

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DE CALIDAD DE DATOS PARA UN DATA LAKE ALOJADO EN AWS”

BASTIÁN ANDRÉS GAETE AGUILERA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Mauricio Figueroa Colarte
Profesor Correferente: Nicolás Rojas Morales

Julio - 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Diseño e implementación de un módulo de calidad de datos para un data lake alojado en AWS

Nombre del candidato(a): Barbón Andrés Isidro Aguilera

Carrera / Grado: Ingeniería Civil Informática

Campus: Casa Central Departamento: Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Mauricio Ignez Castro, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente DEJO CONSTANCIA que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo NO contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo CONTIENE información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 29 de julio de 2026

Firma: 

Estudiante o Candidato(a):

Fecha: 29 de julio de 2026

Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

RESUMEN

Resumen—

En el ámbito del análisis de datos, ya sea para entrenar un modelo de Machine Learning, una Red Neuronal o para la Inteligencia de Negocios, la calidad de los datos es un factor crucial. Dentro de un data lake, esto no es tan simple de conseguir. En este trabajo se diseñó e implementó un módulo operativo simple y efectivo para monitorear la calidad de los datos, específicamente sus dimensiones de consistencia, unicidad y exactitud, enviando un reporte diario acerca del estado de los datos.

Para lograr esto se investigaron distintas metodologías, integrando lenguajes como SQL y Python para obtener el mejor rendimiento por el menor costo posible. El enfoque inicial se centró en bases de datos Oracle, comparando su información con la almacenada en un data lake en AWS. Esto permitió mejorar la consistencia de los datos consumidos por las distintas áreas de la organización respecto de las fuentes originales. Así, se buscó mejorar la reportería, ya sea de consumo interno o externo, y apoyar la toma de decisiones con información veraz y actualizada.

Palabras Clave— Calidad de Datos; Data lake; SQL; Python; AWS; Monitoreo de Datos; Automatización.

ABSTRACT

Abstract— In the field of data analysis, whether is for Machine Learning model training, a Neural Net, or Business Intelligence, a crucial factor is data quality. In a data lake, it is not an easy task. In this project, a simple and effective model was implemented in order to monitor data quality, specifically its dimensions in terms of consistency, uniqueness and accuracy by generating a daily report providing the current data status.

To achieve this, various methodologies were investigated, integrating languages such as SQL and Python to achieve the best performance at the lowest possible cost. The initial approach focused on Oracle databases, comparing their data with that stored in an AWS data lake. This improved the consistency of the data used by different areas of the organization

relative to the original sources. The goal was to improve reporting, whether for internal or external use, and to support decision-making with accurate and up-to-date information.

Keywords— Data Quality; Data lake; SQL; Python; AWS; Data Monitoring; Automation.

GLOSARIO

- **AWS:** Amazon Web Services.
- **BI:** Business Intelligence, Inteligencia de Negocio en español.
- **CDC:** Change Data Capture.
- **CMF:** Comisión para el Mercado Financiero.
- **DEV:** Se refiere a development, desarrollo en español, utilizado para separar distintos ambientes.
- **DPU:** Data Processing Unit.
- **ETL:** Extract, Transform, Load. Proceso que se encarga de extraer, transformar y cargar datos desde una fuente para disponibilizarla en un ambiente.
- **GCP:** Google Cloud Platform.
- **IA:** Inteligencia Artificial.
- **JDBC:** Java Database Connectivity.
- **OLAP:** On-Line Analytical Processing, Procesamiento Analítico en Línea en español.
- **PP:** Se refiere a pre-producción, utilizado para separar distintos ambientes.
- **PROD:** Se refiere a producción, utilizado para separar distintos ambientes.
- **PSH:** Python Shell.
- **Sernac:** Servicio Nacional del Consumidor.
- **SQL:** Structured Query Language.

ÍNDICE DE CONTENIDOS

RESUMEN	I
ABSTRACT	I
GLOSARIO	III
ÍNDICE DE FIGURAS	VII
ÍNDICE DE TABLAS	VII
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Contexto	3
1.1.1 Negocio	3
1.1.2 Productos	3
1.1.3 Tecnología	4
1.1.3.1 Cores Transaccionales	4
1.1.3.2 Cloud	5
1.2 Funcionamiento Actual	5
1.3 Problema a abordar	7
1.4 Organización del Equipo	8
1.5 Objetivos	9
1.5.1 Objetivo General	9
1.5.2 Objetivos Específicos	10
1.6 Alcance	10
CAPÍTULO 2: MARCO CONCEPTUAL	11
2.1 Estado de la práctica	11
2.1.1 Data lake	11
2.1.2 Calidad de datos: concepto y dimensiones	13
2.1.3 Frameworks de calidad de datos	14
2.2 Estado del arte	15
2.2.1 Deequ (AWS Glue Data Quality)	16
2.2.2 Great Expectations	16
2.2.3 AWS Glue DataBrew	17
2.2.4 Perfilamiento de datos (Data Profiling)	17
2.3 Metodología de la solución	18
2.3.1 Recolección de requerimientos	18
2.3.2 Elección de tecnologías	18
2.3.3 Implementación del módulo de calidad de datos	18
2.4 Metodología de la validación	19

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	21
3.1 Levantamiento de requerimientos	21
3.2 Selección de tecnologías	23
3.2.1 AWS Glue Python Shell	23
3.2.2 AWS DynamoDB	23
3.2.3 AWS Redshift	23
3.2.4 Great Expectations, Deequ / AWS Glue Data Quality, AWS Glue DataBrew	23
3.2.5 Amazon QuickSight y Microsoft Teams	24
3.3 Arquitectura del módulo	24
3.3.1 Consistencia	24
3.3.2 Unicidad	25
3.3.3 Exactitud	26
3.4 Flujo de validación	27
3.4.1 Consistencia	27
3.4.2 Unicidad	29
3.4.3 Exactitud	29
3.5 Implementación de métricas	30
3.5.1 Consistencia	30
3.5.2 Unicidad	30
3.5.3 Exactitud	31
3.6 Persistencia y reportería	31
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	33
4.1 Estrategia de validación	33
4.2 Validación de la consistencia	33
4.2.1 Preparación del escenario de prueba	33
4.2.2 Ejecución de las validaciones	33
4.2.3 Resultados obtenidos	34
4.2.4 Análisis de resultados	37
4.3 Validación de la unicidad	38
4.3.1 Preparación del escenario de prueba	38
4.3.2 Ejecución de las validaciones	38
4.3.3 Resultados obtenidos	39
4.3.4 Análisis de resultados	40
4.4 Validación de la exactitud	40
4.4.1 Preparación del escenario de prueba	40
4.4.2 Ejecución de las validaciones	41
4.4.3 Resultados obtenidos	41
4.4.4 Análisis de resultados	41
4.5 Evaluación global de la solución	41
4.5.1 Cumplimiento de requerimientos	42
4.5.2 Limitaciones identificadas	42

CAPÍTULO 5: CONCLUSIONES	44
5.1 Conclusiones por capítulo	44
5.1.1 Definición del problema	44
5.1.2 Marco conceptual	44
5.1.3 Propuesta de solución	45
5.1.4 Validación de la solución	45
5.2 Conclusiones generales	45
5.3 Cumplimiento de objetivos	46
5.3.1 Consistencia	46
5.3.2 Unicidad	48
5.3.3 Exactitud	49
5.4 Trabajo futuro	49
REFERENCIAS BIBLIOGRÁFICAS	50

ÍNDICE DE FIGURAS

1	Arquitectura General SDLF	7
2	Organigrama Gerencia Data Management	9
3	Tabla comparativa entre Data warehouse, Data lake y Data lakehouse	12
4	Ciclo TDQM	14
5	Arquitectura módulo consistencia	25
6	Arquitectura módulo unicidad	26
7	Arquitectura módulo exactitud	27
8	Ejecuciones correctas módulo consistencia	34
9	Ejecuciones en timeout módulo consistencia	34
10	Validaciones CyberDay Octubre 2025	35
11	Reporte automático de consistencia del 5 de abril de 2026	36
12	Reporte automático de consistencia del 9 de abril de 2026	37
13	Ejecuciones en timeout módulo unicidad	38
14	Ejecuciones correctas módulo unicidad	39
15	Reporte automático de unicidad del 6 de junio de 2026	39
16	Columna con valores imposibles preparada	40
17	Mensaje resumen enviado a Microsoft Teams tras detección de discrepancias	41
18	Resultados consistencia en septiembre de 2025	47
19	Resultados consistencia en ciertos meses de 2025	47
20	Resultados unicidad en ciertos días de 2026	48
21	Cantidad de tablas evaluadas en unicidad en ciertos días de 2026	48

ÍNDICE DE TABLAS

1	Matriz de Trazabilidad de Requerimientos y Objetivos del Proyecto	22
---	---	----

INTRODUCCIÓN

Las grandes empresas manejan una cantidad de datos inconmensurable, y muchas veces se encuentran repartidos entre distintas fuentes. Esto tiene ventajas y desventajas. Por el lado de la arquitectura, permite una mayor independencia y tolerancia a fallos, ya que cada servicio o área específica puede contar con su propio servidor o base de datos. Sin embargo, también genera una dificultad al momento de realizar reportes desde el área de inteligencia de negocios u otras áreas similares. Debido a esto, se tienen que abrir conexiones a distintas bases de datos que pueden tener versiones diferentes o incluso ser de otros motores SQL (SQL Server / Oracle / MySQL), lo que puede ocasionar que una consulta no sirva para dos servidores distintos por las sutiles diferencias entre los dialectos SQL utilizados.

Debido a lo mencionado en el párrafo anterior, la Gerencia Data Analytics de Banco Ripley propuso la idea de construir un data lake en AWS. Para ello, se utilizaron distintas tecnologías, como AWS Glue y S3, con el fin de disponibilizar la información a los usuarios mediante un clúster de AWS Redshift. Esta iniciativa reemplazó la herramienta SAS, que ya estaba obsoleta.

En los data lake (o data warehouse o data lakehouse, similares pero no iguales), son varias las fuentes de datos que convergen a un mismo lugar, con el fin de que la consulta de esta información se realice de forma rápida y eficiente.

Cuando una empresa decide implementar un data lake, suele ser para que un equipo de analistas de datos trabaje con ellos con el objetivo de generar insights o métricas que apoyen en la toma de decisiones de la empresa. Esto es sumamente valioso ya que puede dirigir el rumbo de la empresa, desde qué tipo de clientes captar, cuáles mantener, y cómo hacerlo. Es por esto que la toma de decisiones requiere de datos que no tengan errores, puesto que una mala calidad en los datos puede afectar el resultado de las métricas analizadas [Saber y Fariha, 2024; Rahm y Do, 2000], o causar que los analistas tarden más tiempo en conseguir esas métricas, al tener que limpiarla de manera manual.

Aquí se empieza a considerar la necesidad de un módulo que verifique y valide la calidad de los datos. Por consiguiente, esto requiere que sea algo constante, pues cada día, y a veces cada hora, está llegando información nueva que es procesada y almacenada en el data lake, por lo que datos que se encontraban en perfectas condiciones, de un momento a otro, puede pasar a contener errores.

Estos errores, principalmente de consistencia, unicidad y exactitud, tendrán que ser detectados comparando la información disponible en el data lake con la información presente en las fuentes de origen, o directamente en el mismo data lake de ser posible.

Los temas mencionados previamente serán detallados más adelante de la siguiente manera:

- **Capítulo 1:** Definición del problema. Se explicará el contexto del problema, la solu-

ción actual que se encuentra implementada y quiénes son los usuarios involucrados. Además se define el objetivo general y los objetivos específicos.

- **Capítulo 2:** Marco Conceptual. Se detallarán conceptos técnicos y se revisa el estado del arte acerca de soluciones implementadas en otras compañías.
- **Capítulo 3:** Propuesta de Solución. Se diseñará y desarrollará la solución propuesta para cumplir con los objetivos definidos.
- **Capítulo 4:** Validación de la Solución. Se validará la solución implementada mediante métricas que permitan confirmar el cumplimiento de los objetivos definidos.
- **Capítulo 5:** Conclusiones. Se detallarán las conclusiones obtenidas a partir de la solución implementada junto a recomendaciones para trabajos futuros.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

En el capítulo 1 se contextualiza al lector respecto a las características que rodean el trabajo. Esto para entender mejor el origen del problema, lo que se busca solucionar y los objetivos a cumplir.

1.1. Contexto

Se hará una ligera explicación para contextualizar, a nivel empresarial y de tecnologías utilizadas, al lector, para que tenga un entendimiento básico que facilite la comprensión de lo que se abordará más adelante.

1.1.1. Negocio

Ripley Corp es la empresa matriz de las empresas Ripley, que se centran en 3 sectores. El retail, con Ripley, representada por las más de 70 tiendas por departamento presentes en Chile y Perú. El negocio bancario, con Banco Ripley, con más de 70 sucursales en Chile y Perú, incluyendo también el rubro de seguros con SegurosRipley. Finalmente, el sector inmobiliario, teniendo la propiedad de distintos centros comerciales tanto en Chile como en Perú.

1.1.2. Productos

Debido a que el presente trabajo está enfocado en la parte de Banco, y en menor medida, en Seguros, se omitirá la información de Ripley Retail.

Dentro de los productos ofrecidos por Banco Ripley, se encuentran:

- Tarjetas Bancarias
 - Tarjeta de Crédito
 - Tarjeta de Débito
- Cuentas Bancarias
 - Cuenta Corriente
 - Cuenta Vista (No se pueden abrir nuevas)

- Créditos / Financiamiento
 - Crédito de Consumo
 - Avance
 - Súper Avance
- Inversiones
 - Depósitos a plazo

Por su lado, los productos ofrecidos por Seguros Ripley, filial de Banco Ripley, corresponden a los siguientes:

- Seguros de Auto
- Seguros de Vida
- Seguros de Salud
- Seguros de Mascotas
- Seguros de Viaje

1.1.3. Tecnología

Dentro de la sección de tecnologías, se detallan 2 aristas, los cores operacionales del negocio, y lo implementando en la nube, ya sea por el área analítica o artefactos productivos que van al público.

