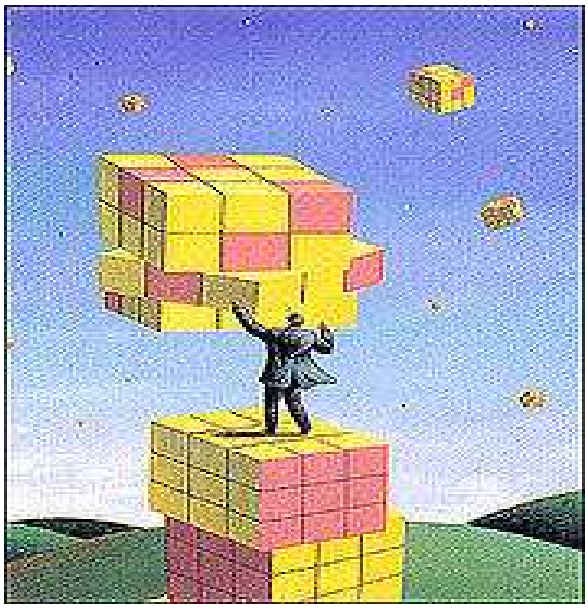
TRABAJO FIN DE CARRERA

MEMORIA



Construcción y explotación de un almacén de datos para la empresa inmobiliaria "Un techo para todos"

Pedro Hervás Bolaños
Consultor : Xavier Plaza Sierra
Ingeniería Técnica en Informática de Sistemas
Fecha: 9 enero 2012

1. DEDICATORIA Y AGRADECIMIENTOS

A mi mujer Ana (la gran culpable de que me embarcase en esta aventura que ahora acaba), gracias a su cariño, comprensión y sobre todo por animarme y darme fuerzas en los momentos de frustración, sin ella no habría sido capaz de llegar a este punto. Gracias a mis padres y hermanos a los que sorprendí cuando me decidí a realizar la carrera.

A mi amigo Shane que siempre estuvo dispuesto para ayudarme con la que para mí era una de las asignaturas más difíciles de la carrera, a Fernando compañero de trabajo y con el que he tenido la suerte de compartir formación (sin su ayuda no habría salido tan bien parado de alguna asignatura).

Gracias a Merche, Antonio y Carla, magníficos profesores que me ayudaron en las materias donde menos conocimientos tenia, y a todas las personas con las que he trabajado a lo largo de estos últimos años, las cuales me han apoyado y ayudado a compaginar mi vida laboral con la formativa.

2. RESUMEN Y PALABRAS CLAVE

2.1. Resumen

Este documento es la memoria del TFC (trabajo fin de carrera) del área de almacenes de datos, correspondiente a la Ingeniería Técnica en Informática de Sistemas de la UOC (Universidad Oberta de Cataluya). En él se describe el plan de trabajo, el análisis y la implementación del TFC.

El objetivo de este trabajo es familiarizarse con la arquitectura de un Date Warehouse, su diseño y el tratamiento de los datos, además de trabajar con la información y realizar los informes que nos solicitan a través de la herramienta Business Intelligent Atlas SBI.

Este proyecto consiste en el diseño, implantación y explotación (a través de informes) de un almacén de datos que sirva de herramienta de apoyo en la toma de decisiones a la empresa inmobiliaria “Un techo para todos”. La base de datos se implementa sobre Oracle 10g Express Edition.

Se parte de un conjunto de datos en formato texto con características y ofertas de los inmuebles que pertenecen a la inmobiliaria. Para lograr el objetivo del proyecto, se crean una serie de scripts, funciones y procedimientos SQL que junto con algunos procesos automatizados crean la estructura del almacén de datos, realizan la extracción, transformación y carga de los datos y los actualizan mensualmente.

Por último a través de Atlas SBI se implementan los informes necesarios para cumplir con los requerimientos de la inmobiliaria, estos informes permitirán obtener información sobre los inmuebles que han bajado de precio, su tiempo medio en venta, superficies, etc., así como facilitar la información necesaria sobre el piso-tipo español y andorrano.

2.2. Palabras clave

Data Warehouse, almacén de datos, OLAP, cubo OLAP, dimensión, modelo dimensional, tabla de dimensiones, tabla de hechos, ETL, SQL*Loader, Oracle, SQL, PL/SQL, Business Intelligence.

Índice

Índice de Figuras

3. INTRODUCCION

3.1. Justificación del proyecto y contexto

Este proyecto es la parte final de la Ingeniería Técnica en Informática de Sistemas, en el se tiene que plasmar todos los conocimientos adquiridos durante la carrera y en especial los relacionados con asignaturas de bases de datos.

El proyecto nace como consecuencia de la petición realizada por la empresa inmobiliaria “Un techo para todos” la cual nos encarga el diseño y la realización de un almacén de datos, que les ayude en la toma de decisiones, de forma que puedan ajustar la oferta y la demanda de sus inmuebles teniendo en cuanta los análisis por zonas geográficas, tipologías de pisos, precios de venta, etc.

La explotación de la información (con la herramienta Businnes Intelligence Atlas SBI), junto con la automatización de la actualización mensual de datos, permitirá cumplir con los requerimientos del cliente a través de un análisis rápido, detallado y fiable de la información.

3.2. Objetivos

En este apartado se describen los objetivos que se quieren alcanzar en este Trabajo Fin de Carrera (TFC). Para los objetivos realizamos la siguiente clasificación:

- Objetivos de la empresa “Un techo para todos”.
- Objetivos generales.
- Objetivos como asignatura.
- Objetivos del proyecto.

3.2.1 Objetivos de la empresa “Un techo para todos”

- Disponer de un sistema de información que les permita comparar los inmuebles por diferentes zonas, tipologías de pisos y precios, de forma que puedan intentar ajustar la oferta y la demanda, obteniendo así ventaja competitiva frente a otras empresas del sector.

- Disponer de un conjunto predefinido de informes, con la información solicitada.

- Obtener la información sobre el piso-tipo español y andorrano (tipología, características y precio del piso estándar de estos dos países), así como de del detalle de los criterios seguidos para llegar al resultado.

3.2.2 Objetivos generales

Objetivos generales son:

- Afianzar los conocimientos adquiridos anteriormente sobre como elaborar un documento de especificaciones basado en los requerimientos del usuario.
- Obtener destreza en la planificación de proyectos de implantación de almacenes de datos.
- Obtener experiencia y ampliar conocimientos en el diseño y explotación de almacenes de datos.
- Adquirir nociones de administración de Oracle.
- Obtener conocimientos y experiencia en el uso de herramientas que faciliten la explotación de un almacén de datos.

3.2.3 Objetivos como asignatura

El principal objetivo de este Trabajo Final de Carrera (TFC) es el de consolidar los conocimientos previamente adquiridos a lo largo de los estudios de Ingeniería Técnica, fundamentalmente los relacionados con asignaturas de las bases de datos.

3.2.4 Objetivos del proyecto

El principal objetivo del proyecto que tenemos que realizar para la empresa “Un techo para todos”, es la creación de un almacén de datos que facilite la toma de decisiones a la empresa. Para entender mejor este objetivo, podemos dividir el mismo en varios objetivos más sencillos y fáciles de entender.

- Objetivo Uno. Crear un proceso que importe los datos (en formato texto) facilitados por la empresa, al sistema de bases de datos elegido. Para asegurar el éxito de esta importación, será necesario analizar previamente los datos, de forma que evitemos datos inconsistentes o erróneos, datos que nos dificultarían posteriormente la obtención de los informes.

- Objetivo Dos. Diseñar y poner en funcionamiento un almacén de datos que sea capaz de trabajar con la información facilitada por la empresa inmobiliaria y que permita la generación de los informes que solicitan en el enunciado.

- Objetivo Tres. Desarrollar una herramienta capaz de generar los informes con una calidad y un consumo de tiempo aceptables.

3.3. Enfoque y metodo seguido

La implantación de un Data Warehouse se suele descomponer en una serie de fases, las cuales pueden variar en función del autor que se consulte. A pesar de estas variaciones, las fases más comunes son reconocidas por la mayoría de autores.

Para el desarrollo de este proyecto, se sigue la metodología descrita en *http://www.1keydata.com/datawarehousing/datawarehouse.html*, adaptando la misma a las necesidades del proyecto a realizar. A continuación se describen con más detalle las fases del proyecto.

1. Análisis de requerimientos (*Requirement Gathering*)

Consiste en recopilar los requisitos de los usuarios finales, los cuales no suelen tener conocimientos técnicos, elaborando con su colaboración un documento de requisitos, dicho documento debe englobar todo aquello que constituya un éxito para el proyecto.

También se deben identificar otros detalles como: el hardware necesario, las fuentes de datos, los requerimientos de capacitación. A la vez que se realiza un plan que concrete las fechas de inicio y finalización del proyecto, un plan de recuperación de desastres, etc.

2. Configuración del entorno físico (*Physical Environment Setup)*

En esta fase se configuran los servidores físicos y bases de datos. Se suelen establecer dos entornos, un entorno de desarrollo y otro entorno de producción, aunque en ocasiones se llegan a establecer tres, siendo el tercer entorno de prueba.

Disponer de diferentes entornos proporciona muchas ventajas como: que los cambios pueden ser probados, sin afectar al entorno de producción, que se puede seguir desarrollando mientras los usuarios continúan accediendo al almacén de datos, etc.

3. Modelado de datos (*Data Modeling*)

Es una fase muy importante del proyecto, un buen modelo de datos permitirá al sistema de almacenamiento crecer con facilidad y obtener un buen rendimiento del mismo.

Con las necesidades de los usuarios se obtiene un modelo lógico que luego se traduce en un modelo de datos físico.

Otra parte del modelado de datos es la identificación de las fuentes de datos, aunque en ocasiones esta fase se retrasa hasta el paso ETL. Suele recomendarse aplicarlo en esta fase, para que en caso de que los datos no estén disponibles dar la alarma lo antes posible.

4. Modelado de datos (*ETL*)

El ETL (Extracción, Transformación, Carga) es una fase que suele tardar más tiempo en desarrollarse, en ocasiones puede tomar hasta el 50% de la implantación de un Data Warehouse. El motivo es que se necesita tiempo para obtener los datos de origen, comprender las columnas, los modelos de datos lógicos y físicos, etc.

5. Diseño cubo OLAP (*OLAP Cube Design*)

En esta fase se define el cubo OLAP, el cual deriva de la captura de requerimientos de los usuarios y de los informes que se deben generar.

6. Desarrollo Front End (*Front End Development*)

Durante esta etapa se define el Front End que se utilizará, escogiéndose normalmente herramientas comerciales, en su elección se debe tener en cuenta la necesidad de ofrecer los informes a través de Internet, así como la posibilidad de poder personalizar dichas herramientas para poder adaptarse a los requisitos de la empresa.

7. Informe sobre el desarrollo (*Report Development*)

Proviene de la fase de requisitos, para el usuario final el punto de contacto que tiene con el sistema de almacenamiento de datos son los informes que pueden ver, por lo que juega un papel importante en el éxito del proyecto.

Es importante que el usuario pueda definir sus propios filtros, los métodos de entrega de los informes, etc., todos estos puntos se deben tener en cuenta en el proceso de desarrollo de los informes.

8. Ajuste de rendimiento (*Performance Tuning*)

En esta etapa se verifican y ajustan los rendimientos de los procesos que suelen consumir más tiempo y potencia de cálculo. Existen tres áreas principales donde se puede utilizar este ajuste de rendimiento, los procesos ETL, el procesamiento de consultas y de informes.

9. Aseguramiento de la calidad (*Quality Assurance*)

En esta fase una vez que el equipo de desarrollo indica que todo está listo, se realizan pruebas junto con los usuarios para comprobar que el software cumple con las funciones solicitadas.

10. El despliegue de la producción (*Rolling out to Production*)

Una vez que el equipo de control de calidad da el visto bueno, se debe poner en marcha y dar acceso al DW. Hoy en día al tener los usuarios finales acceso a Internet, suele bastar con facilitarles una URL para que puedan conectarse al mismo.

11. Mantenimiento (*Production Maintenance*)

En esta etapa se incluyen las tareas de mantenimiento necesarias una vez realizada la puesta en producción, como por ejemplo las copias de seguridad.

12. Mejoras incrementales (*Incremental Enhancements*)

Es esta etapa se contemplan las posibles necesidades de mejora., normalmente serán necesidades no recogidas en el proyecto inicial y/o necesidades que han surgido con el paso del tiempo.

3.4. Planificación del proyecto

Para poder realizar una correcta planificación del proyecto, se deben conocer las tareas a realizar, para posteriormente adaptarlas a las fechas propuestas en el plan docente de la asignatura. Por último se debe distribuir la carga de trabajo entre las distintas fases que se han planificado, para que estas queden equilibradas.

3.4.1. Fases del plan docente

En función del enunciado se pueden deducir las siguientes fases:

- Plan de trabajo y análisis preliminar de requisitos.
- Análisis de requisitos, diseño conceptual y técnico.
- Implementación, que a su vez dividimos en:
    - Construcción del almacén de datos.
    - Instalación de la herramienta de explotación del almacén.
    - Construcción de los informes y análisis de la información.

3.4.2. Adaptación de las fases al plan docente

Desglose en función de las distintas fechas de entrega propuestas en el plan docente de la asignatura.

Pec 1: Plan de Trabajo y Análisis preliminar.

- Realización de un primer análisis de requisitos.
- Planificación del trabajo.
- Análisis de los datos de entrada.
- Borrador del análisis multidimensional.

La preparación del hardware y la instalación del software pertenecen a la fase 2, pero en nuestro caso por necesidad la realizaremos en esta primera fase.

Pec 2: Análisis de requerimientos y diseño técnico.

- Análisis detallado de requerimientos.
- Diseño conceptual del modelo multidimensional.
- Diseño físico del modelo multidimensional.
- Diseño proceso ETL. (Extracción, transformación y carga).

Pec 3: Implementación.

- Construcción del almacén de datos.
- Finalización del proceso ETL.
- Instalación de la herramienta de explotación de datos.
- Generación de informes.

Entrega final.

- Elaboración de la memoria.
- Elaboración del proyecto.
- Presentación del proyecto.

3.4.3. Hitos y fechas clave

El calendario de la asignatura señala las siguientes fechas claves:

Nombre	Entrega
Pec 1: Plan de Trabajo y Análisis preliminar	05/10/2011
Pec 2: Análisis de requerimientos y diseño técnico	09/11/2011
Pec 3: Implementación	21/12/2011
Entrega Final	10/01/2012

3.4.4. Planificación de tareas

Para la realización de esta planificación se han tenido en cuenta tanto las fechas del calendario de la asignatura como mi disponibilidad horaria (entre 2 y 3 horas al día de lunes a jueves y entre 4 y 6 horas los viernes, sábados y domingos).

ID	Nombre	Fecha de inicio	Fecha de fin	Duración días
1	**1. Inicio del proyecto**	21/09/11	21/09/11	1 días
2	1.1. Descarga material asignatura y lectura plan docente	21/09/11	21/09/11	1 días
3	**2. Elaboración PEC1. Plan de Trabajo y Análisis preliminar**	22/09/11	05/10/11	**14 días**
4	2.1. Descarga y lectura del enunciado	22/09/11	22/09/11	1 días
5	2.2. Elaborar plan de trabajo	23/09/11	23/09/11	1 días

6	2.3. Consultar bibliografía	24/09/11	24/09/11	1 días
7	2.4. Análisis de requerimientos	25/09/11	25/09/11	1 días
8	2.5. Elaborar borrador PEC1	26/09/11	28/09/11	3 días
9	2.6. Entrega borrador generado	29/09/11	29/09/11	1 días
10	2.7. Preparar hardware e instalar software necesario	30/09/11	01/10/11	2 días
11	2.8. Elaborar PEC1	02/10/11	04/10/11	3 días
12	2.9. Entregar PEC1	05/10/11	05/10/11	1 días
13	**3. Elaboración PEC2. Análisis de requerimientos y diseño técnico**	06/10/11	09/11/11	**34 días**
14	3.1. Lectura y comprensión	06/10/11	06/10/11	1 días
15	3.2. Diseño Almacén de datos	07/10/11	10/10/11	4 días
16	3.3. Diseño lógico	11/10/11	15/10/11	5 días
17	3.4. Extracción datos de origen	16/10/11	21/10/11	6 días
18	3.5. Transformar datos de origen	22/10/11	26/10/11	5 días
19	3.6. Elaborar borrador PEC2	27/10/11	31/10/11	5 días
20	3.7. Entrega borrador generado	01/11/11	01/11/11	1 días
21	3.8. Elaborar PEC2	02/11/11	08/11/11	7 días
22	3.9. Entregar PEC2	09/11/11	09/11/11	1 días
23	**4. Elaboración PEC3. Implementación**	10/11/11	21/12/11	**40 días**
24	4.1. Lectura y comprensión	10/11/11	10/11/11	1 días
25	4.2. Configurar base de datos	11/11/11	14/11/11	4 días
26	4.3. Carga de datos	15/11/11	19/11/11	5 días
27	4.4. Pruebas	20/11/11	27/11/11	8 días
28	4.5. Elaborar informes	28/11/11	06/12/11	9 días
29	4.6. Elaborar borrador PEC3	07/12/11	10/12/11	4 días
30	4.7. Entrega borrador generado	11/12/11	11/12/11	1 días
31	4.8. Elaborar PEC3	12/12/11	20/12/11	7 días
32	4.9. Entregar PEC3	21/12/11	21/12/11	1 días
33	**5. Entrega Final**	22/12/11	10/01/12	**18 días**
34	5.1. Revisión final	22/12/11	24/12/11	3 días
35	5.2. Conclusión final	25/12/11	26/12/11	2 días
36	5.3. Elaborar memoria	27/12/11	01/01/12	6 días
37	5.4. Elaborar presentación	02/01/12	09/01/12	6 días
38	5.5. Entrega final	10/01/12	10/01/12	1 días
39	**6. Debate virtual y defensa del proyecto**	11/01/12	20/01/12	**8 días**
40	6.1. Intervención en el debate	11/01/12	20/01/12	8 días

3.4.5. Diagrama de Gantt

Diagrama de Gantt generado únicamente para este apartado, en el que se han agrupado algunas tareas para conseguir su visualización en una sola imagen. Al final del documento (anexo), se adjuntan imágenes del diagrama de Gantt que coincide exactamente con la planificación de tareas.

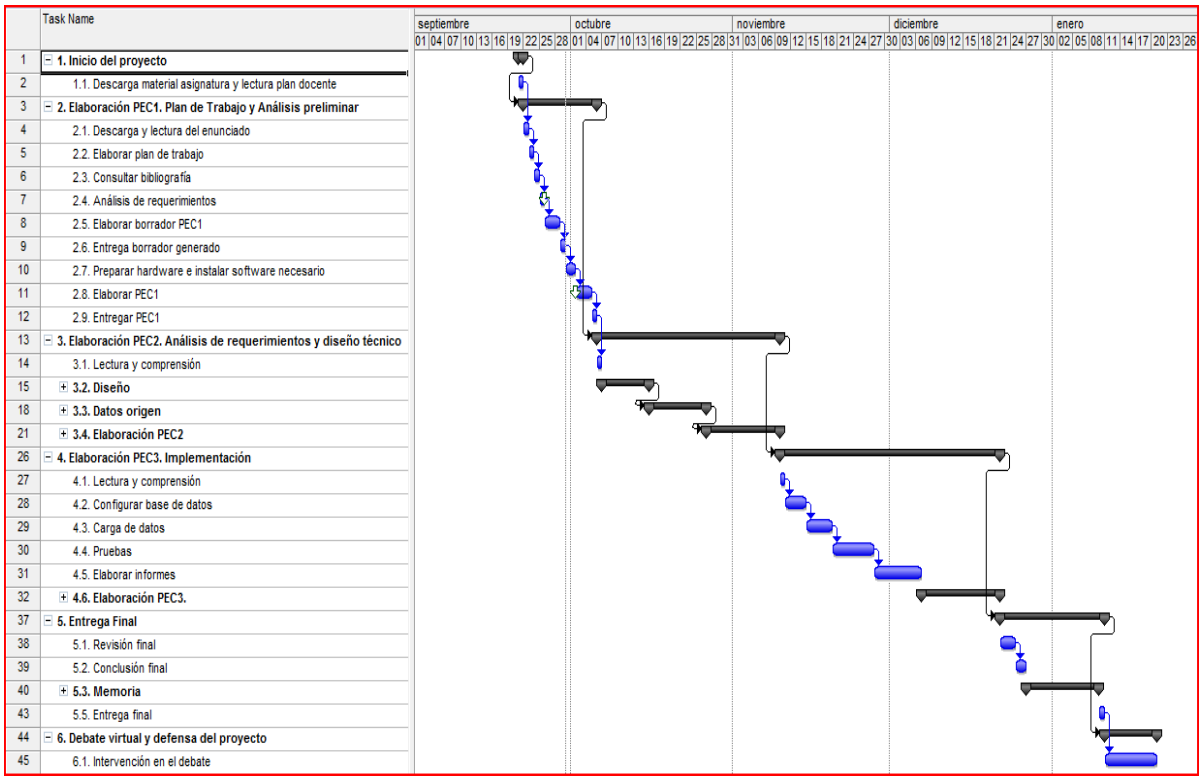


Figura 1. Diagrama de Gantt

3.4.6. Incidencias y riesgos

A continuación se muestran los posibles riesgos e incidencias que se deben tener en cuenta en el desarrollo del proyecto, así como las soluciones previstas para cada uno de ellos.

3.4.7. Problemas por motivos laborales

Por peticiones puntuales, urgentes o desplazamientos laborales que exijan por mi parte una mayor dedicación en mi trabajo.

