http://www.ucentral.edu.ec/facultades/facinge/manualpower/Principal.html

Introducción

Power Builder

Qué es Power Builder ?

PowerBuilder es un ambiente para desdarrollar aplicaciones graficas. Usando PowerBuilder, usted puede facilmente desarrollar poderosas aplicaciones grafica que accesa a servidores de base de datos. PowerBuilder provee todas las herramientas que Ud. necesita para construir aplicaciones industriales , tales como , contabilidad, sistemas de manufactura, ect. PowerBuilder es un entorno de desarrollo comprensivo para construir aplicaciones cliente /servidor de alto desempeño para la familia Windows, que combina una interface gráfica intuitiva con un poderoso lenguaje de programación orientado a objetos. Power Builder soporta multi-plataformas desarrolladas y desplegadas. Por ejemplo, Ud. puede desarrollar una aplicacion usando PowerBuilder bajo windows(Win'95 o Win NT) y desplegar la misma aplicacion -sin hacer cambios- sobre máquinas Win 3.11, Macintosh, o Unix.

Desarrolo en Internet: Power Builder incluye herramientas que le permiten construir aplicaciones basadas en Web y extender la existencia de su aplicacion al Internet. Es un front-end que puede interactuar con la mayoría de DBMS basados en ODBC

Acerca de los Pintores(Painters)

Se puede construir los componentes de una aplicación usando pintores, los cuales proveen una variedad de herramientas para construir objetos. Power Builder provee un pintor para cada tipo de objeto que se puede construir. Por ejemplo: se puede construir una ventana con el Pintor de Ventanas(Window painter).

Acerca de Eventos y Scripts

Las aplicaciones de Power Builder son manejadoras de eventos: los usuarios controlan el flujo de la aplicación . Cuando un usuario hace click en un botón, elige una opción de un menú, o ingresa datos en una caja de texto,un evento se dispara. Se encribe codigo(script) que especifica el proceso que deberia suceder cuando el evento acurre.
Por ejemplo, Buttons tiene un evento Clicked. Se escribe un script(código) para el evento clicked de Buttton que especifica que sucede cuando el usuarios hace click en el botón.
Se escribe Script usando PowerScript del lenguaje Power Builder.
Un **Script** consiste de comandos de PowerScript, funciones, y sentencias que realizan un proceso en respuesta a un evento.

Acerca de las Librerias

Se puede gravar objetos, tales como , ventanas y menúes, en Librerias de PowerBuilder(archivos .PBL). Cuando se corre una aplicación, PowerBuilder recupera los objetos de la libreria. PowerBuilder provee un pintor Library(Libreria) para manejar librerias.

Creando un ejecutable

Cuando se ha creado una aplicación completa, se puede crear un ejecutable de la aplicacion para dar a diferentes usuarios y lo utilicen.

Explicación de Front-End y Back-End

Un front-end es un constructor de interfaces, es una herramienta de programación donde se definen los formatos mediante los cuales se van a visualizar y manipular los datos. Un back-end es la herramienta que almacena los datos y los entrega al front-end para su manipulación

Explicación de Cliente/Servidor.

Cliente/Servidor es una organización de procesos, donde un proceso específico al que se le denomina servidor se dedica exclusivamente a atender los requerimientos que le envían, un grupo de procesos denominados clientes.

Objetos de PowerBuilder

· PowerBuilder es una herramienta orientada a objetos.
· Cada objeto tiene sus propios atributos y eventos

Objeto

Un objeto es cualquier entidad o cosa que se pueda representar o concebir mediante una serie de características que lo definan

Atributos

Un atributo es una característica que define al objeto.

Evento

Es una circunstancia a la cual se asocia una porción de código de programación, que se ejecuta cuando el evento se dispara. Ej: click del mouse, al abrir una ventana, al hacer doble-click, etc..
Cada evento tiene asociado un espacio en donde se puede programar, a este espacio se le conoce como script.

Los tipos de objetos más importantes son:

- Aplicación
- Ventana
- Menús
- DataWindows, ChildDatawindows
- Gráficos
- ListBox
- DropDownListBox
- Multilineedi
- CommandButton

- PictureButton
- Editmask
- Checkbox
- Radiobutton
- Groupbox, etc...

Definición de SQLCA

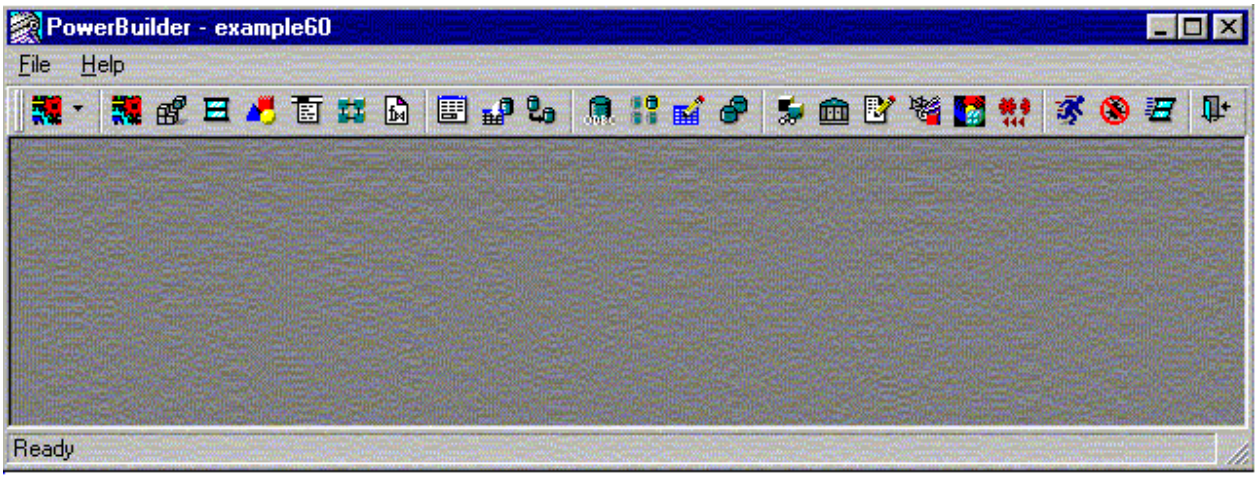
SQL Communications Area(SQLCA) es un objeto transaccional. Un objeto transaccional es el área de comunicación entre el script (lugar donde se programa un evento) y la base de datos. PowerBuilder define este tipo de objeto para facilitar la comunicación con la base de datos desde el código de programación. Este objeto es accesado por default, cada vez que se utiliza una sentencia SQL dentro de un script.

Convenciones de Nomenclatura de Objetos

En PowerBuilder se acostumbra utilizar la primera letra del objeto con un underscore antes del nombre en sí de la instancia del objeto
Ej: ventana w_alumno, datawindow dw_calculo, cb_cerrar, etc.. w_ ventanas, dw_ datawindows, m_ menus, wf_ funciones de ventana, em_ editmask, cb_ commandbutton, dddw_ dropdowndatawindow, etc..

El Entorno de PowerBuilder

Cuando se empieza PowerBuilder, se abre una ventana que contiene un Menú y una barra(PowerBar). Se puede utilizar pintores de PowerBuilder para crear ventanas, menues, tables para una base de datos, y otros objetos que se necesite para la aplicación.



Acerca del PowerBar

La barra PowerBar se despliega cuando se inicia una sesión en PowerBuilder. PowerBar es un parte principal de control para construir aplicacioones en PowerBuilder. Desde esta

barra se puede abrir un pintor de PowerBuilder , correr una aplicación, personalizar el Power Builder de acuerdo a sus necesidades.

Si desea saber como utilizar cada uno de los pintores, haga click en la barra de herramientas que está a continuación.

Lista de los Pintores(Painters)

Pintor	Lo que realiza
Aplicacion	Especifica información acerca de la aplicación, tales como , su nombre, las librerias de Power Builder en las cuales los objetos de la aplicacion serán grabados
Project	Crea un ejecutable especificando los componentes que conforman la aplicación
Window	Construye una ventana que será usada en la aplicación
User Object(Objetos de usuario)	Construye objetos credaos por el usuario, que se puede grabar y usar repetidamente en una ventana.
Menu	Construye menues que las ventanas lo utilizarán
Structure(Estructura)	Define estructuras(grupo de variables) para usarlas en la aplicacion
Function(Función)	Construye funciones para desempeñar un proceso específico en la aplicación
DataWindow	Construye objetos inteligentes llamados objetos DataWindow que presentan información de una Base de datos.
Report(Reporte)	Construye vistas previas o reportes(un objetos DataWindow sin la capacidad de actualizar)
Run Report(ejecuta un reporte)	Realiza una vista previa de un reporte
Query(Consultas)	Realiza consultas graficas con sentecias SQL SELECT .
Data PipeLine(Tuberia de datos)	Transfiere datos desde una fuente de datos a otra
Configurar ODBC	Define una Base de Datos que usa ODBC
Databases Profiles(Perfiles de Base de datos)	Define y usa nombres puestos como parametros para conectarse a una base de datos en particular
Table(Tabla)	Crea tablas para una base de datos, altera tablas existentes, define claves primarias, relaciones entre tablas, indices

Database(Base de datos)	Sirve para mantener una base de datos, un usuario puede controlar el acceso a la base de datos y manipular los datos de una base de datos.
Database Administration(Administrador de base de datos)	Desempeña tareas de administracion de una base de datos,tales como, mantenimiento por parte del usuario y seguridades.
Browser	Permite visualizar acerca de los objetos del sistema y los objetos de una aplicación, tales como, propiedades, eventos, funciones ,variables globales.
Library (Libreria)	Crea y mantiene librerias de objetos de PowerBuilder.
File Editor(Editor de archivo)	Edita archivos de texto, tales como, fuentes, archivos de inicialización.
Run(Correr)	Ejecuta la aplicación actual que esta cargada en PowerBuilder
Debug	Corre una aplicacion paso a paso, permitiendo colocar puntos de quiebre.
Run Window(Corre una ventana)	Corre una ventana sinple en la aplicación.
System Options (Opciones de sistema)	Coloca preferencias del PowerBuilder, tales como, camino(path) de inicialzación perfiles preferidos, etc.
Help	Invoca a la ayuda en línea de PowerBuilder. Presionar F1. o elegir la Opción Help

Pintor Aplicación(Application Painter)

Vista Global de un Objeto Aplicación
Una **Aplicación** es una colección de ventanas de PowerBuilder que desempeñan actividades relacionadas.

El ***Objeto Aplicación*** es el punto de entrada dentro de la ventana que desempeña estas actividades. Cuando un usuario corre una aplicación, el script(código) que es escrito en los eventos son disparados en el objeto Aplicación.

Eventos en el objeto Aplicación

Evento	Lo que ocurre cuando se dispara
Close	Cuando el usuario cierra la aplicación.
ConnetionBegin	Cuando una aplicación Cliente intenta establecer una conexión a la aplicación servidor. Este evento se dispara solo cuando una aplicación servidor esta corriendo en un ambiente distribuido.
ConnectionEnd	Cuando la conexión de una aplicación cliente es terminada. Este evento se dispara solo cuando una aplicación servidor esta corriendo en un ambiente distribuido.
Idle	Cuando la Función Idle ha sido llamada en el script del objeto aplicación y se especifica el número de segundos que han transcurrido cuando el mouse o teclado no están en actividad.
Open	Cuando el usuario corre la aplicación.
SystemError	Cuando en tiempo de ejecución ocurren serios errores.

Creando una Nueva Aplicación

El Primer Paso para Construir una nueva aplicación en PowerBuilder es crear un objeto aplicación para la aplicación.

Pasos para crear una aplicación

1. Haga click en el pintor Aplicación de la barra de herramientas PowerBar y aparece un ambiente de trabajo del objeto Application.

2. Selecione del menú principal en la opción **File** la opción **New** y luego aparece una ventana en donde ingresa un nuevo nombre o elige uno ya existente para el archivo .PBL principal.

3. Luego aparece la ventana (Fig. 1) en donde se debe ingresar el nombre de la aplicación. Además puede ingresar un comentario de la aplicación (este esopcional).

4. Por último debe elegir la libreria en donde se va almacenar todos los objetos creados en la aplicación y presione OK.

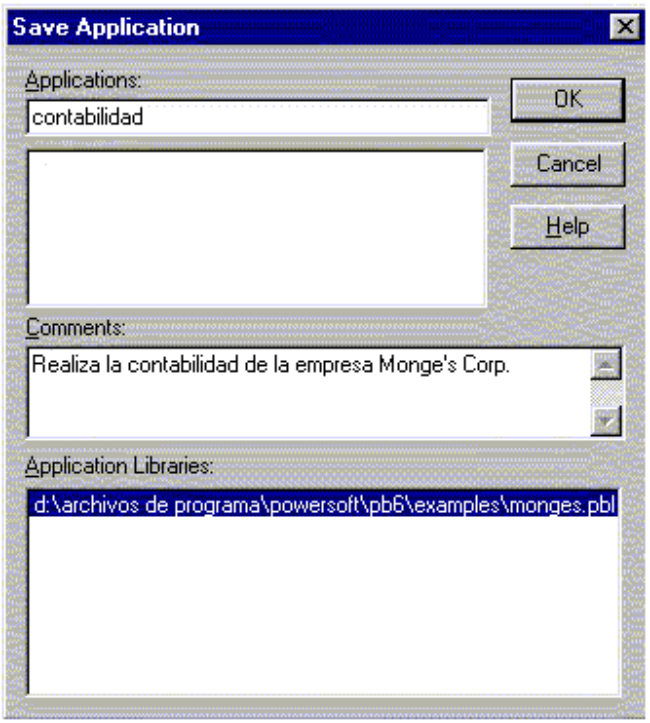


Fig. 1

Pintor Proyecto (Project Painter)

Acerca del Pintor Project:
Existen dos maneras básicas de empaquetar y crear una aplicación.

1. Como un archivo ejecutable independiente que contiene todos los objetos de la aplicación.

2. Como un archivo ejecutable y una o más librerias dinámicas que contienen objetos que son ligados en tiempo de ejecución.

El pintor Project permite que por una línea de flujo la generación de archivos ejecutables y librerias dinámicas. Cuando se quiere contruir una objeto project, se debe especificar los siguientes componentes de la aplicación:

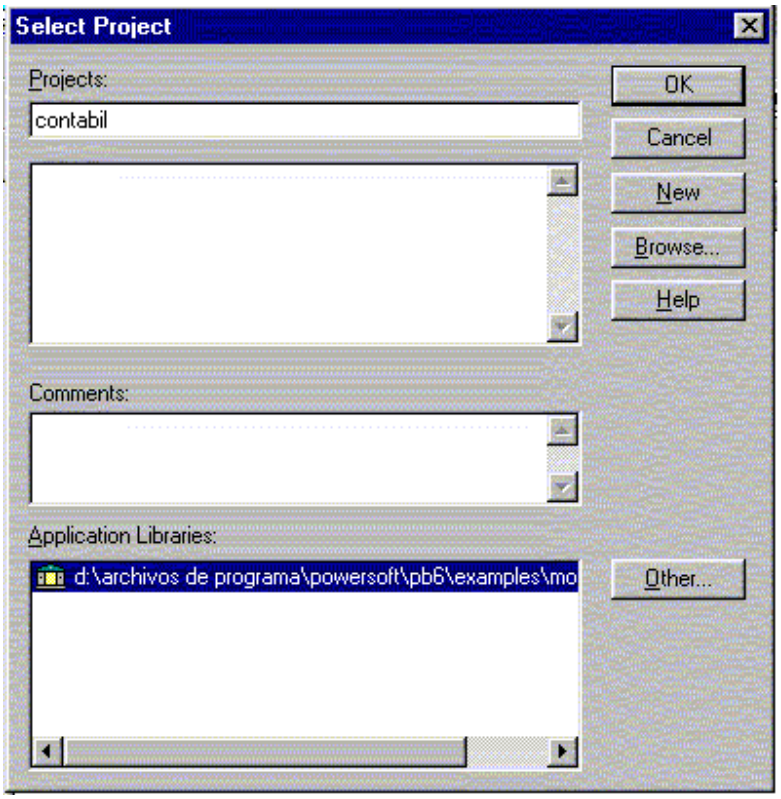
- Nombre del archivo ejecutable
- Cuales de las librerias quiere que se distribuyan como librerias dinámicas.
- Qué opciones de contrucción desea para usar en el proyecto.
- Qué opciones de generación de código desea usar.

Una vez que se ha definido el proyecto , ya se puede construir la aplicación con solo hacer click en el botón Build(contruir).

Contruyendo una Aplicación :

Pasos para crear un proyecto

1. Haga click en el pintor Project de la barra de herramientas PowerBar y aparece la pantalla siguiente:



2. Puede elejir un proyecto que exista ya o ingresar uno nuevo. Si ya existe elija el archivo que ya existe y presiona el botón *OK* y si es uno nuevo presiona el botón en *New*(nuevo).

3. Luego aparece un espacio de trabajo para crear el protecto (Fig. 2), en donde se ingresa el nombre del archivo ejecutable y varias de la opciones que se pueden ver en la figura(más adelante se explican estas opciones para construir una aplicación.)

4. Una vez que ha ingresado todos los datos, en el menú pricipal en la opción ***Design***(diseño) elija la opción ***Build Project***(contruir proyecto) para contruir un ejecutable de la aplicación.

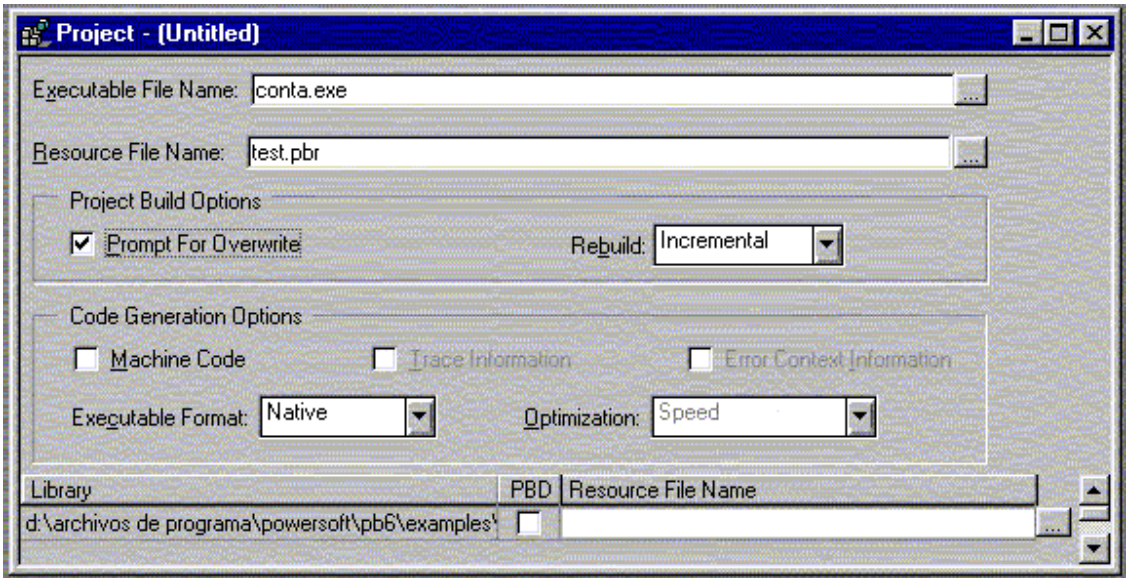


Fig. 2

Explicación de las Opciones para contruir una Aplicación:
Opciones del Pintor Project:
Executable File Name*(Nombre del archivo ejecutable)*
El nombre que se especifica para el ejecutable debe tener una extensión .EXE en la plataforma Windows.

Resource File Name*(Nombre del archivo de recursos)*
Se necesita especificar un archivo de recursos en PowerBuilder(.PBR File) para el archivo ejecutable si dinamicamente se hace referencia a recursos (tales como bitmaps y iconos) en algún script y si se quiere incluir recursos en el archivo ejecutable en lugar de tener que distribuir los recursos separadamente.

Project Build Options (Opciones para construir un proyecto)
Prompt Overwrite
PowerBuilder sobrescribe algunos archivos creados cuando esta construyendo una aplicación. Elija esta opción si desea sobreescribir. ***Rebuild***
Especifica una de las dos o Full o Incremental de una lista deplegable Rebuild indicando si se se quiere regenerar todos los onjetos en la librerias de la aplicación antes de hacerlo ejecutable ejecutable y librerias dinámicas, opción Full. Si elije Incremental, PowerBuilder regenera aquellos objetos referenciados por objetos que han cambiando desde la última vez que se contruyó la aplicación.

Code generation options (Opciones para generación de código)
MUchas de estas opciones están deshabilitadas if el compilador no es soportado por la plataforma. El codigo de compilación esta soportado sobre 32 bits en Windows, Unix y Mac.

Machine Code (código de maquina)
Selecione esta opción si Ud, quiere generar código compilado el lugar de Pcode.

Trace Information (Copia de la información)
Selecciona esta opción cuando desea crear un archivo de copia cuando corre el codigo compilado.

Error context Information (Contexto de información de errores)
PowerBuilder Despliega un contextode información, tales como un objeto, eventos, algún script para errores en tiempo de ejecución.

Excutable Options
Si elije Native es sobre 32 Bits o sino sobre 16 bits. ***Dynamic Library Options (Opciones de librerias dinámicas)***
Se puede reducir el tamaño del archivo ejecutable para distribuir algunos de los objetos requeridos en una libreria dinámica.

Pintor Ventana (Window Painter)

Vista Global de una Ventana

Las Formas Windows(ventanas) dan una interface entre el usuario y una aplicación de PowerBuilder. Las ventanas(windows) pueden desplegar información, pedir información a un usuario, y respoder a las acciones que realiza el usuario con el mouse y teclado.
Una ventana consiste de :

- **Propiedades** que definen la apariencia de la ventana y su comportamiento.
- **Eventos** Una ventana tiene eventos como otros objetos de PowerBuilder
- **Controles** ubicados dentro de la ventana. Controles como: CheckBoxes,CommandButton, etc.

Tipos de Ventanas

- Main
- PoPup
- Child
- Response
- Múltiple Document Interface **(MDI) Frame**
- MDI Frame con MicroHelp

Main Windows (Ventana Pricipales)

Las Main Windows son ventanas independientes que actuan de forma independiente con el resto de las ventanas.

Si usa una Main Window como un ancla para su aplicación. La primera ventana que la aplicación abre es una main window - a menos que Ud. haya contruido una aplicación con

Multiple Document Interface (MDI) , en este caso la primera ventana que se abre es una MDI Frame.

Si Ud. quiere que una ventana siempre esté a disposición del usuario, que puede ser desplegada en cualquier momento, en cualquier parte de la pantalla.

Popup Windows (Ventanas Popup)

Las Ventanas Popup son abiertas desde otra ventana, que en la mayoria de los casos llegan a ser padres de las ventanas popup.

Las ventanas popup son utilizadas a menudo como ventanas de soporte. Por ejemplo: si se tiene una ventana que contiene información principal, tales como una lista de películas. Se puede usar una ventana Popup para permitir al usuario vea en detalle los datos de una película en particular.
Utilizando open **Open(popupwindow,parentwindow)**, por ejmeplo:
open(w_popup,w_padre).

Child Windows (Ventana Hijo)

Las ventanas hijo son siempre abiertas desde el interior de una ventana Main o Popup, que llegan a ser padres de la ventana hijo(Child window). *Una ventana Hijo existe solo dentro de una ventana padre.* Las ventanas Hijo(Child window) no pueden tener menúes, y nunca se consideran como ventanas activas. Una ventana hijo(Child window) se cierra cuando se cierra la ventana que es padre.

