TRABAJO DE FINAL DE CARRERA

CONSTRUCCIÓN Y EXPLOTACIÓN DE UN ALMACÉN DE DATOS PARA EL ANÁLISIS DEL SISTEMA DE PRESTACIONES SOCIALES

Antonio Vaquero de Oro

ETIG

CONSULTOR: Xavier Plaza Sierra

Memoria.

RESUMEN Y PALABRAS CLAVE

Resumen

El objetivo principal del proyecto que se presenta a continuación, es el de diseñar, construir y explotar un *almacén de datos*, a partir de la información entregada por las Comunidades Autónomas (excepto País Vasco y Navarra, que tienen régimen foral propio) y el Instituto Nacional de Estadísticas (INE), para determinar la sostenibilidad del sistema de prestaciones sociales y conocer la evolución de dichas prestaciones.

La información de partida viene dada en una serie de archivo texto en formato CSV.

Disponiendo de toda la información se procede a realizar un análisis previo y al diseño de los distintos modelos que vamos a necesitar (conceptual, lógico y físico).

Posteriormente se procede a extraer, tratar y almacenar la información en una base de datos para la creación de un cubo multidimensional (OLAP).

Por último, se crea un área de negocio que sirve de diseño e implementación de una serie de consultas e informes solicitados por el cliente.

Palabras clave:

Almacén de datos, AtlasSBI, *Data Warehouse,* ETL, OLAP, Oracle, SQL, PL/SQL, Script.

Índice Contenidos

Índice Figuras

1 INTRODUCCIÓN

El presente documento corresponde a la memoria del Trabajo Fin de Carrera donde se sitúa a lector ante el proyecto presentando en el enunciado.

El objetivo final es el de mostrar la capacidad de síntesis del alumno en referencia a los conocimientos logrados a lo largo de toda la carrera.

El Trabajo Fin de Carrera o TFC que se presenta a continuación, corresponde al área de Almacenes de Datos o *Data Warehause*.

1.1 Justificación

El presente proyecto es consecuencia de la petición realizada por el Organismo de Análisis del Departamento de Trabajo (OADT), que ha decido realizar un estudio para determinar la sostenibilidad del sistema de prestaciones sociales y conocer la evolución de dichas prestaciones.

El Organismo se ha visto obligado a solicitar a las Comunidades Autónomas (CCAA) y al INE la información necesaria para realizar el estudio, debido a la gestión distribuida de las competencias sociales y la ley de protección de datos.

Una vez finalizado el proyecto el cliente tendrá una serie de mejoras entre las que podemos destacar:

- La automatización de los procesos de recogida y manipulación de los datos de los que disponen.
- La gran funcionalidad que ofrecen los informes finales que proporcionarán al usuario una mayor rapidez y agilidad en la toma de decisiones.
- La estructuración y almacenaje de los datos en un modelo físico implementado en Oracle 10g para su posterior explotación con herramientas de Business Intelligence (AtlasSBI).

1.2 Objetivos de la asignatura

En este apartado se describen los objetivos que el Trabajo de Fin de Carrera pretende alcanzar en relación con los objetivos generales de la asignatura y los objetivos específicos del proyecto expuesto en el enunciado.

Los objetivos principales de esta asignatura se pueden resumir en los siguientes:

- Consolidar los conocimientos adquiridos en otras asignaturas del plan de estudios, acerca de la elaboración de documentos de especificaciones, basado en los requerimientos del usuario.
- Analizar y estudiar la problemática mostrada en el enunciado del proyecto.
- Planificar el proyecto a realizar durante el periodo de tiempo marcado por el plan docente.
- Realizar la solución práctica del problema planteado.
- Realizar una presentación del desarrollo y resultados finales del proyecto.

1.3 Objetivos del Trabajo de Fin de Carrera

Los objetivos específicos son los solicitados por el Organismo de Análisis del Departamento de Trabajo (OADT) que den soporte a la toma de decisiones.

Los objetivos principales del proyecto son los siguientes:

- Crear un proceso automatizado de importación de los datos facilitados por las diferentes Comunidades Autónomas y el INE.
- Diseñar y poner en marcha el Almacén de Datos.
- Generar los informes solicitados por la empresa.
    - Total retribución.
    - Total retención.
    - Retribución media.
    - % de retención medio.
    - Número de retribuciones medias
    - % de población por segmento.
    - Número de Trabajadores/Número de perceptores no activos
    - Número de Trabajadores/Habitantes.
    - Número de Trabajadores/Número de personas activas
- Proyección de los indicadores Número de Trabajadores/Número de perceptores no activos y Número de Trabajadores/Población para determinar el punto de colapso que sería el momento en que dos trabajadores mantengan a un perceptor o se iguale el número de

trabajadores y población perceptora (no activos + perceptores de prestación social).

1.4 Enfoque y metodología.

El enfoque y el método a seguir para la realización de este trabajo es el determinado por los conocimientos que se han adquirido en los estudios de diferentes asignaturas.

Básicamente podemos resumir el método a seguir según el esquema siguiente:

- **Fase de análisis previo:**
  - Búsqueda de información adicional.
  - Recogida de la bibliografía.
  - Lectura de los diferentes conceptos del proyecto.
  - Realización del plan de trabajo.
- **Fase de requisitos**
  - Análisis y documentación de requerimientos de los usuarios.
- **Diseño**
  - Realización del modelo dimensional.
  - Realización del diseño procedimental.
  - Construcción de toda la documentación del proyecto.
- **Implementación**
  - Construcción Base de datos en Oracle con sus relaciones e índices.
  - Extracción, transformación y carga de los datos facilitados.
  - Análisis de la información que poseemos hasta el momento.
  - Instalación del software de Business Intelligence (AtlasSBI).
  - Familiarización con la herramienta y posibles problemas de instalación.
  - Realización Informes y conclusiones finales.

1.5 Planificación.

Se toman como base las fases de la implantación de un almacén de datos comentadas en el punto anterior, se combinan con las fechas claves previstas en el plan docente.

En el cuatrimestre actual las fechas claves de entrega son las siguientes:

TRABAJO	TITULO	FECHA INICIO	FECHA FIN
PEC 1	Plan de trabajo y análisis preliminar de los requerimientos.	01-MAR-2012	16-MAR-2012
PEC 2	Análisis de requerimientos y diseño conceptual y técnico	17-MAR-2012	20-ABR-2012
PEC 3	Implementación	21-ABR-2012	25-MAY-2012
MEMORIA	Entrega Final	26-MAY-2012	13-JUN-2012
DEBATE	Debate	25-JUN-2012	28-JUN-2012

Figura 1 Tabla Fechas clave

1.5.1 Tareas

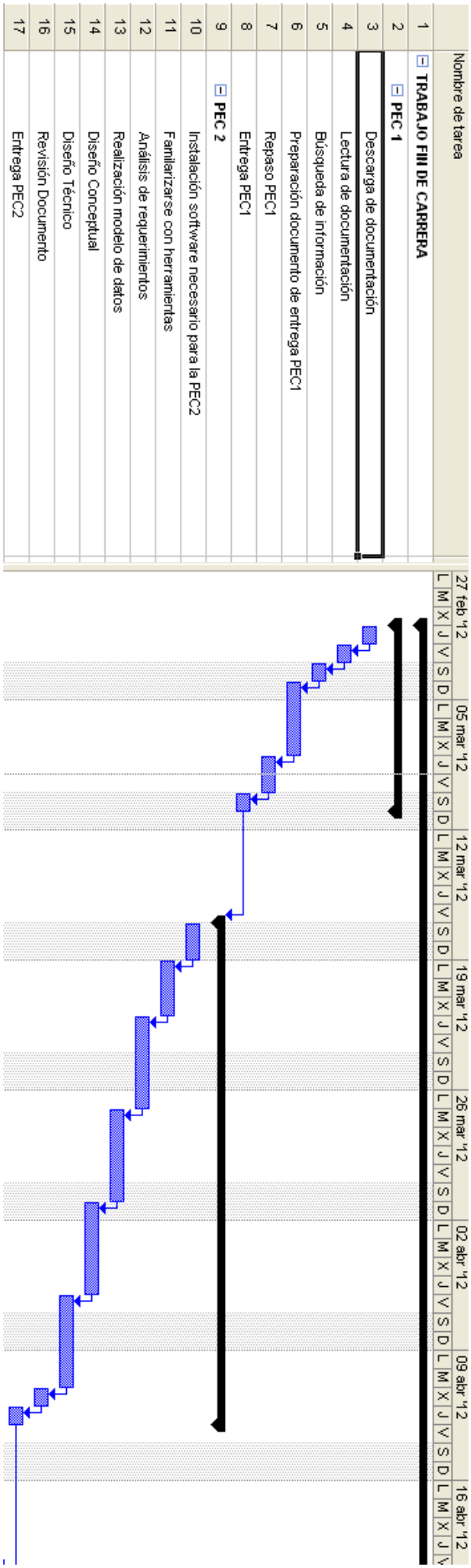
A continuación se muestra la división del Trabajo Fin de Carrera en actividades y tareas. El ritmo de trabajo será de una hora en los días laborales y de las necesarias en los fines de semana.

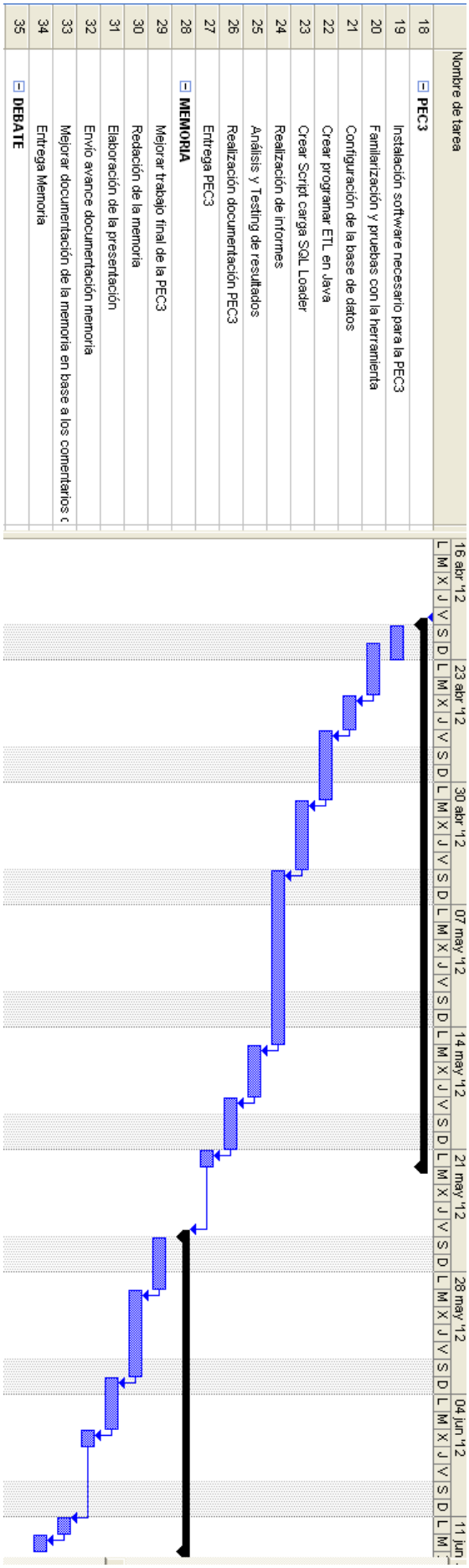
	Nombre de tarea	Comienzo	Duración	Fin	Predecesoras
1	**TRABAJO FIN DE CARRERA**	**jue 01/03/12**	**120 días**	**vie 29/06/12**	
2	**PEC 1**	**jue 01/03/12**	**9,5 días**	**sáb 10/03/12**	
3	Descarga de documentación	jue 01/03/12	1 día	vie 02/03/12	
4	Lectura de documentación	vie 02/03/12	1 día	sáb 03/03/12	3
5	Búsqueda de información	sáb 03/03/12	1 día	dom 04/03/12	4
6	Preparación documento de entrega PEC1	dom 04/03/12	4 días	jue 08/03/12	5
7	Repaso PEC1	jue 08/03/12	2 días	sáb 10/03/12	6
8	Entrega PEC1	sáb 10/03/12	0,5 días	sáb 10/03/12	7
9	**PEC 2**	**sáb 17/03/12**	**27 días**	**vie 13/04/12**	**8**
10	Instalación software necesario para la PEC2	sáb 17/03/12	2 días	lun 19/03/12	
11	Familarizarse con herramientas	lun 19/03/12	3 días	jue 22/03/12	10
12	Análisis de requerimientos	jue 22/03/12	5 días	mar 27/03/12	11
13	Realización modelo de datos	mar 27/03/12	5 días	dom 01/04/12	12
14	Diseño Conceptual	dom 01/04/12	5 días	vie 06/04/12	13
15	Diseño Técnico	vie 06/04/12	5 días	mié 11/04/12	14
16	Revisión Documento	mié 11/04/12	1 día	jue 12/04/12	15
17	Entrega PEC2	jue 12/04/12	1 día	vie 13/04/12	16
18	**PEC3**	**sáb 21/04/12**	**31 días**	**mar 22/05/12**	**17**
19	Instalación software necesario para la PEC3	sáb 21/04/12	2 días	lun 23/04/12	
20	Familarización y pruebas con la herramienta	dom 22/04/12	3 días	mié 25/04/12	
21	Configuración de la base de datos	mié 25/04/12	2 días	vie 27/04/12	20
22	Crear programar ETL en Java	vie 27/04/12	4 días	mar 01/05/12	21
23	Crear Script carga SQL Loader	mar 01/05/12	4 días	sáb 05/05/12	22
24	Realización de informes	sáb 05/05/12	10 días	mar 15/05/12	23
25	Análisis y Testing de resultados	mar 15/05/12	3 días	vie 18/05/12	24
26	Realización documentación PEC3	vie 18/05/12	3 días	lun 21/05/12	25
27	Entrega PEC3	lun 21/05/12	1 día	mar 22/05/12	26
28	**MEMORIA**	**sáb 26/05/12**	**18 días**	**mié 13/06/12**	**27**
29	Mejorar trabajo final de la PEC3	sáb 26/05/12	3 días	mar 29/05/12	
30	Redación de la memoria	mar 29/05/12	5 días	dom 03/06/12	29
31	Elaboración de la presentación	dom 03/06/12	3 días	mié 06/06/12	30
32	Envío avance documentación memoria	mié 06/06/12	1 día	jue 07/06/12	31
33	Mejorar documentación de la memoria en base a los comentarios c	lun 11/06/12	1 día	mar 12/06/12	32
34	Entrega Memoria	mar 12/06/12	1 día	mié 13/06/12	33
35	**DEBATE**	**lun 25/06/12**	**4 días**	**vie 29/06/12**	
36	Inicio Debate	lun 25/06/12	3 días	jue 28/06/12	
37	Fin Debate	jue 28/06/12	1 día	vie 29/06/12	36

Figura 2 Tabla división de trabajo

1.5.2 Diagrama de Gannt

A continuación se muestra el diagrama de Gannt correspondiente al Trabajo de Fin de Carrera.


Nombre de tarea
1 TRABAJO FIN DE CARRERA
2 PEC 1
3 Descarga de documentación
4 Lectura de documentación
5 Búsqueda de información
6 Preparación documento de entrega PEC1
7 Repaso PEC1
8 Entrega PEC1
9 PEC 2
10 Instalación software necesario para la PEC2
11 Familiarizarse con herramientas
12 Análisis de requerimientos
13 Realización modelo de datos
14 Diseño Conceptual
15 Diseño Técnico
16 Revisión Documento
17 Entrega PEC2
27 feb '12
05 mar '12
12 mar '12
19 mar '12
26 mar '12
02 abr '12
09 abr '12
16 abr '12


Nombre de tarea
PEC3
Instalación software necesario para la PEC3
Familiarización y pruebas con la herramienta
Configuración de la base de datos
Crear programar ETL en Java
Crear Script carga SQL Loader
Realización de informes
Análisis y Testing de resultados
Realización documentación PEC3
Entrega PEC3
MEMORIA
Mejorar trabajo final de la PEC3
Redacción de la memoria
Elaboración de la presentación
Envío avance documentación memoria
Mejorar documentación de la memoria en base a los comentarios c
Entrega Memoria
DEBATE
16 abr '12
23 abr '12
30 abr '12
07 may '12
14 may '12
21 may '12
28 may '12
04 jun '12
11 jun
L M X J V S D

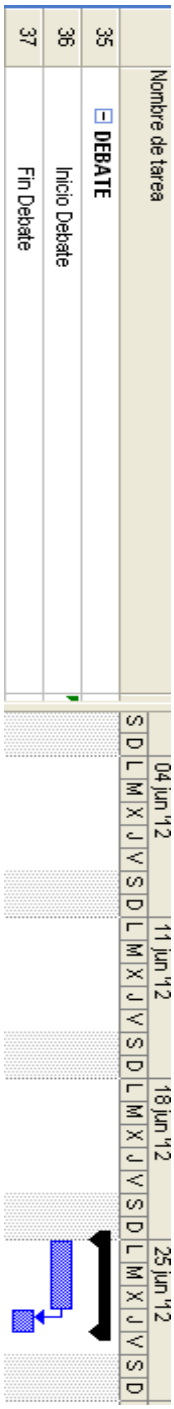


Figura 3 Diagrama de Gantt

1.5.3 Incidencias y Riesgos

A continuación se exponen los posibles riesgos e incidencias que son posibles prever en el momento de realizar la planificación del proyecto, así como las posibles soluciones aplicables en cada caso concreto.

1.5.3.1 Problemas por motivos laborales

Se pueden producir coincidencias de fechas en los momentos de mayor carga de trabajo con situaciones excepcionales en el trabajo que exijan un mayor tiempo de presencia en la oficina o viajes no previstos. La solución prevista en estos casos es la de desplazar las tareas y acomodarlas al nuevo calendario. Para poder realizarlo se han planificado las tareas con algún día de colchón que permita hacer estos ajustes.

1.5.3.2 Problemas de salud

Esta contingencia es difícilmente solucionable si el problema de salud se alarga en los días, para solucionarlo como se ha indicado en el punto anterior las tareas han sido planificadas con holgura para poder desplazadas y acomodarlas a la nueva situación.

1.5.3.3 Problemas técnicos

A lo largo de la elaboración del proyecto se pueden encontrar problemas técnicos con el equipo empleado, problemas con la instalación del software y problemas con el envío de la máquina virtual, ya que requerirá el envío de un archivo de un tamaño elevado. Para solucionarlo al igual que en los anteriores problemas se han planificado las tareas con amplitud de fechas, además de tener previsto una copia de seguridad diaria en un disco duro portátil.

1.6 Productos obtenidos

Durante la elaboración de este proyecto de final de carrera y como resultado del mismo se han generado además de los documentos que han constituido las PECS I,II,III, los siguientes documentos y archivos.

- Aplicaciones Java que realizarán el tratamiento inicial de los datos entregados por las Comunidades Autónomas y INE, dando como producto tres archivos de extensión CSV para la carga automática de la tablas temporales Tributaciones_tmp,Poblacion_tmp y PorcPoblActiva_tmp a través de *Oracle SQL Loader*.
- 3 ficheros planos CSV con la información a cargar en las tablas temporales
- Script de carga de datos.
- Script de borrado de datos.
- Fichero de control para la carga de datos con *Oracle SQL Loader*.
- Fichero log de errores correspondiente a la carga de datos con *Oracle SQL Loader.*
- Máquina virtual con el software del proyecto instalado y cargado.
    - Base de datos Oracle con la información actualizada
    - Informes solicitados en AtlasSBI.
- Memoria final del proyecto.
- Presentación en vídeo del proyecto.

1.7 Próximos capítulos

En los próximos capítulos se describirán los procesos de análisis, diseño y se presentará una breve descripción de los informes realizados junto con una captura de pantalla de cada reporte.

En el capítulo correspondiente al análisis se estudiarán los diagramas de casos de uso y del modelo conceptual con sus correspondientes explicaciones.

En el capítulo referente al diseño se expondrán los siguientes apartados:

- Diagrama de arquitectura de software.
- Diagrama de arquitectura de hardware.
- Diseño de la base de datos y diagrama del modelo físico.

