DISEÑO DE UN SISTEMA DE CONTROL DE PROYECTOS

Nombre Estudiante: José Ramón Rodríguez Goas
ITG/ITS

Nombre Consultor: Gladys Utrera Iglesias

Fecha entrega: 10/01/2005

RESUMEN
DISEÑO DE UN SISTEMA DE CONTROL DE PROYECTOS

El principal objetivo del proyecto fin de carrera es afianzar los conocimientos adquiridos durante la carrera aplicados a la gestión y creación de bases de datos relaciones. El proyecto esta basado en la gestión de datos de una empresa dedicada a la creación de software para el sector inmobiliario. Dicha empresa tiene sede en varias ciudades (centros) los cuales están divididos en departamentos incluyendo estos a sus respectivos empleados. Los clientes solicitan proyectos a estos centros asignándoles los correspondientes jefes de proyectos y empleados juntos a los gastos que este proyecto conlleva. Todos estos datos serán almacenados y gestionados por la base de datos.

La base de datos del sistema se ha diseñado a partir del diagrama Entidad/Relación y se ha implementado utilizando Oracle versión 9i. Oracle es una base de datos potente que ha permitido implementar fácilmente todos los requisitos del diagrama Entidad/Relación. Una vez especificadas todas las tablas se procede a la creación de los procedimientos para la correcta gestión de la base de datos.

Durante la realización del mismo se han tenido en cuenta apartados tales como: planificación, análisis de requisitos, diseño, esquemas E/R, gestión de tablas, banco de pruebas, datos aleatorios.

Una vez implementado el proyecto en si se incluye una pequeña aplicación en java que creara datos de forma aleatoria para las diferentes tablas, manteniendo siempre una buena gestión y creación de los mismos datos. De esta manera podemos obtener un banco de pruebas.

INDICE

1. INTRODUCCION

1.1 JUSTIFICACION TFC

La organización de la empresa de "software Goas" esta muy definida, es una empresa a nivel nacional que se encarga del diseño e implantación de software dedicado al sector inmobiliario. Es por eso que tendremos una sucursal de la empresa en determinadas ciudades. Esta empresa lleva 10 años desarrollando software de gestión de bases de datos inmobiliarias. Su desarrollo a lo largo de este tiempo ha permitido asentarse en dicho mercado además de ir perfeccionándolo.

Las grandes empresas y franquicias inmobiliarias de cada ciudad gestionan sus datos a través del software realizado por dicha empresa. El sector inmobiliario refleja una gran demanda y oferta de inmuebles tanto en venta como alquiler, dichos datos deben quedar reflejados de alguna manera. Anteriormente todos estos datos se almacenaban en formato papel, hoy en día gracias a las nuevas tecnologías y fundamentalmente gracias al nuevo software informático dichos datos se recogen en servidores de bases de datos que almacenan una gran información sobre los clientes. Dicho software debe mantener gran cantidad de datos además deben almacenarse de manera fiable y segura

Para una correcta gestión de los proyectos que gestiona dicha empresa se debe crear una base de datos que permita acceder a los datos lo más fácil y eficientemente posible.

1.2 OBJETIVOS DEL TFC

El desarrollo del proyecto permite afianzar varios objetivos. El principal de ellos es la creación de un sistema que permita controlar los gastos que un determinado trabajador tiene en su vida laboral en la empresa para ello se creara una base de datos que permita una correcta gestión de dichos gastos.

Dentro de este objetivo principal podemos desglosar otros objetivos:

1.1.1. Planificación de un proyecto
1.1.2. Diseño de bases de datos relacionales
1.1.3. Gestión de tablas
1.1.4. Creación de procedimientos sobre tablas
1.1.5. Generación de datos aleatorios
1.1.6. Banco de pruebas.

1.3 ENFOQUE Y METODO SEGUIDO

El enfoque que se le ha dado al proyecto es puramente practico, he planificado dicho proyecto basándome en una empresa de creación de software inmobiliario para darle un aspecto real al proyecto. Dicha empresa tiene varias sucursales y dentro de estas se desarrollan los diferentes proyectos.

Para la realización del proyecto se han utilizados conocimientos adquiridos a lo largo de la carrera. Así se usan conceptos de asignaturas tales como: fundamentos de programación I y II, bases de datos, estructura de la información, ingeniería de software, además de bibliografía adicional.

1.4 PLANIFICACION PROYECTO

Para el desarrollo de la PEC 1 tenemos un margen de 7 días en el que se desarrollaran las siguientes tareas:

1º) Lectura del enunciado del proyecto
2º) Segunda lectura desglosando contenidos
3º) Aplicar el enunciado a una determinada empresa
4º) Desarrollo explicativo de la empresa, en cuestión, con una introducción
5º) Desarrollo del plan de trabajo.
6º) Análisis GANTT

Como se observa esta primera PEC 1 trata esencialmente del correcto entendimiento del proyecto así como la planificación del mismo

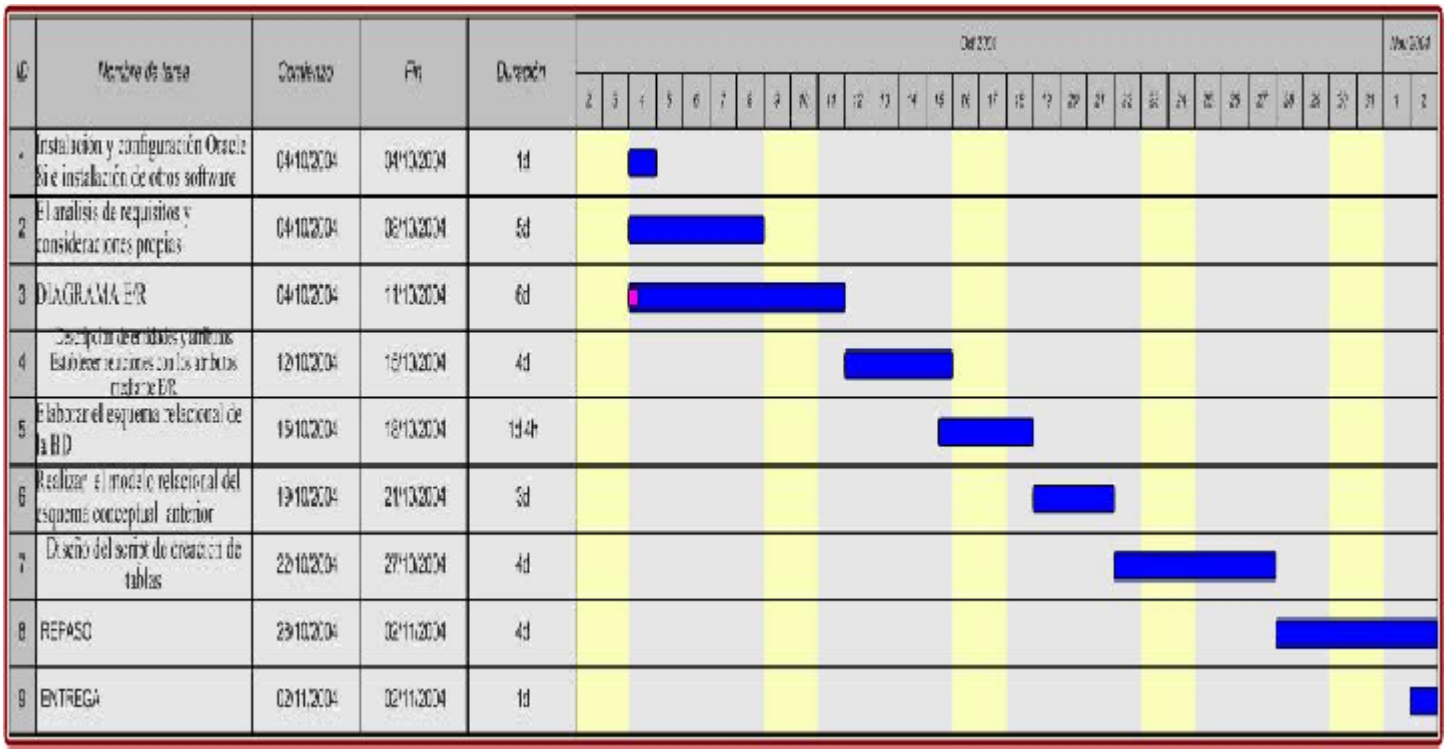
ID	Nombre de tarea	Comienzo	Fin	Duración
1	Lectura del enunciado del proyecto	27/09/2004	29/09/2004	1d
2	Segunda lectura	27/09/2004	28/09/2004	2d
3	Aplicar el enunciado a la empresa	27/09/2004	29/09/2004	3d
4	Desarrollo de la empresa mas introducción	30/09/2004	30/09/2004	1d
5	Desarrollo del plan de trabajo y analisis Gannt	01/10/2004	01/10/2004	1d
6	Diagrama de Gannt pec 1 y entrega PEC 1	01/10/2004	01/10/2004	1d

Para la realización de la segunda parte del proyecto (PEC2) tendremos de plazo 1 mes en que se realizara el diseño en profundidad de la aplicación así como los requisitos necesarios para la realización del mismo.

1º) Como primer objetivo será necesario instalar el entorno que utilizaremos para el desarrollo. En este caso se ha decidido usar el Oracle 8i o superior. Este entorno deberá ser correctamente configurado para su buen uso y para una vez avanzado el proyecto comenzar a implementarlo.
2º) Una vez configurado e instalado el entorno de trabajo podemos instalar otros software para uso de aplicaciones de diseño y herramientas case, en mi caso he optado por instalar el Microsoft Visio profesional que ayudara a agilizar el proyecto.
3º) El análisis de requisitos es una de las fases mas importantes del diseño del proyecto, en esta fase se deberá analizar el proyecto a conciencia, tomando nota de todos los puntos importantes para que el futuro la implementación de las tablas sean correctas y eficientes. Para ello cada punto del enunciado será desarrollado concienzudamente y con ejemplos para una correcta comprensión.
4º) Establecer además consideraciones propias en referencia al proyecto.

5º) Una vez obtenidos todos los datos desarrollados pasaremos al desarrollo del diagrama E/R con las especificaciones anteriores. Este diagrama nos facilitara la idea del proyecto así como futuras implementaciones.
6º) Elaborar aparte una descripción de las entidades con sus atributos.
7º) Establecer las relaciones con los atributos mediante el diagrama E/R
8º) Elaborar el esquema relacional de la BD.
9º) Realizar la transformación al modelo relacional del esquema conceptual obtenido en el apartado anterior.
10º) Análisis de los procedimientos requeridos(altas, bajas..) se comentara como se llevaran a cabo.
11º) Diseño del script de creación de tablas.

Una vez obtenidos todos los desgloses aportados por los apartados anteriores pasaremos a diseñar el proyecto mediante la implementación de los procedimientos almacenados que se requieren para que las aplicaciones los utilicen. Esto formara parte de la PEC 3.



Mediante esta temporización se ha conseguido llegar a un punto en el que la parte del proyecto de diseño ha finalizado. De aquí a la siguiente PEC nos encargaremos de crear e implementar la aplicación mediante ORACLE (SQL) y crear y probar todos los procedimientos pedidos.

La ultima parte del proyecto se basara principalmente en la creación de procedimienos y la generacion de pruebas de la aplicación. Debemos realizar esta parte correctamente y con todo lujo de detalles para que quede bien implementada. Esta PEC debe encargarse de las siguientes tareas:

1º) Definir cada uno de los procedimientos indicando que es lo que realizara cada uno específicamente. Cada procedimiento se probara con datos que posteriormente inventaremos de tal manera que todo sea coherente.

2º) Implementar los procedimientos almacenados que se requieren para que las aplicaciones los utilicen. Estos procedimientos deberán ser como mínimo:

- Alta de proyecto
- Cierre de proyecto
- Alta, Baja, Modificación del Centro, Departamento, Gasto, tipo de gasto …
- Asignación / desasignación de trabajadores y gastos al proyecto.
- Trabajador entra a trabajar.
- Trabajador sale de trabajar.
- Alta de un trabajador.
- Baja de un trabajador.
- Modificación de los datos del trabajador.
- Cambio de centro/departamento/tipo de trabajador.
- Consulta de trabajador, departamento…

3º) Realizar mecanismo de inicialización de la base de datos tal que simule un año de funcionamiento. Datos de todos los tipos y ordenados por tablas de tal manera que los resultados sean correctos.

4º) Crear datos de prueba de tal manera que refleje todas las situaciones posibles.

5º) Tratamiento de errores y excepciones tanto de implementación como de datos erróneos.

6º) Prueba de la implementación con los datos obtenidos previamente. Se observara si hemos realizado todo correctamente de lo contrario debemos volver al paso 2 y 3.

7º) Resumen de funcionamiento y modo de uso. Guía que nos mostrara como se ejecuta dicha aplicación y si los datos obtenidos son los correctos.

8º) Bibliografía

1.5 PRODUCTOS OBTENIDOS

- Script de creación de tablas (jrodriguezgoas_script.txt)
- Script de procedimientos (jrodriguezgoas_procedimiento.txt)
- Script borrado tablas (jrodriguezgoas_scriptdelete.txt)

- script borrado procedimientos(jrodriguezgoas_procedelete.txt)
- fichero java de creación de datos aleatorios (COracle.java)
- Presentación proyecto (jrodriguezgoas_presentacion.ppt)
- Memoria proyecto (jrodriguezgoas_memoria.doc)

1.6 DESCRIPCION OTROS CAPITULOS

Dentro del proyecto podemos encontrar apartados comunes entre ellos que podemos desglosarlos a groso modo en los siguientes:

- Recursos físicos y de hardware: específica cuales son los recursos mínimos y necesarios para la correcta gestión del proyecto.
- Instalación Oracle en Windows
- Análisis del sistema o base de datos y sus requisitos: especificaremos aquí cuales son los condicionantes de los datos y puntos clave a tener en cuenta para la realización del proyecto.
- Diseño de la base de datos: gestionaremos aquí la creación del modelo E/R, incluyendo diagrama, puntos clave, claves primarias…. Y también teniendo en cuenta las consideraciones propias.
- Creación de tablas del modelo E/R: veremos en este apartado la generación del script de las tablas.
- Volumen de datos del sistema: en este apartado se analiza el volumen posible de datos del sistema.
- Creación de procedimientos: desglosaremos cada uno de los procedimientos con sus precondiciones y postcondiciones.
- Eliminación de tablas y procedimientos
- Generación de datos aleatorios: mediante una aplicación en java que genera datos aleatorios para cada una de las tablas.

2. RECURSOS FISICOS Y DE SOFTWARE

La primera tarea de la planificación del desarrollo de software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de software. Es un aspecto importante para el desarrollo y posterior gestión de la base de datos. Podemos desglosarlos en recursos físicos y de software.

2.1. RECURSOS FISICOS

La maquina donde se va a desarrollar el proyecto debe tener unos recursos mínimos:

- Pentium 4 a 3 GHZ
- 512 Ram DDR
- Disco Duro 40 Gb Para el almacenamiento de datos
- Tarjeta Red 10/100 para la conexiona Internet
- Grabadoras de CD para copias de seguridad
- Ratón y teclado
- Conexión a internet (cable MODEM 600Kb)
- Tarjeta grafica 128 mb /estandar

2.2. RECURSOS DE SOFTWARE

Igual que utilizamos hardware como herramienta para construir nuevo hardware, utilizamos software como ayuda en el desarrollo de nuevo software. Podemos desglosar varios software importantes

2.2.1.Sistema operativo Windows XP

Desde el lanzamiento de Windows 95, Microsoft cambió completamente el concepto que se tenía de este popular programa, ya que pasó de ser un programa que simplificaba las funciones del DOS. a un completo sistema operativo mucho más fácil de usar, con un diseño más cómodo para el usuario, con nuevas funciones y muchas ventajas.

Windows 95 fue evolucionando, apareciendo posteriormente el Windows 98, y Windows Me. (Sin contar el Windows NT y Windows 2000 que han sido especialmente diseñados para trabajo en red).

Microsoft Windows XP es la última versión de este nuevo sistema operativo. Dentro de lo que es Windows XP tenemos la versión casera (Home Edition) y la

versión profesional (Professional), en pocas palabras podríamos decir que Windows XP Home Edition es la versión de este sistema operativo ideal para entretenimiento y el hogar, ya que le ayuda a hacer más cosas con su computadora y la Internet, proporcionando una interface novedosa sobre todo por los llamativos colores y facilidad de uso. La versión Professional incluye todas las características del Home Edition y además otras características que lo hacen ideal para negocios y computadoras portátiles.

