Alibaba Cloud
Apsara Stack Agility SE

Technical Whitepaper

Version: 1912, Internal: V3.1.0

Issue: 20200311

Legal disclaimer

Document conventions

Style	Description	Example
⛔	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⛔ Danger: Resetting will result in the loss of user configuration data.
⚠	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	⚠ Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
ⓘ	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	ⓘ Notice: If the weight is set to 0, the server no longer receives new requests.
📋	A note indicates supplemental instructions, best practices, tips , and other content.	📋 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings > Network > Set network type.
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands.	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`

Style	Description	Example
{} or {a|b}	**This format is used for a required value, where only one item can be selected.**	`switch {active|stand}`

Contents

1 Object Storage Service (OSS)

1.1 What is OSS?

Apsara Stack Object Storage Service (OSS) is a secure, cost-effective, and highly reliable storage service that is capable of processing large amounts of data.

It can be considered as an immediately available storage solution featuring unlimited storage capacity. Compared with the user-created server storage, OSS has many outstanding advantages in reliability, security, cost, and data processing capabilities. OSS enables you to store and retrieve a variety of unstructured data objects, such as texts, images, audios, and videos over the network at any time.

OSS uploads data files as objects to buckets. OSS is a distributed object storage service that uses the key-value pair format. You can retrieve object content based on unique object names that act as keys.

In OSS, you can:

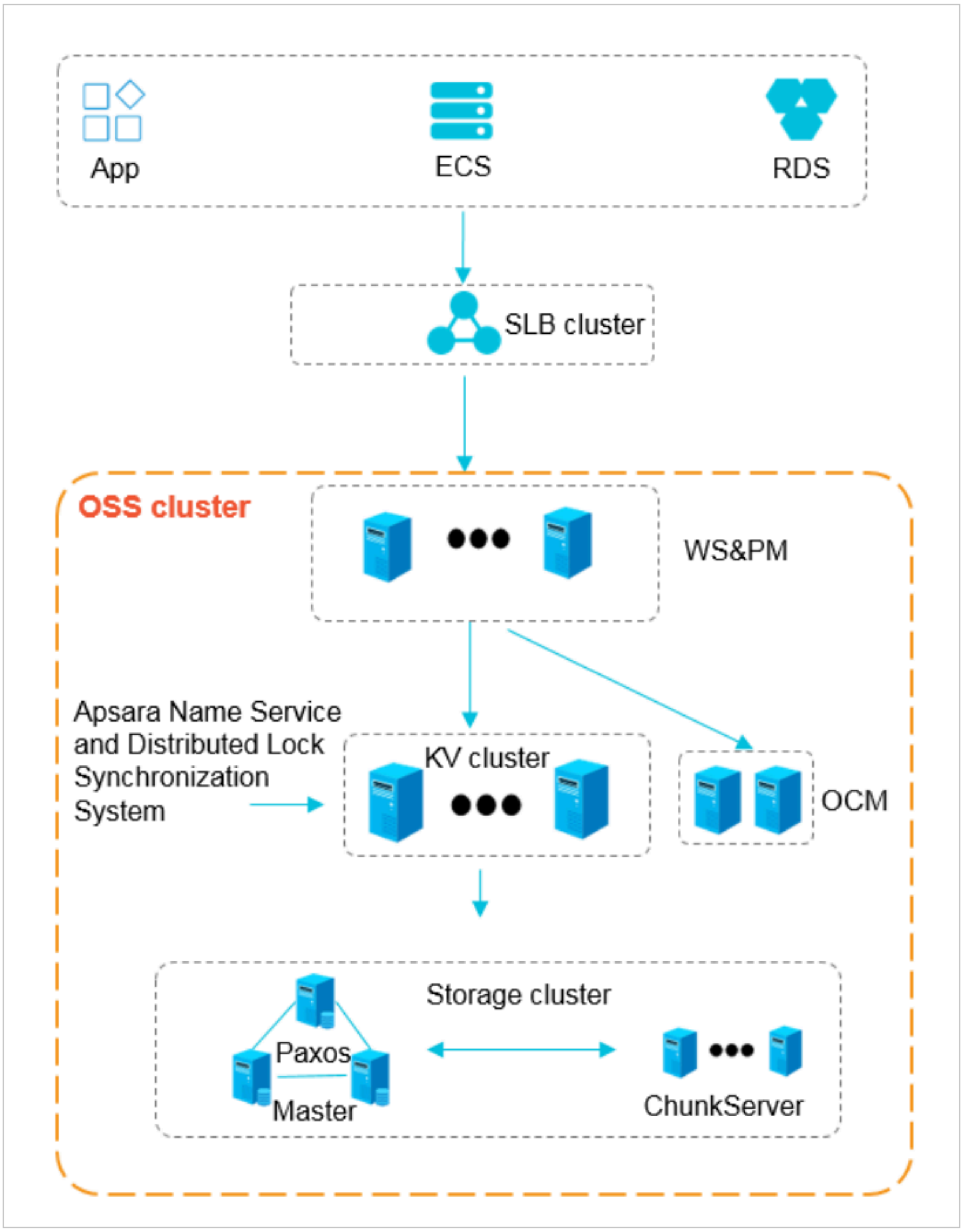
- Create a bucket and upload objects to the bucket.
- Obtain an object URL to share or download the object.
- Modify attributes or metadata of an object, and configure ACL for the object.
- Perform basic and advanced OSS operations in the OSS console.
- Perform basic and advanced operations by using SDKs or calling RESTful API operations in your application.

1.2 Product architecture

Object Storage Service (OSS) is a storage solution built on the Apsara system. It is based on the infrastructure such as Apsara Distributed File System and SchedulerX . This infrastructure provides Apsara Stack services such as OSS with important

features such as distributed scheduling, high-speed networks, and distributed storage. The following figure shows the OSS architecture.

Figure 1-1: OSS architecture



- WS & PM: the protocol layer that receives and authenticates the request sent by using a RESTful protocol. If the authentication succeeds, the request is forwarded to KVEngine for further processing. If the authentication fails, an error message is returned.
- KV cluster: used to process structured data, including reading and writing data based on object names. The KV cluster also supports sporadic bursts of requests . When a service has to run on a different physical server due to a change to the service coordination cluster, the KV cluster can coordinate and find the access point.
- Storage cluster: Metadata is stored in the master node. A distributed message consistency protocol of Paxos is adopted between Master nodes to ensure the consistency of metadata. This method ensures efficient distributed storage of and access to objects.

1.3 Features

Bucket overview

All buckets of the requester are displayed. By default, if you use HTTP to access an OSS endpoint, all of your buckets are displayed.

Set and query the ACL of a bucket

You can set and view the ACL of a bucket. You can set any one of the following ACLs for a bucket:

- Private: Only the owner or authorized users of this bucket can read and write objects in the bucket. Other users, including anonymous users cannot access the objects in the bucket without authorization.
- Public read: Only the owner or authorized users of this bucket can write objects in the bucket. Other users, including anonymous users can only read objects in the bucket.
- Public read/write: Any users, including anonymous users can read and write objects in the bucket.

Create or delete buckets

By default, you can create a maximum of 100 buckets. Bucket names must comply with the naming rules.

The following scenarios may exist when you create a bucket:

- If the bucket you want to create does not exist, the system creates a bucket of a specified name and returns a flag, indicating that the bucket is created.
- If the bucket you want to create exists and the requester is the original bucket owner, the original bucket is retained and a flag is returned, indicating that the bucket is created.
- If the bucket you want to create exists and the requester is not the original bucket owner, a flag is returned, indicating that the bucket fails to create.

If you want to delete a bucket, ensure that the following conditions are met:

- The bucket exists.
- You have the permissions to delete the bucket.
- The bucket contains no data.

List all objects in a bucket

To list all objects in a specified bucket, you must have the corresponding operation permissions on the bucket. If the specified bucket does not exist, an error message is returned.

OSS allows you to search for buckets by prefix and configure the number of objects that can be returned for each search. The maximum number of objects that can be returned for each search is 1,000.

Upload or delete objects

You can upload objects to a specified bucket. You can upload objects to a bucket if the bucket exists and you have the corresponding operation permissions on the bucket. If the object you want to upload has the same name as that of an existing object in the bucket, the new object will overwrite the original object. You can delete a specified object if you have the corresponding operation permissions on the object.

Obtain the content or metadata of objects

To obtain the content or metadata of an object, you must have the corresponding operation permissions on the object.

Access objects

OSS allows you to use a URL to access an object.

Logging and monitoring

You can enable logging for a bucket. After you enable this feature, OSS pushes the access logs on an hourly basis. You can view information such as buckets, traffic , and requests on the Object Storage Service homepage in the Apsara Stack Cloud Management (ASCM) console.

2 ApsaraDB for RDS

2.1 What is ApsaraDB for RDS?

ApsaraDB for RDS is a stable, reliable, and automatically scaling online database service. Based on the distributed file system and high-performance storage, ApsaraDB for RDS allows you to easily perform database operations and maintenance with its set of solutions for disaster recovery, backup, restoration, monitoring, and migration.

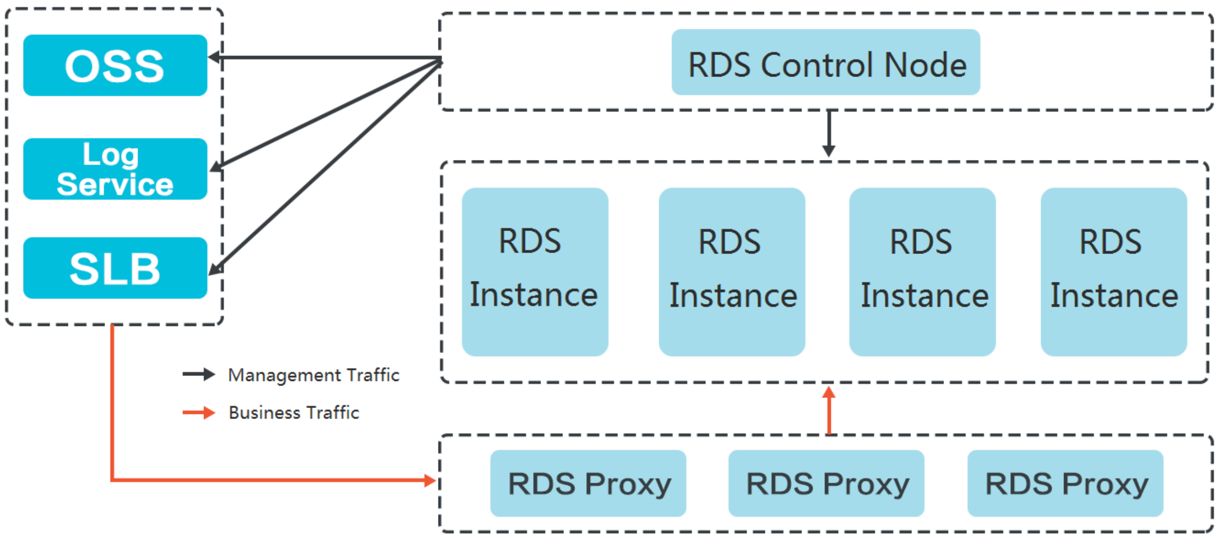
Originally based on a branch of MySQL, ApsaraDB RDS for MySQL has proven its performance and throughput during the high-volume concurrent traffic of Double 11. ApsaraDB RDS for MySQL provides whitelist configuration, backup and restoration, transparent data encryption, data migration, and management for instances, accounts, and databases. It also provides the following advanced features:

- Read-only instance: In scenarios where RDS has a small number of write requests but a large number of read requests, you can enable read/write splitting to distribute read requests away from the primary instance. Read-only instances allow ApsaraDB RDS for MySQL 5.6 to automatically scale the reading capability and increase the application throughput when a large amount of data is being read.
- Data compression: ApsaraDB RDS for MySQL 5.6 allows you to compress data by using the TokuDB storage engine. Data transferred from the InnoDB storage engine to the TokuDB storage engine can be reduced by 80% to 90% in volume. 2 TB of data in InnoDB can be compressed to 400 GB or less in TokuDB. In addition to data compression, TokuDB supports transaction and online DDL operations. TokuDB is compatible with MyISAM and InnoDB applications.

2.2 Architecture

The following figure shows the system architecture of ApsaraDB for RDS.

Figure 2-1: RDS system architecture

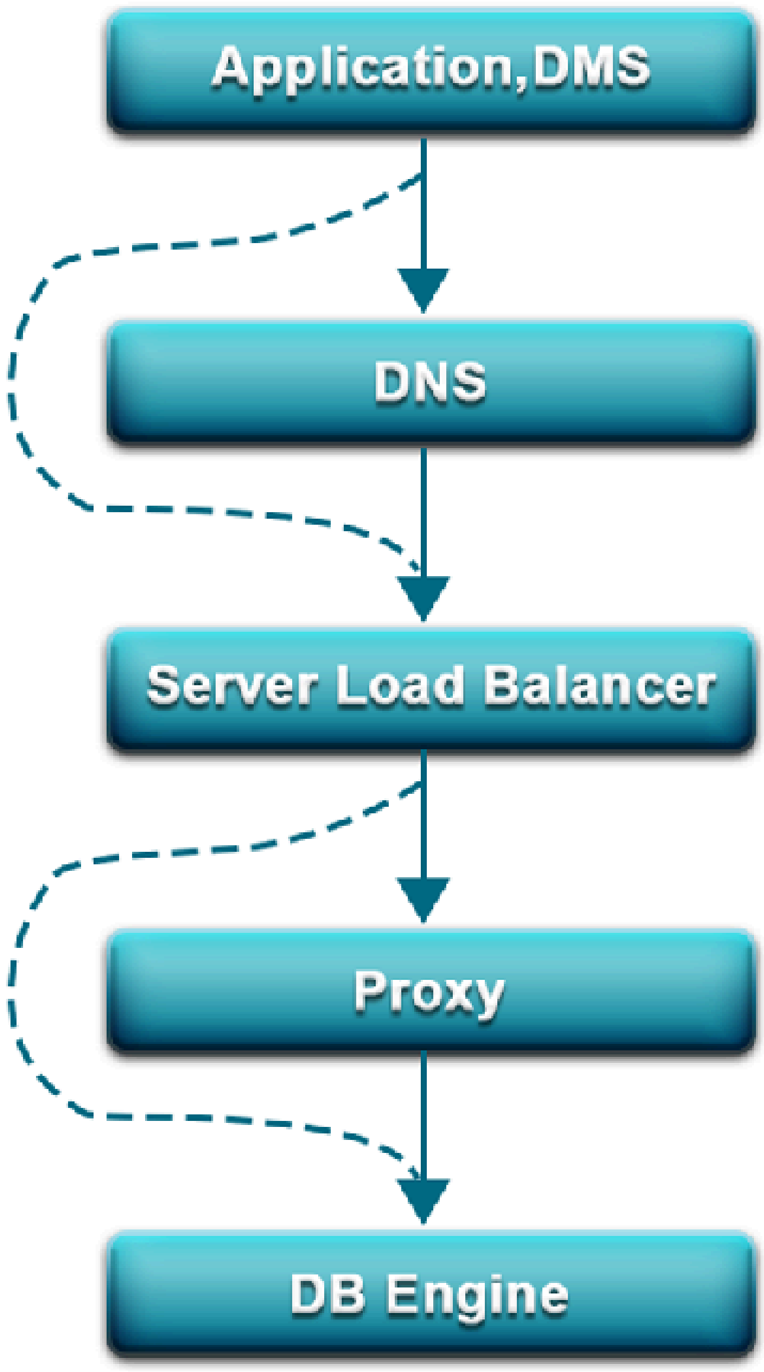


2.3 Features

2.3.1 Data link service

The data link service allows you to add, delete, modify, and query the table schema and data.

Figure 2-2: RDS data link service



DNS

The DNS module can dynamically resolve domain names to IP addresses. Therefore , IP address changes do not affect the performance of RDS instances.

For example, the domain name of an ApsaraDB for RDS instance is test.rds.aliyun .com, and its corresponding IP address is 10.1.1.1. The instance can be accessed when either test.rds.aliyun.com or 10.1.1.1 is configured in the connection pool of a program.

After a zone migration or version upgrade is performed for this ApsaraDB for RDS instance, the IP address may change to 10.1.1.2. If the domain name test.rds. aliyun.com is configured in the connection pool, the instance can still be accessed . However, if the IP address 10.1.1.1 is configured in the connection pool, the instance will no longer be accessible.

SLB

The SLB module provides both the internal IP address and public IP address of an ApsaraDB for RDS instance. Therefore, server changes do not affect the performanc e of the instance.

For example, the internal IP address of an RDS instance is 10.1.1.1, and the corresponding Proxy or DB Engine runs on 192.168.0.1. The SLB module typically redirects all traffic destined for 10.1.1.1 to 192.168.0.1. If 192.168.0.1 fails, another server in hot standby status with the IP address 192.168.0.2 will take over for the initial server. In this case, the SLB module will redirect all traffic destined for 10.1.1 .1 to 192.168.0.2, and the RDS instance will continue to provide services normally.

Proxy

The Proxy module provides a number of features including data routing, traffic detection, and session persistence.

- **Data routing: aggregates the distributed complex queries found in big data scenarios and provides the corresponding capacity management capabilities.**
- **Traffic detection: reduces SQL injection risks and supports SQL log backtracking when necessary.**
- **Session persistence: prevents interruptions to the database connection when faults occur.**

DB Engine

The following table describes the mainstream database protocols supported by RDS.