Response Window (Ventana de respuesta)

Las ventanas de respuesta solicitan información desde el usuario. Estas ventanas siempre son abiertas desde el interior de otra ventana(padre). Por lo general, una ventana de respuesta es abierta luego que algún evento ha ocurrido en la ventana padre.

Las ventanas de respuesta son de modo aplicación(application modal). Esto es, cuando una ventana de respuesta es desplegada, por tanto esta ventana se hace activa(obtiene el enfoque) y las demas ventanas de la aplicación no son accesibles mientras el usuarios no responda a la ventana de respuesta(response window).
Suelen usarse como cajas de mensajes, para dar información cuando ocurre un error, cuando se ha realizado alguna tarea, como se muestra en la figura siguiente.

MDI Frame Windows (marco de interface de multiples documentos)

Una MDI Frame es un marco de ventana en el que se puede abrir multiples ventanas como documentos o Sheets(hojas) y moverse entre las hojas(sheets).
Hay dos tipos de ventanas MDI Frame:

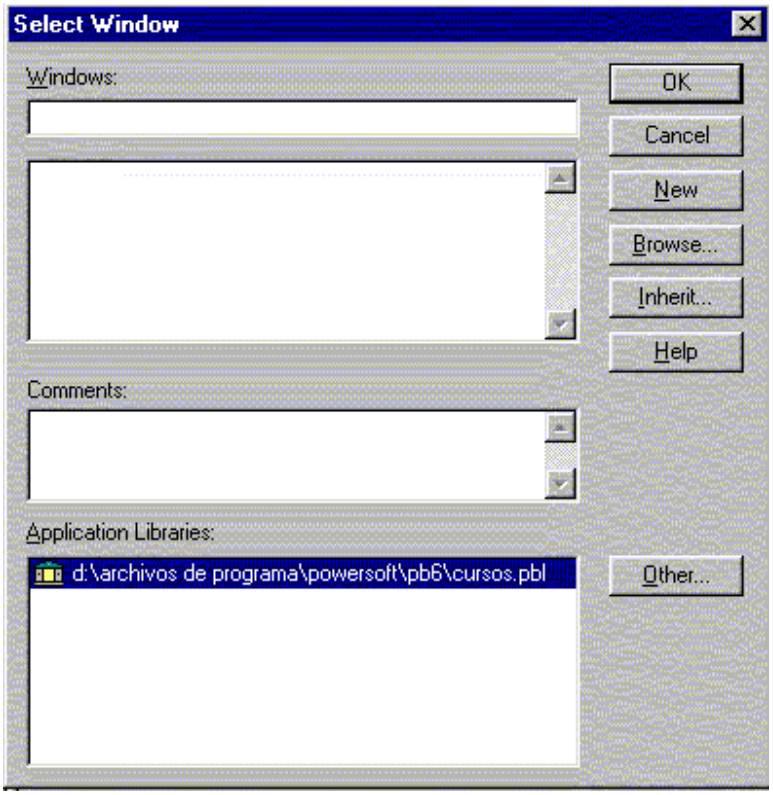
- MDI Frame
- MDI Frame with MicroHelp(con micro ayuda)

Construyendo una ventana Nueva

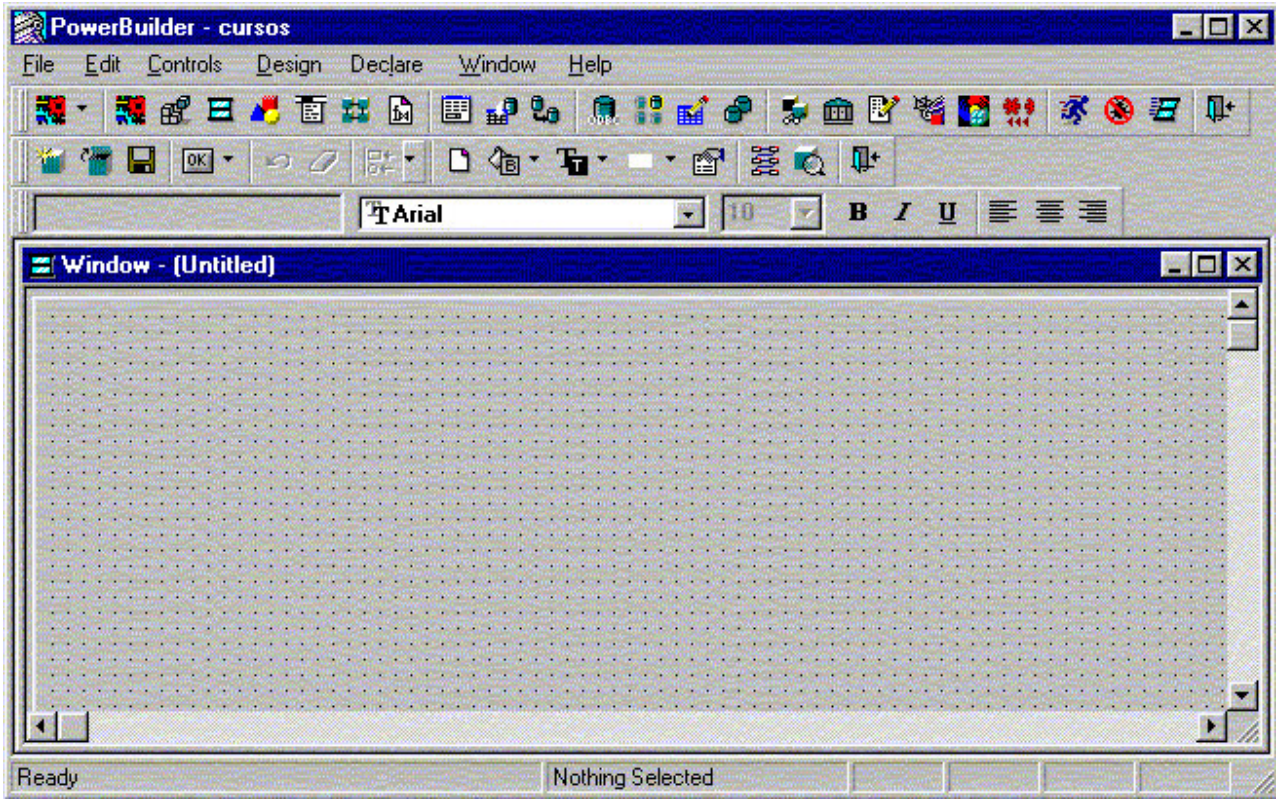
Esta sección describe cómo construir Windows del rasguño.Usted usará esta técnica para crear ventanas que no están basadas sobre ventanas existentes.

Abriendo el Pintor Window(ventana)

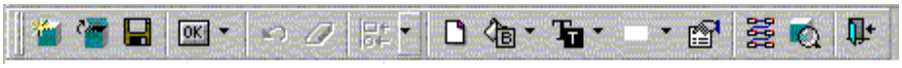
1. Haga click en el botón del pintor Window de la barra de herramientas PowerBar y aparece una ventana en donde puede cargar una ventana ya existente o crear una nueva.



2. Haga Click en el botón *New*(nuevo) para crear una ventana nueva y aparece un espacio de trabajo para crear nuestra nueva ventana.
   Ademas, el botón *OK* , sirve para cuando ya existe una ventana y queremos trabajar sobre ella o modificarla, entonces elegimos una ventana y presionamos *OK*.
   Existe también el botón *Inherit* , esta opción es cuando queremos crear una ventana nueva , pero heredamos todas las características de una ventana que ya existe.



3. Aparecen dos barra de herramientas la barra del pintor (PainterBar) que trabaja de la misma manera como en otros pintores.



4. El pintor Window tiene una Barra de estilos(StyleBar) que se usa para asignar propiedades al texto.

Definiendo las propiedades de la ventana

Cada ventana y control tiene un estilo que determina como apareceran al usuario.
El estilo de una venta comprende lo siguiente:

- Tipo
- Apariencia Básica
- Posicipon inicial sobre la pantalla
- Botones
- Puntero a la pantalla.

Para definir el estilo de la ventana debe elejir de diferentes hojas de la página de propiedades de la ventana, como se muestra en la figura siguiente, en donde cada tiempo va cambiando y muestra todas las hojas de propiedades:

Propiedad General

- Primero debe elegir el tipo de ventana de una lista.(Main Popup,MDI Frame, Child,Response)
- El título de la ventana .
- Una ventana Main o MDI Frame deben tener asociado un menú mientras que las ventanas Hijo o Response no pueden tener nunca un menú.
  Para asociar una ventana con un menú, debe hacer click en *Browser* y elegir un menú que previamente se debe haber creado.
- Elegir color de la ventana *Window Color*, una etiqueta en la opción *Tag*.
- *La Opción Control Menú* ,esta opción para la mayoría de tipos de ventanas, despliega el casillero de cerrar la ventana en la parte superior derecha de la ventana. En la ventana Response no tiene efecto esta opción.

Propiedad Position (posición)

- Elija la hoja *Position*, de la ventana propiedades.
- A la ventana se puede ubicarle en la posición que se quiera en la pantalla con solo haciendo click en la ventana y arrastrar con el mouse a la posición deseada , o también poniendo en los ejes X y Y la posición deseada, además se puede poner el ancho(width) y altura(height) de la ventana.
- Además existe las opciones Normal, Maximizada o Minimizada.

Propiedad Pointer (puntero)

- Elija la hoja *Pointer*, de la ventana propiedades.
- Aqui se puede elegir diferentes tipos de punteros del mouse,Ej: Flecha(arrow) etc, o se puede elegir otro archivo (.CUR) haciendo click en Browser.

Propiedad Icon (icono)

- Elija la hoja *Icon*, de la ventana propiedades.
- En esta opción puede elegir el tipo de ventana que desee, si es la ventana principal de la aplicación entonces el tipo debe ser application!,o si una ventana es tipo response entonces puede elegir el tipo información(Information!) , etc.
- Aqui se puede elegir diferentes tipos de iconos de ventanas ,Ej: application! etc, o se puede elegir otro archivo (.ICO) haciendo click en Browser.

Propiedad Scroll (desplazar)

- Elija la hoja *Scroll*, de la ventana propiedades.
- Aquí puede poner el número en unidades de desplazamiento cuando se hace click en la barra de desplazamiento de la ventana, el default es O.
- Si se quiera tener Barra de desplazamiento horizontal elija en el casillero *HscrollBar* .
  Units Per Column: el número de unidades para desplazarse de Izquierda a derecha o viceversa, cuando el usuario hace click en la flechas izquierda o derecha de la barra horizontal de desplazamiento.
  Column per Page: El número de columnas que se desea desplazar cuando el usuario hace click sobre la barra horizontal.
- Si se quiera tener Barra de desplazamiento vertical elija en el casillero *VscrollBar*
  Units Per Line: el número de unidades para desplazarse de arriba a abajo o viceversa, cuando el usuario hace click en la flechas de arriba o abajo de la barra vertical de desplazamiento.
  Lines per Page: El número de líneas que se desea desplazar cuando el usuario hace click sobre la barra vertical.

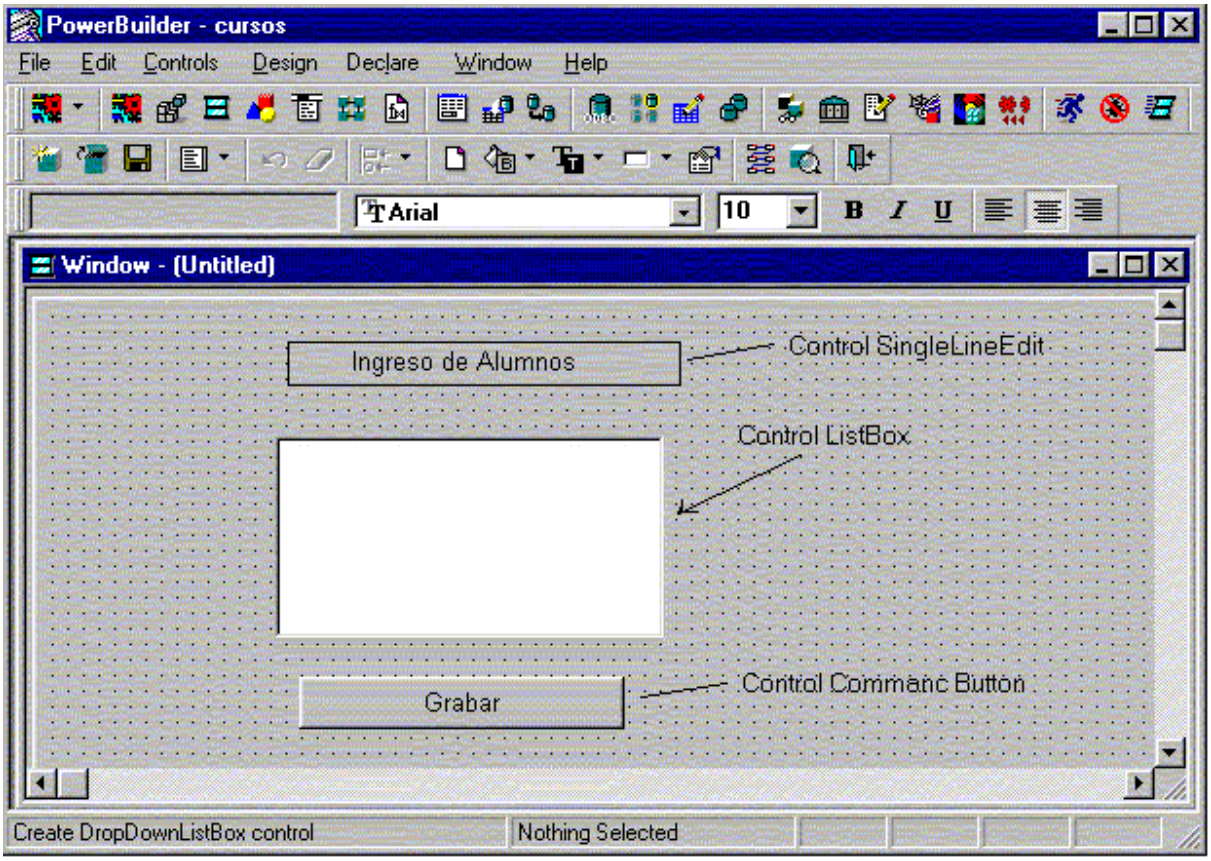
Propiedad ToolBar (barra de herramientas)

- Elija la hoja *ToolBar*, de la ventana propiedades.
- En esta opción Ud. puede elegir en que lugar quiere ubicar la barra de herramientas PowerBuilder, ya sea en la parte superios, a la derecha, ect, o tambien puede dar una localización dando valores a las coordenadas X y Y.

Luego de poner las propiedades a la ventana haga click en ***OK***.

Agregando Controles

Cuando se construye una ventana, se puede colocar controles(tales como: CheckBos, CommadButton, etc.) dentro de la ventana para pedir y recibir información del usuario y presentar información para el usuario.
Despues de colocar un control en la ventana, se puede definir el estilo, moverlo, escribir codigo(script) para que el control responda de acuerdo a un evento.

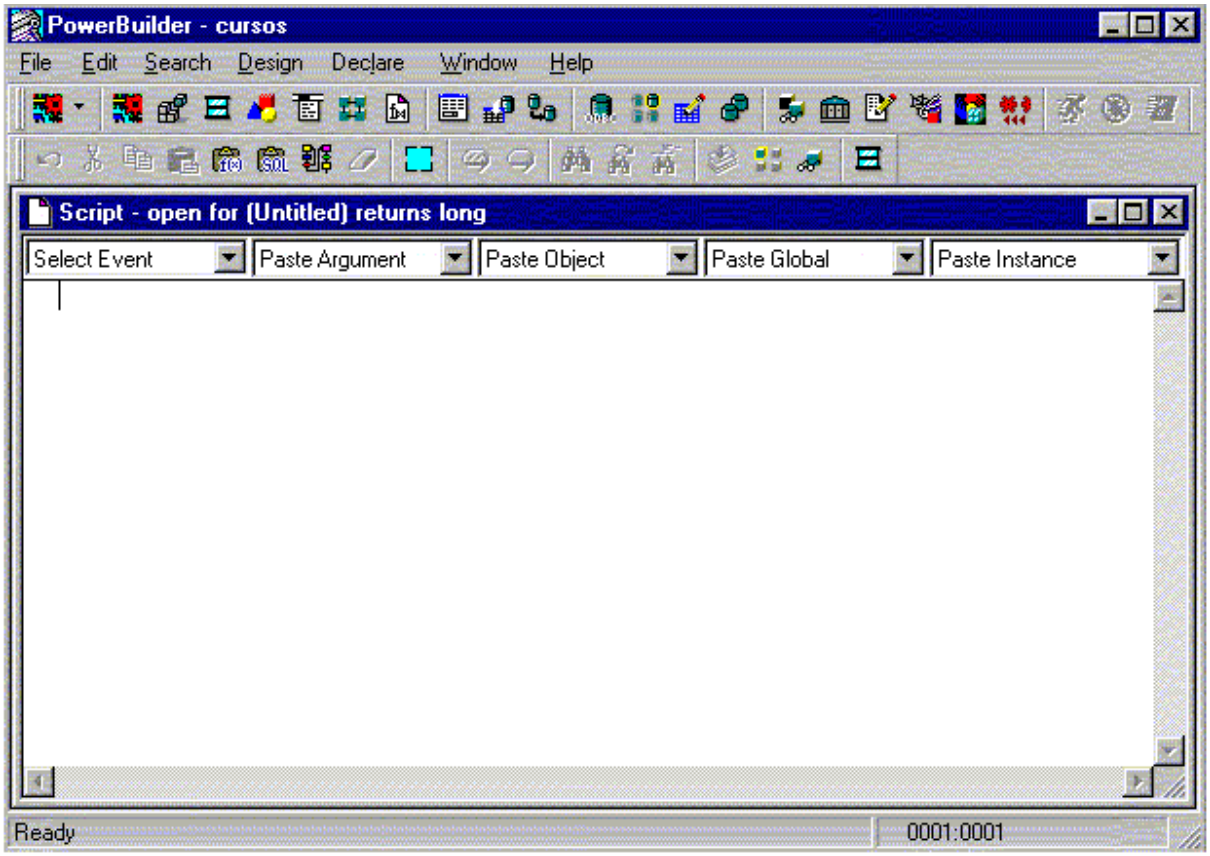


Escribiendo Código(Script) en una ventana

Se puede escribir código(script) en los eventos de una ventana y además en los controles que estan colocados dentro de la ventana.

Para escribir código(script) para un ventana o un control, coloque el mouse sobre la ventana (o el control) y haga click derecho con el mouse y elija la opción ***script*** o por el menú elija la opción ***Edit*** y luego seleccione ***script*** o haga click sobre el icono ***script*** en la barra de herramientas, entonces puede elejir el evento y escribir el código respectivo.

Por ejemplo: puede elegir el evento ***OPEN*** , este evento se dispara cuando se abre la ventana y ecribir el script correspondiente.



Además de elegir el evento en el cual vamos a poner codigo(script), podemos pegar argumentos, objetos ya existenes, variables globales y variables de instancia.

Acerca de los eventos para Ventanas
Las ventanas tiene varios eventos los más importantes son:

Evento	Se dispara cuando
Activate	cuando se abre la ventana, antes del evento open
Close	cuando se cierra la ventana
Open	cuando se abre la ventana

Definiendo sus propios eventos
Ud. también puede definir sus propios eventos,llamados Eventos de Usuario(User Events) para una ventana o control, estos eventos usan la función **TriggerEvent** para disparar un evento creado por un usuario.

Acerca de funciones para Ventanas y controles
Además puede también definir funciones para la ventana el cual puede llamar desde el codigo(script) de cualquier evento de la ventana y de los controles.En las funciones creadas se puede pasar parámetros.

Ud. puede definir su propio nivel de funciones para sus ventanas para facilitar la manipulación de sus ventanas.

Usando Herencia para construir una ventana
Cuando se construye una ventana que hereda todas las caracteristicas (estilo, evento, funciones, estructuras, variables, controles y scripts)de una ventana ya existente. Todo le que se tiene que hacer es modificar las caracteristicas heredadas de acuerdo a las situación actual.

Cómo heredar de una ventana ya existente
Ejemplo: Asumimos que la aplicación tiene una ventana w_padre_ingreso que tiene:

- Un título (Ventana padre para ingreso de datos)
- Un texto de dice: Ingreso de datos
- Un DataWindow en donde se ingresa los datos
- Y dos botones de comando para Grabar los datos o para cancelar.

A continuación mostramos la figura de la ventana descrita anteriormente:

Ahora asumimos que necesitamos construir otra ventana para llevar a cabo un proceso similar. Necesitamos heredar de la ventana padre ingreso , pero ahora para ingresar datos de un alumno especificamente.
Para construir esta ventana tenemos tres opciones:

1. Construir una nueva ventana con esas características de la forma como se explicó anteriormente.
2. Modificando la ventana existente(w_padre_ingreso), y luego grabándola con un nuevo nombre.
3. Usar herencia para construir la ventana que hereda todas las características de una ventana que ya existe(w_padre_ingreso), en otras palabras , construir una ventana decendiente de otra.

Usando las ventajas de la herencia (paso 3)
Usar herencia tiene algunas ventajas:

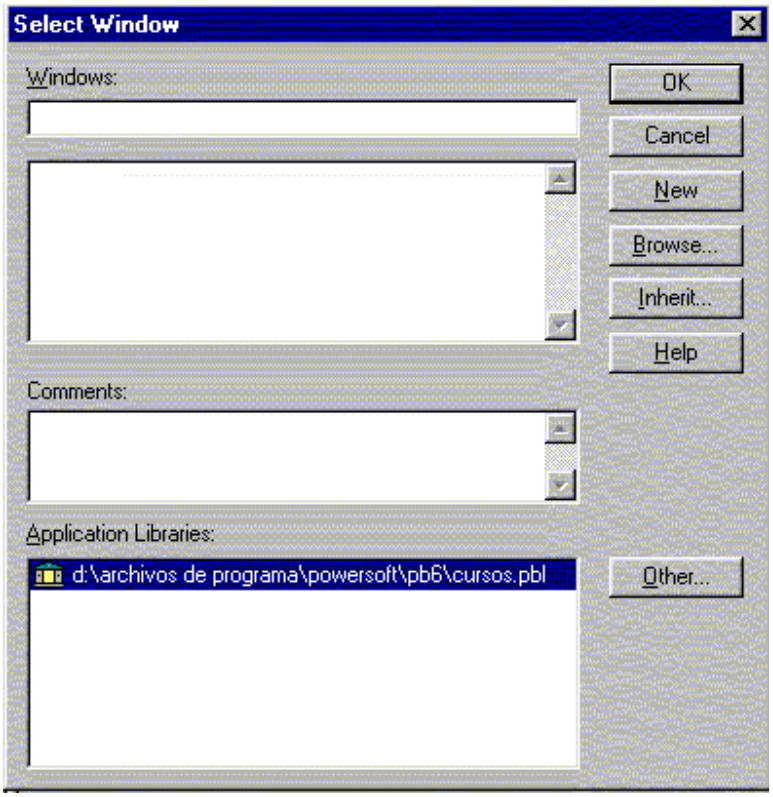
- Cuando Ud. realiza cambios en la ventana *padre*, los cambios son reflejados en todas las ventanas descendientes(hijas). Ud. no tiene que hacer manualmente cambios en los descendientes como lo deberia hacer en una copia(paso 2).
- Los descendientes heredan código script del padre, por tanto Ud. no tiene que reinmgresar el código y agregarlo a la ventana.
- Se obtiene consistencia en el código y en las ventanas de la aplicación.

