T-SQL: Views

\(^o^)/

September, 2019

Section 1

Introduction

An Example

How to hide some details of Vendors table from an application?

Vendors(VendorID, VendorName, VendorAddress1, VendorAddress2, VendorCity, VendorState, VendorZipCode, VendorPhone, VendorContactLName, VendorContactFName, DefaultTermsID, DefaultAccountNo)

Create a view to hide detailed table structure

```
CREATE VIEW VendorsGeneral
AS
SELECT VendorID, VendorName, VendorPhone
FROM Vendors;
```

the created “virtual table”

VendorsGeneral(VendorID, VendorName, VendorPhone)

View Concept

- Single table derived from other tables called the defining tables
- Considered to be a virtual table that is not necessarily populated

Benefits of Using Views

- Design independence
- Data security
- Flexibility
- Simplified queries
- Updatability

Section 2

Create and Manage Views

Create Views

```
CREATE VIEW 'view_name' [(column_name_1 [,column_name_2]...)]
[WITH {ENCRYPTION|SCHEMABINDING|ENCRYPTION,SCHEMABINDING}]
AS
select_statement
[WITH CHECK OPTION]
```

Notes

- Almost any select statement could be used
- Can refer up to 256 tables
- Could be nested (based on other views) up to 32 levels
- No ORDER BY except with TOP / OFFSET and FETCH
- For sorting use ORDER BY while querying views
- Two ways to specify column name
  - After view name (has to name all columns)
  - In select statement (only columns to rename)

Example: view with join

```
CREATE VIEW VendorInvoices
AS
SELECT VendorName, InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Vendors JOIN Invoices ON Vendors.VendorID = Invoices.VendorID;
```

Options

- WITH ENCRYPTION: encrypt SQL code for view
- WITH SCHEMA BINDING: preventing
  - Dropping base tables
  - Modification of base tables affecting the view
  - SELECT * not allowed
- WITH CHECK OPTION: prevent updating a row if it would no longer be included in the view

Example: view with schema_binding

```
CREATE VIEW VendorsDue
WITH SCHEMABINDING
AS
SELECT InvoiceDate AS Date, VendorName AS Name,
    VendorContactFName + ' ' + VendorContactLName AS Contact,
    InvoiceNumber AS Invoice,
    InvoiceTotal - PaymentTotal - CreditTotal AS BalanceDue
FROM dbo.Vendors JOIN dbo.Invoices
    ON Vendors.VendorID = Invoices.VendorID
WHERE InvoiceTotal - PaymentTotal - CreditTotal > 0;
```

Updatable Views

- SELECT statement can't include (for unambiguously determining base tables and columns affected):
- DISTINCT / TOP
- Aggregate functions
- Calculated value
- GROUPBY / HAVING clause
- View can't include UNION operator
- Avoid updating data through views using INSERT, UPDATE & DELETE whenever possible:
- Inflexible & prone to errors
- Better use INSTEAD OF triggers to update

Example: create an updatable view

```
CREATE VIEW VendorPayment
AS
SELECT VendorName, InvoiceNumber, InvoiceDate, PaymentDate,
    InvoiceTotal, CreditTotal, PaymentTotal
FROM Invoices JOIN Vendors ON Invoices.VendorID = Vendors.VendorID
WHERE InvoiceTotal - PaymentTotal - CreditTotal > 0;
```

Example: test an updatable view

```sql
SELECT *
FROM VendorPayment;

UPDATE VendorPayment
SET PaymentTotal = 19351.18, PaymentDate = '2016-04-02'
WHERE VendorName = 'Malloy Lithographing Inc' AND InvoiceNumber = 'P-0608';
```

Modify and Delete Views

- Delete

```
DROP VIEW 'view_name'
```

- Modify

```
ALTER VIEW 'view_name' [(column_name_1 [,column_name_2]...)]
[WITH {ENCRYPTION|SCHEMABINDING|ENCRYPTION,SCHEMABINDING}]
AS
select_statement
[WITH CHECK OPTION]
```

Section 3

Catalog Views

Catalog Views

- Independent of system tables structure
  - Why views? No worry of structure changes on upgrading
- Querying like ordinary views

Examples

View name	Contents
sys.schemas	One row for each schema in the current database.
sys.sequences	One row for each sequence in the current database.
sys.tables	One row for each table in the current database.
sys.views	One row for each view in the current database.
sys.columns	One row for each column in each table, view, or table-valued function in the current database.
sys.key_constraints	One row for each primary or unique key in each table in the current database.
sys.foreign_keys	One row for each foreign key.
sys.foreign_key_columns	One row for each column or set of columns that make up a foreign key.
sys.objects	One row for each user-defined object in the current database, except for triggers.

Figure 1: Some catalog views