Escuela Técnica Superior de Ingeniería Informática

# Guión de Prácticas de Programación de BD y Bases de Datos II

Juan Miguel Medina Rodríguez y Mª Amparo Vila Miranda

Dpto. Ciencias de la Computación e I. A. Universidad de Granada

## Granja "El Cerdito Valiente". Requisitos

- La gestión se centra en dos grupos de funciones de consulta y mantenimiento: el libro de contabilidad y la lista de personas que trabajan en la granja junto con sus direcciones y cualificación.
- El libro de contabilidad registra las transacciones financieras:
  - Cantidades de bienes comprados y vendidos
  - Jornadas pagadas y cantidades recibidas
- Lista de personas con sus direcciones y cualificaciones
- Personas involucradas en las transacciones y en los trabajos de la granja. Representar a ambos tipos de personas en un mismo conjunto de entidades

# Granja "El Cerdito Valiente". Requisitos

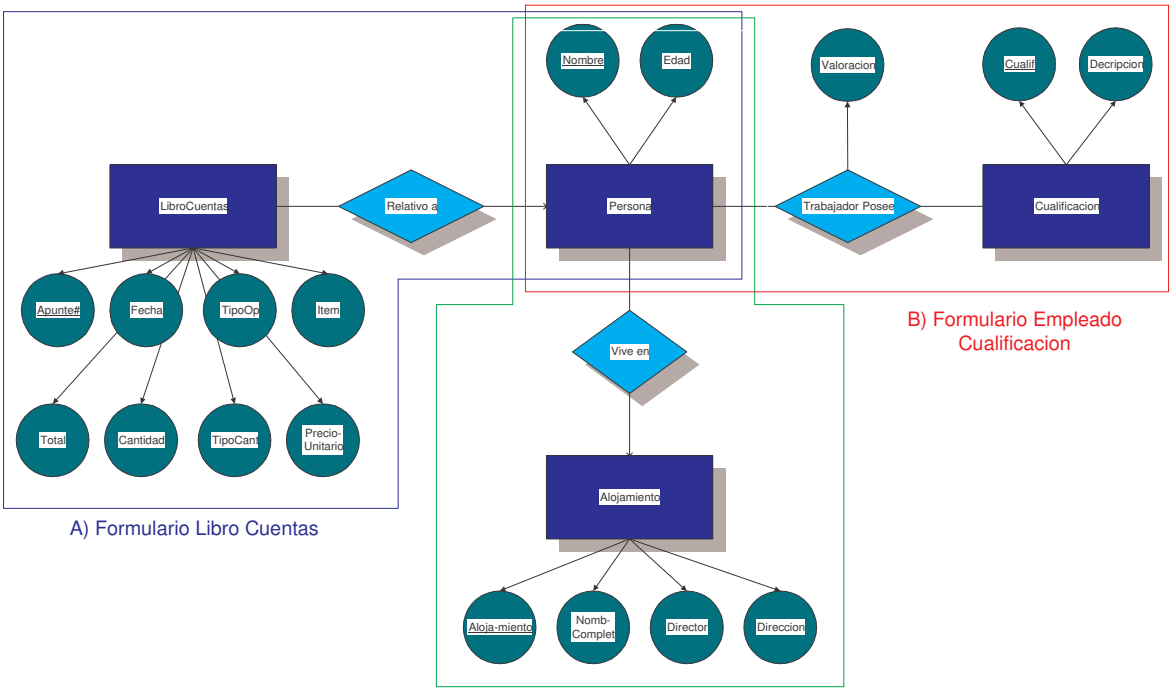
- Funciones que el software de aplicación habrá de tratar:
  - » Representar las personas y organizaciones con las que interactúa en una lista central a disposición de contabilidad y personal para consulta y mantenimiento. Deseable facilitar la corrección de fallos en la lista
  - » Seguimiento de todas las transacciones de la granja, incluyendo tipo (compra, venta, pago, ingreso), descripción del artículo, número unidades, importe unitario, total. Disponible para consulta y realización de auditorias

    Para auditoría todas las transacciones deben quedar registradas. Deseable que la aplicación pudiese generar automáticamente el importe de la transacción a partir de la cantidad y el precio. Que puedan ordenarse por cualquier campo
  - » Mantener una lista de los trabajadores para realizar contactos y pagos. Direcciones correctas y actualizadas. Poder generar etiquetas de correo
  - » Mantener un lista de de cualificaciones y competencias de los trabajadores para asignación de trabajos. Añadir un valoración para cada cualificación



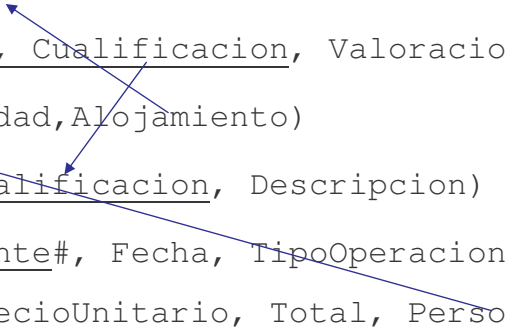
# Granja "El Cerdito Valiente". Diseño BD

- Diseño conceptual utilizando E/R



# Granja "El Cerdito Valiente". Diseño BD

- Diseño lógico relacional:
  - Alojamiento(Alojamiento, NomCompleto, Director,Direccion)
  - EmpCualif(Nombre, Cualificacion, Valoracion)
  - Persona(Nombre,Edad,Alojamiento)
  - Cualificacion(Cualificacion, Descripcion)
  - LibroCuentas(Apunte#, Fecha, TipoOperacion, Item, Cantidad, TipoCantidad, PrecioUnitario, Total, Persona



- Diseño Físico.
  - El archivo `h:\ccia\orawin95\tmp\CreaGranja.sql` contiene las sentencias sql que permiten la creación de la BD, donde se contempla, además, la creación de una secuencia para la generación de códigos de apuntes. `h:\ccia\orawin95\tmp\DatosGranja.sql` introduce algunos datos en esa BD.
  - Además de establecer los tipos y tamaños para los campos, se establece mediante una cláusula `check` los valores permitidos para el campo TipoOperacion. Posteriormente se pueden crear índices para mejorar la consulta



# Granja "El Cerdito Valiente". Diseño Funcional

- El formulario Libro de Contabilidad debe llevar a cabo las siguientes tareas:
  - Debe permitir la introducción de transacciones contables con todos sus datos
  - Generar identificadores únicos para cada apunte
  - La transacción debe ligarse a una persona presente en la BD
  - Calcular el total a partir de precio unitario y cantidad
  - Evitar que alguien no autorizado pueda actualizar o borrar transacciones
  - Consultas y ordenación por distintos criterios
- Adoptaremos una representación tabular basada en el esquema de datos A) de la figura que muestre varias transacciones a la vez y quizás una lista de valores o desplegable para seleccionar la persona implicada en la transacción
- El formulario EmpleadoCualifcación se va encargar de mostrar las cualificaciones que presenta cada empleado. Para eso se utilizará un maestro-detalle basado en el esquema de datos B)
  - El registro maestro (empleado) será de sólo lectura

# Granja “El Cerdito Valiente”. Diseño Funcional

  - » Enlaza con los detalles (las cualificaciones)
  - » Sería interesante poder seleccionar la cualificación de la lista de cualificaciones almacenada en la tabla correspondiente
- Necesarios formularios para mantener Persona, EmpCualif, Cualificaciones y Alojamiento
- Informe Resumen Libro basado en el esquema de datos A):
  - » Obtener las transacciones agrupadas por persona, en orden cronológico y total para cada persona
- Informe Direcciones basado en el esquema de datos C):
  - » Para obtener la dirección de cada persona para correspondencia. Salida de etiquetas de diversos tamaños.
- Antes de comenzar crearemos el siguiente directorio: `u:\CerditoValiente` donde ubicaremos todos los ficheros relacionados con los tutoriales y ejercicios



# Formulario Empleado. 1er Prototipo

- Procedimiento para crear el formulario:
  1. Cread el bloque persona que muestra un única persona
  2. Cread como detalle el bloque cualificaciones asociadas a cada persona
  3. Añadid una lista desplegable que permita seleccionar una cualidad de la misma
  4. Organizad el formulario
  5. Ejecutad y almacenad la aplicación
- Creación del bloque *persona*
  - » Ejecutad Form Builder
  - » Activad *Asistente de Bloques de Datos*
  - » Avanzad hasta visualizar
  - » Pulsad *Siguiente*

# Formulario Empleado. 1er Prototipo

- Pulsad *Examinar*
- Introducid vuestro *Usuario* y *Clave,* dejando en blanco *Cadena de Conexión*
- Visualizad vuestras tablas y vistas y seleccionad *persona*

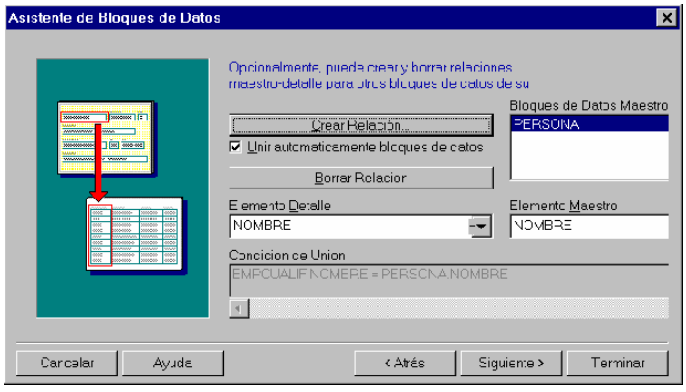


- Pulsad *Terminar* para entrar en el *Asistente de Diseño*
- Avanzad hasta
- Visualizad nombre y pulsad *Terminar*
- ¡Ya tenemos nuestro primer bloque!



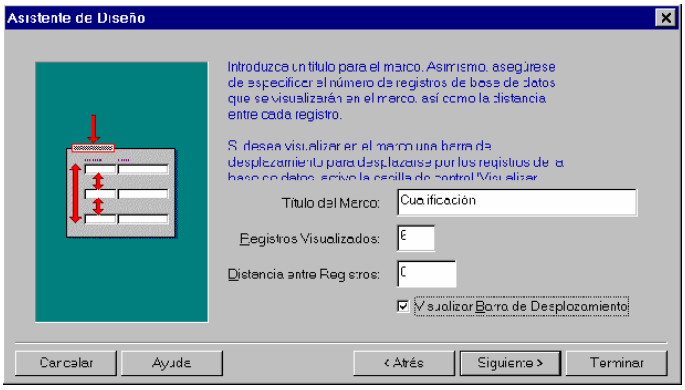
# Formulario Empleado. 1er Prototipo

- Creación del bloque *Cualificación*
  - Marcad el bloque *persona*
  - Para crear otro bloque pulsad 
  - Aceptad la utilización del *Asistente de Bloques de Datos*
  - Seleccionad la tabla *EmpCualif* y utilizad todos sus campos.
  - Avanzad y pulsad *Crear Relación*
  - Seleccionad la tabla *Persona* como maestro



# Formulario Empleado. 1er Prototipo

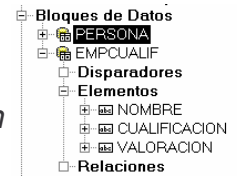
- Continuad con el *Asistente de Diseño*
- Visualizad sólo los campos *Cualificación* y *Valoración*
- Elegid el estilo Tabular para presentar los resultados
- Después poned *Cualificación* en el *Título del Marco*
- Visualizad 6 registros y activad *Visualizar Barra de Desplazamiento*
- Pulsad *Terminar*
- ¡Ya tenemos el esqueleto de nuestro primer formulario!






# Formulario Empleado. 1er Prototipo

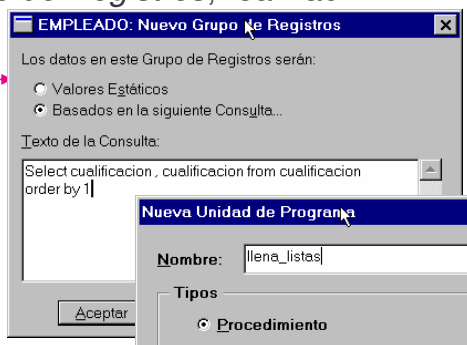
- Lista de Cualificaciones.
  - Vamos a crear una lista dinámica para seleccionar cualificaciones, generada a partir de la tabla Cualificaciones
  - Pulsad sobre *Cualificación,* botón derecho y abrir la *Paleta de Propiedades* con los atributos del *elemento Cualificación*
  - Estableced los atributos *Tipo de Elemento*, *Estilo de la Lista* y *Correspondencia de Otros valores* como muestra la figura
  - Pulsad en el valor de *Elementos de la Lista* y estableced al menos un elemento para la lista que se encuentre en la tabla Cualificación



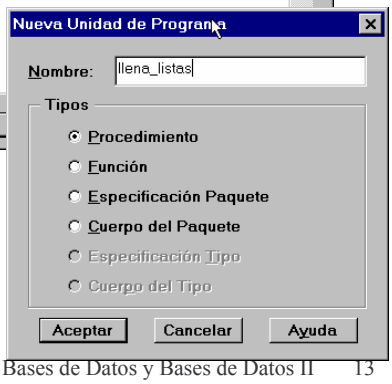

| | |
|---|---|
| Tipo de Elemento | Elemento de Lista |
| Información de Subclase | |
| Comentarios | |
| **Funcional** | |
| Activado | Sí |
| Elementos de la Lista | |
| Estilo de la Lista | Lista Desplegable |
| Correspondencia de Otros Valores | Herrero |

# Formulario Empleado. 1er Prototipo

- Para dotarla de carácter dinámico necesitamos:
  1. Escribir una consulta que obtenga los valores de la tabla Cualificaciones
  2. Escribir un procedimiento PL/SQL que la rellene con los datos de la tabla Cualificación
  3. Crear un disparador que actualice su contenido en el momento oportuno
1. Para crear esta consulta, que es un *Grupo de Registros*, realizad:
   - Marcad *Grupo de Registros* y pulsad 
   - Introducid
   - Llamadlo *Cualificacion*



