ApsaraDB for RDS

Product Introduction

Product Introduction

What is RDS

What is RDS

ApsaraDB for RDS (Relational Database Service) is a stable and reliable online database service, and it also supports elastic scaling function. Based on the Apsara distributed system and high-performance storage of ephemeral SSD, RDS supports MySQL, SQL Server, and PostgreSQL engines. It offers a complete set of solutions for backup, recovery, monitoring, migration, disaster recovery, and troubleshooting database operation and maintenance.

ApsaraDB for MySQL

MySQL is the world’ s most popular open source database. As an important part of LAMP, a combination of open source software (Linux + Apache + MySQL + Perl/PHP/Python), MySQL is widely used in a variety of applications.

In the Web 2.0 era, MySQL serves as the basis of the underlying architecture of the popular BBS software system Discuz and blogging platform WordPress. In the Web 3.0 era, leading Internet companies including Alibaba, Facebook, and Google have built their large-scale mature database clusters by taking advantage of the advanced flexibility of MySQL.

Based on Alibaba’ s MySQL source code branch, ApsaraDB for MySQL proves to have excellent performance and throughput. It withstands the massive data traffic and large number of concurrent users during many November 11 (Singles’ Day) shopping festivals - the Chinese equivalent of Cyber Monday. ApsaraDB for MySQL also offers a range of advanced functions including optimized read/write splitting, data compression, and intelligent optimization.

RDS for MySQL currently supports versions 5.5, 5.6, and 5.7. For specific support details, see the Quick Start guide of your corresponding MySQL version.

ApsaraDB for SQL Server

SQL Server is one of the first commercial databases and is an important part of the Windows platform

(IIS + .NET + SQL Server), with support for a wide range of enterprise applications. The SQL Server Management Studio software comes with a rich set of built-in graphical tools and script editors.

Powered by high-availability architecture and anytime data recovery capabilities, ApsaraDB for SQL Server provides strong support for a variety of enterprise applications. It also covers Microsoft’ s licensing fee without any additional cost required.

RDS for SQL Server currently supports the following versions:

SQL Server 2008 R2 Enterprise

SQL Server 2012 Web, Standard, and Enterprise

SQL Server 2016 Web, Standard, and Enterprise

ApsaraDB for PostgreSQL

PostgreSQL is the world’ s most advanced open source database. As an academic relational database management system, what sets PostgreSQL apart is that its full compliance with SQL specifications and robust support for a diverse range of data formats (including JSON, IP, and geometric data, which are not supported by most commercial databases).

ApsaraDB for PostgreSQL supports a range of features including transactions, subqueries, Multi-Version Concurrency Control (MVCC), and data integrity verification. It also integrates a number of important functions, including high availability and backup recovery, to help mitigate your operation and maintenance burden.

RDS for PostgreSQL currently supports version 9.4.

ApsaraDB for PPAS

Postgres Plus Advanced Server (PPAS) is a stable, secure, and scalable enterprise-level relational database. Based on PostgreSQL, PPAS delivers enhanced performance, application solutions, and compatibility, and provides the capability to run Oracle applications directly. It is a reliable and cost-effective option for running a variety of enterprise applications.

ApsaraDB for PPAS incorporates a number of advanced functions including account management, resource monitoring, backup recovery, and security controls, and it continues to be updated and improved regularly.

RDS for PPAS currently supports version 9.3.

Benefits

Cheap and ease-to-use

Simple deployment

You can customize RDS specifications through Alibaba Cloud’ s official website or the API. After the order is confirmed, RDS generates the specified instance instantly.

RDS can work with ECS to reduce the application response time and save on public traffic fees.

On-demand upgrades

Initially, you can purchase an RDS instance that meets the existing business requirements. When requirements on the database and data storage capacity change, you can flexibly adjust the instance specifications without any interruptions to the service.

Effortless migration

RDS is used similarly to the native database engine, meaning that you can transfer the pre-existing knowledge and skills to RDS management. Data can be migrated to RDS using the commercial off-the-shelf data import and export tools with minimal labor cost required.

Ease of management

Alibaba Cloud is responsible for ensuring the normal operation of RDS through routine maintenance and management, such as hardware/software fault processing and database update patches. You can independently perform database addition, deletion, restart, backup, recovery, and other management operations in the Alibaba Cloud console.

High performance

High performance

Parameter optimization

Alibaba Cloud has accumulated years of experience in production and optimization by gathering key opinions from top database experts in China and aggregating performance data on all the RDS instances. DBA continuously manages RDS over its lifecycle to ensure that RDS is running at the optimal performance.

SQL optimization

Based on your application scenario, RDS locks low-efficiency SQL statements and offers recommendations for optimizing your business code.

High-end backend hardware

All servers used by RDS go through multiple levels of service verification by multiple parties to ensure the exceptional performance and stability.

High security

Anti-DDoS attack

> **Notice:** We recommend that RDS instances are accessed over the intranet to avoid DDoS attacks.

When Internet connection is used to access RDS instances, the risk of DDoS attacks occurring on the network is possible. If this occurs, the RDS security system enables flow cleaning operation first. If the flow cleaning operation fails or the attack reaches the black hole threshold, black hole processing is triggered.

The following describes how flow cleaning and black hole processing work and when they are triggered:

Flow cleaning:

This applies only to inbound traffic from the Internet. During this process, the RDS instance can be normally accessed. Flow cleaning is triggered if a single ApsaraDB instance meets any of the following conditions:

Package Per Second (PPS) reaches 30,000.

Bits Per Second (BPS) reaches 180 Mbps.

The number of concurrent connections created per second reaches 10,000.

The number of concurrent active connections reaches 10,000.

The number of concurrent inactive connections reaches 10,000.

The system automatically triggers and terminates flow cleaning.

Black hole processing:

This only applies to inbound traffic from the Internet. Black hole processing guarantees security of the overall RDS service by blocking malicious attacks. During this process, RDS instances and the services cannot be accessed from the Internet. Black hole processing is triggered if the following conditions are met:

BPS reaches 2 Gbps.

Flow cleaning is ineffective.

The black hole is removed automatically after 2.5 hours.

Access control policy

You can define the IP addresses that are allowed to access RDS. IP addresses that have not been specified are denied access.

Each account can only view and operate its own database.

System security

RDS is protected by multiple firewall layers that can effectively block a variety of malicious attacks and guarantee data security.

Direct logon to the RDS server is not allowed. Only the port required by the specific database service is open.

The RDS server cannot initiate an external connection. It can only accept access requests.