1.1.3.1. Cores Transaccionales

Banco Ripley cuenta con numerosas bases de datos, cada una con objetivos distintos. El core transaccional principal es R-SAT, basado en SAT, pero modificado, obteniendo así R-SAT (Ripley-SAT). Las bases de datos core de la operación son:

- **REPORITF:** Core transaccional con información de tarjetas y transacciones.
- **FISA:** Core que almacena datos de productos del banco.
- **FIDENS:** Core que almacena toda la información relacionada con seguros.
- **REPITFH:** Versión histórica de REPORITF, almacena más información que no es consultada tan frecuentemente.

- **CAMREPORT:** Core que almacena información de campañas.

Además de estas bases de datos core, hay muchos servidores departamentales on premise que están en proceso de migración a Cloud y/o al data lake.

1.1.3.2. Cloud

Banco Ripley cuenta con recursos en distintos proveedores Cloud. Los principales son AWS utilizado principalmente por el área analítica; Azure, destinado a la gestión de correos, grupos, cuentas organizacionales y algunos servidores departamentales; Oracle Cloud, utilizado en algunas bases de datos; y, finalmente, GCP, empleado para bases de datos, migración de on premise a Cloud y aplicaciones productivas tales como la aplicación del Banco.

La organización en las distintas plataformas Cloud es similar. En AWS, existe una cuenta administradora, desde la cual se genera múltiples cuentas con ambientes como DEV, PP y PROD, donde existe una cuenta de cada ambiente para cada proyecto grande. En el caso de GCP es similar, con una cuenta administradora, y muchos proyectos de Google para manejar las distintas iniciativas, nuevamente con ambientes DEV, PP y PROD.

Para el Proyecto Data Management, encargado del data lake, se utilizan 3 cuentas de AWS, DEV, PP y PROD; de estas, DEV y PP muy poco utilizadas, salvo en ocasiones excepcionales donde los cambios que se requieren ejecutar en producción pueden tener un impacto catastrófico en cómo se procesa la data. Para el resto de los desarrollos menores, se realiza todo en producción.

1.2. Funcionamiento Actual

Entre los procesos requeridos para hacer funcionar y mantener un data lake, se encuentran los llamados procesos ETL. Los procesos ETL son los que se encargan de extraer, transformar y cargar los datos [Mohammed Muddasir y Raghuv eer, 2017]. La extracción se encarga de hacer la consulta en la fuente de datos original para llevarla al data lake. La transformación verifica y homologa los esquemas de las tablas resultantes, y si se requiere, suma agregaciones en forma de columnas a las tablas. Por último la carga, que es donde los datos son finalmente guardados en el sistema de almacenamiento definido en la arquitectura del sistema.

Los procesos ETL normalmente se encuentran automatizados, lo cual requiere una mínima, o nula participación de un humano para verificar que todo se ejecute como tiene que ser. Esto implica que una tabla podría estar cargándose erróneamente, y no siempre alguien se dará cuenta, sobre todo si se considera que los data lake sirven para manejar volúmenes enormes de datos, y las personas no pueden estar pendientes de cada uno de estos.

En el caso de estudio, el proceso de extracción se realiza de tres maneras distintas según la criticidad de la tabla. Para las tablas más críticas, la información llega en archivos parquet directo a S3, a través de Golden Gate, un software instalado en una base de datos Oracle que lee los logs de transacciones de la base de datos [Oracle, sf] para enviar los cambios (inserts, updates y deletes) a S3, el servicio de almacenamiento de objetos de AWS [AWS, sfd]. Las siguientes tablas críticas que no cuentan con este sistema, se extraen a través de AWS Glue, el servicio serverless de integración de datos de AWS [AWS, sfd], con una conexión directa con la base de datos a través de PySpark con conectores JDBC, con la que se leen los datos y se escriben en S3 como parquets. La última opción, y la más utilizada, es a través de Apache NiFi, una herramienta de automatización de flujos de datos entre sistemas [Apache Software Foundation, sf], donde una máquina virtual de EC2 tiene el software configurado con la orquestación para extraer cientos de tablas de múltiples de bases de datos distintas cada día, nuevamente, mediante una conexión a las bases de datos a través de conectores JDBC, con la que se extrae la información y se carga en S3 como archivos parquet.

La parte de Transformación y Carga de los datos al data lake se realiza a través de AWS Glue con eventos configurados para que se ejecuten cuando lleguen archivos a S3. A grandes rasgos, se leen los archivos con PySpark, se realiza la transformación con base en la configuración previa de la tabla, y luego se escriben en rutas específicas de S3, con lo que quedan disponibles en el Glue Data Catalog para ser consultadas a través de Redshift [AWS, sfd] o Athena [AWS, sfa]. Todo esto se realiza con una versión modificada del Serverless Data Lake Framework de AWS, una colección de artefactos reutilizables para apoyar en la implementación de data lakes de las empresas [Fait et al., 2024].

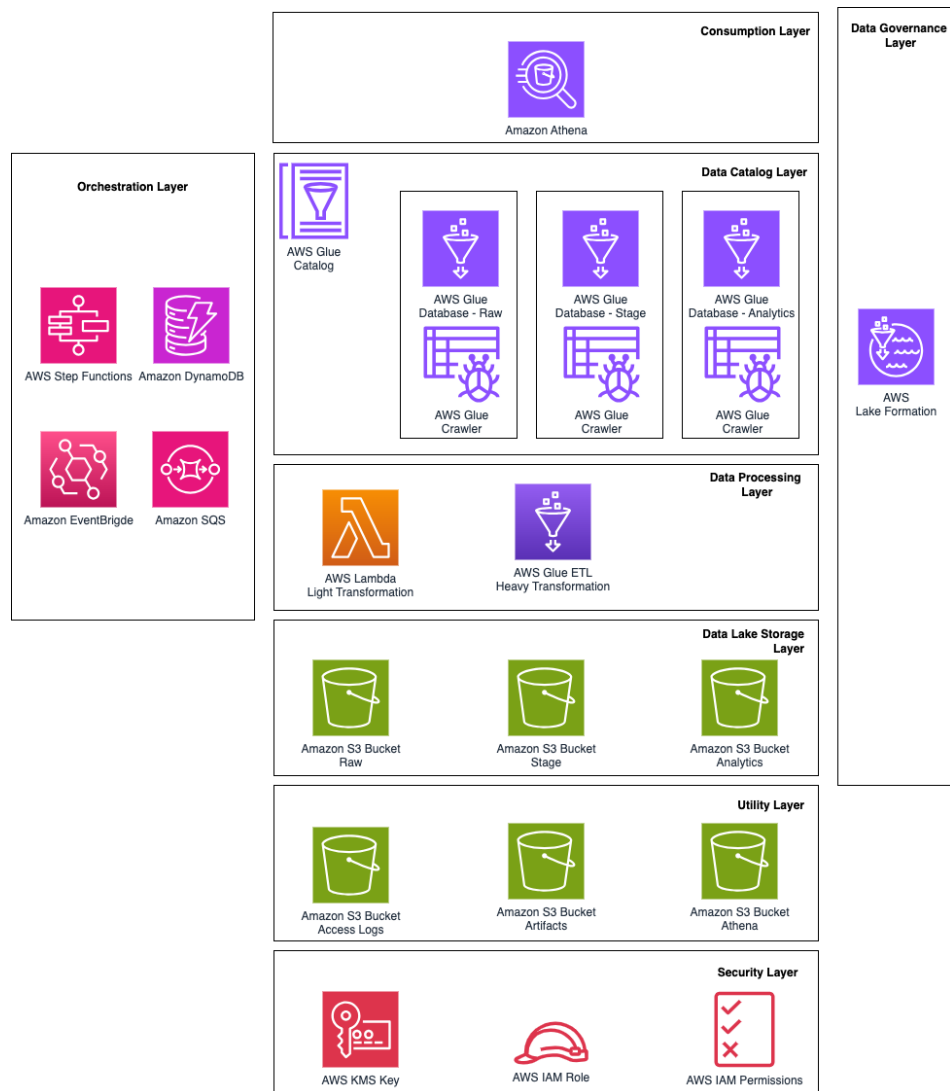


Figura 1: Arquitectura General SDLF.
Fuente: GitHub SDLF.

1.3. Problema a abordar

¿Qué es un problema en la calidad de los datos? ¿Qué tipo de errores hay? Un problema en la calidad de los datos puede estar presente en varias dimensiones, entre las que se encuentran consistencia, duplicación, completitud y exactitud [Yeh y Puri, 2010]. Es decir, si se tiene un registro duplicado en una tabla, se tiene un problema de duplicidad en la tabla. Si existe una columna con un valor X, que en el origen tiene un valor Y, existe un problema de exactitud. Si hay una columna vacía, que en el origen tiene valor Z, hay un problema de completitud. Algunos de estos errores son más difíciles de detectar que otros, lo que los vuelve complejo de corregir.

La toma de decisiones a nivel empresa, en especial en el rubro económico, necesita de datos lo más fieles posible a su origen (dígase origen la base de datos réplica de la productiva). Esto se vuelve aún más importante cuando desde el data lake se empiezan a extraer reportes normativos o a responder oficios del Sernac o de la CMF. Errores presentes en los datos pueden derivar en cuantiosas multas y en pérdida de confianza tanto para la empresa, por parte de los clientes, como al data lake y el Proyecto Data Management, por parte de equipos internos del banco.

Debido a esto, se propone el diseño y la implementación de un módulo de calidad de los datos que verifique las dimensiones de consistencia, unicidad y exactitud, priorizando consistencia y unicidad como controles operativos diarios, para detectar algún posible error en los datos que llegan día a día. Realizar esto necesitará de conocimientos acerca de la nube, al estar el data lake en los servicios de AWS; entender cómo funciona el framework que está detrás del data lake y cómo está armada su arquitectura. Además, se debe saber SQL, al estar trabajando con fuentes SQL en su totalidad, se presenta también un desafío por la capacidad computacional de las bases de datos fuentes, ya que se tiene que trabajar con máquinas antiguas, lo que implica que las consultas SQL deben estar lo más optimizadas posibles para además abaratar costos en la ejecución del script en AWS (al contar con un cobro por hora).

1.4. Organización del Equipo

El equipo que da soporte y mantiene el data lake en el banco corresponde a la Gerencia de Data Management. La gerencia está liderada por el Chief Data Officer, quien a su cargo tiene al gerente de Arquitectura de Datos, al subgerente de Gobierno de Datos y al subgerente de Productos de Datos. En todos los casos, los cargos corresponden al área corporativa de Ripley, teniendo que liderar Banco Ripley y Ripley Retail a la vez. En el caso del gerente de Arquitectura de Datos, aparte de administrar Banco Ripley y Ripley Retail Chile, tiene que gestionar también ambas áreas de Perú.

Todos los equipos tienen un grado de participación en el data lake, y son los de Arquitectura de Datos quienes lo mantienen a nivel técnico. El equipo de Gobierno de Datos se encarga de dar la autorización a los usuarios a que puedan revisar la información, para tener un control correcto de los accesos. En el caso de Productos de Datos, permite tener lineamientos oficiales corporativos respecto a cómo se utilizan los datos, definiendo productos basados en ellos con su respectivo objetivo, ciclo de vida, entre otros. Esta última área todavía se está armando, así que no se detallan más funciones.

Como los que más participan son los de Arquitectura de Datos, se ahondará más en ellos. La gerencia tiene una subgerencia de Arquitectura de Datos, que se divide en dos jefaturas. Por un lado, se cuenta con una dedicada a la operación diaria, encargada de ingestas de tablas, calidad de los datos, campañas, reportería oficial, soporte a BI y apoyos varios. Por otro lado, la otra jefatura se dedica a proyectos internos, agentes con inteligencia artificial, automatizaciones, y todo lo que esté relacionado a proyectos de datos o nuevas iniciativas.

Gerencia Data Management

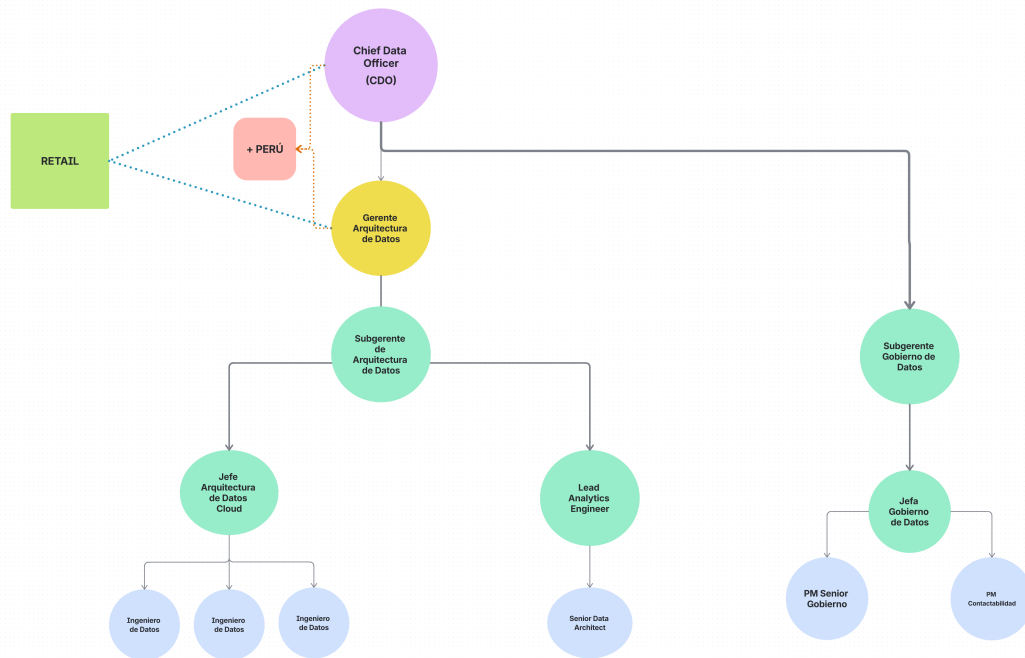


Figura 2: Organigrama Gerencia Data Management.
Fuente: Elaboración Propia.

