

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“DISEÑO E IMPLEMENTACIÓN DE UN PROCESO ETL
SIMPLIFICADO CON VALIDACIÓN DE CALIDAD DE
DATOS PARA LA GENERACIÓN DE REPORTE DE
CAPACITACIÓN PERSONALIZADOS EN OTIC SOFOFA”

TOMÁS BENJAMÍN GUTTMAN POGORZELSKI

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: José Luis Martí Lara
Profesor Correferente: Ricardo Salas Letelier

Junio - 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

Diseño e implementación de un proceso ETL simplificado con validación de calidad de datos para la generación de reportes de

Título del trabajo: capacitación personalizados en OTIC SOFOFA

Nombre del candidato(a): Tomás Benjamín Guttman Pogorzelski

Carrera / Grado: Ingeniería Civil en Informática

Campus: San Joaquín

Departamento: Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, José Luis Martí Lara, 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: 25 - 05 - 2026

Firma:

Estudiante o Candidato(a):

Fecha: 21/05/2026

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.

DEDICATORIA

Quiero comenzar dedicando este trabajo a mi mismo, porque es el reflejo no solo de lo realizado en este proyecto, si no de mi resiliencia que tuve para salir de los años más difíciles de mi vida, con cuadros depresivos y *burnouts*, teniendo que enviar carta para solicitar excepción al reglamento por echarme 3 veces el mismo ramo, por querer dejar la carrera y atreverme a intentarlo una última vez y sacarlo adelante. Lo que se refleja de mi persona y mi historia de vida en este trabajo es muchísimo. También se lo dedico a mi familia, a mi madre que toda la vida ha cuidado y sigue cuidando de mi y ha querido lo mejor para mi; a mi hermana menor que es una luz en mi vida y quien siempre se ha preocupado de manera silenciosa por mi bienestar; a mi tata, que me contagió el gusto por la tecnología y quien siempre me apoyó en los deportes que quisiese practicar, y a mi oma, que siempre me escuchó y supo escuchar, que en paz descansen.

AGRADECIMIENTOS

De igual manera que en la dedicatoria de este trabajo, quiero iniciar agradeciéndome a mí por todo el esfuerzo, llanto, crisis, rabias, frustraciones y festejos que hice. Es justo en este momento cuando me doy cuenta que no fue en vano. Agradecer al profe y jefe de carrera José Luis Martí por mostrar una real preocupación por sus estudiantes, querer motivarme a terminar mi carrera y comprometerse a apoyarme en este desafío. Continúo dando gracias a todas las personas que fueron apoyos vitales en mi vida durante este periodo. A mi madre y a mi hermana por el apoyo y la confianza que otorgan en mi persona y mis capacidades. A la Nacha, *my bestie*, quien en muchas ocasiones la sufrió más que yo incluso, que me apoyó para poder avanzar en el escrito en esas sesiones de *coworking* en un café, biblioteca o donde fuera. Al Shaggy, al JP, al Beto, a la Monse, al Kike, por nombrar a algunas de las personas que me dieron aliento en este proceso, que siempre preguntaron por cómo iba, que estaban preocupados por mi estabilidad emocional y salud mental. A todas estas personas, muchas gracias.

RESUMEN

Resumen— Una empresa cliente de OTIC SOFOFA necesita un reporte personalizado que consolide datos de cursos, costos y participantes de su *holding*, incluyendo un resumen de cuentas de financiamiento. Al no disponer de este reporte, se propone el diseño e implementación de un proceso ETL simplificado en *Python* con *Pandas*, cuya única entrada es un archivo *Excel* y cuyo producto final es otro archivo *Excel* con cuatro hojas: *Participantes*, *Precontrato*, *Competencias Laborales* y *Cuentas*. El desarrollo incluye el modelado mediante diagramas de mapeo basados en UML, la implementación de reglas de transformación y un proceso de validación que mide el índice de calidad (IC) por hoja. Los resultados con cuatro casos de prueba reales muestran que dos alcanzaron un IC del 100 % en todas las hojas, mientras que los otros dos presentaron errores de origen que afectaron los IC de *Participantes* y *Cuentas*, evidenciando la capacidad del proceso para detectar inconsistencias. La solución automatiza la generación de un reporte inexistente, aportando validación y trazabilidad sobre la calidad de los datos.

Palabras Clave— ETL; calidad de datos; ingeniería de datos; capacitación laboral en Chile.

ABSTRACT

Abstract— A client company of OTIC SOFOFA needs a customized report that consolidates data on courses, costs, and participants from its holding company, including a summary of financing accounts. As this report is not available, the design and implementation of a simplified ETL process in Python with Pandas is proposed, whose only input is an Excel file and whose final product is another Excel file with four sheets: Participants, Pre-contract, Job Skills, and Accounts. The development includes modeling using UML-based mapping diagrams, the implementation of transformation rules, and a validation process that measures the quality index (QI) per sheet. The results with four real test cases show that two achieved a QI of 100 % on all sheets, while the other two had source errors that affected the QI of Participants and Accounts, demonstrating the process's ability to detect inconsistencies. The solution automates the generation of a non-existent report, providing validation and traceability on data quality.

Keywords— ETL; data quality; data engineering; job training in Chile.

GLOSARIO

- **ADF:** Del inglés, *Azure Data Factory*, que en español significa *fábrica de datos de Azure*.
- **API:** Del inglés, *Application Programming Interface*, que en español significa *interfaz de programación de aplicaciones*.
- **AWS:** Del inglés, *Amazon Web Services*, que en español significa *servicios web de Amazon*.
- **CDC:** Del inglés, *Changed Data Capture*, que en español significa *captura de datos modificados*.
- **CGS:** *ConsolidadoGestorSchema*.
- **COMLS:** *CompetenciasLaboralesSchema*.
- **CUENS:** *CuentasSchema*.
- **DBT:** Del inglés, *Data Build Tool*, que en español significa *Herramienta de compilación de datos*.
- **DL:** Del inglés, *Data Lake*, que en español significa *lago de datos*.
- **DSA:** Del inglés, *Data Staging Area*, que en español significa *área de almacenamiento temporal de datos*.
- **DW:** del inglés, *Data Warehouse*, que en español significa *almacén de datos*.
- **EMPC:** Elementos de modelado de primera clase.
- **ETL:** Del inglés, *Extract, Transform and Load*, que en español significa *extraer, transformar y cargar*.
- **FT:** Franquicia Tributaria.
- **GCD:** Del inglés, *Google Cloud Dataflow*, que en español significa *flujo de datos en la nube de Google*.
- **IaC:** Del inglés, *Infrastructure as Code*, que en español significa *infraestructura como código*.
- **IC:** Índice de calidad.
- **OC:** Orden de compra.
- **OTEC:** Organismo Técnico de Capacitación.
- **OTIC:** Organismo Técnico Intermedio para Capacitación.

- **PARTS:** *ParticipantesScema*.
- **PRECS:** *PrecontratoSchema*.
- **PK:** Del inglés, *Primary Key*, que en español significa *clave primaria*.
- **RCS:** *ReporteConsolidadoSchema*.
- **RMOCS:** *ReporteMasivoOCSchema*.
- **SENCE:** Servicio Nacional de Capacitación y Empleo.
- **SSIS:** Del inglés, *SQL Server Integration Services*, que en español significa *Servicios de integración para SQL Server*.
- **SOFOFA:** Sociedad de Fomento Fabril.
- **SQL:** Del inglés, *Structured Query Language*, que en español significa *Lenguaje estructurado de consulta*.
- **UML:** Del inglés, *Unified Modeling Language*, que en español significa *lenguaje de modelado unificado*.
- **UTM:** Unidad Tributaria Mensual.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	X
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Sistema nacional de capacitación laboral en Chile	3
1.2 OTIC SOFOFA	3
1.3 El problema	4
1.4 Objetivos y alcance	5
CAPÍTULO 2: MARCO CONCEPTUAL	7
2.1 Procesos ETL	7
2.2 Modelado de procesos ETL	9
2.2.1 Modelado basado en expresiones de mapeo	9
2.2.2 Modelado conceptual	9
2.2.3 Modelado basado en ambiente UML	11
2.3 Calidad de datos	14
2.4 Pruebas en procesos ETL	16
2.4.1 Pruebas de completitud de datos	16
2.4.2 Pruebas de precisión de datos	16
2.4.3 Pruebas de consistencia de datos	17
2.4.4 Pruebas de integridad de datos	17
2.4.5 Pruebas de transformación de datos	18
2.5 Herramientas de desarrollo de procesos ETL	18
2.5.1 <i>Structured Query Language</i> (SQL)	18
2.5.2 Lenguajes de <i>scripting</i>	18
2.5.3 Herramientas de ETL tradicionales	19
2.5.4 Herramientas de ETL modernas	19
2.5.5 Plataformas en la nube	20
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	22
3.1 Diferenciación con el marco conceptual	22
3.2 Diseño del proceso ETL simplificado	23
3.2.1 Diagrama de mapeo de datos de nivel de bases de datos	23
3.2.2 Diagrama de mapeo de datos de nivel de flujo de datos	25

3.2.3	Mapeo de nivel de tablas y atributos	25
3.3	Implementación del proceso ETL simplificado	48
3.3.1	Componentes	49
3.3.2	Módulos	49
3.3.3	Funciones	52
3.3.4	Orquestación	57
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		58
4.1	Reglas de calidad de datos	58
4.2	Descripción de los casos de prueba	59
4.3	Resultados	60
CONCLUSIONES		61
REFERENCIAS BIBLIOGRÁFICAS		65

ÍNDICE DE FIGURAS

1	Diagrama del flujo general de un proceso ETL	7
2	Metamodelo para las entidades lógicas del modelado conceptual	10
3	Niveles de mapeo del diagrama de mapeo de datos del modelado basado en UML	14
4	Diagrama de mapeo de datos de nivel de bases de datos del proceso ETL simplificado	23
5	Modelo de datos relacional del reporte ConsolidadoGestor	23
6	Estructura del paquete ReporteConsolidadoSchema	24
7	Modelo de datos relacional del esquema ParticipantesSchema	24
8	Modelo de datos relacional del esquema PrecontratoSchema	24
9	Modelo de datos relacional del esquema CompetenciasLaborales	25
10	Modelo de datos relacional del esquema CuentasScehema	26
11	Diagrama de mapeo de datos de nivel de flujo de datos del proceso ETL simplificado	27
12	Diagrama de mapeo de datos de nivel de tabla de la clase RazonSocial del esquema ParticipantesSchema	27
13	Diagrama de mapeo de datos de nivel de tabla de la clase SociedadReparto del esquema ParticipantesSchema	28
14	Diagrama de mapeo de datos de nivel de tabla de la clase OTEC del esquema ParticipantesSchema	29
15	Diagrama de mapeo de datos de nivel de tabla de la clase InscripcionInterna del esquema ParticipantesSchema	29
16	Diagrama de mapeo de datos de nivel de tabla de la clase PlanCurso del esquema ParticipantesSchema	31
17	Diagrama de mapeo de datos de nivel de tabla de la clase Curso del esquema ParticipantesSchema	32
18	Diagrama de mapeo de datos de nivel de tabla de la clase Participante del esquema ParticipantesSchema	33
19	Diagrama de mapeo de datos de nivel de tabla de la clase PedidoVenta del esquema PrecontratoSchema	35
20	Diagrama de mapeo de datos de nivel de tabla de la clase Participante del esquema PrecontratoSchema	37
21	Diagrama de mapeo de datos de nivel de tabla de la clase PedidoVenta del esquema CompetenciasLaboralesSchema	38
22	Diagrama de mapeo de datos de nivel de tabla de la clase Participante del esquema CompetenciasLaboralesSchema	39
23	Diagrama de mapeo de datos de nivel de tabla de la clase EstadoCuentaAnual del esquema CuentasSchema en la sección Excedentes	40
24	Diagrama de mapeo de datos de nivel de tabla de la clase EstadoCuentaAnual del esquema CuentasSchema en la sección AporteAnual	43

25	Diagrama de mapeo de datos de nivel de tabla de la clase EstadoCuentaAnual del esquema CuentasSchema en la sección Unporciento	44
26	Diagrama de mapeo de datos de nivel de tabla de la clase EstadoCuentaAnual del esquema CuentasSchema en la sección CostoEmpresa	46
27	Diagrama de mapeo de datos de nivel de tabla de la clase EstadoCuentaTotalAnual del esquema CuentasSchema	48

ÍNDICE DE TABLAS

1	Niveles de un diagrama de mapeo de datos	13
2	Clasificación de problemas típicos de calidad de datos en procesos ETL	15
3	Reglas de validación de consistencia de datos del DataFrame de participantes	59
4	Descripción de los casos de prueba	60
5	Resultados del ciclo de pruebas único	60

INTRODUCCIÓN

El sistema nacional de capacitación laboral chileno, regulado por el Servicio Nacional de Capacitación y Empleo (SENCE), contempla entre sus instrumentos la Franquicia Tributaria, un mecanismo que permite a las empresas financiar la formación de sus trabajadores descontando hasta el 1 % de su planilla anual de sueldos del impuesto a la renta. En este ecosistema operan los Organismos Técnicos Intermedios para Capacitación (OTIC), entidades que actúan como nexo entre las empresas adheridas, y los Organismos Técnicos de Capacitación (OTEC) que ejecutan las capacitaciones. El OTIC SOFOFA, organización en la que se desarrolla este trabajo de título, es uno de los actores más relevantes del sistema, y dispone de la plataforma digital Gestor Capital Humano para que sus empresas cliente administren sus procesos de formación y puedan consultar reportes consolidados de los datos de los cursos, sus costos y participantes.

Sin embargo, dada la diversidad y el volumen de empresas que conforman los grandes *holdings*, suelen surgir requerimientos específicos por parte de las empresas afiliadas al OTIC SOFOFA que los reportes estándar de la plataforma no logran satisfacer, exigiendo la construcción de informes personalizados bajo reglas de negocio específicas para cada empresa. El presente trabajo de título surge a partir de la necesidad de una empresa cliente de OTIC SOFOFA de un reporte de capacitación personalizado que consolide los datos sobre los cursos, sus costos y participantes, junto con un resumen del estado de cuentas de financiamiento de las empresas de su *holding*. Si bien la plataforma Gestor ofrece un reporte consolidado de cursos y participantes, este no responde al formato ni a las reglas de negocio requeridas por el cliente, y tampoco incluye el resumen de cuentas solicitado. Construir este reporte manualmente es inviable por el volumen de datos que se deben manipular y al tiempo disponible para entregar el reporte al cliente.

Para abordar este problema, se propone el diseño e implementación de un proceso ETL simplificado que automatice la generación del reporte personalizado, tomando como única fuente de datos el reporte consolidado de Gestor en formato Excel y produciendo como salida un archivo *Excel* con cuatro hojas, *Participantes*, *Precontrato*, *Competencias Laborales* y *Cuentas*, que reflejan el reporte personalizado solicitado. La calificación de “simplificado” responde a que, a diferencia de un proceso ETL tradicional, la solución no contempla la integración con un sistema de almacenamiento centralizado como un *Data Warehouse* o un *Data Lake*, ni la conexión con APIs externas, si no que opera de manera local y bajo demanda del usuario, decisión fundamentada en los plazos acotados del proyecto y en los requerimientos concretos del cliente.

El desarrollo de la solución se articula en torno a tres grandes ejes. En primer lugar, se realiza el análisis y modelado de los datos involucrados, utilizando diagramas de mapeo de datos del enfoque de modelado de procesos ETL basado en ambiente UML, propuesto por [Luján-Mora et al., 2004], lo que permite documentar con precisión las reglas de transformación requeridas. En segundo lugar, la implementación del proceso ETL simplificado se

lleva a cabo en *Python*, utilizando la librería *Pandas*, adoptando una arquitectura modular que organiza el código en componentes de extracción, transformación y carga, lo que facilita la comprensión, el mantenimiento y la escalabilidad de la solución. Finalmente, se integra un proceso de validación de calidad de datos como componente estructural del ETL, definiendo reglas de calidad específicas para cada conjunto de datos de salida y calculando un índice de calidad que permite cuantificar el porcentaje de registros libres de errores, además de identificar y reportar aquellos con problemas para su corrección oportuna.

La estructura de este documento es la siguiente. El Capítulo 1 define el problema, describiendo el contexto del sistema nacional de capacitación laboral y la organización OTIC SOFOFA, junto con los objetivos y el alcance del proyecto. El Capítulo 2 presenta el marco conceptual, reuniendo las definiciones, estrategias de modelado, dimensiones de calidad de datos, técnicas de prueba y herramientas de desarrollo relevantes de procesos ETL. El Capítulo 3 describe en detalle la propuesta de solución, explicando las decisiones de diseño, los diagramas de mapeo de datos en sus distintos niveles y la implementación del proceso ETL simplificado, incluyendo sus componentes, módulos, funciones y orquestación. El Capítulo 4 presenta la validación de la solución, describiendo las reglas de calidad establecidas, los casos de prueba utilizados y los resultados obtenidos. Finalmente, se exponen las conclusiones del trabajo, analizando el cumplimiento de los objetivos, las contribuciones realizadas, las limitaciones de la solución, las recomendaciones para trabajos futuros y reflexiones finales del autor.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. Sistema nacional de capacitación laboral en Chile

En Chile, el sistema de capacitación y empleo nacional busca promover el desarrollo de las competencias laborales del capital humano del país, contribuyendo en la mejoría de la empleabilidad y la productividad [Chile, 1997]. Una iniciativa a destacar de este sistema es la Franquicia Tributaria (FT), un beneficio para empresas contribuyentes de primera categoría que les permite invertir en la formación de sus trabajadores y trabajadoras, e incluso de personas que no se encuentren vinculadas a la empresa. Para el financiamiento de la FT, las empresas pueden “descontar del impuesto a la renta hasta el 1% de su planilla anual de sueldos por gastos de capacitación autorizados por el Servicio Nacional de Capacitación y Empleo (SENCE)” [ChileAtiende, 2024]. En el sistema de capacitación y empleo, lo que incluye la FT, el SENCE es un organismo técnico del estado que está encargado de, a grandes rasgos, difundir, administrar, gestionar, regular y fiscalizar los programas, cursos y acciones de capacitación [Chile, 1997].

Las empresas que opten por la Franquicia Tributaria podrán adherirse a un Organismo Técnico Intermedio para Capacitación (OTIC), al cual aportan el financiamiento correspondiente para el desarrollo de sus capacitaciones. Los OTIC tienen el objetivo de apoyar a sus empresas adheridas a través de la promoción, organización y supervisión de los programas de capacitación, además de asistencia técnica para el desarrollo de los recursos humanos y la administración de los aportes recibidos. Estas organizaciones no imparten capacitación, sino que son el nexo entre sus empresas adheridas y los Organismos Técnicos de Capacitación (OTEC), quienes si ejecutan los cursos [Chile, 1997].

1.2. OTIC SOFOFA

Uno de los Organismos Técnicos Intermedios para Capacitación más importantes es el OTIC SOFOFA. Con más de 45 años de trayectoria, y siendo un actor relevante en el fortalecimiento del capital humano en Chile, OTIC SOFOFA ha trabajado en consolidarse “como expertos y referentes en la formación de capital humano en Chile, fomentando la capacitación laboral y la educación técnica como herramientas estratégicas para el crecimiento y desarrollo de la actividad industrial” del país [SOFOFA, 2025a, OTIC SOFOFA, 2025]. OTIC SOFOFA, a través del trabajo colaborativo con empresas, especialistas y aliados tanto públicos como privados, ha podido llevar a cabo importantes proyectos como los Observatorios Laborales, Talento Digital para Chile, Relink y ABRA [SOFOFA, 2025a].

OTIC SOFOFA es parte del ecosistema de la Sociedad de Fomento Fabril (SOFOFA), una federación gremial, con más de 140 años de historia en el mundo empresarial privado en Chile, que agrupa a miles de compañías y decenas de asociaciones sectoriales y gremios regionales, que trabaja con el fin de permitir el desarrollo sostenible de la actividad empresarial [SOFOFA, 2025c]. Su ecosistema es una red colaborativa de empresas, instituciones y personas que tienen el propósito común de construir un país más productivo innovador, inclusivo y sostenible [SOFOFA, 2025b].

Enfocándose en la mejora continua de los servicios que OTIC SOFOFA entrega a sus empresas adheridas (o empresas cliente), en el año 2021 lanzan la plataforma Gestor Capital Humano (o simplemente Gestor), la cual “responde a la necesidad de administración de los procesos de formación dentro de una empresa, permitiendo hacer uso del beneficio de la Franquicia Tributaria para capacitación” [OTIC SOFOFA y LICEOS SOFOFA, 2021]. Con Gestor, las empresas cliente de OTIC SOFOFA pueden “autogestionar sus acciones de formación, desde la inscripción de cursos, hasta conocer toda la información relevante de los procesos de formación que faciliten a la organización la toma de decisiones” [OTIC SOFOFA y LICEOS SOFOFA, 2021].

Entre la gran variedad de tareas que las empresas clientes pueden realizar en Gestor, destacan el proceso inscripción de acciones de capacitación donde las empresas cliente agregan toda la información relevante para una administración eficiente de sus recursos para nuevos cursos; el acceso a reportes de datos consolidados, tanto de información interna o de mercado, asociados a las acciones de capacitación inscritas; y la administración de la planilla de colaboradores interna y de empresas que no son parte del *holding* de la empresa cliente, pero que son aliados en la gestión [OTIC SOFOFA y LICEOS SOFOFA, 2021]. Sin embargo, las necesidades particulares de sus empresas cliente para la optimización de la gestión de sus acciones de capacitación, han impulsado a que el OTIC SOFOFA, caracterizado por el trabajo colaborativo, implemente distintas soluciones tecnológicas para el apoyo de sus empresas clientes como la integración de sistemas entre ambas entidades o la construcción de reportes personalizados.