Professional support team

Alibaba Cloud’ s security team provide rapid security technology support for RDS.

High reliability

High reliability

Hot standby

RDS uses hot standby, so if a physical server fails, the service is switched over in seconds without interruptions to application services.

Multi-copy redundancy

The data in RDS server is built on RAID, and the data backup is stored on OSS.

Data backup

RDS offers an automatic backup mechanism. You can set a backup schedule or initiate a temporary backup at any time.

Data recovery

Data can be recovered from a backup. Generally, data can be recovered within 7 days to a temporary RDS instance. After the data is verified, the data can be migrated back to the master RDS instance.

System architecture

Data link service

Data link service

ApsaraDB provides all of the data link services, including DNS, SLB, and Proxy. Because RDS uses the NativeDB Engine, and database operations are highly similar across engines, no learning cost is generated for users who are familiar with these data link services.

DNS

The DNS module supports the dynamic resolution of domain names to IP addresses, to prevent IP address changes from affecting the performance of RDS instances. After its domain name is configured in the connection pool, an ApsaraDB instance continues to be accessed even if the corresponding IP address changes.

For example, the domain name of an ApsaraDB instance is test.rds.aliyun.com, and the IP address corresponding to this domain name is 10.10.10.1. If either test.rds.aliyun.com or 10.10.10.1 is configured in the connection pool of a program, the instance can be accessed. After performing a zone migration or version upgrade for this ApsaraDB instance, the IP address may change to 10.10.10.2. If the domain name configured in the connection pool is test.rds.aliyun.com, the instance can still be accessed. However, if the IP address configured in the connection pool is 10.10.10.1, the instance is no longer accessible.

SLB

The SLB module provides instance IP addresses (including both intranet and Internet IP addresses) to prevent physical server changes from affecting the performance of RDS instances.

For example, the intranet IP address of an RDS instance is 10.1.1.1, and the corresponding Proxy or DB Engine runs on 192.168.0.1. Normally, the SLB module redirects all traffic destined for 10.1.1.1 to 192.168.0.1. If 192.168.0.1 fails, another address in hot standby status, 192.168.0.2, takes over for 192.168.0.1. In this case, the SLB module redirects all traffic destined for 10.1.1.1 to 192.168.0.2, and the RDS instance continues to offer its services normally.

Proxy

The Proxy module performs a number of functions including data routing, traffic detection, and session holding.

Data routing: This supports distributed complex query aggregation for big data and provides the corresponding capacity management.

Traffic detection: This reduces SQL injection risks and supports SQL log backtracking when necessary.

Session holding: This prevents database connection interruptions if any failure occurs.

DB engine

RDS fully supports mainstream database protocols, as shown in the following table:

Database type	Version
MySQL	5.1 (Deprecated), 5.5, 5.6, 5.7
SQL Server	2008 R2, 2012, 2016
PostgreSQL	9.4
PPAS	9.3, highly compatible with Oracle

High-availability service

The high-availability service consists of several modules including the Detection, Repair, and Notification modules. In combination, these modules guarantee the availability of the data link services and process any internal database exception.

In addition, RDS can improve the performance of its high-availability service by migrating to a region that supports multiple zones and by adopting the appropriate high-availability policies.

Detection

The Detection module checks whether the master and slave nodes of the DB Engine offer their services normally. The HA (High Available) node uses heartbeat information, acquired at an interval of

8 to 10 seconds, to check the health status of the master node. This information, combined with the health status of the slave node and heartbeat information from other HA nodes, allows the Detection module to eliminate any risk of misjudgment caused by exceptions such as network jitter and allows that the exception switchover can be completed within 30 seconds.

Repair

The Repair module maintains the replication relationship between the master and slave nodes of the DB Engine. It can also repair any errors that may occur on either node, such as:

Automatic restoration of master/slave replication in case of disconnection

Automatic repair of table-level damage to the master or slave nodes

On-site saving and automatic repair if the master or slave nodes crash

Notice

The Notice module informs the SLB or Proxy of status changes to the master and slave nodes to guarantee that you can continue to access the correct node.

For example, the Detection module discovers that the master node has an exception and instructs the Repair module to fix it. If the Repair module fails to resolve the problem, it directs the Notification module to initiate traffic switching. The Notification module then forwards the switching request to the SLB or Proxy, which begins to redirect all traffic to the slave node. Simultaneously, the Repair module creates a new slave node on another physical server and synchronizes this change back to the Detection module. The Detection module then incorporates this new information and starts to recheck the health status of the instance.

Multi-zone

Multi-zone refers to the physical area that is formed by combining multiple individual zones within the same region. Multi-zone RDS instances can withstand higher level disasters than single-zone instances. For example, a single-zone RDS instance can withstand server and rack failures, while a multi-zone RDS instance can survive a situation such as failure of an entire data center.

Currently no extra charge for multi-zone RDS instances is generated. Users in a region where multi-zone is enabled can purchase multi-zone RDS instances directly or convert single-zone RDS instances into multi-zone RDS instances by using inter-zone migration.

Note: Multiple zones may have a certain amount of network latency. As a result, when a multi-

zone RDS instance uses a semi-synchronous data replication solution, its response time to any individual update may be longer than that of a single-zone instance. In this case, the best way to improve overall throughput is to increase concurrency.

High-availability policies

The high-availability policies use a combination of service priorities and data replication modes to meet the business needs.

The service priorities are as follows:

RTO (Recovery Time Objective) priority: The database must restore services as soon as possible within a specified time frame. This is best for users who require their databases to provide uninterrupted online service.

RPO (Recovery Point Objective) priority: The database must guarantee the data reliability, that is, as little data loss as possible. This is best for users whose highest priority is data consistency.

There are three data replication modes:

Asynchronous replication (Async)

In this mode, the master node does not immediately synchronize data to the slave node. When an application initiates an update request, which may include add, delete, or modify operations, the master node responds to the application immediately after completing the operation but does not necessarily replicate that data to the slave node right away. This means that the operation of the primary database is not affected if the slave node is unavailable, but data inconsistencies may occur if the master node is unavailable.

Forced synchronous replication (Sync)

In this mode, the master node synchronizes all data to the slave node at all times. When an application initiates an update request, which may include add, delete, or modify operations, the master node replicates the data to the slave node immediately after completing the operation and waits for the slave node to return a success message before it responds to the application. This means that the operation of the master node is affected if the slave node is unavailable, but the data on the master and slave nodes is always consistent.

