**Лабораторна робота № 7. Представлення і табличні об'єкти**

## Представлення

Представлення або Views представляють віртуальні таблиці. Але на відміну від звичайних стандартних таблиць в базі даних представлення містять запити, які динамічно витягають використовувані дані. Представлення дають нам ряд переваг. Вони спрощують комплексні SQL-операції. Вони захищають дані, так як представлення можуть дати доступ до частини таблиці, а не до всієї таблиці. Представлення також дозволяють повертати відформатовані значення з таблиць в потрібній і зручній формі.

Для створення представлення використовується команда CREATE VIEW, яка має наступну форму:

|  |
| --- |
| CREATE VIEW назва\_представлення [(стовбчик\_1, стовбчик\_2, ....)]  AS вираз\_SELECT |

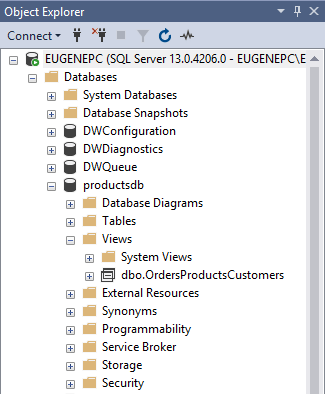
Наприклад, нехай у нас є три пов'язаних таблиці:

|  |
| --- |
| CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  );  CREATE TABLE Customers  (  Id INT IDENTITY PRIMARY KEY,  FirstName NVARCHAR (30) NOT NULL  );  CREATE TABLE Orders  (  Id INT IDENTITY PRIMARY KEY,  ProductId INT NOT NULL REFERENCES Products (Id),  CustomerId INT NOT NULL REFERENCES Customers (Id),  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price MONEY NOT NULL  ); |

Тепер додамо в базу даних, в якій містяться дані таблиці, таке представлення:

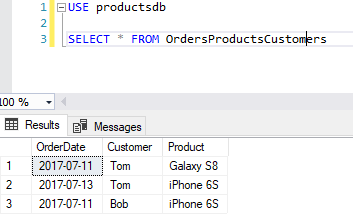
|  |
| --- |
| CREATE VIEW OrdersProductsCustomers AS  SELECT Orders.CreatedAt AS OrderDate,  Customers.FirstName AS Customer,  Products.ProductName As Product  FROM Orders INNER JOIN Products ON Orders.ProductId = Products.Id  INNER JOIN Customers ON Orders.CustomerId = Customers.Id |

Тобто дане подання фактично буде повертати зведені дані з трьох таблиць. І після його створення ми зможемо його побачити в вузлі Views у обраної бази даних в SQL Server Management Studio:



Тепер використовуємо створене вище представлення для отримання даних:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM OrdersProductsCustomers |



При створенні уявлень слід враховувати, що представлення, як і таблиці, повинні мати унікальні імена в рамках тієї ж бази даних.

Представлення можуть мати не більше 1024 стовпців і можуть звертати не більше ніж до 256 таблиць. Також можна створювати представлення на основі інших уявлень. Такі представлення ще називають вкладеними (nested views). Однак рівень вкладеності не може бути больще 32-х.

Команда SELECT, яка використовується в поданні, не може включати вирази INTO або ORDER BY (за винятком тих випадків, коли також застосовується вираз TOP або OFFSET). Якщо ж необхідне сортування даних в поданні, то вираз ORDER BY застосовується в команді SELECT, яка отримує дані з представлення.

Також при створенні вистави можна визначити набір його стовпців:

|  |
| --- |
| CREATE VIEW OrdersProductsCustomers2 (OrderDate, Customer, Product)  AS SELECT Orders.CreatedAt,  Customers.FirstName,  Products.ProductName  FROM Orders INNER JOIN Products ON Orders.ProductId = Products.Id  INNER JOIN Customers ON Orders.CustomerId = Customers.Id |

**Зміна представлення**

Для зміни представлення використовується команда ALTER VIEW. Ця команда має практично той же самий синтаксис, то і CREATE VIEW:

|  |
| --- |
| ALTER VIEW назва\_представлення [(стовбчик\_1, стовбчик\_2, ....)]  AS вираз\_SELECT |

Наприклад, змінимо вище створене представлення OrdersProductsCustomers:

|  |
| --- |
| ALTER VIEW OrdersProductsCustomers  AS SELECT Orders.CreatedAt AS OrderDate,  Customers.FirstName AS Customer,  Products.ProductName AS Product,  Products.Manufacturer AS Manufacturer  FROM Orders INNER JOIN Products ON Orders.ProductId = Products.Id  INNER JOIN Customers ON Orders.CustomerId = Customers.Id |

**Видалення представлення**

Для видалення представлення викликається команда DROP VIEW:

|  |  |
| --- | --- |
| 1 | DROP VIEW OrdersProductsCustomers |

Також варто відзначити, що при видаленні таблиць також слід видалити та подання, які використовують ці таблиці.