Finalmente tendremos un capítulo donde se mostrarán los resultados obtenidos por los diferentes informes creados en *AtlasSBI.*

2 ANÁLIS DE REQUISITOS

En este apartado se analizan los datos de origen entregados por las Comunidades Autónomas (excepto País Vasco y Navarra) y el INE (Instituto Nacional de Estadística) que constituyen el punto de partida del proyecto. A partir de la información obtenida del análisis de dichos datos se realizará el diseño conceptual.

Con el diseño conceptual, el diseño lógico y el diseño físico estaremos en disposición de definir las tablas de la base de datos que soportarán la información necesaria para cubrir las exigencias de OADT.

2.1 Análisis de los datos de origen

En primer lugar se va a proceder al análisis de los archivos enviados en formato CSV por las diferentes Comunidades Autónomas y Ciudades Autónomas (excepto País Vasco y Navarra, que tienen un régimen foral propio).

El OADT nos ha advertido que debido a que la información se ha extraído de diferentes sistemas, hay tres formatos CSV diferentes separados por coma, tabulador y punto y coma.

También nos han indicado que los archivos aunque tengan estructura similar, no todos tienen la misma información.

Todos los archivos incorporan en su estructura el año, la Comunidad Autónoma y los diferentes tipos de receptor.

En cambio para el tipo de retribución aparecen 6 tipos distintos de estructuras de información donde el proceso ETL deberá de identificar y obtener la información necesaria para el Almacén de Datos de forma idónea.

Esta es la tabla resumen que muestra los 6 tipos de estructuras. La tabla incluye 2 columnas, la primera que indica los tipos retribución que se muestra en el archivo y la segunda la información entregada para cada tipo de retribución.

TIPO RETRIBUCIÓN	INFORMACIÓN
TOTAL+SALARIOS+PENSIONES+DESEMPLEO	
	PERSONAS RETRIBUCIONES RETRIBUCIÓN MEDIA ANUAL RETENCIONES TIPO MEDIO DE RETENCIÓN
TOTAL+SALARIOS+PENSIONES+DESEMPLEO	
	PERSONAS RETRIBUCIONES RETENCIONES
SALARIOS+PENSIONES+DESEMPLEO	
	PERSONAS RETRIBUCIÓN MEDIA ANUAL TIPO MEDIO DE RETENCIÓN
SALARIOS+PENSIONES+DESEMPLEO	
	PERSONAS RETRIBUCIONES RETENCIONES
SALARIOS+PENSIONES+DESEMPLEO	
	PERSONAS RETRIBUCIONES RETRIBUCIÓN MEDIA ANUAL RETENCIONES TIPO MEDIO DE RETENCIÓN
*** CEUTA Y MELILLA EN EL MISMO ARCHIVO	

Figura 4 Tabla Estructura de datos Tributaciones

A continuación se procede a describir el significado de cada registro en la columna Tipo de Retribución:

- **Salarios.** Información referente las retribuciones obtenidas por trabajos realizados.
- **Pensiones.** Corresponde a las retribuciones obtenidas por pensionistas (jubilación).

- **Desempleo.** Retribuciones obtenidas por el cobro del subsidio de desempleo.
- **Total.** Resumen de los 3 tipos de retribuciones indicadas anteriormente.

Finalmente se procede a detallar el significado de los registros indicados en la columna Información (ver Figura 1):

- **Personas.** Número de perceptores correspondientes al tipo de retribución al cual están ligados. En el caso que una persona tenga más de un tipo de retribución aparecerá simultáneamente en cada tipo de retribución. En el proyecto este dato se registrará en la tabla de hechos de Tributaciones con el nombre de "num_retribuciones" y será de utilidad para calcular el número de personas en cada tipo de retribución en la tabla de hechos de Población. Esta decisión se explica en el apartado de modelado multidimensional.
- **Retribuciones.** Importe cobrado para el tipo de retribución correspondiente. En la tabla de hechos Tributaciones aparece con el nombre "imp_retribuciones". En algunos archivos no aparece esta información directamente y tendrá que ser calculada en base al número de personas y la retribución media anual.
- **Retribución media anual.** Importe medio cobrado correspondiente al tipo de retribución. Este dato será necesario para los archivos en el que no aparezca el dato de las retribuciones. Esta información no se cargará en ninguna tabla ya que no es necesario para ningún informe solicitado y el caso de requerirlo es fácilmente calculable con la siguiente fórmula Retribuciones/Personas.
- **Retenciones.** Importe total retenido en forma de impuestos. Al igual que el campo Retribuciones en algunos archivos nos aparece y debe de ser calculado con el campo Tipo Retribución media anual y Personas. En la tabla de hechos Tributación aparece con el nombre de "imp_retenciones"
- **Tipo medio Retención.** Porcentaje medio retenido para cada tipo de retribución. Este campo será necesario para calcular el importe retenido en los archivos que no aparezca el campo Retenciones. Esta información al igual que el dato Retribución media anual no será registrado en ninguna tabla. Los motivos son los mismos que los explicados en el punto Retribución media anual.

A continuación se describe la información aportada por el INE (población y porcentaje de población activa)

La información correspondiente a la población ha sido entregada en 7 archivos. Cada archivo identifica a un año (desde el 2005 hasta el 2011). Los datos son aportados distinguiendo el sexo, para este proyecto esta identificación no es útil por lo que el proceso ETL deberá de sumar la población de ambos sexos y calcular cuántas personas corresponden a cada segmento. El dato final será registrado en el campo *habitantes* dentro de la tabla de hecho Población.

El archivo porc población activa, nos indica el porcentaje de población activa para cada Comunidad Autónoma para los años comprendidos desde el 2005 hasta el 2010. Este dato se utilizará para calcular el campo *habitantes* de la tabla de hecho Población. No se registrará en ninguna tabla final del Almacén de Datos.

2.2 Segmentos de población.

En este apartado se describen y se indica cómo se calculan los diferentes segmentos de población.

En primer nivel de clasificación es el correspondiente a si una persona es activa o no.

Persona **Activa** se entiende como la persona en edad laboral que está trabajando o que buscan empleo. Siguiendo esta definición en nuestro proyecto como activa se incluirá a las personas asalariadas y desempleadas (diferenciando las personas que cobran subsidio y las que no).

Como persona **Pasiva** se entiende a las personas que no están en edad laboral (niños, ancianos etc.) además de personas que no buscan empleo (amas de casa, estudiantes, autónomos, etc.).

A continuación se muestra una tabla resumen con los diferentes segmentos.

TIPO	SEGMENTO
ACTIVA	
	ASALARIADOS
	DESEMPLEADOS
	ASALARIADOS+DESEMPLEADOS
	DESEMPLEADOS+PENSIONISTAS
	ASALARIADOS+DESEMPLEADOS+PENSIONISTAS
	DESEMPLEADOS SIN PRESTACIÓN
NO ACTIVA	
	PENSIONISTAS
	OTROS NO ACTIVA

Figura 5 Tabla de segmentos

El significado de cada segmento es el siguiente:

- **Asalariados.** Personas que únicamente reciben salarios por trabajos realizados
- **Desempleados.** Personas que perciben subsidio por desempleo
- **Asalariados+Desempleados.** Personas que reciben a la vez un salario parcial y una prestación por desempleo.
- **Desempleados+Pensionistas.** Personas que están desempleadas pero que reciben a la vez una pensión (por ejemplo por enfermedad)
- **Asalariados+Desempleados+Pensionistas.** Personas que reciben los 3 tipos de retribución a la vez.
- **Desempleado sin prestación.** Personas que en desempleo que no reciben prestación por desempleo, ya sea por haber finalizado el plazo o por no haber cotizado lo suficiente aún.
- **Pensionistas.** Personas que únicamente reciben pensión
- **Otros pasiva.** Personas que no reciben ningún tipo de retribución y que no están buscando empleo ya sea por edad o por no tener desearlo. Por ejemplo serían niños, amas de casa jubilid@s sin prestación etc.

2.2.1. Cálculo personas por segmento

En este apartado se indica el origen del dato o el algoritmo a realizar por el sistema para calcular cada segmento. Indicar que los habitantes de cada segmento se calcularán para cada Comunidad Autónoma y año, haciéndose los cálculos en base a las tablas temporales creadas a través de un procedimiento almacenado de PL/SQL.

TIPO POBLACIÓN	SEGMENTO	ARCHIVO	Tipo de Perceptor	Registro	Tipo de Retribución	Algoritmo
ACTIVA						
	ASALARIADOS	Tributaciones	Asalariado	Personas	Salarios	
	DESEMPLEADOS	Tributaciones	Desempleados	Personas	Desempleo	
	ASALARIADOS+DESEMPLEADOS	Tributaciones	Asalariados y desempleados	Personas	Σ Personas(Salarios y Desempleo)/2	
	DESEMPLEADOS+PENSIONISTAS	Tributaciones	Pensionistas y desempleados	Personas	Σ Personas(Pensiones y Desempleo)/2	
	ASALARIADOS+DESEMPLEADOS+PENSIONISTAS	Tributaciones	Asalariado, Asalariados y desempleados	Personas	Σ (Personas(Salarios,Pensiones y Desempleo)/3)*2	
	DESEMPLEADOS SIN PRESTACIÓN	porc población activa y poblacion por ccaa				%población activa*(habitantes)-(Σ Personas Resto Segmentos Activa)
NO ACTIVA						
	PENSIONISTAS	Tributaciones	Pensionistas	Personas	Pensiones	
	OTROS NO ACTIVA	porc población activa y poblacion por ccaa				100%-(%población activa*(habitantes)-Pensionistas))

Figura 6 Tabla cálculo población por segmento

2.3 Requerimientos funcionales y no funcionales

En este apartado se van a describir los requisitos que el Almacén de Datos debe de cumplir para satisfacer las necesidades del OADT.

Los requerimientos funcionales definen qué debe de hacer el sistema a desarrollar y los requerimientos no funcionales definen cómo debe de ser el sistema.

Siguiendo las definiciones anteriores a continuación se describen los diferentes requerimientos detectados.

2.3.1. Requerimientos funcionales

Los requerimientos funcionales detectados a través del enunciado son los siguientes:

- Proporcionar el conjunto de informes solicitados por el OADT con una temporalidad de un año y que se podrán consultar de forma agregada, por Comunidad Autónoma, tipo de perceptor y tipo de retribución.

  Los informes solicitados son los siguientes:

  - Total retribución.
  - Total retención.
  - Retribución media.

  - % de retención medio.
  - Número de retribuciones medias
  - % de población por segmento.
  - Número de Trabajadores/Número de perceptores no activos
  - Número de Trabajadores/Habitantes.
  - Número de Trabajadores/Número de personas activas

- Proyección de los indicadores Número de Trabajadores/Número de perceptores no activos y Número de Trabajadores/Población para determinar el punto de colapso que sería el momento en que dos trabajadores mantengan a un perceptor o se iguale el número de trabajadores y población perceptora (no activos + perceptores de prestación social).

- El número de habitantes será calculado como media de los valores a 1 de enero del año y el 1 de enero del año siguiente.
- Identificar si un tipo de perceptor es considerado como persona activa o no activa.

2.3.2. Requerimientos no funcionales

Los requerimientos no funcionales detectados en el enunciado son los siguientes:

- Disponer de las Comunidades Autónomas la información sobre prestaciones, retenciones realizadas y número de personas perceptoras de cada prestación.
- Disponer del INE los datos de población y personas en activo por Comunidad Autónoma y año.
- Dar respuesta a los diferentes formatos de información entregados por las Comunidades Autónomas, tanto por el carácter de separación como de la estructura de la información entregada.
- Tratamiento correcto de errores en proceso de carga
- Requerimientos generales de cualquier Almacén de Datos:
  - Seguridad de acceso

- Tiempo de respuesta de las consultas
- Usabilidad
- Facilidad en el proceso de actualización

2.4 Modelo Conceptual

Como se puede comprobar en el diagrama siguiente se han identificado tres tipos de perfil de usuarios del Almacén de Datos. El perfil Proveedor de información que en este proyecto serán las Comunidades Autónomas y el INE, el perfil administrador que se encargará de ejecutar el proceso ETL y el mantenimiento del sistema y el perfil usuario que se encargará de ejecutar las consultas con el fin de estudiar la información generada por el Almacén de Datos.

A continuación se muestra el diagrama de casos de uso del proyecto.

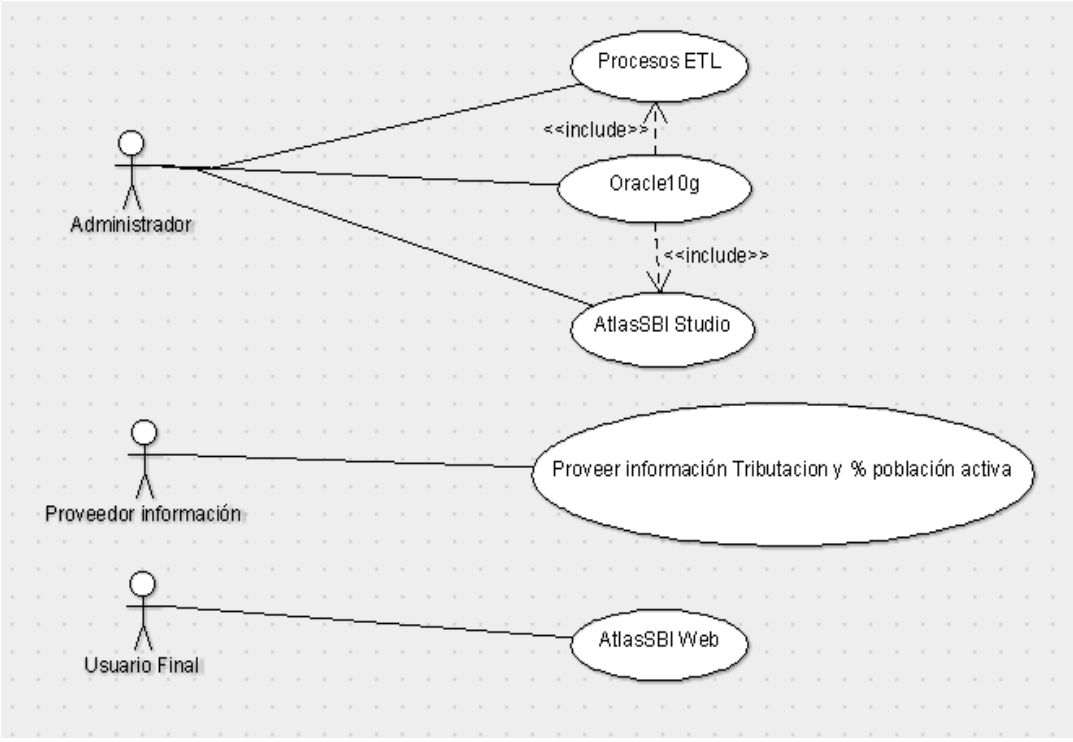


Figura 7 Diagrama de caso de uso

El diagrama de actividades correspondiente al proyecto es el siguiente:

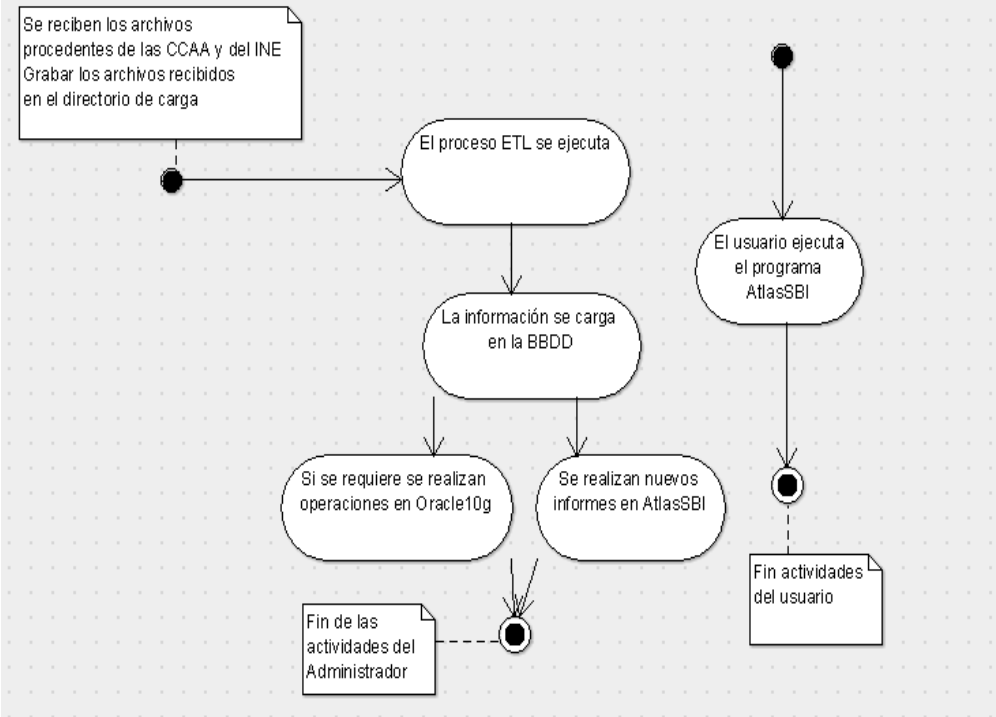


Figura 8 Diagrama de actividades

2.5 Diseño de la Base de Datos. Diagrama ER

El modelo conceptual es un diagrama de constelación de hechos con dos tablas de hechos (Tributación y Personas) y 4 tablas de dimensión (Tiempo, Comunidad_Autonoma, Segmento y TipoRetribucion). En el próximo apartado se realiza un análisis detallado del modelo.

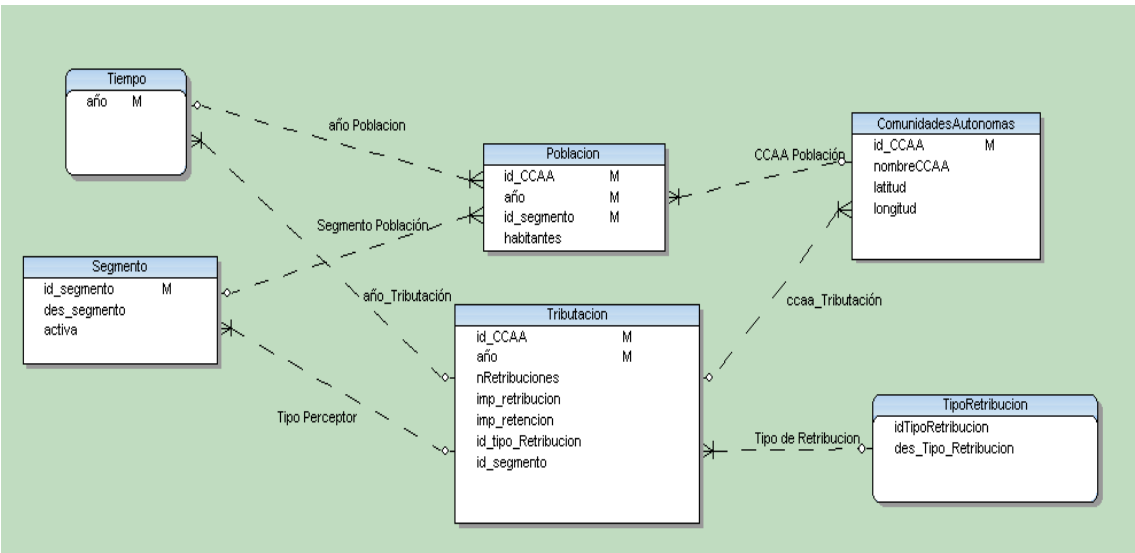


Figura 9 Diagrama de la Base de Datos

A continuación se describen las tablas del modelo de datos. En las tablas siguientes la columna “Tipo” se refiere al tipo de dato. La columna “Long” es la longitud del campo y la columna “Dec”, indica los decimales en los campos numéricos. Finalmente en la columna “Clave” se identifica si el campo es clave principal se identifica con el código PK y si es clave foránea con el código FK.