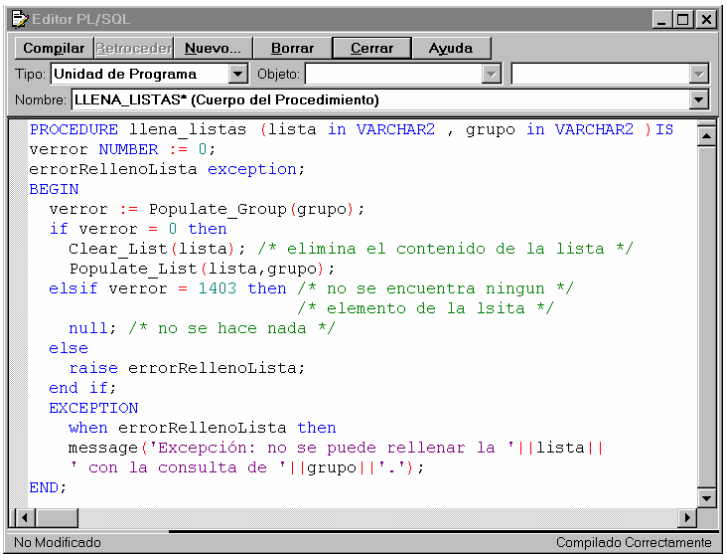

2. Dada la generalidad de este proceso vamos a crear un procedimiento reutilizable que vamos a incorporar a nuestra biblioteca para posteriores usos
   - Creamos una biblioteca.
     - Marcamos Bibliotecas PL/SQL y pulsamos 
     - Marcamos Unidades de Programa y pulsamos 
     - Introducimos






# Formulario Empleado. 1er Prototipo

- En la ventana de edición introducimos el siguiente código:




- Compilamos el módulo
- Salvamos la biblioteca con el nombre MiBiblioteca
- Marcamos la biblioteca y la compilamos (*Programa* | *Compilar Selección*)
- Marcamos *Bibliotecas Asociadas* bajo el Modulo1 y pulsamos  seleccionando MiBiblioteca

# Formulario Empleado. 1er Prototipo

3. Vamos a utilizar un disparador *WHEN-NEW-FORM-INSTANCE* para determinar que la lista de valores se actualice con los valores de la tabla Cualificación
   - » Marcad *Disparadores* pulsad 
   - » Seleccionad el mencionado disparador
   - » En la ventana introducid el código:

```
llena_listas('empcualif.cualificacion','cualificacion');
```

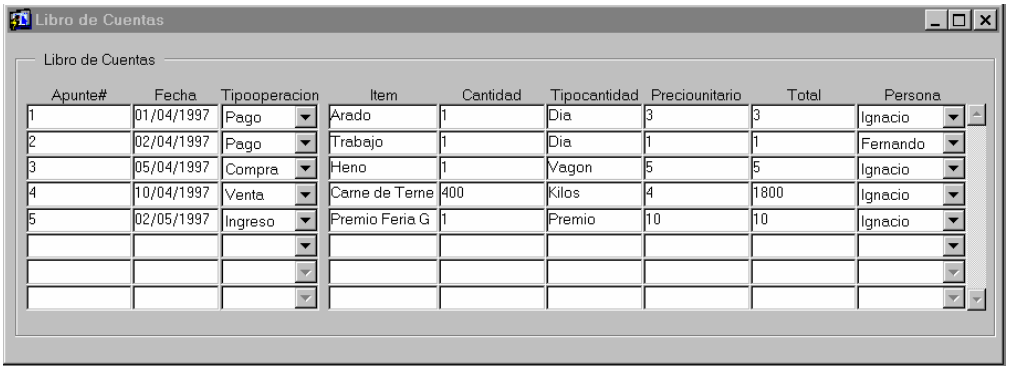
   - » Compilad y cerrad

» Ejecutad el formato, comprobad los resultados y almacenad como *empleado* si todo ha estado conforme



# Formulario Libro de Cuentas. Ejercicio

- Realizad el formulario correspondiente al libro de cuentas (llamadlo libro.fmb) de acuerdo con el aspecto de la figura y los siguientes requisitos:
  - » Debe incluir todos los campos de la tabla correspondiente
  - » Debe tener forma tabular (varias transacciones a la vez)
  - » Una lista estática sobre TipoOperacion (Compra,Venta,Pago,Ingreso)
  - » Una lista dinámica sobre Persona con datos de la tabla Persona



# Informe Contabilidad. 1er Prototipo

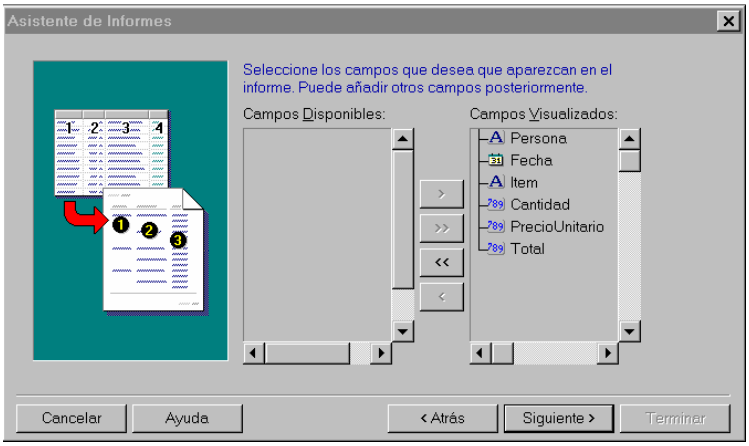
- El informe agrupará por personas las entradas y mostrará los subtotales de las mismas:
  - » Iniciar *Report Builder* y aceptar el uso del *Asistente de Informes*
  - » En **Título** ponemos: *Granja El Cerdito Valiente*
  - » Seleccionamos el estilo *Agrupar a la Izquierda* puesto que queremos agrupar por personas. Pulsad *Siguiente*
  - » En esta pantalla aprovechamos para conectarnos (Usuario, Clave)
  - » Escribimos la consulta:

```
SELECT Persona, Fecha, Item, Cantidad, PrecioUnitario, Total
  FROM LibroCuentas
```

  - » La siguiente pantalla solicita los campos sobre los que agrupar el informe. Introducid Persona. Pulsad *Siguiente*
  - » En esta pantalla se solicita los campos a visualizar en el informe. Introducidlos todos como muestra la figura



# Informe Contabilidad. 1er Prototipo




  - » Pulsad *Siguiente*. La pantalla de los totales solicita los campos a agregar. Nosotros queremos obtener las sumas de los totales por persona. Seleccionamos *Total* y pulsamos el botón *Suma*
  - » Pulsad *Siguiente* dos veces. En la pantalla de plantillas seleccionamos, por ejemplo, Corporate1
  - » Pulsad *Terminar* para obtener en el *Visor Activo* la vista preliminar del informe generado. Es preciso retocarlo un poco para mejorar la presentación

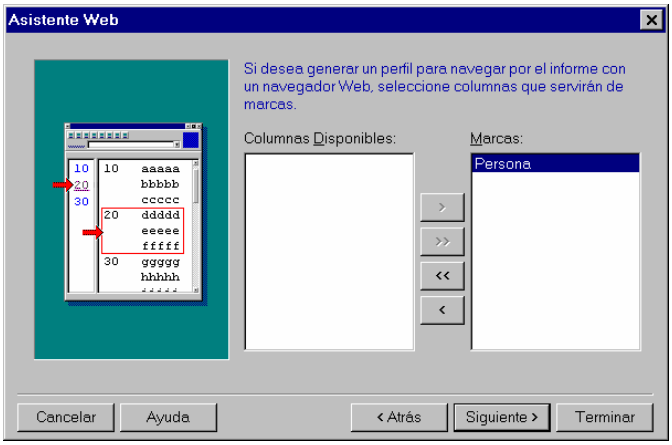
# Informe Contabilidad. 1er Prototipo

- Mediante la utilización del *Live Previewer*:
  - » Dimensionad los campos para que ocupen menos espacio
  - » Cambiad el formato de la fecha y la etiqueta asociada. Salvad como *LibroContabilidad.rdf*



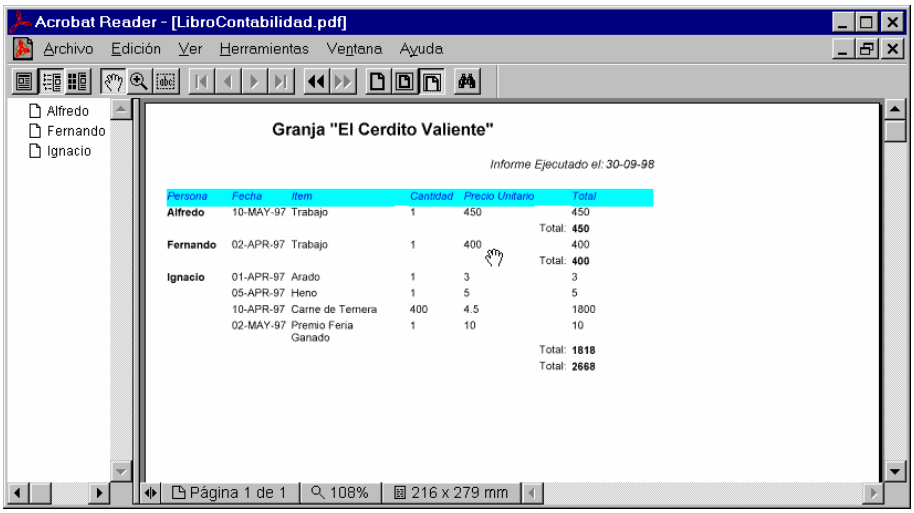

# Informe Contabilidad. En Formato "Web"

- A partir de este informe se puede crear una versión publicable en la "Web", ya sea en formato html o en formato pdf para ello se utiliza el *Asistente Web*:
  - » Seleccionad *Herramientas|Asistente Web*
  - » La pantalla de la figura nos solicita un campo del informe que nos va a permitir navegar por el mismo. En nuestro caso el campo Persona que es por él que se agrupa



# Informe Contabilidad. En Formato "Web"

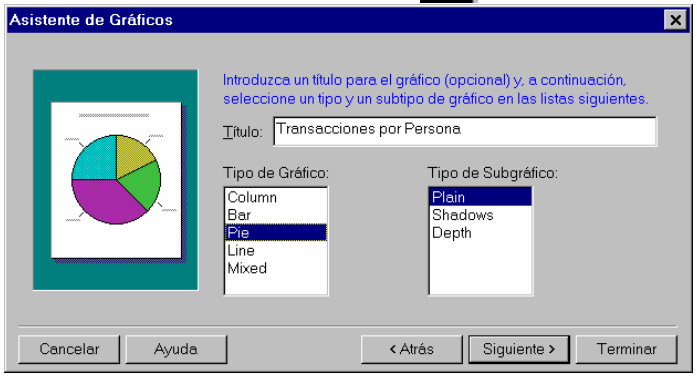
- La siguiente pantalla nos permite seleccionar un documento html que actúe como cabecera y otro como final de informe. En nuestro caso lo dejamos en blanco y seguimos
- Esta pantalla nos permite elegir el formato en que vamos a publicar el informe. Probad con pdf y visualizad con el Acrobat Reader. Después ejecutad de nuevo el Asistente Web y publicad en html visualizando con el navegador.





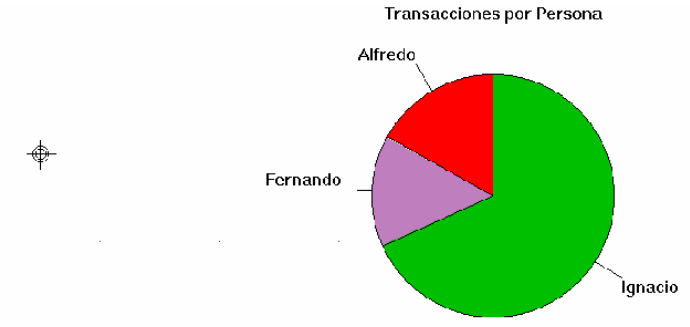

# Informe Contabilidad. Añadir Gráficos

- Podemos generar gráficos en diferentes formatos a partir de los datos e incorporarlos a nuestros formularios o informes
- *Graphics Builder* es la herramienta que se encarga de ello. Se puede invocar de forma separada o desde *Forms o Report Builder*.
- Vamos a generar un gráfico de tarta desde *Report Builder* que muestre las transacciones totales por persona
  - Con el último informe abierto se activa el Modelo de Diseño pulsando 
  - De la Barra de Herramientas seleccionamos la sección *margen* del informe 
  - Seleccionamos el Asistente de Gráficos  rellenamos la ventana como figura abajo



# Informe Contabilidad. Añadir Gráficos

- En la siguiente pantalla seleccionamos Persona como valor para el eje X
- En la siguiente seleccionamos *SumTotalPerPersona* como eje Y puesto que queremos las porciones según el total de cada persona
- En la ventana que nos pide la frecuencia con que va a aparecer el gráfico en el informe seleccionamos que sólo lo queremos una vez y al principio (*Al principio del Informe*)
- En la siguiente ventana indicamos que lo guarde como total.ogd
- Pulsad Terminar, se obtendrá algo parecido a:

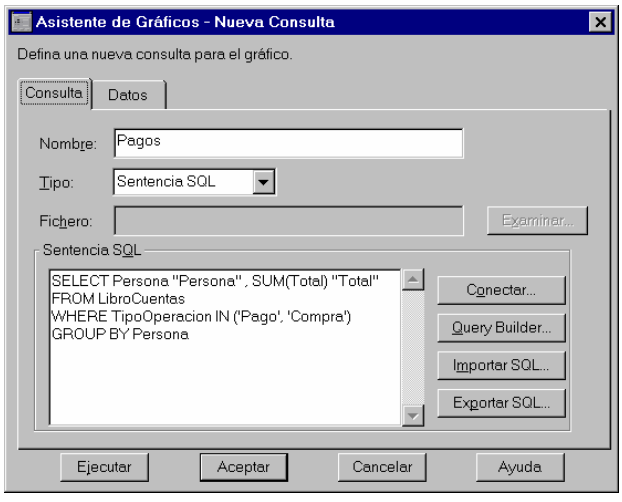



| Persona | Fecha | Item | Cantidad | Precio Unitario | Total |
|---|---|---|---|---|---|
| **Alfredo** | 11-MAY-07 | rahajn | 1 | 451 | 451 |
| | | | | Total: | **450** |
| **Fernando** | 12-APR-07 | rahajn | 1 | 4111 | 411 |
| | | | | Total: | **400** |
| **Ignacio** | 11-APR-07 | Arado | 1 | 3 | 3 |