.-En el siguiente enlace del Cuadro Comparativo se puede ver claramente las diferencias entre Windows XP Home Edition y Professional:

http://www.microsoft.com/latam/windowsxp/home/evaluacion/actualizar/comparacion.asp

Las herramientas que posee Windows xp ayudaran a la realización del proyecto en cuestión. Además debemos tener en cuenta otra serie de herramientas muy útiles para la creación y mantenimiento de la base de datos. Primeramente definiremos una serie de conceptos para después adentrarnos en el entorno Oracle.

El Software de gestión es una solución informática que permite a las empresas simplificar la gestión de su negocio. Todos los procesos de una empresa (producción, compras, logística, distribución, etc) quedan integrados en un único sistema que permite disponer de información y herramientas de análisis muy sencillas y ágiles. Es también denominado ERP o SAP, un ERP es un conjunto de aplicaciones o soluciones que cubren, bajo un mismo paraguas, las necesidades de los usuarios en el entorno operativo de una empresa: área económico-financiera, área logístico-comercial, área de producción y área de recursos.

Una vez adentrados en el mundo del software de gestión analizaremos más profundamente en los entornos de bases de datos. Como primer concepto a definir encontramos ***El Sistema Gestor de Bases de Datos (SGBD)*** es un conjunto de programas, procedimientos y lenguajes que proporcionan a los usuarios las herramientas necesarias para operar con una base de datos. Por tanto, el SGBD actúa como un intermediario entre los usuarios y los datos. Debe cumplir una serie de funciones como descripción de los datos, de manera que debe permitir definir los registros, sus campos, sus relaciones de autorización, etc. Debe manipular los datos permitiendo a los usuarios insertar, suprimir, modificar y consultar datos de la base de datos y por último, debe permitir usar la base de datos, dando un interfaz adecuado a cada tipo de usuario.

Una vez que se ha explicado al motor de bases de datos la forma deseada para los datos, usando por ejemplo un entorno interactivo como es Access, el motor creará algunos objetos físicos en los que guardará los datos. El motor de base de datos es el encargado de realizar las consultas, altas, bajas, modificaciones, procedimientos, etc. de forma transparente al usuario. A la combinación entre estructura y datos será a lo que nos referiremos como base de datos.

Podéis encontrar mas información en la siguiente dirección sobre gestores de bases de datos:

http://traductica.upf.es/Modular2000/gbd/gbd5.htm

A continuación nos sumergiremos mas profundamente con el software de gestión Oracle, con el cual realizaremos la practica.

2.2.2.Software gestión Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas Web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, Access, MySQL, SQL Server, etc.

Nos centraremos ahora en una definición mas clara de Oracle y como funciona la programación sobre éste. Oracle como antes he mencionado se basa en la tecnología cliente/servidor, pues bien, para su utilización primero sería necesario la instalación de la herramienta servidor (Oracle 8i) y posteriormente podríamos gestionar la base de datos desde otros equipos con herramientas de desarrollo como Oracle Designer y Oracle Developer, que son las herramientas básicas de programación sobre Oracle.
Es posible lógicamente atacar a la base de datos a través del SQL plus incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL.

- El Developer es una herramienta que nos permite crear formularios en local, es decir, mediante esta herramienta nosotros podemos crear formularios, compilarlos y ejecutarlos, pero si queremos que los otros trabajen sobre este formulario deberemos copiarlo regularmente en una carpeta compartida para todos, de modo que, cuando quieran realizar un cambio, deberán copiarlo de dicha carpeta y luego volverlo a subir a la carpeta. Este sistema como podemos observar es bastante engorroso y poco fiable pues es bastante normal que las versiones se pierdan y se machaquen con frecuencia. La principal ventaja de esta herramienta es que es bastante intuitiva y dispone de un modo que nos permite componer el formulario, tal y como lo haríamos en Visual Basic o en Visual C, esto es muy de agradecer.
- Los problemas anteriores quedan totalmente resueltos con Designer que es una herramienta que se conecta a la base de datos y por tanto creamos los formularios en ella, de esta manera todo el mundo se conecta mediante Designer a la aplicación que contiene todos los formularios y no hay problemas de diferentes versiones, esto es muy útil y perfecto para evitar machacar el trabajo de otros. Pero el principal y más notable problema es la falta de un entorno visual para diseñar el formulario, es decir, nos aparece una estructura como de árbol en la cual insertamos un formulario, a la vez dentro de éste insertamos bloques o módulos que son las estructuras que contendrán los elementos del formularios, que pueden estar basados en tablas o no.

2.2.3.SQL

Hasta la década de los 80, las personas que preparaban las consultas e informes de una base de datos debían ser programadores. Al aparecer las bases de datos con lenguajes de consulta sencillos y estandarizados, semejantes al lenguaje natural, el proceso de consulta puede hacerlo cualquier usuario mediante un lenguaje escrito asequible.

El lenguaje de gestión de bases de datos más conocido en la actualidad es el SQL, Structured Query Language, que es un lenguaje estándar internacional, comúnmente aceptado por los fabricantes de generadores de bases de datos. En concreto, el gestor de bases de datos Oracle utiliza el lenguaje SQL.

El SQL trabaja con estructura cliente/servidor sobre una red de ordenadores. El ordenador cliente es el que inicia la consulta; el ordenador servidor es que atiende esa consulta. El cliente utiliza toda su capacidad de proceso para trabajar; se limita a solicitar datos al ordenador servidor, sin depender para nada más del exterior. Estas peticiones y las respuestas son transferencias de textos que cada ordenador cliente se encarga de sacar por pantalla, presentar en informes tabulados, imprimir, guardar, etc., dejando el servidor libre.

El SQL permite:
* Definir una base de datos mediante tablas
* Almacenar información en tablas.
* Seleccionar la información que sea necesaria de la base de datos.
* Realizar cambios en la información y estructura de los datos.
* Combinar y calcular datos para conseguir la información necesaria.

2.2.4.Pruebas JAVA

Como bien dice el enunciado será necesario creas un juego de pruebas basados en las tablas. Para ello he empleado el lenguaje java, mediante el cual creo una pequeña aplicación que introducirá aleatoriamente los datos en las tablas creadas. Este programa tiene una funcionalidad únicamente diseñada de comprobación ya que generara una gran cantidad de datos y en todas las formas posibles controlando posibles errores mediante los procedimientos. Esta sección se vera mas desarrollada en apartados posteriores.

2.2.5.Microsoft Visio

Está orientado a crear gráficos de gran impacto visual lo más rápidamente posible, además de hacerlo sencillamente. Se trabaja principalmente haciendo uso de plantillas, de las cuales incluye una gran cantidad, como se puede ver en la tabla adjunta. El uso de estas plantillas conlleva dos beneficios: por una parte pone a disposición del usuario una serie de símbolos y formas adecuados al tipo de diseño que se está creando, y por otra, una serie de herramientas que aceleran el proceso de creación y el uso y gestión de la información, aunque no todas las plantillas disponen de estas herramientas. Como ejemplos, se puede encontrara herramientas en la Ingeniería de Procesos, la creación de mapas de sitios Web, la creación de diagramas UML, etc.

Las herramientas de asistencia están adaptadas a cada tipo de diseño, de forma que, por ejemplo, para crear diagramas UML, se puede realizar ingeniería inversa sobre un proyecto en Visual C++ o Visual Basic 6.0, creándose una completa solución de diseño UML, controlando no sólo el aspecto del diagrama de estructura estática, sino que permite crear los de colaboración, actividades, casos de uso, etc, desde un interfaz similar al entorno gráfico de programación, y capaz de diferenciar las entidades, como clases, escenarios, mensajes….

Este programa contiene numerosas herramientas case de gestión para las bases de datos con este programa facilitare la creación de los modelos E/R así como resto de gráficos y esquemas que aparecen en la realización de todo el proyecto.

2.2.6.Herramientas ayuda.

- Internet Explorer
- Outlook Express
- Word
- Excel
- PowerPoint
- Otras aplicaciones básicas

3. INSTALACIÓN ORACLE EN WINDOWS XP Y OTROS SOFTWARE

3.1.ORACLE Y ENTORNO Y REQUISITOS MÍNIMOS

Podemos encontrar todos los pasos a seguir para la instalación dentro del cd1 de instalación del paquete Oracle. Siguiendo la siguiente ruta:

D:/DOC/INSTALL/install.html

Los requisitos necesarios para la instalación del paquete Oracle también vienen especificados en dicha información de instalación.

4. ANÁLISIS REQUISITOS

El análisis de requisitos es una de las fases más importantes del diseño del proyecto, en esta fase se deberá analizar el proyecto a conciencia, tomando nota de todos los puntos importantes para que en el futuro la implementación de las tablas sea correcta y eficiente.

Para ello cada punto del enunciado será desarrollado concienzudamente y con ejemplos para una correcta comprensión.

4.1.ANÁLISIS ENUNCIADO

✓ Centros de trabajo

Es de esperar que para que la empresa funcione a nivel nacional deberán desarrollarse varias sucursales. En el caso de la empresa software Goas se ha decidido establecer 5 sucursales, de esta forma podemos abarcar geográficamente gran parte del territorio nacional. Las sucursales son las siguientes:

CENTROS DE TRABAJO	AREA DE DESARROLLO
ALANDALUS(SEVILLA)	ANDALUCIA
CASTELL (BARCELONA)	CATALUÑA
MESETA(MADRID)	MADRID
LEVANTE(VALENCIA)	VALENCIA
ASTUR(CORUÑA)	GALICIA

Cada centro de trabajo puede tener distintas cargas de trabajo. Para sopesar estas cargas de trabajo cada centro de trabajo tendrá una plantilla de trabajadores según la demanda.

Cada centro de trabajo dividirá su carga laboral en varios frentes o departamentos de lo contrario los centros serian un desastre. Esto lo veremos en el apartado siguiente.

ESPECIFICACIONES:

- El número máximo de centros es 10 y el mínimo es 5.
- Lo ideal es que en un futuro este mas extendido por el territorio nacional, un centro en cada ciudad.
- De cada centro guardaremos el cif que lo identifica, el nombre del centro, la ciudad donde reside, un teléfono de contacto y el número del fax.

✓ Departamentos

Para que cada centro de trabajo funcione eficientemente será necesario y obligado dividir a los trabajadores según departamentos. Cada departamento tiene una función específica dentro de cada centro, a continuación se especificaran los departamentos:

➢ Dirección

Este departamento se encarga de la gestión del centro. La gestión de un centro contempla varios aspectos:

- Dirección general del centro
- Aspectos económicos
- Aspectos de gestión
- Aspectos de nuevos desarrollos

➢ Desarrollo de software

Se encargaran del desarrollo de software solicitado por los clientes, esta carga de trabajo se desarrollara mediante la gestión de diferentes funciones que especificaremos mas adelante:

- Diseño del entorno de software
- Desarrollo de la aplicación
- Supervisión del desarrollo(Jefe de Proyecto)

➢ Mantenimiento

Este departamento se encarga de las reparaciones, seguridad y mantenimiento. Es un departamento necesario dentro de cada centro para un correcto funcionamiento.

ESPECIFICACIONES:

- El número máximo de departamentos es 4 y el mínimo es 2. En un futuro puede que surjan nuevos departamentos por ejemplo: comerciales que se encargaran de promocionar la empresa por las diferentes inmobiliarias. Aunque en un principio no se ha contemplado este departamento.
- Cada departamento tendrá un mínimo de 5 trabajadores y un máximo de 40. Todo dependerá de la carga de trabajo de cada departamento.
- Un empleado solo puede pertenecer a un departamento o es de dirección o es de desarrollo o es de mantenimiento, nunca a dos al mismo tiempo. Puede ocurrir que cambien de departamento por promoción interna.
- De cada departamento guardaremos el código que lo identifica, un teléfono de contacto y el nombre del departamento, como hemos dicho anteriormente estos pueden ser dirección, mantenimiento, desarrollo. En un futuro cuando el número de departamentos sea elevado podremos crear una tabla auxiliar de departamento donde se listen los departamentos según tipos o lo que es lo mismo mediante herencias en la base de datos ya que esto nos permitirá crear más campos en cada departamento y especializarlo más.

Para la realización de este proyecto se ha establecido el siguiente desglose de departamentos.

DEPARTAMENTO	NUMERO EMPL.	DEPARTAMENTO
DIRECCION	1	DIRECCION GENERAL CENTRO
	1	SECRETARIO/A
	1	GESTOR ECONOMICO
	1	INVESTIGACION Y DESARROLLO
	1	JEFE PROYECTO DIRECCION
DESARROLLO SOFTWARE	5	DISEÑO SOFTWARE
	30	PROGRAMACION SOFTWARE
	5	JEFE DESARROLLO O PROYECTO
MANTENIMIENTO	8	EMPLEADO MANTEMIENTO
	2	JEFE MANTENIMIENTO

Veamos a continuación un posible desglose de los diferentes departamentos en los diferentes centros, SOFTWARE GOAS S.L tiene desarrollados 5 centros de trabajo que son los siguientes (supongamos que a partir de ahora y en adelante estos serán los centros, departamentos y funciones que desarrollaremos en la aplicación ejemplo):

CENTROS DE TRABAJO	AREA DE DESARROLLO	NUMERO TRABAJADORES	DEPARTAMENTOS
ALANDALUS (SEVILLA)	ANDALUCIA	55	DIRECCION ➔5
			DESARROLLO SOFTWARE ➔40
			MANTENIMIENTO ➔10
CASTELL (BARCELONA)	CATALUÑA	40	DIRECCION ➔5
			DESARROLLO SOFTWARE ➔27
			MANTENIMIENTO ➔8
MESETA (MADRID)	MADRID	50	DIRECCION ➔5
			DESARROLLO SOFTWARE ➔35
			MANTENIMIENTO ➔10
LEVANTE (VALENCIA)	VALENCIA	35	DIRECCION ➔5
			DESARROLLO SOFTWARE ➔25
			MANTENIMIENTO ➔5
ASTUR (CORUÑA)	GALICIA	30	DIRECCION ➔5
			DESARROLLO SOFTWARE ➔18
			MANTENIMIENTO ➔7

✓ Cargos en departamentos

Entre los cargos podemos encontrar muchos y variados, en este cado se ha hecho una selección de los más relevantes para cada departamento.

CARGOS	DEFINICION
DISEÑO SOFTWARE	Se encarga del diseño grafico de los proyectos a realizar. Cada agencia Inmobiliaria tiene un diseño de software y este departamento se encarga de gestionarlo.
DIRECCION GENERAL CENTRO	Se encarga de la dirección general del centro, director en funciones del centro, máximo cargo. Máximo responsable del centro, por el pasaran gran parte de las decisiones de gestion.
SECRETARIO/A	Lleva a cabo todo el tema de papeleo, secretario personal del director. Gestión de agendas dedirección,organización del trabajo diario
PROGRAMACION SOFTWARE	Desarrolla el software previsto por la demanda del cliente. Esta función será desarrollada por varios empleados, se debe a la implementación hoy en día de la técnica de programación orientada a objetos, por dicho motivo los empleados podrán trabajar en modulos.
JEFE DESARROLLO o PROYECTO	Coordina el grupo de programación. Da las pautas para desarrollar un proyecto en un plazo establecido
GESTOR ECONOMICO	Lleva las cuentas economicas del centro. Es un gestor de la economia de la empresa. Gestiona sueldos, gastos.
INVESTIGACION Y DESARROLLO	Busca nuevas alternativas al software usando encuestas, estadisticas, intentan mejorar el servicio de la empresa.
EMPLEADO MANTEMIENTO	Encargados de mantener los equipos del centro así como de actualizaciones y seguridad de datos tanto a nivel de aplicaciones para el propio centro como para los proyectos
JEFE MANTENIMIENTO	Coordina al equipo de mantenimiento.
JEFE DIRECCION	Es el encargado de organizar nuevas plantillas y de coordinar todo el departamento de dirección. Trabaja codo con codo con el director general. Podría decirse que es el encargado de recursos humanos del centro.

ESPECIFICACIONES:

- Cada empleado tiene una función especifica dentro del departamento y no puede desempeñar mas de dicha función ni en el mismo departamento ni en departamentos diferentes, así por ejemplo un empleado de programación de software no puede ejercer de empleado de mantenimiento, ni un jefe de proyecto de desarrollo puede ser jefe de mantenimiento…
- Un jefe de proyectos puede tener asignados varios proyectos.
- He considerado que los jefes de departamentos son los denominados jefe dirección, jefe desarrollo, jefe mantenimiento. Siendo los proyectos asignados únicamente y exclusivamente dedicados a estos.
- De cada cargo guardaremos los siguientes campos el código de cargo, una descripción de este y el sueldo por hora que será diferente en cada cargo (esto nos servirá posteriormente para calcular gasto de empleado al trabajar en un proyecto ya que estos cobran por hora).

✓ Proyectos

Los proyectos podemos desglosarlos en proyectos internos (referentes a la empresa) y proyectos externos (aquellos que solicitan los cliente para la realización de un determinado software.