Tabla de Hecho Tributaciones

Campo	Clave	Tipo	Long	Dec
id_CCAA	FK	INT	6	
Año	FK	INT	4	
id_tipo_Retribucion	FK	INT	2	
id_segmento	FK	INT	2	
nRetribuciones		INT	10	
imp_retribucion		FLOAT	15	2
imp_retencion		FLOAT	15	2
imp_retencion		FLOAT	15	2

Tabla de Hecho Poblacion

Campo	Clave	Tipo	Long	Dec
id_CCAA	FK	INT	2	
año	FK	INT	4	
id_segmento	FK	INT	2	
habitantes		INT	10	

Tabla de Dimensión Comunidades_Autonomas

Campo	Clave	Tipo	Long	Dec
id_CCAA	PK	INT	2	
nombreCCAA		VARCHAR	30	
latitud		INT	20	15
longitud		INT	20	15

Tabla de Dimensión Tipo_Retribucion

Campo	Clave	Tipo	Long	Dec
id_TipoRetribucion	PK	INT	2	
des_Tipo_Retribucion		VARCHAR	30	

Tabla de Dimensión Tiempo

Campo	Clave	Tipo	Long	Dec

año	PK	INT	4	

Tabla de Dimensión Segmento

Campo	Clave	Tipo	Long	Dec
id_segmento	PK	INT	2	
des_segmento		VARCHAR	30	
activa		Boolean	1	

Figura 10 Tablas BBDD

2.6 Modelo multidimensional detallado

El diseño del modelo multidimensional elegido para este proyecto es el de un modelo de constelación de hechos. Esta decisión se debe a que se han detectado dos tipos de información muy distintos, por un lado las tributaciones (carácter económico) y por el otro lado la población (carácter sociológico). Esta decisión se desarrolla en el próximo punto.

2.6.1 Identificación de los hechos

Los hechos son las variables de negocio sobre los que se va a totalizar, promediar y en general realizar operaciones de agregación que conduzcan a conclusiones sobre la evolución que se estudia. La denominación estadística de dichas variables de negocio sería la de variables cuantitativas.

Se debe identificar los hechos que constituye el núcleo de la constelación de hechos que se pretende diseñar. Se analizan los procesos del nivel de negocio que se están tratando y se intenta buscar los procesos individuales que en si mismo sean lo suficientemente significativos.

Como se ha indicado en la introducción de este punto en este proyecto se han identificado dos tipos de información distintos, por un lado datos de carácter económico como son las tributaciones realizadas a lo largo de los años para cada Comunidad Autónoma y por otro lado la distribución de la población.

Las ventajas de haber tomado esta decisión son las siguientes:

- Normalización de la base de datos utilizando una tabla para cada tipo de información

- Facilidad del proceso ETL en la carga de información de cada tabla de hechos por provenir cada tabla de archivos distintos.
- Facilidad en la realización de las consultas SQL
- No hay suficiente información en las tablas de hechos que haga que las consultas de SQL se ralenticen a causa de las uniones entre tablas.

- Corrige la dificultad de calcular en consultas SQL el número de habitantes para cada tipo de retribución si la información la tuviéramos repetida en varios registros de la misma tabla de hechos.

A continuación se detallan las métricas o indicadores de cada una de las tablas de hechos.

- **Tributaciones**
  - **nretribuciones**: Número de retribuciones realizadas para cada combinación de Comunidad Autónoma, año, tipo de perceptor y tipo de retribución.
  - **imp_retribucion:** Importe total retribuido para la misma combinación explicada en la métrica anterior.
  - **imp_retencion:** Importe total retenido en combinación explicada en la métrica nretribuciones.

- **Habitantes**
  - **habitantes:** Número de habitantes para la combinación Comunidad Autónoma, año y tipo de perceptor (segmento de población).

2.6.2 Identificación de las dimensiones y sus atributos

Una dimensión se compone de una serie de atributos que podrían estar organizados jerárquicamente. Estos atributos permiten analizar las medidas de los hechos a diferente nivel de detalle según se agreguen o desagreguen los datos.

En nuestro caso obtenemos cuatro dimensiones distintas (tiempo, Comunidad Autónoma, segmento y tipo de retribución). A continuación se detallan cada una ellas.

- **Tiempo**
    - año. Como el propio nombre indica este campo hace referencia al año en el que se quiera realizar el estudio.
- **Comunidad Autónoma**
    - Id_CCAA. Clave principal de la tabla que servirá para realizar la unión con las tablas de hechos.
    - nombreCCAA. Hace referencia al nombre de la Comunidad Autónoma o Ciudad Autónoma.
    - latitud. Latitud de la localización de la Comunidad Autónoma, necesario para realizar gráficos con Google Maps.
    - longitud. Longitud de la localización de la Comunidad Autónoma, necesario para realizar gráficos con Google Maps.

El dato de la latitud y longitud será tomado de la siguiente web:

http://www.businessintelligence.info/varios/longitud-latitud-pueblos-espana.html

- **Segmento.**
    - id_segmento. Clave principal de la tabla que servirá para realizar la unión con las tablas de hechos.
    - Des_segmento. Nombre del segmento de población a estudiar. Indicar que se han detectado 2 segmentos más de los indicados en el enunciado, ya que pueden existir personas activas que no tengan ninguna remuneración y personas no activas que del mismo modo no tengan remuneración.
    - activa. Indicará para cada segmento si pertenece a población activa o no. Este campo es necesario para realizar las consultas necesarias para cubrir las necesidades del Organismo.

Es necesario aclarar que pueden existir personas que pertenezcan a varios segmentos a la vez.

- **TipoRetribución.**
    - idTipoRetribucion. Clave principal de la tabla que servirá para realizar la unión con las tablas de hechos.
    - desc_Tipo_Retribucion. Descripción del tipo de retribución obtenida.

Finalmente indicar que en este proyecto no se han detectado dimensiones con nivel de jerarquías, en posibles mejoras del sistema se podrían añadir jerarquías en la dimensión de CCAA añadiendo las provincias y poblaciones y en la dimensión de año se podrían añadir meses, trimestres etc.

2.6.3 Granularidad Adecuada

En esta fase se trata de fijar cual ha de ser el grado de detalle de las celdas que compondrán el cubo.

Es necesario buscar un equilibrio entre el detalle y la carga del sistema.

Las tributaciones serán registradas por periodos anuales para cada Comunidad Autónoma, año, tipo de perceptor (segmento) y tipo de retribución.

La previsión de carga inicial máxima de información para la tabla de hechos **Tributación** será la siguiente:

17 (CCAA y Ciudades Autónomas) X 3 (tipos de retribución) X 9 (máximo de años) X 7 tipos de perceptores (segmentos)=3213 registros.

La previsión de carga de información para la tabla de hechos **Población** será la siguiente:

19(CCAA y Ciudades Autónomas, se incluye País Vasco y Navarra) X 3 (tipos de retribución) X 10 (máximo de años) =570 registros.

2.7 Diagrama de arquitectura de software

En la figura que aparece a continuación se muestra los logos de las herramientas de software utilizadas y propuestas para el diseño y explotación del proyecto.

Figura 11 Diagrama de arquitectura de software

Si seguimos el sentido de las flechas de izquierda a derecha, empezamos con la obtención de la fuente de datos propietarias, proporcionada en formato Excel por parte del cliente. Dicha información será tratada por una aplicación Java y procedimientos almacenados PL/SQL. Una vez registrada la información en las diferentes tablas en una base de datos de tipo Oracle10g, el usuario final dispondrá de los informes solicitados a través de la herramienta AtlasSBI.

2.8 Diagrama de arquitectura de hardware

En la imagen se muestra el esquema de la arquitectura de hardware correspondiente al diseño y explotación del proyecto.

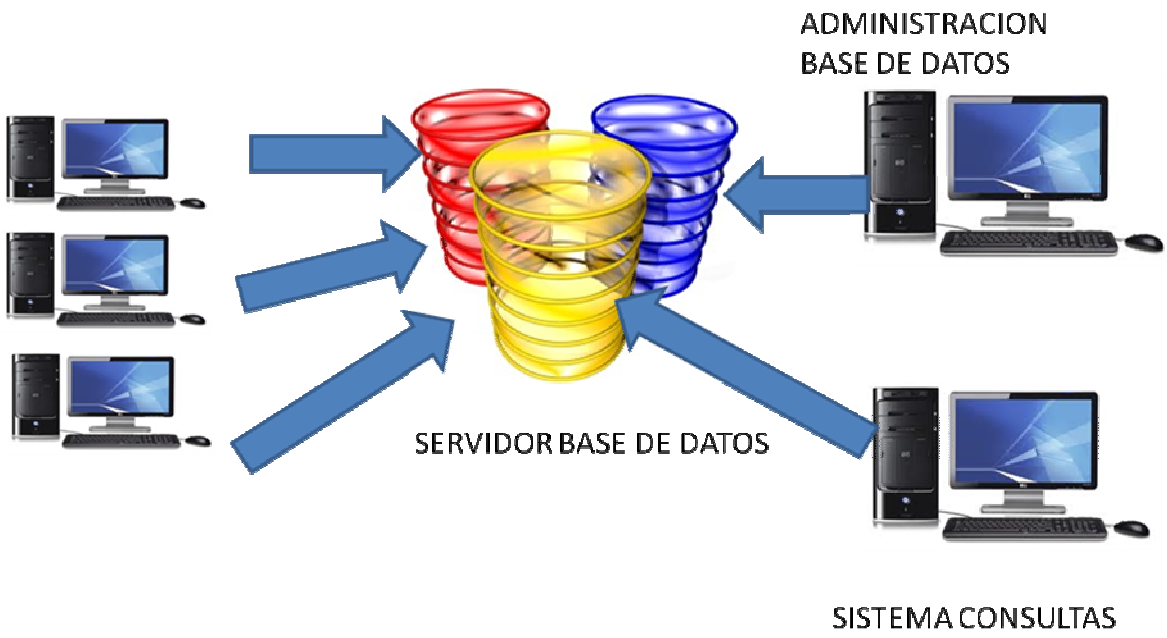


Figura 12 Diagrama de arquitectura de hardware

Cada delegación de la empresa registrará en el servidor de la base de datos la información necesaria para el buen funcionamiento del sistema. Desde el sistema de administración se realizarán los procesos de carga y mantenimiento del almacén de datos además de la gestión de los informes. Finalmente el usuario final a través de su sistema de consultas tendrá acceso a la información en forma de reportes analíticos.

3. PROCESO ETL

3.1 Introducción

Los procesos ETL (*Extraction,Transformation, and Loading*, extracción, transformación y carga) son los encargados de transportar los datos que contienen las fuentes de datos al almacén de datos asegurando la calidad de los datos que serán luego analizados mediante las herramientas de explotación.

Se va a preparar el proceso de carga de información del almacén de datos ya sea porque es la primera vez que se realizada o para regenerarlo en caso de necesidad. En este proyecto no se va a realizar un proceso de carga para el mantenimiento evolutivo ya que Organismo no lo ha solicitado, dejándolo para posibles mejoras futuras.

A continuación se muestra un esquema donde se pueden distinguir las tareas principales.

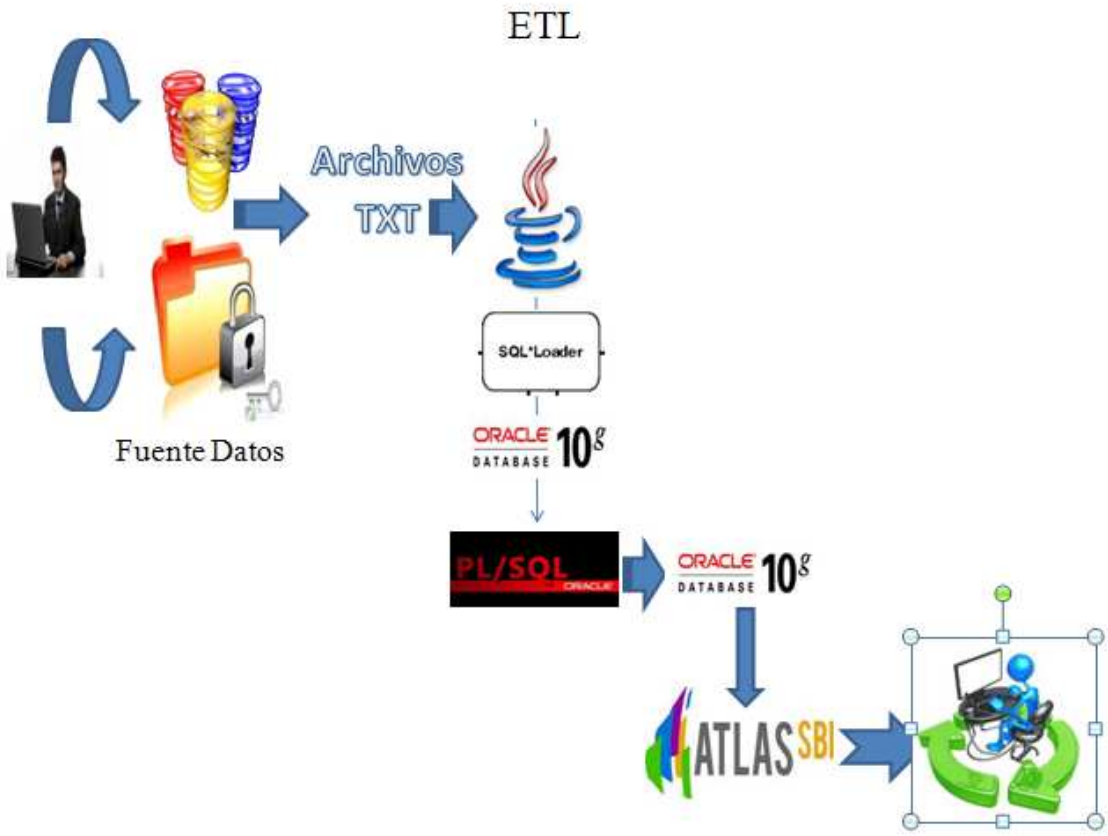


Figura 11 Esquema proceso ETL

3.2 Suposiciones

A lo largo del proyecto se han debido de realizar una serie de suposiciones con el objetivo de tomar la mejor decisión en función de las necesidades del cliente.

A continuación se procede a argumentar cada una de ellas:

- Población Activa. Se entiende como población activa a las personas que tienen edad laboral están buscando empleo. En nuestro caso serían los siguientes segmentos:
  - ASALARIADOS
  - ASALARIADOS-DESEMPLEADOS
  - ASALARIADOS-PENSIONISTAS
  - ASALARIADOS-PENSIONISTAS-DESEMPLEADOS
  - PENSIONISTA-DESEMPLEADO
  - ACTIVA SIN RETRIBUCION

- Población Pasiva. Personas no incluidas dentro de Población Activa. En nuestro caso serían los segmentos:
  - PENSIONISTA
  - NOACTIVA SIN RETRIBUCION

- Perceptores no activos. Personas que reciben una prestación y están dentro de la población pasiva En nuestro caso sería el segmento.
  - PENSIONISTA

- Perceptores activos. Personas que reciben una prestación y están dentro de la población activa En nuestro caso sería el segmento.
  - ASALARIADOS-DESEMPLEADOS
  - ASALARIADOS-PENSIONISTAS
  - ASALARIADOS-PENSIONISTAS-DESEMPLEADOS
  - PENSIONISTA-DESEMPLEADO

- Tasa Paro. Serían las personas desempleadas entre la población activa. Se ha tomado la decisión de incluir a los segmentos donde las personas que están dentro de él hayan estado en desempleo durante el último año. En nuestro caso:

  (DESEMPLEADOS+(ASALARIADOS-DESEMPLEADOS)+(PENSIONISTA-DESEMPLEADO)) / POBLACIÓN ACTIVA

Indicar que en los reportes de proyección el enunciado no deja muy claro que ratios tomar para medir la sostenibilidad del sistema. Se ha decido tomar los siguientes ratios.

- Trabajadores/Perceptores Activos. El sistema tendrá problemas de sostenibilidad cuando dos o menos trabajadores "mantengan" a un perceptor activo (básicamente desempleados).

- Trabajadores/Perceptores. El sistema tendrá problemas de sostenibilidad cuando un trabajador "mantengan" a un perceptor (cualquier persona que reciba una prestación)

3.3 Script de carga

Para que la ejecución por parte del usuario sea lo más sencilla posible se ha creado el archivo de ejecución por lotes ScriptTFC localizado en la siguiente dirección C:\Carga. Dicho script se encargará de ejecutar 4 subprocesos distintos, los cuales se indican a continuación:

- Creación de los archivos con extensión csv definitivos de carga.

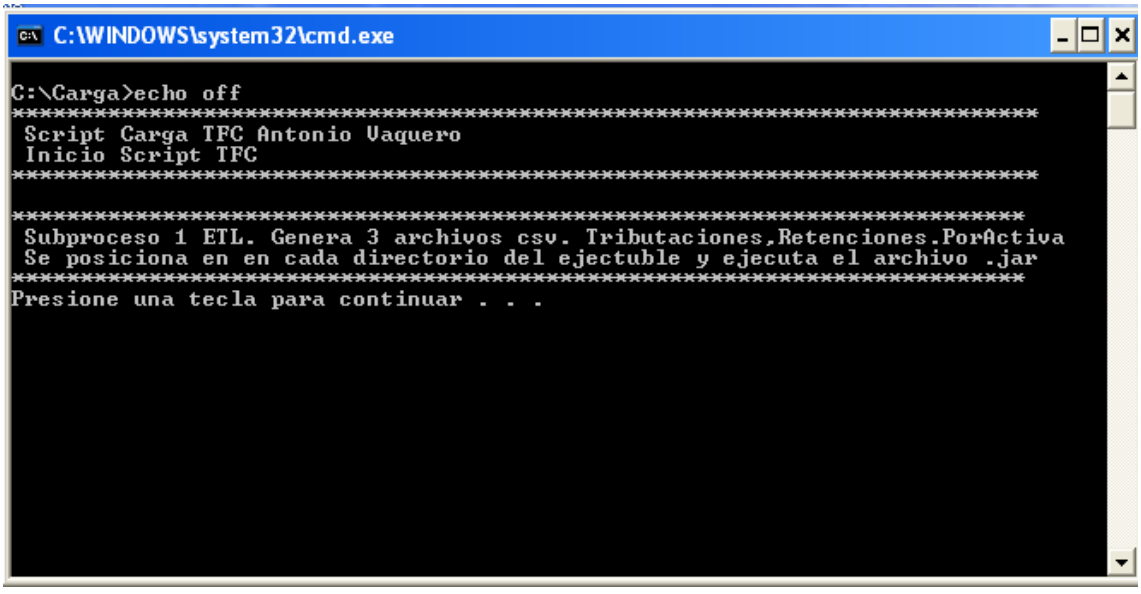


Figura 12 Script carga Subproceso 1

- Borrar la información registrada en las tablas Oracle realizadas en anteriores ejecuciones.

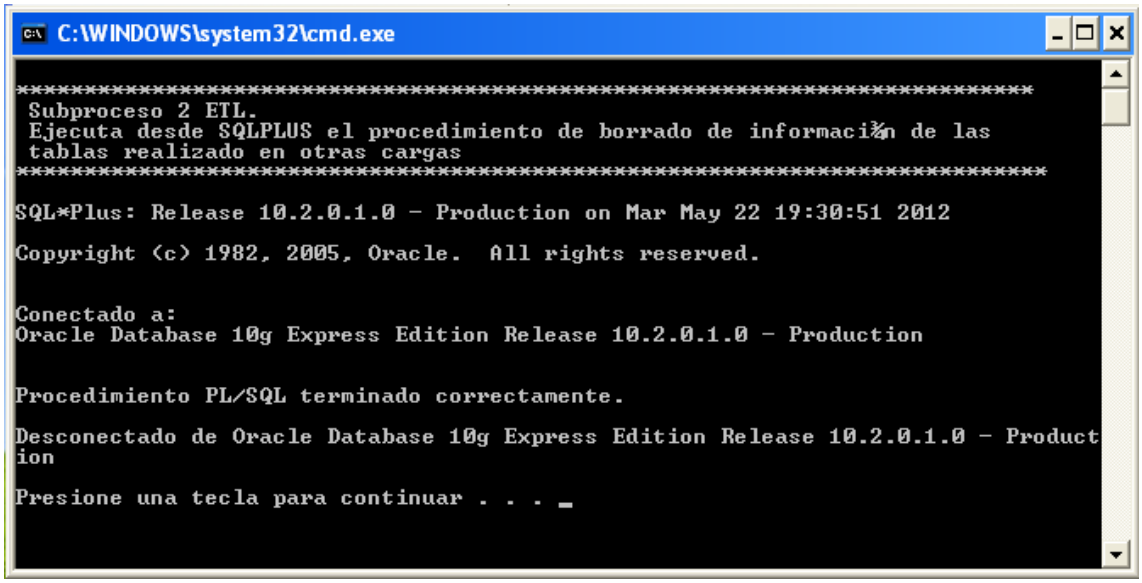


Figura 13 Script carga Subproceso 2

- Carga de la información en las tablas temporales del proyecto.

