UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Construcción de una aplicación para la recopilación y visualización de datos sobre promociones inmobiliarias

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Rubio García, Ana

Tutor/a: Doménech i de Soria, Josep

CURSO ACADÉMICO: 2023/2024

Resumen

Este trabajo expone la manera de construir una aplicación para el correcto almacenamiento de datos de promociones inmobiliarias, con el objetivo de crear un cuadro de mandos dinámico para su visualización. Se destaca la importancia de la recopilación, procesamiento y explotación de los datos en el mundo empresarial, enfatizando en el concepto de "B*usiness Intelligence (BI)"* y los beneficios que este conlleva.

Basándose en un caso práctico desarrollado en una empresa de consultoría para un cliente de ámbito financiero, se descomponen consultas más complejas en partes manejables para su reproducción en aplicaciones como Oracle Data Integrator, con lo que se procesan los datos y se construye la aplicación para almacenar la información. Esto incluye el desarrollo y la ejecución de *mappings* y paquetes que toman datos de múltiples tablas origen para construir una tabla destino, pruebas y validaciones de los datos para asegurar la precisión y fiabilidad, y la subida a producción de los datos. Mediante aplicaciones como Qlik se explotan estos datos construyendo el producto final de visualización, un cuadro de mandos.

En definitiva, se demuestra cómo se pueden transformar datos en bruto en una aplicación altamente explotable, así como la comprensión de métricas críticas para la toma de decisiones futuras por parte de un cliente de ámbito financiero.

Palabras clave: transformación de datos, almacenamiento, visualización, promociones inmobiliarias, business intelligence, Oracle Data Integrator, Qlik.

Abstract

This work presents the construction of an application for the correct storage of data on real estate promotions, with the aim of creating a dynamic dashboard for their visualization. It highlights the importance of data collection, processing, and exploitation in the business world, emphasizing the concept of Business Intelligence (BI) and its benefits.

Based on a practical case developed in a consulting company for a financial client, more complex queries are broken down into manageable parts for reproduction in applications such as Oracle Data Integrator, where data is processed and the application for storing the information is built. This includes the development and execution of mappings and packages that take data from multiple source tables to build a destination table, data testing and validation to ensure accuracy and reliability, and the production deployment of the data. Using applications like Qlik, this data is exploited to build the final visualization product, a dashboard.

In general, it demonstrates how raw data can be transformed into a highly exploitable application, as well as the understanding of critical metrics for future decision-making by a financial client.

Keywords: data transformation, storage, visualization, real estate promotions, business intelligence, Oracle Data Integrator, Qlik.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

Vivimos en una era dominada por los datos donde la conectividad a través de internet facilita el flujo constante de información. Cada acción de nuestro día a día genera gran cantidad de datos que son recopilados y utilizados de diferentes maneras y con diferentes fines.

Con hechos cotidianos como pagar la compra del supermercado con nuestra tarjeta de crédito o navegar por internet con nuestro teléfono móvil facilitamos la recopilación de datos acerca de nosotros. Actualmente, con el simple hecho de llevar nuestro *smartphone* en el bolsillo o usar un *smartwatch* hacemos posible la lectura de una gran cantidad de información de nuestra rutina, como los pasos que damos, nuestra ubicación o incluso nuestro ritmo cardiaco.

Con la entrada de internet en la ecuación, esta recopilación de datos es aún mayor, pues es difícil no toparse con las famosas '*cookies*' incluso al buscar qué temperatura hará mañana. Al aceptarlas, estos ficheros de datos se almacenan en nuestra computadora con la finalidad de recordar accesos, datos específicos y conocer hábitos de navegación, todo ello para, posteriormente, mostrarnos anuncios personalizados, por ejemplo.

Si es tal la importancia y magnitud de la recopilación de datos en nuestro día a día, en el mundo empresarial, altamente competitivo, ha adoptado un papel primordial para el éxito de una compañía. Al haberse convertido en uno de los pilares fundamentales para mantener la competitividad de estas, la inversión en el desarrollo y aprovechamiento de técnicas de ciencia de datos, como el análisis o la ingeniería de datos, permite a la empresa obtener una mejor perspectiva de su situación. Con esto, son capaces de tomar mejores decisiones para el crecimiento de la compañía, así como de afrontar posibles riesgos de manera proactiva, antes incluso de que se conviertan en contingencias. Es por todo esto que la información está pasando a ser uno de los principales activos de las compañías, no solo del sector tecnológico sino de todos los ámbitos.

Para realizar todo este trabajo con datos, en la actualidad empresarial se encuentran varios perfiles profesionales, como son analistas de datos, ingenieros de datos o científicos de datos, perfiles cada vez más demandados por las compañías al haberse convertido en uno de los departamentos más esencial y novedoso.

En este escenario surge el concepto de *Business Intelligence* (BI) o inteligencia de negocio, definido como el aprovechamiento de las herramientas tecnológicas y aplicaciones para el desarrollo de estrategias, análisis de datos y toma de decisiones efectivas en el mundo empresarial (Murillo Junco & Cáceres Castellanos). Con esto, las empresas son capaces de comprender mejor el mercado con una visión global del contexto y gestionar sus recursos de manera más eficiente, incrementando así su rendimiento y competitividad.

La inteligencia de negocio surge de la necesidad de hacer frente a limitaciones como, por ejemplo, la falta de conocimientos técnicos, una deficiente integración de los datos, falta de información histórica del negocio o bajo rendimiento en la realización de informes. El flujo de trabajo al adoptar esta forma de negocio tiene cuatro fases: extracción de los datos de las diferentes fuentes disponibles; modelación de los datos mediante procesos

ETL (*Extract*, *Transform*, *Load*) para su limpieza, consistencia e integridad; almacenamiento de los datos en un *Data Warehouse* y *reporting* o explotación de los datos según las necesidades de negocio.

En definitiva, hoy en día los datos deben considerarse uno de los principales activos dentro de las compañías. Es tan importante su recopilación como una buena gestión de estos, incluyendo su limpieza, almacenamiento y análisis para la extracción de conocimiento que impulse la toma de decisiones de las empresas.

1.1 Objetivos

El objetivo principal de este trabajo de fin de grado consiste en la construcción de una aplicación para la recopilación de datos sobre promociones inmobiliarias de manera que permita la explotación de dicha información mediante una visualización adecuada con la construcción de un cuadro de mandos.

Para alcanzar este objetivo se desglosan varios objetivos secundarios a cumplir antes de completar el principal, además de unos objetivos adicionales en cuanto a la explotación de la información almacenada. Dentro de los objetivos secundarios se encuentran los siguientes:

- Revisión de los fundamentos del negocio para contextualizar adecuadamente la aplicación.
- Revisión de las principales herramientas de gestión de datos con las que es posible implementar la aplicación.
- Definición de las necesidades de información que debe cumplir la aplicación.
- Diseño del proceso ETL.
- Implementación de la aplicación.

Dentro de los objetivos adicionales se encuentra la familiarización del estudiante con una nueva aplicación como es *Oracle Data Integrator (ODI)* y la puesta en práctica de los conocimientos adquiridos durante los estudios en un caso real.

Finalmente, se destaca la finalidad global de mostrar cómo la implementación de herramientas de BI permite transformar datos crudos en información útil para las empresas del sector inmobiliario, ya que se facilita la comprensión de los datos y se ofrecen métricas relevantes para apoyar la toma de decisiones futuras.

1.2 Metodología

Para alcanzar todos los objetivos expuestos anteriormente, se empieza por el desarrollo de los objetivos secundarios. A raíz de la definición de las necesidades de información se construye una *query* o consulta que, al ser ejecutada en bases de datos, devuelve todos los registros actuales con los atributos que se desean almacenar en esta aplicación. Para el diseño del proceso ETL que permitirá la creación de la aplicación, en primer lugar se analiza y entiende dicha *query* original para poder hacer la división de esta en partes más sencillas y computacionalmente menos costosas, además de crear las tablas oportunas en bases de datos. A continuación, se comienza con la familiarización con la aplicación Oracle Data Integrator, para poder definir las variables y construir cada uno de los *mappings* y el paquete final.

Con todo esto, se comprueban y pulen todos los errores encontrados, además de realizar ciertas pruebas para validar los resultados obtenidos. Con todo lo construido funcionando correctamente, se suben los elementos oportunos a producción, lo que cargará los datos al ejecutar el paquete.

Finalmente, se expone brevemente la explotación de los datos almacenados, comentando la construcción de un cuadro de mandos que facilitará la comprensión de la información almacenada.

1.3 Estructura del TFG

Este trabajo de fin de grado sigue la estructura común de este tipo de proyectos. En primer lugar, en este capítulo de introducción, se pone en contexto el tema a tratar y se exponen todos los objetivos a alcanzar y la manera en la que se hará. A continuación, el capítulo de contexto describe brevemente el campo de negocio en el que se centra este proyecto, introduciendo la compañía con la que se trabaja, además de contextualizar las herramientas tecnológicas empleadas en el desarrollo de este trabajo, que son a su vez las disponibles en la empresa.

En el capítulo referente al análisis del problema se expone el marco legal, comentando la confidencialidad y la protección de datos. Tras esto, con la metodología se presenta la muestra de datos, con sus orígenes y variables principales, así como el preprocesamiento que se efectúa a los datos y una presentación de las diferentes capas en las que se trabaja con estos datos.

A continuación, el trabajo realizado se centra en la última de estas capas para explicar los resultados, donde se desarrollan la mayoría de los objetivos presentados en el apartado anterior. Este capítulo incluye la construcción de la aplicación para la recopilación de datos de promociones inmobiliarias, llamada ECIAPI, la ejecución del paquete correspondiente de ODI y la subida a producción para la carga de datos.

Tras esto, el capítulo de validación explica la realización de ciertas pruebas y validaciones necesarias para asegurar la corrección de la aplicación antes de subir los datos a producción. Finalmente, se desarrolla brevemente el cuadro de mandos construido en el capítulo de visualización, con el que se explotan los datos almacenados en ECIAPI y se transmite el conocimiento a la compañía cliente.

Para concluir este trabajo de fin de grado, se redactan unas breves conclusiones acerca del proyecto desarrollado.

2. Contexto

En esta sección se expone el contexto del trabajo realizado, tanto tecnológica como conceptualmente. En primer lugar, se desarrollará una descripción del negocio, contextualizando el sector inmobiliario y el papel de la banca en este. Además, se definirá el concepto de promociones inmobiliarias con sus fases, así como otros derivados de su financiación, como préstamo, aval y confirming. Una vez ubicados en el marco que abarca este trabajo, se pasará a explicar en qué consisten los procesos ETL diferenciando las fases que los componen. Seguidamente, se contextualizará este trabajo tecnológicamente, describiendo las diferentes herramientas empleadas para el desarrollo del caso práctico que se expone. Finalmente, se resumirán las principales características contextuales a modo de recapitulación.

2.1. Descripción del negocio

Este proyecto se basa en un caso práctico desarrollado en una empresa de consultoría para un cliente de ámbito financiero. Dicho cliente, cuyo nombre se mantendrá anonimizado por temas de confidencialidad, se trata de un banco de cierto renombre, siendo uno de los más importantes y grandes de España. Se posiciona como líder en banca minorista en España y trabaja en diversas áreas de negocio, como pueden ser el segmento centrado en clientes particulares con diferente posición monetaria o el segmento centrado en clientes autónomos, profesionales y comercios.

Dentro de las diferentes áreas en las que opera este cliente, cabe destacar la del sector inmobiliario. Uno de los puntos de partida clave de este sector es la denominada burbuja inmobiliaria entre finales de la década de los 90 y los 2000. Durante este periodo, los bajos tipos de interés, la creciente demanda y el fácil acceso al crédito hicieron que se disparara la construcción, suponiendo así un rápido crecimiento del mercado inmobiliario y un aumento exponencial en los precios de la vivienda. A mediados de la década de 2000 esta burbuja estalló y el sector inmobiliario se vio altamente afectado por la crisis económica y financiera en la que se sumió el país. Esto llevó a una caída drástica de los precios y la construcción. Tras un largo periodo de ajuste y recuperación gradual, además de la implementación de ciertas medidas, se consiguió estabilizar de nuevo el mercado inmobiliario y recuperar la confianza de inversores. Esta recuperación también se vio apoyada por el crecimiento de la economía y una mayor estabilidad financiera.

“El mercado inmobiliario resistió en 2023 mejor de lo esperado”, según se concluye en el informe de CaixaBank Research (Instituto Nacional de Estadística, 2024). En otras palabras, el mercado inmobiliario se desaceleró, efectuándose menos operaciones y marcando un ritmo de crecimiento de los precios más suave respecto al año anterior. Además, califican el mercado inmobiliario en 2023 como resiliente, ya que la elevada demanda se mantuvo, suponiendo un desajuste con la oferta de vivienda nueva. Junto con esto, la situación financiera de las familias no sufrió tanto como se esperaba, alcanzando el ratio de endeudamiento su nivel más bajo desde 2002. Finalmente, CaixaBank Research previó una mejora en el mercado inmobiliario español para este año 2024.

Según podemos leer en este informe, para el 2024 apuntan la continuación en el apoyo del sector inmobiliario por esos factores económicos que mejoraron su comportamiento a lo largo de 2023, aunque con algo menos de intensidad. Un factor adicional para argumentar estas buenas perspectivas se basa en el descenso previsto en los tipos de interés a mediados de año. En conclusión, se estima un crecimiento en el precio de la vivienda en España para este año 2024, aunque a un ritmo menor que en el año anterior. En cuanto a la oferta de vivienda, también seguirá avanzando moderadamente, con unos costes de financiación y construcción aún elevados. Por su parte, el crecimiento de la renta de las familias gracias al aumento de salarios y la creación de empleo favorecerá la demanda de vivienda.

El mercado inmobiliario es un sector vital de la economía. En este, la banca juega un papel fundamental en cuanto a financiación, ya que es imprescindible en el acceso a la vivienda para la mayoría de las personas. Es por ello que cambios en el sector bancario pueden tener un impacto significativo en las transacciones de bienes raíces. Más concretamente, la mayoría de los compradores de viviendas recurren a préstamos hipotecarios para financiar dicha compra, teniendo la banca el poder de establecer las políticas crediticias como las tasas de interés. Además, podría aumentar o disminuir los requisitos de pago inicial, teniendo como consecuencia un impacto tanto en la demanda como en los precios.

Pero esta no es la única función de la banca en el sector inmobiliario. Junto con la oferta de préstamos hipotecarios, el sector bancario también se encarga de financiar la construcción de edificaciones y proyectos residenciales y proporcionar líneas de crédito para promotores inmobiliarios y constructores. Además, la avanzada experiencia y conocimiento de la banca en la gestión de riesgos financieros la hace indispensable para evaluar la rentabilidad y viabilidad de dichos proyectos, así como para mitigar los posibles riesgos.

2.1.1. Promoción inmobiliaria

Según encontramos en el diccionario de la Real Academia Española el término inmobiliario se define como “empresa o sociedad que se dedica a construir, arrendar, vender y administrar viviendas”. Según esto, se puede definir una promoción inmobiliaria como una actividad económica que engloba desde la compra del suelo o terreno hasta la venta del producto terminado, pasando por la adquisición de financiación, desarrollo del proyecto arquitectónico, obtención de licencias y construcción o reconstrucción del edificio.

Dentro de esta actividad económica se pueden encontrar tres actores principales: el propietario del terreno, el promotor o, en este caso, empresa que impulsa, programa y financia la construcción, y el constructor que se encarga de materializar y edificar los proyectos.

Por otra parte, se pueden encontrar las fases ya mencionadas, además de una fase previa de estudio de viabilidad. Las promociones inmobiliarias se inician con las primeras ofertas y negociaciones entre el propietario de una finca y el promotor. Si la negociación resulta positiva es imprescindible aplazar la compra hasta que no se concluya acerca de la viabilidad de la promoción mediante un análisis de la demanda y la viabilidad económica del proyecto. Es entonces cuando se lleva a cabo la adquisición del terreno.

La adquisición de financiación es una parte crucial en el desarrollo de una promoción inmobiliaria. Los promotores suelen recurrir a préstamos bancarios para financiar varias de las fases mencionadas.

Por otra parte, en la fase de diseño y planificación del proyecto arquitectónico, se debe asegurar que dicho proyecto cumpla con los requisitos legales y las necesidades del mercado. Para ello, se obtienen los permisos y licencias necesarios de las autoridades locales, como el permiso de construcción. Una vez adquirido este último, se puede iniciar la construcción del proyecto, siempre realizando un seguimiento del progreso para garantizar que se cumple con los plazos y presupuestos preestablecidos.

Por último, se procede a la comercialización y venta de las unidades inmobiliarias, aunque puede darse el caso de comercialización sobre plano, es decir, durante la construcción o incluso antes de iniciarla. Además, es importante ofrecer servicios post-venta, como son las garantías de construcción y atención al cliente.

Como se ha mencionado, la financiación es una fase muy importante para el correcto desarrollo del proyecto, estando presente en varias de las fases arriba explicadas. La Real Academia Española define financiar como:

1. *tr. Aportar el dinero necesario para el funcionamiento de una empresa.*
2. *tr. Sufragar los gastos de una actividad, de una obra, etc.*

Siguiendo esta definición, se destacan tres de los componentes importantes de la financiación en el contexto de promociones inmobiliarias, como son los préstamos, los avales y los confirmings.