Cuando llegue un determinado proyecto se deberán asignar a los jefes de proyectos del departamento en cuestión, por ejemplo:

- Imaginemos que hay un proyecto interno de la empresa en el cual se necesita hacer encuestas de calidad a los clientes que han solitita software para su agencia. Este proyecto será asignado al jefe de proyecto de dirección el cual se encargara de dar las responsabilidades necesarias y como primera opción contrata una empresa de encuestas solicitando al secretario que se encargue de la búsqueda.
- Otro ejemplo más claro: supongamos que llega un cliente (agencia inmobiliaria) a la empresa y solicita la creación de un software informático. Este proyecto será presentado al jefe de proyecto de desarrollo de software el cual se encarga de definir los trabajos a realizar por sus trabajadores de departamento para que se cumpla en los plazos previstos.

Una vez asignado el jefe de proyecto al proyecto en cuestión será necesario asignarle trabajadores del mismo departamento. En un proyecto determinado no podrán trabajar empleados de un departamento que no haya sido asignado previamente, es decir, supongamos que tenemos un proyecto (interno) para realizar encuestas no seria correcto asignar empleados del departamento de desarrollo de software.

Estos trabajadores asignados al proyecto se pueden volver a asignar a un proyecto del mismo departamento, así pues por ejemplo un empleado de desarrollo de software puede estar encargándose de la creación de dos módulos gráficos para dos software diferentes bajo la tutela del mismo jefe de proyectos.

ESPECIFICACIONES:

- Un proyecto solo tiene un jefe de proyecto y debe ser asignado desde el principio.
- Un proyecto puede tener n trabajadores asignados de un mismo departamento.
- Cada trabajador podrá tener asignados varios proyectos a realizar siempre que sean del mismo departamento.

- Dentro de cada proyecto guardaremos los siguientes campos: un código del proyecto, nombre del proyecto, el presupuesto(dato muy importante), fecha de inicio, la fecha fin y las horas dedicadas para ese proyecto.

✓ Empleados

Los empleados serán fundamentales en la gestión de las tablas deberemos crear una tabla que se encargue de almacenar a estos cumpliendo las siguientes condiciones:

- Los empleados no pueden estar en más de un departamento.
- Cada empleado tiene una función tipificada, jefe de departamento, jefe de proyecto, dibujante, ingeniero, economista, …
- Un empleado puede cambiar de departamento, de centro de trabajo y de función, cuando cambie alguna de éstas características, no se deberá modificar los datos del trabajador sino que se dará de alta otra vez y "el trabajador antiguo" se "desactivará".

Dentro de cada empleado guardaremos los siguientes campos: un código del empleado (DNI), nombre, apellidos, teléfono, nss.

✓ Gastos

Podemos encontrar diferentes gastos a la hora de gestionarlos. En primer lugar podemos encontrar los gastos generados en la realización de un proyecto, cada proyecto acarrea gastos que pueden ser de diferentes motivos (materiales, licencias, seguridad, mantenimiento, contratos…). Estos gastos deben de estar relacionados con una persona. Veámoslo en el siguiente ejemplo: Tenemos un proyecto de creación de software al desglosar los gastos nos encontramos con el siguiente esquema:

- Gastos de desarrollo que recaerán sobre el empleado o los empleados encargados de desarrollar el software.
- Gastos de gestión de proyecto: recaerán sobre el jefe de proyecto de desarrollo
- Gastos de diseño: recaerán sobre el o los empleados de diseño

Para llevar un control exhaustivo de los que un trabajador genera en un determinado proyecto deberá implementarse una gestión determinada. Se ha optado por registrar cuando un trabajador sale y entra del centro de trabajo de esta forma se controla el tiempo que este ha estado trabajando sobre dicho proyecto. De esta forma los trabajadores cobraran por horas trabajadas en el proyecto al que están asignados. Puede darse el caso de que un empleado del centro tenga asignado varios proyectos de esta forma puede desglosar su trabajo y trabajar en todos parcialmente y así controlar las horas empleadas en cada uno.

Otro gastos a tener en cuenta serán especificados en la plantilla del trabajador y únicamente los gastos referidos a estos. Por ejemplo no tiene sentido dar un gasto de diseño de software a un empleado de mantenimiento.

ESPECIFICACIONES:

- Cada trabajador podrá entrar y salir del centro de trabajo n veces y las mismas quedarán registradas.
- Cada trabajador tendrá asignado un costo por hora.
- Cada proyecto se ha presupuestado y está aceptado por el cliente, por lo tanto antes de comenzar ya se sabe que cobrará. Este dato es importante y el sistema deberá almacenarlo al abrir el proyecto.

- . Una vez un proyecto ha finalizado no se podrá realizar operaciones sobre él, es decir asignarle un trabajador o gastos o el que sea.
- Los gastos se controlaran en una tabla (estática, es decir con unos datos ya tipificados y no variados Ej: gasto material, gasto gestión…) los tipos de gastos lo almacenaremos en una tabla con los siguientes campos: código del gasto y descripción del gasto

✓ Presupuestos

Cuando un proyecto es aceptado se ha de hacer un presupuesto para presentarlo al cliente. Este presupuesto se hará a la alza y se basara en la experiencia acumulada en años de experiencia por la empresa, que tiene conocimiento de cuanto podrá llegar a presupuestarse un determinado proyecto. Supongamos un proyecto de desarrollo de software:

PRESUPUESTO
Jefe de proyectos (20 Horas): 100h X 9€ = 900 €
Empleados programación (200 horas):
Empleado 1 (100 horas): 100 X 6€ = 600 €
Empleado 2 (100 horas): 100 X 6€ = 600 €
Empleado de diseño (50 horas): 50 X 8€ = 400 €
TOTAL: 2500 €

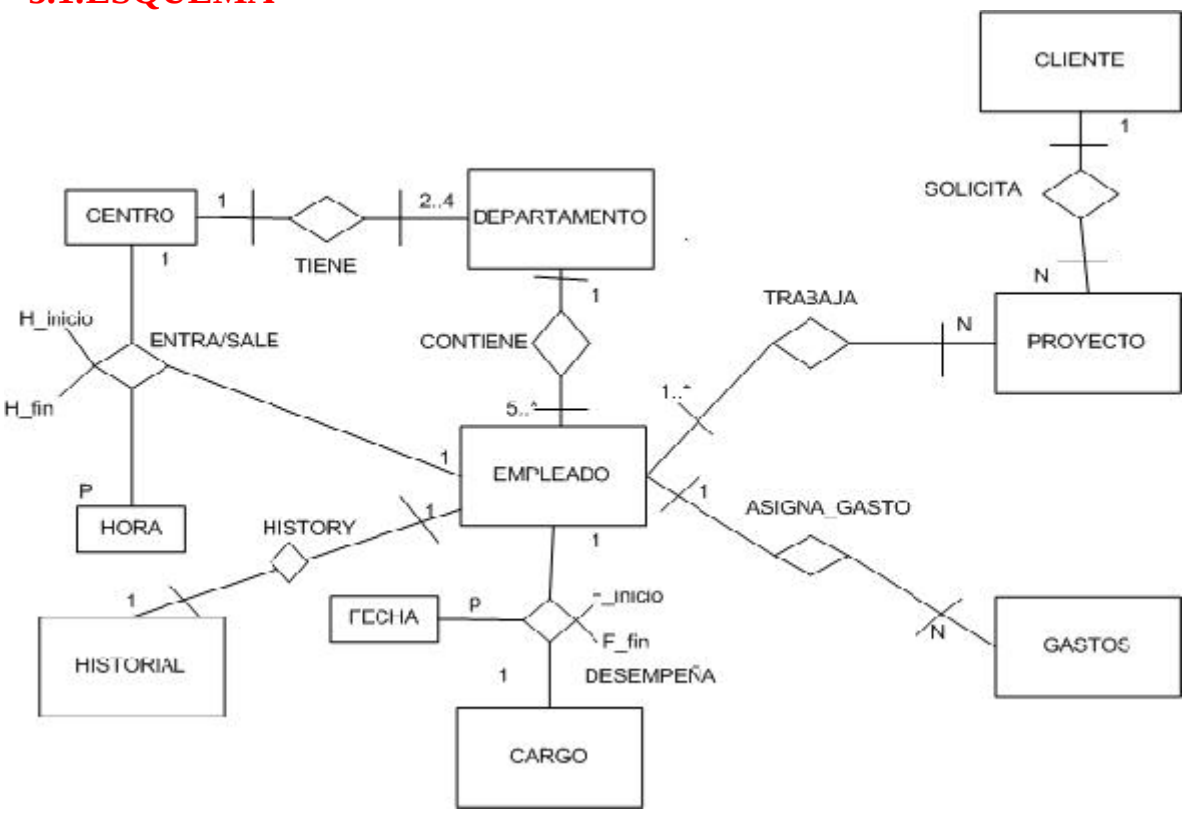
Una vez presupuestado el proyecto deberá ser presentado al cliente para que este lo acepte y se empiece a llevar a cabo la gestión de este.

4.2.CONSIDERACIONES PROPIAS

- Los empleados cuando cambien de centro de trabajo de departamento o de función no modificaran los datos sino que primeramente se dará de baja (borra de Base de datos) y posteriormente se da de alta en la base de datos.
- Los centros de trabajo son inamovibles no se modificaran datos sobre estos.
- En el caso de que se diseñen nuevos departamentos el número de empleados máximos podrá superar los 50.
- No se contempla la creación de nuevas funciones dentro de los departamentos, a no ser que se cree departamentos nuevos en cuyo caso las funciones serán diferentes a las creadas anteriormente teniendo en cuenta que siempre se deberá crear un función principal que es jefe de departamento

5. DIAGRAMA E/R

5.1.ESQUEMA



5.2.ANÁLISIS

Se analizara cada punto dado en el esquema E/R:

- **CENTRO**: Cada centro estará situado en ciudades diferentes no podrán haber dos centros en una misma ciudad. Un centro puede tener 2 o mas departamentos. En cada centro podemos encontrar varios atributos :
  - *CIF: código del centro con formato CifCentrox donde x es el numero del centro.*
  - *Nombre: corresponde a la etiqueta del centro EJ: meseta, alhandales…*
  - *Teléfono: de contacto con el centro*
  - *Fax: del centro en cuestión.*

- **DEPARTAMENTO**: Un departamento de una determinada ciudad pertenece únicamente a ese centro no pertenece a otro centro. Cada departamento tiene un número determinado de personal no inferior a 5 y no superior a 40. De la entidad Departamento podemos sacar varios atributos:
  - *Codigo_departamento: código del departamento con formato D00x donde x es el numero del departamento EJ: CodDepxx donde xx tomara valores diferentes*

- *Teléfono: de contacto con el centro para información general.*
- *Nombre: del departamento en cuestión.*

La inclusión de estos teléfonos proporcionara en un futuro una base para la implantación de CAU en los centros de trabajo además de un desglose mas rápido de las distintas administraciones del centro en caso de emergencia.

- **EMPLEADO**: Un empleado solo pertenece a un solo departamento y todo empleado debe pertenecer al menos a un departamento. Como atributos encontramos:
  - *DNI: código del empleado formato DNIx* donde x tomara diferentes valores
  - *Nombre*
  - *Apellidos*
  - *Telefono*
  - *NSS: Numero seguridad social*

- **CARGO**: cada empleado desempeñara un determinado cargo que en el caso que he planteado son:

CARGOS
JEFE DEPARTAMENTO
SECRETARIO/A
GESTOR ECONOMICO
INVESTIGACION Y DESARROLLO
JEFE DIRECCION
DISEÑO SOFTWARE
PROGRAMACION SOFTWARE
JEFE DESARROLLO
EMPLEADO MANTEMIENTO
JEFE MANTENIMIENTO

Un empleado puede desarrollar todos los cargos (mediante promoción interna) pero no más de 1 a la vez. Por ejemplo el jefe desarrollo no ejercerá funciones de diseño de software. Solo es jefe del departamento desarrollo de software. Además cada empleado tendrá asignado un sueldo por horas que vendrá reflejado según el cargo en el atributo sueldo_hora.

Tiene tres atributos:

- *Codigo_cargo: con formato x donde x es el numero de cargo siendo el 1 el cargo de jefe de departamento(que a su vez será jefe de proyecto)*
- *Descripción del cargo en cuestión.*
- *Sueldo_hora: honorarios según horas y según cargos.*

- **EMPLEADO/HORA/CENTRO:** En esta relación 11P (P por horas) mantenemos el control de las horas en que un empleado ha estado trabajando en el departamento y en dicho centro. En el diagrama de tablas lo veremos como ENTRASALE. En una implementación posterior seria ideal mantener un historial sobre que empleado y el número de horas y sueldo que ha ido acumulando durante su periodo laboral, así mantenemos un desglose de gastos para dicho empleado.

- **EMPLEADO/CARGO/FECHA:** En esta relación 11P (P por fechas) mantenemos el control de los días en que un empleado ha estado trabajando de un determinado cargo. En el diagrama de tablas lo veremos como DESEMPEÑA. Estos

datos incluso podemos meterlos en un futuro en el historial del empleado indicando así los días que ha trabajado junto a las horas y demás datos.

- **PROYECTO:** Los proyectos tendrán un formato definido como el siguiente:
  Tiene tres atributos:
  - *Cod_proyecto: con formato CODPROx donde x es el numero de proyecto siendo*
  - *nombre*
  - *presupuesto*
  - *f_inicio*
  - *f_fin*
  - *horas*

- **PROYECTO/EMPLEADO:** Un proyecto podrá ser realizado por 1 o mas empleados y a su vez un empleado puede trabajar en más de un proyecto. Se vera en la relación TRABAJA

- **GASTOS:** Un proyecto en concreto genera unos determinados gastos y los gastos pueden ser atribuidos a 1 o mas empleados.

OTRAS CONSIDERACIONES:

La definición de tablas se basa en los siguientes criterios aplicados al diagrama entidad relación:

- Definir una tabla por cada entidad.
- Definir una tabla extra para las relaciones M:N y las relaciones en que intervienen tres entidades.
- Definir claves foráneas para las relaciones 1:1 o 1:N.
- Voy a incluir una tabla mas que haga de historial de todos los empleados de todos los centros y departamentos, cada empleado tendrá asignado un numero de historial y en dicho historial se mostrara lo ganado hasta el momento, asi podemos mantener un control de ganancias del personal.
- Ademas he incluido una tabla auxiliar mas denominada errores que nos ayudara a la hora de detecatar los errores en los procedimientos posteriores a la creación de tablas.

5.3.ATRIBUTOS DEL MODELO E/R

CENTRO
Atributo 1: CIF (clave primaria)
Atributo 2: nombre
Atributo 3: ciudad
Atributo 4: telefono
Atributo 5: fax

DEPARTAMENTO
Atributo 1: codigo_departamento (clave primaria)
Atributo 2: telef_infor
Atributo 3; nombre

EMPLEADO
Atributo 1: DNI (clave primaria)

Atributo 2: nombre
Atributo 3: Apellidos
Atributo 4: teléfono
Atributo 5: NSS

CARGO
Atributo 1: codigo_cargo (clave primaria)
Atributo 2: descripción
Atributo 3: sueldo_hora

HORA
Atributo 1: Hora (clave primaria)

FECHA
Atributo 1: Fecha (clave primaria)

PROYECTO
Atributo 1: Codigo_proyecto (clave primaria)
Atributo 2: dni_jefep
Atributo 3: nombre
Atributo 4: presupuesto
Atributo 5: f_inicio
Atributo 6: f_fin
Atributo 7: horas

GASTOS
Atributo 1: cod_gasto (clave primaria)
Atributo 2: descripción

CLIENTE
Atributo 1: Cod_cli(clave primaria)
Atributo 2: Nombre
Atributo 3: apellidos
Atributo 4: Telefono
Atributo 5: Presupuesto

HISTORIAL
Atributo 1: num_historial (clave primaria)
Atributo 2: DNI
Atributo 3: gastos_total
Atributo 4: , h_trabajadas
Atributo 5: Dias_trabajados

ERRORES
Atributo 1: codigo_error (clave primaria)
Atributo 2: msg_error
Atributo 3: procedimiento_ejecutado
Atributo 4: , linea_trabajadas
Atributo 5: fecha_hora

5.4. TRANSFORMACIÓN AL MODELO RELACIONAL DEL ESQUEMA CONCEPTUAL OBTENIDO EN EL APARTADO ANTERIOR.

Una vez definidos los atributos en el apartado anterior pasaremos a realizar la transformación al modelo relacional, hay que tener en cuenta que se han incluido algunas tablas auxiliares para llevar a cabo la correcta gestión de la base de datos. Entre ellas podemos encontrar ENTRASALE, TRABAJA, ASIGNAGASTOS,DESEMPEÑA

CENTRO (CIF, nombre, ciudad, telefono, fax)

DEPARTAMENTO (codigo_departamento, telef_infor, nombre, CIF)
Donde {CIF} referencia a CENTRO

EMPLEADO (DNI, nombre, apellidos, telefono, NSS, codigo_departamento, codigo_cargo)
Donde { codigo_departamento } referencia a DEPARTAMENTO
{codigo_cargo} referencia a CARGO

ENTRASALE (CIF, DNI, h_inicio, h_fin)
Donde {CIF} referencia a CENTRO
Donde {DNI} referencia a EMPLEADO

DESEMPEÑA (codigo_cargo, DNI, f_inicio, f_fin)
Donde {codigo_cargo} referencia a CARGO
y {DNI} referencia a EMPLEADO

PROYECTO(cod_proyecto, cod_cli nombre, dni_jefep,nombre, presupuesto, f_inicio, f_fin, horas)
Donde {cod_cli} referencia a cliente
y { dni_jefep } referencia a EMPLEADO

TRABAJA (cod_proyecto, DNI, f_inicio, f_fin)
Donde {cod_proyecto} referencia a PROYECTO
Y {DNI} referencia a empleado

CARGO(codigo_cargo, descripción, sueldo_hora)

GASTOS (cod_gasto, descripción)

ASIGNAGASTO (cod_gasto, cantidad,DNI,f_gasto)
Donde { cod_gasto } referencia a GASTOS
Donde {DNI} referencia a EMPLEADO

CLIENTE (cod_cli,nombre, apellidos,telefono)

Seria interesante incluir dos tablas finales auxiliares en estas mostraríamos los gastos generados por cada empleado mediante un historial (estaría formado por num_historial, codigo del empleado, gastos totales acumulados por el empleado, horas trabajadas en todos sus proyectos) y controlar los errores mediante una tabla errores .