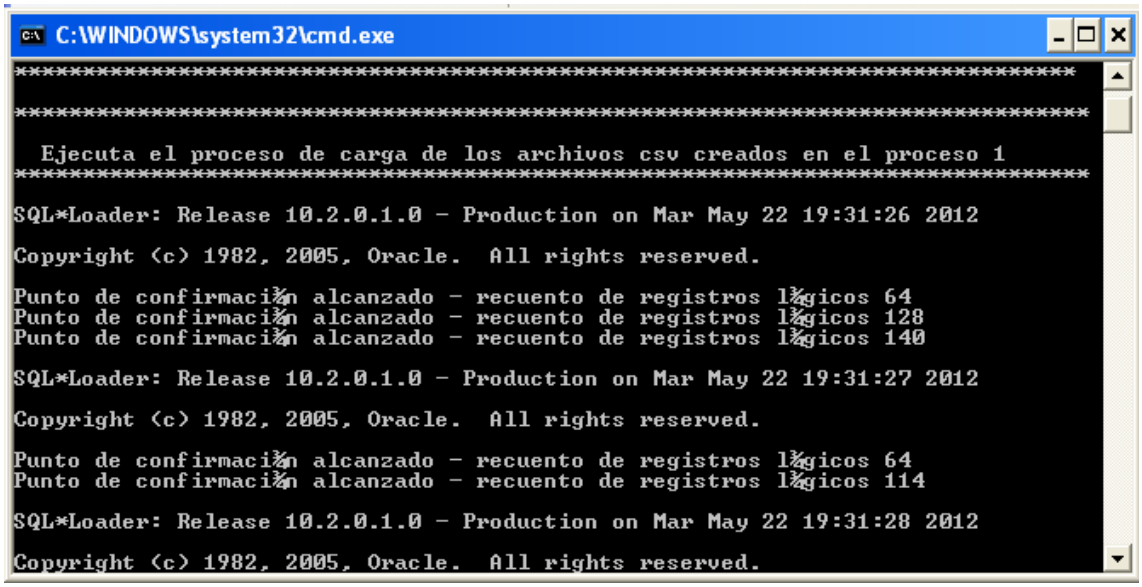


Figura 14 Script carga Subproceso 3.a

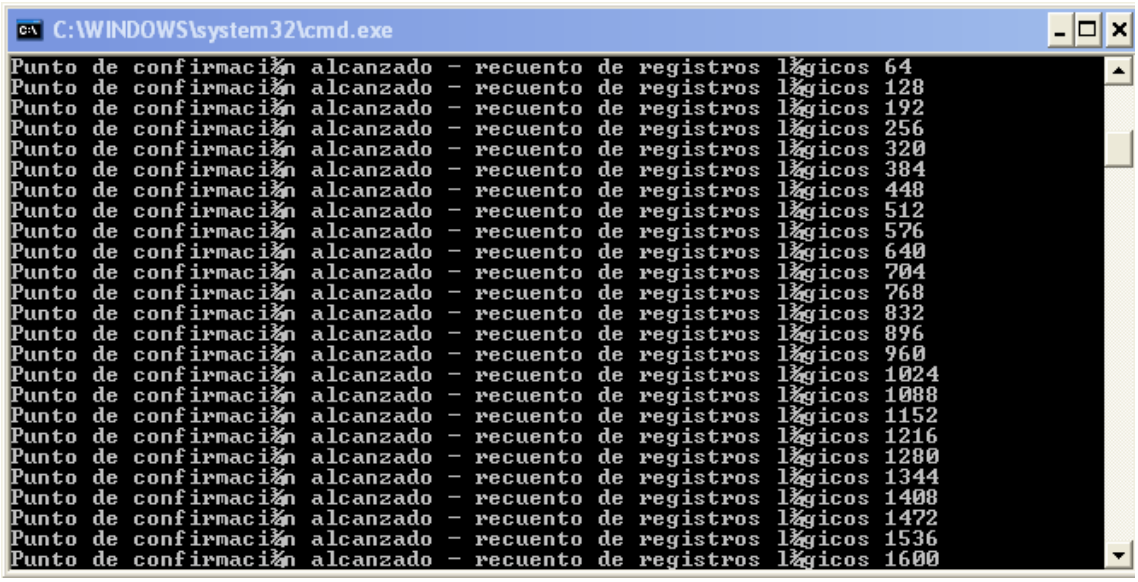


Figura 15 Script carga Subproceso 3.b

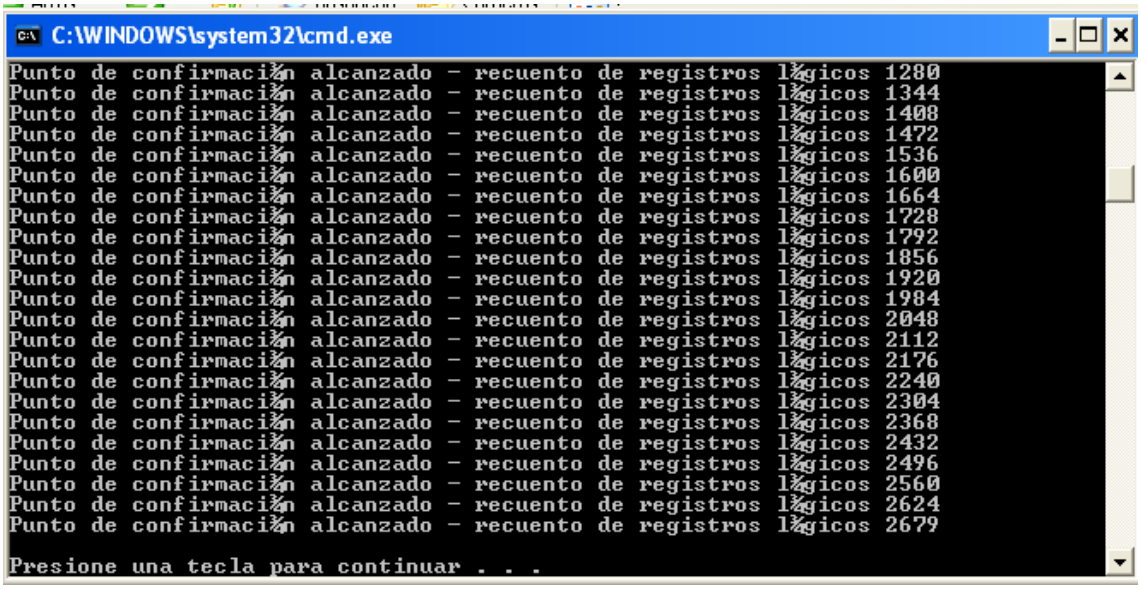


Figura 16 Script carga Subproceso 3.c

- Ejecutar procedimientos almacenados PL/SQL realizados para el tratamiento final de la información y carga en las tablas definitivas.

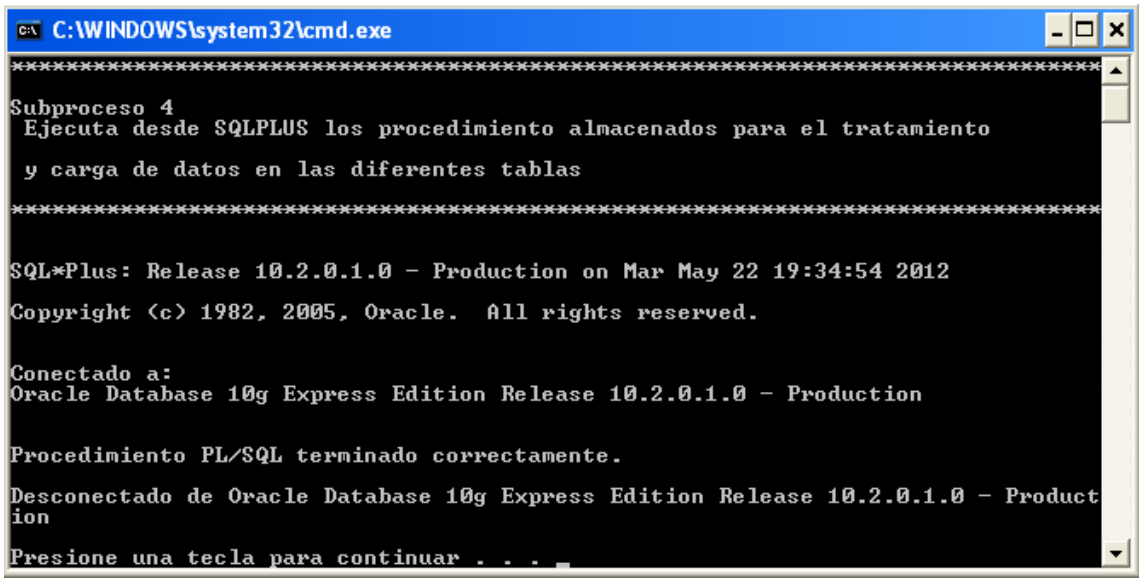


Figura 177 Script carga Subproceso 4

El tiempo aproximado de carga será de menos de 2 minutos.

En los siguientes apartados se detalla el funcionamiento de los diferentes procesos.

3.4 Creación de archivos definitivos de carga

Como se indició en el apartado 2.1 de la PEC2 (Análisis de los datos de origen), la información correspondiente a las tributaciones, población y porcentaje de población activa entregada por las diferentes Comunidades Autónomas y Ciudades Autónomas aunque similar no tienen exactamente el mismo formato, por lo que se han debido de realizar programas de tratamiento que se adecuaran a esta circunstancia.

Se han realizado tres programas en el lenguaje de programación Java, uno para cada tipo de información, con el fin de crear tres archivos definitivos (uno para cada tipo) con la información tratada y unificada que facilitará después la carga definitiva en las diferentes tablas temporales de Oracle.

En los próximos apartados se detallan las operaciones realizadas para conseguir el objetivo.

3.4.1 Tributación

Con el ejecutable tributaciones.jar (C:\Carga\Clases\Tributaciones\dist\), se consigue tratar y unificar la información entregada por las Comunidades Autónomas en diferentes archivos.

El requisito de este programa es que todos los archivos con la información sobre las tributaciones estén en el directorio siguiente C:\Carga\Tributacion

Para cada archivo que aparezca en el directorio anterior se realizarán las siguientes operaciones:

- Abrir el archivo
- Lee la información del archivo con la extensión ISO-8859-1 que hará que los acentos sean entendidos por el programa.
- Detectar cual es el carácter de separación (token). El enunciado ya nos avisa que en cada archivo puede existir un signo de separación distinto.
- Calcular el número de filas y columnas del archivo con el fin de dimensionar correctamente la matriz donde se cargará los datos que servirá para obtener la información necesaria.
- Creación de la matriz.
- Lee los datos fila a fila del archivo y los carga en la matriz, previa corrección de los siguientes datos.
    - Quitar los espacios en blanco que pudieran aparecer por delante o por detrás del dato.
    - Si el dato actual es “..” la sustituye por un 0.
    - Pasar el dato decimal entregado a String para la carga en la matriz.
    - En el caso que el importe de tributación y el importe de retención no se aporte se deberá de calcular de la siguiente forma:

      Retribuciones= Retribución media anual * Personas

      Retenciones= Tipo medio de retención * Personas
    - Calcular el número de segmentos aportado en el archivo.
    - Calcular el número de años aportado en el archivo.

- Calcular el número de filas con información aportado en cada tipo de retribución.
- Obtención de la información a cargar en el archivo final dependiendo de los tres apartados anteriores.
- Unificar el nombre de la Comunidad Autónoma
- Sustituir el caracter "." por "," para el tratamiento de los decimales en SQLLOADER.
- Unificar el nombre del tipo de receptor.
- Añadir en el archivo final Tributaciones.csv (C:\Carga\SQLLOADER\) la información registrada en la matriz.

3.4.2 Población

Con el ejecutable tributaciones.jar (C:\Carga\Clases\Poblacion\dist\), se consigue tratar y unificar la información entregada por el INE en diferentes archivos.

El requisito de este programa es que todos los archivos con la información sobre las tributaciones sean cargados en el directorio siguiente C:\Carga\Poblacion

Para cada archivo que aparezca en el directorio anterior se realizarán las siguientes operaciones:

- Abrir el archivo
- Lee la información del archivo con la extensión ISO-8859-1 que hará que los acentos sean entendidos por el programa.
- Detectar cual es el carácter de separación (token). El enunciado ya nos avisa que en cada archivo puede existir un signo de separación distinto.
- Calcular el número de filas y columnas del archivo con el fin de dimensionar correctamente la matriz donde se cargará la información y que ayudará a la obtención de la información necesaria.
- Creación de la matriz.
- Lee los datos fila a fila y los carga en la matriz, previa corrección de los siguientes datos.

- Quitar los espacios en blanco que pudieran aparecer por delante o por detrás del dato.
- Pasar el dato decimal entregado a String para la carga en la matriz.
- Obtiene el año del nombre del archivo.
- Obtener el número de personas sumando el valor dado en varones y mujeres.
- Unificar el nombre de la Comunidad Autónoma
- Sustituir el caracter "." por "," para el tratamiento de los decimales en SQLLOADER.
- Añadir en el archivo final Poblacion.csv (C:\Carga\SQLLOADER\) la información registrada en la matriz.

3.4.3 Porcpoblacionactiva

Con el ejecutable tributaciones.jar (C:\Carga\Clases\Porcpoblacionactiva\dist\), se consigue tratar y unificar la información entregada en el archivo Porc Poblacion Activa por el INE .

El requisito de este programa es que el archivo Porc Poblacion Activa se encuentre en el directorio siguiente C:\Carga\PorcentajePoblActiva

En el archivo se realizarán las siguientes operaciones de tratamiento.

- Abrir el archivo
- Lee la información del archivo con la extensión ISO-8859-1 que hará que los acentos sean entendidos por el programa.
- Detectar cual es el carácter de separación (token). El enunciado ya nos avisa que en cada archivo puede existir un signo de separación distinto.
- Calcular el número de filas y columnas del archivo con el fin de dimensionar correctamente la matriz donde se cargará la información y que ayudará a la obtención de la información necesaria.
- Creación de la matriz.
- Lee los datos fila a fila y los carga en la matriz, previa corrección de los siguientes datos.

- Quitar los espacios en blanco que pudieran aparecer por delante o por detrás del dato.
- Pasar el dato decimal entregado a String para la carga en la matriz.
- Unificar el nombre de la Comunidad Autónoma
- Sustituir el caracter "." por "," para el tratamiento de los decimales en SQLLOADER.
- Añadir en el archivo final PorcentajePoblActiva.csv (C:\Carga\SQLLOADER\) la información registrada en la matriz.

3.5 Borrar información

Si queremos realizar una nueva carga de información en el almacén de datos creado en Oracle, es necesario borrar toda la información que pudiera existir en las diferentes tablas.

Para realizar esta operación se ha creado el ejecutable sql BorradoSQL, que realiza la operación de borrado de las diferentes tablas gracias al procedimiento almacenado borrartablas creado en Oracle.

3.6 Carga información

En este proceso se realizarán las operaciones necesarias para la carga de información en las diferentes tablas temporales creadas en Oracle.

La utilidad de Oracle sqlloader nos permite registrar en tablas de Oracle la información procedente de un archivo con extensión csv.

Para el proyecto necesitamos cargar los tres archivos con extensión csv creados en el subproceso 1.

En el directorio C:\Carga\SQLLOADER tenemos los 3 programas con extensión CTL que se encargarán de cargar la información en las diferentes tablas.

Los programas son los siguientes:

- TributacionesCTL (carga el archivo Tributaciones.csv)
- PorcPoblacionActivaCTL (carga el archivo PorcentajePoblActiva.csv)
- PoblacionCTL (carga el archivo Poblacion.csv)

Cada vez que se ejecuta el proceso se genera un archivo con el mismo nombre pero con extensión txt, donde nos indica el número de registros cargado y si hay algún error en el proceso.

En el caso de producirse un error se han configurado los ejecutables CTL, para que se cree un archivo con el mismo nombre pero acabado en BAD, donde nos indicará los registros erróneos no cargados en el proceso.

Se ha requerido utilizar la siguiente instrucción (CHARACTERSET WE8ISO8859P1) para que la carga en Oracle de las palabras con acento se realice correctamente.

3.7 Procedimientos almacenados

Para tratar la información registrada ya en las tablas temporales y cargar la información necesaria en las tablas definitivas de Oracle es necesario ejecutar una serie de procedimientos almacenados creados en lenguaje PL/SQL.

A continuación se detallan los procedimientos creados y se explica el objetivo de cada uno de ellos.

- cargaaños. Con este procedimiento se obtiene los años en los cuales se han aportado tanto tributaciones como población. Con ello evitamos que en los informes se pudiera elegir un año para el cual no tuviéramos toda la información necesaria, o bien tributación o bien la población. La información se registrará en la tabla de dimensión años.

- cargaccaa. Registra en la tabla de dimensión comunidades_autonomas, las Comunidades y Ciudades Autónomas de las cuales tengamos información sobre la tributación, evitando así que aparezca en los combos de selección de los reportes las Comunidades Autónomas de País Vasco y Navarra.

- cargatiporetributacion. Procedimiento encargado de registrar en la tabla de dimensión Tipo_Retribucion los diferentes tipos de retribución.

- cargasegmentos. Registra en el tabla de dimensión segmento, los diferentes segmentos que aparecen en los archivos de tributación y además añade los segmentos ACTIVA SIN RETRIBUCIÓN y NO ACTIVA SIN RETRIBUCIÓN.

- poblacionmedia. En el enunciado se indica que se calculen el número de habitantes como media de los valores entre dos años consecutivos. Este procedimiento es el encargado de realizar esta operación actualizando la información de la tabla temporal Personas_tmp. La nueva información se

registra en otra tabla temporal que se ha llamado Personal_tmp2. Indicar que siempre se deja el último año de cada Comunidad con la población sin corregir.

- cargapoblacion. Este procedimiento será el encargado de cargar en la tabla de hechos Personas la población correspondiente a la siguiente combinación (Comunidad Autónoma-año-segmento). La información a cargar se obtiene de la tabla temporal Tributaciones_tmp. En dicha tabla aparece para cada (Comunidad Autónoma-año-tipo de retribución-segmento) el número de retribuciones. Si el número de retribuciones no estuviera repetido en los segmentos que agrupan a dos o tres tipos de retribución distintos el número de retribuciones correspondería al número de habitantes, para solucionarlo lo que se ha hecho es sumar el número de retribuciones de dicha tabla para la combinación (Comunidad Autónoma-año-segmento) y después se ha divido dicha suma entre 2 o 3 (dependiendo del segmento) obteniendo así el número de personas perceptoras reales de esta combinación).Finalmente también se crea el registro en la tabla de hechos Personas con los segmentos ACTIVA SIN RETRIBUCIÓN y NO ACTIVA SIN RETRIBUCIÓN, indicando en este paso el número de habitantes a 0, ya que este número se calculará en los procedimientos siguientes.

- cargapoblacionactivanoret. Este procedimiento será el encargado de calcular el número de habitantes que aún estando dentro de la población activa de cada Comunidad Autónoma y año, no perciben ningún tipo de retribución, como podrían ser parados de larga duración, amas de casa etc. En primer lugar se debe de obtener el número de habitantes que conformar la población activa para el año y Comunidad Autónoma que se trate. El dato se obtiene multiplicando el porcentaje de población activa de la Comunidad Autónoma-año (tabla Porcpoblacionactiva) por el número de habitantes de la misma Comunidad Autónoma y año (tabla Poblacion_tmp2). El numero resultante se le resta la suma de habitantes calculados ya (procedimiento anterior) para los diferentes segmentos de población activa (de la misma Comunidad Autónoma y año).El resultado final será el número de habitantes del segmento ACTIVA SIN RETRIBUCIÓN.