Para solventar estos problemas se harán ajustes en la programación del TFC, si con esto no es suficiente, los fines de semana se dedicarán más horas de las previstas inicialmente. En caso de problemas graves se emplearán días de vacaciones para poder dedicarle más tiempo al proyecto.

3.4.8. Problemas por motivos de salud

No todos los problemas de salud tendrán una solución fácil, por lo que podemos dividirlos en problemas de salud leves y graves.

Problemas de salud leves: Problemas que conlleven un retraso de 1 a 3 días, estos problemas no deberían afectar, ya que la carga de trabajo del TFC está distribuida de forma que es posible adelantar o retrasar fácilmente el trabajo, por lo que estos problemas son asumibles, sin modificar el plan de trabajo.

Problemas de salud graves: Problemas que conlleven un retraso superior a 4 días, si con la planificación establecida no se puede asumir todo el tiempo perdido, se solucionará cogiendo días de vacaciones.

En cualquier caso si con estas medidas no se llega a cumplir el plan de trabajo se comunicaría al tutor la situación, para que en caso de ser posible entregar directamente las PEC's, sin la entrega previa del borrador, esto permitirá dedicar el tiempo que está previsto para realizar el borrador a recuperar los días perdidos.

3.4.9. Problemas técnicos

A lo larga del TFC se pueden presentar problemas técnicos con el equipo o software empleado. La fase de preparación de hardware e instalación de software se realizará en la PEC1 para adelantar en la medida de lo posible este trabajo.

En caso de que se produzcan problemas que nos impidan trabajar con el ordenador habitual, se dispone de un segundo ordenador que a pesar de ser más lento nos proporcionará poder seguir trabajando en el proyecto sin perder ningún día.

Para evitar pérdidas de información y en caso de ser necesario poder restaurar el TFC en un segundo equipo, se realizarán periódicamente copias de seguridad, que se almacenarán en un disco duro externo.

3.4.10. Solapamiento con otra asignatura

Junto con el proyecto se está cursando la asignatura de TALFII. La solución planteada en caso de solapamiento de trabajo de las dos asignaturas, es la de dar prioridad al TFC, si existen problemas de solapamiento graves se planteará abandonar la otra asignatura.

3.5. Productos obtenidos

En la elaboración del proyecto se han generado además de los documentos que han constituido las PEC1, PEC2 y PEC3, los siguientes documentos o ficheros:

- Script de creación de las tablas del TFC.
- Scripts, funciones y procedimientos para la limpieza y carga de datos.
- Scripts para la actualización mensual de datos.
- Informes con la herramienta Atlas SBI.
- Memoria del proyecto.
- Presentación del proyecto.

3.6. Contenido de los siguientes capitulos

A continuación en el capítulo 4 se pueden ver los diagramas de casos de uso y el diagrama del modelo conceptual. El capítulo 5 detalla la arquitectura y el diseño de la base de datos, así como el diseño y descripción de los informes creados. En el capítulo 6 se muestran las capturas de pantalla de los informes. Por último en los capítulos 7 y 8 tendremos respectivamente las conclusiones y las líneas de evolución futura.

4. ANALISIS

4.1. Diagramas de casos de uso

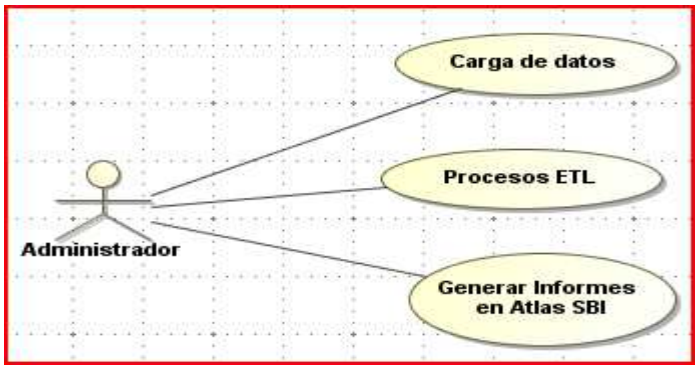


Figura 2. Diagrama casos de uso Administrador

Carga de datos: El administrador se encargará de realizar la carga de datos que le faciliten desde la inmobiliaria en la tabla temporal, donde se tratarán los mismos.

Proceso ETL: El administrador realizará la extracción, tratamiento y carga de los datos en las tablas diseñadas.

Generar informes en Atlas SBI: El administrador generará los informes que sean necesarios para cumplir con los requisitos del cliente.

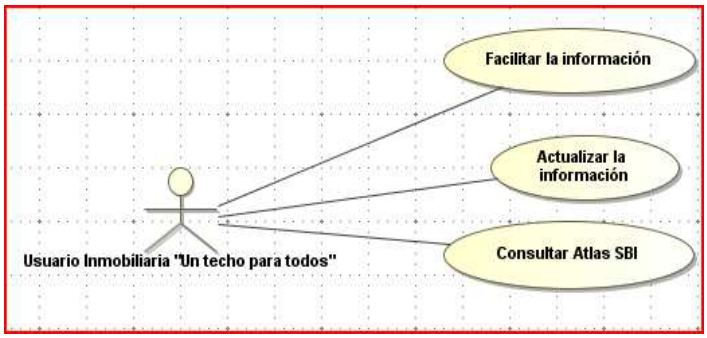


Figura 3. Diagrama casos de uso Usuario inmobiliaria

Facilitar la información: La inmobiliaria debe facilitarnos los datos iniciales a cargar.

Actualizar la información: La inmobiliaria se encargará de actualizar la información y de ejecutar el proceso automatizado mensual que realiza la actualización de la información en la base de datos.

Consultar Atlas SBI: Los usuarios de la inmobiliaria consultarán los informes generados por el administrador.

4.2. Diagrama del modelo conceptual

En este apartado se especifican los pasos seguidos para abordar el diseño conceptual y se detallan los pasos para obtener el modelo en estrella con el que vamos a modelizar nuestro diseño. Para obtener la estrella, debemos detallar los hechos, dimensiones y atributos necesarios de acuerdo con las necesidades del cliente.

Para poder llevar a cabo un análisis de datos basado en dimensiones, será necesario definir un cubo OLAP (*On-Line Analytical Processing*), para lo que es necesario seguir los siguientes pasos.

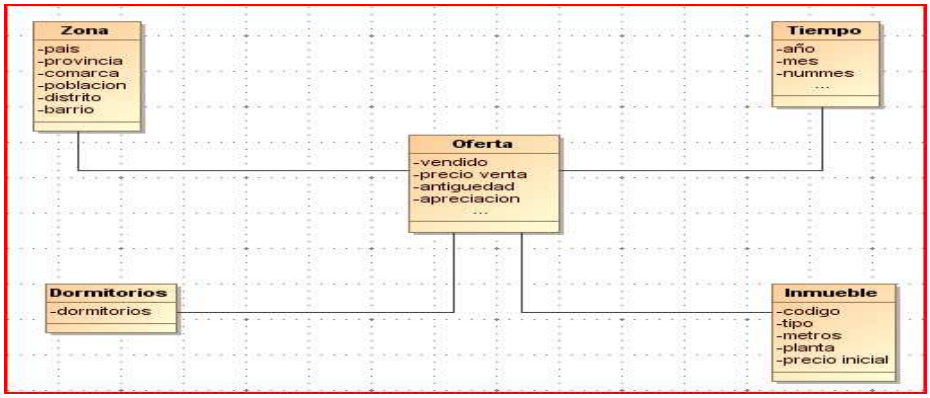


Figura 4. Diagrama conceptual

4.2.1 Identificar el Hecho

En primer lugar debemos identificar el hecho que constituye el centro de la estrella que vamos a diseñar, el cual corresponderá con la actividad que se quiere analizar. Para conseguirlo se analizan los procesos del negocio, e intentamos buscar uno que sea lo bastante identificativo como para concretar la actividad de la empresa.

En nuestro caso al tratarse de una agencia inmobiliaria y con la información de la que disponemos, parece que refleja la venta de inmuebles en diferentes momentos (fechas), junto con información (características) asociada a dichos inmuebles.

En un primer análisis se puede definir el hecho, como la venta de un inmueble, pero si analizamos en detalle los datos, comprobamos que la información nos muestra la evolución de las ofertas de inmuebles en el tiempo, siendo la venta un hecho puntual que no se produce siempre.

Por este motivo se establece como hecho la oferta de inmuebles en una fecha determinada, siendo la venta un caso especial de oferta que trataremos de distinguir de esta a través de un procedimiento que se especificará más adelante.

4.2.2 Encontrar la granularidad adecuada

En este apartado se debe definir el grado de detalle de las celdas que formarán nuestro cubo. Definir correctamente el nivel de detalle es muy importante, será lo que haga que nuestro diseño sea o no eficiente.

En una aproximación inicial podemos inclinarnos por establecer una granularidad que permita un nivel de detalle muy preciso, pero un exceso de detalle puede ocasionar que el sistema tenga que manejar grandes cantidades de información, provocando que las consultas sean lentas y poco eficaces. Por lo que será necesario encontrar la granularidad más baja posible sin llegar a perder funcionalidad.

En nuestro caso, la oferta y venta de inmuebles se encuentra registrada en periodos mensuales. Después de analizar los informes que nos solicitan, parece que estos pueden realizarse con la granularidad que tienen los datos facilitados originalmente por la empresa (un registro por inmueble en oferta y mes).

4.2.3 Escoger las dimensiones a emplear en el análisis

Una vez analizado el hecho, debemos buscar el resto de datos que nos ayuden a definirlo o a situarlo en su contexto. Estos serán los candidatos a las dimensiones que se utilizarán para desgranar la información en nuestro cubo.

Estas dimensiones o catálogos de información, serán las variables a través de las que se analizarán los datos. Es donde se almacenará la información complementaria de cada uno de los registros de la tabla hechos.

En nuestro caso se identifican cuatro dimensiones:

- Tiempo, fecha a la que pertenece la oferta.
- Zona, conjunto de zonas que forman la ubicación geográfica a la pertenece el inmueble.
- Dormitorios, número de dormitorios del inmueble.
- Inmueble, resto de características que constituyen el inmueble (código, planta, metros, etc.).

4.2.4 Establecer los atributos de cada dimensión

Una vez definidas las dimensiones que formarán parte de nuestro análisis, procederemos a establecer los atributos para cada una de ellas, atributos que permitirán filtrar los valores de la dimensión.

Dimensión 'Tiempo', el atributo que represente esta dimensión debe ser la fecha expresada como mes/año.

Dimensión 'Zona', disponemos de cinco niveles de precisión para la definición de zona, pero es la unión de todos ellos la que nos muestra la descripción de la zona geográfica a la que pertenece el inmueble. En esta dimensión se puede establecer el atributo zona como aquel que representa el nombre del área al que pertenece el inmueble.

Dimensión 'Dormitorio', tiene un único atributo, el número de dormitorios del que dispone el inmueble.

Dimensión 'Inmueble' en la que se encuentran los siguientes atributos: código del inmueble, tipo de inmueble, metros cuadrados y planta del inmueble.

4.2.5 Identificar las medidas a emplear

Las medias son los datos relevantes que se quieren estudiar, son los valores que contienen cada celda del cubo, siendo las dimensiones las coordenadas que sirven para identificarla.

Después de analizar los informes que nos solicita la inmobiliaria 'Un techo para todos', se llega a la conclusión de que el precio del inmueble podría considerarse como un atributo del inmueble, pero como este dato variará en el tiempo, finalmente lo consideraremos como una medida de la oferta.

A igual que en el caso anterior, para el informe en que debemos realizar consultas que diferencien entre los inmuebles vendidos y en oferta, se entiende que el atributo que nos indique si está o no vendido puede cambiar en cualquier momento, motivo por el cual se añade en el hecho como un atributo que indicará si el inmueble esta o no vendido.

En los informes también nos solicitan que sepamos separar los inmuebles que han bajado de precio, por lo que añadimos un atributo al hecho (precio inicial del inmueble) que nos permita comprobar esta posible variación.

Para el informe en el que solicitan el tiempo medio de venta, se añade un nuevo atributo (antigüedad) que nos permita comprobar el tiempo (reflejado en meses) que lleva cada inmueble en venta.

Todos estos atributos con valores variables en función de la fecha, se almacenarán en la tabla hechos. Esta información podría verse modificada en el futuro en función del desarrollo del proyecto, en cuyo caso se indicarán y justificarán los posibles cambios.

4.2.6 Definir las celdas

En este apartado se definen que celdas de las que surgen del análisis pueden considerarse interesantes para nuestro modelo. Se tiene que decidir que celdas serán almacenadas y cuáles podemos considerar que son derivadas de otras y por lo tanto no será necesario almacenarlas.

En nuestro caso solo existe una celda que refleja el hecho analizado, por lo que usaremos dicha celda para representar la oferta unitaria de un inmueble en una fecha determinada.

4.2.7 Establecer restricciones de integridad

En este apartado se establecen las bases y restricciones que se deben tener en cuenta. La base nos señalará las dimensiones que son necesarias para definir de forma única una celda.

En nuestro caso, de las cuatro dimensiones que se han definido con anterioridad (tiempo, zona, dormitorio e inmueble), las que definen de forma única una celda, son tiempo e inmueble.

La dimensiones zona y dormitorios, quedan determinadas por la dimensión inmueble, ya que un inmueble siempre estará asociado a la misma zona y tendrá el mismo nº de dormitorios, durante toda su vida. Por lo que para determinar una celda individual, bastará con las dimensiones tiempo e inmueble.

4.2.8 Estudio de viabilidad

Una vez definido el diseño conceptual, debemos calcular el espacio que ocuparían los datos almacenados en el sistema, esto nos permitirá determinar si nuestro modelo es aplicable o no.

Partimos de unos datos en los que nos facilitan datos de aproximadamente 92.555 inmuebles que están en oferta o lo han estado en algún momento. Por otro lado tenemos precios de oferta que hacen referencia a 25 meses, por lo que inicialmente tenemos alrededor de 2.313.875 instancias del hecho con el que vamos a trabajar, teniendo cada una de estas instancias los atributos correspondientes.

A pesar de que la zona y los dormitorios constituyen dimensiones, cada inmueble solo puede pertenecer a una zona geográfica y solo puede tener un nº concreto de dormitorios, por lo que no procede multiplicar por ninguna de estas dimensiones.

En cada celda hemos incluido el precio, el precio inicial del inmueble, su antigüedad y un campo que nos indicará si se encuentra vendido o no. Si hacemos una estimación del nº de bits que necesitamos, se obtiene:

- Antigüedad. 32 bits.
- Precio inicial. 32 bits.
- Precio. 32 bits.
- Vendido (Si/no). 8 bits.
- Índice que apunta a la dimensión dormitorio 24 bits.
- Índice que apunta a la dimensión zona 24 bits.
- Índice que apunta a la dimensión tiempo 24 bits.
- Índice que apunta a la dimensión Inmueble 24 bits.

Total 200 bits (25 bytes) de información asociada a la instancia. Con esta información ya podemos realizar una estimación del tamaño total que necesitaremos, 2.313.875 instancias por 25 bytes por instancia 2.313.875*25=57.846.875/1024=56.491,08 es decir aproximadamente 56 Mbyte.

Este volumen de información es asumible y puede ser manejado por cualquier equipo, y tampoco debe ser un problema para el motor de bases de datos.

No obstante, el espacio que se necesitará inicialmente será menor del indicado, ya que no todos los inmuebles se ofertan durante todo el tiempo (25 meses) disminuyendo de esta forma el nº de instancias. Aunque por otro lado se debe tener en cuenta que es muy probable que en el futuro se disponga de nuevos datos de inmuebles y fechas, lo que aumentaría de nuevo la carga del sistema.

5. DISEÑO

5.1. Arquitectura de Software

En la siguiente imagen se muestra el esquema de la arquitectura del software propuesto para el diseño y la explotación de este proyecto.

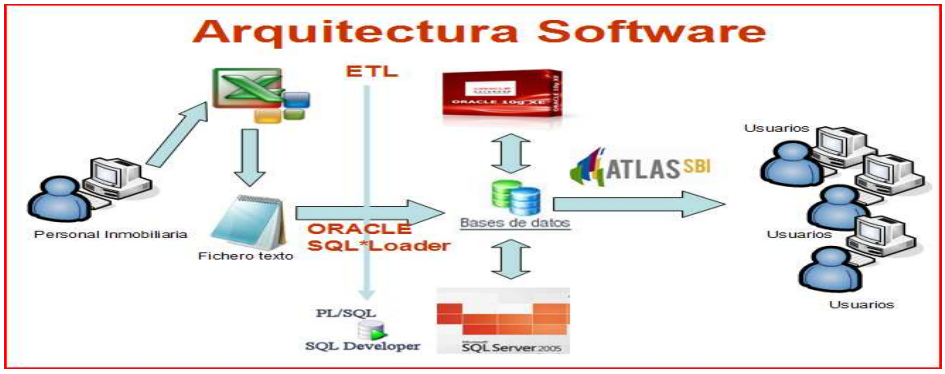


Figura 5. Arquitectura del Software

Según esta arquitectura la información se encuentra inicialmente en ficheros Microsoft Excel, que son convertidos a un fichero de texto por personal de la inmobiliaria, una vez nos facilitan este fichero, procedemos a cargar su contenido en Oracle a través de SQL*Loader. Posteriormente se realiza la transformación y carga de los datos mediante PL/SQL y sentencias SQL, para finalizar realizando los informes y permitiendo el acceso a los mismos a los usuarios a través de la herramienta Atlas SBI.

5.2. Arquitectura de Hardware

La siguiente imagen muestra el esquema de la arquitectura del hardware propuesta.

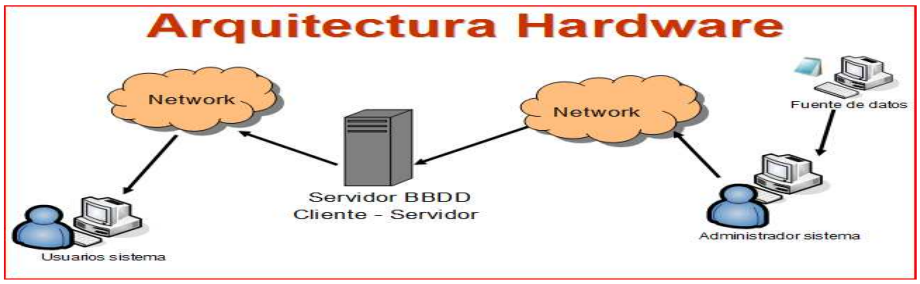


Figura 6. Arquitectura del Hardware

El administrador se encargará de cargar y mantener los datos obtenidos de la fuente en el servidor de base de datos, y los usuarios accederán al servidor para obtener los informes.

5.3. Diseño de la base de datos

Para realizar el diseño de la base de datos, se ha tomado como base el esquema en estrella que se ha definido en documentos anteriores.

La tabla de hechos (OFERTA) enlaza con las tablas de dimensiones (TIEMPO, DORMITORIOS, ZONA e INMUEBLE) mediante las claves primarias de estas tal y como puede verse en el siguiente gráfico.

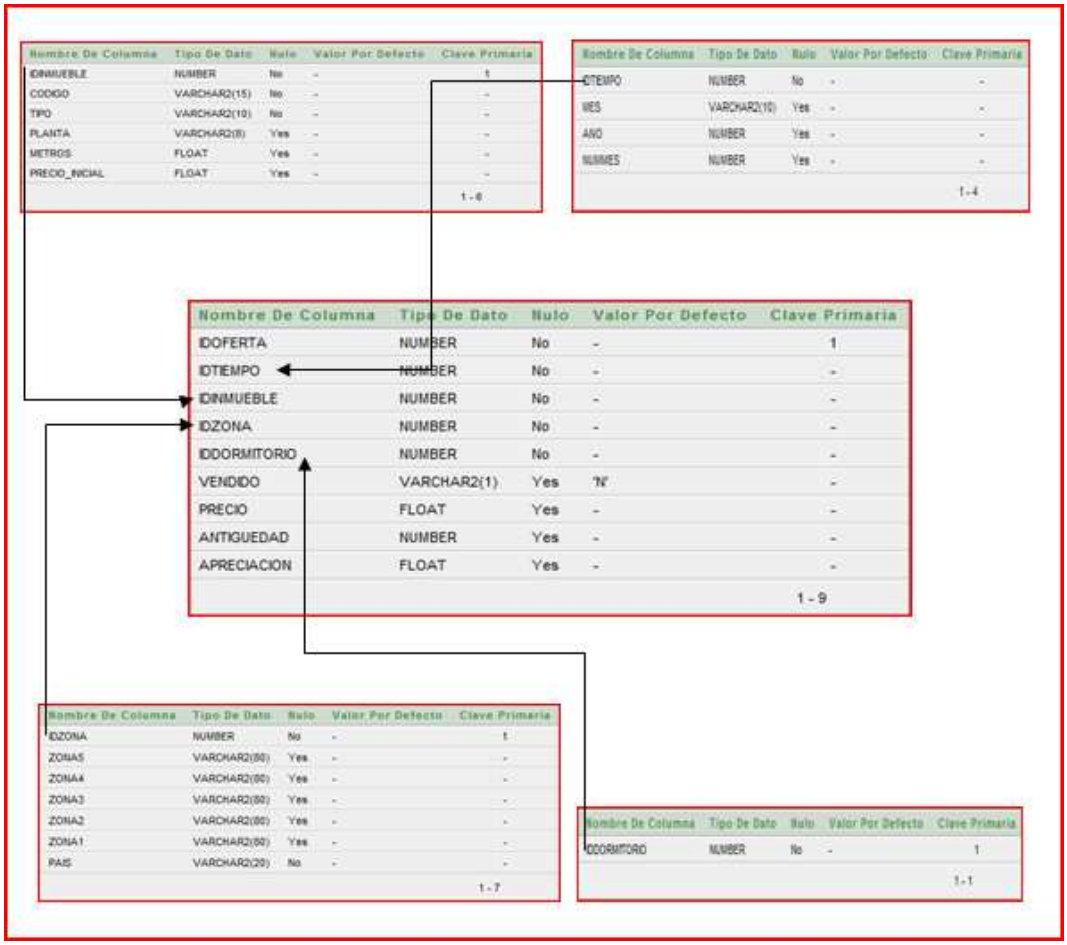


Figura 7. Diseño base de datos

5.3.1. Tabla hechos Oferta

Se trata de la tabla central del esquema, la cual además de contener las claves foráneas por las que podrá conectarse con el resto de tablas (dimensiones), también dispondrá de los campos que identifican si el inmueble está vendido, el importe de la oferta/venta, su antigüedad y la posible apreciación del inmueble desde su puesta en venta.

Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria
IDOFERTA	NUMBER	No	-	1
IDTIEMPO	NUMBER	No	-	-
IDINMUEBLE	NUMBER	No	-	-
IDZONA	NUMBER	No	-	-
IDDORMITORIO	NUMBER	No	-	-
VENDIDO	VARCHAR2(1)	Yes	'N'	-
PRECIO	FLOAT	Yes	-	-
ANTIGUEDAD	NUMBER	Yes	-	-
APRECIACION	FLOAT	Yes	-	-
				1 - 9

Figura 8. Tabla hechos oferta

Los campos de la tabla son:

IDOFERTA: Identificador único de la oferta, a pesar de disponer de otras claves (IdTiempo + IdInmueble) que podrían identificar como única la oferta, se decide establecer este nuevo campo, el cual nos permitirá identificar de forma única a la oferta.

IDTIEMPO: Se trata de una clave foránea que nos permitirá conectar con la tabla de dimensión TIEMPO a través de su clave primaria.

IDINMUEBLE: Se trata de una clave foránea que nos permitirá conectar con la tabla de dimensión INMUEBLE a través de su clave primaria.

IDZONA: Se trata de una clave foránea que nos permitirá conectar con la tabla de dimensión ZONA a través de su clave primaria.

IDDORMITORIO: Se trata de una clave foránea que nos permitirá conectar con la tabla de dimensión DORMITORIOS a través de su clave primaria.

VENDIDO: Este campo nos indica si el inmueble se encuentra vendido o no, por defecto esta inicializado a ‘N’ (no vendido). Aparecerá como ‘S’ (vendido) en el registro de la oferta correspondiente al mes en el que se vende el inmueble.

PRECIO: Este campo contiene el importe de la oferta del inmueble si el anterior campo se encuentra con valor ‘N’, o el importe de venta si se encuentra con valor ‘S’.

ANTIGÜEDAD: Este campo contiene el número de meses que lleva el inmueble en venta.

APRECIACION: Este campo contiene la diferencia positiva o negativa, entre el precio inicial del inmueble (primera oferta que se tiene del mismo) y el precio de la oferta del mes en el que nos encontramos.

5.3.2. Tabla dimensión Inmueble

En esta tabla se almacenan datos de los inmuebles. Se relaciona con la tabla de hechos a través de la clave IDINMUEBLE que a su vez se corresponde con la clave foránea IDINMUEBLE de la tabla de hechos.

Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria
IDINMUEBLE	NUMBER	No	-	1
CODIGO	VARCHAR2(15)	No	-	-
TIPO	VARCHAR2(10)	No	-	-
PLANTA	VARCHAR2(8)	Yes	-	-
METROS	FLOAT	Yes	-	-
PRECIO_INICIAL	FLOAT	Yes	-	-
				1 - 6

Figura 9. Tabla dimensión Inmueble

IDINMUEBLE: Es el índice e identificador único del inmueble, nos permitirá identificar el inmueble de forma única.

CODIGO: Este campo contiene el código identificador del inmueble que nos facilita la empresa 'Un techo para todos'.

TIPO: Este campo contiene la descripción del tipo de inmueble, en nuestro caso puede ser: ático, piso, chalet, estudio o dúplex.

PLANTA: Es el numero de planta en la que se encuentra ubicado el inmueble.

METROS: Numero de metros cuadrados que tiene de superficie el inmueble.

PRECIO_INICIAL: Este campo contiene el precio inicial del inmueble, este importe corresponderá con el de la primera oferta que se tenga del inmueble.

5.3.3. Tabla dimensión Zona

En esta tabla se almacenan las distintas zonas en las que se encuentran ubicados los inmuebles. Se relaciona con la tabla de hechos a través de la clave IDZONA que a su vez se corresponde con la clave foránea IDZONA de la tabla de hechos.

Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria
IDZONA	NUMBER	No	-	1
ZONA5	VARCHAR2(80)	Yes	-	-
ZONA4	VARCHAR2(80)	Yes	-	-
ZONA3	VARCHAR2(80)	Yes	-	-
ZONA2	VARCHAR2(80)	Yes	-	-
ZONA1	VARCHAR2(80)	Yes	-	-
PAIS	VARCHAR2(20)	No	-	-
				1 - 7

Figura 10. Tabla dimensión Zona

IDZONA: Es el índice e identificador único de la zona, nos permitirá identificar de zona única cada una de las zonas que nos facilitan.

ZONA5: En este campo se almacena la descripción correspondiente a la 5ª zona del inmueble, esta zona es equivalente a la descripción del barrio en el que se encuentra ubicado el inmueble.

ZONA4: En este campo se almacena la descripción correspondiente a la 4ª zona del inmueble, esta zona es equivalente a la descripción del distrito en el que se encuentra ubicado el inmueble.

ZONA3: En este campo se almacena la descripción correspondiente a la 3ª zona del inmueble, esta zona es equivalente a la descripción de la población en la que se encuentra ubicado el inmueble.

ZONA2: En este campo se almacena la descripción correspondiente a la 2ª zona del inmueble, esta zona es equivalente a la descripción de la comarca en la que se encuentra ubicado el inmueble.

ZONA1: En este campo se almacena la descripción correspondiente a la 1ª zona del inmueble, esta zona es equivalente a la descripción de la provincia en la que se encuentra ubicado el inmueble.

PAIS: En este campo se almacena la descripción del país al que pertenece el inmueble, en nuestro caso solo puede pertenecer a ‘España’ o ‘Andorra’.

5.3.4. Tabla dimensión Tiempo

En esta tabla se almacenan las fechas para las que existen ofertas para los inmuebles. Se relaciona con la tabla de hechos a través de la clave IDTIEMPO que a su vez se corresponde con la clave foránea IDTIEMPO de la tabla de hechos.

Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria
IDTIEMPO	NUMBER	No	-	1
MES	VARCHAR2(10)	Yes	-	-
AÑO	NUMBER	Yes	-	-
NUMMES	NUMBER	Yes	-	-
				1 - 4

Figura 11. Tabla dimensión Tiempo

IDTIEMPO: Es el índice e identificador único del tiempo, nos permitirá identificar la fecha (mes/año) de la oferta/venta de forma única.

MES: Es el mes en texto al que pertenece la oferta.

AÑO: Es el año en número al que pertenece la oferta.

NUMMES: Es el numero del mes en el año (Enero=1, Febrero=2, etc.).

5.3.5. Tabla dimensión Dormitorios

En esta tabla se almacenan el número de dormitorios de los inmuebles. La tabla se relaciona con la tabla de hechos (OFERTA) mediante la clave IDDORMITORIO que corresponde con la clave foránea IDDORMITORIO de la tabla de hechos.

Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria
IDDORMITORIO	NUMBER	No	-	1
				1 - 1

Figura 12. Tabla dimensión Dormitorios

IDDORMITORIO: Es el índice e identificador único de los dormitorios, contiene la agrupación de los dormitorios que existen en la tabla temporal catalogo.

5.3.6. Tabla temporal CATALOGO

Esta tabla, es una tabla temporal en la que se almacenarán los datos facilitados en el fichero de texto por la empresa 'Un techo para todos'.

Esta tabla nos servirá para la posterior carga de los datos en la tabla de hechos y dimensiones.

Nombre De Columna	Tipo De Dato	Nulo	Valor Por Defecto	Clave Primaria
CODIGO	VARCHAR2(180)	Yes	-	-
ZONA	VARCHAR2(20)	Yes	-	-
TIPO	VARCHAR2(10)	Yes	-	-
PLANTA	VARCHAR2(4)	Yes	-	-
DORMITORIOS	NUMBER	Yes	-	-
METROS	NUMBER	Yes	-	-
PRECIO_ENERO_2006	NUMBER	Yes	-	-
PRECIO_FEBRERO_2006	NUMBER	Yes	-	-
PRECIO_MARZO_2006	NUMBER	Yes	-	-
PRECIO_ABRIL_2006	NUMBER	Yes	-	-
PRECIO_MAYO_2006	NUMBER	Yes	-	-
PRECIO_JUNIO_2006	NUMBER	Yes	-	-
PRECIO_JULIO_2006	NUMBER	Yes	-	-
PRECIO_AGOSTO_2006	NUMBER	Yes	-	-
PRECIO_SEPTIEMBRE_2006	NUMBER	Yes	-	-
PRECIO_OCTUBRE_2006	NUMBER	Yes	-	-
PRECIO_NOVIEMBRE_2006	NUMBER	Yes	-	-
PRECIO_DICIEMBRE_2006	NUMBER	Yes	-	-
PRECIO_ENERO_2007	NUMBER	Yes	-	-
PRECIO_FEBRERO_2007	NUMBER	Yes	-	-
PRECIO_MARZO_2007	NUMBER	Yes	-	-
PRECIO_ABRIL_2007	NUMBER	Yes	-	-
PRECIO_MAYO_2007	NUMBER	Yes	-	-
PRECIO_JUNIO_2007	NUMBER	Yes	-	-
PRECIO_JULIO_2007	NUMBER	Yes	-	-
PRECIO_AGOSTO_2007	NUMBER	Yes	-	-
PRECIO_SEPTIEMBRE_2007	NUMBER	Yes	-	-
PRECIO_OCTUBRE_2007	NUMBER	Yes	-	-
PRECIO_NOVIEMBRE_2007	NUMBER	Yes	-	-
PRECIO_DICIEMBRE_2007	NUMBER	Yes	-	-
PRECIO_ENERO_2008	NUMBER	Yes	-	-

Figura 13. Tabla temporal catalogo

No se considera necesario describir los campos de esta tabla por coincidir con los de las anteriores tablas.

5.4. Diseño y descripción de los informes creados

Los informes se han realizado con la herramienta Business Intelligence Atlas SBI, y aquí se detallan las consultas que se ejecutan en cada uno de los informes solicitados.

1.- Nº inmuebles por zona, tipología y características.

En esta primera consulta se obtiene el número de inmuebles agrupado por zona, tipología y características. En esta consulta se incluyen todos los inmuebles los vendidos y los que no lo están.

```
Select count (distinct inmueble.idinmueble) as NºInmuebles, inmueble.tipo,
zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca, zona.zona3 as
Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio, oferta.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros
From inmueble, zona, oferta
Where inmueble.idinmueble=oferta.idinmueble and zona.idzona=oferta.idzona
Group by inmueble.tipo, zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, oferta.iddormitorio, inmueble.planta,
inmueble.metros
```

Con la segunda consulta obtenemos el listado agrupado de los inmuebles por zona, tipología y características. En este listado tendremos los siguientes datos del inmueble: código, planta, metros y precio inicial.

```
Select inmueble.codigo, zona.pais as Pais, zona.zona1 as Provincia,
zona.zona2 as Comarca, zona.zona3 as Poblacion, zona.zona4 as Distrito,
zona.zona5 as Barrio, inmueble.tipo, dormitorios.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros, precio_inicial
From inmueble, oferta, Zona, Dormitorios
Where inmueble.idinmueble = oferta.idinmueble and
oferta.idzona=zona.idzona and oferta.iddormitorio = dormitorios.iddormitorio
and zona.idzona=$[ControlParameter:Idzona]$ and
inmueble.tipo=$[ControlParameter:tipo]$ and  dormitorios.iddormitorio
=$[ControlParameter:dormitorios]$
Group By zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, dormitorios.iddormitorio, inmueble.planta,
inmueble.metros, precio_inicial, inmueble.codigo
```

2.- Inmuebles que han bajado de precio.

En este informe obtenemos el listado de los inmuebles que han bajado de precio, respecto a su precio inicial. Este listado tendrá la relación de todos los inmuebles que han bajado de precio con su código, zona, tipo, planta, dormitorios y metros. El filtro se realiza a través el campo apreciación para aquellos que tengan el campo con valor menor que '0', en este campo en la carga se almacena el valor del precio inicial del inmueble menos el precio de la oferta.

```
Select inmueble.codigo, zona.pais as Pais, zona.zona1 as Provincia,
zona.zona2 as Comarca, zona.zona3 as Poblacion, zona.zona4 as Distrito,
zona.zona5 as Barrio, inmueble.tipo, inmueble.planta, dormitorios.iddormitorio
as NºDormitorios, inmueble.metros
From inmueble, oferta, Zona, Dormitorios
Where inmueble.idinmueble = oferta.idinmueble and oferta.idzona =
zona.idzona and oferta.iddormitorio = dormitorios.iddormitorio and
oferta.apreciacion<0
Group By zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo,  inmueble.codigo, dormitorios.iddormitorio,
inmueble.planta, inmueble.metros
Order by zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5
```

En este otro apartado se obtienen los mismos inmuebles que en la consulta anterior, pero agrupados por tiempo.

```
Select inmueble.codigo, zona.pais as Pais, zona.zona1 as Provincia,
zona.zona2 as Comarca, zona.zona3 as Poblacion, zona.zona4 as Distrito,
zona.zona5 as Barrio, inmueble.tipo, inmueble.planta, dormitorios.iddormitorio
as NºDormitorios, inmueble.metros, oferta.apreciacion, tiempo.año,
tiempo.nummes, oferta.precio as Precio_oferta, inmueble.precio_inicial,
oferta.idoferta
From inmueble, oferta, Zona, Dormitorios, tiempo
Where inmueble.idinmueble = oferta.idinmueble and oferta.idzona =
zona.idzona and oferta.iddormitorio = dormitorios.iddormitorio and
oferta.apreciacion<0 and oferta.idtiempo=tiempo.idtiempo and
$[PeriodicalFilter:Year(tiempo.año); Month(tiempo.nummes)]$
Group By zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, inmueble.codigo, dormitorios.iddormitorio,
inmueble.planta, inmueble.metros, oferta.apreciacion, tiempo.año,
tiempo.nummes, oferta.precio, inmueble.precio_inicial, oferta.idoferta
Order by zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.codigo
```

3.- Precios máximos, mínimos y medios por zona, tipología y características.

Se obtienen la relación de precios máximos, mínimos y medios por zona, tipología y características. Para obtener estos valores, se utilizan las funciones ‘max’, ‘min.’ y ‘avg’ respectivamente. Estas funciones calculan directamente los valores que buscamos.

```
Select zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca,
zona.zona3 as Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio,
inmueble.tipo, max (oferta.precio) as "OFERTA MAX.", min (oferta.precio) as
"OFERTA MIN.", Round(avg (oferta.precio), 2) as "OFERTA MED.",
oferta.iddormitorio as Nº_Dormitorios, zona.idzona
From Oferta, Inmueble, Zona
Where Oferta.idInmueble = Inmueble.idinmueble and Oferta.idZona =
Zona.idzona
Group by zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, inmueble.tipo, oferta.iddormitorio, zona.idzona
Order by zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, oferta.iddormitorio, zona.idzona
```

4.- Precios de venta real máximos, mínimos y medios por zona, tipología y características.

Obtenemos los precios de venta real máximos, mínimos y medios por zona, tipología y características. Al igual que en la consulta anterior se utilizarán las funciones ‘max’, ‘min’ y ‘avg’, para obtener los valores máximos, mínimos y medios. Esta consulta se realiza solo sobre los inmuebles que se han vendido, por eso en las clausulas where se incluye que el campo vendido debe ser igual a ‘S’ que es lo que indica que el inmueble se ha vendido.

```
Select zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca,
zona.zona3 as Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio,
inmueble.tipo, max (oferta.precio) as "VENTA MAX.", min (oferta.precio) as
"VENTA MIN.", Round(avg (oferta.precio), 2) as "VENTA MED.",
oferta.iddormitorio as Nº_Dormitorios, zona.idzona
From Oferta, Inmueble, Zona, tiempo
Where Oferta.idInmueble = Inmueble.idinmueble and Oferta.idZona =
Zona.idzona and oferta.idtiempo=tiempo.idtiempo and oferta.vendido='S'
Group by inmueble.tipo, zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, oferta.iddormitorio, zona.idzona
Order by zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, inmueble.tipo, oferta.iddormitorio, zona.idzona
```

5.- Diferencia entre precios ofrecidos y de venta real.

Obtenemos la diferencia entre los precios ofrecidos y de venta real. En la consulta se tienen en cuenta solo los inmuebles vendidos (campo vendido igual a 'S') y que su precio de venta se diferente al precio inicial (campo apreciación diferente de '0'), estas dos clausulas nos permiten obtener los resultados que nos solicitan.

```
Select  inmueble.codigo, tiempo.año, tiempo.mes, zona.pais, zona.zona1 as
Provincia, zona.zona2 as Comarca, zona.zona3 as Poblacion, zona.zona4 as
Distrito, zona.zona5 as Barrio, inmueble.tipo, oferta.precio as Precio_Venta,
inmueble.precio_inicial, oferta.apreciacion as Diferencia, oferta.iddormitorio as
Nº_Dormitorios, inmueble.planta, inmueble.metros
From Oferta, Inmueble, Zona, tiempo
Where Oferta.idInmueble = Inmueble.idinmueble and Oferta.idZona =
Zona.idzona and oferta.idtiempo=tiempo.idtiempo and oferta.vendido='S' AND
oferta.apreciacion<>0
Group by tiempo.año, tiempo.mes, inmueble.tipo, zona.pais, zona.zona1,
zona.zona2, zona.zona3, zona.zona4, zona.zona5, oferta.iddormitorio,
oferta.apreciacion,  inmueble.codigo, oferta.precio, inmueble.precio_inicial,
inmueble.planta, inmueble.metros
Order by zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, oferta.iddormitorio
```

6.- Metros cuadrados máximos, mínimos y medios por zona, tipología y características.

Obtenemos los metros cuadrados máximos, mínimos y medios por zona, tipología y características. Para obtener estos valores se utilizan las mismas funciones que en las consultas anteriores. En esta consulta se incluyen todos los inmuebles.

```
Select zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca,
zona.zona3 as Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio,
inmueble.tipo, oferta.iddormitorio as Nº_Dormitorios, max (inmueble.metros)
as Metros_Maximos, min (inmueble.metros) as Metros_Minimos, Round(avg
(inmueble.metros), 2) as Metros_Medios, zona.idzona
From Oferta, Inmueble, Zona, tiempo
Where Oferta.idInmueble = Inmueble.idinmueble and Oferta.idZona =
Zona.idzona and oferta.idtiempo=tiempo.idtiempo
and inmueble.metros is not null
Group by inmueble.tipo, zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, oferta.iddormitorio, zona.idzona
Order by zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, oferta.iddormitorio, zona.idzona
```

7.- Distribución.

Con esta consulta podremos obtener las distintas distribuciones por nº de habitaciones, zona, tipología y características.

```
Select count (distinct inmueble.idinmueble) as NºInmuebles, inmueble.tipo,
zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca, zona.zona3 as
Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio, oferta.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros
From inmueble, zona, oferta
Where inmueble.idinmueble=oferta.idinmueble and zona.idzona=oferta.idzona
Group by inmueble.tipo, zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, oferta.iddormitorio, inmueble.planta,
inmueble.metros
```

8.- Tiempo medio de venta por zona, tipología y características.

A través de esta primera consulta, obtenemos el tiempo medio que el inmueble está en venta, es decir la media de tiempo (en nº de meses) que el inmueble se encuentra en oferta. El campo que utilizamos para filtrar si un inmueble se ha vendido es oferta.vendido='N', que indica que el inmueble no se ha vendido.

```
Select inmueble.codigo, zona.pais as Pais, zona.zona1 as Provincia,
zona.zona2 as Comarca, zona.zona3 as Poblacion, zona.zona4 as Distrito,
zona.zona5 as Barrio, inmueble.tipo, dormitorios.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros, precio_inicial as "PRECIO
INICIAL", oferta.precio as "PRECIO VENTA", oferta.apreciacion,
oferta.vendido, MAX (oferta.idoferta), oferta.antiguedad
From inmueble, oferta, Zona, Dormitorios
Where inmueble.idinmueble = oferta.idinmueble and oferta.idzona =
zona.idzona and oferta.iddormitorio = dormitorios.iddormitorio and
zona.idzona=$[ControlParameter:Idzona]$ and
inmueble.tipo=$[ControlParameter:tipo]$ and
dormitorios.iddormitorio=$[ControlParameter:dormitorios]$ and
oferta.vendido='N'
Group By zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, dormitorios.iddormitorio, inmueble.planta,
inmueble.metros, precio_inicial, inmueble.codigo, oferta.vendido, oferta.precio,
oferta.apreciacion, oferta.antiguedad
Order by inmueble.codigo, oferta.antiguedad
```

Con la segunda de las consultas de este informe, se obtiene el tiempo medio que el inmueble tarda en venderse. Es decir la media de tiempo (en nº de meses) que transcurre hasta que el inmueble se vende. El campo que utilizamos para filtrar si un inmueble se ha vendido es oferta.vendido='S' que indica que se encuentra vendido.

```
Select inmueble.codigo, zona.pais as Pais, zona.zona1 as Provincia,
zona.zona2 as Comarca, zona.zona3 as Poblacion, zona.zona4 as Distrito,
zona.zona5 as Barrio, inmueble.tipo, dormitorios.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros,  precio_inicial as "PRECIO
INICIAL", oferta.precio as "PRECIO VENTA", oferta.apreciacion,
oferta.vendido, MAX (oferta.idoferta), oferta.antiguedad
From inmueble, oferta, Zona, Dormitorios
Where inmueble.idinmueble = oferta.idinmueble and oferta.idzona =
zona.idzona and oferta.iddormitorio = dormitorios.iddormitorio and
oferta.vendido='S' and zona.idzona=$[ControlParameter:Idzona]$ and
inmueble.tipo=$[ControlParameter:tipo]$ and
dormitorios.iddormitorio=$[ControlParameter:dormitorios]$
Group By zona.pais, zona.zona1, zona.zona2, zona.zona3, zona.zona4,
zona.zona5, inmueble.tipo, dormitorios.iddormitorio, inmueble.planta,
inmueble.metros, precio_inicial, inmueble.codigo, oferta.vendido, oferta.precio,
oferta.apreciacion, oferta.antiguedad
```

9.- Nº inmuebles vendidos y existentes por zona, tipología y características.

