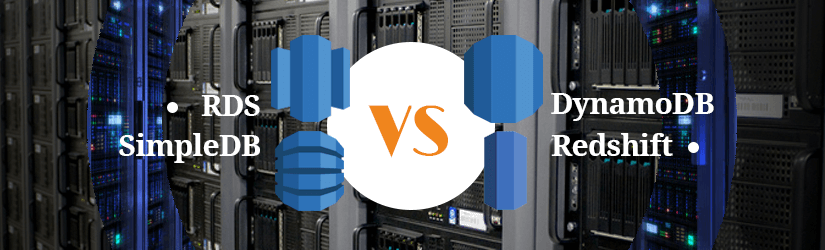
AWS Database Services Complete Overview: RDS vs Redshift vs DynamoDB vs SimpleDB



Cloud database platforms are considered to be a worthy representative of cloud services. No patching, hardware troubles or other maintenance efforts, easy integration and scaling up to the natural growth and customers' demands, high availability and security – all these make the cloud database market growing. Today we are going to have a look at the database platform, which Amazon Web Services (AWS) offers, and find out how to apply it in business and production.

Amazon RDS

Amazon Relational Database Service (RDS) is a good solution for those, who want to run a common database engine with no need for dealing with administration and maintenance. AWS presupposes RDS to be a fully functional alternative to common hardware databases. It is fast, scalable and can be replicated among Availability Zones for greater accessibility.

The following database engines are available:

* **Amazon Aurora.MySQL** v.5.1, 5.5, 5.6 and 5.7 (Community Edition) with InnoDB as the default database storage engine
* **MariaDB** v.10.0
* **Oracle Database** v.11gR2, 12c
* **SQL Server** 2008 R2, SQL Server 2012 (SP2), SQL Server 2014
* **PostgreSQL** 9.3, 9.4

You should have noticed that only InnoDB subsystem is available for MySQL, and it’s highly recommended to be used by Amazon. Nevertheless, it has a 2 TB limit per table. Since MySQL proved itself to be good with big data arrays, use of InnoDB may be challengeable.

So, what about other storage engines? There are no strict restrictions relating to engines use. MyISAM, for instance, breaks Point-In-Time-Restore and Snapshot Restore features of RDS. When you run other engines, you have to stop, lock and flush your tables manually before taking a snapshot or the active content can be damaged.

There are also some inner software limits for each database instance you may want to know:

|  |  |
| --- | --- |
| **Database** | **Limit** |
| Amazon Aurora | No |
| MySQL | No |
| MariaDB | No |
| Oracle Database | One database per instance; no limit on number of schemas per database |
| SQL Server | 30 databases per instance |
| PostgreSQL | No |

**Computing Resources**

Another key question to analyse is computer resources required to run a database. AWS provides for automatic horizontal scalability by deploying additional instances or manual vertical scaling by instance type changing. Nevertheless, the starting point is instance capacities. Amazon RDS has the following offers:

**Standard – Latest Generation**instance family, which includes virtual machines (**VM**) equipped with:

* 2 to 40 **vCPU** (**v**irtual **C**entral **P**rocessing **U**nits).
* 8 to 180 GB RAM.
* Provisioned Input/output Operations Per Second (**PIOPS**) support.
* Network performance from Moderate to 10 Gigabit.

**Standard – Previous Generation** family has VMs equipped with:

* 1 to 8 vCPU.
* 3.75 to 30 GB RAM.
* PIOPS support by top-tier *db.m3.xlarge* and *db.m3.2xlarge*.
* Network performance from Moderate to High, top-tier.

**Memory Optimized – Current Generation** family provides VMs equipped with:

* 2 to 32 vCPU.
* 15 to 244 GB RAM.
* PIOPS does not supported by the weakest *db.r3.large* and the strongest *db.r3.8xlarge*.
* Network performance from Moderate to 10 Gigabit.

**Micro instances** family provides inexpensive VMs equipped with:

* 1-2 vCPU.
* 1 to 8 GB RAM.
* No PIOPS support.
* Network performance from Low to Moderate.