2.1.2. Préstamo

Según se recoge en el Código Civil (Título X del libro IV, 1889), "por el contrato de préstamo, una de las partes entrega a la otra, o alguna cosa no fungible para que use de ella por cierto tiempo y se la devuelva, en cuyo caso se llama comodato, o dinero u otra cosa fungible, con condición de devolver otro tanto de la misma especie y calidad, en cuyo caso conserva simplemente el nombre de préstamo"

En otras palabras, el préstamo es un acuerdo entre dos partes en el que una entrega algo al otro con la condición de que sea devuelto en un futuro determinado. Siguiendo esta definición, se pueden encontrar diversos tipos de préstamos con y sin intereses, aunque en nuestro caso nos centraremos en aquellos préstamos otorgados por los bancos con el añadido de unos intereses.

Todo préstamo queda definido por un conjunto de elementos siempre presentes, como son el capital principal o monto económico prestado, el interés o coste financiero del préstamo, la cuota o pagos de amortización entre los que se fracciona la cuantía prestada, el plazo para amortizar el capital, el prestamista o entidad financiera que presta el dinero, y el prestatario o persona que recibe el dinero. Estos dos últimos establecen las condiciones en las que se presta el dinero en el correspondiente contrato. En el caso práctico que se expone en este trabajo se distingue como prestamista a la empresa del sector bancario con la que se trabaja, mientras que los prestatarios son sus clientes, en este caso empresas promotoras.

Entre los diferentes tipos de préstamos, nos centraremos en los préstamos hipotecarios. En este tipo de operación financiera el prestatario solicita al prestamista una cantidad de dinero destinada a la compra o restauración de un bien inmueble. Al ser un préstamo que habitualmente considera presupuestos elevados entra en juego el concepto de aval, que se explicará más adelante. Además, suelen tener plazos largos y una comisión con tipo de interés reducido.

2.1.3. Aval

En relación con el concepto de préstamo definido anteriormente, encontramos el término de aval, el cual surge para el respaldo del pago del primero. Según define el Banco de España (Martínez Sanz & Broseta Pont, 2014), "el aval es un contrato por el que una persona física o jurídica garantiza o asegura el cumplimiento de obligaciones, asumiendo el pago una deuda de otra persona si esta no lo realiza".

En este caso, nos centraremos en los avales presentados a la entidad bancaria, en los que la promotora garantiza al banco la existencia de una tercera parte que se hará cargo del cumplimiento de una obligación asumida, y cubrirá su pago en caso de que incumpla.

Al ser el banco el emisor de un préstamo en el contexto de promociones inmobiliarias, es habitual que la promotora presente un aval, bien porque lo aporte como garantía de cumplimiento del pago o bien porque el banco prestamista lo exija. Esto se puede ver en situaciones como la construcción de una vivienda, pero también para el alquiler de una propiedad.

Se pueden distinguir diferentes tipos de avales, entre ellos el aval de pago, el cual garantiza el pago de una deuda si el cliente o deudor principal no lo hace; el aval de cumplimiento, que podría ser exigido por el banco para garantizar el cumplimiento de las obligaciones contractuales del promotor; el aval de licitación, donde si una empresa presenta una oferta para desarrollar un proyecto, pero luego no puede cumplir con los términos del contrato, se compensará la pérdida incurrida por el incumplimiento; y el aval de mantenimiento, que garantiza la corrección de cualquier defecto o problema surgido tras la finalización del proyecto, habitualmente en el ámbito de la construcción.

2.1.4. Confirming

Finalmente, se presenta el confirming (Seegmuller de Carvalho, 2023) como una herramienta financiera cada vez más popular que permite la gestión de los pagos de una empresa a sus proveedores, permitiendo adelantar el pago de las facturas financiándolas antes de la fecha de vencimiento pactada. Esta operación permite fortalecer la relación entre el cliente y el proveedor, ya que favorece la confianza entre ambas partes, pudiendo traducirse en una colaboración más sólida a largo plazo.

Este servicio lo otorga la entidad financiera con la que se contrata, encargada de realizar el pago a los proveedores a cambio de unos intereses y comisión de gestión. En primer lugar, el cliente envía al banco las órdenes de pago antes de su vencimiento, y este informa a los proveedores del importe y fecha de pago, dándoles a estos últimos la opción de elegir entre esperar al vencimiento de las facturas, o cobrarlas anticipadamente financiándolas.

Según la fórmula de pago del cliente se diferencian tres tipos de confirming. Con pago financiado el cliente paga la factura a su proveedor en la fecha de vencimiento y lo

financia a un plazo determinado, cuando el banco le cargará el pago financiado más comisiones e intereses. Con el confirming estándar el banco gestiona los pagos, ofreciendo la posibilidad a los proveedores de adelantar las facturas. Finalmente, con el pronto pago financiado cliente y proveedor pactan una fecha de pago anterior a la de vencimiento obteniendo así un descuento en la factura. Llegada la fecha de vencimiento el importe adelantado se carga al cliente más intereses y comisiones.

Por otra parte, en función de quién asume el riesgo de impago se distinguen el confirming sin recurso, donde el proveedor no asume ningún riesgo y este es trasladado en su totalidad a la entidad financiera, y el confirming con recurso, si el proveedor asume parte de las responsabilidades en caso de impago.

2.2. Procesos ETL

En un ecosistema de inteligencia de negocios, como el que tiene lugar en este proyecto, se da un proceso de tratamiento de los datos que consta de cuatro fases o etapas principales: en primer lugar, se **generan** unos datos, los cuales sufrirán ciertas **transformaciones** según las políticas de gobierno de datos establecidas, para, posteriormente, ser **almacenados** a la espera de su **explotación**. A continuación, se detallará la fase de transformación.

Los procesos ETL (Martínez, 2018) constan de tres etapas a su vez. Según sus siglas, extraer, transformar y cargar (*extract, transform and load* en inglés), son los encargados de transformar los datos. Estos procesos cumplen dos funciones principales en las primeras fases del tratamiento de datos. En primer lugar, se lleva a cabo el formateo y la limpieza de los datos para que, a continuación, puedan ser movidos entre las diferentes fuentes y cargados en el lugar que corresponda dentro del *Data Warehouse* o almacén de datos.

Estos procesos se usan por su notable utilidad a la hora de integrar nuevas aplicaciones tanto en nuestro sistema como en sistemas heredados. Estos últimos constituyen antiguas aplicaciones ya existentes que deben actualizarse, habitualmente por nuevos aplicativos.

Entre los beneficios del uso de las herramientas ETL se pueden destacar la consistencia, actualización, sincronización y detalle que proporcionan en una compañía. Además, un bueno uso de estas supone un importante ahorro económico dentro de un *Data Warehouse* frente a la programación manual, tanto a la hora de la construcción como en su mantenimiento.

Como se ha mencionado antes, un proceso ETL consta de tres etapas: extracción, transformación y carga. La **fase de extracción**, como su nombre indica, tiene como resultado extraer los datos de las fuentes de origen para poder convertirlos a un formato preestablecido que nos permita iniciar la transformación. Según la fuente de origen de los datos, podemos tratar con diversos formatos a la hora de la extracción, por lo que es primordial establecer pautas comunes de manera que se puedan integrar con el conjunto del sistema. Adicionalmente, se debe tener en cuenta el impacto que la extracción tiene sobre el sistema de origen y tratar de minimizarlo, ya que se pueden provocar ralentizaciones o incluso bloqueos en el proceso. Es por ello que una práctica común

suele consistir en programar la extracción en un horario donde la actividad del sistema de origen sea lo más baja posible.

Una vez extraídos los datos, se pasa a la **fase de transformación**. En esta etapa se aplican las reglas establecidas por las políticas de gobierno de datos de la compañía, de manera que estos datos se integren con la globalidad del sistema. Además, estas reglas de negocio se encargan de asegurar la calidad y la utilidad del flujo de datos, por lo que es imprescindible que sean claras e inteligibles. Esta fase de transformación surge fundamentalmente de la necesidad de corrección de los datos, de la traducción de códigos y de la codificación de valores libres, entre otros.

Es común encontrar entre los datos de la fuente de origen diferentes valores que representan la misma información. Un ejemplo de esto lo constituyen los términos "señora" y "sra." o "hombre" y "H" dentro del tratamiento de personas, para lo que se debe establecer un código común que unifique aquellos registros que representan lo mismo. Otras transformaciones habituales incluyen el cálculo o generación de nuevos atributos como resultado de agregados. Este caso se puede ejemplificar con el cálculo del total de contratos efectuados en un mismo centro. No obstante, las primeras transformaciones y comprobaciones que se deben hacer habitualmente vienen ejemplificadas con la transformación de los valores nulos a '0' o '-1', según se trate de un dato de tipo numérico o varchar, o la comprobación de que los datos de tipo fecha constituyan fechas existentes, entre otros.

Finalmente, tras efectuar las transformaciones necesarias a los datos, estos están listos para ser cargados en nuestro sistema de almacenamiento, dando así nombre a la **fase de carga**. La manera de proceder en esta etapa dependerá de las necesidades de la empresa ante esos datos concretos, encontrando diferentes opciones como pueden ser sobrescribir la información antigua, añadir los datos nuevos a los ya almacenados o almacenar un promedio de estos. Las estructuras de almacenamiento tienen historiales de registros, los cuales permiten tanto hacer un seguimiento de las cargas realizadas como tener un control constante sobre la información almacenada en cada instante de tiempo. Una vez más, mantener la calidad de los datos en esta fase es crucial, ya que existe una interacción directa con el almacén de destino y la globalidad de los datos.

Los procesos ETL modernos incorporan métodos de procesamiento en paralelo, lo cual permite una gran mejora en el rendimiento de las operaciones al tratar con grandes volúmenes de datos. Entre los diferentes tipos de paralelismo que se pueden encontrar son habituales el paralelismo de datos, efectuando una división de los ficheros en pequeños archivos; el paralelismo de *pipeline*, el cual permite realizar acciones en paralelo sobre diferentes partes del flujo de datos, como la modificación del valor de un registro al tiempo que se accede al valor de otro registro; y paralelismo de componente, cuyo objetivo es permitir la ejecución simultánea de múltiples procesos.

Entre las dificultades de los procesos ETL se destaca la necesidad de alcanzar una sincronización total entre las distintas fuentes de origen. Para lograrlo es fundamental la consistencia de los datos que se cargan acorde con las reglas de negocio dadas, además de la sincronización de las fuentes de origen en la medida de lo posible. Para esto último es útil conocer el estado de las fuentes en los diferentes momentos para establecer periodos de actualización.

2.3. Herramientas utilizadas

En cuanto a la parte tecnológica empleada para el desarrollo de este proyecto se destacan las herramientas proporcionadas por la reconocida compañía Oracle, ya que son conocidas por estar entre las más utilizadas del mercado al tratar con bases de datos relacionales. El caso práctico que expone este trabajo se da en el ámbito financiero, por lo que se ha optado por una base de datos de tipo relacional que facilite la creación de una base de datos fiable y consistente.

Una alternativa a las herramientas elegidas es MySQL, gratuita y de código abierto. No obstante, en la comparativa en términos de soporte, seguridad y funcionalidades más complejas Oracle obtiene un resultado más favorable, lo que lo convierte en la mejor opción para el caso que se expone.

2.3.1. Oracle Data Integrator (ODI)

Oracle es una buena opción tanto para almacenamiento de datos como en tecnologías más recientes como son la computación en la nube, el big data o arquitecturas orientadas a servicios. Sin embargo, todas estas tecnologías no serían posibles sin la integración de datos. Oracle Data Integrator (en adelante ODI) es la plataforma de Oracle que se encarga de dicha integración (Hofman Miquel, 2015).

ODI introduce la arquitectura ELT (extracción, carga y transformación), invirtiendo el orden original de las siglas de los procesos ETL. Esta modificación en la arquitectura se debe al propio tratamiento que esta plataforma da a los datos. Como hemos visto anteriormente, las plataformas ETL extraen los datos en primer lugar para, posteriormente, realizar las transformaciones oportunas y finalmente cargar los datos ya listos. ODI, por su parte, tras la extracción de datos procede a cargarlos en destino, efectuando las transformaciones necesarias directamente sobre estas tablas destino. De esta manera se aprovecha en mayor medida el potencial propio del lenguaje SQL, buscando esencialmente un ahorro de recursos hardware.

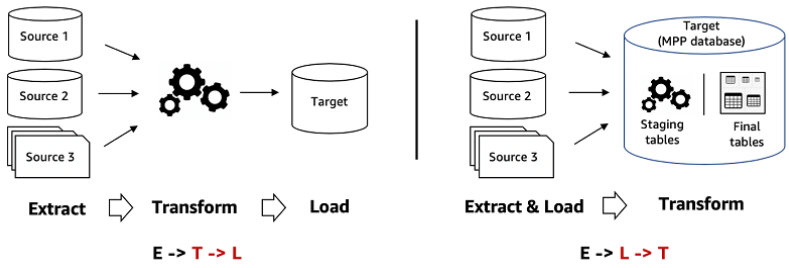

Figura 1. Comparativa ETL (izquierda) y ELT (derecha)
Fuente: (Amazon Web Services, s.f.)

No obstante, ODI también permite alterar su propio flujo de trabajo realizando las transformaciones de los datos tanto sobre la tabla fuente como sobre una base de datos utilizada para hacer de intermediaria.

Dentro de las diferencias que los procesos en ODI suponen respecto a los procesos ETL originales se destacan tres de ellas. En primer lugar, la ya mencionada posibilidad de gestionar el lugar donde se realizan las transformaciones de los datos de manera

dinámica, bien en el origen, bien en el destino. Además, esta arquitectura ofrece la opción de trabajar con SQL de forma global para cualquier base de datos, mientras que otras herramientas solo proporcionan el lenguaje propio de su interfaz. Finalmente, se destaca la posibilidad tanto de generar *Data Manipulation Language* y *Data Definition Language*, como de programar secuencias y operaciones en sistemas homogéneos.

La plataforma de ODI se compone de un repositorio, el cual almacena toda la información necesaria para su funcionamiento, y la interfaz gráfica de la plataforma, a la que se denomina Studio.

En este último se puede encontrar dos esquemas, el esquema lógico y el esquema físico. Ambos esquemas están asociados a un contexto de manera que se conectan entre sí, siendo el esquema lógico una abstracción de las conexiones físicas. Para hacer posible el hecho de que esta abstracción sea traducida para poder ser ejecutada en el esquema físico se introduce la figura de los agentes, los cuales se explicarán más adelante. De esta manera se consigue una independencia entre los parámetros y transformaciones programadas, y la localización física de la base de datos, y viceversa.

A su vez, ODI Studio está dividido en los siguientes apartados:

- Operador: permite revisar el código generado en las ejecuciones, mediante la realización de validaciones, la comprobación y gestión de errores y la visualización de las propias ejecuciones en tiempo real.
- Topología: este apartado se reserva generalmente para los administradores. En él se encuentran los sistemas físicos con los que se trabaja, como las bases de datos de fuente y destino o los contextos, entre otros.
- Seguridad: se destina a la gestión de usuarios, roles y administración.
- Diseñador: como su nombre indica, es el apartado encargado del diseño de las transformaciones y de las reglas ETL, lo cual lleva a cabo mediante modelos, paquetes, asignaciones y variables.

A continuación, se explican los principales elementos de la plataforma de manera que se logre un mayor entendimiento acerca de cómo funciona el proceso de integración de datos con ODI.

2.3.1.1. Agentes

En ODI la figura de los agentes es el componente que dirige todas las operaciones. Desempeñan dos funciones principales, comenzando por la configuración de todos los parámetros oportunos para la ejecución de los procesos. Además, se encargan de la conexión con la base de datos de manera que sea posible enviar el código SQL.

Se distinguen dos tipos de agentes:

- Agentes autónomos o *standalone*, son aplicaciones Java ligeras y fácilmente instalables en cualquier plataforma.
- Agentes JEE (Java Enterprise Edition), reconocidos por aprovechar todas las ventajas que ofrece el servidor dedicado de Oracle llamado Weblogic, el cual permite tanto alta disponibilidad como agrupación de conexiones.

Entre las diversas configuraciones posibles, se encuentra la consistente en utilizar un agente JEE como agente maestro dedicado a distribuir las ejecuciones de manera jerárquica entre varios agentes del tipo *standalone*.

Por su parte, todas estas órdenes de ejecución pueden ser dadas desde ODI Studio, la consola, comandos, planificadores o servicios web.

2.3.1.2. Modelos

Dentro del apartado *Diseñador* se encuentran los modelos. Estos contienen los metadatos o datos acerca de la descripción de los objetos de la base de datos, constituyendo así una representación de los almacenes de datos existentes. Estos modelos pueden incluir, dependiendo del tipo de fuente de datos, tablas, vistas, procedimientos almacenados, entre otros objetos.

Estos metadatos son importados a ODI mediante procesos de ingeniería inversa. Es decir, se analiza la estructura de la fuente de datos y se extraen metadatos relevantes, como nombres de columnas, tipos de datos, relaciones entre tablas, claves primarias y secundarias, entre otros. Se pueden diferenciar dos tipos de ingeniería inversa:

- Ingeniería inversa estándar, la cual recopila los metadatos haciendo uso de un driver JDBC, obtenido mediante la API de Java para conexiones con bases de datos.
- Ingeniería inversa customizada, basada en un módulo de conocimiento para la recopilación de los metadatos de manera que sigue un método específico para la tecnología seleccionada. Este método permite recopilar un rango de información más amplio que la ingeniería estándar.

