Data Warehousing on AWS

March 2016

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Data engineers, data analysts, and developers in enterprises across the globe are looking to migrate data warehousing to the cloud to increase performance and lower costs. This whitepaper discusses a modern approach to analytics and data warehousing architecture, outlines services available on Amazon Web Services (AWS) to implement this architecture, and provides common design patterns to build data warehousing solutions using these services.

Introduction

In today's world, data and analytics are indispensable to business. Almost all large enterprises have built data warehouses for reporting and analytics purposes using the data from a variety of sources, including their own transaction-processing systems and other databases.

But building and running a data warehouse—a central repository of information coming from one or more data sources—has always been complicated and expensive. Most data warehousing systems are complex to set up, cost millions of dollars in upfront software and hardware expenses, and can take months in planning, procurement, implementation, and deployment processes. After you have made the initial investments and set your data warehouse up, you have to hire a team of database administrators to keep your queries running fast and protect against data loss.

Traditional data warehouses are also difficult to scale. When data volumes grow or you want to make analytics and reports available to more users, you have to choose between accepting slow query performance or investing time and effort on an expensive upgrade process. In fact, some IT teams discourage augmenting data or adding queries to protect existing service-level agreements. Many enterprises struggle with maintaining a healthy relationship with traditional database vendors. They are often forced to either upgrade hardware for a managed system or enter a protracted negotiation cycle for an expired term license. When they reach the scaling limit on one data warehouse engine, they are forced to migrate to another engine from the same vendor with different SQL semantics.

Amazon Redshift has changed how enterprises think about data warehousing by dramatically lowering the cost and effort associated with deploying data warehouse systems without compromising on features and performance. Amazon Redshift is a fast, fully managed, petabyte-scale data warehousing solution that makes it simple and cost-effective to analyze large volumes of data using existing business intelligence (BI) tools. With Amazon Redshift, you can get the performance of columnar data warehousing engines that perform massively parallel processing (MPP) at a tenth of the cost. You can start small for $0.25 per hour with no commitments and scale to petabytes for $1,000 per terabyte per year.

Since launching in February 2013, Amazon Redshift has been one of fastest growing AWS services, with many thousands of customers across industries and company sizes. Enterprises such as NTT DOCOMO, FINRA, Johnson & Johnson, Hearst, Amgen, and NASDAQ have migrated to Amazon Redshift. As a result, Amazon Redshift ranked as a leader in the *Forrester Wave: Enterprise Data Warehouse, Q4 2015* report.[1]

In this whitepaper, we provide you the information you need to take advantage of the strategic shift happening in the data warehousing space from on-premises to the cloud:

- Modern analytics architecture
- Data warehousing technology choices available within that architecture
- A deep dive on Amazon Redshift and its differentiating features
- A blueprint for building a complete data warehousing system on AWS with Amazon Redshift and other services
- Practical tips for migrating from other data warehousing solutions and tapping into our partner ecosystem

Modern Analytics and Data Warehousing Architecture

Again, a *data warehouse* is a central repository of information coming from one or more data sources. Data typically flows into a data warehouse from transactional systems and other relational databases, and typically includes structured, semi-structured, and unstructured data. This data is processed, transformed, and ingested at a regular cadence. Users including data scientists, business analysts, and decision-makers access the data through BI tools, SQL clients, and spreadsheets.

Why build a data warehouse at all—why not just run analytics queries directly on an online transaction processing (OLTP) database, where the transactions are recorded? To answer the question, let's look at the differences between data warehouses and OLTP databases. Data warehouses are optimized for batched write operations and reading high volumes of data, whereas OLTP databases are optimized for continuous write operations and high volumes of small read operations. In general, data warehouses employ denormalized schemas like the Star schema and Snowflake schema because of high data throughput requirements, whereas OLTP databases employ highly normalized schemas, which are more suited for high transaction throughput requirements. The Star schema consists of a few large fact tables that reference a number of dimension tables. The Snowflake schema, an extension of the Star schema, consists of dimension tables that are normalized even further.

To get the benefits of using a data warehouse managed as a separate data store with your source OLTP or other source system, we recommend that you build an efficient data pipeline. Such a pipeline extracts the data from the source system, converts it into a schema suitable for data warehousing, and then loads it into the data warehouse. In the next section, we discuss the building blocks of an analytics pipeline and the different AWS services you can use to architect the pipeline.

Analytics Architecture

Analytics pipelines are designed to handle large volumes of incoming streams of data from heterogeneous sources such as databases, applications, and devices.

A typical analytics pipeline has the following stages:

1. Collect data.
2. Store the data.
3. Process the data.
4. Analyze and visualize the data.

For an illustration, see Figure 1, following.