Intrigued how fast Low, Moderate, and High [bandwidths](https://www.cloudberrylab.com/blog/how-to-control-your-bandwidth-with-cloudberry-s3-explorer/) are? There is no official benchmark, though there are some [officious estimates](http://stackoverflow.com/questions/18507405/ec2-instance-typess-exact-network-performance/35806587#35806587). "Low" is anywhere from 50 Mbit to 300 Mbit, "Moderate" is 300-900 Mbit, "High" is 0.9-2.2Gbit. The exact figure strongly depends on the selected region and routing between Amazon data center and the end user.

The instance type to choose also depends on the database engine you want to use. For example, Aurora DB can be deployed only on **Memory Optimized** VMs, while SQL Server Enterprise Edition isn’t available on **Micro** instances. So, it should be checked whether your desired hardware meets software requirements.

***Note:****Amazon RDS limits the number of simultaneously used instances up to 40.*

**Storage Facilities**

Amazon RDS provides three types of attached storage for databases and logs, based on various storage technologies, which differ in performance characteristics and price. All storage types are powered by Amazon Elastic Block Store (EBS) technology, which stripes across multiple Amazon EBS volumes to enhance IOPS performance.

* **Magnetic** (or **Standard)** storage is based on HDD and suitable for the use of a database with low input/output requirements and burst possibilities (for example, latency-tolerable workloads, large data blocks processing and data warehousing). Size limits fall between 5 GB and 3 TB and are determined by the database engine. Their performance is around 100-200 IOPS, and the ceiling is 500.

***Note:****Magnetic storage can’t be reserved for a single instance, so the final capacity also depends on other users.*

* **General Purpose (SSD)** storage is designed for basic workloads and databases, which shall be quick but not too big. SSD storage has minimal latencies and its performance is around three IOPS per gigabyte, which can be boosted up to 3,000 IOPS for a long time and 10,000 IOPS as the upper limit. There are also the following size restrictions:

1. **MySQL, MariaDB, PostgreSQL, Oracle DB** support volumes from 5 GB to 6TB.
2. **SQL Server** supports 20 GB to 4 TB volumes.

*Here is another handy note for you:* while the I/O of General Purpose volumes is 16KB, the I/O of a Magnetic disk is 1 MiB. It definitely makes a distinction between performance and big data processing creating a necessity to use multiple volumes or databases for complex needs.

* **Provisioned IOPS (PIOPS)** storage is based on virtualized volumes, which can provide a stable capacity of 10,000-20,000 IOPS. This is the best choice for intensive database workloads and interactive applications attached to database engines. PIOPS has the following limits:

1. **MySQL, MariaDB, PostgreSQL, Oracle DB** can vary in size between 100 GB and 6TB.
2. **SQL Server Express and Web Editions** vary between 100 GB and 4 TB.
3. **SQL Server Standard and Enterprise** **Edition**varies between 200 GB and 4 TB.

The most appealing feature of PIOPS is that the number of IOPS is dedicated and configured while creating a volume. This capacity is guaranteed by Amazon with ±10% fluctuation 99.9% of time yearly, which helps to rely on a cloud database in case of a big workflow.

The size of a storage block provided for by IOPS storage is 32KiB, and it slightly exceeds the size of a General Purpose Volume.

***Note:****the maximum capacity of all storages is 100TB. To process bigger data you should use another AWS database platform.*

**Accessibility and Manageability**

Among the first RDS promoted features is Multi-AZ ([Availability Zones](https://www.cloudberrylab.com/blog/how-to-move-amazon-ec2-to-different-availability-zone-vpc-region/)) deployment. The feature presupposes database replication with all its settings to an idling VM instance in a different Availability Zone. The main instance and multi-AZ instances are not connected by hardware or a network and belong to different infrastructure objects.

Failures and disasters can hardly affect two data centers at the same time, so Multi-AZ deployment makes them highly durable. In case of a trouble, AWS performs an automatic failover, and a reserved VM starts with the same network settings and endpoint allowing applications and users work with the database as if nothing has happened.

Multi-AZ instances cost more than Single-AZ ones, but they have a number of extra advantages:

* It is fail-resistant, so your database is always available for users.
* Amazon RDS SLA (Service Level Agreement) Terms cover only Multi-AZ instances. If a Single-AZ database is down, there is no credit or compensation.
* Maintenance Window downtime cannot influence Multi-AZ instances.