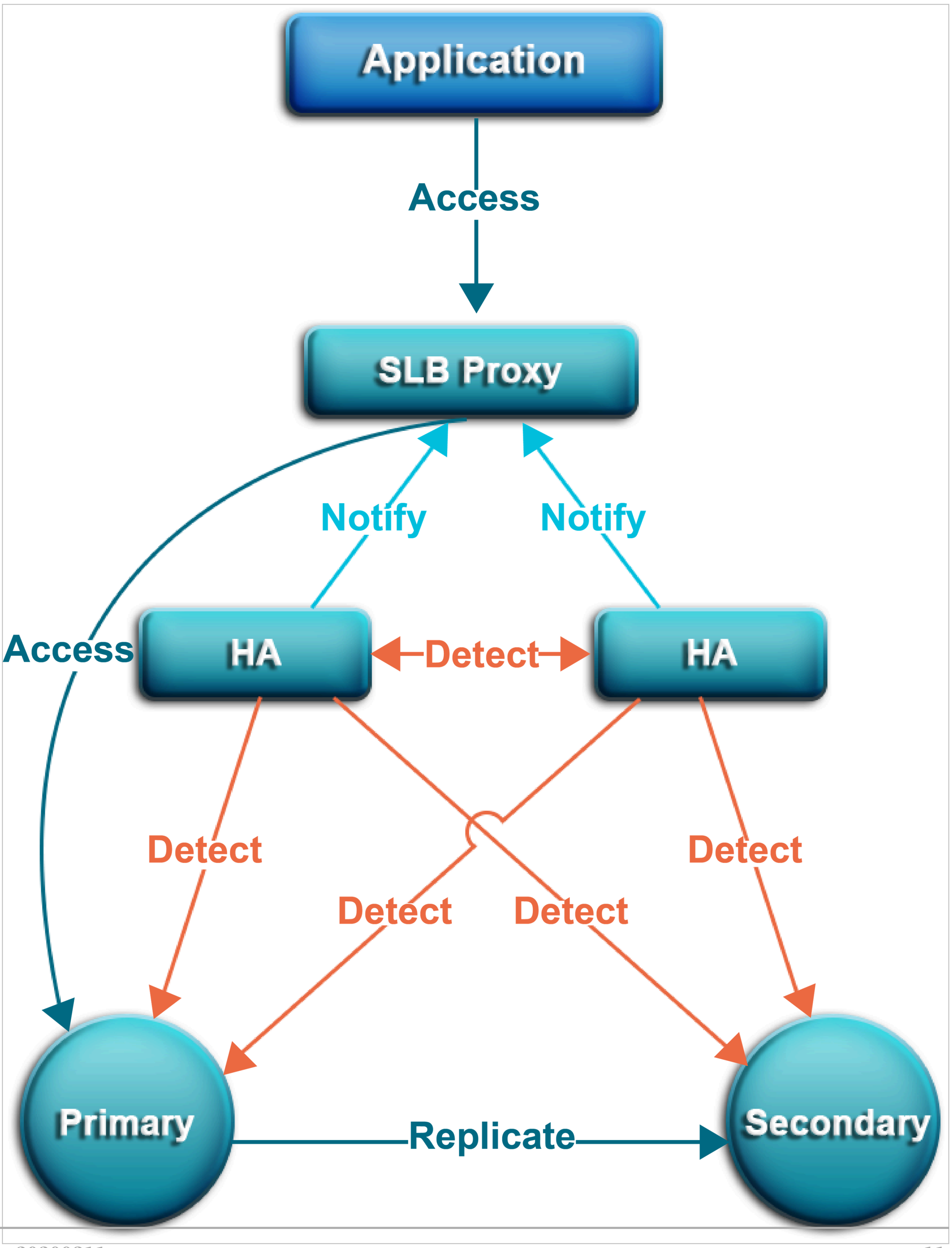
Table 2-1: RDS database protocols

RDBMS	Version
MySQL	**5.6 (including read-only instances)**

2.3.2 High-availability service

The high-availability (HA) service ensures the availability of data link services and processes internal database exceptions. The HA service is implemented by multiple HA nodes.

Figure 2-3: RDS HA service



Detection

The Detection module checks whether the primary and secondary nodes of the DB Engine are providing services normally.

The HA node uses heartbeat information taken at 8 to 10 second intervals to determine the health status of the primary node. This information, along with the health status of the secondary node and heartbeat information from other HA nodes, provides a reference for the Detection module. All this information helps the module avoid misjudgment caused by exceptions such as network jitter. Failover can be completed quickly.

Repair

The Repair module maintains the replication relationship between the primary and secondary nodes of the DB Engine. It can also correct errors that occur on either node during normal operations. For example:

- **It can automatically restore primary/secondary replication after a disconnection .**
- **It can automatically repair table-level damage to the primary or secondary node.**
- **It can save and automatically repair the primary or secondary node in case of crashes.**

Notice

The Notice module informs the SLB or Proxy module of status changes to the primary and secondary nodes to ensure that you always access the correct node.

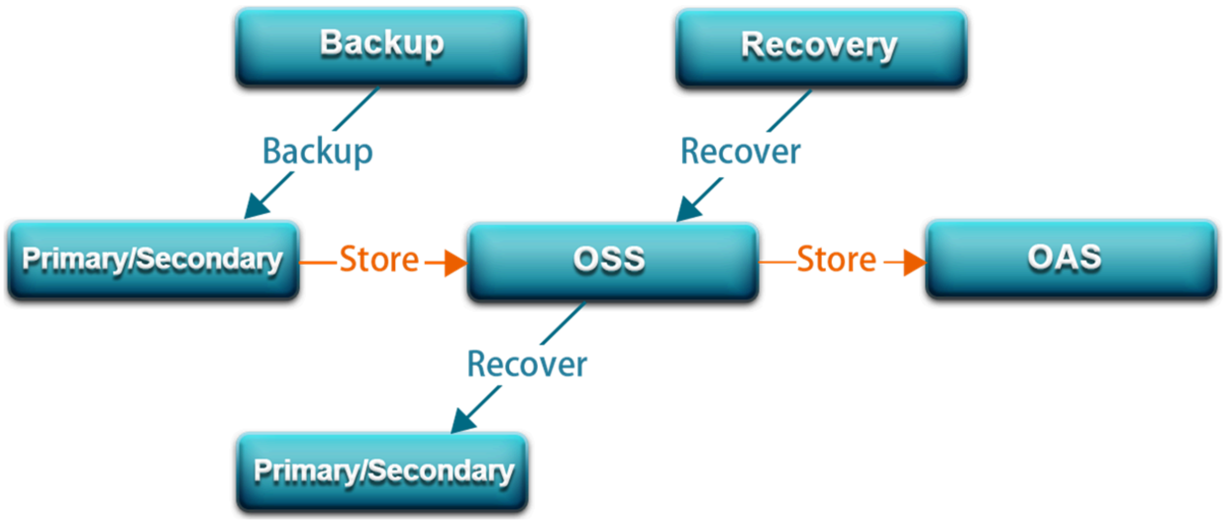
For example, the Detection module discovers problems with the primary node and instructs the Repair module to resolve these problems. If the Repair module fails to resolve a problem, it instructs the Notice module to perform traffic switchover. The Notice module forwards the switching request to the SLB or Proxy module, and then all traffic is redirected to the secondary node.

Meanwhile, the Repair module creates a new secondary node on a different physical server and synchronizes this change back to the Detection module. The Detection module rechecks the health status of the instance.

2.3.3 Backup service

This service supports offline data backup, dumping, and recovery.

Figure 2-4: RDS backup service



Backup

The Backup module compresses and uploads data and logs on both the primary and secondary nodes. ApsaraDB for RDS uploads backup files to OSS and dumps the backup files to a more cost-effective and persistent Archive Storage system. When the secondary node is operating normally, backup is always created on the secondary node so as not to affect the services on the primary node. When the secondary node is unavailable or damaged, the Backup module creates backups on the primary node.

Recovery

The Recovery module restores backup files stored in OSS to a destination node. The Recovery module provides the following features:

- **Primary node rollback: rolls back the primary node to a specified point in time when an operation error occurs.**
- **Secondary node repair: creates a new secondary node to reduce risks when an irreparable fault occurs on the secondary node.**
- **Read-only instance creation: creates a read-only instance from backup files.**

Storage

The Storage module uploads, dumps, and downloads backup files.

All backup data is uploaded to OSS for storage. You can obtain temporary links to download backups as necessary.

Note:
ApsaraDB RDS for PPAS does not allow you to download backup files.

In certain scenarios, the Storage module allows you to dump backup files from OSS to Archive Storage for more cost-effective and longer-term offline storage.

2.3.4 Monitoring service

ApsaraDB for RDS provides multilevel monitoring services across the physical, network, and application layers to ensure service availability.

Service

The Service module tracks the status of services. For example, the Service module monitors whether SLB, OSS, and other cloud services on which RDS depends are operating normally. The monitored metrics include functionality and response time. The Service module also uses logs to determine whether the internal RDS services are operating properly.

Network

The Network module tracks statuses at the network layer. The monitored metrics include:

- Connectivity between ECS and RDS
- Connectivity between physical RDS servers
- Rates of packet loss on VRouters and VSwitches

OS

The OS module tracks the statuses of hardware and OS kernel. The monitored metrics include:

- Hardware maintenance: The OS module constantly checks the operating status of the CPU, memory, motherboard, and storage device. It can predict faults in advance and automatically submit repair reports when it determines a fault is likely to occur.
- OS kernel monitoring: The OS module tracks all database calls and analyzes the causes of slow calls or call errors based on the kernel status.

Instance

The Instance module collects the following information about ApsaraDB for RDS instances:

- **Instance availability information**
- **Instance capacity and performance metrics**
- **Instance SQL execution records**

2.3.5 Scheduling service

The scheduling service allocates resources and manages instance versions.

Resource

The Resource module allocates and integrates underlying RDS resources when you enable and migrate instances. When you use the RDS console or an API operation to create an instance, the Resource module calculates the most suitable host to carry traffic to and from the instance. A similar process occurs during ApsaraDB for RDS instance migration.

After repeated instance creation, deletion, and migration operations, the Resource module calculates the degree of resource fragmentation. It also regularly integrates resources to improve the service carrying capacity.

2.3.6 Migration service

The migration service can migrate data from your on-premises databases to ApsaraDB for RDS.

DTS

DTS can migrate data from on-premises databases to ApsaraDB RDS for MySQL without stopping services.

DTS is a data exchange service that streamlines data migration, real-time synchronization, and subscription. DTS is dedicated to implementing remote and millisecond-speed asynchronous data transmission in various scenarios. Based on the active geo-redundancy architecture designed for Double 11, DTS can implement security, scalability, and high availability by providing real-time data streams to up to thousands of downstream applications.

3 AnalyticDB for PostgreSQL

3.1 What is AnalyticDB for PostgreSQL?

AnalyticDB for PostgreSQL (formerly known as HybridDB for PostgreSQL) is a distributed analytic database that adopts a massive parallel process (MPP) architecture and consists of multiple compute nodes. AnalyticDB for PostgreSQL provides MPP warehousing services and supports horizontal scaling of storage and compute capabilities, online analysis for petabyte levels of data, and offline extract, transform, and load (ETL) task processing.

AnalyticDB for PostgreSQL is developed based on the PostgreSQL kernel and has the following features:

- Supports the SQL:2003 standard, OLAP aggregate functions, views, Procedural Language for SQL (PL/SQL), user-defined functions (UDFs), and triggers. AnalyticDB for PostgreSQL is partially compatible with the Oracle syntax.
- Uses the horizontally scalable MPP architecture and supports range and list partitioning.
- Supports row store, column store, and multiple indexes. It also supports multiple compression methods based on column store to reduce storage costs.
- Supports standard database isolation levels and distributed transactions to ensure data consistency.
- Provides the vector computing engine and the CASCADE-based SQL optimizer to ensure high-performance SQL analysis capabilities.
- Supports the primary/secondary architecture to ensure dual-copy data storage.
- Provides online scaling, monitoring, and disaster recovery to reduce O&M costs.

3.1.1 Scenarios

AnalyticDB for PostgreSQL is applicable to the following OLAP data analysis services.

- **ETL for offline data processing**

  AnalyticDB for PostgreSQL has the following features that make it ideal for optimizing complex SQL queries and aggregating and analyzing large amounts of data:

  - **Supports standard SQL, OLAP window functions, and stored procedures.**
  - **Provides the CASCADE-based SQL optimizer to make complex queries without the need for tuning.**
  - **Built on the MPP architecture for horizontal scaling and PB/s data processing.**
  - **Provides high performance, column store-based storage and aggregation of tables at a high compression ratio to save storage space.**
- **Online high-performance query**

  AnalyticDB for PostgreSQL provides the following benefits for real-time exploration, warehousing, and updating of data:

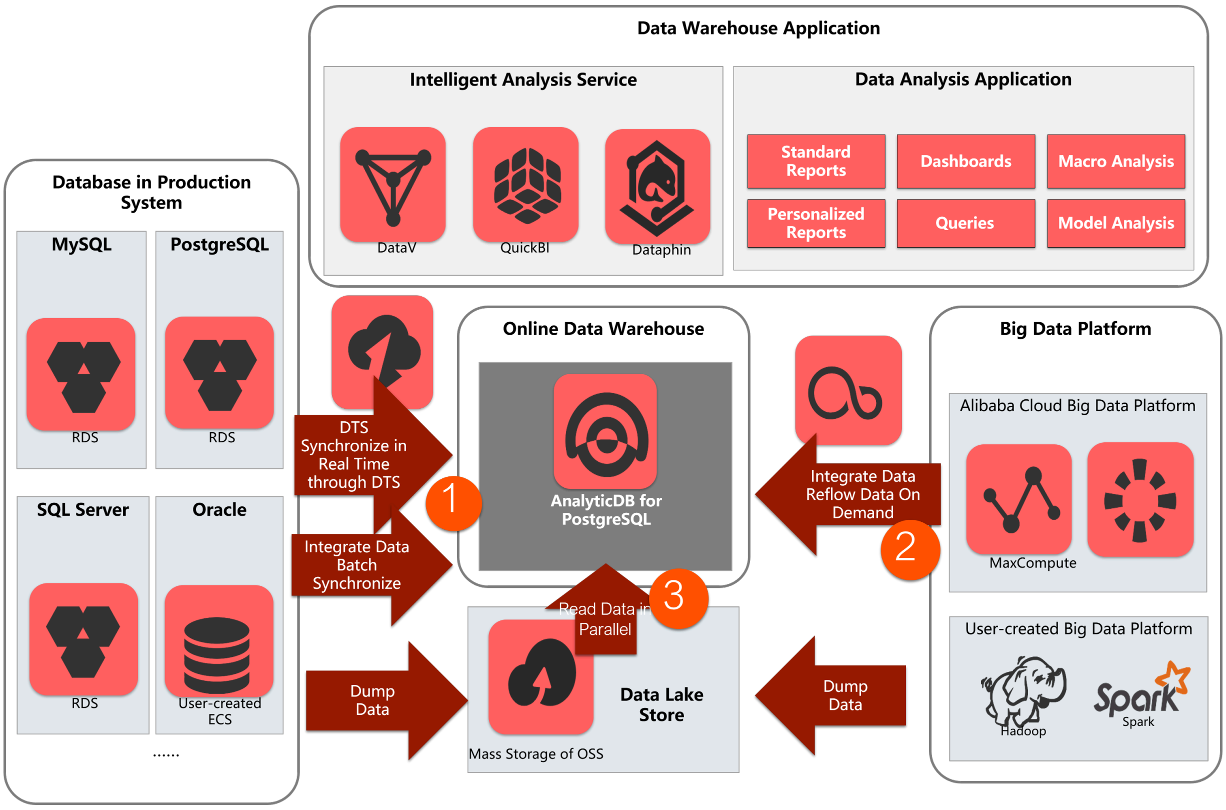
  - **Allows you to write and update high-throughput data through INSERT, UPDATE, and DELETE operations.**
  - **Allows you to query data based on row store and multiple indexes (B-tree, bitmap, and hash) to obtain results in milliseconds.**
  - **Supports distributed transactions, standard database isolation levels, and HTAP.**
- **Multi-model data analysis**

  AnalyticDB for PostgreSQL provides the following benefits for processing of a variety of unstructured data sources:

  - **Supports the PostGIS extension for geographic data analysis and processing.**
  - **Uses the MADlib library of in-database machine learning algorithms to implement AI-native databases.**
  - **Provides high-performance retrieval and analysis of unstructured data such as images, speech, and text through vector retrieval.**
  - **Supports formats such as JSON and can process and analyze semi-structured data such as logs.**

Typical scenarios

AnalyticDB for PostgreSQL is applicable to the following three scenarios:


Data Warehouse Application
Intelligent Analysis Service
DataV
QuickBI
Dataphin
Data Analysis Application
Standard Reports
Dashboards
Macro Analysis
Personalized Reports
Queries
Model Analysis
Database in Production System
MySQL
RDS
PostgreSQL
RDS
SQL Server
RDS
Oracle
User-created ECS
......
DTS Synchronize in Real Time through DTS
Integrate Data Batch Synchronize
1
Online Data Warehouse
AnalyticDB for PostgreSQL
Integrate Data Reflow Data On Demand
2
Read Data in Parallel
3
Dump Data
Data Lake Store
Mass Storage of OSS
Dump Data
Big Data Platform
Alibaba Cloud Big Data Platform
MaxCompute
User-created Big Data Platform
Hadoop
Spark

- Data warehousing service

  Data Transmission Service (DTS) can synchronize data in real time in production system databases such as ApsaraDB RDS for MySQL, ApsaraDB RDS for PostgreSQL, Apsara PolarDB, and traditional databases such as Oracle and SQL Server. Data can also be synchronized in batches to AnalyticDB for PostgreSQL through the data integration service (DataX). AnalyticDB for PostgreSQL supports complex extract, transform, and load (ETL) operations on large amounts of data. These tasks can also be scheduled by Dataworks. AnalyticDB for PostgreSQL also provides high-performance online analysis capabilities and can use Quick BI, DataV, Tableau, and FineReport for report presentation and real-time query.
- Big data analytics platform

  You can import huge amounts of data from MaxCompute, Hadoop, and Spark to AnalyticDB for PostgreSQL through DataX or OSS for high-performance analysis, processing, and exploration.

- **Data lake analytics**

  AnalyticDB for PostgreSQL can use an external table mechanism to access the huge amounts of data stored in OSS in parallel and build an Alibaba Cloud data lake analytics platform.