Figure 1: Analytics Pipeline

Data Collection

At the data collection stage, consider that you probably have different types of data, such as transactional data, log data, streaming data, and Internet of Things (IoT) data. AWS provides solutions for data storage for each of these types of data.

Transactional Data

Transactional data, such as e-commerce purchase transactions and financial transactions, is typically stored in relational database management systems (RDBMS) or NoSQL database systems. The choice of database solution depends on the use case and application characteristics. A NoSQL database is suitable when the data is not well-structured to fit into a defined schema, or when the schema changes very often. An RDBMS solution, on the other hand, is suitable when transactions happen across multiple table rows and the queries require complex joins. Amazon DynamoDB is a fully managed NoSQL database service that can be used as an OLTP store for your applications. Amazon RDS allows you to implement a SQL-based relational database solution for your application.

Log Data

Reliably capturing system-generated logs will help you troubleshoot issues, conduct audits, and perform analytics using the information stored in the logs. Amazon Simple Storage Service (Amazon S3) is a popular storage solution for nontransactional data, such as log data, that is used for analytics. Because it provides 11 9's of durability (that is, 99.999999999 percent durability), Amazon S3 is also a popular archival solution.

Streaming Data

Web applications, mobile devices, and many software applications and services can generate staggering amounts of streaming data—sometimes terabytes per hour—that need to be collected, stored, and processed continuously.[2] Using Amazon Kinesis services, you can do that simply and at a low cost.

IoT Data

Devices and sensors around the world send messages continuously. Enterprises see a growing need today to capture this data and derive intelligence from it. Using AWS IoT, connected devices interact easily and securely with the AWS cloud. AWS IoT makes it easy to use AWS services like AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, and Amazon DynamoDB to build applications that gather, process, analyze, and act on IoT data, without having to manage any infrastructure.

Data Processing

The collection process provides data that potentially has useful information. You can analyze the extracted information for intelligence that will help you grow your business. This intelligence might, for example, tell you about your user behavior and the relative popularity of your products. The best practice to gather this intelligence is to load your raw data into a data warehouse to perform further analysis.

To do so, there are two types of processing workflows, batch and real time. The most common forms of processing, online analytic processing (OLAP) and OLTP, each use one of these types. Online analytic processing (OLAP) processing is generally batch-based. In contrast, OLTP systems are oriented towards real-time processing and are generally not well-suited for batch-based processing. If you decouple data processing from your OLTP system, you keep the data processing from affecting your OLTP workload.

First, let's look at what is involved in batch processing.

Extract Transform Load (ETL)

ETL is the process of pulling data from multiple sources to load into data warehousing systems. ETL is normally a continuous ongoing process with a well-defined workflow. During this process, data is initially extracted from one or more sources. The extracted data is then cleansed, enriched, transformed, and loaded into a data warehouse. Hadoop framework tools such as Apache Pig and Apache Hive are commonly used in an ETL pipeline to perform transformations on large volumes of data.

Extract Load Transform (ELT)

ELT is a variant of ETL where the extracted data is loaded into the target system first. Transformations are performed after the data is loaded into the data warehouse. ELT typically works well when your target system is powerful enough to handle transformations. Amazon Redshift is often used in ELT pipelines because it is highly efficient in performing transformations.

Online Analytical Processing (OLAP)

OLAP systems store aggregated historical data in multidimensional schemas. Used widely in data mining, OLAP systems allow you to extract data and spot trends on multiple dimensions. Because it is optimized for fast joins, Amazon Redshift is often used to build OLAP systems.

Now, let's look at what's involved in real-time processing of data.

Real-Time Processing

We talked about streaming data earlier and mentioned Amazon Kinesis as a solution to capture and store streaming data. You can process this data sequentially and incrementally on a record-by-record basis or over sliding time windows, and use the processed data for a wide variety of analytics including correlations, aggregations, filtering, and sampling. This type of processing is called real-time processing. Information derived from real-time processing gives companies visibility into many aspects of their business and customer activity—such as service usage (for metering or billing), server activity, website clicks, and geolocation of devices, people, and physical goods—and enables them to respond promptly to emerging situations. Real-time processing requires a highly concurrent and scalable processing layer.

To process streaming data in real time, you can use AWS Lambda. Lambda can process the data directly from AWS IoT or Amazon Kinesis Streams. Lambda lets you run code without provisioning or managing servers.

Amazon Kinesis Client Library (KCL) is another way to process data from Amazon Kinesis Streams. KCL gives you more flexibility than AWS Lambda to batch your incoming data for further processing. You can also use KCL to apply extensive transformations and customizations in your processing logic.