Como usar herencia para construir una ventana

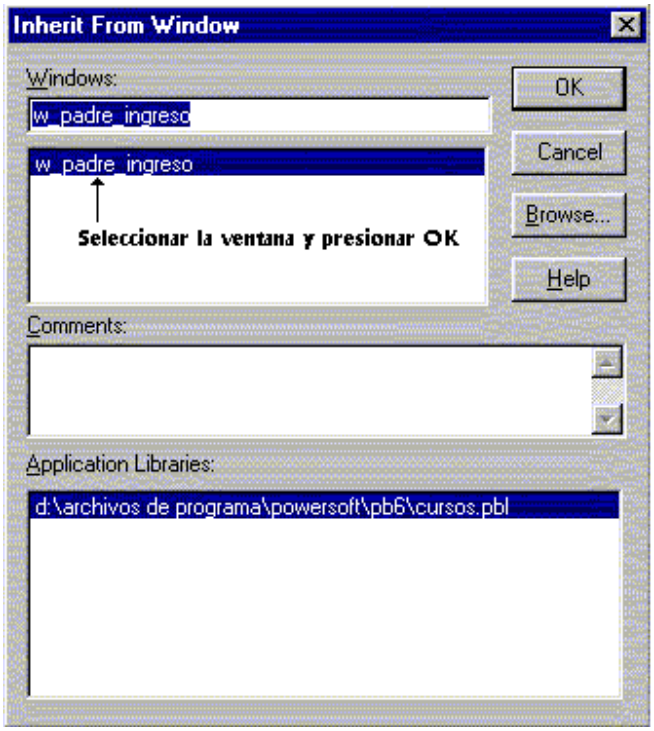
Esta sección describe cómo construir Windows del rasguño.Usted usará esta técnica para crear ventanas que están basadas sobre ventanas existentes.

Abriendo el Pintor Window(ventana)

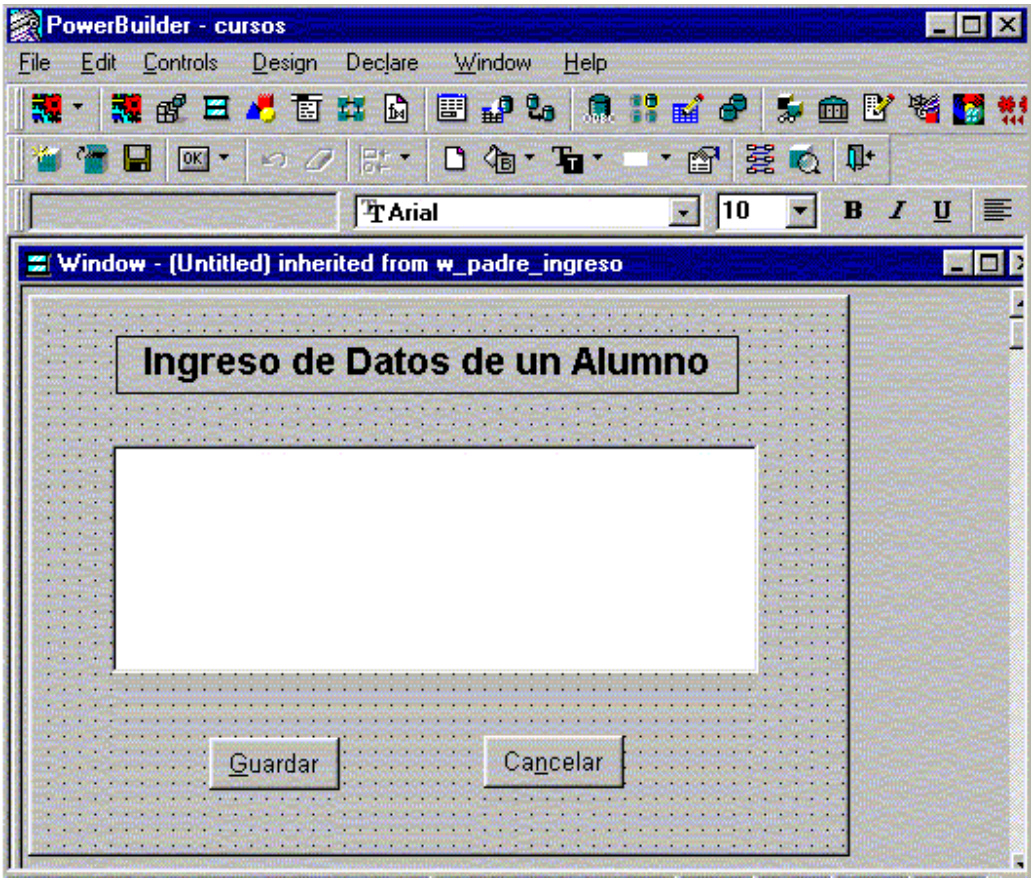
1. Haga click en el botón del pintor Window de la barra de herramientas PowerBar y aparece una ventana de donde podemos heredar una ventana



2. Haga Click en el botón *Inherit*(heredar) para heredar de una ventana que ya existe.
3. Luego aparece otra ventan, en donde debemos seleccionar la ventana de la cual vamos heredar, en este caso, w_padre_ingreso.



4. Luego que hemos seleccionado la ventana , nos aparece la nueva ventana heredada, en la cual podemos ponerle un nombre(w_ingreso_alumnos), y agregarle las características propias para esta nueva ventana. Podemos cambiar las características de la ventana. agregar controloles, contruir nuevos scripts pasra otros eventos, agregar codig´po a los scriptis existentes, referenciar a las funciones y eventos de la ventana padre, declarar nuevas variables, etc.



5. Si Ud. no necesita de algún control heredado, ud. puede harcerlo invisible a ese control en la ventana descendiente.

Pintor Objetos de Usuario (User Objets Painter)

Vista de un Objeto de Usuario Las aplicaciones a menudo tienen características comunes. Por ejemplo, Ud. podría muchas veces reutilizar esas características como las siguientes:

- Un Botón Cerrar, que realiza un cierto número de operaciones y luego cierra una ventana.
- Un ListBox que lista todos los departamentos.
- Un control DataWindow que lleva a cabo un chequeo del mismo tipo de error.
- Procesos que se realizan en varias partes del sistema.

Si Ud. está usando en la misma aplicación características repetidas, Ud. podría definir un Obejto de usuraio(User Objets): Ud. define el objeto de usuario una vez en el pintor Objeto de usuario(User Objets) y puede usar tantas veces como lo necesite.

Hay dos tipos de Objetos de usuario:

- Visuales
- Clase(Class)

Objetos de Usuarios Visuales
Existen tres Tipos de objetos de usuarios visuales:

- ***Standard:*** Un objeto de usuario visual standard hereda las características de un control standard de Power Builder. Usted Modifica las características y hace un control específico para su aplicación.
  Por ejemplo: Ud. asume frecuentemente usar un CommandButton llamado Cerrar, que cierra una ventana padre , el script sería : close(parent).

- ***Custom(creado por el usuario):*** Un objeto de usuario visual custom son objetos que tienen varios controles que funcionan como una unidad. Ud. puede pensar de un objeto de usuario visual custom como una ventana que es una unidad simple y es usada como un control.
  Digamos que Ud. usa un grupo de botones, cada uno de ellos realiza un proceso standard , si ud. construye un objeto de usuario Custom que contiene todos los botones, Ud. puede colocar todos estos botones en una ventana como una unidad , osea coloca el objeto usuario.

- ***External:*** Un objeto de usuario visual external contiene contoles de objetos en el sistema subyacente de las ventanas(Underlying windowing system) que fueron creadas fuera de PowerBuilder. Se puede usar una DLL(Libreria dinámica) hecha im Power Builder para crear un objeto de usuario externo.

Objetos de Usuario tipo Clase (Class)
Un Objeto de Usuario tipo Clase le permite que reutilice un conjunto de reglas de negocios u otros procesos que actúan como una unidad pero que *no son componentes visuales*. Por ejemplo, Ud. podría definir una clase que calcule comisiones de ventas o estadísticas sobre análisis de desempeño.
Hay dos tipos de Objetos de usuario tipo Clase:

- ***Standard:*** Este Objeto hereda las características de un objeto propio del PowerBuilder y que no es visual, tales como, un objeto Transaccional(transaction Object) o un objeto de error (Error Object).
  Un uso importante de este objeto, es cuando se usa herencia de un objeto Transaccional que hace llamadas a procedimientos en una base de datos remota desde la aplicación mismo.

- ***Custom:*** Estos objetos son diseñadas por Ud. mismo, propios, que encapsulan propiedades y funciones no visibles al usuario. No se derivan de Objetos de PowerBuilder. Ud. los define y los crea como unidades de procesamiento que tienen componentes no visuales.
  Por ejemplo: para calcular comisiones en una aplicación, Ud. puede definir un objetode usuario tipo Clase Custom que contiene propiedades y funciones creadas por el usuario que hace el proceso de cálculo de comisiones. Siempre que Ud.

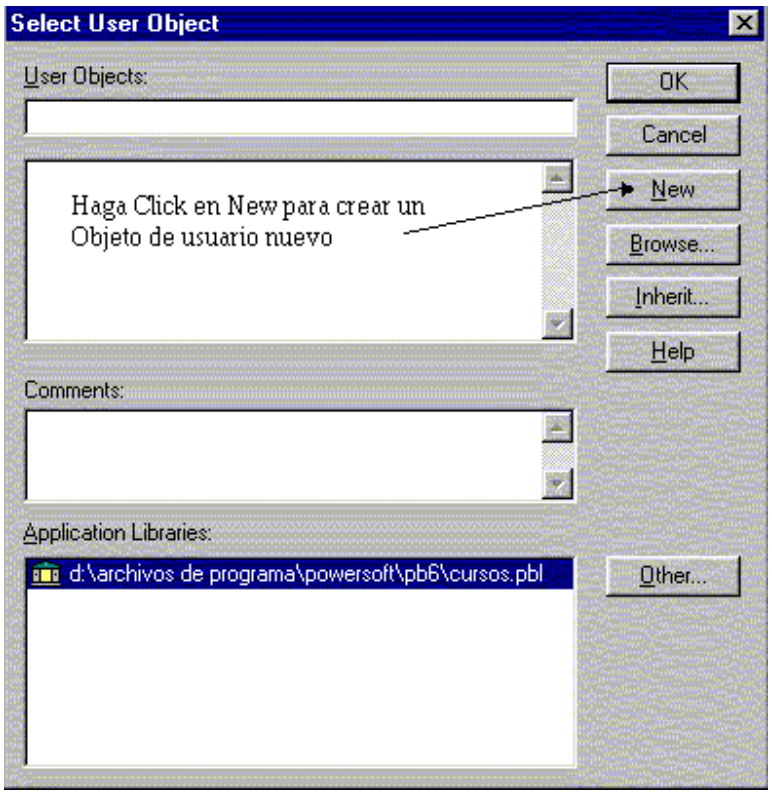
necesite usar este procesamiento, Ud. crea una instancia del objeto de usuario en un script , desde el cual puede accerder a la lógica del objeto de usuario.

Contruyendo un Objeto de Usuario (User Object)

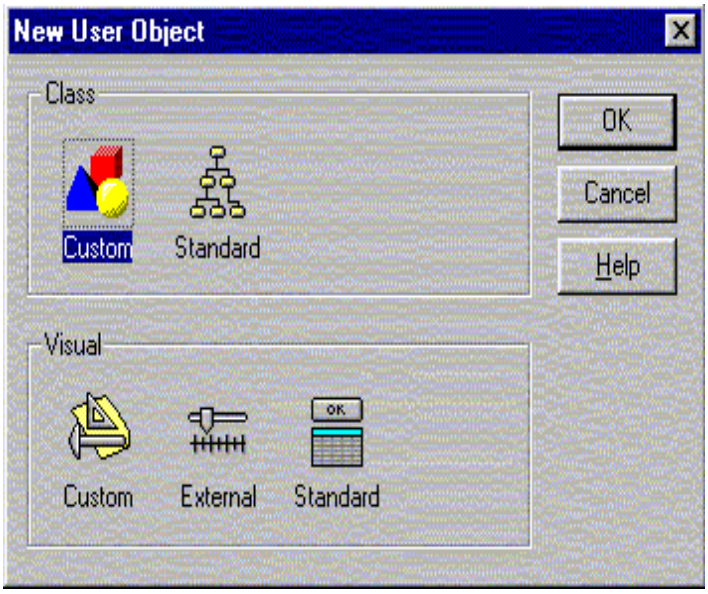
Ud. puede crear un objeto de usuario empezando desde cero, o puede crear objetos de usuario que hereden un estilo determinado , eventos, funciones, estructuras, variables y código(script) de un objeto de usuario que ya existe.

Como contruir un Objeto de Usuario Nuevo

1. Haga Click en Pintor de Objetos de Usuario(User Object) en la barra de herramientas PowerBar.
2. Luego debemos hacer click en el botón ***New*** para cear un nuevo Objeto de usuario.



3. Por último, en la pantalla de Objetos de Usuario, debemos elegir el tipo de Objetode Usuario que se desee crear, de tipo Clase o Visual, luego que elige el tipo haga click en ***OK***.



Pintor Menú (Menu Painter)

Acerca de los Menúes y Objetos Menú(Menu Objects)
Todas las ventanas en una aplicación , excepto en las ventanas de respuesta(Response) y las ventanas hijo(Child), deberían tener menúes. Los **Menúes** son listas de comandos relacionados u opciones(elemento del menú) que un usuario puede seleccionar en la ventana que esta actualmente activa. Cada opción en un menú es definida como un *Objeto Menú(Menu Objetc)* en PowerBuilder. Los **Objetos Menú** se pueden desplegar en una barra de menú o en menúes DropDown o Cascada.

Acerca del uso de menúes
Para usar menúes en PowerBuilder se puede construir de dos formas:

- ***En la barra de menú de una Ventana:*** Los menúes para ventanas(Windows) son asociados con una ventana en el Pintor ventana(Window Painter) y se despliega siempre que que la ventana es abierta.
- ***Como Menúes Popup:*** estos menúes se despliegan solamente cuando en un script se ejecuta la función PopMenu.

Acerca de como diseñar un menú
PowerBuilder da una completa libertad cuando se va a diseñar un menú. Pero se debería seguir ciertos convenios para que el ambiente operativo este en orden para hacer fácil el uso de la aplicación. Por ejemplo: Ud. debería mantener menúes simples y consistentes . Se debería agrupar en opciones relacionadas en un menú dropdown. Debería usar menúes en cascada para economizar y restrigir opciones en un solo nivel.

Acerca de cómo construir Menúes
Cuando se construye un menú , Ud. debe:

- Especificar la apariencia y comportamiento de los objetos menú colocando sus propiedades.
- Construir scripts(código PowerScript) que determinen como responde a un evento en los objetos menú. Para que soporte estos scripts, Ud. puede declarar funciones, estructuras y variables para el menú.

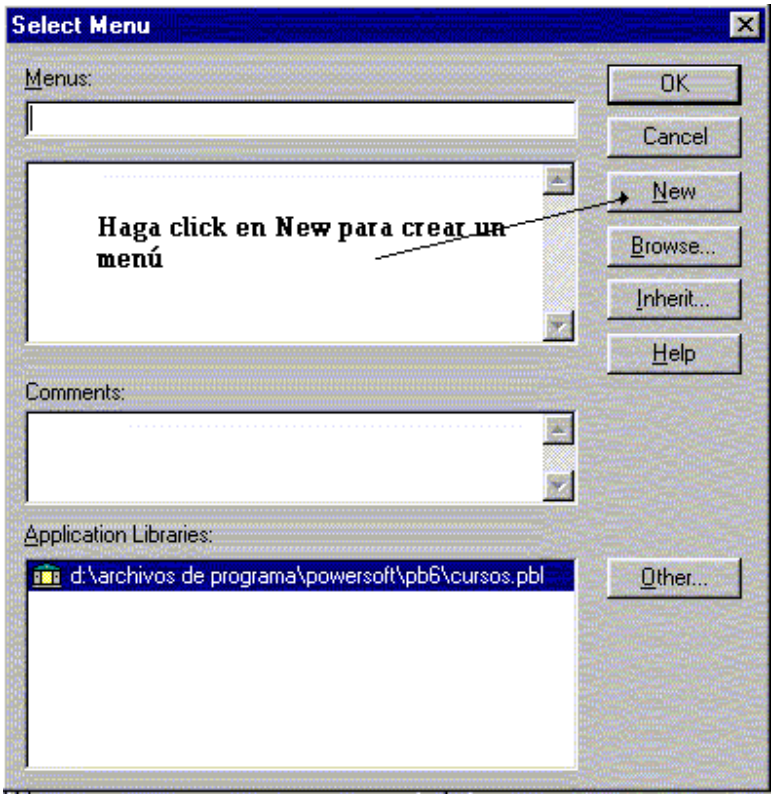
Hay dos maneras para construir un menú:

1. Construir un menu desde cero.
2. Construir un menú que herede un estilo, estructuras, variables y scripts(código) de un menú ya existente.

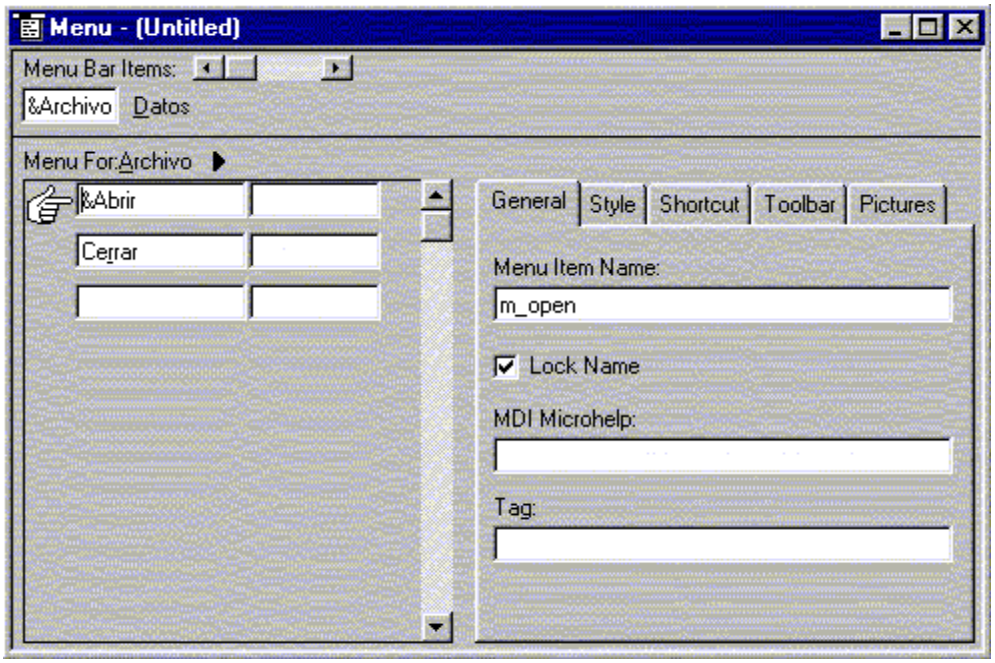
Construyendo un Menú nuevo

Abriendo el Pintor Menú(Painter Menu)

1. Haga click en el pintor Menú de la barra de herramientas PowerBar para que le aparezca una ventana que me permite hacer un menú nuevo o heredar de uno ya existente.



2. Luego haga Click en el botón ***New*** para contruir un menú nuevo. El pintor Menú despliega un espacio de trabajo pra crear el nuevo menú.



El pintor Menú tiene varios areas de trabajo en las cuales se especifica las diferentes partes de un menú:

Menú Bar: Seleccione un Objeto de menu ya existente de la barra de menú(menu bar) o uno nuevo al inicio de la ventana para crear un menú. El menú en la imagen de la ventana anterior tiene dos objetos en la barra de menú(menu bar): Archivos y Datos.

Menúes DropDown y en Cascada: Seleccione un objeto Menú existente de un menú Dropdown o en cascada o especifique uno nuevo en la hoja de menu izquierda. La imagen anterior muestra 2 opciones en el menú dropdown Archivo: Abrir y Cerrar.

Propiedades de los Objetos Menú: Especificando la apariencia y comportamiento del objeto menu selecionado en la hoja derecha de la ventana de menú. Ud. puede especificar propiedades para las opciones de la barra de menú y opciones en el menú. En la imagen anterior se muestra en la página de propiedades ***General*** para la opcion del menu dropdown Abrir,por ejemplo su nombre es *m_open*.

Pintor Estructura (Structure Painter)

Qué son las Estructuras(Structures)?
Una ***estructura*** es una colección de una o más variables relacionadas de un mismo o diferente tipo de datos agrupados bajo un mismo nombre. En Algunos lenguajes como COBOL o Pascal las estructuras son llamadas *registros*.
Las estructuras le permiten referirse como entidades relacionadas como una unidad mas que individualmente.
Hay dos tipos de estructuras:

1. ***Estructuras Globales:*** Estas estruturas no están asociadas a ningún objeto en su aplicación. Ud. puede declarar una instancia de la estructura y referirse ala instancia en algún script de la aplicación.

2. ***Estructura a nivel-de un objeto:*** Estas estructuras están asociadas a un tipo particular de, ventana(window), menú, u objeto de usuario, o con el objeto Aplicación. Estas estructuras pueden siempre ser usadas en un script pero solo de objeto dado. Se puede además se puede escoger como hacer estructuras accesibles desde otros scripts.

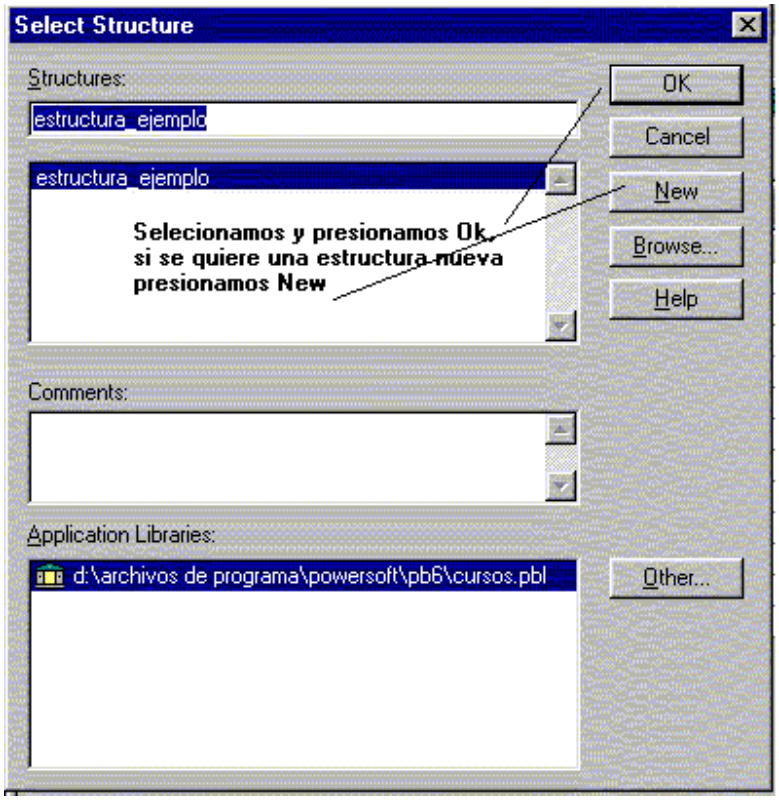
Definiendo estructuras:

1. **Abrir el pintor de Estructura(Structure Painter)**

    - ***Abrir el pintor Estructura para crear una estrcutura Global***

        1. Haga Click en el botón del pintor estructura en la barra de herramientas PowerBar y aparece una ventana de diálogo para selecionar una estructura

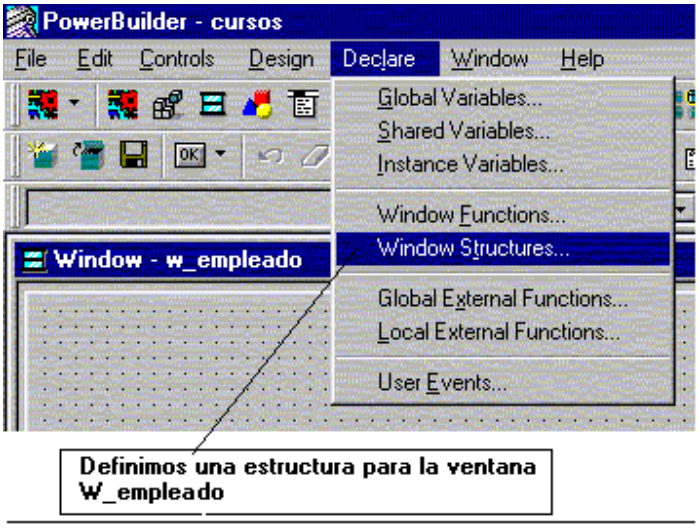
2. Si existe ya una estructura , seleccione la estructura y haga click en ***OK*** o haga click en el botón de New para declarar una estructura nueva.