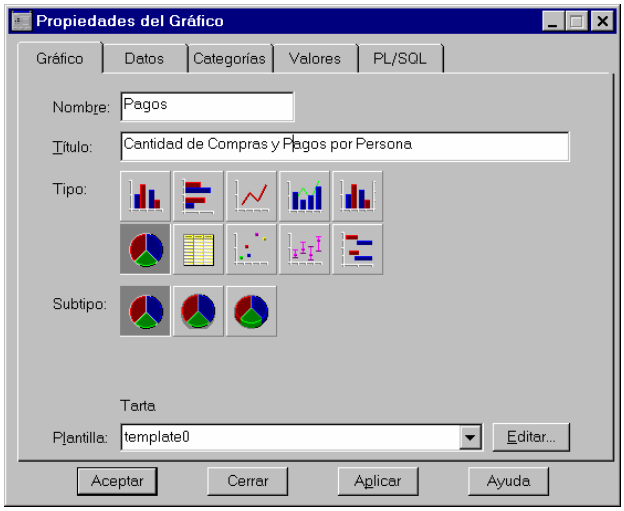
# Creación de Gráficos con Graphics Builder

- Ejercicio: Podéis jugar con los diferentes tipos de gráficos
- Vamos a crear un gráfico reutilizable desde el programa *Graphics Builder*
- El modelo de datos de un gráfico es una sentencia SQL que generalmente incorpora una cláusula GROUP BY
  - Ejecutamos *Graphics Builder*
  - Para crear un gráfico pulsamos  en la barra de herramientas del *Editor de Diseño*
  - Nos muestra el *Asistente de Gráficos*
  - Llenadlo con los datos de la figura. Pulsad *Ejecutar* para aseguraos del resultado



# Creación de Gráficos con Graphics Builder

» Pulsamos *Aceptar*, entramos en el cuadro para elegir las propiedades del gráfico a generar. Rellenamos según muestra la figura

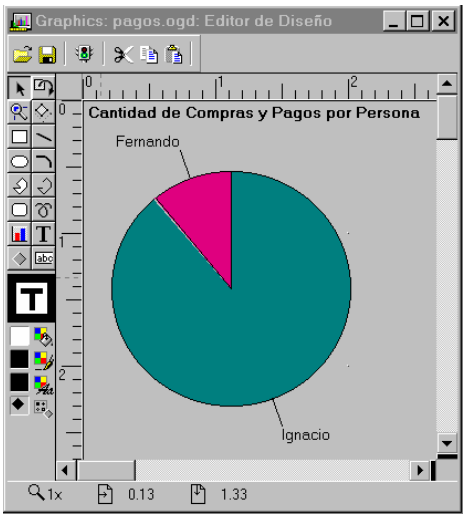



» Pulsamos *Aceptar* y se visualiza el gráfico. Podemos cambiar la fuente del título para resaltarlo más

» Salvad como *pagos.ogd*



# Creación de Gráficos con Graphics Builder




» Se puede utilizar *Gráfico*|*Marco* para organizar la presentación de los resultados

» Marcando el gráfico y pulsando el botón derecho se puede acceder a un menú en el que se puede indicar mediante *Actualizar* que se actualice la presentación del gráfico

» Pulsando *Herramientas*|*Consultas* accedemos a la consulta para una eventual modificación

# Integración de Gráficos con Formularios

- Veamos como se puede integrar un gráfico en un formulario. Se necesita un disparador que ejecute *Graphics* para producir el gráfico:
  - » Abrimos el formulario libro.fmb
  - » Creamos un bloque de control. Seleccionamos *Bloq de Datos* y pulsamos 
  - » Optamos por crearlo manualmente
  - » Abrimos la ventana de propiedades y ponemos: **Nombre**: *Control* , **Bloque de Datos de Base de Datos:** *No*
  - » Hacemos hueco bajo el frame que rodea al bloque LibroCuentas. Para ello establecemos la propiedad **Altura** a *400 en Window1 y Canvas2*
  - » Seleccionamos el bloque Control en el Navegador de Obj. se pulsa 
  - » Se dimensiona un área suficiente bajo el bloque LibroCuentas, optamos por construir el gráfico manualmente
  - » Visualizamos las propiedades del elemento creado y establecemos **Nombre**: *Pagos*, **Lienzo**: *Canvas2*
  - » Asociamos la biblioteca gráfica al formulario. Seleccionamos Bibliotecas Incorporadas y luego  , en el campo Biblioteca: ponemos OG, pulsamos el botón Incorporar



# Integración de Gráficos con Formularios

- » Asociamos la biblioteca gráfica al formulario. Seleccionamos Bibliotecas Incorporadas y luego  en el campo Biblioteca: ponemos OG, pulsamos el botón Incorporar
- » Necesitamos un disparador que active el gráfico y otro para cerrarlo
- » Editamos el disparador WHEN-NEW-FORM-INSTANCE para que contenga:

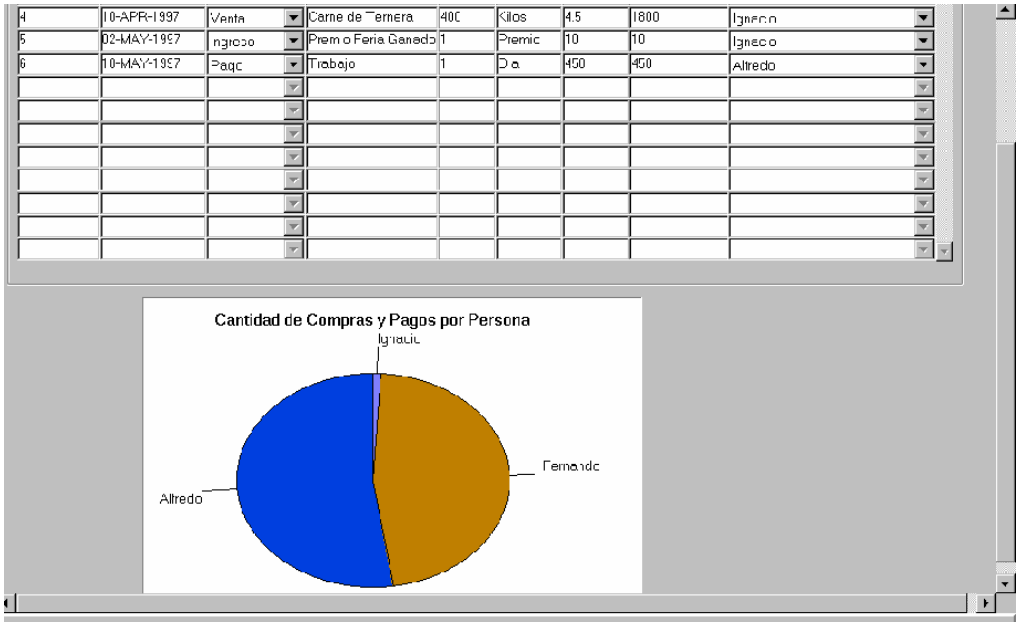
```
BEGIN
        llena_listas('librocuentas.persona','personas');
        OG.Open('pagos.ogd','control.pagos');
END;
```

- » Creamos un nuevo disparador de tipo POST-FORM con el siguiente código:

```
        OG.Close('pagos.ogd','control.pagos');
```

- » Ejecutamos, corregimos los posibles desajustes y salvamos el formulario
- » Si queremos refrescar en el gráfico los cambios que se deriven de la actuación sobre el formulario, debemos utilizar un disparador de tipo Post-Commit que cierre y vuelva a abrir el gráfico

# Integración de Gráficos con Formularios






# Elementos de los Formularios. Ventanas

- Canvas o Lienzos Son elementos de diseño sobre el que se sitúan los otros elementos. Una ventana actúa como un visor sobre el lienzo permitiendo desplazarse por su contenido. Tipos:
  - » Contenido: Elemento contenedor de otros. Una ventana debe tener al menos un lienzo de este tipo
  - » Apilado: Un lienzo que se muestra encima de otros. Suelen tener elementos independientes
  - » Tabular: Lienzo de contenido que organiza el contenido de una ventana a modo de fichas
  - » Barra de Herramientas: Lienzo que contiene botones con iconos que la ventana muestra en barras verticales y horizontales
- El tipo del lienzo se puede seleccionar mediante la propiedad **Tipo de Lienzo**
- La ventana que lo visualiza también se puede seleccionar desde la hoja de propiedades
- Todo elemento debe de estar asignado a un lienzo para ser visualizado. Si perseguimos que no se visualice ponemos *NULL* en su propiedad **Lienzo**

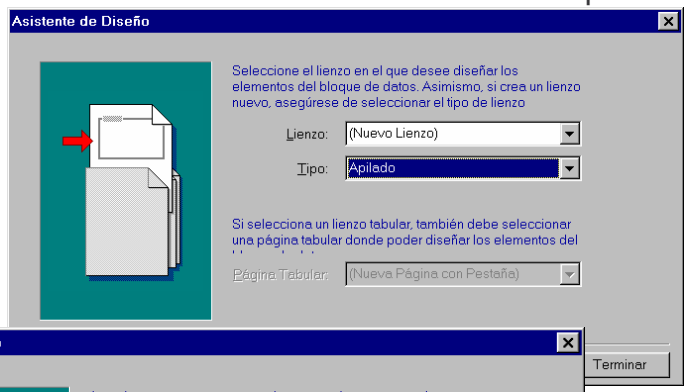
# Elementos de los Formularios. Ventanas

- Cuando se crea un *Bloque de Datos* se crea un lienzo por defecto
- También se puede crear manualmente desde *Navegador de Objetos* y dimensionar en su hoja de propiedades o en el *Editor de Diseño*
- Cuando se visualiza una ventana se visualiza el lienzo base, para visualizar otros lienzos se utilizan los procedimientos predefinidos *Show_View* y *Hide_View* desde disparadores
- Ejemplo: Lienzos apilados. Vamos a crear un formulario con Persona como maestro y LibroCuentas como detalle en un lienzo apilado que permita visualizar los apuntes de cada persona desplazándose horizontalmente por los mismos mientras se mantiene visualizados los datos de la persona correspondiente:
  - Abrid *Forms Builder* y utilizad el asistente para crear un bloque sobre Persona del que sólo vamos a visualizar el campo nombre
  - En el *Editor de Diseño* Desplazad el frame un poco hacia arriba
  - Seleccionad Canvas2 en el *Navegador de Objetos* y dimensionadlo en el *Editor de Diseño* de forma que se ajuste más o menos al frame
  - Ejecutad de nuevo el *Asistente de Bloques* para crear el bloque LibroCuentas



# Ventanas. Lienzos Apilados

- Seleccionad la tabla LibroCuentas y todos sus campos
- Solicitad la creación de una Relación con la tabla Persona entre los campos *persona* y *nombre*
- Entrad en el *Asistente de Diseño* y estableced los valores de la figura
- Visualizad todos los campos excepto persona
- Modificad los valores de *Ancho* de acuerdo con la figura
- Visualizad unos 5 registros con barra de desplaza-miento



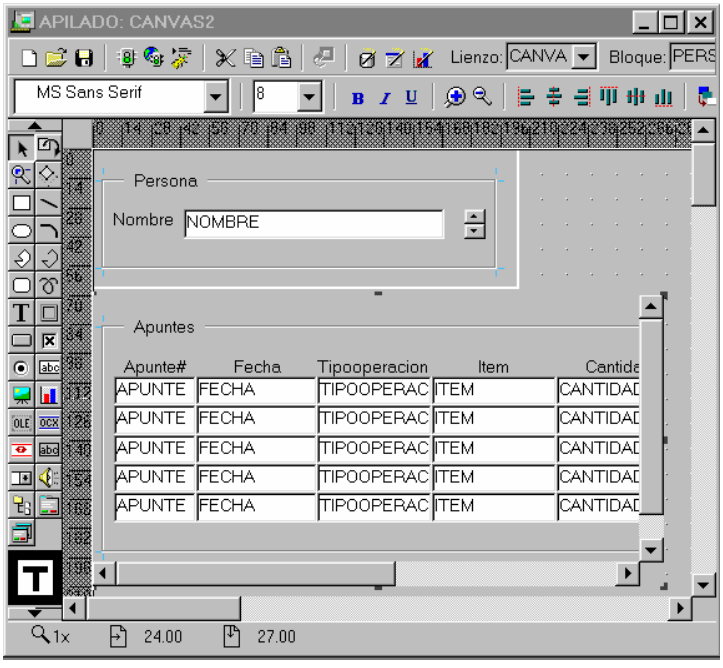

# Ventanas. Lienzos Apilados

| Viewport | |
|---|---|
| Posición X en la Puerta de Vista | 0 |
| Posición Y en la Puerta de Vista | 70 |
| Ancho de la Puerta de Vista | 275 |
| Altura de la Puerta de Vista | 140 |
| **Física** | |
| Visible | Sí |
| Ventana | WINDOW1 |
| Posición X de la Puerta de Vista en el Lienzo | 0 |
| Posición Y de la Puerta de Vista en el Lienzo | 0 |
| Ancho | 470 |
| Altura | 143 |
| Bisel | Hundido |
| Mostrar Barra de Desplazamiento Horizontal | Sí |
| Mostrar Barra de Desplazamiento Vertical | No |

- Editad las propiedades del lienzo apilado y poned los valores de la figura
- Con el *Editor de Diseño* dimensionad los lienzos y frames de los dos bloques de forma que rodeen los grupos de registros sin superponerse
- Editad el lienzo del bloque persona y pulsad en el menú *Ver|Vistas Apliadas...*
- Seleccionad el lienzo que aparece por defecto y aceptar. Esta operación permite superponer los dos lienzos en la misma ventana de edición.
- Si queremos eliminar la superposición realizamos la anterior operación y, mientras pulsamos la tecla Ctrl, pinchamos el lienzo que queremos desactivar, aceptando después
- Ved figura de la transparencia siguiente
- Editad las propiedades de *WINDOW1* y estableced: **Width**: *300* y **Height**: *280*
- Ejecutad, consultad y desplazaos utilizando la barra horizontal del canvas de LibroCuentas
- Modificad lo que sea necesario y salvad como *apilado.fmb*

