**Escenario Actual**

En base a los relevamientos realizados y a partir de las necesidades comentadas por los distintos stakeholders de cada uno de los productos se puede detectar que: (Estos puntos los agregue como escenario para mejorar. Y abajo poner lo otro que sería la propuesta del data)

1. No existe una base de datos del tipo OLAP que centralice la información y ofrezca la performance adecuada para las consultas del tipo analítico, las cuales necesitan un tipo de cómputo distinto a las bases OLTP (Transaccionales) y, por lo tanto, están enfocadas en resolver ejecución de queries que requieren agrupación y sumarización de datos.
2. Debido al crecimiento exponencial de los datos y con el objetivo de mejorar los tiempos de respuesta de las consultas se realizó una separación en el almacenamiento de los datos, dividiendo los mismos por ventanas de tiempo en: Datos históricos, Datos recientes. Esto generó una mayor dispersión en los datos ya que cuando se requiere compilar información de varios periodos se genera la necesidad de consultar en más de una base.
3. El negocio necesita de una mirada global de los datos, lo cual para una arquitectura basada en microservicios (Una/Varias bases de datos por producto), genera dolores de cabeza al usuario ya que requiere efectuar consultas de manera separada en cada uno de los productos y luego compilar esa información para poder tener datos a nivel cross.
4. No realizar estimaciones de futuras ventas. Este es un punto crucial para que en el futuro se puedan entrenar modelos de Machine Learning, los cuales requieren que la data histórica sea almacenada y no sufra cambios en el tiempo para poder efectuar predicciones basadas en estos datos.
5. Ausencia de una herramienta de Gobierno de datos en la cual se pueda centralizar toda la documentación relacionada a cada una de las fuentes de información, el linaje de los datos, y que sirva como diccionario de datos a los usuarios finales. Hoy la mayoría del conocimiento reside en los stakeholder o se encuentra documentada únicamente a nivel producto por lo tanto no hay una mirada global de los datos.

**Escenario Esperado**

Se plantea el diseño de una arquitectura de datos que pueda resolver una a una todas las necesidades del negocio, stakeholders y usuarios técnicos. Para tal fin se aconseja el desarrollo de un proyecto que pueda ser implementado de manera evolutiva, permitiendo que haya una implementación por fases que continúe robusteciendo la arquitectura y agregando valor a las partes interesadas, además de otorgar flexibilidad a la hora de decidir el alcance de las implementaciones, las cuales podrán ser evaluadas en función del costo que genere cada fase previa y el esfuerzo demandado.

La solución propuesta a la problemática de los datos es la adopción de un DataLakeHouse[[1]](#footnote-2), el cual es una evolución del Data Lake y permite combinar las ventajas de un tradicional Data Warehouse, donde los datos estructurados se encuentran de manera consolidada y modelada en base a dimensiones y hechos que facilitan las consultas y permiten responder mejor y mas rápido a las preguntas que tiene el negocio, con las ventajas de un Data Lake el cual es un gran repositorio centralizado de datos donde se puede almacenar y centralizar toda la data histórica y transaccional de la empresa tanto estructurada, semi-estructurada o no estructurada.