- ***Abrir el pintor Estructura para crear una estrcutura a nivel de un objeto***

1. Abra el Pintor para el objeto que Ud. quiere declarar una estructura Ud. puede declarar estructuras para ventanas(windows), menus, objetos de usuario, o aplicaciones. Por ejemplo, si Ud. quiere declarar una estructura para una ventana w_empleado abra el pintor de ventana(painter window) y seleccione w_empleado.

2. Luego de haber elegido la ventana w_empleado, del menú la opción *Declare* , luego seleccione la opción apropiada, en este caso, *Window Structure*, si son otros objetos, Menu Structure, User object Structure, o Aplicación Structure.



Luego aparece una ventana de diálogo, donde debemos elegir una estrcutura ya existente y presionamos OK, o si desea declarar uno nuevo presione el botón New.

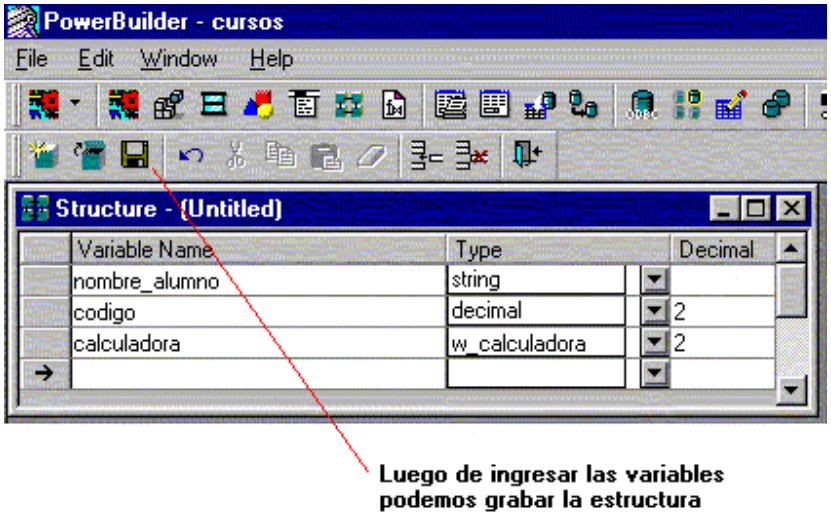
2. **Definir las variables que componen la estructura**

    1. Ingrese el nombre de las variables que usted quiere incluir en la estructura.

    2. Ingrese el tipo de datos de la variable. El default para la primera variable es string(cadena). Se puede especificar cualquier tipo de datos de PowerBuilder, incluyendo tipo de datos standard, tales como enteros(integer) y Cadenas(string), asi como tambipen objetos y controles, tales como ventanas(window), MuiltiLineEdit, ect.
    Además se puede especificar cualquier tipo de datos que Ud. ha definido. Por ejemplo, si Ud. está usando una ventana llamada w_calculadora que Ud. tiene definida y quiere incluir en una estructura la ventana, escriba *w_calculadora* como un tipo de datos.(No se puede seleccionar w_calculadora de la lista , ya que ésta solo muestra tipos propios creados por PowerBiulder).
    Una estrcutura como variable
    Una variable en una estructura puede ser en sí otra estructura. Especifique el nombre de la estructura como un tipo de variable de datos.

    3. Ingrese el número de decimales en el lugar de los DEC si el tipo de datos es decimal. El default es 2.

    4. Repita los pasos hasta que haya terminado de ingresar todas la variables.



3. **Grabar la estructura**
   El nombre de la estructura puede tener un máximo de 40 caracteres.
   Se puede adoptar una convención para poner el nombre a una estructura. Por ejemplo, si está definiendo una estructura global, de puede llamar con el prefijo **s_** (s_nombre), y si una estructura es a nivel de un objeto , su nombre con el prefijo **os_** (os_datos_alumno).
   Una vez haya definido la estrcutura, se debe grabar y ponerle un nombre, adenás puede ponerse comentarios, pero solo cuando es una estructura global).

Usando Estructuras

Pintor Funciones (Function Painter)

Qué son funciones definidas por el usuario ?

El Lenguaje PowerScript tiene muchas funciones que vienen construidas por PowerBuilder. Pero Ud. podría encontrar que necesita programar el mismo proceso una y otra vez. Por ejemplo, Ud. podría necesitar llevar a cabo cierto cálculo en varias partes de una aplicación o en diferentes aplicaciones. En tal situación ud. querrá crear una ***Función definida por el usuario*** para realizar el proceso.

Una Función definida por el usuario es una colección de sentencias que desempeñan algún proceso. Ud. puede usar el pintor de Función(Painter Function) para definir funciones credas por el usuario. Después de definir la función y grabar dentro de una librería, alguna aplicación accesará a esa librería para usar la función.

Hay dos tipos de definir funciones creadas por el usuario:

1. ***Funciones Globales:*** que no están asociadas a ningún objeto en la aplicación y que están siempre accesibles desde cualquier parte de la aplicación.

2. ***Funciones a nivel-de un objeto:*** que están definidas para un tipo particular de ventana(window), menu, u objeto de usuario, o para un objeto aplicación. Estas

funciones son parte de la definición de un objeto y pueden siempre ser usadas en el script solo de ese objeto. Ud. puede escoger y hacer que estas funciones sean accesibles desde otro script también.

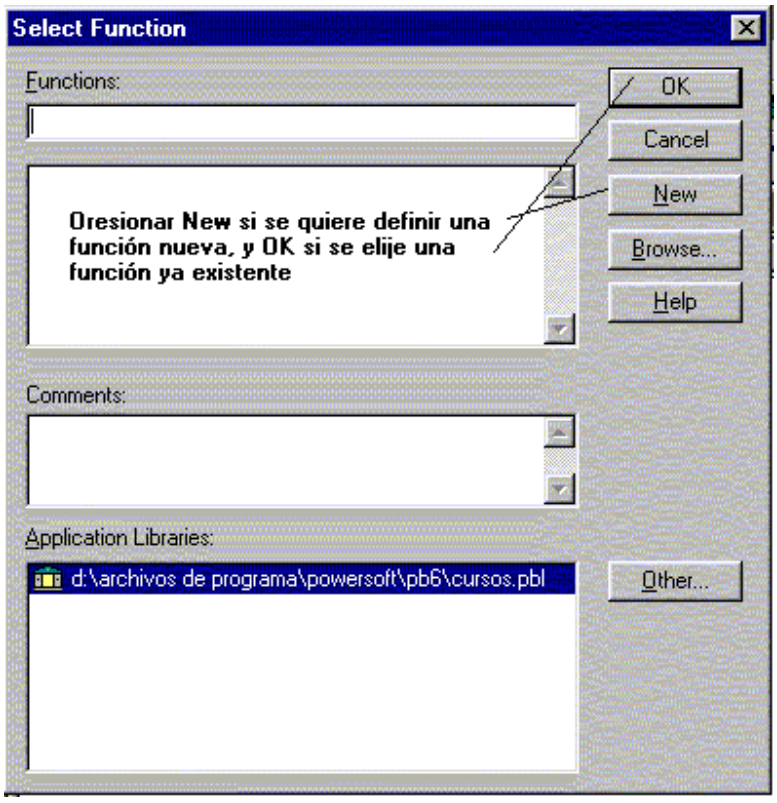
Definiendo Funciones creadas por un usuario

1. ***Abrir el pintor Función(Function painter)***

   - ***Abrir el pintor Función para crear una Función Global***

     1. Haga Click en el botón del pintor función en la barra de herramientas PowerBar y aparece una ventana de diálogo para selecionar una función

     2. Si existe ya una función , seleccione la función y haga click en ***OK*** o haga click en el botón de **New** para declarar una función nueva.



   - ***Abrir el pintor Función para crear una función a nivel de un objeto***

     1. Abra el Pintor para el objeto que Ud. quiere declarar una función Ud. puede declarar funciones para ventanas(windows), menus,

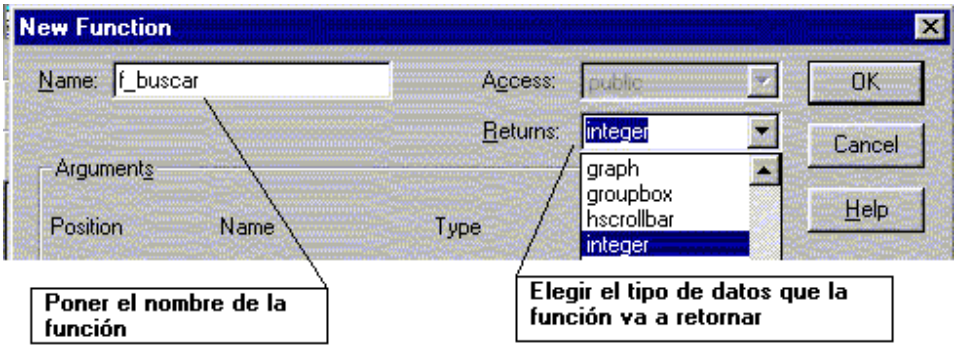
objetos de usuario, o aplicaciones. Por ejemplo, si Ud. quiere declarar una función para una ventana w_empleado abra el pintor de ventana(painter window) y seleccione w_empleado.

2. Luego de haber elegido la ventana w_empleado, del menú la opción *Declare* , luego seleccione la opción apropiada, en este caso, *Window Functions*, si son otros objetos, Menu Functions, User object Functions, o Aplicación Functions.

Luego aparece una ventana de diálogo, donde debemos elegir una función ya existente y presionamos OK, o si desea declarar una nueva presione el botón New.

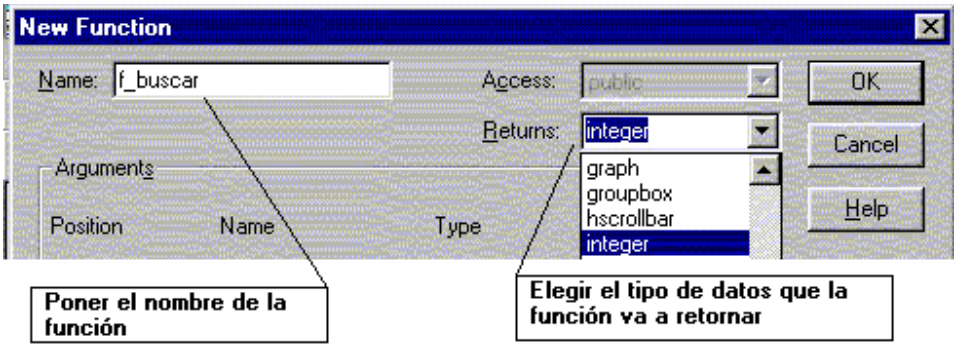
2. ***Poner el nombre de la función***
   El Nombre de las funciones pueden tener hasta 40 caracteres.
   Es buena idea desarrollar una convención de nombres para las funciones creadas por el usuario asi Ud. puede reconocer las funciones fácilmente y distinguirlas de las funciones propias de PowerBuilder.
   Por ejemplo: se podría dar el nombre a las funciones globales con un prefijo **f**_ por ejemplo: (f_encontrar) y para funciones a nivel de objetos con un prefijo **of**_ por ejemplo: of_chequearpadre, of_refreshwindow.

3. ***Definir el tipo que va a retornar la función***
   Muchas funciones desempeñan algunos procesos y entonces deben retornar un valor. Este valor puede ser el resultado de un procesamiento o un valor que indica si la función se ejecutó satisfactoriamente o no. Si tiene su función que retornar un valor, ud. necesita definir el tipo que va ha retornar(*return type*), que especifica el tipo de dato del valor que se va a retornar.
   Cuando de va a definir una función, ya sea global o a nivel de objeto, podemos elegir el tipo que se va a retornar, como se muestra en la figura:



Si se quiere que la función no retorne ningún valor, se debe elegir ***None*** de la lista *return type*.

4. **Para una función a nivel de un objeto, definir un nivel de acceso**
   La ventana de diálogo tiene un lista dropdown que tiene los tipos de acceso a una función.



En esta ventana se especifica el nivel de acceso de la función -el lugar desde el cual Ud. puede llamar a la función en una aplicación.

Para Funciones Globales

Las funciones globales pueden ser llamadas desde cualquier parte de la aplicación. En términos de PowerBuilder, estas funciones son **Públicas(Public)**. Además cuando Ud. está definiendo una función global, no se puede modificar el acceso.

Para Funciones a nivel de un objeto

Se puede restringir el acceso a una función a nivel de objetos por un conjunto de niveles de acceso como se muestra a continuación:

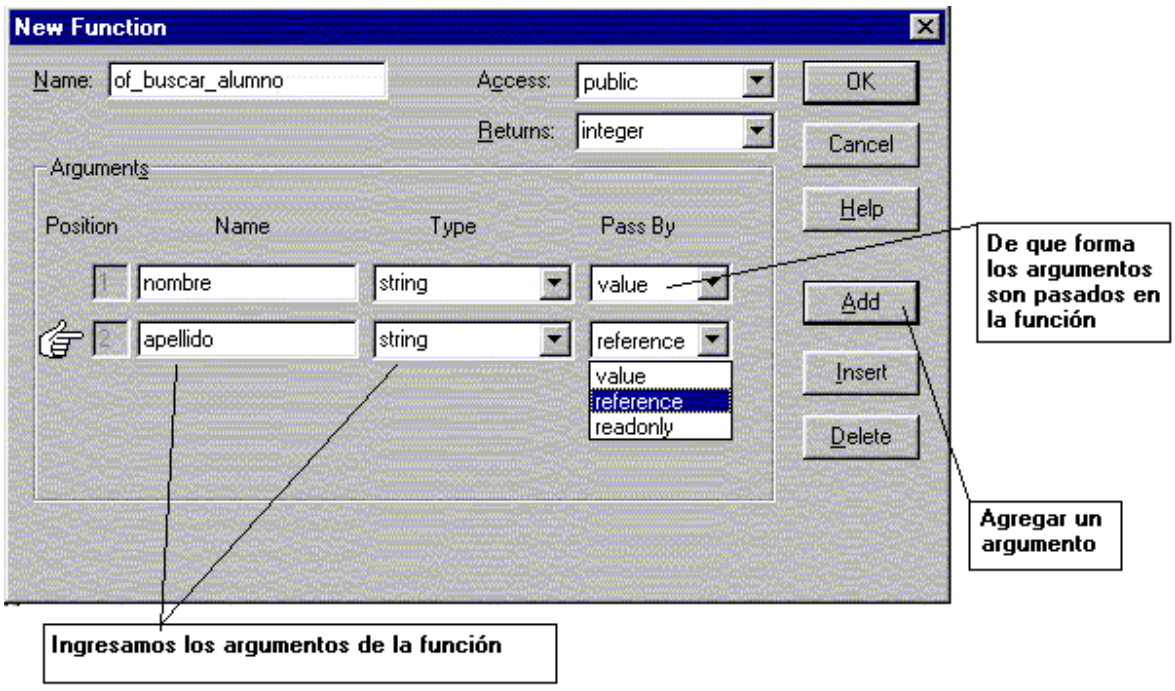
Acceso	Desde donde se puede llamar a la función
Public	Desde cualquier script de la aplicación
Private	Solo en scripts de eventos en el objeto en el cual la función es definida. No se puede llmar a la función desde los objetos decendientes de otro objeto.
Protected	Solo en scripts para objetos en los cuales la función fue definida y para los objetos descendientes del objeto padre.

Si una función está solamente para ser usada dentro de un objeto, debería definir el acceso como private o protected. De esta forma. ud. garantiza que la función nunca sea llamada inapropiadamente desde fuera del objeto.(En términos orientados a objetos, definiendo funciones como private o protected se ***encapsula*** la función dentro del objeto).

- ***Definir los argumentos para la función***

Como en las funciones hechas en PowerBuilder, las funciones definidas por el usuario pueden tener un determinado número de argumentos o ninguno. Ud. declara los argumentos y sus tipos cuando define la función.
Pasos para definir los argumentos:

1. Poner el nombre del argumento(el orden como especifica los argumentos es el orden cuando se llama a la función)
2. Declarar el tipo de los argumentos, existen varios tipos:
    - Tipos de datos propios del PowerBuilder(Integer, real, etc)
    - Tipos de Objetos(tales como window) u objetos específicos(tales como w_empleado)
    - Objetos de usuario
    - Controles (tales como CommmandButtons)
3. Declarar como se quiere que los argumentos sean pasados, existen tres tipos para pasar los argumentos:



- ***Por referencia:*** Cuando se pasa un argumento por referencia, la función tiene acceso a los argumentos originales y se puede cambiar los datos directamente.

- ***Por valor:*** Cuando se pasa por valor, se esta pasando a la función una copia temporal y local del argumento. La función puede cambiar el valor de la copia local del argumento dentro de la función, pero el valor del argumento no es cambiado desde el script que es llamada la función.

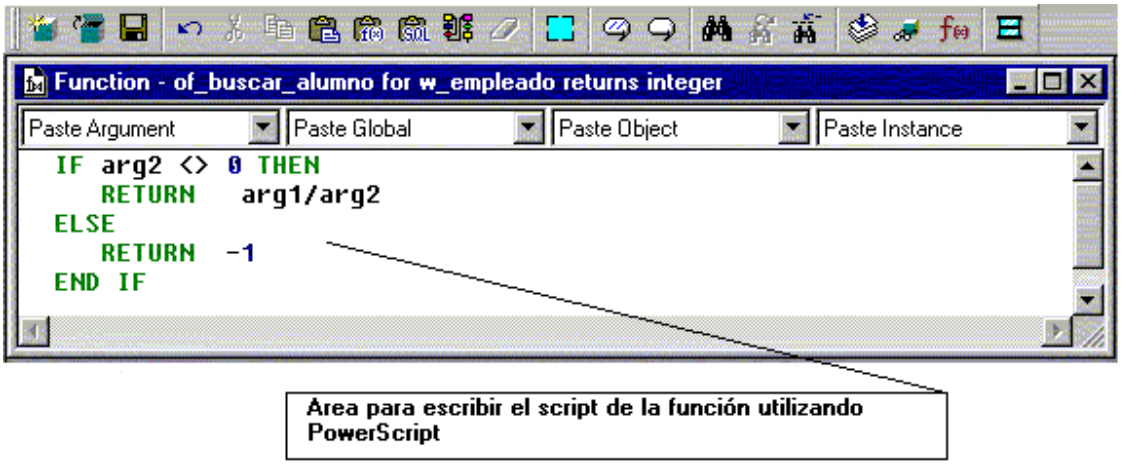
- ***Solo lectura(ReadOnly):*** Cuando pasa un argumento solo lectura, el valor de la variable está disponible en la función, pero es tratada como una constante. Este tipo provee un gran desempeño para valores como cadenas (strings), Bolbs, Date, time, DateTime, por que no crea una copia del dato que es pasado.

4. Si se quiere agregar otro argumento haga click en el boton **Add** y repita los pasos del 1 al 3.

5. Finalmente haga click en **OK**.

- ***Implementar el código para la función***

El pintor Función, una vez que se ha definido la función con sus parámetros, despliega un espacio de trabajo para poder implementar el codigo en la función, esto es, con PowerScript.



Una función definida por el usuario puede contener sentencias PowerScript, sentencias SQL integradas y llamadas a funciones propias del PowerBuilder, y otras funciones.
Para retornar una valor se utiliza la **sentencia RETURN**:
RETURN ***expesion***
Ejemplo:

```
IF arg2 <> 0 THEN
                    RETURN   arg1/arg2
                ELSE
                    RETURN  -1
                END IF
```

Pintor DataWindow (DataWindow Painter)

Introducción a los objetos DataWindow

Un Objeto DataWindow es un objeto que se usa para recuperar, presentar y manipular datos de una base de datos relacional u otra fuente de datos(tales como una tabla de excel o un archivo de dBase).

Los objetos DataWindow tienen el conocimiento acerca de los datos que ellos están recuperando. Ud. puede formatos para desplegar los datos, estilos de presentación.

Cómo usar los objetos dataWindow

Antes que Ud. pueda usar el objeto DataWindow en una aplicación, necesita construir el objeto. Se debe utilizar el pintor Datawindow(Datawindow painter), que le permite crea y editar objetos DataWindow. Adicionalmente, permite hacer archivos PSR(PowerSoft Report) que además le permitirían usar en una aplicación. Un archivo PSR contiene la definición de un reporte (esencialmnete objetos DataWindow sin actualizar tablas) asi como el contenido de datos en este reporte cuando el archivo PSR fue creado.

Ejemplos de objetos DataWindow

Ud. puede desplegar datos de la mejor forma de presentación para el usuario:

Estilos de Edición(Edit styles)

Si una columna puede tomar solamnete u pequeño numeros de valores, ud. puede hacer aparecer los datos como botones de radio(radio buttons) en un objeto datawindow y el usuario sabe que debe elegir uno de ellos.

Formatos de presentación (Display formats)

Si una columna despliega un número telefónico , salarios, o fechas, ud. puede especificar el formato apropiado para el dato.

Reglas de validación

Si una columna puede tomar números solamente en un rango específico, ud. puede especificar una regla simple de validación para la columna, sin escribir ningún código, y así asegurarse que el usuario ingrese datos válidos.