2.3.1.3. Mappings

Las asignaciones, mapeos o *mappings* contienen las definiciones tanto lógicas como físicas de las transformaciones realizadas sobre los datos. Dichas transformaciones se encuentran aisladas de la definición de los datos y de la integración de estos, lo que supone una separación entre el esquema lógico y el esquema físico, constituyendo así un diseño declarativo.

Esta separación aporta ciertas ventajas o beneficios, como el hecho de que los cambios por un lado no afectan al otro lado. En otras palabras, si cambiamos las asignaciones entre tablas, por ejemplo, esto no va a afectar al tipo de tecnología empleada o a la estrategia de integración de los datos. Por otra parte, se puede destacar el aumento de la productividad como ventaja, ya que, usualmente, los *mappings* son únicos en cuanto a la asignación de los campos de las tablas, pero las estrategias de integración que siguen son comunes.

El esquema lógico del *mapping* permite trabajar de manera gráfica e intuitiva. En él se pueden diseñar las transformaciones de los datos de forma esquemática utilizando diversos componentes, como pueden ser almacenes de datos, conectores o filtros.

En el esquema físico, por su parte, se encuentra la planificación de la estrategia a seguir a través de los módulos de conocimiento, los cuales se explicarán más adelante.

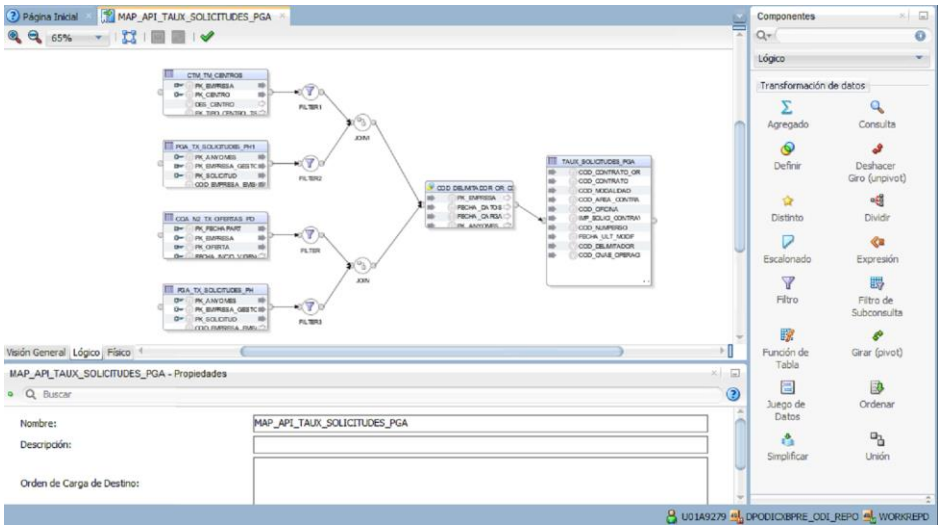

Figura 2. Esquema lógico de un mapping en ODI

Según se visualiza en la figura, el editor de *mappings* de la interfaz de ODI permite diseñar las asignaciones de forma clara e intuitiva, como se ha mencionado anteriormente. En él se destacan los tres apartados principales:

- Diagrama, en el centro de la figura. Representa el esquema físico o lógico, según la pestaña seleccionada en la barra inferior. Este puede ser editado directamente arrastrando a él los componentes deseados, uniendo estos últimos con flechas o moviéndolos para un resultado más claro.
- Paleta de componentes, a la derecha de la figura. Lista los componentes que pueden ser utilizados en el esquema lógico del *mapping*. Para incluirlos basta con arrastrar hacia el diagrama como se ha mencionado antes.
- Inspector, en la parte inferior de la figura. Al seleccionar un componente específico del diagrama, el inspector nos muestra las propiedades de este, además de permitir su modificación.

Dentro del conjunto de componentes que se pueden usar mostrados en la paleta, se diferencian dos tipos. Por un lado, están los componentes proyectores, los cuales determinan los atributos presentes en los datos que son procesados a través del *mapping* en cuestión. Entre ellos destacan los siguientes:

- Pivotes. Permiten transformar los datos contenidos originalmente en múltiples filas en una única fila.
- Distintos. Utilizados para agrupar los atributos de tal manera que cada valor es único.
- Agregados. Empleados en casos similares a los distintos, agrupan los atributos además de utilizar funciones de agregados, como son el máximo, el mínimo, un sumatorio o el valor medio.

Por otra parte, se encuentran los componentes selectores, cuya función hace uso de atributos de componentes predecesores. Los más importantes son:

- Filtros. Permiten establecer condiciones y seleccionar únicamente aquellos datos que cumplan una condición especificada sobre alguno de sus atributos.

- Uniones. Como su propio nombre indica, empleados para unir o combinar los atributos de dos o más componentes.

Al incorporar todos estos componentes al diagrama, es necesario configurar los atributos, las propiedades y las condiciones necesarias en función del componente elegido. Todo ello se hará a través del inspector.

Centrándonos en el esquema físico del *mapping*, se visualiza un diagrama que muestra la relación física entre los diferentes componentes. El recuadro azul indica el almacén de datos que actúa como destino. En el ejemplo dado por la figura se puede ver cómo el tratamiento de los datos se realiza en su totalidad en el destino, siguiendo la arquitectura ELT propia de ODI explicada anteriormente.

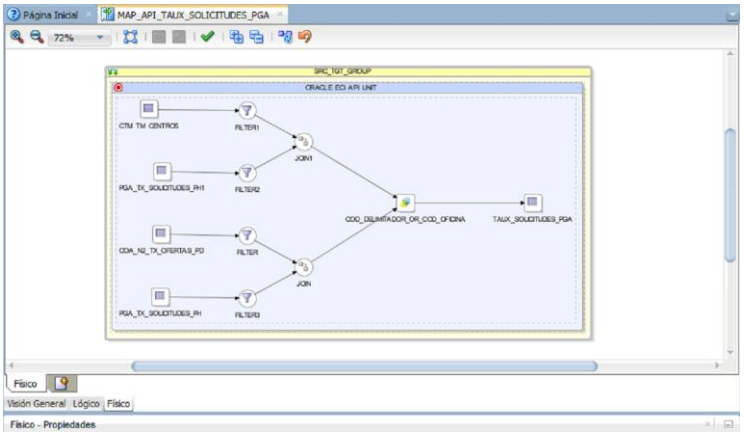

Figura 3. Modelo físico de un mapping en ODI

Al seleccionar el componente destino del *mapping* en el esquema físico del mismo, se pueden configurar a través del inspector los módulos de conocimiento que se explican a continuación. Para ello, en cada proceso se debe aplicar el que más convenga de manera que se consiga el funcionamiento deseado.

2.3.1.4. Módulos de conocimiento

Los módulos de conocimiento son plantillas de código que definen las *queries* SQL y los scripts necesarios para realizar la extracción, transformación y carga de los datos en el proceso de integración. De esta manera, el desarrollador es capaz de utilizar estas plantillas sin necesidad de estudiar el funcionamiento interno de las mismas, ya que se proporciona una abstracción del código.

Son varias las ventajas ofrecidas por los módulos de conocimiento. Entre ellas se destaca un aumento de la productividad, ya que estos módulos solo han de ser programados una vez para permitir su reutilización en múltiples procesos constantemente. Además, ofrecen la posibilidad de personalización al ser posible adaptar cada plantilla al ecosistema en el que se encuentra. Por último, estos módulos de conocimiento permiten a los desarrolladores utilizar la misma plantilla homogeneizando el código, lo que se puede denominar encapsulación del código.

En ODI se encuentras los seis tipos de módulos de conocimiento listados a continuación:

- Loading Knowledge Module (LKM). Este módulo determina las labores de extracción y carga de los datos.
- Integration Knowledge Module (IKM). Empleado cuando se desea que determine la estrategia de integración de los datos en el destino.
- Reverse-engineering Knowledge Module (RKM). Permite la importación de metadatos de la base de datos al repositorio de ODI. No obstante, no todas las operaciones de ingeniería inversa requieren de un RKM.
- Service Knowledge Module (SKM). Convierte y genera el código para ser utilizado en operaciones *web service*.
- Check Knowledge Module (CKM). Implementa controles de calidad en los datos de manera que se validan las transformaciones realizadas, las claves primarias y foráneas, las reglas de negocio y todos aquellos elementos imprescindibles para una correcta integración.
- Journalizing Knowledge Module (JKM). Este módulo permite realizar un seguimiento de los cambios efectuados sobre los datos.

2.3.1.5. Variables y procedimientos

Las variables son creadas con el objetivo de poder ser referenciadas en paquetes, asignaciones y otros componentes. Estas permiten esencialmente almacenar valores, pero también construir y refrescar la información necesaria para la ejecución de procesos.

Se distinguen variables de ámbito global, las cuales pueden ser usadas en cualquier proyecto dentro del repositorio, y variables de proyecto, limitadas a un único proyecto para el que se crean. Los atributos almacenados en las variables son el tipo de dato, entre los que se encuentran numérico, alfanumérico limitado a 255 caracteres, texto sin limitación y fecha; un valor por defecto y una descripción de la variable.

En la definición de las variables en ODI se puede establecer un valor directamente, pero también asociar una *query* en SQL que devuelva un único valor. Esta última opción tiene gran utilidad ya que permite actualizar el valor asociado a la variable en tiempo de ejecución.

En cuanto a los procedimientos, se definen como una serie de comandos que pueden ser ejecutados por un agente. Su uso debe limitarse a la realización de operaciones que no pueden ser efectuadas en los *mappings*; no obstante, pueden ser reutilizados. A diferencia de los *mappings*, los procedimientos requieren el desarrollo del código al completo. De esta manera se permite considerar scripts que se ejecuten dentro de la plataforma de ODI, en vez de fuera de la misma.

2.3.1.6. Paquetes y escenarios

Cada paquete corresponde a una unidad de ejecución en ODI, consistiendo estos en una secuencia de varios pasos organizada. Esta secuencia se organiza de manera que se sigue un esquema ramificado en el que se distinguen las ejecuciones correctas de las erróneas.

El paquete, antes de ser ejecutado, debe ser montado como un diagrama de manera similar a la construcción de los *mappings*. Dentro de este diagrama se pueden encontrar tanto los *mappings* desarrollados como variables, procedimientos, asignaciones,

escenarios y funciones propias de ODI. Todo ello ha de ser conectado de manera que se establezca el orden del flujo de datos en la ejecución, como se puede apreciar en el ejemplo que se visualiza en la figura.

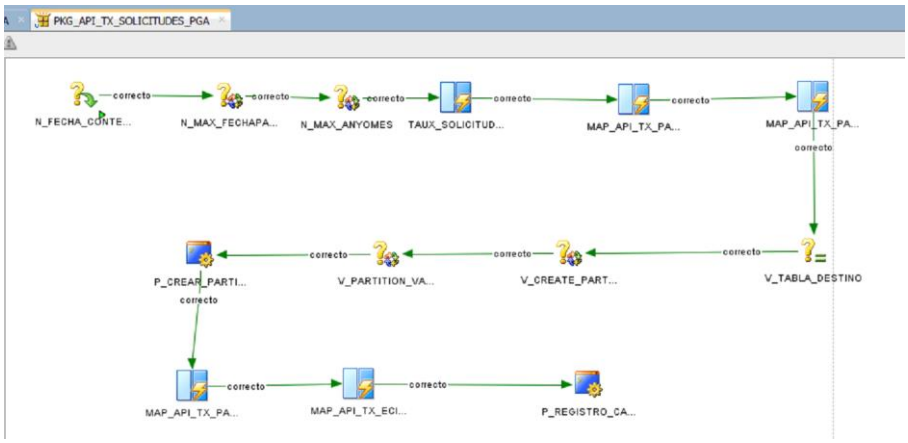

Figura 4. Ejemplo de un paquete en ODI

La manera de añadir variables en un paquete dependerá del objetivo propio de dicha variable. Se distinguen cuatro formas diferentes:

- Declarar variable. Declara explícitamente la variable utilizada.
- Refrescar variable. Refresca el valor almacenado en la variable mediante la ejecución de la *query* asociada.
- Definir variable. Permite establecer el valor de la variable antes de la ejecución bien asignando un valor concreto, bien incrementando el valor por defecto.
- Evaluar variable. Compara el valor actual de la variable con un valor dado. Como resultado, la secuencia de ejecución del paquete se ramifica dependiendo de si la condición se cumple o no.

Una vez se ha construido el paquete se pasa a la generación del escenario, el cual constituye la versión compilada del paquete. El lanzamiento de la ejecución se realiza siempre sobre el escenario generado.

2.3.2. Oracle SQL Developer

Para crear las nuevas tablas destino en las que se almacenarán los datos correspondientes de cada mapping desarrollado se ha utilizado SQL Developer en su versión 18.3.0. Adicionalmente, durante el proceso de pruebas y validación se usó de esta herramienta para lanzar consultas de diversos niveles de complejidad.

Oracle SQL Developer es una interfaz gráfica ofrecida por la compañía Oracle de manera gratuita. Esta permite trabajar con bases de datos no solo de Oracle sino también de terceras empresas, facilitando herramientas para arquitectos, administradores, desarrolladores y modeladores de datos. Fundamentalmente permite explorar, crear, borrar y modificar objetos de bases de datos a través de scripts programables en SQL y PL-SQL.

En nuestro caso nos centramos únicamente en las funciones para desarrolladores, entre las que se encuentran editores para programar en SQL, ejecución de consultas o creación y modificación de estructuras de datos.

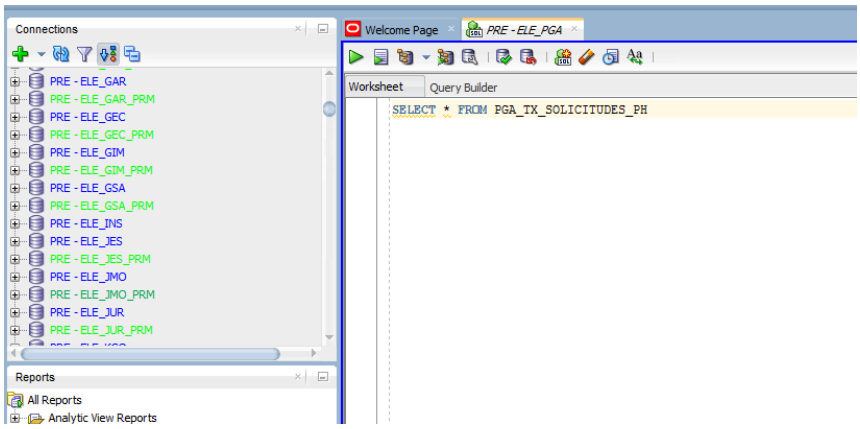

Figura 5. Interfaz gráfica SQL Developer

Como se puede ver en la interfaz gráfica del programa (*Figura 5*) hay dos ventanas principales. La ventana de la izquierda se encuentran las conexiones a las diferentes bases de datos corporativas. Entre ellas se distingue, para cada aplicación, la conexión a los datos disponibles en PRE, que mantiene siempre la misma contraseña, y la de los datos disponibles en PRO, para la cual se necesita generar una nueva clave de acceso cada 24 horas.

A la derecha de la imagen se encuentra la segunda ventana mencionada conocida como la hoja de trabajo. En ella se escribe el código SQL y los scripts a ejecutar. Una vez se ha ejecutado una consulta, aparece en la interfaz una tercera ventana con los registros que devuelve dicha consulta.

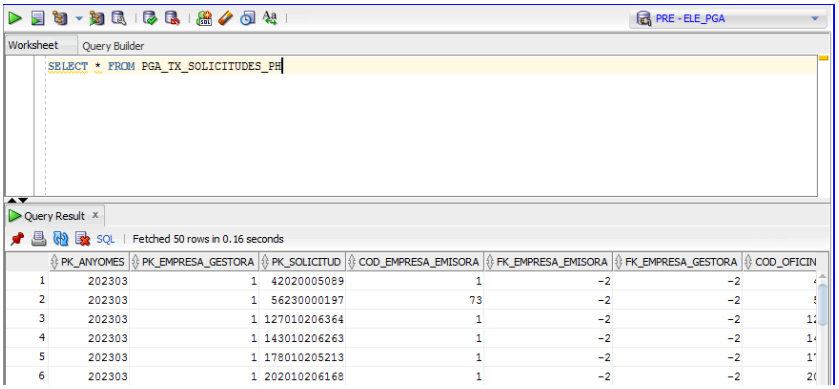

Figura 6. Resultado de la ejecución de una query en SQL Developer

En ocasiones en las que se necesita conocer la estructura de los almacenes de datos SQL Developer permite abrir una ventana adicional en la que se muestran los nombres de todos los campos o atributos, las restricciones sobre estos, claves primarias y foráneas, particiones y los roles asignados. Para acceder a esta visualización de la estructura de cualquier tabla basta con navegar en la ventana de conexiones y hacer doble click sobre la tabla deseada. Otra manera habitual de abrirla es seleccionando el nombre completo de la tabla desde la hoja de trabajo y clicando sobre '*Open declaration*'. Adicionalmente, con esta vista se permite editar manualmente los atributos deseados, siendo así una interfaz intuitiva, fácil de comprender y utilizar.

