How-To Library

Informatica

Amazon Redshift Connector Best Practices

Abstract

This article contains information about how to configure Amazon Redshift Connector, PowerExchange for Amazon Redshift, and PowerExchange for Amazon Redshift for PowerCenter to get the best performance and efficiency.

This document captures the concepts, best practices, and recommendations for tuning Informatica Cloud, Big Data Management, and PowerCenter for using Amazon Redshift. These best practices have been derived from real-world experiences that covers different use cases.

Supported Versions

- Informatica Cloud® Amazon Redshift Connector
- PowerExchange® for Amazon Redshift 10.2 or later
- PowerExchange® for Amazon Redshift for PowerCenter® 10.2 or later

Table of Contents

Overview

You can use Amazon Redshift and Informatica Cloud, Big Data Management, or PowerCenter to rapidly and cost-effectively build a cloud first data warehouse. Then, extend your data warehouse to any on-premises or to other data sources. With the connectivity and power of this solution, you can deliver the data-driven agility required for business success today and tomorrow.

Amazon Redshift is one of the first data warehouse to offer a massively parallel, distributed, columnar, relational data store that is incredibly fast and can analyze billions of rows of data. You can create an Amazon Redshift cluster in few minutes and can scale as data volumes increases easily.

While data warehouse on cloud offers power and scale for analyzing petabytes of data, you still need to apply the fundamental data management processes such as data integration, data quality, data masking, data governance, master data management to the data present in the data warehouse. Transferring of on-premises data to the cloud is a big undertaking task that needs to be done efficiently to maximize the benefits of the data warehouse. Loading large data sets take a long time and consumes huge computing resources. The data integration workflows such as ETL or ELT job needs to run quickly and load the data sets in the right format into the data warehouse to avoid slow analytic queries.

Informatica Intelligent Enterprise Cloud Data Management offers a complete data management solution for Amazon Redshift that covers data integration (ETL or ELT), data quality, data masking, data security, streaming, mass migration, master data management, and data governance.

You can enhance the security to secure and protect the data. You can optimize the performance and allows optimum utilization of the system resources to achieve fast query execution when you read data from the source and write data to the target. The following lists the features that you can use to optimize the performance:

- Security enhancements
- Transformation Tunning
- Partitioning
- Distribution Key, Primary Key, and Sort Key
- Analyze and Vacuum target table

Amazon Redshift Overview

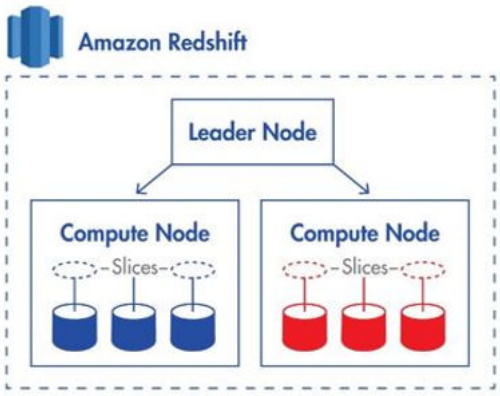
Amazon Redshift is a cloud-based petabyte-scale distributed data warehouse that the organization uses to analyze and store data.

Amazon Redshift uses columnar data storage, parallel processing, and data compression to store data and to achieve fast query execution. Amazon Redshift distributes data across a cluster of compute nodes and processes the nodes in parallel.

Amazon Redshift uses a cluster-based architecture that consists of a single leader node and compute nodes. The leader node manages the compute nodes and communicates with the external client programs. The leader node interacts with the client applications and communicates with compute nodes. The leader node takes query requests such as JDBC or ODBC interface, performs query planning, and then distributes the SQL query across compute nodes for massive scale and performance.

A compute node stores data and runs queries for the leader node. Any client that uses a PostgreSQL driver can communicate with Amazon Redshift. The ANSI standard SQL interface makes it easy to point Business Intelligence and Analytics Visualization tools at Amazon Redshift for interactive analytic queries.

The following image shows the overview of an Amazon Redshift cluster:



ETL and ELT Overview

ETL (Extract, Transform, and Load) and ELT (Extract, Load, and Transform) are methods used to transfer data from a source to a data warehouse.

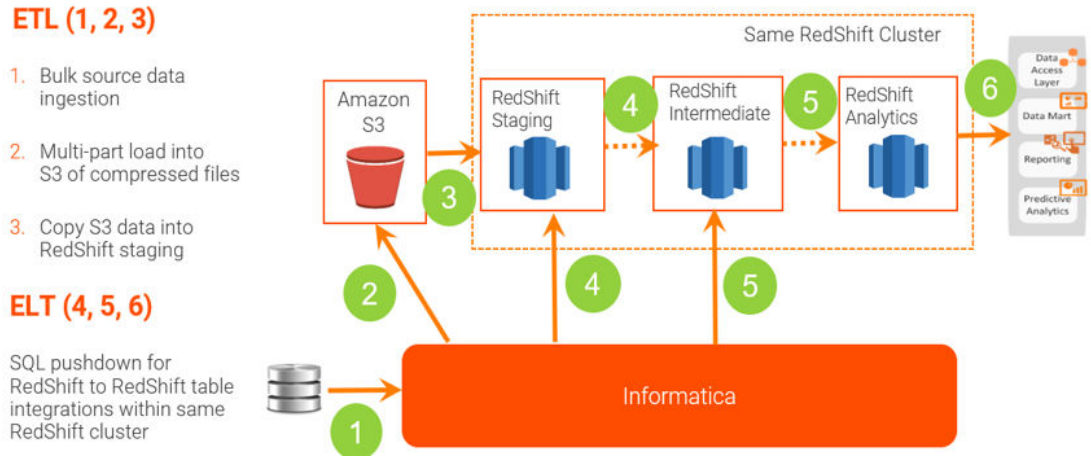
The ETL approach is ideal for transferring external data from either a single source or combined heterogeneous sources into Amazon Redshift cluster. Amazon Redshift recommends that you load the data to Amazon Redshift through Amazon S3 staging location as this approach is the fastest data ingestion option into Amazon Redshift. The ETL approach reads and writes data at high speeds. This allows faster data processing and performing complex calculations in less time to read from or write to disk.

Note: In the ETL approach, memory space of the staging location is the only limiting factor.

However, after loading the data to the Amazon Redshift cluster, you must use a different data transformation approach to transform. Then, move the data across staging location, intermediate, and analysis Amazon Redshift tables in the same Amazon Redshift cluster. Once the data is in the Amazon Redshift cluster, the ETL pattern of moving data in and out of Amazon Redshift tables within the same Amazon Redshift cluster through Amazon S3 is often sub-optimal. Use the ELT approach of pushing the data transformation operations to execute as SQL operations by the Amazon Redshift layer to optimize the process. The ELT approach optimises the process because the data is already available in the data warehouse and the data transformation operations such as filter, join, aggregation, or sorting are supported at the SQL layer enabling faster data processing.