- cargapoblacionnoactivanoret. Este procedimiento será el encargado de calcular el número de habitantes que aún estando dentro de la población pasiva (no activa) de cada Comunidad Autónoma y año, no perciben ningún tipo de retribución, como podrían ser jubilados sin retribución ,niños, religiosos etc. En primer lugar se debe de obtener el número de habitantes que conformar la población pasiva para el año y Comunidad Autónoma que se trate. El dato se obtiene multiplicando la diferencia entre 100 y el porcentaje de población activa de la Comunidad Autónoma-año (tabla Porcpoblacionactiva) por el número de habitantes de la misma Comunidad

Autónoma y año (tabla Poblacion_tmp2). El numero resultante se le resta la suma de habitantes calculados ya (procedimiento anterior) para los diferentes segmentos de población pasiva (de la misma Comunidad Autónoma y año).El resultado final será el número de habitantes del segmento NOACTIVA SIN RETRIBUCIÓN.

- actimportes. Este es el procedimiento que se ha creado para solucionar el problema encontrado en los datos entregados por las Comunidades Autónomas a partir del año 2006, en el cual los importes de retribución y retención aparecen divididos entre mil. He optado a realizar este procedimiento y corregir el problema ya que el ratio de retribución media se veía muy afectado.

4 DOCUMENTOS ANALÍTICOS

Los reportes se han agrupado en un único *Dashboard* donde los usuarios podrán acceder a la información solicitada de una forma muy cómoda. El objetivo buscado para la presentación de los informes ha sido la aunar sencillez y vistosidad gráfica.

A continuación se detallan cada uno de los documentos analíticos realizados.

4.1 Ratios

En la pestaña 'Ratios' se han realizado dos informes analíticos que muestran al cliente los requisitos de información que ha solicitado.

La información se puede consultar de forma agregada, por Comunidad Autónomas, tipo de perceptor y tipo de retribución con una temporalidad de nivel de año.

El primer informe en forma de tabla muestra al usuario la información requerida bajo los criterios de selección de los combos. Como mejora se ha añadido la información del año anterior, el porcentaje de diferencia entre los dos años y una flecha que nos indica de forma gráfica la evolución del ratio respecto al año anterior.

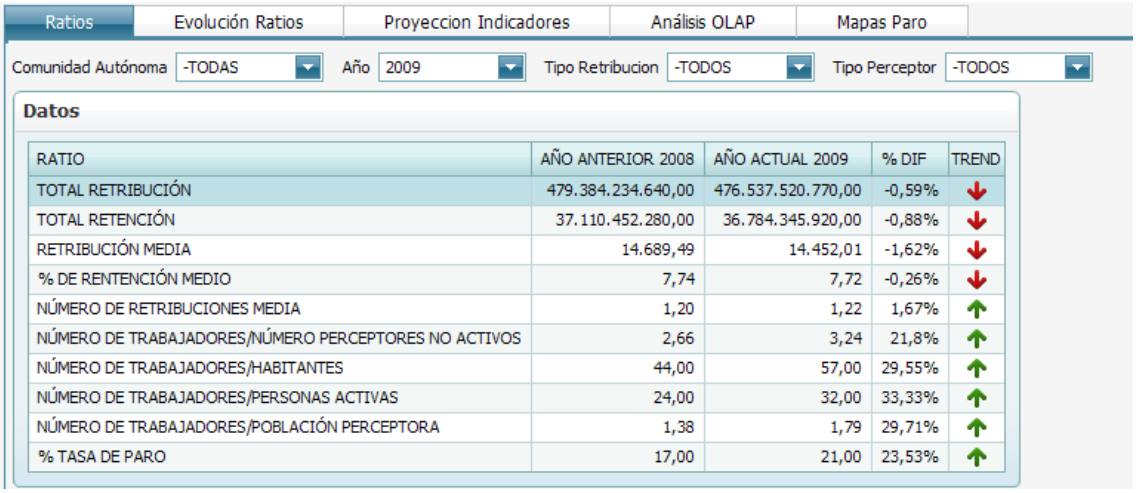
Los ratios que muestra la tabla son los siguientes:

- Total retribución.
- Total retención.
- Retribución media.
- % de retención medio.

- Número de retribuciones medias
- % de población por segmento.
- Número de Trabajadores/Número de perceptores no activos
- Número de Trabajadores/Habitantes.
- Número de Trabajadores/Número de personas activas
- Número de Trabajadores/Población Perceptora
- Tasa de Paro

En la segunda parte del reporte se muestra en formato gráfico de tartas la información solicitada sobre el porcentaje de población por segmento.

A continuación se muestran capturas donde podemos ver el resultado final de lo explicado en los párrafos anteriores.



RATIO	AÑO ANTERIOR 2008	AÑO ACTUAL 2009	% DIF	TREND
TOTAL RETRIBUCIÓN	479.384.234.640,00	476.537.520.770,00	-0,59%	↓
TOTAL RETENCIÓN	37.110.452.280,00	36.784.345.920,00	-0,88%	↓
RETRIBUCIÓN MEDIA	14.689,49	14.452,01	-1,62%	↓
% DE RENTENCIÓN MEDIO	7,74	7,72	-0,26%	↓
NÚMERO DE RETRIBUCIONES MEDIA	1,20	1,22	1,67%	↑
NÚMERO DE TRABAJADORES/NÚMERO PERCEPTORES NO ACTIVOS	2,66	3,24	21,8%	↑
NÚMERO DE TRABAJADORES/HABITANTES	44,00	57,00	29,55%	↑
NÚMERO DE TRABAJADORES/PERSONAS ACTIVAS	24,00	32,00	33,33%	↑
NÚMERO DE TRABAJADORES/POBLACIÓN PERCEPTORA	1,38	1,79	29,71%	↑
% TASA DE PARO	17,00	21,00	23,53%	↑

Figura 188 Tabla Ratios

Figura 19 Gráfico % Población por Segmento

4.2 Evolución Ratios

La pestaña 'Evolución Ratios' se encuentra un informe analítico que ha sido creado para que el usuario pueda comprobar de forma gráfica cómo han evolucionado a lo largo de los años los principales ratios.

Los ratios elegidos para mostrar su evolución han sido los siguientes:

- Retribuciones (expresado en Millones de €. MM€)
- Retenciones (expresado en Millones de €. MM€)
- Retribución Media
- Retención Media
- % Parados
- Número de Retribuciones Medias

Estos informes se pueden consultar de forma agregada por Comunidad Autónomas, tipo de perceptor y por tipo de retribución.

A continuación se muestran capturas de con información correspondiente a Andalucía.

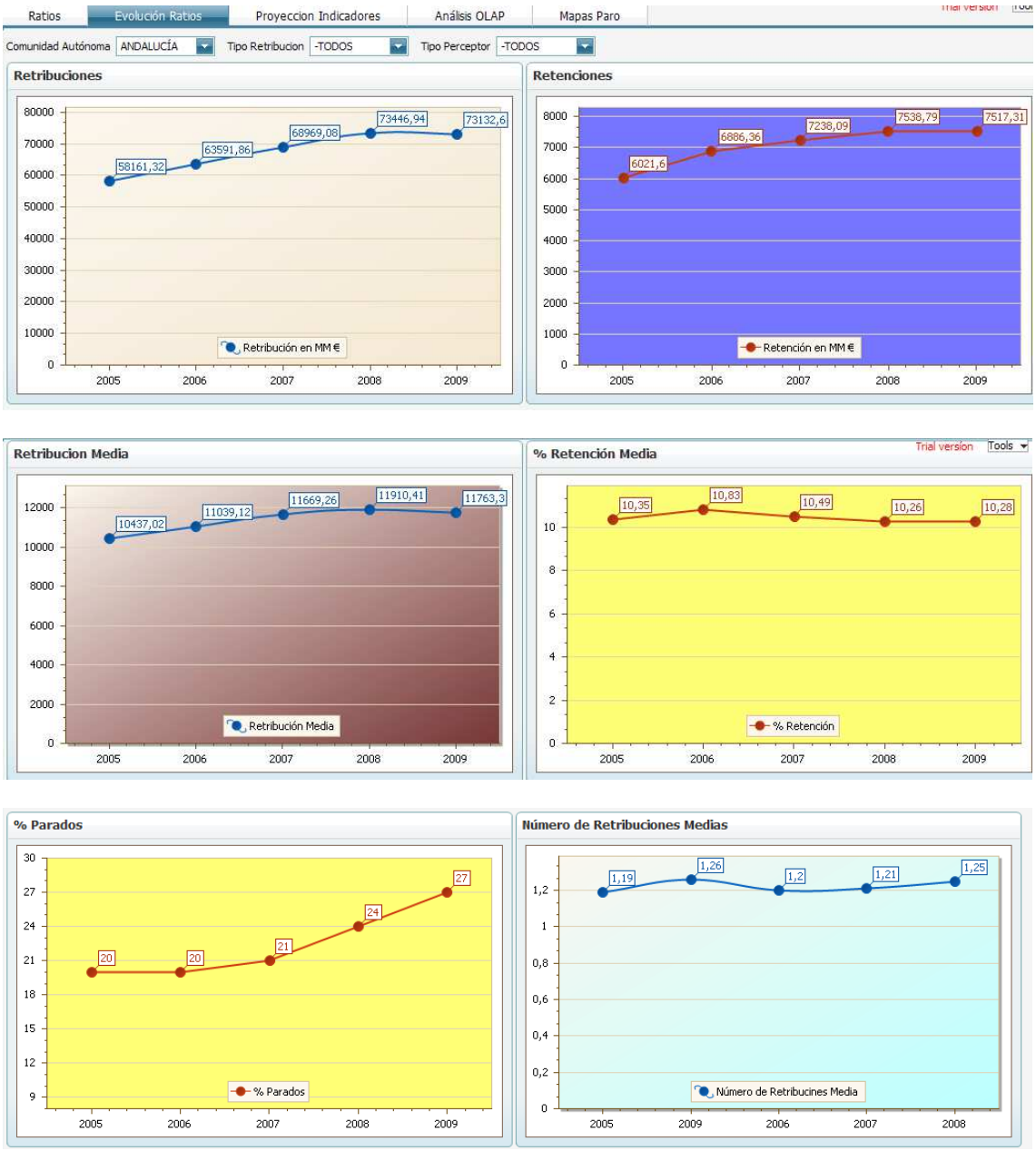


Figura 20 Gráficos de Evolución

4.3 Proyección de Indicadores

En la pestaña 'Proyección Indicadores' se encuentra el informe analítico que responde a la solicitud realizada por Organismo donde se requería una proyección de los indicadores Num Trabajadores/Perceptores activos y Num Trabajadores/Perceptores, para averiguar en qué momento dos trabajadores mantengan un perceptor activo o que se iguale el número de trabajadores y total de perceptores.

Para realizar este proyecto se ha utilizado la técnica estadística de regresión lineal.

A partir del cálculo de la recta de regresión se ha podido estimar el año donde se podría producir el punto de colapso.

Recordar que la fórmula de la recta de regresión sigue la siguiente fórmula:

$$y=mx+b$$

donde :

m es la pendiente de la recta y b es la intersección del eje y.

La pendiente de la recta se ha obtenido con la función de Oracle REGR_SLOPE y el punto de intersección con la función REGR_INTERCEPT.

Una vez obtenida la fórmula, se iguala a 2 para la primera proyección y a 1 para la segunda proyección. Finalmente se despeja la variable x y se obtiene el año donde se estima el "colapso".

Se ha averiguado el coeficiente de determinación R^2 para medir la bondad del ajuste realizado, más exacto cuanto más se acerque a 1 y menos cuanto más se acerque a 0 (relación lineal estocástica). En nuestro caso para la primera proyección R^2 es 0,35 por lo que nos indica que el ajuste tiene poca posibilidad de que se cumpla. La segunda proyección el coeficiente de determinación es 0,85 que nos indica que la recta se ajuste bien a la distribución de los puntos.

Como se puede observar en las figuras siguientes la información disponible se muestra en formato de tabla y de gráfico de evolución. En el cuadro final se muestra el año donde posiblemente se produzca el colapso.

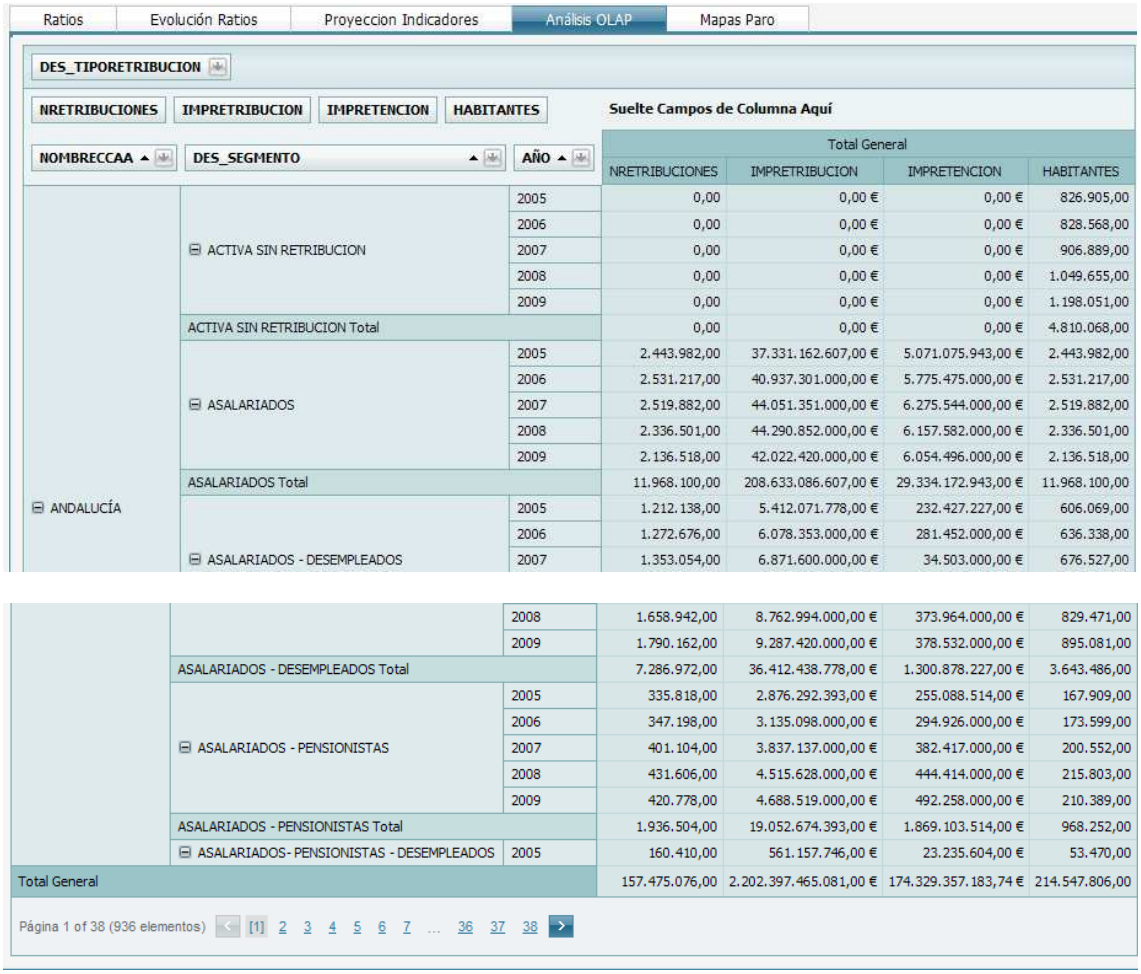
Figura 21 Proyección de Indicadores

4.4 Análisis OLAP

En la pestaña 'Análisis OLAP' se ha añadido un informe analítico en forma de tabla dinámica, donde el usuario tendrá libertad para poder estudiar la información del cubo.

Se ha decidido mostrar el nombre de la Comunidad Autónoma por si se requiere estudiar la información a este nivel, pudiéndose contraer el campo si se requiere pinchando a al flechita que aparece en el nombre del campo.

A continuación se muestra una figura donde podemos ver un ejemplo de tabla.



Ratios | Evolución Ratios | Proyeccion Indicadores | Análisis OLAP | Mapas Paro

DES_TIPORETRIBUCION

NRETRIBUCIONES | IMPRETRIBUCION | IMPRETENCION | HABITANTES — Suelte Campos de Columna Aquí

NOMBRECCAA	DES_SEGMENTO	AÑO	Total General			
			NRETRIBUCIONES	IMPRETRIBUCION	IMPRETENCION	HABITANTES
ANDALUCÍA	ACTIVA SIN RETRIBUCION	2005	0,00	0,00 €	0,00 €	826.905,00
		2006	0,00	0,00 €	0,00 €	828.568,00
		2007	0,00	0,00 €	0,00 €	906.889,00
		2008	0,00	0,00 €	0,00 €	1.049.655,00
		2009	0,00	0,00 €	0,00 €	1.198.051,00
	ACTIVA SIN RETRIBUCION Total		0,00	0,00 €	0,00 €	4.810.068,00
	ASALARIADOS	2005	2.443.982,00	37.331.162.607,00 €	5.071.075.943,00 €	2.443.982,00
		2006	2.531.217,00	40.937.301.000,00 €	5.775.475.000,00 €	2.531.217,00
		2007	2.519.882,00	44.051.351.000,00 €	6.275.544.000,00 €	2.519.882,00
		2008	2.336.501,00	44.290.852.000,00 €	6.157.582.000,00 €	2.336.501,00
		2009	2.136.518,00	42.022.420.000,00 €	6.054.496.000,00 €	2.136.518,00
	ASALARIADOS Total		11.968.100,00	208.633.086.607,00 €	29.334.172.943,00 €	11.968.100,00
	ASALARIADOS - DESEMPLEADOS	2005	1.212.138,00	5.412.071.778,00 €	232.427.227,00 €	606.069,00
		2006	1.272.676,00	6.078.353.000,00 €	281.452.000,00 €	636.338,00
		2007	1.353.054,00	6.871.600.000,00 €	34.503.000,00 €	676.527,00
		2008	1.658.942,00	8.762.994.000,00 €	373.964.000,00 €	829.471,00
		2009	1.790.162,00	9.287.420.000,00 €	378.532.000,00 €	895.081,00
	ASALARIADOS - DESEMPLEADOS Total		7.286.972,00	36.412.438.778,00 €	1.300.878.227,00 €	3.643.486,00
	ASALARIADOS - PENSIONISTAS	2005	335.818,00	2.876.292.393,00 €	255.088.514,00 €	167.909,00
		2006	347.198,00	3.135.098.000,00 €	294.926.000,00 €	173.599,00
		2007	401.104,00	3.837.137.000,00 €	382.417.000,00 €	200.552,00
		2008	431.606,00	4.515.628.000,00 €	444.414.000,00 €	215.803,00
		2009	420.778,00	4.688.519.000,00 €	492.258.000,00 €	210.389,00
	ASALARIADOS - PENSIONISTAS Total		1.936.504,00	19.052.674.393,00 €	1.869.103.514,00 €	968.252,00
	ASALARIADOS- PENSIONISTAS - DESEMPLEADOS	2005	160.410,00	561.157.746,00 €	23.235.604,00 €	53.470,00
Total General			157.475.076,00	2.202.397.465.081,00 €	174.329.357.183,74 €	214.547.806,00

Página 1 of 38 (936 elementos) [1] 2 3 4 5 6 7 ... 36 37 38

Figura 22 Análisis OLAP

4.5 Mapa Paro

En la última pestaña se ha añadido un informe analítico en forma de mapa donde podemos ver la tase de desempleo por Comunidad Autónoma.

La implementación se ha realizado en formato mapa *GoogleMap* donde se muestran en círculos las Comunidades Autónomas.

Los colores de los círculos siguen la siguiente lógica.

- En verde si la tasa de paro está comprendida entre el 0% y el 10 %
- En naranja si la tase supera el 10 % pero es inferior al 15 %
- En rojo si la tasa de paro supera el 15 %.

Indicar que si se posiciona el ratón encima de un círculo se muestra la tasa de paro correspondiente a la Comunidad Autónoma seleccionada.