1.3. El problema

Una empresa cliente en específico, en adelante “el Cliente”, solicita al OTIC SOFOFA un reporte de capacitación personalizado que consolide los datos relacionados a la ejecución de los cursos, sus costos y participantes, de las empresas pertenecientes al *holding* del Cliente. Además, debe incluir un resumen de cuentas que, en función de los datos consolidados de los cursos y participantes, refleje el estado de cuenta de las empresas en diferentes conceptos asociados a costos de capacitación. El entregable es un archivo *Excel* que contiene las hojas con los datos solicitados.

Si bien, OTIC SOFOFA dispone de un reporte consolidado en Gestor, este no responde al formato que el Cliente necesita y ciertas estructuras de datos no cumplen las mismas reglas de negocio, además de no tener un resumen de costos. Por consecuencia, y para responder al

requerimiento del cliente, se deben hacer transformaciones en los datos del consolidado de Gestor del Cliente. Realizar esta tarea manipulando manualmente los datos implica demasiadas *horas persona* y una gran posibilidad de ocurrencia de errores que afecten la calidad de los datos y la credibilidad de los reportes que se construyan. Por lo tanto, es fundamental que se diseñe y desarrolle un proceso automatizado para la generación de los reportes personalizados, permitiendo verificar la calidad e integridad de los datos, asegurando que el reporte es fidedigno a la realidad de cursos del Cliente.

1.4. Objetivos y alcance

A continuación se presenta el objetivo general de este trabajo de título para solucionar el problema definido, los objetivos específicos que permitan medir el cumplimiento del objetivo general y el alcance de la memoria, especificando lo que está considerado y lo que no.

Objetivo general

Diseñar e implementar un proceso ETL simplificado para la confección de un reporte de capacitación personalizado para una empresa cliente del OTIC SOFOFA, incorporando pruebas de validación de calidad de datos que permitan verificar que todos los datos del reporte personalizado cumplen con las reglas de transformación definidas, midiendo el porcentaje de registros sin errores de calidad de datos que debe ser igual a 100 %. Esto debe ser logrado entre diciembre de 2025 y marzo de 2026.

Objetivos específicos

1. Analizar y documentar las fuentes de datos y el reporte de salida para la generación del reporte personalizado, identificando la estructura de los datos, sus relaciones y las reglas de transformación asociadas, mediante una estrategia de modelado de procesos ETL adecuada.
2. Diseñar una arquitectura del proceso ETL simplificado para definir el flujo de los datos en el proceso, cómo se ordenan las etapas del proceso y sus tareas, y los lineamientos de implementación.
3. Implementar el proceso ETL simplificado, utilizando las herramientas de desarrollo adecuadas al problema presentado, el requerimiento de la empresa cliente y los plazos establecidos, para la confección del reporte de capacitación personalizado.
4. Validar la solución mediante la implementación y ejecución de un proceso de pruebas de calidad de datos que verifique que el reporte de capacitación personalizado cumpla

con todas las reglas de transformación establecidas, estableciendo un indicador que mida la cantidad de registros sin errores de calidad de datos. Esto permite verificar que el reporte generado es correcto en función a las reglas definidas y, además, identificar errores en el reporte para gestionar su corrección.

Alcance

El presente trabajo se limita al diseño, implementación y validación de un proceso ETL simplificado que considera como única fuente de origen un archivo *Excel* descargado de la plataforma Gestor, y como producto final de salida un reporte personalizado en *Excel* que contiene los datos consolidados sobre los cursos, sus costos y participantes, de la empresa cliente, y un informe de cuentas de financiamiento de capacitación. Debido a los requerimientos y plazos del proyecto, queda fuera del alcance de esta memoria:

- La integración con APIs, ecosistemas en la nube como *Data Warehouse* o *Data Lake* y el desarrollo de una interfaz gráfica de usuario.
- El uso de herramientas de orquestación y ejecución automática de código para el proceso ETL. La solución propuesta se ejecuta de manera local bajo demanda de usuario.
- La adaptación del entregable final a otras empresas clientes del OTIC SOFOFA, centrándose exclusivamente en las necesidades de la empresa cliente solicitante.
- La corrección de errores de calidad en los datos de origen solo serán reportados al área correspondiente del OTIC SOFOFA para su correspondiente análisis.

CAPÍTULO 2

MARCO CONCEPTUAL

Antes de presentar la solución que se propone en este trabajo de título para abordar el problema presentado en la [Sección 1.3](#), es necesario describir los los conceptos técnicos, metodologías y herramientas en los que se fundamenta esta solución. Por lo anterior, en este capítulo se definen qué son los procesos ETL; las principales estrategias de modelado de ETL; la calidad de datos, sus dimensiones y principales problemas de calidad de datos en ETL; las pruebas en ETL; y las herramientas principales para el desarrollo de procesos ETL.

2.1. Procesos ETL

Los procesos de ETL (del inglés, *Extract-Transform-Load*) consisten en la extracción de datos organizacionales desde distintas fuentes, transformarlos y consolidarlos según los requerimientos del negocio, y cargarlos en un sistema de almacenamiento centralizado [[Deb Nath et al., 2021](#), [Sreemathy et al., 2020](#)], como puede ser un *Data Warehouse* (DW) o un *Data Lake* (DL), donde los usuarios finales puedan consultar los datos y utilizarlos para análisis o reportería [[Kumaran, 2021](#)]. Estos procesos se dividen en tres etapas que, como su nombre lo indica, son: extracción (*extract*), transformación (*transform*) y carga (*load*). El flujo de un proceso general de ETL se muestra en la [Figura 1](#) y es detallado en los próximos párrafos.

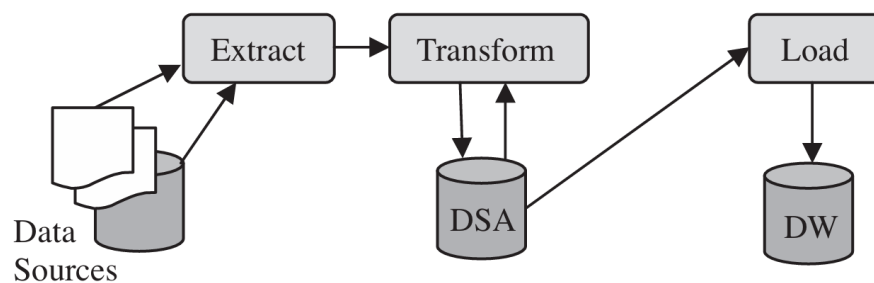


Figura 1: Diagrama del flujo general de un proceso ETL

Fuente: [[El-Sappagh et al., 2011](#)]

Extracción: En esta primera etapa se obtienen los datos desde sus fuentes de origen correspondientes, considerando en la implementación que cada fuente de datos podría diferenciarse de las otras en el formato de entrega de los datos, en el sistema de gestión de bases de datos utilizado o protocolos de comunicación. Existen dos tipos de extracción [[El-Sappagh et al., 2011](#), [Sreemathy et al., 2020](#)]:

- **Extracción completa:** se realiza solo la primera vez que se extraen los datos desde las fuentes de origen, obteniendo la totalidad de los datos para copiarlos en el sistema de almacenamiento de destino.
- **Extracción parcial o incremental:** se extraen solo los datos modificados o agregados en las fuentes de datos de origen desde la última extracción. Se realiza periódicamente según un ciclo de actualización, definido por las necesidades del negocio, o por manejo de eventos si las herramientas utilizadas así lo permiten.

Transformación: En esta fase los datos extraídos son convertidos al formato requerido por el sistema de almacenamiento de destino, según las reglas de transformación definidas por el negocio. Además, se debe validar que los datos cumplan con un cierto nivel de calidad para asegurar que los usuarios utilicen datos precisos, consistentes y completos (más detalles en la [Sección 2.3](#)). Los principales tipos de transformación son [[Mohammed, 2025](#), [Singu, 2022](#)]:

- **Limpieza de datos:** eliminar duplicados, corregir inconsistencias, comprobación de errores y gestión de incidencias de datos faltantes, entre otras. Este tipo de reglas de transformación son esenciales para asegurar la precisión de los datos.
- **Filtrado de datos:** filtra los datos en función de condiciones establecidas [[Sreemathy et al., 2020](#)].
- **Agregación y enriquecimiento:** integra nuevos datos que, en general, son campos calculados, como sumas, totales, porcentajes, etc., o información externa a la organización que no provienen de las fuentes de origen de las cuales se extraen los datos.
- **Mapeo de esquema y conversión de tipos de datos:** ordena las estructuras de datos con el esquema de destino y convierte los tipos de datos para asegurar la compatibilidad de los campos.
- **Implementación de las reglas del negocio:** aplica las reglas de transformación definidas por el negocio.
- **Anonimización de datos:** protege los datos sensibles mediante encriptación, tokenización o técnicas de ofuscación.

En la fase de transformación, los datos son almacenados en un *Data Staging Area* (DSA), en español, *área de almacenamiento temporal de datos* (ver [Figura 1](#)).

Carga: En esta tercera y última etapa, los datos extraídos y transformados son escritos en el sistema de almacenamiento de destino (DW o DL), donde los usuarios podrán consultar la información para reportes o análisis. Los tipos de carga son [[Mohammed, 2025](#), [Sreemathy et al., 2020](#)]:

- Carga completa: reemplaza todos los datos en el sistema de destino. Usualmente es utilizada para migraciones iniciales.
- Carga incremental: para mejorar la eficiencia, solo se cargan los nuevos datos o aquellos que han sido modificados desde la última carga realizada.
- *Changed Data Capture* (CDC): la *captura de datos modificados* monitorea los cambios en las fuentes de origen para ejecutar los procesos ETL, manteniendo actualizado el sistema de destino en tiempo real.

Otras actividades a realizar en esta etapa es el manejo de fallas en el proceso de carga, permitiendo deshacer los cambios realizados para mantener la integridad de los datos [Mohammed, 2025].

2.2. Modelado de procesos ETL

Para diseñar un proceso ETL, es primordial realizar el modelado de los datos que serán manipulados. En los siguientes apartados se describirán distintas estrategias de modelado discutidas en la literatura. Como base teórica de los siguientes apartados, se ha tomado de base lo propuesto en [El-Sappagh et al., 2011, Sharma y Jain, 2014].

2.2.1. Modelado basado en expresiones de mapeo

Este enfoque, presentado por [Rifaieh y Benharkat, 2002], mantiene expresiones de mapeo entre los datos de origen y los de destino, representadas como consultas a las fuentes de datos, implicando que las fuentes no solo los almacenen, si no que también sean el motor de las transformaciones. Se definen y documentan asignaciones sobre los atributos entre diferentes esquemas, permitiendo aplicar y rastrear las transformaciones. El modelado basado en expresiones presenta varias desventajas como la alta dificultad de mantenimiento, el aumento de la complejidad de evolución del sistema; y que realizar actualizaciones del proceso puede ser engorroso.

2.2.2. Modelado conceptual

Propuesto por [Vassiliadis et al., 2002], este enfoque introduce un marco para el modelado de actividades de procesos ETL compuesto por tres capas (ver Figura 2). La capa inferior, denominada *Schema Layer*, implica un escenario de ETL específico. Las entidades de esta capa son instancias de las clases: tipos de datos, conjuntos de objetos, tipos de funciones, actividades elementales y relaciones. La capa superior, denominada *Metamodel Layer*, implica

las clases mencionadas anteriormente. La interrelación entre las capas *Metamodel Layer* y *Schema Layer* se realizan mediante *relaciones de instanciación* (instanceOf). La capa intermedia, denominada *Template Layer*, contiene construcciones, que corresponden a *metaclases*, bastante personalizadas para casos habituales de procesos ETL. Entonces, estas clases representan especializaciones (subclases) de las clases genéricas definidas en la *Metamodel Layer*, representadas como relaciones *IsA*.

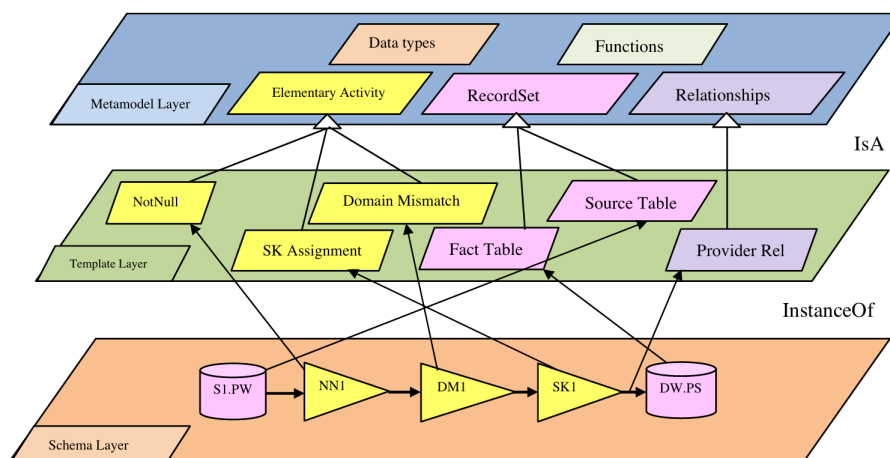


Figura 2: Metamodelo para las entidades lógicas del modelado conceptual

Fuente: [El-Sappagh et al., 2011]

A continuación se definen las entidades, o clases, del modelado conceptual [Vassiliadis et al., 2002, El-Sappagh et al., 2011]:

1. Tipos de datos: cada tipo es caracterizado por un nombre y un dominio, un conjunto numerable de valores, también llamados constantes.
2. Conjunto de objetos: caracterizado por su nombre y su esquema lógico, o estructura del conjunto de registros que, típicamente, suelen ser tablas relacionales o archivos de registros.
3. Tipos de funciones: compuesta por un nombre, una lista finita de tipos de datos de parámetros y un único tipo de dato de retorno. Una función es una instancia de un tipo de función.
4. Actividades elementales: abstracciones lógicas que representan partes o módulos completos de código. Se recomienda emplear la programación lógica con un lenguaje declarativo y, con esto, poder ignorar cualquier peculiaridad de un lenguaje de programación en concreto.
5. Relaciones: describen el seguimiento de los datos desde las fuentes de datos hasta el sistema de almacenamiento centralizado de destino.

2.2.3. Modelado basado en ambiente UML

Este enfoque, planteado por [Luján-Mora *et al.*, 2004], extiende el comportamiento de la notación UML (del inglés, *Unified Modeling Language*), que solo permite relacionar clases y no los atributos que las componen, habilitando la relación entre ellos, modelándolos como *elementos de modelado de primera clase* (EMPC). Se basa en el uso de paquetes UML para ampliar y reducir el diseño de un escenario. En el modelado basado en UML, el desarrollo de un sistema de almacenamiento centralizado, a través de procesos ETL, se estructura en cinco fases:

1. Fuente: define las fuentes de datos de origen.
2. Integración: establece la correspondencia entre las fuentes de datos y el sistema de almacenamiento.
3. Sistema de almacenamiento: determina la estructura del sistema de almacenamiento centralizado.
4. Personalización: precisa la correspondencia entre el almacén de datos y las estructuras de los clientes.
5. Cliente: define estructuras especiales que los usuarios utilizan para acceder al sistema de almacenamiento.

Cada fase puede analizarse en tres niveles o perspectivas:

1. Conceptual: especifica el sistema de almacenamiento centralizado desde un punto de vista conceptual.
2. Lógico: establece los aspectos lógicos del diseño del sistema de almacenamiento, como la definición de los procesos de ETL.
3. Físico: detalla los aspectos físicos del sistema de almacenamiento, como la persistencia de estructuras lógicas en diferentes discos o la configuración de servicios compatibles con sistemas de almacenamiento centralizado (DW y DL).

Atributos como elementos de modelado de primera clase (EMPC)

Los *elementos de modelado de primera clase* son el concepto fundamental de modelado sobre el cual se basa la construcción de modelos de ETL basados en ambiente UML. En los *diagramas de clases* tradicionales, hay dos tipos de elementos de modelado que se tratan como EMPC:

- Clases: representaciones abstractas de entidades del mundo real y son, naturalmente, el centro del modelado. Al ser EMPC, las clases son entidades independientes que también actúan como contenedores de atributos. Las relaciones entre clases se capturan por asociaciones.
- Asociaciones: como EMPC, las asociaciones se conocen como *clases de asociación* y conectarse a otras clases, como también contener atributos. Si bien, una clase de asociación se dibuja como una asociación y una clase, esto es un único elemento de modelado.

Para que los atributos de clases se comporten de la misma manera que los elementos anteriores, se representan como EMPC. Las clases y atributos se definen como se haría en notación UML clásica, pero cuando sea necesario tratar los atributos como EMPC, las clases se importan al *diagrama de atributos/clases*, donde los atributos se representan automáticamente como clases. Las clases que contienen atributos se definen como *clases contenedoras* y las clases que representen atributos son llamadas *clases de atributo*. Las clases de atributos son materializaciones de un nuevo estereotipo «attribute» introducido para representar a los atributos de una clase y deben satisfacer las siguientes restricciones para estar correctamente definidas:

- Convención de nomenclatura: el nombre de una clase de atributo se estructura del nombre de su clase contenedora seguido de un punto y el nombre del atributo.
- Sin funciones: una clase de atributo no puede contener ni atributos ni métodos.

La relación que existe entre una clase contenedora y sus clases de atributo correspondientes, son llamadas *relaciones de contención*, y se representan por una agregación compuesta, resaltadas con el estereotipo «contain». Así, se puede definir un diagrama de atributos/-clases como un diagrama de clases UML ampliado con clases de atributo y relaciones de contención.

Diagramas de mapeo de datos

Los conceptos introducidos permiten describir los *diagramas de mapeo de datos*, el núcleo de este enfoque de modelado, diseñados para el seguimiento del flujo de datos, con diversos grados de detalle, en un entorno de *Data Warehouse*. Complementan los diagramas de clases clásicos de UML, centrándose en las particularidades del flujo de datos y las interconexiones de las fuentes de datos involucradas.

El término *mapeo* refiere a la relación semántica que se establece entre dos elementos de diseño cuando estos comparten las mismas partes de información, posiblemente mediante una transformación o filtro. En un contexto de DW, este tipo de relaciones involucran tres

partes lógicas: el *origen*, entidad responsable de generar los datos que serán propagados (esquema, tabla o atributo); el *destino* que recibe los datos del origen; y el *mapping* intermedio que especifica la forma en que se mapean los datos desde el origen al destino, junto con cualquier transformación o filtrado. Para manejar la complejidad que un diagrama de mapeo de datos implica, se propone una estructura en capas de cuatro niveles detallada en la [Tabla 1](#).

Tabla 1: Niveles de un diagrama de mapeo de datos

Nivel	Descripción
Nivel de bases de datos (Nivel 0)	cada esquema de datos, tanto de las orígenes como del destino, se representan como paquetes de UML. Los mapeos entre diferentes esquemas son modelados por un único paquete de mapeo que encapsula todos los mapeos de bajo nivel.
Nivel de flujo de datos (Nivel 1)	describe la relación de datos entre las tablas de origen de los esquemas involucrados con respecto a los esquemas de destino en el sistema de almacenamiento.
Nivel de tablas (Nivel 2)	mientras que el nivel de flujo de datos describe las relaciones de datos entre fuentes y destinos usando un solo paquetes de mapeo, este nivel detalla todas las reglas de transformación y validaciones intermedias que tienen lugar durante este flujo.
Nivel de atributos (Nivel 3)	hace una ampliación del diagrama de nivel de tablas para capturar el mapeo de datos entre atributos del origen y del destino, junto con cualquier transformación o limpieza intermedia.

Fuente: [[Luján-Mora et al., 2004](#)]

En los niveles 0 y 1 se utilizan artefactos de la notificación tradicional UML. En el nivel 2 se introducen los estereotipos «input» para indicar una clase de asociación entre una clase de origen y un paquete de mapeo; «output» para denotar asociaciones entre paquetes de mapeo y clases de salida; y «intermediate» para definir clases intermedias que pueden o no almacenarse en el sistema de almacenamiento de destino y sirven para guardar los datos de los pasos intermedios de un proceso de transformación, ayudando a entender el proceso de mejor manera. A las clases se les agrega la leyenda (From <table>) para indicar a qué esquema pertenecen y los paquetes de mapeo representan las reglas de transformación del proceso y contienen diagramas de atributos/clases.

En la [Figura 3](#) se muestra un diagrama de mapeo de datos simplificado de ejemplo. En el Nivel 0, el diagrama de mapeo se compone de dos paquetes que abstraen el esquema de la fuente de datos de origen (SCS) y del *Data Warehouse* de destino (DWCS), y un paquete de mapeo. En el nivel 1, el paquete de mapeo es ampliado en un diagrama de tres paquetes

de mapeo, uno para cada una de tres supuestas tablas de destino. El diagrama de mapeo del Nivel 2 describe el detalle de los paquetes de mapeos del nivel anterior. Notar que se modelaron los pasos de mapeo como paquetes y que, además, existe una tabla intermedia para contener la salidas de cada paso. El diagrama del Nivel 3 muestra la forma en que los atributos se asignan entre sí.