In Informatica, the ELT approach is referred to as SQL Pushdown and Amazon Redshift Full SQL Pushdown is supported in the Informatica Cloud platforms. Informatica supports both ETL as well as ELT approach and provides the option to the users to choose based on the scenario.

The following image illustrates the ETL and ELT approaches:



Use Cases

The database administrators, who are well-versed with relational database constructs find it easy to create and provision new Amazon Redshift nodes and clusters. However, when you load data to Amazon Redshift, writing manual SQL scripts can make the loading hard to be agile and adapt to rapidly change the requirements from business stakeholders.

The following are the use cases:

- Batch Data Integration
- Streaming Data into Amazon Redshift
- Mass Ingestion task to upload data to Amazon S3 and Amazon Redshift

Batch Data Integration

When the data warehouse is in the same network as the source data, the data flow patterns can involve multiple interactions with the data warehouse.

Even though data are loaded to the data warehouse in batches, data are loaded to the data warehouse in a record-level interaction such as lookups or customized update statements. With the data warehouse on cloud, it is important to address the record-level interactions to ensure if you can perform read or write operation in bulk. As Amazon Redshift is best suited for bulk insert, update or read operation, you must design the mapping to perform such bulk operations.

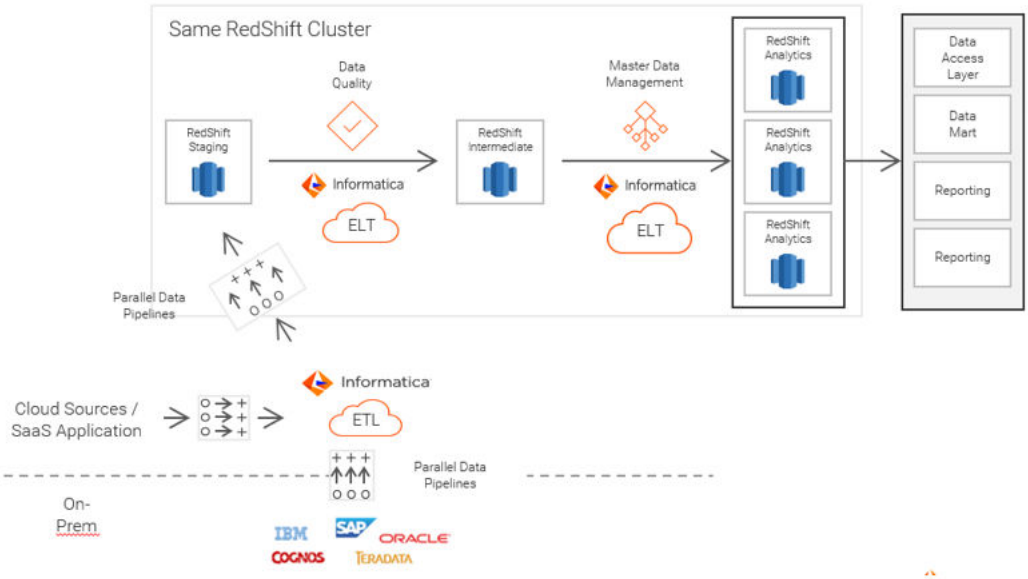
The recommended data load patterns for the data warehouse involves staging of data from the source first by transforming the data before or after loading (ETL or ELT) to the staging location. Informatica conforms to the Amazon Redshift best practices and implements several of the recommendations in the Amazon Redshift Connector, PowerExchange for Amazon Redshift adapter, and PowerExchange for Amazon Redshift for PowerCenter adapter.

The data fetched from on-premises and cloud sources are staged in Amazon Redshift mostly with few transformations such as filters that makes sure that only the required data are fetched. By default, data are staged on Amazon S3. However, Informatica recommends that you stage the data to Amazon Redshift directly.

Note that the staging of data to Amazon S3 is an internal step performed to follow the AWS recommendations for performance. In this case, we are discussing about the logical staging of source data in an Amazon Redshift table. You can even choose to use an SQL query when you load the data. Such staging is performed by splitting the data into multiple threads and loading the data to the staging area in parallel. Informatica also uses the node and slice information to configure the number of threads.

For all source data that resides on cloud, you can use the Informatica Secure Agent. For any data that resides inside the firewall, Informatica recommends that you download the Informatica Secure Agent. Once the data is in the staging area, depending on the design, the data can be either processed through an intermediate steps or loaded directly into the data warehouse table. These steps can be designed using Mapping Designer and at the run time. You can choose to run the mapping either using Informatica Secure Agent (ETL) or Amazon Redshift SQL (ELT).

The following image illustrates the batch data integration:



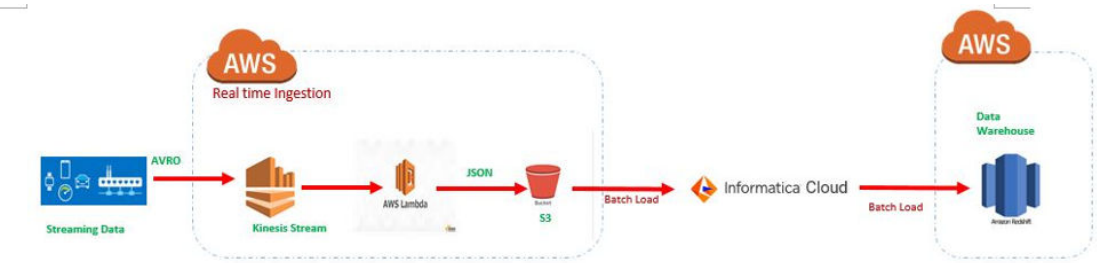
While the image shows an Amazon Redshift staging storage and an intermediate Amazon Redshift storage, the staging of the data largely depends on the data transformation patterns. In most of the cases, you might need to stage the data in one of the staging locations.

Streaming Data into Amazon Redshift

Streaming messages are ingested by Informatica Edge Data Streaming using Kinesis or Kafka processed by Informatica Big Data Streaming.

You can perform streaming data processing to enrich or refine the messages and then use Kinesis to consume the steaming data from Informatica Big Data Streaming. Kinesis can be configured to write the steaming data to Amazon S3 from where the steaming data undergoes further batch processing through Informatica Big Data Streaming or any other tool.