Reforzar los Objetos DataWindow

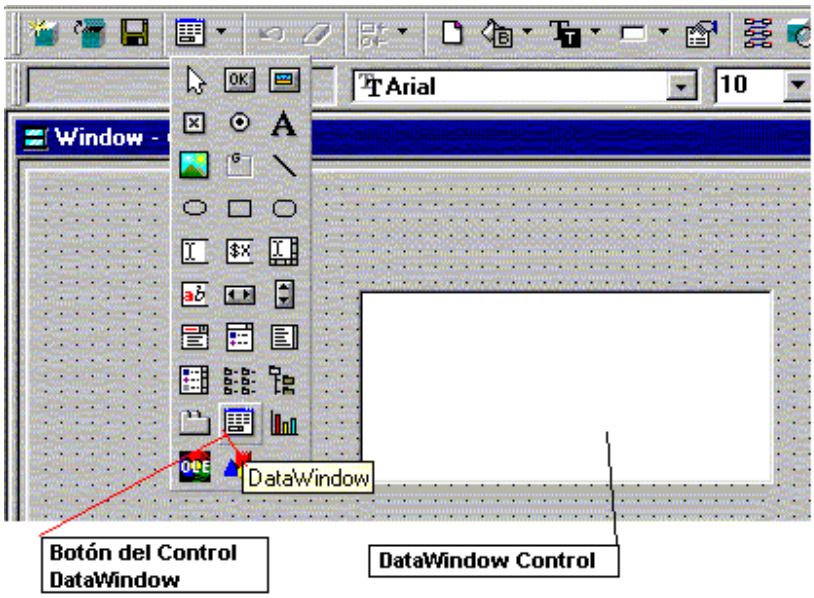
Si ud. quiere mejorar la presentación y manipulación de los datos en un objeto DataWindow, ud. puede incluir campos calculados(computed fields), imágenes(pictures) y gráficos que son ligados directamente a los datos recuperados por el objeto.

Esta sección describe los procesos sobretodo para la creación y uso de los objetos DataWindow.

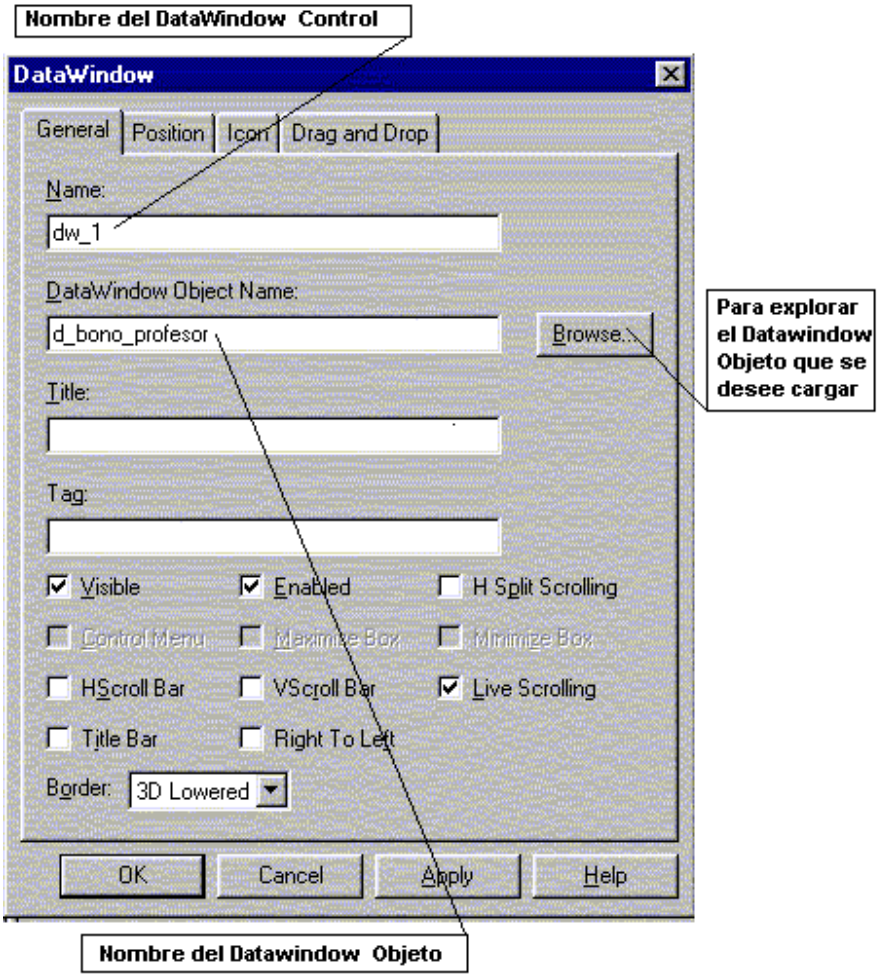
Usando los objetos Datawindow

1. Construir un objeto DataWindow (o archivo PSR) haciendo click en el botón del Pintor DataWindow. En este pintor, Ud. define la fuente de datos, estilo de presentación y todas las otras propiedades del objeto, tales como, reglas de validación, ordenamiento y filtros y gráficos.

2. Colocar un control DataWindow en una ventana(o en objeto de Usuario).
   A través de este control su aplicación se puede comunicar con el objeto datawindow que ha creado con el pintor DataWindow.



3. Asociar el control DataWindow con el objeto DataWindow. Hacemos doble click sobre el control datawindow o hacemos click derecho sobre el control datawindow para ver las propiedades del control y poder ligar con el datawindow objeto.



4. Escribir código script en una ventana para manipular de control DataWindow y su contenido.
   Por ejemplo: Ud. puede usar la función Retrieve de PowerScript para recuperar datos dentro de un control DataWindow.
   Ud. puede escribir scripts para el control DataWindow y tratar de manipular errores, compartiendo datos entre controles DataWindow.

5. Escribir código para controlar un proceso un proceso que es iniciado cuando ocurre un evento en el control DataWindow.
   Ud. puede escribir scripts para el control DataWindow y tratar de manipular errores, compartiendo datos entre controles DataWindow.

Objetos Datawindow versus Reportes

Pintor Reporte (Reporte Painter)

Los Reportes presentan datos. El pintor Reporte en PowerBuilder provee de muchas maneras para presentar los datos. Ud. podría requerir de reportes tabulares con filas y columnas llenas de información. A veces un gráfico o un crosstab es una mejor manera para presentar los datos.

Los reportes en PowerBuilder pueden además estar con etiquetas para envíar por correo o muchos reportes jerarquizados que se encuentran en la misma página. PowerBuilder además tiene reportes de forma libre(FreeForm) que le permite colocar texto, datos, líneas, cajas de texto, y gráficos en cualquier parte que Ud. desee.

Reportes versus Objetos Datawindow

Para crear un reporte nuevo se hace de la misma manera como para crear un nuevo objeto DataWindow.
Construyendo un Reporte

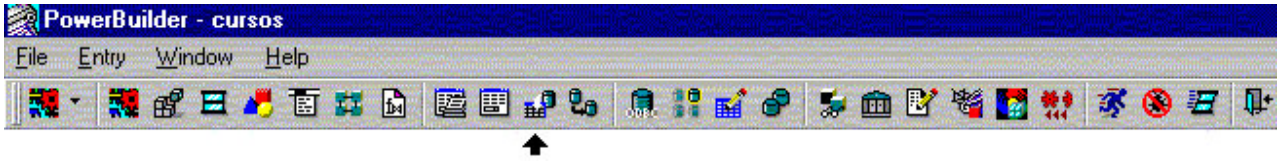
Pintor Consulta (Painter Query)

Una Consulta es una sentencia SQL SELECT creada con el pintor Consulta(Query Painter) y grabada con un nombre y puede ser usada repetidamnete como una fuente de datos para un objerto DataWindow.

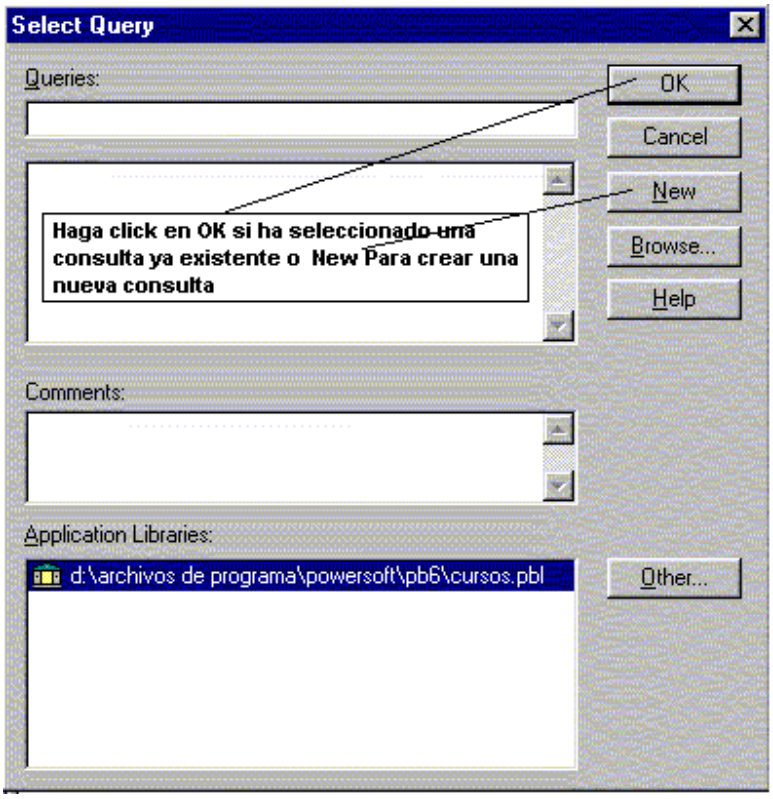
Las consulta ahorran tiempo, porque ud. especifica todos los requerimientos de datos solo una vez. Por ejemplo, se puede especificar las columnas, cuales filas se van a recuperar, y el ordenamiento de una consulta. Las veces que Ud. quiera crea objetos DataWindow usando estos datos, simplemente especificando la Consulta(Query) como la fuente de datos.

Definiendo una Consulta(Query)

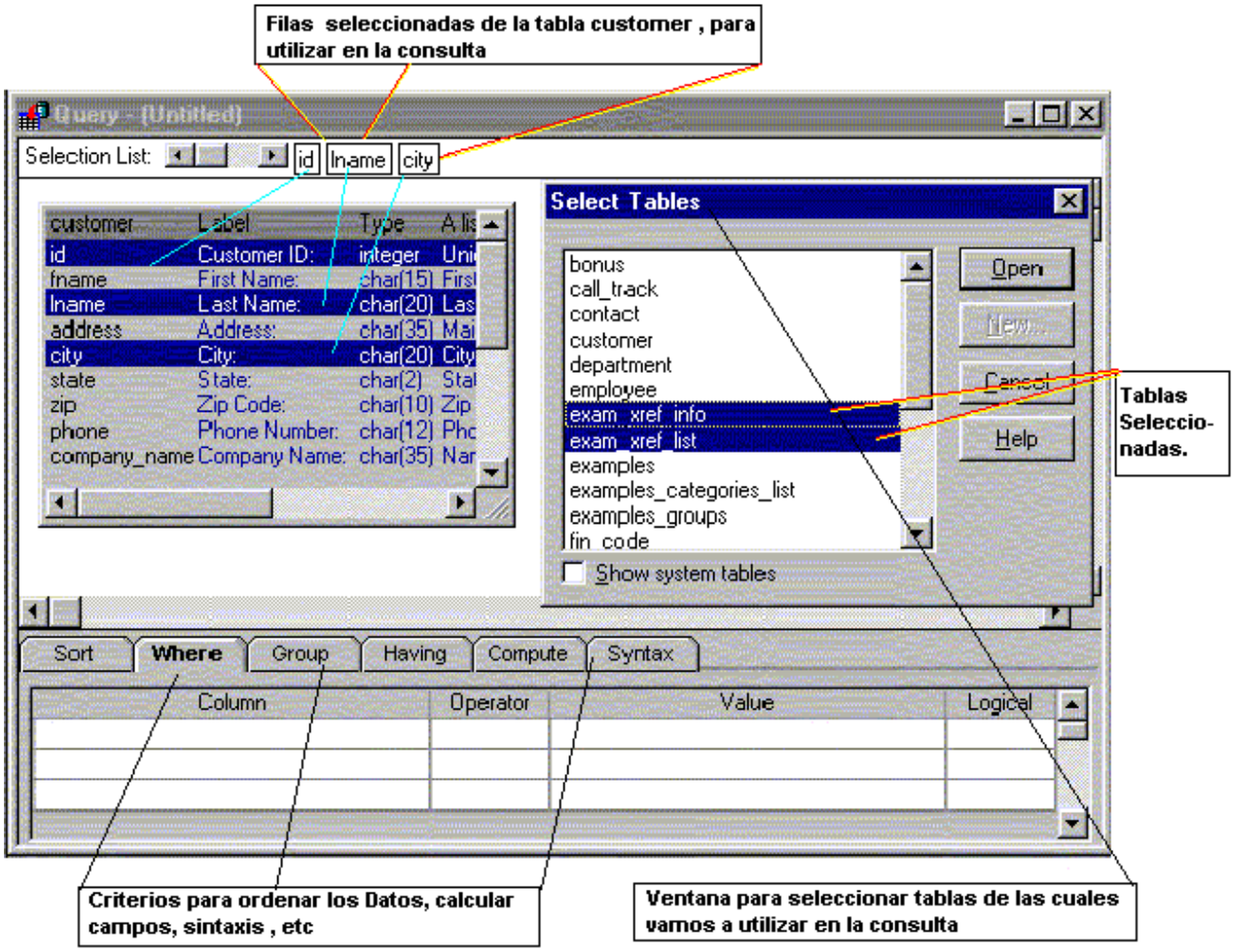
1. Haga click en el botón del Pintor Consulta(Query painter) en la barra de herramientas PowerBar.



2. Haga Click en el Botón **New** para crear una Consulta nueva en la ventana de diálogo o elija una consulta ya existente y haga click en **OK** para modificar la consulta. Entonces se despliega un área de trabajo para crear o modificar una consulta.



3. Seleccione las tablas de la ventana de diálogo que aparece y luego las columnas que desea que se despliegen en la consulta.



Además se puede definir criterios para ordenar datos de acuerdo a una columna o más, criterios para agrupar datos, definir campos calculados(Competed), y así sucesivamente, de manera parecida cuando crea objetos datawindow usando fuentes de datos con SQL select.

Pintor Tubería de Datos (Painter Data Pipeline)

Acerca de los Data PipeLines

El pintor Data Pipeline le da la capacidad para reproducir rápidamente datos dentro de una base de datos, a través de base de datos, o incluso a través de DBMS´s. Para hecer eso, ud. crea una tubería de datos que, cuando es ejecutada, trasmite los datos según lo especificado en la definición de la tubería de los datos.

Qué Puede hacer Ud. ?

Con el pintor Data Pipeline, puede desempeñar algunas tareas que serían de otra manera muy desperdiciadoras de tiempo. Por ejemplo ud. puede:

- Conducir datos(y atributos extendidos) de una o más tablas a una tabla en el mismo DBMS o diferente DBMS.
- Conducir una Base de datos íntegra, una tabla a la vez, a otro DBMS.
- Crear una tabla con el mismo diseño como una tabla ya hecha pero sin datos.
- Conducir datos agrupados de una base de datos servidora a una base de datos SQL Anywhere sobre su computadora y que ud. pueda trabajar sobre los datos y sacar reportes sin necesidad de acceder a la red.
-

Base de Datos Fuente y Destino
Ud. puede usar el pintor Data Pipeline para conducir datos de una o más tablas de una **Base de Datos fuente** a una tabla en una **Base de Datos Destino**.
Se puede conducir los todos los datos o seleccionar datos en una o más tablas. Por ejemplo, ud. puede conducir una pocas columnas de datos de una tabla o datos seleccionados de un join multitabla.
Cuando se conduce datos, los datos en la base de datos fuente se quedan en la base de datos fuente y son reproducidos en una tabla nueva o ya existente en la base de datos destino. Aunque el origen como el destino pueden ser la misma base de datos, son usualmente distintos, y pueden tener aún diferentes DBMS's. Por ejemplo, puede conducir datos de una base de datos SQL Server a una base de datos SQL Anywhere en su computadora.

Definiendo un Data Pipeline
Se puede usar el pintor Data Pipeline para crear un pipeline(tubería de datos), se definir lo siguiente:

- La base de datos fuente(origen).
- La base de datos destino.
- La tablas en el origen y acceder a los datos y recuperarlas de ellas.
- Operaciones para el pipeline (tubería de datos).
- Tablas que van a ser destino.

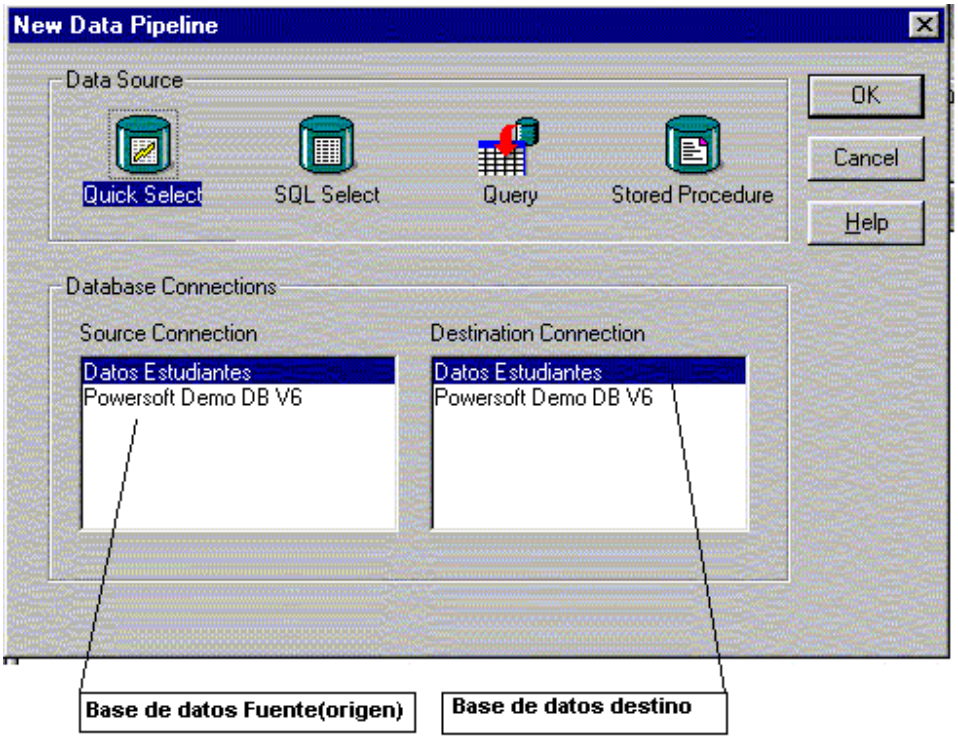
Una vez que se ha definido el pipeline, se puede ejecutarlo inmediatamente. Si ud. quiere, puede además grabarlo como un objeto y ponerle un nombre para usarlo las veces que se quiera.

Creando un Data Pipeline

1. Hacer click en el pintor Data Pipeline de la barra PowerBar.
   Luego en la ventana de diálogo seleccione un data Pipeline existente y haga click en **OK**, pero si desea crear uno nuevo haga click en **NEW**

2. Si hace click en NEW, aparece el cuadro de diálogo para el nuevo Data Pipeline:
   En la opción Databese Connection, La conexión origen(Source connection) y la conexión destino(Destination connection) se despliegan los profiles de las base de datos que han sido definidos.
   Nota: Si se quiere usar una base de datos como fuente o destino se debe crear un profile para que aparesca en la ventana de diálogo para un nuevo data pipeline, pero si su base de datos no aparece es que todavía no se crea un profile, por tanto debe definir un profile para su base de datos.

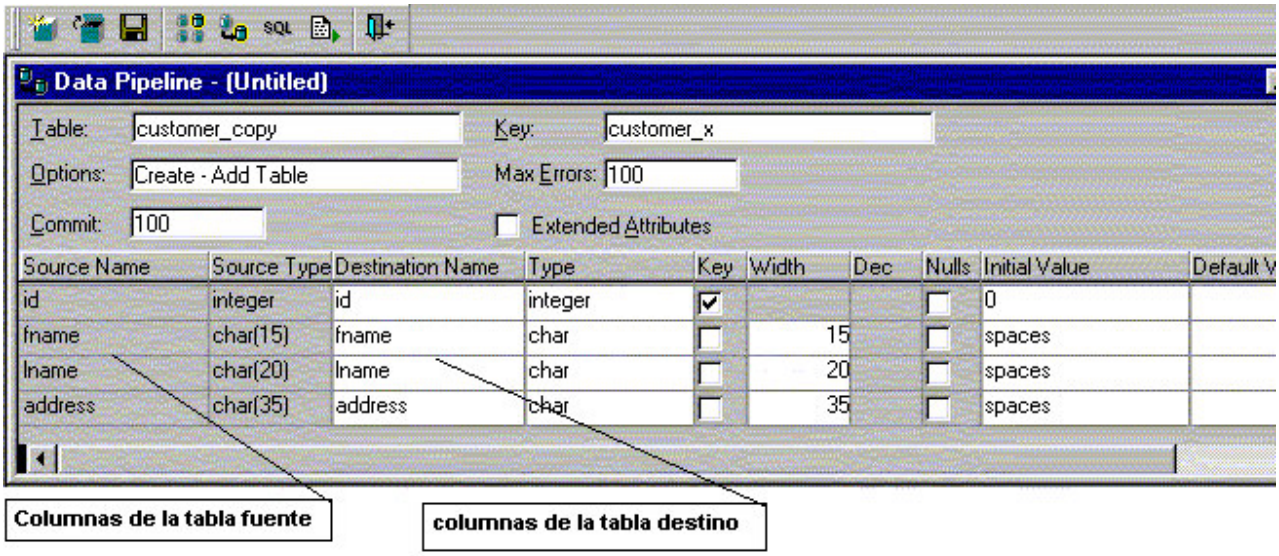


3. Selecionar una fuente de datos.
   La fuente de datos(Data Source) determina cómo el powerBuilder recupera los datos cuando ud. ejecute el pipeline. (Source connection)

Fuente de datos(Data Source)	**Uselo si**
Quick Select	Los datos son de tablas que están conectadas a través de una clave y solamnete se necesita ordenarlos y datos limitados.
SQL Select	Ud. quiere más controles sobre sentencias SQL Select generadas por la fuente de datos o sus datos estan en tablas que no están conectadas a través de una clave.
Query	Los datos han sido definidos como una consulta(query)
Stored Procedure	Los datos están definidos como procedimientos alamacenados.

4. Seleccione las conexiones tanto la fuente como el destino y haga click en OK.

5. Defina los datos a transmitir. Esto depende según la fuente datos que eligió en el paso 3.
   Cuando termina la definición de los datos a conducir, el pintor Data Pipeline despliega un espacio de trabajo para definir el pipeline, que incluye operaciones de pipeline, una casilla de verificación para especificar si se transmite datos con atributos extendidos y opciones para la fuente y el destino.



6. Modifique la definición del pipeline como sea necesario

7. (Opcional) Modifique la fuente de datos según sea necesario.
   Haga click en el botón Edit SQL , o en el menú la opción *Design* y de ahí la opción *Edit data Source*.

8. Si ud. quiere empezar el pipeline, haga click en el botón Execute del menu-barra o en el menú la opción *Design* y de ahí la opción *Execute*.
   Power Builder recupera la fuente de datos y ejecuta el pipeline. Si usted especifica argumentos de recuperación con el pintor Select, PowerBuilder primero le sugiere que los proporcione.

