SQL Server Views

When you use the SELECT statement to query data from one or more tables, you get a result set.

For example, the following statement returns the product name, brand, and list price of all products from the `products` and `brands` tables:

```
SELECT
    product_name,
    brand_name,
    list_price
FROM
    production.products p
INNER JOIN production.brands b
        ON b.brand_id = p.brand_id;
```

Next time, if you want to get the same result set, you can save this query into a text file, open it, and execute it again.

SQL Server provides a better way to save this query in the database catalog through a view.

A view is a named query stored in the database catalog that allows you to refer to it later.

So the query above can be stored as a view using the CREATE VIEW statement as follows:

```
CREATE VIEW sales.product_info
AS
SELECT
    product_name,
    brand_name,
    list_price
FROM
    production.products p
INNER JOIN production.brands b
        ON b.brand_id = p.brand_id;
```

Code language: SQL (Structured Query Language) (sql)

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

```
SELECT * FROM sales.product_info;
```

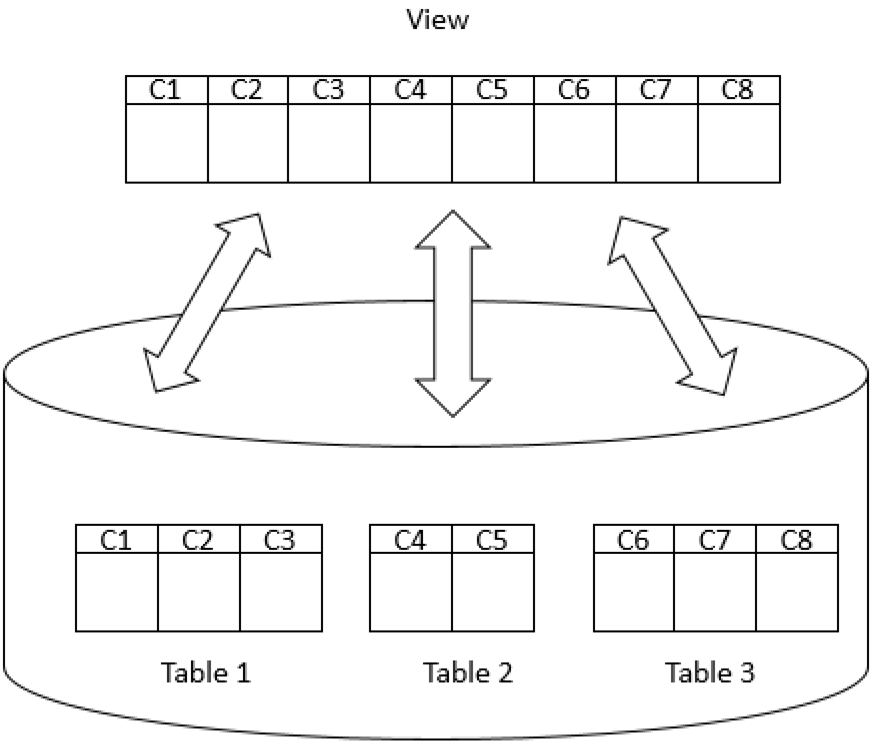
When receiving this query, SQL Server executes the following query:

```
SELECT
    *
FROM (
    SELECT
        product_name,
        brand_name,
        list_price
    FROM
        production.products p
    INNER JOIN production.brands b
        ON b.brand_id = p.brand_id;
);
Code language: SQL (Structured Query Language) (sql)
```

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

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.

The following picture illustrates a view that includes columns from multiple tables:



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.

Simplicity

A relational database may have many tables with complex relationships e.g., one-to-one and one-to-many 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 reference the logic from the views rather than rewriting it in separate queries.