



Las vistas

Una vista es un objeto de la base de datos que almacena una consulta predefinida. Las vistas permiten que el foco del usuario se concentre en la data que es relevante para él, ocultándole la complejidad del modelo de datos. También simplifican las consultas complejas, y la administración de los permisos del usuario.

Esta página se ha dejado en blanco intencionalmente.

Capítulo 12

Las vistas

Contenido

- ❑ *Vistas*
 - ✓ **Ejercicio 118:** *Ejemplo de vista*
 - ✓ *Ventajas de las vistas*
- ❑ *Creación de vistas – La instrucción CREATE VIEW*
 - ✓ *Consideraciones al crear las vistas*
 - ✓ **Ejercicio 119:** *Balance entrada/salida en el almacén de MarketPERU*
 - ✓ *Obtención de la definición de una vista*
 - ✓ **Ejercicio 120:** *Uso de sp_helptext y sp_depends*
- ❑ *Modificación y eliminación de una vista*
 - ✓ *Modificación de una vista – La instrucción ALTER VIEW*
 - ✓ *Eliminación de una vista – La instrucción DROP VIEW*
 - ✓ *Ocultando la definición de una vista*
 - ✓ **Ejercicio 121:** *Uso de WITH ENCRYPTION*
- ❑ *Modificación de datos a través de vistas*
 - ✓ **Ejercicio 122:** *Uso de WITH CHECK OPTION*
- ❑ *Vistas indexadas*
 - ✓ *Combinación de vistas indexadas con consultas*

Esta página se ha dejado en blanco intencionalmente.

Las vistas

En ocasiones, una consulta compleja se puede simplificar si previamente se diseñan vistas que contienen consultas parciales que combinadas permiten llegar al resultado final requerido. Adicionalmente, las vistas mejoran el rendimiento de las consultas ya que la instrucción SELECT asociada se guarda compilada y con su plan de ejecución ya definido.

Vistas

Una es un objeto que almacena una consulta predefinida y que proporciona un modo alternativo de visualización de datos sin tener que redefinir la consulta. Las tablas requeridas en una vista se llaman tablas base. Con algunas excepciones, cualquier declaración SELECT puede nombrarse y guardarse como una vista. Los ejemplos comunes de vistas incluyen:

- Un subconjunto de filas o columnas de una tabla base.
- Una unión de dos o mas tablas base.
- Un join de dos o mas tablas base.
- Un resumen estadístico de una tabla base.
- Un subconjunto de otra vista, o alguna combinación de vistas y tablas base.

Ejercicio 118: Ejemplo de vista

El ejemplo siguiente crea la vista **v_ListaPrecios** en la base de datos **MarketPERU**.

```
USE MarketPERU
go
```

```
CREATE VIEW v_ListaPrecios
AS
SELECT Categoria.categoria, Producto.idProducto,
       Producto.nombre, Producto.unidadMedida,
       Producto.precioProveedor
FROM Producto INNER JOIN Categoria
       ON Producto.idCategoria = Categoria.idCategoria
go

SELECT * FROM v_ListaPrecios
go
```

SQLQuery1.sql...5.MarketPERU* Summary					
	categoria	idProducto	nombre	unidadMedida	precioProveedor
1	GOLOSINAS	1	CARAMELOS BASTON...	PAQUETE 454 GR	1.50
2	GOLOSINAS	2	CARAMELOS SURTIDO...	PAQUETE 450 GR	1.00
3	GOLOSINAS	3	CARAMELOS FRUTAS ...	PAQUETE 520 GR	1.50
4	GOLOSINAS	4	CARAMELOS FRUTAS ...	PAQUETE 454 GR	1.30
5	GOLOSINAS	5	CHUPETES LOLY AMB...	KILOGRAMO	1.20
6	GOLOSINAS	6	FRUNA SURTIDA DON...	PAQUETE X 24 UN...	1.80
7	GOLOSINAS	7	CHOCOLATE DOÑA PE...	PAQUETE X 6 UNL...	2.20
8	GOLOSINAS	8	CHOCOLATE CUA CUA...	PAQUETE X 6 UNL...	1.60
9	GOLOSINAS	9	MELLOWS FAMILIAR FI...	PAQUETE 454 GR	2.10
10	GOLOSINAS	10	WAFER CHOCOLATE FI...	PAQUETE X 9 UNL...	0.70
11	GOLOSINAS	11	CHOCOLATE BARRA R...	BARRA 2 ONZAS	0.40
12	GOLOSINAS	12	CHOCOLATE MOSTRO...	PAQUETE X 6 UNL...	1.50
13	GOLOSINAS	13	CHOCOLATE BARRA M...	BARRA 2.15 ONZAS	0.80
14	GOLOSINAS	14	SNICKERS BAR KING S...	BARRA 3.7 ONZAS	1.20
15	GOLOSINAS	15	CHOCOLATE BARRA M...	UNIDAD	1.30
16	GOLOSINAS	16	CHOCOLATE BARRA D...	UNIDAD	1.30
17	GOLOSINAS	17	MILKY WAY BAR KING...	UNIDAD 3.6 ONZAS	5.00