## Оновлюване представлення

Представлення можуть бути поновлюваними (updatable). У таких представленнях ми можемо змінити або видалити рядки або додати в них нові рядки.

При створенні подібних уявлень є безліч обмежень. Зокрема, команда SELECT в поданні не може містити:

* **TOP**
* **DISTINCT**
* **UNION**
* **JOIN**
* агрегатні функції типу COUNT або MAX
* **GROUP BY**і HAVING
* підзапити
* похідні стовпці або стовпці, які обчислюються на підставі кількох значень
* звернення одночасно до кількох таблиць

Припустимо, у нас є наступна таблиця:

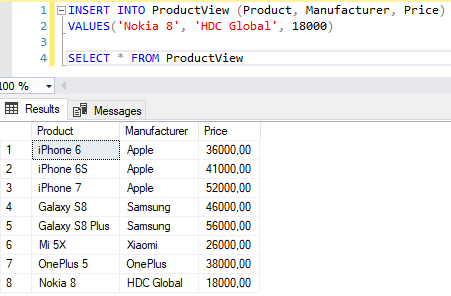
|  |
| --- |
| CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  ) |

І створимо оновлюється представлення:

|  |
| --- |
| CREATE VIEW ProductView  AS SELECT ProductName AS Product, Manufacturer, Price  FROM Products |

Додамо в нього дані:

|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO ProductView (Product, Manufacturer, Price)  VALUES ( 'Nokia 8', 'HDC Global', 18000)  SELECT \* FROM ProductView |



Варто відзначити, що при додаванні фактично буде додано об'єкт в таблицю Products, яку використовує представлення ProductView. І тому треба враховувати, що якщо в цій таблиці є які-небудь стовпці, в які подання не додає дані, але які не допускають значення NULL або не підтримують значення за замовчуванням, то додавання завершиться з помилкою.

Оновлення рядки представлення:

|  |  |
| --- | --- |
| 1  2 | UPDATE ProductView  SET Price = 15000 WHERE Product = 'Nokia 8' |

Видалення рядка в поданні:

|  |  |
| --- | --- |
| 1  2 | DELETE FROM ProductView  WHERE Product = 'Nokia 8' |

Оновлення та видалення також зачіпають ту таблицю, яку використовує представлення.

## **Тимчасові і похідні таблиці**

**Тимчасові таблиці**

На додаток до табличних змінним можна визначати тимчасові таблиці. Такі таблиці можуть бути корисні для зберігання табличних даних всередині складного комплексного скрипта.

Тимчасові таблиці існують протягом сесії бази даних. Якщо така таблиця створюється в редакторі запитів (Query Editor) в SQL Server Management Studio, то таблиця буде існувати поки відкрите редактор запитів. Таким чином, до тимчасової таблиці можна звертатися з різних скриптів всередині редактора запитів.

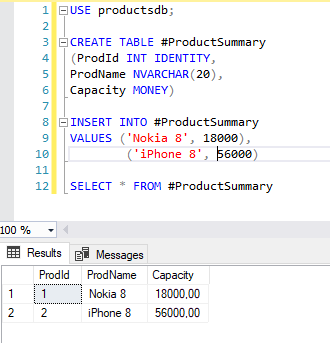
Після створення все тимчасові таблиці зберігаються в таблиці tempdb, яка є за замовчуванням в MS SQL Server.

Якщо необхідно видалити таблицю до завершення сесії бази даних, то для цієї таблиці слід виконати команду DROP TABLE.

Назва тимчасової таблиці починається зі знака решітки #. Якщо використовується один знак #, то створюється локальна таблиця, яка доступна в протягом поточної сесії. Їли використовуються два знака ##, то створюється глобальна тимчасова таблиця. На відміну від локальної глобальна тимчасова таблиця доступна всім відкритим сесій бази даних.

Наприклад, створимо локальну тимчасову таблицю:

|  |
| --- |
| CREATE TABLE #ProductSummary  (ProdId INT IDENTITY,  ProdName NVARCHAR (20),  Price MONEY)  INSERT INTO #ProductSummary  VALUES ( 'Nokia 8', 18000),  ( 'IPhone 8', 56000)  SELECT \* FROM #ProductSummary |



І з цією таблицею можна працювати в більшій мірі як і зі звичайною таблицею - отримувати дані, додавати, змінювати і видаляти їх. Тільки після закриття редактора запитів ця таблиця перестане існувати.