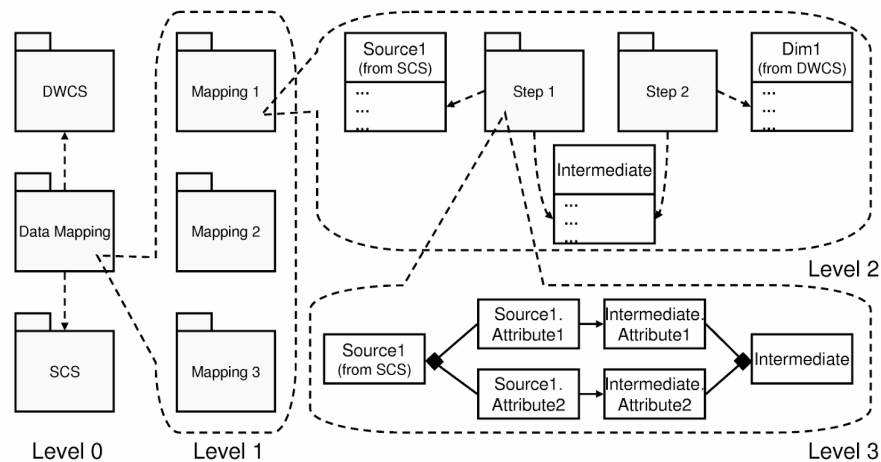


Figura 3: Niveles de mapeo del diagrama de mapeo de datos del modelado basado en UML
Fuente: [Luján-Mora et al., 2004]

2.3. Calidad de datos

En la definición de la fase de transformación de ETL, se establece que se debe validar la calidad de los datos, mencionando incluso algunas dimensiones (ver Sección 2.1). La *calidad de datos* refiere al grado en el que los datos satisfacen las necesidades y requisitos de una empresa u organización (negocio), y está determinada por factores como la cultura y tamaño organizacional, sus procesos y procedimientos, y cómo utilizan sus datos [Ridzuan y Zainon, 2024]. La calidad de datos puede ser definida, medida y administrada en términos de **dimensiones de calidad**. Entre las más comunes se tienen las siguientes [Ridzuan y Zainon, 2024]:

- **Precisión:** los datos se consideran precisos cuando reflejan el estado real de la información, sin incertidumbres, correspondiéndose con el mundo real. Cuando se tiene un conjunto acotado de valores posibles, como “Masculino”, “Femenino” o “No binario” para el género de las personas, es fácil evaluar la precisión. En cambio, si no hay un valor de referencia, puede ser una tarea más difícil.
- **Consistencia:** refiere al incumplimiento de reglas semánticas definidas sobre un sistema y mide la equivalencia entre la información resultante y la de origen, verificando

que sean coincidentes en concepto, en dominio de valores y en formatos. La consistencia se puede clasificar en dos: la *consistencia estructural* que describe la estructura de los valores de los datos, y la *consistencia semántica* que explica la relación entre los campos del conjunto de datos.

- **Complejidad:** corresponde al grado en el que los datos describen un estado significativo del mundo real representado y, en general, se expresa como el porcentaje del volumen de datos sin procesar en relación al volumen de datos requerido. Por lo tanto, no solo refleja la tasa de valores perdidos, sino que también la medida en que todos los datos están disponibles y tienen un alcance suficiente para su contexto de uso [Souibgui et al., 2019].
- **Puntualidad:** consiste en dos dimensiones: la *vigencia* que mide la validez temporal en el sistema o en el mundo real, y la *volatilidad* que expresa la frecuencia de cambios en los datos.

En un proceso ETL es primordial validar la calidad de los datos, pues los formatos de los datos pueden diferir entre los diversos orígenes, y los formatos y estructuras pueden ser difíciles de interpretar o ser cambiantes en el tiempo [Souibgui et al., 2019]. En la [Tabla 2](#) se clasifican los problemas típicos de calidad de datos según etapas del proceso ETL.

Tabla 2: Clasificación de problemas típicos de calidad de datos en procesos ETL

Etapas	Problemas
Extracción	Falta de restricciones de integridad que definan la consistencia de los datos respecto a sus fuentes, por ejemplo, por clave primaria. Diseño de esquema deficiente, donde la definición es incompleta o errónea. Múltiples valores en un solo atributo (valores incrustados). Datos duplicados que pueden originarse por faltas ortográficas o diferentes formas de escribir un mismo nombre. Valores faltantes que pueden clasificarse en dos tipos: los datos que aparecen como nulos o vacíos (incompletitud directa) y los valores perdidos por operaciones de datos como una actualización [Yang et al., 2017].
Transformación	Diferentes tipos de datos entre el esquema de origen y el de destino. Conflicto de nombres, por ejemplo, género y sexo de las personas. Formatos inconsistentes, por ejemplo, fechas.
Carga	Mapear incorrectamente los datos de origen y destino. Mala implementación del versionamiento de los datos al ingresar nuevos datos o actualizar los datos actuales.

Fuente: [Souibgui et al., 2019]

2.4. Pruebas en procesos ETL

Para manejar los posibles problemas que se puedan presentar en un proceso ETL, es necesario realizar un proceso de pruebas que garantice la calidad de los datos en sus distintas dimensiones [Seenivasan, 2023]. Cada una de las tres etapas de un proceso ETL debe ser sometida a un proceso de pruebas propio [Seenivasan, 2023]. Las pruebas de extracción verifican que los datos se obtengan correctamente desde los orígenes, comprobando que no se pierdan datos y que sean consistentes con el origen. En la etapa de transformación, las pruebas validan que las transformaciones se apliquen sin errores, lo que significa revisar que los datos se han limpiado, estandarizado y enriquecido, según sea necesario, y que las transformaciones de negocio se han aplicado correctamente. Por último, las pruebas de carga consisten en comprobar que los datos se carguen exitosamente en el sistema de almacenamiento de destino, de forma precisa, completa y consistente con el esquema de destino. También se verifica que no existan errores de validación en los datos. En [Seenivasan, 2023] se proporciona una detallada lista de técnicas de pruebas de ETL. En las siguientes secciones se describirán las técnicas de interés de esta memoria.

2.4.1. Pruebas de completitud de datos

Las pruebas de completitud de datos tienen por objetivo garantizar que todos los datos necesarios se hayan extraído desde el origen y cargados en el destino. A continuación se proponen algunas técnicas de pruebas de completitud de datos.

- Recuento de datos de origen a destino: esta técnica consiste en comparar el número de registros en los orígenes y el número de registros cargados en el sistema de almacenamiento de destino. Si estos no son iguales, entonces se pudo haber perdido datos en el proceso.
- Muestreo de datos de origen: se selecciona aleatoriamente una muestra de datos del origen y se comprueba que se carguen correctamente en el destino.
- Validación de transformaciones de negocio: valida que las transformaciones de negocio se apliquen a los datos implicados correctamente.

2.4.2. Pruebas de precisión de datos

Las pruebas de precisión de datos garantizan que:

- Los datos extraídos del origen sean precisos, verificando que los valores de los datos sean correctos y que no existan errores en el tipo o formato de los datos.

- Los datos se hayan transformado y cargado en el destino con precisión.
- Los datos sean consistentes con los orígenes y que los campos calculados y agregaciones presenten los resultados esperados.

Algunas técnicas que se pueden emplear son:

- Pruebas basadas en reglas: consisten en probar que cada regla o transformación de negocio se aplique correctamente en los datos.
- Perfilado de datos: los datos son analizados con el objetivo de identificar valores atípicos o anómalos que produzcan problemas de calidad de datos para que sean corregidos.
- Pruebas de agregación: se comprueba que las agregaciones y cálculos, como sumas o medias, se realicen sin errores.

2.4.3. Pruebas de consistencia de datos

Las pruebas de consistencia de datos garantizan que los datos extraídos sean consistentes con los orígenes y con el destino. Las siguientes técnicas se pueden implementar para realizar este tipo de pruebas:

- Pruebas de integridad referencial: valida que las relaciones entre elementos de datos existentes en los orígenes estén también en el destino.
- Validación de datos históricos: verifica que los datos sean consistentes a lo largo del tiempo.
- Identificación de registros duplicados: se identifican y eliminan los registros duplicados que puedan existir tanto en los orígenes como en el destino.

2.4.4. Pruebas de integridad de datos

Las pruebas de integridad de datos buscan garantizar que los datos mantengan su integridad a lo largo de todo el proceso ETL, comprobando que no se pierdan, ni se dañen ni se alteren. A continuación se presentan algunas técnicas que se pueden usar:

- Validación del mapeo de datos: valida que los datos se mapeen debidamente desde los orígenes al destino.
- Validación de datos: verifica la consistencia de datos entre los orígenes y el destino.

2.4.5. Pruebas de transformación de datos

Las pruebas de transformación de datos se centra en comprobar que los datos se transformen de acuerdo a las reglas y requisitos de negocio. Algunas técnicas que se pueden aplicar son:

- Pruebas de entrada y salida: los datos de entrada y salida son comparados para verificar que los datos se hayan transformado correctamente.
- Validación de transformación de datos: valida que las transformaciones se apliquen sin errores.
- Pruebas de manejo de errores: comprueba que los errores son manejados correctamente durante la etapa de transformación.
- Pruebas de reglas de transformación: asegura que las reglas de transformación estén correctamente implementadas.

2.5. Herramientas de desarrollo de procesos ETL

Para implementar los procesos ETL modelados con algunas de las técnicas vistas en la [Sección 2.2](#), es esencial seleccionar una herramienta de desarrollo adecuada al contexto del caso de aplicación. En las siguientes secciones se describirán diferentes herramientas mencionadas en [[Guntupalli y Vamshi ch, 2025](#)].

2.5.1. *Structured Query Language (SQL)*

Las bases de datos relacionales son frecuentemente utilizadas para el almacenamiento de datos y el lenguaje más común para consultarlas es *Structured Query Language* (SQL, en español, lenguaje estructurado de consulta) [[Kanburoğlu y Tek, 2024](#)]. Si bien, las herramientas de datos y lenguajes de *scripting* disponibles en la actualidad son rápidos y fáciles de usar, SQL sigue siendo el más confiable para consultar y operar sobre bases de datos relacionales.

2.5.2. Lenguajes de *scripting*

Mientras que SQL destaca en operaciones declarativas de datos, los lenguajes de programación de *scripting* son importantes herramientas para la orquestación, lógica de transformación y automatización en procesos ETL. *Python* es el lenguaje por excelencia en la ingeniería de datos gracias a su simplicidad; su amplio ecosistema con librerías como *Pandas*, *PySpark*

y *SQLAlchemy*; y su fácil integración con herramientas de orquestación y plataformas en la nube. El lenguaje de programación *Scala* ofrece un excelente soporte para la computación distribuida, sobre todo en sistemas basados en *Spark*, y es una herramienta clave en *pipelines* de bajo rendimiento, con baja latencia y con gran dependencia de paralelismo.

2.5.3. Herramientas de ETL tradicionales

Las siguientes herramientas son utilizadas en procesos ETL tradicionales [Guntupalli y Vamshi ch, 2025]:

- *Informatica PowerCenter*: destaca por su compatibilidad con sistemas legados, su canalización basada en datos y sus funcionalidades de técnicas tradicionales de integración ETL de datos empresariales.
- *Talend*: garantiza flexibilidad y su interfaz gráfica de usuario acelera el diseño de procesos ETL.
- *Microsoft SQL Server Integration Services (SSIS)*: conjunto de servicios para la integración con *Microsoft SQL Server*, comúnmente utilizado por empresas que utilizan *software* de *Microsoft*. SSIS ofrece funcionalidades complejas de transformación y programación de flujos de datos.

El diseño gráfico y el registro de *logs* significativo son características relevantes de estas herramientas. Como desventaja se tienen que no suelen contar con características escalables o ágiles, inherentes de las plataformas de la nube.

2.5.4. Herramientas de ETL modernas

Entre las herramientas modernas para la implementación de procesos ETL que destacan en el mundo empresarial se encuentran [Guntupalli y Vamshi ch, 2025]:

- *Data Build Tool (DBT)*: permite desarrollar transformaciones SQL modulares, gestionar el control de versionamiento a través de *Git* e implementar plataformas de integración. Su compatibilidad con *RedShift*, *BigQuery* y *Snowflake*, convierte a DBT en una herramienta fundamental en sistemas analíticos modernos.
- *Apache NiFi* y *Airbyte*: son tecnologías visuales orientadas al flujo de datos que faciliten el consumo de estos y son particularmente útiles para aplicaciones de *streaming* y tiempo real.

- *Airflow* y *Perfect*: en pos de priorizar la orquestación del proceso sobre las transformaciones, estos sistemas permiten realizar programación de eventos, reintentos en caso de fallos y monitoreo para rastrear la actividad de los datos más demandados.

2.5.5. Plataformas en la nube

En la actualidad, los procesos de ETL y las plataformas en la nube están estrechamente relacionadas. A medida que las empresas migran a estos marcos de trabajo, cobra mayor relevancia tener la capacidad de construir, controlar y maximizar los *pipelines* ETL en sistemas basados en la nube rentables y escalables. A continuación se presentan las robustas herramientas que tienen las principales empresas proveedoras de servicios en la nube [Guntupalli y Vamshi ch, 2025].

- *AWS Glue*: herramienta ETL completamente gestionada que interactúa perfectamente con el entorno de *Amazon Web Services* (AWS). Aborda la inferencia automática de esquemas; la catalogación de datos, por medio de *Glue Data Catalog*, para captar los metadatos de los servicios *S3*, *Redshift* y *Athena*; la ejecución tareas sin servidor en formato de código de *Python* o *Scala*, por medio de los trabajos de *AWS Glue*, facilitando el desarrollo de transformaciones complejas; y la integración con los servicios *AWS Lake Formation*, *Step Functions* y *EventBridge* para la gestión clara de los procesos.
- *Azure Data Factory* (ADF): dispone opciones de integración de datos con y sin código, y facilita la transferencia de datos entre entornos locales y en la nube, junto con ser compatible con más de 90 conectores. Sus funcionalidades claves son la fijación en segundo plano del flujo de datos en *Spark* para una lógica de transformación visible; la perfecta integración con los servicios *Azure Functions*, *Blob Storage* y *Azure Synapse*; y otras funcionalidades destacables como la ramificación, lógica condicional, reintento en caso de ocurrencia de fallos y parametrización en la orquestación de *pipelines*.
- *Google Cloud Dataflow* (GCD): sus principales fortalezas son un marco de programación ETL coherente para lotes y para flujo. Otras características interesante son el escalado automático y equilibrio dinámico de la carga de trabajo para operaciones imprevistas y de grandes volúmenes de datos; y la fácil conexión con los servicios *Cloud Storage*, *Pub/Sub* y *BigQuery*.

Otra tecnología de las plataformas en la nube relevante en la era *DevOps* es la *Infrastructure as Code* (IaC), en español, infraestructura como código, la cual apoya la automatización de la instalación, configuración, implementación y despliegue de recursos en la nube consumidos por los procesos ETL. La IaC facilita el versionamiento, los despliegue escalables dentro de procesos de integración continua y despliegue continuo, y la coherencia entre los ambientes de desarrollo, pruebas y producción. Se destacan las siguientes herramientas [Guntupalli y Vamshi ch, 2025]:

- *Terraform*: aplicación de IaC de código abierto utilizada para la definición y aprovisionamiento de infraestructura en la nube entre varios proveedores (AWS, Azure y Google) mediante archivos de configuración declarativos. Permite crear funciones *lambda* de AWS, *pipelines* de ADF, trabajos de AWS *Glue*, entre otras características de servicios de los diferentes proveedores; y habilitar la reversión de cambios y el registro de auditoría por medio de *Git*. Ideal para empresas que operan con varios proveedores de plataformas en la nube.
- *AWS CloudFormation*: posibilita la definición y el aprovisionamiento de infraestructura mediante plantillas de IaC específicas de AWS y garantiza una gestión automática y segura de los recursos. Sus principales ventajas son los controles de pila y la detección de desviaciones que ayudan a evitar modificaciones accidentales; la IaC mejorada con *Python* o *TypeScript* mediante la integración con el kit de desarrollo de la nube de AWS; y la previsualización de cambios estructurales antes de su ejecución.

El propósito de utilizar alguna de estas herramientas es eliminar el aprovisionamiento manual de infraestructura, reducir los errores humanos y simplificar la colaboración entre desarrolladores, ingenieros de infraestructura y equipos de seguridad.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Para abordar el problema presentado en la [Sección 1.3](#), se propone como solución un proceso de ETL simplificado utilizando el lenguaje de programación *Python* y la librería *Pandas*. Este capítulo está estructurado con el fin de describir la solución de forma detallada para que el proceso pueda ser comprendido en profundidad. Para esto, se presentan las diferencias de implementación respecto a las definiciones del marco conceptual, las metodologías y herramientas elegidas; el diseño de la solución mediante diagramas de mapeo de datos; y, para finalizar, la implementación de la solución.

3.1. Diferenciación con el marco conceptual

Para el desarrollo de una solución adecuada al contexto de aplicación, se tomaron decisiones de diseño, basadas el marco conceptual, que tienen relación a cómo se estructura el proceso ETL, qué metodologías de modelado y qué herramientas se eligieron utilizar.

- *Proceso ETL simplificado*: por definición, los procesos ETL extraen datos de diversas fuentes de origen para transformarlos y consolidarlos en un sistema de almacenamiento centralizado de destino, *Data Warehouse* o *Data Lake* (véase la [Sección 2.1](#)). Sin embargo, la solución propuesta considera como entrada y salida reportes en formato *Excel* y, dado los plazos de desarrollo, no se considera una conexión a una API o integración en un ambiente de DW o DL para el producto mínimo viable. Por consiguiente, se propone un ETL simplificado que considera en la etapa de extracción solo un origen correspondiente a un archivo *Excel* y en la etapa de carga, en vez de considerar un sistema de almacenamiento como destino, considera la generación de un *Excel* que contienen distintas hojas con distintas categorías de datos.
- *Diseño y modelado*: se ha elegido diseñar y modelar el flujo de datos del proceso ETL simplificado, y sus transformaciones, con diagramas de mapeo de datos (véase la [Sección 2.2.3](#)). Esto se basa en la universalidad de la notación UML, lo que ayuda a nuevos desarrolladores, que se integren al proyecto en el futuro, entender el proceso rápidamente.
- *Herramientas de desarrollo*: se ha decidido implementar el ETL simplificado en *Python* con la librería *Pandas*, por la fácil extracción y carga de datos desde, y hacia, archivos *Excel*; el alto nivel de conocimiento alto del equipo desarrollador en estas herramientas; y la gran compatibilidad de integración con distintas herramientas para posibles futuras mejoras.

3.2. Diseño del proceso ETL simplificado

El proceso ETL simplificado será modelado mediante diagramas de mapeo de datos en sus distintos niveles definidos en la [Tabla 1](#).

3.2.1. Diagrama de mapeo de datos de nivel de bases de datos

En la [Figura 4](#), se muestra el diagrama de mapeo de datos de nivel de bases de datos del proceso ETL simplificado. En el lado izquierdo del diagrama se encuentra el paquete ConsolidadoGestorSchema (CGS) que encapsula el esquema del único origen correspondiente a un Excel llamado ConsolidadoGestor.

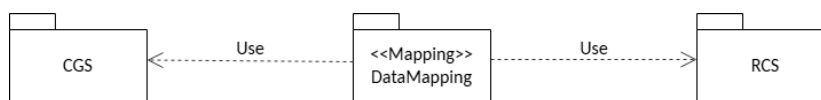


Figura 4: Diagrama de mapeo de datos de nivel de bases de datos del proceso ETL simplificado

Fuente: Elaboración propia

Este archivo proviene de la plataforma Gestor Capital Humano de OTIC SOFOFA y consolida los datos de los cursos, sus costos y participantes, de las empresas del *holding* del Cliente. El esquema del reporte ConsolidadoGestor, contenido en el paquete CGS, es representado por el modelo de datos propuesto en la [Figura 5](#), donde los atributos que no participan del proceso han sido omitidos para simplificar la comprensión de este trabajo.

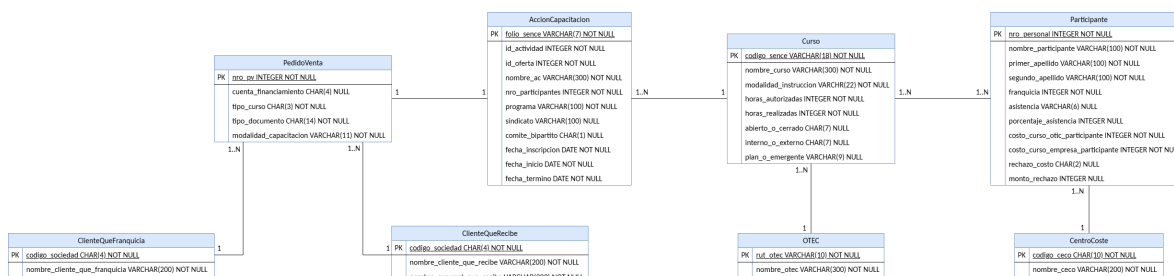


Figura 5: Modelo de datos relacional del reporte ConsolidadoGestor

Fuente: Elaboración propia

El paquete ubicado en el lado derecho de la [Figura 4](#), ReporteConsolidadoSchema (RCS), encapsula los esquemas del reporte de salida del proceso y está compuesto por cuatro paquetes respectivos a las cuatro hojas del ReporteConsolidado: Participantes, Precontrato, Competencias Laborales y Cuentas (ver [Figura 6](#)).