Semi-synchronous replication (Semi-Sync)

This functions as a hybrid of the two preceeding replication modes. In this mode, when both

nodes are functioning normally, data replication is identical to the forced synchronous replication mode. However, when there is an exception, such as the slave node becoming unavailable or a network exception occurring between the two nodes, the master node only attempts to replicate data to the slave node and suspend its response to the application for a set period of time. Once the replication mode has timed out, the master node degrades to asynchronous replication. At this point, if the master node becomes unavailable and the application updates its data from the slave node, it is consistent with the data on the master node. When data replication between the two nodes returns to normal, because the slave node or network connection is recovered, forced synchronous replication is reinstated. The amount of time it takes for the nodes to return to forced synchronous replication depends on how the semi-synchronous replication mode was implemented. For instance, ApsaraDB for MySQL 5.5 is different from ApsaraDB for MySQL 5.6 in this regard.

Several combinations of service priorities and data replication modes are available to meet your database and business needs. The characteristics of key combinations are detailed in the following table.

Cloud data engine	Service priority	Data replication mode	Combination characteristics
MySQL 5.1	RPO	Async	- If the master node fails, the slave node switches over after applying all of the relay logs. - If the slave node fails, application operations on the master node are not affected. The data on the master node is synchronized after the slave node

			recovers.
MySQL 5.5	RPO	Async	- If the master node fails, the slave node switches over after applying all of the relay logs. - If the slave node fails, application operations on the master node are not affected. The data on the master node is synchronized after the slave node recovers.
MySQL 5.5	RTO	Semi-Sync	- If the master node fails and data replication degrads, RDS immediately triggers the switchover and direct traffic to

			the slave node because data consistency is guaranteed. - If the slave node fails, application operations on the master node times out, and data replication degrades to asynchronous replication. After the slave node recovers and the data on the master node is synchronized completely, data replication returns to forced synchronization. - If the master node fails while the two nodes have

			inconsistent data and the data replication mode degrads to asynchronous replication, the slave node switches over after applying all of the relay logs.
MySQL 5.6	RPO	ASync	- If the master node fails, the slave node switches over after applying all of the relay logs. - If the slave node fails, application operations on the master node are not affected. The data on the master node is synchronized after the slave node

			recovers.
MySQL 5.6	RTO	Semi-Sync	- If the master node fails and data replication degrads, RDS immediately triggers the switchover and direct traffic to the slave node because data consistency is guaranteed. - If the slave node fails, application operations on the master node times out, and data replication degrades to asynchronous replication. After the slave node recovers and the data on the master

			node is synchronized completely, data replication returns to forced synchronization. - If the master node fails while the two nodes have inconsistent data and the data replication mode degrads to asynchronous replication, the slave node switches over after applying all of the relay logs.
MySQL 5.6	RPO	Semi-Sync	- If the master node fails and data replication has not degraded, RDS immediatel

			y triggers the switchover and direct traffic to the slave node because data consistency has been guaranteed. - If the slave node fails, application operations on the master node times out, and data replication degrades to asynchrono us replication. When the slave node can obtain information from the master node again, because the slave node or network connection recovers, data replication returns to forced synchroniza

			tion. - If the master node fails while the two nodes have inconsistent data and the data difference on the slave node cannot be reconciled completely, you can obtain the time of the slave node through the API. Then you can decide when to switchover and which method you plan to use to reconcile the data.
MySQL 5.7	X	X	Currently this engine does not support policy adjustments.
SQL Server 2008 R2	X	X	Currently this engine does not support policy adjustments.
SQL Server 2012	X	X	Currently this engine does not support policy adjustments.
PostgreSQL	X	X	Currently this engine does not support policy adjustments.

PPAS	X	X	Currently this engine does not support policy adjustments.

Backup recovery service

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

Backup

The Backup module compresses and uploads the data and logs on both the master and slave nodes.

RDS uploads this data to OSS by default, but the backup files can also be dumped to a cheaper and more persistent Archive Storage. Normally, backup is initiated on the slave node so as not to affect the performance of the master node. However, if the slave node is unavailable or damaged, the Backup module initiates backup on the master node.

Recovery

The Recovery module recovers a backup file stored on OSS to the target node.

Master node roll-back: This can be used to restore a node to the state that it was in at a specific point in time.

Slave node repair: This can be used to automatically create a new slave node to reduce risks when an irreparable failure occurs to the slave node.

Read-only instance creation: This creates a read-only instance from a backup.

Storage

The Storage module is responsible for uploading, dumping, and downloading backup files.

Currently all backup data is uploaded to OSS for storage, and users can obtain temporary links to download their data as needed. The Storage module also supports dumping backup files from OSS to an Archive Storage for cheaper and steady offline storage.

Monitoring service

Monitoring service

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

Service

The Service module tracks service-level statuses. It monitors whether the SLB, OSS, Archive Storage, Log Service, and other cloud products on which RDS depends are normal, including their functionality and response time. It also uses logs to determine whether the internal RDS services are operating normally.

Network

The Network module tracks statuses at the network layer. It monitors the connectivity between ECS and RDS and between physical RDS servers, as well as the rates of packet loss on the router and switch.

OS

The OS module tracks statuses at the hardware and OS kernel layer, including:

Hardware overhaul: Constantly checks the operational status of the CPU, memory, main board, and storage, pre-judges whether a fault will occur, and automatically submits a repair report in advance.

OS kernel monitoring: Tracks all database calls and uses kernel statuses to analyze the reasons for slowdowns or call errors.

Instance

The Instance module collects RDS instance-level information, including:

Available instance information.

Instance capacity and performance indicators.

Instance SQL execution records.

Scheduling service

The scheduling service consists of a Resource module and a Version module. It mainly implements resource allocation and instance version management.

Resource

The Resource module allocates and integrates the underlying RDS resources. For example, when you create an instance through the RDS console or API, the Resource module determines which physical server is best suited to carry traffic. This module also allocates and integrates the underlying resources required for the inter-zone migration of RDS instances. After lengthy instance creation, deletion, and migration operations, the Resource module calculates the degree of resource fragmentation in a zone and initiates resource integration regularly to improve the service carrying capacity of the zone.

Version

The Version module is applicable to version upgrades of RDS instances. For example:

MySQL major version upgrade: Upgrade MySQL 5.1 to MySQL 5.5, and upgrade MySQL 5.5 to MySQL 5.6.

MySQL minor version upgrade: Fix a bug in the MySQL source code.