9. Grabar la definición del Pipeline si cree apropiado.

Pintor Base de Datos(Painter DataBase)

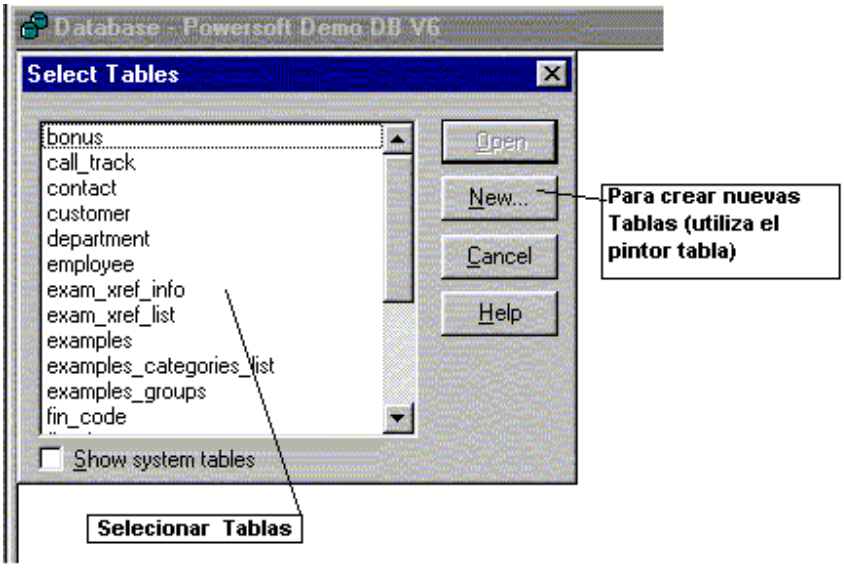
Usando el Pintor de Base de Datos
Abriendo el Pintor de Base de Datos

1. Haga click en el pintor de base de datos en la barra PowerBar

Se despliega una ventana de diálogo Select Tablas(seleccionar tablas) donde aparece una lista de todas las tablas y vistas(views) en la base de datos actual

2. Selecione una o mas tablas y haga click en el botón Open para desplegarlas gráficamente

   o
   Haga click en el botón *New Table* y vamos al pintor de Tabla para crear una nueva Tabla.



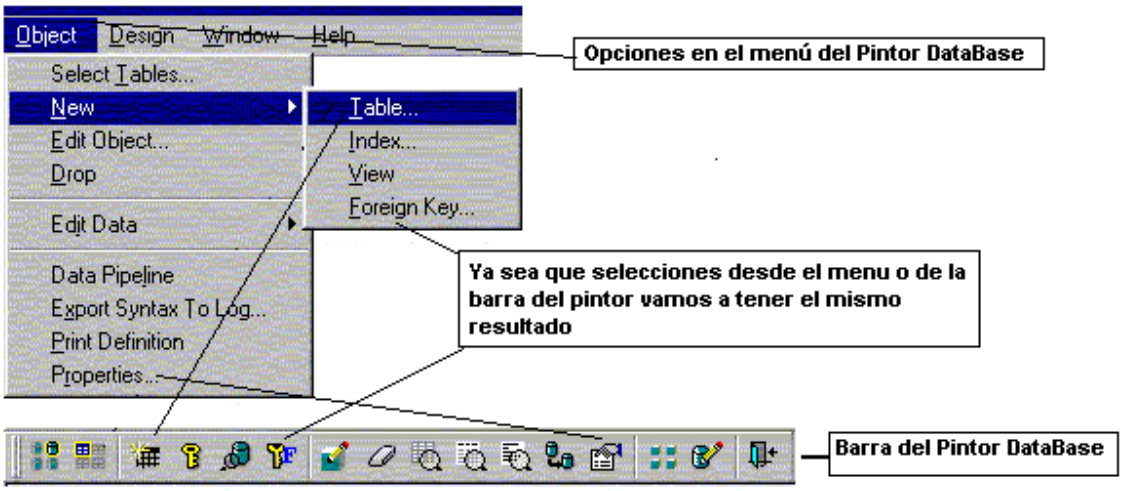
Cambiando la conexión con la Base de Datos
Cuando ud. abre el pintor que comunica con la base de datos (tales como Pintor Bese de Datos o pintor DataWindow). PowerBuilder lo conecta a la ultima base de datos que fue usada si ud. no esta ya conectado. Ud. puede cambiarse a otra bese de datos en cualquier momento.

Acerca del Pintor Base de Datos
Como los otros pintores de PowerBuilder, el pintor de Base de Datos(Database Painter) contiene una barra menu, una barra del pintor PainterBar que se le puede personalizar y un espacio de trabajo donde podemos ver las tablas y sus relaciones.

**********////falta dib//////////**********

PowerBuilder despliega las tablas con sus columnas e iconos que marcan una columna o un conjunto de columnas como una clave primaria, una clave foranea, o un indice. Estas claves e indices fueron definidos con anterioridad.

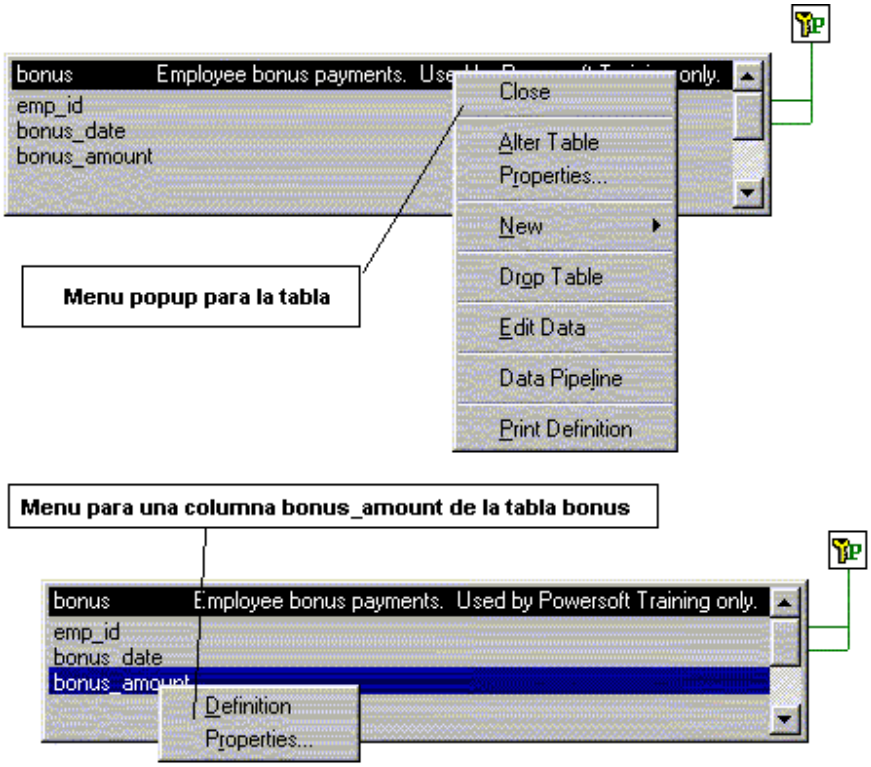
La Menu Bar y en la barra PainterBar
Se puede hacer en la mayoria de actividades comunes en una base de datos desde el pintor DataBase en la opciòn Object del menu del pintor.



La barra del Pintor database PainterBar contiene botones que realizan todasl las actividades listadas en la siguiente tabla:

Como el Pintor Base de datos y el Pintor Tabla trabajan juntos
El pintor base de datos(Database) y el pintor tabla(Table) son pintores independientes, pero cuando estos pintores están abiertos al mismo tiempo estos trabajan juntos. Por ejemplo, cuando ud. crea y graba la definición de una nueva tabla en el pintor Tabla, el pintor Tabla notifica al pintor Base de datos que una nueva tabla existe y que la nueva tabla se desplegará en todas las instancias del pintor de Base de datos. Si ud. agrega un índice para la tabla en el pintor de base de datos, ud. puede ver estos cambios si abre la pagina de propiedad de las propiedades de la tabla para la tabla en el Pintor Tabla.

Trabajando con los objetos(tablas) en el espacio de trabajo del pintor DataBase
Moviendo objetos(tablas) Ud. puede mover los objetos a traves del espacio de trabajo solo con hacer click en el objeto arrastrarlo y soltarlo en el lugar deseado.
Cambiando tamaño de los objetos Se puede cambiar el tamaño de los objetos con solo hacer click en una de las esquinas de los objetos.
Usando en una Tabla o Columna un menú popup Cuando una tabla esta abierta en el espacio de trabajo y tiene columnas podemos desplegar dos menues popup , tanto para la tabla como para una columna determinada, tan solo nos ubicamos en la tabla o coluna deseada y presionamos click derecho y nos aparece los siguientes menúes:



Para el menu popup de la tabla se puede elegir las siguientes opciones:

Seleccione esto	Hace esto
Close	Cerrar la tabla
Alter Table	Abre el pintor tabla de manera que puede modificar la tabla
Properties	Abre la página de propiedades para la página
New	Crea un nuevo índice o clave foránea
Drop Table	Elimina la tabla
Edit Data	Abre el pintor para manipular datos de manera que se puede modificar los datos de la tabla
Data Pipeline	Abre el pintor Data Pipeline de manera que puede crear y definir una tuberia de datos y que la tabla sea la fuente.
Print definition	Imprime la definición de la tabla

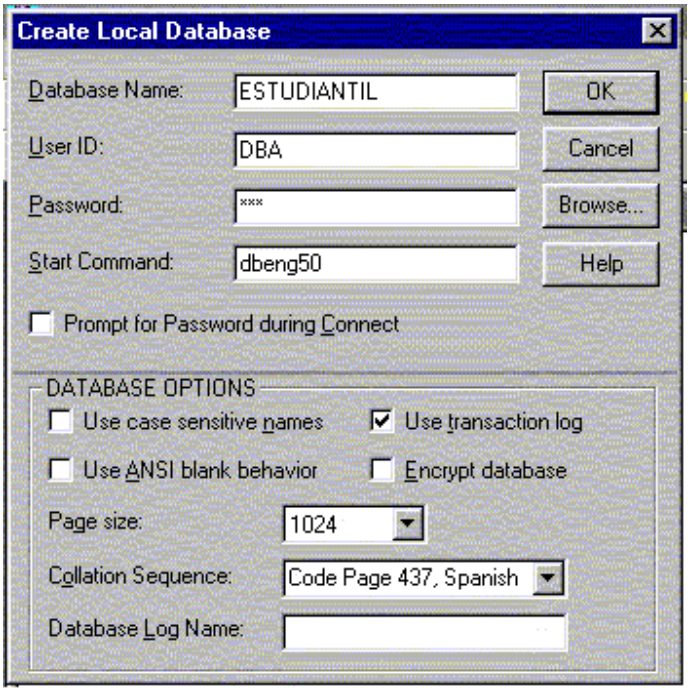
Para el menu popup de una columna de la tabla se puede elegir las siguientes opciones:

Seleccione esto	Hace esto
Definition	Despliega y se puede modificar la definición de la tabla dentro del pintor Tabla

Properties	Despliega las propiedades de la columna y se las puede modificar

Creando y Borrando una Base de Datos SQL AnyWhere
En PowerBuilder ud. puede trabajar dentro de base de datos existentes. Con una excepción, crear y borrar una base de datos es una tarea adiministrativa que no es desempeñada directamente por PowerBuilder. La única excepción, es que ud. puede crear y borra una base de datos local SQL AnyWhere desde PowerBuilder.
Crear una Base de datos Local SQL AnyWhere

1. Abrir el Pintor Base de datos
2. Seleccione en el menu File la opción *Create Database* desde el menu del pintor. Se despliega una ventana de diálogo para crear la base de datos Local.
3. Luego se debe ingresar el nombre de la base de datos y el camino en donde se va a crear.



4. Defina otras propiedades si cree necesario, presione en el botón *More...*
5. Luego presione OK.

Para conectar a la Base de datos creada, vamos la menu File y elegimos la opción ***Connect*** y luego nos sale otro menú en cascad y elegimos la base de datos creada y se conecta al a base de datos.

Pintor Tabla(Table Painter)

Trabajando con tablas en el Pintor Tabla
En el pintor Tabla Ud. puede crear y definir una nueva tabla o modificar una definición de una tabla ya existente, además puede modificar las propiedades de la tabla, y trabajar con índices y claves(primarias o foráneas).

Acerca del pintor Tabla
Aunque el Pintor Tabla(Table Painter) no es un editor, tiene algunas caracteríticas que son similares a la de un editor. En lugar de trabajar con texto, ud. trabaja con las columnas de la tabla. Por ejemplo, se puede copiar una columna y pegar en la definición de otra tabla diferente.

Trabajando con mas de una tabla cada vez ud. puede abrir el pintor tabla , puede elegir la definición de una tabla y cambiarse o crear una nueva tabla. Si ud. quiere trabajar sobre mas de una tabla a la vez, abre nuevas instancias del pintor Tabla para cada tabla.

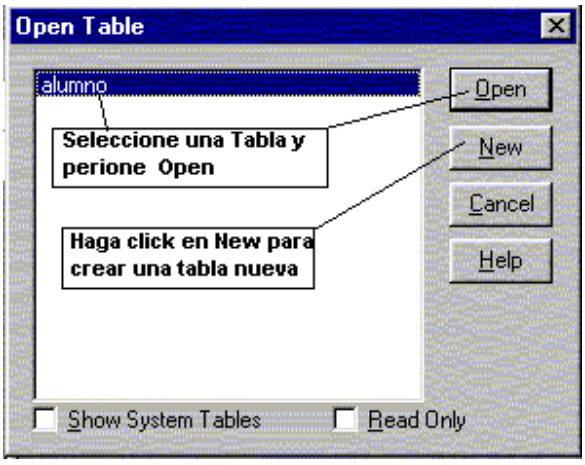
Creando una tabla desde el Pintor Tabla
Ud. puede crear una nueva tabla en PowerBuilder en la base de datos actual. La base de datos actual es la base de datos a la cual PowerBuilder está conectada.

Crear una tabla desde la base de datos actual

1. Haga Click en el botón del pintor Tabla de la barra de herramientas PowerBar.
   Se abre la ventana de diálogo para abrir una tabla o crear una nueva.

2. Luego si hace click en el botón ***New*** , para crear una nueva tabla
   o
   elige una tabla ya existente, para modificar la definición de dicha tabla y haga click en ***Open*** para abrir la tabla.

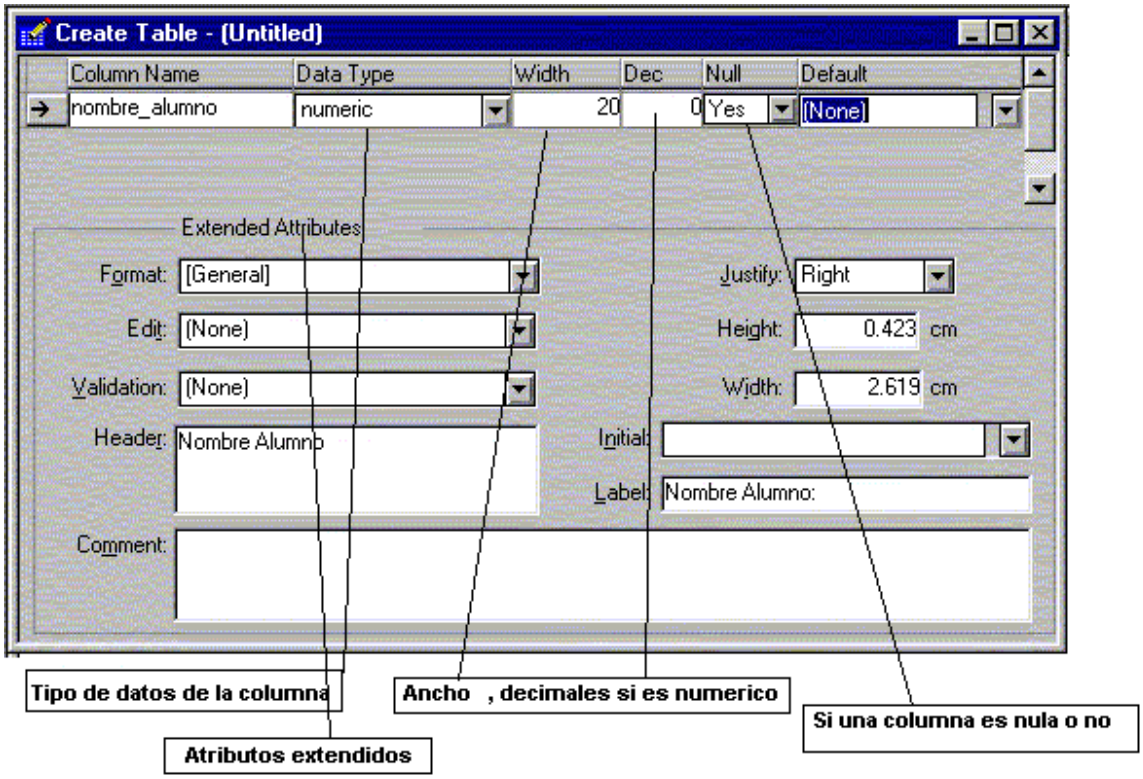
   

   Aparece un espacio de trabajo para crear nuevas columnas.

3. Si elegimos una nueva tabla, debemos ingresar los requerimientos para la primera columna.

Puede ingresar el nombre de la columna , su tipo de datos, si es nulo o no y algún valos por default.

4. (Opcional) Especifique los atributos extendidos(Extended Attributes) para la columna.
   Se puede ingresar ahora o mas tarde cuando modifique la tabla.



5. Repita los pasos 3 y 4 hasta que ingrese todas las columnas de la tabla.
6. Luego presione el botón para grabar la nueva tabla y poner un nombre a la tabla.
7. Haga click en Close para cerrar la tabla.

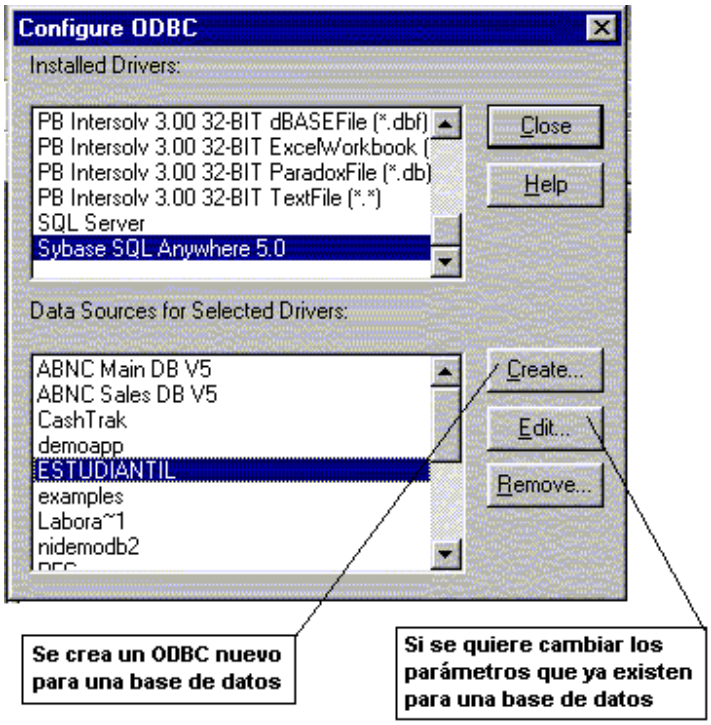
Pintor Configurar ODBC(Configure ODBC Painter)

Acerca de configurar el ODBC
Una vez que se ha creado una base de datos , podemos ingresar ciertos parámetros que necesita el administrador ODBC , para según eso conectarnos al motor de base de datos y luego elegir el archivo de la base de datos y poner los parametros que sean necesarios.

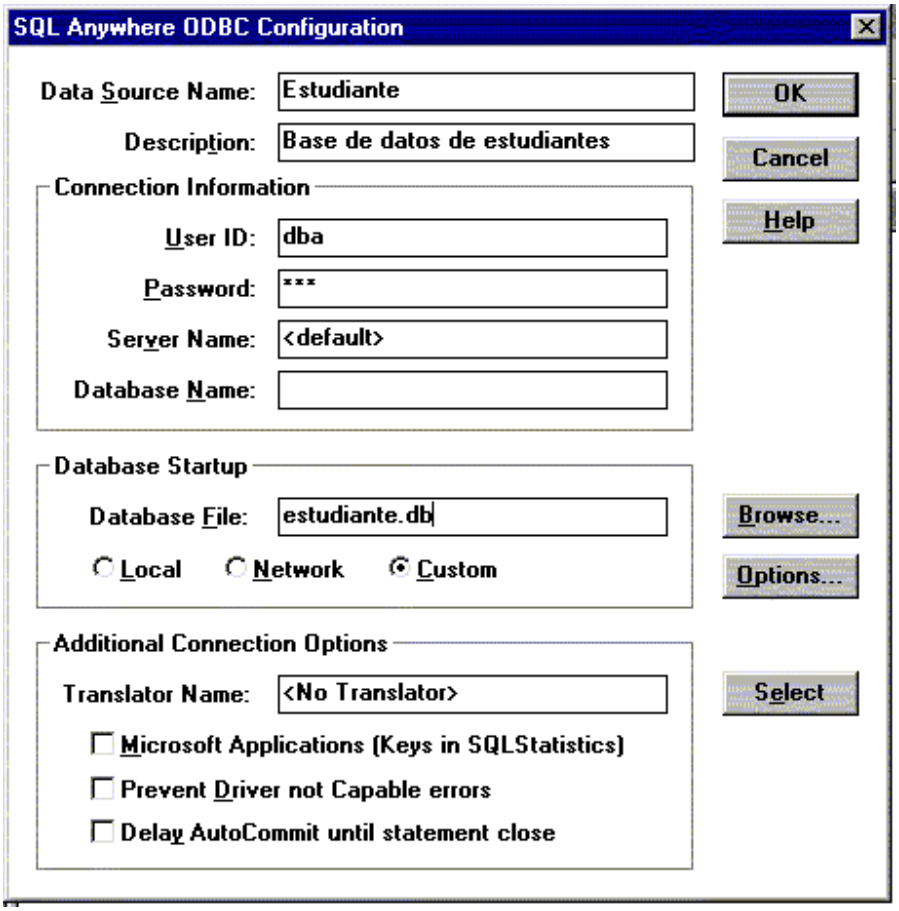
Cómo configurar una base de datos

1. Haga Click en el botón del pintor Configurar ODBC de la barra de herramientas PowerBar.

2. Luego aparece una ventana de diálogo para configurar el ODBC, en donde primero tenemos los manejadores instalados de los diferentes Motores de base de datos, por ejemplo Sybase SQL AnyWhere 5.0 , y en otra ventana están los archvios de las bases de datos que tiene cada motor de base de datos.



3. Si queremos crear una nueva configuración ODBC para un archivo de base de datos presionamos el boton ***Create...***, y aparece la siguiente pantalla:



La ventana de diálogo de la configuración ODBC SQL Anywhere DBC contiene los siguietes campos. Estos campos corresponden a los parámetros de conexión. Mire Connection Parameters para una descripcion de los parametros de conexión y una descripción de la manera en que estos se usan para establecer una conexión con una base de datos.