The following image illustrates the streaming data integration into Amazon Redshift:



Mass Ingestion Task to Upload Data into Amazon S3 or Amazon Redshift

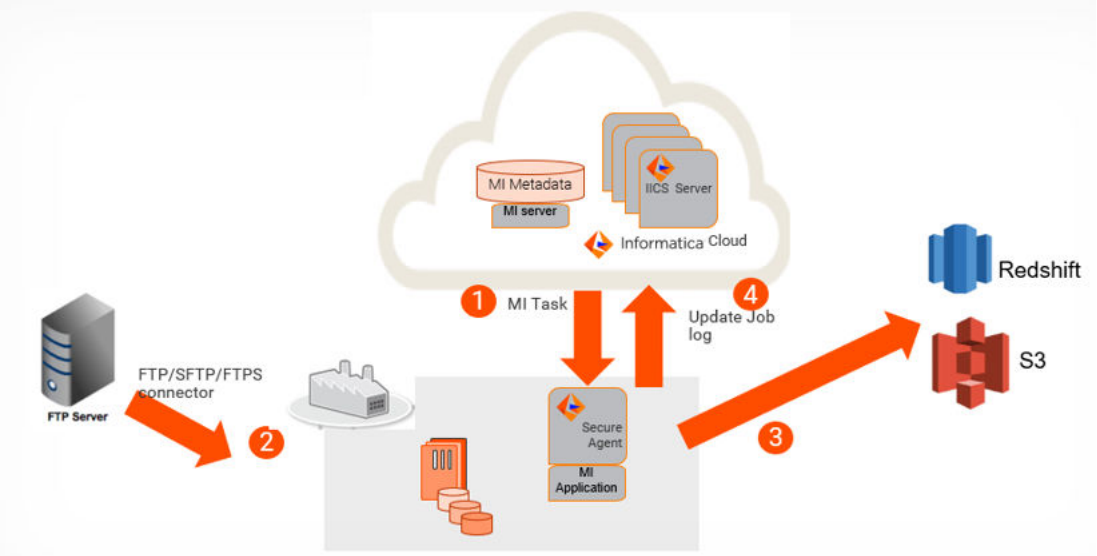
On cloud, as more and more companies wants to create data lakes in addition to the data warehouse, it has become a common requirement to upload data from various sources to a data lake, such as a data lake on Amazon S3.

Once the data is available in the data lake, you can process data and load the data in the data warehouse. In the data warehouse, the data are stored in a structured way that is more suitable for conventional analytics and data scientists can use the data lake for huge data analyses.

In addition to the data that is in applications, such as ERP, CRM, or database, nowadays most of the companies store data in various file formats, such as Avro, CSV, ORC, JSON, and Parquet. It is common to have large amount of files that are generated or received from the third-parties or other applications on a regular basis. You typically want ongoing loads set up for such files that can provide Managed File Transfer (MFT) features as data are loaded to the data lake.

On cloud, you can use mass ingestion tasks on Informatica Intelligent Cloud Services to upload a large number of files of any file type between on-premises and cloud repositories, and track or monitor file transfers. You can upload files from any file container, such as local folders, FTP folders, or Amazon S3 bucket using a name or expression pattern. You can also upload files to Amazon S3 or Amazon Redshift staging location directly at once, instead of moving single row of data separately. When you create a mass ingestion task to upload files, you can perform the Managed File Transfer (MFT) features, such as encryption, compression, override, and rename.

The following image illustrates the mass ingestion task to upload data to Amazon S3 or Amazon Redshift staging location:



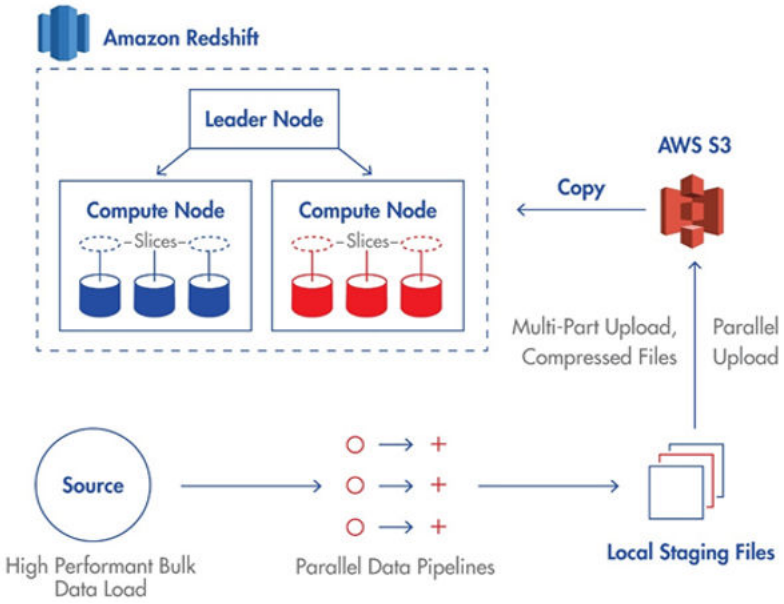
Amazon Redshift Connectivity Architecture Overview

When you use Amazon Redshift Connector, PowerExchange for Amazon Redshift, and PowerExchange for Amazon Redshift for PowerCenter to read data from sources, the data are read using multiple partitions that can be configured either using specific values, ranges of values, or partition specific queries. Then, the data are processed in parallel to apply any transformations that you configure.

The metadata are fetched from Amazon Redshift based on the number of slices in the compute nodes and splits the data accordingly. The source side partitions can split the data into several stage files that are written in parallel.

The files are loaded into Amazon S3 staging location in parallel with each file uploaded as a single part of multi-part based on the size. Then, loads the data from these files into the Amazon Redshift table by running the COPY command.

The following image shows how data is read from the source, loaded in the Amazon S3 staging file, and written to the Amazon Redshift target:



Security

You can secure and protect the data when you read data from the source and write data to the target. You can enhance the security for the Amazon Redshift clusters, data in-transit, and data at-rest.

Sign-In Credentials

At the database level, users are authenticated using the user name and password credentials when the connection is establish with Amazon Redshift.

The users can be of the following types:

- **Superuser**: A superuser user is the master user name that you create when you launch the Amazon Redshift cluster. Using this user name, you can query system tables, views, or catalog tables.
- **User**: A user name that is used to access the cluster. Based on privileges of the user, you can perform select, insert, update, delete, create, or reference operations.

The following image shows the sign-in credentials page:

Access Management

At cluster level, there are two security management options available to prevent unauthorized access.

The following lists the security management options:

- **Role Based Access**: With AWS Identity and Access Management (IAM) Role, you can securely control access to Amazon Redshift resources for the users in your AWS account. If you have multiple users that require access to Amazon Redshift, create an IAM Role for each users and provide a specific access policies to each users. You do not have to share your credentials. For more information about how to create an IAM Role and add IAM Role to a cluster, see the AWS documentation.
- **Key Based Access**: You can provide the **Access Key ID** and **Secret Access Key** for all the users with IAM Role that are authorized to access the Amazon resources that contains the data.
  The following image shows the **Access Key ID** and **Secret Access Key** connection properties:

  Note: Informatica recommends that you use the IAM Role based access option to access resources and sensitive data.

VPC

To protect the cluster access using a virtual networking environment, you can launch the cluster in an Amazon Virtual Private Cloud (VPC).

You can use the Amazon S3 VPC endpoints. Amazon S3 VPC endpoints are easy to configure, highly reliable, and provides a secure connection to Amazon S3 that does not require a gateway or NAT instances.

The EC2 instances that runs in the private subnets of a VPC have controlled access to Amazon S3 buckets, objects, and API functions that resides in the same region as the VPC. You can use an Amazon S3 bucket policy to indicate which VPCs and VPC endpoints have access to the Amazon S3 buckets.