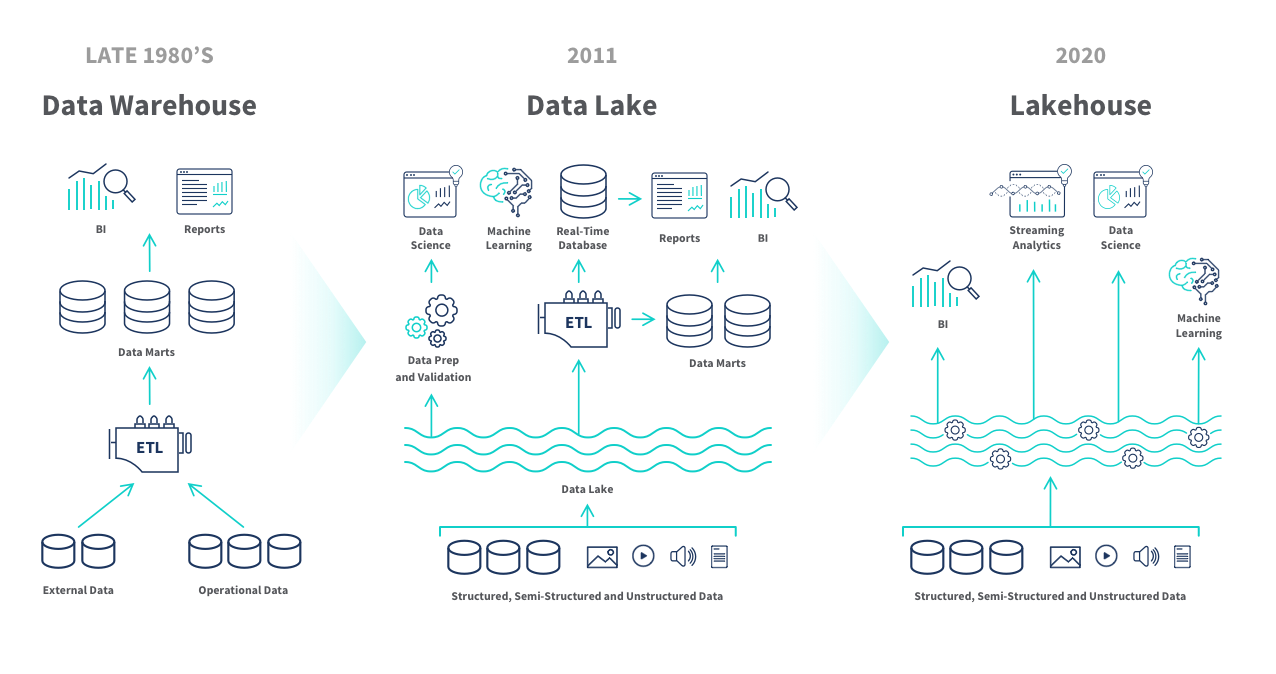


Figure 1. Evolución de los repositorios de datos

**Arquitectura y estrategia de Explotación de los Datos**

La implementación de una solución de datos no queda solamente en llevar a producción un DataLakeHouse que centralice la información, sino que requiere el desarrollo de toda una arquitectura de datos, la cual soporte y acompañe los distintos estadíos de la data, las estrategias de ingesta y los casos de uso de cada uno de los productos. Para esto se propone una adaptación de la Arquitectura del tipo Lambda[[2]](#footnote-3), la cual ofrece la combinación de datos en real time y de manera batch los cuales pueden ser consultados por los distintos usuarios.

A picture containing text, screenshot, font, diagram

Description automatically generated

Figure 2. Arquitectura Lambda con serving layer unificada

La adopción de esta arquitectura se puede realizar de manera evolutiva\*, en una primera fase se puede implementar solamente la capa de datos batch, en la cual se realizaría una ingesta inicial y posterior replicación de los datos con una ventana de tiempo de mins/hrs. Una segunda fase implementaría la capa de datos real-time, mediante la cual se podrá realizar una ingesta de cada registro nuevo que se guarda en una base datos y sus múltiples updates con una ventana de tiempo de ms/segs.

\**(Estas estrategias se detallan en el apartado de: Estrategia de Ingesta por Consumidor/Producto)*

Los componentes utilizados en esta arquitectura serán principalmente el stack de productos ofrecidos por Azure que son actualmente estándares en la industria.

A picture containing text, screenshot, diagram, font

Description automatically generated

Figure 3. Borrador de arquitectura propuesta, se resalta Fase 1.