Con esta primera consulta, obtenemos el nº de inmuebles vendidos y por zona, tipología y características. El campo que utilizamos para filtrar si un inmueble se ha vendido es oferta.vendido='S' que indica que se encuentra vendido.

```
Select count (distinct inmueble.idinmueble) as NºInmuebles, inmueble.tipo,
zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca, zona.zona3 as
Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio, oferta.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros, inmueble.codigo
From inmueble, zona, oferta
Where inmueble.idinmueble=oferta.idinmueble and zona.idzona=oferta.idzona
and oferta.vendido='S'
Group by inmueble.tipo, zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, oferta.iddormitorio, inmueble.planta,
inmueble.metros, inmueble.codigo
```

Con esta consulta obtenemos los inmuebles existentes por zona, tipología y características. Es decir aquellos inmuebles que siguen en oferta y que por lo tanto no se han vendido. En esta ocasión utilizamos un doble filtro para localizar los inmuebles que todavía no se han vendido, por un lado los inmuebles con oferta.vendido='N' y que además no tengan oferta.vendido='S', de esta forma obtenemos los inmuebles que todavía se encuentran en venta.

```
Select count (distinct inmueble.idinmueble) as NºInmuebles, inmueble.tipo,
zona.pais, zona.zona1 as Provincia, zona.zona2 as Comarca, zona.zona3 as
Poblacion, zona.zona4 as Distrito, zona.zona5 as Barrio, oferta.iddormitorio as
NºDormitorios, inmueble.planta, inmueble.metros, inmueble.codigo
From inmueble, zona, oferta
Where inmueble.idinmueble=oferta.idinmueble and zona.idzona=oferta.idzona
and oferta.vendido='N' and inmueble.idinmueble not in (select
inmueble.idinmueble from inmueble, zona, oferta Where
inmueble.idinmueble=oferta.idinmueble and zona.idzona=oferta.idzona and
oferta.vendido='S')
Group by inmueble.tipo, zona.pais, zona.zona1, zona.zona2, zona.zona3,
zona.zona4, zona.zona5, oferta.iddormitorio, inmueble.planta,
inmueble.metros, inmueble.codigo
```

10.- Piso-Tipo español y andorrano.

Otro de los requerimientos del cliente es el de determinar cuál es el piso-tipo español y andorrano, por tipología, características y precio inicial, tomando como base los datos que nos facilitan en el fichero de texto.

Para obtener este dato, podemos utilizar distintos criterios (media aritmética, mediana, moda), pero en nuestro caso nos hemos decantado por utilizar la moda (valor que se produce con mayor frecuencia), ya que este criterio se adapta perfectamente al requerimiento que nos realiza el cliente.

Obtendremos la moda, calculando la frecuencia con la que aparece cada elemento y seleccionando el valor más alto que obtengamos. Oracle nos proporciona una función estadística (STATS_MODE) que nos permite calcular de forma automática este valor sin necesidad de realizar ninguna otra operación.

A continuación se detallan los scripts que se ejecutarán para obtener esta información.

Para obtener el piso-tipo español y andorrano, teniendo en cuenta todos los datos de los que dispone la inmobiliaria y agrupando los mismos solo por el país, se ejecutará la siguiente consulta.

```
Select zona.pais, stats_mode(inmueble.tipo) as "Moda por tipología",
stats_mode(inmueble.metros) as "Moda por metros",
stats_mode(inmueble.precio_inicial) as "Moda por precio inicial",
stats_mode(inmueble.planta) as "Moda por
planta",stats_mode(oferta.iddormitorio)  as "Moda por dormitorios"
From oferta, inmueble, zona
Where inmueble.idinmueble=oferta.idinmueble and
zona.idzona=oferta.idzona and oferta.antiguedad=1
Group by zona.pais
```

A continuación se detalla la información obtenida a través de estos criterios.

País	Tipología	Dormitorios	Superficie	Planta	Precio Inicial
España	Pisos	3	90 m^2	1	240.000 €
Andorra	Pisos	3	100 m^2	1	210.000 €

Para obtener la misma información, pero agrupada también por el año y por el mes de forma que el usuario final tenga la posibilidad de seleccionar el espacio temporal con el que desea trabajar, se ejecutará la siguiente consulta.

```
Select zona.pais, tiempo.año, tiempo.mes, stats_mode(inmueble.tipo) as
"Moda por tipología", stats_mode(inmueble.metros) as "Moda por metros",
stats_mode(inmueble.precio_inicial) as "Moda por precio inicial",
stats_mode(inmueble.planta) as "Moda por planta",
stats_mode(oferta.iddormitorio)  as "Moda por dormitorios", tiempo.idtiempo
From oferta, inmueble, zona, tiempo
Where inmueble.idinmueble=oferta.idinmueble and zona.idzona=oferta.idzona
and oferta.idtiempo=tiempo.idtiempo and oferta.antiguedad=1
Group by zona.pais, tiempo.año, tiempo.mes, tiempo.idtiempo
Order by zona.pais, tiempo.idtiempo asc
```

6. CAPTURAS DE PANTALLA DE LOS INFORMES

A continuación se muestran las capturas de pantalla de cada uno de los informes que han sido implementados a petición del cliente.

6.1. Nº inmuebles por zona, tipología y características

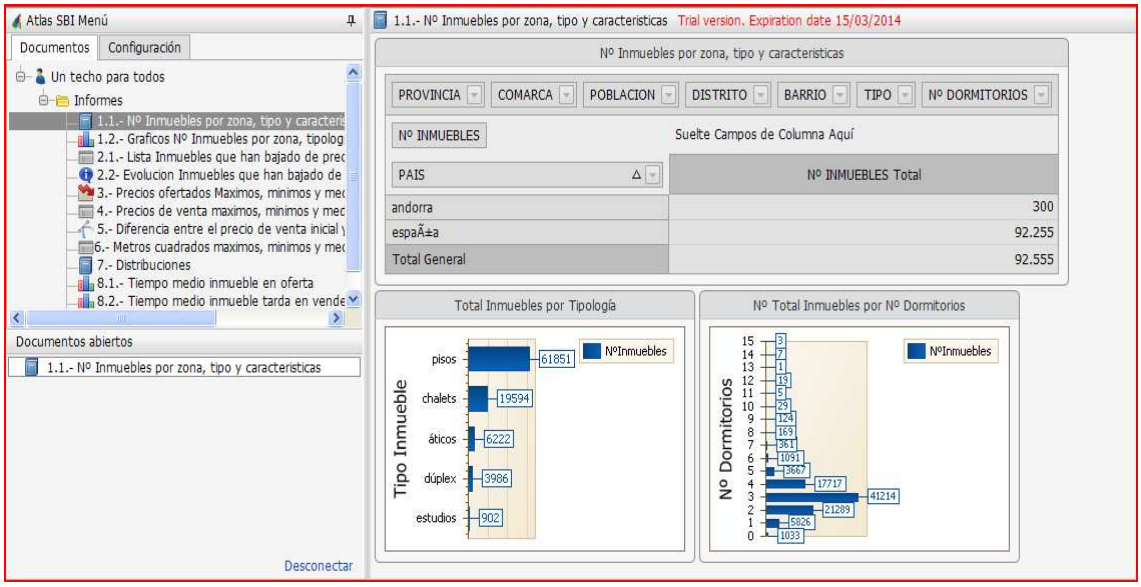


Figura 14a. Nº inmuebles por zona, tipología y características

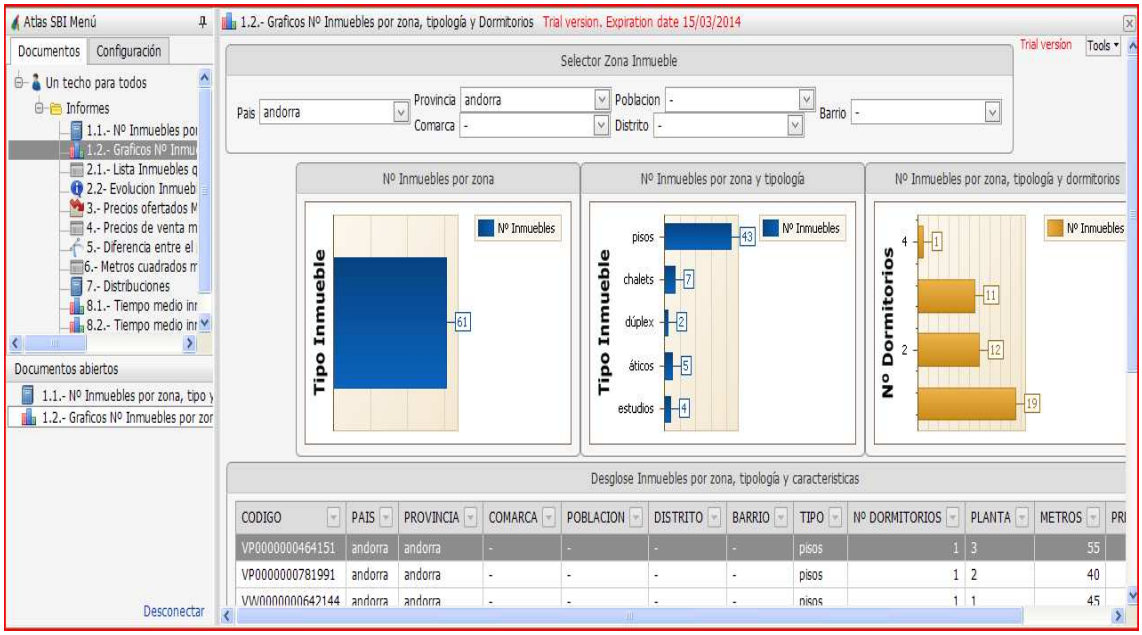


Figura 14b. Desglose inmuebles por zona, tipología y características

6.2. Inmuebles que han bajado de precio

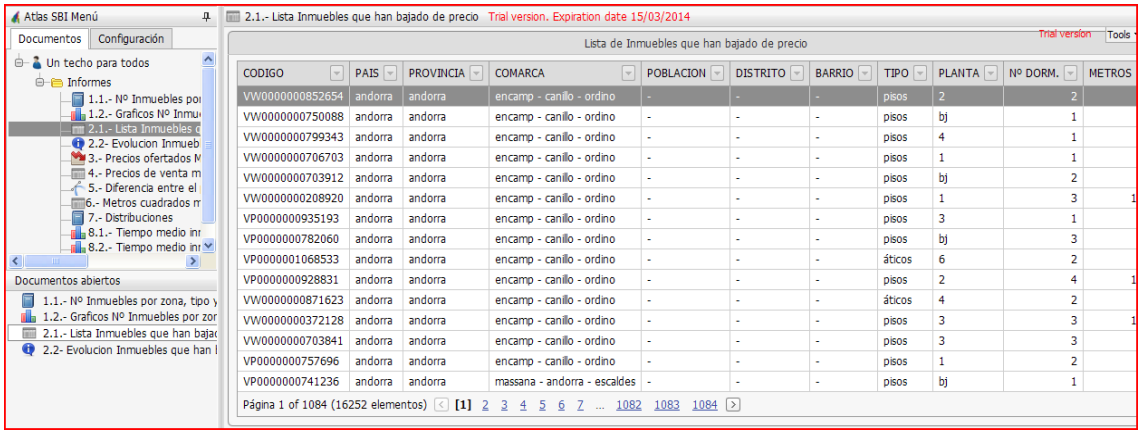


Figura 15a. Relación de Inmuebles que han bajado de precio

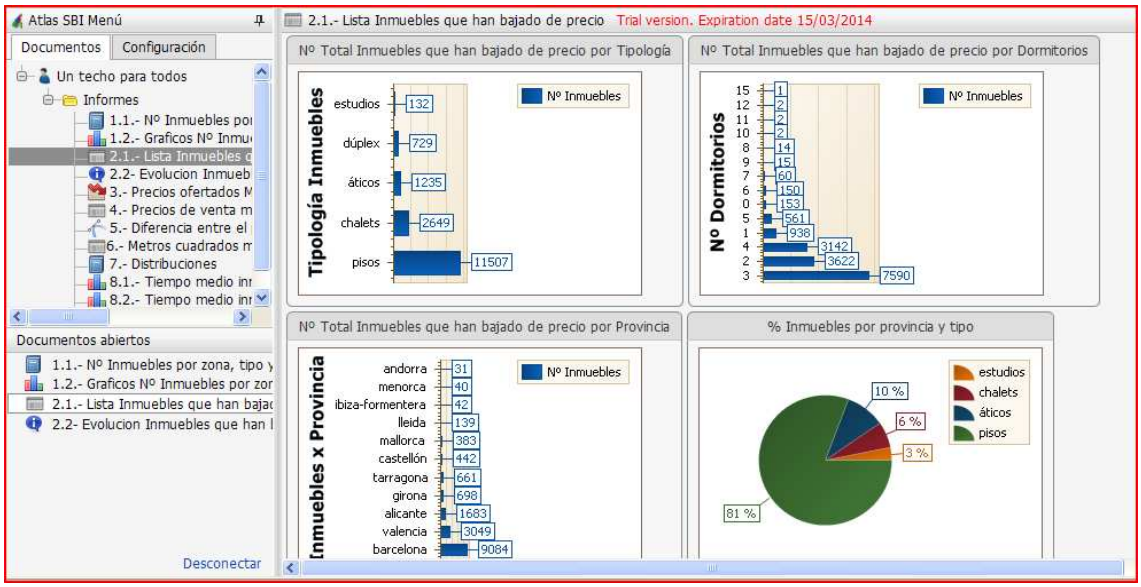


Figura 15b. Gráficos inmuebles que han bajado de precio

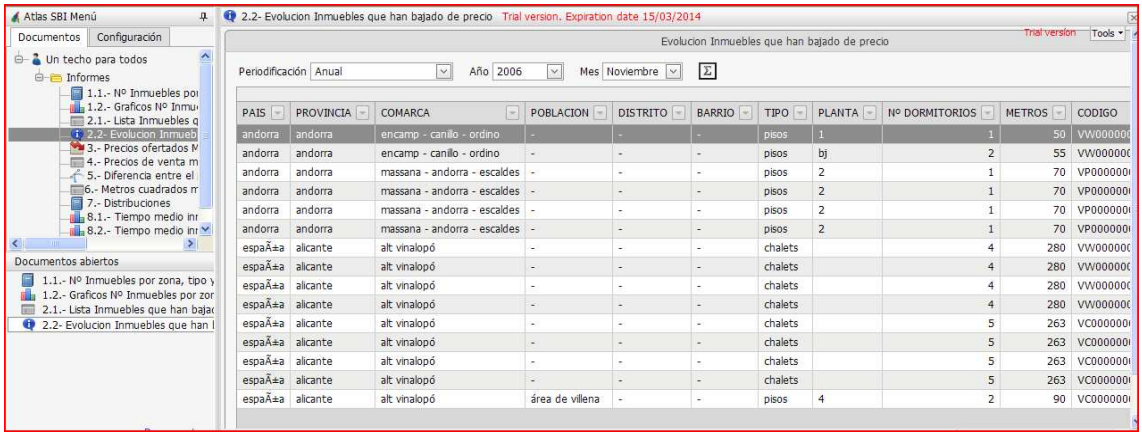


Figura 15c. Desglose inmuebles que han bajado de precio

6.3. Precios máximos, mínimos y medios por zona, tipología y características

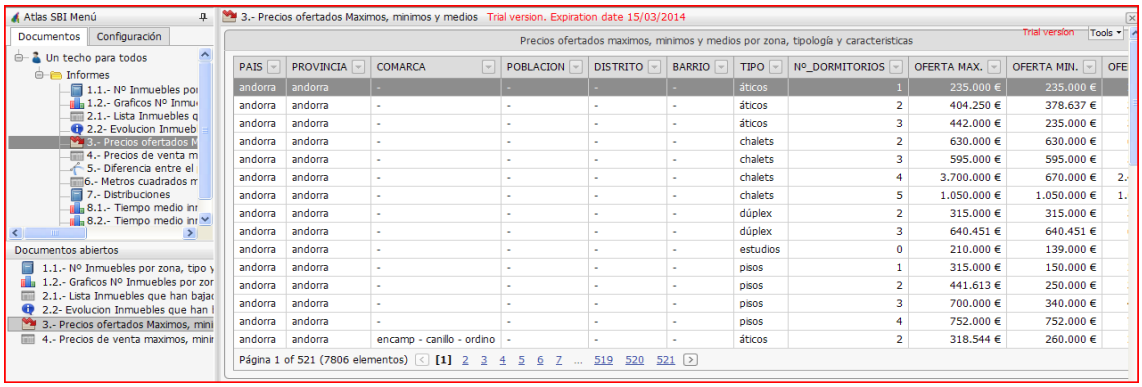


Figura 16a. Precios máximos, mínimos y medios por zona, tipología y características

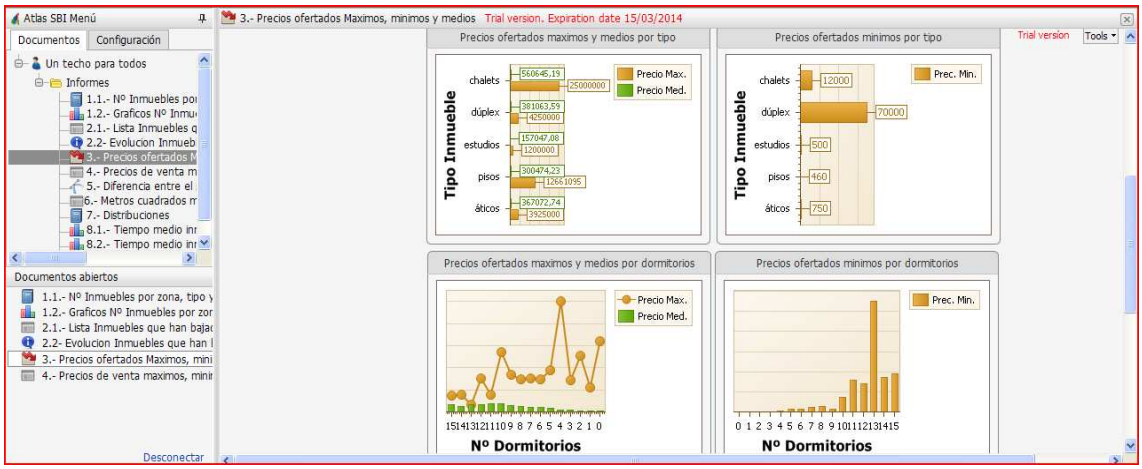


Figura 16b. Gráficas precios máximos, mínimos y medios por zona, tipo y carac.

6.4. Precios de venta real máximos, mínimos y medios por zona, tipología y características

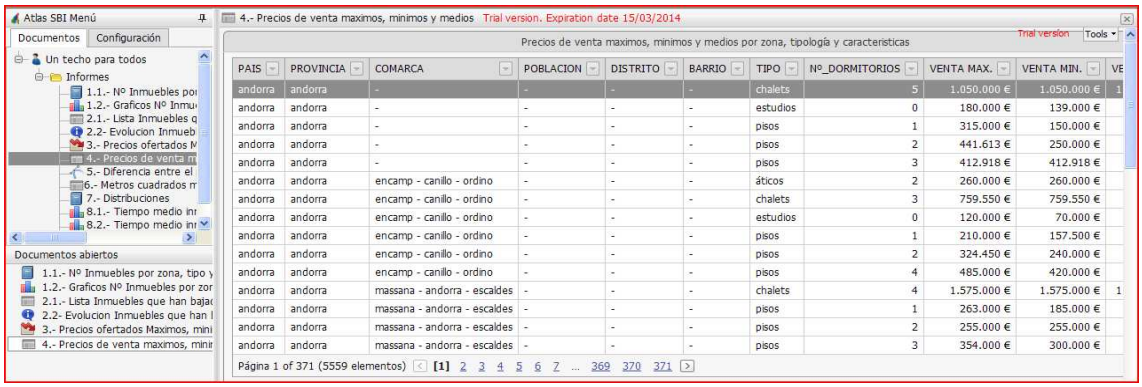


Figura 17a. Precios venta real máximos, mínimos y medios por zona, tipo y carac.