HISTORIAL (num_historial, DNI, gastos_total, h_trabajadas, dias_trabajados)
Donde {DNI} referencia a EMPLEADO
ERRORES (codigo_error, msg_error, procedimiento_ejecutado, linea, fecha_hora)

6. CREACION DE TABLAS DEL MODELO E/R

SCRIPT CREACION BASE DE DATOS

El siguiente script sirve para crear las tablas de la base de datos, contiene las claves primarias y las referencias cruzadas entre tablas. Los campos obligatorios se han definido como no nulos.

```
-- CREACIÓN DE TABLAS
-- TABLA CENTRO
create table centro(
cif varchar(10) not null,
nombre Varchar (32) not null,
ciudad Varchar (32) not null,
telefono Varchar(32) not null,
fax Varchar(32) not null,
primary key(cif));

-- TABLA DEPARTAMENTO
create table departamento(
codigo_departamento Varchar (32)  not null,
telef_infor varchar(12) not null,
nombre Varchar(32) not null,
cif varchar(10) not null,
primary key(codigo_departamento),
foreign key(cif) references centro(cif));

-- TABLA CARGO
create table cargo(
codigo_cargo integer not null,
descripcion varchar (50),
sueldo_hora integer not null,
primary key(codigo_cargo));

-- TABLA EMPLEADO
create table empleado(
dni varchar (10) not null,
nombre varchar (32) not null,
apellidos varchar (50) not null,
telefono varchar (20),
nss varchar (20) not null,
codigo_departamento varchar (32)  not null,
codigo_cargo integer not null,
primary key(dni),
foreign key (codigo_departamento) references departamento (codigo_departamento),
```

```
foreign key (codigo_cargo) references cargo (codigo_cargo));

-- TABLA DESEMPEÑA
create table desempeña(
codigo_cargo integer not null,
dni varchar (10) not null,
f_inicio date not null,
f_fin date not null,
primary key(codigo_cargo, dni),
foreign key (codigo_cargo) references cargo (codigo_cargo),
foreign key (dni) references empleado (dni));

-- TABLA ENTRASALE
create table entrasale(
cif varchar(10) not null,
dni varchar (10) not null,
h_inicio date not null,
h_fin date not null,
primary key(cif, dni),
foreign key (cif) references centro (cif),
foreign key (dni) references empleado (dni));

-- TABLA CLIENTE
create table cliente(
cod_cli varchar (10) not null,
nombre varchar (32) ,
apellidos varchar (32) ,
telefono varchar(12),
primary key(cod_cli));

-- TABLA PROYECTO
create table proyecto(
cod_proyecto varchar (10) not null,
cod_cli varchar (10) not null,
dni_jefep varchar (10) not null,
nombre varchar (32) ,
presupuesto integer not null,
f_inicio date not null,
f_fin date not null,
horas integer not null,
primary key(cod_proyecto),
foreign key (cod_cli) references cliente (cod_cli));


-- TABLA TRABAJA
create table trabaja(
cod_proyecto varchar (10) not null,
dni varchar (10) not null,
f_inicio date not null,
f_fin date not null,
primary key(cod_proyecto, dni),
foreign key (cod_proyecto) references proyecto (cod_proyecto),
foreign key (dni) references empleado (dni));
```

```
-- TABLA GASTOS
create table gastos(
cod_gasto varchar (10) not null,
descripcion varchar (50),
primary key(cod_gasto));

--TABLA ASIGNAGASTOS
create table asignagastos(
cod_gasto varchar (10) not null,
cantidad integer not null,
dni varchar (10) not null,
f_gasto DATE not null,
primary key (cod_gasto),
foreign key (cod_gasto) references gastos (cod_gasto),
foreign key (dni) references empleado (dni));

-- TABLA HISTORIAL
create table historial(
num_historial integer,
dni varchar (10) not null,
gastos_total integer,
h_trabajadas integer not null,
dias_trabajados integer not null,
primary key(num_historial),
foreign key (dni) references empleado (dni));

-- TABLA ERRORES
create table errores(
codigo_error integer,
msg_error VARCHAR2(200),
procedimiento_ejecutado VARCHAR(64),
linea integer,
fecha_hora date);

commit;
```

7. VOLUMEN DATOS DEL SISTEMA

A continuación calcularemos volúmenes de datos en el caso de que la base de datos se realice completamente sobre todo el sistema especificado en el enunciado:

Centros: 10
Número trabajadores: 150 máximo por cada centro
Número de departamentos máximos: 4
Máximo numero de empleados por departamento: 40

Sobre estos centros desarrollaremos los proyectos, los cuales deberán ser asignados a un cliente, por lo tanto por cada proyecto un cliente, habrá tantos proyectos como clientes. Supongamos que en el plazo de un año hemos realizado 100 proyectos.

Número de proyectos: indeterminado pueden ser más de 100 ya que cada cliente puede tener varios proyectos.
Número de clientes: 100
Número de jefes de proyectos (diferentes): 100
Gastos totales: 500.000 €
Gastos medios de proyectos: 14000 €

Una vez hecho el análisis de los datos a tener en cuenta podemos aproximarnos a calcular el TAMAÑO BASE DE DATOS sobre el número de entradas que se habrán de guardar en cada una de las tablas de la base de datos. Para estimar el tamaño de la base de datos, se tiene en cuenta el espacio que Oracle reserva para cada tipo de datos:

DATE: 7 bytes
INTEGER: 4 bytes
FLOAT: 8 bytes
VARCHAR: depende de la longitud de los strings almacenados. Para estimar el tamaño se utilizarán los tamaños máximos definidos para cada uno de los campos. En la práctica el tamaño necesario será algo menor, ya que los strings no ocuparán el tamaño máximo definido.

Estimación de tamaño máximo por tablas. Se calcula multiplicando el número estimado de entradas por el tamaño máximo de cada una de las entradas:

CENTRO = 10 * 76 = 760 Bytes
DEPARTAMENTO = 4 * 48 = 192 Bytes
EMPLEADO = 1500 * 178 = 267 KBytes
CLIENTE = 100*92= 920 Bytes
PROYECTO = 100 * 62 = 620 Bytes
GASTO = 1500 empleados*100(proyectos) * 70 = 11 MBytes
HISTORIAL = 1500 * 26 = 39 KBytes
CARGO = 10*58 = 580 bytes
DESEMPEÑA = 1500 empleados*30 = 45 Kbytes
ASIGNAGASTO = 150.000 (personal gastos) * 100 proyectos *34 = 510 Mbytes
ENTRASALE = 1500 empleados * 4 entrada/salida* 365 días * 30 = 66 Mbytes
JEFE = 100 * 52 = 6 kbytes
TRABAJA = 1500 * 36 = 54 kbytes

Tamaño total = 587`414 mbytes ➔ 590 Mbytes

8. CREACION DE PROCEDIMIENTOS

Hasta el momento hemos trabajado con la base de datos de manera interactiva, es decir, el usuario introducía un comando y Oracle daba una respuesta. Esta no seria la manera correcta de gestionar una base de datos en una empresa ya que se debe dar por hecho que todos los usuarios deben conocer perfectamente el lenguaje y sus usos.

También hemos creado pequeños scripts para la gestión de la base de datos pero estos tienen limitaciones, para superar estas Oracle incorpora un gestor PL/SQL en el servidor de la base de datos y en las principales herramientas, este lenguaje basado en ADA incorpora todas las características de los lenguajes de tercera generación: variables, estructurada modular, estructuras de control, excepciones...

Estos programas se pueden almacenar en la base de datos como cualquier otro objeto de esta; de esta forma facilita a los usuarios la utilización de la base de datos. Además los programas se ejecutan en el servidor con el consiguiente ahorro en los clientes y disminución del tráfico de red.

A continuación implementaremos los diferentes procedimientos requeridos en el enunciado.

- **'ALTA CENTRO'**

  Damos en este procedimiento de alta un centro, previamente debemos buscar si existe dicho centro si existe se pueden modificar sus datos si no existe introduciremos el centro siempre y cuando el numero de centros no supere 10.

  Precondiciones:
  cif no puede tener valores null
  nombre no puede tener valores null
  ciudad no puede tener valores null
  telefono no puede tener valores null
  fax no puede tener valores null
  Postcondiciones:
  Una nueva entrada en la tabla CENTRO
  Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
  correctamente o 'error' en caso contrario.
  Una variable de salida con el código del centro si ha sido dada de alta.
  Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE ALTA_CENTRO
( var_cif VARCHAR, aux_nombre  VARCHAR, aux_ciudad VARCHAR, aux_telefono VARCHAR,
aux_fax VARCHAR, Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS

var_cod integer;
vT INTEGER;
foundcif INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de centro .
-- Si no existe  daremos de alta centro siempre que no supere el 10 centros
vT:= 0;
SELECT count(*) INTO var_cod FROM centro WHERE cif=var_cif;
IF (var_cod = 0) THEN
      -- el centro no existe podemos dar de alta mientras no supere el valor 10
vT:= 10;
SELECT COUNT(*) INTO foundcif from centro;
IF (foundcif < 10) THEN
-- insertamos centro
vT:= 20;
INSERT INTO centro (cif, nombre, ciudad, telefono, fax)
Values (var_cif, aux_nombre, aux_ciudad, aux_telefono, aux_fax);
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := var_cif;
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
```

```
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'ALTA_PROYECTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'MODIFICAR CENTRO '**

Con este procedimiento modificaremos lo datos del centro que entra como parámetro. Modificaríamos datos como nombre, ciudad, telefono, fax.

Precondiciones:
cif no puede tener valores null
nombre no puede tener valores null
ciudad no puede tener valores null
telefono no puede tener valores null
fax no puede tener valores null

Postcondiciones:
Una entrada actualizada en la tabla centro
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE MOD_CENTRO
(var_cif VARCHAR, aux_nombre VARCHAR, aux_ciudad VARCHAR, aux_telefono VARCHAR,
aux_fax VARCHAR, Rst OUT VARCHAR) AS
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
vT INTEGER;
BEGIN
-- Actualizar el centro
vT:=0;
UPDATE centro SET nombre=aux_nombre, ciudad=aux_ciudad, telefono=aux_telefono, fax=aux_fax
WHERE cif=var_cif;
--la actualizacion  se ha ejecutado correctamente
COMMIT;
Rst:='ok';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' MOD_CENTRO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'BAJA CENTRO'**

Con este procedimiento daremos de baja lo datos del centro que entra como parámetro. Es evidente que hay que controlar el resto de tablas para que la eliminación del centro no provoque errores en el resto.

Precondiciones:
cif no puede tener valores null
nombre no puede tener valores null
ciudad no puede tener valores null
telefono no puede tener valores null
fax no puede tener valores null

Postcondiciones:
Una entrada borrada en la tabla centro
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE BAJA_CENTRO
( var_cif VARCHAR, aux_nombre  VARCHAR, aux_ciudad VARCHAR, aux_telefono VARCHAR,
aux_fax VARCHAR, Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS
var_cod integer;
vT INTEGER;
foundcif INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de centro .
-- Si no existe  no es necesario dar de baja si existe si podemos darlo de baja.
 vT:= 0;
SELECT count(*) INTO var_cod FROM centro WHERE cif=var_cif;
IF (var_cod > 0) THEN
      -- el centro  existe podemos dar de baja
vT:= 10;
-- para dar de baja centro debemos tener en cuenta el resto de tablas para no
--provocar errores.primeramente daremos de baja en la entrada ENTRASALE
DELETE FROM entrasale WHERE cif=var_cif;
-- posteriormente damos de baja el centro en la tabla centro y como consecuencia
 --los departamentos de dicho centro
DELETE FROM departamento WHERE cif=var_cif;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := var_cif;
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'BAJA_CENTRO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'ALTA DEPARTAMENTO'**

Una vez dado un centro de alta procederemos a dar de alta sus departamentos debemos considerar las siguientes precondiciones y poscondiciones:

Precondiciones:
codigo_departamento no puede tener valores null
nombre no puede tener valores null

telef_infor no puede tener valores null
num_empleados no puede tener valores null
sueldo_medio no puede tener valores null
cif no puede tener valores null

Postcondiciones:
Una nueva entrada en la tabla Departamento
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario.
Una variable de salida con el código del departamento si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
      CREATE OR REPLACE
PROCEDURE ALTA_DEPARTAMENTO
(var_cod_departamento VARCHAR, aux_telef_infor VARCHAR,aux_nombre VARCHAR, aux_cif
VARCHAR, Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS
   var_cod INTEGER;
   var_coddepar INTEGER;
                  vT INTEGER;
                     foundcod INTEGER;
   Codigo_Error NUMBER;
   Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de CENTRO.
-- Si no existe NO daremos de alta departamento siempre que no supere los 40  departamentos
--(maximo 10 centros por 4 departamentos máximos en cada centro)
vT:= 0;
SELECT count(*) INTO var_cod FROM centro WHERE cif=aux_cif;
IF (var_cod > 0) THEN
      -- el centro existe podemos dar de alta departamento mientras no supere el valor 40
vT:= 10;
SELECT COUNT(*) INTO foundcod from departamento;
IF (foundcod < 40) THEN
-- insertamos departamento SI NO EXISTE DICHO DEPARTAMENTO
vT:= 20;
SELECT count(*) INTO var_coddepar FROM departamento WHERE
codigo_departamento=var_cod_departamento;
IF (var_coddepar = 0) THEN
INSERT INTO departamento(codigo_departamento,telef_infor,nombre,cif)
Values (var_cod_departamento, aux_telef_infor,aux_nombre,aux_cif);
END IF;
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_cod_departamento ';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' ALTA_ DEPARTAMENTO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'MODIFICAR DEPARTAMENTO'**

  Modificaremos lo datos del departamento que entra como parámetro. Lo realizaremos mediante el siguiente procedimiento:

Precondiciones:
codigo_departamento no puede tener valores null
nombre no puede tener valores null
telef_infor no puede tener valores null
num_empleados no puede tener valores null
sueldo_medio no puede tener valores null
cif no puede tener valores null

Postcondiciones:
Una modificacion en la tabla Departamento
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
      CREATE OR REPLACE
PROCEDURE MOD_DEPARTAMENTO
(var_cod_departamento VARCHAR, aux_telef_infor VARCHAR, aux_nombre VARCHAR, Rst OUT
VARCHAR) AS
vT INTEGER;
   Codigo_Error NUMBER;
   Msg_Error VARCHAR2(200);
BEGIN
-- Actualizar el departamento
vT:= 0;
UPDATE departamento SET telef_infor=aux_telef_infor,nombre=aux_nombre WHERE
codigo_departamento = var_cod_departamento;
--la actualizacion  se ha ejecutado correctamente
COMMIT;
Rst:= 'ok';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' MOD_DEPARTAMENTO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'BAJA DEPARTAMENTO'**

  Borraremos lo datos del departamento que entra como parámetro. Lo realizaremos mediante el siguiente procedimiento y con las siguientes precondiciones y postcondiciones:

Precondiciones:
codigo_departamento no puede tener valores null
nombre no puede tener valores null
telef_infor no puede tener valores null
num_empleados no puede tener valores null
sueldo_medio no puede tener valores null
cif no puede tener valores null