Migration service

Migration service

The migration service can migrate data from a local database to ApsaraDB, or migrate an ApsaraDB instance to another instance. ApsaraDB includes a Data Transfer Service (DTS) tool to facilitate quick database migration.

DTS is a cloud data transfer service for efficient instance migration in local databases, and between RDS instances. For more information, see DTS product overview.

DTS provides three migration modes, that is, structural migration, full migration, and incremental migration. The detailed introduction is as follows:

Structural migration: DTS migrates the structural definitions of migration objects to the target instance. Currently, this mode can migrate tables, views, triggers, stored procedures, and stored functions.

Full migration: DTS migrates all existing data in the source database migration objects to the target instance.

> **Note:** To guarantee data consistency, non-transaction tables that do not have a Primary Key are locked during full migration. Locked tables cannot be written to, and the duration of the lock depends on the volume of data in a table. The locks are released only after the non-transaction tables without a Primary Key are fully migrated.

Incremental migration: DTS synchronizes data changes made in the migration process to the target instance.

> **Note:** If a DDL operation is performed during data migration, structural changes are not synchronized to the target instance.

Product series

General introduction to product series

Currently, RDS instances are divided into three series: Basic Edition, High-availability Edition and Finance Edition. Different series support different engine types and instance types. See Instance type list for more information.

Note: Currently the Finance Edition is applicable only to the regions in China.

Brief introductions

Series	Introduction	Use cases
Basic Edition	It uses the storage-computing isolated architecture and a single computing node, which realizes super high cost effectiveness.	- Personal learning. - Micro websites. - Development and testing environments for and small and medium-sized enterprises.
High-availability Edition	It uses the classic high-availability architecture with one master node and one slave node. The ephemeral SSD storage provides the best and balanced performance.	- Production databases of large enterprises. - Applications for the industries of Internet, Internet of things (IoT), retail e-commerce sales, logistics, game, and other industries.
Finance Edition	It uses the three-node architecture with one master node and two slave nodes. This architecture guarantees strong data consistency through synchronizing multiple log copies and provides financial-level data reliability and cross-IDC disaster tolerance.	Core databases with extremely high data security requirements in the financial, securities, insurance and other industries.

Feature differences

Feature	Basic Edition	High-availability Edition	Finance Edition
Engine	- MySQL 5.7 - SQL Server 2012 Web, Enterprise - SQL Server 2016 Web	- MySQL 5.5/5.6/5.7 - SQL Server 2008 R2 Enterprise - SQL Server 2012 Standard, Enterprise - SQL Server 2016 Standard, Enterprise - PostgreSQL 9.4 - PPAS 9.3	MySQL 5.6
Number of nodes	1	2	≥3
Type configuration	Maximum 32-core 128 GB/2 TB	Maximum 60-core 470 GB/3 TB	Maximum 60-core 470 GB/3 TB
Monitoring and alert	Supported	Supported	Supported
IP whitelist	Supported	Supported	Supported
Backup and recovery	Supported	Supported	Supported
Parameter settings	Supported	Supported	Supported
SSL and TDE	Not supported	Supported (Currently MySQL 5.7 does not support TDE)	Supported
Log management	Not supported	Supported	Supported
Performance optimization	Not supported	Supported	Supported
Read-only instance (Additional instances required)	Not supported	Supported only by MySQL 5.6/5.7	Supported
Read/write splitting	Not supported	Supported only in MySQL 5.6	Supported
Built-in read/write	Not supported	Not supported	Supported (Phase II)

splitting			
SQL auditing	Not supported	Supported (Additional payment required)	Supported (For free)
High-frequency monitoring	Not supported	Supported (Additional payment required)	Supported (For free)

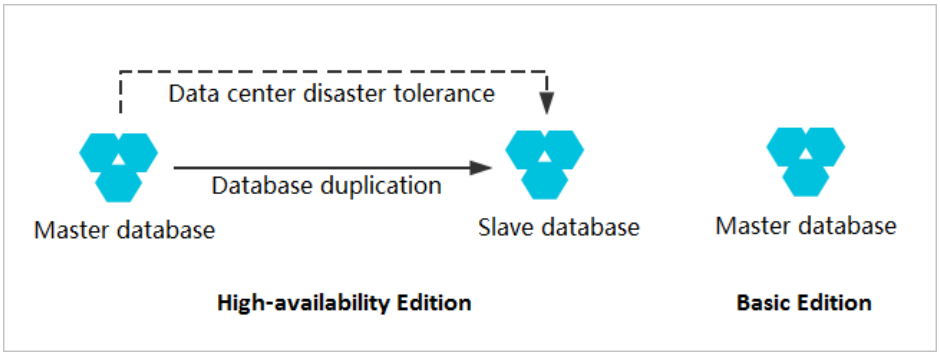
Basic Edition

Basic Edition

General introduction

The Basic Edition is a new series of ApsaraDB for RDS, of which the architecture is deployed on a single database node. Compared with the mainstream master/slave High-availability Edition, the Basic Edition contains only one node and has no slave node for fault recovery. Currently, both MySQL and SQL Server support this new series.

The following picture shows the architecture of the Basic Edition and High-availability Edition.



Comparative advantages

The slave database in the High-availability Edition is used only for failover but does not provide services, and the database duplication adds performance cost to the master database. In this respect, the Basic Edition is not inferior but superior to the High-availability Edition in terms of performance.

The Basic Edition of RDS uses the underlying data distributed storage layer to guarantee the stability

of multiple replicas, that is, the fault in or damage to one physical node does not result in data loss. Besides, the costs can be greatly saved by reducing one database node, making the price of the Basic Edition half that of the High-availability Edition.

> **Note:** Because there is only one database node, when the node fails, it takes longer to recover the node. Therefore, the High-availability Edition is recommended for sensitive businesses that have higher requirements on database availability.