A Maintenance Window is an obligatory downtime period for service tasks. When it’s used for scaling a database instance or software patching, the virtual machine will be offline while maintenance works are in progress. It is automatically scheduled for requested changes within an instance, security and durability patches, and lasts 30 minutes by default. Such actions are commonly required every few months.

**Usage**

Here are the most typical applications of Amazon RDS:

* You already have a database with a familiar engine, which needs to be offsite.
* The platform for an application that requires the database to be fast, durable, scalable or all of these.
* There is unrationed workflow, which requires a highly scalable database in order to avoid expenses
* The Data shall be processed quickly without storing too much onsite.

Amazon Redshift

Amazon RedShift is a tool designed to work with data of up to dozens of petabytes. Powered by PostgreSQL, it is mostly applied to any kind of SQL applications with minimum changes. The target feature of the service is creating a data warehouse, where a user may focus on data management without keeping an effortful and complex infrastructure. From the technical point of view, Redshift is a cluster database without such consistency features as a foreign key and the uniqueness of field values.

The cluster includes a number of nodes with virtual databases powered by Amazon Elastic Compute Cloud 2 (EC2) instances. Those nodes are basic database units that you can use for your tasks.

**Computing and Storage**

The cluster architecture of Redshift is based on two main roles – a leading node and a computing one:

* A leading node is connected to the outer network; it gets a user request, compiles an executable code for a computing node, makes a query and forwards tasks to computing nodes.
* Сomputing nodes perform user requests and send responses, which are again queried by leading nodes and sent back to the user.
* If there is just one node in a cluster, it plays leading and computing roles, however, the minimum number of nodes in big clusters is two.
* Moreover, each computing node is subdivided into slices, conventional computing units that get tasks from a leading node and take part in queries.

So, the first thing to choose with Redshift is node instances. They are subdivided into the following two tiers:

* **Dense Storage (DS)** nodes are designed for large data workflow, equipped with an HDD for higher capacity at a lower price and available in two variations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Node name** | **vCPU** | **RAM** | **Storage** | **Number of slices** | **Maximum number of nodes per cluster** |
| ds1.xlarge node | 2 | 15GB | 2TB | 2 | 1-32 |
| ds2.xlarge | 4 | 31GB | 2TB | 2 | 1-32 |
| ds1.8xlarge | 16 | 120GB | 16TB | 16 | 2-128 |
| ds2.8xlarge | 36 | 244GB | 16TB | 16 | 2-128 |

* **Dense Compute (DC)** nodes are used for tasks with intensive performance and extremely low latency. They use an SSD as basic storage. Also, these nodes are much faster than DS nodes, that’s why they are considered to be the best choice for the role of a leading node. DC nodes are available in two variations:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Node name** | **vCPU** | **RAM** | **Storage** | **Number of slices** | **Maximum number of nodes per cluster** |
| dc1.xlarge | 2 | 15GB | 160GB | 2 | 32 |
| dc1.8xlarge | 32 | 244GB | 2.56TB | 32 | 2-128 |

How to complete a cluster with nodes? The first criterion to consider is the data volume and its growth rate. If you have 32 TB of data, and this amount remains almost unchanged, 2 **ds1.8xlarge** nodes will perfectly fit your demands. If the amount of data increases by small portions, it will be better to choose 16 **ds1.xlarge** nodes with a possibility of horizontal scaling by 2 TB increments. Like with RDS, you also get storage for backups, which size is the same as the size of the main storage, thus facilitating maintenance.

The second criterion is the required performance. It can be easily increased by scaling the database horizontally, namely, adding DC nodes to the cluster. With Redshift technology, computing nodes mirror their disks to another one making data processing persistent. You can create a data warehouse of any capacity and complexity combining different cluster builds and node types.

**Accessibility and manageability**

While Redshift’s special appeal is its large scale, there are also some limits:

* Number of active nodes: 200.
* Parameter, Security, Subnet groups: 20.
* Subnets within a Subnet group: 20.
* Tables (including temporary ones) per cluster: 9.900.
* Databases per cluster: 60.
* Concurrent user connection to a cluster: 500.