Amazon Kinesis Firehose is the easiest way to load streaming data into AWS. It can capture streaming data and automatically load it into Amazon Redshift, enabling near-real-time analytics with existing BI tools and dashboards you're already using today. You can define your batching rules with Firehose, and then it takes care of reliably batching the data and delivering to Amazon Redshift.

Data Storage

You can store your data in either a data warehouse or data mart, as discussed in the following.

Data Warehouse

As we've said, a *data warehouse* is a central repository of information coming from one or more data sources. Using data warehouses, you can run fast analytics on large volumes of data and unearth patterns hidden in your data by leveraging BI tools. Data scientists query a data warehouse to perform offline analytics and spot trends. Users across the organization consume the data using ad hoc SQL queries, periodic reports, and dashboards to make critical business decisions.

Data Mart

A *data mart* is a simple form of data warehouse focused on a specific functional area or subject matter. For example, you can have specific data marts for each division in your organization or segment data marts based on regions. You can build data marts from a large data warehouse, operational stores, or a hybrid of the two. Data marts are simple to design, build, and administer. However, because data marts are focused on specific functional areas, querying across functional areas can become complex because of the distribution.

You can use Amazon Redshift to build data marts in addition to data warehouses.

Analysis and Visualization

After processing the data and making it available for further analysis, you need the right tools to analyze and visualize the processed data.

In many cases, you can perform data analysis using the same tools you use for processing data. You can use tools such as SQL Workbench to analyze your data in Amazon Redshift with ANSI SQL. Amazon Redshift also works well with popular third-party BI solutions available on the market.

Amazon QuickSight is a fast, cloud-powered BI service that makes it easy to create visualizations, perform ad hoc analysis, and quickly get business insights from your data. Amazon QuickSight is integrated with Amazon Redshift and is currently in preview, with general availability planned for later in 2016.

If you are using Amazon S3 as your primary storage, a popular way to perform analysis and visualization is to run Apache Spark notebooks on Amazon Elastic MapReduce (Amazon EMR). Using this process, you have the flexibility to run SQL or execute custom code written in languages such as Python and Scala.

For another visualization approach, Apache Zeppelin is an open source BI solution that you can run on Amazon EMR to visualize data in Amazon S3 using Spark SQL. You can also use Apache Zeppelin to visualize data in Amazon Redshift.

Analytics Pipeline with AWS Services

AWS offers a broad set of services to implement an end-to-end analytics platform. Figure 2 shows the services discussed preceding and where they fit within the analytics pipeline.