Observe que para ejecutar la vista se utiliza la instrucción SELECT. Una vista se manipula como si fuera una tabla; es decir, que se le puede aplicar las declaraciones SELECT, INSERT, UPDATE y DELETE. Sin embargo, no debe ver la vista como una tabla, sino pensar en ella como un programa que ejecuta una sola instrucción (la instrucción SELECT).

Ventajas de las vistas

El uso de las vistas proporciona las siguientes ventajas:

- El usuario accede a la data importante o apropiada para él. Limita el acceso a datos sensibles.
- Oculta la complejidad del modelo de datos. Un join de múltiples tablas se convierte en un simple SELECT para el usuario.
- Desde el punto de vista del usuario, una vista es una "tabla" pues puede ejecutar en ella todas las operaciones de datos: SELECT, INSERT, UPDATE y DELETE.
- Debido a que una vista es un objeto de la base de datos, puede asignarle permisos de usuario. Esto es mucho mas eficiente que colocar los mismos permisos sobre columnas individuales en una tabla.
- Los datos pueden exportarse desde una vista por medio de la utilidad **bcp**.

Creación de vistas – La instrucción CREATE VIEW

Sintaxis

```
CREATE VIEW nombre_vista [ ( lista_columnas ) ]  
[ WITH ENCRYPTION ]  
AS  
sentencia_select  
[ WITH CHECK OPTION ]
```

- WITH ENCRYPTION indica que se debe encriptar la sentencia con la que se define la vista, de modo tal que cuando se utiliza el procedimiento **sp_helptext** no se pueda observar la instrucción que creó la vista.
- WITH CHECK OPTION indica que si la definición de la vista contiene la cláusula WHERE, y la vista se utiliza para operaciones de mantenimiento de datos, estas operaciones deben respetar la condición definida en el WHERE.

Las siguientes restricciones se aplican a la creación de vistas:

- No se puede definir una vista con ORDER BY, COMPUTE, COMPUTE BY o SELECT INTO.
- No pueden hacer referencia a tablas temporales.
- No pueden hacer referencia a mas de 1024 columnas.
- La sentencia CREATE VIEW no puede combinarse con otras sentencias Transact-SQL en el mismo batch.

Consideraciones al crear las vistas

Tenga en cuenta lo siguiente al momento de crear una vista:

- Para ejecutar la declaración CREATE VIEW, debe ser miembro de alguno de los siguientes roles: **sysadmin**, **db_owner**, **db_ddladmin**, o debe tener el permiso CREATE VIEW. Además, también debe tener el permiso SELECT en todas las tablas o vistas que son referenciadas dentro de la vista.
- Para evitar situaciones en las que el dueño de una vista y el dueño de las tablas subyacentes son distintos, se recomienda que el usuario **dbo** tenga todos los objetos de una base de datos.
- Si la declaración SELECT de la vista contiene columnas computadas o constantes, éstas deben tener un nombre asignado.

Recomendación:

Antes de crear la vista, es importante probar la declaración SELECT que define la vista para asegurar que el servidor devuelve el resultado esperado.

