**VISTAS**

**(PPT-2)**

Una **vista** es una **consulta guardada** con un nombre, que se comporta como si fuera una **tabla virtual**. No almacena datos físicamente (salvo que sea indexada), sino que presenta los datos al momento en que se consulta.

Las vistas son muy útiles para organizar, proteger y reutilizar consultas, pero deben usarse con cuidado en escenarios de alto rendimiento o donde se requiere mucha edición de datos.

**📌 Definición técnica**

Una vista es un **objeto de base de datos** que contiene una sentencia SELECT predefinida. Cuando se consulta la vista, SQL Server ejecuta esa sentencia y devuelve el resultado como si fuera una tabla.

**🔒 Definición orientada a seguridad**

Una vista permite **restringir el acceso** a ciertas columnas o filas de una tabla, proporcionando una forma controlada de exponer información sin permitir acceso directo a toda la tabla base.

**♻️ Definición orientada a reutilización**

Una vista permite **reutilizar consultas complejas** (con joins, filtros, cálculos) bajo un nombre simple y reutilizable, facilitando el mantenimiento del código SQL.

**📌 ¿Para qué se usan?**

* **Simplificar consultas complejas**  
  Puedes ocultar JOINs, filtros o cálculos detrás de un nombre fácil de usar.
* **Reutilizar lógica**  
  Puedes centralizar lógica de negocio (filtros, reglas) en una sola vista.
* **Seguridad y control de acceso**  
  Puedes dar permisos solo a ciertas columnas o registros, sin exponer toda la tabla.

Ejemplo básico:

Supón que tienes esta tabla =>

CREATE TABLE Empleados (

ID INT,

Nombre VARCHAR(100),

Departamento VARCHAR(50),

Salario DECIMAL(10,2)

);

Puedes crear una vista para mostrar solo empleados del área de TI =>

CREATE VIEW Vista\_TI AS

SELECT ID, Nombre, Salario

FROM Empleados

WHERE Departamento = 'TI';

Y luego usarla como si fuera una tabla =>

SELECT \* FROM Vista\_TI;

**🧠 Notas importantes:**

* Las vistas **no almacenan datos** (a menos que sean *indexadas* o *materializadas*).
* Puedes **anidar vistas** (una vista que usa otra).
* Puedes **actualizar datos** a través de una vista, pero con ciertas restricciones (por ejemplo, no debe tener GROUP BY o DISTINCT si vas a hacer INSERT, UPDATE o DELETE).

**✅ ¿Qué permiten hacer las vistas? (PPT-5)**

1. **Ocultar información**
   * Permitiendo el acceso a algunos datos y manteniendo oculto el resto de la información que no se incluye en la vista.
   * El usuario opera con los datos de una vista como si se tratara de una tabla, pudiendo modificar tales datos.
2. **Simplificar la administración** 
   * Los permisos de usuario: se pueden dar al usuario permisos para que solamente pueda acceder a los datos a través de vistas, en lugar de concederle permisos para acceder a ciertos campos, así se protegen las tablas base de cambios en su estructura.
3. **Simplificar consultas complejas**
   * Puedes encapsular consultas con múltiples JOIN, WHERE, GROUP BY, etc., y usarlas como si fueran una tabla.
   * Ejemplo: en lugar de escribir una consulta larga cada vez, usas SELECT \* FROM VistaResumenVentas.
4. **Reutilizar lógica de negocio**
   * Centralizas reglas, filtros o cálculos en una sola definición, y la reutilizas en distintas consultas o reportes.
5. **Restringir el acceso a los datos (seguridad)**
   * Puedes mostrar solo ciertas columnas o filas a determinados usuarios, sin darles acceso a toda la tabla.
   * Ejemplo: una vista que no muestre sueldos ni datos sensibles.
6. **Facilitar reportes**
   * Son útiles para preparar y organizar los datos de forma clara para herramientas de análisis o reportes.
7. **Trabajar como una tabla virtual**
   * Puedes hacer SELECT, y en muchos casos también INSERT, UPDATE y DELETE, dependiendo de cómo esté construida la vista.
8. **Independencia del esquema**
   * Si cambias internamente cómo se relacionan las tablas, puedes mantener la vista con el mismo nombre para que las aplicaciones no se vean afectadas.

