Putra Pandu Adikara, S.Kom

VIEW & TABLE

Basis Data 2

View

View

- **View** merupakan virtual table di mana isinya (kolom dan baris) didefinisikan dari suatu query (yang dapat melibatkan beberapa tabel sekaligus melalui JOIN, menggunakan agregasi, grouping, dll).
- Beberapa tujuan penggunaan View:
  - Untuk memfokuskan, menyederhanakan, dan mengkustomisasi dari persepsi tiap user yang memiliki database
  - Sebagai mekanisme keamanan untuk memberikan akses ke data melalui view, tapi tidak memberikan ijin untuk akses langsung/pengubahan ke base table
  - Untuk memberikan antarmuka backward compatible untuk mengemulasikan table suatu skema yang berubah

View

- Data yang ditampilkan pada view tidak dapat diubah kecuali pada kondisi batasan tertentu.
- Pengubahan data dapat melalui mekanisme pilihan:
  - **INSTEAD OF Trigger**
    - Dijelaskan pada materi Trigger
  - **Partitioned Views**
    - Djelaskan kemudian

Deklarasi View

- Syntax

```
CREATE VIEW [ schema_name . ]view_name
[(column [ ,...n ] )]
[ WITH <view_attribute> [ ,...n ] ]
AS
select_statement
```

Contoh: Deklarasi View

```
USE AdventureWorks2008R2;
GO

CREATE VIEW hiredate_view
AS
SELECT p.FirstName, p.LastName,
   e.BusinessEntityID, e.HireDate
FROM HumanResources.Employee e
JOIN Person.Person AS p ON e.BusinessEntityID
= p.BusinessEntityID;
GO
```

Contoh: Deklarasi View

```
USE AdventureWorks2008R2 ;
GO

CREATE VIEW Sales.SalesPersonPerform
AS
SELECT TOP (100) SalesPersonID, SUM(TotalDue)
AS TotalSales
FROM Sales.SalesOrderHeader
WHERE OrderDate >
CONVERT(DATETIME,'20001231',101)
GROUP BY SalesPersonID;
GO
```

Partitioned Views

Partitioned View

- **Partitioned View** adalah suatu View yang didefinisikan dari UNION ALL dari tabel-tabel yang dibuat dengan struktur yang sama, tapi disimpan pada beberapa tabel pada instance SQL Server atau group dari server dengan instance autonomous SQL Server, disebut *federated database server*.
  - Data di partisi untuk optimasi (misal untuk query, dll), memudahkan pengelolaan ketika data sangat besar

Contoh: Deklarasi Partitioned View 1.1

```
--Create the tables and insert the values.
CREATE TABLE dbo.SUPPLY1 (
supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 1 and 150),
supplier CHAR(50)
);
CREATE TABLE dbo.SUPPLY2 (
supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 151 and 300),
supplier CHAR(50)
);
CREATE TABLE dbo.SUPPLY3 (
supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 301 and 450),
supplier CHAR(50)
);
GO
```

Contoh: Deklarasi Partitioned View 1.2

```
INSERT dbo.SUPPLY1 VALUES ('1', 'CaliforniaCorp'), ('5',
'BraziliaLtd');

INSERT dbo.SUPPLY2 VALUES ('231', 'FarEast'), ('280',
'NZ');

INSERT dbo.SUPPLY3 VALUES ('321', 'EuroGroup'), ('442',
'UKArchip');
GO
```

Contoh: Deklarasi Partitioned View 1.3

```
--Create the view that combines all supplier tables.
CREATE VIEW dbo.all_supplier_view
WITH SCHEMABINDING
AS
SELECT supplyID, supplier FROM dbo.SUPPLY1
UNION ALL
SELECT supplyID, supplier FROM dbo.SUPPLY2
UNION ALL
SELECT supplyID, supplier FROM dbo.SUPPLY3
GO
```

Contoh: Deklarasi Partitioned View

- Contoh partitioned view dari beberapa node/server

```
--Partitioned view as defined on Server1
CREATE VIEW Customers
AS
--Select from local member table.
SELECT * FROM CompanyData.dbo.Customers_33
UNION ALL
--Select from member table on Server2.
SELECT * FROM Server2.CompanyData.dbo.Customers_66
UNION ALL
--Select from mmeber table on Server3.
SELECT * FROM Server3.CompanyData.dbo.Customers_99
```

