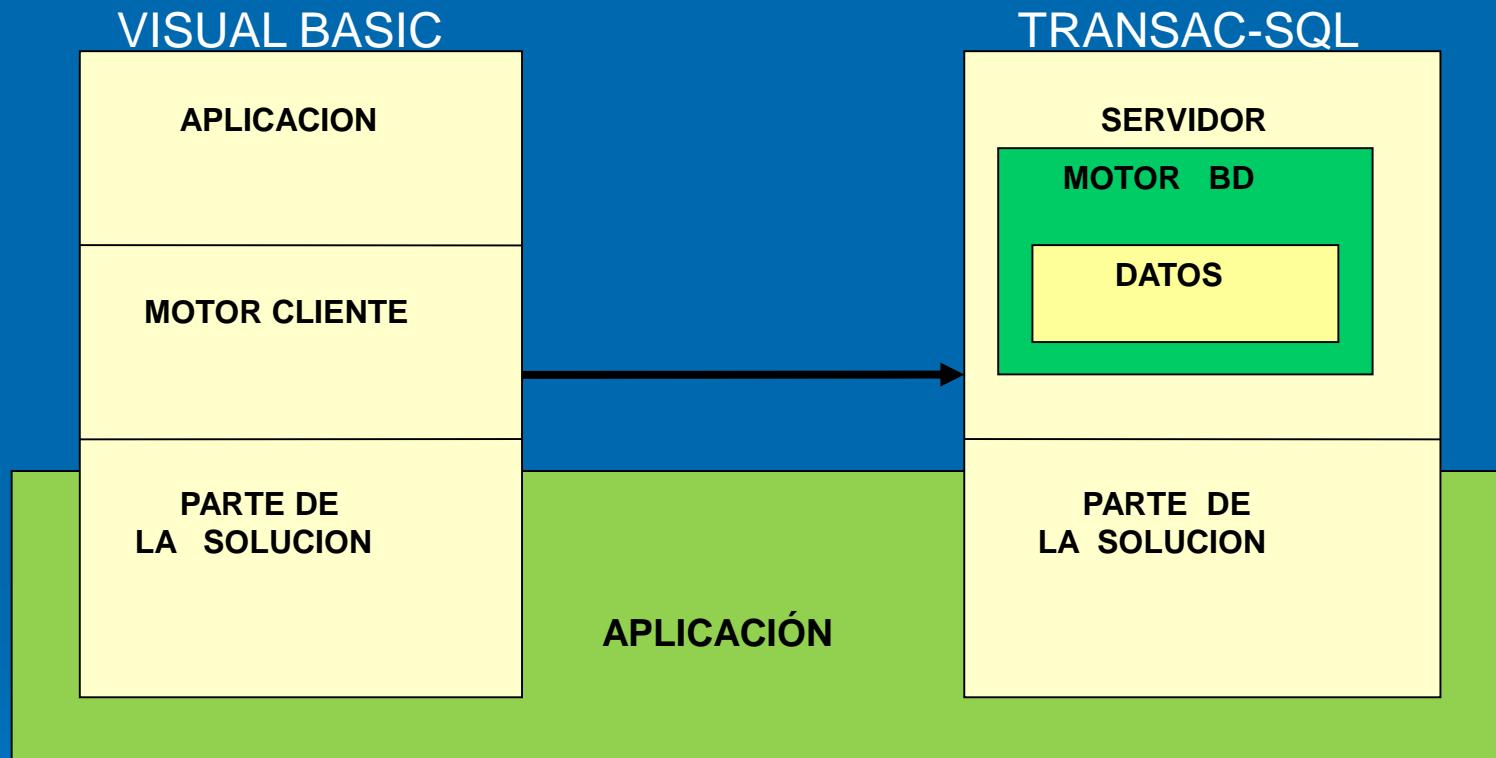


Cap.V Aplicaciones Cliente/Servidor



Busca la separación del código para aislar dominios de la solución diferentes.

Aplicaciones Cliente/Servidor

- Por un lado está el servidor de la BD, responsable de garantizar la integridad de los datos y proveer servicios para las operaciones con ellos.
- Por el otro lado está la aplicación cliente, responsable de darle al usuario las herramientas necesarias para enviar y recibir información.

Aplicaciones Cliente/Servidor

Ventajas

- La carga se trabajo se distribuye entre el servidor y el cliente.
- Se pueden centralizar controles y operaciones que pueden ser aprovechadas por el cliente.
- Se disminuye el tráfico en la red, porque sólo viajan las peticiones y los resultados.
- Se esconde la complejidad del código.

Desventajas :

- ❖ se requiere conocer VB y el servidor de BD.
- ❖ Como se construye parte de la aplicación en el servidor, ésta es dependiente de ese producto.
- ❖ La aplicación VB tiene código solapado con el código de peticiones de ejecución de servicios del servidor.

Programar al Cliente

- Se compone de interfaces gráficas, manejo de eventos, configuración de propiedades, manejo de controles, peticiones de ejecución al servidor de BD entre otros.

- Puede requerir en ocasiones usar SQL dinámico o con parámetros para resolver un aspecto específico de la aplicación.

Aspectos de la Conexión

Existen dos alternativas :

- **Conexión por Sesión** : se establece en el instante en que el usuario abre la aplicación y se mantiene permanentemente.