✅ **Ventajas de las vistas (PPT-9)**

|  |  |
| --- | --- |
| **Ventaja** | **Descripción** |
| 🔹 **Simplicidad** | Permiten encapsular consultas complejas en un nombre sencillo. |
| 🔹 **Reutilización de lógica** | Puedes centralizar reglas de negocio y evitar repetir consultas. |
| 🔹 **Seguridad** | Permiten mostrar solo ciertas columnas o filas, protegiendo datos sensibles. |
| 🔹 **Facilitan el mantenimiento** | Si cambia la estructura interna de las tablas, puedes mantener la misma vista y evitar tocar el código de las aplicaciones. |
| 🔹 **Compatibles con herramientas de reportes** | Se pueden usar directamente en herramientas como Power BI o Excel. |
| 🔹 **Pueden admitir operaciones DML** | Algunas vistas permiten INSERT, UPDATE y DELETE, si están bien diseñadas. |

❌ **Desventajas de las vistas (PPT-9)**

|  |  |
| --- | --- |
| **Desventaja** | **Descripción** |
| ⚠️ **No almacenan datos** | Cada vez que se consulta una vista, se ejecuta la consulta base (salvo vistas indexadas). |
| ⚠️ **Rendimiento limitado** | Si la consulta es compleja o la vista es usada dentro de otras, puede afectar el rendimiento. |
| ⚠️ **Algunas vistas no son modificables** | Vistas con JOIN, GROUP BY, DISTINCT, TOP, etc., no permiten modificar datos directamente. |
| ⚠️ **No admiten parámetros** | No puedes pasarle valores directamente como a una función o procedimiento (se requieren filtros en el WHERE). |
| ⚠️ **Dependencias ocultas** | Cambios en las tablas subyacentes pueden romper la vista si no se actualiza adecuadamente. |

**✅ Creación de vistas en SQL Server (PPT-11)**

Una vista se crea usando la sentencia CREATE VIEW, seguida por el nombre de la vista y una consulta SELECT.

📌 **Sintaxis básica:**

CREATE VIEW nombre\_de\_la\_vista AS

SELECT columna1, columna2, ...

FROM tabla

WHERE condición;

**⚠️ Restricciones y consideraciones importantes (PPT-11)**

**🔹 1. No puede contener ORDER BY directamente**

* Solo se permite ORDER BY si usas TOP junto con ORDER BY.

**🔹 2. Restricciones para modificar datos (INSERT/UPDATE/DELETE)**

Solo puedes modificar datos a través de una vista si:

* No tiene DISTINCT, GROUP BY, UNION, TOP, funciones agregadas (como SUM), JOIN complejos, subconsultas escalar.
* Se basa en una única tabla.
* Todas las columnas modificables están directamente en la tabla base y no son calculadas.

**🔹 3. No puede tener parámetros**

Las vistas no aceptan parámetros como procedimientos o funciones. Pero puedes filtrar la vista desde la consulta que la llama.

**🔹 4. No puedes usar algunas cláusulas**

En una vista no se permiten directamente:

* COMPUTE, INTO, TEMPORARY TABLES, OPTION (RECOMPILE), etc.