Restrictions

Compared with the High-availability Edition, the Basic Edition does not support the following functions:

- Master/slave switchover
- Zone switch
- Safe connection mode
- Log management
- Performance diagnosis
- Read-only instances
- Disaster tolerance instances

Instance type

Instance type overview

The instances of ApsaraDB for RDS are now available in different types: the common instance, the dedicated instance, and the dedicated host. To distinguish from the instance type available earlier

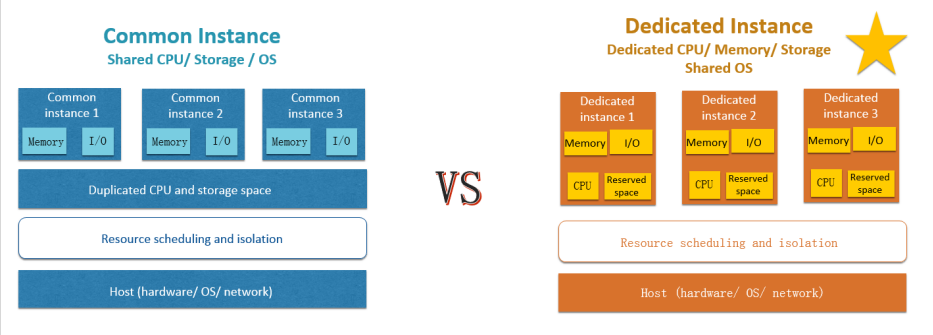
(before January 2017) which we defined as common instance, the new instance type is defined as dedicated instance. The dedicated host can be deemed as the top-level configurations of the dedicated instance. The dedicated host has been so upgraded that it is equivalent to a single full physical machine. Therefore, its resources are dedicated too.

The following list shows the features and applicable scenarios of various instance types:

Type	Description	Applicable scenarios
Common instance	- It is a highly cost-effective instance type which maximizes utilization by reusing resources and allows you to enjoy the benefits of scale. - Its storage capacity is not linked with CPU/memory, allowing for flexible configurations.	- It is applicable to price-sensitive customers. - It is applicable for scenarios where high-performance stability is not required.
Dedicated instance/ dedicated host	- It is a new RDS instance type which features fixed computing capabilities, storage space and IO performance. - A number of fully dedicated CPU cores and threads are assigned to it to ensure long-term stability in computing performance. - It has a storage space reserved for higher stability. - The dedicated host is the dedicated instance type with	It is applicable for business scenarios where a database-centric system is used, such as finance, e-commerce, government, medium and large -size Internet businesses.

	top-level configurations.	

The differences between the common and dedicated instances are shown in the following figure.



Dedicated instance

Product introduction

As a new instance type of ApsaraDB for RDS, the dedicated instance features fixed computing capabilities, storage space and IO performance. It differs from other instance types in terms of resource allocation strategies used. With more stable performance, a dedicated instance is the best option for business scenarios where a database-centric system is used, such as finance, e-commerce, government, medium and large -size Internet businesses.

The dedicated instances are available in many types. For more information, see Instance type list. The dedicated host is the top-level configuration of dedicated instance. You can change the configuration of a instance type at will. You are free of restrictions on instance types, which means configuration changes are allowed across different instance types.

Design principles and performance features

With the help of the operating system (Linux/Windows) kernel, RDS isolates computing resources on physically different instances. A dedicated instance uses a slightly different CPU allocation strategy from a common instance. A number of fully dedicated CPU cores and threads are assigned to the dedicated instance to ensure long-term stability and predictability in computing performance. This avoids noisy neighborhood on physical machines.

A dedicated instance has a reserved storage space. As compared with a common instance, the dedicated instance can fully avoid instance migration across physical machines as a result of the disk capacity increase on your instance or other instances to provide higher stability. Furthermore, the dedicated instance supports hot standby, so you can fail over at any time in case of disk failure on one instance, guaranteeing the availability of the instance. After the fail-over, you can hot swap the faulty host transparently yet unawares.

Contrast benifits

It is not likely to make direct comparison between dedicated and common instances, because their metrics do not fully match. However, to help make purchase decisions, we select two similar instance types for a cost performance analysis.

Type	Type ID	CPU/Memory	Disk space	Maximum number of connections	Maximum IOPS	Monthly price (effective before January 2017)
Common instance	rds.mysql.m1.medium	4-core 16 GB	500 GB	4000	7000	2,100 RMB
Dedicated instance	mysql.x8.large.2	4-core 32 GB	500 GB	5000	9000	3,650 RMB

The above table shows that the dedicated instance costs 70% more than the common instance, but it offers twice the memory capacity as the common instance, 25% more in the maximum number of connections, and 28% more in the maximum IOPS. In addition, it has stable CPU computing performance. Therefore, the dedicated instance provides higher overall cost performance in applicable business scenarios. For more information about the prices of the dedicated instances, see Pricing.

Instance type list

Instance type list

RDS for MySQL

Series	Version	Type	Type	CPU/M	Maximu	Maximu	Storage

			code	emory	m number of connections	m IOPS	size
Basic series	5.7	Common instance	mysql.n1.micro.1	1 C 1 GB	2000	IOPS=min{30*Storage size, 20000}	20 GB - 1000 GB
			mysql.n2.small.1	1 C 2 GB	2000		
			mysql.n2.medium.1	2 C 4 GB	4000		
			mysql.n4.medium.1	2 C 8 GB	6000		20 GB - 2000 GB
			mysql.n4.large.1	4 C 16 GB	8000		
			mysql.n4.xlarge.1	8 C 32 GB	10,000		
			mysql.n4.2xlarge.1	16 C 64 GB	15,000		
			mysql.n4.4xlarge.1	32 C 128 GB	20,000		
			mysql.n8.4xlarge.1	32 C 256 GB	64,000		
			mysql.n4.8xlarge.1	56 C 224 GB	64,000		
			mysql.n8.8xlarge.1	56 C 480 GB	64,000		
High-availability series	5.5/5.6/5.7	Common instance	rds.mysql.t1.small	1 C 1 GB	300	600	5 GB - 2000 GB
			rds.mysql.s1.small	1 C 2 GB	600	1000	
			rds.mysql.s2.large	2 C 4 GB	1200	2000	

			rds.mysql.s2.xlarge	2 C 8 GB	2000	4000	
			rds.mysql.s3.large	4 C 8 GB	2000	5000	
			rds.mysql.m1.medium	4 C 16 GB	4000	7000	
			rds.mysql.c1.large	8 C 16 GB	4000	8000	
			rds.mysql.c1.xlarge	8 C 32 GB	8000	12,000	
			rds.mysql.c2.xlarge	16 C 64 GB	16,000	14,000	5 GB - 3000 GB
			rds.mysql.c2.xlp2	16 C 96 GB	24,000	16,000	
		Dedicated instance (large Memory)	mysql.x8.medium.2	2 C 16 GB	2500	4500	250 GB
			mysql.x8.large.2	4 C 32 GB	5000	9000	500 GB
			mysql.x8.xlarge.2	8 C 64 GB	10,000	18,000	1000 GB
			mysql.x8.2xlarge.2	16 C 128 GB	20,000	36,000	2000 GB or 3000 GB
		Dedicated instance (more CPUs)	mysql.x4.large.2	4 C 16 GB	2500	4500	250 GB or 500 GB
			mysql.x4.xlarge.2	8 C 32 GB	5000	9000	500 GB or 1000 GB
			mysql.x4.2xlarge.2	16 C 64 GB	10,000	18,000	1000 GB, 2000 GB or 3000 GB
			mysql.x4.4xlarg	32 C 128 GB	20,000	36,000	2000 GB or