Welcome Page | PRE - ELE_PGA | PGA_TX_SOLICITUDES_PH

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PK_ANYOMES	NUMBER(6,0)	Yes	(null)	1	(null)
2	PK_EMPRESA_GESTORA	NUMBER(5,0)	Yes	(null)	2	(null)
3	PK_SOLICITUD	NUMBER(13,0)	Yes	(null)	3	(null)
4	COD_EMPRESA_EMISORA	NUMBER(5,0)	Yes	(null)	4	(null)
5	FK_EMPRESA_EMISORA	NUMBER(5,0)	Yes	(null)	5	(null)
6	FK_EMPRESA_GESTORA	NUMBER(5,0)	Yes	(null)	6	(null)
7	COD_OFICINA	NUMBER(5,0)	Yes	(null)	7	(null)
8	COD_MODALIDAD	NUMBER(3,0)	Yes	(null)	8	(null)
9	COD_CUENTA	NUMBER(7,0)	Yes	(null)	9	(null)
10	COD_TIPO_SOLICITUD	VARCHAR2(3 CHAR)	Yes	(null)	10	(null)
11	COD_TIPO_OPERACION	VARCHAR2(2 CHAR)	Yes	(null)	11	(null)
12	IND_GENERICO	NUMBER(1,0)	Yes	(null)	12	(null)
13	COD_PROD_NORMALIZ_TGC	NUMBER(5,0)	Yes	(null)	13	(null)

Figura 7. Vista de tabla en SQL Developer

2.3.3. Qlik

Para la explotación de los datos una vez se han sometido a las transformaciones necesarias y han sido cargados en la tabla destino se ha utilizado la herramienta Qlik (semantic systems). Esta prometedora plataforma es usada habitualmente para extraer el valor oculto de los datos que generan las empresas mediante su visualización, permitiendo así una mejor toma de decisiones y salvaguardando las políticas de seguridad de la empresa.

El principal beneficio que brinda Qlik se basa en la analítica de datos, ya que permite realizar una interpretación visual sencilla, incluyendo gráficos e informes que facilitan la comprensión de la realidad de la empresa mediante una imagen completa. De esta manera cualquier usuario puede tomar decisiones mejor informadas, además de más rápidas y precisas.

Esta plataforma puede ofrecer numerosas ventajas a la compañía, entre las que se destaca la integración total de datos procedentes de cualquier fuente y de cualquier tamaño. Qlik pone a disposición de los usuarios una amplia gama de APIs RESTful abiertas que han desarrollado con el objetivo de que funcionen con diferentes tecnologías web y aplicaciones. De esta forma, facilitan a los desarrolladores la integración de datos de manera flexible y sencilla, consiguiendo satisfacer las necesidades de cualquier negocio.

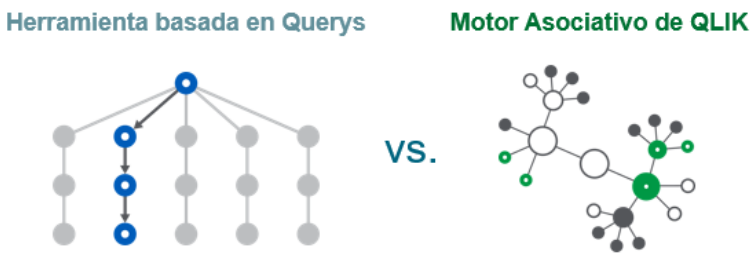

Figura 8. Comparativa enfoque tradicional y motor asociativo

Por otra parte, Qlik cuenta con un motor asociativo que relaciona cualquier parte con el resto, haciendo frente al esquema clásico basado en el modelo pregunta-respuesta, como se puede observar en la *figura 8*. Con esto se consigue evitar el lanzamiento de

numerosas *queries* hasta dar con la respuesta deseada. La manera de presentar la información al usuario en esta plataforma pasa por una codificación de tres colores: verde para resaltar la selección, blanco para indicar los datos asociados a dicha selección y gris para aquellos datos sin relación.

Qlik ofrece una amplia variedad de productos en función de los objetivos de visualización de la empresa. Entre ellos se destaca Qlik Sense como el más utilizado y conocido, al cual describen como "una solución completa de analítica de datos" (Qlik, 2023). Su plataforma en la nube de alto rendimiento, junto con su motor de analítica asociativa y una IA sofisticada, ayudará a crear un entorno corporativo realmente basado en datos.

Por su parte, Qlik View permite desarrollar y producir rápidamente aplicaciones y cuadros de mandos de analítica guiada interactivos. Con esta plataforma se facilita el uso de datos disponibles en las empresas, además de poner el *Business Intelligence* a disposición de cualquier empresa.

Finalmente, Qlik AutoML ofrece una analítica más compleja y diferente, ya que permite generar modelos de aprendizaje automático fácilmente a través de una interfaz de usuario intuitiva y sin código. De esta manera cualquier usuario será capaz de realizar un análisis tanto predictivo como de escenarios hipotéticos.

Junto a esta herramienta se encuentran otras buenas opciones líderes en el mercado actual como Tableau. Ambas ofrecen numerosas capacidades de visualización y análisis de datos, además de la integración de estos de diversas fuentes. No obstante, Tableau sigue el enfoque tradicional de conexión directa a fuentes de datos, dejando a Qlik y su motor asociativo en una mejor posición.

2.3.4. IBM Tivoli Workload Scheduler (TWS)

Para la gestión de la ejecución de cargas diarias de este proyecto se ha utilizado IBM Tivoli Workload Scheduler (en adelante TWS) (Gucer, Jones, Jojic, Patrick, & Bain, 2003), herramienta que ofrece soluciones a problemas de gestión de la carga de trabajo de producción de una empresa. Entre las principales ventajas que esta herramienta aporta a la compañía se encuentra la optimización de los recursos disponibles y la maximización del rendimiento, ya que permite automatizar, controlar y planear la totalidad de los procesos. Además, facilita la posibilidad de intervención manual en todo momento.

En cuanto a los beneficios que ofrece este planificador, se pueden agrupar en tres categorías diferentes. En primer lugar, se encuentran las soluciones de negocio, que engloban diversas funcionalidades como son dirigir la producción según los objetivos de negocio, gestionar eficientemente un mayor número de procesos, automatizar los procesos de producción consiguiendo una mayor productividad corporativa y obtener información sobre las cargas de trabajo tanto actuales como futuras.

Por otra parte, se encuentra el crecimiento de negocio, el cual incluye una interfaz abierta ejecutable en diversos sistemas, la utilidad de absorber una carga de trabajo creciente sin necesidad de aumentar el número de recursos, la simulación de cargas futuras de trabajo y la integración con diferentes plataformas y sistemas.

Finalmente, la productividad de usuario engloba no solo una fácil implementación y respuesta inmediata a peticiones en tiempo real, sino también la automatización de los

procesos de limpieza y recuperación de datos y una guía con información detallada para detectar y corregir errores.

2.3.4.1. Conceptos básicos de TWS

El planificador de trabajo de IBM trabaja con los siguientes objetos:

- *Job*. Este objeto constituye una unidad de trabajo y especifica una acción, lo cual queda preestablecido por las reglas de gobernanza de datos. En este proyecto, cada *job* corresponde a la carga de datos en una tabla del almacén de datos. Los *jobs* pueden ejecutarse tanto individualmente como formando parte de un *Job Stream*.
- *Job Stream*. Se define como una secuencia de *jobs* que son ejecutados en función a ciertas órdenes de prioridad y restricciones de dependencia. Estas últimas pueden ser bien internas, cuando se dan entre *jobs* del mismo *job stream*, bien externas, si se producen entre *jobs* de diferentes secuencias. Los *job streams* surgen de la necesidad de pertenencia de los *jobs* a alguna secuencia, de manera que se puedan planificar las ejecuciones y automatizar el proceso. No obstante, como se ha mencionado antes, los *jobs* también pueden ser ejecutados de manera individual.

Estos objetos se planifican o configuran en función a un ciclo de ejecución o tiempo entre las diferentes ejecuciones. Se encuentran las siguientes posibilidades:

- Diario. Los procesos se ejecutan siguiendo un periodo diario según la frecuencia especificada, la cual puede establecerse como días naturales, días laborales o cada x días.
- Semanal. Los procesos se ejecutan semanalmente de manera periódica. Dentro de este ciclo pueden ejecutarse varios días de la semana o solo uno.
- Mensual. Los procesos se ejecutan con carácter mensual, que, nuevamente, puede configurarse de diferentes maneras, como por ejemplo meses alternos o cada dos meses.
- Anual. Los procesos siguen un periodo anual, que puede estar configurado de manera similar al anterior. Por ejemplo, podría establecerse a cada año o años alternos.
- Simple. Con este tipo se permite especificar manualmente los días y tiempos de ejecución del *job* o *job stream*.
- Exclusivo. En este ciclo se puede pueden especificar los tiempos en los que no se permite ejecutar un proceso.
- Inclusivo. Ciclo de ejecución en el que se especifican los tiempos en los que se planea la ejecución de un *job* o *job stream*.

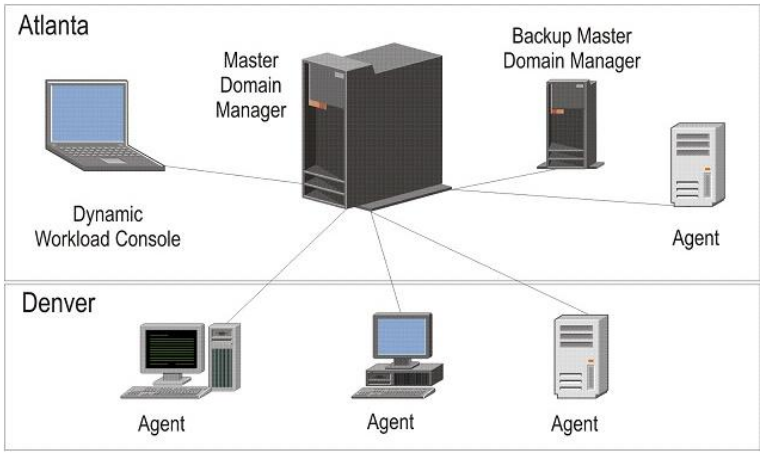

Figura 9. Ejemplo red de ejecución de TWS

En la figura siguiente se visualiza una red que integra varias estaciones de trabajo en las que los procesos son ejecutados. Dichas redes se dividen en dominios, siendo el manager el maestro de dominio, es decir, aquel que actúa de centro y es el encargado de realizar las comunicaciones oportunas con el resto de los agentes de la red. Adicionalmente, se conecta con la base de datos y lleva un control de los logs o evidencias de comportamiento del sistema y reportes de la red.

2.6. Recapitulación

Para cerrar el capítulo de contexto, se enumera a continuación el flujo que siguen los datos a través de las herramientas tecnológicas utilizadas y el objetivo de su uso.

En primer lugar, el caso práctico que expone este proyecto se basa en el servicio ofrecido por una consultoría a una empresa de ámbito financiero. Esta última trabaja en el sector inmobiliario con empresas promotoras como clientes, a las cuales facilita préstamos hipotecarios y la posibilidad de acordar contratos de *confirmings*. Por su parte, las promotoras presentan avales al banco para garantizar el pago de los préstamos, los cuales también serán almacenados como datos.

Una vez se han recopilado todos los datos necesarios, dicha empresa del sector bancario los facilita a la consultoría, de manera que se inician los primeros procesos ETL necesarios para poder trabajar con los datos. El siguiente paso consiste en la construcción de las estructuras necesarias con Oracle Data Integrator, hasta llegar a una ejecución perfecta del paquete que da como resultado la tabla destino. Esta fase se apoya en SQL Developer para la ejecución de consultas simples y pruebas de volumetría y datos. Finalmente, esta tabla destino está lista para ser consumida en Qlik, la herramienta de visualización utilizada en este proyecto. Adicionalmente, se lleva un seguimiento, gestión y planificación de la ejecución de cargas diarias mediante IBM Tivoli Workload Scheduler.

3. Análisis del problema

3.1. Marco legal

El caso práctico que expone este trabajo es el resultado de la colaboración con una compañía de ámbito financiero. Por motivos de confidencialidad se mantiene el nombre de dicha compañía anonimizado a lo largo de todo el proyecto.

En cuanto a los datos con los que se trabaja, es el cliente el encargado de recopilar los datos de sus propios clientes. El equipo de trabajo de la consultoría tiene acceso a estos datos en su totalidad y, como se explica a continuación en el capítulo de metodología, recae en ellos la responsabilidad de limpiar los datos y almacenarlos correctamente para su posterior explotación. Una vez más, por temas de protección de datos no está permitido el acceso a estos datos por parte de alguien externo al departamento dedicado de la consultoría.

A lo largo de los capítulos de resultados y visualización se comenta el uso de portales y aplicaciones, como por ejemplo en la carga de datos, para los cuales no se especifica el nombre comercial, ya que constituyen productos propios de la compañía cliente.

Asimismo, se incluyen imágenes del cuadro de mandos construido en el capítulo de visualización. Estos gráficos son modificados para constituir únicamente un ejemplo visual de la representación empleada con la estética definida, aunque los valores mostrados no son los reales. De esta manera se mantiene la protección de datos comentada.

3.2. Marco ético

En cuanto al marco ético, cabe señalar la presencia de atributos sensibles dentro de las bases de datos utilizadas, como son el nombre completo de los clientes, su dirección o su DNI, entre otros. Es por ello por lo que se ha adoptado un enfoque que preserva la privacidad y protección de datos personales como se estipula en el Reglamento General de Protección de Datos. Para ello se han anonimizado dichos atributos sensibles a la hora de exponer este trabajo, además de asegurar el consentimiento de las partes involucradas en el desarrollo del mismo.

4. Metodología

En este cuarto capítulo se presenta y describe la metodología empleada en el trabajo. Para ello, en primer lugar se describe la muestra de datos, incluyendo tanto el origen de estos como las distintas variables presentes. Posteriormente se desarrolla el preprocesamiento de los datos antes de ser usados para el caso práctico que se expone. Dentro de este procesamiento se distinguen las tres capas que tienen lugar desde la ingesta de datos hasta su consumo.

4.1. Muestra

La amplia muestra de datos utilizada para el desarrollo de este caso práctico constituye un conjunto representativo de varias promociones inmobiliarias gestionadas por un cliente del sector bancario con el que trabaja esta consultoría. Esta muestra se almacena en las bases de datos de la compañía, donde el acceso está restringido, por lo que cada nuevo empleado necesita pedir los permisos necesarios para poder trabajar con ellas.

Al llegar un nuevo empleado al proyecto, tras aportar la información necesaria, se le otorgan los permisos de acceso al *data pool* o repositorio de datos proporcionado por el cliente en cuestión y se establecen las conexiones oportunas con las bases de datos. Dentro de la compañía, este data pool está estructurado en diferentes bases de datos en función al contenido de estas, enfocadas a un proyecto concreto. A su vez, se distinguen dos ramificaciones por cada proyecto, dependiendo de la capa del proceso de la cual son resultado.

Para el desarrollo de este caso práctico solo se hará uso de una parte de estas bases de datos, pero es conveniente disponer de acceso a todas ellas para futuros enfoques o ramificaciones del proyecto. En total, ECIAPI se construye extrayendo datos de doce ELEs diferentes. Las tablas más importantes de algunas de ellas recogen datos históricos desde el año 2018, lo que supone un gran número de registros ya que estos datos se actualizan diariamente. No obstante, el proyecto desarrollado data del año 2024, por lo que solo se usarán datos desde enero hasta el mes actual de este año, reduciendo así el tamaño de la muestra considerablemente. En cualquier caso, el número de registros filtrando por el año sigue ascendiendo a millones en algunas de las tablas más importantes.

4.1.1. Origen de datos

El cliente ya mencionado se encarga de recoger una serie de datos de las promotoras que son sus clientes, junto con otros conjuntos de datos relacionados con los diferentes centros físicos de la entidad financiera.

El proceso de recopilación comienza con la colaboración entre el equipo de la consultoría dedicado a este proyecto y el banco cliente. Este último es el encargado de recolectar y facilitar todos los datos e información relevante para el estudio, proporcionando acceso al *data pool* ya mencionado, donde se almacena dicha información. Esta última se encuentra repartida en multitud de tablas origen simples, que serán tratadas para, posteriormente, construir una única tabla destino objeto de explotación en este caso práctico.

Para realizar la transferencia se establecen mecanismos seguros y eficientes, de manera que se garantiza la integridad y confidencialidad durante el proceso. Por otra parte, se establecen políticas acerca del alcance y la periodicidad en la entrega de datos, acordando la frecuencia de actualización o las variables necesarias, entre otros. Sin embargo, no existen políticas estrictas en cuanto al formato y la naturaleza de los datos, ya que es trabajo del equipo dedicado realizar los procesos ETL necesarios antes de que estos datos puedan ser explotados.

4.1.2. Variables

Para el desarrollo de este caso práctico, como se ha mencionado anteriormente, solo se hace uso de un conjunto de todas las bases de datos disponibles en el *data pool*. Los datos resultantes de la *Enterprise Layer* (ELE) son almacenados en diferentes bases de datos. Para la construcción de la aplicación objeto de este caso práctico, la tabla destino tendrá su origen en varias de estas ELEs. En otras palabras, los datos iniciales tienen diversas procedencias con multitud de variables, por lo que se sintetiza este subapartado a la explicación de las variables almacenadas en la tabla destino.