SSL Connections

You can use the Secure Sockets Layer (SSL) connections to encrypt data and validate the server certificate that you connect to.

The SSL connections ensure that all JDBC calls to Amazon Redshift, such as data preview, UNLOAD, and COPY commands goes through the SSL encrypted connection. For more information about how to set up Secure Agent to support SSL enabled connections with Amazon Redshift, see the AWS documentation.

Data Encryption

You can encrypt data while loading the table data as staging files to Amazon S3.

When you read data from an Amazon Redshift source, you can select one of the following encryption types:

- Server-side encryption with Amazon S3-managed keys (SSE-S3)
- Client-side encryption with a customer-managed key (CSE-CMK)

When you write data to an Amazon Redshift target, you can select one of the following encryption types:

- Server-side encryption with Amazon S3-managed keys (SSE-S3)
- Server-side encryption with AWS KMS-managed keys (SSE-KMS)

- Client-side encryption with master symmetric key.

To encrypt data using AWS KMS-managed keys, you must provide the customer master key ID when you configure the Amazon Redshift connection.

Amazon S3 Upload Optimizations

When you upload data to Amazon S3, you can optimize the performance.

To optimize the performance, use the following methods:

Compress data before uploading to Amazon S3

When you read data from a source, local staging files are created before you upload the data to Amazon S3. You can use the multiple compression format to compress the data. When you write data to the target, the files are decompressed. Even though compression and decompression process is a CPU-intensive task, you can optimize the performance when you load large amounts of data to Amazon S3.

Encrypt data before uploading to Amazon S3

You can encrypt the data before you upload the data to Amazon S3. You can use either the client-side or server-side encryption to encrypt the data when you write data stored in Amazon S3 to Amazon Redshift target.

Uploads file to Amazon S3 in parallel

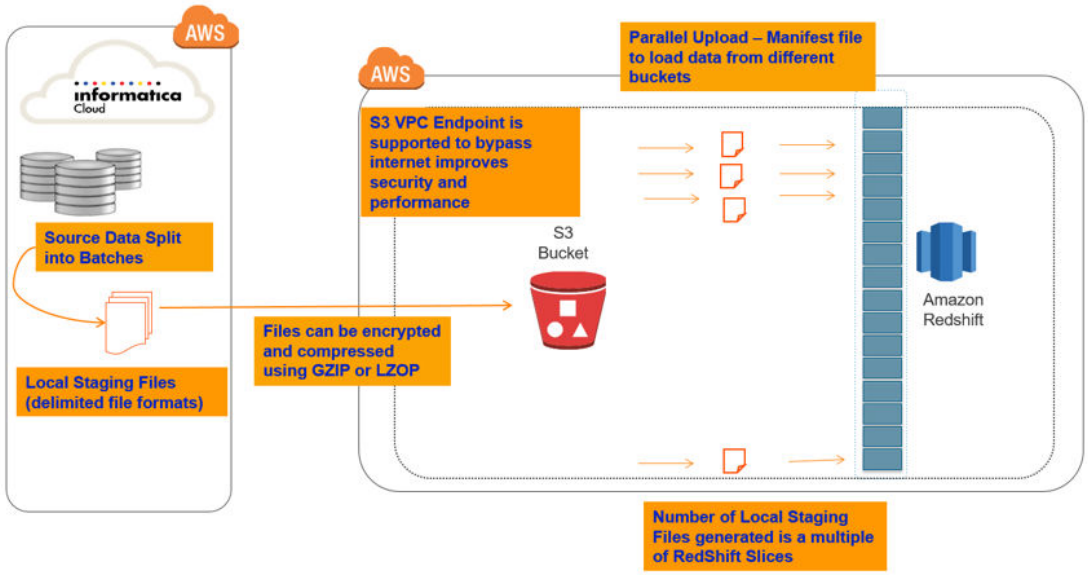
You can upload files to Amazon S3 in parallel. The Secure Agent, Data Integration Service, or PowerCenter Integration Service uses the data available in the compute nodes of the Amazon Redshift cluster to determine how to split the files and upload the files to Amazon S3. The number of files staged and uploaded in parallel is a combination of the partitions used in the mapping and the number of slices in the compute nodes.

The Secure Agent, Data Integration Service, or PowerCenter Integration Service processes data in parallel for the partitioned mapping.

Parallelizing Copy to Amazon Redshift

When you write data from Amazon S3 to Amazon Redshift target, you can split up the data into multiple slices that are equal to or greater than the number of slices in the cluster. Then, load the data in parallel with each slices taking the data from its own dedicated file.

The following image shows how the data are uploaded to Amazon S3 and written to Amazon Redshift target:



SSL Configuration on Cloud

On cloud, you must configure the Secure Agent to support an SSL connection to Amazon Redshift.

Perform the following step to configure the Secure Agent:

1. Download the Amazon Redshift certificate from the following location:

   https://s3.amazonaws.com/redshift-downloads/redshift-ssl-ca-cert.pem
2. Run the following command to add the certificate file to the key store:

   ```
   ${JAVA_HOME}/bin/keytool - keystore {JAVA_HOME}/lib/security/cacerts -import -alias
   <string_value> -file<certificate_filepath>
   ```
3. On the Data Integration, click **Configure** > **Runtime Environment**.
4. In the **Actions** tab, click **Edit**.

   The **Edit Secure Agent** page appears.
5. In the **System Configuration Details** section, set the value of the **Type** to **DTM**.
6. Click **Edit Agent Configuration** icon and add the following value for **JVMOption1**:

   ```
   Djavax.net.ssl.trustStore=<keystore_name>
   ```
7. Click **Edit Agent Configuration** icon and add the following value for **JVMOption2**:

   ```
   Djavax.net.ssl.trustStorePassword=<password>
   ```
8. Add the following parameter in the `JDBC URL` that you specify in the Amazon Redshift connection property:

   ```
   ssl=true
   ```

   For example,

   ```
   jdbc:redshift://mycluster.xyz789.us-west- 2.redshuft.amazonaws.com:5439/dev?ssl=true
   ```
9. Click **OK**.

Transformation Tuning

When you initialize a mapping, blocks of memory are allocated to hold the source and target data. If enough memory blocks to hold the data are not allocated, the mapping fails.

The DTM buffer is a temporary storage area used to store temporary data for caching data used by transformation. The buffer is divided into blocks. The buffer size and block size for a mapping are tunable. If the performance of read, write, or any transformations are slow, this is an indication that DTM buffer tuning is required.