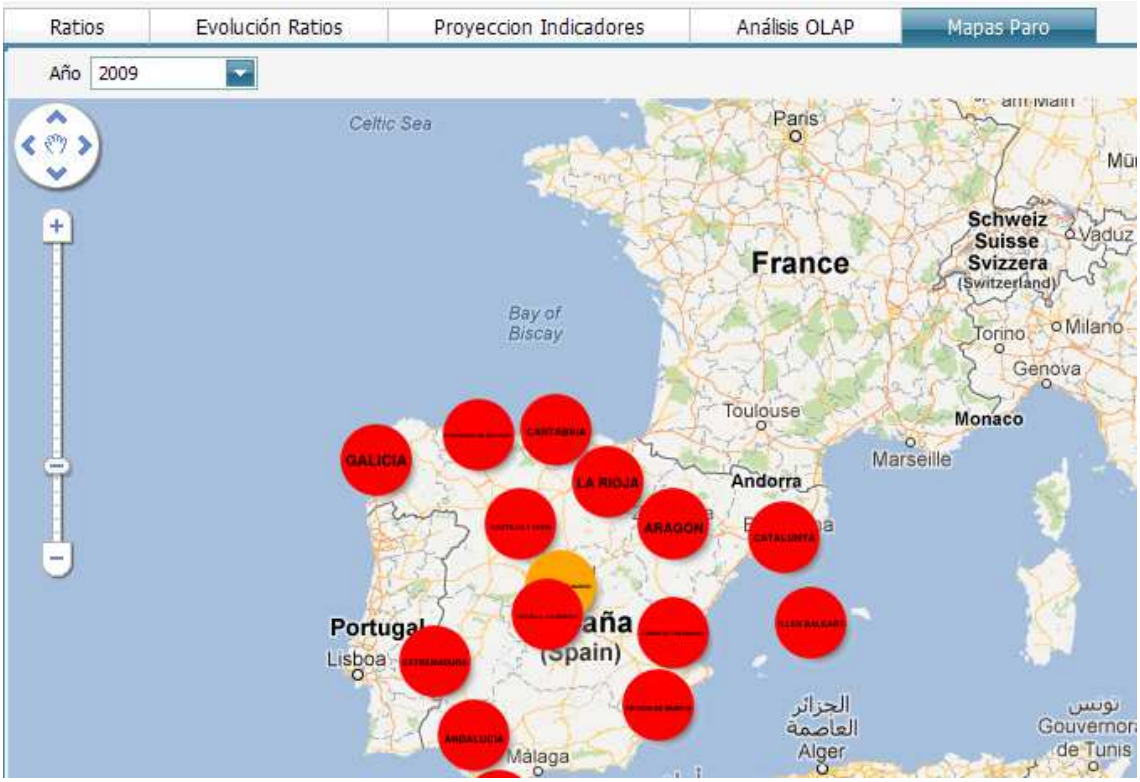


Figura 23 Mapas Paro

5 CONCLUSIONES

Una vez finalizada la exposición de los diferentes procesos seguidos y de los resultados obtenidos en cada fase a continuación expreso mis conclusiones finales.

La introducción en el área de los almacenes de datos ha sido satisfactoria y progresiva durante las PECS realizadas.

Respecto a la necesidad de aprender nuevas herramientas y lenguajes de programación requeridos para la realización del proyecto indicar que se debería de aportar mejores manuales de introducción ya que si tienes varias asignaturas o el tiempo limitado por razones laborales pues tener problemas de cumplimiento de plazos al tener que buscar la información en internet o obtener la información por libros de consulta en los temas relacionados.

Por último indicar que el proyecto ha sido muy interesante y espero que la experiencia adquirida pueda ser útil en la vida laboral.

6 LÍNEAS DE EVOLUCIÓN FUTURA

La evolución futura de la herramienta se podría enfocar en añadir niveles de jerarquía en las tablas de dimensiones, pudiéndose añadir nuevos niveles que hagan el estudio más concreto.

Por ejemplo en la tabla año, se podría estudiar la información a nivel mes, en la tabla Comunidades_Autónomas se podría añadir la provincia.

También creo que podría ser interesante el añadir una tabla de dimensión más a nivel de industria, con lo que se podría estudiar las variaciones producidas en Turismo, Industria, Automoción, Servicios, Construcción etc.

7 GLOSARIO

Almacén de datos: Colección de datos orientada a un determinado ámbito, integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.

AtlasSBI: Es un sistema en entorno web que permite implementar todas las necesidades de información.

Data Warehouse: Ver almacén de datos.

Dimensión: Se entiende por dimensión el eje de análisis (tiempo, producto, cliente, proveedor, etc.) por el que se desea categorizar la información. Si se utilizase una denominación estadística, estos ejes recibirían el nombre de variables cualitativas.

ETL: *Extract, Transform and Load* (Extraer, transformar y cargar en inglés, frecuentemente abreviado a ETL) es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos, data mart, o data warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

Hecho: Los hechos son las variables de negocio (ventas, costes, consumos, tiempos, etc.) sobre los que se va a totalizar, promediar, y en general realizar operaciones de agregación que conduzcan a conclusiones sobre la evolución del área o departamento que se estudie. La denominación estadística de dichas variables de negocio sería la de variables cuantitativas.

OLAP: Es el acrónimo en inglés de procesamiento analítico en línea (On-*Line Analytical Processing*). Es una solución utilizada en el campo de la llamada Inteligencia empresarial (o *Business Intelligence*) cuyo objetivo es agilizar la consulta de grandes cantidades de datos. Para ello utiliza estructuras multidimensionales (o Cubos OLAP) que contienen datos resumidos de

grandes Bases de datos o Sistemas Transaccionales (OLTP). Se usa en informes de negocios de ventas, marketing, informes de dirección, minería de datos y áreas similares.

Oracle: Es un sistema de gestión de base de datos objeto-relacional (o ORDBMS por el acrónimo en inglés de *Object-Relational Data Base Management System*), desarrollado por *Oracle Corporation*.

PL/SQL: (*Procedural Language/Structured Query Language*) es un lenguaje de programación incrustado en Oracle.

SQL Loader: *SQL*Loader (sqlldr)* Utilidad para la carga masiva de datos en base de datos.

8 BIOGRAFIA

La bibliografía consultada para la realización de esta PEC ha sido:

8.1 Publicaciones

- Diseño y Explotación de Almacenes de Datos, Editorial Club Universitario, Juan Carlos Trujillo, José Norberto Mazón y Jesús Pardillo.
- Gestión de proyectos con Project 2007, Anaya Multimedia
- Apuntes sobre Almacenes de Datos, Bases de Datos Multidimensionales y Herramientas OLAP, Bubok, Jesús Pardillo.
- Programación en Oracle 11g SQL, SQL *Plus y PL/SQL. Ra-Ma 2011
- Programador Certificado Java 2 Curso Práctico. Ra-Ma 2007

8.2 Webs

- http://es.wikipedia.org/wiki/
- http://www.java2s.com/Code/Oracle/CatalogOracle.htm
- http://kbase.atlassbi.com/2011/11/creacion-selectores.html
- http://sql.1keydata.com/es/sql-insert-into.php
- http://www.orafaq.com/wiki/SQL*Loader_FAQ
- http://es.classora.com/reports/h143452/ranking-de-las-comunidades-autonomas-de-espana-con-mayor-numero-de-parados-registrados-en-el-inem

9 ANEXOS

9.1 Códigos Java proceso ETL

9.1.1 Tributaciones.java

```
package tributaciones;


/**
 *
 * @author Administrador
 */
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.StringTokenizer;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import com.csvreader.CsvWriter;



public class Tributaciones {


  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) throws FileNotFoundException {
```

```
// Elimina el archivo Tributaciones.csv generado en anteriores cargas
File fichero = new File("C:\\Carga/SQLLOADER/Tributaciones.csv");

if (fichero.delete());
File Rs = new File("C:\\Carga/Tributacion");
File ff[]=Rs.listFiles();
int narchivos=ff.length;
String año="";
String ccaa;
String tRetribucion;
String tReceptor;
double impRetribucion;
double impRetencion;
double retmedanual;
double tmretencion;
double nRetribuciones=0.0;
int nsegmentos;
int nfilasretribucion;
int ños;

// Abre cada uno de los archivos que está en la ruta c:/TFC

    for(int i=0;i<=narchivos-1;i++)
    {

// Crea las variables para el tratamiento de los archivos

        FileReader fr = null;
        FileReader fr2 = null;
        BufferedReader br = null;
```

```
                BufferedReader br2 = null;
                File archivoactual=null;
                String rutaarcactual=ff[i].getAbsolutePath();
                archivoactual = new File (rutaarcactual);

        // Abre el archivo actual

                    try {

                        fr = new FileReader (archivoactual);
                        fr2 = new FileReader (archivoactual);

        // Abre el archivo aceptando los acentos

                        br = new BufferedReader(new InputStreamReader(new FileInputStream(archivoactual),"ISO-8859-1"));
                        br2 = new BufferedReader(new InputStreamReader(new FileInputStream(archivoactual), "ISO-8859-
1"));
                        String linea="";
                        String linea2="";

        // Lee la primera linea
                        linea=br.readLine();
                        linea2=br2.readLine();

        //Detecta cual es el caracter de separación ( token )
                            char token=linea.charAt(0);
                            String tokenactual=Character.toString(token);
        // Parte la linea en partes separadas por el token

                            StringTokenizer st = new StringTokenizer(linea, tokenactual, true);
```

```
// Cálculo del número de filas del archivo para dimensional correctamente la matriz

int lNumeroLineas = 0;
while ((br2.readLine())!=null) {
 lNumeroLineas++;
}
```

// Cálculo del número de particiones de cada linea para dimensional correctamente la matriz, (el caracter de token es una partición

```
int longitud=linea.length();
int ntokens=0;
 for(int f=0;f<longitud;f++) {

    char caracter=linea.charAt(f);

    if (caracter==token) {

      ntokens++;
     }
 }
```

// Crea la matriz que usaremos posteriormente para la obtención de la información necesaria

```
String [][] matriz = new String[lNumeroLineas+1][2*ntokens+1];
```

// Variables para la carga de la información procedente del archivo actual a la matriz

```
String current="";    // current es el valor actual
int fila=0;
int columna=0;
```

/* Carga de la información del archivo actual a la matriz. Detecta si el campo es un caracter de token*/

```
/* Trata si el dato es un token, si es numérico,si el dato es float, si el dato es .. */

for(int j=0;j<=lNumeroLineas;j++) {

 while (linea !=null) {
st = new StringTokenizer(linea, tokenactual, true);
 while (st.hasMoreTokens()) {
    current = st.nextToken();
    current=current.trim();
    current=current.replace("\"","");
    current=current.trim();

     if (current.equals("..")) {
       matriz[fila][columna]="0";
        columna=columna+1;
     }

     else if(!current.equals(tokenactual)) {
          if (current.contains("E+")) {
               current=current.replace(",",".");
               Float f=Float.valueOf(current).floatValue();
               matriz[fila][columna]=current;
               columna=columna+1;
          }
           else {
               matriz[fila][columna]=current;
               columna=columna+1;
           }

     }
```

```
                    else {
                        matriz[fila][columna]=null;
                        columna=columna+1;
                    }

                }
                fila++;
                columna=0;
                linea=br.readLine();

            }
        }

// Obtener la información de la matriz y cargar los datos en las variables

// Calculo del número de segmentos

    if (matriz[1][1].equals("Todos")) {

        nsegmentos=8;

    }

    else {

        nsegmentos=7;

    }

// Calculo del número de años
```

```
                    naños=(ntokens)/nsegmentos;

                // Calculo del número de filas(información) de cada tipo de retribucion ( Se diferencia el que tiene 3 campos de información ya que tiene información distinta ).

                    if (matriz[8][0].equals("Tipo medio de retención")) {

                    nfilasretribucion=5;

                    }

                    else if (matriz[7][0].equals("Retenciones")) {

                    nfilasretribucion=4;

                    }


                    else if (matriz[6][0].equals("Retenciones")) {

                     nfilasretribucion=3;

                    }


                    else {

                        nfilasretribucion=31;

                    }


                // Obtención de los datos de la matriz para cargar las variables

                int ncol=0;


                // Cara cada año

                for(int x=0;x<=naños-1;x++) {


                // Busqueda de la fila donde está el tipo de retribución Salario

                    int nfilasalario=0; /* Número de fila que se encuentra el dato SALARIOS */


                    boolean valor = true;

                    int k=3;

                        while ( valor ) {
```

```
                if (matriz[k][0].equals("SALARIOS")) {
                    nfilasalario=k;
                    break;
                }
                k++;
            }

        // Para cada segemento según el tipo de archivo que sea
        // Para cada tipo de retribución  se obtiene el año,ccaa,el tipo de Retribución, el tipo de Receptor, importe de la retribución, el importe de la renteción.
        // Llama al metodo EscribirArchivo con la información de cada linea.
            for (int y=0;y<=nsegmentos-1;y++) {
             switch(nfilasretribucion)
               {
                  case 3:
                     int nfi=0;
                     ncol=ncol+2;
                     int contador2=1;

                     for(int z=1;z<=3;z++) {

                       int nfilaactual=nfilasalario+1+nfi;
                       año=(matriz[0][(x*(nsegmentos+1))+1]);
                       ccaa=(matriz[2][0]);
                       nRetribuciones=Double.valueOf(matriz[nfilaactual][ncol]).doubleValue();
                       impRetribucion=Double.valueOf(matriz[nfilaactual+1][ncol]).doubleValue();
                       impRetencion=Double.valueOf(matriz[nfilaactual+2][ncol]).doubleValue();

                       tRetribucion=(matriz[nfilaactual-1][0]);
                       tReceptor=(matriz[1][ncol-1]);
```

```
                nfi=nfi+4;

EscribirArchivo(año,ccaa,tRetribucion,tReceptor,nRetribuciones,impRetribucion,impRetencion);

              }
            break;

            case 31:
              nfi=0;
              ncol=ncol+2;
              contador2=1;
               for(int z=1;z<=3;z++) {
                 int nfilaactual=nfilasalario+1+nfi;
                 año=(matriz[0][(x*(nsegmentos+1))+1]);
                 ccaa=(matriz[2][0]);
                 nRetribuciones=Double.valueOf(matriz[nfilaactual][ncol]).doubleValue();
                 retmedanual=Double.valueOf(matriz[nfilaactual+1][ncol]).doubleValue();
                 tmretencion=Double.valueOf(matriz[nfilaactual+2][ncol]).doubleValue();
                 impRetribucion=retmedanual*nRetribuciones;
                 impRetencion=tmretencion*nRetribuciones;

                  if(Integer.parseInt(año)>=2006){
                    /* Se divide entre 1000 porque en un procedimiento almacenado se multiplica por 1000
por error en
                     los datos proporcionados por las CCAA a partir del año 2006*/
                     impRetribucion=impRetribucion/1000;
                     impRetencion=impRetencion/1000;
                  }
                 tRetribucion=(matriz[nfilaactual-1][0]);
                 tReceptor=(matriz[1][ncol-1]);
```

```
                    nfi=nfi+4;

EscribirArchivo(año,ccaa,tRetribucion,tReceptor,nRetribuciones,impRetribucion,impRetencion);
                  }
                break;
                  case 5:
                  nfi=0;
                  ncol=ncol+2;
                  contador2=1;
                  for(int z=1;z<=3;z++) {
                    int nfilaactual=nfilasalario+1+nfi;
                    año=(matriz[0][(x*(nsegmentos+1))+1]);
                    ccaa=(matriz[2][0]);
                    nRetribuciones=Double.valueOf(matriz[nfilaactual][ncol]).doubleValue();
                    impRetribucion=Double.valueOf(matriz[nfilaactual+1][ncol]).doubleValue();
                    impRetencion=Double.valueOf(matriz[nfilaactual+3][ncol]).doubleValue();
                    retmedanual=Double.valueOf(matriz[nfilaactual+2][ncol]).doubleValue();
                    tmretencion=Double.valueOf(matriz[nfilaactual+2][ncol]).doubleValue();
                    tRetribucion=(matriz[nfilaactual-1][0]);
                    tReceptor=(matriz[1][ncol-1]);
                    nfi=nfi+6;

EscribirArchivo(año,ccaa,tRetribucion,tReceptor,nRetribuciones,impRetribucion,impRetencion);
                  }

                break;
                 case 4:
                  nfi=0;
                  ncol=ncol+2;
                  int nfiM=0;
                  int ncolM=ncol+2;
```

```
                    for(int z=1;z<=3;z++) {
                      int nfilaactual=nfilasalario+1+nfi;
                      año=(matriz[0][(x*(nsegmentos+1))+1]);
                      ccaa=(matriz[2][0]);
                      nRetribuciones=Double.valueOf(matriz[nfilaactual][ncol]).doubleValue();
                      impRetribucion=Double.valueOf(matriz[nfilaactual+1][ncol]).doubleValue();
                      impRetencion=Double.valueOf(matriz[nfilaactual+3][ncol]).doubleValue();
                      retmedanual=Double.valueOf(matriz[nfilaactual+2][ncol]).doubleValue();
                      tmretencion=Double.valueOf(matriz[nfilaactual+2][ncol]).doubleValue();
                      tRetribucion=(matriz[nfilaactual-1][0]);
                      tReceptor=(matriz[1][ncol-1]);
                      nfi=nfi+5;

EscribirArchivo(año,ccaa,tRetribucion,tReceptor,nRetribuciones,impRetribucion,impRetencion);
                      año=(matriz[0][(x*(nsegmentos+1))+1]);
                      ccaa=(matriz[18][0]);
                      nRetribuciones=Double.valueOf(matriz[nfilaactual+16][ncol]).doubleValue();
                      impRetribucion=Double.valueOf(matriz[nfilaactual+16+1][ncol]).doubleValue();
                      impRetencion=Double.valueOf(matriz[nfilaactual+16+3][ncol]).doubleValue();
                      retmedanual=Double.valueOf(matriz[nfilaactual+16+2][ncol]).doubleValue();
                      tmretencion=Double.valueOf(matriz[nfilaactual+16+2][ncol]).doubleValue();
                      tRetribucion=(matriz[nfilaactual+16-1][0]);
                      tReceptor=(matriz[1][ncol-1]);
                      nfiM=nfiM+5;

EscribirArchivo(año,ccaa,tRetribucion,tReceptor,nRetribuciones,impRetribucion,impRetencion);
                    }
                  };
                }
              }
```

```
                }
                  catch(Exception e){

                     e.printStackTrace();
                  }finally{
                // En el finally cerramos el fichero, para asegurarnos
                // que se cierra tanto si todo va bien como si salta
                // una excepcion.

                  try{
                      if( null != fr ){
                      fr.close();
                      }
                     }
                  catch (Exception e2){
                       e2.printStackTrace();
                       }
                } // TODO code application logic here

            }

        }

// Procedimiento que escribe en el archivo Tributaciones la información de cada línea.

        public static void EscribirArchivo(String año, String ccaa,String tRetribucion,String tReceptor,Double nRetribuciones,Double impRetribucion,Double impRetencion) throws IOException {

        FileWriter fichero = null;
```

```
try
{       File file = new File("C:\\Carga/SQLLOADER/Tributaciones.csv");
// BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file, true),"ISO-8859-1"
));
CsvWriter out = new CsvWriter(new FileWriter(file, true),';' );
// Unifica el nombre de las Comunidades Autónomas
try {
    if (ccaa.equals("Navarra (Comunidad Foral de)")) {
    ccaa="Comunidad Foral de Navarra";
  }
  if (ccaa.equals("Navarra  Comunidad Foral de")) {
    ccaa="Comunidad Foral de Navarra";
  }
  if (ccaa.equals("Asturias  Principado de")) {
    ccaa="Principado de Asturias";
  }
  if (ccaa.equals("Asturias (Principado de)")) {
    ccaa="Principado de Asturias";
  }

  else if(ccaa.equals("Balears (Illes)")) {
    ccaa="Illes Balears";
  }

  else if(ccaa.equals("Balears  Illes")) {
    ccaa="Illes Balears";
  }
  else if(ccaa.equals("Balears")) {
    ccaa="Illes Balears";
  }
```

```
else if(ccaa.equals("MURCIA (REGIÓN DE)")) {
    ccaa="REGIÓN DE MURCIA";
}
else if(ccaa.equals("Murcia  Región de")) {
    ccaa="REGIÓN DE MURCIA";
}
 else if(ccaa.equals("Rioja La")) {
    ccaa="LA RIOJA";
}

 else if(ccaa.equals("RIOJA (LA)")) {
    ccaa="LA RIOJA";
}
else if(ccaa.equals("Cataluña")) {
    ccaa="CATALUNYA";
}
 else if(ccaa.equals("Ciudad autónoma de Ceuta")) {
    ccaa="CEUTA";
}
else if(ccaa.equals("Ciudad autónoma de Melilla")) {
    ccaa="MELILLA";
}
 else if(ccaa.equals("Madrid  Comunidad de")) {
    ccaa="COMUNIDAD DE MADRID";
}
else if(ccaa.equals("Madrid (Comunidad de)")) {
    ccaa="COMUNIDAD DE MADRID";

}

 // Corrige el punto por coma para el decimal
```

```
String SnRetribuciones=String.valueOf(nRetribuciones);
SnRetribuciones=SnRetribuciones.replace(".", ",");
String SimpRetribucion=String.valueOf(impRetribucion);
SimpRetribucion=SimpRetribucion.replace(".", ",");
String SimpRetencion=String.valueOf(impRetencion);
SimpRetencion=SimpRetencion.replace(".", ",");
        /* Tratamiento del tipo de Receptor ( segmento ) */

tReceptor=tReceptor.replace(",","-" );
tReceptor=tReceptor.replace("\"","" );
tReceptor=tReceptor.replace(";","-" );
tReceptor=tReceptor.replace("y","-" );
tReceptor=tReceptor.replace("-d","-D" );
tReceptor=tReceptor.replace("-p","-P" );

// Escribe en el archivo final las variables año,ccaa y Población

 out.write(ccaa.toUpperCase());
 out.write(año);
 out.write(SnRetribuciones);
 out.write(SimpRetribucion);
 out.write(SimpRetencion);
 out.write(tRetribucion);
 out.write(tReceptor.toUpperCase());
 out.endRecord();
 out.close();
}
finally {
  out.close();
}
```

```
 }
      catch (Exception e) {
         e.printStackTrace();
      } finally {

         // Nuevamente aprovechamos el finally para
      // asegurarnos que se cierra el fichero.
      if (null != fichero)
          fichero.close();
      try {

      } catch (Exception e2) {
           e2.printStackTrace();
      }

      }

   }
}
```