3.2 Benefits

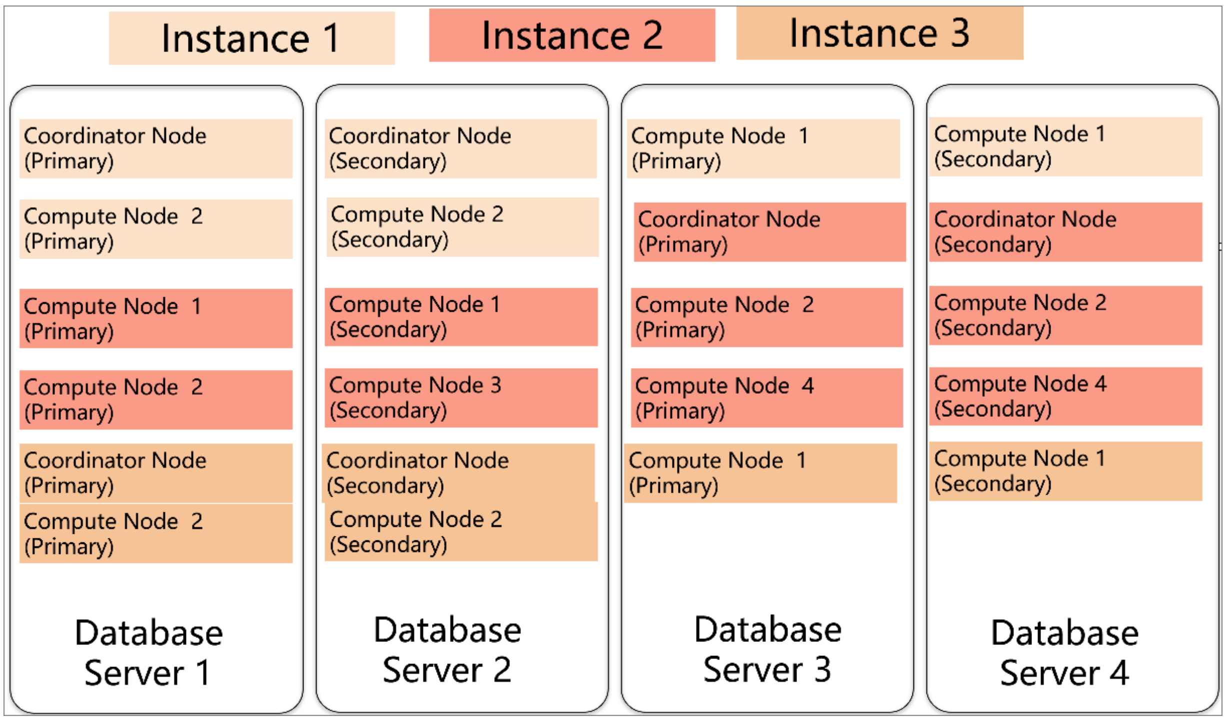
Real-time analysis	**Built on the MPP architecture for horizontal scaling and PB /s data processing. AnalyticDB for PostgreSQL supports the leading vector computing feature and intelligent indexes of column store. It also supports the CASCADE-based SQL optimizer to make complex queries without the need for tuning.**
Stability and reliability	**Provides ACID properties for distributed transactions. Transactions are consistent across nodes and all data is synchronized between primary and secondary nodes. AnalyticDB for PostgreSQL supports distributed deployment and provides transparent monitoring, switching, and restoration to secure your data infrastructure.**
Easy to use	**Supports a large number of SQL syntax and functions, Oracle functions, stored procedures, user-defined functions (UDFs), and isolation levels of transactions and databases. You can use popular BI software and ETL tools online.**
Ultra-high performance	**Supports row store, column store, and multiple indexes. The vector engine provides high-performance analysis and computing capabilities. The CASCADE-based SQL optimizer enables complex queries without the need for tuning. It supports high-performance parallel import of data from OSS.**
Scalability	**Enables you to scale up compute nodes, CPU, memory, and storage resources on demand to improve OLAP performance.** **Supports transparent OSS operations. OSS offers a larger storage capacity for cold data that does not require online analysis.**

3.3 Architecture

Physical cluster architecture

The following figure shows the physical cluster architecture of AnalyticDB for PostgreSQL.

Figure 3-1: Physical cluster architecture


Instance 1
Instance 2
Instance 3
Coordinator Node (Primary)
Compute Node 2 (Primary)
Compute Node 1 (Primary)
Compute Node 2 (Primary)
Coordinator Node (Primary)
Compute Node 2 (Primary)
Database Server 1
Coordinator Node (Secondary)
Compute Node 2 (Secondary)
Compute Node 1 (Secondary)
Compute Node 3 (Secondary)
Coordinator Node (Secondary)
Compute Node 2 (Secondary)
Database Server 2
Compute Node 1 (Primary)
Coordinator Node (Primary)
Compute Node 2 (Primary)
Compute Node 4 (Primary)
Compute Node 1 (Primary)
Database Server 3
Compute Node 1 (Secondary)
Coordinator Node (Secondary)
Compute Node 2 (Secondary)
Compute Node 4 (Secondary)
Compute Node 1 (Secondary)
Database Server 4

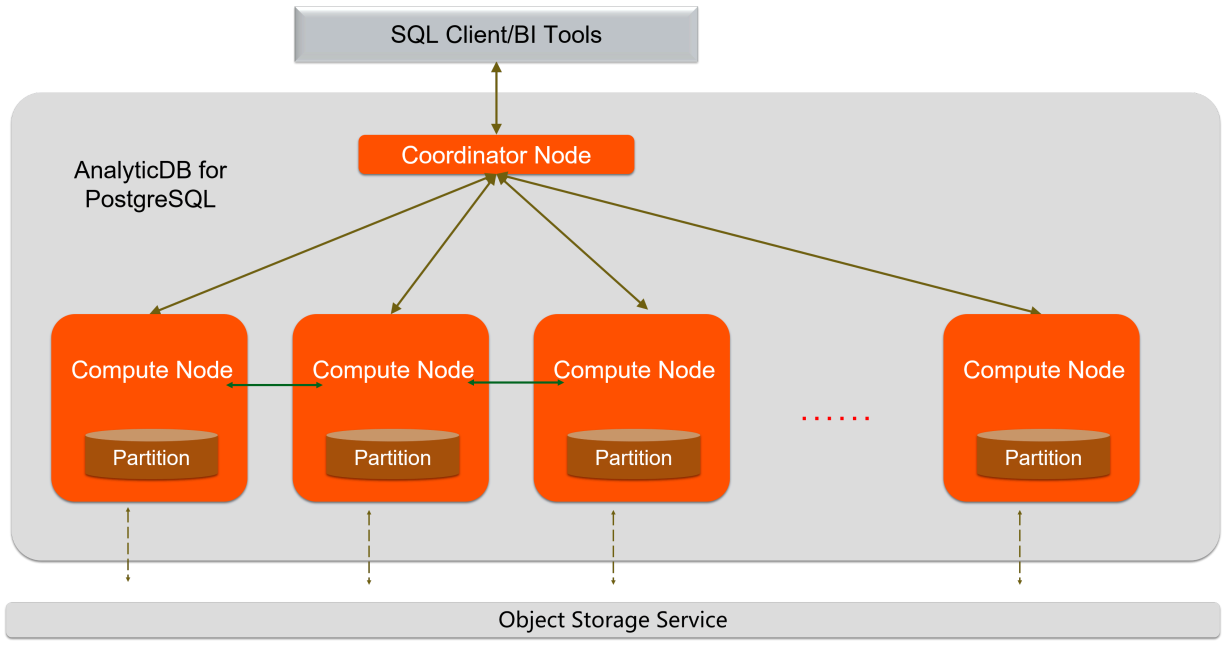
You can create multiple instances within a physical cluster of AnalyticDB for PostgreSQL. Each cluster includes two components: the coordinator node and the compute node.

- **The coordinator node is used for access from applications. It receives connection requests and SQL query requests from clients and dispatches computing tasks to compute nodes. The cluster deploys a secondary node of the coordinator node on an independent physical server and replicates data from the primary node to the secondary node for failover. The secondary node does not accept external connections.**
- **Compute nodes are independent instances in AnalyticDB for PostgreSQL. Data is evenly distributed across compute nodes by hash value or RANDOM function , and is analyzed and computed in parallel. Each compute node consists of a primary node and a secondary node for automatic failover.**

Logical architecture of an instance

You can create multiple instances within a cluster of AnalyticDB for PostgreSQL. The following figure shows the logical architecture of an instance.

Figure 3-2: Logical architecture of an instance



Data is distributed across compute nodes by hash value or RANDOM function of a specified distributed column. Each compute node consists of a primary node and a secondary node to ensure dual-copy storage. High-performance network communication is supported across nodes. When the coordinator node receives a request from the application, the coordinator node parses and optimizes SQL statements to generate a distributed execution plan. After the coordinator node sends the execution plan to the compute nodes, the compute nodes will perform an MPP execution of the plan.

3.4 Features

3.4.1 Distributed architecture

AnalyticDB for PostgreSQL is built on MPP architecture. Data is distributed evenly across nodes by hash value or RANDOM function, and is analyzed and computed in parallel. Storage and computing capacities are scaled horizontally as more nodes are added to ensure a quick response as the data volume increases.

AnalyticDB for PostgreSQL supports distributed transactions to ensure data consistency among nodes. It supports three transaction isolation levels: SERIALIZABLE, READ COMMITTED, and READ UNCOMMITTED.

3.4.2 High-performance data analysis

AnalyticDB for PostgreSQL supports column store and row store for tables. Row store provides high update performance and column store provides high OLAP aggregate analysis performance for tables. AnalyticDB for PostgreSQL supports the B-tree index, bitmap index, and hash index that enable high-performance analysis, filtering, and query.

AnalyticDB for PostgreSQL adopts the CASCADE-based SQL optimizer. AnalyticDB for PostgreSQL combines the cost-based optimizer (CBO) with the rule-based optimizer (RBO) to provide SQL optimization features such as automatic subquery decorrelation. These features enable complex queries without the need for tuning.

3.4.3 High-availability service

AnalyticDB for PostgreSQL builds a system based on the Apsara system of Alibaba Cloud for automatic monitoring, diagnostics, and error handling to reduce O&M costs.

The coordinator node compiles and optimizes SQL statements by storing database metadata and receiving query requests from clients. The coordinator node adopts a primary/secondary architecture to ensure strong consistency of metadata. If the primary coordinator node fails, the service will be automatically switched to the secondary coordinator node.

All compute nodes adopt a primary/secondary architecture to ensure strong data consistency between primary and secondary nodes when data is written into or updated. If the primary compute node fails, the service will be automatically switched to the secondary compute node.

3.4.4 Data synchronization and tools

You can use Data Transmission Service (DTS) or DataWorks to synchronize data from MySQL or PostgreSQL databases to AnalyticDB for PostgreSQL. Popular extract, transform, and load (ETL) tools can import ETL data and schedule jobs on AnalyticDB for PostgreSQL databases. You can also use standard SQL syntax to query data from formatted files stored in OSS by using external tables in real time.

AnalyticDB for PostgreSQL supports Business Intelligence (BI) reporting tools, including Quick BI, DataV, Tableau, and FineReport. It also supports ETL tools, including Informatica and Kettle.

3.4.5 Data security

AnalyticDB for PostgreSQL supports IP whitelist configuration. You can add up to 1,000 IP addresses of servers to the whitelist to allow access to your instance and control risks from access sources. AnalyticDB for PostgreSQL also supports Anti-DDoS that monitors inbound traffic in real time. When large amounts of malicious traffic is identified, the traffic is scrubbed through IP filtering. If traffic scrubbing is not sufficient, the black hole process will be triggered.

3.4.6 Supported SQL features

- Supports row store and column store.
- Supports multiple indexes, including the B-tree index, bitmap index, and hash index.
- Supports distributed transactions and standard isolation levels to ensure data consistency among nodes.
- Supports character, date, and arithmetic functions.
- Supports stored procedures, user-defined functions (UDFs), and triggers.
- Supports views.
- Supports range partitioning, list partitioning, and the definition of multi-level partitions.
- Supports multiple data types. The following table provides a list of data types and their information.

Parameter	Alias	Storage size	Range	Description
bigint	int8	8 bytes	-9223372036854775808 to 9223372036854775807	Large-range integer
bigserial	serial8	8 bytes	1 to 9223372036854775807	Large auto-increment integer

Parameter	Alias	Storage size	Range	Description
bit [(n)]	N/A	n bits	Bit string constant	Fixed-length bit string
bit varying [(n)]	varbit	Variable-length bit string	Bit string constant	Variable-length bit string
boolean	bool	1 byte	true/false, t/f, yes/no, y/n, 1/0	Boolean value (true/false)
box	N/A	32 bytes	((x1,y1),(x2,y2))	A rectangular box on a plane , not allowed in distribution key columns
bytea	N/A	1 byte + binary string	Sequence of octets	Variable-length binary string
character [(n)]	char[(n)]	1 byte + n	String up to n characters in length	Fixed-length, blank-padded string
character varying [(n)]	varchar [(n)]	1 byte + string size	String up to n characters in length	Variable length with limit
cidr	N/A	12 or 24 bytes	N/A	IPv4 and IPv6 networks
circle	N/A	24 bytes	24 bytes	A circle on a plane, not allowed in distribution key columns
date	N/A	4 bytes	4,713 BC to 294,277 AD	Calendar date (year, month, day)
decimal [(p, s)]	numeric [(p, s)]	variable	No limit	User-specified precision, exact
double precision	float8	8 bytes	Precise to 15 decimal digits	Variable precision, inexact
	float			

Parameter	Alias	Storage size	Range	Description
inet	N/A	12 or 24 bytes	N/A	IPv4 and IPv6 hosts and networks
Integer	int or int4	4 bytes	-2.1E+09 to + 2147483647	Typical choice for integer
interval [(p)]	N/A	12 bytes	-178000000 years to 178000000 years	Time span
json	N/A	1 byte + JSON size	JSON string	Unlimited variable length
lseg	N/A	32 bytes	((x1,y1),(x2,y2))	A line segment on a plane, not allowed in distribution key columns
macaddr	N/A	6 bytes	N/A	Media Access Control (MAC) address
money	N/A	8 bytes	-9223372036 8547758.08 to +9223372036 8547758.07	Currency amount
path	N/A	16+16n bytes	[(x1,y1),...]	A geometric path on a plane, not allowed in distribution key columns
point	N/A	16 bytes	(x,y)	A geometric point on a plane, not allowed in distribution key columns

Parameter	Alias	Storage size	Range	Description
polygon	N/A	40+16n bytes	((x1,y1),...)	A closed geometric path on a plane, not allowed in distribution key columns
real	float4	4 bytes	Precise to 6 decimal digits	Variable precision, inexact
serial	serial4	4 bytes	1 to 2147483647	Auto-increment integer
smallint	int2	2 bytes	-32768 to 32767	Small-range integer
text	N/A	1 byte + string size	Variable-length string	Unlimited variable length
time [(p)] [without time zone]	N/A	8 bytes	00:00:00[.000000] to 24:00:00[.000000]	Time of day (without time zone)
time [(p)] with time zone	timetz	12 bytes	00:00:00+1359 to 24:00:00-1359	Time of day (with time zone)
timestamp [(p)] [without time zone]	N/A	8 bytes	4,713 BC to 294,277 AD	Date and time
timestamp [(p)] with time zone	timestamptz	8 bytes	4,713 BC to 294,277 AD	Date and time (with time zone)
xml	N/A	1 byte + XML size	Variable-length XML string	Unlimited variable length

4 Data Transmission Service (DTS)

4.1 What is DTS?

Data Transmission Service (DTS) is a data service provided by Alibaba Cloud. DTS supports data transmission between various types of data sources, such as relational databases.

DTS provides data transmission capabilities such as data migration and change tracking. DTS can be used in many scenarios, such as interruption-free data migration, geo-disaster recovery, cross-border data synchronization, and cache updates. DTS helps you build a data architecture that features high availability, scalability, and security.

- **DTS allows you to simplify data transmission and focus on business development .**
- **DTS supports MySQL as the data source type.**

4.1.1 Environment requirements

You must use DTS on hosts of the following models:

- **PF51. ***
- **PV52P2M1. ***
- **DTS_E. ***
- **PF61. ***
- **PF61P1. ***
- **PV62P2M1. ***
- **PV52P1. ***
- **Q5F53M1. ***
- **PF52M2. ***
- **Q41. ***
- **Q5N1.22**
- **Q5N1.2B**
- **Q46.22**
- **Q46.2B**

- W41.22
- W41.2B
- W1.22
- W1.2B
- W1.2C
- D13.12

You must use the following operating system:

AliOS7U2-x86-64

Notice:

- Do not use DTS on hosts that are excluded from the preceding models.
- The `/apsara` directory used by DTS resides on only one hard disk. Make sure that the available space in the directory is larger than 2 TB.

  If the available space in the `/apsara` directory is less than 2 TB, tasks cannot run as expected and errors will occur. If a task fails, the task recovery and data pulling are affected.

4.2 Benefits

DTS supports data transmission between data sources such as relational databases and OLAP databases. DTS provides data transmission capabilities such as data migration and change tracking. Compared with other data migration and synchronization tools, DTS provides transmission channels with higher compatibility, performance, security, and reliability. DTS also provides a variety of features to help you create and manage transmission channels.

High compatibility

DTS supports data migration and synchronization between homogeneous and heterogeneous data sources. For migration between heterogeneous data sources, DTS supports schema conversion.

DTS provides data transmission capabilities such as data migration and change tracking. In change tracking, data is transmitted in real time.

DTS minimizes the impact of data migration on applications to ensure service continuity. The application downtime during data migration is minimized to several seconds.

High performance

DTS uses high-end servers to ensure the performance of each data synchronization or migration channel.

DTS uses a variety of optimization measures for data migration.

Security and reliability

DTS is implemented based on clusters. If a node in a cluster is unavailable or faulty , the control center switches all tasks on this node to another node in the cluster.

Secure transmission protocols and tokens are used for authentication across DTS modules to ensure reliable data transmission.

Ease of use

The DTS console provides a codeless wizard for you to create and manage channels.

To facilitate channel management, the DTS console shows information about transmission channels, such as transmission status, progress, and performance.

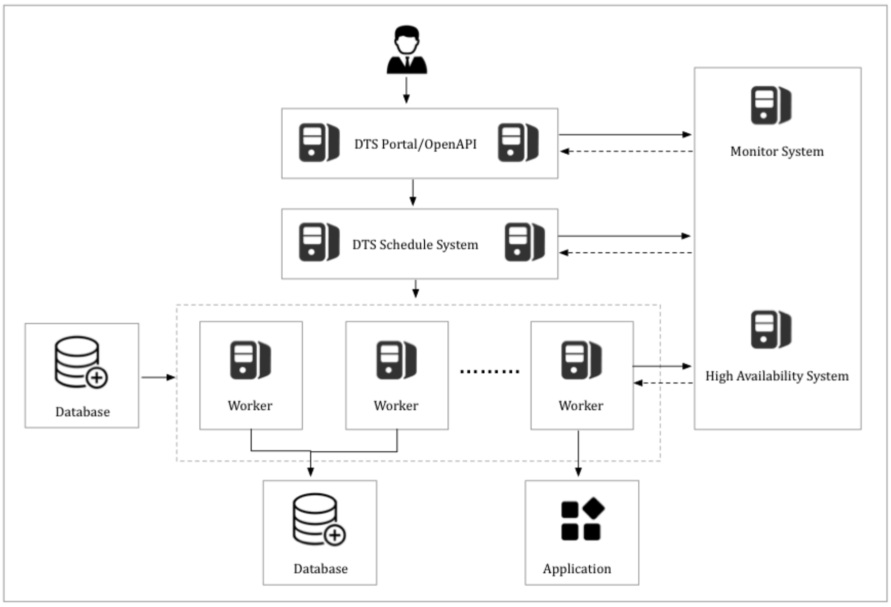
DTS supports resumable transmission, and monitors channel status on a regular basis. If DTS detects a network failure or system error, DTS automatically fixes the failure or error and restarts the channel. If the failure or error persists, you must manually repair and restart the channel in the DTS console.