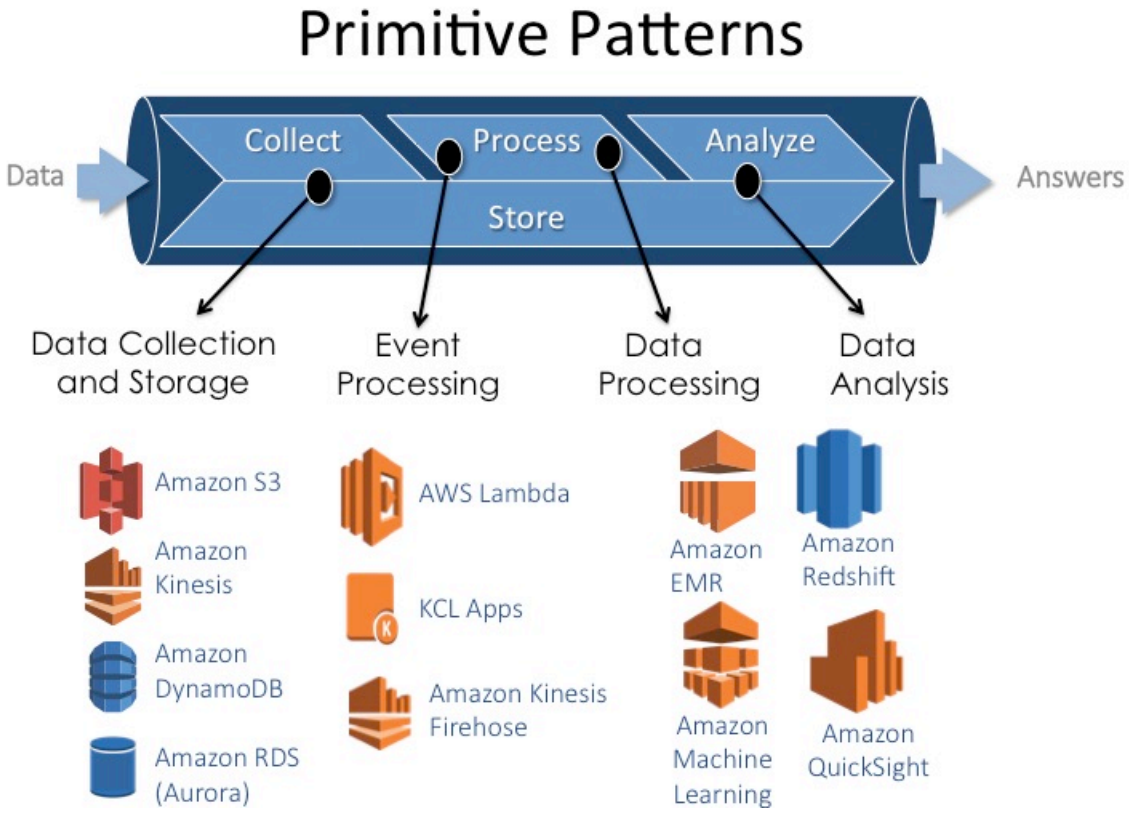


Figure 2: Analytics Pipeline with AWS Services

Data Warehouse Technology Options

In this section, we discuss options available for building a data warehouse: row-oriented databases, column-oriented databases, and massively parallel processing architectures.

Row-Oriented Databases

Row-oriented databases typically store whole rows in a physical block. High performance for read operations is achieved through secondary indexes. Databases such as Oracle Database Server, Microsoft SQL Server, MySQL, and PostgreSQL are row-oriented database systems. These systems have been traditionally used for data warehousing, but they are better suited for transactional processing (OLTP) than for analytics.

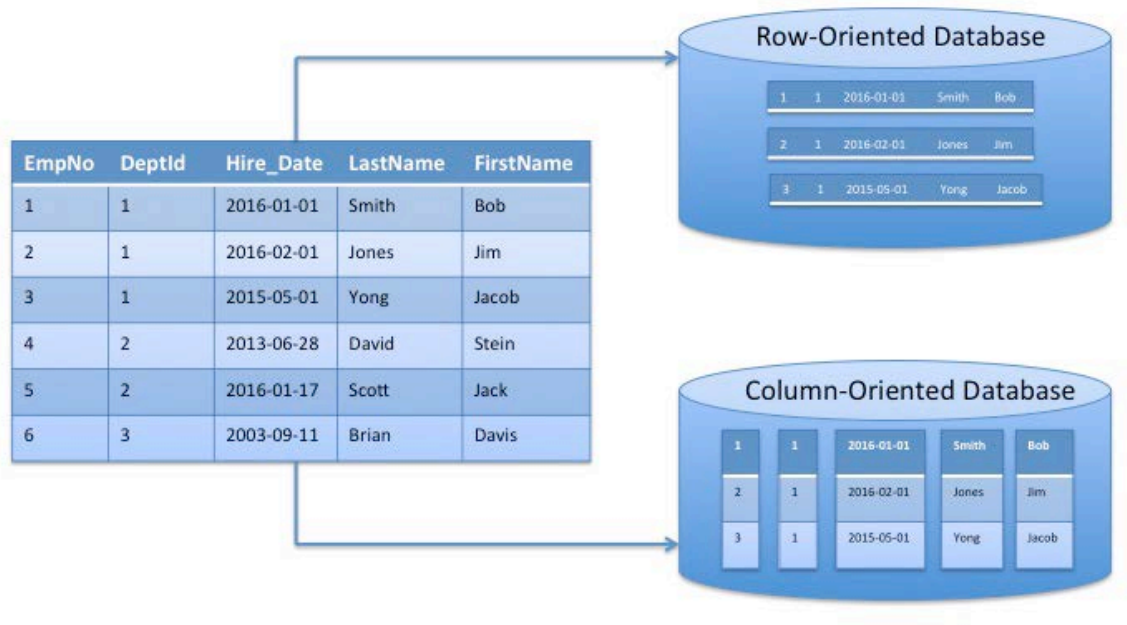
To optimize performance of a row-based system used as a data warehouse, developers use a number of techniques, including building materialized views, creating pre-aggregated rollup tables, building indexes on every possible predicate combination, implementing data partitioning to leverage partition pruning by query optimizer, and performing index based joins.

Traditional row-based data stores are limited by the resources available on a single machine. Data marts alleviate the problem to an extent by using functional sharding. You can split your data warehouse into multiple data marts, each satisfying a specific functional area. However, when data marts grow large over time, data processing slows down.

In a row-based data warehouse, every query has to read through all of the columns for all of the rows in the blocks that satisfy the query predicate, including columns you didn't choose. This approach creates a significant performance bottleneck in data warehouses, where your tables have more columns, but your queries use only a few.