9.1.2 Poblacion.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package poblacion;

/**
 *
```

```
 * @author avaquero
 */
import com.csvreader.CsvWriter;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.StringTokenizer;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;



public class Poblacion {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {



  // Elimina el archivo Poblacion.csv generado en anteriores cargas
  File fichero = new File("C:\\Carga/sqlloader/Poblacion.csv");
  if (fichero.delete());

  File Rs = new File("C:\\Carga/Poblacion");
  File ff[]=Rs.listFiles();
  int narchivos=ff.length;
  String año;
  String ccaa;
  Double poblacion=0.0;
```

```
// Abre cada uno de los archivos que está en la ruta c:/TFC
    for(int i=0;i<=narchivos-1;i++)
    {

// Crea las variables para el tratamiento de los archivos

        FileReader fr = null;
        BufferedReader br = null;
        BufferedReader br2 = null;
        File archivoactual=null;
        String rutaarcactual=ff[i].getAbsolutePath();
        archivoactual = new File (rutaarcactual);

// Abre el archivo actual

          try {
            fr = new FileReader (archivoactual);
// Abre el archivo aceptando los acentos
            br = new BufferedReader(new InputStreamReader(new FileInputStream(archivoactual),"ISO-8859-1"));
            br2 = new BufferedReader(new InputStreamReader(new FileInputStream(archivoactual), "ISO-8859-
1"));
            String linea="";

// Lee la primera linea
            linea=br.readLine();
//Detecta cual es el caracter de separación ( token )
              char token=linea.charAt(0);
              String tokenactual=Character.toString(token);
// Parte la linea en partes separadas por el token

              StringTokenizer st = new StringTokenizer(linea, tokenactual, true);
```

// Cálculo del número de filas del archivo para dimensional correctamente la matriz

```
int lNumeroLineas = 0;
while ((br2.readLine())!=null) {
 lNumeroLineas++;
}
```

// Cálculo del número de particiones de cada linea para dimensional correctamente la matriz, (el caracter de token es una partición

```
int longitud=linea.length();
int ntokens=0;
 for(int f=0;f<longitud;f++) {
     char caracter=linea.charAt(f);
     if (caracter==token) {
        ntokens++;
      }
  }
```

// Crea la matriz que usaremos posteriormente para la obtención de la información necesaria

```
String [][] matriz = new String[lNumeroLineas][2*ntokens+1];
```

// Variables para la carga de la información procedente del archivo actual a la matriz

```
String current="";    // current es el valor actual
int fila=0;
int columna=0;
```

/* Carga de la información del archivo actual a la matriz. Detecta si el campo es un caracter de token*/

/* Trata si el dato es un token, si es numérico*/

```
for(int j=1;j<=lNumeroLineas;j++) {

        while (linea !=null) {

        st = new StringTokenizer(linea, tokenactual, true);

          while (st.hasMoreTokens()) {

           current = st.nextToken();
           current=current.trim();
           current=current.replace("\"","");
           current=current.replace(";","");
           current=current.trim();


             if (current.equals("..")) {

                matriz[fila][columna]="0";
                 columna=columna+1;

             }


            else if(!current.equals(tokenactual)) {
                    int longit=current.length();
                    if (current.substring(longit-1,longit).equals("E")) {
                        current=current.replace(",",".");
                        Float f=Float.valueOf(current).floatValue();
                        matriz[fila][columna]=current;
                        columna=columna+1;
```

```
                    }
                     else {
                         matriz[fila][columna]=current;
                         columna=columna+1;
                     }

                }

                 else {

                     matriz[fila][columna]=null;
                     columna=columna+1;
                 }

            }
             fila++;
             columna=0;
             linea=br.readLine();

        }

        }

for (int j=1;j<=lNumeroLineas-1;j++) {
```

```
                    longitud=rutaarcactual.length();
                    año=rutaarcactual.substring(longitud-8, longitud-4);
                    ccaa=matriz[j][0];
                    poblacion=Double.valueOf(matriz[j][2])+Double.valueOf(matriz[j][4]) ;
                    EscribirArchivo(año,ccaa,poblacion);

                }

            }
              catch(Exception e){

                e.printStackTrace();
              }finally{
            // En el finally cerramos el fichero, para asegurarnos
            // que se cierra tanto si todo va bien como si salta
            // una excepcion.

              try{
                 if( null != fr ){
                 fr.close();
                 }
                }
              catch (Exception e2){
                  e2.printStackTrace();
                 }
            } // TODO code application logic here

    }

}
```

```
            // En el finally cerramos el fichero, para asegurarnos
        // Procedimiento que escribe en el archivo Tributaciones la información de cada línea.

        // Procedimiento que escribe en el archivo Tributaciones la información de cada línea.

static void EscribirArchivo(String año,String ccaa,Double poblacion) throws IOException {

 FileWriter fichero = null;

 try
 {

  File file = new File("C:\\/Carga/SQLLOADER/Poblacion.csv");

   CsvWriter out = new CsvWriter(new FileWriter(file, true),';' );

  // Unifica el nombre de las Comunidades Autónomas


  if (ccaa.equals("Navarra (Comunidad Foral de)")) {

      ccaa="Comunidad Foral de Navarra";
   }
   if (ccaa.equals("Navarra  Comunidad Foral de")) {

      ccaa="Comunidad Foral de Navarra";
   }
  if (ccaa.equals("Asturias  Principado de")) {

      ccaa="Principado de Asturias";
   }
```

```
if (ccaa.equals("Asturias (Principado de)")) {

    ccaa="Principado de Asturias";
}

else if(ccaa.equals("Balears (Illes)")) {

    ccaa="Illes Balears";
}

else if(ccaa.equals("Balears  Illes")) {

    ccaa="Illes Balears";
}
else if(ccaa.equals("Balears")) {

    ccaa="Illes Balears";
}

else if(ccaa.equals("Murcia (Región de)")) {

    ccaa="REGIÓN DE MURCIA";
}
else if(ccaa.equals("Comunidad Valenciana")) {

    ccaa="COMUNITAT VALENCIANA";
}
else if(ccaa.equals("Murcia  Región de")) {

    ccaa="REGIÓN DE MURCIA";
}
```

```
else if(ccaa.equals("Rioja  La")) {

    ccaa="LA RIOJA";
}

else if(ccaa.equals("Rioja (La)")) {

    ccaa="LA RIOJA";
}
else if(ccaa.equals("Cataluña")) {

    ccaa="CATALUNYA";
}
else if(ccaa.equals("Ciudad autónoma de Ceuta")) {

    ccaa="CEUTA";
}
else if(ccaa.equals("Ciudad autónoma de Melilla")) {

    ccaa="MELILLA";
}
else if(ccaa.equals("Madrid  Comunidad de")) {

    ccaa="COMUNIDAD DE MADRID";
}
else if(ccaa.equals("Madrid (Comunidad de)")) {

    ccaa="COMUNIDAD DE MADRID";
}

// Corrige el punto por coma para el decimal
```

```
        String SPoblacion=String.valueOf(poblacion);
        SPoblacion=SPoblacion.replace(".", ",");

        // Escribe en el archivo final las variables año,ccaa y Población
        out.write(año);
        out.write(ccaa.toUpperCase());
        out.write(SPoblacion);
        out.endRecord();
        out.close();
      }
      catch (Exception e) {
        e.printStackTrace();
      } finally {

        // Nuevamente aprovechamos el finally para
      // asegurarnos que se cierra el fichero.
      if (null != fichero)
         fichero.close();
      try {

      } catch (Exception e2) {
          e2.printStackTrace();
      }
        }
    }
}
```

9.1.3 Porcpoblacionactiva.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package porcpoblacionactiva;

/**
 *
 * @author avaquero
 */

import com.csvreader.CsvWriter;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.StringTokenizer;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

public class porcpoblacionactiva {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
```

```
    // Elimina el archivo PorcentajePoblActiva.csv generado en anteriores cargas
   File fichero = new File("C:\\Carga/SQLLOADER/PorcentajePoblActiva.csv");
   if (fichero.delete());

   File Rs = new File("C:\\Carga/PorcentajePoblActiva");
   File ff[]=Rs.listFiles();
   int narchivos=ff.length;
   String año;
   String ccaa;
   Double porcentaje=0.0;

// Abre cada uno de los archivos que está en la ruta c:/TFC

     for(int i=0;i<=narchivos-1;i++)
     {
// Crea las variables para el tratamiento de los archivos

          FileReader fr = null;
          BufferedReader br = null;
          BufferedReader br2 = null;
          File archivoactual=null;
          String rutaarcactual=ff[i].getAbsolutePath();
          archivoactual = new File (rutaarcactual);

// Abre el archivo actual

              try {
                 fr = new FileReader (archivoactual);

// Abre el archivo aceptando los acentos
```

```
br = new BufferedReader(new InputStreamReader(new FileInputStream(archivoactual),"ISO-8859-1"));

br2 = new BufferedReader(new InputStreamReader(new FileInputStream(archivoactual), "ISO-8859-1"));

String linea="";



// Lee la primera linea

linea=br.readLine();



//Detecta cual es el caracter de separación ( token )

char token=linea.charAt(0);

String tokenactual=Character.toString(token);

// Parte la linea en partes separadas por el token



StringTokenizer st = new StringTokenizer(linea, tokenactual, true);



// Cálculo del número de filas del archivo para dimensional correctamente la matriz



int lNumeroLineas = 0;

while ((br2.readLine())!=null) {

 lNumeroLineas++;

}

// Cálculo del número de particiones de cada linea para dimensional correctamente la matriz, (el caracter de token es una partición



 int longitud=linea.length();

 int ntokens=0;

 for(int f=0;f<longitud;f++) {



    char caracter=linea.charAt(f);
```

```
            if (caracter==token) {

              ntokens++;
             }
           }
// Crea la matriz que usaremos posteriormente para la obtención de la información necesaria

            String [][] matriz = new String[lNumeroLineas][2*ntokens+1];

// Variables para la carga de la información procedente del archivo actual a la matriz

            String current="";    // current es el valor actual
            int fila=0;
            int columna=0;

/* Carga de la información del archivo actual a la matriz. Detecta si el campo es un caracter de token*/
/* Trata si el dato es un token, si es numérico*/

            for(int j=1;j<=lNumeroLineas;j++) {

                while (linea !=null) {
                st = new StringTokenizer(linea, tokenactual, true);
                  while (st.hasMoreTokens()) {
                   current = st.nextToken();
                   current=current.trim();
                   current=current.replace("\"","");
                   current=current.replace(";","");
                   current=current.trim();
                     if (current.equals("")) {

                        matriz[fila][columna]="";
```

```
                columna=columna+1;
            }
            else if(!current.equals(tokenactual)) {

                matriz[fila][columna]=current;
                columna=columna+1;
            }
              else {
                matriz[fila][columna]=null;
                columna=columna+1;
              }

          }
          fila++;
          columna=0;
          linea=br.readLine();

      }

      }

for (int j=4;j<=lNumeroLineas-1;j++) {
    for (int z=0;z<=5;z++) {
    año=matriz[1][2*z+1];
    ccaa=matriz[j][0];
    porcentaje=Double.valueOf(matriz[j][z*2+2]).doubleValue();
    EscribirArchivo(año,ccaa,porcentaje);
  }
```

```
                }



            }
              catch(Exception e){
                 e.printStackTrace();
              }finally{
             // En el finally cerramos el fichero, para asegurarnos
             // que se cierra tanto si todo va bien como si salta
             // una excepcion.

              try{
                 if( null != fr ){
                 fr.close();
                 }
                }
              catch (Exception e2){
                  e2.printStackTrace();
                 }
            } // TODO code application logic here


    }

}

  // En el finally cerramos el fichero, para asegurarnos
  // Procedimiento que escribe en el archivo Tributaciones la información de cada línea.

   // Procedimiento que escribe en el archivo Tributaciones la información de cada línea.

static void EscribirArchivo(String año,String ccaa,Double porcentaje) throws IOException {
```

```
FileWriter fichero = null;
try
{

 File file = new File("C:\\/Carga/SQLLOADER/PorcentajePoblActiva.csv");

   CsvWriter out = new CsvWriter(new FileWriter(file, true),';' );

// Unifica el nombre de las Comunidades Autónomas


   if (ccaa.equals("Navarra (Comunidad Foral de)")) {

      ccaa="Comunidad Foral de Navarra";
   }
   if (ccaa.equals("Navarra  Comunidad Foral de")) {

      ccaa="Comunidad Foral de Navarra";
   }
  if (ccaa.equals("Asturias  Principado de")) {

      ccaa="Principado de Asturias";
   }
  if (ccaa.equals("Asturias (Principado de)")) {

      ccaa="Principado de Asturias";
   }

   else if(ccaa.equals("Balears (Illes)")) {
```

```
    ccaa="Illes Balears";
}

else if(ccaa.equals("Balears  Illes")) {

    ccaa="Illes Balears";
}
 else if(ccaa.equals("Balears")) {

    ccaa="Illes Balears";
}

else if(ccaa.equals("Murcia (Región de)")) {

    ccaa="REGIÓN DE MURCIA";
}
else if(ccaa.equals("Murcia  Región de")) {

    ccaa="REGIÓN DE MURCIA";
}
 else if(ccaa.equals("Rioja  La")) {

    ccaa="LA RIOJA";
}

 else if(ccaa.equals("Rioja (La)")) {

    ccaa="LA RIOJA";
}
else if(ccaa.equals("Cataluña")) {
```

```
    ccaa="CATALUNYA";
  }
  else if(ccaa.equals("Ciudad autónoma de Ceuta")) {

    ccaa="CEUTA";
  }
  else if(ccaa.equals("Ciudad autónoma de Melilla")) {

    ccaa="MELILLA";
  }
  else if(ccaa.equals("Madrid  Comunidad de")) {

    ccaa="COMUNIDAD DE MADRID";
  }
  else if(ccaa.equals("Madrid (Comunidad de)")) {

    ccaa="COMUNIDAD DE MADRID";

  }

// Corrige el punto por coma para el decimal

  String SPorcentaje=String.valueOf(porcentaje);
  SPorcentaje=SPorcentaje.replace(".", ",");

// Escribe en el archivo final las variables año,ccaa y porcentaje de población

  out.write(año);
  out.write(ccaa.toUpperCase());
  out.write(SPorcentaje);
  out.endRecord();
```

```
            out.close();
        }

        catch (Exception e) {
            e.printStackTrace();
        } finally {

            // Nuevamente aprovechamos el finally para
        // asegurarnos que se cierra el fichero.
        if (null != fichero)
            fichero.close();
        try {

        } catch (Exception e2) {
            e2.printStackTrace();
        }

          }


    }

}
```

9.2 Script Carga

9.2.1 Script Borrado

```
begin
borrartablas();
commit;
end;
/
exit;
```

9.2.2 Script Carga

```
begin
tfc.EjecutarProcedimientos();
commit;
end;
/
exit;
```

9.3 Procedimientos almacenados

9.2.1 Poblacion media

```
create or replace PROCEDURE PoblacionMedia
is
 añoact poblacion_tmp.año%TYPE;
 añoant poblacion_tmp.año%TYPE;
 poblacionact poblacion_tmp.habitantes%TYPE;
 poblacionant poblacion_tmp.habitantes%TYPE;
 poblacionmed poblacion_tmp.habitantes%TYPE;
 ccaaant poblacion_tmp.ccaa%TYPE;
```

```
 ccaaact poblacion_tmp.ccaa%TYPE;

 cursor cca is

select distinct ccaa,año

from poblacion_tmp

order by ccaa,año;


 BEGIN


poblacionant:=0;

ccaaant:='';

open cca;

fetch cca into ccaaact,añoact;


while cca%found

LOOP



 if (ccaaant=ccaaact and añoant<>añoact) then


select habitantes into poblacionact

from poblacion_tmp

where año=añoact and ccaa=ccaaact

order by ccaa,año;


poblacionmed:= (poblacionact+poblacionant)/2;



dbms_output.put_line(añoact||' '||ccaaact||' '||poblacionant||' '|| poblacionact||' '||poblacionmed);


INSERT INTO Poblacion_tmp2 (año,ccaa,habitantes)

VALUES (añoact, ccaaact, round(poblacionact));
```

```
UPDATE Poblacion_tmp2
SET año=añoant,ccaa=ccaaant,habitantes=round(poblacionmed)
WHERE año=añoant and ccaa=ccaaant;

poblacionant:=poblacionact;
ccaaant:=ccaaact;
añoant:=añoact;

else

select habitantes into poblacionact
from poblacion_tmp
where año=añoact and ccaa=ccaaact
order by ccaa,año;
poblacionmed:=poblacionact;

INSERT INTO Poblacion_tmp2 (año,ccaa,habitantes)
VALUES (añoact, ccaaact, round(poblacionact));

dbms_output.put_line('No es igual'||' '||añoact||' '||ccaaact||' '||poblacionant||' '|| poblacionact||' '||poblacionmed);

poblacionant:=poblacionact;
ccaaant:=ccaaact;
añoant:=añoact;

end if;
```

```
fetch cca into ccaaact,añoact;

end loop;

close cca;

END ;
```

9.2.2 Ejecutar Procedimientos

```
create or replace PROCEDURE EjecutarProcedimientos

is

BEGIN

cargaaños();

cargarccaa();

cargatiporetribucion();

cargasegmentos();

cargatributaciones();

poblacionmedia();

cargapoblacion();

cargapoblacionactivanoret();

cargapoblacionnoactivanoret();

actimportes();

commit;

end;
```

