View

NGUYEN HongPhuong
Email: phuongnh@soict.hust.edu.vn
Site: http://users.soict.hust.edu.vn/phuongnh
Face: https://www.facebook.com/phuongnhbk
Hanoi University of Science and Technology

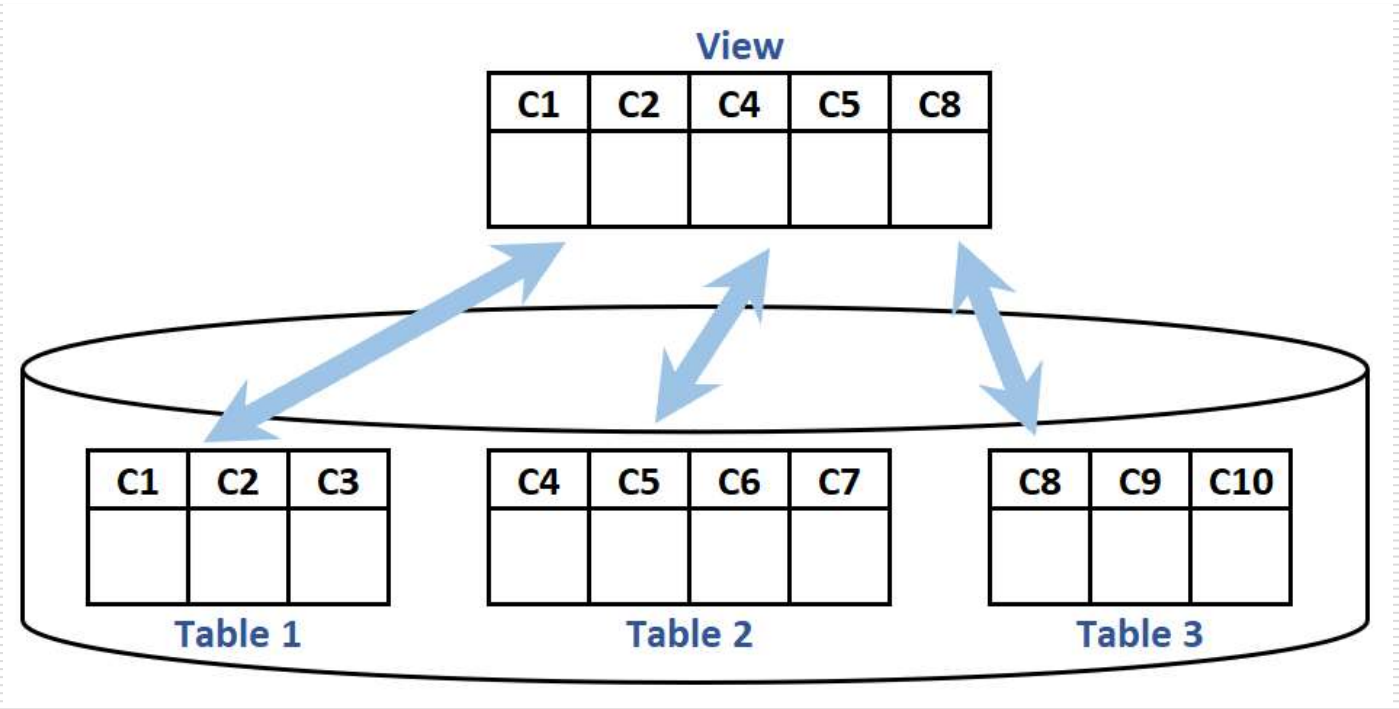
Contents

1. Introduction

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view may consist of columns from multiple tables using joins or just a subset of columns of a single table. This makes views useful for abstracting or hiding complex queries.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

1. Introduction (2)

- By definition, views do not store data except for **indexed views**.



2. Advantages of views

Generally speaking, views provide the following advantages:

- Security
  - You can restrict users to access directly to a table and allow them to access a subset of data via views.
  - For example, you can allow users to access customer name, phone, email via a view but restrict them to access the bank account and other sensitive information.

2. Advantages of views (2)

- Simplicity
  - A relational database may have many tables with complex relationships (e.g., 1-1, 1-n) that make it difficult to navigate. However, you can simplify the complex queries with joins and conditions using a set of views.
- Consistency
  - Sometimes, you need to write a complex formula or logic in every query. To make it consistent, you can hide the complex queries logic and calculations in views. Once views are defined, you can refer the logic from the views rather than rewriting it in separate queries.

3. Types of Views

- Besides the standard role of basic user-defined views, SQL Server provides the following types of views that serve special purposes in a database:
  - Indexed Views
  - Partitioned Views
  - System Views

Indexed Views

- An indexed view is a view that has been materialized. This means the view definition has been computed and the resulting data stored just like a table. You index a view by creating a unique clustered index on it. Indexed views can dramatically improve the performance of some types of queries. Indexed views work best for queries that aggregate many rows. They are not well-suited for underlying data sets that are frequently updated.

Partitioned Views

- A partitioned view joins horizontally partitioned data from a set of member tables across one or more servers. This makes the data appear as if from one table. A view that joins member tables on the same instance of SQL Server is a local partitioned view.

System Views

- System views expose catalog metadata. You can use system views to return information about the instance of SQL Server or the objects defined in the instance. For example, you can query the sys.databases catalog view to return information about the user-defined databases available in the instance.