El paquete ParticipantesSchema (PARTS) contiene el modelo de datos que se muestra en la [Figura 7](#). En esta hoja del reporte, los datos de entrada del proceso de transformación



Figura 6: Estructura del paquete ReporteConsolidadoSchema

Fuente: Elaboración propia

corresponden al ConsolidadoGestor filtrado según las siguientes condiciones:

- tipo_curso == 'NOR' y modalidad_capacitacion == 'NOR' (cursos normales)
- tipo_curso == 'NIV' (cursos de nivelación)
- tipo_curso == 'REP' (cursos de reparto)

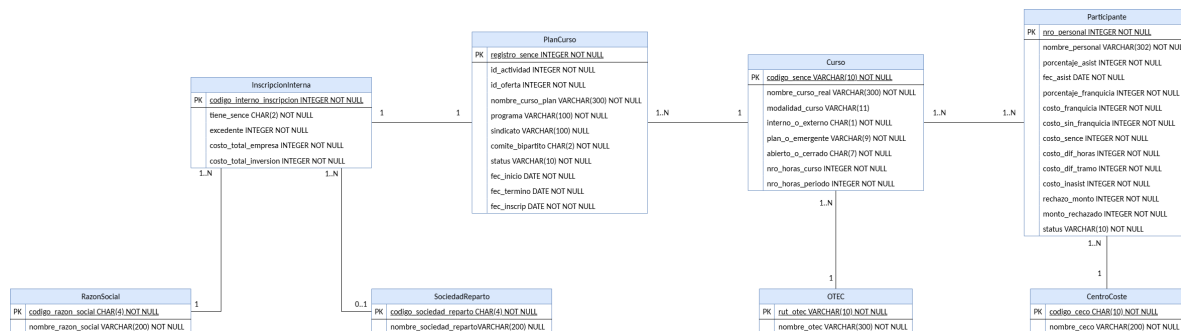


Figura 7: Modelo de datos relacional del esquema ParticipantesSchema

Fuente: Elaboración propia

El paquete PrecontratoSchema (PRECS) encapsula el modelo de datos de la [Figura 8](#). Los datos del ConsolidadoGestor fueron filtrados para procesar y transformar solo los registros donde modalidad_capacitacion == 'PRE' (cursos con precontrato)

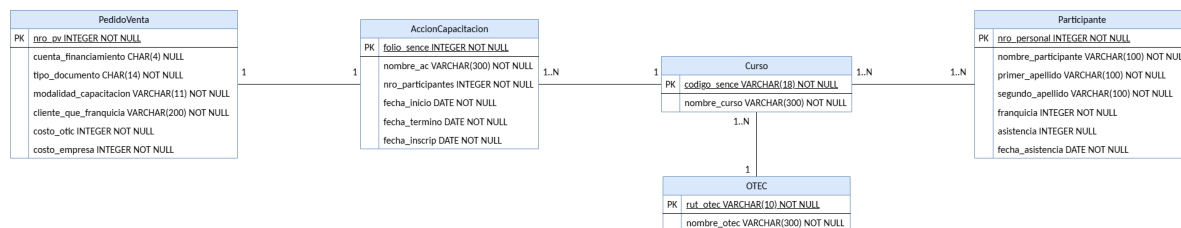


Figura 8: Modelo de datos relacional del esquema PrecontratoSchema

Fuente: Elaboración propia

El paquete CompetenciasLaboralesSchema (COMLS) abstrae el modelo de datos de la [Figura 9](#). Los datos fueron filtrados según la condición modalidad_capacitacion == 'COL' (cursos de competencias laborales).

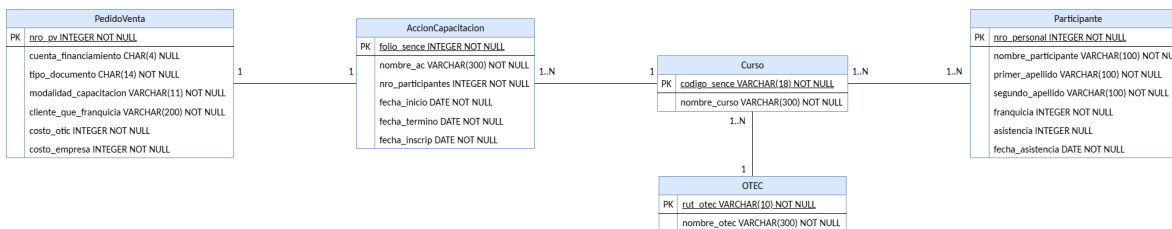


Figura 9: Modelo de datos relacional del esquema CompetenciasLaborales
Fuente: Elaboración propia

El CuentasSchema (CUENS), cómo último paquete, se compone por un modelo de datos que tiene dos clases, como se puede ver en la [Figura 10](#). EstadoCuentaAnual refleja el estado de cuenta por año de las distintas empresas del *holding* del Cliente, mientras que EstadoCuentaTotalAnual contiene la suma de la cuenta anual de todas las empresas. Es decir, refleja el estado de cuenta anual total del Cliente.

3.2.2. Diagrama de mapeo de datos de nivel de flujo de datos

Con los esquemas del diagrama de datos de nivel 0 definidos, ahora corresponde “abrir” el paquete DataMapping e identificar los distintos paquetes de mapeo. Para este proceso ETL simplificado, se ha decidido hacer un mapeo de datos por cada tabla de destino. En la [Figura 11](#) se muestran todos los paquetes de mapeo.

3.2.3. Mapeo de nivel de tablas y atributos

En este apartado se describen los diagramas de mapeos de datos de nivel de tablas asociados a cada paquete de mapeo de la [Figura 11](#). Sin embargo, antes de presentar cada diagrama, es necesario considerar las siguientes decisiones de diseño:

- Las especificaciones de los atributos de las clases se han omitido en todos los diagramas para simplificar la lectura.
- Las tablas intermedias que participan de clases de asociación de estereotipo «otuput» no son almacenadas y solo se han agregado para la comprensión de los diagramas, a menos que se especifique lo contrario.
- Se ha modificado el comportamiento del estereotipo «intermediate» para permitir participar de clases de asociación de estereotipo «input». Esto representa que las tablas intermedias que se utilizan en las diferentes reglas de transformación, no son extraídas desde el origen, sino que son gestionados en la implementación, lo que se detallará en la [Sección 3.3](#).

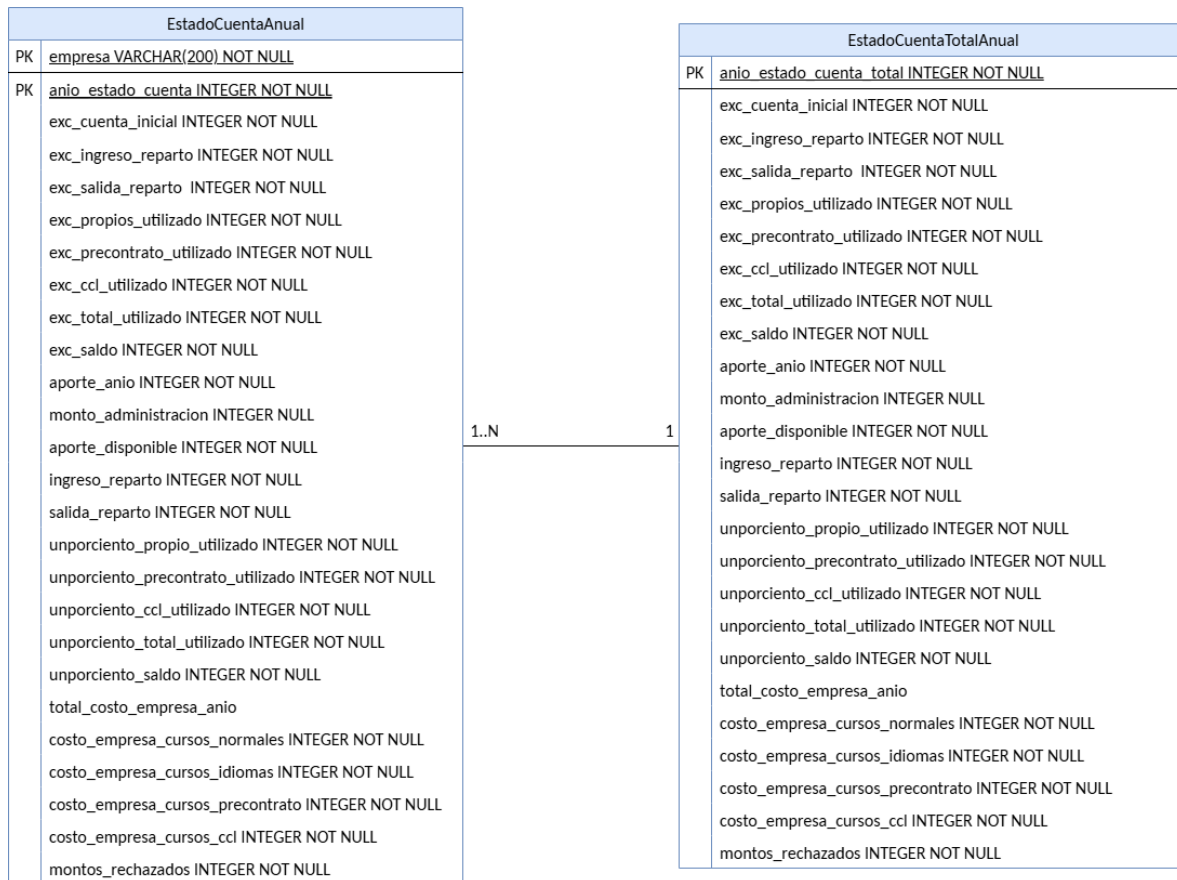


Figura 10: Modelo de datos relacional del esquema CuentasScehema
Fuente: Elaboración propia

- Los diagramas de mapeo de datos de nivel de tablas se presentarán separados por esquema y en el orden que aparecen en el diagrama de mapeo de datos de nivel 1 (ver [Figura 11](#)).
- Las reglas de transformación, representadas en los diagramas con paquetes de mapeo, debiesen ser definidas en el nivel de atributos con diagramas de atributo/clase. El proyecto de este trabajo de título considera más de 100 reglas de transformación y modelar cada una extendería demasiado este documento. Por lo tanto, se ha decidido no realizar estos diagramas y cada regla será descrita en lenguaje natural cuando se presente su mapeo de datos respectivo.

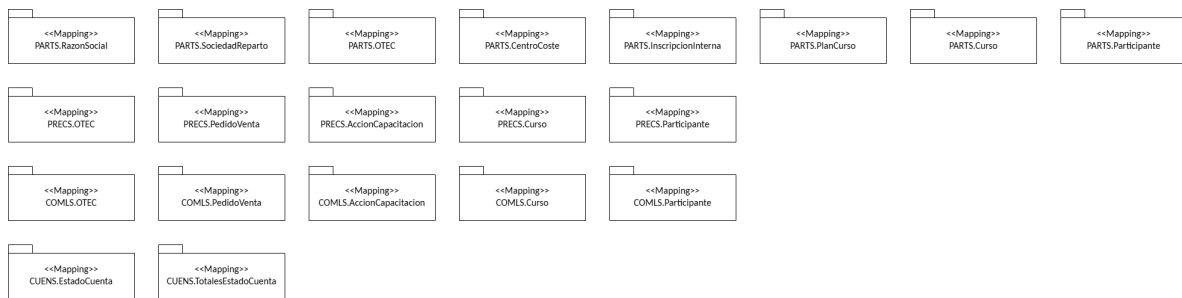


Figura 11: Diagrama de mapeo de datos de nivel de flujo de datos del proceso ETL simplificado

Fuente: Elaboración propia

ParticipantesSchema

RazonSocial: El diagrama de mapeo de datos de nivel de tablas de la [Figura 12](#) utiliza la clases *ClienteQueRecibe* desde el esquema *ConsolidadoGestorSchema* y *EstandarSucursales* que almacena el pareo entre nombre original y nombre estandar, como entradas del proceso. En este mapeo se realizan las siguientes reglas de transformación:

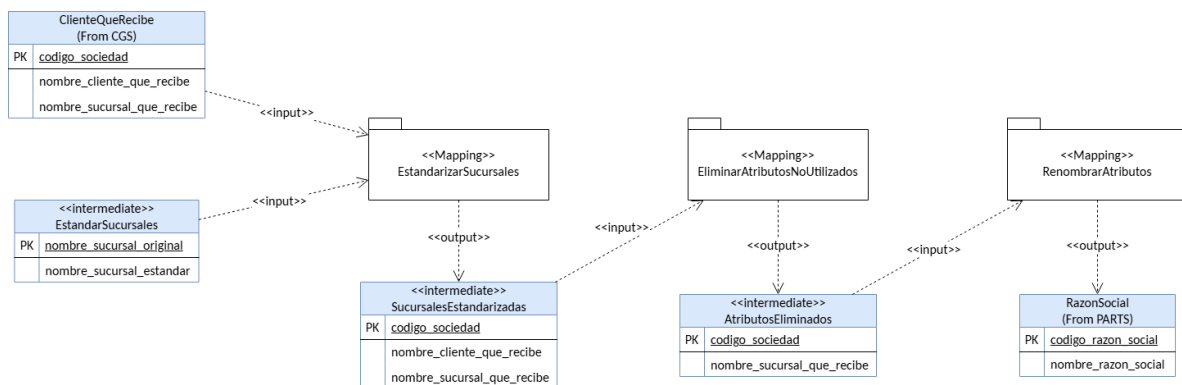


Figura 12: Diagrama de mapeo de datos de nivel de tabla de la clase RazonSocial del esquema ParticipantesSchema

Fuente: Elaboración propia

En el mapeo de nivel 2 de la clase *Razon social* se realizan las siguientes reglas de transformación:

1. **EstandarizarSucursales:** reemplaza en *ClienteQueRecibe* los valores de *nombre_sucursal_que_recibe* por los nombres estandarizados extraídos desde la clase *EstandarSucursales*.
2. **EliminarAtributosNoUtilizados:** elimina el atributo *nombre_cliente_que_recibe* de la clase intermedia *SucursalesEstandarizadas*.

3. RenombrarAtributos: en AtributosEliminados, cambia el nombre de los atributos `codigo_razon_social` y `nombre_sucursal_que_recibe` a `codigo_social_razon` y `nombre_razon_social` respectivamente.

SociedadReparto: Como se muestra en la [Figura 13](#), el mapeo de datos utiliza cuatro clases como entradas del proceso. `ClienteQueFranquicia` y `PedidoVenta` desde el esquema CGS, y las tablas intermedias, `EstandarizarSociedades` que contiene el pareo de entre los nombres de sociedades de reparto originales y los nombres estandarizados, y `CodigosSociedades` que contiene el pareo de códigos únicos de los nombres de sociedades de reparto estandarizados.

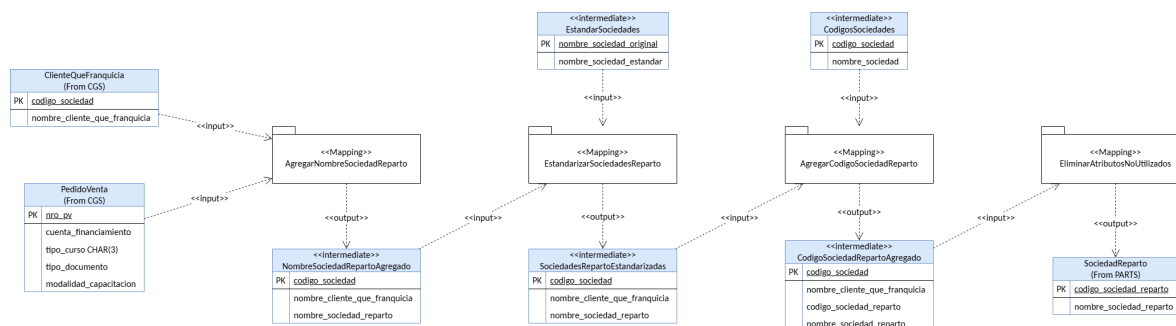


Figura 13: Diagrama de mapeo de datos de nivel de tabla de la clase `SociedadReparto` del esquema `ParticipantesSchema`
Fuente: Elaboración propia

Las reglas de transformación aplicadas en la [Figura 13](#) son:

1. `AgregarNombreSociedadReparto`: agrega un nuevo atributo en `ClienteQueFranquicia`, llamado `nombre_sociedad_reparto`, con el valor del campo `nombre_cliente_que_franquicia` si, en las asociaciones de `ClienteQueFranquicia` con `PedidoVenta`, `tipo_documento` es igual a 'Con Franquicia' y, además, `cuenta_financiamiento` es igual a 'EXCP' o 'EXRP'. En caso contrario, el valor es nulo.
2. `EstandarizarSociedadesReparto`: reemplaza en la clase intermedia `NombreSociedadRepartoAgregado` los valores de `nombre_sociedad_reparto` por los nombres estandarizados extraídos desde la clase `EstandarSociedades`.
3. `AgregarCodigoSociedadReparto`: agrega el atributo `codigo_sociedad_reparto` en la clase intermedia `SociedadesRepartoEstandarizadas` con los códigos de identificación asociados a los nombres estandarizados de las sociedades de reparto, extraídos desde la clase intermedia `CodigosSociedades`.
4. `EliminarAtributosNoUtilizados`: elimina los atributos `codigo_sociedad` y `nombre_cliente_que_franquicia` de la clase interme-

dia CodigoSociedadRepartoAgregado, y asigna el atributo codigo_sociedad_reparto como *primary key* (PK).

OTEC: Recibe como entrada las clases OTEC, desde el esquema CGS, y la tabla intermedia EstandarNombresOTEC, que almacena el pareo entre los nombres originales de los OTEC y los nombres estandarizados. Se aplica una sola regla de transformación, de nombre EstandarizarNombresOTEC, que reemplaza en la clase OTEC, del esquema CGS, los valores del atributo nombre_otec por los nombres estandarizados extraídos desde la clase intermedia EstandarNombresOTEC (ver Figura 14).

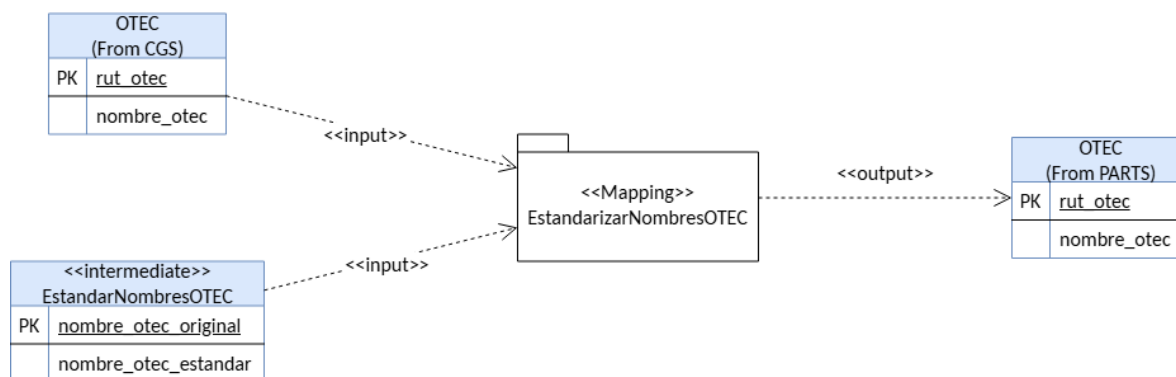


Figura 14: Diagrama de mapeo de datos de nivel de tabla de la clase OTEC del esquema ParticipantesSchema
Fuente: Elaboración propia

CentroCoste: Esta clase no recibe transformaciones entre los esquemas CGS y PARTS. Por consiguiente, no se presenta su diagrama de mapeo de datos respectivo.

InscripcionInterna: Como se refleja en la Figura 15, de entrada se reciben cinco clases desde el esquema CGS, PedidoVenta, AccionCapacitacion, Curso y Participante, y una tabla intermedia EstandarTipoDocumento que almacena el pareo entre los tipos de documentos originales y los tipos estandarizados.

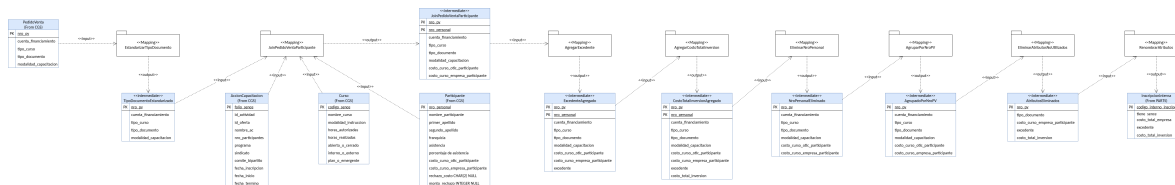


Figura 15: Diagrama de mapeo de datos de nivel de tabla de la clase InscripcionInterna del esquema ParticipantesSchema
Fuente: Elaboración propia

En el diagrama de mapeo de datos de nivel 2 de la Figura 15 se emplean las siguientes reglas de transformación:

1. **EstandarizarTipoDocumento:** modifica los valores del atributo `tipo_documento` de la clase intermedia `PedidoVenta`, asignando los valores 'Si' donde `tipo_documento` sea igual a 'Con Franquicia' y 'No' donde sea igual a 'Sin Franquicia'.
2. **JoinPedidoVentaParticipante:** combina la clases intermedia `TipoDocumentoEstandarizado` con la clase `Participante` para obtener los costos de curso OTIC y empresa por participante asociados a cada par (`nro_pv`, `nro_personal`). Las clases `AccionCapacitacion` y `Curso` son utilizadas para hacer el cruce entre `TipoDocumentoEstandarizado` y `Participante`.
3. **AgregarExcedente:** agrega el atributo `excedente` en la clase `AgrupadoPorNroPV` con el valor de `costo_curso_otic_participante` si `cuenta_financiamiento` es igual a 'EXCP' o 'EXRP'. En caso contrario asigna el valor 0.
4. **AgregarCostoTotalInversion:** agrega el atributo `costo_total_inversion` en la clase `ExcedenteAgregado` con la suma de `costo_curso_otic_participante` y `costo_curso_empresa_participante`.
5. **EliminarNroPersonal:** elimina el atributo `nro_personal` de la clase intermedia `JoinPedidoVentaParticipante`.
6. **AgruparPorNroPV:** agrupa los registros de la clase `NroPersonalEliminado` por `nro_pv`, sumando `costo_curso_otic_participante` y `costo_curso_empresa_participante` agrupados.
7. **EliminarAtributosNoUtilizados:** elimina los atributos `nro_personal`, `cuenta_financiamiento`, `tipo_curso` y `costo_curso_otic_participante` de la clase intermedia `CostoTotalInversionAgregado`.
8. **RenombrarAtributos:** en la clase `AtributosEliminados`, se cambia el nombre del atributo `nro_pv` a `codigo_interno_inscripcion` y, también, del atributo `tipo_documento` a `tiene_sence`.