Características:

- Mejor tiempo de respuesta, considerando que al ejecutar la petición la conexión está abierta.
- **Conexión por Acción** : se establece la conexión en el instante en que se hace una petición al servidor y luego cerrarla inmediatamente después de finalizar la ejecución.
- Supone más demora porque se tiene que establecer la conexión cada vez que se ejecuta una petición.

Programación del Servidor

Un servidor de BD es un conjunto de programas que permiten definir estructuras de datos y su almacenamiento, procesar esos datos, transformarlos, transferirlas y administrarlos.

- El lenguaje se identifica con el nombre de Transact-SQL y se compone de operadores , funciones , estructuras de control, etc.

Programación de Procedimientos Almacenados

Representa la interfase de comunicación entre la aplicación y el servidor de la BD.

- Cada PA en el servidor, proporciona servicios para satisfacer necesidades de la aplicación.
- Es un programa Transac-SQL que se almacena en el servidor.
- Pueden ser de acción o de recuperación.

Programación de Procedimientos Almacenados

Ofrecen la posibilidad de realizar operaciones como :

1. Realizar cálculos con datos
2. Tomar o devolver parámetros
3. Ordenar datos
4. Devolver datos de un modo más sencillo y eficiente.

CREACION

Los PA se crean mediante la instrucción CREATE PROCEDURE de Transact-SQL.

Formato:

```
CREATE PROCEDURE nombreProc  
[ lista de parámetros ]  
As Bloque de Instrucciones
```

➤ Ejemplo:

```
CREATE PROCEDURE ARTXCATEGORIA  
@cantidadporunidad varchar (50)  
AS  
SELECT * FROM Producto WHERE cantidadop like  
@cantidadporunidad  
return
```

Programación de Procedimientos Almacenados

Ejecución

Para ejecutar un PA se utiliza el **EXECUTE** de Transac-SQL .

Difieren de las funciones en :

- no devuelven valores
- no se pueden usar directamente en la expresión

Programación de Procedimientos Almacenados

Tipos de Procedimientos Almacenados

➤ Procedimientos de Acción

Son los que realizan actualizaciones sobre los datos de la BD. Para adicionar, modificar o eliminar uno o varios registros.

- Ejemplo:

CREATE PROCEDURE CateAcciónl

@IDCategoria int,

@Descripción varchar(100)

AS

Insert INTO Categoría (IDCategoria, Descripción)

Values (@IDCategoria, @Descripción)

Procedimientos de Recuperación

- Devuelven resultados a la aplicación cliente. Cada aplicación puede devolver un valor de retorno (valor entero), parámetros de salida o un conjunto de resultados (con el Select).

Ejemplo :

```
CREATE PROCEDURE ArtiBuscarFK  
    @IDCategoria int  
AS  
Select * from Articulo  
Where IDCategoria = @IDCategoria
```

Especificación de Parámetros

Un PA se enlaza con el programa que lo llama usando parámetros.

- Cuando un programa ejecuta un PA, a éste se le pueden pasar valores mediante los parámetros del procedimiento .
- Todo parámetro tiene un nombre, tipo de datos, dirección y valor
- Cada parámetro debe tener un nombre único y empezar con el carácter @ (se consideran variables locales).

Especificar la dirección del parámetro

Todos los parámetros de un procedimiento almacenado pueden recibir datos de entrada cuando el PA es ejecutado por el programa que lo llama.

Ejemplo :

Create procedure CateAccionl

@IDCategoria int, -- parametro de entrada

@Descripción varchar(100) -- parámetro de entrada

Especificar la dirección del parámetro

- Se pueden devolver valores desde el PA, si se especifica un parámetro como salida (output).

Ejemplo :

```
Create procedure CateAccionI
```

```
@IDCategoria int,
```

```
@Flag varchar(100) output
```

Especificar la dirección del parámetro

Devolver datos

Un PA puede devolver datos de tres maneras:

- **Parámetros de Salida:** declara un parámetro output.
- **Valores de retorno :** devuelve un entero y se envía con la instrucción Return.
- **Conjunto de resultados :** se devuelve un conjunto de resultados por cada SELECT que esté en el procedimiento.

Pasos para definir Procedimientos almacenados en un programa:

1. Establecer la propiedad CommandType del comando a CommandType.StoredProcedure.
2. Cargar en la propiedad Text el nombre del Procedimiento Almacenado
3. Usar la colección Parameters del comando para indicar los parámetros que se le pasarán al procedimiento.

Ejemplo (Aplicación Cliente/servidor) :

Procedimiento Almacenado en el Servidor

```
/* Se crea un PA de recuperación          BUSCARSHIPPER */
create procedure sp_BuscarShipper
@shipperid int
as
    Select * from shippers
    Where shipperID = @shipperid
```

Aplicación Visual Basic en el Cliente

```
Call conexion()
cadena = "sp_BuscarShipper"
cmd = New SqlCommand(cadena, cn)
cmd.CommandType = CommandType.StoredProcedure
valor = Val(InputBox("Teclee un código"))
cmd.Parameters.AddWithValue("@shipperid", valor)
dr = cmd.ExecuteReader
dr.Read()
TextBox1.Text = dr.Item("CompanyName")
dr.Close()
```