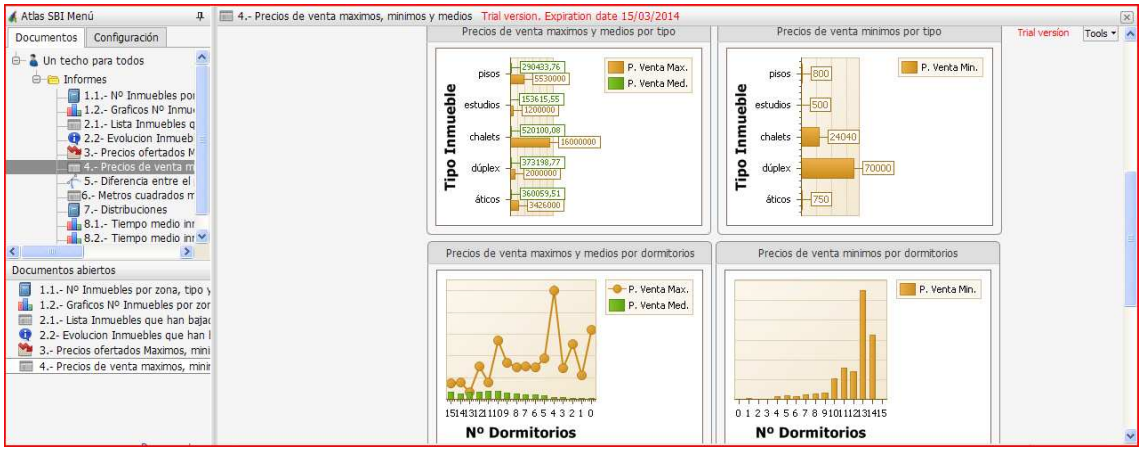


Figura 17b. Precios venta real máximos, mínimos y medios por zona, tipo y carac.

6.5. Diferencia entre precios ofrecidos y de venta real

Figura 18a. Diferencia entre precios ofrecidos y de venta real

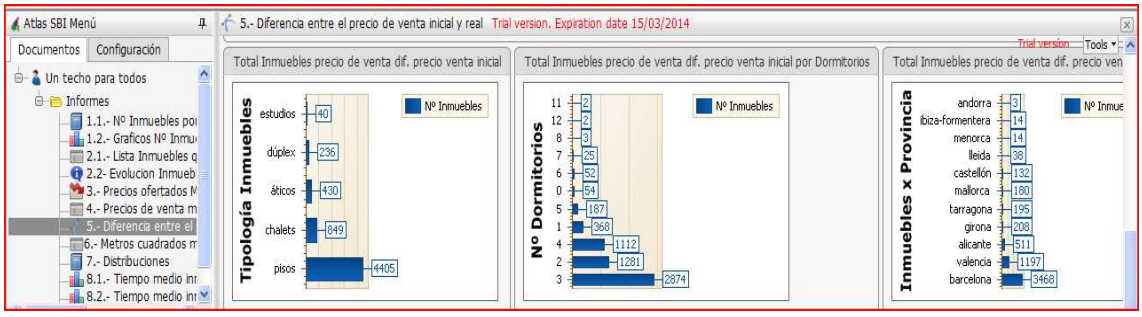


Figura 18b. Gráficos diferencia entre precios ofrecidos y de venta real

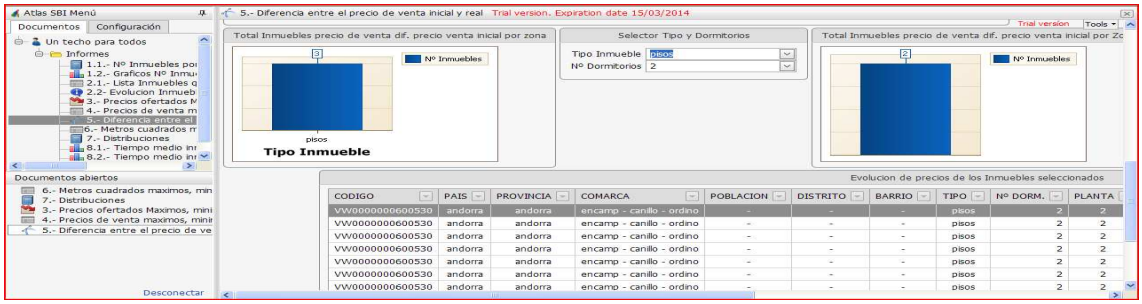


Figura 18c. Desglose diferencia entre precios ofrecidos y de venta real

6.6. Metros cuadrados máximos, mínimos y medios por zona, tipología y características

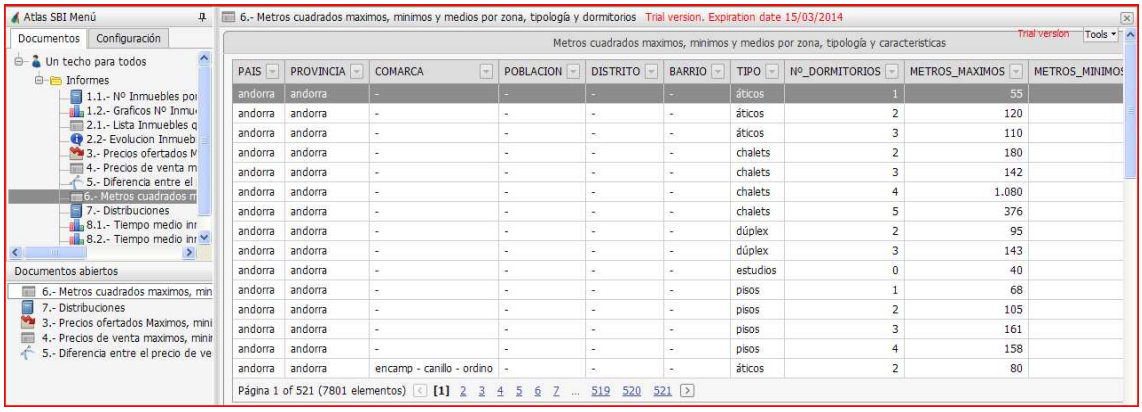


Figura 19a. Metros cuadrados máximos, mínimos y medios por zona, tipo y carac.

Figura 19b. Gráficos metros máximos, mínimos y medios por zona, tipo y carac.

6.7. Distribución

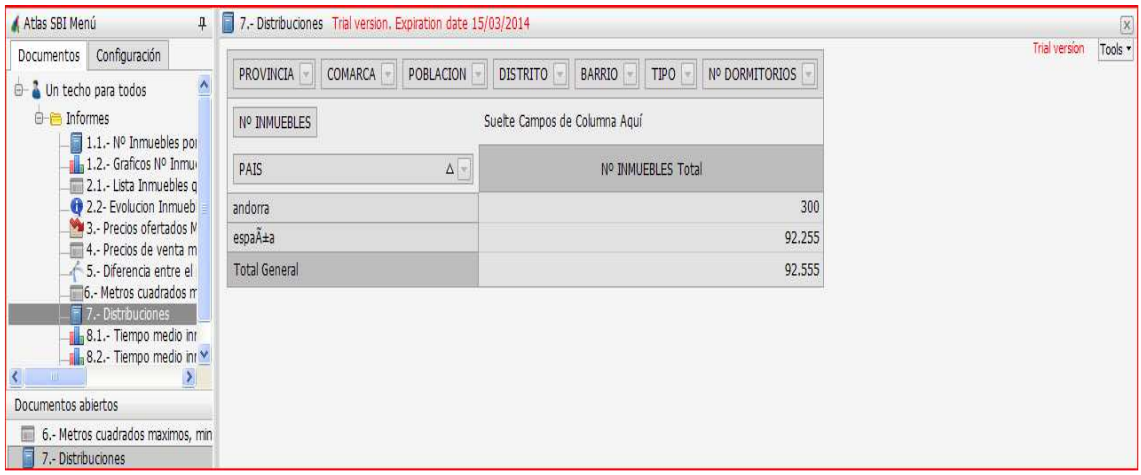


Figura 20. Distribución

6.8. Tiempo medio de venta por zona, tipología y características

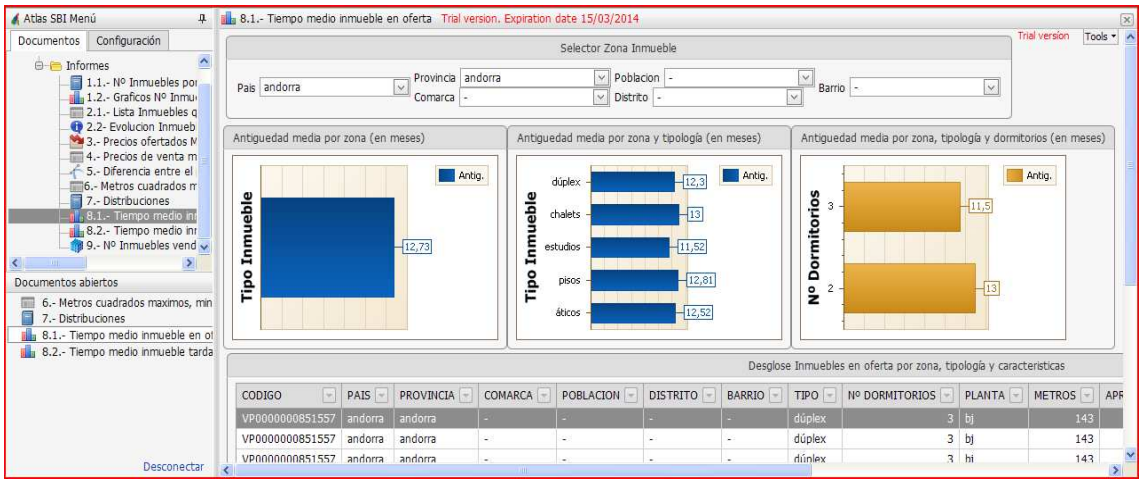


Figura 21a. Tiempo medio en oferta por zona, tipología y características

Figura 21b. Tiempo medio de venta por zona, tipología y características

6.9. Nº inmuebles vendidos y existentes por zona, tipología y características

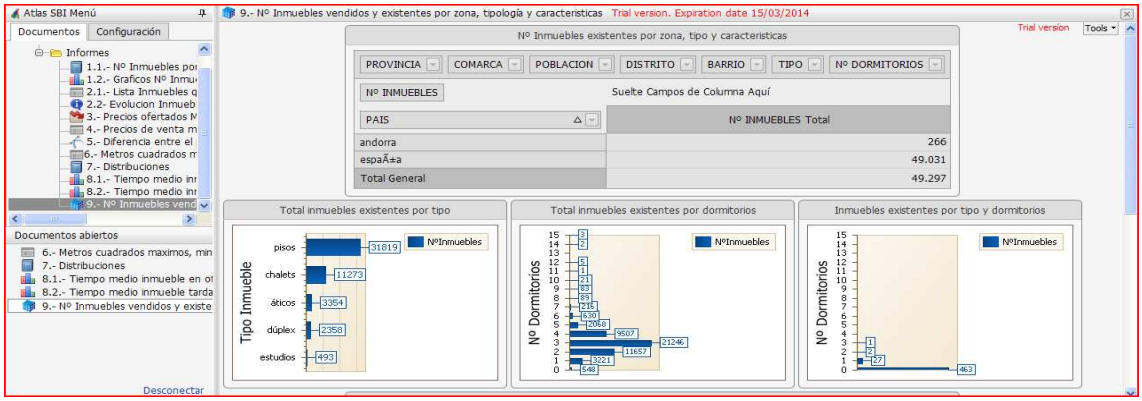


Figura 22a. Nº inmuebles vendidos y existentes por zona, tipología y características

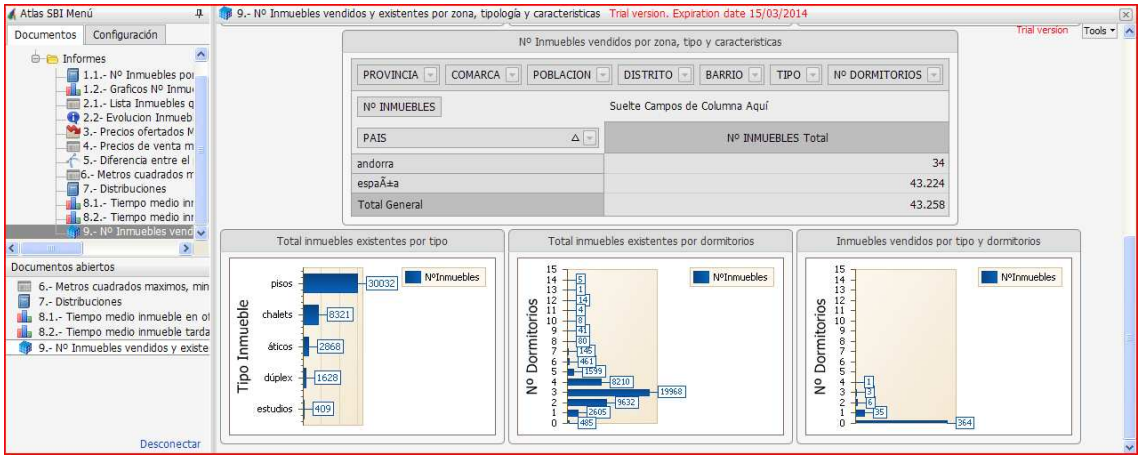
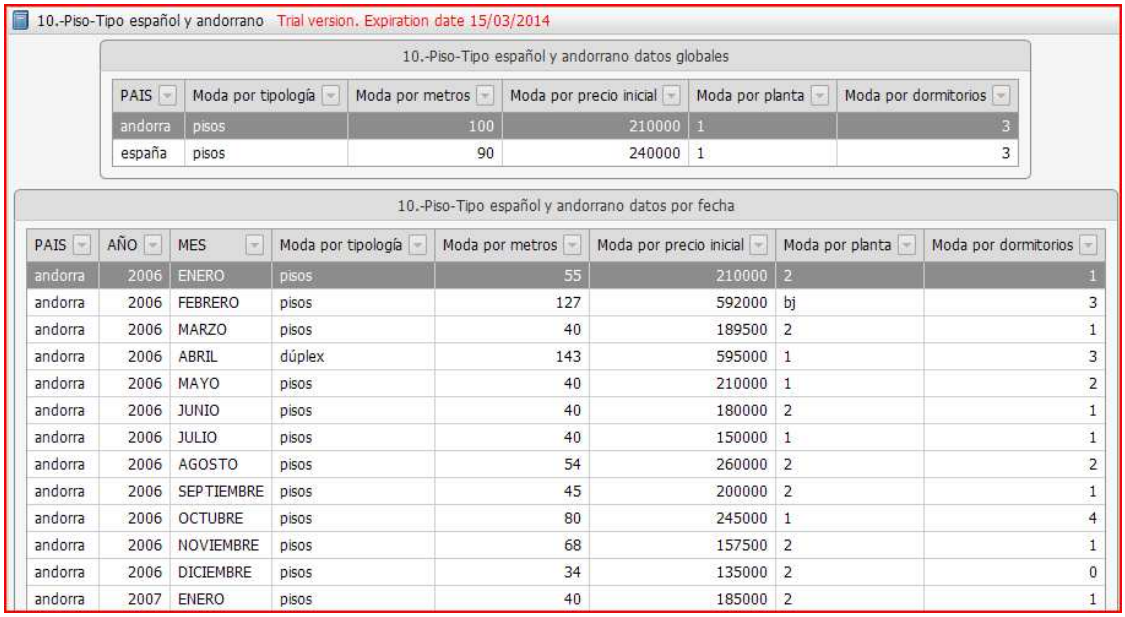


Figura 22b. Nº inmuebles vendidos y existentes por zona, tipología y características

6.10. Piso-Tipo español y andorrano



PAIS	Moda por tipología	Moda por metros	Moda por precio inicial	Moda por planta	Moda por dormitorios
andorra	pisos	100	210000	1	3
españa	pisos	90	240000	1	3

PAIS	AÑO	MES	Moda por tipología	Moda por metros	Moda por precio inicial	Moda por planta	Moda por dormitorios
andorra	2006	ENERO	pisos	55	210000	2	1
andorra	2006	FEBRERO	pisos	127	592000	bj	3
andorra	2006	MARZO	pisos	40	189500	2	1
andorra	2006	ABRIL	dúplex	143	595000	1	3
andorra	2006	MAYO	pisos	40	210000	1	2
andorra	2006	JUNIO	pisos	40	180000	2	1
andorra	2006	JULIO	pisos	40	150000	1	1
andorra	2006	AGOSTO	pisos	54	260000	2	2
andorra	2006	SEPTIEMBRE	pisos	45	200000	2	1
andorra	2006	OCTUBRE	pisos	80	245000	1	4
andorra	2006	NOVIEMBRE	pisos	68	157500	2	1
andorra	2006	DICIEMBRE	pisos	34	135000	2	0
andorra	2007	ENERO	pisos	40	185000	2	1

Figura 23. Piso-tipo español y andorrano

7. CONCLUSIONES

Tras finalizar el proyecto se llega a las siguientes conclusiones:

- Se ha realizado un análisis previo de los datos que nos han facilitado, este análisis nos sirve como punto de partida para el resto de fases del proyecto.
- Se realiza el diseño del almacén de datos, los procesos ETL y la implementación del almacén.
- Una vez creado el almacén se generan los informes necesarios para cubrir las necesidades del cliente.
- Mediante consultas SQL se determina cual es el piso-tipo español y andorrano.
- Podemos afirmar que se han alcanzado los objetivos del proyecto.

Los trabajos de documentación de la PEC1 y PEC2 supusieron una carga acorde con lo que se había previsto inicialmente, incluso a lo largo de estas dos etapas se consiguió adelantar algo de trabajo de la siguiente entrega (PEC3).

Sin embargo la carga de trabajo para completar la PEC3 fue superior a la esperada, lo que origino un retraso en la planificación inicial. Los principales motivos de este retraso han sido tener que conocer desde cero la herramienta Atlas SBI, herramienta de la que no se tenía ningún conocimiento anterior y que nuestro diseño inicial no era del todo correcto y que por lo tanto la carga de datos que se realizaba tampoco lo era, motivos por los que fue necesario modificar el diseño y los scripts de carga de datos.

Para solucionar este retraso y poder realizar la entrega de la PEC3 en fecha, se dedicaron más horas de las previstas en la planificación (horas que estaban programadas para dedicárselas a la asignatura que se cursa junto al TFC). Por último indicar que para elaborar la entrega final no se han producido retrasos significativos.

No obstante a pesar de estos problemas y de que mis conocimientos iniciales eran muy bajos, estoy contento con el trabajo realizado, aunque ahora con la experiencia que he ido obteniendo a lo largo del proyecto seguro que podría mejorarlo y hacerlo de una forma más eficiente.

En general el proyecto me ha servido para profundizar dentro del mundo del Data Warehouse, iniciarme con las herramientas Business Intelligence y para realizar un proyecto completo desde el inicio hasta la puesta en marcha del mismo.

8. LINEAS DE EVOLUCION FUTURAS

A pesar de que la carga inicial de la información facilitada por el cliente se realiza correctamente, la misma tarda muchas horas en realizarse, por lo que será necesario realizar las modificaciones necesarias en el mismo para mejorarlo y reducir el tiempo de la carga.

Otra evolución que ayudaría a mejorar la carga inicial, consistiría en que el proceso mostrase mensajes en pantalla para que el usuario este informado en todo momento de las acciones que se están realizando. Así el usuario final sabría lo que está haciendo el proceso y no tendría dudas sobre si este ha finalizado o si se encuentra bloqueado.

La información de las ofertas que nos facilitan abarca desde enero del 2006 hasta enero del 2008, pero para que la explotación del almacén de datos proporcione realmente información útil de cara a la toma de decisiones, es imprescindible actualizar la misma con las ofertas mensuales que se produzcan.

Esta actualización está contemplada en el proyecto mediante el proceso automatizado que se ha creado, aunque sería recomendable que se realizarán pruebas con más datos e incluso analizar algunos con el cliente. Esto nos permitiría localizar excepciones no contempladas en el planteamiento inicial y mejorar el proceso de actualización evitando errores en el mismo.

Respecto a los informes, los que se han generado en este proyecto cubren totalmente los requerimientos planteados por la empresa, pero con algo más de tiempo se pueden mejorar. Adaptar los informes a futuras necesidades y ofrecerles formación para que puedan crear sus propios informes sería otra opción.

Como en nuestro proyecto se ha creado un solo usuario con acceso a toda la información, puede ser necesario la creación de nuevos usuarios que solo puedan consultar la información de determinadas zonas o con otro tipo de restricciones.

Con el tiempo es posible que algunas zonas en las que están ubicados los inmuebles desaparezcan, cambien de nombre se unan con otras, etc., esto obligaría a un mantenimiento de datos no previsto en este proyecto.

Motivos por los que este trabajo se puede considerar como una primera solución que debe mejorarse mediante un proyecto de mayor alcance y duración.

9. GLOSARIO

Data Warehouse (DW): Almacén de datos.

Almacén de datos: Un almacén de datos es un conjunto de datos orientado a una temática concreta, integrado, variable en el tiempo y no volátil, que ayuda a la toma de decisiones.

OLAP (On-Line Analytical Processing): Procesamiento analítico en línea, solución utilizada en la llamada inteligencia empresarial, su objetivo es agilizar la consulta de grandes cantidades de datos, para lo que utiliza estructuras muldimensionales (Cubos OLAP).

Cubo OLAP: También llamado cubo multidimensional o hipercubo, está compuesto por hechos numéricos llamados medidas que se clasifican por dimensiones. El cubo de metadatos suele crearse a partir de un esquema en estrella o copo de nieve, esquema de las tablas en una base de datos relacional. Las medidas se obtienen de los registros de una tabla de hechos y las dimensiones se derivan de la dimensión de los cuadros.

Dimensión: Una estructura que representa una de las caras de un cubo. Cada dimensión representa una categoría diferente, como zona, tiempo, inmueble, etc., son las perspectivas de análisis de las variables (forman parte de las tablas de dimensiones).

Modelo dimensional: Un tipo de modelo de datos empleado en el Almacén de Datos. En un modelo dimensional existen dos tipos de tablas de dimensiones y tablas de hechos.

Tabla de dimensiones: Guarda los registros relativos a una dimensión concreta. No se almacenan hechos en las tablas de dimensiones.