9.2.3 Cargar Tributaciones

```
create or replace PROCEDURE Cargatributaciones

is

 ccaaactual tributaciones_tmp.ccaa%TYPE;

 añoactual tributaciones_tmp.año%TYPE;
```

```
segactual tributaciones_tmp.segmento%TYPE;
treactual tributaciones_tmp.tiporetribucion%TYPE;
nreactual tributaciones_tmp.nretribuciones%TYPE;
iretractual tributaciones_tmp.imp_retribucion%TYPE;
ireteactual tributaciones_tmp.imp_retribucion%TYPE;
cidccaa Comunidades_Autonomas.id_ccaa%TYPE;
cidsegmento segmento.id_segmento%TYPE;
cidtiporetribucion tipo_retribucion.id_tiporetribucion%TYPE;

cursor ctributacion is

select ccaa,tributaciones_tmp.año,nretribuciones,imp_retribucion,imp_retencion,tiporetribucion,segmento
from Tributaciones_tmp,AÑo,Segmento,Tipo_Retribucion
where Tributaciones_tmp.año=AÑo.año
and Tributaciones_tmp.segmento=Segmento.des_segmento
and Tributaciones_tmp.tiporetribucion=Tipo_Retribucion.des_tiporetribucion
and NRetribuciones<>0;

BEGIN
  DBMS_OUTPUT.PUT_LINE('Se han cargado las siguientes tributaciones');

open ctributacion;
 fetch ctributacion into ccaaactual,añoactual,nreactual,iretractual,ireteactual,treactual,segactual;

while ctributacion%found
LOOP

      select id_ccaa into cidccaa
      from Comunidades_Autonomas
      where nombreccaa=ccaaactual;
```

```
select id_segmento into cidsegmento

from Segmento

where des_segmento=segactual;


select id_tiporetribucion into cidtiporetribucion

from Tipo_Retribucion

where des_tiporetribucion=treactual;




DBMS_OUTPUT.PUT_LINE(cidccaa||','||cidsegmento||','||cidtiporetribucion);


 INSERT INTO
Tributaciones(ID_CCAA,AÃ‘O,ID_TIPORETRIBUCION,ID_SEGMENTO,NRETRIBUCIONES,IMPRETRIBUCION,IMPRETENCION,ID_TRIBU
TACION)

VALUES (cidccaa,añoactual,cidtiporetribucion,cidsegmento,nreactual,iretractual,ireteactual,secidtributacion.nextval);



fetch ctributacion into ccaaactual,añoactual,nreactual,iretractual,ireteactual,treactual,segactual;


end loop;

close ctributacion;

end;
```

9.2.4 CargarTipoTributacion

```
create or replace PROCEDURE CargaTipoRetribucion

is


 tractual tributaciones_tmp.tiporetribucion%TYPE;


 cursor ctret is

 Select distinct tributaciones_tmp.tiporetribucion

  from tributaciones_tmp

  where tributaciones_tmp.tiporetribucion<>'TODOS'

  order by tributaciones_tmp.tiporetribucion;

BEGIN

  DBMS_OUTPUT.PUT_LINE('Se han cargado los siguientes tipos de retribucion');


 open ctret;

 fetch ctret into tractual;


 while ctret%found

 LOOP

DBMS_OUTPUT.PUT_LINE(tractual);

INSERT INTO Tipo_Retribucion(ID_TIPORETRIBUCION,DES_TIPORETRIBUCION)

VALUES (sectiporetribucion.nextval,tractual);



 fetch ctret into tractual;



end loop;
```

```
close ctret;

end;
```

9.2.5 Cargasegmentos

```
create or replace PROCEDURE Cargasegmentos

is

 segactual tributaciones_tmp.segmento%TYPE;

 cursor csegmento is

 Select distinct tributaciones_tmp.segmento

  from tributaciones_tmp

  where tributaciones_tmp.segmento<>'TODOS'

   order by tributaciones_tmp.segmento;

BEGIN

  DBMS_OUTPUT.PUT_LINE('Se han cargado los siguientes segmentos');


 open csegmento;

 fetch csegmento into segactual;


 while csegmento%found

 LOOP

DBMS_OUTPUT.PUT_LINE(segactual);

 INSERT INTO Segmento(ID_SEGMENTO,DES_SEGMENTO,ACTIVA)

VALUES (secsegmento.nextval,segactual,1);




 fetch csegmento into segactual;



end loop;

close csegmento;
```

```
UPDATE SEGMENTO
SET ACTIVA=0
WHERE DES_SEGMENTO='PENSIONISTAS';

INSERT INTO Segmento(ID_SEGMENTO,DES_SEGMENTO,ACTIVA)
VALUES (secsegmento.nextval,'ACTIVA SIN RETRIBUCION',1);
DBMS_OUTPUT.PUT_LINE('ACTIVA SIN RETRIBUCION');

INSERT INTO Segmento(ID_SEGMENTO,DES_SEGMENTO,ACTIVA)
VALUES (secsegmento.nextval,'NOACTIVA SIN RETRIBUCION',0);
DBMS_OUTPUT.PUT_LINE('NOACTIVA SIN RETRIBUCION');


end;
```

9.2.6 Cargarccaa

```
create or replace PROCEDURE CargarCCAA

is

/* DeclaraciÓNn variables */

 nombreccaa tributaciones_tmp.ccaa%TYPE;

/* Cursor, obtiene las Comunidades AutÓNnomas que tienen tributaciones, por lo que no
se carga ni PaÃs Vasco ni Navarra */

cursor cca is
select distinct ccaa
```

```
from tributaciones_tmp

where ccaa<>'TOTAL'

order by ccaa;



 BEGIN


open cca;

fetch cca into nombreccaa;


/* Cada CCAA es cargada en la tabla COMUNIDADES_AUTONOMAS */



while cca%found

LOOP



INSERT INTO Comunidades_Autonomas(ID_CCAA,nombreCCAA,latitud,longitud)

VALUES (secidccaa.nextval,nombreCCAA,1,1);


DBMS_OUTPUT.PUT_LINE(nombreCCAA);


fetch cca into nombreCCAA;




end loop;

close cca;


/* Actualiza en cada CCAA la latitud y longitud para los informes en Google Maps */
```

```
UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=37.38264,LONGITUD=-5.996295
WHERE NOMBRECCAA='ANDALUCÃA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=41.65629,LONGITUD=-0.765379
WHERE NOMBRECCAA='ARAGÃ"N';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=43.36026,LONGITUD=-5.844759
WHERE NOMBRECCAA='PRINCIPADO DE ASTURIAS';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=28.46981,LONGITUD=-16.25486
WHERE NOMBRECCAA='CANARIAS';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=43.46096,LONGITUD=-3.807934
WHERE NOMBRECCAA='CANTABRIA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=39.85678,LONGITUD=-4.024476
WHERE NOMBRECCAA='CASTILLA - LA MANCHA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=41.65295,LONGITUD=-4.728388
WHERE NOMBRECCAA='CASTILLA Y LEÃ"N';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=41.38792,LONGITUD=2.169919
WHERE NOMBRECCAA='CATALUNYA';
```

```
UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=38.8786,LONGITUD=-6.970284
WHERE NOMBRECCAA='EXTREMADURA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=42.88045,LONGITUD=-8.546304
WHERE NOMBRECCAA='GALICIA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=39.68446,LONGITUD=2.847511
WHERE NOMBRECCAA='ILLES BALEARS';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=42.46577,LONGITUD=-2.449995
WHERE NOMBRECCAA='LA RIOJA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=40.41669,LONGITUD=-3.700346
WHERE NOMBRECCAA='COMUNIDAD DE MADRID';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=37.98344,LONGITUD=-1.12989
WHERE NOMBRECCAA='REGIÃ“N DE MURCIA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=39.47024,LONGITUD=-0.768049
WHERE NOMBRECCAA='COMUNITAT VALENCIANA';
```

```
UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=35.88829,LONGITUD=-5.316195
WHERE NOMBRECCAA='CEUTA';

UPDATE COMUNIDADES_AUTONOMAS
SET LATITUD=35.29234,LONGITUD=-2.938794
WHERE NOMBRECCAA='MELILLA';
END ;
```

9.2.7 Cargapoblacionnoactivanoret

```
create or replace PROCEDURE CargapoblacionNoActivaNoRet
is

 idccaaanterior tributaciones.id_ccaa%TYPE;
 idccaaactual tributaciones.id_ccaa%TYPE;
 ccaaactual poblacion_tmp2.CCAA%TYPE;
 añoactual  tributaciones.año%TYPE;
 añoanterior  tributaciones.año%TYPE;
 idsegactual tributaciones.id_segmento%TYPE;
 idactnorec segmento.des_segmento%TYPE;
 idnoactnorec segmento.des_segmento%TYPE;
 segactual segmento.des_segmento%TYPE;
 nreactual tributaciones.nretribuciones%TYPE;
 pobnoactiva poblacion_tmp2.habitantes%TYPE;
 pobnoactivaacum poblacion_tmp2.habitantes%TYPE;
 habactual poblacion_tmp2.habitantes%TYPE;
 habitacumulado poblacion_tmp2.habitantes%TYPE;
 porcpobl porcpoblacionactiva_tmp.porcentaje%TYPE;
```

```
cursor cnoret is
select distinct poblacion.id_ccaa,poblacion.año
from poblacion
order by id_ccaa,año;

BEGIN

DBMS_OUTPUT.PUT_LINE('Calculando las personas por segmento');

idccaaanterior:=0;
añoanterior:=0;
pobnoactiva:=0;
habitacumulado:=0;
habactual:=0;

select id_segmento into idnoactnorec
from segmento
where des_segmento='NOACTIVA SIN RETRIBUCION';

open cnoret;
fetch cnoret into idccaaactual,añoactual;
while cnoret%found
LOOP

select distinct nombreccaa into ccaaactual
from poblacion,comunidades_autonomas
where poblacion.id_ccaa=comunidades_autonomas.id_ccaa
and poblacion.id_ccaa=idccaaactual;

select sum(poblacion.habitantes)into pobnoactivaacum
```

```
from poblacion,segmento

where poblacion.id_segmento=segmento.id_segmento

and id_ccaa=idccaaactual and año=añoactual

and activa=0;


select sum(poblacion_tmp2.habitantes) into habitacumulado

from poblacion_tmp2

where ccaa=ccaaactual and año=añoactual;


select porcentaje INTO porcpobl

from porcpoblacionactiva_tmp

where ccaa=ccaaactual and año=añoactual;


pobnoactiva:= habitacumulado*((100-porcpobl)/100);


habactual:=pobnoactiva-pobnoactivaacum;


DBMS_OUTPUT.PUT_LINE(idccaaactual||','||añoactual||','||pobnoactivaacum||','||habitacumulado||','||pobnoactiva||','||hab
actual);

 UPDATE Poblacion

SET habitantes=habactual

WHERE id_ccaa=idccaaactual and id_segmento=idnoactnorec and año=añoactual;


fetch cnoret into idccaaactual,añoactual;

end loop;

close cnoret;


end;
```

9.2.8 Cargapoblacionactivanoret

```
create or replace PROCEDURE CargapoblacionActivaNoRet

is


 idccaaanterior tributaciones.id_ccaa%TYPE;

 idccaaactual tributaciones.id_ccaa%TYPE;

 ccaaactual poblacion_tmp2.CCAA%TYPE;

 añoactual  tributaciones.año%TYPE;

 añoanterior  tributaciones.año%TYPE;

 idsegactual tributaciones.id_segmento%TYPE;

 idactnorec segmento.des_segmento%TYPE;

 idnoactnorec segmento.des_segmento%TYPE;

 segactual segmento.des_segmento%TYPE;

 nreactual tributaciones.nretribuciones%TYPE;

 pobactiva poblacion_tmp2.habitantes%TYPE;

 pobactivaacum poblacion_tmp2.habitantes%TYPE;

 habactual poblacion_tmp2.habitantes%TYPE;

 habitacumulado poblacion_tmp2.habitantes%TYPE;

 porcpobl porcpoblacionactiva_tmp.porcentaje%TYPE;


cursor cnoret is

select distinct poblacion.id_ccaa,poblacion.año

from poblacion

order by id_ccaa,año;



BEGIN
```

```
idccaaanterior:=0;

añoanterior:=0;

pobactiva:=0;

habitacumulado:=0;

habactual:=0;


select id_segmento into idactnorec

from segmento

where des_segmento='ACTIVA SIN RETRIBUCION';


open cnoret;

 fetch cnoret into idccaaactual,añoactual;

while cnoret%found

LOOP


select distinct nombreccaa into ccaaactual

from poblacion,comunidades_autonomas

where poblacion.id_ccaa=comunidades_autonomas.id_ccaa

and poblacion.id_ccaa=idccaaactual;


select sum(poblacion.habitantes)into pobactivaacum

from poblacion,segmento

where poblacion.id_segmento=segmento.id_segmento

and id_ccaa=idccaaactual and año=añoactual

and activa=1;



select sum(poblacion_tmp2.habitantes) into habitacumulado

from poblacion_tmp2

where ccaa=ccaaactual and año=añoactual;
```

```
select porcentaje INTO porcpobl
from porcpoblacionactiva_tmp
where ccaa=ccaaactual and año=añoactual;
pobactiva:= habitacumulado*(porcpobl/100);
habactual:=pobactiva-pobactivaacum;

UPDATE Poblacion
SET habitantes=habactual
WHERE id_ccaa=idccaaactual and id_segmento=idactnorec and año=añoactual;

fetch cnoret into idccaaactual,añoactual;
end loop;
close cnoret;
end;
```

9.2.9 CargaPoblacion

```
create or replace PROCEDURE Cargapoblacion
is

/*Declaracion de variables */

idccaaanterior tributaciones.id_ccaa%TYPE;
idccaaactual tributaciones.id_ccaa%TYPE;
ccaaactual poblacion_tmp2.CCAA%TYPE;
añoactual  tributaciones.año%TYPE;
añoanterior  tributaciones.año%TYPE;
idsegactual tributaciones.id_segmento%TYPE;
idactnorec segmento.des_segmento%TYPE;
idnoactnorec segmento.des_segmento%TYPE;
segactual segmento.des_segmento%TYPE;
```

```
 nreactual tributaciones.nretribuciones%TYPE;

 pobactiva poblacion_tmp2.habitantes%TYPE;

 habactual poblacion_tmp2.habitantes%TYPE;

 habitacumulado poblacion_tmp2.habitantes%TYPE;


/*Cursor. Obtiene las combinaciones distintas de CCAA,año y segmento. */


cursor cpoblacion is

select distinct tributaciones.id_ccaa,tributaciones.año,tributaciones.id_segmento

from tributaciones

order by id_ccaa,año,tributaciones.id_segmento;


BEGIN



/* Inicializa los valores de las variables  */


idccaaanterior:=0;

añoanterior:=0;

pobactiva:=0;

habitacumulado:=0;

habactual:=0;


/* Obtiene el id del segmento ACTIVA SIN RETRIBUCION */


select id_segmento into idactnorec

from segmento

where des_segmento='ACTIVA SIN RETRIBUCION';


/* Obtiene el id del segmento NOACTIVA SIN RETRIBUCION */
```

```
select id_segmento into idnoactnorec
from segmento
where des_segmento='NOACTIVA SIN RETRIBUCION';

/* Abre el cursor */

open cpoblacion;

fetch cpoblacion into idccaaactual,añoactual,idsegactual;
while cpoblacion%found
LOOP

/* Obtiene el nombre correspondiente al id del segmento actual del cursor */

select DISTINCT DES_SEGMENTO into segactual
from tributaciones,segmento
where tributaciones.id_segmento=segmento.id_segmento
AND tributaciones.id_segmento=idsegactual;

/* 1Âº se inserta en la tabla poblaciÓNn la combinaciÓNn actual del cursor con habitantes a 0 que
se calcularÃ¡n mÃ¡s adelante */

INSERT INTO poblacion(id_ccaa,año,id_segmento,habitantes)
VALUES(idccaaactual,añoactual,idsegactual,0);

/* 1Âº.b Se inserta en la tabla poblaciÓNn el registro correspondiente al sector activa sin retribuciÓNn
y no-activa sin retribuciÓNn para una ccaa y año.  */

/* Se averigua si es la primera combinaciÓNn de ccaa y año */
```

```
if(idccaaanterior<>idccaaactual or añoanterior<>añoactual) then
 DBMS_OUTPUT.PUT_LINE('Estoy en el primer condicional');

/* Obtiene el nombre de la CCAA actual */

select DISTINCT CCAA into ccaaactual
from poblacion_tmp2,comunidades_autonomas
where CCAA=NOMBRECCAA
AND ID_CCAA=idccaaactual;

/* Obtiene el nÃºmero de habitantes correspondiente a la CCAA y año actual */

select habitantes into pobactiva
from poblacion_tmp2
where ccaa=ccaaactual and año=añoactual;

INSERT INTO poblacion(id_ccaa,año,id_segmento,habitantes)
VALUES(idccaaactual,añoactual,idactnorec,0);

 INSERT INTO poblacion(id_ccaa,año,id_segmento,habitantes)
VALUES(idccaaactual,añoactual,idnoactnorec,0);

idccaaanterior:=idccaaactual;
añoanterior:=añoactual;
habactual:=0;
habitacumulado:=0;
```

```
end if;

/* 2Â º SegÃºn el tipo de segmento ( tiporeceptor ) se registra la poblaciÓNn dividiendo
el nÃºmero de retribuciones por el nÃºmero de tipo de retribuciÓNn distinta que tenga
que pueden ser 1,2 o 3 */

if (segactual='ASALARIADOS' OR segactual='PENSIONISTAS' OR segactual='DESEMPLEADOS') THEN

select nretribuciones into habactual
from tributaciones
where id_ccaa=idccaaactual and año=añoactual and id_segmento=idsegactual;

UPDATE Poblacion
SET habitantes=habactual
WHERE id_ccaa=idccaaactual and id_segmento=idsegactual and año=añoactual;

elsif (segactual='ASALARIADOS - PENSIONISTAS' OR segactual='ASALARIADOS - DESEMPLEADOS' OR
segactual='PENSIONISTAS - DESEMPLEADOS') THEN

select sum(nretribuciones) into habactual
from tributaciones
where id_ccaa=idccaaactual and año=añoactual and id_segmento=idsegactual;

UPDATE Poblacion
SET habitantes=habactual/2
WHERE id_ccaa=idccaaactual and id_segmento=idsegactual and año=añoactual;
```

```
else

select sum(nretribuciones) into habactual
from tributaciones
where id_ccaa=idccaaactual and año=añoactual and id_segmento=idsegactual;

UPDATE Poblacion
SET habitantes=habactual/3
WHERE id_ccaa=idccaaactual and id_segmento=idsegactual and año=añoactual;

END IF;

fetch cpoblacion into idccaaactual,añoactual,idsegactual;
end loop;
close cpoblacion;

end;
```

9.2.10 Cargaaños

```
create or replace PROCEDURE Cargaaños
is
 /* DeclaraciÓNn de variables */
 añoactual tributaciones_tmp.año%TYPE;

 /* Cursor, obtiene Ãºnicamente los años que tienen tributaciÓNn por lo que los años anteriores no se cargarÃ¡n
 en la tabla años */

 cursor caño is
   Select distinct tributaciones_tmp.año
```

```
  from tributaciones_tmp,poblacion_tmp

  where tributaciones_tmp.año=poblacion_tmp.año

  order by tributaciones_tmp.año;


BEGIN


/* Carga de los años en la tabla años */


 DBMS_OUTPUT.PUT_LINE('Se podrÃ¡ obtener informaciÓNn de los siguientes años');


 open caño;

 fetch caño into añoactual;


 while caño%found

 LOOP


DBMS_OUTPUT.PUT_LINE(añoactual);


INSERT INTO AÃ‘O(año)


values(añoactual);




fetch caño into añoactual;




end loop;

close caño;

end;
```

9.2.11 Borrartablas

```
create or replace PROCEDURE BorrarTablas

is

BEGIN


DELETE FROM POBLACION;

DELETE FROM POBLACION_TMP;

DELETE FROM POBLACION_TMP2;

DELETE FROM TRIBUTACIONES;

DELETE FROM TRIBUTACIONES_TMP;

DELETE FROM PORCPOBLACIONACTIVA_TMP;

DELETE FROM AÃ‘O;

DELETE FROM COMUNIDADES_AUTONOMAS;

DELETE FROM TIPO_RETRIBUCION;

DELETE FROM SEGMENTO;

COMMIT;

END;
```

9.2.12 Actimportes

```
create or replace PROCEDURE ActImportes

is

begin

UPDATE tributaciones

SET impretribucion=impretribucion*1000,impretencion=impretencion*1000

WHERE año>=2006;

end;
```

9.2.13 Actimpretribuciones

```
create or replace PROCEDURE ActImpRetribuciones

is

begin

UPDATE tributaciones

SET impretribucion=impretribucion*1000

WHERE año>=2006;

end;
```