4.3 Architecture

System architecture

The following figure shows the system architecture of DTS.

Figure 4-1: System architecture



- **High availability**

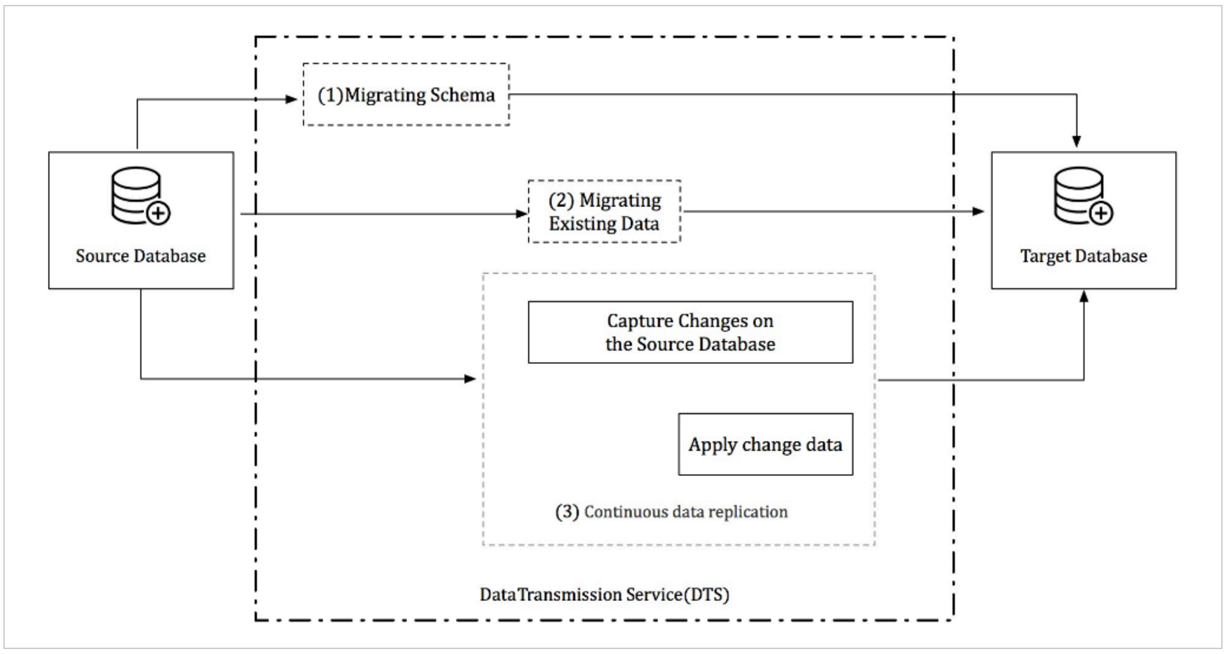
  Each module in DTS has primary and secondary nodes to ensure high availability . The disaster recovery module runs a health check on each node in real time. If a node failure is detected, the module requires only a few seconds to switch the channel to a healthy node.
- **Connection reliability**

  To ensure the connection reliability of change tracking channels, the disaster recovery module checks for configuration changes, such as changes of a data source address. If a data source address is changed, the module allocates a new connection method to ensure the stability of the channel.

Design concept of data migration

The following figure shows the design concept of data migration.

Figure 4-2: Design concept of data migration


(1)Migrating Schema
Source Database
(2) Migrating Existing Data
Target Database
Capture Changes on the Source Database
Apply change data
(3) Continuous data replication
DataTransmission Service(DTS)

Data migration supports schema migration, full data migration, and incremental data migration. The following processes ensure service continuity during data migration:

1. **Schema migration**
2. **Full data migration**
3. **Incremental data migration**

To migrate data between heterogeneous databases, DTS reads the source database schema, translates the schema into the syntax of the destination database, and imports the schema to the destination database.

A full data migration requires a long period of time. During this process, incremental data is continuously written to the source database. To ensure data consistency, DTS starts the incremental data reading module before full data migration. This module retrieves incremental data from the source database, and parses, encapsulates, and locally stores the data.

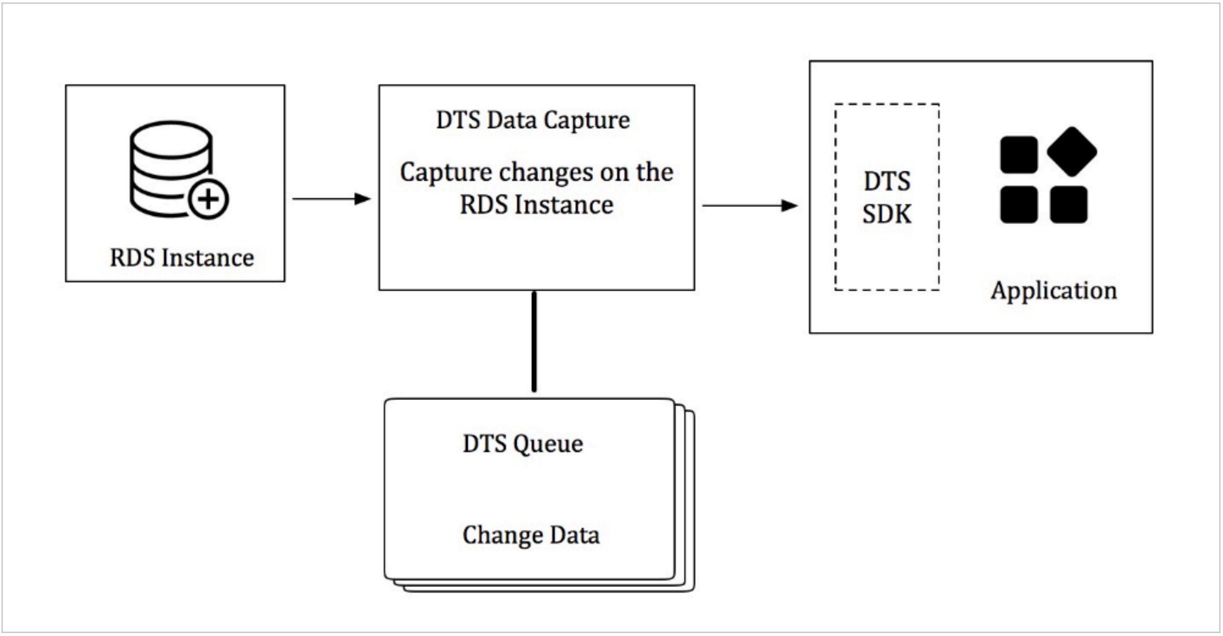
After the full data migration is complete, DTS starts the incremental data loading module. This module retrieves incremental data from the incremental data reading

module. After reverse parsing, filtering, and encapsulation, incremental data is migrated to the destination database in real time.

Design concept of change tracking

The following figure shows the design concept of change tracking.

Figure 4-3: Design concept of change tracking



The change tracking feature allows you to obtain incremental data from an RDS instance in real time. You can subscribe to incremental data on the change tracking server by using DTS SDKs. You can also customize data consumption rules based on your business requirements.

The incremental data reading module on the server side of DTS retrieves raw data from the source instance. After parsing, filtering, and syntax conversion, incremental data is locally stored.

The incremental data reading module connects to the source instance by using a database protocol and retrieves incremental data from the source instance in real time. If the source instance is an ApsaraDB RDS for MySQL instance, the incrementa l data reading module connects to the source instance by using the binary log dump protocol.

DTS ensures high availability of the incremental data reading module and consumption SDK processes.

If an error is detected in the incremental data reading module, the disaster recovery module restarts the incremental data reading module on a healthy node. This ensures high availability of the incremental data reading module.

DTS ensures high availability of consumption SDK processes on the server. If you start multiple consumption SDK processes for the same change tracking channel, the server pushes incremental data to only one process at a time. If an error occurs on a process, the server pushes data to another healthy consumption process.

4.4 Features

4.4.1 Data migration

You can use DTS to migrate data between various types of data sources. Typical scenarios include data migration to the cloud, data migration between instances within Apsara Stack, and database sharding and scaling. DTS supports data migration between homogeneous and heterogeneous data sources. It also supports extract, transform, and load (ETL) features such as data filtering and object name mapping for databases, tables, and columns.

Data source and data migration types

The following table lists the data sources and data migration types that are supported by DTS.

Table 4-1: Data source and data migration types

Data source	Schema migration	Full data migration	Incremental data migration
MySQL database	Supported	Supported	Supported

DTS supports migrating data from the following types of data sources:

- User-created on-premises database

DTS supports migrating data to the following types of data sources:

- User-created on-premises database

Online migration

DTS uses online migration. You only need to configure the source instance, destination instance, and objects to be migrated. DTS automatically completes the entire data migration process. You can select all of the supported migration types to minimize the impact of online data migration on your services. However, you must ensure that DTS servers can connect to both the source and destination instances.

Data migration types

DTS supports schema migration, full data migration, and incremental data migration.

- **Schema migration: migrates schemas from the source instance to the destination instance.**
- **Full data migration: migrates historical data from the source instance to the destination instance.**
- **Incremental data migration: migrates incremental data that is generated during migration from the source instance to the destination instance in real time. You can select schema migration, full data migration, and incremental migration to ensure service continuity.**

ETL features

Data migration supports the following extract, transform, and load (ETL) features:

- **Object name mapping for databases, tables, and columns: You can migrate data between two databases, tables, or columns that have different names.**
- **Data filtering: You can use SQL conditions to filter the required data in a specific table. For example, you can specify a time range to migrate only the latest data.**

Alerts

If an error occurs during data migration, DTS immediately sends an SMS alert to the task owner. This allows the owner to handle the error at the earliest opportunit y.

Migration task

A migration task is a basic unit of data migration. To migrate data, you must create a migration task in the DTS console. To create a migration task, you must configure the required information such as the source and destination instances

, migration types, and objects to be migrated. You can create, manage, stop, and delete migration tasks in the DTS console.

The following table describes the statuses of a migration task.

Table 4-2: Statuses of a migration task

Status	Description	Available operation
Not Started	The migration task is configured but the precheck is not performed.	· Perform the precheck · Delete the migration task
Prechecking	A precheck is being performed but the migration task is not started.	Delete the migration task
Passed	The migration task has passed the precheck but has not been started.	· Start the migration task · Delete the migration task
Migrating	Data is being migrated.	· Pause the migration task · Stop the migration task · Delete the migration task

Status	Description	Available operation
Migration Failed	An error occurred during migration. You can identify the point of failure based on the progress of the migration task.	Delete the migration task
Paused	The migration task is paused.	· Start the migration task · Delete the migration task
Completed	The migration task is completed, or you have stopped data migration by clicking End.	Delete the migration task

4.4.2 Change tracking

You can use DTS to retrieve incremental data from user-created MySQL databases in real time. This feature applies to the following scenarios: cache updates, business decoupling, asynchronous data processing, and real-time synchronization of heterogeneous data and extract, transform, and load (ETL) operations.

Features

You can use DTS to retrieve incremental data from user-created MySQL databases.

Data sources

The change tracking feature supports the following types of data source:

- MySQL database

Objects for change tracking

The objects for change tracking include tables and databases. You can specify one or more tables from which you want to track data changes.

In change tracking, incremental data includes data manipulation language (DML) operations and data definition language (DDL) operations. When you configure change tracking, you must select the type of operation.

Change tracking channel

A change tracking channel is the basic unit of incremental data tracking and consumption. To subscribe to incremental data of an RDS instance, you must create a change tracking channel in the DTS console for the RDS instance. The change tracking channel pulls incremental data from the RDS instance in real time and locally stores incremental data. You can use the DTS SDK to consume incrementa l data from the change tracking channel. You can also create, manage, or delete change tracking channels in the DTS console.

A change tracking channel can be consumed by only one downstream SDK client . To subscribe to an RDS instance by using multiple downstream SDK clients, you must create an equivalent number of change tracking channels. The channels pull incremental data from the same RDS instance.

The *Table 4-3: Statuses of a change tracking channel* **table describes the statuses of a change tracking channel.**

Table 4-3: Statuses of a change tracking channel

Channel status	Description	Available operation
Prechecking	The configuration of the change tracking channel is complete and a precheck is being performed.	Delete the change tracking channel
Not Started	The change tracking channel has passed the precheck but has not been started.	· Start the change tracking channel · Delete the change tracking channel
Performing Initial Change Tracking	The initial change tracking is in progress. This process takes about one minute.	Delete the change tracking channel
Normal	Incremental data is being pulled from the source RDS instance.	· View sample code · View tracked data changes · Delete the change tracking channel

Channel status	Description	Available operation
Error	An error occurs when the change tracking channel is pulling incremental data from the source RDS instance.	· View sample code · Delete the change tracking channel

Advanced features

You can use the following advanced features that are provided for change tracking:

- Add and remove objects for change tracking

  You can add or remove the objects for change tracking.
- View tracked data changes

  You can view the incremental data in the change tracking channel.
- Modify consumption checkpoints

  You can modify consumption checkpoints.
- Monitor the change tracking channel

  You can monitor the status of the change tracking channel and receive an alert if the threshold for downstream consumption is reached. You can set the alert threshold based on the sensitivity of your businesses to consumption latency.

5 KVStore for Redis

5.1 What is KVStore for Redis?

5.1.1 What is KVStore for Redis?

KVStore for Redis is an online key-value storage service compatible with open-source Redis protocols. KVStore for Redis supports various types of data, such as strings, lists, sets, sorted sets, and hash tables. The service also supports advanced features, such as transactions, message subscription, and message publishing. Based on the hybrid storage of memory and hard disks, KVStore for Redis can provide high-speed data read/write capability and support data persistence.

As a cloud computing service, KVStore for Redis works with hardware and data deployed in the cloud, and provides comprehensive infrastructure planning, network security protections, and system maintenance services.

5.1.2 Scenarios

Game industry applications

KVStore for Redis can be an important part of the business architecture for deploying a game application.

Scenario 1: KVStore for Redis works as a storage database

The architecture for deploying a game application is simple. You can deploy a main program on an ECS instance and all business data on a KVStore for Redis instance. The KVStore for Redis instance works as a persistent storage database. KVStore for Redis supports data persistence, and stores redundant data on primary and secondary nodes.

Scenario 2: KVStore for Redis works as a cache to accelerate connections to applications

KVStore for Redis can work as a cache to accelerate connections to applications. You can store data in a Relational Database Service (RDS) database that works as a backend database.

Reliability of the KVStore for Redis service is vital to your business. If the KVStore for Redis service is unavailable, the backend database is overloaded when processing connections to your application. KVStore for Redis provides a two-node hot standby architecture to ensure high availability and reliability of services. The primary node provides services for your business. If this node fails, the system automatically switches services to the secondary node. The complete failover process is transparent.

Live video applications

In live video services, KVStore for Redis works as an important measure to store user data and relationship information.

Two-node hot standby ensures high availability

KVStore for Redis uses the two-node hot standby method to maximize service availability.

Cluster editions eliminate the performance bottleneck

KVStore for Redis provides cluster instances to eliminate the performance bottleneck that is caused by Redis single-thread mechanism. Cluster instances can effectively handle traffic bursts during live video streaming and support high-performance requirements.

Easy scaling relieves pressure at peak hours

KVStore for Redis allows you to easily perform scaling. The complete upgrade process is transparent. Therefore, you can easily handle traffic bursts at peak hours .

E-commerce industry applications

In the e-commerce industry, the KVStore for Redis service is widely used in the modules such as commodity display and shopping recommendation.

Scenario 1: rapid online sales promotion systems

During a large-scale rapid online sales promotion, a shopping system is overwhelmed by traffic. A common database cannot properly handle so many read operations.

However, KVStore for Redis supports data persistence, and can work as a database system.

Scenario 2: counter-based inventory management systems

In this scenario, you can store inventory data in an RDS database and save count data to corresponding fields in the database. In this way, the KVStore for Redis instance reads count data, and the RDS database stores count data. KVStore for Redis is deployed on a physical server. Based on solid-state drive (SSD) high-performance storage, the system can provide a high-level data storage capacity.

5.1.3 Benefits

High performance

- **Supports cluster features and provides cluster instances of 128 GB or higher to meet large capacity and high performance requirements.**
- **Provides primary/secondary instances of 32 GB or smaller to meet general capacity and performance requirements.**

Elastic scaling

- **Easy scaling of storage capacity: you can scale instance storage capacity in the KVStore for Redis console based on business requirements.**
- **Online scaling without interrupting services: you can scale instance storage capacity on the fly. This does not affect your business.**

Resource isolation

Instance-level resource isolation provides enhanced stability for individual services.

Data security

- **Persistent data storage: based on the hybrid storage of memory and hard disks, KVStore for Redis can provide high-speed data read/write capability and support data persistence.**
- **Dual-copy backup and failover: KVStore for Redis backs up data on both a primary node and a secondary node and supports the failover feature to prevent data loss.**
- **Access control: KVStore for Redis requires password authentication to ensure secure and reliable access.**
- **Data transmission encryption: KVStore for Redis supports encryption based on Secure Sockets Layer (SSL) and Secure Transport Layer (TLS) to secure data transmission.**

High availability

- **Primary/secondary structure: each instance runs in this structure to eliminate the possibility of single points of failure (SPOFs) and guarantee high availability.**
- **Automatic detection and recovery of hardware faults: the system automatica lly detects hardware faults and performs the failover operation within several seconds. This can minimize your business losses caused by unexpected hardware faults.**

Easy to use

- **Out-of-the-box service: KVStore for Redis requires no setup or installation. You can use the service immediately after purchase to ensure efficient business deployment.**
- **Compatible with open-source Redis: KVStore for Redis is compatible with Redis commands. You can use any Redis clients to easily connect to KVStore for Redis and perform data operations.**

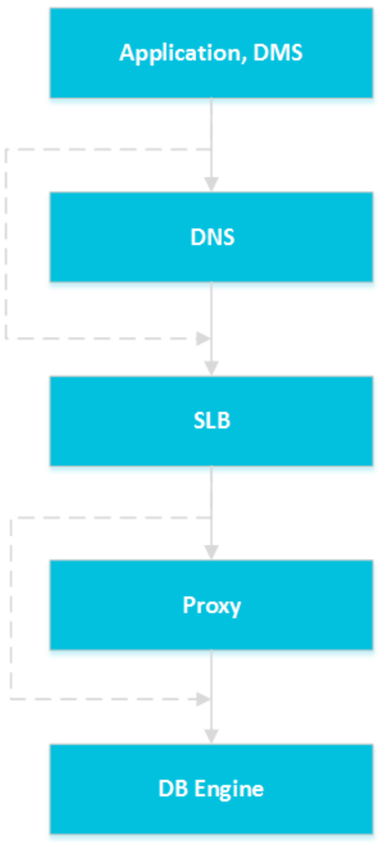
5.2 Features

5.2.1 Data link services

5.2.1.1 Overview

The data link service allows you to add, delete, modify, and search data.