1.5. Objetivos

Los objetivos de esta memoria son los siguientes:

1.5.1. Objetivo General

Diseñar e implementar un módulo de control de calidad de datos para un data lake, con el fin de mejorar la consistencia y confiabilidad de los datos almacenados.

Donde la confiabilidad se evaluará conjuntamente a través de las tres dimensiones a implementar en el módulo, siendo una mejora en la confiabilidad equivalente a una reducción de errores detectados y en la estabilidad de los resultados históricos almacenados por el módulo.

1.5.2. Objetivos Específicos

- Definir e implementar reglas y métricas de calidad de datos orientadas a detectar, registrar y reportar errores comunes.
- Identificar las diferencias entre los datos almacenados en el data lake con sus fuentes de origen.
- Desarrollar un dashboard que permita visualizar de manera rápida el estado de calidad de las tablas monitoreadas por el módulo.
- Establecer un proceso de mejora continua para mantener y elevar los niveles de calidad de los datos en el data lake.

1.6. Alcance

En un principio la idea era implementar data quality en todo el data lake, pero dados los costos de implementación, el tiempo disponible y la complejidad de establecer la conexión con algunas bases de datos, el proyecto se centrará en monitorear y evaluar que los datos del core **REPORTIF** que llegan a través de Golden Gate estén con un nivel de calidad monitoreado y controlado.

CAPÍTULO 2

MARCO CONCEPTUAL

En este capítulo se definirán los conceptos fundamentales y metodologías que guiarán el diseño e implementación del módulo de calidad de datos. Se comenzará describiendo el estado de la práctica en relación con la gestión de calidad de datos en entornos corporativos, donde se revisará cómo las organizaciones abordan este problema hoy en día. Luego, se presentará el estado del arte, donde se detallarán las herramientas y enfoques más modernos disponibles para implementar controles de calidad sobre un data lake. A continuación, se describirá la metodología seleccionada para desarrollar la solución, donde se justificará su elección por sobre otras alternativas. Finalmente, se presentará la metodología que permitirá validar que la solución implementada cumple con los objetivos planteados.

2.1. Estado de la práctica

El estado de la práctica comprende el conjunto de métodos, procesos y técnicas que son ampliamente utilizados y aceptados por los profesionales de una industria en un momento determinado. Refleja el conocimiento colectivo aplicado en contextos reales, es decir, lo que las organizaciones efectivamente hacen para resolver un problema, más allá de lo que la teoría propone como ideal. En esta oportunidad se analizará el estado de la práctica de la gestión de calidad de datos en organizaciones que operan con data lakes. Se revisará cómo se define la calidad de datos, qué dimensiones se consideran relevantes, y cuáles son los estándares y marcos de referencia más utilizados en la industria para medirla y gestionarla.

2.1.1. Data lake

Un data lake es una solución para la gestión de datos, centrada en almacenar datos a bajo costo. A diferencia de un data warehouse, que está centrado en data estructurada, los data lakes no tienen que estar anclados a datos de este tipo, sino que también puede contener otro tipo de archivos, como imágenes, videos, audios, entre otros [Prasad y Sayegh, 2026].

	Data warehouses	Data lakes	Data lakehouses
Types of data	Mostly structured data	Raw structured, unstructured and semistructured data	Raw or processed structured, unstructured and semistructured data
Storage type	Relational databases	Cloud object storage	Cloud object storage
Storage cost	High	Low	Low
Native query performance	High	Low	High
Schema approach	Schema-on-write	Schema-on-read	Supports both schema-on-write and schema-on-read
Defining feature	Built-in analytics engine supports high performance	Low-cost, scalable storage for all data types	Warehouse performance and lake storage in one solution
Common use cases	Business intelligence and data analytics	Artificial intelligence, machine learning and data science	Business intelligence, data analytics, artificial intelligence, machine learning and data science

Figura 3: Tabla comparativa entre Data warehouse, Data lake y Data lakehouse.

Fuente: [Kosinski, sf].

Una ventaja que tienen los data lakes es la posibilidad de recibir muchos datos de distintas fuentes de manera simultánea. Esto es logrado mediante el almacenamiento escalable que ofrecen. Dicho almacenamiento suele ser en un ambiente cloud, como Amazon S3 o Google Cloud Storage, por lo que el almacenamiento puede escalar sin las restricciones típicas de un servidor on premise.

La organización de los datos que ingresan al data lake depende de la arquitectura implementada. Una de las más comunes es la arquitectura de tipo medallón, donde los datos son separados en tres capas distintas, bronce, plata y oro [Databricks, sf]. La capa de bronce representa los datos en su manera cruda, tal como llegan. En la capa de plata se encuentran los datos ya limpiados, ligeramente procesados. Finalmente, en la capa de oro, se encuentran los datos con agregaciones. La idea es que los datos vayan progresando a través de las capas hasta llegar a la de oro.

Esta ventaja los vuelve muy atractivos para ser utilizados como centros de análisis de datos. Generalmente se utilizan sistemas OLAP o servicios especializados en Machine Learning o IA, que se conectan a los datos almacenados en los data lake de manera centralizada. Algunos ejemplos de estos servicios serían Redshift, Amazon SageMaker Studio o similares.

La falta de gobernabilidad de los datos presentes en un data lake puede terminar en lo que es conocido como un data swamp. Como comenta [de León, 2025], “esta falta de control puede derivar en lo que se conoce como un data swamp, donde la utilidad de los datos almacenados en el data lake se ve comprometida, dificultando el análisis y la obtención de datos confiables”. Al ingresar tantos datos al data lake, es fácil perder la trazabilidad de lo que hay y para qué sirven, o incluso por qué están ahí. Para evitar esto, es fundamental contar con procesos que validen que los datos ingresados sean consistentes con su fuente de origen.

2.1.2. Calidad de datos: concepto y dimensiones

Algunos autores definen la calidad de datos como el estado o los atributos de los datos que define la habilidad para cumplir su función específica [Vágner, 2026]. La norma ISO 8000-2 define como calidad el grado en el cual un conjunto de características intrínsecas de un objeto cumplen ciertos requerimientos [ISO, 2020]. En ese sentido, se podría decir que la calidad de datos no es algo fijo, sino que es algo que cada persona o equipo de desarrolladores debe definir a su propia manera, con base en los distintos factores que les afectan, y con ello entender qué les es aceptable en su entorno.

Una de las maneras de medir la calidad de datos, es a través de algunas de sus dimensiones. Hay muchas dimensiones posibles, y por eso uno debe elegir correctamente las que utilizará para su caso de uso. La gran mayoría de las dimensiones son medibles cuantitativamente, lo que permite obtener valores estructurados con los que evaluar qué tan buena es la calidad de los datos.

Algunas de las principales dimensiones, o de las más comunes, son [Aquino y Jonker, 2026]:

- *Accuracy (Exactitud)*: Mide qué tan exactos son los datos, ya sea con eventos de la vida real, o con una fuente de confianza.
- *Completeness (Compleitud)*: Mide si los datos necesarios están presentes, dígame las columnas de monto en una tabla de saldos, por ejemplo.
- *Consistency (Consistencia)*: Mide si los datos son consistentes entre distintas fuentes.
- *Timeliness (Temporalidad)*: Mide si los datos están disponibles cuando son necesarios, o básicamente, qué tan frecuente es el refresco de los datos.
- *Validity (Validez)*: Mide si los datos son válidos conforme a reglas predispuestas, ejemplo, una fecha de nacimiento no puede ser mayor a la fecha actual.
- *Uniqueness (Unicidad)*: Mide si cada registro es único y no se encuentra duplicado.

Como se mencionó previamente, los datos y las métricas a utilizar depende del uso que se le dé y del área a la que pertenezcan los datos. Esto hace que los datos, y su respectiva calidad, caigan en lo que es conocido como el **fit for purpose** o **adecuada para su propósito**. Un ejemplo claro es que si los datos son solo para uso analítico, se podrían aceptar valores más relajados en la exactitud o consistencia de los datos. Sin embargo, para un uso normativo, o legal, los datos tienen valores mucho más estrictos por cumplir, pues el propósito de los datos así lo requiere. Esto invita a los consumidores de los datos a definir los valores mínimos que deben cumplir los datos para ser considerados como correctos, para así todos aprovechar al máximo la información disponible y poder tomar decisiones mejores.

A veces se discute si es realmente necesario el uso de métricas objetivas calculadas de manera automática, o si basta con una evaluación subjetiva del usuario final de los datos. A

medida que la automatización de los procesos se va haciendo cada vez más masiva, es necesario contar con valores objetivos que no se vean sesgados por los usuarios finales, pues muchas veces ese sesgo, intencional o no, puede ocasionar valores fuera de lo común y con ello, decisiones incorrectas para los objetivos finales del usuario/empresa. El uso de valores objetivos, y la independencia de un consumidor de los datos para evaluar, permite la detección automática de errores en los datos y en algunos casos, incluso la corrección automática de estos errores, lo que da mucho más valor a los insights generados con los datos.

2.1.3. Frameworks de calidad de datos

La medición de la calidad de los datos no es algo nuevo. No obstante, se ha visto intensificada por la expansión de la inteligencia artificial en la cotidianidad de la vida. Esto debido a que, para entrenar los modelos de inteligencia artificial, se necesitan datos de buena calidad para evitar agujeros en la información y directamente errores en las respuestas.

Son varios los frameworks que existen con respecto a la calidad de datos y a su manejo, algunos son de uso general, mientras que otros son de un uso específico para un área, como finanzas o salud.

Según [Miller et al., 2025], algunos de estos frameworks son:

- *Total Data Quality Management (TDQM)*: Estrategia holística para asegurar y monitorear la calidad de los datos en organizaciones, creada en los años 80 por el MIT. Se centra en distintas dimensiones de calidad de datos, como credibilidad, reputación, acceso, seguridad, entre otras. Se basa en 4 etapas que forman un ciclo, definición, medición, análisis y mejora.

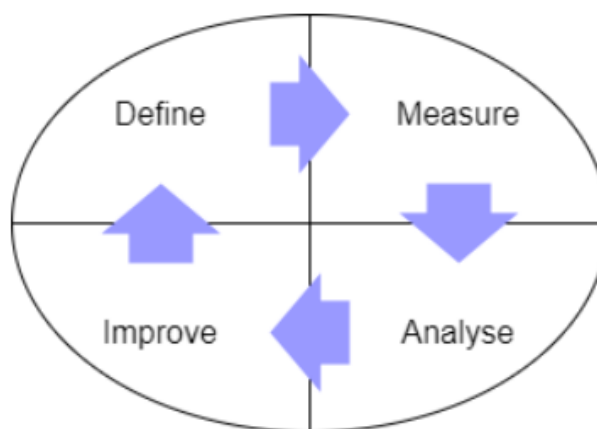


Figura 4: Ciclo TDQM.
Fuente: [Miller et al., 2025].

- *Quality Assurance Framework of the European Statistical System (ESS QAF)*: Sigue los mismos principios que el Total Quality Management Framework, pero más centrado en la calidad de los datos generados en procesos estadísticos. Sigue 5 dimensiones fundamentales, como relevancia, accesibilidad, claridad, entre otros. Su aplicabilidad es menor al del TDQM, al estar centrado en datos estadísticos, a diferencia del TDQM que es de uso general.
- *Data Management Body of Knowledge (DAMA DMBok)*: Publicado por la organización DAMA International, tiene una serie de guías detalladas para manejar los desafíos de la administración de datos. Algunos extractos del framework son utilizados por otros frameworks, como el *The UK Government Data Quality Framework*, con ciertos aspectos de este último estando basados en el de DAMA. Algunas áreas del conocimiento abarcadas por el DMBok son gobernanza de datos, arquitectura de datos, calidad de datos, entre otros, lo que lo hace muy adaptable a las necesidades específicas de cada uno.
- *IMF Data Quality Assessment Framework (DQAF)*: Publicado por el Fondo Monetario Internacional, está focalizado al sector de finanzas y estadísticas. Busca también que los distintos países que le reportan al FMI tengan métodos para asegurar la calidad de los datos estandarizados, promoviendo la transparencia financiera de las instituciones.
- *ALCOA+ Principles*: Son principios centrados en el mundo de la medicina. Permite asegurar la integridad de los datos con énfasis particular en la manufactura de medicinas. Además de esto, permite identificar falsificación de datos médicos, algo crítico en la salud de los pacientes y en los resultados de estudios.

Como se revisó, los frameworks existentes abordan la calidad de datos desde una perspectiva organizacional amplia, que definen procesos, roles y políticas para gestionar la calidad a nivel institucional. Sin embargo, ninguno de ellos ofrece una guía técnica específica para validar la integridad de los datos entre un sistema origen y un repositorio analítico como un data lake. Por esta razón, si bien estos frameworks sirven como referencia conceptual para entender qué dimensiones de calidad son relevantes, no se adoptará ninguno de forma directa en la implementación de la solución.

2.2. Estado del arte

En esta sección se describe el estado del arte de las distintas herramientas y técnicas que existen para medir la calidad de datos de conjuntos de datos en distintos ámbitos.

2.2.1. Deequ (AWS Glue Data Quality)

Deequ es una herramienta de código abierto desarrollada y utilizada en Amazon, que permite calcular de manera automática métricas de calidad de datos en un conjunto de datos, definir y validar restricciones en los datos, y notificar acerca de cambios en la distribución de los datos [Schmidt et al., 2019]. Deequ funciona a través del mismo motor de AWS Glue, que utiliza PySpark para los análisis y con costos asociados a DPU por hora. En el fondo, son un AWS Glue Job con un código definido con parámetros que se pueden personalizar para validar las métricas que al usuario le interese para el contexto donde lo está aplicando. También es posible utilizarlo desde AWS EMR para aprovechar el uso de contenedores aprovisionados para optimizar costos.