You must tune the DTM buffer size and buffer block size for optimal throughput in the following manner:

Default Buffer Block Size

To adjust the buffer block size, use the following code:

```
Buffer Block Size = sum of all column widths (in bytes) * n,
 where n is a positive integer > 1
```

In the advanced session property, select the **Default buffer block size** option and set the required value. The following image shows the **Default buffer block size** advanced session property:

DTM Buffer Size

To adjust the DTM buffer size, use the following code:

```
DTM Buffer Size = Buffer Block size * m,
 Where m is a positive integer >1
```

In the advanced session property, select the **DTM Buffer Size** advanced session propertyoption and set the required value. The following image shows the **DTM Buffer Size** advanced session property:

Note: The DTM buffer size is based on the size of the row and data. If data partitioning is enabled, the DTM buffer size is the total size of all memory buffer allocated to all partitions. For a mapping that contains `n` numbers of partitions, set the value of the **DTM Buffer Size** option to at least `n` times for the mapping with one partition.

Partitioning

On cloud and PowerCenter, you can partition the data to optimize the performance and allows optimum utilization of the system resources.

Apart from configuring the DTM buffer size, buffer block size, and batch size to optimize the performance, you can further improve the performance at the external sources, targets, or transformations level. External source or target connection performance is more dependent on external application optimization, network latency, DTM buffer block size, and commit interval. The transformation level optimization is related to I/O operations and partitions, or partitions points.

Informatica supports creation of parallel data pipelines with a thread-based architecture. The partitioning of data across the parallel data pipelines is handled automatically. The partitioning option executes optimal parallel mapping by dividing the data processing into subsets, which runs in parallel and are spread among available CPUs in a multiprocessor system.

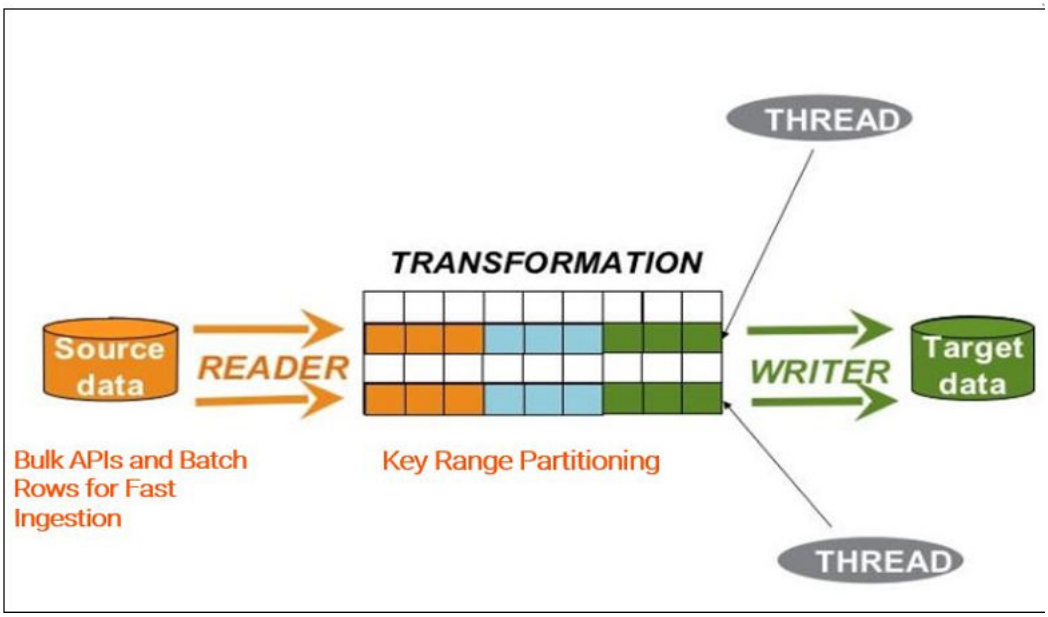
Unlike the approaches that require manual data partitioning, the partitioning option automatically guarantees data integrity because Informatica provides parallel engine that dynamically realigns data partitions for set-oriented transformations. Configurable mapping options, such as error handling, recovery strategy, memory allocation, and logging makes the mapping easier to gather statistics used to maximize the performance. By enabling the hardware and software to scale for handling large volumes of data and users, the partitioning option improves the performance and productivity.

Amazon Redshift Connector supports pass-through and key-range partitioning. PowerExchange for Amazon Redshift for PowerCenter supports pass-through partitioning.

Informatica partitioning for Amazon Redshift follows the principle of serializable isolation. This enable you to run two concurrently transactions T1 and T2, giving the same results as at least one of the following:

1. T1 and T2 run serially in that order.
2. T2 and T1 run serially in that order.

The following image shows how the source data are partitioned and loaded to the target:



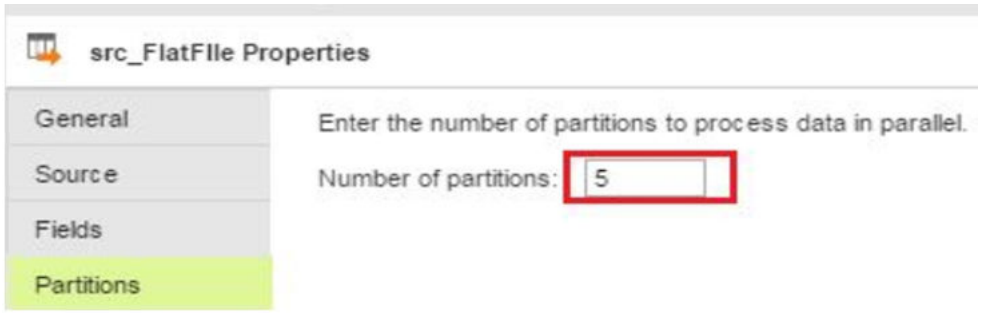
Partitioning on Cloud

When you read data from an Amazon Redshift source, Informatica recommends that you use the key-range partitioning. The key can be based on a column or a set of columns that are distributed across partitions evenly and across the compute nodes of the Amazon Redshift cluster.

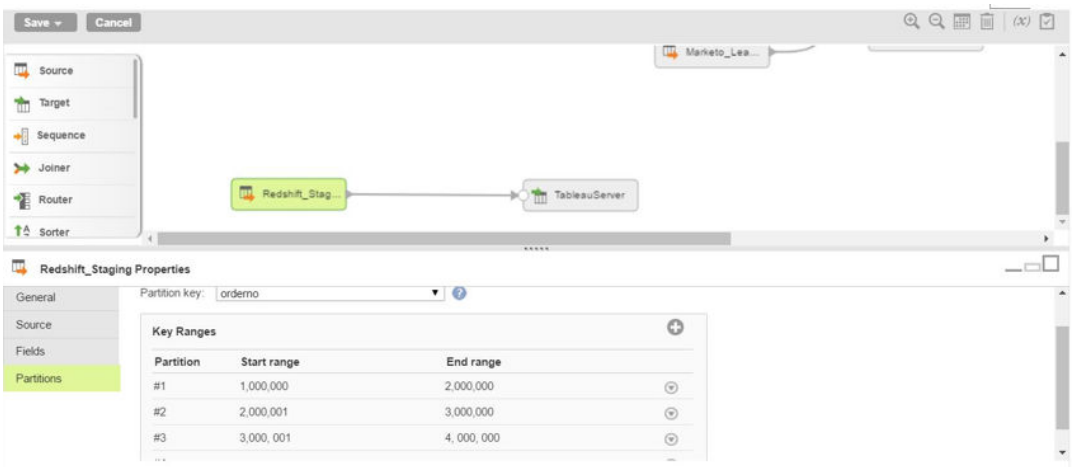
When the key column that you select for the source partitioning is same as the distribution key of the Amazon Redshift cluster, the Secure Agent reads the data faster.