Entre estas variables, que corresponden a aquellas conseguidas tras la construcción de la aplicación, se encuentran las siguientes:

- PK_ANYOMES: variable de tipo numérico que se usa como clave primaria para, junto a otras, identificar el contrato. Almacena el año y el mes de este en el formato 'aaaamm'.
- COD_CONTRATO: variable de tipo numérico que almacena el código que identifica cada contrato.
- COD_OFICINA: código que identifica la oficina em la cual se ha procesado dicho contrato.
- DES_CENTRO: variable de tipo alfanumérico para la descripción o nombre del centro en el que se efectúa el contrato.
- COD_CONTRATO_ORIGEN: código identificador del contrato origen. En definitiva, un *COD_CONTRATO* se forma concatenando empresa, modalidad, área y *COD_CONTRATO_ORIGEN*.
- COD_MODALIDAD: valor numérico con el código de la modalidad del contrato. Con él se puede saber a qué ámbito pertenece el contrato (morosidad, solicitudes, etc.).
- COD_AREA_CONTRATO_RELACIONADO: valor numérico con el código del área del contrato. Con él se puede saber a qué ámbito pertenece el contrato (morosidad, solicitudes, etc.).
- NIF_PERSONA: identificador de la persona que firma el contrato.
- RAZON_SOCIAL: varchar con el nombre de la empresa.
- IMPO_SOLICI_CONTRAVALORADO: se incluyen todos los importes, ya sean de préstamos, cuentas de crédito, avales o confirming.
- PCT_DIFERENCIAL_A_0: porcentaje de diferencial promotor.
- DES_LOCALIDAD: variable alfanumérica que hace referencia a la descripción o nombre de la localidad en la que se halla el centro.
- DES_PROVINCIA: descripción o nombre de la provincia en la que se encuentra el centro.

- NUM_FINCAS_FISICAS_TASA: número de fincas físicas que recoge la tasación en cuestión.
- FECHA_ULT_MODIF: variable de tipo fecha que guarda la fecha en la cual se realizó la última modificación del contrato.
- FECHA_CARGA: variable de tipo *timestamp*, alberga la fecha en la que se ejecuta la carga de datos.
- COD_FLAG_ORIGEN: etiqueta que puede tomar un valor entre dos para distinguir aquellos contratos del CENI de las promociones.

Todas estas variables se obtienen y juntan en una tabla destino con el objetivo de explotarlas visualmente, como se explicará en el capítulo 7.

4.2. Preprocesamiento de datos

Como se ha comentado anteriormente, las tablas origen de datos necesitan ser procesadas antes de poder ser explotadas. El flujo de los datos comienza en su recolección y transferencia a la empresa como varias tablas origen, sobre las que se aplican una serie de procesos ETL para limpiar y unificar esos datos. A continuación, se lleva a cabo un minucioso estudio de la información que se desea extraer de estos datos para su posterior visualización, así como de la mejor manera de obtener dicha información.

Una vez están claros los objetivos del proyecto, se realizan las segundas transformaciones al tiempo que se establecen las relaciones necesarias entre los campos útiles de cada tabla para, finalmente, construir una única tabla destino que recoja toda la información que se desea explotar con las herramientas y tecnologías de visualización mencionadas en el capítulo de contexto, apartado 2.3.

Más concretamente, el proceso de transformación de los datos desde la ingesta hasta el consumo de los mismos consta de tres capas en el siguiente orden: *Staging Layer*, *Enterprise Layer* y *Consumer Layer*. A continuación se introducen todas ellas brevemente; sin embargo, las dos primeras quedan fuera del alcance del proyecto.

4.2.1. Staging Layer (SLE)

Se conoce la *Staging Area* como la puerta de entrada al *Data Warehouse* o almacén de datos. En esta capa se realiza la ingesta de todos los datos procedentes de origen sin apenas transformaciones. Puede verse como un almacenamiento preliminar de los datos, donde estos últimos permanecen a la espera de ser transformados.

Dado el caso que se expone en este trabajo, esta capa corresponde a la consolidación de todos los datos en el entorno de Oracle Exadata. Estos datos pueden proceder de diferentes aplicaciones, siendo en su mayoría procedentes de ficheros de tipo Excel dentro de lo que concierne a este proyecto.

4.2.2. Enterprise Layer (ELE)

La *Enterprise Layer* corresponde a la capa de negocio. Es en esta capa donde se transforman los datos ya almacenados en la *Staging Layer* mediante procesos ETL. Tras esto, se almacenan en la base de datos corporativa que corresponda dentro del *data pool*.

A su vez, se pueden distinguir dos fases dentro de esta capa. En la primera fase los datos son tratados, limpiados y transformados siguiendo un modelo relacional. Dentro de estas transformaciones se encuentran la sustitución de valores nulos por '0' o '-1', o la comprobación de posibles fechas inexistentes. Respecto a la segunda sección, se incluye la aplicación de cálculos y métricas con el objetivo de generar un modelo dimensional en estrella que será utilizado posteriormente de manera que se consiga generar la información necesaria acorde con las necesidades de la organización en el proyecto.

4.2.3. Consumer Layer (ECI)

Se trata de la capa de consumo. Es en esta capa donde se va a desarrollar el caso práctico que se expone en este trabajo, del cual se hablará más en detalle en el siguiente capítulo. Constituye la capa de acceso al almacén de datos por parte del usuario final.

En la *Consumer Layer* se pueden encontrar las diferentes bases de datos departamentales. Para su construcción se accede a los datos tratados en la capa de negocio, los cuales pasan por ODI para ser relacionados, agregados, calcular métricas o incluso efectuar alguna transformación adicional. El resultado de esta capa contiene estructuras pensadas para facilitar la recogida y consumo de los datos necesarios para la realización de informes, cuadros de mandos y reportes, entre otros.

5. Resultados

En este capítulo se presenta la construcción de la aplicación desarrollada para el almacenamiento de los datos sobre promociones inmobiliarias. Para ello, se explica más en detalle la capa de consumo ya mencionada en el apartado 4.2, durante la cual se da este caso práctico. Se habla tanto de la preparación de los datos como de los diferentes pasos seguidos hasta llegar al resultado final, con el que se consigue una tabla destino con la información necesaria para ser explotada posteriormente.

5.1. Construcción de ECIAPI

Para la construcción de la aplicación final se han usado varias bases de datos almacenadas en diferentes ELEs tras pasar por la capa de negocio. Tomando todas estas como orígenes para construir diversos *mappings*, se ha conseguido una tabla final con los atributos o variables comentadas anteriormente. En la imagen siguiente se pueden distinguir todas las ELEs utilizadas para la construcción de ECIAPI.

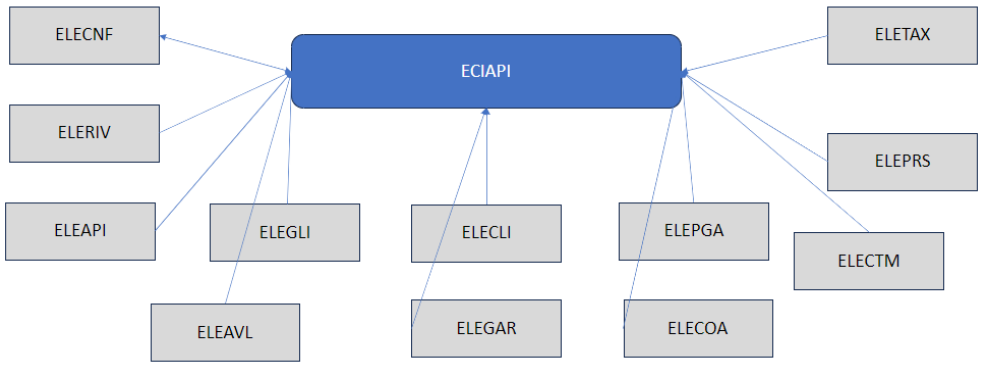

Figura 10. Datapool orígenes ECIAPI

El punto de partida para la construcción de esta aplicación en ODI es una consulta SQL que devuelve los registros oportunos con sus correspondientes variables. Para hacer este proceso más sencillo, se ha dividido en subconsultas, lo que se traduce a ODI en varios *mappings*. Finalmente, se construye un paquete que une todas las variables y *mappings* en el orden adecuado para la generación e inserción de registros en la tabla final.

A continuación se explican dos pasos previos, la creación de las tablas vacías y la definición de algunas variables, para dar paso a la construcción de todos los *mappings* y el paquete final. Dentro de cada *mapping* se comentan algunos filtros con valores concretos, aunque por simplicidad, además de no considerarse relevante para la explicación, no se entra en detalle de los valores dados.

5.1.1. Creación de tablas

Antes de comenzar a trabajar con ODI es necesario crear una tabla por cada partición de la consulta general. En otras palabras, al dividir la consulta en varias partes y, por consiguiente, varios *mappings*, cada una de estas partes resulta en una serie de registros con sus atributos correspondientes. Esto implica la creación de una tabla donde almacenarlos como pasos intermedios antes de llegar a la tabla destino final.

Dentro de cada *mapping* se distinguen una o varias tablas origen y una única tabla destino donde se almacenarán los datos, por lo que debe estar bien definida antes de la inserción de registros. Más adelante se explica qué información se almacena en cada una.

Tras establecer claramente las distintas partes de la consulta, se estudia qué atributos devuelven. Con esta información, se realiza una operación 'CREATE TABLE' en SQL Developer dentro de la base de datos de ECIAPI, indicando tanto el nombre de la tabla o parte de la consulta, como todos los atributos que almacena junto con su tipo de datos.

Una vez se ha ejecutado esta operación para cada una de las subconsultas se realiza un procedimiento equivalente por tabla en ODI, de manera que se cargan las tablas creadas en bases de datos en ODI, nuestra interfaz de trabajo para los siguientes pasos.

Este procedimiento constituye una unidad de ejecución creada para todos los proyectos desarrollados en esta compañía, por lo que ya se dispone del programa, almacenado en la carpeta común de proyecto dentro de la aplicación. Con ella se importa la estructura de las tablas creadas anteriormente desde la base de datos, además de crearse los correspondientes metadatos en el modelo. Con esto, ODI reconoce y utiliza estas estructuras de datos ya existentes en sus procesos de integración y transformación.

5.1.2. Definición de variables

Durante la construcción de los *mappings* y previo a la construcción del paquete final surgen necesidades en cuanto a las variables definidas. Algunas de ellas permiten filtrar campos en base a un valor concreto dado o al cálculo matemático de una variable, mientras que otras son necesarias para la correcta ejecución del paquete y la inserción de filas en la tabla destino. En este caso, se definen las siguientes variables de proyecto:

- N_FECHA_CONTENIDO. Variable introducida en el momento de ejecución del paquete. Se indica la fecha para la que se desea ejecutar en formato 'AAAAMMDD'.
- N_MAX_ANYOMES. Variable calculada como la máxima fecha en formato 'AAAAMM' que se tiene en la tabla ELE_COA.COA_N2_TX_OFERTAS_PD, usada como origen en algún *mapping*. Se usa para filtrar datos con un FK_ANYOMES igual a este valor.
- N_MAX_ANYOMES_ELE_GAR. Variable calculada de manera similar a la anterior, usada para filtrar los datos según su PK_ANYOMES, pero tomando la tabla ELE_GAR.GAR_TM_CERTF_EFIC_ENERG_PH para el cálculo.
- N_MAX_FECHAPART_COA_OFERTAS. Variable de tipo texto calculada como la máxima fecha de partición en formato 'DD/MM/AA' que se tiene de la tabla ELE_COA. Se usa para filtrar datos por el campo PK_FECHAPART.
- V_CREATE_PARTITION_VALUE. Variable calculada como la fecha actual en la variable N_FECHA_CONTENIDO en formato 'AAAAMM' sumándole un mes. Esto indica la fecha final de la partición que se crea sin incluir esta última.
- V_PARTITION_VALUE. Variable que almacena la fecha de la variable N_FECHA_CONTENIDO en formato 'AAAAMM'. Indica la primera fecha de los datos de la partición que se crea.
- V_TABLA_DESTINO. Variable que almacena el nombre de la tabla destino donde se debe crear la partición al ejecutar el paquete. Se usa directamente dentro del mismo.

5.1.3. Tabla auxiliar (MAP_API_TAUX_SOLICITUDES_PGA)

En primer lugar, la consulta SQL de todo ECIAPI repite a lo largo de todo el proceso una pequeña partición de una tabla con una serie de filtros. Por comodidad y simplicidad se decide generar una tabla auxiliar con esta parte de la *query* que se usará en los *mapppings* siguientes.

Para ello se construye un primer *mapping* que almacena el resultado de esta *query* en la tabla auxiliar, la cual podrá ser utilizada directamente en los siguientes mapeos. Esta tabla auxiliar tiene como origen principal ELE_PGA.PGA_TX_SOLICITUDES_PH, donde se almacena información mensual referente a las solicitudes del producto activo de diferentes operaciones.

Esta información es filtrada bien para que el código delimitador coincida con una selección de ofertas obtenida a través de la tabla ELE_COA.COA_N2_TX_ofertas_PD, bien para que el código de oficina corresponda a aquellos centros que están registrados en ELE_CTM.CTM_TM_CENTROS con un código de tipo de oficina ‘4’, es decir, aquellos que corresponden con centros de empresas de negocio inmobiliario o CENI.

Aplicando estos filtros junto a otros en términos de fechas se obtienen los registros de aquellos contratos referentes a operaciones formalizadas. Por otra parte, es importante tener en cuenta que un contrato puede haber sido formalizado en viernes pero no registrarse hasta el lunes siguiente la fecha de última modificación.

Una vez se ha construido este *mapping* auxiliar podemos hacer uso de la tabla resultante en los *mappings* posteriores, por lo que en adelante se hará referencia a este como ‘TAUX_SOLICITUDES_PGA’. En resumen, esta tabla almacena los campos referentes a códigos, como son el del contrato de origen, contrato, modalidad, área del contrato relacionado, oficina, cliente, delimitador y operación, junto con la fecha de última modificación y el importe de la solicitado contravalorado.

5.1.4. Mapping parte 1 (MAP_API_TX_PARTE1)

La primera parte consiste en la unión de todos los contratos, referentes a préstamos, avales y confirmings. A esto se le añaden diversos cruces con varias tablas para completar la información extraída.

En primer lugar, se obtienen los contratos existentes de préstamos, para lo cual se hace uso de la tabla auxiliar explicada anteriormente aplicándole un filtro que permita seleccionar solo aquellos registros con valor ‘3’ o ‘14’ como COD_AREA_CONTRATO_RELACIONADO. Por su parte, los avales requerirán un filtro similar que seleccione el valor ‘57’ para este campo, junto con un código delimitador correspondiente a 'K0003A' o 'K0003R'.

En cuanto a los confirmings, se requiere una operación ligeramente más compleja. Para ello, se seleccionan los registros de la tabla auxiliar con COD_AREA_CONTRATO_RELACIONADO igual a ‘32’, además de COD_CNAE_OPERACION como ‘4110’, ‘4121’ o ‘4122’. Todo esto se traduce en ODI como un filtro donde se indican los valores concretos en el campo de la expresión. Tras esto, se añade un componente *join* de ODI, el cual permite unir dos tablas, indicando una unión interna. Es decir, se seleccionan únicamente aquellos registros para los que coincide la condición indicada en ambas tablas. En este caso, se une la tabla auxiliar

mediante el código del contrato con ELE_CLI.CL_TM_CONTRATOS_PH, donde se almacena información principalmente acerca de los clientes y, en este caso, los detalles de los contratos llevados a cabo.

Adicionalmente, se añaden una serie de filtros a esta nueva tabla, entre los que se destaca la selección del código del centro gestor entre aquellos centros que, una vez más, se registran con tipo de oficina '4'. Es necesario volver a establecer este filtro ya que en TAUX_SOLICITUDES_PGA solo se establece como una condición a cumplir entre dos, por lo que podrían seleccionarse registros que no la cumplieran. Este último filtro se traduce en ODI en una unión interna con la tabla ELE_CTM.CTM_TM_CENTROS mediante el código del centro, ya que no se dispone de este campo en ELE_CLI.

Tras estas tres subselecciones de la tabla auxiliar correspondientes a préstamos, avales y confirmings, se realiza una unión de todas ellas. En este caso se debe emplear el componente de ODI llamado 'Definir', estableciendo el operador en 'UNION'. Este se diferencia del *join* en la posibilidad de unir más de dos orígenes a la vez, además de no ser necesario indicar una expresión para la condición de unión, sino que se indica una expresión para cada uno de los *inputs* por cada campo resultante.

A la unión resultante le siguen varios *left joins* de manera que se agrega a la información ya extraída descripciones o nueva información relevante. Esta se almacena como campos nuevos para aquellos contratos que dispongan de dicha información. En primer lugar, se extrae la descripción del centro de ELE_CTM.CTM_TM_CENTROS nuevamente. Tras esto, se obtiene el NIF y el nombre del cliente a través de ELE_CLI.CL_TM_PERSONAS_NIF, filtrando por COD_NUMPERSO. De ELE_RIV.RIV_TX_REVISION_INTERES_PH se extrae el PCT diferencial. El código de tasación y el código de tasación interno junto con el número de fincas físicas que dispone ese cliente se obtienen de ELE_API y ELE_TAX respectivamente. Mediante las tablas de ELE_GAR referentes a eficiencia energética y garantías hipotecarias se agregan los campos código de contrato certificado, código de finca certificada, COD_TIPO_CERTIFICADO_IEE y COD_TIPO_CALIF IEE, primario y secundario.