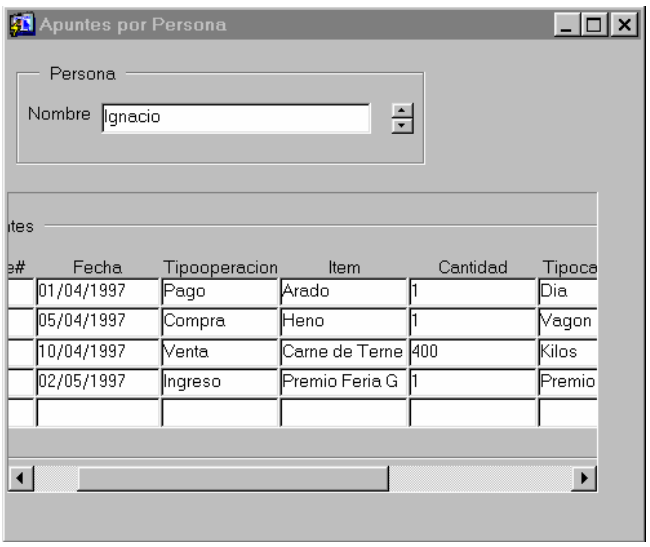


# Ventanas. Lienzos Apilados




Aspecto de los lienzos de Persona y LibroCuentas apilados en el *Editor de Diseño*

# Ventanas. Lienzos Apilados




Aspecto del formulario en ejecución



# Ventanas. Lienzos Tabulares

- Es un tipo de carpeta que, cuando se pulsa sobre la solapa, se muestra la página correspondiente
- Vamos a crear el formulario Recursos Humanos, que asocie a cada persona (maestro) sus capacidades (detalle) por un lado y los datos de alojamiento (detalle) por otro:
  - » Abrid *Forms Builder* y utilizad el asistente para crear el bloque Persona con las siguientes características:
    - Basado en la tabla persona, que utilice todos los campos de dicha tabla
    - En el *Asistente de Diseño* asignadle la creación de un lienzo nuevo de tipo *Contenido*
    - Visualizad únicamente nombre y edad
    - Visualizadlo en formato Formulario, el titulo del *Marco* Persona, visualizad un sólo registro y la barra de desplazamiento
    - Verificad el correcto funcionamiento y salvad como *RecursosHumanos.fmb*
  - » Cread el bloque detalle EmpCualif con las siguientes características:
    - Basado en la tabla del mismo nombre con todos sus campos
    - La condición que lo relaciona con Persona, que va a ser su bloque maestro, estará basada en la igualdad del campo nombre

# Ventanas. Lienzos Tabulares

- Asignadle un lienzo nuevo (*Nuevo Lienzo*) de tipo de tabular (*Tabular*) y (*Nueva Página con Pestaña*)
- Visualizad cualificación y valoración
- Elegid una presentación tabular visualizando 5 registros y la barra de desplazamiento
- Visualizad la Paleta de Propieades de PAGE5 y estableced: **Etiqueta:** *Cualificacion*
- Ejecutad, sólo vereis el lienzo tabular. Salvad con el mismo nombre

» Cread el bloque Alojamiento:

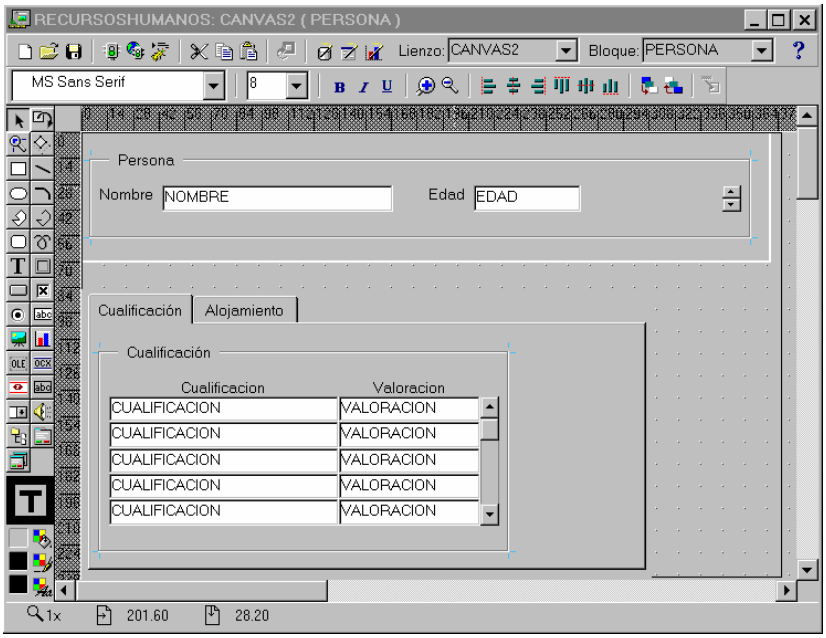
- Sobre la tabla del mismo nombre utilizad todos sus campos
- Cread la relación detalle con el bloque persona como maestro (desactivad antes *Unir automaticamente bloques de datos*) basada en una condición de unión sobre el campo alojamiento
- Utilizad el mismo lienzo que para el bloque EmpCualif y una nueva página (*Nueva Página con Pestaña*)
- Visualizad todos los campos salvo alojamiento pulsad *Terminar* para aceptar el resto de los valores por defecto
- Visualizad la Paleta de Propiedades de PAGE7 y estableced: **Etiqueta:** *Alojamiento*

» Redimensionad y ubicad los lienzos:

- Seleccionad *CANVAS4* y en la Paleta de Propiedades poned: **Posición Y en la Puerta de Vista:** *85* para situad el lienzo bajo él de persona;
- Redimensionad el lienzo de Persona para que rodee el Marco
- Redimensionad *CANVAS4* para que rodee la solapa de Alojamiento, en sus propiedades poned **Bisel**: *Ninguno* para que no se visualice en ejecución el borde del lienzo. Dimesionad WINDOW1 para que rodee el lienzo.
- Aseguraos de que el bloque Persona está el primero en la lista de bloques
- Ejecutad, corregid y salvad

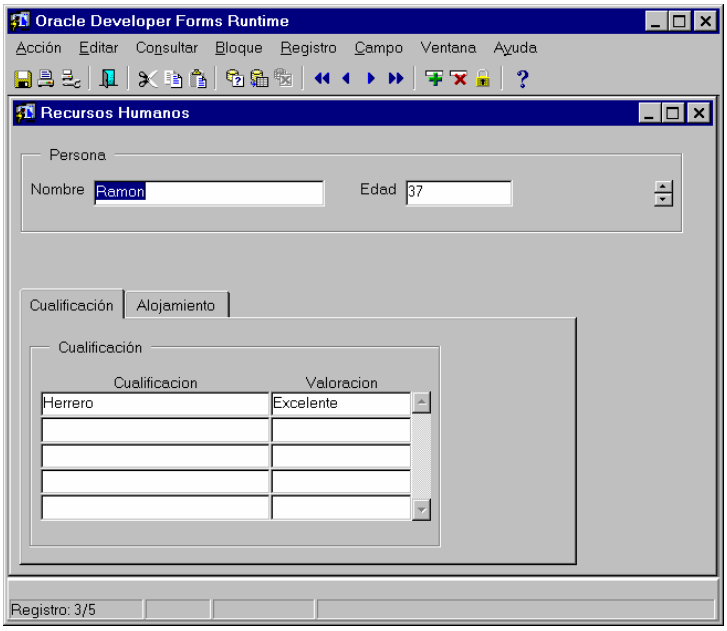


# Ventanas. Lienzos Tabulares




Diseño visualizado en modo *Ver|Vistas Apiladas...*

# Ventanas. Lienzos Tabulares




Formulario RecursosHumanos en Ejecución



# Visualización de Lienzos mediante Ventanas

- Developer2000 utiliza la arquitectura MDI (Interfaz de Múltiples Documentos). En ejecución, Developer2000 dispone una ventana principal de aplicación, que no dispone de lienzos, mediante la que visualiza el resto de las ventanas de documento o de diálogo
- Estas últimas ventanas deben tener asociado un lienzo al menos
- El menú principal pertenece a la ventana principal, que siempre está abierta
- Las ventanas de tipo documento tienen restringida su visualización a la ventana principal
- Las ventanas de diálogo pueden visualizarse fuera
- Otro criterio de clasificación de ventanas modales o no modales:
  - » Las modales requieren de una respuesta del usuario al mensaje que ofrecen para desaparecer y, en algunos casos, permitir continuar con la aplicación (cuadros de diálogo)
  - » Las no modales permite acceder a otra ventana sin hacerla desaparecer

# Visualización de Lienzos mediante Ventanas

- La ventana de aplicación se corresponde con el formulario que se ejecuta actualmente. De esta forma:
  - El menú principal que se ejecuta es el menú del formulario
  - Podemos incluir barras de herramientas horizontales y verticales mediante el empleo de las propiedades *Lienzo de la Barra de Herramientas Horizontal* y *Lienzo de la Barra de Herramientas Vertical* del formulario, respectivamente
  - El procedimiento predefinido *Set_Window_Property* permite establecer de forma programada las propiedades de las ventanas. Para modificar las correspondientes a la ventana principal debemos pasar como primer argumento la referencia FORMS_MDI_WINDOW
- En entornos que no soportan MDI, como Java (Web), Motif y Macintosh hay que recurrir al empleo de una ventana raiz. El empleo de esta ventana sólo tiene sentido en estos entornos si queremos dotarlos de MDI. La ayuda en línea muestra la forma de utilizar esta característica
- Propiedades Funcionales de las ventanas:
  - **Cierre Permitido** (Sí|No) si permitimos que, en ejecución, el usuario cierre la ventana
  - **Dimensionamiento Permitido** (Sí|No) si permitimos que, en ejecución, el usuario redimensione la ventana



# Visualización de Lienzos mediante Ventanas

  - **Minimización Permitida** (Sí|No) permitir o no la minimización
  - **Maximización Permitida** (Sí|No) permitir o no la maximización
  - **Movimiento Permitido** (Sí|No) permitir o no el desplazamiento de la ventana
  - **Heredar Menú** (Sí|No) si muestra o no el menú del formulario (no es operativo en Microsoft Windows)
- Propiedades Estructurales de las ventanas:
  - **Lienzo Primario**, el lienzo que muestra en primer lugar
  - **Lienzo de Barra de Herramientas Vertical** y **Horizontal**, las barras de herramientas verticales y horizontales a mostrar
  - **Estilo de Ventana** (Documento|Diálogo), tipo de ventana
  - **Modal** (Sí|No) si es o no modal
  - **Ocultar al Salir** (Sí|No) si oculta la ventana o no cuando se navega a otra
  - **Nombre de Fichero del Icono**, **Título minimizado**, nombre del archivo del icono que aparece cuando se iconifica y título del mismo
  - **Mostrar Barra de Desplazamiento Horizontal** y **Vertical**, si muestra o no las correspondientes barras de desplazamiento
- Abrir y cerrar ventanas:
  - Utilizando disparadores que desencadenen la navegación hacia, o fuera de una ventana

# Elementos de los Formularios. Ventanas

- Canvas Son elementos de diseño sobre el que se sitúan los otros elementos. Una ventana actúa como un visor sobre el canvas permitiendo desplazarse por su contenido. Tipos:
  - » Contenido: Elemento contenedor de otros. Una ventana debe tener al menos un canvas de este tipo
  - » Apilado: Un canvas que se muestra encima de otros. Suelen tener elementos independientes
  - » De fichas: Canvas de contenido que organiza el contenido de una ventana a modo de fichas
  - » Barra de Herramientas: Canvas que contiene botones con iconos que la ventana muestra en barras verticales y horizontales
- El tipo del canvas se puede seleccionar mediante la propiedad **Canvas Type**
- La ventana que lo visualiza también se puede seleccionar desde la hoja de propiedades
- Todo elemento debe de estar asignado a un canvas para ser visualizado. Si perseguimos que no se visualice ponemos *NULL* en su propiedad **Canvas**



# Ventanas de Aviso o Alertas

- Developer dispone de un mecanismo específico para alertas o avisos que simplifica su programación, de hecho lo utiliza automáticamente para visualizar ciertos mensajes
- Para crear un aviso seleccionamos el objeto *Alertas* en el *Navegador de Objetos* y pulsamos create
- Dispone de tres clases de avisos que se pueden especificar mediante la propiedad **Estilo de Alerta:** Parar, Precaucion y Nota este último es del estilo del que hemos creado recientemente
- La propiedad **Mensaje** permite introducir el mensaje a mostrar en el aviso. Las propiedades **Etiqueta del Botón 1, Etiqueta del Botón 2 y Etiqueta del Botón 3** permiten etiquetar los botones que muestra el aviso, si no se introduce etiqueta el aviso no muestra el botón correspondiente
- Se debe designar un botón como botón por defecto cuando el usuario pulsa Enter
- Para determinar la aparición de un aviso se utiliza la función predefinida *Show_Alert* desde el código de un disparador, la función devuelve el número *ALERT_BUTTON1, ALERT_BUTTON2, ALERT_BUTTON3*, dependiendo de que botón haya sido pulsado, en el código del disparador se programarán las acciones apropiadas en función del botón pulsado

# Ventanas de Aviso o Alertas

- Ejercicio: Realizad el ejemplo anterior utilizando un aviso. Considerad la utilización del siguiente código para el disparador:

```
DECLARE
  boton NUMBER;
BEGIN
  boton := Show_Alert('Bienvenido');
  IF boton = ALERT_BUTTON1 THEN
    NULL;
  END IF;
END;
```

- Podemos crear elementos de aviso reutilizables. Consideremos el siguiente código que permite gestionar genéricamente la aparición de avisos de tipo informativo:

```
PROCEDURE muestra_informacion (titulo IN VARCHAR2, mensaje IN VARCHAR2) IS
  avisoID Alert := Find_Alert('Informativo');
  ignora  NUMBER;
BEGIN
  Set_Alert_Property(avisoID, TITLE, titulo);
  Set_Alert_Property(avisoID, ALERT_MESSAGE_TEXT, mensaje);
  ignora := Show_Alert(avisoID);
END;
```