**Componentes:**

1. Fuentes de datos: Los orígenes de datos principales son bases SQL Server. Por cada producto se detecta una base con los registros históricos, una con los datos recientes y por último una base de logs de las operaciones realizadas en cada microservicio. Sin embargo, la ingesta de datos no necesariamente se limite a estas bases de datos, sino que puede incorporar datos tanto Batch como streaming desde otros origines.
2. Ingesta de datos: Por medio del servicio de Azure DataFactory se pueden conectar cada una de las bases de datos on-premise/cloud para efectuar una replicación de datos hacia el storage de almacenamiento. Existen varias alternativas técnicas con respecto a este punto. Se detallan con mayor detalle en secciones posteriores.
3. Datalakehouse: Se compone de varios servicios, en primer lugar, Azure Blob Storage el cual es un servicio que permite almacenar grandes volúmenes de datos y cuyo costo varía en función del ciclo de vida del dato\* *(Se detalla en el apartado de estrategia de depuración del datalake).* Dentro del servicio de almacenamiento los datos se encuentran divididos en diferentes stages \*\**(Se detalla en el apartado de diseño de los diferentes stages*). Para procesar los datos en los diferente stages se hace uso Synapse Pipelines. Esta herramienta simplifica el proceso de transformación de datos ofreciendo la posibilidad combinar SQL con sintaxis Python, lo cual contribuye al desarrollo de ETLs robustos y escalables proporcionando modularidad en los mismos. Este modularidad permite efectuar y aislar cambios sin afectar a toda la arquitectura.
4. Serving Layer: Azure Synapse es el motor SQL que permite efectuar consultas a grandes volúmenes de datos almacenados en Azure Blob Storage. Es un motor orientado en el procesamiento de queries que requieren la lectura y procesamiento de millones de registros y en tiempos de respuesta cortos.
5. Consume: Es el conjunto de herramientas que hace posible la explotación de los datos almacenados, para el desarrollo de tableros de BI se estaría haciendo uso de PowerBi, combinado con las Azure Notebooks, las cuales permiten realizar análisis e inteligencia de datos por medio de código Python. \*\**(Se detalla a continuación en la explotación de datos).*
6. Orquestación: Para que las distintas herramientas puedan comunicarse entre si y ejecutar los procesos que corresponde en cada una de ellas, es necesario contar con una herramienta que orqueste estos procesos. En este caso Azure Data Facotry cuenta con capacidades de orquestación y es una solución robusta y la cual es usada en la gran mayoría de implementaciones de arquitectura de datos basadas en Azure.

**Explotación de Datos**

Para la explotación de datos del Datalakehouse se propone el uso de herramientas centralizadas, para esto se dispondrá de un motor SQL Azure Synapse que permite efectuar cada una de las diferentes queries que requiera el negocio sin importar su volumen, producto y ventana de tiempo del dato. Esta explotación puede ser efectuada a través de un cliente SQL de escritorio, aunque se aconseja el uso de un cliente SQL en la nube, ya que permite la interacción de los distintos equipos sobre una misma query, compartir queries, un historial de uso y cambios sobre queries almacenadas. Synapse puede disponer tanto de consultas serverless como pools dedicados, y se pueden utilizar ambos según el caso de uso.

Del lado de las necesidades de BI proponemos utilizar PowerBI, la cual es una herramienta que ya tiene casos de uso en la arquitectura actual, pero que al realizar el cambio en la manera en que se consulta los datos (bases de datos OLTP hacia el datalakehouse (Base de datos OLAP)) se evidenciará una mejora en lo que a performance refiere y la posibilidad de explotar la información y, sobre todo, poder cruzar datos de distintas bases de datos.

El datalake puede ser consumido a través de plataformas como Azure Notebooks, Azure ML Studio o bien un entorno de Databricks, los cuales son entornos de código Python + SQL que facilitan la explotación de datos a través de un entorno colaborativo y que sirve para realizar minería de datos, analítica de datos y modelos de machine learning.

Otros modos de explotar las capacidades del datalake es disponibilizando los datos para consultas dinámicas de baja latencia sobre datos agregados o streaming procesado. Para esto se propone realizar un “reverse ETL” en el cual se cargan datos procesados en una base de datos apta para su consumo a través de APIs (por ejemplo, CosmoDB o Redis). A su vez los datos procesados por streaming pueden ser consumidos como eventos a través de una cola (e.g. Kafka, Events Hub, etc), facilitando integraciones entre microservicios u otros procesos.

