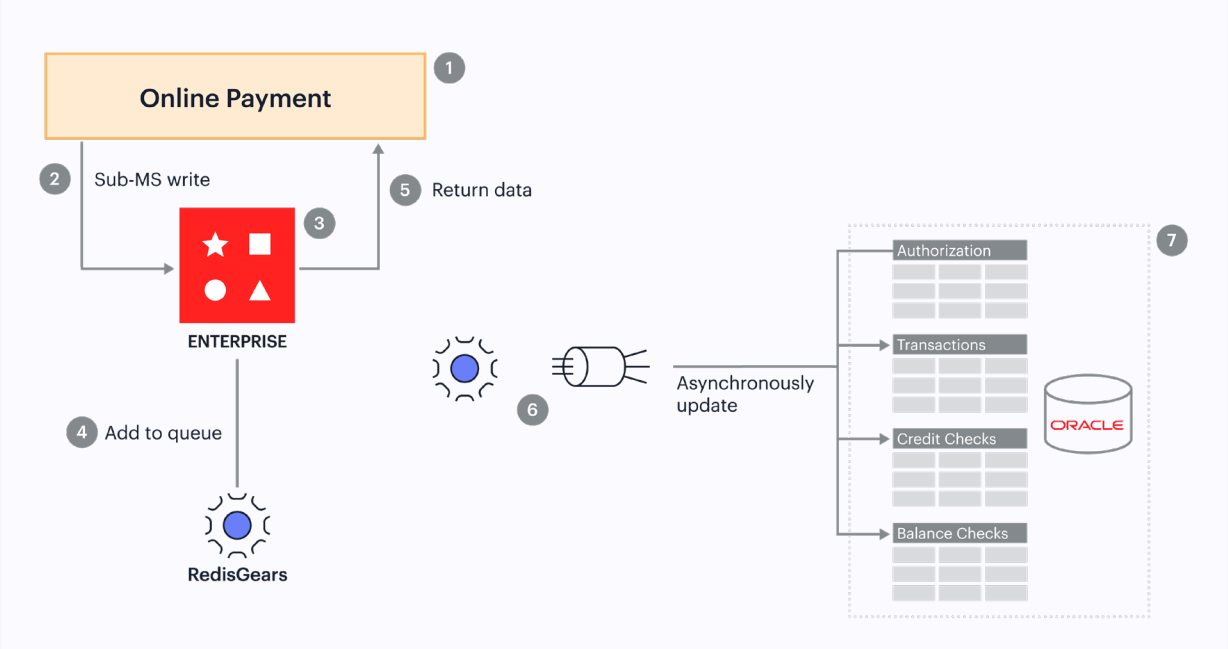
Redis Enterprise can be used as a write-behind cache, directly receiving and processing thousands of write requests from the application in less than a millisecond, and asynchronously updating any subsequent tables in Oracle, which acts as the system of record.

Using Redis Enterprise as a write-behind cache for your Oracle applications greatly speeds up write-heavy workloads, while reducing load on your database and ensuring data is accurate.

Customer case study: Real-time payment processing using Redis Enterprise as a write-behind cache

A major U.S. financial institution has an online banking application that supports millions of customers. At any given second, they process tens of thousands of payments, each of which requires records of the transaction to be written to over ten tables in Oracle. The customer found Oracle alone unable to perform with this high volume of user activity and writes.

To improve performance they used Redis Enterprise as a write-behind cache to process their banking application's online payment in real time, while ensuring data integrity by asynchronously updating their system of record in Oracle after the transaction had been approved.



Explore a scenario where a banking customer uses Redis Enterprise as a write-behind cache to process online financial transactions in real time and store records in Oracle.