You can connect to the KVStore for Redis service by using your application.



5.2.1.2 DNS

The Domain Name System (DNS) module can dynamically resolve domain names to IP addresses. Therefore, IP address changes cannot affect the performance of KVStore for Redis.

For example, the domain name of a KVStore for Redis instance is `test.kvstore.aliyun.com`, and the IP address corresponding to this domain name is `10.1.1.1`. You can connect to the KVStore for Redis instance if you add `test.kvstore.aliyun.com` or `10.1.1.1` to the connection pool of your application. If you migrate the KVStore for Redis instance to another host after a failure occurs or upgrades the instance version, the IP address may change to `10.1.1.2`. You can connect to the KVStore for Redis instance if you add `test.kvstore.aliyun.com` to the connection pool of your application. However, if you add `10.1.1.1` to the connection pool, you cannot connect to the instance.

5.2.1.3 SLB

The Server Load Balancer (SLB) module can forward traffic to available instance IP addresses. Therefore, physical server changes cannot affect the performance of KVStore for Redis.

For example, the private IP address of a KVStore for Redis instance is 10.1.1.1. The IP address of the Proxy or DB Engine module is 192.168.0.1. The SLB module forwards all traffic destined for 10.1.1.1 to 192.168.0.1. When the Proxy or DB Engine module fails, the secondary Proxy or DB Engine module with the IP address 192.168.0.2 takes over for 192.168.0.1. The SLB module redirects access traffic from 10.1.1.1 to 192.168.0.2 and the KVStore for Redis instance continues to run normally.

5.2.1.4 Proxy

The Proxy module provides some features such as data routing, traffic detection, and session persistence.

- Data routing: supports partition policies and complex queries for distributed routes based on a cluster architecture.
- Traffic detection: reduces the risks from cyberattacks that exploit Redis vulnerabilities.
- Session persistence: prevents connection interruptions in the case of failures.

5.2.1.5 DB Engine

KVStore for Redis supports standard Redis protocols of the corresponding engine versions as described in the following table.

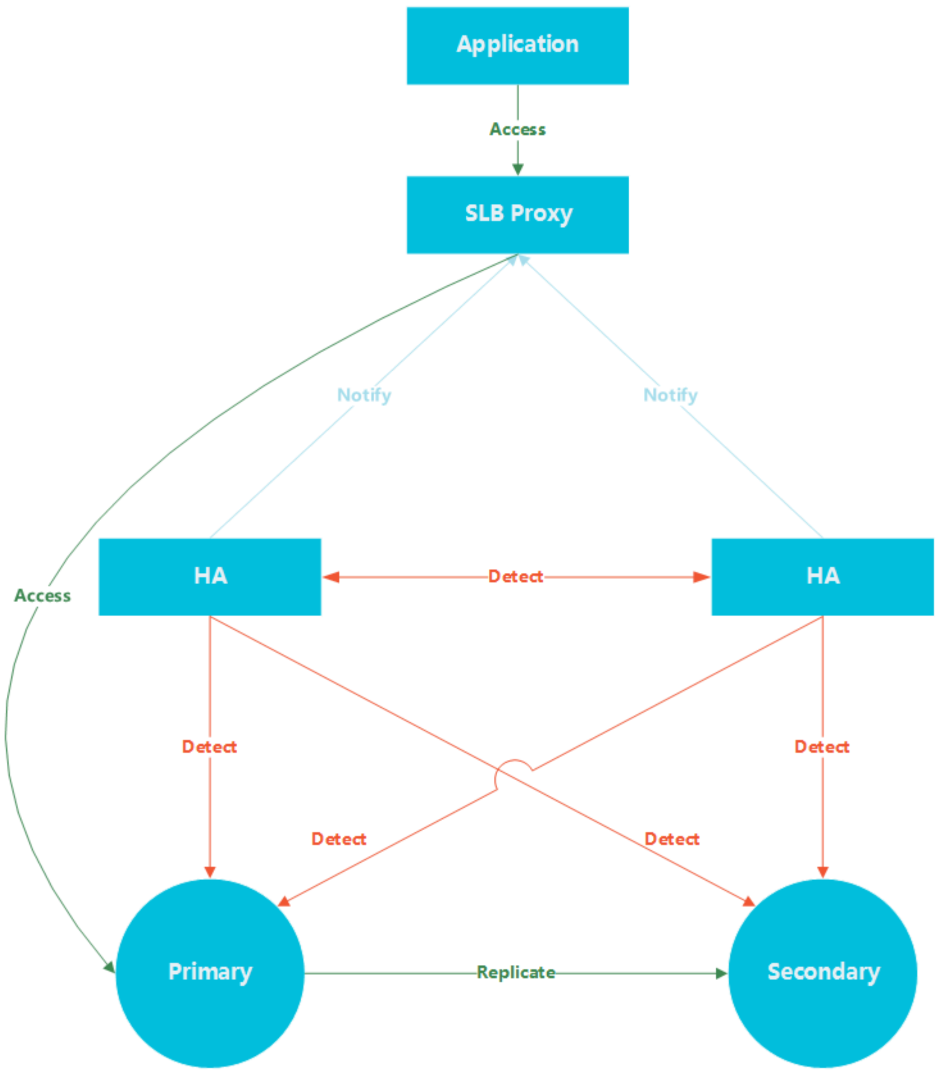
Engine	Version
Redis	Compatible with Redis 2.8 and Redis 3.0 GEO.
Redis	Redis 4.0

5.2.2 HA services

5.2.2.1 Overview

The high-availability (HA) service guarantees the availability of data link services and handles internal database exceptions.

The HA service is also highly available because this service contains multiple HA nodes.



5.2.2.2 Detection

The Detection module checks whether the primary and secondary nodes of the database engine are operating normally.

An HA node receives the heartbeat from the primary database engine node at an interval of 8 to 10 seconds. This information, combined with the heartbeat information of the secondary and other HA nodes, allows the Detection module to eliminate false negatives and positives caused by exceptions such as network jitter. As a result, switchover can be completed within 30 seconds.

5.2.2.3 Repair

The Repair module maintains replications between the primary node and the secondary node of DB Engine. This module also fixes errors that occur on either node during normal operations as follows:

- Automatically fixes exceptionally disconnected replications between these nodes .
- Automatically fixes table-level damages on both nodes.
- Automatically saves crash events and fixes the failures on both nodes.

5.2.2.4 Notice

The Notice module notifies the SLB or Proxy module of status changes of primary and secondary nodes. Therefore, you can connect to available nodes.

For example, the Detection module locates an exception on a primary node and notifies the Repair module to fix the exception. If the Repair module fails to resolve the issue, the Repair module notifies the Notice module to perform failover. Afterward, the Notice module forwards the failover request the Server Load Balancer (SLB) or Proxy module to switch all traffic to the secondary node. Meanwhile, the Repair module creates a secondary node on a different physical server and synchronizes this change to the Detection module. The Detection module checks the health status of the instance again to verify that the instance is healthy.

5.2.3 Monitoring

5.2.3.1 Service-level monitoring

The independent Service module provides service-level monitoring. The Service module of KVStore for Redis monitors features, response time, and other metrics of other dependent cloud services such as Server Load Balancer (SLB), and checks whether these services run normally.

5.2.3.2 Network-level monitoring

The Network module traces the network status. The monitoring metrics include:

- Connection conditions between ECS instances and KVStore for Redis instances.
- Connection conditions between physical servers of KVStore for Redis.
- Packet loss rates of routers and VSwitches.

5.2.3.3 OS-level monitoring

The operating system (OS) module traces status of hardware and the kernel of an operating system. The monitoring metrics include:

- Hardware inspection: the OS module regularly checks the running status of devices such as CPUs, memory modules, motherboards, and storage devices. When locating any potential hardware failures, the module automatically raises a request for repair.
- OS kernel monitoring: the OS module traces all kernel requests for databases, and analyzes the cause of a slow or error response to a request according to the kernel status.

5.2.3.4 Instance-level monitoring

The Instance module collects information of KVStore for Redis instances. The monitoring metrics include:

- Instance availability.
- Instance capacity.

5.2.4 Scheduling service

The scheduling service allocates and integrates underlying resources of KVStore for Redis, so you can activate and migrate instances.

When you create an instance in the console, the scheduling service computes and selects the most suitable physical server to handle the traffic.

After long-term operations such as instance creation, deletion, and migration, a data center generates resource fragments. The scheduling service can calculate resource fragmentation in the data center and regularly initiates resource integration to improve service performance of the data center.

6 Distributed Relational Database Service (DRDS)

6.1 What is Distributed Relational Database Service?

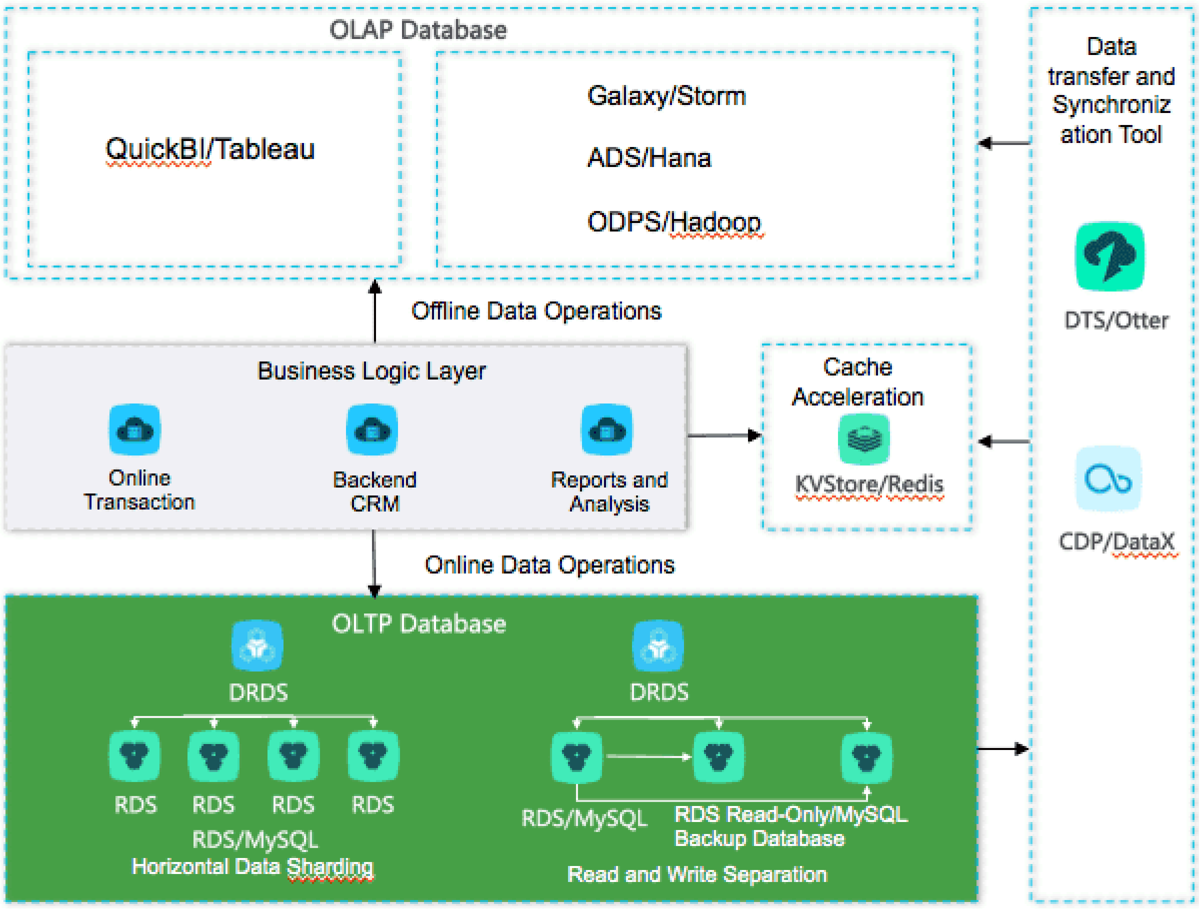
Distributed Relational Database Service (DRDS) is a middleware product independently developed by Alibaba Group for scale-out of single-instance relational databases.

DRDS is the relational database access standard for Alibaba Group. It shares the database sharding logic and team with Taobao Distributed Data Layer (TDDL). Compatible with the MySQL protocol, DRDS supports most MySQL Data Manipulati on Language (DML) and Data Definition Language (DDL) syntax. It provides the core capabilities of distributed databases, such as database sharding and table sharding, smooth scale-out, configuration changing, and transparent read/write splitting. DRDS features lightweight (stateless), flexibility, stability, and efficiency , and provides you with O&M capabilities throughout the lifecycle of distributed databases.

DRDS is mainly used for large-scale online data operations, which focuses on frontend businesses for writing data to databases. The business-specific partitioni

ng maximizes the operation efficiency, effectively meeting the high-concurrency and low-latency database operation requirements.

Figure 6-1: DRDS architecture



DRDS mainly solves the following problems:

- **Capacity bottleneck of single-instance databases: As the data volume and access volume increase, this problem cannot be completely solved by hardware upgrades. In distributed database solutions of DRDS, multiple instances work jointly, which effectively resolves the data storage capacity and access volume bottlenecks.**
- **Difficult scale-out of relational databases: Due to the inherent attributes of distributed databases, shard storage nodes can be changed through smooth data migration, supporting the dynamic scale-out of relational databases.**

6.2 Benefits

Distributed architecture

The distributed architecture allows horizontal partitioning of data and the cluster deployment of a single service. In this way, single-instance bottlenecks of Server Load Balancer (SLB), DRDS, and ApsaraDB for RDS are resolved and service scalability is achieved.

Auto scaling

DRDS instances and RDS instances can be dynamically added and removed for flexible service capabilities.

High performance

DRDS for RDS (MySQL) partitions data in specific business scenarios and clusters data based on major business operations, speeding up the response to online transactional operations. By using the columnar storage and knowledge grid, DRDS for HiStore significantly speeds up the response to common analytic operations such as large-scale data aggregation and ad hoc queries. DRDS for HiStore also helps reduce costs by achieving high compression ratio.

Secure and controllable

DRDS supports an account permission system similar to that of single-instance databases, and provides useful functions, such as the IP address whitelist and default disabling of high-risk SQL statements. It offers comprehensive API operations for support even if they need to be integrated into the local management system. We also provide complete product support and architecture services.

6.3 Architecture

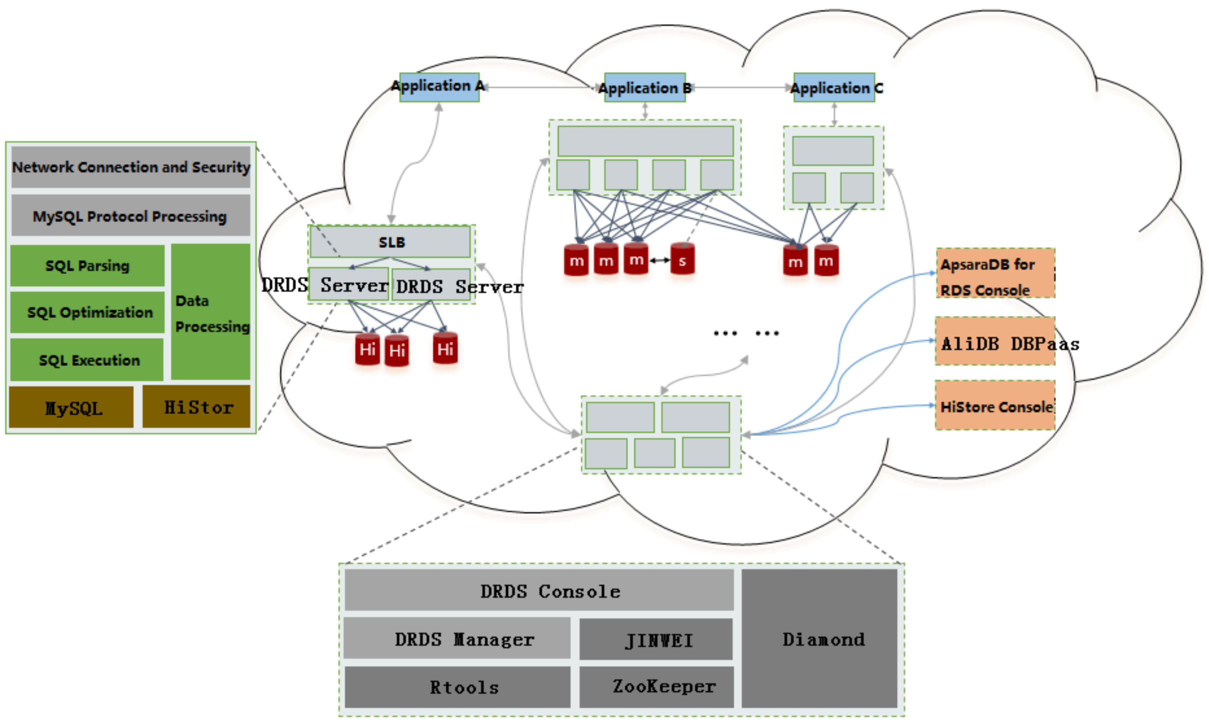
Distributed Relational Database Service (DRDS) supports two data output methods: overall output by Apsara Stack and separate output by Alibaba middleware. The two output methods differ in features and dependent components of DRDS.

The following table describes the differences between them.

Item	Overall output by Apsara Stack	Separate output by Alibaba middleware
MySQL	ApsaraDB RDS for MySQL	Alibaba Group database system (DBPaaS)
Load balancing	Centralized Server Load Balancer (SLB)	Client load balancer (VIPServer)
Special storage support	None	Storage with a high compression ratio (HiStore)

The following figure shows the system architecture of DRDS.

Figure 6-2: DRDS system architecture



DRDS server

The DRDS servers are the service layer of DRDS. Multiple DRDS servers form a cluster to provide distributed database services, including read/write splitting , routed SQL execution, result merging, dynamic database configuration, and globally unique ID (GUID).

MySQL (m and s in the figure)

ApsaraDB RDS for MySQL stores and operates online data. It implements high availability (HA) through MySQL primary-secondary replication, and implements dynamic database failover with the primary-secondary failover mechanism of ApsaraDB RDS for MySQL.