Column-Oriented Databases

Column-oriented databases organize each column in its own set of physical blocks instead of packing the whole rows into a block. This functionality allows them to be more I/O efficient for read-only queries because they only have to read those columns accessed by a query from disk (or from memory). This approach makes column-oriented databases a better choice than row-oriented databases for data warehousing.



EmpNo	DeptId	Hire_Date	LastName	FirstName
1	1	2016-01-01	Smith	Bob
2	1	2016-02-01	Jones	Jim
3	1	2015-05-01	Yong	Jacob
4	2	2013-06-28	David	Stein
5	2	2016-01-17	Scott	Jack
6	3	2003-09-11	Brian	Davis

Figure 3: Row-Oriented vs. Column-Oriented Databases

Figure 3, preceding, illustrates the primary difference between row-oriented and column-oriented databases. Rows are packed into their own blocks in a row-oriented database, and columns are packed into their own blocks in a column-oriented database.

After faster I/O, the next biggest benefit to using a column-oriented database is improved compression. Because every column is packed into its own set of blocks, every physical block contains the same data type. When all the data is the same data type, the database can use extremely efficient compression algorithms. As a result, you need less storage compared to a row-oriented database. This approach also results in significantly lesser I/O because the same data is stored in fewer blocks.

Some column-oriented databases that are used for data warehousing include Amazon Redshift, Vertica, Teradata Aster, and Druid.

Massively Parallel Processing Architectures

An MPP architecture allows you to use all of the resources available in the cluster for processing data, thereby dramatically increasing performance of petabyte-scale data warehouses. MPP data warehouses allow you improve performance by simply adding more nodes to the cluster. Amazon Redshift, Druid, Vertica, GreenPlum, and Teradata Aster are some of the data warehouses built on an MPP architecture. Open source frameworks such as Hadoop and Spark also support MPP.

Amazon Redshift Deep Dive

As a columnar MPP technology, Amazon Redshift offers key benefits for performant, cost-effective data warehousing including efficient compression, reduced I/O, and lower storage requirements. It is based on ANSI SQL, so you can run existing queries with little or no modification. As a result, it has become a popular choice for enterprise data warehouses and data marts today. In this section, we dive deeper into Amazon Redshift and discuss more about its capabilities.

Amazon Redshift delivers fast query and I/O performance for virtually any data size by using columnar storage, and by parallelizing and distributing queries across multiple nodes. It automates most of the common administrative tasks associated with provisioning, configuring, monitoring, backing up, and securing a data warehouse, making it easy and inexpensive to manage. Using this automation, you can build petabyte-scale data warehouses in minutes instead of the weeks or months taken by traditional on-premises implementations.

Performance

Amazon Redshift uses columnar storage, data compression, and zone maps to reduce the amount of I/O needed to perform queries. Interleaved sorting enables fast performance without the overhead of maintaining indexes or projections.

Amazon Redshift employs an MPP architecture to take advantage of all available resources by parallelizing and distributing SQL operations. The underlying hardware is designed for high performance data processing, using local attached storage to maximize throughput between the CPUs and drives, and a 10 GigE

mesh network to maximize throughput between nodes. Performance can be tuned based on your data warehousing needs: AWS offers Dense Compute (DC) with solid-state drives and also Dense Storage (DS) options. Continuous deployment of software upgrades delivers ongoing performance improvements without any user intervention.

Durability and Availability

To provide the best possible data durability and availability, Amazon Redshift automatically detects and replaces any failed node in your data warehouse cluster. It makes your replacement node available immediately and loads your most frequently accessed data first so that you can resume querying your data as quickly as possible. Because Amazon Redshift mirrors your data across your cluster, it uses the data from another node to rebuild the failed node. The cluster is in read-only mode until a replacement node is provisioned and added to the cluster, which typically takes only a few minutes.

Amazon Redshift clusters reside within one Availability Zone.[3] However, if you want to a Multi-AZ setup for Amazon Redshift, you can create a mirror and then self-manage replication and failover.

With just a few clicks in the Amazon Redshift Management Console, you can set up a robust disaster recovery (DR) environment with Amazon Redshift. You can keep copies of your backups in multiple AWS Regions. In case of a service interruption in one AWS Region, you can restore your cluster from the backup in a different AWS Region. You can gain read/write access to your cluster within a few minutes of initiating the restore operation.