1. The bank powers an online banking application used by customers to check balances, make deposits, and make online payments.
2. When a user requests to make a payment, details about the transaction are written to Redis Enterprise with sub-millisecond speed.
3. Redis Enterprise handles writes because it enables a high volume of transactions to be processed in real time, without making users wait while for slow Oracle write operations, by updating Oracle tables in the background after the transaction has been processed in Redis.
4. For each write, a write operation to a Redis Hash key triggers the execution of a RedisGears function that reads data from the Hash, writes it into a Redis Stream, and then adds it to a queue. RedisGears is an engine for batch and event-driven processing of Redis data. Redis Streams is a Redis data structure that acts as an append-only log to syndicate events in real time.
5. The transaction is first processed. Data and a confirmation message are then returned to the user.
6. In the background, another RedisGears function reads the Redis Stream from the queue and writes the details to multiple tables in Oracle. The operation is split into two functions so that the RedisGears function that interacts with Redis Enterprise can operate in real time without being slowed down by the function that is dependent on slower writes to the Oracle tables.
7. Using Redis Streams, data from the transaction is then asynchronously updated from Redis Enterprise into the bank's tables in Oracle, which acts as its system of record and holds a large volume of historical customer and transaction data in tables. Tables hold data around authorization, transactions, credit checks, balance checks, and more.

Customer benefits:

- The customer was able to cut their payment processing time from seconds to sub-milliseconds, greatly enhancing the customer experience in a competitive market.
- They were able to retain Oracle as their system of record, while adding the real-time performance of Redis Enterprise, without any major application refactoring or disruption to their application.
- Because they were processing less Oracle writes, they reduced Oracle usage leading to saved costs.

Cache prefetching and CQRS

Cache prefetching is a technique where data is read from its original storage in slow disk-based storage, like Oracle, which is then written to a much faster in-memory database, Redis Enterprise, before it is needed by your application. Using this approach to offload reads to Redis Enterprise improves application speed while lowering costs via reduced Oracle usage.

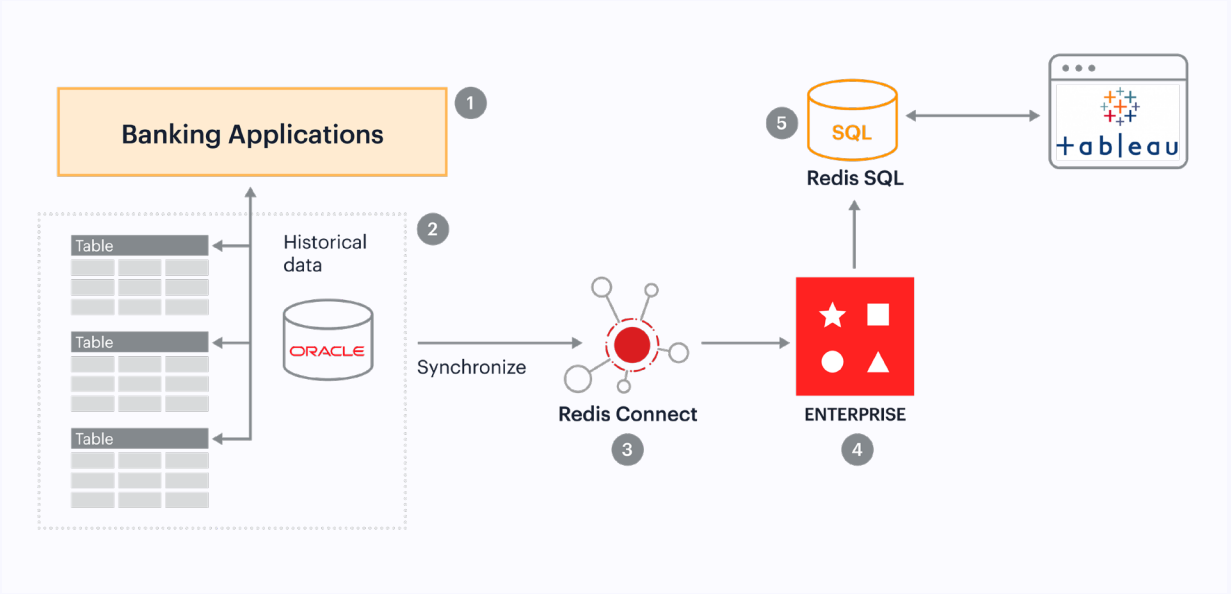
CQRS (Command Query Responsibility Segregation) is an application architecture pattern often used in cache prefetching solutions. CQRS is a critical pattern within microservice architectures that decouple reads and writes. Oracle and Redis Enterprise can use this pattern to enable an application to write only to Oracle, while prefetching reads into Redis Enterprise for the application to read from data that's cached in-memory, for far better read performance.