Tabla de hechos: una de las tablas que constituyen el modelo dimensional. Una tabla de hechos típica contiene dos tipos de columnas, los hechos en sí y las claves relativas a las tablas de dimensiones

ETL: Denominación del proceso de extracción, transformación y carga (Extraction, Transformation, and Loading).

SQL*Loader: Utilidad que proporciona Oracle para cargar datos a una base de datos desde un fichero externo.

Oracle: Sistema de gestión de base de datos.

SQL (Structured query language): Lenguaje declarativo de acceso a bases de datos relacionales, permite diversos tipos de operaciones en estas.

PL/SQL (Procedural Language/Structured Query Language): Lenguaje de programación incrustado en Oracle, los programadores suelen construir bloques PL/SQL para utilizarlos como procedimientos o funciones.

Business Intelligence (BI): Inteligencia empresarial, conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos.

10. BIBLIOGRAFIA

10.1. Publicaciones

The Data Warehouse Toolkit. (2nd Edition). Ralph Kimball and Margy Ross.

Building the Data Warehouse (3rd Edition). W. H. Inmon.

The Data Warehouse ETL Toolkit. Joe Caserta and Ralph Kimball.

Data Modeling Techniques for Data Warehousing. Chuck Ballard y otros.

Otros proyectos de Almacenes de datos de la U.O.C.

10.2. Webs

http://www.inmoncif.com

http://www.rkimball.com

http://www.dwreview.com

http://www.1keydata.com/datawarehousing/datawarehouse.html

http://www.intranetjournal.com/features/datawarehousing.html

http://es.wikipedia.org/wiki/Esquema_en_estrella

http://www.redbooks.ibm.com

http://www.dwreview.com

http://www.oracle.com/technetwork/developer-tools/warehouse/documentation/index.html

http://www.oracle.com/technetwork/indexes/documentation/index.html

http://es.wikipedia.org/wiki/Almac%C3%A9n_de_datos

http://informationmanagement.wordpress.com/2007/11/19/diseno-de-un-data-warehouse-estrella-y-copo-de-nieve/

http://informationmanagement.wordpress.com/2007/12/26/diseno-de-un-data-warehouse-slowly-changing-dimensions/

http://informationmanagement.wordpress.com/2008/06/12/diseno-de-un-data-warehouse-medidas-i/

http://informationmanagement.wordpress.com/2008/06/13/diseno-de-un-data-warehouse-medidas-ii/

http://informationmanagement.wordpress.com/2008/07/12/diseno-de-un-data-warehouse-tabla-de-hecho/

http://atlassbikb.blogspot.com

http://es.wikipedia.org/wiki/Cubo_OLAP

http://es.wikipedia.org/wiki/OLAP#Tipos_de_sistemas_OLAP

http://es.wikipedia.org/wiki/Almac%C3%A9n_de_datos

http://es.wikipedia.org/wiki/SQL

http://es.wikipedia.org/wiki/PL/SQL

11. ANEXOS

11.1. Anexo - Referencias Scripts Carga inicial

A continuación se describe el contenido de los scripts, procedimientos y funciones, que se han generado para la creación de tablas, unificación, carga de datos y automatización del proceso. Los elementos que se enumeran en este apartado se han empaquetado en un fichero denominado carga_inicial.zip que se adjunta con esta entrega. Estos ficheros se encuentran ubicados en la siguiente ruta de la maquina virtual 'E:\oraclexe\TFC\TFC2011'.

- **1-Crea_Tabla_Catalogo.sql:** Script que primero borra y posteriormente crea la tabla temporal catalogo, tabla donde se cargarán los datos que nos facilitan.

```
/* Script creación tabla temporal catalogo, primero se elimina la tabla */
drop table catalogo;
/* Crea la tabla con los campos que aparecen en el fichero de texto */
create table catalogo (codigo varchar(180), zona varchar(20), tipo
varchar(10), planta varchar (4), dormitorios int, metros int,
precio_enero_2006 int, precio_febrero_2006 int, precio_marzo_2006
int, precio_abril_2006 int, precio_mayo_2006 int, precio_junio_2006 int,
precio_julio_2006 int, precio_agosto_2006 int, precio_septiembre_2006
int, precio_octubre_2006 int, precio_noviembre_2006 int,
precio_diciembre_2006 int, precio_enero_2007 int,
precio_febrero_2007 int, precio_marzo_2007 int, precio_abril_2007 int,
precio_mayo_2007 int, precio_junio_2007 int, precio_julio_2007 int,
precio_agosto_2007 int, precio_septiembre_2007 int,
precio_octubre_2007 int, precio_noviembre_2007 int,
precio_diciembre_2007 int, precio_enero_2008 int);
```

- **2-Crear_Tablas_TFC.sql:** Script que genera las tablas, índices y secuencias que se han definido en el diseño.

```
/* Se crean el resto de tablas, índices y secuencias */
DROP TABLE SYSTEM.ZONA CASCADE CONSTRAINTS;
CREATE TABLE  "ZONA"
("IDZONA" NUMBER(*,0) NOT NULL ENABLE,
"ZONA5" VARCHAR2(80),
"ZONA4" VARCHAR2(80),
"ZONA3" VARCHAR2(80),
"ZONA2" VARCHAR2(80),
"ZONA1" VARCHAR2(80),
"PAIS" VARCHAR2(20) NOT NULL ENABLE,
```

```
CONSTRAINT "PK_ZONA" PRIMARY KEY ("IDZONA") ENABLE) /
DROP TABLE SYSTEM.DORMITORIOS CASCADE CONSTRAINTS;
CREATE TABLE  "DORMITORIOS"
("IDDORMITORIO" NUMBER(*,0) NOT NULL ENABLE,
 CONSTRAINT "PK_DORMITORIOS" PRIMARY KEY
("IDDORMITORIO") ENABLE )
/
DROP TABLE SYSTEM.TIEMPO CASCADE CONSTRAINTS;
CREATE TABLE  "TIEMPO"
("IDTIEMPO" NUMBER(*,0) NOT NULL ENABLE,
"MES" VARCHAR2(10), "AÑO" NUMBER(*,0),
"NUMMES" NUMBER (*,0), CONSTRAINT "PK_TIEMPO" PRIMARY
KEY ("IDTIEMPO") ENABLE)
/
DROP TABLE SYSTEM.INMUEBLE CASCADE CONSTRAINTS;
CREATE TABLE  "INMUEBLE"
("IDINMUEBLE" NUMBER(*,0) NOT NULL ENABLE, "CODIGO"
VARCHAR2(15) NOT NULL ENABLE, "TIPO" VARCHAR2(10) NOT
NULL ENABLE, "PLANTA" VARCHAR2(20), "METROS" FLOAT(126),
"PRECIO_INICIAL" FLOAT(126), CONSTRAINT "PK_INMUEBLE"
PRIMARY KEY ("IDINMUEBLE") ENABLE)
/
DROP TABLE SYSTEM.OFERTA CASCADE CONSTRAINTS;
CREATE TABLE  "OFERTA"
("IDOFERTA" NUMBER(*,0) NOT NULL ENABLE, "IDTIEMPO"
NUMBER(*,0) NOT NULL ENABLE, "IDINMUEBLE" NUMBER(*,0) NOT
NULL ENABLE, "IDZONA" NUMBER(*,0) NOT NULL ENABLE,
"IDDORMITORIO" NUMBER(*,0) NOT NULL ENABLE, "VENDIDO"
VARCHAR2(1) DEFAULT 'N', "PRECIO" FLOAT(126),
"ANTIGUEDAD" NUMBER(*,0), "APRECIACION" FLOAT(126),
 CONSTRAINT "PK_OFERTA" PRIMARY KEY ("IDOFERTA") ENABLE,
 CONSTRAINT "FK_OFERTAINMUEBLE" FOREIGN KEY
("IDINMUEBLE") REFERENCES  "INMUEBLE" ("IDINMUEBLE")
ENABLE,
CONSTRAINT "FK_OFERTATIEMPO" FOREIGN KEY ("IDTIEMPO")
REFERENCES  "TIEMPO" ("IDTIEMPO") ENABLE,
CONSTRAINT "FK_OFERTADORMI" FOREIGN KEY
("IDDORMITORIO")
REFERENCES  "DORMITORIOS" ("IDDORMITORIO") ENABLE,
CONSTRAINT "FK_OFERTAZONA" FOREIGN KEY ("IDZONA")
REFERENCES  "ZONA" ("IDZONA") ENABLE)
/
CREATE UNIQUE INDEX "I_INMUEBLE" ON "INMUEBLE"
("CODIGO")
/
CREATE UNIQUE INDEX  "I_TIEMPO" ON  "TIEMPO" ("MES", "AÑO")
/
```

```
CREATE UNIQUE INDEX "I_ZONA" ON "ZONA" ("ZONA1", "ZONA2",
"ZONA3", "ZONA4", "ZONA5", "PAIS")
/
CREATE UNIQUE INDEX "I_OFERTA" ON "OFERTA" "IDINMUEBLE",
"IDTIEMPO")
/
DROP SEQUENCE SYSTEM.SQ_IDZONA;
CREATE SEQUENCE SYSTEM.SQ_IDZONA MINVALUE 1
MAXVALUE 999999999999999999999999999 INCREMENT BY 1
START WITH 1 NOCACHE  NOORDER  NOCYCLE
/
DROP SEQUENCE SYSTEM.SQ_IDDORMITORIO;
CREATE SEQUENCE SYSTEM.SQ_IDDORMITORIO MINVALUE 1
MAXVALUE 999999999999999999999999999 INCREMENT BY 1
START WITH 1 NOCACHE  NOORDER  NOCYCLE
/
DROP SEQUENCE SYSTEM.SQ_IDTIEMPO;
CREATE SEQUENCE SYSTEM.SQ_IDTIEMPO MINVALUE 1
MAXVALUE 999999999999999999999999999 INCREMENT BY 1
START WITH 1 NOCACHE  NOORDER  NOCYCLE
/
DROP SEQUENCE SYSTEM.SQ_IDINMUEBLE;
CREATE SEQUENCE SYSTEM.SQ_IDINMUEBLE MINVALUE 1
MAXVALUE 999999999999999999999999999 INCREMENT BY 1
START WITH 1 NOCACHE  NOORDER  NOCYCLE
/
DROP SEQUENCE SYSTEM.SQ_IDOFERTA;
CREATE SEQUENCE SYSTEM.SQ_IDOFERTA MINVALUE 1
MAXVALUE 999999999999999999999999999 INCREMENT BY 1
START WITH 1 NOCACHE  NOORDER  NOCYCLE
/
```

- **3-Unificar_Datos.sql:** Script que unifica los datos que nos facilitan antes de realizar el proceso de carga definitivo.

```
/* Actualización campo planta de '--' a null */
UPDATE SYSTEM.CATALOGO SET CATALOGO.planta = Null
WHERE (((CATALOGO.planta)='--'));
/* Campo planta de 'bj' a null para inmuebles tipo 'Áticos' */
UPDATE SYSTEM.CATALOGO SET CATALOGO.planta = Null
WHERE (((CATALOGO.planta)='bj') AND ((CATALOGO.tipo)='áticos'));
/* Actualización campo metros < 10 Y > 2000 a null */
UPDATE SYSTEM.CATALOGO SET CATALOGO.metros = Null
WHERE (((CATALOGO.metros)<10 Or (CATALOGO.metros)>2000));
```

```
/* Actualización campo dormitorios > 15 A '0' */
UPDATE SYSTEM.CATALOGO SET CATALOGO.dormitorios = 0
WHERE (((CATALOGO.dormitorios)>15));
/* Actualización campo dormitorios a ‘0’. Todos los registros con
dormitorio igual a null los ponemos a ‘0’ */
UPDATE SYSTEM.CATALOGO SET CATALOGO.dormitorios = 0
WHERE (CATALOGO.dormitorios is null and zona is not null);
/* Actualización campo planta a null para inmuebles tipo 'Chalets'.
Estimamos que un chalet no está ubicado en ninguna planta y por lo
tanto el campo planta para chalets se pone a nulo */
UPDATE SYSTEM.CATALOGO SET CATALOGO.planta = Null
WHERE (((CATALOGO.tipo)='chalets'));
```

- **4-Unificar_Precios.sql:** Script que unifica los precios de las ofertas que nos facilitan.

```
/* Campos precio a null para los que tengan el valor ‘0’ */
UPDATE CATALOGO SET CATALOGO.precio_enero_2006 = Null
WHERE CATALOGO.precio_enero_2006=0;
UPDATE CATALOGO SET CATALOGO.precio_febrero_2006 = Null
WHERE CATALOGO.precio_febrero_2006=0;
UPDATE CATALOGO SET CATALOGO.precio_marzo_2006 = Null
WHERE CATALOGO.precio_marzo_2006=0;
……..
……..
……..
UPDATE CATALOGO SET CATALOGO.precio_noviembre_2007 = Null
WHERE CATALOGO.precio_noviembre_2007=0;
UPDATE CATALOGO SET CATALOGO.precio_diciembre_2007 = Null
WHERE CATALOGO.precio_diciembre_2007=0;
UPDATE CATALOGO SET CATALOGO.precio_enero_2008 = Null
WHERE CATALOGO.precio_enero_2008=0;
```

- **5-Rellenar_Dormitorios.sql:** Script que se encarga de rellenar el con los datos que nos facilitan el contenido de la tabla dormitorios.

```
/* Rellena la tabla dormitorios */
INSERT INTO DORMITORIOS (IdDormitorio) SELECT
CATALOGO.DORMITORIOS FROM CATALOGO
WHERE (((CATALOGO.DORMITORIOS) Is Not Null)) GROUP BY
CATALOGO.DORMITORIOS ORDER BY
CATALOGO.DORMITORIOS;
```

- **6-Carga_Tiempo.sql:** Procedimiento que se encarga de rellenar el contenido de la tabla Tiempo con los años y meses de las ofertas que nos facilitan.

```
/* Procedimiento para cargar la tabla tiempo */
CREATE OR REPLACE PROCEDURE Carga_Tiempo
IS
/* Se define el cursor para recorrer las columnas de la tabla
CATALOGO */
CURSOR c2 IS SELECT * FROM USER_TAB_COLUMNS WHERE
TABLE_NAME = 'CATALOGO';
v_columns_names c2%ROWTYPE;
col INTEGER := 1;
existe INTEGER;
nummes INTEGER;
str_query VARCHAR2(200);
v_mes VARCHAR2(200);
v_año VARCHAR2(200);
BEGIN
OPEN c2;
/* Bucle mientras tengamos columna de la tabla CATALOGO que leer */
LOOP
FETCH c2 into v_columns_names;
Exit when c2%NOTFOUND;
/* Si columna >= que 7 ó < que 0 columna 7 primera columna con
precio */
IF(col >= 7) THEN
/* Hacemos tratamiento de columnas para obtener el nombre de la
misma */
v_mes := TRIM(SUBSTR(v_columns_names.column_name, 8,
LENGTH(v_columns_names.column_name)-12));
v_año := TRIM(SUBSTR(v_columns_names.column_name,
LENGTH(v_columns_names.column_name)-3));
DBMS_OUTPUT.PUT_LINE ('MESES: '||v_mes||' - '||v_año);
 /* Comprobamos el numero de mes que le corresponde al literal mes */
        IF (v_mes='ENERO') THEN
        nummes:=1;
        ELSIF (v_mes='FEBRERO') THEN
        nummes:=2;
        ELSIF (v_mes='MARZO') THEN
        nummes:=3;
        ELSIF (v_mes='ABRIL') THEN
        nummes:=4;
        ELSIF (v_mes='MAYO') THEN
        nummes:=5;
```

```
            ELSIF (v_mes='JUNIO') THEN
            nummes:=6;
            ELSIF (v_mes='JULIO') THEN
            nummes:=7;
            ELSIF (v_mes='AGOSTO') THEN
            nummes:=8;
            ELSIF (v_mes='SEPTIEMBRE') THEN
            nummes:=9;
            ELSIF (v_mes='OCTUBRE') THEN
            nummes:=10;
            ELSIF (v_mes='NOVIEMBRE') THEN
            nummes:=11;
            ELSIF (v_mes='DICIEMBRE') THEN
            nummes:=12;
            END IF;
            /* Consultamos si ya existe el mes y el año */
           str_query:='SELECT COUNT (*) FROM TIEMPO where mes=
          :mes and año = :año';
           EXECUTE IMMEDIATE str_query INTO existe USING v_mes,
           v_año;
/* Si no existe se crea el nuevo registro en tabla TIEMPO, su existe se
omite */
           IF (existe = 0) THEN
                 str_query:='INSERT INTO TIEMPO (IDTIEMPO,
                 NUMMES, MES, AÑO) VALUES
                 (SYSTEM.SQ_IDTIEMPO.nextval, :nummes, :mes, :año)';
                 EXECUTE IMMEDIATE str_query USING nummes,
                 v_mes, v_año;
           END IF;
         END IF;
/* Se aumenta la variable col en 1, para pasar a leer la siguiente
columna */
col := col+1;
/* Fin del bucle mientras tengamos columnas de la tabla CATALOGO */
END LOOP;
CLOSE c2;
COMMIT;
END;
/
```

- **7-Mytable.sql:** Script que crea o reemplaza la tabla myTableType.

```
/* Crea la tabla */
create or replace type myTableType as table of varchar2(255);
/
```

- **8-Str2tbl.sql:** Función a la que se llama desde el procedimiento skeleton.sql para obtener las distintas zonas existentes.

```
/* Crea la función */
create or replace function str2tbl
        (p_str   in varchar2, p_delim in varchar2 default '.')
        return     myTableType
 as     l_str      long default p_str || p_delim;
        l_n        number;
        l_data     myTableType := myTabletype();
 begin
        loop
           l_n := instr( l_str, p_delim );
           exit when (nvl(l_n,0) = 0);
           l_data.extend;
           l_data( l_data.count ) := ltrim(rtrim(substr(l_str,1,l_n-1)));
           l_str := substr( l_str, l_n+length(p_delim) );
        end loop;
        return l_data;
end;
/
```