You can implement management, monitoring, alerting, and resource management in the RDS instance lifecycle in the MySQL console.

HiStore

When DRDS outputs data separately (not overall output by Apsara Stack), it uses HiStore as the physical storage. HiStore is a low-cost and high-performanc e database developed by Alibaba to support columnar storage. By using the columnar storage, knowledge grid, and multiple cores, HiStore provides higher data aggregation and ad hoc query capabilities, with lower costs than row storage (such as MySQL).

You can implement management, monitoring, alerting, and resource management in the HiStore instance lifecycle in the HiStore console.

DBPaaS

When DRDS outputs data separately (not overall output by Apsara Stack), the MySQL O&M platform DBPaaS implements management, monitoring, alerting, and resource management in the MySQL lifecycle.

SLB

You do not need to install a client on user instances. User requests are distributed through SLB. When an instance fails or a new instance is added, SLB ensures that traffic on the bound instances is distributed evenly.

VIPServer

You need to install a client on user instances, with a weak dependency on the central controller (interaction is performed only when the load configuration changes). User requests are distributed through VIPServer. When an instance fails or a new instance is added, VIPServer ensures that traffic on the bound instances is distributed evenly.

Diamond

Diamond is a system responsible for DRDS configuration storage and management . It provides the configuration storage, query, and notification functions. Diamond stores the database source data, sharding rules, and DRDS switch configuration.

Data Replication System

Data Replication System is responsible for data migration and synchronization of DRDS. Its core capabilities include full data migration and incremental data synchronization. Its derived features include smooth data import, smooth scale-out , and global secondary index (GSI). Data Replication System requires the support of ZooKeeper and DRDS Rtools.

DRDS console

The DRDS console is designed for business database administrators (DBAs) to isolate resources as required and perform operations, such as instance management, database and table management, read/write splitting configuration, smooth scale-out, monitoring data display, and IP address whitelisting.

DRDS manager

The DRDS manager is designed for global O&M personnel and DBAs. It provides the DRDS resource management and system monitoring functions:

- **Manages all resources on which RDS instances depend, including virtual machines, SLB instances, and domain names.**
- **Monitors DRDS instance statuses, including queries per second (QPS), active threads, connections, node network I/O, and node CPU usage.**

Rtools

Rtools is the O&M support system of DRDS. It allows you to manage database configuration, read/write weight, connection parameters, database and table topologies, and sharding rules.

6.4 Features

6.4.1 Horizontal partitioning (sharding)

The core principle of Distributed Relational Database Service (DRDS) is horizontal partitioning of data, where data in a logical database is distributed and stored to multiple stable ApsaraDB RDS for MySQL databases according to certain rules. The ApsaraDB RDS for MySQL databases can be distributed across multiple instances or even across data centers, but provide external services (add, delete, modify, and query operations) as a single ApsaraDB RDS for MySQL database. After partitioning , a physical database on an ApsaraDB RDS for MySQL instance is called a database shard and a physical table is called a table shard (each table shard is a part of the complete data). By moving database shards on different ApsaraDB RDS for MySQL instances, DRDS implements database scale-out and improves the overall access to and storage capacity of DRDS databases.

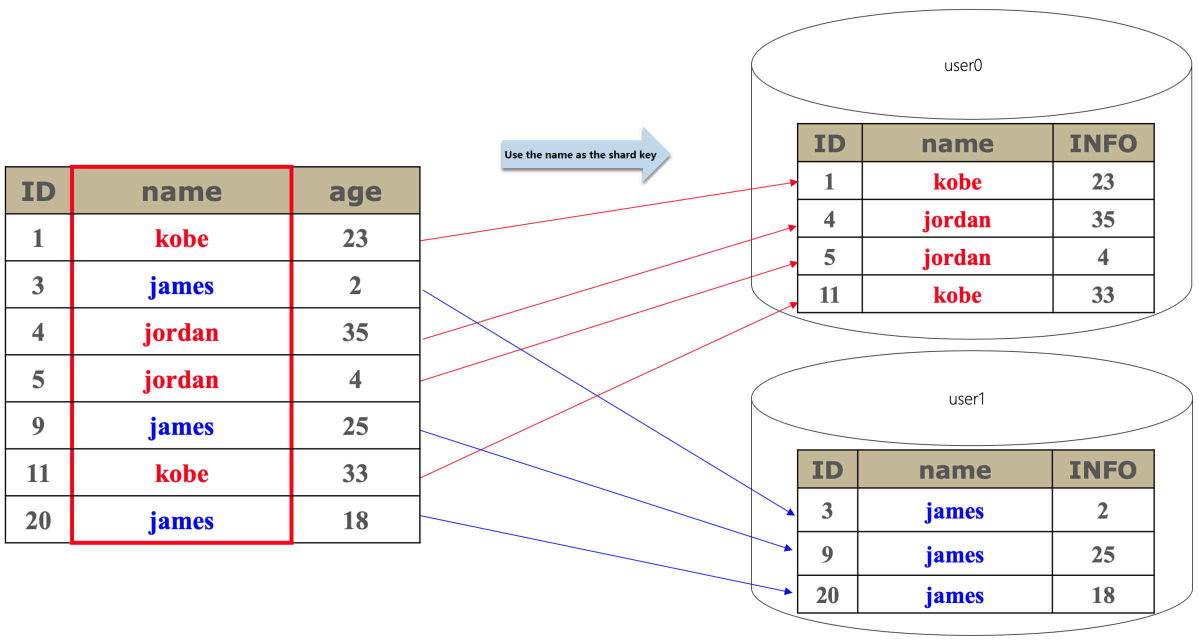
DRDS provides sharding rules, allowing you to select a partitioning policy that fits your business data characteristics. This ensures low latency for online transactio nal database operations in high-concurrency scenarios. Therefore, when DRDS is used, choosing the shard key is one of the important steps in database table structure design. The general principles are as follows:

- **DRDS is best at storing data at the frontend. Most operations of such businesses are performed based on a specific database entity. For example, the business operations of the Internet are performed for users, the business operations of Internet of Things (IoT) are performed for devices and vehicles, the business operations of banks and government agencies are performed for customers, and the business operations of e-commerce independent software vendors (ISVs) and catering ISVs are performed for merchants. The data of such businesses can be partitioned by database entity. This, combined with global secondary indexes and finally consistent transactions, can address the requirements on databases for large data volume, high concurrency, and low latency.**
- **For backend businesses, a batch of data is filtered and displayed on pages by condition and then processed and written back to the database. This is a business scenario in which DRDS can partially address the needs. In this case, a large number of single-table associations and multi-table associations may exist , multiple filtering conditions are combined for DELETE and SELECT operations , and a large number of multi-table transactions are processed. Data partitioning**

by entity is recommended for such scenarios. If database processing is tightly related to time, data can be partitioned by time.

The following figure shows how data partitioning works.

Figure 6-3: Data partitioning



ID	name	age
1	kobe	23
3	james	2
4	jordan	35
5	jordan	4
9	james	25
11	kobe	33
20	james	18

ID	name	INFO
1	kobe	23
4	jordan	35
5	jordan	4
11	kobe	33

ID	name	INFO
3	james	2
9	james	25
20	james	18

6.4.2 Smooth scale-out

To scale out a Distributed Relational Database Service (DRDS) instance, you can add ApsaraDB RDS for MySQL instances and migrate the original database shards to the new ApsaraDB RDS for MySQL instances.

DRDS scale-out principle

Follow these steps:

1. **Create a scale-out plan.**

   Select a new ApsaraDB RDS for MySQL instance and the database shards to be migrated to the new ApsaraDB RDS for MySQL instance. After the task is submitted, the system automatically creates a database and an account on the destination ApsaraDB RDS for MySQL instance and submits a task for data migration and synchronization.
2. **Perform full data migration.**

   The system selects a time point before the current time and copies and migrates all data generated before this time point.

3. Perform incremental synchronization.

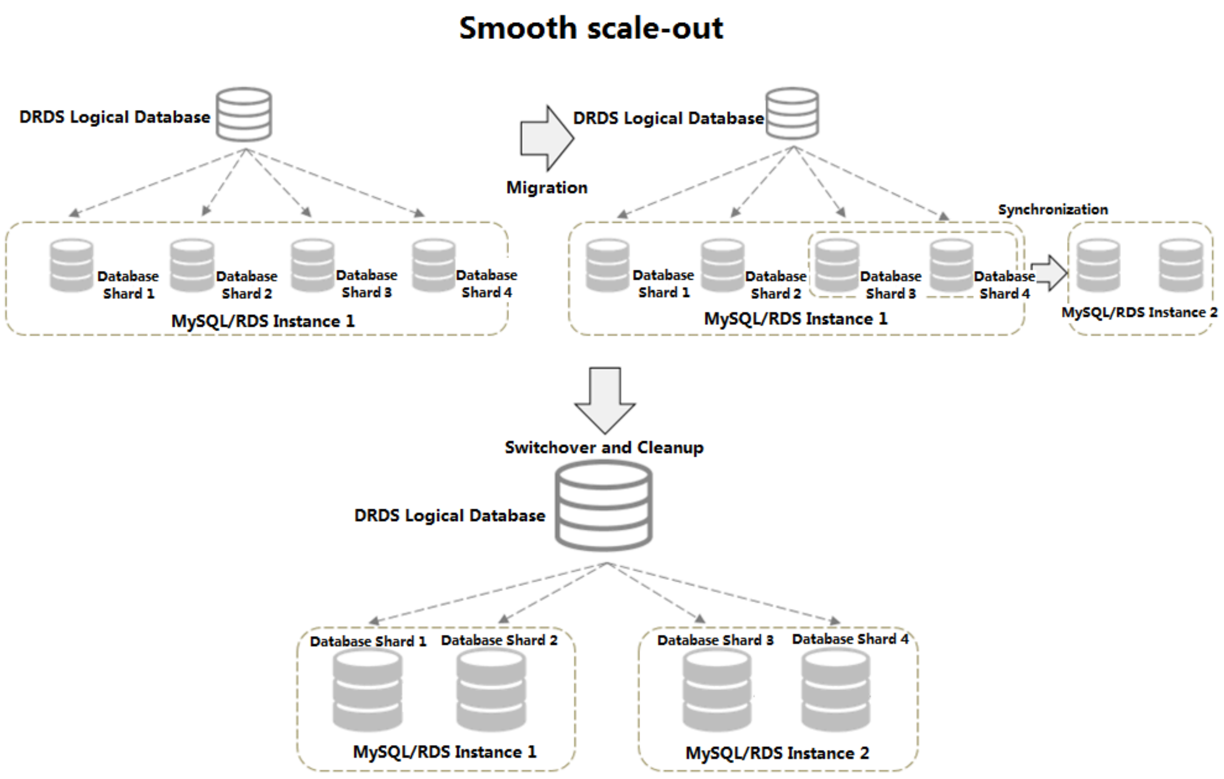
   After a full migration is completed, incremental data is synchronized according to the incremental change logs generated between a time point before the full migration and the current time, and eventually, the data is synchronized from the source database shard to the destination database shard in real time.
4. Verify data.

   When the incremental data is synchronized in quasi-real time, the system automatically performs full data verification and corrects inconsistent data caused by synchronization latency.
5. Disable the application service and switch routes.

   After verification, the incremental data is still synchronized in quasi-real time, and a specified time is selected for the failover. To ensure strict data consistency , we recommend that you disable the service (you can also not disable the service but the same data may be written at a high concurrency). The engine layer switches routes based on database sharding rules to switch subsequent traffic to the new database. The switching process can be completed within seconds.

The following figure shows data migration between database shards.

Figure 6-4: Scale-out



To ensure data security and facilitate rollback of a scale-out task, data synchroniz ation continues after the routing rule is switched. After the data O&M personnel confirm that the service is normal, you can clean up data in the source database shard in the console.

The whole scale-out process has little impact on services of the upper layer (some services may be affected if the instance type of the ApsaraDB RDS for MySQL instance is not satisfactory or its traffic pressure is high). If the service is not disabled during the failover, we recommend that you perform this operation when the database access traffic is low to reduce the possibility of concurrently updating the same data.

6.4.3 Read/write splitting

The read/write splitting function of Distributed Relational Database Service (DRDS) is a relatively transparent read traffic failover policy based on read-only ApsaraDB RDS for MySQL (MySQL) instances.

Business applications can add MySQL read-only instances and adjust their read weights in the DRDS console without code modification when they can tolerate the latency of data synchronization between read-only instances and the primary instance. The read traffic is proportionally adjusted between the MySQL primary instance and multiple MySQL read-only instances. Write operations and transaction operations are performed on the MySQL primary instance.

Note that a latency exists for data synchronization between the MySQL primary instance and MySQL read-only instances. When a large Data Definition Language (DDL) statement is executed or a large volume of data is being corrected, the latency may be over one minute. Therefore, consider whether your business can tolerate the impact before using this function.

Adding read-only instances linearly improves the read performance. For example, if there is one read-only instance, the read performance is doubled after one other read-only instance is added or tripled after two other read-only instances are added .

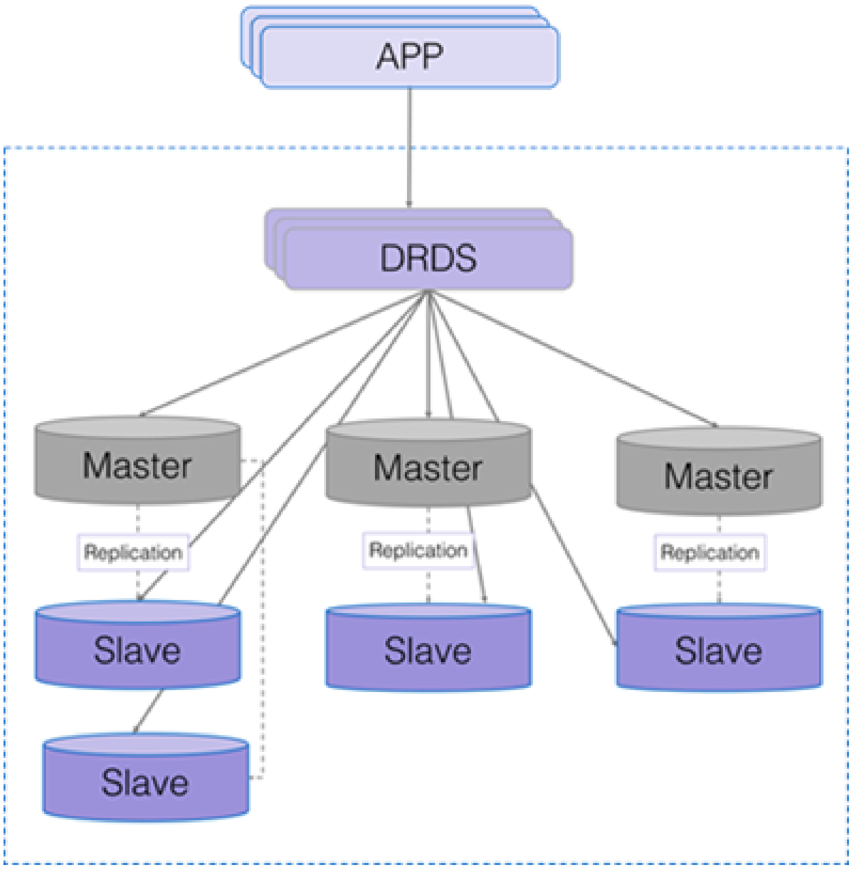
Traffic distribution and instance addition for read/write splitting

The DRDS read/write splitting function requires no modification of application code. You only need to add read-only instances and adjust the read weights in the DRDS console, to proportionally adjust the read traffic between the primary instance and multiple read-only instances. The write operations are performed on the primary instance.

Adding read-only instances linearly improves the read performance. For example, if there is one read-only instance, the read performance is doubled after one other

read-only instance is added or tripled after two other read-only instances are added , as shown in the following figure.

Figure 6-5: Traffic distribution and instance addition for read/write splitting

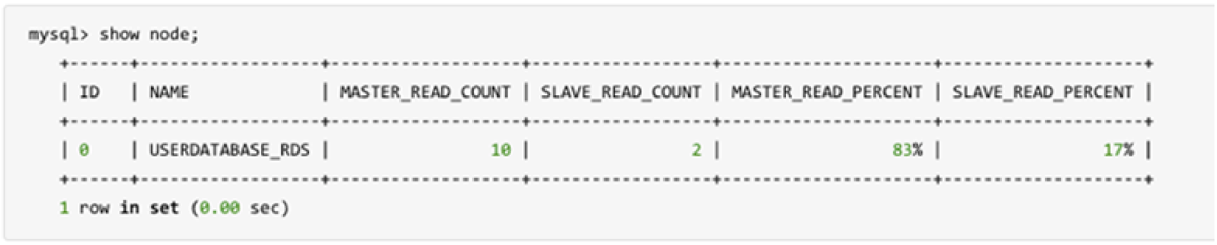


All data read on a read-only instance is asynchronously synchronized from the primary instance with a latency of milliseconds. For SQL statements that require high real-time performance, you can specify the primary instance for executing these SQL statements through DRDS hints as follows:

```
/*TDDL:MASTER/select * from tddl5_users;
```

DRDS allows you to run SHOW NODE to view the actual distribution of read traffic, as shown in the following figure.

Figure 6-6: SHOW NODE to view the actual distribution of read traffic



```
mysql> show node;
    +------+-----------------+-------------------+------------------+---------------------+--------------------+
    | ID   | NAME            | MASTER_READ_COUNT | SLAVE_READ_COUNT | MASTER_READ_PERCENT | SLAVE_READ_PERCENT |
    +------+-----------------+-------------------+------------------+---------------------+--------------------+
    | 0    | USERDATABASE_RDS |               10 |                2 |                 83% |                17% |
    +------+-----------------+-------------------+------------------+---------------------+--------------------+
    1 row in set (0.00 sec)
```

Read/write splitting in non-partition mode

The read/write splitting function of DRDS can be used independently in non-partition mode.