Customer case study: Powering real-time business intelligence and analytics with cache prefetching from Oracle to Redis Enterprise

A major financial institution had an Oracle database that held over five years of historical data, consisting of millions of records generated by their application portfolio. The customer's executive team needed to use this data for reporting, compliance, and to guide data-driven business decisions.

Data was visualized and analyzed in a Tableau® dashboard that was querying data from Oracle. Due to the incredibly slow process of querying such a high volume of data from Oracle, executives found the dashboards to be nearly unusable.

The customer decided that they needed a new approach to their analytics and BI tools, and decided to use Redis Enterprise alongside Oracle. Oracle was still used as the primary system of record for the customer's historical dataset, however the data was cached into Redis Enterprise to power real-time analytics in their Tableau dashboard.



Explore a scenario where a Redis Enterprise financial services customer prefetches data from Oracle into Redis Enterprise to power real-time analysis on years of historical data held in Oracle.

1. The customer has a portfolio of financial services applications that perform a multitude of functions.
2. The applications store years of historical data, consisting of millions of records, in Oracle.
3. The data is synchronized from the system of record in Oracle into Redis Enterprise using RedisConnect. RedisConnect is used to automatically capture, transform, and load data from Oracle into Redis Enterprise to ensure integrity of data viewed in Tableau.
4. Data is synchronized from Oracle into Redis Enterprise, allowing data to be queried and retrieved in real-time.
5. Redis SQL is a plug-in for RediSearch that acts as an interface between Tableau and Redis Enterprise, allowing Tableau to perform SQL queries against data stored in Redis Enterprise using a JDBC connection, and display it in Tableau dashboards.
6. Executives can query, filter, and analyze large quantities of data from Redis Enterprise in a Tableau dashboard. Data is returned within seconds, rather than minutes.

Customer benefits:

- Data load times were reduced from over twenty minutes to seconds.
- The customer greatly improved their reporting capabilities. Doing so unlocked the value of years of historical business data.
- The customer continued operating their Oracle-based applications without interruption, while synchronizing data into Redis Enterprise to power BI and analytics in Tableau.