Scalability and Elasticity

With a few clicks in the console or an API call, you can easily change the number and type of nodes in your data warehouse as your performance or capacity needs change.[4] Amazon Redshift enables you to start with as little as a single 160 GB node and scale up all the way to a petabyte or more of compressed user data using many nodes. For more information, see About Clusters and Nodes in the *Amazon Redshift Cluster Management Guide*.[5]

While resizing, Amazon Redshift places your existing cluster into read-only mode, provisions a new cluster of your chosen size, and then copies data from your old cluster to your new one in parallel. During this process, you pay only for the active Amazon Redshift cluster. You can continue running queries against your old cluster while the new one is being provisioned. After your data has been copied to your new cluster, Amazon Redshift automatically redirects queries to your new cluster and removes the old cluster.

You can use Amazon Redshift API actions to programmatically launch clusters, scale clusters, create backups, restore backups, and more. Using this approach, you can integrate these API actions into your existing automation stack or build custom automation that suits your needs.

Interfaces

Amazon Redshift has custom Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) drivers that you can download from the **Connect Client** tab of the console, which means you can use a wide range of familiar SQL clients. You can also use standard PostgreSQL JDBC and ODBC drivers. For more information about Amazon Redshift drivers, see Amazon Redshift and PostgreSQL in the *Amazon Redshift Database Developer Guide.* [6]

You can also find numerous examples of validated integrations with many popular BI and ETL vendors.[7] In these integrations, loads and unloads execute in parallel on each compute node to maximize the rate at which you can ingest or export data to and from multiple resources, including Amazon S3, Amazon EMR, and Amazon DynamoDB. You can easily load streaming data into Amazon Redshift using Amazon Kinesis Firehose, enabling near real-time analytics with existing BI tools and dashboards. You can locate metrics for compute utilization, memory utilization, storage utilization, and read/write traffic to your Amazon Redshift data warehouse cluster by using the console or Amazon CloudWatch API operations.

Security

To help provide data security, you can run Amazon Redshift inside a virtual private cloud based on the Amazon Virtual Private Cloud (Amazon VPC) service. You can use the software-defined networking model of the VPC to define firewall

rules that restrict traffic based on the rules you configure.[8] Amazon Redshift supports SSL-enabled connections between your client application and your Amazon Redshift data warehouse cluster, which enables data to be encrypted in transit.

The Amazon Redshift compute nodes store your data, but the data can be accessed only from the cluster's leader node. This isolation provides another layer of security. Amazon Redshift integrates with AWS CloudTrail to enable you to audit all Amazon Redshift API calls.[9] To help keep your data secure at rest, Amazon Redshift encrypts each block using hardware-accelerated AES-256 encryption as each block is written to disk. This encryption takes place at a low level in the I/O subsystem; the I/O subsystem encrypts everything written to disk, including intermediate query results. The blocks are backed up as is, which means that backups are also encrypted. By default, Amazon Redshift takes care of key management, but you can choose to manage your keys using your own hardware security modules (HSMs) or manage your keys through AWS Key Management Service.[10,11]

Cost Model

Amazon Redshift requires no long-term commitments or upfront costs. This pricing approach frees you from the capital expense and complexity of planning and purchasing data warehouse capacity ahead of your needs. Charges are based on the size and number of nodes in your cluster.

There is no additional charge for backup storage up to 100 percent of your provisioned storage. For example, if you have an active cluster with two XL nodes for a total of 4 TB of storage, AWS provides up to 4 TB of backup storage on Amazon S3 at no additional charge. Backup storage beyond the provisioned storage size, and backups stored after your cluster is terminated, are billed at standard Amazon S3 rates.[12] There is no data transfer charge for communication between Amazon S3 and Amazon Redshift. For more information, see Amazon Redshift Pricing.[13]

Ideal Usage Patterns

Amazon Redshift is ideal for online analytical processing (OLAP) using your existing BI tools. Organizations are using Amazon Redshift to do the following:

- Run enterprise BI and reporting
- Analyze global sales data for multiple products
- Store historical stock trade data
- Analyze ad impressions and clicks
- Aggregate gaming data
- Analyze social trends
- Measure clinical quality, operation efficiency, and financial performance in health care

Anti-Patterns

Amazon Redshift is not ideally suited for the following usage patterns:

- **Small datasets** – Amazon Redshift is built for parallel processing across a cluster. If your dataset is less than 100 gigabytes, you're not going to get all the benefits that Amazon Redshift has to offer and Amazon RDS may be a better solution.
- **OLTP** – Amazon Redshift is designed for data warehousing workloads delivering extremely fast and inexpensive analytic capabilities. If you require a fast transactional system, you might want to choose a traditional relational database system built on Amazon RDS or a NoSQL database such as Amazon DynamoDB.
- **Unstructured data** – Data in Amazon Redshift must be structured by a defined schema. Amazon Redshift doesn't support an arbitrary schema structure for each row. If your data is unstructured, you can perform extract, transform, and load (ETL) on Amazon EMR to get the data ready for loading into Amazon Redshift. For JSON data, you can store key value pairs and use the native JSON functions in your queries.[14]
- **BLOB data** – If you plan on storing binary large object (BLOB) files such as digital video, images, or music, you might want to consider storing the data in Amazon S3 and referencing its location in Amazon Redshift. In this scenario, Amazon Redshift keeps track of metadata (such as item name, size, date created, owner, location, and so on) about your binary objects, but the large objects themselves are stored in Amazon S3.