# Ventanas de Aviso o Alertas

- Desde el *Navegador de Objetos,* en el objeto Bibliotecas PL/SQL abrimos la biblioteca *mibiblioteca.pll*
- Creamos una nueva unidad de programa que va a llamarse y contener el código mostrado en la transparencia anterior. Lo compilamos y cerramos el editor y, seleccionando mibiblioteca, pulsamos Guardar.
- En Bibliotecas Incorporadas del Formulario RecursosHumanos, no de Bibliotecas PL/SQL, asociamos la biblioteca mibiblioteca si todavía no la tenemos asociada
- Creamos una nueva alerta (Alertas) que se va a llamar informativo y va a ser de tipo nota (**Estilo de Alerta:** Nota)
- Creamos o utilizamos un disparador a nivel de formulario de tipo WHEN-NEW-FORM-INSTANCE cuyo código será el siguiente:

```
muestra_informacion('Bienvenido','Bienvenido a la Aplicacion Recursos
  Humanos');
```

- Ejecutamos y salvamos
- Situad el bloque dialogo al final para que no se muestre la ventana de ayuda que lo contiene
- Ejercicio. Modificad el ejercicio para que permita seleccionar el tipo de alerta

# Barras de Herramientas

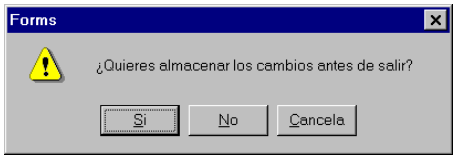
- Vamos a construir una barra de herramientas con el siguiente aspecto


y con la siguiente funcionalidad:

» El primer botón visualiza la persona anterior, el segundo la persona siguiente

» El botón salvar almacena todos los cambios producidos

» El botón salir muestra la ventana de aviso: donde, la primera opción almacena y sale la segunda sale sin almacenar y la tercera cancela la salida




- Para ello acometemos los siguientes pasos:

» Cread un lienzo de tipo barra de herramientas (**Tipo del Lienzo**: *Barra de Herramientas Horizontal*) denominado *Herramientas*, visualizadlo en Window1 (**Ventana:** *WINDOW1*) estableced **Altura**: *28*

» Editad este lienzo en el *Editor de Diseño* y aseguraos que en Bloque: ponga persona para que los elementos que añadamos estén asociados a ese bloque

» Utilizando la barra de herramientas vertical del *Editor de Diseño* seleccionar, ubicad y dimensionad cuatro botones en forma similar a la figura de arriba



# Barras de Herramientas

» Los nombres y etiquetas respectivas de esos botones son: anterior y <, posterior y >, salva y Salvar, salida y Salir

» Seleccionad de forma múltiple los cuatro botones (mientras se mantiene pulsada la tecla Mayúsculas se van seleccionado los botones con el ratón) y visualizad la *Paleta de Propiedades* y estableced las propiedades **Teclado de Navegación y Navegación del Ratón** *No*. Esto hace que la navegación no permanezca en ninguno de estos botones

» Cread un nuevo aviso (Alertas) y estableced las propiedades: **Nombre**: *Salir*, **Estilo de Alerta:** *Precaución*, **Etiqueta de Botón 1**: *Si*, **Etiqueta de Botón 2:** *No*, **Etiqueta de Botón 3:** *Cancela*

» Ahora debemos asociar a cada botón la función a realizar cuando se pulsa. El disparador WHEN-BUTTON-PRESSED se activa cuando pulsamos ese botón y ejecuta el código asignado al mismo, por tanto utilizaremos ese disparador para cada uno de los botones:

– Creamos un disparador de este tipo para el botón ANTERIOR con el código: `do_key('up');`

– Para el botón POSTERIOR el código es: `do_key('down');`

– Para el botón SALVA el código es: `commit_form;`

# Barras de Herramientas

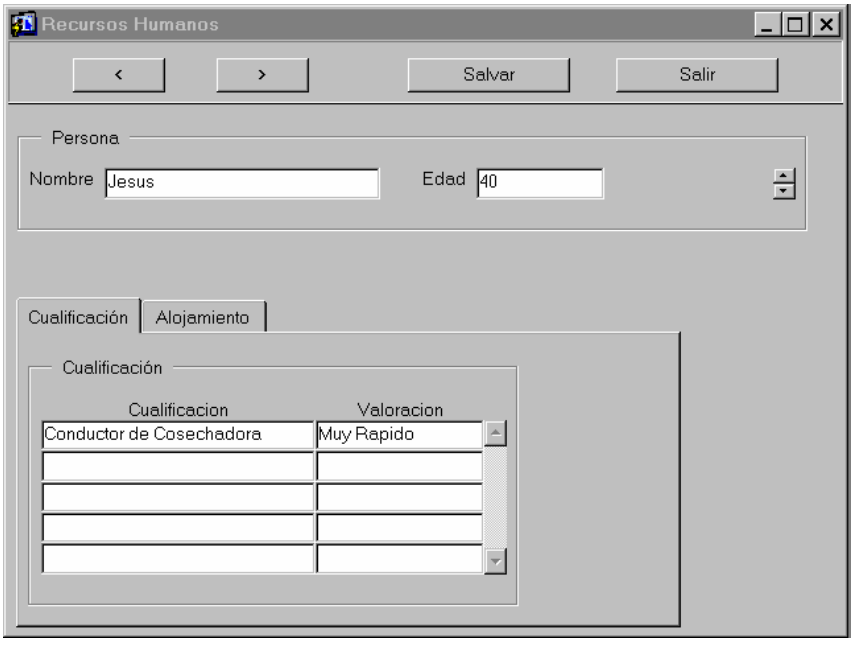
– Para el botón SALIDA el código es algo más complejo:

```
DECLARE
 boton NUMBER;
BEGIN
SET_ALERT_PROPERTY('salir',ALERT_MESSAGE_TEXT,'¿Quieres almacenar los
  cambios antes de salir?');
boton := show_alert('salir');
if boton = ALERT_BUTTON1 then
     commit_form;
end if;
if boton <> ALERT_BUTTON3 then
  exit_form(no_commit);
end if;
end;
```

» Ejecutad y verificad el funcionamiento, corregid y salvad



# Barras de Herramientas




Formulario en Ejecución

# Bloques. Elementos

- Los bloques son elementos funcionales normalmente asociados a tablas aunque pueden actuar como marco que agrupa elementos de control
- Los elementos que contiene pueden recurrir al empleo de algunos de los siguientes componentes o características:
  - » Listas que pueden ser de cuatro tipos:
    - – Lista emergente: Una lista desplegable del tipo visto
    - – Lista T: Un cuadro que muestra todos los elementos de texto en una lista desplazable, como un canvas independiente, pero dentro de un único campo de formulario
    - – Recuadro de Combinación: Combinación de lista emergente y de campo de texto, permite introducir valores distintos a los de la lista
    - – Lista de Valores (LOV List of Values): Un cuadro de diálogo con posibilidades de búsqueda que se puede asociar a un campo de texto
  - » Botones de opción para crear opciones mutuamente exclusivas
  - » Campos con resultados de cálculos para facilitar la tarea de mostrar totales y otros cálculos que se realizan en el formulario
  - » Etc.

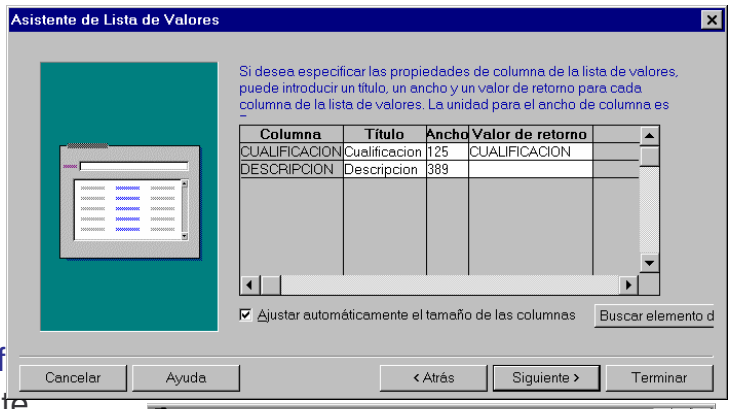


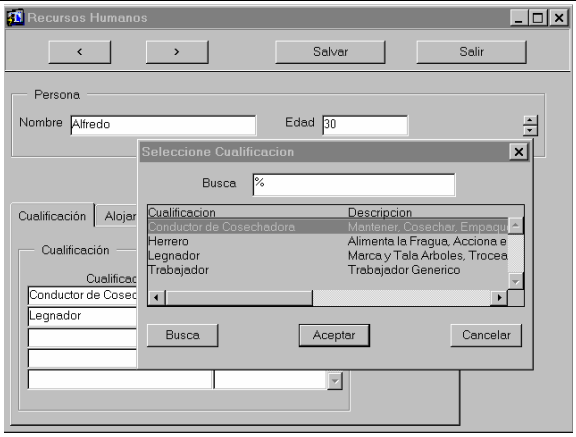
# Bloques. LOV

- A diferencia de las listas permiten asociar más de un valor por tupla.
- Vamos a ilustrar su empleo mediante la creación de una LOV para el campo *cualificación* del bloque *empcualif* del formulario *Recursos Humanos* utilizando el asistente
  - » Seleccionamos la cabecera de los elementos LOV en el *Navegador de Objetos* y pulsamos create
  - » Dejamos marcado *Nuevo Grupo de Registros basado en una consulta*
  - » Introducimos la sentencia de la figura y comprobamos
  - » Desplazamos las dos columnas a la ventana derecha
  - » Avanzamos

# Bloques. LOV

- Disponemos la forma en que se asocian los valores visualizados a los campos del formulario, estableciendo *Cualificacion* como valor de retorno para la columna CUALIFICACION. Esto devuelve únicamente el valor cualificación a ese campo del bloque empCualif
- Activamos Ajustar automáticamente ...
- En Titulo ponemos Seleccione Cualificacion
- Despazamos cualificacion a la ventana derecha. Terminamos
- Ejercicio: Cread un formulario sobre Persona con todos los campos, añadid los campos NomCompleto, Director y Direccion que no estén asociados a tabla. Cread un LOV que rellene el campo alojamiento y visualice estos campos. Cread un disparador que los visualize en modo consulta. Considerad la utilización de disparadores Post-Query y Post-Change



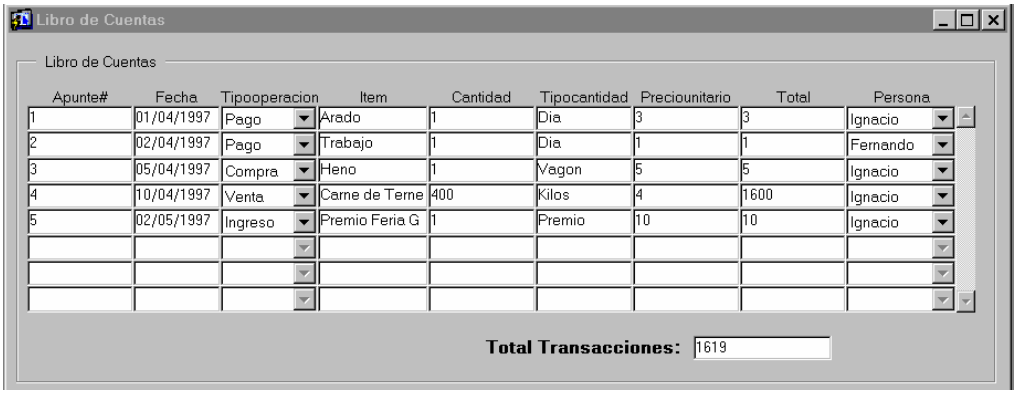






# Bloques. Elementos con Resultados de Cálculos

- Se pueden crear campos que muestren cálculos elaborados en función de otros campos (total en función de cantidad y de precio unitario) o que realicen agregaciones para grupos de registros (cantidad total de transacciones). Estos campos no almacenan el resultado en la BD
- Modifiquemos el campo total para que calcule automáticamente el total en función de la cantidad y el precio unitario:
  - Visualizamos las propiedades del campo total y establecemos: **Modo de Cálculo:** *Fórmula* y **Fórmula**: `:librocuentas.cantidad * :librocuentas.preciounitario`
  - Ejecutad y cambiad los valores de los campos cantidad y preciounitario para ver que sucede
- Para mostrar el total de las transacciones realizamos las siguientes operaciones:
  - Cread un campo de tipo Elemento de Texto en el bloque librocuentas con las propiedades: **Nombre**: *SumTotal*, **Tipo de Dato:** *Number*, **Número de Elementos Visualizados:** *1*, **Elemento de Base de Datos:** *No*, **Modo de Cálculo:** *Total*, **Función Totalizar:** *Sum*, **Bloque de Total:** *librocuentas* y **Elemento de Total:** *Total*
  - Ubicad el campo bajo el campo total y cread y ubicad una etiqueta a su izquierda que ponga "Total Transacciones:"
  - Seleccionad el bloque librocuentas y estableced: **Consultar Todos los Registros:** *Sí* y **Precalcular Totales:** *No*