			e.2				3000 GB
		Dedicated Host	rds.mysql.st.d13	30 C 220 GB	64,000	20,000	3000 GB
			rds.mysql.st.h43	60 C 470 GB	100,000	50,000	3000 GB
Finance series (formerly known as Enterprise Multi-node Edition)	5.6	Dedicated Instance (more CPUs)	mysql.x4.large.3	4 C 16 GB	2500	4500	250 GB or 500 GB
			mysql.x4.xlarge.3	8 C 32 GB	5000	9000	500 GB or 1000 GB
			mysql.x4.2xlarge.3	16 C 64 GB	10,000	18,000	1000 GB, 2000 GB or 3000 GB
			mysql.x4.4xlarge.3	32 C 128 GB	20,000	36,000	2000 GB or 3000 GB
		Dedicated instance (large memory)	mysql.x8.medium.3	2 C 16 GB	2500	4500	250 GB
			mysql.x8.large.3	4 C 32 GB	5000	9000	500 GB
			mysql.x8.xlarge.3	8 C 64 GB	10,000	18,000	1000 GB
			mysql.x8.2xlarge.3	16 C 128 GB	20,000	36,000	2000 GB or 3000 GB
			mysql.x8.4xlarge.3	32 C 256 GB	40,000	72,000	3000 GB
		Dedicated Host	mysql.st.8xlarge.3	60 C 470 GB	100,000	120,000	3000 GB
Read-only instance	5.6/5.7	Common instance	rds.mysql.t1.small	1 C 1 GB	300	600	5 GB - 2000 GB
			rds.mysql.s1.sm	1 C 2 GB	600	1000	

			all				
			rds.mysql.s2.large	2 C 4 GB	1200	2000	
			rds.mysql.s2.xlarge	2 C 8 GB	2000	4000	
			rds.mysql.s3.large	4 C 8 GB	2000	5000	
			rds.mysql.m1.medium	4 C 16 GB	4000	7000	
			rds.mysql.c1.large	8 C 16 GB	4000	8000	
			rds.mysql.c1.xlarge	8 C 32 GB	8000	12,000	
			rds.mysql.c2.xlarge	16 C 64 GB	16,000	14,000	5 GB - 3000 GB
			rds.mysql.c2.xlp2	16 C 96 GB	24,000	16,000	
		Dedicated instance (large memory)	mysqlro.x8.medium.1	2 C 16 GB	2500	4500	250 GB
			mysqlro.x8.large.1	4 C 32 GB	5000	9000	500 GB
			mysqlro.x8.xlarge.1	8 C 64 GB	10,000	18,000	1000 GB
			mysqlro.x8.2xlarge.1	16 C 128 GB	20,000	36,000	2000 GB or 3000 GB
		Dedicated instance (more CPUs)	mysqlro.x4.large.1	4 C 16 GB	2500	4500	250 GB or 500 GB
			mysqlro.x4.xlarge.1	8C 32GB	5000	9000	500 GB or 1000 GB
			mysqlro.x4.2xlarge.1	16 C 64 GB	10,000	18,000	1000 GB, 2000

							GB or 3000 GB
			mysqlro.x4.4xlarge.1	32C 128GB	20,000	36,000	2000 GB or 3000 GB
		Dedicated Host	rds.mysql.st.d13	30 C 220 GB	64,000	20,000	3000 GB

RDS for SQL Server

Series	Version	Type	Type code	CPU/Memory	Maximum number of connections	Maximum IOPS	Storage size
Basic series	2012 Enterprise (formerly known as 2012)	Common instance	rds.mssql.s2.large	2 C 4 GB	Not limited	IOPS=min{30*Storage size, 20000}	20 GB - 2000 GB
			rds.mssql.s2.xlarge	2 C 8 GB			
			rds.mssql.s3.large	4 C 8 GB			
			rds.mssql.m1.medium	4 C 16 GB			
			rds.mssql.c1.large	8 C 16 GB			
			rds.mssql.c1.xlarge	8 C 32 GB			
			rds.mssql.c2.xlarge	16 C 64 GB			
	2012/2016 Web	Dedicated instance	mssql.x2.medium.w1	2 C 4 GB	Not limited	IOPS=min{30*Storage size, 20000}	20 GB - 2000 GB
			mssql.x2.large.w1	4 C 8 GB			

			mssql.x2.xlarge.w1	8 C 16 GB			
			mssql.x2.2xlarge.w1	16 C 32 GB			
			mssql.x4.medium.w1	2 C 8 GB			
			mssql.x4.large.w1	4 C 16 GB			
			mssql.x4.xlarge.w1	8 C 32 GB			
			mssql.x4.2xlarge.w1	16 C 64 GB			
High-availability series	2008 R2 Enterprise	Common instance	rds.mssql.s1.small	1 C 2 GB	600	1000	10 GB - 2000 GB
			rds.mssql.s2.large	2 C 4 GB	1200	2000	
			rds.mssql.s2.xlarge	2 C 8 GB	2000	4000	
			rds.mssql.s3.large	4 C 8 GB	2000	5000	
			rds.mssql.m1.medium	4 C 16 GB	4000	7000	
			rds.mssql.c1.large	8 C 16 GB	4000	8000	
			rds.mssql.c1.xlarge	8 C 32 GB	8000	12000	
			rds.mssql.c2.xlarge	16 C 64 GB	16000	14000	
			rds.mssql.c2.xlp2	16 C 96 GB	24000	16000	
		Dedicated	mssql.x8.mediu	2 C 16 GB	2500	4500	250 GB