These are not all technical and structural limits of Redshift, but still they are the most important ones. You can learn more on [AWS Limits page](http://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html#limits_redshift) and [Redshift Limits page](http://docs.aws.amazon.com/redshift/latest/mgmt/amazon-redshift-limits.html).

**Maintenance**

Like in RDS, the entire infrastructure is maintained and patched by AWS, and a user doesn’t have a root access. While the data warehouse architecture is really complicated, and it’s really effortful and expensive to replicate Redshift using EC2 instances or any other cloud platform, there is one consequent pitfall – the Maintenance Window.

It is exactly the same as in RDS: scheduled manually or automatically, takes place once per week, the exact time can be adjusted. Unlike RDS, in Redshift you have to manage the database downtime manually.

At last, Redshift supports all auto balancing, autoscaling, monitoring and networking AWS features, SQL commands and API, so it will be easy to deploy and control it.

**Usage**

The most common use cases of Amazon Redshift are as follows:

* Data warehousing – the name speaks for itself
* Big corporate or scientific data processing, with loads related to big amounts of data and large computing loads
* Analytical databases for businesses required to store, analyze and transfer big data within a short time
* Customer activity monitoring for analysis and statistics

DynamoDB

DynamoDB is a NoSQL database service by AWS designed for fast processing of small data, which dynamically grows and changes. The main non-relative feature of DynamoDB is the unstrict structure of a table – it consists of items (as compared to rows in a traditional table) and attributes (an analogue to columns). Carrying over of relational engines, it resembles a table with an individual number of columns in each row. Database mutability and fast I/O rate is powered by an SSD used as the basic (and the only) storage hardware.

**Features and Limits**

With DynamoDB there are no hardware instances on which capacities and billing depend. The main value is the read/write throughput used by the database. There is no limit on storage resources – they grow as the database grows with no replication of instances or any other typical cloud scaling. The multi-AZ feature, which requires an additional fee with RDS, comes from the box here: your data is automatically replicated among 3 Availability Zones (AZ) within the selected region. Total absence of administering activities, data replication, and final-performance scaling models make DynamoDB extremely durable.

Meanwhile, DynamoDB doesn’t support such complex functions as advanced querying and transactions. Since data is partitioned for durability, it takes some time to re-write it in each replica after a successful write operation in the main one. The balance between read and write capacities is called Read Consistency, and it can be adjusted in the following way:

* Eventually Consistent Reads option gives a priority to a read operation, which forwards data even if it is already modified but hasn’t been yet replicated to a local AZ. This option bursts the reading performance, but read requests shall be performed again to get up-to-date data.
* Strongly Consistent Reads option is targeted at getting the latest data. It takes more time but it returns the result, which reflects all successful writes made before read initialization.

Read consistency is not the only one unique peculiarity of DynamoDB. We stated some of its main features below:

* Maximum R&W throughput – 10.000 R&W units per table, 20.000 R&W units per account.  
  *Note: the maximum R&W* *throughput for the US East region is 40.000 and 80.000 R&W units respectively.*
* Maximum item size (item key + all attributes) – 400 KB.
* Maximum table size – unlimited.
* Tables per account: 256.
* Supported data: Number, String, Binary, Boolean, collection data (Number Set, String Set, Binary Set) heterogeneous List and heterogeneous Map (NULL values).
* String data encoding: UTF-8.

More limits can be found on Amazon DynamoDB [Limits Page](http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html) and in its [FAQ](https://aws.amazon.com/dynamodb/faqs/) [section](https://aws.amazon.com/dynamodb/faqs/). There are also additional features like:

* Streams – time-ordered sequences of item changes.
* Triggers – Integration with AWS Lambda to execute a custom function if certain item changes are detected.
* Integration – an effortless interaction between DynamoDB and Redshift, Data Pipeline, Elastic MapReduce, Hadoop, etc.
* Compatibility – supports all AWS networking, monitoring and management services.