Indexed View

Indexed View

- Konsep awal Index digunakan untuk meningkatkan kinerja query
- Indexed views memberikan manfaat kinerja yang tidak dapat dicapai dengan menggunakan indeks standar.
- Indexed views dapat meningkatkan query performance dengan cara berikut:
  - Agregasi dapat di-precompute dan disimpan dalam index untuk mengurangi komputasi yang mahal selama eksekusi query.
  - Tabel dapat di-prejoine dan data set yang dihasilkan disimpan.
  - Kombinasi dari join atau agregasi dapat disimpan.

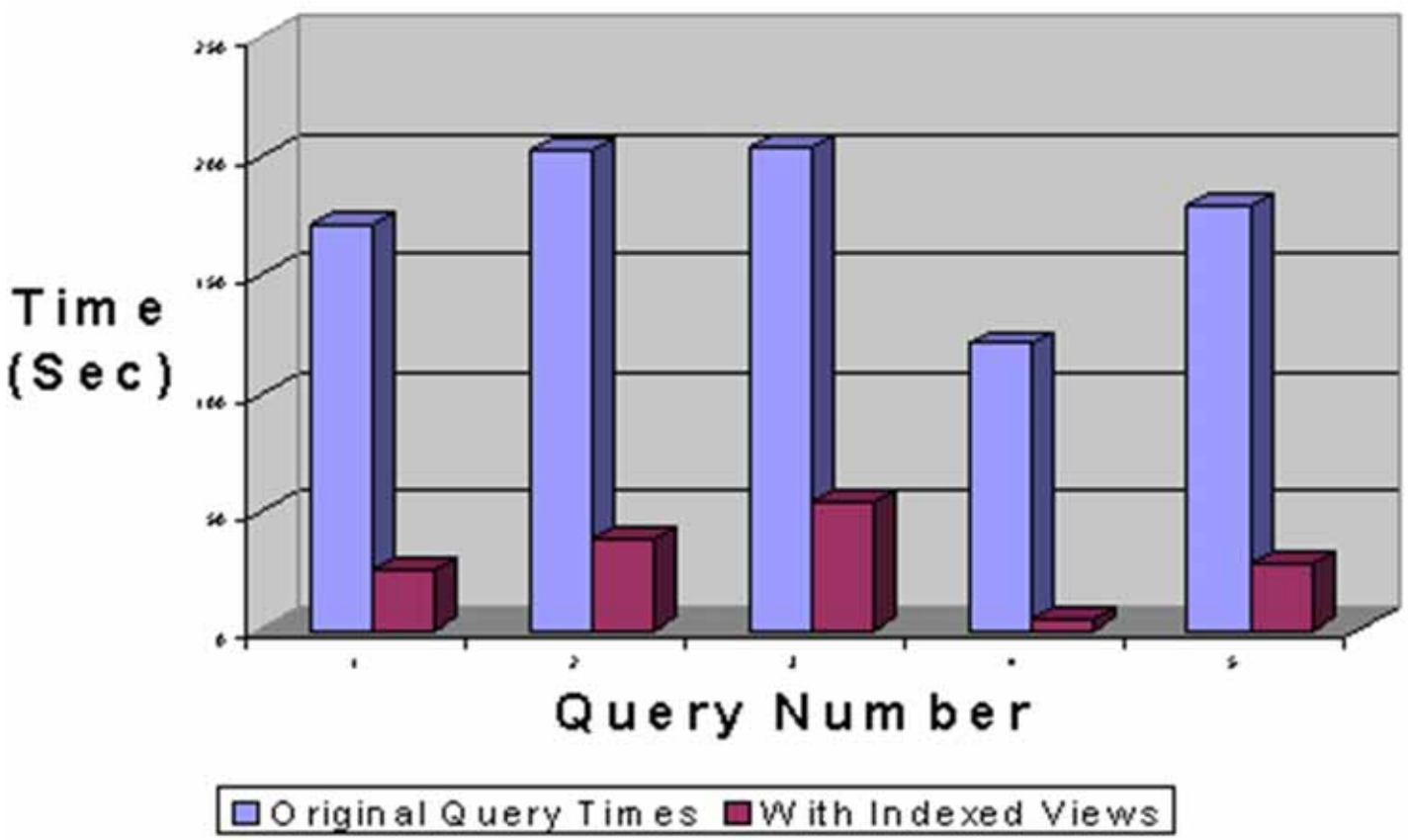
Pengunaan Indexed View

- Sebelum menggunakan Indexed View, analisis terlebih dahulu beban kerja database.
- Gunakan tool seperti SQL Profiler untuk mengetahui mana View yang diuntungkan melalui Indexed View
  - Biasanya query dengan agregasi dan join adalah kandidat terbaik sebagai Indexed View
  - Tidak semua query diuntungkan dengan Indexed View
- Walaupun diuntungkan secara performansi query, tapi dibutuhkan pula tambahan ruang hardisk, maintenance, dan optimisasi.

Pengunaan Indexed View

- Penggunaan Indexed View berguna untuk diimplementasikan pada:
  - Decision support workloads
  - Data marts
  - Data warehouses
  - Online analytical processing (OLAP) stores and sources
  - Data mining workloads

Waktu Query Biasa vs Indexed View



Contoh Indexed View

- **Query 1**

```
SELECT TOP 5 ProductID, Sum(UnitPrice*OrderQty) -
Sum(UnitPrice*OrderQty*(1.00-UnitPriceDiscount)) AS Rebate
FROM Sales.SalesOrderDetail
GROUP BY ProductID
ORDER BY Rebate DESC
```

- **View 1**

```
CREATE VIEW Vdiscount1 WITH SCHEMABINDING AS
SELECT SUM(UnitPrice*OrderQty) AS SumPrice,