PlanCurso: En el diagrama de la [Figura 16](#) se consideran dos clases de entrada, `AccionCapacitacion` desde el esquema CGS y la tabla intermedia `EstandarProgramas` que contiene el pareo de los programas originales con los estandarizados.

Las reglas de transformación aplicadas en el diagrama presentado en la [Figura 16](#) son las siguientes:

1. **EstandarizarProgramas:** asigna el valor 'Programa no informado' en el atributo `programa` de la clase `AccionCapacitacion` donde sus valores no aparezcan en la clase `ProgramasInformados`.
2. **EstandarizarComiteBipartito:** modifica los valores del atributo `comite_bipartito` de la clase intermedia `ProgramasEstandarizados`, asignando los valores 'Si' donde `comite_bipartito` sea igual a 'X' y 'No' donde sea nulo.

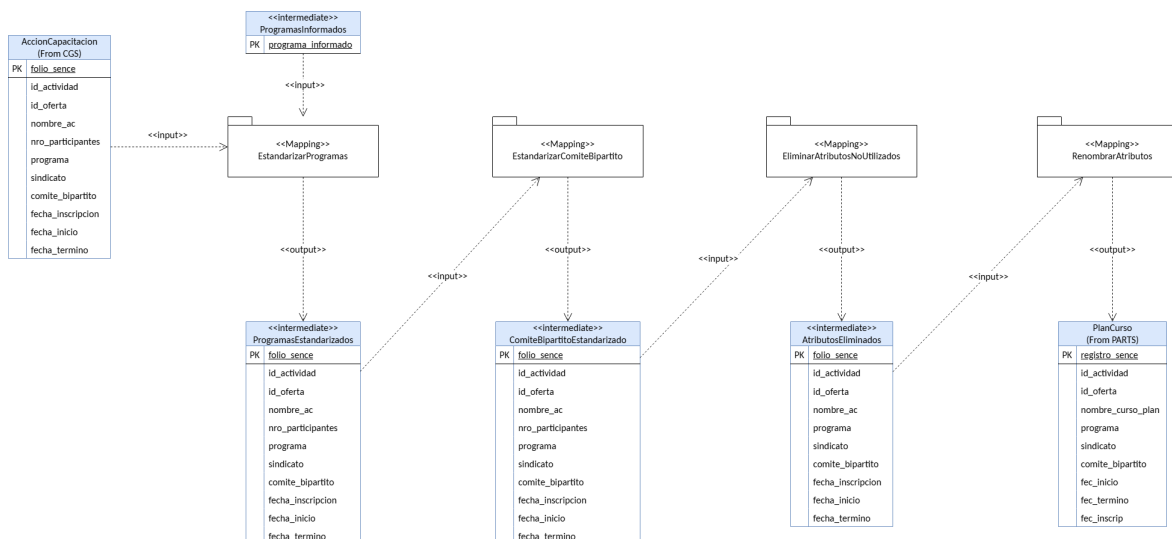


Figura 16: Diagrama de mapeo de datos de nivel de tabla de la clase PlanCurso del esquema ParticipantesSchema
Fuente: Elaboración propia

3. EliminarAtributosNoUtilizados: elimina el atributo `nro_participantes` de la clase `ComiteBipartitoEstandarizado`.
4. RenombrarAtributos: en la clase `AtributosEliminados`, se cambia el nombre de los atributos `folio_sence` y `nombre_ac` a `registro_sence` y `nombre_curso_plan`, respectivamente, y los atributos que empiezan con `fecha_` se renombran para que comiencen con `fec_`.

Curso: Como se muestra en la Figura 17, se reciben como entrada dos clases, `Curso` desde el esquema CGS y la tabla intermedia, `EstandarModalidadInstruccion` que almacena el pareo entre los tipos de modalidades de instrucción originales y los estandarizados.

Las reglas de transformación aplicadas en la Figura 17 son:

1. EstandarizarModalidadInstruccion: reemplaza en la clase `Curso` los valores del atributo `modalidad_instruccion` por los tipos estandarizados extraídos desde la clase `EstandarModalidadInstruccion`.
2. EstandarizarInternoExterno: modifica los valores del atributo `interno_o_externo` de la clase intermedia `ModalidadInstruccionEstandarizada`, asignando los valores 'I' donde `interno_o_externo` sea igual a 'Interno', y 'E' donde sea igual a Externo o nulo.
3. RenombrarAtributos: en la clase `InternoExternoEstandarizado`, se cambia el nombre de los atributo `nombre_curso`, `modalidad_instruccion`,



Figura 17: Diagrama de mapeo de datos de nivel de tabla de la clase Curso del esquema ParticipantesSchema
Fuente: Elaboración propia

horas_autorizadas y horas_realizadas a nombre_curso_real, modalidad_curso, nro_horas_curso y nro_horas_periodo, respectivamente.

Participante: las entradas del mapeo son cuatro clases del esquema CGS, Participante, Curso, AccionCapacitacion y PedidoVenta, como se muestra en la Figura 18.

En el diagrama de mapeo de datos de nivel de tablas presentado en la Figura 18 se aplican las siguientes reglas de transformación:

1. **AgregarNombrePersonal:** agrega un nuevo atributo nombre_personal en la clase Participante con la concatenación de nombre_participante, primer_apellido y segundo_apellido separados por un espacio.
2. **JoinCursoACPedidoVenta:** combina las clases Curso, AccionCapacitacion y PedidoVenta para obtener los atributos modalidad_instruccion, fecha_termino, cuenta_financiamiento y tipo_documento asociados a cada tupla de llaves (codigo_sence, folio_sence, nro_pv).
3. **TransformarPorcentajeAsistencia:** Mantiene el valor original del atributo porcentaje_asistencia en la clase NombrePersonalAgregado, si en sus asociaciones con la clase JoinCursoACPedidoVenta se cumple que tipo_documento es igual a 'Con Franquicia' y que modalidad_instruccion es igual a 'A distancia', 'E-learning asincronico' o 'E-learning sincronico'. En caso contrario, se cambia el valor por el del atributo asistencia.
4. **AgregarCostoFranquicia:** agrega el atributo costo_franquicia a la clase PorcentajeAsistenciaTransformado con el valor de

gica que se detallará en la [Sección 3.3](#) y, segundo, modifica el atributo creado asignando el valor 0 si costo_inasist es mayor a 0. En caso contrario mantiene el valor original.

9. **AgregarCostoDifTramo:** agrega el atributo costo_dif_tramo a la clase CostoDifHorasAgregado en dos pasos. Primero agrega el atributo con la resta de costo_curso_empresa_participante menos costo_dif_horas, si en sus asociaciones con la clase JoinCursoACPedidoVenta, tipo_documento es igual a 'Con Franquicia' y, en caso contrario, con el valor 0. Segundo, modifica el atributo creado asignando el valor 0 si costo_inasist es mayor a 0. En caso contrario mantiene el valor original.
10. **TransformarCostoFranquicia:** modifica el valor del atributo costo_franquicia en la clase CostoDifTramoAgregado con el valor 0 si costo_inasist es mayor a 0 y, en caso contrario, se mantiene el valor original.
11. **AgregarFecAsist:** agrega el atributo fec_asist a la clase CostoFranquiciaTransformado con el valor de fecha_termino de la clase JoinCursoACPedidoVenta si porcentaje_asistencia es un valor no nulo y. En caso contrario asigna el valor nulo.
12. **AgregarStatus:** agrega un nuevo atributo, de nombre status, a la clase FecAsistAgregada con tres posibles valores, en función de sus asociaciones con la clase JoinCursoACPedidoVenta. 'Inscrito' si la fecha de ejecución del proceso ETL simplificado es menor a fecha_termino. 'Finalizado' si la fecha de ejecución es mayor o igual a fecha_termino y asistencia es nula. 'Cerrado' si la fecha de ejecución es mayor o igual a fecha_termino y asistencia es no nula.
13. **LimpiarRechazoCostoMonto:** reemplaza los valores nulos de los atributos de la clase StatusAgregado, rechazo_costo y monto_rechazado, por los valores 'No' y 0, respectivamente.
14. **EliminarAtributosNoUtilizados:** en la clase RechazoCostoMontoLimpiados, se eliminan los atributos nombre_participante, primer_apellido, segundo_apellido, asistencia, costo_curso_otic_participante y costo_curso_empresa_participante.
15. **RenombrarAtributos:** en la clase AtributosEliminados, cambia el nombre de los atributos franquicia, porcentaje_asistencia, rechazo_costo y monto_rechazo de la clase AtributosEliminados a porcentaje_franquicia, porcentaje_asist, rechazo_monto y monto_rechazado, respectivamente.

PrecontratoSchema

OTEC: Esta clase no recibe transformaciones entre los esquemas CGS y PARTS. Por consiguiente, no se presenta su diagrama de mapeo de datos respectivo.

PedidoVenta: El diagrama de mapeo de datos de nivel de tabla de la [Figura 19](#) considera de entrada cuatro clases del esquema CGS, PedidoVenta, AccionCapacitacion, Curso y Participante, y la tabla intermedia EstandarClienteQueFranquicia que almacena el pareo entre los nombres originales de clientes que franquician cursos y los nombres estandarizados.

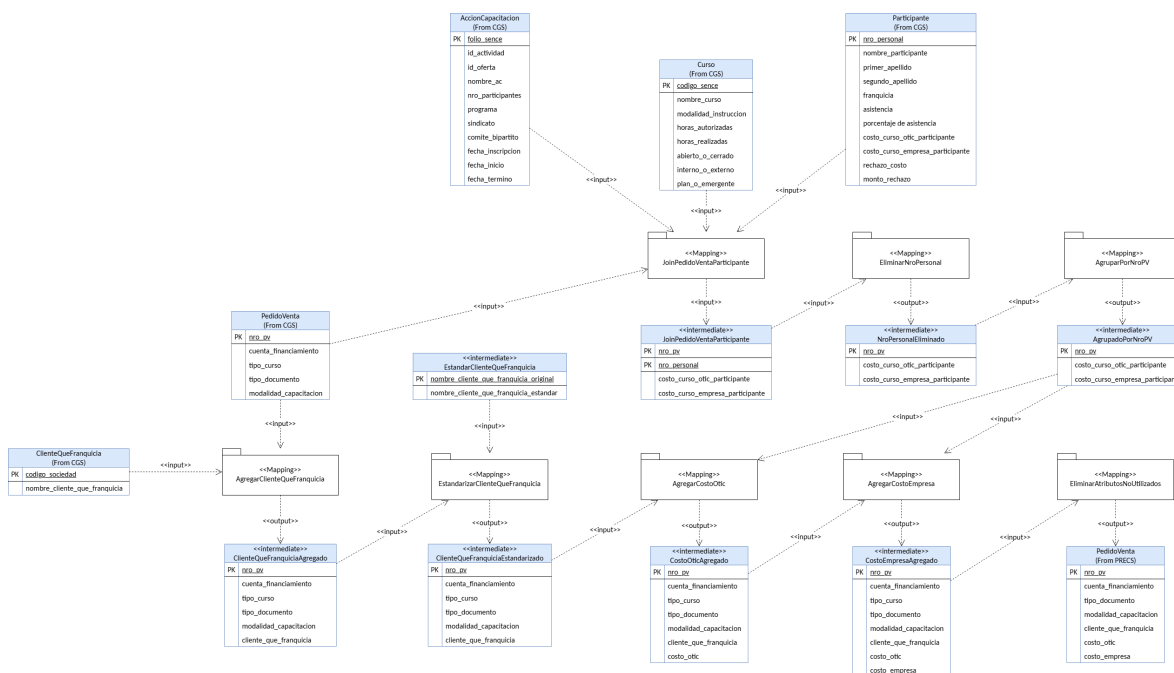


Figura 19: Diagrama de mapeo de datos de nivel de tabla de la clase PedidoVenta del esquema PrecontratoSchema

Fuente: Elaboración propia

Las reglas de transformación aplicadas en el mapeo de datos definido en la [Figura 19](#) son las siguientes:

1. **AgregarClienteQueFranquicia:** agrega un nuevo atributo en la clase PedidoVenta, llamado `cliente_que_franquicia`, con el nombre del cliente que franquicia de las asociaciones que sostiene con la clase ClienteQueFranquicia.
2. **EstandarizarClienteQueFranquicia:** reemplaza en la clase ClienteQueFranquiciaAgregado los valores del atributo `cliente_que_franquicia` por los valores estandarizados extraídos desde la clase EstandarClienteQueFranquicia.

3. **JoinPedidoVentaParticipante:** combina las clases PedidoVenta y Participante extrayendo los costos de curso OTIC y empresa, utilizando las clases AccionCapacitacion y Curso para hacer el cruce.
4. **EliminarNroPersonal:** elimina el atributo nro_personal de la clase JoinPedidoVentaParticipante.
5. **AgruparPorNroPV:** agrupa los registros de la clase NroPersonalEliminado por nro_pv, sumando costo_curso_otic_participante y costo_curso_empresa_participante agrupados.
6. **AgregarCostoOtic:** agrega el atributo costo_otic en la clase ClienteQueFranquiciaEstandarizado con los valores de costo_curso_otic_participante, de la clase AgrupadoPorNroPV, según las asociaciones que estan tengan.
7. **AgregarCostoEmpresa:** agrega el atributo costo_empresa en la clase CostoOticAgregado con los valores de costo_curso_empresa_participante, de la clase AgrupadoPorNroPV, según las asociaciones que estan tengan
8. **EliminarAtributosNoUtilizados:** elimina el atributo tipo_curso de la clase CostoEmpresaAgregado.

AccionCapacitacion: En este mapeo solo se eliminan los atributos no utilizados desde AccionCapacitación del esquema CGS para que coincida con el esquema PRECS. Por la simpleza de esta transformación se omitirá su diagrama de mapeo correspondiente.

Curso: En este mapeo solo se eliminan los atributos no utilizados desde Curso del esquema CGS para que coincida con el esquema PRECS. Por la simpleza de esta transformación se omitirá su diagrama de mapeo correspondiente.

Participante: Como se muestra en la [Figura 20](#), las clases Participante, Curso, y AccionCapacitacion del esquema CGS son las entradas del mapeo.

Las reglas de transformación que se aplican en la [Figura 20](#) son:

1. **LimpiarAsistencia:** aproxima los valores del atributo asistencia de la clase Participante hacia el número entero menor más cercano.
2. **JoinParticipanteAC:** combina las clases Participante y AccionCapacitacion extrayendo la fecha de término asociada al par de llaves (nro_personal, folio_sence), utilizando las clase Curso para hacer el cruce.
3. **AgregarFechaAsistencia:** agrega un nuevo atributo llamado fecha_asistencia en la clase AsistenciaLimpiada con el valor de fecha_termino, de la clase JoinParticipanteAC, en función de las asociaciones que estas clases sostienen, si asistencia es no nula. En caso contrario, asigna el valor nulo.

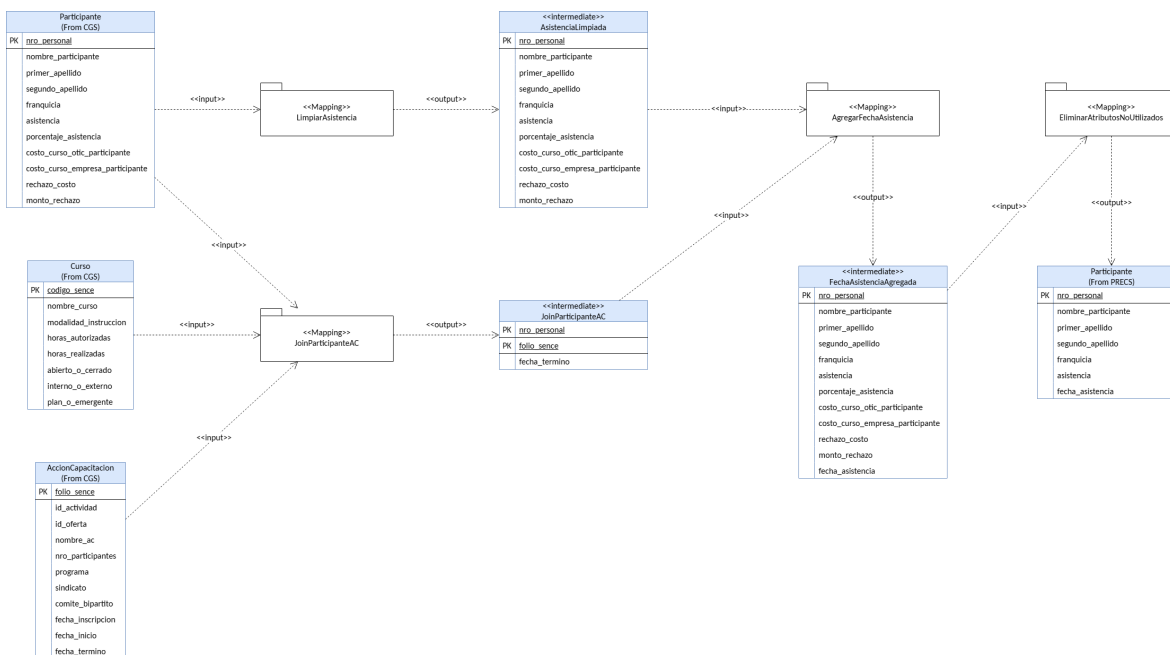


Figura 20: Diagrama de mapeo de datos de nivel de tabla de la clase Participante del esquema PrecontratoSchema
Fuente: Elaboración propia

4. EliminarAtributosNoUtilizados: en la clase FechaAsistenciaAgregada, se eliminan los atributos porcentaje_asistencia, costo_curso_otic_participante, costo_curso_empresa_participante, rechazo_costo y monto_rechazo.

CompetenciasLaboralesSchema

OTEC: Esta clase no recibe transformaciones entre los esquemas CGS y PARTS. Por consiguiente, no se presenta su diagrama de mapeo de datos respectivo.

PedidoVenta: El diagrama de mapeo de datos de nivel de tabla de la [Figura 21](#) considera como entrada dos clases del esquema CGS, PedidoVenta y ClienteQueFranquicia, y una tabla intermedia EstandarTipoCurso que almacena el pareo entre los tipos de curso originales y los estandarizados.

A continuación se definen las reglas de transformación utilizadas en el diagrama de la [Figura 21](#):

1. EstandarizarTipoCurso: reemplaza en la clase PedidoVenta los valores del atributo tipo_curso por los valores estandarizados extraídos desde la clase EstandarTipoCurso.

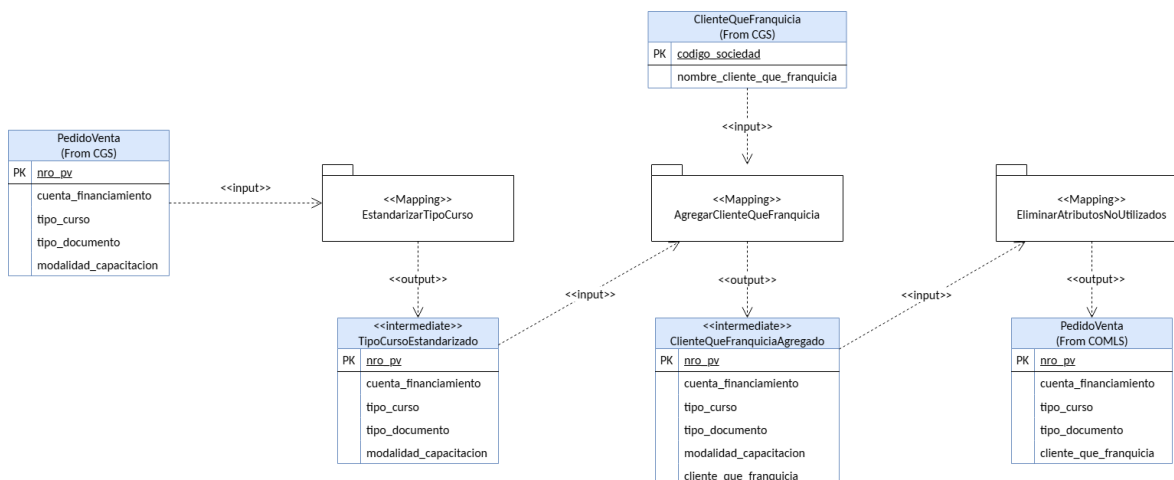


Figura 21: Diagrama de mapeo de datos de nivel de tabla de la clase PedidoVenta del esquema CompetenciasLaboralesSchema
Fuente: Elaboración propia

2. AgregarClienteQueFranquicia: agrega el nuevo atributo cliente_que_franquicia desde la clase ClienteQueFranquicia en la clase intermedia TipoCursoEstandarizado según las asociaciones que estas tengan.
3. EliminarAtributosNoUtilizados: elimina el atributo modalidad_capitacion de la clase ClienteQueFranquiciaAgregado.