4. Creating a view

- Limitations
  - A view can be created only in the current database.
  - A view can have a maximum of 1,024 columns.
- Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

4. Creating a view (2)

- Examples

```
CREATE VIEW vCompany1 AS
SELECT Name, Address, Telephone
FROM Company

CREATE VIEW vCompany2 AS
SELECT Name, Address, Telephone
FROM Company
WHERE Address LIKE '%Japan%'

CREATE VIEW vComSupPro1(ComName, ProdName, Qty) AS
SELECT Company.Name, Product.Name, Quantity
FROM Company INNER JOIN Supply ON Company.CompanyID =
Supply.CompanyID
INNER JOIN Product ON Supply.ProductID = Product.ProductID
```

5. Querying on a view

- Later, you can refer to the view in the SELECT statement like a table as follows:

```
SELECT * FROM vCompany1
WHERE Address LIKE'%Germany%';
```

- When receiving this query, SQL Server executes the following query:

```
SELECT * FROM
(SELECT Name, Address, Telephone
FROM Company)
WHERE Address LIKE'%Germany%';
```

6. Modifying a view

- Modifying a view does not affect any dependent objects, such as stored procedures or triggers, unless the definition of the view changes in such a way that the dependent object is no longer valid.
- ALTER VIEW can be applied to indexed views; however, ALTER VIEW unconditionally drops all indexes on the view.

6. Modifying a view (2)

- Syntax

```
ALTER VIEW view_name AS
SQL_statement
```

- Example

```
ALTER VIEW vCompany2 AS
SELECT Name, Address, Telephone
FROM Company
WHERE Address LIKE '%Germany%'
```

7. Deleting a view

- Syntax

```
DROP VIEW view_name;
```

- Example

```
DROP VIEW vCompany1;
DROP VIEW vCompany2;
DROP VIEW vComSupPro1;
```

8. Indexed view

- Regular views are the saved queries that provide some benefits such as query simplicity, business logic consistency, and security. However, they do not improve the underlying query performance.
- Unlike regular views, indexed views are *materialized views* that stores data physically like a table, hence, may provide some the performance benefit if they are used appropriately.

8. Indexed view (2)

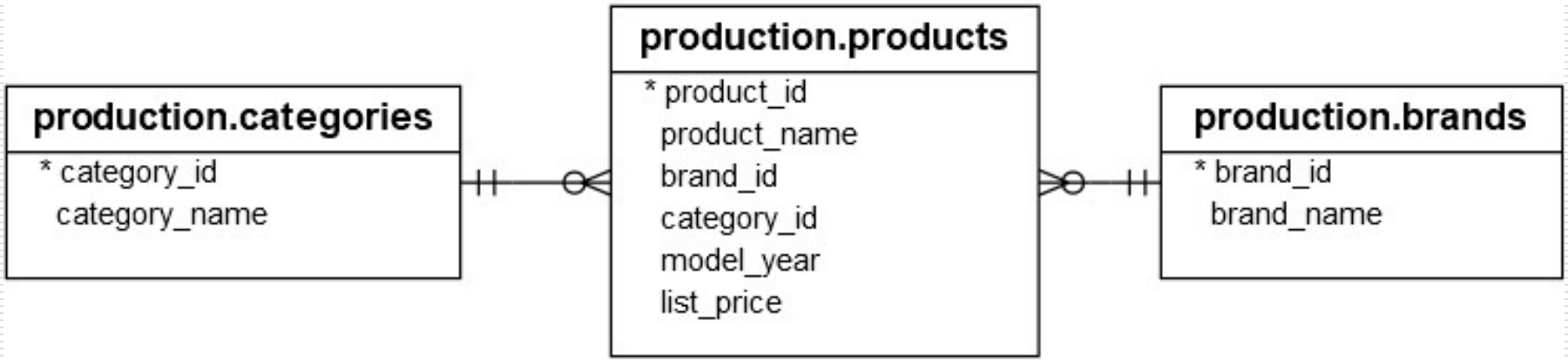
- To create an indexed view, follow these steps:
  - First, create a view that uses the WITH SCHEMABINDING option which binds the view to the schema of the underlying tables.
  - Second, create a unique clustered index on the view. This materializes the view.

8. Indexed view (3)

- Because of the WITH SCHEMABINDING option, if you want to change the structure of the underlying tables which affect the indexed view's definition, you must drop the indexed view first before applying the changes.

8. Indexed view (4)

- An example



8. Indexed view (5)

- An example

```
CREATE VIEW product_master WITH SCHEMABINDING
AS
SELECT product_id, product_name, model_year,
list_price, brand_name, category_name
FROM production.products p  INNER JOIN
production.brands b ON b.brand_id = p.brand_id
INNER JOIN production.categories c ON
c.category_id = p.category_id;
```

8. Indexed view (6)

- Before creating a unique clustered index for the view, let's examine the query I/O cost statistics by querying data from a regular view and using the SET STATISTICS IO command:

```
SET STATISTICS IO ON
GO
SELECT  *
FROM production.product_master
ORDER BY  product_name;
GO
```

- Let's add a unique clustered index to the view:

```
CREATE UNIQUE CLUSTERED INDEX ucidx_product_id
ON production.product_master(product_id);
```

8. Indexed view (7)

- You can also add a non-clustered index on the product_name column of the view:

```
CREATE NONCLUSTERED INDEX ucidx_product_name
ON production.product_master(product_name);
```