**Protección de Información de las vistas (PPT-17)**

Proteger la información de las vistas en **SQL Server** es una práctica importante para **mantener la seguridad y la integridad de los datos**, especialmente cuando se exponen a usuarios o aplicaciones. Aquí te dejo algunas **estrategias y métodos** que puedes usar para proteger la información de las vistas:

**✅ Métodos para proteger la información en vistas**

1. **Controlar el acceso mediante permisos**  
   La forma más directa de proteger las vistas es **gestionar los permisos**. Puedes conceder acceso a las vistas solo a ciertos usuarios y roles, asegurando que solo los usuarios autorizados puedan ver los datos.
   * **Ejemplo: Conceder permisos solo de lectura**

sql

GRANT SELECT ON VistaEmpleadosActivos TO UsuarioEjemplo;

* + **Ejemplo: Denegar permisos**

sql

DENY SELECT ON VistaEmpleadosActivos TO UsuarioNoAutorizado;

1. **Usar vistas para restringir columnas y filas**  
   Puedes crear vistas que solo muestren **ciertas columnas** o **filas específicas** de una tabla, ocultando datos sensibles.

* **Ejemplo: Vista que omite datos sensibles**

sql

CREATE VIEW VistaEmpleadosSegura AS

SELECT ID, Nombre, Departamento

FROM Empleados

WHERE Activo = 1;

De esta manera, puedes evitar que los usuarios vean campos como Salario, FechaNacimiento, etc.

1. **Aplicar filtrado por usuario (seguridad a nivel de fila)**  
   Puedes diseñar vistas que restrinjan los datos basados en **atributos del usuario** mediante una función de seguridad que dependa de quien ejecute la consulta.

* **Ejemplo: Filtrar por usuario**

sql

CREATE VIEW VistaVentasSegura AS

SELECT \* FROM Ventas

WHERE VendedorID = USER\_ID();

Esto filtraría los resultados según el usuario que está ejecutando la consulta, protegiendo los datos de otros vendedores.

1. **Usar funciones de cifrado para datos sensibles**  
   Si tienes datos sensibles, como números de tarjetas de crédito o información personal, puedes cifrar esos datos antes de almacenarlos y **descifrarlos en las vistas** solo cuando sea necesario.

* **Ejemplo: Cifrado de datos sensibles**:

sql

CREATE VIEW VistaEmpleadosCifrados AS

SELECT Nombre, CONVERT(VARCHAR(100), DecryptByKey(CIFRADO), 1) AS NumeroSeguro

FROM Empleados;

1. **Aplicar roles y seguridad a nivel de objeto**  
   SQL Server permite crear **roles** de base de datos y asignarles permisos específicos. Puedes usar esto para garantizar que solo los usuarios que pertenezcan a ciertos roles puedan acceder a las vistas.

* **Ejemplo: Crear un rol con permisos restringidos**:

sql

CREATE ROLE RolLecturaEmpleados;

GRANT SELECT ON VistaEmpleadosActivos TO RolLecturaEmpleados;

1. **Uso de WITH ENCRYPTION para proteger la definición de la vista**  
   Si deseas **proteger el código** de la vista, puedes cifrar su definición para que otros no puedan ver la consulta SELECT que utiliza la vista.

* **Ejemplo: Crear una vista con encriptación**:

sql

CopiarEditar

CREATE VIEW VistaEmpleadosSegura

WITH ENCRYPTION AS

SELECT ID, Nombre

FROM Empleados

WHERE Activo = 1;

Esto evitará que otros usuarios puedan ver el código fuente de la vista, aunque podrán consultar los datos.

**🛠️ Cuándo se pueden modificar los datos a través de una vista (PPT-18)**

Las vistas actualizables permiten modificar datos, pero deben cumplir con ciertos requisitos.

Las vistas no actualizables (por ejemplo, aquellas con JOIN, GROUP BY, etc.) no permiten modificaciones directas, pero puedes usar triggers como INSTEAD OF para permitirlo.

1. **Vistas actualizables**  
   Una vista es **actualizable** si cumple con ciertos requisitos y está basada en una única tabla. En estos casos, puedes realizar operaciones de modificación directamente a través de la vista.