Ejercicio 119: Balance entrada/salida en el almacén de MarketPERU

Se desea obtener un reporte que muestre el balance entrada/salida de todos los productos registrados en la base de datos **MarketPERU**. El reporte debe mostrar para cada producto: el código y el nombre del producto, la cantidad de unidades entrantes en el almacén, y la cantidad de unidades salientes del almacén.

Este problema se resuelve fácilmente si crea una vista para las entradas, y una segunda vista para las salidas. Luego, debe combinar las vistas anteriores para obtener el reporte final.

Creación de la vista para el reporte de Entradas

```
USE MarketPERU
go

-- Vista para el listado de entradas
CREATE VIEW v_UnidadesEntrantes
AS
SELECT Producto.idProducto, Producto.nombre,
       SUM( Orden_detalle.cantidadRecibida ) AS Entradas
FROM Producto LEFT OUTER JOIN Orden_detalle
  ON Producto.idProducto = Orden_detalle.idProducto
GROUP BY Producto.idProducto, Producto.nombre
go

SELECT * FROM v_UnidadesEntrantes
go
```

SQLQuery1.sql...5.MarketPERU* Summary			
	idProducto	nombre	Entradas
1	1	CARAMELOS BASTON VIENA ARCOR	NULL
2	2	CARAMELOS SURTIDO DE FRUTAS	1500
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	1250
4	4	CARAMELOS FRUTAS MASTICABLES	3000
5	5	CHUPETES LOLY AMBROSOLI	3000
6	6	FRUNA SURTIDA DONOFRIO	3000
7	7	CHOCOLATE DOÑA PEPA FIELD	3000

Creación de la vista para el reporte de Salidas

```
-- Vista para el listado de salidas
CREATE VIEW v_UnidadesSalientes
AS
SELECT Producto.idProducto, Producto.nombre,
       SUM( Guia_detalle.cantidad ) AS Salidas
FROM Producto LEFT OUTER JOIN Guia_detalle
  ON Producto.idProducto = Guia_detalle.idProducto
GROUP BY Producto.idProducto, Producto.nombre
go
```

```

SELECT * FROM v_UnidadesSalientes
ORDER BY idProducto
go

```

SQLQuery1.sql...5.MarketPERU* Summary			
	idProducto	nombre	Salidas
1	1	CARAMELOS BASTON VIENA ARCOR	400
2	2	CARAMELOS SURTIDO DE FRUTAS	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	400
4	4	CARAMELOS FRUTAS MASTICABLES	400
5	5	CHUPETES LOLY AMBROSOLI	NULL
6	6	FRUNA SURTIDA DONOFRIO	600
7	7	CHOCOLATE DOÑA PEPA FIELD	500

Combinación de las vistas para obtener el reporte final

```

-- Combinación de las vistas para el balance
SELECT v_UnidadesEntrantes.idProducto,
       v_UnidadesEntrantes.nombre,
       v_UnidadesEntrantes.entradas,
       v_UnidadesSalientes.salidas
FROM v_UnidadesEntrantes INNER JOIN v_UnidadesSalientes
ON v_UnidadesEntrantes.idProducto =
   v_UnidadesSalientes.idProducto
ORDER BY v_UnidadesEntrantes.idProducto
go

```

SQLQuery1.sql...5.MarketPERU* Summary				
	idProducto	nombre	entradas	salidas
1	1	CARAMELOS BASTON VIENA ARCOR	NULL	400
2	2	CARAMELOS SURTIDO DE FRUTAS	1500	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	1250	400
4	4	CARAMELOS FRUTAS MASTICABLES	3000	400
5	5	CHUPETES LOLY AMBROSOLI	3000	NULL
6	6	FRUNA SURTIDA DONOFRIO	3000	600
7	7	CHOCOLATE DOÑA PEPA FIELD	3000	500