AccionCapacitacion: En esta mapeo solo se eliminan los atributos no utilizados de la clase AccionCapacitacion del esquema CGS para que coincida con el esquema COMLS. Por la simpleza de este transformación se omitirá su diagrama de mapeo correspondiente.

Curso: En esta mapeo solo se eliminan los atributos no utilizados de la clase Curso del esquema CGS para que coincida con el esquema COMLS. Por la simpleza de este transformación se omitirá su diagrama de mapeo correspondiente.

Participante: En la [Figura 22](#) se muestra que el diagrama de mapeo de datos utiliza cuatro entradas correspondientes a las clases del esquema CGS, Participante, Curso, AccionCapacitacion y PedidoVenta

En la [Figura 22](#) se tienen las siguientes reglas de transformación:

1. EliminarIVACostoEmpresa: divide los valores de costo_curso_empresa_participante de la clase Participante por 1.19 y aproxima el resultado al número entero mayor más cercano.
2. JoinParticipanteCursoACPedidoVenta: combina las clases Participante, Curso, AccionCapacitacion y PedidoVenta, extrayendo los atributos

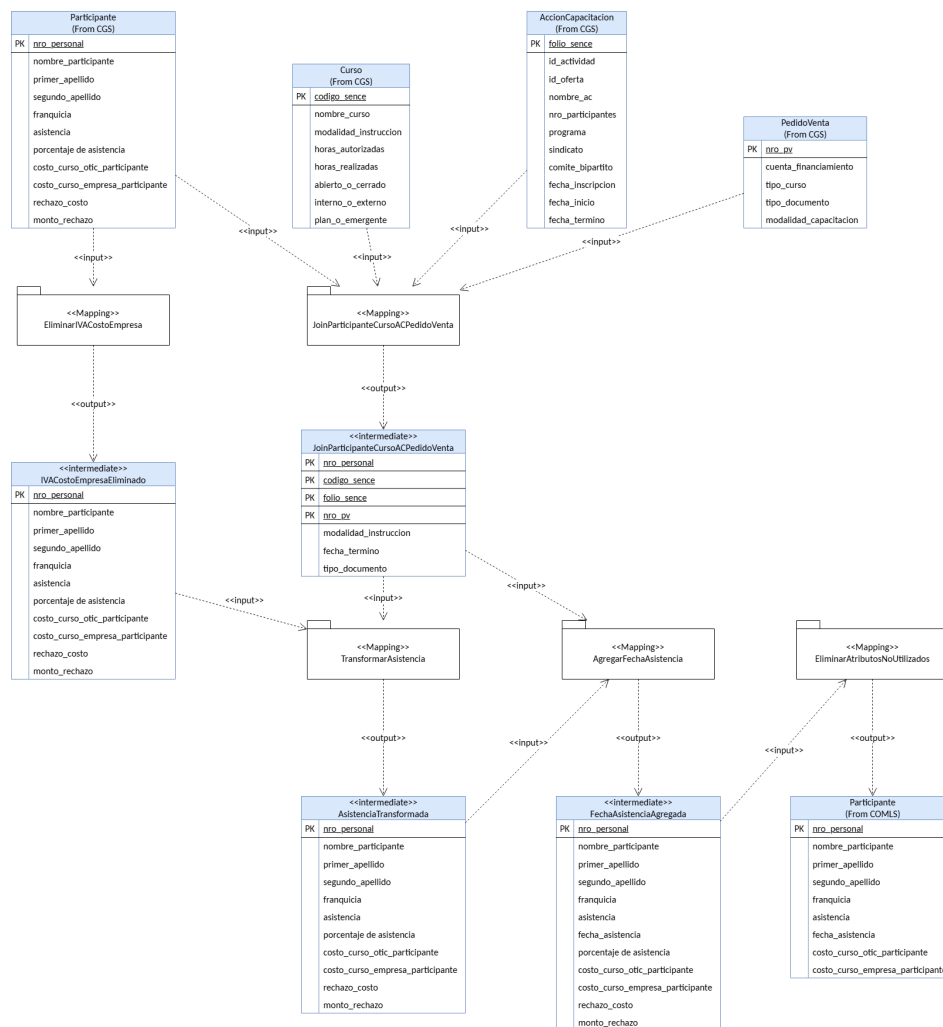


Figura 22: Diagrama de mapeo de datos de nivel de tabla de la clase Participante del esquema CompetenciasLaboralesSchema
Fuente: Elaboración propia

modalidad_instruccion, fecha_termino y tipo_documento asociados a cada tupla de llaves (nro_personal, codigo_sence, folio_sence, nro_pv).

3. TransformarAsistencia: Modifica el valor del atributo asistencia, de la clase IVACostoEmpresaEliminado, por el valor de porcentaje_asistencia si, en las asociaciones que tiene con la clase JoinParticipanteCursoACPedidoVenta, se cumple que tipo_documento es igual a 'Con Franquicia' y modalidad_instruccion es igual a 'A distancia', 'E-learning asincronico' o 'E-learning sincronico'. En caso contrario, se mantiene el valor original.

1. EmpresaAnio como tabla inicial con la empresa y el año del estado de cuenta.
2. CuentaExcedentes que almacena las cuentas de excedentes iniciales de las empresas.
3. RazonSocial desde ParticipantesSchema.
4. SociedadReparto desde ParticipantesSchema.
5. InscripcionInterna desde ParticipantesSchema.
6. PlanCurso desde ParticipantesSchema.
7. Curso desde ParticipantesSchema.
8. Participante desde ParticipantesSchema.
9. PedidoVenta desde PrecontratoSchema.
10. PedidoVenta desde CompetenciasLaboralesSchema.
11. AccionCapacitacion desde CompetenciasLaboralesSchema.
12. Curso desde CompetenciasLaborales.
13. Participante desde CompetenciasLaborales.

y las reglas de transformación realizadas se definen a continuación:

1. AgregarExcCuentaInicial: agrega el atributo `exc_cuenta_inicial` en la tabla base `EmpresaAnio`, extrayendo el monto desde la tabla intermedia `CuentaExcedentes`.
2. JoinInscInternaRazonSocialSocReparto: combina las clases `InscripcionInterna` `RazonSocial` y `SociedadReparto`, extrayendo los nombres de ambos tipos de empresa asociados a cada `codigo_interno_inscripcion`.
3. FiltrarSociedadesRepartoNulas: elimina todos los registros de la clase `JoinRazonSocialSocRepartoInscInterna` donde el atributo `nombre_sociedad_reparto` sea nulo.
4. AgregarExcIngresoReparto: agrega el atributo `exc_ingreso_reparto` en la clase `ExcCuentaInicialAgregada` con la suma de excedente por `nombre_razon_social` de la clase `FiltrarSociedadesRepartoNulas`.
5. JoinSociedadesRepartoNulasFiltradasParticipante: combina las clases `SociedadesRepartoNulasFiltradas` y `Participante`, extrayendo el atributo `costo_franquicia`, junto a los que se propagan desde el *join* anterior, por cada tupla de claves (`codigo_razon_social`, `codigo_sociedad_reparto`, `codigo_interno_inscripcion`, `nro_personal`). Las clases `PlanCurso` y `Curso` se utilizan para hacer el cruce entre las clases anteriores.

6. **AgregarExcSalidaReparto:** agrega el atributo `exc_salida_reparto` en la clase `ExcIngresoRepartoAgregado` con la suma de excedente por `nombre_sociedad_reparto` de la clase `JoinSociedadesRepartoNulasFiltradasParticipante`.
7. **FiltrarSociedadesRepartoNoNulas:** elimina todos los registros de la clase `JoinInscInternaRazonSocialSocReparto` donde el atributo `nombre_sociedad_reparto` sea no nulo.
8. **AgregarExcPropiosUtilizados:** agrega el atributo `exc_propios_utilizados` en la clase `ExcSalidaRepartoAgregado` con la suma de excedente por `nombre_razon_social` de la clase `SociedadesRepartoNoNulasFiltradas`.
9. **AgregarExcPrecontratoUtilizado:** agrega el atributo `exc_precontrato_utilizado` en la clase `AgregarExcPropiosUtilizadosAgregado` con la suma de `costo_otic` por `cliente_que_franquicia` de la clase `PedidoVenta` del esquema PRECS.
10. **JoinPedidoVentaParticipante:** combina las clases `PedidoVenta` y `Participante` del esquema COMLS, extrayendo los atributos `cliente_que_franquicia`, `cuenta_financiamiento`, `costo_curso_otic_participante` y `costo_curso_empresa_participante` por cada dupla de claves (`nro_pv`, `nro_personal`). Las clases `AccionCapacitacion` y `Curso` se utilizan para hacer el cruce entre las clases anteriores.
11. **FiltrarCuentaFinanciamiento:** elimina todos los registros de la clase `JoinPedidoVentaParticipante` donde el atributo `cuenta_financiamiento` sea distinto a 'EXCP'.
12. **AgregarExcCCLUtilizado:** agrega el atributo `exc_ccl_utilizado` en la clase `ExcPrecontratoUtilizadoAgregado` con la suma de `costo_curso_otic_participante` por `cliente_que_franquicia` de la clase `CuentaFinanciamientoFiltrada`.
13. **AgregarExcTotalUtilizado:** agrega el atributo `exc_total_utilizado` en la clase `ExcCCLUtilizadoAgregado` con la suma de los atributos `exc_salida_reparto`, `exc_propios_utilizados`, `exc_precontrato_utilizado` y `exc_ccl_utilizado`.
14. **AgregarExcSaldo:** agrega el atributo `exc_saldo` en la clase `ExcTotalUtilizadoAregado` con la resta del atributo `exc_cuenta_inicial` menos `exc_total_utilizado`.

AporteAnual: La [Figura 24](#) muestra la sección `AporteAnual` del mapeo de datos de la clase `EstadoCuentaAnual`.

El diagrama mostrado en la [Figura 24](#) utiliza tres clases intermedias como entradas:

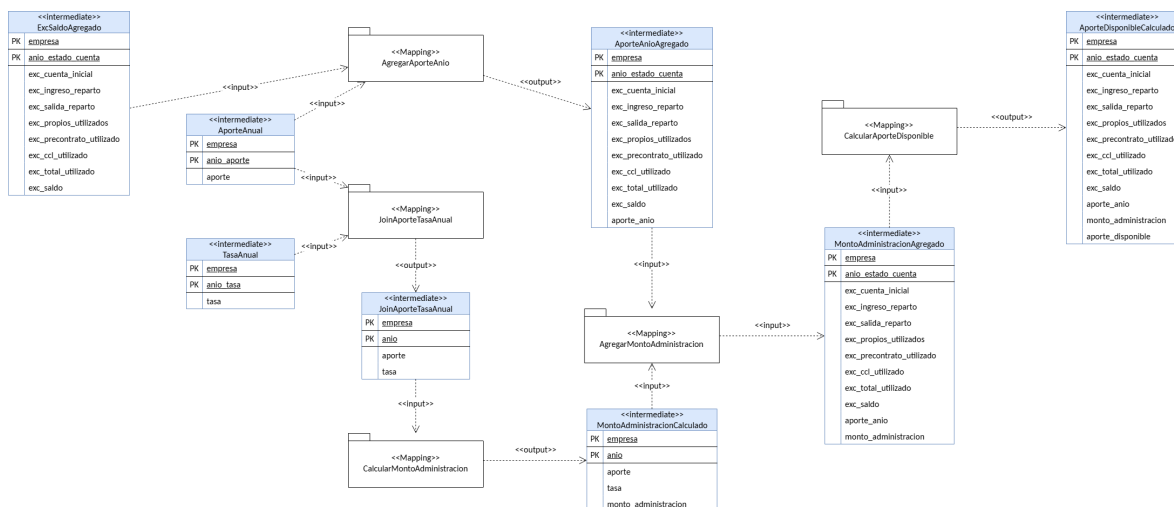
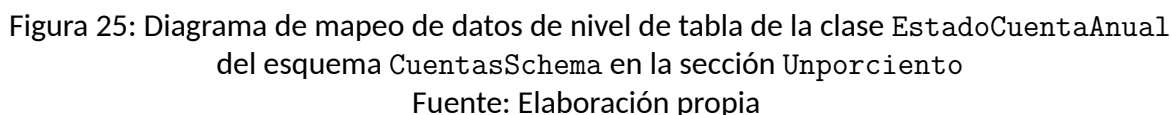


Figura 24: Diagrama de mapeo de datos de nivel de tabla de la clase EstadoCuentaAnual del esquema CuentasSchema en la sección AporteAnual
Fuente: Elaboración propia

1. ExcSaldoAgregado desde el resultado del mapeo de datos de la sección Excedentes (Figura 23).
2. AporteAnual que almacena los aportes realizados por las diferentes empresas del cliente por año.
3. TasaAnual que contiene las tasas de administración anuales de cada empresa.

y se aplican las siguientes reglas de transformación:

1. AgregarAporteAnio: agrega el atributo aporte_anio en ExcSaldoAgregado extrayendo el aporte desde la clase intermedia AporteAnual.
2. JoinAportesTasaAnual: combina las clases intermedias AporteAnual y TasaAnual, extrayendo los aportes y las tasas por cada par de llaves (empresa, año).
3. CalcularMontoAdministracion: agrega el atributo monto_administracion en la clase JoinAportesTasaAnual con la multiplicación de aporte y tasa.
4. AgregarMontoAdministracion: agrega el atributo monto_administracion en la clase AporteAnioAgregado, extrayendo el monto de la clase MontoAdministracionCalculado.
5. CalcularAporteDisponible: agrega el atributo aporte_disponible en la clase MontoAdministracionAgregado con la resta de aporte_anio menos monto_administracion.



1. AporteDisponibleCalculado desde el resultado del mapeo de datos de la sección Aporte (Figura 24).
2. JoinSociedadesRepartoNulasFiltradasParticipante desde la sección Excedentes (Figura 23).
3. PlanCurso desde ParticipantesSchema.
4. Curso desde ParticipantesSchema.
5. Participante desde ParticipantesSchema.
6. SociedadesRepartosNoNulasFiltradas desde la sección Excedentes.
7. PedidoVenta desde PrecontratoSchema.
8. JoinPedidoVentaParticipante desde la sección Excedentes.

Página 44 de 66

1. **AgregarIngresoReparto:** agrega el atributo ingreso_reparto en AporteDisponibleCalculado con la suma de costo_franquicia por nombre_razon_social de la clase intermedia JoinSociedadesRepartoNulasFiltradasParticipante.
2. **AgregarSalidaReparto:** agrega el atributo salida_reparto en IngresoRepartoAgregado con la suma de costo_franquicia por nombre_sociedad_reparto de la clase intermedia JoinSociedadesRepartoNulasFiltradasParticipante.
3. **JoinSociedadesRepartoNoNulasFiltradasParticipante:** combina las clases SociedadesRepartoNoNulasFiltradas desde la sección Excedentes y Participante del esquema PARTS, extrayendo los atributos nombre_razon_social, nombre_sociedad_reparto, excedente y costo_franquicia por cada tupla de llaves (codigo_razon_social, codigo_sociedad_reparto, codigo_interno_inscripcion, nro_personal).
4. **AgregarUnporcientoPropioUtilizado:** agrega el atributo unporciento_propio_utilizado en la clase SalidaRepartoAgregado con la suma de costo_franquicia por nombre_razon_social de la clase JoinSociedadesRepartoNoNulasFiltradasParticipante.
5. **AgregarUnporcientoPrecontratoUtilizado:** agrega el atributo unporciento_precontrato_utilizado en la clase UnporcientoPropioUtilizadoAgregado con la suma de costo_otic por cliente_que_franquicia de la clase PedidoVenta del esquema PRECS.
6. **FiltrarCuentaFinanciamiento:** elimina todos los registros de la clase JoinPedidoVentaParticipante, de la sección Excedentes, donde cuenta_financiamiento sea distinto a 'NORM'.
7. **AgregarUnporcientoCCLUtilizado:** agrega el atributo unporciento_ccl_utilizado en la clase UnporcientoPrecontratoUtilizadoAgregado con la suma de costo_curso_otic_participante por cliente_que_franquicia de la clase CuentaFinanciamientoFiltrada.
8. **AgregarUnporcientoTotalUtilizado:** agrega el atributo unpociento_total_utilizado con la suma de salida_reparto, unporciento_propio_utilizado, unporciento_precontrato_utilizado y unporciento_ccl_utilizado.
9. **AgregarUnporcientoSaldo:** agrega el atributo unporciento_saldo con la resta de aporte_disponible menos unporciento_total_utilizado.

CostoEmpresa: En la [Figura 26](#) se muestra el diagrama de mapeo de datos de la última sección CostoEmpresa. Como es la última sección, el resultado del mapeo es la clase EstadoCuentaAnual del esquema CuentasSchema (ver [Figura 10](#)).

3. AgregarCostoEmpresaCursosNormales: agrega el atributo `costo_empresa_cursos_normales` con la suma de `costo_total_empresa` por `nombre_razon_social` de la clase `ProgramasIdiomasFiltrado`.
4. FiltrarProgramasNoIdiomas: elimina los registros de la clase `JoinRazonSocialInscInternaPlan` donde programa sea distinto a 'Idiomas'.
5. AgregarCostoEmpresaCursosIdiomas: agrega el atributo `costo_empresa_cursos_idiomas` con la suma de `costo_total_empresa` por `nombre_razon_social` de la clase `ProgramasNoIdiomasFiltrado`.
6. AgregarCostoEmpresaCursosPrecontrato: agrega el atributo `costo_empresa_cursos_precontrato` con la suma de `costo_empresa` por `cliente_que_franquicia` de la clase `PedidoVenta` del esquema PRECS.
7. AgregarCostoEmpresaCursosCCL: agrega el atributo `costo_empresa_cursos_ccl` con la suma de `costo_curso_empresa_participante` por `cliente_que_franquicia` de la clase `JoinPedidoVentaParticipante` de la sección Excedentes.
8. AgregarTotalCostoEmpresaAnio: agrega el atributo `total_costo_empresa_utilizado` con la suma de `costo_empresa_cursos_normales`, `costo_empresa_cursos_idiomas`, `costo_empresa_cursos_precontrato` y `costo_empresa_cursos_ccl`.
9. JoinRazonSocialInscInternaPlanParticipante: combina la clase intermedia `JoinRazonSocialInscInternaPlan` con la clase `Participante` del esquema PARTS, extrayendo `monto_rechazado` junto con los atributos propagados por el join anterior, por cada tupla de llaves (`codigo_razon_social`, `codigo_interno_inscripcion`, `registro_sence`, `nro_personal`).
10. AgregarMontosRechazados: agrega el atributo `montos_rechazados` con la suma de `monto_rechazado` por `nombre_razon_social` de la clase `JoinRazonSocialInscInternaPlanParticipante`.

El diagrama de mapeo de datos de nivel de tablas de `EstadoCuentaTotalAnual`, segunda clase del esquema `CuentasSchema`, es mostrado en la [Figura 27](#). Este mapeo considera de entrada solo la clase `EstadoCuentaAnual`, que contiene los estados de cuenta por año de cada empresa del Cliente, también del esquema CUENS. Esto implica que debe ejecutarse las transformaciones de la clase `EstadoCuentaAnual` antes que las correspondientes a la clase `EstadoCuentaTotalAnual`. Se aplica una única regla de transformación, llamada `CalcularEstadoCuentaTotalAnual`, que suma las cuentas de todas las empresas del *holding* del Cliente por cada concepto de cuenta, o en otras palabras, por atributo.