# Bloques. Elementos con Resultados de Cálculos



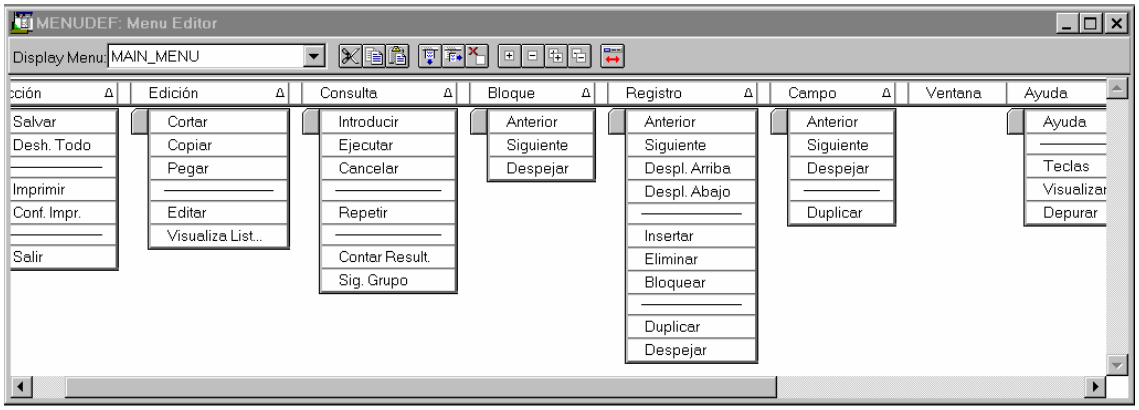

Aspecto del Formulario Libro en Ejecución



# Menús

- Developer dispone de un menú por defecto en la ventana de aplicación que permite ejecutar las órdenes habituales en ejecución, *Next Record*, *Enter_Query*, etc.
- Cada una de estas órdenes se corresponde directamente con la pulsación de una tecla y existen procedimientos predefinidos para desencadenar su ejecución
- La utilización de este menú y de su barra de herramientas asociada es por defecto, a menos que se defina y utilice un menú personalizado. El contenido y funcionalidad del menú por defecto no se puede alterar
- **Menus Personalizados**. Es conveniente partir de un menú previamente definido, como el menú `menudef.mmb` que se encuentra definido en `h:\ccia\orawin95\tmp`
  - » Comenzaremos por abrir el formulario *libro.fmb*
  - » Seleccionamos la clase Menus debajo de Forms e invocamos *Abrir*, seleccionando el menú mencionado
  - » Ejecutamos el *Editor de Menús* y editamos las etiquetas de las diferentes opciones para traducirlas al español de acuerdo con la figura de siguiente transparencia

# Menús

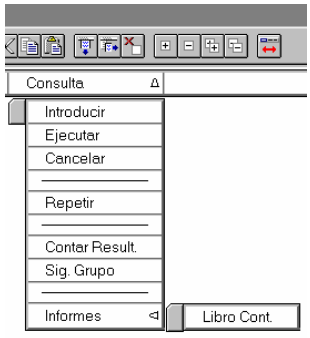



- Vamos a introducir una entrada en la opción de menú Consulta para invocar al informe *LibroContabiblidad.rdf*
- Para ello, desde el *Editor de Menús*, utilizamos el botón  para introducir bajo *Sig. Grupo* un separador (**Nombre**: *Sep20*, **Etiqueta**: *Sep20*, **Tipo de Elemento de Menú:** *Separador*), la opción Informes (**Nombre:** *Informes*, **Etiqueta:** *Informes*, **Tipo de Elemento de Menú:** *Normal*). Con esta última opción seleccionada pulsamos  para crear una opción lateral (**Nombre:** *Libro*, **Etiqueta:** *Libro Cont.*, **Tipo de Elemento de Menú:** *Normal*)

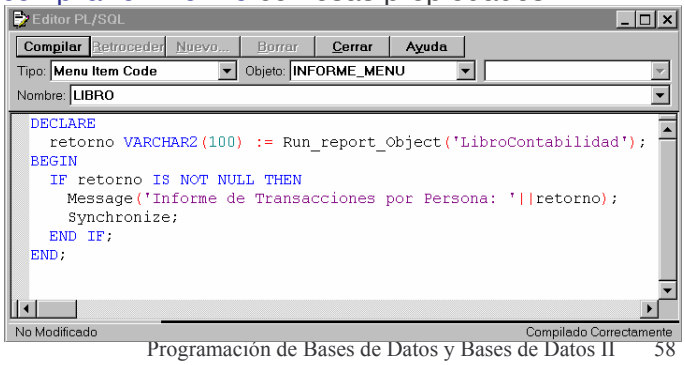


# Menús

- La opción de menú Consulta debe presentar el siguiente aspecto:
- Ahora vamos a incorporar el informe *LibroContabilidad* a nuestro formulario, para ello:
  - Seleccionamos en el Navegador de Objetos la clase Informes dentro del formulario libro e invocamos crear, en la ventana de diálogo optamos por abrir el informe LibroContabilidad.rdf previamente creado
  - Modificamos las propiedades del informe: **Nombre**: *LibroContabilidad*, **Modo de Ejecución** *Runtime*, **Modo de Comunicación:** *Síncrono*, **Tipo de Destino del Informe:** *Pantalla*
  - Manteniendo seleccionado el informe, pulsamos *Fichero|Administración|Compilar Fichero* en el Menú Principal para recompilar el informe con esas propiedades




- Para invocar la ejecución del informe desde la opción Libro Cont. editamos sus propiedades y en **Código del Elemento de Menú** escribimos el código de la figura. Compilad



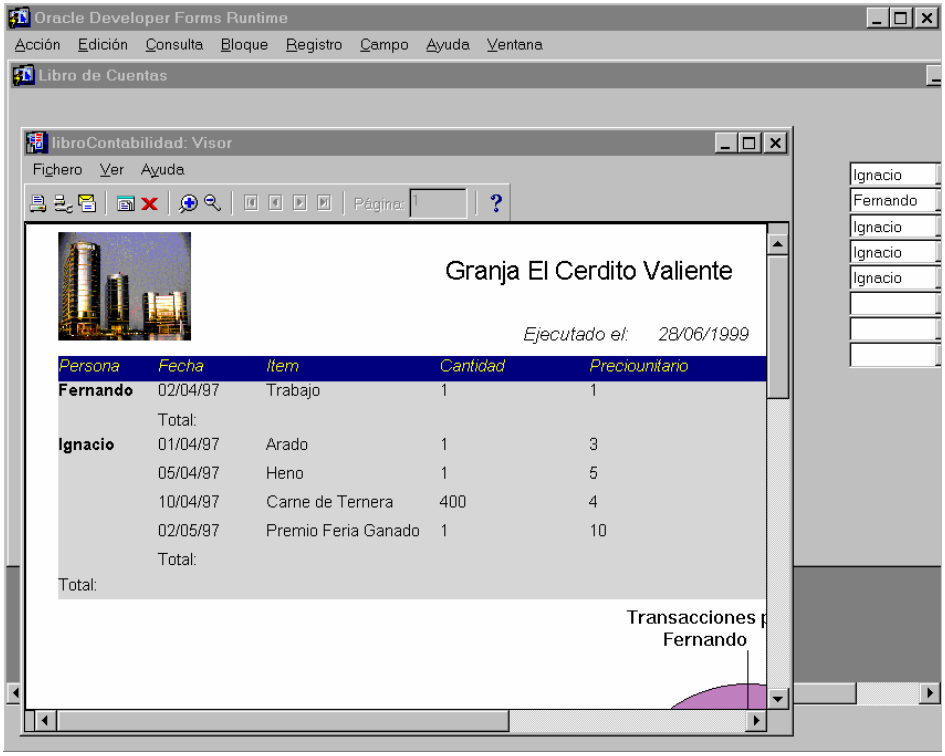
# Menús

- Sálvemos el menú en Cerdito Valiente como mimenu.mmb
- Con dicho menú seleccionado, procedemos a compilarlo mediante la opción *Fichero|Administración|Compilar Fichero* del Menú Principal
- Sólo nos queda asociar el menú recién generado al formulario libro. Para ello editamos sus propiedades y establecemos: **Módulo de Menús:** *mimenu.mmx*
- Ejecutamos y seleccionamos la nueva opción para visualizar el informe que debe presentar un aspecto similar al de la próxima transparencia

- Además de la función predefinida *Run_Report_Object*, dispuesta específicamente para la invocación de Informes, podemos utilizar *Run_Product* para invocar la ejecución de cualquier módulo de Oracle. La utilización de este procedimiento se puede consultar en la ayuda interactiva
- Disponemos también de los procedimientos *Disable_item('nomb_menu', 'elemento')* y *Enable_item()* para deshabilitar y habilitar, respectivamente, un elemento del menú dependiendo de la fase de operación sobre la aplicación en la que estemos



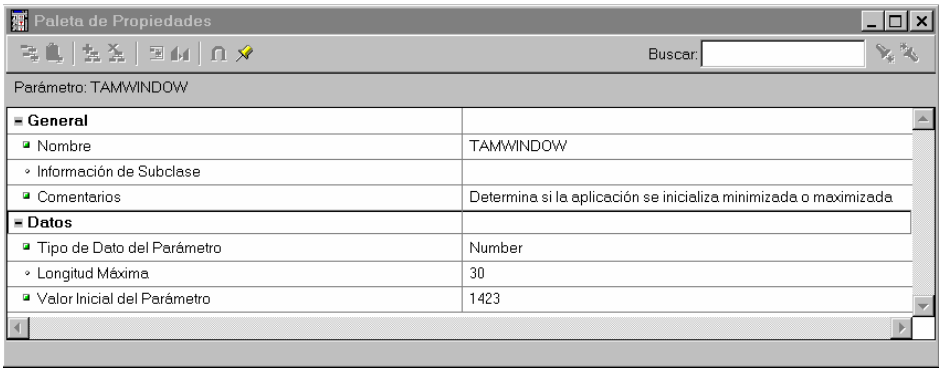
# Menús




Aspecto del formulario libro con el informe en ejecución

# Creación y Uso de Parámetros

- Un parámetro es una variable asociada a un Formulario, Informe o Gráfico que parametriza el funcionamiento del módulo en función del valor que se envíe en el momento de ejecución del módulo (línea de comandos o invocación desde otro módulo)
- Para Formularios e Informes se pueden utilizar cuadros de diálogo mediante los que establecer los valores de los parámetros
- **Creación de Parámetros en Formularios**. Vamos a crear un parámetro para determinar si la aplicación de ejecuta maximizada o minimizada.
  - » Con el formulario libro abierto, se pulsa create en sobre la clase *Parámetros*. Y se rellena como en la figura






# Creación y Uso de Parámetros

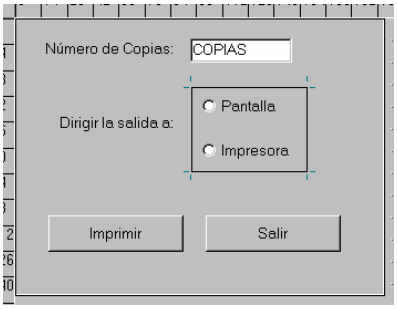
- **Utilización del parámetro**.
  - » Se crea o edita un disparador WHEN-NEW-FORM-INSTANCE para que aparezca el código:

```
BEGIN
  Set_Window_Property(FORMS_MDI_WINDOW, WINDOW_STATE, :Parameter.Tamwindow);
  llena_listas ('librocuentas.persona','persona');
  OG.Open('pagos.ogd','control.pagos');
END;
```

  - » Compilamos libro (seleccionándolo, en el menú *Programa|Compilar|Todo* )
- **Paso de argumentos**.
  - » Abrimos una ventana MS-DOS y nos situamos en `u:\CerditoValiente`
  - » Ejecutamos: `IFRUN60 libro TAMWINDOW=1421`
  - » Vemos que la aplicación se ejecuta maximizada
  - » Si paso 1422 como parámetro se ejecuta minimizada y si paso 1423 se ejecuta con el tamaño establecido en diseño
- Desde un módulo se puede invocar a otro pasándole los parámetros a través de una lista de ellos previamente construida

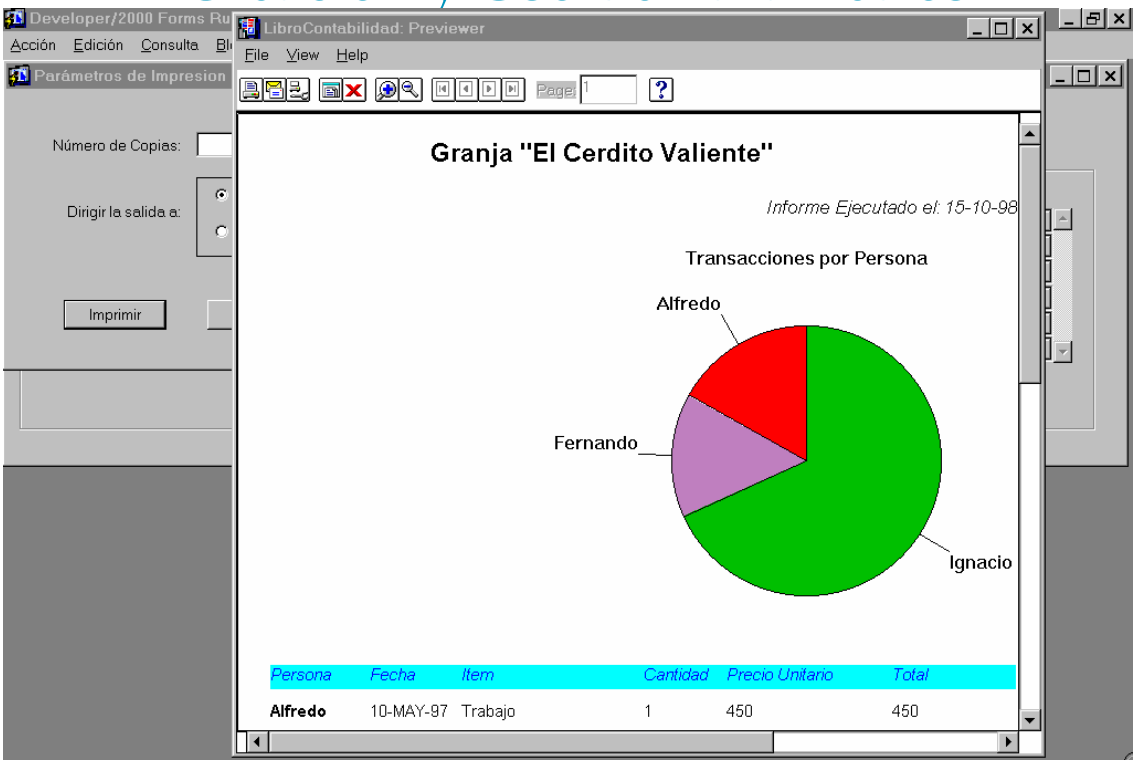
# Creación y Uso de Parámetros

- Vamos a modificar la opción Libro Cont. del menú personalizado para que invoque al informe a través de una lista de parámetros cuyo valor vamos a definir previamente desde una ventana de diálogo del formulario
  - » Creación de la ventana de diálogo.
    - Cread una nueva ventana con las propiedades: **Nombre**: *Impresion*, **Título**: *Parámetros de Impresión*, **Estilo de Ventana** *Documento*, **Modal**: *Sí*, **Cierre Permitido**, **Dimensionamiento**, **Maximización** y **Minimización PermitIdos**: *No*, **Ancho:** *200*, **Altura:** *150*
    - Cread un nuevo lienzo con las propiedades: **Nombre:** *Impresión*, **Ventana:** *Impresión*, **Ancho:** *200*, **Altura:** *150*
    - Cread manualmente un nuevo bloque con las propiedades: **Nombre:** *Impresión*, **Bloque de Datos de Navegación Anterior y Siguiente:** *Impresión*, **Bloque de Datos de Base de Datos:** *No*
    - Visualizad las propiedades del bloque LibroCuentas y estableced **Bloque de Datos de Navegación Anterior y Siguiente:** *LibroCuentas*
    - Editad en el *Editor de Diseño* el nuevo lienzo procurando que el lienzo sea Impresión y el block también. Ubicad los componentes que aparecen en la figura