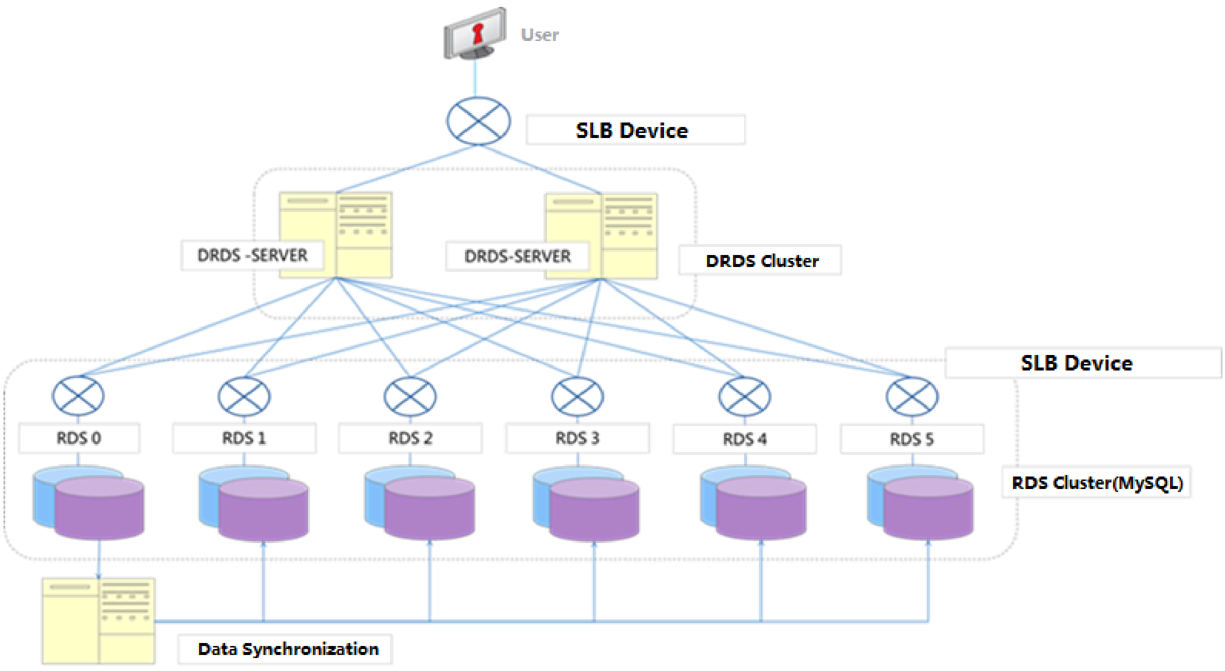
When you select an ApsaraDB for RDS instance for creating a DRDS database in the DRDS console, you can directly introduce a logical database on the ApsaraDB for RDS instance to the DRDS database for read/write splitting without data migration.

6.4.4 DRDS server scaling

Distributed Relational Database Service (DRDS) servers are deployed in clusters. A DRDS instance consists of multiple DRDS servers and provides services externally through Server Load Balancer (SLB) and Domain Name System (DNS). Multiple DRDS servers are in their own states and process external requests in a balanced manner. When the processing capability of a DRDS server cluster is insufficient, DRDS servers can be added in real time to expand the service capability. If the

resource utilization of all DRDS servers in the cluster is low, you can remove DRDS servers to achieve elastic scaling, as shown in *Figure 6-7: DRDS server scaling*.

Figure 6-7: DRDS server scaling



6.4.5 Account and permission system

The account and permission system of Distributed Relational Database Service (DRDS) is used in the same way as MySQL, but does not support authorization across multiple databases, and has fewer permissions than MySQL. The supported statements include GRANT, REVOKE, SHOW GRANTS, CREATE USER, DROP USER, and SET PASSWORD. Currently, you can grant database- and table-level permissions, but not global or column-level permissions.

Account rules:

- **The administrator account created in the console has all permissions.**
- **Only the administrator account can create accounts and grant permissions. Other accounts can only be created and granted permissions by the administra tor account.**
- **The administrator account is bound to a database and does not have permission s on other databases. It can be connected only to the bound database, and cannot grant permissions of other databases to an account. For example, the EasyDB administrator account can only connect to the EasyDB database, and can**

only grant certain permissions of the EasyDB database or tables in the EasyDB database to an account.

Currently, eight table-associated basic permissions are supported: CREATE, DROP, ALTER, INDEX, INSERT, DELETE, UPDATE, and SELECT. Among these operations:

- The TRUNCATE operation requires the table-level DROP permission.
- The REPLACE operation requires the table-level INSERT and DELETE permission s.
- The CREATE INDEX and DROP INDEX operations require the table-level INDEX permission.
- The CREATE SEQUENCE operation requires the database-level CREATE permission.
- The DROP SEQUENCE operation requires the database-level DROP permission.
- The ALTER SEQUENCE operation requires the database-level ALTER permission.
- The INSERT ON DUPLICATE UPDATE statement requires the table-level INSERT and UPDATE permissions.

6.4.6 DRDS sequence

A Distributed Relational Database Service (DRDS) sequence (a 64-digit number of the signed BIGINT type in MySQL) aims to generate a globally unique number sequence (not necessarily in increments) to generate keys such as a primary key column and a unique key.

The DRDS sequence can be implicitly used. When a table is partitioned to shards , with the primary key of auto_increment, or business data is inserted with no primary key specified, the globally unique primary key is automatically set, just like the single-instance MySQL database.

The DRDS sequence can also be explicitly used. You can run `select xxx_seq. nextval from dual where count= ?` to obtain one or more DRDS sequences for other use in the application.

6.4.7 Second-level monitoring

Distributed Relational Database Service (DRDS) allows users to run `SHOW FULL STATS` to implement second-level monitoring. This command operates with the business monitoring system of DRDS or third-party open-source monitoring software to provide better monitoring and alerting experience.

The following table describes the metrics supported by this command.

Metric	Description
QPS	The logical queries per second (QPS) specifying queries from an application to a DRDS instance.
RDS_QPS	The physical QPS specifying queries from a DRDS instance to an ApsaraDB for RDS instance.
ERROR_PER_SECOND	The number of errors per second, which is the sum of SQL syntax errors, primary key conflicts, system errors, and connectivity errors.
VIOLATION_PER_SECOND	The number of primary key or unique key conflicts per second.
MERGE_QUERY_PER_SECOND	The number of queries merged from multiple table shards.
ACTIVE_CONNECTIONS	The number of connections in use.
CONNECTION_CREATE_PER_SECOND	The number of connections created per second.
RT(MS)	The logical response time (RT) for a response from an application to a DRDS instance.
RDS_RT(MS)	The physical RT for a response from a DRDS instance to an ApsaraDB RDS for MySQL instance.
NET_IN(KB/S)	The inbound network traffic of DRDS.
NET_OUT(KB/S)	The outbound network traffic of DRDS.
THREAD_RUNNING	The number of running threads.
HINT_USED_PER_SECOND	The number of queries with hints per second.
HINT_USED_COUNT	The total number of queries with hints since startup.
AGGREGATE_QUERY_PER_SECOND	The number of aggregate queries per second.

Metric	Description
AGGREGATE_QUERY_COUNT	The total number of historical aggregate queries.
TEMP_TABLE_CREATE_PER_SECOND	The number of temporary tables created per second.
TEMP_TABLE_CREATE_COUNT	The total number of temporary tables created since startup.
MULTI_DB_JOIN_PER_SECOND	The number of cross-database JOIN queries per second.
MULTI_DB_JOIN_COUNT	The total number of cross-database JOIN queries since startup.

6.4.8 Distributed SQL engine

The distributed SQL engine of Distributed Relational Database Service (DRDS) is designed to achieve high compatibility with a single-instance MySQL database and implement SQL offload. DRDS allows you to perform SQL operations, such as SQL analysis, SQL optimization, SQL routing, and data aggregation.

The core principles of SQL offload are as follows:

- Process data as close as to the data itself.
- Reduce data transmission over the network.
- Reduce computing on DRDS and offload computing to the lower-level data nodes whenever possible.
- Make full use of the features and capabilities of database storage.

6.4.9 High-availability architecture

Automatic failover of DRDS servers

A Distributed Relational Database Service (DRDS) instance consists of multiple DRDS servers and provides services as a single connection through a load balancing service. When a DRDS server fails, its services fail over to another DRDS server in seconds. The entire failover process is transparent to users, with no need to change the application code or restart the application.

Automatic failover of read-only ApsaraDB RDS for MySQL instances

Distributed Relational Database Service (DRDS) supports the read/write splitting function. You can sign in to the DRDS console, choose DRDS Database > Read and

Write Separation, and configure the function. This function allocates some read traffic to the secondary ApsaraDB RDS for MySQL instances. DRDS identifies the read SQL requests and routes them to the primary and secondary ApsaraDB RDS for MySQL instances based on the configured ratio for read/write splitting. A secondary ApsaraDB RDS for MySQL instance allocated with only read traffic is called a read-only ApsaraDB RDS for MySQL instance.

If multiple read-only ApsaraDB RDS for MySQL instances are configured but one of them fails (the connection fails), DRDS automatically withdraws the read traffic from the failed ApsaraDB RDS for MySQL instance and re-allocates the traffic based on the read traffic ratio of the remaining normal read-only ApsaraDB RDS for MySQL instances.

The automatic failover process of read-only ApsaraDB RDS for MySQL instances is transparent to users, with no need to restart applications. When no read-only ApsaraDB RDS for MySQL instance is available, read SQL requests are still allocated proportionally to both the read-only and the primary ApsaraDB RDS for MySQL instances, but errors are reported for the read SQL requests allocated to read-only ApsaraDB RDS for MySQL instances to prevent the primary ApsaraDB RDS for MySQL instance from being overloaded.

Note:
All write SQL requests and transactions are automatically routed to the primary ApsaraDB RDS for MySQL instance for execution, regardless of the availability of read-only ApsaraDB RDS for MySQL instances.

6.4.10 Software upgrade

- Distributed Relational Database Service (DRDS) automatically provides you with new versions of installed database software.
- Software upgrade is optional. It is carried out only upon your request.
- If DRDS determines that your version has major security risks, it will instruct you to schedule the upgrade. The DRDS team will provide support during the upgrade process.
- The DRDS upgrade process is generally completed within 5 minutes. During the upgrade, there may be several transient database disconnections. There is minimal interruption to applications if the database reconnection (or a connection pool) is properly configured for applications.

6.4.11 SQL compatibility

Distributed Relational Database Service (DRDS) is compatible with the MySQL protocol and supports most MySQL query syntax, common data manipulation language (DML) syntax, and data definition language (DDL) syntax. However, the pronounced architectural differences between distributed databases and single-instance databases restrict SQL usage. The compatibility and SQL restrictions are described as follows.

Note:
Since there are many MySQL versions and the MySQL syntax and DRDS versions are updating, the compatibility discussed in this document is for reference only. Determine whether the selected MySQL version matches the business according to the actual test results.

DRDS SQL restrictions

SQL restrictions are as follows:

- **Custom data types and functions are not supported at present.**
- **Views, stored procedures, triggers, and cursors are not supported at present.**
- **Compound statements such as** `BEGIN...END, LOOP...END LOOP, REPEAT...UNTIL...END REPEAT, and WHILE...DO...END WHILE` **are not supported at present.**
- **Process control statements such as IF and WHILE are not supported at present.**

Small syntax restrictions

DDL:

- **CREATE TABLE tbl_name LIKE old_tbl_name does not support table sharding.**
- **CREATE TABLE tbl_name SELECT statement does not support table sharding.**

DML:

- **SELECT INTO OUTFILE, SELECT INTO DUMPFILE, and SELECTvar_name are not supported at present.**
- **INSERT DELAYED is not supported at present.**
- **Subqueries irrelevant to the WHERE condition are not supported at present.**
- **SQL subqueries that contain aggregation conditions are not supported at present .**

- **Variable references and operations in SQL statements are not supported at present, for example,** `SET @c=1, @d=@c+1; SELECT @c, @d`**.**

Database management:

- **SHOW WARNINGS does not support the LIMIT/COUNT combination.**
- **SHOW ERRORS does not support the LIMIT/COUNT combination.**

DRDS SQL compatibility

Compatibility with MySQL protocols:

DRDS supports mainstream clients such as MySQL Workbench, Navicat For MySQL, and SQLyog.

Note:
DRDS supports the add, delete, modify, and query operations on databases. However, other special functions (such as import and diagnosis) have not been thoroughly tested.

Compatibility with DDL statements:

- `CREATE TABLE`
- `CREATE INDEX`
- `DROP TABLE`
- `DROP INDEX`
- `ALTER TABLE`
- `TRUNCATE TABLE`

Compatibility with DML statements:

- **INSERT**
- **REPLACE**
- **UPDATE**
- **DELETE**
- **Subquery**
- **Scalar subquery**
- **Comparisons subquery**
- **Subquery with ANY, IN, or SOME**
- **Subquery with ALL**

- Subquery by column
- Subquery with EXISTS or NOT EXISTS
- Subquery in the FROM clause
- SELECT

Compatibility with PREPARE:

- PREPARE
- EXECUTE
- DEALLOCATE PREPARE

Compatibility with database management syntax

- SET
- SHOW
- KILL 'PROCESS_ID' (DRDS only supports the KILL 'PROCESS_ID' command but does not support the KILL QUERY command.)
- SHOW COLUMNS
- SHOW CREATE TABLE
- SHOW INDEX
- SHOW TABLES
- SHOW TABLE STATUS
- SHOW TABLES
- SHOW VARIABLES
- SHOW WARNINGS
- SHOW ERRORS

Notice:

Other SHOW commands are delivered to the database for processing by default, and the returned result data in different shards are not merged.

Database tool commands:

- DESCRIBE
- EXPLAIN
- USE

Custom DRDS commands:

- SHOW SEQUENCES, CREATE SEQUENCE, ALTER SEQUENCE, and DROP

- SEQUENCE [manage the globally unique numerical sequence of DRDS]
- SHOW PARTITIONS FROM TABLE [query table shard keys]
- SHOW TOPOLOGY FROM TABLE [query the physical topology of a table]
- SHOW BROADCASTS [query all broadcast tables]
- SHOW RULE [FROM TABLE] [query the table sharding rule]
- SHOW DATASOURCES [query data sources of the backend database connection pool]
- SHOW DBLOCK/RELEASE DBLOCK [define the distributed lock]
- SHOW NODE [query the database read and write traffic]
- SHOW SLOW [query the slow SQL statements]
- SHOW PHYSICAL_SLOW [query slow SQL statements executed by the physical database]
- TRACE SQL_STATEMENT/SHOW TRACE [trace the SQL statement execution process]
- EXPLAIN [DETAIL/EXECUTE] SQL_STATEMENT [analyze the SQL execution plans of DRDS and physical databases]
- RELOAD USERS [synchronize the user information from the DRDS console to the DRDS servers]
- RELOAD SCHEMA [clear data caches in the corresponding DRDS database, such as the SQL parsing, syntax tree, and table structure caches]
- RELOAD DATASOURCES [rebuild a connection pool that connects the backend to all databases]

Database functions:

- SQL statements with shard keys are supported by all MySQL functions.
- SQL statements without a shard key are supported by some functions.
- Operator functions

Function	Description
AND, &&	Logical AND
=	Assigns a value (a part of the SET statement or a part of the SET clause in the UPDATE statement).
BETWEEN... AND ...	Determines a certain range of a value.

Function	Description
BINARY	Converts a string into a binary string.
&	Bitwise AND
~	Bitwise negation
^	Bitwise XOR
DIV	Returns an integer obtained from integer division.
/	Division operator
<=>	NULL-safe equal operator
=	Equal operator
>=	Greater than or equal operator
>	Greater than operator
IS NOTNULL	Tests for a non-NULL value.
ISNOT	Tests for a non-Boolean value.
ISNULL	Tests for a NULL value.
IS	Tests for a Boolean value.
<<	Bitwise left shift operator
<=	Less than or equal operator
<	Less than operator
LIKE	Finds a specific character string matches a specified pattern .
-	Minus operator
%,	Returns the remainder of a number divided by another number.
NOTBETWEEN... AND...	Determines a certain range that a value is not in.
! =, <>	Inequality operator
NOTLIKE	Finds a specific character string that does not match a specified pattern.
NOTREGEXP	NOT operator in regular expressions
NOT, !	NOT
OR	Logical OR
+	Plus operator

Function	Description
REGEXP	Uses a regular expression for matching.
>>	Bitwise right shift operator
RLIKE	Uses a regular expression for matching.
*	Multiplication operator
-	Changes the parameter sign to take the opposite value.
XOR	Logical XOR
Coalesce	Returns the first non-NULL parameter.
GREATEST	Returns the maximum parameter value.
LEAST	Returns the minimum parameter value.
STRCMP	Compares two strings.

- Process control functions

Function	Description
CASE	Case operator
IF()	If/else structure
IFNULL()	Null if/else structure
NULLIF()	If expr1 = expr2, NULL is returned.

- Numeric functions

Function	Description
ABS()	Returns the absolute value.
ACOS()	Returns the arc cosine of a number.
ASIN()	Returns the arc sine of a number.
ATAN2()	Returns the arc tangent of two parameters.
ATAN()	Returns the arc tangent of a parameter.
CEIL()	Obtains the smallest integer greater than or equal to a number.
CEILIG()	Obtains the smallest integer greater than or equal to a number.
CONV()	Converts a number between different cardinal numbers.
COS()	Returns the cosine of a number.
COT()	Returns the cotangent of a number.

Function	Description
CRC32()	Calculates the cyclic redundancy check (CRC) value.
DEGREES()	Converts a radian to a degree.
DIV	Returns an integer obtained from integer division.
EXP()	Returns e raised to the power of the specified number.
FLOOR()	Obtains the largest integer less than or equal to a number.
LN()	Returns the natural logarithm of a parameter.
LOG10()	Returns the logarithm with the base 10 of the parameter.
LOG2()	Returns the logarithm with the base 2 of the parameter.
LOG()	Returns the natural logarithm of the first parameter.
MOD()	Returns the remainder of a number divided by another number.
%,MOD	Returns the remainder of a number divided by another number.
PI()	Returns the value of Pi.
POW()	Returns N power of the first parameter, where N is the second parameter.
POWER()	Returns N power of the first parameter, where N is the second parameter.
RADIANS()	Converts a parameter into a radian.
RAND()	Returns a random floating-point number.
ROUND()	Rounds up or down to an integer.
SIGN()	Returns the symbol of a parameter.
SIN()	Returns the sine of a parameter value.
SQRT()	Returns the square root of a parameter.
TAN()	Returns the tangent of a parameter value.
TRUNCATE(	Truncates to the specified decimal place.

- String functions

Function	Description
ASCII()	Returns the ASCII value of a character.
BIN()	Returns the binary value of a character.
BIT_LENGTH()	Returns the bit length of a string.