Data Source Name Se ingresa un nombre corto de la fuente de datos, tal como Ordenes de ventas.
Descripción Una descripción detallada de la fuente de datos.
User ID (Opcional) El nombre del usuario que va usar cuando se realiza la conexión.
Password (Opcional) El password para proporcionar un identificador de usuario. Desde que se proporciona el Password es almacenado en el odbc.ini, colocando el password aqui deberia ser una seguridad contra riesgos.
Server Name El nombre de un motor de base de datos SQL Anywhere o el nombre de un servidor de red SQL Anywhere. Si no es especificado, por default se carga el motor local (el primer motor de base de datos iniciado). Este campo corresponde al parámetro de conexión EngineName.
Database Name Si se especifica, este corresponde al nombre de la base de datos que ya esta corriendo sobre el motor de base de datos SQL Anywhere o el servidor de red SQL Anywhere. Este campo corresponde al parámetro de conexión

DatabaseName.
Database File Si se especifica, este contiene el nombre del archivo de la base de datos--tales como c:\sqlany50\sademo.db. Ud. hacer click en el boton Browse para localizar al archivo de base de datos. Este campo corresponde al parámetro de conexión DatabaseFile.
Local, Network, Custom El commando usado para correr el software de base de datos cuando el nombre del motor de base de datos o servidor no se está ejecutando. Ud. puede seleccionar Local o Network, como aproppiado, si el conjunto de parametros están correctamente. De otra manera, seleccione Custom e ingrese el comando incluyendo algún comando en la linea parametros presionando el boton **Opciones**.
Translator Name Si se especifica, este contiene el nombre de un traductor ODBC. Un Traductor DLL causa que todos los datos pasen entre un aplicación y la base de datos a traducirse.

 Pintor Perfil de una Base de Datos(Profile DataBase Painter)

La conexión a una base de datos incluye información acerca de perfiles de la base de datos. Perfiles, que son definidos en el PB.INI, provee de información necesaria para conectarse a la fuente de datos desde el ambiente en que fue desarrollado.

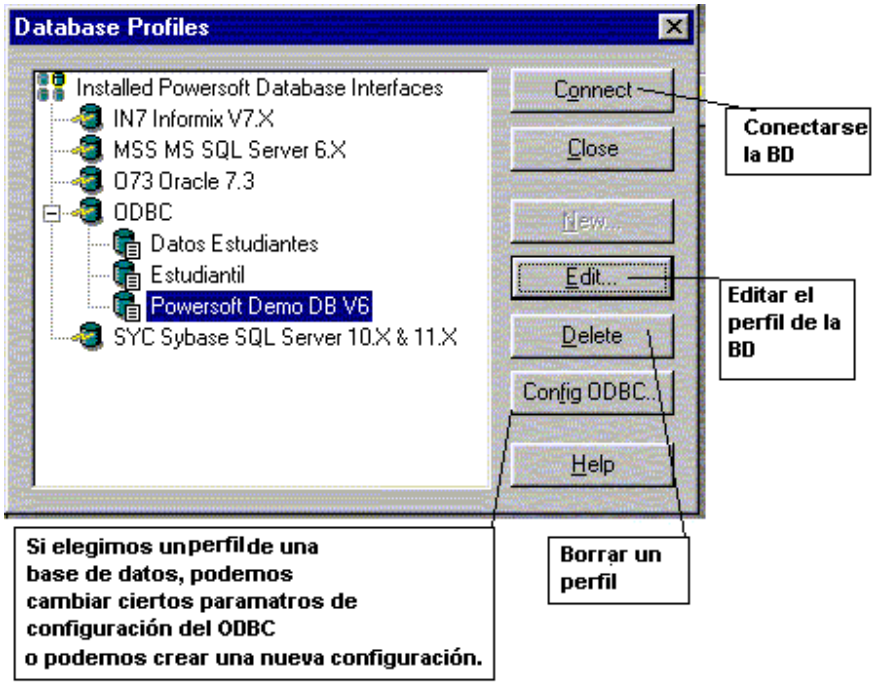
Antes de tener un Perfil de una Base de datos
Para crear un perfil de Base de datos, debemos primero configurar el ODBC, e ingresar todos los parámetros necesarios de la base de datos , e ingresar un *nombre de la fuente de datos*, éste nombre, es para PowerBuilder el Perfil de la base de datos que va a utilizar cuando se va a conectar a la base de datos.
Luego de haber configurado el ODBC para una base de datos, podemos ingresar a través del pintor de Perfiles de base de datos y poder ver un determinado perfil de una base de datos, y ver sus parámetros y agregar más opciones sobre el perfil.
Además si queremos conectarnos a una base de datos que ya tiene un perfil creado , podemos hacerlo a través de este pintor, elegimos el perfil de la base de datos y automáticamente el PowerBuilder se conecta a la base de datos.

Como Modificar/Eliminar Un Perfile de una Base de Datos

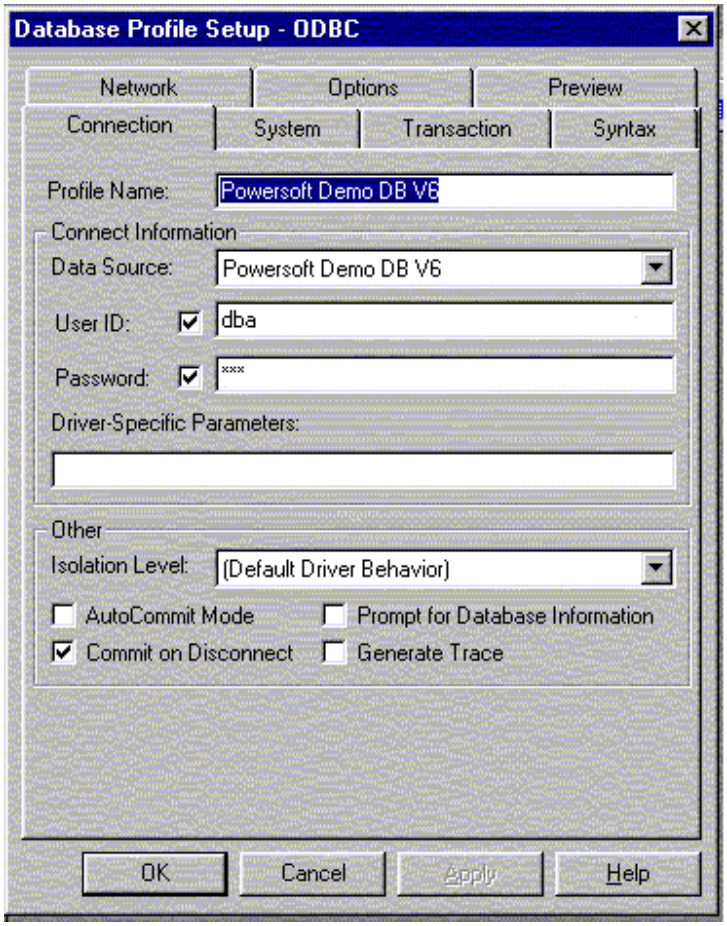
1. Haga Click en el botón del pintor Perfil de la base de dato de la barra de herramientas PowerBar.

2. Luego aparece una ventana de diálogo para ver los diferentes perfiles de Base de Datos de los diferentes Motores de base de datos existentes.



3. Si presionamos el botón **Edit**, podemos cambiar ciertos parámetros o agregar otros al perfil, por ejemplo: agregar ciertos parámetros para hacer que la base se pueda conectarse en red.



****

Manejando el archivo de Inicialización

Cuando inicia el Power Builder, el PowerBuilder busca por el archivo de inicialización y configura su entorno.

Acerca del archivo de Inicialización

El Archivo de inicialización es un archivo de texto que contiene variables que especifican las preferencias de PowerBuilder. Estas preferencias incluyen cosas como, la última base de datos a la cual estaba conectada, el PBL que se estaba usando.

Formato de Archivos .INI

El archivo de inicialización de PowerBuilder el formato de los archivos INI de windows en todas las plataformas. Hay tres tios de elemntos:

- Nombre de la Sección, los cuales son encerradas entre corchetes Ej: [database]

- Palabras Claves, en los cuales son los nombres del conjunto de preferencias
- Valores, pueden ser numéricos o cadenas de texto, asignadas como un valor que se asocian a la palabra Clave.

Sección	Lo que contiene
Application	El nombre y localización de la aplicación actual y librerias de PowerBuilder, y una historia de aplicaciones pervias .
PB	Barra de herramientas básico, tamaño de las ventanas, y preferencias en generación de código, así como los nombres de los objetos más recientemente abiertos.
DataBase	El perfil de la base de datos actual. la lista de los DBMS's disponibles, y otras preferencias del pintor de Base de datos(DataBase painter).
DBMS_Profiles	El nombre del perfil de la base de datos actual y una lista de otros perfiles.
Profile *name*	El perfil de la base de datos en donde su nombre es *name*
Debug	El esquema de la ventana actual que se esta depurando, y los puntos de quiebre y las ventanas de expresión

Ejemplo de un archivo de inicialización laboratorio.ini :

```
[Database]
DBMS=ODBC
Database=Laboratorio.db
UserId=
DatabasePassword=
LogPassword=
ServerName=
DbParm=Connectstring='DSN=Laboratorio'
Prompt=0

[Debug]
VariablesWindow=0
WatchWindow=0

[DBMS_PROFILES]
CURRENT=Laboratorio.pbl
PROFILES='Laboratorio.pbl','ABNC Main DB V5','Powersoft Demo DB V5','ABNC
Sales DB V5'
History='Laboratorio.pbl','ABNC Main DB V5','Powersoft Demo DB V5','ABNC
Sales DB V5'
```

Lenguaje Básico

Comentarios

Descripción
Se puede usar comentarios para documentar su scripts y evitar sentencias dentro de un script de ejecución. hay dos métodos:

Sintaxis
Método Doble-slash
Código // Comentario

Método Slash-y-asterisco
/* Comentario */

Uso
Agregar comentarios
En los pintores tanto en el PowerScript y Funciones, ud. puede usar el botón de Seleción de comentarios(Comment Selection button) o seleccionar del menu la opción ***Edit>Comment Selection*** o de la barra de menú) para poner comentarios en una línea o seleccionar un grupo de líneas.

Ejemplos
Metodo Doble-slash
// Esta línea completa es un comentario.
amt = qty * cost // El resto de la linea es un comentario como un apoyo

Método Slash-y-asterisco
/* Esta linea sinple en un comentario. */
A = B + C /* Este es un comentario luego de la suma /*

Nombre de Identificadores

Descripción
Puede usar identificadores par nombrar variables, etiquetas(labels), funciones, ventanas(windows), controles, menus, y cualquier otro que se refiera en un script.

Sintaxis
Reglas para identificadores:

- debe empezar con una letra o un _ (underscore) Ej: _nombre
- No puede ser una palabra reservada Ej: Else, while , etc.
- Puede tener hasta 40 caracteres pero sin espacios en blanco.
- Estos casos para PowerBuilder son idénticos: PART, Part, y part
- Puede incluir combinaciones entre letras, números y los siguientes caracteres especiales:
  - Guión(Dash)
  _ Línea Baja(Underscore)
  $ Signo de dolar
  # Signo de número
  % signo de porcentaje.

Uso
Por defecto, PowerBuilder permite que ud. use guiones en todo identificador,incluyendo en nombre de variables en un script. Esto significa que cuando usa el operador de resta(sustracción) o el operador -- en un script, ud debe rodearlos con espacios en blanco(de otra manera, PowerBuilder piensa que la expresión es un nombre de unidentificador).
If you want to disallow dashes in variable names in scripts, you can change the setting of the Allow Dashes in Identifiers option in the script editor's property sheet. This way you do not have to surround the subtraction operator and the decrement assignment shortcut (--) with spaces. Be carefulIf you disallow dashes and have previously used dashes in variable names, you will get errors the next time you compile.

Ejemplos
Identificadores Válidos
ABC_Code
Child-Id
FirstButton
response35
pay-before%deductions$
ORDER_DATE
Actual-$-amount
Part#

Identificadores no Válidos
2nd-quantity // No empieza con una letra
ABC Code // contiene un espacio en blanco
Child'sId // Contiene un caracter especial invalido(').

Etiquetas(Labels)

Descripción
Ud. puede incluir etiquetas en un script para usar con la sentencia GOTO

Sintaxis
Identificador :

Uso
Una etiqueta puede ser un identificador válido. Ud. puede ingresarlo en la misma línea al inicio de la sentencia o sobre la sentencia.

Ejemplo
On a line by itself above the statement
FindCity: IF city=cityname[1] THEN ...
Inicia antes de la sentencia en la misma línea.

Sentencias SQL

COMMIT

Descripción

Permanente actualiza todas las operaciones desde antes de un COMMIT, ROLLBACK, o CONNECT para el objeto transacción especificado.
Commit termina la unidad lógica de trabajo, garantiza todos los cambios hechos en la base de datos desde el inicio de la unidad de trabajo actual y que es permanente, y empieza la nueva unidad lógica de trabajo.

Sintaxis

COMMIT {USING TransactionObject} ;

Parámetro	Descripción
TransactionObject	El nombre del objeto transacción para el cual ud. quiere permanentemente actualizar todas las operaciones de la base de datos desde el ultimo COMMIT, ROLLBACK, o CONNECT. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso

COMMIT no causa una desconexión, pero cierra todos los cursores o procedimientos abiertos.
(Pero note que la sentencia DISCONNECT en PowerBuilder si emite un COMMIT.)

Ejemplos

Ejemplo 1
Esta sentencia comete todas las operaciones para la base de datos especificada en el objeto transaccion por defecto.

```
COMMIT ;
```

Ejemplo 2
Esta sentencia comete todas las operaciones para la base de datos especificada en el objeto transacción llamado emp_tran

```
COMMIT USING emp_tran;
```

CONNECT

Descripción

Conecta a una base de datos especifica.

Sintaxis

CONNECT {USING TransactionObject} ;

Parámetro	Descripción
TransactionObject	El nombre del obieto transacción conteniendo la

	información requerida de conexión para la base de datos a la cual ud. quiere conectarse. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso
Esta sentencia debe ser ejecutada antes de alguna acción(tales como INSERT, UPDATE, o DELETE) que puede ser procesada usando el objeto transacción por defecto o el objeto transacción especificado.

Ejemplos
Ejenplo 1
Esta sentencia conecta a la bse de datos epecificada en el objeto transacción por defecto:
CONNECT ;

Ejemplo 2
Esta sentencia conecta a la bse de datos epecificada en el objeto transacción llamado Emp_tran:
CONNECT USING Emp_tran ;

DELETE

Descripción
Borra las filas de la tabla(TableName) especificada dado algún creterio.

Sintaxis
DELETE FROM TableName **WHERE** Criteria {**USING** TransactionObject} ;

Parámetro	Descripción
TableName	El nombre de la tabla desde la cual ud. quiere borrar las filas
Criteria	Criterios que especifica cuales filas serán borradas
TransactionObject	El nombre del objeto transacción que identifica la base de datos que contiene la tabla. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso
Cuando se desea eliminar filas de una tabla específica, a la cual se puede dar criterios de eliminación.

Ejemplos
Ejemplo 1
Esta sentencia borra las filas de la tabla Empleado en donde la columna Emp_num es menor que 100:

DELETE FROM Empleado
WHERE Emp_num (menor que) 100 ;

Ejemplo 2
Esta sentencia borra las filas de la tabla empleado en la base de datos especificada en el objeto transacción llamado Emp_tran donde Emp_num es igual a un valor ingresado en la SingleLineEdit sle_number:
int Emp_num
Emp_num = Integer(sle_number.Text)
DELETE FROM Employee
WHERE Empleado.Emp_num = :Emp_num ;
USING Emp_tran;

DISCONNECT

Descripción
Ejecuta un COMMIT para el objeto transacciòn especificado y entonces se desconecta desde la base de datos especificada.

Sintaxis
DISCONNECT {USING TransactionObject} ;

Parámetro	Descripción
TransactionObject	El nombre del objeto transacción que identifica la base de datos que ud. quiere desconectar. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso
Se usa para deconectar una base de datos.

Ejemplos
Ejemplo 1
Esta sentencia desconecta a la base de datos especificada en el objeto transacciòn por default.
DISCONNECT ;

Ejemplo 2
Esta sentencia desconecta a la base de datos especificada en el objeto transacciòn llamado Emp_tran:
DISCONNECT USING Emp_tran ;

INSERT

Descripción

Inserta una o más nuevas filas en una tabla especificada en RestOfInsertStatement.

Sintaxis

INSERT RestOfInsertStatement {USING TransactionObject} ;

Parámetro	Descripción
RestOfInsertStatement	El apoyo de la sentencia INSERT (la clausula INTO, lista de columnas y valores o el origen)
TransactionObject	El nombre del objeto transacción que identifica la base de datos que contiene la tabla. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso

La sentencia INSERT se utiliza para insertar filas a una tabla de un base de datos.

Ejemplos

Ejemplo 1

Esta sentencia inserta una fila con los valores en tienen la variables Emp_Nbr y Emp_Name en las columnas Empnbr y Empname de la tabla Empleado identificada por el objeto transacción por defecto transaction object:

```
int Emp_Nbr
string Emp_Name
INSERT INTO Empleado (empleado.Empnbr,empleado.Empname) VALUES (:Emp_Nbr,
:Emp_Name) ;
```

Ejemplo 2

Este ejemplo inserta una fila con los valores ingresados en el SingleLineEdits(campo de edición) sle_number y sle_name en las columnas Emp_nbr and Emp_name dela tabla empleado en el objeto transacción llamado Emp_tran:

```
int EmpNbr
EmpNbr = Integer(sle_number.Text)
INSERT INTO Employee (employee.Emp_nbr, employee.Emp_name)
USING Emp_tran ;
```

ROLLBACK

Descripción

Cancela todas las operaciones de base de datos en la Base de datos Especidicada desde el último COMMIT, ROLLBACK, or CONNECT. RollBack que no cause una desconexión.

Sintaxis

ROLLBACK {USING TransactionObject} ;

Parámetro	Descripción
TransactionObject	El nombre del objeto transacción que identifica la base de datos en la cual ud. quiere cancelar todas las operaciones desde el ultimo COMMIT, ROLLBACK, or CONNECT. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso
ROLLBACK no causa una disconexión, pero hace que se cierren todos los cursores o procedimientos abiertos.

Ejemplos
Ejemplo 1
Esta sentencia cancela todas las operaciones en la base de datos especificada en el objeto transacción por defecto(SQLCA)
ROLLBACK ; Ejemplo 2
Esta sentencia cancela todas las operaciones en la base de datos especificada en el objeto transacción llamada Emp_tran.
ROLLBACK USING Emp_tran;

SELECT

Descripción
Selecciona una fila de la tabla especificada en RestOfSelectStatement.

Sintaxis
SELECT RestOfSelectStatement {USING TransactionObject} ;

Parámetro	Descripción
RestOfInsertStatement	El apoyo de la sentencia SELECT (las clausula INTO, FROM, WHERE, y otras clausulas)
TransactionObject	El nombre del objeto transacción que identifica la base de datos que contiene la tabla. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso
Un error ocurre si la sentencia SELECT retorna mas que una fila.

Ejemplo
Los siguientes datos de la sentencia SELECT en las columnasd Emp_LName y Emp_FName de una fila en la tabla empleado y pone los datos en los SingleLineEdits(campo de edición) sle_LName y sle_FName (el objeto transacción Emp_tran es usado):

int Emp_num
Emp_num = Integer(sle_Emp_Num.Text)
SELECT empleado.Emp_LName, empleado.Emp_FName
INTO :sle_LName.text, :sle_FName.text FROM Employee
WHERE Empleado.Emp_nbr = :Emp_num
USING Emp_tran ;

```
if Emp_tran.SQLCode = 100 then
   MessageBox("Información Empleado", "Empleado no encontrado")
elseif Emp_tran.SQLCode > 0 then
   MessageBox("Error Database", Emp_tran.SQLErrText, Exclamation!)
End If
```

UPDATE

Descripción
Actualiza las filas especificadas en el paràmetro RestOfUpdateStatement.

Sintaxis
UPDATE TableName RestOfUpdateStatement {USING TransactionObject} ;

Parámetro	Descripción
TableName	El nombre de la tabla en la cual ud. quiere actualizar las filas.
RestOfUpdateStatement	El apoyo de la sentencia UPDATE (las clausula SET y WHERE)
TransactionObject	El nombre del objeto transacción que identifica la base de datos que contiene la tabla. Esta clausula es requerida solamente para los objetos transacción que son diferentes al objeto transacción por default (SQLCA)

Uso
Esta sentencia sirve para actualizar datos de una tabla.

Ejemplos
Esta sentencia actualiza filas de la tabla empleado en la base de datos especificada en el objeto transacciòn llamado Emp_tran donde Emp_num es igual al valor ingresado en el SingleLineEdit sle_Number:
int Emp_num
Emp_num=Integer(sle_Number.Text)
UPDATE Empleado
SET emp_name = :sle_Name.Text
WHERE Employee.emp_num = :Emp_num
USING Emp_tran ; IF Emptran.SQLNRows > 0 THEN
COMMIT USING Emp_tran ; END IF

Funciones de Power Builder

AcceptText()

Descripción
Aplica el contenido del control de edición(edit control) del Datawindow al item actual en el buffer de un control DataWindow o DataStore. El dato en el control de edición debe pasar la regla de validación para la columna antes de poder ser almacenado en el item.

Aplicado a
Controles DataWindow, objetos DataStore, y DataWindows child().

Sintaxis
dwcontrol.AcceptText()

Argumento	Descripción
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow que el que ud. quiere aceptar datos ingresados en el control de edición(edit control)
transaction	El Nombre del objeto transacción que ud. quiere usar en el dwcontrol

Valor que Returna
Integer. Returna 1 si ha sicedido y -1 si a acurrido un error(por ejemplo, si el dato no ha pasado la validación). Si el dwcontrol es nulo, AcceptText retorna NULL.

Uso
Cuando el usuario se mueve de un item a otro en un control datawindow, el control valida y acepta lo que el usuario ha ingresado. Cuando el usuario a modificado un item en un Datawindow e inmediatamente cambia el enfoque a otro control en la ventana, el control Datawindow no acepta el dato modificado -los restos de los datos en el edit control . Use la función AcceptText en esta situación y asegurarse que el objeto Datawindow contenga los datos que el usuario a ingresado. Un tipico lugar para llamar a la función AcceptText es en el evento LoseFocus del Datawindow. AcceptText en el evento ItemChanged no tiene efecto.

Ejemplos
En este ejemplo, el usuario espera ingresar un valor de código(tal como numero de empleado) en una columna de un objeto Datawindow , y luego hacer click en el botón Ok. Este es el script para el evento clicked del botón OK , llama a AcceptText y valida la entrada y si esta correcto recupera datos del empleado.

```
IF dw_emp.AcceptText() = 1 THEN
dw_emp.Retrieve(dw_emp.GetItemString(dw_emp.GetRow(), dw_emp.GetColumn()))
END IF
```

SetTransObject()

Descripción
Causa que un programador específico use el Control Datawindow o DataStore como un objeto transaccional. Un objeto transaccional provee de la información necesaria para la comunicación con la Base. De Datos.

Sintaxis
dwcontrol.SetTransObject (transaction)

Argumento	Descripción
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow que el programador quiere usar como objeto transaccional
transaction	El Nombre del objeto transacción que ud. quiere usar en el dwcontrol

Valor que Returna
Integer. Returna 1 si ha sicedido y -1 si a acurrido un error. Si el valor del argumento es nulo, SetTransObject retorna NULL.

Uso
Un objeto transaction que usa el programador le da mas control sobre las transacciones en la base de datos y provee un desempeño eficiente en la aplicación. Ud. controla la conexión a la base de datos para usar sentencias SQL tales como CONNECT, COMMIT, and ROLLBACK.
Ejemplos :
dw_profesor.SetTransObject(SQLCA)

dw_profesor.SetTransObject(emp_transobject)

```
IF dw_Empleado.Update()>0 THEN
      COMMIT USING emp_transobject;
ELSE
       ROLLBACK USING emp_transobject;
```

Retrieve()

Descripción
Recupera filas de la Base de Datos para un Control Datawindow o DataStore. Si son incluidos argumentos , los valores de los argumentos son usados para la recuperación de filas en la sentencia SQL SELECT para el objeto DataWindow o DataWindow hijo.