# Creación y Uso de Parámetros




Aspecto del formulario libro en ejecución con el informe LibroContabilidad invocado

# Creación y Uso de Parámetros

  - Las propiedades para el Grupo de Botones Radio son: **Nombre** *Salida*, **Valor Inicial** *screen*, **Elemento de Dase de Datos** *No*
  - Las del primer botón: **Nombre** *Pantalla*, **Etiqueta** *Pantalla*, **Valor de Botón de Radio:** *screen*
  - Las del segundo: **Nombre** *Impresora*, **Etiqueta** *Impresora*, **Valor de Botón de Radio:** *printer*

» Funcionalidad botones Salir e Imprimir:

  - Cread un disparador de tipo WHEN-BUTTON-PRESSED para el botón Salir y escribid el código: `go_block(‘librocuentas’);`
  - Cread un disparador del mismo tipo para Imprimir con el siguiente código:

```
DECLARE
  id ParamList;
BEGIN
  commit_form;
  id := Create_Parameter_List('datos');
  Add_Parameter(id,'destype',TEXT_PARAMETER,:impresion.salida);
  Add_Parameter(id,'copies',TEXT_PARAMETER,:impresion.copias);
  Add_Parameter(id,'PARAMFORM',TEXT_PARAMETER,'NO');
  Run_Product(REPORTS,’LibroContabilidad',SYNCHRONOUS, RUNTIME,FILESYSTEM,
  id,NULL);
  Destroy_Parameter_List(id);
  Go_Block(’librocuentas');
END;
```



# Creación y Uso de Parámetros

» Modificacion del menú personalizado:

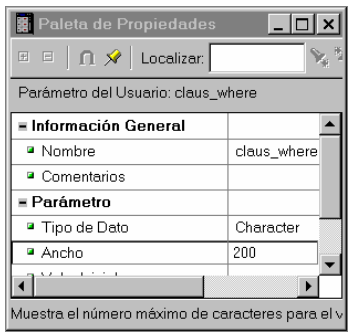
  - En la sección Menus, abrid mimenu.mmb
  - Editad el código de la opción Libro para que se lea: `go_block('impresion');`
  - Salvad y Compilad el menú *Fichero|Administración|Compilar Fichero* (o Ctrl + T)

» Ejecutad y comprobad el funcionamiento, no pulseis imprimir con la opción de salida a impresora activada puesto que no se dispone de ella

- Se pueden pasar valores de parámetros mediante la creación de una lista de parámetros, como en el ejemplo anterior, desde un formulario a otro mediante los procedimientos *New_Form(), Open_Form() y Call_Form()*
- Hemos utilizado los parámetros predefinidos de Report: *destype* con el que se establece a donde va la salida (printer, screen, preview, etc), *copies* que establece el número de copias a imprimir. Consúltese la ayuda para la utilización de otros parámetros
- Report dispone de su propia ventana para la asignación de valores en ejecución. Se edita fácilmente desde el menú pulsando *Herramientas|Constructor de Pantalla de Parámetros*
- Report también permite la definición de parámetros de usuario. Incluso permite parametrizar la consulta en que basa su ejecución

# Creación y Uso de Parámetros

- Vamos a parametrizar la cláusula Where del informe LibroContabilidad
  - » Creamos un parámetro de usuario pulsando crear sobre *Parámetros del Usuario* dentro de *Modelo de Datos*
  - » Establecemos las propiedades de la figura
  - » Seleccionamos *Herramientas|Constructor de Pantalla de Parámetros* seleccionando el parámetro CLAUS_WHERE y poniendo en **Etiqueta:** *Cláusula Where*
  - » Editamos, a través de sus propiedades, la consulta Q_1 (bajo la clase *Consultas* de *Modelo de Datos*) para que ponga:

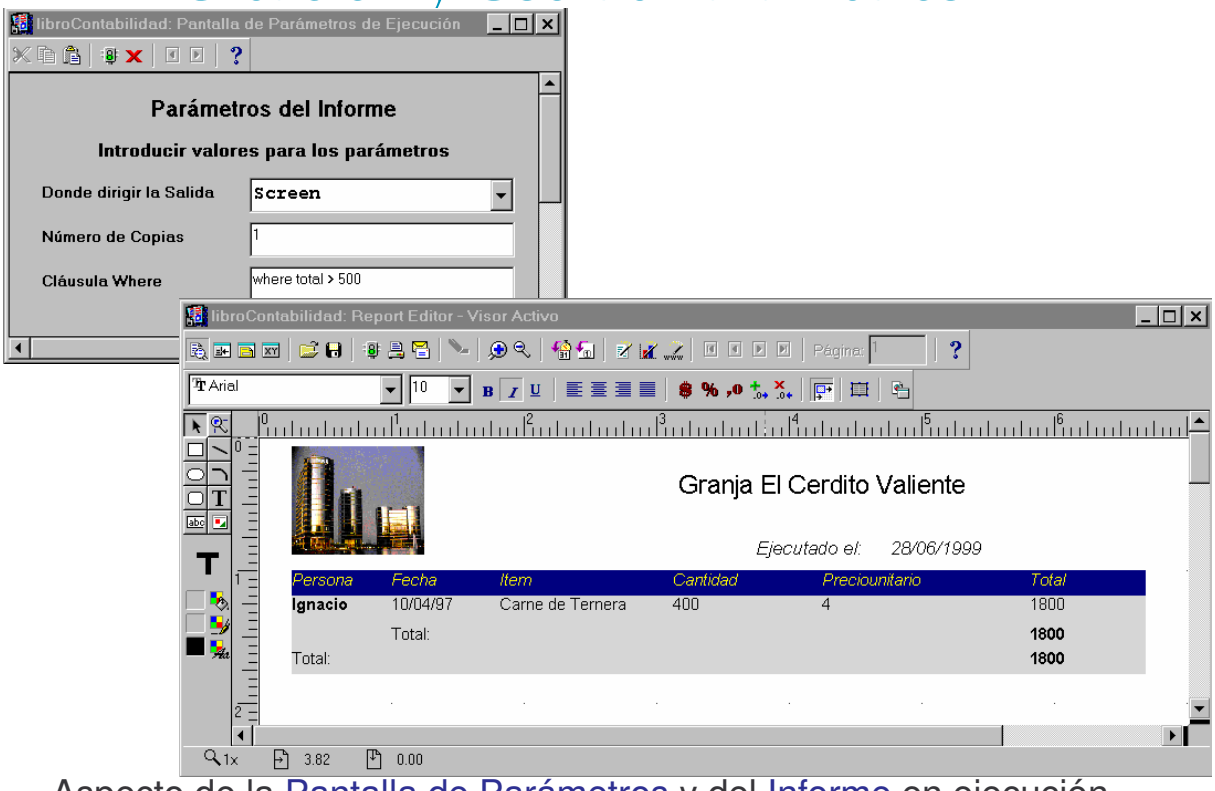




  - » Ejecutamos el informe y, para el último parámetro, introducimos, por ejemplo: *where Total > 500*



# Creación y Uso de Parámetros




Aspecto de la Pantalla de Parámetros y del Informe en ejecución

# Empaquetado y Distribución de Aplicaciones

- La implantación de una aplicación Developer es una tarea compleja ya que puede implicar diversas tareas de administración para su puesta en funcionamiento
- La relación de tareas necesarias depende de diferentes factores:
  - » Modo de publicación: Web o Cliente/Servidor
  - » Configuración actual de los servidores y clientes: disposición, o no, del entorno "runtime" de Developer en los clientes, disposición de los servidores de Bases de Datos, servidores Web, servidores de ficheros, etc.
  - » El grado de compromiso contractual con el cliente: si estamos desarrollando para un tercero este grado determina los cometidos en cuanto a la instalación y administración de las herramientas necesarias para el funcionamiento o, si el cliente dispone de un administrador, suministrar la información precisa para la correcta configuración e integración de nuestra aplicación con el sistema
- Los **requisitos comunes** para que una aplicación funcione son:
  - » Disponer de un servidor de Bases de Datos (preferiblemente Oracle aunque se puede acceder a cualquier Servidor mediante ODBC) configurado adecuadamente
  - » Disponer de una red (preferiblemente con soporte TCP/IP) con PC clientes debidamente configurados y conectados con el servidor o servidores. Sería interesante disponer de un servidor de archivos que permita a los clientes compartir ficheros comunes



# Empaquetado y Distribución de Aplicaciones

  - » Crear o adaptar los esquemas utilizados por la aplicación (creación de usuarios y métodos de acceso, gestión de recursos, creación de tablas vistas y otros objetos con los adecuados privilegios de acceso) y, en su caso, cargar los datos en esta base de datos
- Si distribuimos para un entorno **Cliente/Servidor**:
  - » Los PCs clientes deben estar configurados para poder acceder al Servidor de Bases de Datos (Cliente de SQL*Net para el caso de Servidores Oracle) y con el componente "runtime" de Developer (Form Runtime, Report Runtime, Graphics Runtime, etc.). Existen dos formas de realizar esta instalación:
    - – Utilizar el CD-ROM de instalación de Developer y, al ejecutar el instalador, seleccionar la opción Deployment Instalación, opción ésta que instala todo lo necesario
    - – Si se dispone de un servidor de ficheros, utilizar "Oracle Software Client Manager", que es un modo de instalación que permite instalar el software en un directorio del servidor accesible por los PCs clientes y una pequeña parte, relativa a la configuración particular de cada PC, en el directorio local de los PCs. Además, gestiona de forma automática la actualización de todo el software y configuración de cada PC cliente
  - » Debemos empaquetar convenientemente los módulos que componen la aplicación propiamente dicha (versiones ejecutables de los formularios, informes, etc.). *Proyect Builder* facilita enormemente esta tarea, veamos como podemos empaquetar nuestro proyecto

# Gestión del Proyecto con Proyect Builder

- Esta herramienta permite la integración y gestión de todos los módulos (formularios, informes, gráficos, bibliotecas, etc.) que componen un proyecto
- Copiad la carpeta *h:\ccia\orawin95\pj60* a *u:*
- Ejecutad *Proyect Builder* y utilizad el *Asistente de Proyectos*.
- Introducid los siguientes datos en las ventanas del Asistente:
  - **Nombre de Fichero del Registro del Proyecto:** *u:\CerditoValiente*
  - **Titulo:** *Granja El Cerdito Valiente*
  - **Directorio del Proyecto:** *u:\CerditoValiente*
- Continuad e introducid vuestro nombre y los comentarios que describan el proyecto
- En la siguiente ventana seleccionad *Crear un Proyecto Vacío*



# Gestión del Proyecto con Proyect Builder

- Continuad e introducid vuestro nombre y los comentarios que describan el proyecto
- Ventajas del empleo de *Proyect Builder:*
  - Centraliza la ejecución de las diversas herramientas de Developer
  - Organiza los archivos en varios directorios incluso en ordenadores diferentes
  - Proporciona una interfaz a herramientas de control de versiones como PVCS
  - Permite la actualización y generación automática de archivos en base a dependencias. Por ejemplo, si una serie de aplicaciones utilizan un módulo PL/SQL y variamos éste, todas ellas se actualizan
  - Permite automatizar el proceso de empaquetado de la aplicación para su implantación. Pueden programarse tareas en este proceso
  - Permite adaptar el entorno a las necesidades de cada proyecto
  - Permite la creación de subproyectos y la compartición entre equipos de desarrollo

# Empaquetado y Distribución de Aplicaciones

- En primer lugar debemos disponer de un programa de compresión o de un programa de creación de instalaciones. En nuestro caso vamos a utilizar Winzip.
- Con Proyect Builder en ejecución vamos a utilizar el proyecto CerditoValiente anteriormente creado
- Con este proyecto seleccionado pulsamos desde el menú *Proyecto|Añadir Ficheros al Proyecto*. Navegamos al directorio `u:\CerditoValiente` y realizamos una selección múltiple de todos los ficheros salvo los .bak y otros que no formen parte de la aplicación. Aceptamos para incorporarlos a nuestro proyecto
- Nos aseguramos de que tenemos definida adecuadamente la conexión oracle0 en el apartado *Conexiones* y de que la tenemos asignada a nuestro proyecto
- Antes de empaquetar debemos asegurarnos que todos los ficheros están compilados. Para ello, con el proyecto seleccionado podemos pulsar *Proyecto|Crear Incremental* o, mejor, *Proyecto|Crear Todo* para compilarlos todos
- *Proyect Builder* invoca al correspondiente compilador de cada fichero y lo compila automáticamente, salvo para los ficheros de gráficos que, al no existir un compilador separado, los abre en Graphics Builder y tenemos que desencadenar manualmente la compilación (*Fichero|Compilar Todo*) y salir



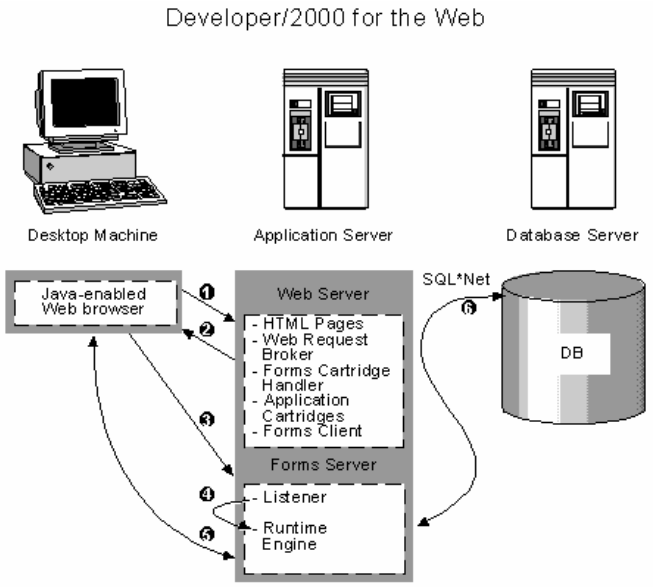
# Empaquetado y Distribución de Aplicaciones