**Usage**

In the upshot, the best practices with DynamoDB are as follows:

* Data blocks systematization and processing.
* Advertising services: collection of customer data, making trend charts, etc.
* Messaging and blogging: building message selections, the list of blog entries by author, etc.
* Gaming: high-scores, world changes, player status and statistics, etc.
* Any other case when you have to process data rather than store and the data shall be highly available rather transactable.

SimpleDB

Amazon SimpleDB is another NoSQL database platform, which resembles DynamoDB technically. It has a similar non-relational item-attributable structure, replicates among a few regions for durability, and provides read consistency options to adjust an appropriate access mode. Nevertheless, SimpleDB should be treated as the database core, which supports only the basic non-relational index, the query and storage function. The main distinctive features of the database platform are as follows:

* The basic structural unit is a domain, which is referred to as a table in a relational database. Domains are multiplied in order to increase performance.
* Domain size limit is 10 GB, and it’s scaled up by means of deployment of additional domains, which mirror its disks to create a database medium.
* Maximum query execution time is 5 seconds.

SimpleDB differs from DynamoDB in capacity too. Let’s compare them to clear up all things:

|  |  |  |
| --- | --- | --- |
|  | **DynamoDB** | **SimpleDB** |
| **Write Capacity (per table)** | 10.000-40.000 units | 25 writes/sec |
| **Performance Scaling Method** | Presettable Throughput | Horizontal (no bursts available) |
| **Attributes per table** | Unlimited | 1 billion/td> |
| **Attributes per item** | Unlimited | 256 |
| **Items per table (with maximal size)** | Unlimited | 3.906.250 |
| **Tables per account** | 256 | 250 |
| **Maximum size of item** | 400KB | 1KB |
| **Data types supported** | Number, String, Binary, Boolean, NULL values, collection data | String |
| **Encoding of string data** | UTF-8 | UTF-8 |

Thus, the main billing metrics are hours of service, running and data storage capacity. As a NoSQL database, SimpleDB doesn’t support complex transactions, but still it can run conditional PUT/DELETE operations. Domains are easily accessed via web interfaces, managed via API or Management Console, and can be integrated with any AWS product.

**Usage**

Lightweight and easily managed, SimpleDB doesn’t stand out against other database platforms by performance, computing capacity or storage facilities. Nevertheless, it’s beneficial to use it as an auxiliary service for other AWS products or as a simple database for non-complex needs. SimpleDB common usage scenarios are as follows:

* Logging facility
* Gaming database for scores, player items, client settings, etc
* Indexing object metadata like rating, format or geolocation

Conclusion

The choice of a database platform always depends on computing resources and flexibility – an external index, a data warehouse and a business activity tracker require different storage capacities, database engines and performance rates. The depth of administering is also significant. If you want to adjust everything within the database, it would be better to deploy one of the preconfigured database images for [EC2](https://www.cloudberrylab.com/blog/azure-vm-vs-amazon-ec2-vs-google-ce-cloud-computing-comparison/) having all software installed with a possibility to access root features. To facilitate your decision and brush up the features of every platform, we created a little chart below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Amazon RDS** | **Amazon Redshift** | **Amazon DynamoDB** | **Amazon SimpleDB** |
| **Database engine** | Amazon Aurora, MySQL, MariaDB, Oracle Database, SQL Server, PostgreSQL | Redshift (adapted PostgreSQL) | NoSQL | NoSQL (with limited capacity) |
| **Computing resources** | Instances with 32 vCPU and 244 GB RAM | Nodes with vCPU and 244 GB RAM | Not specified, software as a service | Not specified, software as a service |
| **Data storage facilities (max)** | 6 TB per instance, 20.000 IOPS | 16 TB per instance | Unlimited storage size, 40.000 Read/Write per table | 10 GB per domain, 25 Writes/Sec |
| **Maintenance Windows** | 30 minutes per week | 30 minutes per week | No effect | No effect |
| **Multi-AZ replication** | As an additional service | Manual | Built-in | Built-in |
| **Tables (per basic structural unit)** | Defined by the database engine | 9.900 | 256 | 250 |
| **Main usage feature** | Conventional database | Data warehouse | Database for dynamically modified data | Simple database for small records or auxiliary roles |