Postcondiciones:
Un borrado en la tabla Departamento
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado

correctamente o 'error' en caso contrario.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE BAJA_DEPARTAMENTO
(var_cod_departamento VARCHAR, aux_telef_infor VARCHAR,aux_nombre VARCHAR, aux_cif
VARCHAR, Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS
        var_coddepar INTEGER;
                    vT INTEGER;
                       foundcod INTEGER;
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de DEPARTAMENTO.
-- Si no existe NO daremos de BAJA departamento , si existe podremos darlo de baja
vT:= 0;
SELECT count(*) INTO var_coddepar FROM departamento WHERE codigo_departamento =
var_cod_departamento;
IF (var_coddepar > 0) THEN
      -- el departamento existe podemos dar de baja departamento mientras cumpla
     -- unos requisitos, primeramente anularemos los empleados que tengan asignados
      -- dichos departamentos
vT:= 10;
DELETE FROM empleado WHERE codigo_departamento = var_cod_departamento;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_cod_departamento ';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' BAJA_ DEPARTAMENTO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **' ALTA CARGO '**

Antes de dar de alta a empleados debemos definir los cargos posibles para los empleados, lo haremos mediante el procedimiento ALTA_CARGO en la Pág. 15 de este documento podemos encontrar más definición sobre los cargos:

Precondiciones:
codigo_cargo no puede tener valores null
sueldo_hora no puede tener valores null
Postcondiciones:
Una nueva entrada en la tabla cargos
Hay en total (en este ejemplo) de 10 cargos teniendo en cuenta que el numero de departamentos puede aumentar estos podrian incrementarse, pero puede servir de base.
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario.
Una variable de salida con el código del cargo si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE ALTA_CARGO
```

```
(var_codigo_cargo INTEGER, aux_descripcion VARCHAR, aux_sueldo_hora INTEGER, Rst OUT
VARCHAR, Rst2 OUT VARCHAR) AS

    var_cod INTEGER;
                vT INTEGER;
                foundcargo INTEGER;
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
begin
-- Obtener el  codigo del cargo. Si no existe  daremos de alta cargo
vT:= 0;
SELECT count(*) INTO var_cod FROM cargo WHERE codigo_cargo=var_codigo_cargo;
IF (var_cod = 0) THEN
      -- el cargo no existe podemos dar de alta mientras no supere el valor 10
vT:= 10;
SELECT COUNT(*) INTO foundcargo from cargo;
IF (foundcargo < 10) THEN
vT:= 20;
INSERT INTO cargo(codigo_cargo, descripcion, sueldo_hora)
Values (var_codigo_cargo, aux_descripcion, aux_sueldo_hora);
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_codigo_cargo ';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' ALTA_CARGO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **' MODIFICAR CARGO '**

Dado de alta un departamento es posible modificar sus datos para ello comprobamos una serie de requisitos y a continuación se modifican los datos.

Precondiciones:
codigo_cargo no puede tener valores null
sueldo_hora no puede tener valores null
Postcondiciones:
Una nueva entrada en la tabla cargos
Hay en total (en este ejemplo) de 10 cargos teniendo en cuenta que el numero de departamentos puede aumentar estos podrian incrementarse, pero puede servir de base.
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario.
Una variable de salida con el código del cargo si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE MOD_CARGO
(var_codigo_cargo INTEGER, aux_descripcion VARCHAR, aux_sueldo_hora INTEGER, Rst OUT
VARCHAR) AS

vT INTEGER;
```

```
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
BEGIN
-- Actualizar el cargo
vT:= 0;
UPDATE cargo SET descripcion=aux_descripcion,sueldo_hora=aux_sueldo_hora WHERE
codigo_cargo = var_codigo_cargo;
--la actualizacion  se ha ejecutado correctamente
COMMIT;
Rst:= 'ok';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' MOD_CARGO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **' ALTA GASTO '**

Los gastos se introducirán en la tabla y habrá un numero limitado básicamente estarán delimitados por la descripción aquellos que tengan descripciones parecidas serán clasificados dentro del mismo código de gasto

Precondiciones:
codigo_gasto no puede tener valores null
sueldo_descripcion no puede tener valores null
Postcondiciones:
Una nueva entrada en la tabla gastos
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE ALTA_GASTOS
(var_cod_gasto VARCHAR, aux_descripcion VARCHAR, Rst OUT VARCHAR, Rst2 OUT
VARCHAR) AS
    var_cod INTEGER;
                vT INTEGER;
                foundgasto INTEGER;
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
begin
-- Obtener el  codigo del gasto. Si no existe  daremos de alta gasto
vT:= 0;
SELECT count(*) INTO var_cod FROM gastos WHERE cod_gasto=var_cod_gasto;
IF (var_cod = 0) THEN
      -- el gasto no existe podemos dar de alta
vT:= 10;
INSERT INTO gastos(cod_gasto, descripcion)
Values (var_cod_gasto, aux_descripcion);
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_cod_gasto ';
--Se ha producido algún error inesperado
EXCEPTION
```

```
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' ALTA_GASTOS ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **' MODIFICAR GASTO '**

Los gastos se podrán modificar, más específicamente la descripción del gasto que se podrá gestionar a gusto del usuario.

Precondiciones:
codigo_gasto no puede tener valores null
descripcion no puede tener valores null
Postcondiciones:
Una nueva modificación en la tabla gastos
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE MOD_GASTO
(var_cod_gasto INTEGER, aux_descripcion VARCHAR, Rst OUT VARCHAR) AS
vT INTEGER;
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
BEGIN
-- Actualizar el gasto
vT:= 0;
UPDATE gastos SET descripcion=aux_descripcion WHERE cod_gasto = var_cod_gasto;
--la actualizacion  se ha ejecutado correctamente
COMMIT;
Rst:= 'ok';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' MOD_GASTO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **' BAJA GASTO '**

El Gasto se podrá de dar de baja, primeramente deberá darse de baja de la relación asignagasto y posteriormente de la tabla gastos, de esta manera evitamos posibles incoherencias en las relaciones de las tablas.

Precondiciones:
codigo_gasto no puede tener valores null
descripcion no puede tener valores null
Postcondiciones:

Una nueva modificación en la tabla gastos
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE BAJA_GASTOS
(var_cod_gasto VARCHAR, aux_descripcion VARCHAR, Rst OUT VARCHAR, Rst2 OUT
VARCHAR) AS
    var_cod INTEGER;
                    vT INTEGER;
                    foundgasto INTEGER;
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
begin
-- Obtener el  codigo del gasto. Si  existe  daremos de BAJA gasto
vT:= 0;
SELECT count(*) INTO var_cod FROM gastos WHERE cod_gasto=var_cod_gasto;
IF (var_cod > 0) THEN
      -- el gasto  existe podemos dar de baja
vT:= 10;
DELETE FROM gastos WHERE cod_gasto = var_cod_gasto;
DELETE FROM asignagastos WHERE cod_gasto = var_cod_gasto;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_cod_gasto ';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' BAJA_GASTOS ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **' ALTA EMPLEADO '**

Para dar de alta el empleado debemos tener en cuenta que estos deben de pertenecer a centros y un departamento en concreto. Y además debe estar muy definido el cargo del este empleado. Mostraremos una serie de precondiciones:

Precondiciones:
dni no puede tener valores null
nombre no puede tener valores null
apellidos no puede tener valores null
telefono no puede tener valores null
nss no puede tener valores null
codigo_departamento no puede tener valores null
codigo_cargo no puede tener valores null

Postcondiciones:
Una nueva alta en la tabla empleado
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado

Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE ALTA_EMPLEADO
( var_dni VARCHAR,aux_nombre VARCHAR, aux_apellidos VARCHAR, aux_telefono VARCHAR,
aux_nss VARCHAR, aux_codigo_departamento VARCHAR, aux_codigo_cargo INTEGER, Rst OUT
VARCHAR, Rst2 OUT VARCHAR) AS
var_coddni INTEGER;
var_codigo_departamento INTEGER;
var_codigo_cargo INTEGER;

                    vT INTEGER;
   Codigo_Error NUMBER;
   Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de empleado.
-- Si no existe  daremos de alta al empleado, tambien controlaremos que el empleado tengo correctos los
campos codigo cargo y codigo departamento, es decir que existan
vT:= 0;
vT:= 0;
SELECT count(*) INTO var_coddni FROM empleado WHERE dni=var_dni;
IF (var_coddni = 0) THEN
      -- el empleado no existe podemos dar de alta mientras supere unas condiciones:
      --que exista el departamento y el cargo
vT:= 10;
SELECT count(*) INTO var_codigo_departamento FROM departamento WHERE
codigo_departamento=aux_codigo_departamento;
IF (var_codigo_departamento > 0) THEN
-- insertamos empleado si existe el cargo
vT:= 20;
SELECT count(*) INTO var_codigo_cargo FROM cargo WHERE codigo_cargo=aux_codigo_cargo;
IF (var_codigo_cargo > 0) THEN
vT:= 30;
INSERT INTO empleado (dni, nombre, apellidos, telefono, nss, codigo_departamento, codigo_cargo)
Values (var_dni, aux_nombre, aux_apellidos, aux_telefono, aux_nss, aux_codigo_departamento,
aux_codigo_cargo);
END IF;
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_dni ';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' ALTA_EMPLEADO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

• MODIFICAR EMPLEADO

Modificando los datos de empleado tendremos lo siguiente:

Precondiciones:
dni no puede tener valores null
nombre no puede tener valores null
apellidos no puede tener valores null
telefono no puede tener valores null
nss no puede tener valores null

codigo_departamento no puede tener valores null
codigo_cargo no puede tener valores null

Postcondiciones:
Una nueva modificación en la tabla empleado
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE MOD_EMPLEADO
(var_dni VARCHAR, aux_nombre VARCHAR, aux_apellidos VARCHAR, aux_telefono VARCHAR,
aux_nss VARCHAR, Rst OUT VARCHAR) AS
vT INTEGER;
    Codigo_Error NUMBER;
    Msg_Error VARCHAR2(200);
BEGIN
-- Actualizar el empleado
vT:= 0;
UPDATE empleado SET
nombre=aux_nombre,apellidos=aux_apellidos,telefono=aux_telefono,nss=aux_nss WHERE dni =
var_dni;
--la actualizacion  se ha ejecutado correctamente
COMMIT;
Rst:= 'ok';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' MOD_EMPLEADO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'BAJA EMPLEADO'**

Para dar de baja a empleado hay que eliminarlo del resto de tablas para que no provoque incoherencias, primeramente daremos de baja al empleado en la relación trabaja, en la relación asignagastos y por ultimo al empleado en la tabla.

Precondiciones:
dni no puede tener valores null
nombre no puede tener valores null
apellidos no puede tener valores null
telefono no puede tener valores null
nss no puede tener valores null
codigo_departamento no puede tener valores null
codigo_cargo no puede tener valores null

Postcondiciones:
Una nueva alta en la tabla empleado
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.

Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE BAJA_EMPLEADO
( var_dni VARCHAR,aux_nombre VARCHAR, aux_apellidos VARCHAR, aux_telefono VARCHAR,
aux_nss VARCHAR, aux_codigo_departamento VARCHAR, aux_codigo_cargo INTEGER, Rst OUT
VARCHAR, Rst2 OUT VARCHAR) AS
var_coddni INTEGER;
var_codigo_departamento INTEGER;
var_codigo_cargo INTEGER;

                    vT INTEGER;
   Codigo_Error NUMBER;
   Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de empleado.
-- Si  existe  daremos de baja al empleado, tambien controlaremos que el empleado tengo correctos los
campos codigo cargo y codigo departamento, es decir que existan
vT:= 0;
vT:= 0;
SELECT count(*) INTO var_coddni FROM empleado WHERE dni=var_dni;
IF (var_coddni > 0) THEN
      -- el empleado  existe podemos dar de baja
vT:= 10;
 -- primeramente daremos de baja al empleado en la relacion trabaja, en la relacion asignagastos y por
ultimo
 -- al empleado en la tabla
DELETE FROM trabaja WHERE dni=var_dni;
DELETE FROM asignagastos WHERE dni=var_dni;
DELETE FROM empleado WHERE dni=var_dni;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := ' var_dni ';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, ' BAJA_EMPLEADO ', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'ALTA CLIENTE '**

Los clientes son los que solicitan a la empresa la creación de proyectos estos serán almacenados debidamente siguiendo las condiciones siguientes:

Precondiciones:
cod_cli no puede tener valores null
nombre no puede tener valores null
apellidos no puede tener valores null
telefono no puede tener valores null

Postcondiciones:
Una nueva alta en la tabla clientes
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.

Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE ALTA_CLIENTE
( var_cod_cli VARCHAR, aux_nombre  VARCHAR, aux_apellidos VARCHAR, aux_telefono
VARCHAR, Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS
var_codcli integer;
vT INTEGER;
foundcif INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de cliente .
-- Si no existe  daremos de alta cliente
vT:= 0;
SELECT count(*) INTO var_codcli FROM cliente WHERE cod_cli=var_cod_cli;
IF (var_codcli = 0) THEN
      -- el cliente no existe podemos dar de alta
vT:= 10;
INSERT INTO cliente (cod_cli, nombre,apellidos,telefono)
Values (var_cod_cli, aux_nombre, aux_apellidos, aux_telefono);
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_cod_cli';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'ALTA_CLIENTE', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'ALTA PROYECTO'**

El procedimiento ALTA_PROYECTO se utiliza para dar de alta un determinado proyecto en el sistema. Al dar de alta un proyecto este debe ser asignado a un cliente (esto es obligatorio por cada cliente habrá 1 proyecto), por lo tanto debemos buscar el cliente en cuestión, si existe se modifican los datos del proyecto para ese cliente. Si no existe debemos darlo de alta, a su vez, al dar de alta un proyecto es necesario asignarle un jefe de proyectos, este se buscara dentro de la lista de empleados y si no lo encuentra de la misma manera lo daremos de alta. Debemos tener en cuenta una serie de precondiciones y poscondiciones.

Precondiciones:
cod_proyecto no puede tener valores null
f_inicio no puede tener valores null
f_fin no puede tener valores null
horas no puede tener valores null

Postcondiciones:
Una nueva entrada en la tabla PROYECTO
Una nueva entrada en la tabla CLIENTE si el cliente no existía en la base de datos
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.

Una variable de salida con el código del proyecto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
        CREATE OR REPLACE
PROCEDURE ALTA_PROYECTO
( var_cod_proyecto VARCHAR, var_cod_cli  VARCHAR, var_dni_jefep VARCHAR, aux_nombre
VARCHAR,aux_presupuesto INTEGER, aux_f_inicio DATE,aux_f_fin DATE,aux_horas INTEGER,
Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS
var_codpro integer;
var_codcli INTEGER;
var_coddni INTEGER;
var_codcargo INTEGER;
vT INTEGER;
foundcif INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de proyecto .
-- Si no existe  daremos de alta proyecto siempre y cuando se cumpla una serie de condiciones
vT:= 0;
SELECT count(*) INTO var_codpro FROM proyecto WHERE cod_proyecto=var_cod_proyecto;
IF (var_codpro = 0) THEN
      -- el proyecto no existe podemos dar de alta segun condiciones
      --debe de existir el cliente
vT:= 10;
SELECT count(*) INTO var_codcli FROM cliente WHERE cod_cli=var_cod_cli;
IF (var_codcli > 0) THEN
  --el cliente existe ahora buscaremos que jefeproyecto exista y nos aseguraremos que sea jefe de
  -- proyectos buscando al empleado en cargos con codigo jefe de proyecto que tendra valor 1
  SELECT count(*) INTO var_coddni FROM empleado WHERE dni=var_dni_jefep AND
codigo_cargo=1;
IF (var_coddni > 0) THEN
  --existe empleado y buscaremos si es jefe de proyecto en cargo
     INSERT INTO proyecto (cod_proyecto, cod_cli,dni_jefep,nombre,presupuesto,f_inicio,f_fin,horas)
     Values (var_cod_proyecto,var_cod_cli,var_dni_jefep,aux_nombre,
aux_presupuesto,aux_f_inicio,aux_f_fin, aux_horas);
END IF;
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_cod_proyecto';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'ALTA_PROYECTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'BAJA PROYECTO'**

Precondiciones:
cod_proyecto no puede tener valores null
f_inicio no puede tener valores null
f_fin no puede tener valores null
horas no puede tener valores null

Postcondiciones:
Una nueva baja en la tabla PROYECTO

Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del proyecto si ha sido dada de baja.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE BAJA_PROYECTO
( var_cod_proyecto VARCHAR, var_cod_cli  VARCHAR, var_dni_jefep VARCHAR, aux_nombre
VARCHAR,aux_presupuesto INTEGER, aux_f_inicio DATE,aux_f_fin DATE,aux_horas INTEGER,
Rst OUT VARCHAR, Rst2 OUT VARCHAR) AS
var_codpro integer;
var_codcli INTEGER;
var_coddni INTEGER;
var_codcargo INTEGER;
vT INTEGER;
foundcif INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de proyecto .
-- Si  existe  daremos de baja proyecto siempre y cuando se cumpla una serie de condiciones
vT:= 0;
SELECT count(*) INTO var_codpro FROM proyecto WHERE cod_proyecto=var_cod_proyecto;
IF (var_codpro > 0) THEN
      -- el proyecto  existe podemos dar de baja primeramente en la relacion trabaja (empleado proyecto)
 DELETE FROM trabaja WHERE cod_proyecto=var_cod_proyecto;
DELETE FROM proyecto WHERE cod_proyecto=var_cod_proyecto;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_cod_proyecto';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'BAJA_PROYECTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'EMPLEADO TRABAJA PROYECTO'**

Esta es una asignación muy importante dentro de la base de datos en ella nos indica durante que tiempo ha estado trabajando el empleado en un determinado proyecto:

Precondiciones:
cod_proyecto no puede tener valores null
f_inicio no puede tener valores null
f_fin no puede tener valores null
horas no puede tener valores null

Postcondiciones:
Una nueva baja en la tabla PROYECTO
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del proyecto si ha sido dada de baja.

Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE EMPLEADO_TRABAJA_PROYECTO
( var_cod_proyecto VARCHAR, var_dni VARCHAR,aux_f_inicio DATE, aux_f_fin DATE, Rst OUT
VARCHAR, Rst2 OUT VARCHAR) AS
var_codpro integer;
var_coddni INTEGER;
var_coddnitrab INTEGER;
deparjefepro VARCHAR(10);
depemple VARCHAR(10);
vT INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de proyecto .
-- Si  existe  asignaremos empleado
vT:= 0;
SELECT count(*) INTO var_codpro FROM proyecto WHERE cod_proyecto=var_cod_proyecto;
IF (var_codpro > 0) THEN
      -- el proyecto existe podemos asignarle trabajador ya que estos pueden estar asignados
      -- a mas de un proyecto debe de existir el trabajador y ademas deben pertenecer al mismo
      -- departamento que el jefe de proyecto
vT:= 10;
 SELECT count(*) INTO var_coddni FROM empleado WHERE dni=var_dni;
IF (var_coddni > 0) THEN
 --existe empleado y buscaremos si ese empleado ya esta trabajando en ese proyecto
vT:= 20;
 SELECT count(*) INTO var_coddnitrab FROM trabaja WHERE dni=var_dni AND
cod_proyecto=var_cod_proyecto;
IF (var_coddni = 0) THEN
 --no trabaja en ese proyecto actualmente y por lo tanto comprobaremos a que departamento esta
asignado
 --el jefe de proyecto y compararemos con el departamento del empleado a asignar
 SELECT codigo_departamento INTO deparjefepro FROM proyecto,empleado WHERE dni_jefep=dni
AND cod_proyecto=var_cod_proyecto;
 SELECT codigo_departamento INTO depemple FROM empleado WHERE dni=var_dni;
IF (deparjefepro = depemple) THEN
    INSERT INTO trabaja (cod_proyecto,dni,f_inicio,f_fin)
    Values (var_cod_proyecto,var_dni,aux_f_inicio,aux_f_fin);
END IF;
END IF;
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_dni';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'EMPLEADO_TRABAJA_PROYECTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'EMPLEADO DESASIGNA PROYECTO'**

Precondiciones:
cod_proyecto no puede tener valores null
f_inicio no puede tener valores null
f_fin no puede tener valores null
horas no puede tener valores null

Postcondiciones:
Una nueva baja en la tabla PROYECTO
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del proyecto si ha sido dada de baja.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
        CREATE OR REPLACE
PROCEDURE EMPLEADO_DESASIGNA_PROYECTO
( var_cod_proyecto VARCHAR, var_dni VARCHAR,aux_f_inicio DATE, aux_f_fin DATE, Rst OUT
VARCHAR, Rst2 OUT VARCHAR) AS
var_codpro integer;
var_coddni INTEGER;
var_coddnitrab INTEGER;
deparjefepro VARCHAR(10);
depemple VARCHAR(10);
vT INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- Obtener el  identificador de proyecto .
-- Si  existe  desasignaremos empleado
vT:= 0;
SELECT count(*) INTO var_codpro FROM proyecto WHERE cod_proyecto=var_cod_proyecto;
IF (var_codpro > 0) THEN
      -- el proyecto existe podemos desasignarle trabajador ya que estos pueden estar asignados
      -- a mas de un proyecto. debe de existir el trabajador y ademas deben pertenecer al mismo
      -- departamento que el jefe de proyecto
vT:= 10;
  SELECT count(*) INTO var_coddni FROM empleado WHERE dni=var_dni;
IF (var_coddni > 0) THEN
  --existe empleado
   DELETE FROM trabaja WHERE dni=var_dni;
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_dni';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'EMPLEADO_TRABAJA_PROYECTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'EMPLEADO ASIGNA GASTO'**

Precondiciones:
cod_gasto no puede tener valores null
cantidad no puede tener valores null
dni no puede tener valores null
f_gasto no puede tener valores null

Postcondiciones:
Una nueva asignacion en la tabla gasto

Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.
Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE EMPLEADO_ASIGNA_GASTO
( var_cod_gasto VARCHAR, aux_cantidad INTEGER, var_dni VARCHAR,aux_f_gasto DATE, Rst
OUT VARCHAR, Rst2 OUT VARCHAR) AS
var_codgasto integer;
var_coddni INTEGER;
vT INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- comprobamos que existe el tipo de gasto .
-- Si  existe  asignaremos gasto a empleado
vT:= 0;
SELECT count(*) INTO var_codgasto FROM gastos WHERE cod_gasto=var_cod_gasto;
IF (var_codgasto > 0) THEN
      -- el tipo de gasto existe  podemos asignarle empleado, comprobaremos que el empleado esta
asignado
      -- a un proyecto
vT:= 10;
 SELECT count(*) INTO var_coddni FROM trabaja WHERE dni=var_dni;
IF (var_coddni > 0) THEN
 --existe empleado trabajando en proyecto con lo que podemos asignarle gasto
vT:= 20;
    INSERT INTO asignagastos (cod_gasto,cantidad,dni,f_gasto)
    Values (var_cod_gasto,aux_cantidad,var_dni,aux_f_gasto);
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_dni';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'EMPLEADO_ASIGNA_GASTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

• 'EMPLEADO DESASIGNA GASTO'

Precondiciones:
cod_gasto no puede tener valores null
cantidad no puede tener valores null
dni no puede tener valores null
f_gasto no puede tener valores null

Postcondiciones:
Una nueva asignacion en la tabla gasto
Una variable de salida que contendrá el valor 'ok' si el procedimiento se ha ejecutado
Correctamente o 'error' en caso contrario.
Una variable de salida con el código del gasto si ha sido dada de alta.

Una nueva entrada en la tabla AVISOS_ERRORES si la variable de salida contiene 'error'

```
CREATE OR REPLACE
PROCEDURE EMPLEADO_DESASIGNA_GASTO
( var_cod_gasto VARCHAR, aux_cantidad INTEGER, var_dni VARCHAR,aux_f_gasto DATE, Rst
OUT VARCHAR, Rst2 OUT VARCHAR) AS
var_codgasto integer;
var_coddni INTEGER;
vT INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- comprobamos que existe el empleado trabajando en proyecto con lo que podemos desasignarle gasto .
vT:= 0;
 --existe empleado trabajando en proyecto con lo que podemos desasignarle gasto
  SELECT count(*) INTO var_coddni FROM trabaja WHERE dni=var_dni;
IF (var_coddni > 0) THEN
     DELETE FROM asignagastos  WHERE dni=var_dni;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_dni';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'EMPLEADO_DESASIGNA_GASTO', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

- **'EMPLEADO ENTRASALE'**

Este procedimiento controla los horarios realizados por los trabajdores de esta forma se tiene el control de horas de trabajo en un determinado proyeto o en el centro.

```
CREATE OR REPLACE
PROCEDURE EMPLEADO_ENTRASALE
( var_cif VARCHAR, var_dni VARCHAR,aux_h_inicio DATE,aux_h_fin DATE, Rst OUT VARCHAR,
Rst2 OUT VARCHAR) AS
var_codcif integer;
var_coddni INTEGER;
var_depcen VARCHAR(10);
var_depemple VARCHAR(10);
vT INTEGER;
Codigo_Error NUMBER;
Msg_Error VARCHAR2(200);
begin
-- comprobamos que existe el centro .
-- Si  existe  asignaremos entrasale a empleado
vT:= 0;
SELECT count(*) INTO var_codcif FROM centro WHERE cif=var_cif;
IF (var_codcif > 0) THEN
      -- el centro existe  podemos asignarle entrasale a empleado, comprobaremos que el empleado existe
y
      -- que ademas el departamento del empleado pertenece a ese centro
vT:= 10;
  SELECT count(*) INTO var_coddni FROM empleado WHERE dni=var_dni;
IF (var_coddni > 0) THEN
  --existe empleado trabajando en proyecto con lo que podemos asignarle gasto
vT:= 20;
  --comprobaremos que el departamento del centro es igual que de empleado
```

```
    SELECT cif INTO var_depcen FROM centro,empleado WHERE dni=var_dni AND cif=var_cif;
    SELECT codigo_departamento INTO var_depemple FROM empleado WHERE dni=var_dni;
    IF (var_depcen = var_depemple) THEN
      INSERT INTO entrasale (cif,dni,h_inicio,h_fin)
      Values (var_cif,var_dni,aux_h_inicio,aux_h_fin);
END IF;
END IF;
END IF;
-- se ha ejecutado correctamente
COMMIT;
--Guardaremos los valores out para ver posibles errores y poder mostrar mensajes
Rst:= 'ok';
Rst2 := 'var_dni';
--Se ha producido algún error inesperado
EXCEPTION
WHEN OTHERS THEN
Codigo_Error := SQLCODE;
Msg_Error := SUBSTR(SQLERRM,1,200);
ROLLBACK;
--Se guarda dentro de la tabla avisos_errores el código del error y el mensaje del error
INSERT INTO errores
(codigo_error,msg_error,procedimiento_ejecutado, linea, fecha_hora)
VALUES
(Codigo_Error, Msg_Error, 'EMPLEADO_ENTRASALE', vT, sysdate);
COMMIT;
Rst:= 'error';
End;
```

•' MOVER EMPLEADO '

Lo ideal para mover un empleado es usar los procedimientos anteriores para realizar el cambio, podríamos pensar que la mejor manera es eliminar el empleado y volver a crearlo, pero esto no es correcto ya que perderíamos los datos que teníamos hasta el momento de dicho empleado.

Una alternativa coherente y mas correcta es crear una tabla auxiliar que podremos denominar “auxempleado” en la cual guardaremos momentáneamente al empleado que queremos cambiar de departamento, centro o cargo. Es decir, el proceso es el siguiente, guardamos al empleado en la tabla auxiliar una vez guardado eliminamos al empleado de la tabla con lo cual deja de estar asignado a departamentos, cargos y centro es en ese momento cuando volvemos a dar de alta al empleado que tenemos en la tabla auxiliar de esta manera guardamos sus datos y no perdemos su historial ya que de este nunca se borra. Ya que una de las principales características de esta base de datos es que conservemos valores de todos los empleados de la empresa.

TABLA_AUXLIAR

```
create table tabla_auxiliar(
dni varchar (10) not null,
nombre varchar (32) not null,
apellidos varchar (50) not null,
telefono varchar (20),
nss varchar (20) not null,
primary key(dni),
```

Después de introducir en tabla podemos dar de alta de nuevo asignando nuevo cargo departamento y centro. Teniendo en cuenta que dicho empleado también esta almacenado dentro de la tabla historial no perderemos datos de este en su trayecto en la empresa. MIRAR el procedimiento ALTA_EMPLEADO.

•Consulta trabajador

Para consultar todos los datos del empleado o trabajador tenemos posibilidad de gestionarla mediante la tabla HISTORIAL. Esta tabla es muy importante dentro del

diseño de la base de datos en ella almacenamos todos los datos necesarios para las consultas de los empleados, a partir de esta y mediante el DNI de los empleados podemos averiguar cuanto a gastado cada departamento haciendo consultas reiterativas. Veamos los datos de la tabla:

- num_historial
- DNI
- gastos_total
- h_trabajadas
- Dias_trabajados

Por cada empleado tendremos una entrada, en cada una de estas filas mostraremos los gastos totales del empleado según las horas trabajadas y los gastos de este empleado asignados al proyecto o proyectos que ha realizado durante su carrera y los días totales que ha estado trabajando o lleva trabajando en la empresa. Además permitirá hacer consultas de todo tipo.

9. SCRIPT ELIMINACION TABLAS Y PROCEDIMIENTOS

```
--BORRADO PROCEDIMIENTOS
drop PROCEDURE ALTA_CENTRO
drop PROCEDURE MOD_CENTRO
drop PROCEDURE BAJA_CENTRO
drop PROCEDURE ALTA_DEPARTAMENTO
drop PROCEDURE MOD_DEPARTAMENTO
drop PROCEDURE BAJA_DEPARTAMENTO;
drop PROCEDURE ALTA_CARGO;
drop PROCEDURE MOD_CARGO;
drop PROCEDURE ALTA_GASTO;
drop PROCEDURE MOD_GASTO;
drop PROCEDURE BAJA_GASTO;
drop PROCEDURE ALTA_EMPLEADO;
drop PROCEDURE MOD_EMPLEADO;
drop PROCEDURE BAJA_EMPLEADO;
drop PROCEDURE ALTA_CLIENTE;
drop PROCEDURE ALTA_PROYECTO;
drop PROCEDURE BAJA_PROYECTO;
drop PROCEDURE EMPLEADO_TRABAJA_PROYECTO;
drop PROCEDURE EMPLEADO_DESASIGNA_PROYECTO;
drop PROCEDURE EMPLEADO_ASIGNA_GASTO;
drop PROCEDURE EMPLEADO_DESASIGNA_GASTO;
drop PROCEDURE EMPLEADO_ENTRASALE;

--BORRADO TABLAS
drop table errores;
drop table historial;
drop table asignagastos;
drop table gastos;
drop table trabaja;
drop table proyecto;
drop table cliente;
drop table entrasale;
drop table desempeña;
drop table empleado;
drop table cargo;
drop table departamento;
drop table centro;
commit;
```

10. GENERACIÓN DE DATOS ALEATORIOS

Para la generacion de datos aleatorios he creado un pequeño programa en JAVA que permite introducir datos de forma aleatoria mediante los procedimientos.Dicho programa controla las inserciones ya que gran parte de comprobación de los datos se realiza en los procedimientos.