		instance	m.2				
			mssql.x8.large.2	4 C 32 GB	5000	9000	500 GB
			mssql.x8.xlarge.2	8 C 64 GB	10000	18000	1000 GB
			mssql.x8.2xlarge.2	16 C 128 GB	20000	36000	2000 GB
		Dedicated host	rds.mssql.st.d13	30 C 220 GB	64000	20000	2000 GB
			rds.mssql.st.h43	60 C 470 GB	100000	50000	2000 GB
	SQL Server 2012/2016 Enterprise	Dedicated instance	mssql.x4.xlarge.e2	8 C 32 GB	Not limited	Depends on SSD cloud disk performance	20 GB - 2 TB
			mssql.x4.2xlarge.e2	16 C 64 GB			
			mssql.x4.3xlarge.e2	24 C 96 GB			
			mssql.x8.xlarge.e2	8 C 64 GB			
			mssql.x8.2xlarge.e2	16 C 128 GB			
			mssql.x8.4xlarge.e2	32 C 256 GB			
			mssql.x8.7xlarge.e2	56 C 480 GB			
	SQL Server 2012/2016 Standard	Dedicated instance	mssql.x4.medium.s2	2 C 8 GB			
			mssql.x4.large.s2	4 C 16 GB			
			mssql.x4.xlarge.s2	8 C 32 GB			

			mssql.x4.2xlarge.s2	16 C 64 GB			
			mssql.x4.3xlarge.s2	24 C 96 GB			

RDS for PostgreSQL

Series	Version	Type	Type code	CPU/Memory	Maximum number of connections	Maximum IOPS	Storage size
High-availability series	9.4	Common instance	rds.pg.t1.small	1 C 1 GB	100	600	5 GB - 2000 GB
			rds.pg.s1.small	1 C 2 GB	200	1000	
			rds.pg.s2.large	2 C 4 GB	400	2000	
			rds.pg.s3.large	4 C 8 GB	800	5000	
			rds.pg.c1.large	8 C 16 GB	1500	8000	
			rds.pg.c1.xlarge	8 C 32 GB	2000	12,000	
			rds.pg.c2.xlarge	16 C 64 GB	2000	14,000	
		Dedicated instance (large memory)	pg.x8.medium.2	2 C 16 GB	2500	4500	250 GB
			pg.x8.large.2	4 C 32 GB	5000	9000	500 GB
			pg.x8.xlarge.2	8 C 64 GB	10,000	18,000	1000 GB
			pg.x8.2xlarge.2	16 C 128 GB	12,000	36,000	2000 GB
		Dedicated instance (more CPUs)	pg.x4.large.2	4 C 16 GB	2500	4500	250 GB or 500 GB
			pg.x4.xlarge.2	8 C 32 GB	5000	9000	500 GB or 1000 GB
			pg.x4.2	16 C 64	10,000	18,000	1000

			xlarge.2	GB			GB or 2000 GB
			pg.x4.4 xlarge.2	32 C 128 GB	12,000	36,000	2000 GB or 3000 GB
		Dedicated Host	rds.pg.st.d13	30 C 220 GB	4000	20,000	3000 GB
			rds.pg.st.h43	60 C 470 GB	4000	50,000	3000 GB

RDS for PPAS

Series	Version	Type	Type code	CPU/Memory	Maximum number of connections	Maximum IOPS	Storage size
High-availability series	9.3	Common instance	rds.ppas.t1.small	1 C 1 GB	100	600	5 GB - 2000 GB
			rds.ppas.s1.small	1 C 2 GB	200	1000	
			rds.ppas.s2.large	2 C 4 GB	400	2000	
			rds.ppas.s3.large	4 C 8 GB	800	5000	
			rds.ppas.m1.medium	4 C 16 GB	1500	8000	
			rds.ppas.c1.xlarge	8 C 32 GB	2000	12,000	
			rds.ppas.c2.xlarge	16 C 64 GB	2000	14,000	
		Dedicated instance	ppas.x8.medium.2	2 C 16 GB	2500	4500	250 GB
			ppas.x8.large.2	4 C 32 GB	5000	9000	500 GB

			ppas.x8.xlarge.2	8 C 64 GB	10,000	18,000	1000 GB
			ppas.x8.2xlarge.2	16 C 128 GB	12,000	36,000	2000 GB
		Dedicated Host	rds.ppas.st.d13	30 C 220 GB	4000	20,000	3000 GB
			rds.ppas.st.h43	60 C 470 GB	4000	50,000	3000 GB

History instance type

RDS for MySQL

The history instance type of RDS for SQL Server is in the following table. It’ s not provide for creating the new RDS instance. Recommend the user to use the latest specifications.

Type code	CPU(Core)	Memory (MB)	Maximum number of connections	Maximum IOPS
rds.mys2.small	2	240 MB	60	150
rds.mys2.mid	4	600 MB	150	300
rds.mys2.standard	6	1200 MB	300	600
rds.mys2.large	8	2400 MB	600	1200
rds.mys2.xlarge	9	6000 MB	1500	3000
rds.mys2.2xlarge	10	12000 MB	2000	6000
rds.mys2.4xlarge	11	24000 MB	2000	12,000
rds.mys2.8xlarge	13	48000 MB	2000	14,000

RDS for SQL Server

The history instance type of RDS for SQL Server is in the following table. It’ s not provide for creating the new RDS instance. Recommend the user to use the latest specifications.

Type code	CPU(Core)	Memory (MB)	Maximum number of connections	Maximum IOPS
rds.mss1.small	6	1000 MB	100	500

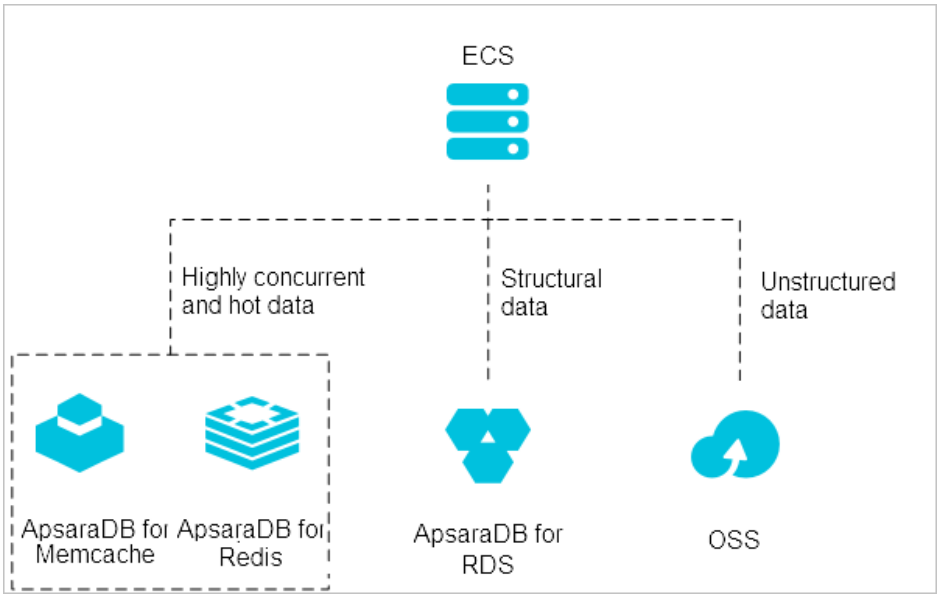
rds.mss1.mid	8	2000 MB	200	1000
rds.mss1.standard	9	4000 MB	400	2000
rds.mss1.large	10	6000 MB	600	3000
rds.mss1.xlarge	11	8000 MB	800	4000
rds.mss1.2xlarge	12	12000 MB	1200	6000
rds.mss1.4xlarge	13	24000 MB	2000	12,000
rds.mss1.8xlarge	13	48000 MB	2000	14,000