**Estrategia de Ingesta por Consumidor/Producto.**

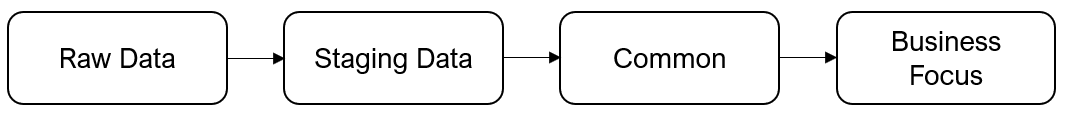
En una primera fase se realizaría la implementación de la capa de ingesta Batch. En base al relevamiento realizado con cada producto se debe definir la ventana de tiempo mínima con la que se requiere contar con los datos actualizados para su consumo.

Por medio de Azure Data Factory [[3]](#footnote-4) se conectará cada una de las bases de datos las cuales se replicarán en su totalidad (Se exceptúan tablas de configuración/parametría). Esta replicación se basa generalmente en campos del tipo timestamp que permitan en una primera instancia una ingesta inicial de toda la base y a partir de ese punto, ingestar únicamente los nuevos registros, es decir, los posteriores al último timestamp ingestado. Una buena práctica demanda que estos datos se ingesten de manera bruta, esto es, sin la intervención de ningún proceso de ETL. De esta manera, los datos son un fiel reflejo de la base de datos. La replicación de cada una de las tablas de las bases de datos hacia el servicio de almacenamiento Azure Blob Storage hace uso de un formato de datos llamado Delta[[4]](#footnote-5). Los archivos del tipo Delta permiten el almacenamiento y posterior consulta de grandes volúmenes de datos gracias a que son formatos columnares, además, en el caso de Delta, provee ACID con lo cual las ventajas de las bases de datos SQL también se mantienen en el Datalakehouse.

Para una segunda fase proponemos realizar la implementación de la capa de ingesta Real Time. Para esta capa se requiere que las bases de datos dispongan de la opción de CDC (Change Data Capture) lo cual permite leer en vivo cada transacción realizada en la base de datos y enviar las mismas a un sistema de colas, el cual posteriormente procesa cada uno de los registros y la ingesta en el mismo tipo de archivo Delta dentro del Datalakehouse. La importancia de la utilización de CDC pasa por el hecho de capturar la transacción en el momento que se realiza, dado que, si usara una estrategia de timestamp, siempre la replicación sería más lenta que la velocidad en la cual un dato puede cambiar.

**Diseño de los diferentes stages**

El diseño de una solución de datalakehouse requiere la implementación de varios estadíos de los datos. A continuación, se muestra un esquema y la descripción de cada uno de ellos:

****

Cada stage es conocido como una layer (capa) del Datalakehouse.

* En un primer stage, los datos que se replican desde las bases de datos se ingestan de manera bruta, es decir, sin alteraciones de por medio. Esta capa funciona como una copia fiel de las bases de datos transaccionales/históricas/logs o bien también puede funcionar como único lugar de almacenamiento de los datos *(esto se encuentra en relevamiento. Resta definir si, para algunos productos, esta capa reemplaza las bases históricas)*
* El siguiente stage de la data es una capa de Staging. En esta etapa los datos comienzan su procesamiento a través de la herramienta de ETL seleccionada, este stage es usado para que los datos sean estandarizados/limpiados, es decir que las transformaciones solamente impliquen cómputos básicos (ej; pesos a dólar), casteo de datos (ej; texto a fecha, o texto a número, etc) y renombre de campos (esto si hay campos que en las bases de datos tienen nombres distintos a como los conoce el producto).
* La etapa intermedia de los datos ocurre en el stage de Common, en esta capa se realiza la simplificación de estructura, es decir que muchas tablas pasen a estar consolidadas en pocas para facilitar las consultas y donde se puede realizar un cambio en la granularidad del dato. Es decir, en esta etapa se realiza el modelado lógico, en base a los distintos agrupamientos o apertura de conceptos que el negocio requiere.
* Por último, la capa de Business Focused, o Marts, son tablas construídas para el negocio y las cuales incluye toda la lógica de este, por ejemplo, una tabla de facturación donde está consolidada todo el consumo de los clientes, una tabla donde se encuentran todos los clientes y sus productos de manera consolidada. El objetivo de esta capa es que, tanto para el negocio como para la reportería, las consultas sean lo más sencilla posible, con tablas que contengan las diferentes métricas que se consultan día a día. El escenario ideal es que en esta capa se puedan resolver la mayoría de las preguntas de los stakeholders, de las áreas comerciales y del management. Es decir, una capa que dote a Coelsa de una visión macro y simplificada.