Es una buena alternativa para analizar conjuntos de datos grandes. Al utilizar PySpark tiene la libertad de dividir internamente las lecturas de información para evitar saturaciones o falta de memoria. Sin embargo, es una opción costosa de implementar a un nivel masivo, debido a cómo es cobrado el servicio de AWS Glue, además de la manualidad que tiene ingresar las restricciones requeridas, y el conocimiento necesario para poder definir esas mismas restricciones para que estén acorde al contexto de los datos.

2.2.2. Great Expectations

Great Expectations, generalmente abreviado como GX, es una librería de código abierto de Python para obtener métricas de calidad de datos o realizar validaciones. Utiliza un lenguaje entendible para humanos para llamar las validaciones, generando reportes HTML de manera automática al terminar sus pruebas [Great Expectations, sf]. Permite conectarse a distintas fuentes de datos, como BigQuery, PostgreSQL, Snowflake, Redshift, entre otras. Además, esta cuenta con dos versiones, GX Cloud, manejada por ellos con sus respectivos costos, y GX Core, que es la librería en sí que es de uso libre por quien la quiera utilizar.

Por una parte, GX Cloud está pensado para integraciones más simples, donde seleccionas la fuente de datos, se ingresan las credenciales de acceso y se seleccionan qué tablas se quieren validar y cómo validarlas, donde se muestran en la misma interfaz web los resultados de una manera simple. Los costos empiezan con un tier gratuito, luego pasa a uno por equipo para terminar en uno enterprise para organizaciones grandes, sin especificar en ningún caso los costos que tiene de manera explícita. GX Core, por otro lado, es de libre uso y sin costo asociado (a menos que se lance desde servicios con costos asociados), lo que lo vuelve sencillo de implementar, y también conserva la posibilidad de conectarse a distintas fuentes de datos de manera programática.

2.2.3. AWS Glue DataBrew

DataBrew es una herramienta de preparación de datos vía interfaz web, que permite definir reglas que los conjuntos de datos deben cumplir, como porcentaje de nulos, valores fuera de rango, entre otros; así como también la transformación de datos y la normalización de columnas [AWS, sf]. Está integrado con el ecosistema de AWS, lo que permite analizar datos en data lakes, Redshift y más. Su sistema de pricing es similar al de AWS Glue y al de AWS Glue Data Quality, sumando un costo fijo por cada sesión en base a su duración. Es similar a GX Cloud, en el sentido de que provee una interfaz gráfica que permite seleccionar lo que se quiere validar, que lo vuelve amigable no solo para los ingenieros de datos que administran los datos, sino que también para analistas con poco o nulo dominio de Python.

Comparación y selección

Las tres herramientas revisadas comparten una característica en común: están diseñadas para evaluar la calidad de datos dentro de un mismo ecosistema, ya sea sobre tablas ya almacenadas en un data lake o sobre conjuntos de datos accesibles desde una única fuente. Ninguna ofrece soporte nativo para comparar datos entre una base de datos transaccional Oracle on premise y un data lake en AWS, que es el requerimiento central de este trabajo. Esta limitación, junto con las restricciones de conectividad propias del entorno corporativo donde se implementa la solución, justifica el desarrollo de un módulo propio, el cual será detallado en el capítulo de propuesta de solución.

2.2.4. Perfilamiento de datos (Data Profiling)

El perfilamiento de datos se refiere al proceso de revisar y analizar datos para entender de mejor manera cómo están estructurados los datos de un conjunto de datos [IBM, sf]. Es una etapa crucial para definir métricas de calidad de datos, pues el análisis previo de los datos permite entender mejor qué información contiene la tabla, identificar patrones en algunas de las columnas y con ambos detalles. También se pueden definir reglas y métricas de calidad de datos para medir, que los conjuntos de datos deben cumplir para que se consideren aptos para el uso que se les tiene planificado.

El perfilamiento de los datos está intrínsecamente relacionado con la definición de las métricas de calidad de datos en base al uso que se le da o dará a los datos, siempre dependiendo del contexto en el que se utilizan. Estos análisis se pueden realizar de manera automática, con scripts y procesos que buscan patrones o evalúan la integridad de los datos. También es posible hacerlo de manera manual, caso a caso, o basarse en documentación existente de los conjuntos de datos que se quieren evaluar.

Algunas de las métricas más comunes a la hora de hacer un perfilamiento de datos, son la

completitud, la unicidad, la cantidad de nulos, valores máximos y mínimos, distribuciones, entre otras, que permiten comprender de mejor manera qué significan los valores que se están viendo. Este proceso se puede realizar antes de la ingesta de los datos, para que la configuración sea la ideal a la hora de procesar la información. Para esto es necesario contar con acceso a los datos de antemano. A pesar de esto, también es posible realizar el perfilamiento después de ingestar los datos, con una carga completa y sin hacer ninguna modificación en las filas y columnas, y, si se detectan patrones o errores, se pueden corregir para que los datos nuevos que lleguen cumplan con los mismos patrones (o reprocesar completamente los datos para que todo cumpla con lo definido).

2.3. Metodología de la solución

A continuación se describen las partes de la metodología para diseñar e implementar el módulo de calidad de datos para el data lake.

2.3.1. Recolección de requerimientos

El primer paso es definir con los stakeholders qué es lo que se requiere realizar, cómo se necesita realizar, y cuál es el presupuesto que se tiene considerado. Qué tablas considerar, qué bases de datos utilizar y qué métricas obtener para determinar el estado de una tabla para en base a esto, buscar la mejor manera de llevarlo a cabo.

2.3.2. Elección de tecnologías

El segundo paso es entender cuál de las distintas opciones de tecnología es mejor para el caso presentado, ya que si bien varias realizan las mismas cosas, o similares, pueden existir detalles que hagan que una sea más útil o directa de utilizar que otras, o que incluso haya que desarrollar una propia que cumpla con lo solicitado.

2.3.3. Implementación del módulo de calidad de datos

El último paso es la implementación del módulo de calidad de datos, cómo es la arquitectura y cómo se adapta a los artefactos que ya existen en el ambiente del data lake.

2.4. Metodología de la validación

Con la implementación de la medición de calidad de datos en una de las bases de datos, se evaluará si la solución cumple o no con los requerimientos necesarios. Las métricas de calidad a evaluar serán las siguientes:

- **Consistencia:** El principal requerimiento de calidad que se debe medir en este proyecto es el de consistencia, pues se requiere confirmar que la cantidad de registros almacenados en el data lake sea lo suficientemente similar a lo que se encuentra en las bases de datos de origen. El umbral definido para esta métrica es del 1 % hacia abajo (de 100 registros no puede faltar más de 1), y del 20 % hacia arriba (de 100 registros no se pueden tener más de 120). Esto se justifica porque se cuenta con Golden Gate, entonces se pueden generar diferencias entre la cantidad de registros que hay presentes en la base de datos origen y el data lake según a qué hora se realiza el refresco de la base de datos y si la tabla origen recibe actualizaciones en tiempo real o no, generando un desfase en la disponibilidad de la información entre ambos sistemas. Un umbral inferior haría que tablas que se encuentran bien puedan ser consideradas malas. No se tolera que falte mucha información ya que eso implicaría que hubo información que debería haber llegado que no llegó, lo cual es crítico para los procesos dependientes de los datos como reportes normativos, informes de contabilidad u oficios a organismos externos como SERNAC o la CMF, donde la ausencia de registros puede derivar en información incompleta y/o incorrecta. Por su lado, el umbral superior es más holgado puesto que el exceso de información no representa una pérdida, sino solo un desfase temporal entre el data lake y la base de datos origen, generado por la ventaja de contar con Golden Gate.
- **Unicidad:** La segunda métrica considerada es la de unicidad, pues no deben existir registros duplicados si en la base de datos original no están. Para este requerimiento no hay un umbral, pues no se puede tener ningún registro duplicado. Esto se medirá a través de las llaves primarias de cada tabla, la cual se encuentra almacenada tanto en la base de datos original como en una tabla de DynamoDB con metadatos en AWS.
- **Exactitud:** La tercera métrica a considerar es la de exactitud. Esto pues hay tablas que no se ingestan completas siempre, sino solo los datos de los últimos 2 meses. Y además, pueden no contar con una columna con la última fecha de actualización, por lo que si un registro antiguo se actualiza, puede no verse reflejado en el data lake al no poder identificar fácilmente estos cambios, lo que podría generar reportes inexactos. En el caso de la base de datos que cuenta con Golden Gate, es validar que los cambios que se realizaron en la base de datos original se repliquen correctamente en el data lake, pues no hay una buena manera de realizar esta medición actualmente.

Estas evaluaciones serán realizadas de manera automática, y a diario, con la finalidad de tener el estado de calidad de las tablas el mismo día que los datos son utilizados y así mismo,

que el equipo pueda actuar de manera rápida y efectiva en caso de presentarse alguna discrepancia. La periodicidad se justifica con la frecuencia de actualización que tienen los datos en el data lake, ya que no tendría valor evaluar la calidad cada día si los datos son actualizados una vez a la semana. Sin embargo, como los datos se actualizan cada una hora (aquellas tablas con Golden Gate), se vuelve crucial la evaluación al menos diaria.

Los resultados de las evaluaciones serán almacenados en tablas dentro del mismo data lake, para poder consultar de manera centralizada el estado. Además de esto, se implementará un sistema de reportería que enviará a diario por correo un reporte de tipo semáforo con el estado de cada tabla, esto para las métricas de Consistencia y Unicidad. El almacenamiento de los resultados en una tabla dentro del data lake permite trazabilidad histórica, incluyendo la posibilidad de ver cómo evolucionó el estado de calidad de las tablas a lo largo del tiempo desde que se implementó el módulo de calidad de datos. Para los resultados de Exactitud, se implementará un sistema de alertas que notificará al equipo a través de Microsoft Teams en caso de encontrar alguna discrepancia.

El proceso de mejora continua será responsabilidad de los ingenieros de datos del equipo, quienes revisarán diariamente los reportes de Consistencia y Unicidad recibidos por correo, y las alertas de Exactitud notificadas por Microsoft Teams. Ante cada discrepancia reportada, se seguirá el siguiente proceso:

1. **Priorización:** Las discrepancias de Unicidad serán tratadas con prioridad alta al no tener umbral tolerable. Las de Consistencia serán priorizadas según la criticidad de la tabla afectada y la magnitud de la diferencia respecto al umbral definido.
2. **Investigación y corrección:** El ingeniero responsable identificará la causa raíz de la discrepancia, determinando si el origen es un problema en el proceso de ingesta, en la fuente de datos, o en la lógica de transformación, e implementará los cambios necesarios.
3. **Seguimiento de reincidencias:** Al almacenarse los resultados históricamente en el data lake, es posible identificar si una discrepancia ya fue reportada anteriormente y no fue resuelta de forma definitiva, permitiendo escalar su prioridad en caso de reincidencia.
4. **Incorporación de nuevas reglas:** A medida que se incorporen nuevas tablas al data lake, el módulo puede extenderse agregando registros a la tabla de configuración, sin necesidad de modificar el código del módulo.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

En esta sección se presentará la propuesta de solución en base a lo definido en el capítulo 2.

3.1. Levantamiento de requerimientos

Durante el levantamiento de requerimientos, los stakeholders mencionaron discrepancias en la cantidad de registros que tienen las tablas en el data lake. También reportan diferencias en la configuración de las tablas en comparación a su versión original. Otra problemática mencionada fue la de valores distintos en ciertas columnas para una misma fila.

Tras varias reuniones, se determinó que los stakeholders necesitan cumplir con varias exigencias:

- **Entorno de ejecución:** El entorno de ejecución debe ser AWS Glue, pues se requiere de una Glue Connection en particular que llegue a la red privada de la empresa, donde están alojadas las bases de datos.
- **Costos:** Los costos deben ser los menores posibles, pues son muchas tablas a considerar. Esto implica el uso de artefactos AWS Glue Python Shell, en vez de Glue ETL, es decir, Python puro y no PySpark.
- **Herramientas:** Se requiere diseñar un módulo propio que permita comparar los datos en el data lake con los datos en la base de datos origen, más que la detección de nulls o valores fuera de parámetros.
- **Escalabilidad:** La solución debe ser escalable, con la posibilidad de añadir nuevas tablas para su evaluación de una manera simple.
- **Replicabilidad:** La solución debe ser replicable fácilmente, con la idea a futuro de implementar lo mismo en otros data lakes presentes en el entorno de Ripley Corporativo.

Se desarrollará una primera versión, que validará la base de datos cdcreporitf dentro del data lake. Esta corresponde a la base de datos que cuenta con replicación mediante Golden Gate.

Los requerimientos se relacionan con los objetivos de la siguiente manera:

Tabla 1: Matriz de Trazabilidad de Requerimientos y Objetivos del Proyecto

Requerimiento Stakeholder	Objetivo Específico Vinculado	Métrica / Criterio de Calidad	Componente Imple- mentado y Evidencia
Módulo Propio: Comparación directa origen vs destino.	OE-2: Identificar las diferencias entre los datos almacenados en el data lake con sus fuentes de origen.	Consistencia: Umbral -1 % y +20 % (Daily, Monthly, Full). Unicidad: $U = \frac{\text{count(distinct(pk))}}{\text{count(*)}} = 1$ Exactitud: Muestreo de 10k registros.	Módulo Python propio en Glue (consultas COUNT y DISTINCT en Oracle, Athena y Redshift). Evidencia: Tablas de resultados históricos en el data lake.
Escalabilidad: Añadir nuevas tablas para su evaluación de una manera simple.	OE-4: Proceso de mejora continua.	Tiempo y complejidad de configuración al acoplar una nueva tabla.	Tabla de metadatos en Amazon DynamoDB. Evidencia: Inserción de nuevos registros en DynamoDB sin alterar el código base.
Replicabilidad: Facilidad de despliegue en otros data lakes del entorno Corporativo.	OE-4: Proceso de mejora continua.	Modularidad, desacoplamiento del código y parametrización.	Código modular en Python parametrizado.
Monitoreo Diario y Mejora Continua: Automatización diaria y flujos operativos ante incidentes.	OE-1: Definir e implementar reglas. OE-4: Proceso de mejora continua. OE-3: Desarrollar un dashboard.	Frecuencia de evaluación (Diaria frente al refresco de Golden Gate).	Reportes Email (Semáforo) para Consistencia/Unicidad y alertas en MS Teams para Exactitud. Evidencia: Correos/alertas recibidas.