```
/*
 * COracle.java
 *
```

```
 * Created on 12 de diciembre de 2004, 17:21
 */
import java.sql.*;
import java.util.*;
/**
 *
 * @author  apren
 */
public class COracle {
    public int azar(int min,int max){
        return (int)(max*Math.random()+min);
    }
    public long azarfecha(long min,long max){
        return (long)(max*Math.random()+min);
    }
    public void rellenar(){
    //CONSTANTES
        final int NDEPxCENTRO = 4;
        final int NCENTROS = 10;
        final int NEMPLExDEP = 40;
        final int NCLIENTES = 100;
        final int NPROYECTOS = 150;

        //Variables utiles
        Vector departamentos = new Vector();
        Vector centros = new Vector();
        Vector jefesp = new Vector();
        Vector clientes = new Vector();
        Vector empleados = new Vector();
        Vector proyectos = new Vector();

        Connection con;
        Statement sentencia;
        ResultSet result;

        System.out.println("Iniciando...");
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }catch (Exception e){
            System.out.println("No se pudo cargar el driver Oracle..oohhh");
            return;
        }
        try{
            con =
DriverManager.getConnection("jdbc:oracle:thin:@192.168.0.4:1521:proyecto","system","goasito");
            sentencia = con.createStatement();
            DatabaseMetaData meta = con.getMetaData();
            result = sentencia.executeQuery("Select * FROM SYSTEM.centro");
            while (result.next()){
                String cif = result.getString("cif");
                System.out.println(cif);
            }

            //CENTROS Y DEPARTAMENTOS//////////////////////////////////////////////
           ////////////////////////////////////////////////
            //RELLENO CENTROS Y DEPARTAMENTOS
            CallableStatement stmt;
            CallableStatement stmt2;
            System.out.println("Cargando centros y departamentos...");
            stmt = con.prepareCall("{ call ALTA_CENTRO(?, ?, ?, ?, ?, ?, ?) }");
```

```
stmt2 = con.prepareCall("{ call ALTA_DEPARTAMENTO(?, ?, ?, ?, ?, ?) }");
for (int i = 0; i < NCENTROS; i++) {
   stmt.setString(1,"CifCentro"+i);
   stmt.setString(2,"Nombre"+i);
   stmt.setString(3,"Ciudad"+i);
   stmt.setString(4,"Tlf"+i);
   stmt.setString(5,"Fax"+i);
   stmt.registerOutParameter(6, Types.VARCHAR);
   stmt.registerOutParameter(7, Types.VARCHAR);
   stmt.execute();
   //Se añade a la lista de centros cargados
   centros.add("CifCentro"+i);
   for(int j = 0;j < NDEPxCENTRO;j++){
       stmt2.setString(1,"CodDep"+i+j);
       stmt2.setString(2,"Telefono"+i+j);
      stmt2.setString(3,"Nombre"+i+j);
      stmt2.setString(4,"CifCentro"+i); //CifCentro
       stmt2.registerOutParameter(5, Types.VARCHAR);
      stmt2.registerOutParameter(6, Types.VARCHAR);
      stmt2.execute();
      //Añado a lista de departamentos
      departamentos.add("CodDep"+i+j);
   }
}
//CARGOS//////////////////////////////////////////////
//////////////////////////////////////////////////////
 /* //Borrado completo cargos
 try{
    System.out.println("Borrando cargos...");
    int estado = sentencia.executeUpdate("DELETE * FROM SYSTEM.cargos");
    System.out.println("Completado.");
 }catch (Exception x){
    System.out.println(x.getMessage());
    return;
 }*/
 //RELLENO Cargos
 System.out.println("Cargando cargos de prueba...");
 stmt = con.prepareCall("{ call ALTA_CARGO(?, ?, ?, ?, ?) }");
 String[] cargos = new String[11];
 cargos[1]="Jefe Departamento";
 cargos[2]="Jefe desarrollo";
 cargos[3]="jefe mantenimiento";
 cargos[4]="secretaria";
 cargos[5]="desarrollador";
 cargos[6]="diseñador";
 cargos[7]="Servicio limpieza";
 cargos[8]="gestion";
 cargos[9]="Administracion";
 cargos[10]="Jefe gestion";

 for (int i = 1; i < 11; i++) {
    stmt.setInt(1,i);//Codigoo cargo
    stmt.setString(2,cargos[i]);
    stmt.setInt(3,12+i);//Sueldo hora
    stmt.registerOutParameter(4, Types.VARCHAR);
    stmt.registerOutParameter(5, Types.VARCHAR);

    stmt.execute();
 }
 //GASTOS//////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////
/*  //Borrado completo gastos
 try{
    System.out.println("Borrando gastos...");
    int estado = sentencia.executeUpdate("DELETE * FROM SYSTEM.gastos");
    System.out.println("Completado.");
 }catch (Exception x){
    System.out.println(x.getMessage());
    return;
 }*/
 //RELLENO GASTOS
 System.out.println("Cargando tipificación de gastos de prueba...");
 stmt = con.prepareCall("{ call ALTA_GASTOS(?, ?, ?, ?) }");

 for (int i = 10; i < 21; i++) {
    stmt.setString(1,"CodGasto"+i);
    stmt.setString(2,"Descripcion"+i);
    stmt.registerOutParameter(3, Types.VARCHAR);
    stmt.registerOutParameter(4, Types.VARCHAR);

    stmt.execute();
    //System.err.println("Loop[" + i + "] = " + stmt.getString(2));
 }

  /*//Borrado completo empleados
 try{
    System.out.println("Borrando empleados...");
    int estado = sentencia.executeUpdate("DELETE * FROM SYSTEM.empleados");
    System.out.println("Completado.");
 }catch (Exception x){
    System.out.println(x.getMessage());
    return;
 }*/
 //RELLENO EMPLEADOS
 System.out.println("Cargando empleados...");
 stmt = con.prepareCall("{ call ALTA_EMPLEADO(?, ?, ?, ?, ?, ?, ?, ?, ?) }");
 int ncargo = 0;

 for (int i = 1; i < departamentos.size(); i++) {
    boolean jproyecto = false;
    String[] empledepar = new String[2];
    for(int j=1;j < NEMPLExDEP;j++){
    stmt.setString(1,"DNI"+i+j);
    stmt.setString(2,"Nombre"+i+j);
    stmt.setString(3,"Apell"+i+j);
    stmt.setString(4,"6754"+i+j); //telefono
    stmt.setString(5,"898-"+i+j); //seg social
    String depar = (String)departamentos.get(i);
    empledepar[0]="DNI"+i+j;
    empledepar[1]=depar;
    stmt.setString(6,depar); //Cod departamento
    if (jproyecto)
              ncargo = azar(2,cargos.length);
    else
          ncargo = azar(1,cargos.length);

    if (ncargo == 1) {
       jproyecto = true;
       jefesp.add(empledepar);
    }else
```

```
            empleados.add(empledepar);

        stmt.setInt(7,ncargo);// codigo cargo
        stmt.registerOutParameter(8, Types.VARCHAR);
        stmt.registerOutParameter(9, Types.VARCHAR);

        stmt.execute();
        //System.err.println("Loop[" + i + "] = " + stmt.getString(2));
        }
    }


    //RELLENO CLIENTES
    System.out.println("Cargando clientes con pasta...");
    stmt = con.prepareCall("{ call ALTA_CLIENTE(?, ?, ?, ?, ?, ?) }");

    for (int i = 0; i < NCLIENTES; i++) {

        stmt.setString(1,"CCliente"+i);
        stmt.setString(2,"NCliente"+i);
        stmt.setString(3,"ACliente"+i);
        stmt.setString(4,"TCliente"+i); //telefono
        stmt.registerOutParameter(5, Types.VARCHAR);
        stmt.registerOutParameter(6, Types.VARCHAR);
        clientes.add("CCliente"+i);
        stmt.execute();
        //System.err.println("Loop[" + i + "] = " + stmt.getString(2));

    }

    //RELLENO PROYECTOS
    System.out.println("Cargando proyectos...");
    stmt = con.prepareCall("{ call ALTA_PROYECTO(?, ?, ?, ?, ?, ?, ?, ?, ?, ?) }");

    for(int j=1;j < NPROYECTOS;j++){
        stmt.setString(1,"CODPRO"+j);
        String cli = (String)clientes.get(azar(0,NCLIENTES-1));
        System.out.println(cli);
        stmt.setString(2,cli);
        String jefecito = (String)((String[])jefesp.get(azar(0,jefesp.size()-1)))[0];
        System.out.println(jefecito);
        stmt.setString(3,jefecito);
        stmt.setString(4,"NombreProy"+j); //telefono
        stmt.setInt(5,50000*j); //presupuesto

        long tinicio = azarfecha(600000000000L,1700000000000L);
        long tfinal=tinicio*2;
        //stmt.setDa
        stmt.setDate(6,new java.sql.Date(tinicio)); //fecha inicio
        stmt.setDate(7,new java.sql.Date(tfinal)); //fecha fin
        stmt.setInt(8,azar(100,2000)); //hora
        stmt.registerOutParameter(9, Types.VARCHAR);
        stmt.registerOutParameter(10, Types.VARCHAR);
        String[] pro = new String[4];
        pro[0] = "CODPRO"+j;
        pro[1] = jefecito;
        pro[2] = Long.toString(tinicio);
        pro[3] = Long.toString(tfinal);
        proyectos.add(pro);
        stmt.execute();
```

```
            //System.err.println("Loop[" + i + "] = " + stmt.getString(2));
        }


        //RELLENO TRABAJA EMPLEADO
        /* System.out.println("Cargando trabajo para empleados...");
        stmt = con.prepareCall("{ call EMPLEADO_TRABAJA_PROYECTO(?, ?, ?, ?) }");

        for(int j=0;j < proyectos.size();j++){

            stmt.setString(1,"CODPRO"+j);
            stmt.setString(2,"DNIVAR");
            long tinicio = azarfecha(600000000000L,1700000000000L);
            long tfinal=tinicio*2;
            //stmt.setDa
            stmt.setDate(3,new java.sql.Date(tinicio)); //fecha inicio
            stmt.setDate(4,new java.sql.Date(tinicio)); //fecha inicio
            stmt.registerOutParameter(4, Types.VARCHAR);
            stmt.registerOutParameter(5, Types.VARCHAR);

            stmt.execute();
            //System.err.println("Loop[" + i + "] = " + stmt.getString(2));
        }


        //RELLENO empleados gastos
        System.out.println("Cargando trabajo para empleados...");
        stmt = con.prepareCall("{ call EMPLEADO_GASTO(?, ?, ?, ?, ?) }");

        for(int j=1;j < NPROYECTOS;j++){
            stmt.setString(1,"CODgasto var"+j);
            stmt.setString(2,"cantidad int");
            stmt.setString(2,"dni var");

            long tinicio = azarfecha(600000000000L,1700000000000L);
            long tfinal=tinicio*2;
            //stmt.setDa
            stmt.setDate(3,new java.sql.Date(tinicio)); //fecha gasto
            stmt.registerOutParameter(4, Types.VARCHAR);
            stmt.registerOutParameter(5, Types.VARCHAR);

            stmt.execute();
            //System.err.println("Loop[" + i + "] = " + stmt.getString(2));
        }

        */
        System.out.println("Fin");
    }catch(SQLException x){
        System.out.println("sql: " + x.getMessage());
    }catch (Exception y){
        System.out.println("error" + y.getMessage());
    }
}