Tras todas estas uniones se realiza una operación *group by*, lo que en ODI se traduce en el componente *aggregate*. Con esto se elimina cualquier duplicado que haya podido surgir al agrupar los registros por todos los atributos recabados hasta el momento. La salida de este *aggregate* se une con un último *left join* a la salida de otro *aggregate* para completar los detalles acerca de la ubicación. Tanto la localidad como la provincia de las fincas se obtienen de ELE_API, a lo que se aplica un máximo de manera que, para los registros duplicados con y sin valor en estos campos, seleccione únicamente aquellos para los que se tiene información. Junto con esto, se indica la agrupación por los atributos restantes.

El último *join* efectuado da como resultado una tabla con los campos y registros correspondientes a esta primera parte, la cual será integrada en posteriores *mappings*. Resumidamente, a algunos de los campos extraídos con TAUX_TX_SOLICITUDES_PGA se le añaden otros como la descripción del centro, el NIF y el nombre del cliente, localidad y provincia, el número de fincas en esa localización, códigos de contrato relacionado y certificado, tasación y finca.

5.1.5. Mapping parte D1 (MAP_API_TX_PARTED1)

Este *mapping* consiste en una ampliación de MAP_API_TX_PARTE1. Se separa en este segundo *mapping* con el objetivo de conseguir una visión más general y un entendimiento más claro, ya que la parte 1 resulta en un esquema bastante complejo.

Esencialmente, lo que se encuentra con el mapeo correspondiente a la parte 1 es una promoción. El objetivo de esta pequeña ampliación es determinar si el contrato está relacionado o no.

Para ello se incorpora como orígenes de este *mapping* la tabla resultado de la parte 1 y la tabla ELE_API.API_TM_REL_CONTRACTUALES_PH, donde se recoge con qué otros contratos se relaciona una promoción inmobiliaria.

Como resultado de MAP_API_TX_PARTED1 se obtiene la tabla PARTE_D1 tras efectuar una unión interna condicionada por la igualdad entre el código de contrato de la parte 1 y el código del contrato relacionado de la tabla de ELE_API. Con esto se almacenan únicamente aquellos contratos que sí están relacionados, dejando en PARTE_1 todos los contratos, relacionados o no.

Para diferenciar los contratos posteriormente se añade un nuevo campo como último paso, al que se le da el valor 'Promociones' para todos los registros de esta tabla. Este campo obtiene el nombre COD_FLAG_ORIGEN, que permitirá comparar los contratos relacionados o promociones con los contratos no relacionados u otras operaciones del CENI, extraídos a continuación.

En resumen, esta tabla almacena esencialmente los campos extraídos en la parte 1 junto con la etiqueta asignada a las promociones con contratos relacionados.

5.1.6. Mapping parte 2 (MAP_API_TX_PARTE2)

El objetivo de este *mapping*, al contrario que MAP_API_TX_PARTED1, es quedarse únicamente con aquellos contratos que no están relacionados o, en otras palabras, otras operaciones firmadas en el CENI que no son promociones inmobiliarias. Para ello se debe obtener el conjunto de todos los contratos y restarle o quitarle aquellos contratos guardados en PARTE_D1, es decir, aquellos contratos que sí están relacionados.

Este *mapping* en ODI vuelve a resultar algo más complejo, ya que, a pesar de corresponder con una idea principal sencilla, a la hora de pulir errores, falta de información y duplicados innecesarios se amplían los orígenes y uniones del mapeo con el fin de informar todos los campos correctamente.

Para traducir esta parte de la consulta en ODI es necesario recrear el *mapping* hecho para la tabla auxiliar pero estableciendo dos de los filtros a un valor diferente. En este caso se construye el diagrama en el propio mapeo de esta parte, ya que, a pesar de ser muy similar a la tabla auxiliar, solo se utiliza una vez, por lo que no se gana en términos de simplicidad si se construye un mapeo adicional.

Se comienza por construir un diagrama con la misma estructura comentada en el primer *mapping*, pero, en este caso, el componente que filtra los datos de ELE_COA.COA_N2_TX_ofertas_PD para hacer una selección de las ofertas se modifica. La TAUX original filtraba aquellas ofertas con un valor de '900520' o '900500' para el atributo AGRUP_PRESENT, tipos de agrupaciones en las ofertas que

corresponden a promociones inmobiliarias, excluyendo así el resto de ofertas. Como en este caso se buscan otras operaciones firmadas en el CENI, se selecciona el grupo de ofertas complementario al que se filtra en TAUX, es decir, todas las que no tengan los dos valores indicados o no correspondan a promociones inmobiliarias. Con el resultado de esto se tienen todas las operaciones de ofertas no consideradas anteriormente que figuran como firmadas en el CENI, junto a todas aquellas promociones del CENI.

A este resultado se le agrega el resultado de la TAUX original, uniendo ambas partes por el código de contrato, con lo que se asegura que no haya duplicados. Tras esto, se une el resultado con la tabla ELE_CLI.CL_TM_PERSONAS_NIF para asegurar que se informan todos los campos referentes a los clientes, como son su nombre o su NIF. Con objetivos similares se cruza con la tabla ELE_CTM.CTM_TM_CENTROS recuperando así la descripción del centro de aquellos registros con el campo no informado. Ambos procedimientos se realizan mediante un *left join* dejando en el lado izquierdo la unión de todo lo mencionado hasta el momento. Con esto se conservan todos los registros recuperados, incluidos aquellos no emparejados con las nuevas tablas, además de no añadir ningún registro adicional proveniente de dichas tablas.

Todo esto se agrega al resultado almacenado en PARTE_1 mediante el componente 'Definir' seleccionando 'UNION' como operador. De esta manera se ha recreado la parte 1 de la consulta asegurando que todos los campos queden informados, además de añadir los contratos de operaciones que no son promociones inmobiliarias, recuperados a través de las ofertas que no se tenían en cuenta. En otras palabras, se recuperan todos los contratos.

En este punto se dispone de los contratos relacionados o promociones, almacenados en PARTE_D1, y de todas las operaciones firmadas en el CENI. Para eliminar aquellos registros correspondientes a promociones de todo el conjunto de realiza un *left join* junto con un filtrado de nulos. La unión de ambas partes se realiza sobre los atributos de código de contrato y código de oficina, almacenando todos los registros del conjunto, incluidos los no emparejados con PARTE_D1. De esta manera, al filtrar posteriormente aquellos registros con código de contrato a nulo por parte de las promociones, se almacenan únicamente los registros correspondientes a las operaciones firmadas en el CENI.

En resumen, los campos que se almacenan en esta tabla corresponden con los campos almacenados en PARTE_D1, ya que se desea tener la misma información para ambos tipos de contratos. Estos se diferencian de los anteriores mediante la etiqueta añadida como 'Operaciones CENI'. De esta manera se podrán juntar en una tabla final sin transformaciones adicionales.

5.1.7. Mapping tabla final (MAP_API_TX_ECI_PROMOCIONES_PH)

Este último *mapping* recoge toda la información que se deseaba obtener como objetivo principal. Todo ello se almacena en la tabla destino API_TX_ECI_PROMOCIONES_PH, donde se encuentran todos los contratos de todas las promociones junto con sus descripciones y otra información relevante que se ha ido extrayendo hasta el momento.

En este se juntan todos los contratos relacionados extraídos mediante el *mapping* MAP_API_TX_PARTED1 con aquellos contratos no relacionados extraídos a través del *mapping* MAP_API_TX_PARTE2. Para ello se incluyen como orígenes las tablas resultantes de ambos mapeos. La operación en cuestión se realiza mediante un

componente 'Definir' indicando tipo de operación 'UNION', lo que da como resultado la tabla destino de ECIAPI. Una vez más, cada campo requiere una expresión para cada input. Todos ellos son comunes en ambas partes, por lo que los orígenes ya tienen el campo definido, salvo los cuatro últimos.

Estos últimos campos son los códigos de la tasación, del contrato relacionado, del contrato certificado y de la finca. Al ser atributos característicos de contratos relacionados únicamente, se establece la expresión para aquellos no relacionados a NULL. Como se ha comentado anteriormente, se dispone de un campo *flag* en ambos tipos de contratos que los diferencia, por lo que basta con arrastrarlo hasta la tabla destino.

Todas las transformaciones necesarias sobre los atributos ya han sido aplicadas en mapeos anteriores, por lo que el único campo a destacar de esta tabla es la fecha de carga, calculada en el momento de ejecución con la función 'SYSTIMESTAMP', que obtiene la fecha actual.

Por tanto, ya se dispone de toda la información necesaria tanto para los contratos del CENI como para las promociones de contratos relacionados. Los campos almacenados en esta tabla corresponden con las variables explicadas en el capítulo de metodología, apartado 4.1.2.

5.1.8. Paquete ECIAPI (PKG_API_TX_SOLICITUDES_PGA)

Por último, se construye el paquete de ECIAPI que unirá todo lo mencionado anteriormente de manera que se permita su ejecución para insertar las filas en la tabla destino. Los paquetes en ODI establecen un flujo de los datos único, por lo que es importante incluir todos los elementos en el orden adecuado.

En primer lugar, se introduce la variable N_FECHA_CONTENIDO estableciendo 'Declarar variable'. Este tipo se utiliza a la hora de almacenar un valor dado por el usuario en una de las variables de proyecto que será utilizada más adelante. Seguidamente, se conectan por orden las tres variables calculadas con una sentencia SQL que devuelve una fecha máxima. Estas tres variables se establecen como 'Refrescar variable', lo que indica a ODI la necesidad de ejecutar cada una de las *queries* con las que se da valor a dichas variables en tiempo real. Es decir, al lanzar el paquete ODI ejecuta las *queries* asociadas a las variables obteniendo el valor máximo en ese momento.

Una vez se han añadido todas estas variables que funcionarán como filtros dentro de los *mappings* correspondientes, se continúa el flujo de los datos por el *mapping* correspondiente a la tabla auxiliar. Es imprescindible establecer este mapeo al principio ya que se usa a lo largo del resto de *mappings*. De este, el flujo continúa con los mapeos de la parte 1 y de la parte d1 en ese orden.

En este punto es necesario definir la variable V_TABLA_DESTINO para, seguidamente, refrescar las dos variables restantes referentes a la partición. Estas últimas son muy importantes ya que la estructura de las particiones agrega a su nombre la fecha de inicio de la partición, además de guardar la última fecha sin incluir para la que se ejecuta. Esto queda ilustrado con la imagen inferior.

	PARTITION_NAME	LAST_ANALYZED	NUM_ROWS	BLOCKS	SAMPLE_SIZE	HIGH_VALUE
1	PTX_API_TX_ECI_PROMOCIO_202312	13/05/24	0	0	(null)	202312
2	PTX_API_TX_ECI_PROMOCIO_202401	13/05/24	66	4	66	202401
3	PTX_API_TX_ECI_PROMOCIO_202402	13/05/24	0	0	(null)	202403
4	PTX_API_TX_ECI_PROMOCIO_202403	13/05/24	0	0	(null)	202404
5	PTX_API_TX_ECI_PROMOCIO_202404	13/05/24	0	0	(null)	202405
6	PTX_API_TX_ECI_PROMOCIO_202405	(null)	(null)	(null)	(null)	202406
7	PTX_API_TX_ECI_PROMOCIO_MAXVAL	(null)	(null)	(null)	(null)	MAXVALUE

Figura 11. Particiones tabla destino

A continuación, con todas estas variables definidas, se añade un procedimiento que permite crear la partición. Como se puede observar en la figura anterior, la existencia de una última partición denominada 'MAXVAL' impide crear la nueva partición simplemente añadiéndola a la tabla almacenada hasta el momento. Esta estructura es algo común dentro de las diferentes aplicaciones que se construyen para este cliente en ODI, por lo que ya se disponía de dicho procedimiento. Esto se debe a la estructura común que siguen la mayoría de aplicaciones, consistente en la carga y almacenamiento de datos mensualmente. Este procedimiento se define con el código necesario para añadir la nueva partición entre la última fecha para la que se ejecutó y la partición correspondiente al máximo valor.

Tras crear la partición, se continúa por el *mapping* de la parte 2, seguido del mapeo que tiene como tabla destino la tabla final que se desea construir; es decir, la aplicación de ECIAPI. Por último, la salida de este va a un procedimiento que registra las cargas. Nuevamente, este procedimiento es común para varios paquetes, por lo que ya se disponía de su código. La principal función de este último procedimiento reside en el registro de las cargas, así como de las estadísticas oportunas (tema fuera del alcance de este proyecto).

5.2. Ejecución del paquete

Como se ha comentado en el capítulo de contexto, apartado 2.3.1., un paquete corresponde a una unidad de ejecución en ODI. Al ejecutar la secuencia de varios pasos organizada que compone al paquete se pueden distinguir las ejecuciones correctas de las erróneas.

Una vez se ha construido el paquete se pasa a la generación del escenario, el cual constituye la versión compilada del paquete. Para ello, se selecciona la opción de generar escenario en el menú que se muestra al pinchar sobre el propio paquete. Es conveniente tener una única versión del escenario para evitar problemas con la ejecución. El lanzamiento de la ejecución se realiza siempre sobre el escenario generado, seleccionando el nivel 6, siendo este el mayor. Al ejecutarlo, como se ha comentado antes, se ha de introducir la fecha para la que se desea generar la partición con el formato 'AAAAMMDD'.

Una vez se ha iniciado la sesión, en la ventana del operador aparece la ejecución en cuestión, dividida en los diferentes pasos que tiene el paquete indicando su correcto funcionamiento o el fallo de alguno. En este paso es común encontrar ciertos errores en los *mappings*, como por ejemplo un fallo sintáctico en la definición de la expresión para un campo.

En el caso práctico que desarrolla este trabajo se han necesitado varias ejecuciones por fallos en los *mappings*. Es por ello por lo que es importante revisar la ejecución todas las

veces necesarias hasta el correcto funcionamiento del paquete, lo que permitirá una carga de datos exitosa. La manera de proceder al encontrar un fallo es, en primer lugar, corregirlo en el *mapping* en cuestión para, a continuación, regenerar el escenario, lo cual permitirá actualizar las correcciones, y ejecutarlo de nuevo. Este proceso se repite hasta conseguir una ejecución limpia y sin errores como la que se muestra en la imagen siguiente.

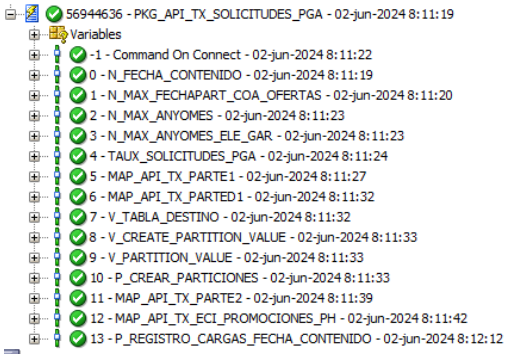

Figura 12. Ejecución del paquete en ventana Operador

El siguiente paso consiste en la comprobación de los resultados, es decir, comparar los resultados obtenidos con la *query* que genera ODI tras ejecutar el paquete, con los resultados que deseamos obtener. Este paso se explicará más detalladamente en el capítulo de validación.

5.3. Carga de datos

Una vez se tiene todo construido y se ha comprobado su correcto funcionamiento se debe planificar la carga de datos. Para ello, se distinguen dos entornos: preproducción, en adelante PRE, y producción, en adelante PRO. El primero se conoce como un entorno de desarrollo que se utiliza para realizar multitud de comprobaciones y pruebas antes de cargar lo construido en PRO. Este entorno no dispone de los datos oficiales completos, sino que se cargan los datos bajo demanda según las necesidades de trabajo. Como regla general se intentan mantener los datos al día, aunque en ocasiones no se logra debido a la necesidad de disponer de todos los datos de otras aplicaciones para cargar los nuevos datos. Este entorno está oculto para el cliente.

Por su parte, el entorno de PRO es aquel visible para el cliente, donde se dispone de todos los datos oficiales correctamente actualizados. Estos datos se cargan diariamente mediante los *jobs* programados en TWS. Además, se asegura la carga de todos los orígenes facilitados por el cliente, así como de todas las bases tras la aplicación de los procesos ETL.

Para que los datos puedan cargarse automáticamente y se actualice el cuadro de mandos que se explica más adelante en el capítulo de visualización, es necesario subir a producción todo lo que está construido en preproducción. Más concretamente, se sube tanto el paquete construido en ODI, que podrá ser ejecutado para la carga de datos en la tabla destino, como los *jobs* creados, que permitirán la planificación de la ejecución del paquete, diariamente en este caso. Esta última parte es esencial ya que, a pesar de tener el paquete subido a producción, no se ejecutará en caso de no disponer de un *job* que se encargue de indicar dicha ejecución.

Para ello se debe comprobar que el paquete esté correctamente actualizado y que su ejecución no devuelva ningún error. Una vez más, desde ODI PRE, se regenera el escenario y se ejecuta para una fecha ejemplo. Tras las comprobaciones oportunas, se debe crear o actualizar el paquete en el apartado de planes de carga y casos, dentro de la ventana del diseñador. Para ello basta con arrastrar el paquete desde el apartado de proyecto.

