Redshift to Databricks Migration Plan and Approach

# Introduction

This document details the structured plan and approach for migrating from Amazon Redshift to Databricks. The objective is to ensure a seamless transition of data, procedures, and functions with minimal disruption to ongoing operations, leveraging Databricks' advanced analytics and processing capabilities.

# Objectives

- Migrate Data: Seamlessly transfer data from Redshift to Databricks.  
- Recreate Procedures: Recreate essential procedures and functions in Databricks.  
- Ensure Data Integrity: Maintain data integrity and consistency during migration.  
- Automate Migration: Develop automation processes to streamline migration tasks.

# Tools and Technologies

- Ingestion Tools: Fivetran for efficient data extraction and transfer.  
- Storage: Proprietary storage with sharing mechanisms in Databricks.  
- Target Platform: Databricks on Delta Lake for optimized data storage and processing.

# Migration Phases

## Phase 1: Planning and Assessment

1. Inventory Assessment: Identify and catalog all data, procedures, and functions in the Redshift environment.
   1. The types of workloads (ETL, BI, ingress/egress, etc.) and their size by warehouses and databases
   2. The scope of data and queries/workloads to be migrated
   3. The upstream and downstream technologies and applications involved in the architecture
   4. The current security setup and protocols
   5. Estimates for infrastructure costs
2. Requirements Gathering: Define the technical and business requirements for the Databricks environment.
3. Gap Analysis: Identify and address gaps between Redshift and Databricks capabilities, such as handling stored procedures or user-defined functions.

## Phase 2: Data Migration

Data Extraction: Utilize Fivetran for data extraction from Redshift, ensuring minimal downtime.

Data Transformation: Transform data to conform with Databricks' schema and data structures.

Data Loading: Load transformed data into Databricks, ensuring optimal partitioning and storage formats (e.g., Delta Lake).

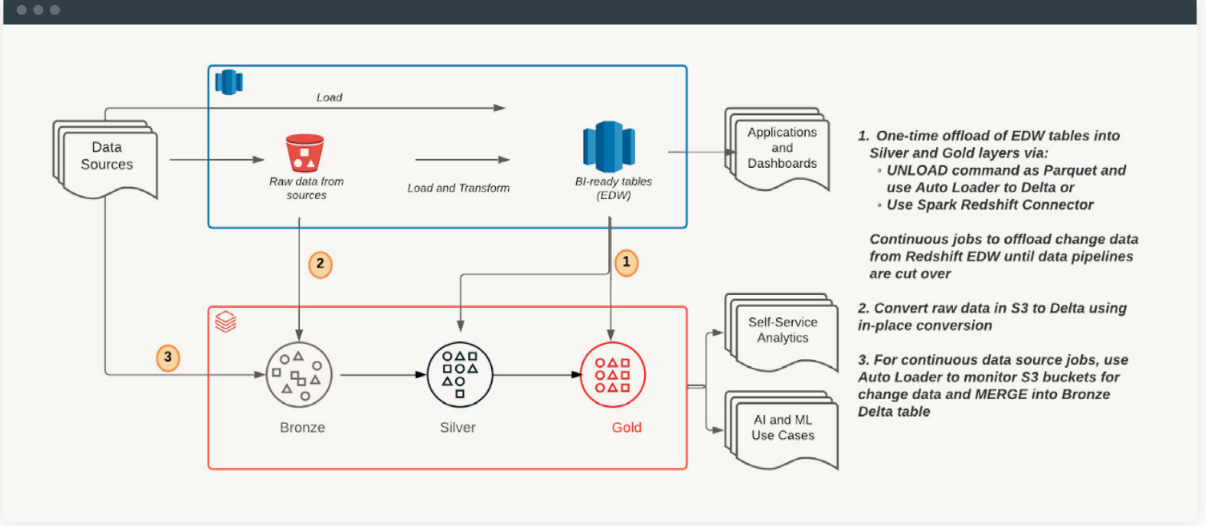
When migrating data out of Amazon Redshift, there are key questions that need to be

answered. Some of these could include:

What is the target design for the tables being migrated?

Should the destination retain the same hierarchy of catalogs, databases, schemas, tables?

Should there be any cleanup of duplicated data sets or organization of the existing data footprint in Amazon Redshift?



## Phase 3: Procedures and Functions Migration

1. Simple Procedures and Functions: Migrate and test simple procedures and functions in Databricks, refactoring as necessary.  
2. Medium Complexity Procedures: Recreate medium complexity procedures, optimizing for Databricks' distributed architecture.  
3. Complex Procedures: Re-engineer complex procedures to leverage Databricks' processing power and advanced capabilities.

## Phase 4: Testing and Validation

1. Data Validation: Compare and validate data between Redshift and Databricks to ensure accuracy and consistency.  
2. Functionality Testing: Rigorously test all recreated procedures and functions to confirm they perform as expected.  
3. Performance Testing: Benchmark Databricks performance, ensuring it meets or exceeds Redshift's performance metrics.