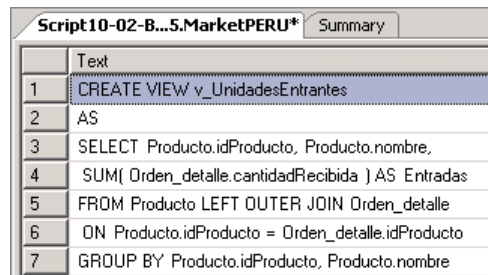
Obtención de la definición de una vista

Para averiguar cómo está definida una vista, puede utilizar los siguientes procedimientos almacenados del sistema para consultar las siguientes tablas del catálogo de una base de datos:

Vista del sistema	Almacena	Procedimiento
sys.sysobjects	Nombre de la vista	<code>sp_help nombre_vista</code>
sys.sysdepends	Nombre de los objetos dependientes de la vista	<code>sp_depends nombre_vista</code>
sys.syscomments	Sentencia que definió la vista	<code>sp_helptext nombre_vista</code>
sys.syscolumns	Columnas definidas en la vista	<code>sp_columns nombre_vista</code>

Ejercicio 120: Uso de sp_helptext y sp_depends

```
sp_helptext v_UnidadesEntrantes
go
```



The screenshot shows a window titled 'Script10-02-B...5.MarketPERU*' with a 'Summary' tab. Below the tab is a table with two columns: 'Text' and a line number column. The table contains the following text:

	Text
1	CREATE VIEW v_UnidadesEntrantes
2	AS
3	SELECT Producto.idProducto, Producto.nombre,
4	SUM(Orden_detalle.cantidadRecibida) AS Entradas
5	FROM Producto LEFT OUTER JOIN Orden_detalle
6	ON Producto.idProducto = Orden_detalle.idProducto
7	GROUP BY Producto.idProducto, Producto.nombre

Muestra la instrucción con la que se creó la vista **v_UnidadesEntrantes**.

```
sp_depends v_UnidadesEntrantes
go
```

Script10-02-B...5.MarketPERU*					
Summary					
	name	type	updated	selected	column
1	dbo.ORDEN_DETALLE	user table	no	no	IdProducto
2	dbo.ORDEN_DETALLE	user table	no	no	CantidadRecibida
3	dbo.PRODUCTO	user table	no	no	IdProducto
4	dbo.PRODUCTO	user table	no	no	Nombre

Muestra los objetos que dependen de la vista **v_UnidadesEntrantes**.

Modificación y eliminación de una vista

A menudo, en respuesta a las demandas de los usuarios por información adicional, o a los cambios en la definición de la tabla subyacente, es necesario modificar la definición de una vista. Por ejemplo, si la tabla a la que una vista hace referencia se ha eliminado, los usuarios recibirán un mensaje del error cuando ellos intenten utilizar la vista. Se puede modificar una vista eliminándola y luego recreándola, o ejecutando la sentencia ALTER VIEW.

Modificación de una vista – La instrucción ALTER VIEW

La sentencia ALTER VIEW cambia la definición de una vista permitiéndole retener los permisos para la vista. Esta sentencia está sujeta a las mismas restricciones que la sentencia CREATE VIEW. Si en vez de modificar la vista, la elimina y luego la recrea, se verá obligado a recrear los permisos.

Sintaxis

```
ALTER VIEW nombre_vista [ ( lista_columnas ) ]  
[ WITH ENCRYPTION ]  
AS  
nueva_sentencia_select  
[ WITH CHECK OPTION ]
```

Eliminación de una vista – La instrucción DROP VIEW

La siguiente sentencia se utiliza para eliminar una vista.

Sintaxis

```
DROP VIEW nombre_vista
```

Ocultando la definición de una vista

El texto de la sentencia que define una vista se almacena en la tabla de sistema **syscomments**. Si al crear la vista utiliza la opción **WITH ENCRYPTION**, el texto se almacenará en dicha tabla en forma encriptada.

Recomendación

Antes de encriptar una vista, asegúrese que la definición de vista (el script) se guarde en un archivo. Para desencriptar el texto de una vista, debe eliminar y luego recrear o modificar la vista utilizando la sintaxis original.