El siguiente paso consiste en gestionar las peticiones de versionado desde una plataforma facilitada por el cliente. En primer lugar, se debe crear la petición de versionado de componentes correspondiente a ODI, seleccionando la aplicación correspondiente, ECIAPI. Dentro de esto se crea una petición específicamente para el paquete. A continuación, se entra en las peticiones de versionado de componentes correspondiente al planificador TWS, desde donde se crea una petición para la subida del *Job Dummy* y, después, otra para la del *Job Stream*.

Tras esto, se procede con la subida de los tres elementos a PRO desde el *Developer Portal* facilitado también por el cliente. Una vez dentro de esta interfaz, se selecciona como plataforma la opción de '*DataPool*' y la aplicación correspondiente, en este caso ECIAPI. En primer lugar, se sube el paquete de dicha aplicación, por lo que se selecciona ODI como componente. Para la subida hay cuatro pasos: *version*, *build*, *promote* y *deploy*.

El primero de ellos se genera solo al crear la petición mencionada anteriormente, el cual, al completar su ejecución, genera el segundo paso. Este paso, creado para ejecutar el lanzamiento del paquete, sí requiere ciertas acciones, como es la planificación de dicho lanzamiento indicando fecha y hora. Una vez se ha ejecutado correctamente se crea el tercer paso, donde se debe introducir un código de autorización, así como volver a planificarse la ejecución. En este caso se debe planificar unos minutos más tarde, ya que, como la ejecución de este paso es algo más lenta, podría devolver un error si intenta lanzarlo antes de que finalice la ejecución. Finalmente se alcanza el último paso que, nuevamente, no requiere ninguna acción. Al terminar la ejecución de este se puede dar por concluida la subida del paquete a producción.

Por su parte, se suben ambos *jobs* siguiendo el mismo procedimiento explicado para el paquete. Para ello, se debe cambiar la selección de componentes de ODI a TWS. Es importante subir en primer lugar el *Job Dummy* y, una vez se ha subido correctamente, iniciar la subida del *Job Stream*. En caso de no seguir este orden la subida puede dar algún error.

Para comprobar que los *jobs* se han subido a producción correctamente se abre la interfaz de TWS PRO. Dentro de esta se accede a la sección '*Dynamic Workload Console*', donde se puede buscar la aplicación de ECIAPI al entrar en la gestión de definiciones de cargas de trabajo, en administración.

Dado que los *jobs* se han subido correctamente, el paquete se ejecutará cuando corresponda cargando así los datos actuales en la tabla destino dentro de producción. Con esto se puede dar por concluida la carga de datos en PRO.

6. Validación

En este capítulo se presenta la validación de la aplicación construida. Para ello, se explican una serie de comprobaciones en cuanto a los diversos pasos seguidos para su construcción, además de efectuar ciertas pruebas con los datos disponibles de manera que se detecten posibles errores.

La construcción de la aplicación en ODI se realiza en el entorno PRE hasta que se tiene total seguridad de que todo funciona correctamente y se cargan los datos en PRO. Es por ello por lo que las primeras comprobaciones dadas en la *checklist* explicada a continuación se efectúan en PRE.

En cuanto a las pruebas de volumetría y datos desarrolladas en el apartado 6.2., se indica el uso de PRO ya que, como se ha comentado anteriormente, en ocasiones no se logra mantener las bases de datos disponibles en PRE completamente actualizadas. Al optar por una manera de proceder no invasiva para este entorno no se interfiere en la oficialidad ofrecida en PRO, por lo que no hay ningún inconveniente en su uso.

6.1. *Checklist* desarrollo PRE

Esta *checklist* es común dentro todas las aplicaciones construidas, por lo que constituye una plantilla ya creada. En ella se comprueba la realización de todos los pasos importantes dentro del desarrollo en PRE, pudiendo indicar 'en espera' para aquellos pasos que aún no se han completado y 'ok' cuando el paso esta correctamente efectuado. Además, se ha de adjuntar una imagen que sirva como evidencia de dicho paso una vez ha alcanzado el valor 'ok'. En la siguiente imagen se puede observar la estructura de esta *checklist*.

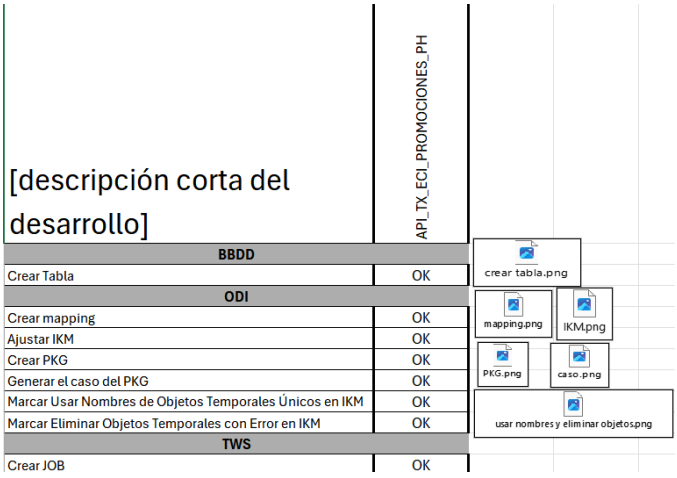

Figura 13. Ejemplo checklist

En el caso práctico que expone este proyecto únicamente se ha construido una tabla destino donde se cargan los datos e información objetivo, por lo que esta *checklist* solo completa una columna. Dentro de ella se encuentran pasos como la creación de tablas en bases de datos y la creación de los *mappings* y el paquete en cuestión en ODI, así como la generación del escenario y su ejecución, pasos explicados anteriormente. Por otra parte, se indican las acciones necesarias para ajustar el IKM (módulo de conocimiento de integración), marcar el uso de nombres de objetos temporales únicos y la eliminación

de objetos temporales con error. Estos últimos pasos se han completado desde el esquema físico de ODI para el mapeo de la tabla final, entrando en las propiedades de la tabla destino API_TX_ECI_PROMOCIONES_PH.

Adicionalmente, se documenta tanto la creación del *job* en TWS, añadiendo las dependencias correspondientes, como la asignación de dicho *job* al *JobStream* que corresponda añadiendo las dependencias al *JobStream Dummy*. Esta parte, con mayor explicación en el apartado de contexto en TWS (2.3.4.), queda fuera del trabajo realizado por el alumno.

Finalmente, se comprueban varios pasos dentro del pase a producción, entre los que se encuentran la creación del *job dummy* y del J*obStream dummy*, añadirles dependencias y la apertura de un RTC de pase a producción. Nuevamente todos estos pasos no han sido completados por el alumno.

6.2. Pruebas con datos en PRO

Una vez se ha construido ECIAPI en ODI y su ejecución no ha devuelto ningún error se efectúan una serie de pruebas de volumetría y datos. La manera de proceder consiste en la ejecución de la consulta original en bases de datos dentro del entorno dedicado a ECI_API en PRO. La elección de este entorno para dichas pruebas se debe a la completitud de los datos, como se ha mencionado en la introducción del capítulo. Por su parte, PRE alberga un subconjunto de todos estos datos.

Paralelamente, se ejecuta la misma consulta resultado de la ejecución del paquete en ODI. La separación de las *queries* generadas por este último en las distintas partes es tarea sencilla, ya que en la ventana del operador se desglosa la ejecución en los distintos pasos, indicando asimismo su correcto funcionamiento o si hay algún error. Dentro de cada paso se puede acceder al código generado para la inserción de filas, con lo que bastará con copiarlo para la parte en cuestión.

No obstante, al haber construido la aplicación dividiendo la *query* general en varios *mappings*, cada uno con una tabla resultado, el código de aquellas partes que hagan uso de alguna de estas tablas en su mapeo hará referencia a la tabla con el nombre dado. Por su parte, en bases de datos no se dispone de dichas tablas por lo que una ejecución del código devuelto por ODI daría errores.

Para solucionar este problema se han estudiado dos posibilidades. Por una parte, se conoce la estructura ‘with *nombre* as’, seguido de toda la *query* que se desea guardar para poder referenciarla luego por *nombre*. De está manera se añaden al principio de la *query* todas las *subqueries* que se utilizan o a las que se hace referencia dentro de la principal. Por otra parte, se pensó en la creación de tablas auxiliares dentro de bases de datos donde almacenar los resultados obtenidos al ejecutar las distintas partes por orden. En cualquier caso, la primera opción es favorable ya que, a pesar de ser más compleja, la creación de tablas auxiliares en PRO no se alinea con la profesionalidad de esta compañía.

Es en este punto cuando se pueden efectuar las comparaciones correctamente. En primer lugar se realizan las pruebas de volumetría, ejecutando cada una de las cuatro partes en las que se divide la *query* y realizando una operación ‘count’ que devuelva el número total

de registros recopilados. Esto se efectúa tanto con el código de la *query* original como con el código desarrollado con ODI.

En caso de coincidir, se realizan las comparaciones de datos. Para ello se vuelven a ejecutar ambos códigos, sin la operación 'count', pero añadiendo una ordenación por alguna de las claves primarias. De esta manera se permite la comparación de algunas filas seleccionadas aleatoriamente, ya que siguen el mismo orden dentro del conjunto de registros.

Tras todas estas pruebas, se decide volverlas a realizar para los meses restantes para los que se tienen datos dentro de este año. La principal motivación de esto es la corrección de posibles errores obtenidos por excepciones en los propios datos. Un ejemplo de esto lo constituye un caso en el que Palma de Mallorca se registró como localidad como Palma. No coincidiendo así con el resto de los registros, cuya localidad era Palma de Mallorca.

Finalmente, gracias a estas pruebas se permite la corrección no solo de errores en la construcción de ECIAPI, sino también errores originados por problemas o excepciones en las bases de datos.

Para facilitar la posterior comprensión del trabajo realizado en caso de necesitar alguna modificación, corrección o simplemente servir de ejemplo, así como para evidenciar el correcto funcionamiento de la aplicación, todas estas pruebas se documentan en un pdf con imágenes de las evidencias junto a una pequeña descripción de la prueba.

7. Visualización

En el desarrollo de este capítulo se habla de la visualización de los datos resultado de la aplicación mediante las herramientas tecnológicas comentadas anteriormente. Esta última parte del proyecto queda fuera del alcance del trabajo, aunque se ha decidido incluir una breve explicación de las principales ideas de visualización y de los objetivos y utilidades de la aplicación construida, debido a su capacidad de representar gráficamente los resultados de la aplicación construida.

Para la explotación de los datos visualmente se hace uso de Qlik, herramienta ya introducida con el contexto en el capítulo 2. Dentro de esta se encuentran tres partes:

- Extractor. Es el encargado de sacar la información. Habitualmente se compone de una *query* de SQL para seleccionar los campos necesarios de las bases de datos.
- Transformador. Fase en la que se realizan las modificaciones oportunas a los datos cargados con el extractor. Entre esta serie de transformaciones se encuentran establecer con formato fecha aquellos datos de tipo *date* o codificar con 'SI' y 'NO' aquellos valores '0'/'1'.
- *Dashboard*. En esta parte, con los datos ya transformados cargados, se construyen las métricas, los *kpis*, las tablas y los gráficos, entre otros.

En esta compañía se hace uso de varios entornos dentro de Qlik Sense en función del objetivo de trabajo con los datos. Dentro de los diferentes entornos se encuentran los siguientes:

- Qlik DEV. Empleado para cargar toda la información necesaria de base de datos, transformarla y crear las métricas, kpis y gráficas oportunas. Usa datos de PRE principalmente para el desarrollo.
- Qlik PRE. Similar a Qlik DEV, pero solo permite modificar el *dashboard*. La extracción de datos y las transformaciones deben estar cargadas. Se centra en la visualización de los datos de PRE.
- Qlik PROVAL. Este entorno se alimenta de las bases de datos de PRO, es decir, carga los datos definitivos de la aplicación. Está oculto al cliente y permite hacer pruebas y comprobaciones con datos certeros.
- Qlik PRO. Accesible para todos los usuarios. Se publica tras comprobar el correcto funcionamiento de la aplicación en PROVAL.

Por otra parte, existe Qlik Sandbox, un entorno más simple que permite la realización de las tres partes, extracción, transformación y visualización, todo en un mismo espacio de trabajo.

7.1. Objetivos

El cliente de esta consultoría originalmente trataba sus propios datos. Dentro de su compañía se disponía de un departamento encargado de aplicar las transformaciones oportunas a los datos y almacenarlos adecuadamente. Posteriormente, desarrollaban un informe acerca de los detalles de las promotoras, así como un mapa donde situaban las

viviendas actuales. Todo esto lo publicaban en un documento PowerPoint semanalmente.

El principal objetivo de esta última parte del proyecto es cumplir con las expectativas de visualización del cliente pero automatizando el proceso, motivación por la cual se ha contratado a la compañía y encargado dicho trabajo. Para ello se reconstruyen los gráficos, indicadores y mapas que ya disponían de manera que se actualizan automáticamente en la aplicación tras la carga de datos.

Como objetivos secundarios se destaca la creación de otros gráficos que se consideran relevantes para el entendimiento de los datos por parte del cliente, además de aportar opiniones y mejoras en el ámbito estético dentro de aquellos gráficos que se piden replicar.

7.2. Usuarios

El usuario final de este proyecto de visualización es el cliente que contrata el trabajo que se expone con este caso práctico. El cuadro de mandos que se construye con esta aplicación de visualización, incluyendo los dashboards y mapas desarrollados, es el producto final que se entrega al cliente.

Dentro de la compañía del cliente será el departamento encargado del análisis de negocio el responsable de valorar este trabajo y solicitar cambios o añadidos a la visualización. Además, podrán navegar dentro del cuadro de mandos seleccionando fechas o filtrando datos, entre otros, ya que se construye como elemento interactivo.

7.3. Implementación

Para el desarrollo del cuadro de mandos, producto final que se entrega al cliente, se hace uso de todos los entornos de Qlik introducidos en este capítulo. En primer lugar, se construye el cuadro de mandos con todos sus gráficos y mapas en Qlik Sandbox. Este entorno nos permite manipular de manera sencilla el diseño de las distintas hojas o *dashboards*, así como de cada gráfico y mapa construido. Este diseño está disponible para el cliente de manera que puede ver la evolución del producto y dar *feedback*. Con los comentarios y nuevas peticiones del cliente se perfecciona el cuadro de mandos en su totalidad y, una vez está terminada correctamente la visualización, se replica todo esto en los siguientes entornos, siguiendo el orden en el que se han presentado.

Con esto, se comienza por construir la visualización en DEV, donde ya se dispone del extractor, el transformador y el *dashboard* como tres aplicaciones separadas, por lo que la implementación varia ligeramente respecto a Sandbox. Tras esto, se sube al entorno de PRE, a PROVAL y a PRO, siguiendo este orden.

Finalmente, es en este último entorno de producción (Qlik PRO) donde el cliente tiene acceso a la aplicación final que consumirán.

Dado que la parte de visualización de la aplicación queda fuera del alcance de este proyecto no se detalla su desarrollo.

7.4. Diseño de la interfaz y resultado final

El resultado final de este capítulo de visualización está constituido por un cuadro de mandos interactivo que gestiona la carga de datos de manera automática diariamente.

Este cuadro de mandos se compone de siete hojas distintas por las que navegar, las cuales dividimos en las seis primeras, pedidas específicamente por el cliente, y la última titulada 'Otros', con gráficos considerados como valor adicional. Para ello se sitúa en la parte superior de la interfaz una barra que permite seleccionar la hoja que se desea visualizar. Además, en todas ellas se indica la fecha de la última carga de datos y se dispone de tres filtros. El primero de ellos, un filtro temporal, permite seleccionar los datos dentro de un periodo de tiempo desplegando un calendario. Los dos restantes se nombran como filtros generales y permiten seleccionar un CENI o promotor concreto de una lista desplegable para cada tipo con todos los nombres.

Las dos primeras hojas se basan en la comparación entre las operaciones del CENI y las operaciones por promotor. La primera, titulada *'Operaciones formalizadas'*, divide la hoja en dos mitades donde muestra en un gráfico de barras el límite total, los avales y los confirming, permitiendo cambiar el gráfico de CENI a promotor. En la mitad derecha de la hoja se visualiza el número total de viviendas por provincias, pudiendo alternar entre un mapa que muestra la localización y un gráfico de barras con el nombre de las provincias. En la imagen inferior se muestran las operaciones por CENI y el mapa con el número total de viviendas.