**Estrategia de Depuración del DataLake.**

La depuración del datalakehouse contiene 2 aristas principales:

1. **Reducción o eliminación de bases históricas.** En este punto, una vez realizada la ingesta inicial de las bases históricas en la capa Raw del datalakehouse, se debería empezar a eliminar los registros en dichas bases hasta que se pueda realizar el apagado de la misma. Para esto se recomienda realizar un snapshot final de la base de datos previo al apagado definitivo del mismo. *(este punto se encuentra en etapa de relevamiento con cada producto).*
2. **Ciclo de vida del dato:** Es la parte central de la estrategia de depuración de un datalakehouse. Dado que los datos se encuentran almacenados en un servicio de storage (Azure blob storage), el costo del servicio tiene asociado un costo exponencial, ya que los datos siempre van a tender a crecer con el tiempo. Es por esto por lo que se requiere la definición del ciclo de vida. Azure Blob Storage ofrece distintos tiers de almacenamiento[[5]](#footnote-6) en las cuales los costos se reducen si los datos no requieren ser consultados con frecuencia. Para esto es necesario definir la ventana de tiempo para cada producto. De esta manera, los datos almacenados se estarían transfiriendo entre los tiers, optimizando los costos de almacenamiento:

* **Hot Tier:** nivel en línea optimizado para almacenar datos a los que se accede o se modifican con frecuencia. El nivel de acceso frecuente tiene los costos de almacenamiento más altos, pero los costos de acceso más bajos.
* **Cool Tier:** nivel en línea optimizado para almacenar datos a los que se accede o se modifican con poca frecuencia. Los datos del nivel de acceso esporádico se deben almacenar durante un mínimo de 30 días. El nivel de acceso esporádico tiene menores costos de almacenamiento y mayores costos de acceso en comparación con el nivel de acceso frecuente.
* **Cold Tier:** nivel en línea optimizado para almacenar datos a los que se accede o se modifican con poca frecuencia. Los datos del nivel de acceso esporádico se deben almacenar durante un mínimo de 90 días. El nivel de acceso esporádico tiene menores costes de almacenamiento y mayores costes de acceso en comparación con el nivel de acceso esporádico.
* **Archieve Tier:** nivel sin conexión optimizado para almacenar datos a los que se accede muy pocas veces y que tienen requisitos de latencia flexibles, del orden de horas. Los datos del nivel de acceso de archivo se deben almacenar durante un mínimo de 180 días.

*(este punto se encuentra en etapa de relevamiento con cada producto y con el equipo de arquitectura. Coelsa ya cuenta con el concepto de datos calientes, frios, etc).*

1. https://www.qlik.com/us/data-lake/data-lakehouse [↑](#footnote-ref-2)
2. https://www.databricks.com/glossary/lambda-architecture [↑](#footnote-ref-3)
3. https://azure.microsoft.com/en-us/products/data-factory#overview [↑](#footnote-ref-4)
4. https://docs.delta.io/latest/delta-intro.html [↑](#footnote-ref-5)
5. https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview?tabs=azure-portal [↑](#footnote-ref-6)