## Phase 5: Automation

1. Automation Scripts: Develop robust scripts for automating the migration process, reducing manual intervention.  
2. CI/CD Integration: Integrate migration scripts into the CI/CD pipeline for continuous deployment and testing.

## Phase 6: Deployment and Monitoring

1. Deployment: Roll out the migrated data, procedures, and functions to the production Databricks environment.  
2. Monitoring: Continuously monitor the Databricks environment for any issues post-migration, ensuring smooth operations.

# Tangible Outcomes

- Successful Data Migration: Complete migration of data, procedures, and functions to Databricks.  
- Automation Efficiency: Streamlined migration process through automation, reducing manual effort.  
- Enhanced Performance: Improved performance and scalability in the Databricks environment.

# Key Considerations for Optimizing Performance

**Data Partitioning and Clustering:** Optimize query performance through effective partitioning and clustering strategies in Databricks, leveraging Delta Lake.

**Data Transformation and Cleaning:** Ensure data quality with thorough transformations and cleaning during the migration process.

**Efficient Data Loading:** Use parallel data loading techniques and optimize Fivetran configurations for efficient data transfer.

**Resource Allocation:** Allocate appropriate compute resources in Databricks, adjusting as needed based on workload demands.

**Indexing and Caching:** Implement indexing strategies and caching mechanisms for faster data retrieval.

**Code Optimization:** Refactor SQL queries and Spark jobs to harness Databricks' distributed computing power, reducing execution time.

**Testing and Validation:** Perform extensive testing to validate data accuracy and performance, addressing bottlenecks identified during testing.

**Monitoring and Tuning:** Utilize Databricks' monitoring tools to track performance and apply necessary optimizations post-migration.

**Data Compression:** Employ data compression techniques to minimize storage costs and enhance I/O performance.

**Incremental Data Migration:** Consider incremental migration techniques to ensure data consistency and minimize downtime, using Delta Lake for efficient updates.

# AMAZON REDSHIFT PROFILER

The Amazon Redshift Profiler reads system views and catalog tables (e.g., PG\_TABLE\_DEF) and returns insights such as workload types, long-running ETL queries and user access patterns. The profiler classifies queries into T-shirt sizes for complexity, evaluates function calls for compatibility, and extracts other metadata information to aid with the data migration. This analysis provides guidance on identifying databases and pipelines that contribute to high costs and complexity. The results assist in workload prioritization and migration execution planning. Work with your Databricks representative to get access to the Profiler.

A screenshot of a computer screen

Description automatically generated

# Redshift vs. Databricks Comparison

|  |  |  |
| --- | --- | --- |
| OBJ ECTS/ WOR K LOAD | AMA ZO N R E D S H I F T | DATA B R I C KS |
| Compute | Amazon Redshift Clusters | Databricks Clusters optimized for workload types with a runtime:  All-purpose for interactive/developer use Jobs for scheduled pipelines  SQL warehouse for BI workloads and ad hoc SQL queries |
| Storage | Amazon Redshift Managed Storage and S3 | Cloud storage  (Amazon S3, Azure Blob Storage, Azure Data Lake Storage Gen2, Google Cloud Storage) |
| Tables | Amazon Redshift Tables | Delta Tables |
| Format | Amazon Redshift proprietary | Delta Lake (Parquet) |
| Interface | Amazon Redshift Query Editor v2.0  Redshift Data API | Databricks collaborative notebooks Databricks SQL workspace Databricks CLI |
| Database Objects | Tables, Temporary Tables, Views, Materialized Views, Stored Procedures, UDFs | Tables, Views, Temporary Views, Materialized Views, UDFs |
| Metadata Catalog | Built-in catalog, Glue Catalog | Unity Catalog |
| Data Sharing | Amazon Redshift Data Sharing AWS Data Exchange | Delta Sharing  Delta Sharing Marketplace |
| Data Ingestion | AWS Glue, Amazon EMR, custom pipelines using connectors, COPY (S3), AWS Kinesis, third- party tools | COPY INTO CONVERT TO DELTA  Auto Loader DataFrame Reader  Structured Streaming APIs for Kafka and Kinesis  Integrations via Partner Connect Add data UI |
| Data Types | [Data Types in Amazon Redshift](https://docs.aws.amazon.com/redshift/latest/dg/c_Supported_data_types.html) | [Data Types in Databricks](https://docs.databricks.com/sql/language-manual/sql-ref-datatypes.html) |
| Workload Management | Available through WLM | Intelligent Workload Management (IWM), cluster configuration (policies), cluster metrics |
| Access Control | IAM, Glue Catalog | IAM, role-based access, Unity Catalog |
| Sorting | Multipart Sort Keys | Z-Ordering |

# Conclusion

This migration plan provides a comprehensive and structured approach to transitioning from Redshift to Databricks. By leveraging automation, best practices, and thorough testing, the plan aims to achieve a seamless migration that maximizes performance and minimizes disruptions to business operations.