3.2. Selección de tecnologías

A continuación se presentará la selección de distintas tecnologías y su justificación de por qué se eligen por sobre otras.

3.2.1. AWS Glue Python Shell

Para la ejecución de los scripts se utilizará AWS Glue en su versión de Python Shell, con 0,0625 DPU. Esto se hace para seguir dentro del ambiente de AWS y tener la conexión configurada a las bases de datos. Otra opción plausible era la de AWS Lambda, pero como estas ejecuciones pueden tardar desde pocos minutos a horas, al ser muchas tablas y grandes, se descartó el uso de AWS Lambda por la relación que existe entre la duración de una ejecución con el costo de esta [AWS, sff].

3.2.2. AWS DynamoDB

Para el almacenamiento de consultas SQL a utilizar y la configuración para la orquestación del módulo, se creará una tabla en DynamoDB. Se utilizará DynamoDB por su simpleza y su buena integración con el resto de servicios de AWS [AWS, sfb], además de su fácil acceso a través de librerías como boto3 de manera programática.

3.2.3. AWS Redshift

Algunas de las validaciones serán realizadas a través de Redshift, esto para ahorrar en costos asociados a la cantidad de datos leídos al utilizar otras herramientas como AWS Athena. Esto también permite aprovechar al máximo los recursos que están disponibles, al utilizar el cluster aprovisionado de Redshift en horas de poco o nulo tráfico (tardes, noches y madrugadas).

3.2.4. Great Expectations, Deequ / AWS Glue Data Quality, AWS Glue DataBrew

No se utilizará ninguna de las herramientas mencionadas en el capítulo 2, esto porque los requerimientos exigen el desarrollo de un módulo propio que permita comparar con el origen de los datos. Además, los costos del módulo deben ser los menores posibles, por lo que el uso de cualquiera de estas herramientas no cumpliría con dicha exigencia.

3.2.5. Amazon QuickSight y Microsoft Teams

La reportería asociada al módulo se realizará a través de Amazon QuickSight, para no salir del ambiente AWS y aprovechar los accesos existentes y directos entre Amazon QuickSight y Athena / Redshift [AWS, sfg], que no están presentes en otras herramientas de visualización como Power BI. Por otro lado, se elige Microsoft Teams por sobre Slack porque el equipo no utiliza Slack, lo que obliga que las alertas críticas deban ser a través de Microsoft Teams.

3.3. Arquitectura del módulo

Se presentarán tres arquitecturas distintas, una para cada dimensión evaluada de las tablas, pues cada una tiene un enfoque distinto.

3.3.1. Consistencia

Para evaluar la consistencia de las tablas, se utilizan dos tablas en DynamoDB con las siguientes estructuras para cada elemento:

```

1  {
2      "tableName": "tabla_trx",
3      "database": "cdcreporitf",
4      "aws_query_daily": "select count(1) from cdcreporitf.tabla_trx
5                          where fecha_trx >= current_date - interval '1' day and fecha_trx <
6                          current_date",
7      "ora_query_daily": "select count(1) from database.tabla_trx where
8                          fecha_trx >= trunc(current_date - 1) and fecha_trx < trunc(
9                          current_date)"
10     }

```

Listing 1: Estructura de la tabla consistencia-queries.

```

1  {
2      "workflowName": "cdcreporitf_daily",
3      "datacheck": "daily",
4      "tables": ["tabla_trx", "tabla_clientes"]
5  }

```

Listing 2: Estructura de la tabla consistencia-workflow.

La tabla consistencia-queries almacena las tablas que se pueden procesar, con la base de datos a la que corresponde, y las consultas SQL de validación, contando con validaciones por día, semana, mes y de la tabla completa. La tabla consistencia-workflow almacena los flujos a evaluar, con el lapso de tiempo a evaluar, y las tablas a validar.

La idea sería incorporar un elemento en la tabla consistencia-queries por cada tabla presente en cdcreporitf, y crear tres elementos en la tabla consistencia-workflow, uno para daily, otro

para monthly y otro más con el full, cada uno con todas las tablas de cdcreporitf en el atributo de tables.

Con esto definido, la arquitectura quedaría así:

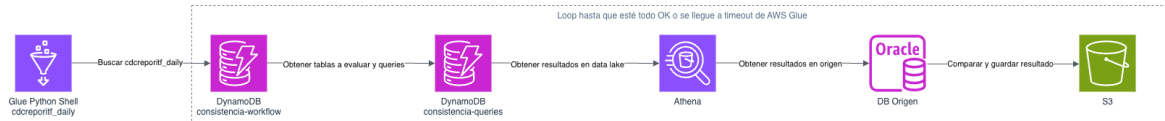


Figura 5: Arquitectura módulo consistencia.

Fuente: Elaboración propia.

Contando con una tabla de resultados, tabla dq_consistencia_logs, disponible en el data lake con la siguiente estructura:

```

1 CREATE TABLE br_arquitectura.dq_consistencia_logs (
2     nombre_tabla varchar,
3     base_datos varchar,
4     check varchar,
5     status integer,
6     year integer,
7     month integer,
8     day integer
9 );

```

Listing 3: DDL de la tabla dq_consistencia_logs.

Donde por cada tabla y cada check, se registra si el count externo coincide con el interno del data lake.

3.3.2. Unicidad

Para validar que cada registro es único en las tablas, y aprovechando las horas de bajo uso del cluster aprovisionado de Redshift, se realizó un stored procedure en SQL que ejecuta las consultas.

Esto se realizó generando nuevamente dos tablas, pero esta vez tablas SQL en Redshift. La tabla queries_unicidad presenta la siguiente estructura:

```

1 CREATE TABLE schm_arquitectura.queries_unicidad (
2     nombre_tabla varchar,
3     base_datos varchar,
4     query varchar,
5     query_pk varchar,
6     primary_key varchar
7 );

```

Listing 4: DDL de la tabla queries_unicidad.

Y la tabla queries_unicidad_result:

```

1 CREATE TABLE schm_arquitectura.queries_unicidad_result (
2     nombre_tabla varchar,
3     base_datos varchar,
4     result integer,
5     result_pk integer,
6     diferencia integer,
7     fecha_ejecucion timestamp
8 );

```

Listing 5: DDL de la tabla queries_unicidad_result.

Luego Glue invoca la ejecución del stored procedure, y almacena en una variable las filas de la base de datos especificada, procediendo a ejecutar los inserts, a la vez que realiza las queries de cada una de estos registros.

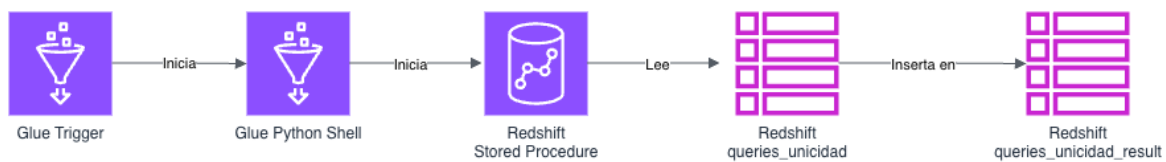


Figura 6: Arquitectura módulo unicidad.

Fuente: Elaboración propia.

3.3.3. Exactitud

Para evaluar la exactitud de los registros, se decidió crear una nueva tabla en DynamoDB con el fin de guardar las tablas que se compararán y con qué columna se extraerá una muestra de los registros.

La estructura de cada elemento en la tabla de DynamoDB sería:

```

1 {
2     "tableName": "tabla_trx",
3     "database": "cdcreporitf",
4     "columnName": "n_cuenta"
5 }

```

Listing 6: Estructura de la tabla exactitud-conf-tabla.

Con esta tabla creada, se lanzaría un Glue Job que primero extraería de la base de datos origen (Oracle) una muestra aleatoria en base a la columna columnName, y luego extraería todas las columnas de la tabla original y de la tabla del data lake de esos valores específicos de la columna columnName, comparándolos y emitiendo una alerta en caso de encontrar discrepancias con el detalle de la tabla y el valor con discrepancia.

Este mensaje será almacenado en otra tabla (tabla result_discrepancias) del data lake con la finalidad de poder evaluar a futuro la frecuencia de las detecciones.

```

1 CREATE TABLE br_arquitectura.result_discrepancias (
2     nombre_tabla varchar,
3     base_datos varchar,
4     mensaje varchar,
5     column_name varchar,
6     column_value varchar,
7     fecha_ejecucion timestamp
8 );

```

Listing 7: DDL de la tabla result_discrepancias.

Y la arquitectura de esta sección del módulo quedaría:

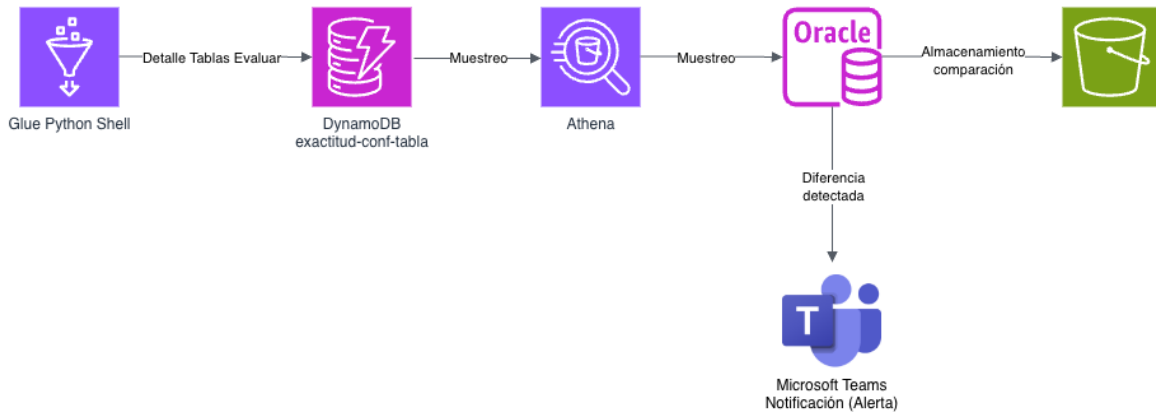


Figura 7: Arquitectura módulo exactitud.

Fuente: Elaboración propia.

3.4. Flujo de validación

En esta sección se describirá cómo funciona el flujo completo que realiza la validación. Para esto se volverá a separar en tres segmentos distintos, uno por cada dimensión que se analiza. Se menciona que lo que tienen en común las tres validaciones son que todas son iniciadas por un Glue Trigger iniciando la ejecución de un Glue Job en una hora y con parámetros específicos.

3.4.1. Consistencia

Al iniciar la ejecución del Glue Job, lo primero que se hace es obtener el valor del parámetro entregado en la ejecución. Este parámetro es el workflowName. En caso de no pasarse el

parámetro, la ejecución fallará. Ya obtenido el valor del parámetro, con la librería boto3 se realiza la consulta en la tabla consistencia-workflow en DynamoDB, filtrando por workflow-Name. Con esto se obtiene qué tipo de validación se debe realizar (daily, monthly, full) y qué tablas se deben evaluar. Con las tablas a evaluar, se consulta nuevamente en DynamoDB, pero esta vez a la tabla consistencia-queries, para obtener las queries a ejecutar de cada tabla.

El módulo concurrent.futures, propio de Python, permite la ejecución, en paralelo y de manera asíncrona, de las tareas que se requieran. Esto se consigue enviando a distintos hilos cada instrucción, en vez de realizar todo en un único hilo. La ventaja principal es la paralelización, y con ello, eficiencia en tiempo de ejecución, lo que resulta atractivo a la hora de tener que hacer consultas masivas y simples. El nivel de paralelismo que se puede obtener depende de la cantidad de hilos disponibles en el procesador y de la cantidad de paralelismo que puede soportar lo que sea que se requiere realizar. Para este caso, se utiliza un valor de diez hilos ejecutando consultas a la vez.

La librería oracledb habilita la conexión desde Python a bases de datos Oracle en Thin Mode. Debido a la versión de las bases de datos de la organización, fue necesario cargar a S3 un .zip con una librería de Oracle y con ello habilitar el Thick Mode en el código.

Para aprovechar mejor los recursos disponibles, se paralelizaron las consultas a las bases de datos Oracle y las consultas a Athena. Primero se realizan todas las ejecuciones en Oracle (de todas las tablas), y una vez terminadas, se procede con las ejecuciones en Athena (también de todas las tablas). Los resultados son almacenados en una lista de tuplas, estando esta última conformada por el par ("tabla", resultado). Luego, se procede con ordenar las listas de manera alfabética según el nombre de la tabla, y se itera en cada elemento de cada lista asegurando que las tablas existen en ambas listas, en caso contrario, se agrega con el resultado en cero en la lista donde falte alguna de las tablas.

El paralelismo no se pudo aumentar a más de diez hilos, puesto que Athena cuenta con una cuota limitada de 20 consultas. Dejar el valor mayor a diez podría ocasionar problemas en los procesamiento de datos y en otros procesos ejecutados dentro del ambiente de AWS.

Los resultados de las consultas a la base de datos son comparados con los resultados de las consultas a Athena. Si falta más de 1 % de los registros presentes en Oracle, o si hay más de un 20 % extra de registros en el data lake, se marca la tabla como mala. Esto se explica en la siguiente relación matemática:

$$diff = count_{oracle} - count_{aws}$$
$$diff > \frac{count_{oracle}}{100} \vee |diff| \geq 0,2 \times count_{oracle} \implies status = 0$$

Realizada la comparación entre ambos valores, se procede a escribir utilizando la librería awswrangler un parquet en la tabla dq_consistencia_logs. Esta consta de las columnas: nombre_tabla con el nombre de la tabla, base_datos con el nombre de la base de datos, check

con el tipo de validación realizada (daily, monthly, full), status con un valor de 0 si está mala la tabla o un valor de 1 si está bien y por último year, month y day, que son columnas partición de la tabla dq_consistencia_logs que permiten una mejor escalabilidad para realizar las consultas una vez que la tabla aumente su volumen de registros.