Ejercicio 121: Uso de WITH ENCRYPTION

La siguiente instrucción crea la vista **v_Inventario** cuya definición se guarda encriptada.

```
CREATE VIEW v_Inventario
    WITH ENCRYPTION
AS
SELECT Categoria.categoria, Producto.idProducto,
    Producto.nombre, Producto.unidadMedida,
    Producto.stockActual, Producto.stockMinimo
FROM Categoria INNER JOIN Producto
    ON Categoria.idCategoria = Producto.idCategoria
go

-- Revisando la definición de la vista
sp_helptext v_Inventario
go
```

Se obtiene el siguiente mensaje del sistema:

```
The object comments have been encrypted.
```

Modificación de datos a través de vistas

Desde el punto de vista del usuario, una vista es una "tabla" ya que él puede ejecutar sobre la vista las sentencias SELECT, INSERT, UPDATE y DELETE. Estas sentencias ejecutadas sobre la vista afectan a las tablas dependientes de ella.

Deben tenerse en cuenta las siguientes consideraciones al momento de crear las vistas:

- La modificación no puede afectar a mas de una de las tablas dependientes.
- Las vistas con columnas computadas deben ser de solo lectura ya que producen error cuando se trata de ejecutar modificaciones a través de ellas.
- Las columnas no nulas que no son referenciadas en la vista pueden producir error cuando se ejecuta una modificación a través de la vista.
- Si el SELECT asociado a la vista ha sido definido con una cláusula WHERE, el uso de la opción WITH CHECK OPTION al momento de crear la vista hará que las modificaciones que se hagan a través de ella respeten el criterio del WHERE.

Ejercicio 122: Uso de WITH CHECK OPTION

A modo de ilustración del uso de WITH CHECK OPTION crearemos una vista que lista los productos de la categoría 2.

```
USE MarketPERU
go

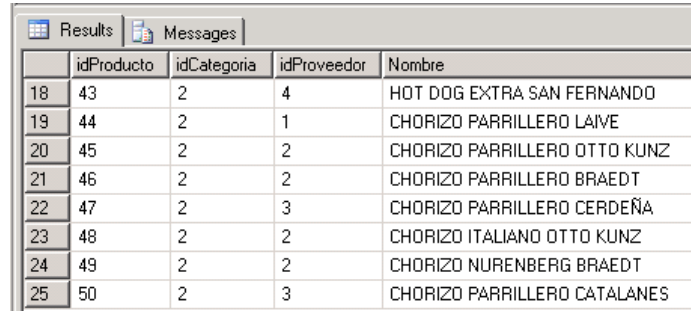
-- Vista que lista de productos de la categoría 2
CREATE VIEW v_MisProductos
AS
SELECT idProducto, idCategoria, idProveedor, Nombre
FROM Producto
WHERE idCategoria = 2
go

SELECT * FROM v_MisProductos
go
```


Inserción de un producto a través de la vista v_MisProductos

```
-- Insertando un producto a través de la vista
INSERT INTO v_MisProductos
    VALUES(1, 14, 'Chocolate Bitter Delicioso')
go

-- Consultando la vista
SELECT * FROM v_MisProductos
go
```



	idProducto	idCategoria	idProveedor	Nombre
18	43	2	4	HOT DOG EXTRA SAN FERNANDO
19	44	2	1	CHORIZO PARRILLERO LAIVE
20	45	2	2	CHORIZO PARRILLERO OTTO KUNZ
21	46	2	2	CHORIZO PARRILLERO BRAEDT
22	47	2	3	CHORIZO PARRILLERO CERDEÑA
23	48	2	2	CHORIZO ITALIANO OTTO KUNZ
24	49	2	2	CHORIZO NURENBERG BRAEDT
25	50	2	3	CHORIZO PARRILLERO CATALANES

Observe que la fila insertada no aparece en el resultado de la ejecución de la vista. Ello debido a que la vista filtra para los productos de la categoría 2.

Si esta vista se le ha entregado a un usuario para que pueda hacer mantenimiento de sus productos, el resultado lo puede confundir ya que él acaba de insertar una fila que no puede ver. Esto podría llevarlo a pensar que ha ocurrido un error, y que la fila no se ha insertado. Entonces, podría tratar de insertarla nuevamente repitiendo la orden de inserción.

Crearemos otra versión de la vista, pero que no permita hacer mantenimiento de los productos que no correspondan al filtro definido. Para ello, haremos uso de `WITH CHECK OPTION`.