Подібні таблиці зручні для якихось тимчасових проміжних даних. Наприклад, нехай у нас є три таблиці:

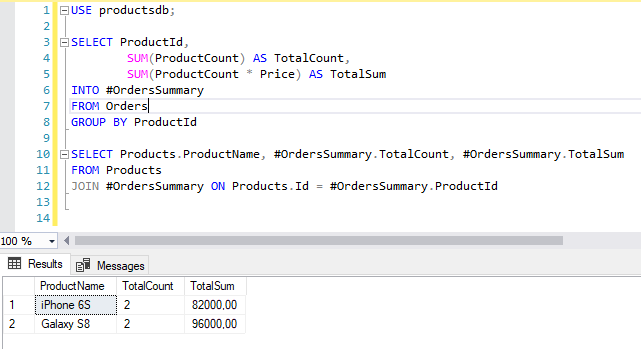
|  |
| --- |
| CREATE TABLE Products  (  Id INT IDENTITY PRIMARY KEY,  ProductName NVARCHAR (30) NOT NULL,  Manufacturer NVARCHAR (20) NOT NULL,  ProductCount INT DEFAULT 0,  Price MONEY NOT NULL  );  CREATE TABLE Customers  (  Id INT IDENTITY PRIMARY KEY,  FirstName NVARCHAR (30) NOT NULL  );  CREATE TABLE Orders  (  Id INT IDENTITY PRIMARY KEY,  ProductId INT NOT NULL REFERENCES Products (Id) ON DELETE CASCADE,  CustomerId INT NOT NULL REFERENCES Customers (Id) ON DELETE CASCADE,  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price MONEY NOT NULL  ); |

Виведемо в тимчасову таблицю проміжні дані з таблиці Orders:

|  |
| --- |
| SELECT ProductId,  SUM (ProductCount) AS TotalCount,  SUM (ProductCount \* Price) AS TotalSum  INTO #OrdersSummary  FROM Orders  GROUP BY ProductId  SELECT Products.ProductName, # OrdersSummary.TotalCount, # OrdersSummary.TotalSum  FROM Products  JOIN #OrdersSummary ON Products.Id = # OrdersSummary.ProductId |

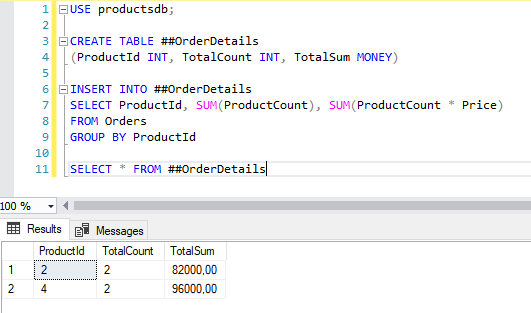
Тут спочатку витягуються дані в тимчасову таблицю #OrdersSummary. Причому так як дані в неї витягуються за допомогою виразу SELECT INTO, то попередньо таблицю не треба створювати. І ця таблиця буде містити id товару, загальна кількість проданого товару і на яку суму був проданий товар.

Потім ця таблиця може використовуватися у виразах INNER JOIN.



Подібним чином визначаються глобальні тимчасові таблиці, єдине, що їх ім'я починається з двох знаків ##:

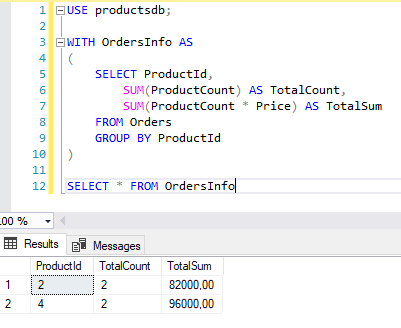
|  |
| --- |
| CREATE TABLE ## OrderDetails  (ProductId INT, TotalCount INT, TotalSum MONEY)  INSERT INTO ## OrderDetails  SELECT ProductId, SUM (ProductCount), SUM (ProductCount \* Price)  FROM Orders  GROUP BY ProductId  SELECT \* FROM ## OrderDetails |



**Похідні таблиці**

Крім тимчасових таблиць MS SQL Server дозволяє створювати похідні таблиці, які в плані продуктивності є більш ефективним рішенням, ніж тимчасові. Похідна таблиця задається за допомогою ключового слова WITH:

|  |
| --- |
| WITH OrdersInfo AS  (  SELECT ProductId,  SUM (ProductCount) AS TotalCount,  SUM (ProductCount \* Price) AS TotalSum  FROM Orders  GROUP BY ProductId  )  SELECT \* FROM OrdersInfo - тут нормально  SELECT \* FROM OrdersInfo - тут помилка  SELECT \* FROM OrdersInfo - тут помилка |



На відміну від тимчасових таблиць похідні зберігаються в оперативній пам'яті і існують тільки під час першого виконання запиту, який представляє цю таблицю.

**Завдання**

1. Створити view, який буде містити дані про всі замовлення з вказанням імені користувача
2. Створити view, який буде містити сумарну кількість замовлень за кожним замовником.
3. Створити view, який буде містити кількість замовлень за кожен день.
4. За допомогою view з завдання 2 знайти замовника з найбільшою кількістю замовлень.
5. За допомогою view з завдання 3 знайти день з найбільшою кількістю замовлень.
6. Зробити на базі таблиці Products view, який дозволить правити тільки назву продукту і виробника. Перевірити роботу через UPDATE.
7. \*Додати в таблицю Customers поле isDelted, яке буде означати, що записи з значенням 1 вважаються видаленими. Створити оновлюваний view, який не буде показувати видалені записи.