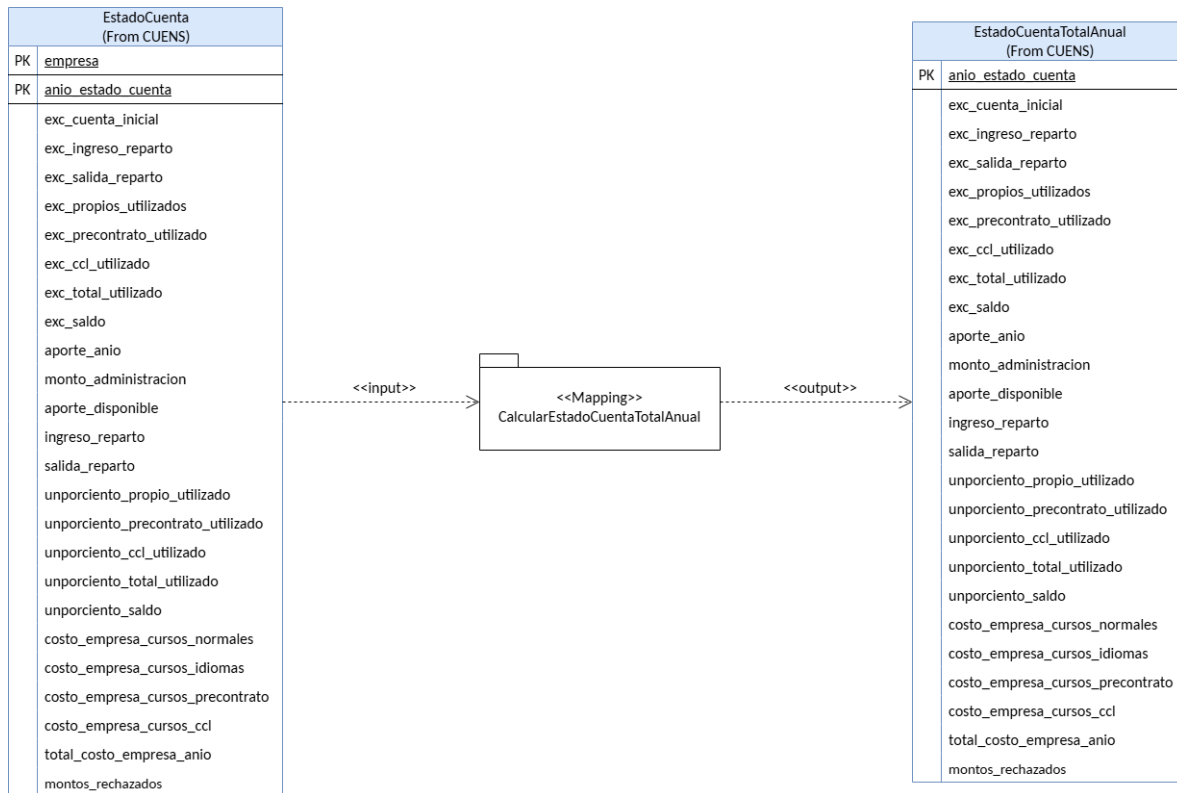


Figura 27: Diagrama de mapeo de datos de nivel de tabla de la clase
EstadoCuentaTotalAnual del esquema CuentasSchema
Fuente: Elaboración propia

3.3. Implementación del proceso ETL simplificado

Como se mencionó en la [Sección 3.1](#), para desarrollar el proceso ETL simplificado se ha elegido el lenguaje de programación *Python* y la librería de manejo de datos *Pandas*. La implementación sigue una estructura modular, ordenando los archivos `.py`, que contienen las funciones necesarias para ejecutar el proceso ETL simplificado, en diferentes carpetas según su funcionalidad. En adelante, se denominará *componentes* a estas carpetas organizativas y *módulos* a los archivos contenidos en ellas. Cuando se mencione un módulo en específico, se omitirá la extensión `.py` pues corresponden a archivos *Python* por definición. Para establecer el flujo del proceso y la ejecución de las funciones definidas en los módulos, se considera un archivo principal `main.py` que orqueste los llamados a funciones. Los siguientes apartados definen los componentes, módulos y funciones de la implementación, y explica la orquestación completa del proceso.

3.3.1. Componentes

Los módulos son organizados en los siguientes componentes:

- **utils:** reúne los módulos con variables y funciones complementarias utilizables por cualquier elemento del proceso.
- **extract:** agrupa los módulos que permiten realizar la extracción de datos desde el reporte consolidado de Gestor y la separación de los datos obtenidos en subconjuntos según tipo de curso y modalidad de capacitación para disponer de las entradas de datos para generar las hojas Participantes, Precontrato y Competencias Laborales del reporte consolidado de salida.
- **transform:** contiene los módulos cuyas funciones aplican las reglas de transformación definidas en la [sección 3.2.3](#) para generar las estructuras de datos para la construcción del reporte consolidado final.
- **load:** sus módulos están destinados a exportar el reporte consolidado final en formato *Excel*.

Además, se consideran dos carpetas adicionales. *input* para almacenar el reporte consolidado de Gestor en formato *Excel* y *output* como destino de carga del reporte consolidado final.

3.3.2. Módulos

A continuación se describen los módulos de los componentes *utils*, *extract*, *transform* y *load*. Esta sección será ordenada por componente para simplificar la lectura.

Módulos del componente *utils*

constants: declara variables constantes para ser importadas desde diferentes archivos. Eso se hace para facilitar la gestión de cambios ante nuevos requerimientos que modifiquen los valores de estas constantes. Las constantes declaradas son las siguientes:

- **consolidado_gestor_filename** con un *string* del nombre del archivo del reporte descargado de Gestor.
- **consolidado_gestor_hoja** con un *string* del nombre de la hoja del consolidado de gestor que contiene los datos.

- `consolidado_final_filename` con un *string* del nombre de archivo de salida del consolidado final.
- `estandar_sucursales` con un diccionario de la forma `{'nombre_sucursal_original': 'nombre_sucursal_estandar'}`.
- `estandar_sociedades_reparto` con un diccionario de la forma `{'sociedad_reparto_original': 'sociedad_reparto_estandar'}`.
- `codigos_sociedades_reparto` con un diccionario de la forma `{'codigo_sociedad_reparto': 'nombre_sociedad_reparto'}`.
- `estandar_nombres_otec` con un diccionario de la forma `{'nombre_otec_original': 'nombre_otec_estandar'}`.
- `estandar_programas` con una lista de los programas informados.
- `estandar_modalidad_instruccion` con un diccionario de la forma `{'modalidad_original' : 'modalidad_estandar'}`.
- `estandar_cliente_que_franquicia` con un diccionario de la forma `{'nombre_cliente_original': 'nombre_cliente_estandar'}`.
- `cuentas_excedentes` con un diccionario de la forma `{anio: {'empresa': exc_cuenta_inicial}}` donde `anio` y `exc_cuenta_inicial` son números enteros.
- `aportes_anuales` con un diccionario de la forma `{anio : {'empresa': aporte_anio}}` donde `anio` y `aporte_anio` son números enteros.
- `tasas_anuales` con un diccionario de la forma `{anio: {'empresa': tasa_anio}}` donde `anio` es un número entero y `tasa_anio` es un *float* mayor a 0 y menor a 1.
- `today` con el valor `pandas.Timestamp.today().normalize()` que retorna la fecha y hora de ejecución en el formato de *Pandas*.

Módulos del componente extract

`extract_consolidado`: extrae los datos del reporte consolidado de gestor y divide los datos obtenidos en los subconjuntos de entrada para las diferentes funciones de los módulos de transformación.

Módulos del componente transform

`commons_rules`: sus funciones definen las reglas de transformación comunes entre los distintos mapeos de la [Sección 3.2.3](#) para reutilizarlas en los demás módulos de transformación.

participantes: sus funciones están destinadas a realizar el mapeo completo entre el *DataFrame* de participantes obtenido en la extracción, las reglas de transformación implicadas y la construcción del *DataFrame* con los datos de la hoja Participantes del reporte consolidado final.

precontrato: sus funciones están destinadas a realizar el mapeo completo entre el *DataFrame* de precontrato obtenido en la extracción, las reglas de transformación implicadas y la construcción del *DataFrame* con los datos de la hoja Precontrato del reporte consolidado final.

competencias: sus funciones están destinadas a realizar el mapeo completo entre el *DataFrame* de competencias laborales obtenido en la extracción, las reglas de transformación implicadas y la construcción del *DataFrame* con los datos de la hoja Competencias Laborales del reporte consolidado final.

cuentas: sus funciones están destinadas a realizar el mapeo completo entre los *DataFrames* de participantes, precontrato y competencias laborales obtenidos en los módulos de transformación respectivos, las reglas de transformación implicadas y la construcción del *DataFrame* con los datos de la hoja Cuentas del reporte consolidado final.

Una particularidad que comparten los módulos participantes, precontrato y competencias es que todas las funciones definidas en cada módulo reciben como parámetro, al menos, los *DataFrames* `df_participantes`, `df_precontrato` y `df_competencias` respectivamente. Estas variables corresponden al retorno de la función de extracción que contiene los datos consolidados necesarios para cada fase de transformación (más detalles en la [Sección 3.3.3](#)). Esto implica que:

1. Los diagramas de mapeo de datos de nivel de tablas de los esquemas `ParticipantesSchema`, `PrecontratoSchema` y `CompetenciasLaboralesEsquema` de la [Sección 3.2.3](#) deben ser considerados como un solo diagrama de mapeo. Por transitividad, el módulo cuentas sigue esta misma regla, ya que su función principal de transformación recibe como parámetros los *DataFrames* `df_participantes`, `df_precontrato` y `df_competencias` resultantes de las funciones de transformación respectivas.
2. Todas las operaciones *join*, y las reglas de transformación originadas para tratar datos combinados, deben ser omitidas. En el caso de los mapeos de datos del esquema CUENS, los *joins* y las reglas de transformación relacionadas, se representan con variables auxiliares a las que se les asignan *DataFrames* con los subconjuntos que se necesitan de las variables `df_participantes`, `df_precontrato` y `df_competencias` para implementar las reglas de transformación del esquema CUENS.
3. Las funciones `renombrar_columnas` y `eliminar_columnas` del módulo `common_rules` (más detalles en la [Sección 3.3](#)) deben ser invocadas, en el mismo orden, como las últimas dos reglas de transformación. Así, el retorno de

`eliminar_columnas` corresponderá al *DataFrame* con los datos asociados a la hoja del reporte consolidado final respectiva a la función de transformación que la invoca. La lista de nombres de columnas que se le pase como parámetro a `renombrar_columnas` debe ser definida en función a la estructura del *DataFrame* de entrada transformado en ese punto del proceso y no a lo establecido en el mapeo de nivel de tablas (ver [Sección 3.2.3](#)). Esto se debe a las diferencias que se generan en la implementación respecto al diseño.

Módulos del componente load

`load_consolidado`: carga los datos obtenidos en los módulos de transformación en un archivo *Excel*, aplicando formatos y estilos según lo requerido por el Cliente, y exporta el archivo en la carpeta de destino output.

3.3.3. Funciones

Los siguientes apartados describen las funciones de los distintos módulos a muy alto nivel.

Funciones del módulo `extract_consolidado`

`extract_consolidado_gestor()`: extrae los datos del archivo de entrada alojado en la carpeta input y los separa en tres *DataFrames* de *Pandas*, `df_participantes`, `df_precontrato` y `df_competencias`, bajo las condiciones establecidas en la [Sección 3.2.1](#). La función retorna una tupla con los tres *DataFrames* del consolidado de gestor dividido. La función debe importar las variables `consolidado_gestor_filename` y `consolidado_gestor_hoja` del módulo constants para extraer los datos del *Excel*.

Funciones del módulo `common_rules`

`eliminar_columnas()`: recibe los parámetros `df`, un *DataFrame* cualquiera, y `cols_to_drop`, una lista de strings con los nombres de las columnas a eliminar. La función retorna la variable `df` sin las columnas especificadas en `columns`. Esta función representa la regla de transformación `EliminarAtributosNoUtilizados` utilizada a lo largo de distintos diagramas de la [Sección 3.2.3](#).

`renombrar_columnas()`: recibe los parámetros `df` y `rename_map` correspondientes a un *DataFrame* cualquiera y a un diccionario, respectivamente. La variable `rename_map` tiene la forma `{nombre_columna_original' : nombre_columna_nuevo}`. La función retorna la

variable `df` con sus nombres de columnas cambiados según el parámetro `rename_map`. Esta función representa la regla de transformación `RenombrarAtributos`.

`estandarizar_columnas()`: recibe los parámetros `df`, un *DataFrame* cualquiera; `columna`, un *string* con el nombre de la columna a estandarizar, `df_estandar`, un *DataFrame* con la estandarización de valores de la columna respectiva; `columna_pareo`, un *string* con el nombre de la columna de la variable `df_estandar` con la cual hacer el pareo; y `columna_valores`, un *string* con el nombre de la columna de la variable `df_estandar` de donde extraer los valores estandarizados. La función retorna la variable `df` con los valores de la columna `columna` estandarizados según el parámetro `df_estandar`. Esta función representa todas las reglas de transformación cuyos nombres comiencen con `Estandarizar` y que utilizan una clase intermedia para el pareo de valores.

`reemplazar_valores()`: recibe los parámetros `df`, un *DataFrame* cualquiera; `columna`, un *string* con el nombre de la columna donde se reemplazarán los valores; y `pareo_valores` un diccionario de la forma `{valor_original' : valor_nuevo}`. La función retorna la variable `df` con los valores de `columna` cambiados según el parámetro `pareo_valores`. Esta función representa todas las reglas de transformación cuyos nombres comiencen con `Estandarizar` y que estandarizar valores binarios o ternarios.

Funciones del módulo participantes

`transformar_participantes()`: aplica el mapeo de datos del esquema `ParticipantesSchema` invocando a las funciones auxiliares que aplican las reglas de transformación de cada mapeo e importando las reglas de transformación comunes necesarias desde el módulo `common_rules`. La función retorna la variable `df_participantes` con todas las reglas de transformación aplicadas.

`agregar_excedente()`: retorna el `df_participantes` con una nueva columna llamada `excedente` a la que se le aplica la regla de transformación `AgregarExcedente` de la [Figura 15](#).

`agregar_costo_total_inversion()`: retorna el `df_participantes` con una nueva columna llamada `costo_total_inversión` a la que se le aplica la regla de transformación `AgregarCostoTotalInversion` de la [Figura 15](#).

`estandarizar_programas()`: retorna el `df_participantes` con la regla de transformación `EstandarizarProgramas` de la [Figura 16](#) aplicada. Esta función importa la variable `estandar_programas` para hacer la comprobación de programas no informados.

`agregar_nombre_personal()`: retorna el `df_participantes` con una nueva columna llamada `nombre_personal` a la que se le aplica la regla de transformación `EstandarizarInternoExterno` de la [Figura 18](#).

`transformar_porcentaje_asistencia()`: retorna el `df_participantes` con la regla de transformación `TransformarPorcentajeAsistencia` de la [Figura 18](#) aplicada.

`agregar_costo_franquicia()`: retorna el `df_participantes` con una nueva columna llamada `costo_franquicia` a la que se le aplica la regla de transformación `AgregarCostoFranquicia` de la [Figura 18](#).

`agregar_costo_inasist()`: retorna el `df_participantes` con una nueva columna llamada `costo_inasist` a la que se le aplica la regla de transformación `AgregarCostoInasist` de la [Figura 18](#).

`agregar_costo_sin_franquicia()`: retorna el `df_participantes` con una nueva columna llamada `costo_sin_franquicia` a la que se le aplica la regla de transformación `AgregarCostoSinFranquicia` de la [Figura 18](#).

`agregar_costo_sence()`: retorna el `df_participantes` con una nueva columna llamada `costo_sence` a la que se le aplica la regla de transformación `AgregarCostoSence` de la [Figura 18](#).

`agregar_costo_dif_horas()`: retorna el `df_participantes` con una nueva columna llamada `costo_dif_horas` a la que se le aplica la regla de transformación `AgregarCostoDifHoras` de la [Figura 18](#).

`agregar_costo_dif_tramo()`: retorna el `df_participantes` con una nueva columna llamada `costo_dif_tramo` a la que se le aplica la regla de transformación `AgregarCostoDifTramo` de la [Figura 18](#).

`agregar_fec_asist()`: retorna el `df_participantes` con una nueva columna llamada `fec_asist` a la que se le aplica la regla de transformación `AgregarFecAsist` de la [Figura 18](#).

`agregar_costo_total_inversion()`: retorna el `df_participantes` con una nueva columna llamada `costo_total_inversion` a la que se le aplica la regla de transformación `AgregarCostoTotalInversion` de la [Figura 18](#).

`agregar_status()`: retorna el `df_participantes` con una nueva columna llamada `status` a la que se le aplica la regla de transformación `AgregarStatus` de la [Figura 18](#).

Funciones del módulo precontrato

`transformar_precontrato()`: aplica el mapeo de datos del esquema `PrecontratoSchema` invocando a las funciones auxiliares que aplican las reglas de transformación de cada mapeo e importando las reglas de transformación comunes necesarias desde el módulo `common_rules`. La función retorna la variable `df_precontrato` con

todas las reglas de transformación aplicadas.

`limpiar_asistencia()`: retorna el `df_precontrato` con la regla de transformación `LimpiarAsistencia` de la [Figura 20](#) aplicada.

`agregar_fecha_asistencia()`: retorna el `df_precontrato` con una nueva columna llamada `fecha_asistencia` a la que se le aplica la regla de transformación `AgregarFechaAsistencia` de la [Figura 20](#).

Funciones del módulo competencias

`transformar_competencias()`: aplica el mapeo de datos del esquema `CompetenciasLaboralesSchema` invocando a las funciones auxiliares que aplican las reglas de transformación de cada mapeo e importando las reglas de transformación comunes necesarias desde el módulo `common_rules`. La función retorna la variable `df_competencias` con todas las reglas de transformación aplicadas.

`eliminar_iva_costo_empresa()`: retorna el `df_competencias` con la regla de transformación `EliminarIVACostoEmpresa` de la [Figura 22](#) aplicada.

`transformar_asistencia()`: retorna el `df_competencias` con la regla de transformación `TransformarAsistencia` de la [Figura 22](#) aplicada.

Funciones del módulo cuentas

`transformar_cuentas()`: aplica el mapeo de datos del esquema `CuentasSchema` invocando a las funciones auxiliares que aplican las reglas de transformación de cada mapeo. La función retorna la variable `df_cuentas` con todas las reglas de transformación aplicadas.

`agregacion_comun()`: implementa la lógica de la gran mayoría de reglas de transformación del esquema `CuentasSchema`, correspondientes a aquellas que se apoyan de una tabla, ya sea intermedia o de los esquemas, `PARTS`, `PRECS` o `COMLS`, para agregar una nueva columna a la tabla de cuentas.

`agregar_exc_cuenta_inicial()`: recibe como parámetro el número entero `anio` y retorna el *DataFrame* `df_cuentas` con las cuentas iniciales de excedentes por empresa del año respectivo. Para esto, la función importa la variable `cuentas_excedentes` desde el módulo `constants`. Esta función representa la regla de transformación `AgregarExcCuentaInicial` de la [Figura 23](#).

`agregar_exc_total_utilizado()`: recibe como parámetro el *DataFrame* `df_cuentas` y retorna esta misma variable con la regla de transformación `AgregarExcTotalUtilizado`

de la [Figura 23](#) aplicada.

`agregar_exc_saldo()`: recibe como parámetro el *DataFrame* `df_cuentas` y retorna esta misma variable con la regla de transformación `AgregarExcSaldo` de la [Figura 23](#) aplicada.

`agregar_aporte_anio()`: recibe como parámetros el *DataFrame* `df_cuentas` y el número entero `anio`, y retorna la variable `df_cuentas` con la regla de transformación `AgregarAporteAnio` de la [Figura 24](#) aplicada. Para esto, la función importa la variable `aportes_anuales` del módulo `constants`.

`agregar_monto_administración()`: recibe como parámetros el *DataFrame* `df_cuentas` y el número entero `anio`, y retorna la variable `df_cuentas` con la regla de transformación `AgregarMontoAdministración` de la [Figura 24](#) aplicada. Para esto, la función importa la variable `tasas_anuales` del módulo `constants`.

`agregar_aporte_disponible()`: recibe como parámetro el *DataFrame* `df_cuentas` y retorna esta misma variable con la regla de transformación `AgregarAporteDisponible` de la [Figura 24](#) aplicada.

`agregar_unporciento_total_utilizado()`: recibe como parámetro el *DataFrame* `df_cuentas` y retorna esta misma variable con la regla de transformación `AgregarUnporcientoTotalUtilizado` de la [Figura 25](#) aplicada.

`agregar_unporciento_saldo()`: recibe como parámetro el *DataFrame* `df_cuentas` y retorna esta misma variable con la regla de transformación `AgregarUnporcientoSaldo` de la [Figura 25](#) aplicada.

`agregar_total_costo_empresa_anio()`: recibe como parámetro el *DataFrame* `df_cuentas` y retorna esta misma variable con la regla de transformación `AgregarTotalCostoEmpresaAnio` de la [Figura 25](#) aplicada.

`calcular_totales()`: recibe como parámetro el *DataFrame* `df_cuentas` y agrega una fila con los totales por columna y el valor 'TOTALES' en la columna `empresa`. Esta función representa la regla de transformación `CalcularEstadoCuentaTotalAnual` de la [Figura 27](#).

Funciones del módulo `load_consolidado`

`export_consolidado_final()`: recibe como parámetros los *DataFrames* `df_participantes`, `df_precontrato`, `df_competencias` y `df_cuentas`. Crea un archivo *Excel* de cuatro hojas, `Participantes`, `Precontrato`, `Competencias Laborales` y `Cuentas`, donde se cargan los datos de sus *DataFrames* respectivos. Además, aplica formatos y estilos de celda a los requeridos por el Cliente. Esta función no tiene retorno.