Function	Description
CHAR_LENGTH()	Returns the number of characters in a string.
CHAR()	Converts an input integer into a character.
CHARACTER_LENGTH()	Returns the number of characters in a string.
CONCAT_WS()	Connects the input parameters by using the specified separator.
CONCAT()	Returns a connection string.
ELT()	Returns the string at the index number.
EXPORT_SET()	-
FIELD()	Returns the index position of the first parameter in subsequent parameters.
FIND_IN_SET()	Returns the index position of the first parameter in the second parameter.
FORMAT()	Returns the formatted numbers of the specified decimal places.
HEX()	Converts an input decimal number or string into a hexadecimal number.
INSERT()	Inserts a substring of a specified number of characters at the specified place.
INSTR()	Returns the index position where the substring appears for the first time.
LCASE()	Converts to lowercase letters.
LEFT()	Returns the characters of the specified number that is the furthest left.
LENGTH()	Returns the number of bytes of a string.
LIKE	Finds a specific character string matches a specified pattern.
LOCATE()	Returns the position where the substring appears for the first time.
LOWER()	Converts to lowercase letters.
LPAD()	Pads the left side of a string with a specific set of characters.
LTRIM()	Removes spaces at the beginning.

Function	Description
MAKE_SET()	Returns a set value (a string containing substrings separated by , characters) consisting of the characters specified in the first argument.
MID()	Extracts a substring from a string (starting at the specified position).
NOTLIKE	Finds a specific character string that does not match a specified pattern.
NOTREGEXP	Performs a pattern match of a string expression against a pattern.
OCT()	Converts a number to an octal number by a string.
OCTET_LENGTH()	Returns the number of bytes of a string.
ORD()	Returns the code for the leftmost character of the given parameter.
POSITION()	Returns the position where the substring occurs for the first time.
QUOTE()	Escapes parameters for use in SQL statements.
REPEAT()	Repeats a string for a specified number of times.
REPLACE()	Replaces the specified string in all the places where it appears.
REVERSE()	Reverses characters in a string.
RIGHT()	Returns the characters of the specified number that is the furthest right.
RPAD()	Pads strings for the specified number of times from the right.
RTRIM()	Removes spaces at the end.
SPACE()	Returns a string consisting of specified spaces.
STRCMP()	Compares two strings.
SUBSTR()	Returns the specified substring.
SUBSTRING_INDEX()	Returns the substring that appears for a specified number of times and in front of a separator in a string.
SUBSTRING()	Returns the specified substring.
TRIM()	Removes spaces at the beginning and end.

Function	Description
UCASE()	Converts a string to uppercase.
UNHEX()	Returns a hexadecimal number.
UPPER()	Converts a string to uppercase.

- Time functions

Function	Description
ADDDATE()	Adds a time value (an interval) to a date.
ADDTIME()	Adds a time interval to a time/datetime and then returns the time/datetime.
CURDATE()	Returns the current date.
CURRENT_DATE()	Returns the current date.
CURRENT_TIME()	Returns the current time.
CURRENT_TIMESTAMP()	Returns the current date and time.
CURTIME()	Returns the current time.
DATE_ADD()	Adds a time value (an interval) to a date.
DATE_FORMAT()	Formats the date as required.
DATE_SUB()	Subtracts a specified time value (an interval) from a date.
DATE()	Extracts the date from the date expression or datetime expression.
DATEDIFF()	Subtracts one date from the other date.
DAY()	Returns the day (0-31) of a month for the specified date.
DAYNAME()	Returns the weekday for a date.
DAYOFMONTH()	Returns the day (0-31) of a month for the specified date.
DAYOFWEEK()	Return the day of a week (1 for Sunday and 7 for Saturday) for a date.
DAYOFYEAR()	Returns the day (1-366) of a year for a date.
EXTRACT()	Extracts a part of a date.
FROM_DAYS()	Converts a day to a date.
FROM_UNIXTIME()	Formats a UNIX timestamp as a date.

Function	Description
GET_FORMAT()	Returns a string of the date format.
HOUR()	Extracts hours from input time parameters.
LAST_DAY()	Returns the last day of the month for the parameter.
LOCALTIME()	Returns the current date and time.
LOCALTIMESTAMP, LOCALTIMESTAMP()	Returns the current date and time.
MAKEDATE()	Returns the date, containing the year and the number of days.
MAKETIME()	Constructs a time containing the hour, minute, and second.
MICROSECOND()	Returns the microsecond of a parameter.
MINUTE()	Returns the minute of a parameter.
MONTH()	Returns the month of the input date.
MONTHNAME()	Returns the name of a month.
NOW()	Returns the current date and time.
PERIOD_ADD()	Adds a period to a date containing the year and month.
PERIOD_DIFF()	Returns the number of months between two periods.
QUARTER()	Returns the quarter of the date parameter.
SEC_TO_TIME()	Converts the second into the time in 'HH:MM:SS' format.
SECOND()	Returns the second (0-59) of a minute.
STR_TO_DATE()	Converts the string to a date.
SUBDATE()	Subtracts a specified time value (an interval) from a date when three parameters are called.
SUBTIME()	Subtracts a time interval from a time/datetime and then returns the time/datetime.
SYSDATE()	Returns the function execution time.
TIME_FORMAT()	Formats the time.
TIME_TO_SEC()	Converts a parameter into a second.
TIME()	Extracts the time of an input parameter.
TIMEDIFF()	Returns the difference between two time/datetime expressions.

Function	Description
TIMESTAMP()	Returns a datetime value or expression based on a date or datetime value. If there are two arguments specified with this function, it first adds the second argument to the first, and then returns a datetime value.
TIMESTAMPADD()	Adds a time interval to the datetime expression.
TIMESTAMPDIFF()	Subtracts a time interval from the datetime expression .
UNIX_TIMESTAMP()	Returns the UNIX timestamp.
UTC_DATE()	Returns the current UTC date.
UTC_TIME()	Returns the current UTC time.
UTC_TIMESTAMP()	Returns the current UTC date and time.
WEEKDAY()	Returns the weekly index, where 1 indicates Sunday and 7 indicates Saturday.
WEEKOFYEAR()	Returns the number of the week on the calendar for a date.
YEAR()	Returns the year.

- Type conversion functions

Function	Description
BINARY	Converts a string into a binary string.
CAST()	Converts a value into a type.
CONVERT()	Converts a value into a type.

6.4.12 Table sharding

Distributed Relational Database Service (DRDS) provides convenient table sharding and changing functions, allowing you to flexibly partition tables into table shards, to glue table shards to tables, and to change table shards to table shards.

6.4.13 Multi-zone instances

Distributed Relational Database Service (DRDS) allows you to select a multi-zone DRDS instance. This ensures the DRDS instance availability when one of the zones is unavailable.

6.4.14 Zone-based disaster recovery

Distributed Relational Database Service (DRDS) provides the zone-based disaster recovery function, supporting migration between single-zone instances and dual-zone instances. Zone-based disaster recovery can be performed if an inappropriate zone is selected for the target DRDS instance or the available ApsaraDB for RDS instances in the target zone are insufficient.

7 AnalyticDB for MySQL

7.1 What is AnalyticDB for MySQL?

AnalyticDB for MySQL (originally named ADS) is an Alibaba Cloud developed real-time online analytical processing (RT-OLAP) service that enables online analytics of large amounts of data at high concurrency. It can analyze hundreds of billions of data records from multiple dimensions at millisecond-level timing to provide you with data-driven insights into your business.

Background

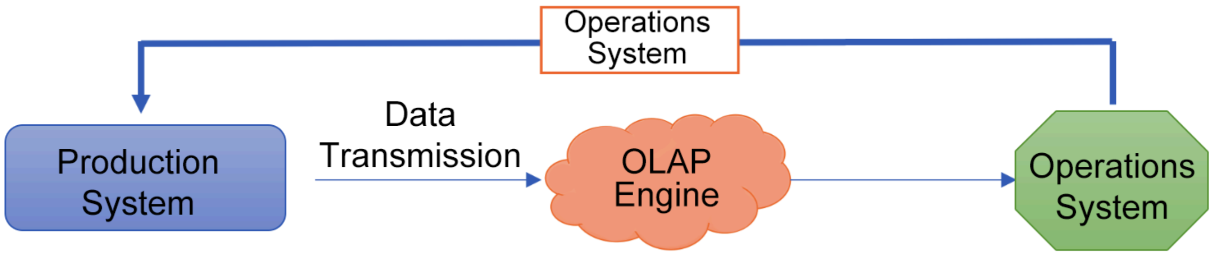
With the rapid development of IT and Internet technology, more data is being created than ever before. Two of the main characteristics of big data analysis , volume and velocity, are rapidly changing to make data processing more independent from business systems.

Online transaction processing (OLTP), such as MySQL or PostgreSQL, is typically used in business systems. OLTP is good at processing transactions and supports frequent data insertion and modification. However, OLTP cannot keep up with the increasing amount of data in the range of billions, nor can it handle complex computation procedures. In these cases, online analytical processing (OLAP) is a better solution for processing such data.

Current situation

AnalyticDB for MySQL uses SQL to build relational data warehouses. AnalyticDB for MySQL allows you to easily manage databases, scale nodes in or out, and scale specifications up or down. AnalyticDB for MySQL provides various visualization and ETL tools to simplify enterprise data processing.

AnalyticDB for MySQL can be used to refine business operations, provide real-time insights into data values, and promote continuous digital transformation for enterprises. A growing number of industries, such as logistics, transport, and new retail, use OLAP to refine their business operations and accordingly adjust production guidelines, operation efficiencies, and enterprise decisions.



AnalyticDB for MySQL supports a large number of concurrent queries and ensures high system availability through dynamic multi-copy storage and computing technology. Therefore, AnalyticDB for MySQL can serve as a backend system for end user products (including Internet products and internal enterprise analysis products). AnalyticDB for MySQL has been used in Internet business systems that have hundreds of thousands to tens of millions of users, such as Data Cube, Taobao Index, Kuaidi Dache, Alimama DMP, and Taobao Groceries.

7.2 Benefits

AnalyticDB for MySQL (originally named ADS) is an Alibaba Cloud developed real-time online analytical processing (RT-OLAP) service that enables online analytics of large amounts of data at high concurrency. It can analyze hundreds of billions of data records from multiple dimensions at millisecond-level timing to provide you with data-driven insights into your business.

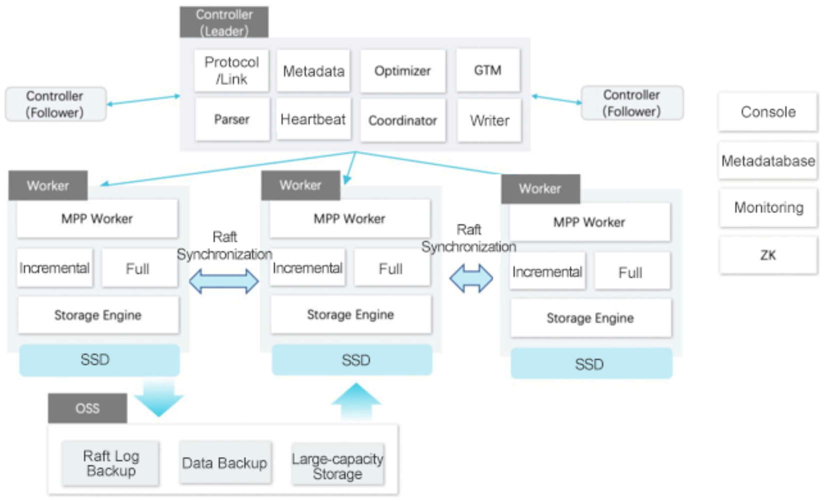
AnalyticDB for MySQL uses the Full MPP Mode (MPP) technology that features hybrid row-column storage to break through the technical barriers of OLTP and traditional data warehouses. AnalyticDB for MySQL builds high-performance and cost-effective data warehouses to process PBs of data.

AnalyticDB for MySQL is fully compatible with MySQL protocols and the SQL:2003 standard. You can migrate all your business to AnalyticDB for MySQL with few to no changes to your existing business.

7.3 Architecture

AnalyticDB for MySQL is a distributed real-time computing system based on the MPP architecture. It is constructed based on the Apsara system and integrated with distributed retrieval technology.

AnalyticDB for MySQL consists of the underlying dependencies, compute nodes, controllers, and storage nodes, as shown in the following figure.



Underlying dependencies

Underlying dependencies include the following parts:

- **Apsara system: is used to isolate resources in a virtualized manner, store data persistently, and construct data schemas and indexes.**
- **Metadatabase: refers to ApsaraDB for RDS or Table Store that stores metadata of AnalyticDB for MySQL.**

  Note:
  Metadata is not involved in actual computations.
- **Apache ZooKeeper module: performs distributed coordination among components.**

Controllers

A controller is used to control the allocation of database resources in compute nodes and distribution of compute resources. It can also manage compute nodes and tasks running in the database background. A controller consists of multiple modules:

- **SLB: manages grouping and load balancing of controllers.**
- **Client access manager.**
- **SQL parser.**
- **AnalyticDB for MySQL console.**

AnalyticDB for MySQL supports the following clients, drivers, programming languages, and middleware:

- **Clients and drivers that support MySQL 5.1, 5.5, or 5.6 protocols: MySQL 5.1.x Connector/J, MySQL 5.3.x Connector/ODBC, and MySQL 5.1.x, 5.5.x, or 5.6.x client.**
- **Programming languages: JAVA, Python, C/C++, Node.js, PHP, and R (RMySQL).**
- **Middleware: Websphere Application Server 8.5, Apache Tomcat, and JBoss.**

Compute nodes

Compute nodes carry out computing tasks issued by controllers to read, filter, merge, and compute data.

Storage nodes

Storage nodes are responsible for writing data, saving data to disk storage, and copying data between nodes. Storage nodes support data backup and restoration.

7.4 System features

Compatibility with MySQL

- **Supports MySQL and standard JDBC and ODBC interfaces.**
- **Supports several MySQL development tools such as the DMS console, MySQL command line client, DBeaver, Navicate, and SQL Workbench/J.**
- **Uses relational models to store data and provides SQL statements to flexibly compute and analyze data. No data modeling is required in advance.**
- **Supports mainstream data types to display numbers, characters, dates, and binary data.**

INSERT, UPDATE, and DELETE operations

AnalyticDB for MySQL provides concurrent INSERT and DELETE operations on one or more tables.

- **Allows you to perform INSERT and DELETE operations on the real-time tables with defined primary keys.**
- **Provides multiple mechanisms to ensure that the written data is not lost. Both REPLACE INTO/INSERT OVERWRITE and INSERT IGNORE INTO statements are supported.**

- Provides `INSERT INTO...SELECT FROM` statements.

AnalyticDB for MySQL provides SELECT operations on one or more tables.

- Over 90% compatible with standard MySQL queries and provides column mapping methods such as expressions, functions, aliases, column names, and CASE WHEN.
- Provides clauses such as FROM table name AS alias and JOIN table name AS alias.
- Provides JOIN operations between real-time tables, between real-time tables and dimension tables, and between ON conditions.
- Provides subqueries with up to three levels, JOIN operations between subqueries under specific conditions, and queries with IN operators for dimension table data.

  Note:
  MPP engines allow you to query with IN operators for data of both dimension and real-time tables.

- Provides WHERE clauses combined with AND and OR operators, function expressions, or BETWEEN and IS operators.
- Provides GROUP BY operations for multiple columns and aliases generated from column mapping expressions such as CASE WHEN.
- Provides ORDER BY operations for expressions and columns in either ascending or descending order.
- Provides common aggregate functions and HAVING operations.
- Provides COUNT(DISTINCT) operations for multiple columns that contain hash partition columns, and for any columns when Full MPP Mode is used.
- Provides operators such as UNION, UNION ALL, MINUS, and INTERSECT for multiple SELECT statements.

Mainstream data types and diversified OLAP functions

AnalyticDB for MySQL provides mainstream data types and a wide range of OLAP functions to display numbers, characters, dates, and binary data.

Hybrid row-column storage

AnalyticDB for MySQL provides hybrid row-column storage for single tables to handle hybrid load scenarios.

- Detail query in OLTP: requires to read and write the detailed data of an entire row by using a single SELECT statement. AnalyticDB for MySQL can quickly return query results at low I/O costs.
- Large-scale multi-dimensional analysis in OLAP: includes statistical analysis and JOIN operations for large amounts of data, typically for several columns in a wide table. AnalyticDB for MySQL has load processing capabilities in OLAP scenarios.
- High throughput: ApsaraDB for MySQL allows you to write hundreds of billions of data entries in real time daily.

High data compression rate

AnalyticDB for MySQL provides adaptive compression algorithms. AnalyticDB for MySQL can automatically select an optimal algorithm at an up to 1:20 compressio n ratio and provide DML operations in data compression status based on different data distribution methods and data types.

Data import

AnalyticDB for MySQL provides multiple methods to import data.

- Allows you to batch import files in parallel by using multiple nodes.
- Allows you to import data in the format of CSV files, TEXT files, or files with multiple delimiters.
- Allows you to import data by using multiple nodes in real-time streaming mode.

Data export

AnalyticDB for MySQL allows you to specify a destination and batch export data in parallel by using multiple nodes.

- Provides SELECT statements to output query results.
- Provides DUMP DATA statements to quickly export large amounts of data to OSS and MaxCompute.

Load balancing management

AnalyticDB for MySQL integrates with Alibaba Cloud SLB to implement load balancing. Failures will go unnoticed by clients. Even if two or more servers fail, AnalyticDB for MySQL can still receive connection requests from clients.

7.5 Unique features

7.5.1 Full-text indexing

This topic describes the features of full-text indexing in AnalyticDB for MySQL.

Full-text indexing has the following features:

- AnalyticDB for MySQL supports the SQL-92 standard and MySQL protocols. AnalyticDB for MySQL provides full-text retrieval based on SQL statements, which greatly reduces learning costs. Additionally, AnalyticDB for MySQL unifies common structured data analysis with flexible unstructured data analysis and uses the same SQL language to operate multiple types of data, which significantly reduces development costs.
- AnalyticDB for MySQL can perform integrated retrieval and multi-modal analysis for structured and unstructured data. Most existing solutions in the industry only focus on creating full-text indexes on text data to retrieve unstructured data, and not structured data. AnalyticDB for MySQL not only supports full-text retrieval, but also provides a variety of classic index structures in traditional databases, such as B-tree index, bitmap index, and inverted index. You can use multiple indexes within the same table to meet a variety of retrieval requiremen ts.
- AnalyticDB for MySQL provides comprehensive distributed computing capabiliti es. Existing services such as Elasticsearch and Solr cannot provide distribute d JOIN solutions. AnalyticDB for MySQL uses the MPP and DAG architecture to provide distributed JOIN, GROUP BY, and aggregation capabilities such as COUNT(DISTINCT). AnalyticDB for MySQL also provides computation based on partition and non-partition keys.

7.5.2 Data consistency

AnalyticDB for MySQL provides strong data consistency. Data that is written or updated can take effect immediately.