En el caso de que una tabla arroje que está mala, se entra en un bucle que vuelve a validar después de cinco minutos, hasta que se llegue al número máximo de reintentos o al timeout de la ejecución del Glue Job. Esto se hace de esta manera porque puede que a la hora en que se inicie la ejecución del Glue Job, se esté procesando la data a evaluar, entonces así se espera el tiempo necesario para comparar los registros finales en el data lake contra los de la base de datos.

3.4.2. Unicidad

Una vez iniciado el Glue Job, lo primero que se hace es obtener los valores de los parámetros entregados en la ejecución. Estos parámetros son el sp_name y el base_datos. En caso de no pasarse alguno de los parámetros, la ejecución fallará. Ya obtenido los valores de los parámetros, con la librería redshift_connector se realiza la conexión al clúster de Redshift donde se ejecutará el procedimiento almacenado. Con la conexión establecida, se lanza la ejecución del procedimiento almacenado dando como parámetro el valor de base_datos.

El procedimiento extrae las tablas a evaluar de dicha base de datos desde la tabla queries_unicidad, y procede a iterar en cada registro ejecutando un count completo de la tabla y un count distinct en base a la primary key de la misma tabla. Cada iteración escribe un registro en la tabla queries_unicidad_result, almacenando el nombre de la tabla, la base de datos, el resultado del count completo, el resultado del count distinct, la diferencia de estos valores y la fecha de ejecución de la consulta. De esta manera, gran parte de lo asociado a medir la unicidad de las tablas es realizado y almacenado exclusivamente en Redshift.

3.4.3. Exactitud

Durante la inicialización del Glue Job, lo primero que se hace es obtener el valor del parámetro entregado en la ejecución. Este parámetro es el base_datos. En caso de no pasarse el parámetro, la ejecución fallará. Ya obtenido el valor del parámetro, con la librería boto3 se realiza la consulta en la tabla exactitud-conf-tabla en DynamoDB, filtrando por database (base_datos). Con esto se obtiene el listado de tablas a las que hay que evaluar su exactitud y en base a qué columna se buscarán los registros.

El primer paso es obtener una muestra de registros, es decir, qué valores de la columna columnName serán comparados. Para esto se realiza una consulta en la base de datos Oracle, utilizando la librería oracledb para establecer la conexión antes. Obtenido los valores

de columnName que serán evaluados, se procede a ejecutar la consulta para obtener estos registros en Athena.

Con los resultados de Oracle y de Athena, se procede a comparar columna por columna, fila por fila, cada uno de los registros, buscando discrepancias en los valores de alguna de las columnas. En caso de encontrar una diferencia en algún valor, se envía una alerta a través de Microsoft Teams a un equipo y canal para la rápida visualización de esta.

El mensaje a través de Microsoft Teams se logra a través de la librería requests de Python (que posibilita enviar solicitudes HTTP de manera programática) y de un flujo en Microsoft Power Automate que se gatilla al enviar una solicitud de tipo POST a un link específico. El flujo inicia al recibir el POST, y envía un mensaje a través de Microsoft Teams. Este mensaje es personalizable desde el mismo código Python.

3.5. Implementación de métricas

En este apartado se explicarán las métricas evaluadas y cómo se utilizan para evaluar el estado de cada tabla.

3.5.1. Consistencia

Como ya se explicó, se realizan consultas SQL, específicamente counts, para obtener la cantidad de registros en un rango de tiempo determinado tanto en la base de datos origen (Oracle) como en el data lake (Athena). Se obtiene la diferencia porcentual entre ambos resultados, si falta un 1 % de registros en Athena de la cantidad de registros en Oracle, o si en Athena se tienen 20 % más registros que en Oracle, se define una tabla como mala para ese rango de tiempo determinado.

Esto por su cuenta no es de mucha ayuda, pues una tabla puede estar bien para el día previo, pero no para el mes completo o para la tabla completa. La idea final consta de tres evaluaciones distintas, una de tipo daily, otra de tipo monthly y otra de tipo full. La suma de los status entregará el estado real de la tabla.

3.5.2. Unicidad

La primary key, o llave primaria por su nombre en español, de una tabla SQL, indica qué grupo de columnas no pueden ser iguales para dos registros distintos. Por ejemplo, si hay una tabla con los habitantes de Chile, la llave primaria correspondería al rut de la persona, pues el rut es único para cada uno, no puede haber dos personas con el mismo rut.

Es por esto que para medir la unicidad se aprovecha esta propiedad de las tablas SQL, al poder obtener las llaves primarias de cada tabla, se definen consultas (`count distinct`) que entregan la cantidad de resultados de agrupaciones distintas de las columnas que son llave primaria. Si una tabla con su `count distinct` tiene 1.000.000 registros y con su `count completo` tiene 1.000.001, es porque existen registros con valores duplicados en su llave primaria, lo que rompe la unicidad dentro de la tabla.

Esto también se puede traducir matemáticamente a:

$$Unicidad = \frac{count(distinct(pk))}{count(*)} = 1$$

3.5.3. Exactitud

Para medir la exactitud, se realiza un muestreo aleatorio para evitar sesgos al poder elegir siempre un grupo de valores repetitivo. También se realiza un muestreo porque hay tablas que cuentan con millones de registros que serían casi imposibles de comparar completamente, recordando el enfoque bajo en costos que se busca.

La comparación por columna de cada registro permitirá detectar discrepancias en valores específicos dentro de las filas evaluadas, lo que disminuiría la exactitud de la tabla, pero permitirá la detección y futura corrección de los registros conflictivos.

Se reconoce que una muestra de 10.000 registros no es estadísticamente representativa de tablas con millones de registros, por lo que este mecanismo no pretende garantizar la exactitud sobre la totalidad de los registros de la tabla. Su propósito es distinto: actuar como un control de detección temprana orientado a identificar errores sistemáticos en los procesos de ingesta, tales como conversiones incorrectas de tipos de datos, truncamiento de valores, o pérdida de precisión numérica.

Este tipo de errores, al ser producto de un proceso defectuoso y no de un evento aislado, tienden a manifestarse de manera consistente a lo largo de los registros. Por lo tanto, una muestra distribuida en el tiempo es suficiente para detectarlos con alta probabilidad, aun cuando no cubra la totalidad de los datos.

3.6. Persistencia y reportería

Los resultados de cada evaluación serán guardados o en el data lake, o en Redshift. La persistencia de estos resultados permitirá la medición histórica, para ver la mejora (o no mejora) de cambios grandes que se pueden realizar a lo largo del tiempo. También permite la conexión desde sistemas de reportería a las tablas con los resultados, con la finalidad de enviar paneles con la información de una manera más legible.

Para la reportería, una de las herramientas elegidas es Amazon QuickSight. Esto se justifica porque así se continúa dentro del ambiente de AWS, y además, QuickSight cuenta con conectores nativos a Redshift y a Athena, lo que permite una sencilla y rápida extracción de los datos a utilizar. Para la consistencia se elige un reporte de tipo semáforo, donde se visualizará el nombre de la tabla, y un círculo con un color, rojo, amarillo o verde. Todos los días se realizará la evaluación de consistencia en tres niveles. Daily para validar los datos del día anterior, Monthly para evaluar los registros del mes completo, y por último, Full, para comparar los registros totales de la tabla. Para que una tabla esté en verde, las tres mediciones deberán estar en rango, es decir, la suma de los status para esa tabla en ese día específico debe sumar 3. Si la suma de los status suma 2, se muestra en amarillo. Y si la suma de los status es 1 o 0, se muestra en rojo. Esto permitirá la rápida visualización del estado de las tablas.

La ejecución de los Glue Jobs estará programada en cierto horario. Poco después de la hora límite de estas ejecuciones se lanzará la actualización de los datos desde QuickSight. Y aproximadamente media hora después de iniciada la actualización, se enviará automáticamente un correo con un archivo PDF adjunto que corresponde a cómo se ve el panel después de la actualización de los datos, con el estado más actualizado de las tablas.

El reporte con el estado de la unicidad de las tablas también se encontrará alojado en QuickSight. El proceso SQL se ejecutará todos los días por la madrugada, en un horario con poco tráfico en el clúster de Redshift. En las mañanas, desde QuickSight se irán a buscar los nuevos datos para el día a Redshift, consultando la tabla en la que se almacenarán los resultados. En este caso el reporte será uno más simple, que solo ahorrará el tener que conectarse manualmente a Redshift a ejecutar la consulta, enviando la misma tabla a través de un correo automático todos los días.

Para el caso de la evaluación de exactitud, debido a que las diferencias detectadas requieren revisión puntual sobre registros específicos, se priorizará el envío de alertas inmediatas a través de Microsoft Teams por sobre la generación de un panel.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

En esta sección se presentará la validación de la solución implementada. La evidencia presentada combina ejecuciones operativas sobre datos reales del data lake y también pruebas controladas con escenarios preparados deliberadamente, utilizadas para verificar que los mecanismos de detección funcionan correctamente. Para cada dimensión de calidad se especifica explícitamente qué tipo de evidencia sustenta los resultados presentados.

4.1. Estrategia de validación

Las validaciones constarán de cuatro aristas. Preparación del escenario de prueba, donde se especifica si se realizó alguna preparación antes de ejecutar el módulo o no. Ejecución de las validaciones, donde se muestran y explican las ejecuciones de los Glue Python Shell que realizaron las validaciones. Resultados obtenidos, donde se muestran los resultados obtenidos en el paso anterior, mostrando si se logra o no detectar las anomalías posibles. Y por último, análisis de resultados, donde se hace un corto análisis de los resultados obtenidos.

4.2. Validación de la consistencia

En la presente subsección se mostrarán los resultados de la implementación de evaluación de consistencia de las tablas.

4.2.1. Preparación del escenario de prueba

Antes de ejecutar el módulo por primera vez, se pensó en crear tablas temporales, con datos faltantes a propósito, con la finalidad de evaluar si efectivamente el módulo de consistencia lograba detectar las discrepancias. Sin embargo, se optó por ejecutar el módulo sin forzar positivos, para probar el código original en las tablas productivas directamente.

4.2.2. Ejecución de las validaciones

Desde el 12 de junio de 2025, todos los días se ha ejecutado la validación de consistencia de manera automática, validando que la cantidad de registros esté correcta para el día anterior, el mes completo y la tabla completa.

Run status	Start time (Local)	End time (Local)	Duration	--workflow
✔ Succeeded	02/13/2026 09:30:20	02/13/2026 09:36:18	5 m 43 s	cdc_daily
✔ Succeeded	02/13/2026 09:20:40	02/13/2026 09:23:56	3 m 10 s	cdc_monthly
✔ Succeeded	02/13/2026 09:13:08	02/13/2026 09:25:59	12 m 45 s	cdc_full

Figura 8: Ejecuciones correctas módulo consistencia.
Fuente: Elaboración propia.

La figura 8 muestra las tres ejecuciones del módulo del día 13 de febrero de 2026. La separación de varios minutos entre cada ejecución, como se mencionó en el capítulo 3, se debe a la cuota máxima que tiene Athena a la hora de realizar consultas. El estado de la ejecución en Succeeded indica que todas las tablas se encontraban bien en su cantidad de registros. Las ocasiones en las que alguna de la validación no es positiva para todas las tablas, se cuenta con un timeout para que no siga reintentando, como se puede ver en la figura 9.

Run status	Start time (Local)	End time (Local)	Duration	--workflow
⚠ Timeout	04/06/2026 09:30:09	04/06/2026 09:45:26	15 m 1 s	cdc_daily
⚠ Timeout	04/06/2026 09:20:47	04/06/2026 09:35:54	15 m 1 s	cdc_monthly
⚠ Timeout	04/06/2026 09:13:42	04/06/2026 09:28:49	15 m 1 s	cdc_full

Figura 9: Ejecuciones en timeout módulo consistencia.
Fuente: Elaboración propia.

4.2.3. Resultados obtenidos

Los resultados de las ejecuciones son almacenados en una tabla dentro del data lake. En la figura 10 se puede ver la cantidad de validaciones realizadas durante la semana del CyberDay de octubre de 2025 (incluye días PreCyber), donde se realizaron validaciones constantes a los datos de las tablas CDC.

year	▼	month	▼	day	▼	validaciones_realizadas
2025		10		10		426
2025		10		9		412
2025		10		8		427
2025		10		7		446
2025		10		6		417
2025		10		5		1346
2025		10		4		1229

Figura 10: Validaciones CyberDay Octubre 2025.

Fuente: Elaboración propia.

Además de esto, para verlo de manera más directa, se cuenta con un panel en QuickSight de tipo semáforo, que se envía por correo todos los días. Las tablas en verde es porque tienen las tres validaciones correctas, las que están en amarillo fallaron en una validación y, por último, las que están en rojo fallaron en dos o más validaciones. La figura 11 muestra el estado de las tablas para el día 5 de abril de 2026, cuando debido a un fallo de conexión entre la base de datos y AWS, se detuvo el envío de archivos CDC, lo que generó discrepancias en los datos, mostrando correctamente el estado de las tablas en colores verde, amarillo y rojo. Por el contrario, en la figura 12 es posible visualizar el estado de las mismas tablas para el día 9 de abril de 2026, con la incidencia ya superada por completo.



Figura 11: Reporte automático de consistencia del 5 de abril de 2026.
Fuente: Elaboración propia.



Figura 12: Reporte automático de consistencia del 9 de abril de 2026.
Fuente: Elaboración propia.