Typical application

Diversified data storage

Diversified data storage

RDS supports diversified storage extension through Memcache, Redis, and OSS storage products.



Cache data persistence

ApsaraDB for Memcache and ApsaraDB for Redis can be used together to form a high-throughput and low-latency storage solution, with a request delay of only a few milliseconds and a cache that supports a higher QPS (queries per second) than standard RDS.

For more information, see Cache data persistence.

Multi-structure data storage

OSS is an Alibaba Cloud storage service that features massive capacity, robust security, low cost, and high reliability. RDS and OSS can work together to form multiple data storage solutions.

For example, when RDS and OSS are used in a forum, resources such as the images of registered users or those posted on the forum can be stored in OSS to reduce the storage pressure on RDS.

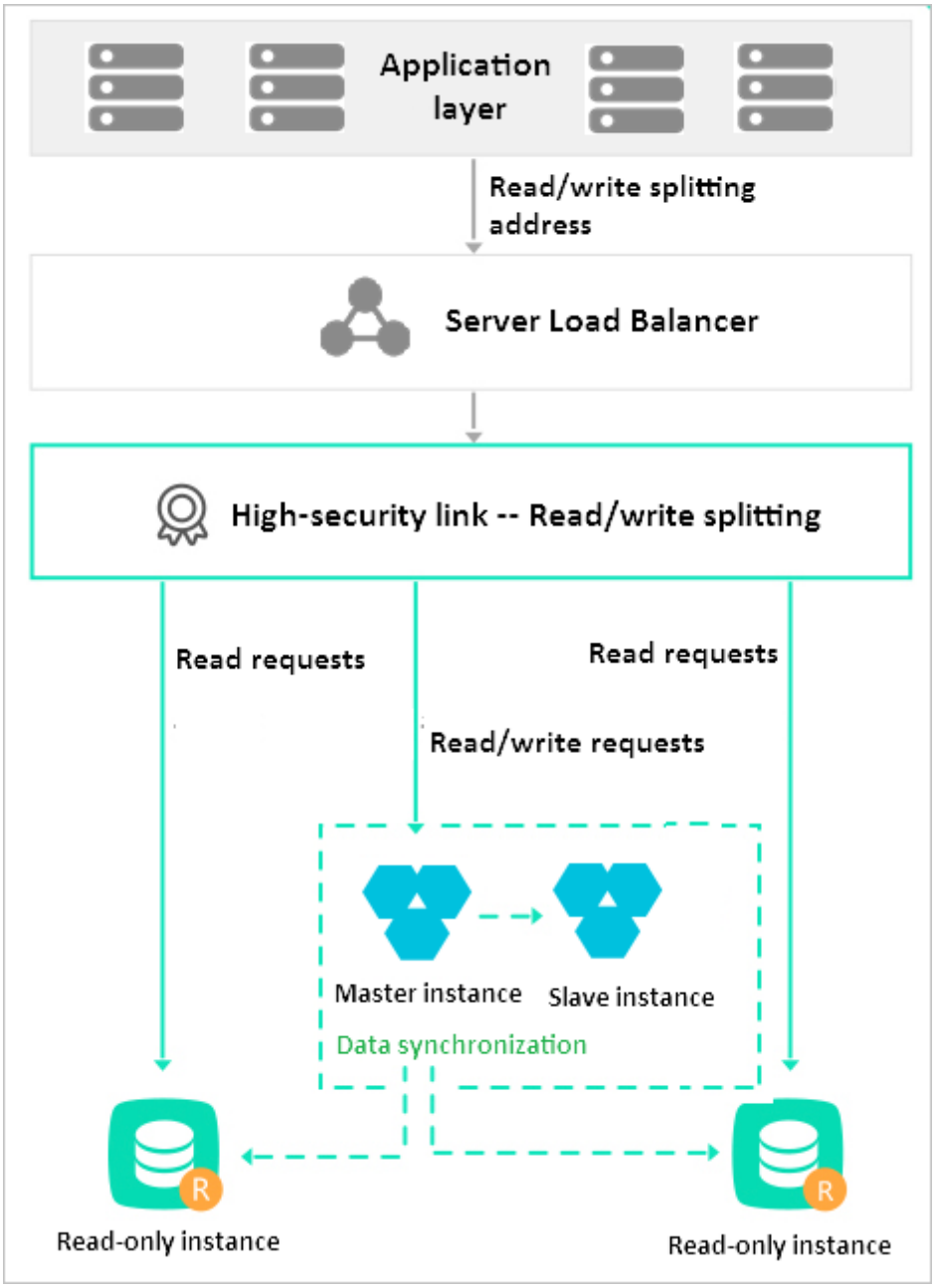
For more information, see Multi-structure data storage.

Read/write splitting function

Read/Write splitting function

ApsaraDB for MySQL allows read-only instances to be directly attached to RDS in order to distribute

the read pressure on the master instance. Each read-only instance has an independent connection string, and the read pressure can be automatically distributed on the application side.



For more information on creating read-only instances on ApsaraDB for MySQL, refer to Create read-only instance.

Concept and terminology

Concept and terminology

Basic concept

Instance: A database service process that takes up physical memory independently. You can set different memory size, disk space, and database type, among which the memory specification determines the performance of the instance. After the instance is created, you can change the configuration and delete the instance.

Database: A logical unit created in an instance. Multiple databases can be created in an instance, and the database name is unique within the instance.

Terminology

Term	Note
Local database/Source database	Refers to the database to be migrated to RDS.
RDS for XX	XX indicates RDS for a specific kind of database: MySQL, SQL Server, PostgreSQL, or PPAS.