static public void main(String[] args){

   COracle mioracle = new COracle();
   mioracle.rellenar();
}
```

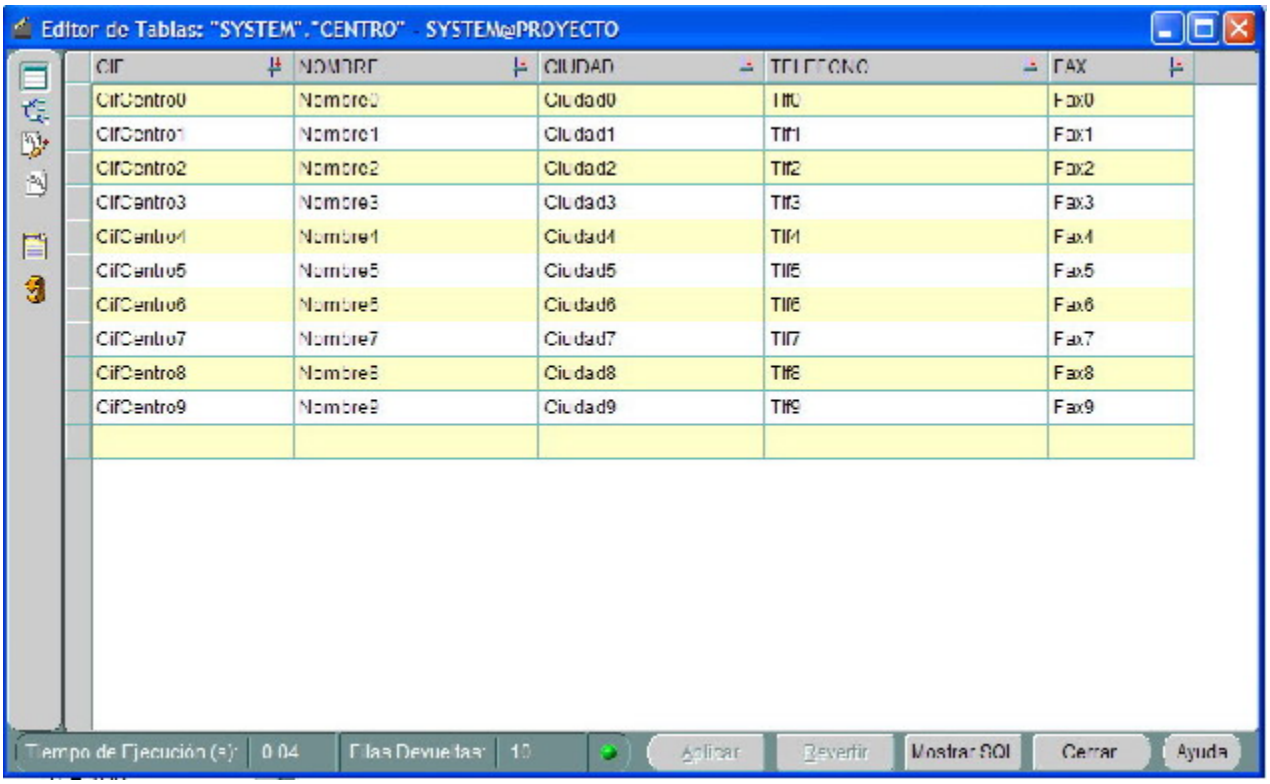
}

Para poder simular la información que se generaría se utilizan las siguientes premisas:

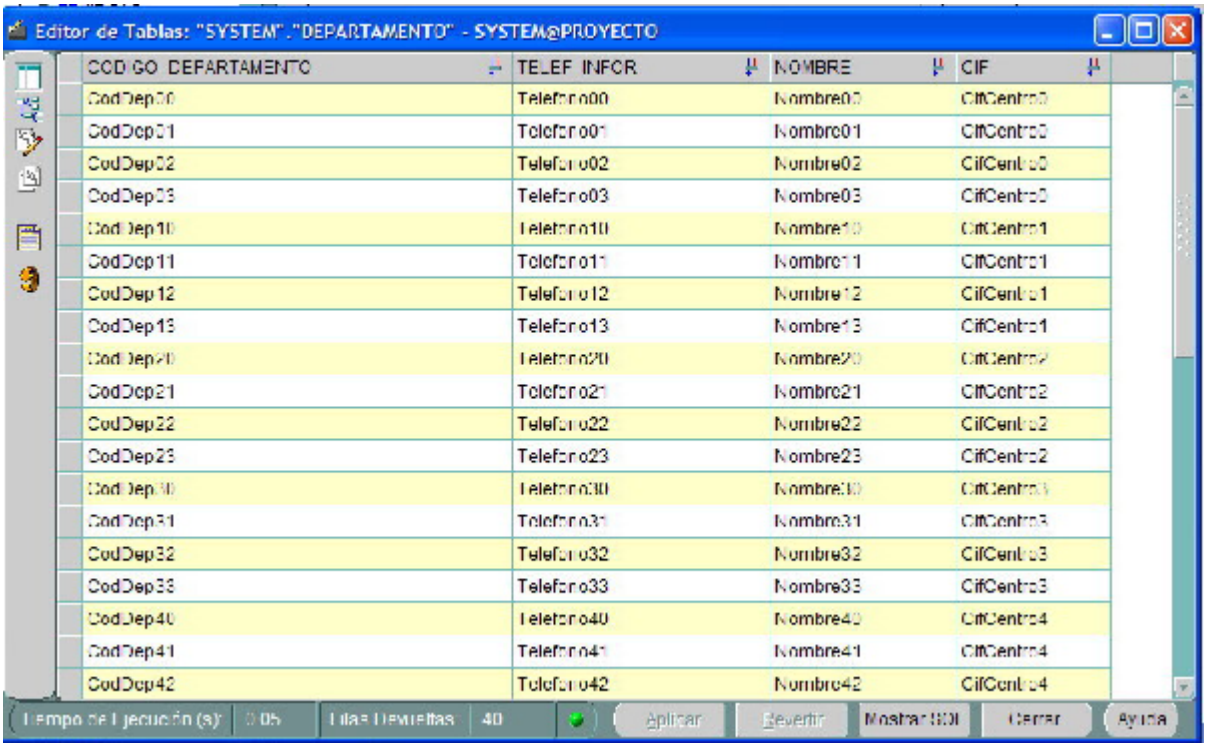
- Se toma como base numero de centros 10.
- Numero de departamentos por centro 4
- Numero de empleados por departamento 40
- Numero de clientes 100
- Numero de proyectos 150
- Los jefes de departamentos(jefes de proyectos) tienen como codigo de cargo 1.
- Los datos se generan utilizando identificadores numéricos, por ejemplo NOMBRE1, NOMBRE2, NOMBRE3, etc
- Para que los datos introducidos en la base de datos sean consistentes, la aplicación de generación de datos aleatorios utiliza los procedimientos SQL

A modo de ejemplo podemos ver en los siguientes pantallazas datos aleatorios introducidos en las tablas:

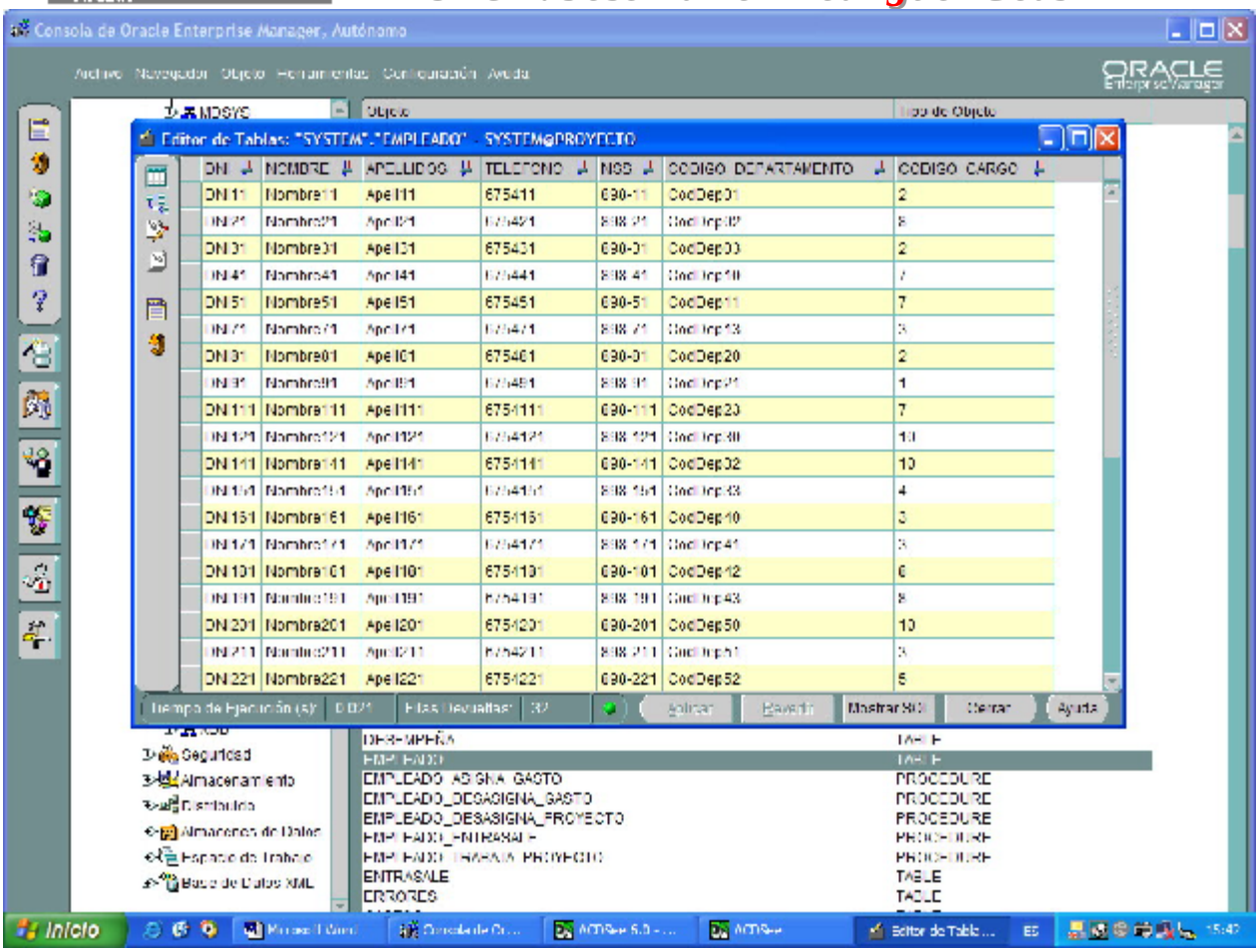
CENTROS



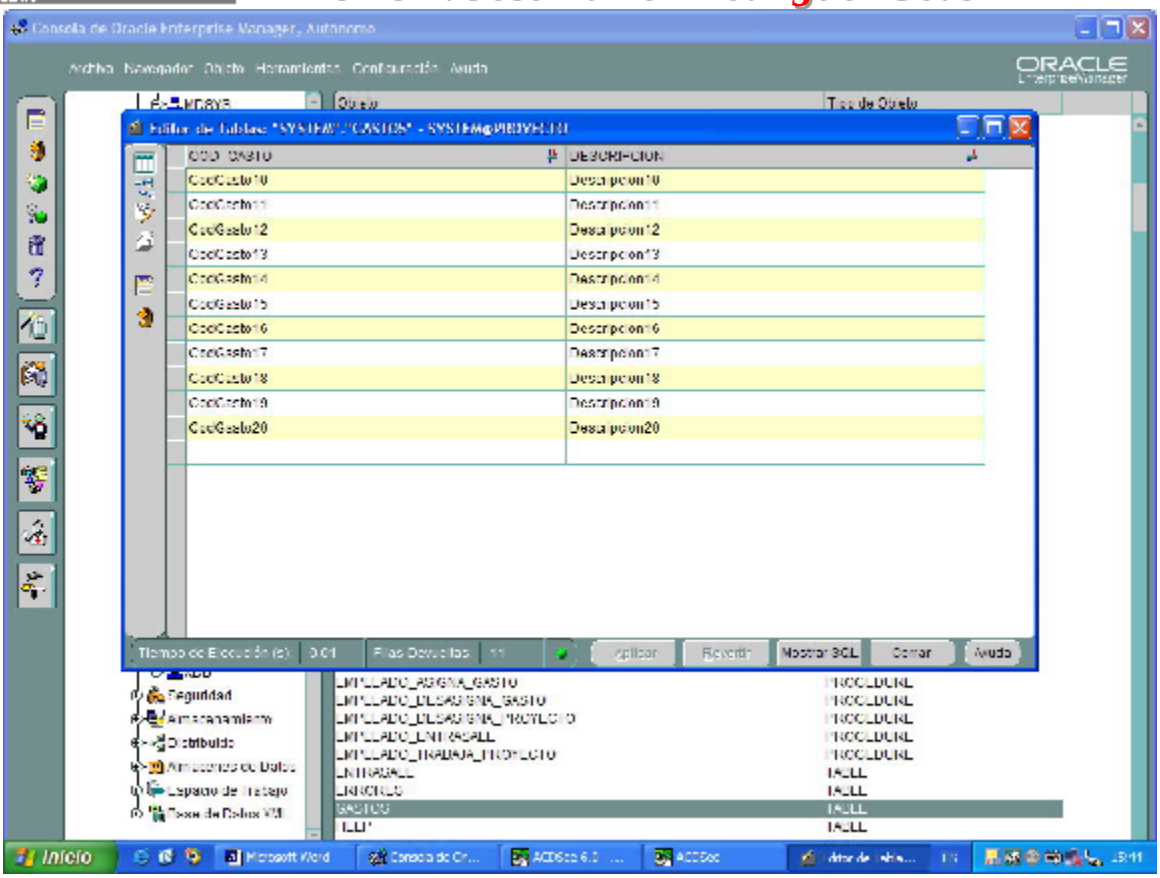
DEPARTAMENTOS



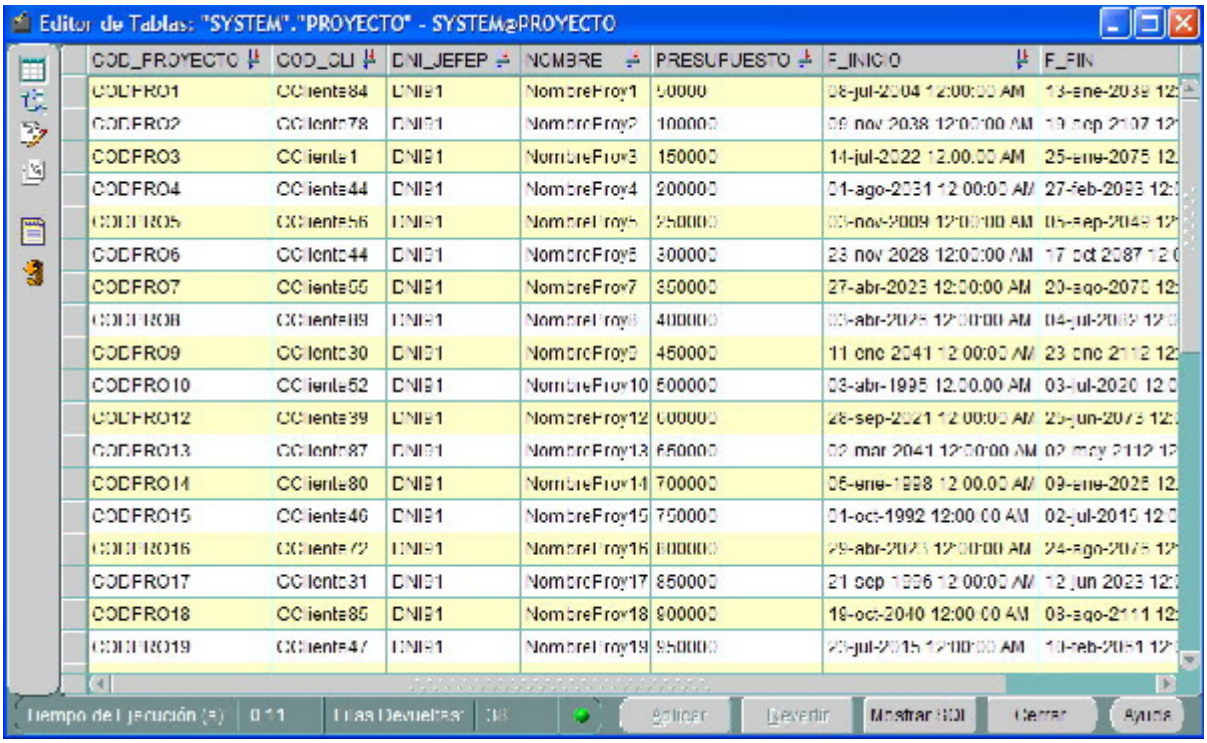
EMPLEADOS


Consola de Oracle Enterprise Manager, Autónomo
Editor de Tablas: "SYSTEM"."EMPLEADO" - SYSTEM@PROYECTO
DNI
NOMBRE
APELLIDOS
TELEFONO
NSS
CODIGO_DEPARTAMENTO
CODIGO_CARGO
Seguridad
Almacenamiento
Distribución
Almacenes de Datos
Espacio de Trabajo
Base de Datos XML
EMPLEADO_ASIGNA_GASTO
EMPLEADO_DESASIGNA_GASTO
EMPLEADO_DESASIGNA_PROYECTO
ENTRASALE
ERRORES
Inicio

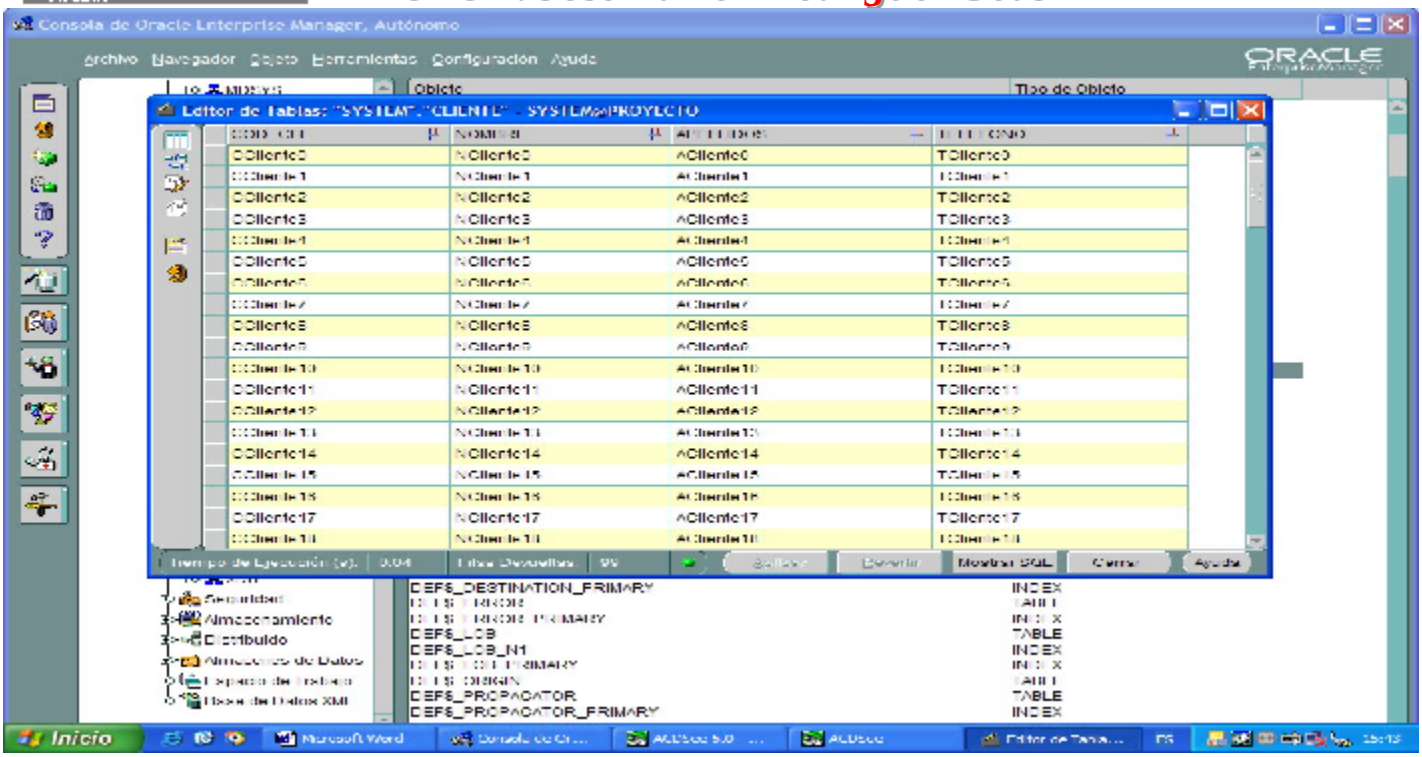
GASTOS



PROYECTO



CLIENTE



11. CONCLUSION

El diseño del sistema creado hasta ahora pone en práctica todos los conocimientos adquiridos durante la carrera, orientados a la creación y gestion de bases de datos ademas de las aplicaciones necesarias para la programación en JAVA de la creación de datos aleatorios para bases de datos. Durante el desarrollo de todo el proyecto se ha tenido en cuenta el diseño generado en un principio el cual ha permitido el desarrollo eficaz y correcto de la creación y gestion practica de la base de datos. Ha sido necesaria para la realización del mismo una correcta planificación ya gestionada en la PEC1 ademas de tener en cuenta todos los aspectos del enunciado, tambien teniendo en cuenta las consideraciones propias.

En un proyecto de este tipo entran en juego varias especializaciones, es decir, desde el diseño grafico de la aplicación, pasando por el mantenimiento de la misma, como la gestion de un jefe de proyectos. Todos estos puntos han sido tocados en la creación de este proyecto teniendo en cuenta la dificultad que implica este.Todos estos aspectos unidos han enriquecido mis conocimientos a lo largo de la creación del mismo.

Todo el proyecto queda documentado y especificado en los puntos anteriores, queda sujeto a nuevas interpretaciones de este o modificaciones futuras.

12. ANEXOS

12.1. BIBLIOGRAFÍA

12.1.1. SOPORTE PAPEL

- **FUNDAMENTOS Y MODELOS DE BASES DE DATOS**
  EDICIONES RAMA 2ª EDICION
  AUTORES:
  ADORACION DE MIGUEL
  MARIO PIATTINI
- **DESARROLLO DE APLICACIONES EN ENTORNOS DE 4ª GENERACION Y CON HERRAMIENTAS CASE.**
  EDICIONES MCGRAW HILL
  AUTORES:
  Mª JESUS RAMOS
  ALICIA RAMOS
  FERNANDO MONTERO
- **JAVA 2 CURSO DE PROGRAMACION**
  EDICIONES RAMA 2ª EDICION
  AUTORES:
  FRANCISCO JAVIER CEBALLOS
- **ESTRUCTURA DE LA INFORMACION**
- **FUNDAMENTOS DE LA PROGRAMACION I**
- **FUNDAMENTOS DE LA PROGRAMACION II**

12.1.2. SOPORTE WEB

- **WINDOWSXP**
  http://www.microsoft.com/latam/windowsxp/home/evaluacion/actualizar/comparacion.asp
- **PROCEDIMIENTOS**
  - **HTTP://WWW.BD.CESMA.USB.VE/CI3391/MANUAL/PROCEDURES.HTML**
  - **HTTP://WWW.LANIA.MX/BIBLIOTECA/SEMINARIOS/BASEDATOS/PLSQL/INTRO/ARCHI01.HTML**
  - **HTTP://WWW.PROGRAMATIUM.COM/MANUALES/ORACLE/8.HTM**

12.1.3. OTROS DOCUMENTOS

DIFERENTES CURSOS DE PROGRAMACION PARTICULARES EN FORMATOS PDF. ENTRE ELLOS, ORACLE, SQL, JAVA, PROGRAMACION, MODELOS E/R