SUM(UnitPrice*OrderQty*(1.00-UnitPriceDiscount)) AS
SumDiscountPrice,
COUNT_BIG(*) AS Count, ProductID
FROM Sales.SalesOrderDetail
GROUP BY ProductID
GO
CREATE UNIQUE CLUSTERED INDEX VDiscountInd ON Vdiscount1
(ProductID)
```

Partitioned Table

Partitioned Table

- **Optimasi kinerja** pada database yang mempunyai t**abel-tabel yang sangat besar (Very Large Database/VLDB)** dapat dilakukan dengan cara **mempartisi tabel-tabel**.
- Pada SQL Server partisi tabel menjadi filegroups yang terpisah.
- Pada SQL Server 2005, dimungkinkan untuk menyebar data disk yang berbeda secara fisik, sehingga mempengaruhi performansi akses bersamaan (*concurrent)* dari disk-disk tsb untuk optimasi kinerja query

Horizontal Partitioning

- **Horizontal Partitioning** adalah partisi dimana tabel dipisahkan menjadi beberapa tabel kecil yang memiliki kolom sama namun lebih sedikit baris.
- Tabel akan disimpan ke filegroup ke tempat yang berbeda secara fisik
- Digunakan bila baris sangat besar misalnya hingga jutaan baris

Horizontal Partitioning

- Manfaat horizontal partitioning:
  - Tiap tabel partisi memiliki lebih sedikit baris, sehingga untuk membutuhkan waktu lebih cepat untuk pencarian
  - Index tiap tabel partisi menjadi lebih kecil sehingga mempercepat pencarian dibandingkan tabel yang tidak terpartisi
  - Bila diperlukan, dapat ditempatkan tiap partisi pada filegroup berbeda di beberapa disk/volume RAID/drive controller
  - Bila membuat partitioned view dari partitioned table, view akan memperlakukan keseluruhan tabel dan Query Processor (QP) hanya akan menggunakan tabel yang digunakan untuk memenuhi query.