- **9-Skeleton.sql:** Procedimiento que realiza la carga de los datos que nos facilitan desde la tabla temporal 'catalogo' al resto de tablas, al mismo tiempo realiza los cálculos correspondientes a la antigüedad y a la apreciación o depreciación del inmueble respecto a su precio inicial. En este procedimiento se controlan los posibles inmuebles, zonas y ofertas duplicadas.

```
/* Se crea el proceso skeleton encargado de la carga de datos */
create or replace PROCEDURE skeleton
IS
  /* Se define el cursor con el que se recorrerá la tabla CATALOGO */
  CURSOR c1 IS SELECT * FROM CATALOGO;
  v_CATALOGO_rec c1%ROWTYPE;
  /* El cursor con el que se recorrerá las columnas de la tabla
CATALOGO */
 CURSOR c2 IS SELECT * FROM USER_TAB_COLUMNS WHERE
TABLE_NAME = 'CATALOGO';
  v_columns_names c2%ROWTYPE;
  /* Se definen las variables necesarias para el procedimiento */
  cou INTEGER := 1;
  var_zonas VARCHAR(500);
  var_array mytabletype;
```

```
  /* Variables para almacenar las zonas */
  var_pais VARCHAR(30);
  v_zona1 VARCHAR(100);
  v_zona2 VARCHAR(100);
  v_zona3 VARCHAR(100);
  v_zona4 VARCHAR(100);
  v_zona5 VARCHAR(100);
  /* Variables para almacenar los precios de las ofertas, para controlar si
existen las zonas, inmuebles, si está vendido, etc. */
  str_query VARCHAR2(200);
  precio NUMBER;
  venta VARCHAR(1);
  precio_inicial FLOAT(126);
  col INTEGER;
  existe INTEGER;
  existe2 INTEGER;
  existe_vendido INTEGER;
  antiguedad INTEGER;
  anti INTEGER;
  /* Variables para almacenar las identificadores de la tablas */
  indice_Inmueble INTEGER;
  indice_Tiempo INTEGER;
  indice_Zona INTEGER;
  indice_Oferta INTEGER;
  /* Variables para almacenar el mes y año de las ofertas */
  v_mes VARCHAR2(200);
  v_año VARCHAR2(200);
  v_din_columns INTEGER;
BEGIN
  /* Se abre el cursor c1 */
  OPEN c1;
  DBMS_OUTPUT.PUT_LINE('Cursor opened!');
  /* Bucle mientras el cursor c1 tenga datos */
  Loop
    FETCH c1 into v_CATALOGO_rec;
    Exit when c1%NOTFOUND;
/* Si el campo código comienza por #, sabemos que nos indica una
zona y se llama a la función str2tbl para montar el registro de la tabla
zona correspondiente  */
    IF(SUBSTR(v_CATALOGO_rec.codigo,1,1) = '#' ) THEN
      var_zonas := SUBSTR(v_CATALOGO_rec.codigo,2);
      var_array := str2tbl(var_zonas, ':::');
      DBMS_OUTPUT.PUT_LINE('TAM: '||var_array.COUNT);
```

```
/* Si el campo código comienza por andorra el país será Andorra en
caso contrario será España */
      IF(var_array.COUNT > 1 and TRIM(var_array(2))='andorra') THEN
        var_pais := 'andorra';
      ELSE
        var_pais := 'españa';
      END IF;
/* Se ponen en blanco las variables donde almacenaremos las zonas */
      v_zona1 := '-';
      v_zona2 := '-';
      v_zona3 := '-';
      v_zona4 := '-';
      v_zona5 := '-';
  /* Se almacenan las zonas en las variables creadas para ello */
      FOR i IN 1 .. var_array.COUNT LOOP
        CASE i
WHEN 2 THEN v_zona1:= var_array(i);
WHEN 3 THEN v_zona2:= var_array(i);
WHEN 4 THEN v_zona3:= var_array(i);
WHEN 5 THEN v_zona4:= var_array(i);
WHEN 6 THEN v_zona5:= var_array(i);
ELSE DBMS_OUTPUT.PUT_LINE ('');
        END CASE;
      END LOOP;
          /* Consultamos si ya existe la zona */
          str_query:='SELECT COUNT (*) FROM ZONA where pais = :pa
          and zona1 = :z1 and zona2 = :z2 and zona3 = :z3 and zona4 =
          :z4 and zona5 = :z5';
          EXECUTE IMMEDIATE str_query INTO existe USING var_pais,
          v_zona1, v_zona2, v_zona3, v_zona4, v_zona5;
    /* Si no existe se genera la nueva zona en la tabla zona y se
    almacena su identificador, en caso contrario se almacena el
    identificador de la zona existente */
          IF (existe = 0) THEN
                INSERT INTO ZONA (IDZONA, ZONA1, ZONA2, ZONA3,
                ZONA4, ZONA5, PAIS) VALUES
                (SYSTEM.SQ_IDZONA.nextval, v_zona1, v_zona2,
                v_zona3, v_zona4, v_zona5, var_pais);
                str_query := 'SELECT SYSTEM.SQ_IDZONA.currval from
                dual';
                execute immediate str_query into indice_Zona;
          ELSE
                str_query:='SELECT IDZONA from ZONA where pais =
                :pa and zona1 = :z1 and zona2 = :z2 and zona3 = :z3 and
                zona4 = :z4 and zona5 = :z5';
```

```
            EXECUTE IMMEDIATE str_query INTO indice_zona
            USING var_pais, v_zona1, v_zona2, v_zona3, v_zona4,
            v_zona5;
END IF;
/* Si el campo no comienza por # se trata de un inmueble */
ELSE
/* Se abre el cursor c2 */
OPEN c2;
/* Se inicializa a 'S' la variable venta y antiguedad a 0 */
venta := 'S';
antiguedad := 0;
/* Consultamos si ya existe el inmueble */
str_query:='SELECT COUNT (*) FROM INMUEBLE where
codigo = :co';
EXECUTE IMMEDIATE str_query INTO existe2 USING
v_CATALOGO_rec.codigo;
/* Si no existe se genera el nuevo inmueble en la tabla inmueble
y se almacena su identificador, en caso contrario se almacena el
identificador del inmueble, su antiguedad y el identificador de su
última oferta */
IF (existe2 = 0) THEN
            DBMS_OUTPUT.PUT_LINE ('Datos nuevos1:
            '||v_CATALOGO_rec.codigo ||' - '||
            v_CATALOGO_rec.tipo||' - '|| v_CATALOGO_rec.planta||' -
            '|| v_CATALOGO_rec.metros);
            INSERT INTO INMUEBLE (IDINMUEBLE, CODIGO,
            TIPO, PLANTA, METROS) VALUES
            (SYSTEM.SQ_IDINMUEBLE.nextval,
            v_CATALOGO_rec.codigo, v_CATALOGO_rec.tipo,
            v_CATALOGO_rec.planta, v_CATALOGO_rec.metros);
            str_query := 'SELECT SYSTEM.SQ_IDINMUEBLE.currval
            from dual';
            execute immediate str_query into indice_inmueble;
ELSE
            str_query:='SELECT IDINMUEBLE from INMUEBLE
            where codigo = :co';
            EXECUTE IMMEDIATE str_query INTO indice_inmueble
            USING v_CATALOGO_rec.codigo;
            str_query:='SELECT MAX(IDOFERTA) FROM OFERTA
            where idinmueble = :id';
            EXECUTE IMMEDIATE str_query INTO indice_oferta
            USING indice_inmueble;
            str_query:='SELECT antiguedad FROM OFERTA where
            idoferta = :idof';
            EXECUTE IMMEDIATE str_query INTO anti USING
            indice_oferta;
```

```
                antiguedad := antiguedad + anti;
        END IF;
/* Si el inmueble que ya está dado de alta se encuentra vendido */
str_query:='SELECT COUNT (*) FROM OFERTA where idinmueble =
:idin and vendido = :ven';
EXECUTE IMMEDIATE str_query INTO existe_vendido USING
indice_inmueble, venta;
/* Si no está vendido se leen las diferentes ofertas que existan para el
inmueble, en caso contrario no se hace nada con el inmueble. */
/* Se entiende que si un inmueble se ha vendido y vuelve a entrar en la
inmobiliaria para gestionar su venta este debe registrarse con un nuevo
código, motivo por el que no haremos nada con un inmueble ya
registrado y vendido */
        IF (existe_vendido = 0) THEN
/* Se inicializa la variable col a 1 */
   col := 1;
/* Bucle mientras tengamos columna de la tabla CATALOGO que leer */
LOOP
FETCH c2 into v_columns_names;
Exit when c2%NOTFOUND;
         /* Si la columna es mayor o igual que 7 o menor que 0 entonces,
         la columna 7 es la primera columna con precio */
       IF(col >= 7 or col < 0) THEN
          /* Almacenamos el precio de la oferta en la variable precio */
        str_query:='SELECT '||v_columns_names.column_name||' FROM
        CATALOGO where codigo = :id';
        EXECUTE IMMEDIATE str_query INTO precio USING
        v_CATALOGO_rec.codigo;
 /* Almacenamos el precio inicial del inmueble en la variable
precio_inicial */
        str_query:='SELECT PRECIO_INICIAL FROM INMUEBLE where
        idinmueble = :id';
        EXECUTE IMMEDIATE str_query INTO precio_inicial USING
        indice_inmueble;
 /* Se hace el tratamiento de columnas para obtener el nombre de la
misma */
        v_mes := TRIM(SUBSTR(v_columns_names.column_name, 8,
        LENGTH(v_columns_names.column_name)-12));
        v_año := TRIM(SUBSTR(v_columns_names.column_name,
        LENGTH(v_columns_names.column_name)-3));
        DBMS_OUTPUT.PUT_LINE ('MESES: '||v_mes||' - '||v_año);
         /* Almacenamos el identificador de tiempo que corresponde al
        mes y año en la variable indice_tiempo */
        str_query := 'SELECT IDTIEMPO from TIEMPO WHERE MES =
        :M AND AÑO = :A';
        execute immediate str_query into indice_Tiempo USING v_mes,
        v_año;
```

```
        DBMS_OUTPUT.PUT_LINE ('DATE: '||v_mes||' - '||v_año||' -
        '||indice_Tiempo);
/* Si el precio de la oferta es null y el precio inicial es null, significa que
el inmueble no tiene registrada ninguna oferta, no se hace nada */
        IF (precio is NULL and precio_inicial is NULL) THEN
          DBMS_OUTPUT.PUT_LINE ('');
/* Si el precio de la oferta es distinto de null y el precio inicial es null,
significa que es la primera oferta que existe para el inmueble, se
aumenta la antiguedad en 1, se actualiza el precio inicial del inmueble
con el precio de la oferta, vendido a 'N' y apreciación a 0 */
/* Por último se registra la oferta correspondiente con los identificadores
del resto de tablas y el resto de campos */
        ELSIF (precio is not NULL and precio_inicial is NULL) THEN
          antiguedad := antiguedad +1;
          UPDATE INMUEBLE SET precio_inicial = precio WHERE
        idinmueble = indice_inmueble;
        INSERT INTO OFERTA (IDOFERTA, IDTIEMPO, IDINMUEBLE,
        IDZONA, IDDORMITORIO, VENDIDO, PRECIO, ANTIGUEDAD,
        APRECIACION)
        VALUES (SYSTEM.SQ_IDOFERTA.nextval, indice_Tiempo,
        indice_inmueble, indice_Zona, v_CATALOGO_rec.dormitorios,
        'N', precio, antiguedad, 0);
 /* Si el precio de la oferta es distinto de null y el precio inicial es
también distinto de null, significa que es una nueva oferta para el
inmueble */
/* Se aumenta la antiguedad en 1, el campo vendido se pone a 'N' y
apreciación será igual al precio de la oferta menos el precio inicial */
/* Por último se registra la oferta correspondiente con los identificadores
del resto de tablas y el resto de campos */
        ELSIF(precio is not NULL and precio_inicial is not NULL) THEN
                antiguedad := antiguedad +1;
                INSERT INTO OFERTA (IDOFERTA, IDTIEMPO,
                IDINMUEBLE, IDZONA, IDDORMITORIO, VENDIDO,
                PRECIO, ANTIGUEDAD, APRECIACION)
                VALUES (SYSTEM.SQ_IDOFERTA.nextval,
               indice_Tiempo, indice_inmueble, indice_Zona,
               v_CATALOGO_rec.dormitorios, 'N', precio, antiguedad,
               precio-precio_inicial);
/* Si el precio de la oferta es null ó 0 y el precio inicial es distinto de null,
significa que se ha producido la venta del inmueble*/
/* Se actualiza el campo vendido a de la última oferta del inmueble a 'S'
para indicar que está vendido, se asigna el valor -2 a la variable col
para poder salir de bucle de lectura de ofertas, al venderse el inmueble
no es necesario seguir leyendo las posibles ofertas del mismo */
        ELSIF( (precio is NULL or precio = 0) and precio_inicial is not
        NULL) THEN
```

```
str_query:='SELECT MAX(IDOFERTA) FROM OFERTA where
idinmueble = :id';
EXECUTE IMMEDIATE str_query INTO indice_oferta USING
indice_inmueble;
UPDATE OFERTA SET vendido = 'S' WHERE idoferta =
indice_oferta;
col := -2;
END IF;
DBMS_OUTPUT.PUT_LINE ('Cols:
'||v_columns_names.column_name||' - '||precio||' -
'||precio_inicial);
/* Fin del if mientras la columna es >= a 7 o menor que 0 */
END IF;
/* Se aumenta la variable col en 1, para leer la siguiente oferta */
col := col+1;
/* Fin del bucle mientras tengamos columnas de la tabla
CATALOGO */
END LOOP;
/* Fin del if que comprueba si el inmueble está vendido */
END IF;
/* Se cierra el cursor c2 */
CLOSE c2;
/* Fin del if que comprueba si el primer valor de código es # */
END IF;
DBMS_OUTPUT.PUT_LINE('Code: ' ||
v_CATALOGO_rec.codigo || ' - Zona: '||
v_CATALOGO_rec.zona);
/* Bucle contador para realizar el commit cada 1000 pasos */
IF(MOD(COU,1000)=0) THEN
        COMMIT;
END IF;
/* Se aumenta el contador de pasos en 1 */
cou:= cou+1;
/* Fin del bucle mientras el cursor c1 tenga datos */
End Loop;
/* Se cierra el cursor c1 */
CLOSE c1;
/* Commit final */
COMMIT;
END;
/
```

- **Lanza1.sql:** Script que realiza la conexión con Oracle y llama al script 1-Crea_Tabla_Catalogo.sql.

```
/* Lanza el script de creación de la tabla temporal catalogo */
conn system/uoc
@e:\oraclexe\TFC\TFC2011\1-Crea_Tabla_Catalogo.sql
```

- **Lanza2.sql:** Script que realiza la conexión con Oracle y llama a los siguientes scripts:

```
/* Lanza el resto de scripts necesarios para la carga inicial */
conn system/uoc
@e:\oraclexe\TFC\TFC2011\2-Crear_Tablas_TFC.sql
@e:\oraclexe\TFC\TFC2011\3-Unificar_Datos.sql
@e:\oraclexe\TFC\TFC2011\4-Unificar_Precios.sql
@e:\oraclexe\TFC\TFC2011\5-Rellenar_Dormitorios.sql
@e:\oraclexe\TFC\TFC2011\6-Carga_Tiempo.sql
execute Carga_Tiempo
@e:\oraclexe\TFC\TFC2011\7-Mytable.sql
@e:\oraclexe\TFC\TFC2011\8-STR2TBL.sql
@e:\oraclexe\TFC\TFC2011\9-SKELETON.sql
execute skeleton
```

- **Sqlldr.bat:** Este proceso esta ubicado en la ruta ‘C:\Documents and Settings\uocdw’ de la maquina virtual, se encarga de realizar la carga de datos que nos facilitan en la tabla temporal ‘catalogo’. Para que funcione correctamente debe existir el fichero ‘cargador.ctl’ y el fichero de texto ‘Catalogo.txt’ con los datos que nos facilitan en la ruta E:\oraclexe de la maquina virtual.

```
/* Lanza SQL*Loader */
cd e:\oraclexe
e:
sqlldr userid=SYSTEM/uoc Control=cargador.ctl data=catalogo.txt
```

- **Cargador.ctl:** Archivo de control que se utiliza para que funcione correctamente SQL*Loader. En este fichero se especifica la ruta del fichero *.txt que contiene los datos que nos facilitan, la tabla destino en la que se cargarán los mismos, los campos que se cargarán y el carácter que hace de separador de estos campos.

```
/* Fichero control de SQL*Loader */
load data
infile 'c:\oraclexe\catalogo.txt'
into table catalogo
insert
fields terminated by x'09'
trailing nullcols
(codigo, zona, tipo, planta, dormitorios, metros, precio_enero_2006,
precio_febrero_2006, precio_marzo_2006, precio_abril_2006,
precio_mayo_2006, precio_junio_2006, precio_julio_2006,
precio_agosto_2006, precio_septiembre_2006, precio_octubre_2006,
precio_noviembre_2006, precio_diciembre_2006, precio_enero_2007,
precio_febrero_2007, precio_marzo_2007, precio_abril_2007,
precio_mayo_2007, precio_junio_2007, precio_julio_2007,
precio_agosto_2007, precio_septiembre_2007, precio_octubre_2007,
precio_noviembre_2007, precio_diciembre_2007, precio_enero_2008)
```

- **Catalogo.txt.** Fichero de texto que nos facilitan con los datos de la inmobiliaria. No se considera necesario reproducir el contenido de este fichero.

11.2. Anexo - Referencias Scripts actualizaciones mensuales

A continuación se describe el contenido de los scripts, procedimientos y funciones, que se han generado para la creación de tablas, unificación, carga de datos y automatización del proceso de las actualizaciones mensuales. Los elementos que se enumeran en este apartado se han empaquetado en un fichero denominado actualización_mensual.zip que se adjunta con esta entrega.

- **1-Crea_Tabla_Catalogo_act.sql:** Script que primero borra y posteriormente crea la tabla temporal catalogo_act, tabla donde se cargarán los datos de actualización que nos facilitan.

```
/* Se crea la tabla temporal de actualización */
drop table catalogo_act;
create table catalogo_act ( codigo varchar(180), zona varchar(20), tipo
varchar(10), planta varchar (4), dormitorios int, metros int, precio int);
```

- **2-Unificar_Datos_act.sql:** Script que unifica los datos mensuales que nos facilitan antes de realizar el proceso de carga definitivo.

```
/* Actualización campo planta de '--' a null */
UPDATE SYSTEM.CATALOGO_ACT
SET CATALOGO_ACT.planta = Null
WHERE (((CATALOGO_ACT.planta)='--'));
/* Campo planta de 'bj' a null para inmuebles tipo 'Áticos' */
UPDATE SYSTEM.CATALOGO_ACT
SET CATALOGO_ACT.planta = Null
WHERE (((CATALOGO_ACT.planta)='bj') AND
((CATALOGO_ACT.tipo)='áticos'));
/* Actualización campo metros < 10 Y > 2000 a null */
UPDATE SYSTEM.CATALOGO_ACT
SET CATALOGO_ACT.metros = Null
WHERE (((CATALOGO_ACT.metros)<10 Or
(CATALOGO_ACT.metros)>2000));
/* Actualización campo dormitorios > 15 A '0' */
UPDATE SYSTEM.CATALOGO_ACT
SET CATALOGO_ACT.dormitorios = 0
WHERE (((CATALOGO_ACT.dormitorios)>15));
/* Actualización campo dormitorios a ‘0’. Todos los registros con
dormitorio igual a null los ponemos a ‘0’ */
UPDATE SYSTEM.CATALOGO_ACT
SET CATALOGO_ACT.dormitorios = 0
WHERE (CATALOGO_ACT.dormitorios is null and zona is not null);
```

```
/* Actualización campo planta a null para inmuebles tipo 'Chalets'.
Estimamos que un chalet no está ubicado en ninguna planta y por lo
tanto el campo planta para chalets se pone a nulo */
UPDATE SYSTEM.CATALOGO_ACT
SET CATALOGO_ACT.planta = Null
WHERE (((CATALOGO_ACT.tipo)='chalets'));
```

- **3-Unificar_Precios_act.sql:** Script que unifica los precios de las ofertas mensuales que nos facilitan.

```
/* Actualización campo precio a null para los que tengan valor '0' */
UPDATE CATALOGO_ACT SET CATALOGO_ACT.precio = Null
WHERE CATALOGO_ACT.precio=0;
```

- **4-Mytable.sql:** Script que crea o reemplaza la tabla myTableType.

```
/* Crea la tabla */
create or replace type myTableType as table of varchar2(255);
/
```