Amazon Redshift Connector supports the following partitions:

- **Pass-through partitioning**: You can specify the number of partitions when you read flat file sources. The following image shows an example of the **Pass-Through Partitioning** tab:



- **Key-Range partitioning**: With key-range partitioning, the number of source partitions you need to create depends on the Amazon Redshift cluster workload management concurrency setting. By default, the value is 5. If the value is set to the default value, you must create five partitions on the source side. The following image shows the **Key-Range Partitioning** tab:



Partitioning on PowerCenter

If you need to extract a large amount of source data, you can partition the sources to improve session performance. Partitioning sources allows the PowerCenter Integration Service to create multiple connections to sources and process partitions of source data concurrently. You can partition sources if the PowerCenter Integration Service can maintain data consistency when it processes the partitioned data.

By default, the Workflow Manager sets the partition type to pass-through for Amazon Redshift tables. In pass-through partitioning, the PowerCenter Integration Service passes all rows at one partition point to the next partition point without redistributing them.

If you create multiple partitions for an Amazon Redshift source session, the PowerCenter Integration Service evaluates the session properties in the following order to run the session:

- SQL Query
- INFA Advanced Filter
- Slices on Amazon Redshift nodes

Data Compression

Amazon Redshift is a columnar data warehouse in which each columns are stored in a separate file. As the data types of the data are the same in a column, you can apply the column encoding compression strategies.

Applying compression to Amazon Redshift columns saves storage space, reduces disk I/O, and improves query performance. Amazon Redshift provides compression strategies and by default applies no compression to the columns.

You can specify the following options for when you apply column compression encoding in the Amazon Redshift columns:

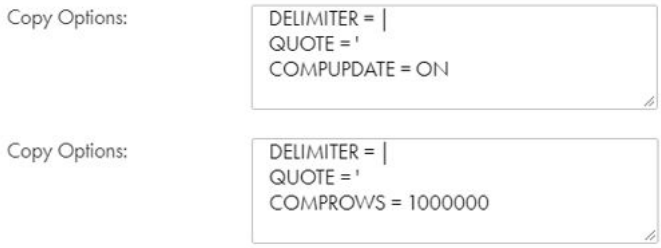
Automatic compression through COPY command in design time

You can specify the COPY command option to load data from Amazon S3 to Amazon Redshift cluster. The COPY command automatically determines the best compression encoding for the column when there is an empty table with no compression encoding specified to load the data to Amazon S3.

When you apply the compression encoding, the first 100,000 rows are read and the best compression encoding for the columns are determined. Then, the compression encoding is applied to the entire table.

You can use the following configuration parameters for managing the automatic compression through COPY command:

- Set `COMPUPDATE=ON` to override the compression if you have defined the compression encoding.
- Set `COMPROWS` to override the default 100,0000 rows.
  The following image shows the **Copy Options** properties:



Set the compression encoding while creating a target in the design time

You can specify the compression encoding manually for the Amazon Redshift columns when you create a target and specify the target schema definition at design time.

Run the ANALYZE command manually

You can run the ANALYZE command manually when the Amazon Redshift table is idle. When you run the ANALYZE command, the compression analysis is performed and a report is generated with the specified compression encoding for the tables.

Keys

As Amazon Redshift is a columnar data warehouse storing each columns in a separate file, you can specify a distribution key and sort key for a very large dimensional table to optimize the performance. You can also specify if the column is a primary key or foreign key to uniquely identify a record in the Amazon Redshift table.

Distribution Keys

In an Amazon Redshift table, every column is an index for each column. Even though, there are no index keys to be specified, you can specify distribution key for partitioning.

Amazon Redshift contains a distributed data architecture where data are vertically partitioned across the Amazon Redshift compute nodes and each compute node in the cluster stores a subset of the data.

You can specify a column as a distribution key by considering the common joins and aggregation keys against a set of table that have data located in the same compute node. Specifying a distribution key in this manner minimizes the data movement over the network. Informatica recommends that you use this option for a very large dimensional table to distribute both the dimension and tables associated with it on the join column.

In addition to the distribution key, when you create a target table, you can select the following options to specify the distribution type to distribute the data:

Even distribution

Even distribution distributes the data uniformly across compute nodes in a round-robin style. Informatica recommends that you use this option for the tables that are not joined with the other tables or the tables that are only joined to tables in which the `ALL` distribution option is specified.

For example, a table that is joined with a smaller dimensional table in which the `ALL` distribution option is specified for the small dimensional table.

ALL

You can specify the `ALL` distribution option to copy the table to all the nodes. Informatica recommends that you use this option for a smaller dimensional tables that needs to be joined frequently with a large distributed tables. For example, a date dimension with a few thousand entries.

Sort Keys

In an Amazon Redshift table, you can specify one or more columns as a sort key to sort data in a table.

Amazon Redshift stores columnar data in 1 MB disk blocks. The minimum and maximum values for each block is stored as part of the metadata. Sorting the data enables you to execute the queries faster with predicates, such as filters and joins that reference the sort key.

Note: You can have only one sort key per table.

If the sort key is derived from a source table column, you can read the source data in the sorted order and then upload the data in the sorted form to Amazon Redshift. This process saves the sorting time in an Amazon Redshift table. However, sorting the data before loading helps only if the sort key is the same table as source table column. Otherwise, you do not have to sort the data before you load the data to Amazon Redshift.

Primary & Foreign Keys

You can specify primary and foreign keys in an Amazon Redshift table to uniquely identify a record in the table.

Amazon Redshift query planner uses the primary and foreign key relationships in the query planner. When you perform an upsert operation, you must define a primary key.

Analyze and Vacuum Target Table

After you load a large amount of data in the Amazon Redshift tables, you must ensure that the tables are updated without any loss of disk space and all rows are sorted to regenerate the query plan.

The leader node uses the table statistics to generate a query plan. The query plan might not be optimal if the table size changes. When you run an update operation in an Amazon Redshift table, the data are deleted and then inserted. However, the records are not deleted. Instead the records are marked as delete that results in consuming spaces. Similarly, when you run an append operation, the tables are sorted periodically if the append order is not inline with the sort key. The newly inserted, updated, and deleted rows are sorted in a separate unsorted partition.

You must run the ANALYZE command to ensure that the query planner on Amazon Redshift updates the statistical metadata to build and choose optimal plans to improve the efficiency of queries. You can run the ANALYZE command during design time to update the table statistics after running the ETL job on the entire table or to specific columns that are frequently used in sorting and joins.

Additionally, in the **COPY command** property, you can set the value of the `STATUPDATE` option to `ON` to run the ANALYZE command and perform an analysis.