Migrating to Amazon Redshift

If you decide to migrate from an existing data warehouse to Amazon Redshift, which migration strategy you should choose depends on several factors:

- The size of the database and its tables
- Network bandwidth between the source server and AWS
- Whether the migration and switchover to AWS will be done in one step or a sequence of steps over time
- The data change rate in the source system
- Transformations during migration
- The partner tool that you plan to use for migration and ETL

One-Step Migration

One-step migration is a good option for small databases that don't require continuous operation. Customers can extract existing databases as comma-separated value (CSV) files, then use services such as AWS Import/Export Snowball to deliver datasets to Amazon S3 for loading into Amazon Redshift. Customers then test the destination Amazon Redshift database for data consistency with the source. Once all validations have passed, the database is switched over to AWS.

Two-Step Migration

Two-step migration is commonly used for databases of any size:

1. **Initial data migration:** The data is extracted from the source database, preferably during nonpeak usage to minimize the impact. The data is then migrated to Amazon Redshift by following the one-step migration approach described previously.
2. **Changed data migration:** Data that changed in the source database after the initial data migration is propagated to the destination before switchover. This step synchronizes the source and destination databases.

Once all the changed data is migrated, you can validate the data in the destination database, perform necessary tests, and if all tests are passed, switch over to the Amazon Redshift data warehouse.

Tools for Database Migration

Several tools and technologies for data migration are available. You can use some of these tools interchangeably, or you can also use other third-party or open-source tools available in the market.

1. AWS Database Migration Service supports both the one-step and the two-step migration processes described preceding.[15] To follow the two-step migration process, you enable supplemental logging to capture changes to the source system. You can enable supplemental logging at the table or database level.
2. Additional data integration partner tools are the following:
   - Attunity
   - Informatica
   - SnapLogic
   - Talend
   - Bryte

For more information on data integration and consulting partners, see Amazon Redshift Partners.[16]

Designing Data Warehousing Workflows

In the previous sections, we discussed the features of Amazon Redshift that make it ideally suited for data warehousing. To understand how to design data warehousing workflows with Amazon Redshift, let's now look at the most common design pattern along with an example use case.

Suppose that a multinational clothing maker has more than a thousand retail stores, sells certain clothing lines through department and discount stores, and has an online presence. From a technical standpoint, these three channels currently operate independently. They have different management, point-of-sale

systems, and accounting departments. No single system merges all the related datasets together to provide the CEO a 360-degree view across the entire business.

Suppose also that the CEO wants to get a company-wide picture of these channels and be able to do ad hoc analytics such as the following:

- What trends exist across channels?
- Which geographic regions do better across channels?
- How effective are the company's advertisements and promotions?
- What trends exist across each clothing line?
- Which external forces have impacts on the company's sales, for example the unemployment rate and weather conditions?
- How do store attributes affect sales, for example tenure of employees and management, strip mall versus enclosed mall, location of merchandise in the store, promotion, endcaps, sales circulars, and in-store displays?

An enterprise data warehouse solves this problem. It collects data from each of the three channels' various systems and also from publicly available data such as weather and economic reports. Each data source sends data daily for consumption by the data warehouse. Because each data source might be structured differently, an extract, transform, and load (ETL) process is performed to reformat the data into a common structure. Then analytics can be performed across data from all sources simultaneously. To do this, we use the following data flow architecture:

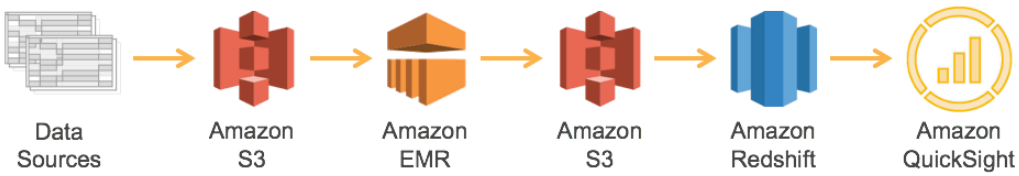


Figure 4: Enterprise Data Warehouse Workflow

1. The first step in this process is getting the data from different sources into Amazon S3. Amazon S3 provides a highly durable, inexpensive, and scalable storage platform that can be written to in parallel from many different sources at a very low cost.