4.2.4. Análisis de resultados

La dimensión de consistencia, como se observó anteriormente, es evaluada correctamente por el módulo, permitiendo ver rápidamente el estado de las tablas, y actuar con poca latencia apenas las discrepancias son detectadas.

Además de esto, es posible ampliar estas validaciones a otras tablas y otras bases de datos, permitiendo escalar el módulo para abarcar más conjuntos de datos dentro del data lake. Como también fue posible revisar, la ejecución es bastante rápida, comparando más de 130 tablas en solo 3 minutos para el caso de la validación mensual, esto gracias al paralelismo incluido dentro del código.

4.3. Validación de la unicidad

En la presente subsección se mostrarán los resultados de la implementación de evaluación de unicidad de las tablas.

4.3.1. Preparación del escenario de prueba

Para la validación del módulo de unicidad no se realizó ninguna preparación de escenario. Esto puesto que otras áreas reportaron a los stakeholders que existían registros duplicados en algunas tablas, y al validar manualmente (ejecutando las mismas consultas pero a mano), constataron que efectivamente había registros duplicados. Esto aumentó la prioridad para implementar el módulo de unicidad para tener la detección automática de registros duplicados en las más de 130 tablas con CDC implementado.

4.3.2. Ejecución de las validaciones

Como ya existían los artefactos para poder ejecutar procedimientos almacenados en Redshift desde Glue, se aprovechó esto. Debido a la cantidad de tablas, y cantidad de registros en cada una, se optó por dejar algunas tablas fuera de la validación. Las tablas excluidas corresponden mayoritariamente a tablas dimensión, con pocos registros y pocas, o nulas, actualizaciones. De esta manera se evita invertir tiempo y recursos en validar tablas que no cuentan con cambios constantes. A pesar de este cambio, la mayoría de las ejecuciones para validar la unicidad llegaban al timeout del Glue Job, tal como se puede ver en la figura 13, y al no terminar correctamente, los resultados parciales no fueron persistidos.

Run status	Start time (Local)	End time (Local)	Duration	--sql_file_key
Timeout	05/30/2026 00:10:17	05/30/2026 03:10:42	3 h 10 s	redshift-sql-scheduled/call_sp_eaqr_count_dup_cdcreporitf.sql
Timeout	05/29/2026 00:10:17	05/29/2026 03:10:49	3 h 10 s	redshift-sql-scheduled/call_sp_eaqr_count_dup_cdcreporitf.sql
Timeout	05/28/2026 00:10:17	05/28/2026 03:10:34	3 h 10 s	redshift-sql-scheduled/call_sp_eaqr_count_dup_cdcreporitf.sql
Timeout	05/27/2026 00:10:17	05/27/2026 03:10:33	3 h 10 s	redshift-sql-scheduled/call_sp_eaqr_count_dup_cdcreporitf.sql
Timeout	05/26/2026 00:10:17	05/26/2026 03:10:33	3 h 10 s	redshift-sql-scheduled/call_sp_eaqr_count_dup_cdcreporitf.sql

Figura 13: Ejecuciones en timeout módulo unicidad.

Fuente: Elaboración propia.

Es por esto que se optó por separar en dos ejecuciones distintas la validación de unicidad, ejecutando solo una limitada cantidad de tablas distintas en cada ejecución. Con esto, se logró reducir la cantidad de timeouts finales, como se puede visualizar en la figura 14.

Run status	Start time (Local)	End time (Local)	Duration	--sql_file_key
✔ Succeeded	06/14/2026 03:10:22	06/14/2026 04:10:35	1 h 7 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/14/2026 00:10:17	06/14/2026 03:02:23	2 h 52 m	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/13/2026 03:10:22	06/13/2026 04:12:19	1 h 1 m 51 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/13/2026 00:10:17	06/13/2026 02:59:14	2 h 48 m 51 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/12/2026 03:10:22	06/12/2026 04:02:21	51 m 53 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/12/2026 00:10:17	06/12/2026 02:59:45	2 h 49 m 22 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/11/2026 03:10:22	06/11/2026 04:02:00	51 m 32 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql
✔ Succeeded	06/11/2026 00:10:17	06/11/2026 02:59:29	2 h 49 m 3 s	redshift-sql-scheduled/call_sp_eaerq_count_dup_cdcreporitf2.sql

Figura 14: Ejecuciones correctas módulo unicidad.

Fuente: Elaboración propia.

4.3.3. Resultados obtenidos

Como se vio en el capítulo 3, los resultados de la validación de unicidad son almacenados en una tabla dentro del mismo Redshift. Para no tener que conectarse a Redshift para ejecutar una consulta para obtener los resultados de la validación, nuevamente se utiliza QuickSight para generar un reporte diario y automático que envía el resultado de la consulta para ver el resultado del día actual, como se puede ver en la figura 15, con el reporte automático del día 6 de junio de 2026.

Tablas con Registros Duplicados (PK)					
base de datos	tabla	fecha_revision	n	n distinct	registros_extra
cdcreporitf	bopers_mae_ide	Jun 6, 2026	12,073,214	12,073,210	4
cdcreporitf	bopers_mae_ing_eco	Jun 6, 2026	10,533,528	10,533,527	1
cdcreporitf	bopers_mae_prd_cli	Jun 6, 2026	23,766,462	23,766,447	15
cdcreporitf	sfadmi_bco_btc_sol	Jun 6, 2026	48,190,156	48,189,271	885
cdcreporitf	sfadmi_bco_sol	Jun 6, 2026	9,344,179	9,344,178	1

Figura 15: Reporte automático de unicidad del 6 de junio de 2026.

Fuente: Elaboración propia.

4.3.4. Análisis de resultados

Los resultados del módulo de unicidad, como se pudo ver con anterioridad, indican que la detección de duplicidades funciona correctamente. Sin embargo, también presenta “falsos positivos”. Estos “falsos positivos” se debe a que en Redshift, las columnas varchar, al hacer una consulta para obtener solo los distintos resultados, elimina los espacios vacíos del texto. Esto implica que dos registros, uno con valor “CC_____” (se utiliza _ para representar espacios vacíos con fines ilustrativos) y otro con valor “CC”, son marcados como iguales. Mientras que al ir a validar a la base de datos origen, estos registros son correctamente considerados como distintos. Si este problema es responsabilidad de los DBA de las bases de datos originales, o de los mantenedores del data lake, es una discusión que se escapa del alcance del escrito.

Estos “falsos positivos” introducen ruido en la interpretación de resultados, pero no invalidan la utilidad del módulo, pues logra detectar registros duplicados no solo dentro del data lake, sino también fuera del mismo, a pesar de que el módulo no consulta las tablas en la base de datos original. Además, se cuenta con positivos reales, por registros que efectivamente están duplicados en el data lake, lo que permite enfocar las correcciones directamente a las tablas con problemas.

4.4. Validación de la exactitud

En la presente subsección se mostrarán los resultados de la implementación de evaluación de exactitud de las tablas.

4.4.1. Preparación del escenario de prueba

Se preparó una ingesta temporal de una de las tablas, donde se definió el valor de una de las columnas de la tabla en un valor que no existe en la base de datos original (ver figura 16), y que es imposible de tener según la lógica de la tabla, con la finalidad de observar si logra detectar y reportar la inconsistencia forzada.

fecalta
2026-12-12
2026-12-12
2026-12-12
2026-12-12

Figura 16: Columna con valores imposibles preparada.
Fuente: Elaboración propia.

4.4.2. Ejecución de las validaciones

La ejecución de la validación se realizó a través de AWS Glue Python Shell, configurado a 0.0625 DPU.

4.4.3. Resultados obtenidos

La ejecución realizada logró detectar las discrepancias forzadas de buena manera, enviando la alerta a través de Microsoft Teams en el canal designado, como se puede ver en la figura

17.

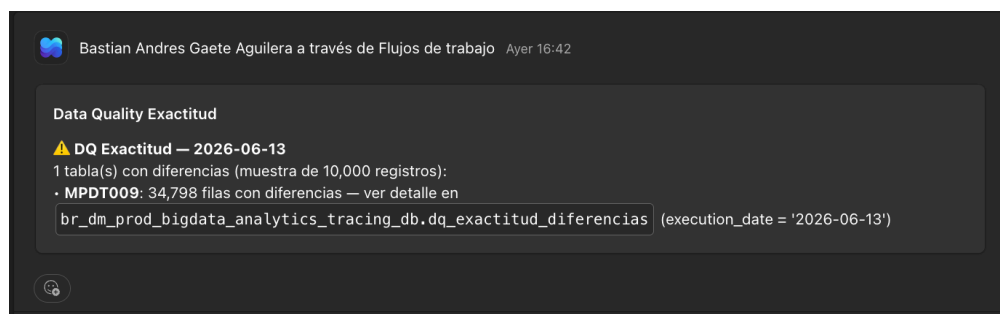


Figura 17: Mensaje resumen enviado a Microsoft Teams tras detección de discrepancias.
Fuente: Elaboración propia.

4.4.4. Análisis de resultados

Los resultados fueron los esperados, logrando enviar alertas de manera automática al detectar en una validación alguna diferencia en algunas de las columnas, guardando correctamente los valores afectados en una tabla en el data lake. Si bien la muestra que se extrae es de 10.000 registros, terminaron siendo más de 34.000 los registros comparados. Esto se debe a que las columnas que se utilizan para obtener la muestra no necesariamente son la llave primaria, entonces al obtener la muestra completa de la tabla (con todas las columnas), la consulta devuelve más de 10.000 registros, lo que amplía la cobertura de validación.

4.5. Evaluación global de la solución

Con los resultados obtenidos, es posible afirmar que las tres dimensiones consideradas fueron correctamente evaluadas, detectando en cada módulo los casos específicos de cada implementación. Se logra también una buena escalabilidad, siendo necesario solo agregar elementos en tablas de DynamoDB para agregar nuevas tablas a evaluar.

Todo esto permite la rápida y correcta detección de discrepancias en tablas del data lake, contribuyendo a mejorar la confiabilidad operativa de los datos y facilitando una acción temprana del equipo responsable. De esta manera, se incrementa la confianza de los equipos analíticos en la calidad de los datos disponibles en el data lake, al contar con un mecanismo que permite detectar tempranamente discrepancias relevantes (asociadas a las tres dimensiones evaluadas).

4.5.1. Cumplimiento de requerimientos

Respecto al entorno de ejecución, se logró cumplir con la limitación de ejecutar desde AWS Glue, logrando establecer conexiones con las bases de datos origen sin mayor complicaciones.

Por el lado de los costos, se lograron mantener acotados al utilizar AWS Glue Python Shell corriendo a 0.0625 DPU solamente, con un costo aproximado de 0.0275 USD por hora, con lo que se pueden ejecutar varias veces los módulos, cada uno por horas, y todo por unos pocos dólares al mes.

Se creó un módulo propio, autodiseñado, que permite comparar los datos presentes en el data lake con los datos originales, logrando escapar de la simplicidad de validar solamente valores nulos.

El uso de parámetros y otros servicios de AWS como DynamoDB y/o Redshift permite que los módulos sean altamente escalables, bastando en varias ocasiones con agregar elementos en tablas de DynamoDB para incorporar nuevas tablas en las validaciones, sin la necesidad de cambiar el código base de los scripts.

El código ya está siendo replicado e implementado en otros data lakes presentes en el entorno corporativo de Ripley, específicamente Retail Chile, sin mayores complicaciones, aprovechando la parametrización presente en las implementaciones.

Con esto se dan por cumplidos los requerimientos solicitados por los stakeholders, validados por ellos mismos.

4.5.2. Limitaciones identificadas

Al tener que utilizar AWS Glue Python Shell a 0.0625 DPU para mantener bajos los costos, se limita la cantidad de registros que se pueden validar en cada ejecución (al menos en el caso del módulo de exactitud) por la cantidad de memoria disponible. Si se utilizara PySpark y una mayor cantidad de DPU se podrían validar, en algunos casos, hasta tablas completas en una única ejecución.

Otra limitación identificada fue la cuota de uso de Athena / Gets de S3, que limita el paralelismo en la validación de consistencia e incluso puede generar errores críticos en el procesamiento de los datos si no es manejado con cuidado.

CAPÍTULO 5

CONCLUSIONES

En este capítulo se presentan las conclusiones, realizando una reflexión acerca del trabajo realizado. También se evalúan los objetivos definidos al principio del trabajo y cómo se cumplieron. Además se propondrán trabajos futuros a realizar para poder darle aún más valor a la solución realizada.

5.1. Conclusiones por capítulo

En esta sección se realizará un resumen de cada capítulo previo, destacando ciertos puntos del proyecto.

5.1.1. Definición del problema

En el capítulo 1 se contextualiza al lector acerca del negocio y los productos que ofrece Banco Ripley, información recopilada de las páginas web del Banco y sus filiales. También se explica el funcionamiento actual del sistema donde encuentra inmerso el problema, el framework que se utiliza, el stack cloud utilizado, el problema identificado, la organización del equipo que da soporte al data lake y la presentación de los objetivos del trabajo.

El análisis del funcionamiento actual permitió identificar que la principal vulnerabilidad del data lake no estaba en su arquitectura, sino en la ausencia de un control que midiera la fidelidad de los datos respecto a sus fuentes de origen.

5.1.2. Marco conceptual

En el capítulo 2 se presenta en primera instancia el estado de la práctica, explicando qué es un data lake, la diferencia entre estos con data warehouses y data lakehouses. Se explican conceptos y dimensiones de la calidad de datos, frameworks aplicables a la calidad de datos y se explica también el por qué estos frameworks no son aplicables para el problema a abordar.

En segunda instancia se exhibe el estado del arte, mostrando algunas soluciones orientadas a medir la calidad de los datos, haciendo un análisis de estas y argumentando por qué no se utilizó ninguna. También se explica lo que es el perfilamiento de datos, un paso crucial para una correcta definición de métricas y umbrales para considerar buena o mala una dimensión a evaluar.

Por último se muestra la metodología de la solución y la metodología de la validación, definiendo las dimensiones de calidad que serán evaluadas y cuál será el umbral aceptado para determinar el estado de las tablas.