**Requisitos para una vista actualizable:**

* + La vista debe estar basada en **una sola tabla** (sin JOIN con otras tablas).
  + No debe contener funciones agregadas (SUM, COUNT, AVG, etc.), GROUP BY, ni DISTINCT.
  + No debe tener **subconsultas** que devuelvan más de un valor o más de una fila.
  + Las columnas de la vista deben ser **directamente modificables** y no derivadas de cálculos o agregaciones.

**❌ Cuándo no se pueden modificar los datos a través de una vista**

1. **Vistas no actualizables**  
   Si la vista no cumple con los requisitos mencionados, no podrás modificar los datos a través de ella. Por ejemplo:
   * **Vistas con JOIN** entre varias tablas:  
     Las vistas que combinan datos de varias tablas no son actualizables, ya que SQL Server no sabe cómo distribuir las actualizaciones en todas las tablas relacionadas.
   * **Vistas con GROUP BY, funciones agregadas o DISTINCT**:  
     Si la vista utiliza estos, no puedes modificar los datos, ya que la vista está agregando datos, y no hay forma de identificar qué fila original corresponde al resultado agregado.

**🔧 Operaciones permitidas en vistas no actualizables**

Aunque no se puede modificar directamente, aún puedes hacer **consultas** a la vista (SELECT) y usarla para obtener datos.

**🛡️ Uso de INSTEAD OF Triggers**

Si deseas permitir modificaciones en vistas no actualizables, puedes usar un **trigger INSTEAD OF** para interceptar las operaciones de modificación y definir cómo deben realizarse en las tablas subyacentes.

**Optimización del Rendimiento de las vistas (PPT-19)**

Las **vistas** en SQL Server no solo sirven para organizar y simplificar consultas, también pueden ayudar —en ciertos casos— a **optimizar el rendimiento** de tus consultas. A continuación te detallo cómo y cuándo las vistas pueden mejorar el rendimiento, y qué debes tener en cuenta para evitar que lo empeoren.

Las vistas pueden ayudarte a optimizar el rendimiento cuando se usan con inteligencia:

* Son ideales para simplificar lógica, filtrar datos, y reutilizar consultas.
* Para grandes volúmenes de datos o consultas frecuentes, considera vistas indexadas.
* Siempre verifica que no estés creando un cuello de botella al apilar vistas complejas.

**✅ Formas en que las vistas pueden mejorar el rendimiento =>**

**1. Evitar consultas repetitivas y costosas**

* Si tienes una consulta compleja (con JOIN, filtros, cálculos) que usas frecuentemente, una vista te permite **almacenar esa lógica** y evitar repetir el costo de escribir y analizar la misma consulta una y otra vez.

⚠️ Aunque la vista **no almacena datos** (excepto si es indexada), sí ayuda a que el optimizador reutilice su plan de ejecución.

**2. Vistas indexadas (materializadas)**

* Una **vista indexada** guarda físicamente los datos resultantes de la consulta, como si fuera una tabla.
* Mejora el rendimiento en consultas sobre grandes volúmenes de datos agregados, resúmenes o cálculos que no cambian frecuentemente.

🟢 Ventajas:

* Acelera notablemente las consultas de lectura.
* Útil en reportes y análisis.

🔴 Desventajas:

* Costosa al insertar/actualizar datos en las tablas base.
* Tiene muchas restricciones (debes usar SCHEMABINDING, no se permiten OUTER JOIN, DISTINCT, UNION, etc.).

**3. Filtrado lógico previo**

Una vista puede limitar la cantidad de filas o columnas que una consulta necesita procesar, reduciendo la carga de lectura en la tabla base.

**4. Abstracción que permite indexación específica**

* Si usas la vista frecuentemente para ciertos reportes, puedes **crear índices** en las tablas base que optimicen esa vista en particular, enfocándote en los filtros o JOINs que usa.

**5. Reducción de ancho de banda**

* Una vista que selecciona solo columnas necesarias puede reducir los datos transferidos en consultas desde aplicaciones o reportes, acelerando la carga.