3.3.4. Orquestación

La tarea de orquestar el proceso ETL reducido completo recae en el archivo principal `main.py` que coordina los llamados a las funciones principales de los módulos de extracción, transformación y carga. Así, la orquestación definida en `main.py` sigue los siguientes pasos:

1. Llama a la función `extract_consolidado()` y almacena los *DataFrames* de retorno, `df_participantes`, `df_precontrato` y `df_competencias`, en variables del mismo nombre.
2. Llama a la función `transformar_participantes()`, pasándole como parámetro la variable `df_participantes` obtenida en la extracción, y se asigna el valor de retorno a la misma variable `df_participantes`. En este punto, `df_participantes` almacena los datos transformados para la hoja Participantes del reporte consolidado de salida.
3. Llama a la función `transformar_precontrato()`, pasándole como parámetro la variable `df_precontrato` obtenida en la extracción, y se asigna el valor de retorno a la misma variable `df_precontrato`. En este punto, `df_precontrato` almacena los datos transformados para la hoja Precontrato del reporte consolidado de salida.
4. Llama a la función `transformar_competencias()`, pasándole como parámetro la variable `df_competencias` obtenida en la extracción, y se asigna el valor de retorno a la misma variable `df_competencias`. En este punto, `df_competencias` almacena los datos transformados para la hoja Competencias Laborales del reporte consolidado de salida.
5. Llama a la función `transformar_cuentas()`, pasándole como parámetros los *Dataframes* `df_participantes`, `df_precontrato` y `df_competencias`, almacenando el *DataFrame* de retorno en una nueva variable llamada `df_cuentas`.
6. Para finalizar el proceso, llama a la función `export_consolidado_final()`, pasándole como parámetros los *Dataframes* `df_participantes`, `df_precontrato`, `df_competencias` y `Cuentas`.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Para validar la solución propuesta se ha decidido implementar un proceso de pruebas de calidad de datos, como la que se describen en la [Sección 2.4](#), sobre los datos de salida del proceso ETL reducido que calcule el porcentaje de registros sin errores de calidad, utilizando la métrica *índice de calidad IC* definida por la [ecuación \(1\)](#), donde *registros_con_errores* es la cantidad de registros identificados con al menos un error de calidad de datos en alguna de sus columnas, y *registros_totales* corresponde a la cantidad de filas que tiene el *DataFrame* que se está validando. Si bien, se reconoce que el numerador de la expresión podría expresarse simplemente como *registros_sin_errores*, se define de esta manera para hacer hincapié en que el proceso de pruebas está enfocado a identificar los registros con errores de calidad.

$$IC = \left(\frac{1 - \text{registros_con_errores}}{\text{registros_totales}} \right) \cdot 100 \quad (1)$$

Además, se genera un reporte de resultados con los registros con errores de calidad de datos y cuáles son las columnas con errores para así poder hacer un análisis y realizar correcciones. En las secciones siguientes se definen las reglas de calidad de datos, los casos de prueba a utilizar para validar la solución y los resultados obtenidos del proceso de pruebas.

4.1. Reglas de calidad de datos

Las reglas de calidad de datos, necesariamente, se establecen según el *DataFrame* que se desee validar y, por consiguiente, se definen cuatro grupos de reglas de calidad respectivos a los *DataFrames* resultantes del proceso ETL reducido. Para el *DataFrame* *df_participantes* se valida que no existan valores nulos en las columnas que, según los modelos de datos de la [Sección 3.2.1](#), no puedan tenerlo (completitud de los datos); las relaciones semánticas que existen entre distintas columnas (consistencia de los datos), resumidas en la [Tabla 3](#); y que no existan filas con el valor 'Programa no informado' en la columna *programa*. Respecto al *DataFrame* *df_precontrato*, solo se valida la completitud de columnas, mientras que en *df_competencias* se valida la completitud y que si la columna *asistencia* es no nula, entonces *fecha_asistencia* también debe ser no nula.

En el caso del *DataFrame* *df_cuentas*, se debe validar que las montos de las columnas *exc_saldo*, *aporte_disponible* y *unporciento_saldo*, sean igual a los montos de la cartola de las empresas del *holding* del Cliente, a la fecha de ejecución del proceso de validación. Para esto, se debe ingresar un archivo *Excel*, con el nombre *cartola.xlsx*, en la carpeta *input* con la cartola actualizada de las empresas del Holding del cliente. Este archivo tiene las

Tabla 3: Reglas de validación de consistencia de datos del *DataFrame* de participantes

Regla	Descripción
Status	valida que si status es igual a 'Cerrado', entonces la columna fec_asist no debe ser nula.
Programa y sindicato	valida que si la columna programa tiene el valor 'Sindicatos', entonces sindicato debe ser no nulo. En caso contrario, sindicato debe ser nula.
Rechazo de monto	valida que si la columna rechazo_monto tiene el valor 'Si', entonces monto_rechazo debe ser distinto de 0. En caso contrario, monto_rechazo debe ser igual a 0.
ID actividad y ID oferta	valida que la operación id_actividad - id_oferta sea menor o igual que 0.
Costo de inasistencia	valida que si la columna costo_inasist es distinta de 0, entonces costo_sence, costo_sin_franquicia, costo_dif_horas y costo_dif_tramo deben ser igual a 0; y porcentaje_asistencia debe ser menor a 75.
Costo total empresa	valida que costo_total_empresa sea igual a la suma de costo_inasist, costo_sin_franquicia, costo_dif_horas y costo_dif_tramo.
Costo total inversión	valida que costo_total_inversion sea igual a la suma de costo_sence y costo_total_empresa.
Sociedad de reparto	valida que las columnas codigo_sociedad_reparto y nombre_sociedad_reparto sean, a la vez, ambas nulas o ambas no nulas. Si bien, en la Figura 7 se indica que ambos atributos son no nulos, la relación que tienen las clases InscripcionInterna y SociedadReparto permite que en el consolidado de los datos, las columnas respectivas si puedan presentar valores nulos.

Fuente: Elaboración propia

columnas empresa, saldo_excedentes, aporte_disponible Y cuenta_unporciento. Así, en la implementación se extraen los datos de este archivo para hacer las comparaciones correspondientes.

4.2. Descripción de los casos de prueba

Para la ejecución del proceso de pruebas se han escogido cuatro casos de prueba. Estos corresponden a reportes consolidados de Gestor en rangos de fechas distintos tal que sus cuentas sean distintas. En la [Tabla 4](#) se indican el ID y nombre de los casos de prueba y las fechas de inicio y término de los cursos a considerar.

Tabla 4: Descripción de los casos de prueba

ID	Nombre	Fecha de inicio	Fecha de término
CG01	Consolidado Gestor 1	1 de enero de 2025	30 de junio de 2025
CG02	Consolidado Gestor 2	1 de julio de 2025	31 de diciembre de 2025
CG03	Consolidado Gestor 3	1 de abril de 2025	30 de septiembre de 2025
CG04	Consolidado Gestor 3	1 de enero de 2025	31 de diciembre de 2025

Fuente: Elaboración propia

4.3. Resultados

Al ejecutar un único ciclo de pruebas, se obtuvieron los resultados mostrados en la [Tabla 5](#). Los casos de prueba CG02 y CG04 presentaron errores en los *DataFrames* de participantes, impactando en los resultados en el *DataFrame* de cuentas. Tras analizar los resultados, se ha identificado que los errores de calidad provienen desde los datos del origen.

Tabla 5: Resultados del ciclo de pruebas único

ID	$IC_{participantes}$ (%)	$IC_{precontrato}$ (%)	$IC_{competencias}$ (%)	$IC_{cuentas}$ (%)
CG01	100,00	100,00	100,00	100,00
CG02	99,75	100,00	100,00	90,00
CG03	100,00	100,00	100,00	100,00
CG04	99,44	100,00	100,00	50,00

Fuente: Elaboración propia

Los resultados de la [Tabla 5](#) muestran que el proceso de ETL simplificado que se ha implementado permite confeccionar el reporte personalizado para el Cliente de OTIC SOFOFA cumpliendo con las reglas de transformación solicitadas. El único caso en que el reporte pueda presentar errores en la calidad de sus datos es cuando el reporte consolidado descargado desde la plataforma Gestor viene con errores en sus datos. Cuando esto sucede, se informa a las áreas de la organización responsable de los datos para que realicen las modificaciones correspondientes.

CONCLUSIONES

El trabajo de título desarrollado tuvo como propósito diseñar e implementar un proceso de ETL simplificado para la generación semi-automatizada de un reporte personalizado para una empresa cliente del OTIC SOFOFA, incorporando un proceso de validación de calidad de datos que permitiese identificar los registros que no cumplen con las reglas de transformación definidas. En los capítulos anteriores se describieron el problema y su contexto, el marco conceptual que sustenta las decisiones de diseño, la solución propuesta y los mecanismos de validación y sus resultados.

En este capítulo se presentan los principales hallazgos del trabajo realizado, discutiendo el cumplimiento de los objetivos específicos y el objetivo general; las principales contribuciones técnicas, organizacionales y académicas de esta trabajo de título; las limitaciones emanadas del alcance encontradas a lo largo del desarrollo; las recomendaciones de trabajos futuros que se pueden realizar para mejorar la solución propuesta y las reflexiones finales del autor.

Cumplimiento de objetivos

Para determinar si se logró cumplir o no el objetivo general de este trabajo de título, primero es necesario verificar el cumplimiento de los objetivos específicos. Comenzando con el objetivo específico N° 1, en la [Sección 3.2](#) se modelaron los esquemas del reporte consolidado de gestor ([Figura 5](#)), entrada del proceso ETL simplificado, y del reporte consolidado de salida, en sus hojas *Participante* ([Figura 7](#)), *Precontrato* ([Figura 8](#)), *Competencias Laborales* ([Figura 9](#)) y *Cuentas* ([Figura 10](#)), describiendo los datos y sus relaciones. Además, se modelaron las reglas de transformación establecidas por el negocio (ver [Sección 3.2.3](#)) a través de diagramas de mapeo de datos (ver [Sección 3.2.3](#)). Por todo lo anterior, se considera que el objetivo específico N° 1 ha sido cumplido.

El objetivo específico N° 2 también se cataloga como logrado, pues en la [Sección 3.3](#) se presenta la arquitectura modular que sigue la implementación del proceso ETL simplificado y el flujo de orquestación. En la misma sección se explica cómo fue implementado el desarrollo, utilizando *Python* y *Pandas*, las herramientas de desarrollo seleccionadas, para la confección del reporte de capacitación personalizado en *Excel*. Ergo, se cumple también el objetivo específico N° 3. Aquí se destaca que el rigor en el diseño demostró ser vital para la definición de una arquitectura de implementación sólida y que la estructura modular permitió gestionar la alta complejidad de las transformaciones de manera ordenada, comprensible y escalable.

Para finalizar, el objetivo específico N° 4 se ha logrado a través del cálculo del *indice_de_calidad* que permite medir el porcentaje de calidad del reporte en sus diferentes hojas y comprobar los datos con problemas de calidad, como se detalla en el Capítulo 4. Considerando que se logró implementar el proceso ETL simplificado, permitiendo disponer el reporte que no existía como un producto formal y reproducible dentro de la organización;

que la validación de calidad de datos que permite detectar inconsistencias antes de entregar el resultado final, fortaleciendo la confiabilidad del producto entregado; y que el trabajo fue realizado dentro de los 3 meses establecidos, se concluye que el objetivo general ha sido cumplido.

Principales contribuciones

El trabajo realizado contribuye en diferentes aspectos. Desde un punto de vista técnico, se ha mostrado un caso de éxito donde un proceso ETL simplificado, modificando los conceptos fundamentales del ETL para un caso de aplicación específico, es una opción válida para la solución de problemas de manipulación de datos. Además, la integración de la calidad de datos como un componente estructural y fundamental del proceso completo es coherente con buenas prácticas modernas de ingeniería de datos.

En la perspectiva organizacional, contar con un la construcción de un reporte a través de un proceso semiautomatizado es un aporte por sí solo. Si bien, la solución propuesta es ejecutada bajo demanda de usuario, el tiempo de ejecución es de pocos segundos, gracias a las herramientas de desarrollo elegidas. Además, la identificación de errores en el reporte de capacitación personalizado, y la gestión de su corrección, se realiza de manera rápida gracias al índice de calidad y al informe de pruebas que genera el proceso de validación de calidad de datos. Si bien, no se establecieron métricas de satisfacción del proceso con respecto a las partes interesadas, el valor del proceso ETL simplificado y el proceso de pruebas de calidad de datos aporta la base inicial de una herramienta que puede escalar a una solución mucho más robusta.

En el ámbito académico, este trabajo demuestra la aplicabilidad de la ingeniería de datos, con la implementación de conceptos ETL, calidad de datos y validación, en contextos organizacionales. Además, la contribución más importante de este trabajo de título, que es transversal a todas las miradas anteriores, es la definición del *índice_de_calidad* y la identificación de errores. La solución propuesta no solo confecciona el reporte personalizado requerido, sino que también expone explícitamente la calidad del reporte construido, identificar rápidamente los datos con problema de calidad y gestionar su corrección rápidamente, antes de la entrega final al cliente. Además, aportó trazabilidad en la calidad de los datos indispensable par el negocio y el problema en cuestión.

Alcance y limitaciones

A pesar de que se lograron cumplir los objetivos de este trabajo, y las contribuciones realizadas, la solución propuesta presenta limitaciones inherentes a su arquitectura y al alcance inicial del proyecto. La primera limitación es la intervención manual que se realiza en el proceso para la actualización de los diccionarios de cuentas de excedentes, aportes y tasas

anuales, el archivo *Excel* con los datos de la cartola de las empresas del *holding* del cliente, y la extracción del reporte consolidado de Gestor. Estas tareas manuales se fundamentan en que no se realizó una integración de APIs con Gestor que permitiese obtener todos estos datos, de manera estructurada de forma directa y con datos actualizados, debido a que se optó por no hacerlo debido a los plazos acotados del proyecto.

La segunda limitación es que no existe un ambiente de datos basado en la nube que permita una manipulación de datos más eficiente, la ejecución del proceso basado en *pipelines* automatizados o basados en eventos, métricas personalizadas de aspectos técnicos y estratégicos del proceso, entre otros beneficios. Cabe destacar que, en la actualidad, el OTIC SOFOFA está implementando un *Data Lake*, el cual se encuentra en sus últimos pasos a producción.

Por último agregar que la documentación formal solo quedó representada en los diagrama de mapeo de datos en sus diferentes nivel modelados en la [Sección 3.2](#) y en el repositorio del código en formato *markdown* y como comentarios en los *scripts*. Estos artefactos podrían complementarse con manuales operativos que contengan mayor detalle y solventen la posible ambigüedad que los diagramas o documentación en repositorio puedan presentar para facilitar la transferencia de conocimiento.

Recomendaciones y trabajos futuros

De lo discutido en las secciones anteriores, la primera recomendación es finalizar la implementación del *Data Lake* en OTIC SOFOFA que permita centralizar los datos de la organización y, en particular, de la plataforma Gestor Capital Humano. Así, el proceso ETL simplificado se puede extender a un proceso ETL completo en un ambiente en la nube y eliminar las tareas manuales existentes del proceso actual. Además, en un entorno de datos de este tipo, con herramientas ETL y de orquestación, como las que fueron nombradas en la [Sección 2.5](#), la solución propuesta puede dejar de depender de la ejecución bajo demanda de usuario a ejecutarse automáticamente según un cronograma de ejecución o basado en eventos de cambios en las fuentes de datos de origen. Así se disminuirían *horas personas* significativas.

Considerando el escenario del *Data Lake* de OTIC SOFOFA implementado y operativo, y “*pensando en grande*” sería interesante mejorar la solución propuesta y el producto entregado a una reportería que el Cliente pueda acceder directamente desde Gestor, sin necesidad de intermediar con el OTIC, con métricas y visualizaciones filtrables y con la opción de descargar el reporte en formato *Excel* si se sigue requiriendo. Por último, un trabajo futuro que se debe realizar es la confección de una documentación a nivel operativo para la gestión del conocimiento a lo largo del tiempo.

Reflexión final

Este trabajo de título logra el objetivo de la construcción de un reporte de capacitación personalizado a través de un proceso ETL reducido, acompañado de métricas de calidad y mecanismos de identificación de problemas de calidad, representando un aporte concreto tanto técnico como organizacional. Pero, por sobre la implementación realizada, el aprendizaje central radica en comprender que la calidad de datos, trazabilidad y validación deben formar parte del análisis y diseño de los procesos ETL desde el inicio del proyecto y no ser considerados elementos secundarios que deben revisarse en la etapa final de desarrollo. En relación con la relevancia de la calidad de datos, el trabajo realizado evidencia que la ingeniería de datos no solo consiste en procesar información, sino en generar confianza sobre la misma. La solución propuesta constituye un avance hacia una cultura organizacional orientada a datos confiables y verificables.

REFERENCIAS BIBLIOGRÁFICAS

- [Chile, 1997] Chile (1997). Ley no. 19518, FIJA NUEVO ESTATUTO DE CAPACITACION Y EMPLEO. En: <https://www.bcn.cl/leychile/navegar?idNorma=76201>. Accedido el 24 de diciembre de 2025.
- [ChileAtiende, 2024] ChileAtiende (2024). Franquicia tributaria. En: <https://www.chileatiende.gob.cl/fichas/44808-franquicia-tributaria>. Accedido el 27 de diciembre de 2025.
- [Deb Nath et al., 2021] Deb Nath, R. P., Romero, O., Pedersen, T. B., y Hose, K. (2021). High-level ETL for semantic data warehouses. *Semantic Web: – Interoperability, Usability, Applicability*, 13(1):85–132.
- [El-Sappagh et al., 2011] El-Sappagh, S. H. A., Hendawi, A. M. A., y El Bastawissy, A. H. (2011). A proposed model for data warehouse ETL processes. *Society Logo Journal of King Saud University - Computer and Information Sciences*, 23(2):91–104.
- [Guntupalli y Vamshi ch, 2025] Guntupalli, B. y Vamshi ch, S. (2025). Top skills every ETL developer needs in 2025. *International Journal of Emerging Research in Engineering and Technology (IJERET)*, 6(1):71–81.
- [Kanburoğlu y Tek, 2024] Kanburoğlu, A. B. y Tek, F. B. (2024). Text-to-SQL: A methodical review of challenges and models. *Turkish Journal of Electrical Engineering and Computer Sciences*, 32(3):403–419.
- [Kumaran, 2021] Kumaran, R. (2021). ETL techniques for structured and unstructured data. *International Research Journal of Engineering and Technology (IRJET)*, 08(12):1727 – 1735.
- [Luján-Mora et al., 2004] Luján-Mora, S., Vassiliadis, P., y Trujillo, J. (2004). Data mapping diagrams for data warehouse design with UML. En Atzeni, P., Chu, W., Lu, H., Zhou, S., y Ling, T. W., editores, *Lecture Notes in Computer Science*, pp. 191–204. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Mohammed, 2025] Mohammed, B. (2025). A systematic taxonomy of ETL activities for modern data pipelines. *International Journal of Advanced Multidisciplinary Research and Studies*, 05(02):106–111.
- [OTIC SOFOFA, 2025] OTIC SOFOFA (2025). Quiénes somos. En: <https://www.oticssofofa.cl/quienes-somos/>. Accedido el 27 de diciembre de 2025.
- [OTIC SOFOFA y LICEOS SOFOFA, 2021] OTIC SOFOFA y LICEOS SOFOFA (2021). Memoria sofofa capital humano 2021. En: <https://s3.amazonaws.com/cdn.oticssofofa.cl/Marketing/Memorias/Memoria2021-SofofaCapitalHumano.pdf>. Accedido el 27 de diciembre de 2025.

- [Ridzuan y Zainon, 2024] Ridzuan, F. y Zainon, W. M. N. W. (2024). A review on data quality dimensions for big data. *Procedia Computer Science*, 234:341–348.
- [Rifaieh y Benharkat, 2002] Rifaieh, R. y Benharkat, N. A. (2002). Query-based data warehousing tool. En *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pp. 35–42, McLean, Virginia, USA. Association for Computing Machinery (ACM).
- [Seenivasan, 2023] Seenivasan, D. (2023). Exploring popular ETL testing techniques. *International Journal of Computer Trends and Technology (IJCTT)*, 71(02):32–39.
- [Sharma y Jain, 2014] Sharma, S. y Jain, R. (2014). Modeling ETL process for data warehouse: An exploratory study. En *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, pp. 271–276, Rohtak, India. IEEE.
- [Singu, 2022] Singu, S. K. (2022). ETL process automation: Tools and techniques. *ESP Journal of Engineering & Technology Advancements*, 02(01):74–85.
- [SOFOFA, 2025a] SOFOFA (2025a). Capital humano. En: <https://www.sofofa.cl/ecosistema-sofofa/capital-humano/>. Accedido el 27 de diciembre de 2025.
- [SOFOFA, 2025b] SOFOFA (2025b). Ecosistema SOFOFA. En: <https://www.sofofa.cl/ecosistema-sofofa/>. Accedido el 27 de diciembre de 2025.
- [SOFOFA, 2025c] SOFOFA (2025c). Quiénes somos. En: <https://www.sofofa.cl/somos/>. Accedido el 27 de diciembre de 2025.
- [Souibgui et al., 2019] Souibgui, M., Atigui, F., Zammali, S., Cherfi, S., y Yahia, S. B. (2019). Data quality in ETL process: A preliminary study. *Procedia Computer Science*, 159:676–687.
- [Sreemathy et al., 2020] Sreemathy, J., Joseph V., I., Nisha, S., Prabha I., C., y Priya R.M., G. (2020). Data integration in ETL using TALEND. En *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1444–1448, Coimbatore, India. IEEE.
- [Vassiliadis et al., 2002] Vassiliadis, P., Simitsis, A., y Skiadopoulos, S. (2002). Conceptual modeling for ETL processes. En *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pp. 14–21, McLean, Virginia, USA. Association for Computing Machinery (ACM).
- [Yang et al., 2017] Yang, Q., Ge, M., y Helfert, M. (2017). Guildlines of data quality issues for data integration in the context of the TPC-DI benchmark. En *Proceedings of the 19th International Conference on Enterprise Information Systems*, pp. 135–144, Porto, Portugal. SciTePress.