SQL Server 檢視表（Views）詳解

基於提供的資料表結構，以下是 SQL Server 檢視表的詳細說明，包含實際範例。

定義

檢視表（View）是基於一個或多個資料表的預定義 SQL 查詢所產生的虛擬資料表。檢視表不儲存實際數據，而是在每次查詢時動態產生結果。檢視表可以像普通資料表一樣被查詢，是一種將複雜查詢封裝的方式。

檢視表的種類

1. 標準檢視表（Standard Views）

最基本且常用的檢視表類型，用於簡化複雜查詢、提供安全性控制等。

```
-- 產品資訊檢視表：結合產品與分類資訊
CREATE VIEW vw_ProductInfo AS
SELECT
    p.product_id,
    p.product_name,
    c.category_name,
    p.price,
    p.stock,
    p.description
FROM Products p
JOIN Categories c ON p.category_id = c.category_id;
```

2. 索引檢視表（Indexed Views）

包含**實體儲存數據的檢視表**，可以顯著提高查詢效能，特別是對於**經常執行的複雜查詢**。

```
-- 熱銷產品索引檢視表
CREATE VIEW vw_PopularProducts
WITH SCHEMABINDING AS
SELECT
    p.product_id,
    p.product_name,
    SUM(od.quantity) AS total_sold,
    COUNT_BIG(*) AS row_count
FROM dbo.Products p
JOIN dbo.OrderDetails od ON p.product_id = od.product_id
GROUP BY p.product_id, p.product_name;

-- 建立唯一叢集索引
CREATE UNIQUE CLUSTERED INDEX IX_vw_PopularProducts
ON vw_PopularProducts(product_id);
```

3. 分割檢視表（Partitioned Views）

將資料**水平分割到多個資料表**中，然後**透過檢視表來整合**它們。

```
-- 假設我們依年份分割了訂單資料
CREATE VIEW vw_AllOrders AS
SELECT * FROM Orders2023
UNION ALL
SELECT * FROM Orders2024
UNION ALL
SELECT * FROM Orders2025;
```

4. 系統檢視表（System Views）

SQL Server 內建的檢視表，用於查詢資料庫元數據和系統狀態。

```
-- 查詢所有使用者資料表
SELECT * FROM sys.tables WHERE type = 'U';

-- 查詢資料庫中的所有外鍵關係
SELECT * FROM sys.foreign_keys;
```

檢視表的優點

1. **簡化複雜查詢**：將複雜的查詢邏輯封裝，提供簡化的介面。

```
-- 訂單摘要檢視表
CREATE VIEW vw_OrderSummary AS
SELECT
    o.order_id,
    c.name AS customer_name,
    e.name AS employee_name,
    o.order_date,
    o.total_amount,
    o.status
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN Employees e ON o.employee_id = e.employee_id;
```

2. **安全性控制**：限制使用者只能訪問特定欄位或資料。

```
-- 員工基本資訊檢視表（隱藏敏感資料）
CREATE VIEW vw_PublicEmployeeInfo AS
SELECT
    employee_id,
    name,
    email
FROM Employees;
```

3. **資料抽象化**：隱藏底層資料結構的複雜性，提供業務邏輯層面的抽象。

```sql
-- 訂單詳細檢視表
CREATE VIEW vw_DetailedOrders AS
SELECT
    o.order_id,
    c.name AS customer_name,
    p.product_name,
    od.quantity,
    od.unit_price,
    od.subtotal,
    o.order_date,
    o.status
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN OrderDetails od ON o.order_id = od.order_id
JOIN Products p ON od.product_id = p.product_id;
```

4. **資料獨立性**：底層資料表結構變更時，可以修改檢視表定義而不影響應用程式。
5. **資料整合**：能夠整合來自多個資料表的資料，提供統一的訪問點。

檢視表的缺點

1. **效能考量**：標準檢視表每次查詢都需要執行基礎查詢，可能影響效能。
2. **更新限制**：複雜的檢視表（如涉及多個資料表的聯結或彙總函數）可能不支援直接更新。
3. **索引檢視表的維護成本**：索引檢視表需要額外的儲存空間和維護開銷。
4. **可讀性和追蹤困難**：大量使用嵌套檢視表可能導致查詢追蹤困難。

實際應用範例

1. 銷售報表檢視表

```sql
CREATE VIEW vw_SalesReport AS
SELECT
    c.category_name,
    p.product_name,
    SUM(od.quantity) AS total_quantity,
    SUM(od.subtotal) AS total_sales
FROM OrderDetails od
JOIN Products p ON od.product_id = p.product_id
JOIN Categories c ON p.category_id = c.category_id
GROUP BY c.category_name, p.product_name;
```

2. 庫存警報檢視表

```
CREATE VIEW vw_LowStockAlert AS
SELECT
    product_id,
    product_name,
    stock,
    price
FROM Products
WHERE stock <= 10;

select * from vw_LowStockAlert ORDER BY stock ASC;
```

3. 客戶訂單歷史檢視表

```
CREATE VIEW vw_CustomerOrderHistory AS
SELECT
    c.customer_id,
    c.name AS customer_name,
    COUNT(o.order_id) AS total_orders,
    SUM(o.total_amount) AS total_spent,
    MAX(o.order_date) AS last_order_date
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name;
```

4. 熱門產品檢視表

```
CREATE VIEW vw_TopSellingProducts AS
SELECT TOP 20
    p.product_id,
    p.product_name,
    c.category_name,
    SUM(od.quantity) AS total_sold,
    SUM(od.subtotal) AS total_revenue
FROM Products p
JOIN OrderDetails od ON p.product_id = od.product_id
JOIN Categories c ON p.category_id = c.category_id
GROUP BY p.product_id, p.product_name, c.category_name
ORDER BY total_sold DESC;
```

5. 員工銷售業績檢視表

```
CREATE VIEW vw_EmployeePerformance AS
SELECT
    e.employee_id,
    e.name AS employee_name,
    COUNT(DISTINCT o.order_id) AS orders_processed,
    SUM(o.total_amount) AS total_sales
FROM Employees e
JOIN Orders o ON e.employee_id = o.employee_id
GROUP BY e.employee_id, e.name;
```

檢視表管理

建立檢視表

```
CREATE VIEW vw_名稱 AS
SELECT 欄位1, 欄位2, ...
FROM 資料表
WHERE 條件;
```

修改檢視表

```
ALTER VIEW vw_名稱 AS
SELECT 修改後的查詢;
```

刪除檢視表

```
DROP VIEW vw_名稱;
```

查詢檢視表

```
SELECT * FROM vw_名稱;
```

最佳實踐

1. **命名規範**：使用前綴（如 vw_）明確識別檢視表
2. **適當文件**：使用 `WITH DESCRIPTION` 或註解說明檢視表用途
3. **避免過深嵌套**：減少檢視表調用其他檢視表的層級
4. **考慮效能影響**：針對頻繁查詢的檢視表考慮使用索引檢視表
5. **定期維護**：檢視並優化檢視表定義，確保與資料庫結構同步

檢視表是 SQL Server 中強大的功能，能夠在維護資料庫結構和查詢彈性間取得平衡，適當使用可以顯著提升開發效率和資料安全性。