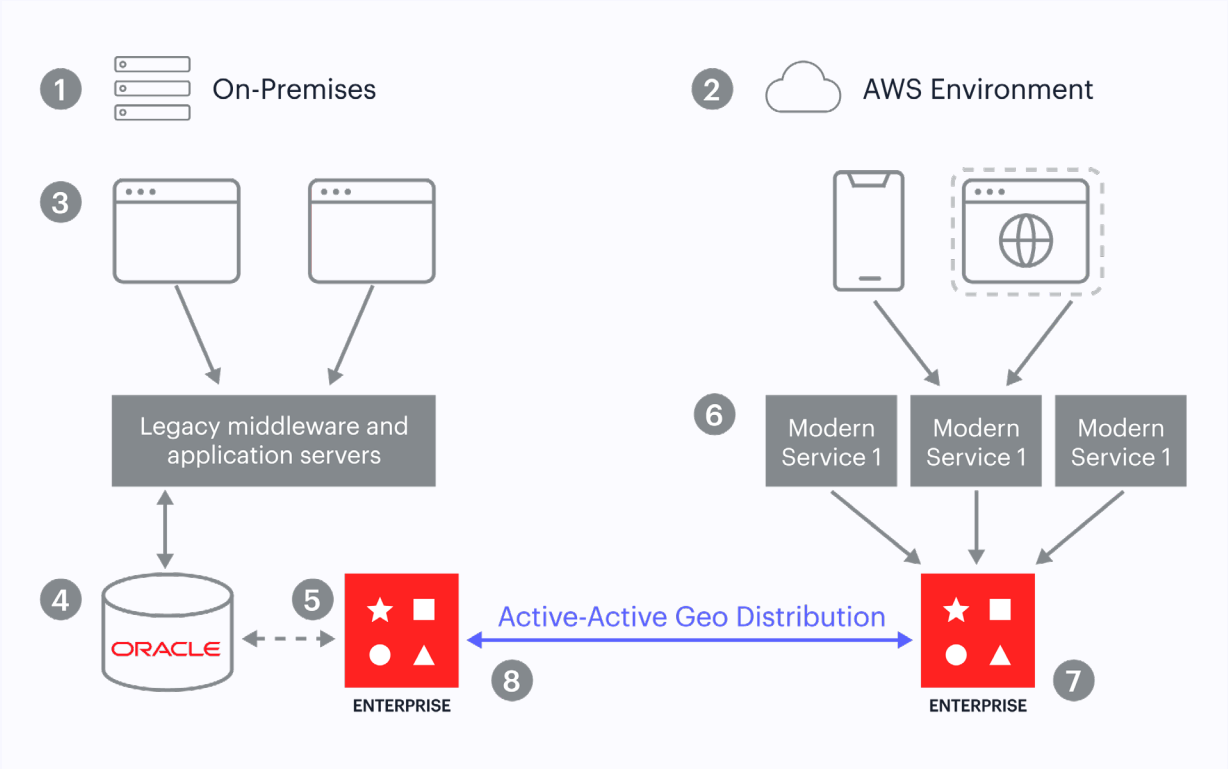
Application modernization

Redis Enterprise is commonly used to free customers' Oracle applications from the limitations of Oracle and facilitate application modernization. Redis Enterprise can unlock data stored in Oracle from the limitations of disk-based storage and relational schemas, and open it up to a multitude of modern use-cases. It also can help organizations move to hybrid or multicloud architectures or support application modernization with a cloud-agnostic data layer that unifies data across all of their environments.

Customer case study: Using Redis Enterprise to modernize Oracle applications

One Redis Enterprise customer had an on-premises environment that hosted legacy applications, their middleware, and an Oracle database that stored their application data. The organization was phasing workloads and applications out of the on-premises environment and replatforming into cloud-native microservices applications hosted in AWS.

Replicating data in Oracle into Redis Enterprise in their cloud environment was critical to operating efficiently in both environments while the customer modernized. It allowed data held in Oracle 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.



Explore how a Redis Enterprise customer's hybrid cloud cache supported application modernization:

1. The on-premises environment contains a number of mature applications that still serve customers, but have not yet been migrated to AWS for modernization.
2. The AWS environment is the destination for workloads that have been moved to the cloud using a phased migration approach.
3. There are a number of aging production applications still in use and hosted in the on-premises environment.
4. The on-premises environment also hosts an Oracle database that contains data needed by **both** the on-premises legacy applications as well as the modernized cloud applications.
5. Data is cached from the Oracle database into a local Redis Enterprise cluster hosted on-premises.
6. A few microservices applications have already been phased out of the on-prem environment and moved into AWS to be modernized, including new mobile and web interfaces for their flagship retail application.
7. A Redis Enterprise cluster is hosted in AWS to provide real-time data with local latency to the new cloud-based applications.
8. 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:

- Using Redis Enterprise enabled the customer to modernize their technology stack over a period of months, without disruption to their application or data held in Oracle. 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.
- Redis Enterprise unlocked data trapped by Oracle's schema and performance limitations to new modern application use-cases.
- 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.

Why your Oracle applications need Redis Enterprise

Today's applications are faster, more powerful, and routinely capable of doing things we considered extraordinary just years ago. But if you're running Oracle applications, chances are your database is hindering your application modernization initiative. There is an easy solution: Redis Enterprise. Redis Enterprise can work alongside your Oracle database, allowing you to continue to operate your existing applications, while simultaneously adding the sub-millisecond performance, scalability, resilience, and flexibility needed to power today's most cutting edge applications. Some of the main benefits of using Redis Enterprise alongside Oracle are:

- **Redis Enterprise brings real-time speed for our real-time world:** It can be used alongside Oracle to add sub-millisecond performance for real-time application experiences, not only providing better user experiences but allowing businesses to unlock new use-cases with their data.
- **Redis Enterprise saves you money by allowing you to use Oracle less:** And can be used to offload reads and writes from Oracle, allowing you to only use Oracle when you really need to, which means less expensive Oracle infrastructure and licenses needed (while boosting performance).
- **Redis Enterprise supports application modernization:** By freeing your Oracle data to move to hybrid or multicloud architectures and supporting application modernization with a cloud-agnostic data layer that unifies data across all of your environments.

Want to learn more?

Now that you know you need the real-time performance that comes with an in-memory cache and database, how do you choose the right solution for you?

Check out our **Buyer's Guide for Enterprise Caching** to learn what enterprise caching is, when it's required, and evaluate criteria for selecting an enterprise-grade caching solution. **Download now.**

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