La conclusión más relevante de esta sección fue que ni los frameworks ni las herramientas estudiadas ofrecen un soporte nativo para comparar un origen Oracle on premise con un data lake en AWS. Esto justificó el desarrollo de un módulo propio en lugar de utilizar una solución existente.

5.1.3. Propuesta de solución

En el capítulo 3 se detallan distintas aristas de la solución que se implementó, como los requerimientos a cumplir, las tecnologías a utilizar, la arquitectura del módulo, cómo se define el flujo de validación, la implementación de métricas y la persistencia y reportería de los resultados obtenidos.

De este capítulo es destacable la persistencia y la reportería, algo fundamental para una rápida detección y la posibilidad de evaluar la evolución de los resultados a lo largo del tiempo.

5.1.4. Validación de la solución

Por último se realiza la validación de la solución, explicando la validación y ejecución por cada dimensión evaluada, y luego analizando los resultados obtenidos, mostrando imágenes y evidencia de lo implementado, también hablando de los problemas encontrados.

Además de todo esto, se detalla el cumplimiento de requerimientos y las limitaciones identificadas, como la obligación de usar Glue Python Shell con solo 0.0625 DPU, la cuota máxima de gets en S3 y la detección de falsos positivos.

5.2. Conclusiones generales

En capítulos anteriores se exploraron distintas opciones para evaluar la calidad de los datos, como Deequ, la librería Great Expectations o AWS Glue DataBrew, sin embargo todas estas tenían una falencia, y es la incapacidad de comparar datos alojados en un data lake con datos almacenados en bases de datos on premise. Es debido a esto que se opta por diseñar e implementar un módulo de calidad de datos propio, con sus propios requerimientos y objetivos.

Lo implementado logra efectivamente la conexión en un mismo código al data lake en AWS y a la base de datos on premise, permitiendo detectar diferencias en las dimensiones de con-

sistencia, unicidad y exactitud sin depender de una revisión manual. Como se evidenció en el capítulo 4, el módulo identificó discrepancias reales, como la caída en el envío de archivos CDC del 5 de abril de 2026, y las reflejó automáticamente en los reportes diarios, lo que da un alto valor al trabajo realizado.

5.3. Cumplimiento de objetivos

Es posible medir la efectividad del trabajo realizado revisando el estado de los objetivos específicos descritos previamente. A continuación se presentan los resultados de cada uno:

- **Definir e implementar reglas y métricas de calidad de datos orientadas a detectar, registrar y reportar errores comunes:** Las reglas y métricas implementadas son descritas en las secciones 2.4, 3.4 y 3.5.
- **Identificar las diferencias entre los datos almacenados en el data lake con sus fuentes de origen:** Los resultados obtenidos en cada dimensión permiten evidenciar la correcta identificación de las diferencias, estos están disponibles en las secciones 4.2.3, 4.3.3 y 4.4.3.
- **Desarrollar un dashboard que permita visualizar de manera rápida el estado de calidad de las tablas monitoreadas por el módulo:** El dashboard principal que permite ver de manera rápida el estado de las tablas es el presentado en la sección 4.2.3, de tipo semáforo, que muestra en verde las tablas buenas, en amarillo las tablas con calidad media y en rojo las tablas que están malas.
- **Establecer un proceso de mejora continua para mantener y elevar los niveles de calidad de los datos en el data lake:** El proceso de mejora continua es detallado al final de la sección 2.4, definiendo responsables y los pasos del proceso.

Con los objetivos específicos cumplidos, y a partir de la evidencia presentada, se considera que el objetivo general, **diseñar e implementar un módulo de control de calidad de datos para un data lake, con el fin de mejorar la consistencia y confiabilidad de los datos almacenados**, fue cumplido dentro del alcance definido para el core REPORITF. Aun con esto, se desglosa el análisis para definir la confiabilidad según resultados de cada dimensión implementada.

5.3.1. Consistencia

Se consulta la tabla de resultados de evaluaciones de consistencia, visible en la figura 18:

día	suma_estados
2025-09-17	284
2025-09-18	284
2025-09-19	284
2025-09-20	284
2025-09-21	283
2025-09-22	284
2025-09-23	411
2025-09-24	411
2025-09-25	411

Figura 18: Resultados consistencia en septiembre de 2025.

Fuente: Elaboración propia.

En todos estos días, la cantidad de tablas evaluadas fue la misma, 137, sin embargo, se empieza a ver una mejora en el estado global de las tablas a partir del día 23 de septiembre de 2025, explicado por la correcta cantidad de registros en la evaluación de consistencia de tipo daily, monthly y full a la vez. En la figura 19 también es posible visualizar la evolución positiva obtenida durante 2025, donde mayo registra el peor valor en el estado promedio diario de las tablas, explicado por ser el primer mes con el módulo implementado de manera parcial y el hecho de que no estuvo habilitado gran parte del mes y octubre registra ya un valor mucho más alto, con casi todas las tablas buenas cada día. En base a estas evidencias, se considera el aporte de la dimensión de consistencia al valor de confiabilidad como positivo.

mes	promedio_estados
202505	35.0
202506	248.0
202507	280.0
202508	286.0
202509	329.0
202510	408.0

Figura 19: Resultados consistencia en ciertos meses de 2025.

Fuente: Elaboración propia.

5.3.2. Unicidad

Se consulta la tabla que contiene los resultados de la evaluación de unicidad, filtrando aquellas mediciones de `cdcreporitf` que contaron con diferencia en la cantidad de registros distintos según llave primaria con la cantidad de registros total de la tabla, los resultados están disponibles en la figura 20.

fecha_revision	tablas_con_duplicados
2026-02-02	1
2026-02-01	1
2026-01-31	1
2026-01-30	2
2026-01-29	2
2026-01-28	4
2026-01-27	5
2026-01-26	5

Figura 20: Resultados unicidad en ciertos días de 2026.
Fuente: Elaboración propia.

fecha_revision	tablas_evaluadas
2026-02-02	88
2026-02-01	88
2026-01-31	88
2026-01-30	88
2026-01-29	88
2026-01-28	88
2026-01-27	88
2026-01-26	88

Figura 21: Cantidad de tablas evaluadas en unicidad en ciertos días de 2026.
Fuente: Elaboración propia.

En todos estos días, como se puede ver en la figura 21, la cantidad de tablas evaluadas fue la misma, 88, y a pesar de esto, se empieza a ver una mejora en la cantidad de tablas que presentan registros duplicados, evidenciando la prioridad que tienen estos casos tan críticos que se buscan detectar. Con estas evidencias, se considera el aporte de la dimensión de unicidad a la confiabilidad de las tablas como positivo.

5.3.3. Exactitud

Las ejecuciones del módulo de exactitud en tablas productivas no arrojaron registros con discrepancias, los cuales se comprobó que son detectables gracias a las pruebas en la tabla con valores definidos a propósito. Si bien este resultado no permite concluir que todos los valores son exactos, debido a las limitaciones en la cantidad de registros comparables de manera simultánea, la ausencia de diferencias en los registros evaluados constituye en sí un indicador positivo del estado de exactitud de las tablas monitoreadas.

Con esto, el aporte de la evaluación de exactitud a la confiabilidad general de lo implementado se define como neutra, ni mejora ni empeora su valor.

5.4. Trabajo futuro

En esta sección se mencionan algunas ideas o mejoras que se pueden realizar a futuro en relación al módulo de calidad de datos implementado:

- **Interfaz gráfica:** A veces se vuelve operacionalmente complejo trabajar con elementos de tipo JSON a la hora de configurar nuevas tablas. Es por esto que se propone la creación de una interfaz gráfica, quizás alojada en una página web, que, conectada con las tablas de DynamoDB, muestre las tablas actuales que se están considerando y tenga las funcionalidades necesarias para agregar nuevas tablas o eliminar ya existentes.
- **Solución automática:** Si bien la detección automática y veloz de diferencias entre las tablas del data lake y las tablas origen es un gran paso, y aporta un valor considerable, se puede ir un paso más allá e implementar en un módulo aparte, o dentro del mismo módulo, la solución automática de errores detectados.
- **Incorporación de nuevas dimensiones:** Actualmente, el módulo implementa tres dimensiones, consistencia, unicidad y exactitud, sin embargo, aún quedan muchas dimensiones más que podrían ser implementadas en el módulo.
- **Consideración de nuevas bases de datos:** La implementación solo considera una de las bases de datos, pero está listo para agregar nuevas bases de datos Oracle. También se podría ampliar e incluir bases de datos SQL Server o MySQL.

REFERENCIAS BIBLIOGRÁFICAS

- [Apache Software Foundation, sf] Apache Software Foundation (s.f.). Apache NiFi. <https://nifi.apache.org/>. Consultado: 02 mayo 2026.
- [Aquino y Jonker, 2026] Aquino, J. y Jonker, A. (2026). What are data quality dimensions? <https://www.ibm.com/think/topics/data-quality-dimensions>. IBM. Consultado: 07 mayo 2026.
- [AWS, sfa] AWS (s.f.a). Amazon Athena. <https://aws.amazon.com/es/athena/>. Consultado: 02 mayo 2026.
- [AWS, sfb] AWS (s.f.b). Amazon DynamoDB. <https://aws.amazon.com/es/dynamodb/>. Consultado: 16 mayo 2026.
- [AWS, sfc] AWS (s.f.c). Amazon Redshift. <https://aws.amazon.com/es/redshift/>. Consultado: 02 mayo 2026.
- [AWS, sfd] AWS (s.f.d). Amazon S3. https://aws.amazon.com/es/s3/?nc=sn&loc=1&trk=32b96db8-849f-4365-bfe1-053cf18c09ae&sc_channel=ps. Consultado: 02 mayo 2026.
- [AWS, sfe] AWS (s.f.e). AWS Glue. <https://aws.amazon.com/es/glue/>. Consultado: 02 mayo 2026.
- [AWS, sff] AWS (s.f.f). Precios de AWS Lambda. <https://aws.amazon.com/es/lambda/pricing/>. Consultado: 16 mayo 2026.
- [AWS, sfg] AWS (s.f.g). Preparación de datos y conectividad de QuickSight. <https://aws.amazon.com/es/quick/quicksight/features/data-preparation/>. Consultado: 16 mayo 2026.
- [AWS, sf] AWS (s.f). What is AWS Glue DataBrew? <https://docs.aws.amazon.com/databrew/latest/dg/what-is.html>. Consultado: 08 mayo 2026.
- [Databricks, sf] Databricks (s.f). What is Medallion Architecture? <https://www.databricks.com/blog/what-is-medallion-architecture>. Consultado: 09 mayo 2026.
- [de León, 2025] de León, J. P. (2025). Calidad de Datos basada en Contexto en Data Lakes. Proyecto de grado, Universidad de la República (Uruguay). Repositorio Colibrí.
- [Fait et al., 2024] Fait, C., Babsalaam, A., Chakraborty, D., Irimpan, N., Shekhawat, S. S., y Joseph, J. (2024). Serverless Data Lake Framework (SDLF). <https://github.com/aws-solutions-library-samples/data-lakes-on-aws>. GitHub repository, README.md. Consultado: 07 marzo 2026.

- [Great Expectations, sf] Great Expectations (s.f.). GX Core. <https://greatexpectations.io/gx-core/>. Consultado: 08 mayo 2026.
- [IBM, sf] IBM (s.f.). What is data profiling? <https://www.ibm.com/think/topics/data-profiling>. Consultado: 08 mayo 2026.
- [ISO, 2020] ISO (2020). ISO 8000-2:2020 — data quality — Part 2: Vocabulary. Norma internacional. International Organization for Standardization, Ginebra, Suiza. <https://www.iso.org/standard/80543.html>. Consultado: 09 mayo 2026.
- [Kosinski, sf] Kosinski, M. (s.f.). Data warehouses vs. data lakes vs. data lakehouses. <https://www.ibm.com/think/topics/data-warehouse-vs-data-lake-vs-data-lakehouse>. Consultado: 09 mayo 2026.
- [Miller et al., 2025] Miller, R., Chan, S. H. M., Whelan, H., y Gregório, J. (2025). A Comparison of Data Quality Frameworks: A Review. *Big Data and Cognitive Computing*, 9(4).
- [Mohammed Muddasir y Raghuveer, 2017] Mohammed Muddasir, N. y Raghuveer, K. (2017). Study of Methods to Achieve Near Real Time ETL. En *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, pp. 436–441.
- [Oracle, sf] Oracle (s.f.). GoldenGate Features. <https://www.oracle.com/integration/goldengate/features/>. Consultado: 02 mayo 2026.
- [Prasad y Sayegh, 2026] Prasad, A. y Sayegh, R. (2026). What is a data lake? <https://learn.microsoft.com/en-us/azure/architecture/data-guide/scenarios/data-lake>. Consultado: 09 mayo 2026.
- [Rahm y Do, 2000] Rahm, E. y Do, H. (2000). Data Cleaning: Problems and Current Approaches. *IEEE Data Eng. Bull.*, 23:3–13.
- [Saber y Fariha, 2024] Saber, R. y Fariha, A. (2024). Formative Study for AI-assisted Data Visualization. *arXiv preprint arXiv:2409.06892*.
- [Schmidt et al., 2019] Schmidt, P., Lange, D., Schelter, S., y Rukat, T. (2019). Test data quality at scale with Deequ. <https://aws.amazon.com/es/blogs/big-data/test-data-quality-at-scale-with-deequ/>. Consultado: 08 mayo 2026.
- [Vágner, 2026] Vágner, A. (2026). Perspective Chapter: The Role of Data Quality Dimensions in Data-Driven Decision-Making. En Ventura, S., Luna, J. M., y Martín-Castaño, A. R. M., editores, *Data Quality Matters - Best Practices for Integrity and Assurance*, capítulo 4. IntechOpen, London.
- [Yeh y Puri, 2010] Yeh, P. Z. y Puri, C. A. (2010). An Efficient and Robust Approach for Discovering Data Quality Rules. En *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volumen 1, pp. 248–255.