When the table changes or is updated, there might be free spaces and unsorted rows in the table. You must set the **Vacuum Target Table** property to restore free spaces and to sort the rows in a specified table or all tables in the database. This results in a faster query performance.

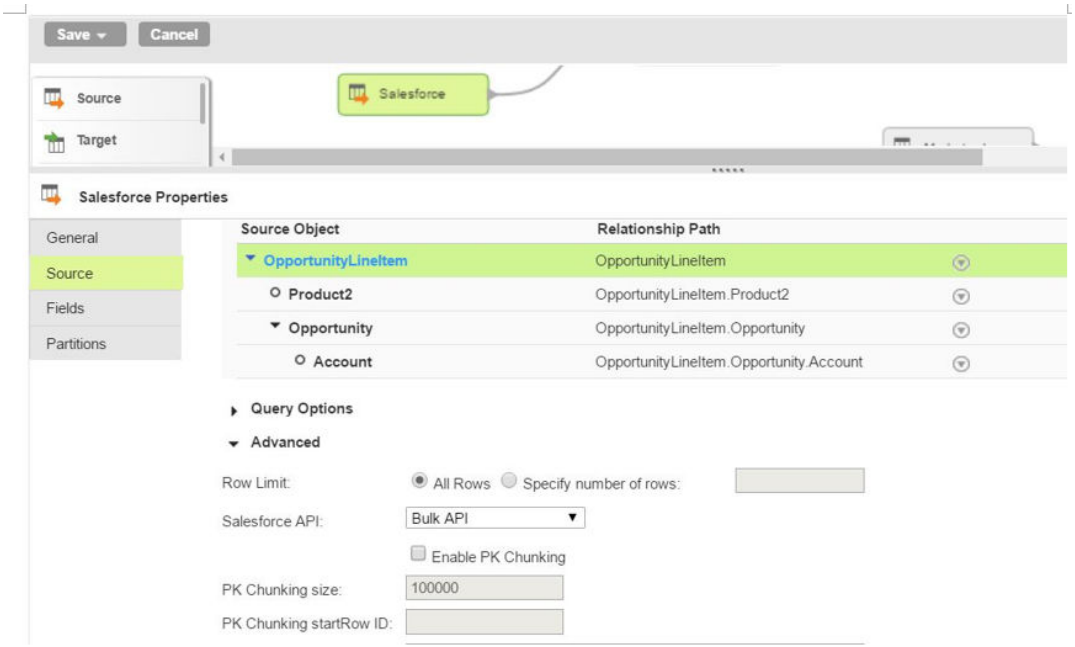
Reading Data From a Non-Amazon Redshift Source

The ODBC and JDBC drivers are best for querying and to perform read or write operations for a lesser amount of data. However, for a native application specific integration, you need a faster performance.

Informatica offers native connectors that supports optimized integrations to read data from a non- Amazon Redshift source faster.

Use the following recommendations when you read large volumes of source data:

- On cloud, use the **Bulk API** option.
  The following image shows the Bulk API option when you read data from a Salesforce source:



- If you want to read data from a single table, use the select statements with an `ORDER BY` or `GROUP BY` clause. You can optimize the time to read bulk data by adding indexes.
- Use the SQL filter options to reduce the volume of data.
- If you read data by joining multiple source tables in one Source Qualifier, you can improve the performance by optimizing the query with optimizing hints. The database administrator can create optimizer hints to inform the database on how to execute the query for a set of source tables. The query used to read data appears in the session log. The database administrator can analyze the query, create optimizer hints, and indexes for the source tables.

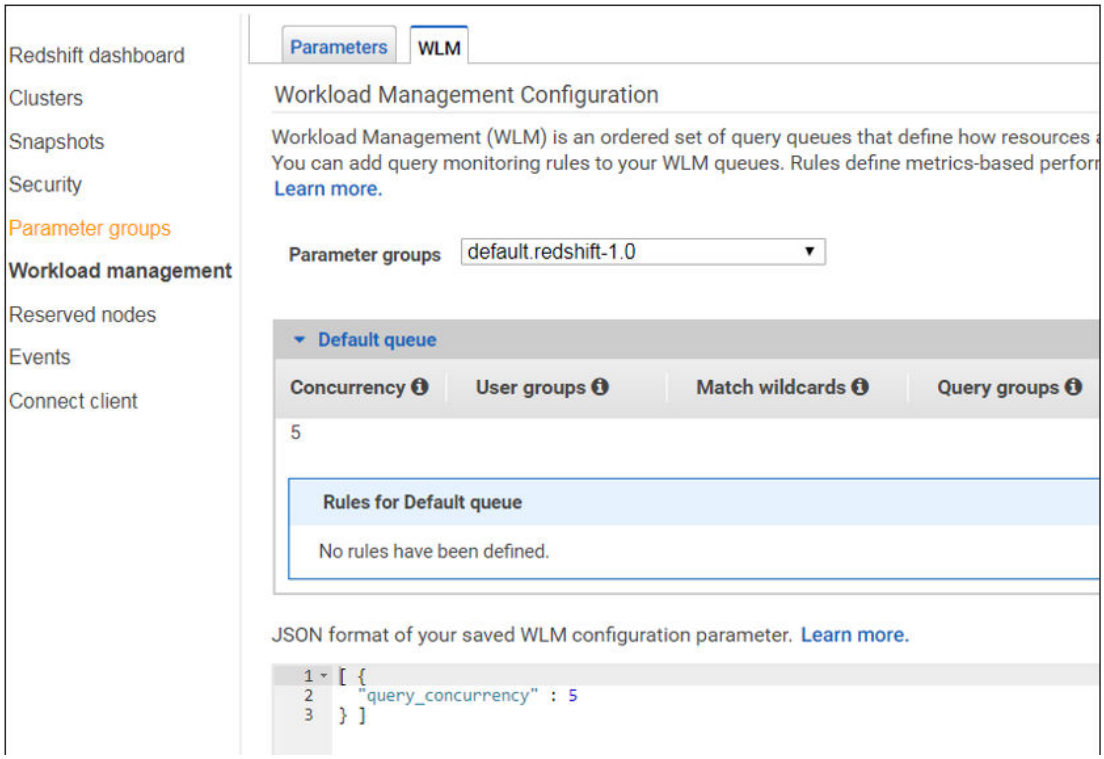
Configure Amazon Redshift Workload Management

In the Online Transaction Processing systems (OLTP), most queries are of uniform size and have similar execution cost for all jobs. However, in the case of data warehouse, the queries vary greatly in the execution cost, time, and size of the result set.

Some are interactive queries that have strict SLAs while others are adhoc queries analyzing large volumes of data that takes longer time to complete the task or session. The ETL jobs are differentiated according to the job execution, such as the incremental updates that takes shorter time while initial loads or massive updates takes longer time.