Sintaxis
dwcontrol.Retrieve ({argumento1, argumento2 ...})

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow que ud. quiere para recuperar filas de la base de datos.
argument(opcional)	Uno o ma´s valores que ud. quiere usar como argumentos de recuperación en la sentencia SQL SELECT definida en el dwcontrol

Valor que Returna
Long. Retorna el número de filas desplegadas si ha sucedido y -1 si ha fallado. Si el valor del argumento es nulo, Retrieve retorna NULL.

Uso
Para recuperar filas de una tabla de la base de datos.
Antes que ud. recupere las filas para un DataWindow control or DataStore, debe especificar objeto transaction con SetTransObject or SetTrans. Si usa SetTransObject, debe además usar sentencia SQL CONNECT para establecer conexión con la base de datos.

Ejemplo:

```
If  dw_profesor.Retrieve() = -1 Then
        RollBack ;
        messagebox("Error","Fallo la recuperación",Exclamation!)
Else
        Commit ;
End If
```

Ej : En el siguiente ejemplo mandamos un parámetro que es el código de un profesor cualquiera y nos recupera datos solo de ese código.

```
If  dw_profesor.Retrieve( li_codigo) = -1 Then
        RollBack ;
        messagebox("Error","Fallo la recuperación",Exclamation!)
Else
        Commit ;
        dw_ profesor.SetRowFocusIndicator(Hand!)
        dw_ profesor.SetFocus()
End If
```

SetItem()

Descripción
Coloca en la fila y columna específica de un datawindow control o dataStore un valor especificado.

Sintaxis
dwcontrol.SetItem (row, column, value)

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child

	DataWindow en el cual ud. quiere colocar en una determinada fila y columna un valor
row	Un entero(long) cuyo valor sea la ubicación de la fila del dato
column	La ubicación de la columna del dato. La Columna puede ser un numero(integer) o el nombre de la cadena(string)
value	El valor que ud. quiere colocar en la ubicación fila y columna. El tipo de dato del valor debe ser el mismo tipo que el de la columna

Valor que retorna
Integer. Returna 1 si ha sucedido y -1 si a acurrido un error.

Uso
SetItem coloca un valor en el buffer del DataWindow.

Ejemplo : En el siguiente ejemplo, se coloca en la fila tres y en la columna Pro_nombre, el valor de José Perez.
dw_profesor.SetItem(3, "pro_nombre", "Jose Perez")

GetItemString()

Descripción
Obtiene un dato cuyo tipo es cadena del buffer especificado de un control DataWindow o un objeto DataStore.

Sintaxis
dwcontrol.GetItemString(row,column{,dwbuffer,originalvalue})

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere obtener la cadena de datos contenida en una fila y columna especificada.
row	Un entero(long) cuyo valor sea la ubicación de la fila del dato
column	La ubicación de la columna del dato. La Columna puede ser un numero(integer) o el nombre de la cadena(string)
dwbuffer(opcional)	Un valor del dwBuffer enumerado por el tipo de dato identificando el buffer del DataWindow
originalvalue (opcional)	Un boolean indicando si ud. quiere el original o el valor actuals para una fila y columna 1. True- Retorna el valor original

	2. False- (Default) Retorna el valor actual <>

Valor que retorna
String. Retorna NULL si el valor de la columna es NULL. Retorna cadena vacia("") si un error ha ocurrido. Si valor de algún argumento es nulo, GetItemString retorna NULL.

Uso
Se usa GetItemString para conseguir información del buffers de un DataWindow .

Ejemplo:
En el ejemplo recupero de la fila cinco y columna "nombre_profesor" el nombre del profesor . string ls_nombre
ls_nombre = dw_profesor.GetItemString(5,"nombre_profesor")

GetItemNumber()

Descripción
Recupera un dato numerico desde el buffer especificado de un control DataWindow o un objeto DataStore. Puede obtener el dato que fue originalmente recuperado y almacenado en la base de datos desde el buffer original.

Sintaxis
dwcontrol.GetItemNumber(fila,columna{,dwbuffer,originalvalue})

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere obtener el dato numérico contenido en una fila y columna especificada.
row	Un entero(long) cuyo valor sea la ubicación de la fila del dato
column	La ubicación de la columna del dato. La Columna puede ser un numero(integer) o el nombre de la cadena(string)
dwbuffer(opcional)	Un valor del dwBuffer enumerado por el tipo de dato identificando el buffer del DataWindow
originalvalue (opcional)	Un boolean indicando si ud. quiere el original o el valor actuals para una fila y columna 1. True- Retorna el valor original 2. False- (Default) Retorna el valor actual <>

Valor que retorna
Un tipo de dato numérico (decimal, double, integer, long, or real). Se diapara el evento

SystemError y retorna -1 si ha ocurrido un error. Si el valor de algún argumento es nulo, GetItemNumber retorna NULL.

Uso
Se usa GetItemNumber para conseguir información del buffers de un DataWindow .

Ej : En el ejemplo recupero de la fila cinco y columna "cod_profesor" el código del profesor . int li_numero
ls_numero = dw_profesor.GetItemNumber(5,"cod_profesor")

GetItemDate()

Descripción
Recupera un dato cuyo tipo es fecha desde el buffer especificado de un control DataWindow o un objeto DataStore. Puede obtener el dato que fue originalmente recuperado y almacenado en la base de datos desde el buffer original.

Sintaxis
dwcontrol.GetItemDate(fila,columna{,dwbuffer,originalvalue})

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere obtener el dato de una fecha contenido en una fila y columna especificada.
row	Un entero(long) cuyo valor sea la ubicación de la fila del dato
column	La ubicación de la columna del dato. La Columna puede ser un numero(integer) o el nombre de la cadena(string)
dwbuffer(opcional)	Un valor del dwBuffer enumerado por el tipo de dato identificando el buffer del DataWindow
originalvalue (opcional)	Un boolean indicando si ud. quiere el original o el valor actuals para una fila y columna 1. True- Retorna el valor original 2. False- (Default) Retorna el valor actual <>

Valor que retorna
Date. Retorna NULL si el valor de la columna es NULL. Retorna 1900-01-01 si ha ocurrido un error. Si el valor de algún argumento es nulo, GetItemDate retorna NULL.

Uso
Se usa GetItemNumber para conseguir información del buffers de un DataWindow .

Ej : En el ejemplo recupero la fecha dela fila cinco y columna "cod_fecha" . date ld_fecha
ld_fecha = dw_profesor.GetItemdate(5,"cod_fecha")

RowCount()

Descripción

Obtiene el número de filas que están actualmente disponibles en un control DataWindow o DataStore.

Sintaxis

dwcontrol.RowCount()

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere el numero de filas actualmente disponibles

Valor que Retorna

Long. Retornas el numero de filasque esta´n actualmente disponible en el dwcontrol, 0 si no hay filas, y -1 si ha ocurrido un error. Si el dwcontrol es NULL, RowCount retorna NULL.

Uso

El Buffer primario para un control DataWindow o DataStore contiene las filas que están disponibles actualmenste desplegadas o por impresora. Esta filas son contadaspor RowCount. El número actualmente de filas disponibles es igual al número total de filas recuperadas menos alñguna fila borrada más alguna fila que ha sido insertada ,menos algunas filas que han sido flitradas.

Ejemplo : If dw_profesor.RowCount()=0
Then MessageBox("Información","No existen registros")

UpDate()

Descripción

Actualiza la Base de datos con los cambios hechos en el control DataWindow o dataStore. Update puede además llamar a AcceptText a la fila y columna actual antes de actualizar la base de datos.

Sintaxis

dwcontrol.Update({accept,resetflag})

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere obtener que contiene la información que ud. quiere usar para actualizar la base de

	datos.
accept (opcional)	Un valor booleano especificando si el control DataWindow o DataStore debería automáticamente realizar un AcceptText antes de de realizar la actualización: l. TRUE (Default) Realiza AcceptText. La actualización Falla si la validacion de los datos falla. 2. FALSE . No realiza el AcceptText
resetflag (optional)	Un valor boleano especificando si dwcontrol debería automáticamnete resetear las banderas de actualización: l. TRUE- (Default) Resetea las banderas 2. FALSE- No resetea las banderas.

Valor que retorna
Integer. Retorna 1 si ha secudido correctamente -1 si ha ocurrido un error.
Si el valor del argumento es NULL, Update retorna NULL.

Uso
Ud. debe usar la función SetTrans o SetTransObject para especificar la connexión con la base de datos antes de ejecutar la función Update.
Cuando usa SetTransObject, es la mas eficiente de las dos funciones, debe hacer su propia transacción, en los cuales puede incluir la sentencias SQL COMMIT o ROLLBACK al finalizar la actualización.

```
Ejemplo :
        If dw_profesor.Update()= -1 Then
                Rollback;
                MessageBox("Error","No se pudo grabar",Exclamation!)
        Else
                Commit;
                MessageBox("Información","Se pudo grabar con éxito")
        End if
```

InsertRow()

Descripción
Inserta una fila en el control DataWindow o DataStore.

Sintaxis
dwcontrol.InsertRow(row)

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere insertar una fila.
row	Un identificador entero(long) de la fila antes que ud. inserte la fila Inserta una fila al final si se especifica row=0.

Valor que retorna
Un Long(entero largo), que es el número de la fila que fue agregada si sucedió y -1 su ocurrió un error. Si el valor del argumento es nulo, la función retorna nulo(NULL).

Uso
InsertRow simplemente inserta una fila sin cambiar el cursor a la fila que se inserta, para desplazarse por la pantalla hasta la fila insertada se utiliza la función ScrollToRow o simpelmente haga de la fila insertada la fila actual llamando a la función SetRow.

Ejemplo : En el ejemplo , cuando se pone parámetro de fila cero, se inserta una fila siempre al último del datawindow.
int li_fila
li_fila = dw_profesor.InsertRow(0)

DeleteRow()

Descripción
Elimina una fila en el control DataWindow o DataStore.
Sintaxis
dwcontrol.DeleteRow(*row*)

Argumento	Descripcion
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud quiere borrar una fila.
row	Un identificador entero(long)de la fila que ud. quiere borrar.Borrar la fila actual se especifica con el valor 0 para row

Valor que retorna
Un entero, retorna 1 si ha borrado satisfactoriamnete y -1 si ha ocurrido un error. Si el valor del argumento es nulo, la función retorna nulo(NULL).

Uso
DeleteRow borra la fila del buffer primario del Datawindow.
La fila no es borrada de la tabla de la base de datos hasta que la aplicación llame a la función Update.

```
Ejemplo :  En el ejemplo , cuando se pone parámetro de fila.
If MessageBox("Confirmar","Eliminar registro?",Question!,YesNo!) = 1 Then
        dw_profesor.DeleteRow(8)
End if
```

GetRow()

Descripción
Nos devuelve el valor de la fila actual en el control DataWindow

Sintaxis
dwcontrol.GetRow()

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere el numero de la fila actual

Valor que retorna
Un entero largo(Long), Retorna el nuemro de la fila actual en el dwcontrol. Retorna 0 sila fila no es la actual y -1 si ha ocurrido un error. Si dwcontrol es Nulo, GetRow returna NULL.

Ejemplo:
int li_fila
li_fila = dw_profesor.GetRow()

SetRow()

Descripción
Coloca la fila actual en el datawindow determinado por el valor de fila determinada.

Sintaxis
dwcontrol.SetRow(row)

Argumento	Description
dwcontrol<>	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere colocar la fila actual

Valor que retorna
Un integer. Retorna 1 si ha sicedido correctamente y -1 si ha ocurrido un error. Si la fila es menor que 1 o mayor que el numero de filas , SetRow falla. Si el valor del argumento es NULL, SetRow retorna NULL.

Uso
SetRow mueve el cursor a la fila actual pero no se desplaza por el control DataWindow o DataStore.
Eventos en los cuales SetRow podría dispararse:

- ItemChanged
- ItemError
- ItemFocusChanged
- RowFocusChanged

Ejemplo:
dw_profesor.SetRow(1), se ubica en la primera fila.

SetColumn()

Descripcion
Coloca la columna actual en el datawindow o dataStore.

Sintaxis
dwcontrol.SetColumn(column)

Argumento	Description
dwcontrol	El nombre del control DataWindow ,DataStore, o child DataWindow en el cual ud. quiere colocar la fila actual
column	La columna que ud quiere harcerla actual. Column puede ser un número de columna(integer) o el nombre de la columna(string)

Valor que retorna
Un integer. Retorna 1 si ha sicedido correctamente y -1 si ha ocurrido un error. Si la fila es menor que 1 o mayor que el numero de filas , SetColumn falla. Si el valor del argumento es NULL, SetColumn retorna NULL.

Uso
SetColumn mueve el cursor a la columna actual pero no se desplaza por el control DataWindow o DataStore.
Solamente una columna editable puede ser actualizada.(Una columna es editable cuando el valor del tabulador(Order Tab) es mayor que 0.
Eventos en los cuales SetColumn podría dispararse:

- ItemChanged
- ItemError
- ItemFocusChanged

Ejemplo :
dw_profesor.SetRow("pro_codigo"), se ubica en la columna pro_codigo.

SetPointer()

Descripción
Coloca el puntero del mouse de la forma especificada.

Syntax
SetPointer (type)

Argumento	Description
type	Un valor del Puntero enumerado según el tipo de dato que

	indica el tipo de puntero que usted desea. Los vaores son: ○ Arrow! ○ Cross! ○ Beam! ○ HourGlass! ○ SizeNS! ○ SizeNESW! ○ SizeWE! ○ SizeNWSE! ○ UpArrow!

Valor que retorna
Puntero(Pointer). Retorna el tipo enumerado del puntero reemplazando para que en el script pueda establecerse, si es necesario. Si el tipo es NULL, SetPointer retorna NULL.

Uso
Se usa SetPointer para desplegar un Reloj de tiempo(hourglass) al inicio de un script cuando el script toma un tiempo largo en ejecutarse.
type Arrow!(flecha) , HourGlass!(reloj) .etc..

Ejemplo : SetPointer(Arrow!)

***P*owerBuilder 6.0 es una parte integral de la Familia de herramientas de Sybase Powersoft, esta herramienta hace posible el desarrollo distribuido de aplicaciones, además de aplicaciones basadas en componentes para las nuevas operaciones de las organizaciones.**

Con la versión 6.0 PowerBuilder satisface las necesidades de los desarrolladores que deseaban crear aplicaciones completas o componentes de aplicación en un medio ambiente 4GL. En esta nueva versión, PowerBuilder promueve el liderazgo en la industria del medio ambiente de 4GL de diferentes formas importantes y relevantes.

PowerBuilder 6.0 esta construido con una filosofía que cumpla con

los retos que viven actualmente las organizaciones, como lo es el incremento en la productividad y en la calidad de los servicios que se desarrollan, y que al mismo tiempo le permitan ser competitivo a nivel mundial, las nuevas características de PowerBuilder que permitirán llegar a las nuevas metas generadas en las organizaciones se pueden dividir en tres fundamentales:

NUEVAS HERRAMIENTAS DE PRODUCTIVIDAD

Procesamiento Asíncrono de aplicaciones

El procesamiento asíncrono provee de un mecanismo para listar las peticiones del procesamiento de una aplicación, y permite que el cliente cancele el procesamiento de estas peticiones antes que este sea terminado por el servidor.

Aplicaciones con Objetos compartidos

Los objetos compartidos permiten que los clientes corran sobre una aplicación en el servidor para compartir instantáneamente los objetos sobre ese servidor. Los nombres de las instancias de objetos pueden ser registrados como objetos compartidos.

Direccionamiento Lógico de Servidores

La utilería para nombrar servidores es un mecanismo de redireccionamiento lógico, esto se hace entre el nombre lógico del servidor y el servidor físico. Esta utilería es además usada para balancear las cargas de los clientes conectados y provee información de las conexiones de aplicaciones distribuidas de PowerBuilder.

Preparado para el Web.

Debido a las necesidades de los desarrolladores de crear aplicaciones basadas en el Web, PowerBuilder 6.0 ha agregado las siguientes características:

- **Integración de las herramientas para el desarrollo de INTERNET (Internet Developer Tools, previamente conocidas como Internet Developer Toolkit) en PowerBuilder Enterprise (PBE).**
- **Todos los componentes previamente empaquetados como Internet Developer Toolkit son incluidos y accesibles solamente para desarrolladores que utilizan PowerBuilder**

Enterprise. Dichos componentes incluyen:

***Web.PB DLL's para CGI, ISAPI e NSAPI Servidores Web.**
*** Librerías de Clases Web.Pb para la generación de HTML y el manejo de la situación de conexión con un navegador.**
*** Web.Pb Wizard, utilizado para crear los elementos de HTML requeridos para invocar los servicios de objetos distribuidos.**
*** PowerBuilder Window Plug-in DLL, en versión para modo estándar y seguro. Nuevo Window ActiveX, en versión para modo estándar y seguro, igual como el Plug-in DLL, con la diferencia de que este permite la invocación de métodos.**
*** PowerBuilder DataWindow Plug-in DLL, solamente en versión estándar. Software O'Reilly WebSite como servidor web.**
***Reestructuración del PowerBuilder Deployment DLLs en PowerBuilder VM (Virtual Machine).**

Window Plug-in/ActiveX modo seguro.

El modo seguro provee DLLs alternos para el PowerBuilder Window Plug-in y ActiveX. Esta versión evita que el Plug-in/ActiveX pueda realizar cambios en las maquinas de los usuarios.

Productividad en el Desarrollo. En su versión 6.0, PowerBuilder continua agregando nuevas capacidades funcionales, fáciles de usar que permiten incrementar la productividad en el desarrollo. Un nuevo Depurador de Errores ("Debugger".) El nuevo debugger de PowerBuilder 6.0 tiene nuevas opciones y funcionalidad para: v Condiciones para puntos de interrupción (breakpoints). v Llamadas a pilas. (Call Stack) v Objetos en memoria. v Código Fuente. v Fuente del Navegador (Browser). v Historia de fuentes. v Variables. v Ver variables (Watch variables). Una nueva interfase del debugger que permite ver diferente tipo de información simultáneamente. Depuración Justo a Tiempo (Debugging just in time) - Los desarrolladores pueden hacer uso del debugger durante el tiempo de corrida. Perfilar y Trazar una aplicación. El trazado y perfilado de una aplicación permite a los desarrolladores de PowerBuilder 6.0 colectar, trazar y analizar información referente a la ejecución de la aplicación. El trazado y perfilado incluye la colección de datos, análisis y un despliegue funcional. Además, nuevos objetos de

PowerBuilder y funciones de PowerScript permiten a los desarrolladores crear rutinas customatizadas. Barra de Herramientas. PowerBuilder 6.0 agregó nuevos botones a la barra de herramientas para generar entradas al registro de Windows y para invocar la herramienta de sincronización. PowerBuilder 6.0 implementa la barra de herramientas planas como el estilo del nuevo Microsoft Office97. Mejoras en los DataWindows. Objetos de botones son soportados en los DataWindows. Los desarrolladores puedes asociar acciones predefinidas o customatizadas con botones. Cuando el usuario presione un botón de un DataWindow, PowerBuilder dispara eventos al control del DataWindow antes y después de que la acción asociada sea tomada. Además, nuevos objetos

PRODUCTIVIDAD EN EL DESARROLLO

En su versión 6.0, PowerBuilder continua agregando nuevas capacidades funcionales, fáciles de usar que permiten incrementar la productividad en el desarrollo.

Un nuevo Depurador de Errores ("Debugger".)

El nuevo debugger de PowerBuilder 6.0 tiene nuevas opciones y funcionalidad para: v Condiciones para puntos de interrupción (breakpoints).

- **Llamadas a pilas. (Call Stack)**
- **Objetos en memoria. v Código Fuente.**
- **Fuente del Navegador (Browser). v Historia de fuentes.**
- **Variables.**
- **Ver variables (Watch variables).**

Una nueva interfase del debugger que permite ver diferente tipo de información simultáneamente.

Depuración Justo a Tiempo (Debugging just in time) - Los desarrolladores pueden hacer uso del debugger durante el tiempo de corrida.

Perfilar y Trazar una aplicación.

El trazado y perfilado de una aplicación permite a los desarrolladores de PowerBuilder 6.0 colectar, trazar y analizar información referente a la ejecución de la aplicación. El trazado y perfilado incluye la colección de datos, análisis y un despliegue

funcional. Además, nuevos objetos de PowerBuilder y funciones de PowerScript permiten a los desarrolladores crear rutinas customatizadas.

Barra de Herramientas.

PowerBuilder 6.0 agregó nuevos botones a la barra de herramientas para generar entradas al registro de Windows y para invocar la herramienta de sincronización. PowerBuilder 6.0 implementa la barra de herramientas planas como el estilo del nuevo Microsoft Office97.

Mejoras en los DataWindows.

Objetos de botones son soportados en los DataWindows. Los desarrolladores puedes asociar acciones predefinidas o customatizadas con botones. Cuando el usuario presione un botón de un DataWindow, PowerBuilder dispara eventos al control del DataWindow antes y después de que la acción asociada sea tomada. Además, nuevos objetos tipo Group Box pueden ser usados para marcar y etiquetar un grupo de objetos en una DataWindow.

DataStore remoto.

DataWindows y DataStores del lado del cliente pueden ahora intercambiar información desde un objeto tipo DataStore que esta del lado del Servidor.

Arquitectura Abierta

PowerBuilder 6.0 continua incrementando su apertura hacia nuevas tecnologías y esto incluye el soporte de nuevas plataformas, nuevas interfaces de bases de datos, la adición de nuevos lenguajes y nuevas tecnologías surgidas de los servidores de datos. Algunas de las características mas relevantes son:

- **Soporta la Plataforma UNIX**

  Se amplia el soporte de plataformas UNIX incluyendo ahora IBM AIX y HP-UX ampliando de esta forma las opciones de desarrollo para las organizaciones y los desarrolladores de PowerBuilder

- **Soporte abierto para modelos de componentes estándares**

  PowerBuilder 6.0 esta construido para soportar la creación futura de múltiples componentes estándares, incluyendo componentes Activex, componentes CORBA, clases C++, y JavaBeans

- **Soporte abierto a servidores de transacciones**

  PowerBuilder actúa tanto como un productor o como consumidor de objetos para servidores de transacciones múltiples, incluyendo ahora para esto el producto Sybase Jaguar CTS y el Microsoft Transaction Server (MTS)

- **Amplia Conectividad con Base de datos**

  PowerBuilder ahora soporta la conexión nativa con Informix, nuevo soporte para Sybase SQL 11.1 incluyendo el soporte para el OpenClient 11.1 security e incrementa el procesamiento en UNIX y Macintosh, así mismo provee el soporte para ODBC 3.0 expande significativamente el soporte de lenguajes, tales como el árabe y el hebreo.

- **Ampliación en el soporte de dispositivos**

  PowerBuilder 6.0 soporta el uso del dispositivo IntelliMouse. Los usuarios del IntelliMouse pueden aprovechar las características especiales de este dispositivo para incrementar los movimientos en los DataWindow, y de la navegación a través de cualquier ventana de control tal como la vista de listas.

Con las nuevas características anteriormente descritas podemos concluir que PowerBuilder 6.0 integra herramientas que le permiten continuar siendo el líder en el soporte de tecnología orientada a incrementar la productividad en el desarrollo de aplicaciones necesarias para que una organización se encuentre preparada para competir a nivel mundial.

http://www.baja.gob.mx/organizacion/dgi/biblioteca/ci/ci10/art_5.htm

****
****