Creación de una vista con WITH CHECK OPTION

```
-- Creando una vista con WITH CHECK OPTION
CREATE VIEW v_MisProductos_CHECK
AS
SELECT idProducto, idCategoria, idProveedor, Nombre
    FROM Producto
    WHERE idCategoria = 2
    WITH CHECK OPTION
go

-- Consultando la vista
SELECT * FROM v_MisProductos_CHECK
go
```

Inserción de un producto que viola el filtro de la vista

```
-- Insertando un producto que viola el filtro de la
-- última vista
INSERT INTO v_MisProductos_CHECK
    VALUES(1, 14, 'Chocolate Bitter c/almendras')
go
```

Se obtiene el siguiente mensaje del sistema:

```
The attempted insert or update failed because the target
view either specifies WITH CHECK OPTION or spans a view
that specifies WITH CHECK OPTION and one or more rows
resulting from the operation did not qualify under the
CHECK OPTION constraint.
The statement has been terminated.
```

Vistas indexadas

Desde la versión anterior del producto (SQL Server 2000) es posible crear vistas indexadas. Las vistas indexadas mejoran de forma considerable el rendimiento de algunos tipos de consultas.

El mejor rendimiento de las vistas indexadas se obtiene cuando se actualizan los datos subyacentes con frecuencia. El mantenimiento de una vista indexada puede ser superior al costo de mantenimiento de un índice de tabla. Si se actualizan frecuentemente los datos subyacentes, el costo de mantener la vista indexada puede no compensar las ventajas que se obtienen en el rendimiento al utilizarla.

Las vistas indexadas mejoran el rendimiento de los siguientes tipos de consultas:

- Las combinaciones y agregaciones que procesan muchas filas.
- Las operaciones de combinación y agregación realizadas frecuentemente por muchas consultas.
- Las cargas de trabajo de la ayuda a la toma de decisiones.
- Las vistas indexadas no suelen mejorar el rendimiento de este tipo de consultas:
- Los sistemas OLTP con muchas escrituras.
- Las bases de datos con muchas actualizaciones.
- Las consultas que no incluyen agregaciones ni combinaciones.
- Las agregaciones de datos con un alto grado de cardinalidad para la clave.
- Las combinaciones expandidas, que son vistas cuyos conjuntos de resultados son mayores que los datos originales de las tablas base.

Combinación de vistas indexadas con consultas

Aunque las restricciones de los tipos de vistas que pueden indexarse pueden impedir diseñar una vista que resuelva completamente un problema, cabe la posibilidad de diseñar varias vistas indexadas más pequeñas que aceleren partes del proceso.

Considere estos ejemplos:

Una consulta de ejecución frecuente que agrega datos a una primera base de datos, luego agrega datos a otra base de datos y, finalmente, combina los resultados. Puesto que una vista indexada no puede hacer referencia a tablas de más de una base de datos, no se puede diseñar una vista única para todo el proceso. Se puede, no obstante, crear una vista indexada en cada base de datos que realice la agregación pertinente.

Si el optimizador puede hacer coincidir las vistas indexadas con las consultas existentes, al menos se acelerará el proceso de agregación sin necesidad de volver a codificar las consultas existentes. Aunque la consulta global es más rápida que el proceso de combinación porque utiliza las agregaciones almacenadas en las vistas indexadas.

Una consulta de ejecución frecuente que agrega datos de varias tablas y, a continuación, utiliza UNION para combinar los resultados. No se permite utilizar UNION en una vista indexada. Se puede volver a diseñar vistas para realizar cada una de las operaciones individuales de agregación. El optimizador puede seleccionar a continuación las vistas indexadas para acelerar las consultas sin necesidad de volver a codificarlas. Si bien no se mejora el procesamiento UNION, sí se mejoran los procesos individuales de agregación.

Diseñe vistas indexadas que puedan satisfacer varias operaciones. Como el optimizador puede utilizar una vista indexada aunque no está especificada en la cláusula FROM, una vista indexada bien diseñada puede acelerar el procesamiento de muchas consultas.