There is a need to manage the priority, memory allocated, and concurrency of the different warehouse jobs. To ensure that the slow running queries are not blocking fast running queries, Amazon Redshift Workload Management divides the cluster availability resource into slots. Then, combines and arranges those slots into queues and assigns the incoming queries to the queues. You can configure priority, memory, and concurrency for queues through the Amazon Redshift Workload Management interface.

The following image shows the Amazon Redshift Workload Management Configuration page:



Rules and Guidelines for Configuring Amazon Redshift Workload Management

Consider the following rules and guidelines when you configure the Amazon Redshift Workload Management interface:

- Separate the data import jobs from the analysis queries and ensure that the jobs have the resources that they need to complete the queries.
- For non-ETL queries, you must set the default concurrency as five and assign sufficient memory to the concurrent analytic query jobs.
- You must separate the queues for batch jobs, adhoc queries, and interactive queries.
- You must separate the queues for ETL jobs.
- You must separate the queues for short and long running ETL jobs. The lower the concurrency of long running ETL jobs, specify a lower memory while short running ETL jobs can have concurrency lesser than one.

Pushdown Optimization

You can use pushdown optimization to push transformation logic to the source or target databases. Use pushdown optimization when you use the database resources to improve the task performance.

When you run a mapping configured for pushdown optimization, the mapping converts the transformation logic to an SQL query. The mapping sends the query to the database and then database executes the query. You can use full and source pushdown optimization for the ODBC connection type that uses Amazon ODBC Redshift drivers for mapping.

For a given mapping, the pushdown optimizer analyzes the transformation logic. If you select the full pushdown optimization to pushdown the data from the transformation to Amazon Redshift, the optimizer generates the appropriate SQL statement automatically.

When you perform an insert operation, the pushdown optimization generates the following sample transaction:

```
BEGIN;
CREATE TEMP VIEW PM_(…) AS SELECT (…) …
CREATE TEMP VIEW PM (…) AS SELECT (…) A JOIN (SELECT …) LEFT JOIN
INSERT INTO STG_CUSTOMER(…) SELECT (…)
INSERT INTO STG_CUSTOMER(…) SELECT(…)
DROP VIEW IF EXISTS …
DROP VIEW IF EXISTS …
COMMIT
```

The following lists the transformation types that you can use for the mapping configured for pushdown optimization:

- Two sources in multiple pipelines
- Expressions
- Lookups with SQL override
- Filters
- Routers
- Target

For more information about Amazon Redshift pushdown optimization, supported functions, and operators, see the following guides:

- Amazon Redshift pushdown optimization on cloud, see *Amazon Redshift Connector User Guide*.
- Amazon Redshift pushdown optimization on Big Data Management, see *PowerExchange for Amazon Redshift User Guide*.
- Amazon Redshift pushdown optimization on PowerCenter, see *PowerExchange for Amazon Redshift for PowerCenter User Guide*.

Benefits of Using Pushdown Optimization

The following lists the benefits of using pushdown optimization:

- **Performance**. The optimal and most appropriate load balancing between Informatica server and Amazon Redshift engine resulting in significant performance gains.
- **Productivity**. Utilizes the same metadata driven interface and architecture.
- **No coding**. Eliminates the need to write complex SQL statements or code.
- **Scale**. Effectively leverages high performance data warehouse engine.
- **Backward compatible and future proof**. Pushdown optimization works with latest version of Amazon Redshift engine. Future versions of ODBC pushdown optimization for Amazon Redshift incorporates new functionality.
- **SLA**. Makes data available on time.
- **OLTP acid properties**. The ODBC pushdown optimization combines all the statements into a single `BEGIN … COMMIT;` statement that ensures transaction control and data integrity.

Rules and Guidelines for Using Pushdown Optimization

The following lists the rules and guidelines for using pushdown optimization with Amazon Redshift:

- You must select the **Full Pushdown Optimization** option for the Amazon Redshift source and target table when you have large amount of data.
- Do not use unconnected lookups.
- Do not use parallel lookups for mappings. Instead, create a sequential lookups.
- Eliminate sorted aggregation and pass-through port in an aggregator.

- Eliminate variable ports.
- Always set the **Allow Temporary View for Pushdown** property when you use pushdown optimization with updates, multiple tables, custom SQLs, and look-ups.
- Tune the custom Amazon Redshift SQL for ODBC pushdown optimization using the Amazon Redshift Execution Plan.
- Filter data using `WHERE` clause before configuring the outer joins.
- Do not use full table scans for a large table.
- Use staging processing if required.
- Validate the use of sort keys. You can create a sort key on the columns that are most commonly used in the `WHERE` clause.
- To optimize the queries, ensure that you choose the right columns for the distribution option.
- If you have a Sequence Generator transformation, do not set the **Allow Temporary View for Pushdown** property.

Data Types and Load Patterns

You can read and write data of multiple types to an Amazon Redshift target.

The following lists the data types that you can write to an Amazon Redshift target:

Dimensional Data Warehouse

Dimensional data warehouse represents data as few facts, such as Invoice Amount and Ordered Quantity, or a large set of dimensions, such as Customer and Region. Each of the data are stored in their own tables with the artificial keys called as the surrogate keys.

Operational Data Store (ODS)

ODS stores data mostly not in the raw form. The stored data either have the data as a system of record, a database where any business record is stored in its most original form, or a simplified data set for a quick analysis. ODS maintains the original transactional table structures.

Other data

These kind of data are very common for the users to store additional data along with a data warehouse that does not fit in either of the two types. However, these data are used either by analysis or other applications.

Informatica offers a wide array of transformations that support all the types of data and load patterns. One of the example is the Slowly Changing Dimension load pattern that are used for the important dimension tables. Informatica provides wizards, templates, and sample mappings that shows how to configure such patterns.

Rules and Guildelines for Connecting to Amazon Redshift

Consider the following rules and guidelines when you connect to Amazon Redshift using Informatica Cloud, Big Data Management, and PowerCenter:

- Amazon Redshift Connector, PowerExchange for Amazon Redshift, and PowerExchange for Amazon Redshift for PowerCenter does not enforce primary key constraints.
- Do not use Amazon Redshift lookups that are not cached.
  If a lookup is not cached, each lookup action is performed in the Amazon Redshift table for each record that is processed. If a lookup is cached, the records for lookup are loaded in memory all at once avoiding the record level.

- Do not use the update override option that are issued by record. Updating an Amazon Redshift table is more efficient when the data are batched together. When you run a mapping to update an Amazon Redshift table, the tables are updated in batches. Instead, of overriding the updates, you must try to translate the logic that either maps the logic or splits the mapping to explore.
- Do not use the functions and expressions that are not support for pushdown optimization.
- Specify the distribution and sort keys on all tables.
- Ensure that the Amazon Redshift columns do not contain null values.
- The distribution and sort keys are more effective in Amazon Redshift compared to the traditional primary key constraints. In many cases, the existing primary keys from the table could be the right choice for the sort keys. However, you must refer to the Amazon Redshift documentation to ensure that these keys are configured in a way that the keys are most suited for the analysis.

Authors

Sara Miles

Subhashree Salam