- Cada uno de los ficheros que cuelgan de nuestro proyecto dispone de la propiedad **Entregar Fichero** (*Sí|No*) que determina si será incluido o no en el paquete
- Por defecto *Proyect Builder* incluye sólo los ejecutables, que es la forma habitual de distribuir la aplicación, nosotros vamos a incluir también los fuentes para que tengamos una copia de nuestro trabajo
- Para ello, seleccionamos el proyecto y pulsamos *Editar|Seleccionar Todo*, abrimos la *Paleta de Propiedades* y ponemos a *Sí* **Entregar Fichero**
- Antes de empaquetar, debemos modificar la propiedad **Entregar los Ficheros Seleccionados** de Cerdito Valiente para que utilice winzip en la forma:
  `{Packfile ? winzip32 -a {Packfile} {1}}`
- Y determinar el nombre y ubicación del fichero comprimido. Establecemos la propiedad **PACKFILE** con el valor `u:\CerditoValiente\cerdito.zip`
- Ya podemos empaquetar. Para ello utilizamos el asistente de entregas *Herramientas|Asistente de Entregas.* Activando la opción *Usar la acción DELIVER definida en mi proyecto,* y aceptando el resto de las opciones

- Si distribuimos para **entorno Web**:
  - Los PCs cliente sólo deben disponer de un navegador Web con soporte para Java 1.1.x (por ejemplo: Explorer 4.x o Netscape 4.0x al que hay que aplicar un parche) o de la versión ejecutable del jdk1.1.x (jre.1.1.x) y de conexión TCP/IP con el/los servidor/es
  - Un servidor Web debidamente configurado e inicializado

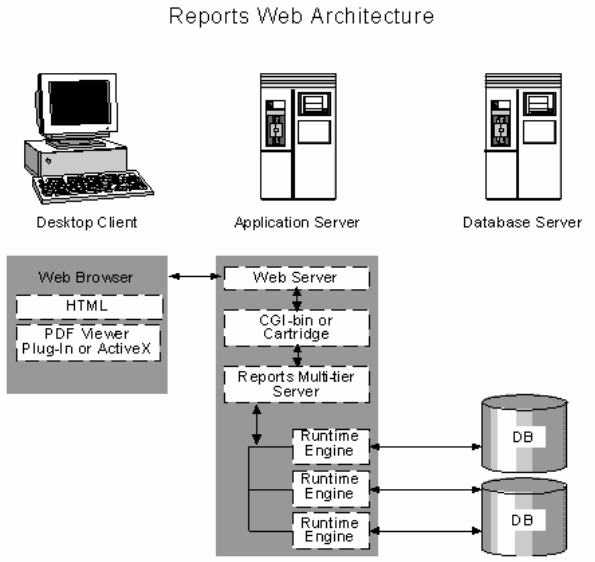
# Empaquetado y Distribución de Aplicaciones

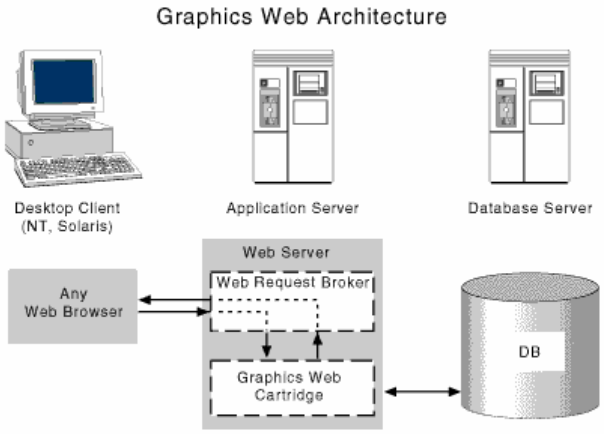
- El servidor de formularios para la Web configurado e inicializado
- El servidor de Informes para la Web configurado e inicializado
- El servidor de Gráficos para la Web configurado e inicializado y un módulo denominado Web Request Broker para conectar con el servidor Web (válido para servidores Netscape, Microsoft y Oracle)
- Los módulos que componen la aplicación compilados para el sistema operativo del servidor de aplicaciones y ubicados en los directorios configurados al efecto
- Las páginas html desde las que se invocan a los correspondientes módulos






# Empaquetado y Distribución de Aplicaciones



# Reutilización

- Mecanismos para beneficiarse de la reutilización:
  - » Utilización de Plantillas: Formularios, Informes, Gráficos
  - » Consultas externas (Query Builder) y parametrizadas: referencias (&) y variables ligadas (:variable)
  - » Atributos visuales nominados
  - » Clases de propiedades
  - » Subclases, copia y referencia de objetos. Creación de bibliotecas de objetos
  - » Parametrización, variables y paso de argumentos
  - » Aplicaciones multiformulario. Reutilización de un formulario en varias aplicaciones
- Plantillas de Formulario
  - » No existen como tal, se puede construir un formulario basado en otro formulario a partir de la ventana inicial *"Bienvenido a Form Builder"* seleccionando la opción *"Crear un form basado en una plantilla"*
  - » Se abre el formulario del que se desea partir, el que actuaría como plantilla
  - » Sobre esta plantilla se completa el diseño
  - » Para beneficiarse de esta aproximación el formulario plantilla debería disponer de los elementos comunes de un conjunto de formularios que se basen en él



# Reutilización. Plantillas

- » Se pueden estandarizar mediante plantillas de formularios algunos de los siguientes objetos que los componen:
  - – Formulario: Estableciendo valores para las propiedades **Sistema de Coordenadas**, **Módulo de Menús** y otras opciones de configuración
  - – Disparadores del Formulario: Acciones en el disparador When-New-Form para configurar tamaños estándar y demás operaciones de inicialización
  - – Alertas: Alertas estándar de error y aviso
  - – Bibliotecas Incorporadas: Bibliotecas estándar compartidas por múltiples formularios
  - – Lienzos: Lienzo estándar con configuración de reglas y diseño
  - – Parametros: Parámetros utilizados por todos los formularios
  - – Menús Desplegables: Menús emergentes compartidos por todas las aplicaciones
  - – Unidades de Programa: Módulos de de Programa específicos de formulario
  - – Informes: Informes estándar compartidos por todos los formularios
  - – Ventanas: Ventanas y caja de diálogo estándar
- Plantillas de Informes
  - » Archivo que contiene texto y gráficos junto con campos de ejemplo para diferentes tipos de datos: caracteres, números y fechas
  - » Developer proporciona una especie de plantillas genéricas que hay que adaptar para que cumplan con nuestros propósitos

# Reutilización. Plantillas

- Ejercicio
  - Estas plantillas se encuentran en `h:\ccia\orawin95\reports60\admin\template\us`
  - Abrimos desde Report Builder la plantilla corp1.tdf y la salvamos como `u:\CerditoValiente\cerdito.tdf`
  - Copiamos `h:\ccia\orawin95\tmp\granja.bmp` a `u:\CerditoValiente\granja.bmp`
  - Eliminamos la imagen de la plantilla
  - Con la sección Margen marcada, creamos  y dimensionamos un Fichero de Enlace cuyas propiedades son: **Nombre** *Cerdito*, **Formato de Fichero Origen:** *Imagen* y **Nombre de Fichero de Origen:** *granja.bmp*
  - Cambiamos el Título por: *Granja el Cerdito Valiente* y *Report Run on*: por *Informe ejecutado el:*
  - La propiedad **Máscara de Formato** de F_DATE1 la establecemos a: *DD-MM-RRRR*
  - Salvamos
  - Creamos nuevamente el informe LibroContabilidad.rdf utilizando esta plantilla (selecionando Fichero de Plantilla en el Asistente de Informes)
- En la sección Cuerpo se pueden personalizar todos los parámetros de visualización de los datos del informe
- También se pueden asociar bibliotecas y unidades de programa para facilitar su utilización desde los informes



# Reutilización. Atributos Visuales

- Atributos Visuales
  - En la sección Atributos Visuales se pueden crear objetos que recogen las características visuales de los elementos.
  - Las propiedades que se pueden establecer son:
    - **Nombre de la Fuente, Tamaño de la Fuente, Grosor de la Fuente, Estilo de la Fuente, Espaciado de la Fuente, Color del Primer Plano, Color del Fondo, Trama de Relleno, Atributo Lógico de Modo Carácter** (para la versión de la aplicación en modo carácter), **Blanco sobre Negro** (si se pone a Sí, texto blanco sobre negro para pantallas monocromas)
  - Para que un elemento asuma las propiedades de un objeto "Atributo Visual" establecemos el nombre de ese objeto en la propiedad **Grupo de Atributos Visuales**
  - Cuando se modifican las propiedades del objeto Atributo Visual, se modifican las de todos los elementos que lo utilizan
  - Deberían crearse atributos visuales nominados para cada tipo de atributos e incluirlos en las plantillas de Formulario
- Clases de Propiedades
  - En la sección Clases de Propiedad se pueden crear objetos que implementan características de herencia en las aplicaciones
  - Posibilita establecer propiedades funcionales además de las visuales
  - Para crear una clase de propiedades se pulsa create en la sección Clases de Propiedad o se abre la hoja de propiedades de un objeto existente y se pulsa la herramienta Clase de Propiedad  de la hoja para crear una Clase de Propiedad basada en ese objeto

# Reutilización. Clases de Propiedades

- Se añaden propiedades a la clase mediante  , se eliminan mediante  , se copian mediante  y se pegan mediante 
- Una clase de propiedades puede heredar las propiedades de otra clase de propiedades
- Para que un elemento herede las propiedades de una clase de propiedades hay que establecerla en su propiedad **Información de Subclase**, desde esta propiedad también se pueden heredar las propiedades de cualquier objeto existente
- Las propiedades heredadas se pueden sobreescribir (reemplazar el valor heredado) poniendo el nuevo valor en la propiedad del objeto. Se puede recuperar el valor heredado seleccionando la propiedad y pulsando 
- La utilización de disparadores en las clases de propiedades es posible, sólo que es preciso generalizar su funcionamiento

- Subclases, copia y referencia de Objetos
  - Desde el Navegador de Objetos se puede utilizar el mecanismo de arrastrar y soltar para mover un objeto de ubicación y para crear un objeto o referencia al mismo (si mantenemos pulsada la tecla Ctrl durante la operación de arrastre). No es muy aconsejable crear referencias
  - Las subclases basadas en objetos se crean de forma similar a la empleada para utilizar las clases de propiedades



# Reutilización. Grupos de Objetos

- Bibliotecas de Objetos
  - Es un contenedor de objetos de tipo Formulario
  - Se pueden incluir cualquier número de objetos individuales, como bloques, elementos, lienzos, ventanas o clases de propiedades
  - Actualmente no se pueden almacenar objetos de tipo menú, informe o gráfico
  - La biblioteca se crea desde la sección Grupos de Objetos.
  - Una doble pulsación sobre la nueva biblioteca abre el editor de Grupos de Objetos que muestra dos fichas para organizar los objetos que se incluyan
  - El nombre y número de fichas se puede variar para lograr una mejor organización de los objetos introducidos
  - Para copiar objetos a esta biblioteca sólo es preciso arrastrarlos desde el Navegador de Objetos a la pestaña correspondiente
  - Un vez completada nuestra biblioteca es preciso salvarla en nuestro directorio
  - Para utilizarla se abre en la sección Grupos de Objetos y se arrastran los objetos que contienen hasta el lugar correspondiente del formulario
  - Se nos preguntará si queremos copiar el objeto o crear una clase del objeto, en el último caso las modificaciones que se hagan en el objeto de la biblioteca se reflejarán en el objeto del formulario
  - Un objeto de una biblioteca puede convertirse en una “Smart Class” y figurar en el menú que se despliega con el botón derecho sobre cualquier elemento bajo la entrada SmartClasses

# Reutilización. Ejercicios

- Para ello seleccionamos el objeto deseado en el editor de Grupos de Objetos de la biblioteca de objetos
- Pulsamos la opción de menú Objeto|SmartClass, aparecerá una marca verde junto a ese objeto en dicho editor
- Para verificarlo seleccionamos en el formulario un objeto del mismo tipo y pulsamos el botón derecho, debe aparecer ese objeto en la entrada SmartClass

- Ejercicios de reutilización
  - Cread un objeto de tipo Atributo Visual que especifique las propiedades visuales para vuestros elementos
  - Cread una Clase de Propiedad a partir del elmento *nombre* del bloque *persona* del formulario RecursosHumanos
  - Utilizad los mecanismos de arrastrar y soltar sobre objetos
  - Cread un Grupo de Objetos y disponed en ella pestañas para elementos, bloques, Clases de Propiedad y Atributos Visuales. Copiad en ella objetos de cada una de estas categorías tomados de los ejercicios anteriores y del formulario RecursosHumanos
  - Cread un nuevo formulario llamado plantilla.fmb que responda a los siguientes requisitos (podéis recurrir a algunos de los elementos disponibles en la biblioteca anterior):



# Reutilización. Ejercicios

  - Que utilice el menú mimenu
  - Que disponga de alertas de información y de error genéricas. Modificad los procedimientos disponibles en mibiblioteca para que utilicen estas alertas de forma parametrizada
  - Que tenga enlazada la biblioteca mibiblioteca
  - Que disponga de un disparador When-New-Form-Instance a nivel de formulario que visualice una ventana de bienvenida a través de las alertas definidas y que ejecute la ventana de aplicación maximizada
  - Que disponga de las Clases de Propiedades y de los Atributos Visuales definidos anteriormente
- Salvad esta plantilla y utilizadla como base para crear una nueva versión de RecursosHumanos llamada RecursosHumanosb.fmb. Podeís utilizar también la biblioteca de objetos

# Bibliografía


- Manual de Oracle Developer/2000. Robert J. Muller. Oracle Press. McGraw-Hill. 1998
- Oracle8. Programación PL/SQL. Scott Urman. Oracle Press. McGraw-Hill. 1998