Proses Pembuatan Partition

- Pembuatan partisi di SQL Server memerlukan tiga proses:
  - Pembuatan partition function
  - Pembuatan partition schema
  - Pemartisian table

1. Pembuatan Partition Function

- **Partition function** adalah suatu fungsi untuk pembagian data. Data mana yang ditaruh pada partisi ke 1, ke 2, ke 3, dst.
- Misalnya fungsi partisi yang membagi berdasarkan ID dari suatu tabel.
- Contohnya: tabel Customer dengan Customer Number (ID) unik dari 1 – 1.000.000, dibagi menjadi 4 partisi.
- Fungsinya:
  - `CREATE PARTITION FUNCTION customer_partfunc (int)`
  - `AS RANGE RIGHT`
  - `FOR VALUES (250000, 500000, 750000)`

1. Pembuatan Partition Function

- Fungsi di atas membagi 4 partisi:
  - Partisi 1: 1-249.999
  - Partisi 2: 250.000-499.999
  - Partisi 3: 500.000-799.999
  - Partisi 4: 750.000-dst

- **`RANGE RIGHT`** → batas yang digunakan: 1-249.999, 250.000-499.999
- **`RANGE LEFT`** → batas yang digunakan: 1-250.000, 250.001-500.000

2. Pembuatan Partition Scheme

- Setelah membuat fungsi partisi untuk memecah data, kemudian buat skema partisi untuk menghubungkan partisi ke filegroups.
- Misal membuat 4 filegroup bernama **fg1** hingga **fg4:**

```
CREATE PARTITION SCHEME customer_partscheme
AS PARTITION customer_partfunc
TO (fg1, fg2, fg3, fg4)
```

3. Pemartisian Table

- Setelah membuat skema, untuk menghubungkan tabel ke skema maka ditambahkan klausa **"ON"** pada DDL pembuatan tabel dan menentukan skema partisi dan kolom mana untuk diaplikasikan pada fungsi partisi.

```
CREATE TABLE customers (
    FirstName nvarchar(40),
    LastName nvarchar(40),
    CustomerNumber int)
ON customer_partscheme (CustomerNumber)
```

Referensi

- **SQL SERVER – 2005 – Database Table Partitioning Tutorial – How to Horizontal Partition Database Table**
  - http://blog.sqlauthority.com/2008/01/25/sql-server-2005-database-table-partitioning-tutorial-how-to-horizontal-partition-database-table/
- **Partitioned Tables in SQL Server 2005**
  - http://www.simple-talk.com/sql/database-administration/partitioned-tables-in-sql-server-2005/
- **Partitioned Tables and Indexes in SQL Server 2005**
  - http://msdn.microsoft.com/en-us/library/ms345146%28SQL.90%29.aspx

Temporary Table
Table Variable

Temporary Table

- **Temporary table** adalah tabel temporer yang tersedia hanya pada sesi yang membuatnya misal stored procedure
- Dibuat didalam database **tempdb** secara fisik, dicatat di transaction log
- Deklarasi menggunakan satu tanda pagar **#table_name**:
  - ```
    CREATE TABLE #people (
        id INT,
        name VARCHAR(32)
    )
    ```
- Table ini otomatis dihapus pada saat penghentian procedure atau sesi yang membuatnya
- Tapi sebaiknya dihapus secara manual setelah selesai menggunakannya dengan menggunakan **drop**
  - ```
    drop #table_name
    ```

Global Temporary Table

- **Global Temporary Tables** adalah temporary table yang secara global terlihat ke seluruh sesi koneksi SQL Server dan seluruh user dapat melihat.
  - Jarang digunakan di SQL Server.
- Akan di-drop otomatis ketika sesi terakhir selesai menggunakan temporary table
- Dibuat didalam database **tempdb** secara fisik, dicatat di transaction log
- Deklarasinya menggunakan dua tanda pagar **##table_name**
  - ```
    CREATE TABLE ##global_people (
        id INT,
        name VARCHAR(32)
    )
    ```
    ```

Table Variable

- **Table Variable** disimpan di dalam memory, ditempatkan seperti table.
- Table variable sebagian disimpan di disk dan sebagian lagi disimpan di memory
- Lebih cepat dibandingkan temporary table
- Deklarasi menggunakan tanda @ (**@table_name):**
  - ```
    DECLARE @people TABLE (
        id INT,
        name VARCHAR(32)
    )
    ```
    ```

Kapan Digunakan?

- Bila memiliki kurang dari 100 baris umumnya gunakan table variable, bila lebih gunakan temporary table. Karena SQL Server tidak membuat statistik pada table variable
- Bila perlu membuat/menggunakan index gunakan temporary table
- Table variable tidak dapat di-**`truncate`**
- Table variable tidak dapat di-**`alter`** setelah dideklarasikan
- Table variable tidak dapat mempunyai constraint