- **5-Skeleton_act.sql:** Procedimiento que realiza la carga de los datos mensual que nos facilitan desde la tabla temporal 'catalogo_act' al resto de tablas, al mismo tiempo realiza los cálculos correspondientes a la antigüedad y a la apreciación o depreciación del inmueble respecto a su precio inicial. En este procedimiento se controlan los posibles inmuebles, zonas y ofertas duplicadas.

```
/* Se crea el procedimiento que realiza la carga de actualización */
create or replace PROCEDURE skeleton_act
IS
/* Se define el cursor con el que se recorre la tabla CATALOGO_ACT */
CURSOR c1 IS SELECT * FROM CATALOGO_ACT;
v_catalogo_rec c1%ROWTYPE;
/* Define el cursor que se recorrerá las columnas de la tabla
catalogo_act */
CURSOR c2 IS SELECT * FROM USER_TAB_COLUMNS WHERE
TABLE_NAME = 'CATALOGO_ACT';
v_columns_names c2%ROWTYPE;
/* Se definen las variables necesarias para el procedimiento */
cou INTEGER := 1;
var_zonas VARCHAR(500);
var_array mytabletype;
```

```
/* Variables para almacenar las zonas */
var_pais VARCHAR(30);
v_zona1 VARCHAR(100);
v_zona2 VARCHAR(100);
v_zona3 VARCHAR(100);
v_zona4 VARCHAR(100);
v_zona5 VARCHAR(100);
/* Variables para almacenar los precios de las ofertas, para controlar si
existen las zonas, inmuebles, si está vendido, etc. */
str_query VARCHAR2(200);
precio NUMBER;
venta VARCHAR(1);
precio_inicial FLOAT(126);
col INTEGER;
existe INTEGER;
existe2 INTEGER;
existe_vendido INTEGER;
antiguedad INTEGER;
anti INTEGER;
/* Variables para controlar que el nuevo mes/año de la oferta solo se
registre una vez en la tabla tiempo y para su tratamiento */
primerpaso INTEGER :=0;
ult_itiempo INTEGER;
ult_nummes INTEGER;
ult_año INTEGER;
sig_mes VARCHAR(20);
sig_nummes INTEGER;
sig_año INTEGER;
/* Variables para almacenar los identificadores de la tablas */
indice_Inmueble INTEGER;
indice_Tiempo INTEGER;
indice_Zona INTEGER;
indice_Oferta INTEGER;
/* Variables para almacenar el mes y año de las ofertas */
v_mes VARCHAR2(200);
v_año VARCHAR2(200);
v_din_columns INTEGER;
BEGIN
/* Se abre el cursor c1 */
OPEN c1;
DBMS_OUTPUT.PUT_LINE('Cursor opened!');
/* Bucle mientras el cursor c1 tenga datos */
Loop
   FETCH c1 into v_catalogo_rec;
   Exit when c1%NOTFOUND;
```

```
/* Si el campo código comienza por #, sabemos que nos indica una
zona y se llama a la función str2tbl para montar el registro de la tabla
zona */
   IF(SUBSTR(v_catalogo_rec.codigo,1,1) = '#' ) THEN
     var_zonas := SUBSTR(v_catalogo_rec.codigo,2);
     var_array := str2tbl(var_zonas, ':::');
     DBMS_OUTPUT.PUT_LINE('TAM: '||var_array.COUNT);
          /* Si el campo código comienza por andorra el país será
          Andorra en caso contrario será España */
     IF(var_array.COUNT > 1 and TRIM(var_array(2))='andorra') THEN
       var_pais := 'andorra';
     ELSE
       var_pais := 'españa';
     END IF;
        /* Se ponen en blanco las variables donde almacenaremos las
        zonas */
     v_zona1 := '-';
     v_zona2 := '-';
     v_zona3 := '-';
     v_zona4 := '-';
     v_zona5 := '-';
          /* Se almacenan las zonas en las variables creadas para ello */
     FOR i IN 1 .. var_array.COUNT LOOP
       CASE i
         WHEN 2 THEN v_zona1:= var_array(i);
         WHEN 3 THEN v_zona2:= var_array(i);
          WHEN 4 THEN v_zona3:= var_array(i);
         WHEN 5 THEN v_zona4:= var_array(i);
         WHEN 6 THEN v_zona5:= var_array(i);
         ELSE DBMS_OUTPUT.PUT_LINE ('');
       END CASE;
     END LOOP;
        /* Consultamos si ya existe la zona */
        str_query:='SELECT COUNT (*) FROM ZONA where pais = :pa
        and zona1 = :z1 and zona2 = :z2 and zona3 = :z3 and zona4 =
        :z4 and zona5 = :z5';
        EXECUTE IMMEDIATE str_query INTO existe USING var_pais,
        v_zona1, v_zona2, v_zona3, v_zona4, v_zona5;
/* Si no existe se genera la nueva zona en la tabla zona y se almacena
su identificador, si existe se almacena el identificador de la zona
existente */
   IF (existe = 0) THEN
        INSERT INTO ZONA (IDZONA, ZONA1, ZONA2, ZONA3,
        ZONA4, ZONA5, PAIS) VALUES
        (SYSTEM.SQ_IDZONA.nextval, v_zona1, v_zona2, v_zona3,
        v_zona4, v_zona5, var_pais);
```

```
            str_query := 'SELECT SYSTEM.SQ_IDZONA.currval from dual';
            execute immediate str_query into indice_Zona;
      ELSE
            str_query:='SELECT IDZONA from ZONA where pais = :pa and
            zona1 = :z1 and zona2 = :z2 and zona3 = :z3 and zona4 = :z4
            and zona5 = :z5';
            EXECUTE IMMEDIATE str_query INTO indice_zona USING
            var_pais, v_zona1, v_zona2, v_zona3, v_zona4, v_zona5;
      END IF;
/* Si el campo no comienza por # se trata de un inmueble */
ELSE
/* Se abre el cursor c2 */
            OPEN c2;
            /* Se inicializa a 'S' la variable venta y antiguedad a 0 */
            venta := 'S';
            antiguedad := 0;
            /* Consultamos si ya existe el inmueble */
            str_query:='SELECT COUNT (*) FROM INMUEBLE where
            codigo = :co';
            EXECUTE IMMEDIATE str_query INTO existe2 USING
            v_catalogo_rec.codigo;
/* Si no existe se genera el nuevo inmueble en la tabla inmueble y se
almacena su identificador, si existe se almacena el identificador del
inmueble, su antiguedad y el identificador de su última oferta */
            IF (existe2 = 0) THEN
                  DBMS_OUTPUT.PUT_LINE ('Datos nuevos1:
                  '||v_catalogo_rec.codigo ||' - '|| v_catalogo_rec.tipo||' - '||
                  v_catalogo_rec.planta||' - '|| v_catalogo_rec.metros);
                  INSERT INTO INMUEBLE (IDINMUEBLE, CODIGO,
                  TIPO, PLANTA, METROS) VALUES
                  (SYSTEM.SQ_IDINMUEBLE.nextval,
                  v_catalogo_rec.codigo, v_catalogo_rec.tipo,
                  v_catalogo_rec.planta, v_catalogo_rec.metros);
                  str_query := 'SELECT SYSTEM.SQ_IDINMUEBLE.currval
                  from dual';
                  execute immediate str_query into indice_inmueble;
            ELSE
                  str_query:='SELECT IDINMUEBLE from INMUEBLE
                  where codigo = :co';
                  EXECUTE IMMEDIATE str_query INTO indice_inmueble
                  USING v_catalogo_rec.codigo;
                  str_query:='SELECT MAX(IDOFERTA) FROM OFERTA
                  where idinmueble = :id';
                  EXECUTE IMMEDIATE str_query INTO indice_oferta
                  USING indice_inmueble;
```

```
            str_query:='SELECT antiguedad FROM OFERTA where
            idoferta = :idof';
            EXECUTE IMMEDIATE str_query INTO anti USING
            indice_oferta;
             antiguedad := antiguedad + anti;
END IF;
/* Consultamos si el inmueble que ya está dado de alta está
vendido */
str_query:='SELECT COUNT (*) FROM OFERTA where
idinmueble = :idin and vendido = :ven';
EXECUTE IMMEDIATE str_query INTO existe_vendido USING
indice_inmueble, venta;
/* Si no está vendido se leen las diferentes ofertas que existan
para el inmueble, en caso contrario no se hace nada con el
inmueble. */
/* Se entiende que si un inmueble se ha vendido y vuelve a
entrar en la inmobiliaria para gestionar su venta este debe
registrarse con un nuevo código, motivo por el que no haremos
nada con un inmueble ya registrado y vendido */
IF (existe_vendido = 0) THEN
/* Se inicializa la variable col a 1 */
    col := 1;
/* Bucle mientras tengamos columnas de la tabla
CATALOGO_ACT que leer */
LOOP
        FETCH c2 into v_columns_names;
        Exit when c2%NOTFOUND;
/* Si la columna es >= que 7 ó < que 0 entonces, la columna 7 es
la primera columna con precio */
        IF(col >= 7 or col < 0) THEN
          /* Comprobamos si es la primera vez que se ejecuta la
          actualización, si es así se debe registrar el nuevo mes/año
          en la tabla tiempo */
            IF (primerpaso=0) THEN
           /* Consultamos la ultima fecha registrada en la tabla
          tiempo, la siguiente a la última es la que damos de alta */
          primerpaso :=1;
          str_query:='SELECT MAX(IDTIEMPO) FROM TIEMPO';
           EXECUTE IMMEDIATE str_query INTO ult_itiempo;
          str_query:='SELECT nummes, año FROM TIEMPO where
          idtiempo = :idt';
          EXECUTE IMMEDIATE str_query INTO ult_nummes,
          ult_año USING ult_itiempo;
```

```
/* Comprobamos si el último mes registrado es diciembre,
ya que esto nos obligaría a cambiar de año y tratar de
forma diferente el campo mes */
IF (ult_nummes=12) THEN
   sig_nummes :=1;
   sig_mes :='ENERO';
   sig_año :=ult_año+1;
ELSIF (ult_nummes=11) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='DICIEMBRE';
   sig_año :=ult_año;
ELSIF (ult_nummes=10) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='NOVIEMBRE';
   sig_año :=ult_año;
ELSIF (ult_nummes=9) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='OCTUBRE';
   sig_año :=ult_año;
ELSIF (ult_nummes=8) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='SEPTIEMBRE';
   sig_año :=ult_año;
ELSIF (ult_nummes=7) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='AGOSTO';
   sig_año :=ult_año;
ELSIF (ult_nummes=6) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='JULIO';
   sig_año :=ult_año;
ELSIF (ult_nummes=5) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='JUNIO';
   sig_año :=ult_año;
ELSIF (ult_nummes=4) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='MAYO';
   sig_año :=ult_año;
ELSIF (ult_nummes=3) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='ABRIL';
   sig_año :=ult_año;
ELSIF (ult_nummes=2) THEN
   sig_nummes :=ult_nummes+1;
   sig_mes :='MARZO';
   sig_año :=ult_año;
```

```
            ELSIF (ult_nummes=1) THEN
              sig_nummes :=ult_nummes+1;
              sig_mes :='FEBRERO';
              sig_año :=ult_año;
            END IF;        /* Fin de los IF que comparan meses */
            /* Consultamos si ya existe el mes y el año */
          str_query:='SELECT COUNT (*) FROM TIEMPO where mes=
          :mes and año = :año';
          EXECUTE IMMEDIATE str_query INTO existe USING
          sig_mes, sig_año;
          /* Si no existe se genera nuevo registro en tabla TIEMPO, si
          existe se omite */
          IF (existe = 0) THEN
            str_query:='INSERT INTO TIEMPO (IDTIEMPO,
            NUMMES, MES, AÑO) VALUES
            (SYSTEM.SQ_IDTIEMPO.nextval, :nummes, :mes, :año)';
             EXECUTE IMMEDIATE str_query USING sig_nummes,
            sig_mes, sig_año;
             /* Almacenamos el identificador de tiempo que acabamos
            de generar */
            str_query := 'SELECT SYSTEM.SQ_IDTIEMPO.currval
            from dual';
            execute immediate str_query into indice_tiempo;
        END IF;
        END IF; /* Fin del IF primerpaso */
         /* Almacenamos el precio de la oferta en la variable precio */
        str_query:='SELECT precio FROM CATALOGO_ACT where
        codigo = :co';
        EXECUTE IMMEDIATE str_query INTO precio USING
        v_catalogo_rec.codigo;
         /* Almacenamos el precio inicial del inmueble en la variable
         precio_inicial */
       str_query:='SELECT PRECIO_INICIAL FROM INMUEBLE where
       idinmueble = :id';
       EXECUTE IMMEDIATE str_query INTO precio_inicial USING
       indice_inmueble;
        /* Si el precio de la oferta es null y el precio inicial es null,
       significa que el inmueble todavía no tiene registrada ninguna
       oferta, no se hace nada */
        IF (precio is NULL and precio_inicial is NULL) THEN
          DBMS_OUTPUT.PUT_LINE ('');
```

```
/* Si el precio de la oferta es distinto de null y el precio inicial es
null, significa que es la primera oferta que existe para el
inmueble, se aumenta la antiguedad en 1, se actualiza el precio
inicial del inmueble con el precio de la oferta, vendido a 'N' y
apreciación a 0. Por último se registra la oferta correspondiente
con los identificadores del resto de tablas y el resto de campos */
ELSIF (precio is not NULL and precio_inicial is NULL) THEN
 antiguedad := antiguedad +1;
 UPDATE INMUEBLE SET precio_inicial = precio WHERE
 idinmueble = indice_inmueble;
 INSERT INTO OFERTA (IDOFERTA, IDTIEMPO, IDINMUEBLE,
 IDZONA, IDDORMITORIO, VENDIDO, PRECIO,
 ANTIGUEDAD, APRECIACION)
 VALUES (SYSTEM.SQ_IDOFERTA.nextval, indice_Tiempo,
 indice_inmueble, indice_Zona, v_catalogo_rec.dormitorios, 'N',
 precio, antiguedad, 0);
/* Si el precio de la oferta es distinto de null y el precio inicial es
también distinto de null, significa que es una nueva oferta para el
inmueble, se aumenta la antiguedad en 1, el campo vendido se
pone a 'N' y apreciación será igual al precio de la oferta menos el
precio inicial. Por último se registra la oferta correspondiente con
los identificadores del resto de tablas y el resto de campos */
ELSIF(precio is not NULL and precio_inicial is not NULL) THEN
 antiguedad := antiguedad +1;
 INSERT INTO OFERTA (IDOFERTA, IDTIEMPO, IDINMUEBLE,
 IDZONA, IDDORMITORIO, VENDIDO, PRECIO,
 ANTIGUEDAD, APRECIACION)
VALUES (SYSTEM.SQ_IDOFERTA.nextval, indice_Tiempo,
indice_inmueble, indice_Zona, v_catalogo_rec.dormitorios, 'N',
precio, antiguedad, precio-precio_inicial);
/* Si el precio de la oferta es null ó 0 y el precio inicial es distinto
de null, significa que se ha producido la venta del inmueble, se
actualiza el campo vendido de la última oferta del inmueble a 'S'
para indicar que está vendido, se asigna el valor -2 a la variable
col para salir de bucle de lectura de ofertas, al venderse el
inmueble no es necesario seguir leyendo sus posibles ofertas */
ELSIF( (precio is NULL or precio = 0) and precio_inicial is not
NULL) THEN
 str_query:='SELECT MAX(IDOFERTA) FROM OFERTA where
 idinmueble = :id';
 EXECUTE IMMEDIATE str_query INTO indice_oferta USING
 indice_inmueble;
 UPDATE OFERTA SET vendido = 'S' WHERE idoferta =
 indice_oferta;
 col := -2;
END IF;
```

```
        /* Fin del if mientras la columna es >= a 7 o menor que 0 */
        END IF;
          /* Se aumenta la variable col en 1, para leer la siguiente oferta */
          col := col+1;
          /* Fin del bucle mientras tengamos columnas de la tabla
          catalogo_act */
      END LOOP;
          /* Fin del if que comprueba si el inmueble está vendido */
      END IF;
      /* Se cierra el cursor c2 */
      CLOSE c2;
     /* Fin del if que comprueba si el primer valor de código es # */
    END IF;
DBMS_OUTPUT.PUT_LINE('Code: ' || v_catalogo_rec.codigo || ' -
Zona: '|| v_catalogo_rec.zona);
/* Bucle contador para realizar el commit cada 1000 pasos */
IF(MOD(COU,1000)=0) THEN
        COMMIT;
END IF;
/* Se aumenta el contador de pasos en 1 */
cou:= cou+1;
/* Fin del bucle mientras el cursor c1 tenga datos */
End Loop;
/* Se cierra el cursor c1 */
CLOSE c1;
/* Commit final */
COMMIT;
END;  /
```

- **Lanza1_act.sql:** Script que realiza la conexión con Oracle y llama al script 1-Crea_Tabla_Catalogo_act.sql.

```
/* Lanza el script de creación de la tabla temporal catalogo_act */
conn system/uoc
@e:\oraclexe\TFC\TFC2011\1-Crea_Tabla_Catalogo_act.sql
```

- **Lanza2_act.sql:** Script que realiza la conexión con Oracle y llama a los siguientes scripts:

```
/* Lanza el resto de scripts necesarios para la actualización */
conn system/uoc
@e:\oraclexe\TFC\TFC2011\2-Unificar_Datos_act.sql
@e:\oraclexe\TFC\TFC2011\3-Unificar_Precios_act.sql
@e:\oraclexe\TFC\TFC2011\4-Mytable.sql
@e:\oraclexe\TFC\TFC2011\5-SKELETON_act.sql
Execute skeleton_act
```

- **Sqlldr_act.bat:** Este proceso esta ubicado en la ruta 'C:\Documents and Settings\uocdw' de la maquina virtual, se encarga de realizar la carga de datos que nos facilitan en la tabla temporal 'catalogo_act'. Para que funcione correctamente debe existir el fichero 'cargador_act.ctl' y el fichero de texto 'Catalogo_act.txt' con los datos mensuales que nos facilitan en la ruta E:\oraclexe de la maquina virtual.

```
/* Lanza SQL*Loader */
cd e:\oraclexe
e:
sqlldr userid=SYSTEM/uoc Control=cargador_act.ctl
data=catalogo_act.txt
```

- **Cargador_act.ctl:** Archivo de control que se utiliza para que funcione correctamente SQL*Loader. En este fichero se especifica la ruta del fichero *.txt que contiene los datos mensuales que nos facilitan, la tabla destino en la que se cargarán los mismos, los campos que se cargarán y el carácter que hace de separador de estos campos.

```
/* Fichero control de SQL*Loader */
load data
infile 'c:\oraclexe\catalogo_act.txt'
into table catalogo_act
insert
fields terminated by x'09'
trailing nullcols
(codigo, zona, tipo, planta, dormitorios, metros, precio)
```

- **Catalogo_act.txt.** Fichero de texto que nos facilitan con los datos mensuales de la inmobiliaria.

- **Actualización.bat.** Proceso batch, al que llama la tarea programada que ejecuta de forma automática el proceso de actualización mensual.

```
/* Proceso que llama a los scripts necesarios para realizar la
actualización mensual de datos de forma automática */
E:\oraclexe\app\oracle\product\10.2.0\server\BIN\sqlplus.exe
system/uoc @e:\oraclexe\TFC\TFC2011\1-
Crea_Tabla_Catalogo_act.sql
cd e:\oraclexe
e:
sqlldr userid=SYSTEM/uoc Control=cargador_act.ctl
data=catalogo_act.txt
c:
E:\oraclexe\app\oracle\product\10.2.0\server\BIN\sqlplus.exe
system/uoc @e:\oraclexe\TFC\TFC2011\lanza2_act.sql
exit
```

11.3. Anexo - Diagrama de Gantt

Diagrama de Gantt completo que se corresponde con la planificación de tareas.

	Task Name	Duración	Comienzo	Fin	Prede
1	**1. Inicio del proyecto**	**1 día**	**mié 21/09/11**	**mié 21/09/11**	
2	1.1. Descarga material asignatura y lectura plan docente	1 día	mié 21/09/11	mié 21/09/11	
3	**2. Elaboración PEC1. Plan de Trabajo y Análisis preliminar**	**14 días**	**jue 22/09/11**	**mié 05/10/11**	
4	2.1. Descarga y lectura del enunciado	1 día	jue 22/09/11	jue 22/09/11	2
5	2.2. Elaborar plan de trabajo	1 día	vie 23/09/11	vie 23/09/11	4
6	2.3. Consultar bibliografía	1 día	sáb 24/09/11	sáb 24/09/11	5
7	2.4. Análisis de requerimientos	1 día	dom 25/09/11	dom 25/09/11	6
8	2.5. Elaborar borrador PEC1	3 días	lun 26/09/11	mié 28/09/11	7
9	2.6. Entrega borrador generado	1 día	jue 29/09/11	jue 29/09/11	8
10	2.7. Preparar hardware e instalar software necesario	2 días	vie 30/09/11	sáb 01/10/11	9
11	2.8. Elaborar PEC1	3 días	dom 02/10/11	mar 04/10/11	10
12	2.9. Entregar PEC1	1 día	mié 05/10/11	mié 05/10/11	11
13	**3. Elaboración PEC2. Análisis de requerimientos y diseño técnico**	**35 días**	**jue 06/10/11**	**mié 09/11/11**	3
14	3.1. Lectura y comprensión	1 día	jue 06/10/11	jue 06/10/11	12
15	3.2. Diseño Almacén de datos	4 días	vie 07/10/11	lun 10/10/11	14
16	3.3. Diseño lógico	5 días	mar 11/10/11	sáb 15/10/11	15
17	3.4. Extracción datos de origen	6 días	dom 16/10/11	vie 21/10/11	16
18	3.5. Transformar datos de origen	5 días	sáb 22/10/11	mié 26/10/11	17
19	3.6. Elaborar borrador PEC2	5 días	jue 27/10/11	lun 31/10/11	18
20	3.7. Entrega borrador generado	1 día	mar 01/11/11	mar 01/11/11	19
21	3.8. Elaborar PEC2	7 días	mié 02/11/11	mar 08/11/11	20
22	3.9. Entregar PEC2	1 día	mié 09/11/11	mié 09/11/11	21

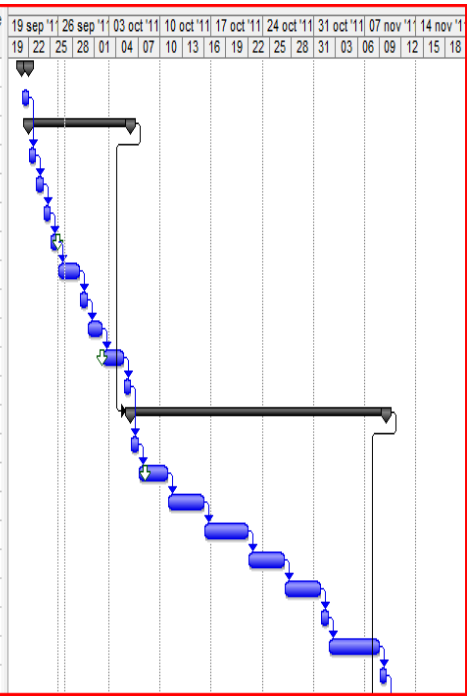


Figura 24a. Diagrama de Gantt completo parte 1

	Task Name	Duración	Comienzo	Fin	Prede
23	**4. Elaboración PEC3. Implementación**	**40 días**	**jue 10/11/11**	**mié 21/12/11**	13
24	4.1. Lectura y comprensión	1 día	jue 10/11/11	jue 10/11/11	22
25	4.2. Configurar base de datos	4 días	vie 11/11/11	lun 14/11/11	24
26	4.3. Carga de datos	5 días	mar 15/11/11	sáb 19/11/11	25
27	4.4. Pruebas	8 días	dom 20/11/11	dom 27/11/11	26
28	4.5. Elaborar informes	9 días	lun 28/11/11	mar 06/12/11	27
29	4.6. Elaborar borrador PEC3	4 días	mié 07/12/11	sáb 10/12/11	28
30	4.7. Entrega borrador generado	1 día	dom 11/12/11	dom 11/12/11	29
31	4.8. Elaborar PEC3	7 días	lun 12/12/11	mar 20/12/11	30
32	4.9. Entregar PEC3	1 día	mié 21/12/11	mié 21/12/11	31
33	**5. Entrega Final**	**18 días**	**jue 22/12/11**	**mar 10/01/12**	23
34	5.1. Revisión final	3 días	jue 22/12/11	sáb 24/12/11	32
35	5.2. Conclusión final	2 días	dom 25/12/11	lun 26/12/11	34
36	5.3. Elaborar memoria	6 días	mar 27/12/11	dom 01/01/12	35
37	5.4. Elaborar presentación	6 días	lun 02/01/12	lun 09/01/12	36
38	5.5. Entrega final	1 día	mar 10/01/12	mar 10/01/12	37
39	**6. Debate virtual y defensa del proyecto**	**8 días**	**mié 11/01/12**	**vie 20/01/12**	33
40	6.1. Intervención en el debate	8 días	mié 11/01/12	vie 20/01/12	38

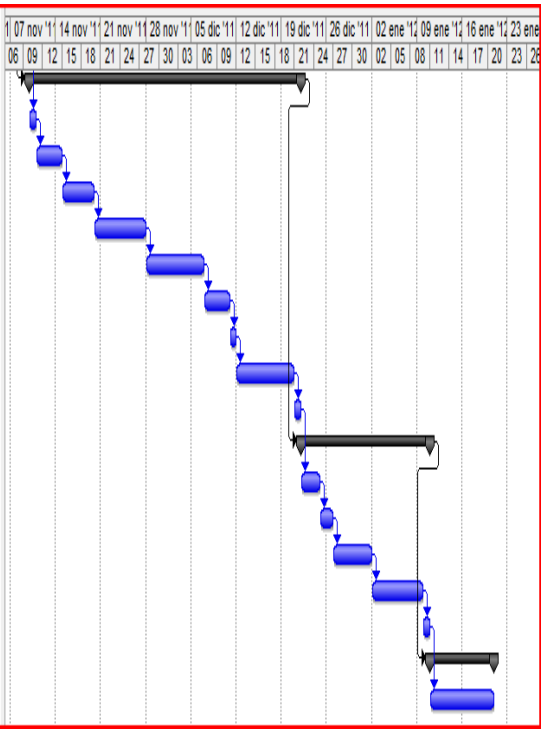


Figura 24b. Diagrama de Gantt completo parte 2