**❌ Cuándo las vistas pueden afectar negativamente el rendimiento**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Problema** | **Explicación** | | ⚠️ Anidamiento excesivo | Si una vista llama a otra vista, y esta a su vez a otra, se pueden generar consultas difíciles de optimizar y más lentas. | | ⚠️ Vistas complejas sin filtros | Usar vistas que consultan toda una tabla sin WHERE, y luego aplicar filtros en la consulta final, puede hacer que el optimizador ejecute más trabajo del necesario. | | ⚠️ No usar índices en tablas base | Si las tablas subyacentes no tienen buenos índices, la vista no será más rápida. | | ⚠️ Mal uso de funciones | Si usas funciones escalares (como ISNULL, LEFT, CAST) en las columnas de la vista, puede evitar el uso de índices. | |  |
|  |  |
|  |  |
|  |  |

🧠 Buenas prácticas para optimizar el rendimiento con vistas

|  |  |
| --- | --- |
| **Práctica** | **Beneficio** |
| Usa SCHEMABINDING cuando sea posible | Requiere que la vista esté ligada a la tabla, lo cual permite crear índices y mejorar integridad. |
| Revisa los planes de ejecución | Para saber si tu vista está generando un plan eficiente o está sobrecargando recursos. |
| Evita SELECT \* | Solo selecciona las columnas necesarias para reducir el ancho de banda y el trabajo de lectura. |
| Limita el número de JOINs y subconsultas | Especialmente en vistas muy usadas por reportes o dashboards. |
| Considera vistas indexadas para reportes pesados | Si tu vista se usa solo para lectura y los datos no cambian con frecuencia. |

**Asignación de permisos a vistas (PPT-21)**

Puedes otorgar permisos sobre una **vista** (como SELECT, INSERT, UPDATE, DELETE) de la misma forma que sobre una tabla, usando los comandos GRANT, REVOKE o DENY.

**✅ 1. Comando GRANT para dar permisos sobre una vista**

**🔹 Sintaxis básica:**

sql

GRANT permiso

ON nombre\_de\_vista

TO usuario\_o\_rol;

**🔹 Ejemplos:**

**➤ Dar permiso de solo lectura (SELECT) sobre una vista:**

sql

GRANT SELECT ON VistaEmpleados TO UsuarioReportes;

**➤ Permitir actualizar datos a través de la vista:**

(Siempre que la vista lo permita)

sql

GRANT UPDATE ON VistaEmpleados TO UsuarioEditores;

**➤ Permitir insertar registros (si la vista lo soporta):**

sql

GRANT INSERT ON VistaEmpleados TO UsuarioAlta;

**➤ Permitir eliminar registros a través de la vista:**

sql

GRANT DELETE ON VistaEmpleados TO UsuarioAdmin;

**➤ Otorgar varios permisos al mismo tiempo:**

sql

GRANT SELECT, UPDATE ON VistaEmpleados TO UsuarioModificador;

**🚫 2. Quitar permisos con REVOKE o DENY**

* **REVOKE**: Quita permisos previamente otorgados.

sql

REVOKE SELECT ON VistaEmpleados FROM UsuarioReportes;

* **DENY**: Prohíbe explícitamente el acceso, incluso si pertenece a un rol con permiso.

sql

DENY SELECT ON VistaEmpleados TO UsuarioCurioso;

**👥 3. Asignar permisos a roles (buena práctica)**

En lugar de dar permisos usuario por usuario, puedes crear un **rol** y asignar permisos al rol:

sql

CREATE ROLE RolLecturaEmpleados;

GRANT SELECT ON VistaEmpleados TO RolLecturaEmpleados;

EXEC sp\_addrolemember 'RolLecturaEmpleados', 'UsuarioReportes';

**📌 Notas importantes:**

* El usuario debe tener acceso a las **tablas subyacentes** si no se usa WITH SCHEMABINDING.
* Si quieres ocultar la lógica de la vista y solo dar acceso al resultado, asegúrate de **negar el acceso a la tabla base** y solo otorgar acceso a la vista.