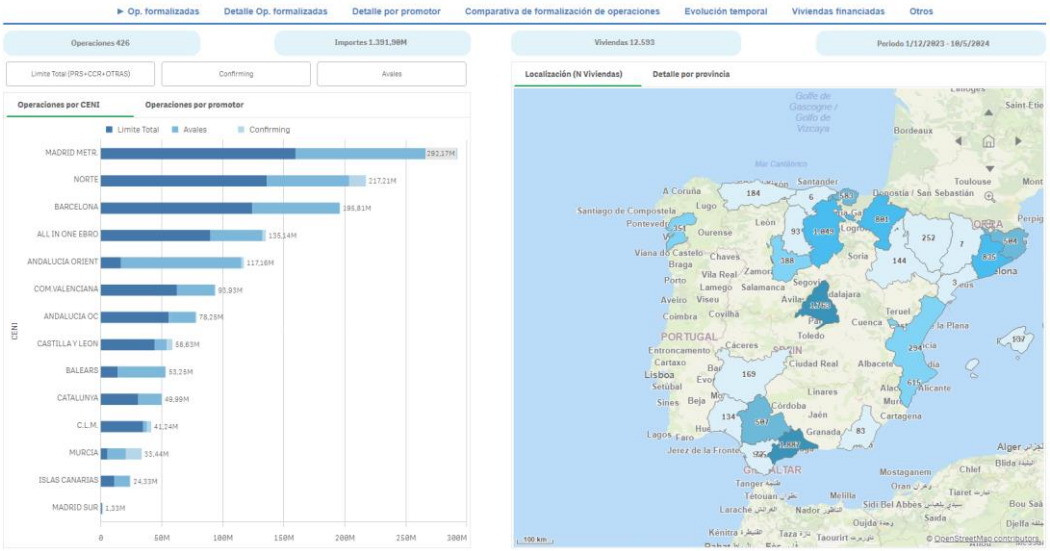

Figura 14. Operaciones por CENI y localización viviendas

Por su parte, la hoja de *'Detalle operaciones formalizadas'* realiza la comparativa entre promotor y CENI incluyendo información más especifica y con mayor detalle de estas operaciones. En esta ocasión se eligen dos tablas para mostrar la información. Para la tercera hoja, 'Detalle por promotor', se escoge nuevamente una tabla para la visualización. En este caso se desglosa cada tipo de importe (límite total, avales y confirming) con su importe concreto para cada promotor.

Figura 15. Hoja 'Comparativa de formalización de operaciones'

A continuación se construyen las tres hojas siguientes, *'Comparativa de formalización de operaciones'*, *'Evolución temporal'* y *'Viviendas financiadas'*, con gráficos considerados algo más visuales. La primera de estas dispone de tres gráficos de barras para representar el número de operaciones, el importe y el número de viviendas. Estos tres gráficos se presentan tanto con datos anuales como mensuales, como se puede observar en la imagen superior.

Para la evolución temporal, así como para las viviendas financiadas, se muestran los datos a elegir entre estos tres totales mencionados (número de operaciones, importe y número de viviendas) localizados en un mapa de España. A esto se le añade una barra que permite seleccionar desde la actualidad hasta tres meses atrás para la primera hoja. En cuanto a la segunda hoja se disponen dos mapas, uno con la ubicación del CENI origen que financia las viviendas y otro para la localización de destino de dichas viviendas, como queda reflejado en la siguiente imagen.

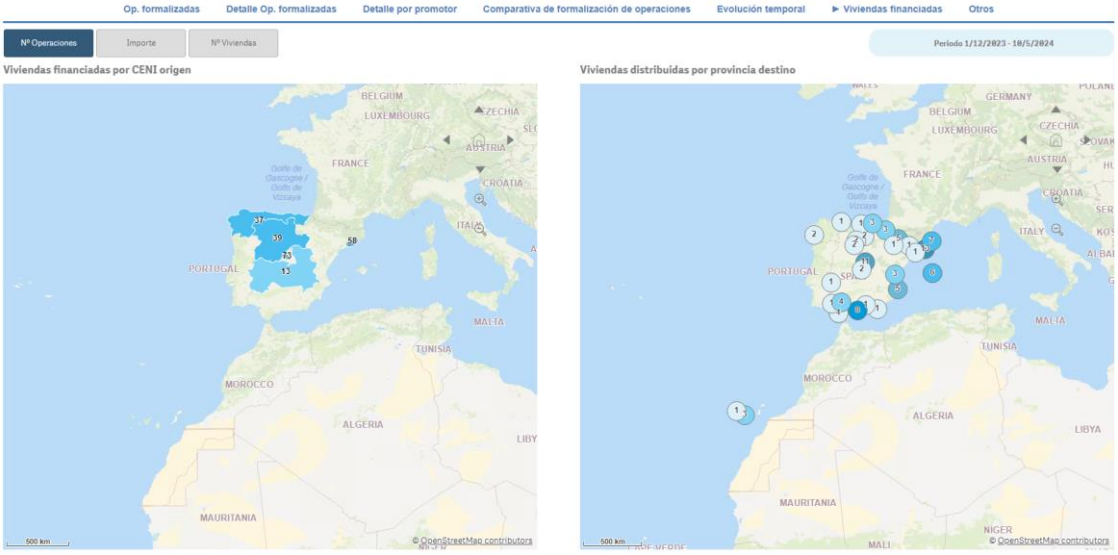

Figura 16. Hoja 'Viviendas financiadas'

Por último, se añade una hoja para complementar la información mostrada en esta última. Como se ha mencionado al inicio del apartado esta hoja no forma parte de los dashboards solicitados por el cliente, sino que se añade por iniciativa propia al considerarla útil en el entendimiento de cierta información. En ella se dispone un diagrama de Sankey, como el que se muestra en la siguiente imagen, que permite conectar los datos del CENI origen de la financiación de ciertas viviendas con la provincia destino de dichas viviendas. Asimismo, se facilita esta misma información en formato tabla.

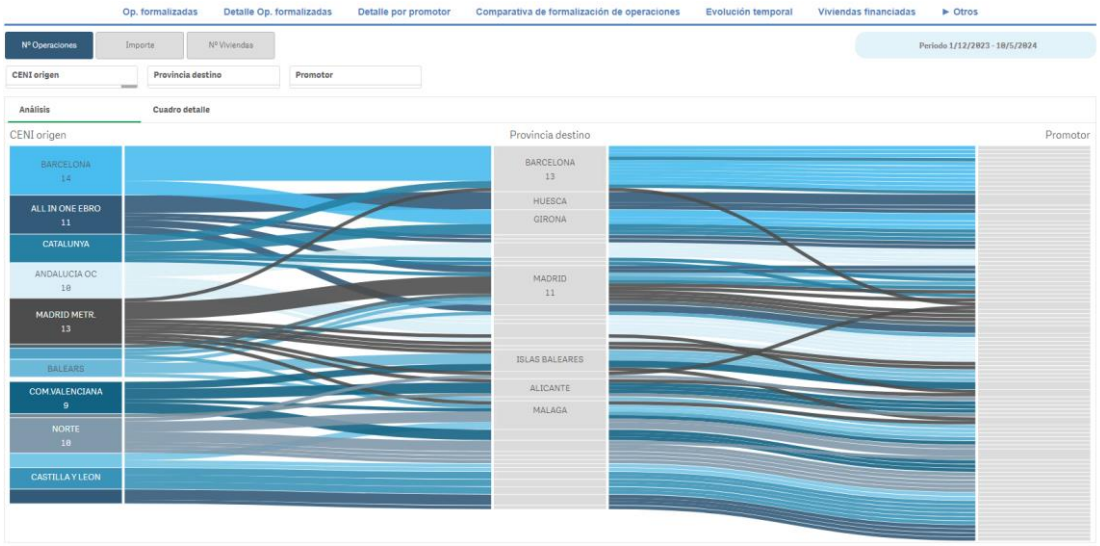

Figura 17. Diagrama de Sankey

8. Conclusiones

Este trabajo de fin de grado se ha desarrollado en torno a la construcción de una aplicación para la recopilación y visualización de datos sobre promociones inmobiliarias. Por aspectos temporales y el propio trabajo del estudiante dentro de la compañía, este proyecto se ha centrado en el desarrollo de la primera parte, la cual dará paso a la segunda.

Para alcanzar el objetivo principal de construcción de una aplicación que recopile e integre los datos necesarios, se ha pasado por varios objetivos secundarios. Empezando por el análisis de la consulta a recrear, se ha particionado dicho código de manera que su implementación resultase más sencilla. Con esto, se ha introducido la herramienta ODI, construyendo los cinco *mappings* en los que se divide la *query* original, así como desarrollando las variables necesarias para su implementación.

Todo esto se une en el paquete, encargado de juntar y relacionar todo. Con su ejecución se extraen los datos de los orígenes necesarios, se aplican las transformaciones y cruces oportunos, y se almacenan en la tabla destino. Como último objetivo secundario se destaca la subida a producción de todo el trabajo desarrollado, posibilitando así la carga de datos.

En resumen, se ha seguido el orden preestablecido de los objetivos a cumplir, completando todas las tareas para la finalización de este trabajo con éxito, además de efectuar las comprobaciones y validación oportunas.

Adicionalmente, se incluye un objetivo complementario, el cual se traduce como la principal utilidad de la tabla destino construida, la visualización de los datos. Con este capítulo, a pesar de no haber sido desarrollado completamente por el estudiante, se concluye la construcción de la aplicación ECIAPI con éxito, adquiriendo antes la aprobación por parte del cliente.

8.1. Legado

En cuanto al legado de este proyecto, se enfatiza en su principal beneficiario, el cliente de ámbito financiero introducido anteriormente. La aplicación desarrollada en este trabajo es una pequeña parte de todo el proyecto que se desarrolla continuamente para este cliente dentro de la compañía, con el objetivo principal de facilitar el conocimiento y comprensión de los datos que recogen, impulsando así la toma de decisiones para dicho cliente. De esta manera, el cliente será capaz de tomar mejores decisiones basándose en el conocimiento extraído de los datos, así como de prevenir y gestionar mejor los posibles riesgos a los que se enfrenta.

Adicionalmente, tendrá un impacto en términos de carga de trabajo, ya que se pasa del desarrollo de un informe semanal a una aplicación que actualiza los datos de manera automática diariamente.

Por su parte, este proyecto ha servido al estudiante para familiarizarse más con términos bancarios, además de trabajar dentro de un caso práctico ejemplificando así la aplicación en la vida real de los estudios cursados, especialmente en ámbitos que, a priori, no son tecnológicos.

8.2. Relación con los estudios cursados

Como se ha reflejado a lo largo del desarrollo de este proyecto, se han puesto en práctica varios de los conocimientos adquiridos durante los estudios. Especialmente, se destaca el uso del lenguaje SQL y la aplicación práctica de todos los conocimientos recibidos sobre bases y gestión de datos. Asimismo, las herramientas para tratar con este lenguaje vistas durante los estudios cursados se han complementado con el uso de otras herramientas nuevas para el estudiante. No obstante, gracias a los conocimientos impartidos a lo largo de los estudios y la familiarización con el lenguaje, estas nuevas herramientas no han supuesto un problema para el desarrollo del trabajo.

Adicionalmente, cabe mencionar las técnicas de visualización desarrolladas en asignaturas como proyecto o visualización, las cuales han sido una gran ayuda a la hora de desarrollar el objetivo secundario referente a la visualización de los datos.

Por último, se destacan ciertas competencias transversales trabajadas durante los cuatro años de estudio que han sido de gran ayuda al estudiante para desarrollar su trabajo dentro de este equipo de manera proactiva, como son el trabajo en equipo y liderazgo o la responsabilidad y toma de decisiones.

Bibliografía

Amazon Web Services. (s.f.). Obtenido de ¿Cuál es la diferencia entre los procesos de ETL y ELT?: https://aws.amazon.com/es/compare/the-difference-between-etl-and-elt/

Bustamante Martínez, A., Galvis Lista, E. A., & Gómez Flórez, L. C. (2013). Técnicas de modelado de procesos de ETL: una revisión de alternativas y su aplicación en un proyecto de desarrollo de una solución de BI. *Scientia Et Technica*, 185-191.

Cristobal, D. (2022). *semantic systems*. Obtenido de https://www.semantic-systems.com/semantic-noticias/articulos-tecnologicos/que-es-qlik-analitica-potente-y-visual-de-manera-rapida-y-sencilla/

Gucer, V., Jones, R., Jojic, N., Patrick, D., & Bain, A. (octubre de 2003). *IBM Tivoli Workload Scheduler*. Obtenido de https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3887e81f58994dc57a1594b8af8d771b88a5340f

Hecksel, D., & Wheeler, B. (2012). *Getting Started with Oracle Data Integrator 11g: A Hands-On Tutorial. Packt Publishing Ltd.*

Hofman Miquel, L. (abril de 2015). Developer's Guide for Oracle Data Integrator. Obtenido de https://docs.oracle.com/middleware/11119/odi/develop/E12643-07.pdf

IBM. (2012). *Workload Scheduler User's Guide and Reference. IBM.*

Instituto Nacional de Estadística. (2024). *Índice de Precios de Vivienda (IPV). Base 2015. Primer trimestre 2024*. Obtenido de https://ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736152838&menu=ultiDatos&idp=1254735976607

Martínez Sanz, F., & Broseta Pont, M. (2014). *Manual de derecho mercantil.* Tecnos.

Martínez, T. (septiembre de 2018). *Gestión de datos empresariales utilizando procesos ETL*. Obtenido de http://ri.uaemex.mx/bitstream/handle/20.500.11799/95251/Gestion%20de%20datos%20empresariales%20utilizando%20procesos%20ETL.pdf?sequence=1&isAllowed=y

Murillo Junco, M. J., & Cáceres Castellanos, G. (s.f.). Business intelligence y la toma de decisiones financieras: una aproximación teórica. *Revista Logos, Ciencia & Tecnología*, 119-138.

Narayanan, A. (2009). *Oracle SQL Developer 2.1. Packt Publishing Ltd.*

Oracle. (2021). *¿Qué es SQL Developer?* Obtenido de https://www.oracle.com/es/database/sqldeveloper/technologies/what-is-sql-developer/

Petri, G. (2005). *A comparison of Oracle and MySQL. TUSC.*

Qlik. (2023). Obtenido de Analitics products: https://www.qlik.com/es-es/products/analytics-products

SAS. (2019). *ETL. What it is and why it matters*. Obtenido de https://www.sas.com/en_us/insights/data-management/what-is-etl.html

Seegmuller de Carvalho, L. F. (2023). El confirming, su posición jurídica y la dificultad de agregar los proveedores a las plataformas. *Revista de Derecho del Sistema Financiero, 5*, 325-366. Obtenido de Banco Sabadell.

Título X del libro IV. (25 de julio de 1889). Obtenido de Código Civil: https://www.boe.es/buscar/act.php?id=BOE-A-1889-4763

Anexo I. Objetivos de Desarrollo Sostenible

Tabla 1. Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza**				**x**
ODS 2. **Hambre cero**				**x**
ODS 3. **Salud y bienestar**				**x**
ODS 4. **Educación de calidad**				**x**
ODS 5. **Igualdad de género**				**x**
ODS 6. **Agua limpia y saneamiento**				**x**
ODS 7. **Energía asequible y no contaminante**				**x**
ODS 8. **Trabajo decente y crecimiento económico**		**x**		
ODS 9. **Industria, innovación e infraestructuras**		**x**		
ODS 10. **Reducción de las desigualdades**				**x**
ODS 11. **Ciudades y comunidades sostenibles**	**x**			
ODS 12. **Producción y consumo responsables**				**x**
ODS 13. **Acción por el clima**				**x**
ODS 14. **Vida submarina**				**x**
ODS 15. **Vida de ecosistemas terrestres**				**x**
ODS 16. **Paz, justicia e instituciones sólidas**				**x**
ODS 17. **Alianzas para lograr objetivos**				**x**

Fuente: Elaboración propia.

Este trabajo de fin de grado se alinea con algunos de los Objetivos de Desarrollo Sostenible recogidos en la Agenda 2030, diseñada como un "plan para lograr un futuro mejor y más sostenible para todos".

En primer lugar, el proyecto desarrollado se relaciona con el octavo objetivo, el cual pretende promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos. Una de las principales utilidades de la aplicación construida es la mejora del análisis y toma de decisiones dentro de una empresa de ámbito financiero, más concretamente en el sector inmobiliario. Al proporcionar las herramientas de visualización adecuadas dicha empresa será capaz de identificar oportunidades de desarrollo económico e incluso de empleo al apoyar la creación de nuevos proyectos inmobiliarios. Asimismo, la demanda de construcción de esta

aplicación por parte de la empresa contribuye indirectamente al crecimiento de puestos de trabajo, especialmente entre trabajadores jóvenes, además de contribuir a la igualdad en el mercado laboral. Al no disponer de los recursos necesarios se contrata a una consultoría, lo cual libera puestos de trabajo en un entorno que promueve la igualdad entre sus trabajadores, independientemente de la edad y el género, así como entornos de trabajo seguros y protegidos.

Por otra parte, el proyecto desarrollado se relaciona en gran medida con el objetivo número nueve, el cual pretende construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación. Se implementan tecnologías avanzadas para la gestión y el análisis de datos en el sector inmobiliario, como son las herramientas de ODI y Qlik, lo que promueve la innovación y progreso tecnológico en este sector. Además, se fomenta la industrialización sostenible, ya que el uso de estas herramientas deriva en una gestión más eficiente de los recursos.

Por último, se destaca el objetivo número once, el cual pretende lograr que las ciudades y los asentamientos humanos sean inclusivos, seguros, resilientes y sostenibles. El trabajo expuesto se alinea en mayor medida con este objetivo, ya que la aplicación construida tiene un impacto directo en el desarrollo urbano sostenible. El análisis de datos sobre el sector inmobiliario es esencial a la hora de optimizar la planificación de ciudades. Además, las herramientas de visualización presentadas facilitan la visión global del estado actual del desarrollo inmobiliario. Más concretamente, los mapas construidos en el cuadro de mandos permiten la identificación de zonas sostenibles pendientes de urbanizar, lo cual puede contribuir a la creación de ciudades más inclusivas, seguras, resilientes y sostenibles.

En conclusión, el proyecto que se expone no solo responde a las necesidades de la empresa, sino que contribuye a algunos de los objetivos de desarrollo sostenible principalmente mediante la promoción del empleo, la innovación tecnológica y el apoyo para lograr un desarrollo urbano más eficiente.