2. Amazon EMR is used to transform and cleanse the data from the source format to go into the destination format. Amazon EMR has built-in integration with Amazon S3, which allows parallel threads of throughput from each node in your Amazon EMR cluster to and from Amazon S3.

   Typically, a data warehouse gets new data on a nightly basis. Because there is no need for analytics in the middle of the night, the only requirement around this transformation process is that it finishes by the morning when the CEO and other business users need to access reports and dashboards. Therefore, you can use the Amazon EC2 Spot market to further bring down the cost of ETL here.[17] A good spot strategy is to start bidding at a very low price at midnight, and continually increase your price over time until capacity is granted. As you get closer to the deadline, if spot bids have not succeeded, you can fall back to on-demand prices to ensure you still meet your completion time requirements. Each source might have a different transformation process on Amazon EMR, but with the AWS pay-as-you-go model, you can create a separate Amazon EMR cluster for each transformation and tune it to be exactly the right capacity to complete all data transformation jobs without contending with resources of the other jobs.

3. Each transformation job loads formatted, cleaned data into Amazon S3. We use Amazon S3 here again because Amazon Redshift can load the data in parallel from Amazon S3, using multiple threads from each cluster node. Amazon S3 also provides a historical record and serves as the formatted source of truth between systems. Data on Amazon S3 can be consumed by other tools for analytics if additional requirements are introduced over time.

4. Amazon Redshift loads, sorts, distributes, and compresses the data into its tables so that analytical queries can execute efficiently and in parallel. As the data size increases over time and the business expands, you can easily increase capacity by adding more nodes.

5. To visualize the analytics, you can use Amazon QuickSight or one of the many partner visualization platforms that connect to Amazon Redshift using ODBC or JDBC. This point is where the CEO and her staff view reports, dashboards, and charts. Now executives can use the data for making better decisions about

company resources, which ultimately increase earnings and value for shareholders.

You can easily expand this flexible architecture when your business expands, opens new channels, launches additional customer-specific mobile applications, and brings in more data sources. It takes just a few clicks in the Amazon Redshift Management Console or a few API calls.

Conclusion

We are seeing a strategic shift in data warehousing as enterprises migrate their analytics databases and solutions from on-premises solutions to the cloud to take advantage of the cloud's simplicity, performance, and cost-effectiveness. This whitepaper offers a comprehensive account of the current state of data warehousing on AWS. AWS provides a broad set of services and a strong partner ecosystem that enable you easily build and run enterprise data warehousing in the cloud. The result is a highly performant, cost-effective analytics architecture that is able to scale with your business on the AWS global infrastructure.

Contributors

The following individuals and organizations contributed to this document:

- Babu Elumalai, solutions architect, Amazon Web Services
- Greg Khairallah, principal BDM, Amazon Web Services
- Pavan Pothukuchi, principal product manager, Amazon Web Services
- Jim Gutenkauf, senior technical writer, Amazon Web Services
- Melanie Henry, senior technical editor, Amazon Web Services
- Chander Matrubhutam, product marketing, Amazon Web Services

Further Reading

For additional help, consult the following sources:

- Apache Hadoop software library[18]
- Amazon Redshift best practices[19]
- Lambda architecture[20]

Notes

1 https://www.forrester.com/report/The+Forrester+Wave+Enterprise+Data+Warehouse+Q4+2015/-/E-RES124041

2 http://aws.amazon.com/streaming-data/

3 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html

4 http://docs.aws.amazon.com/redshift/latest/APIReference/Welcome.html

5 http://docs.aws.amazon.com/redshift/latest/mgmt/working-with-clusters.html#rs-about-clusters-and-nodes

6 http://docs.aws.amazon.com/redshift/latest/dg/c_redshift-and-postgres-sql.html

7 http://aws.amazon.com/redshift/partners/

8 https://aws.amazon.com/vpc/

9 https://aws.amazon.com/cloudtrail/

10 http://docs.aws.amazon.com/redshift/latest/mgmt/working-with-HSM.html

11 https://aws.amazon.com/kms/

12 http://aws.amazon.com/s3/pricing/

13 http://aws.amazon.com/redshift/pricing/

14 http://docs.aws.amazon.com/redshift/latest/dg/json-functions.html

15 https://aws.amazon.com/dms/

16 https://aws.amazon.com/redshift/partners/

17 http://aws.amazon.com/ec2/spot/

18 https://hadoop.apache.org/

19 http://docs.aws.amazon.com/redshift/latest/dg/best-practices.html

20 https://en.wikipedia.org/wiki/Lambda_architecture