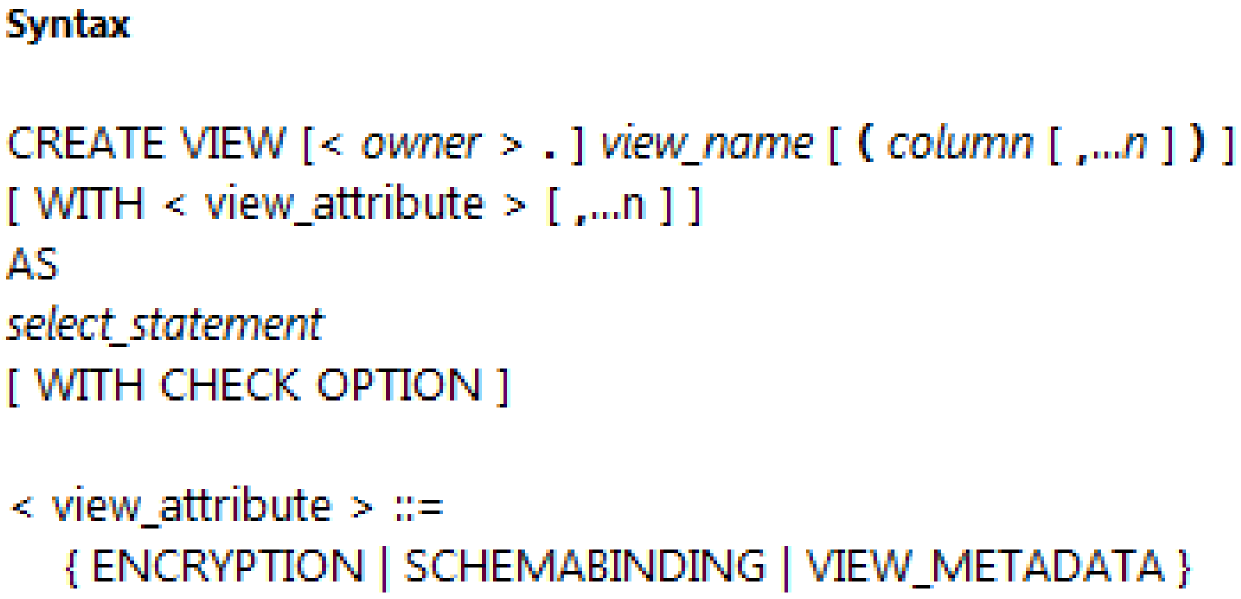
VIEW

Nguyen Minh Dao

CREATE VIEW

- Creates a virtual table that represents the data in one or more tables in an alternative way.
- CREATE VIEW must be the first statement in a query batch.



Syntax

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

< view_attribute > ::=
    { ENCRYPTION | SCHEMABINDING | VIEW_METADATA }
```

CREATE VIEW

- There are a few restrictions on the SELECT clauses in a view definition.
- A CREATE VIEW statement cannot:
  - Include **COMPUTE** or **COMPUTE BY** clauses.
  - Include **ORDER BY** clause, unless there is also a TOP clause in the select list of the SELECT statement.
  - Include the **INTO** keyword.
  - Reference a **temporary table** or a **table variable**.

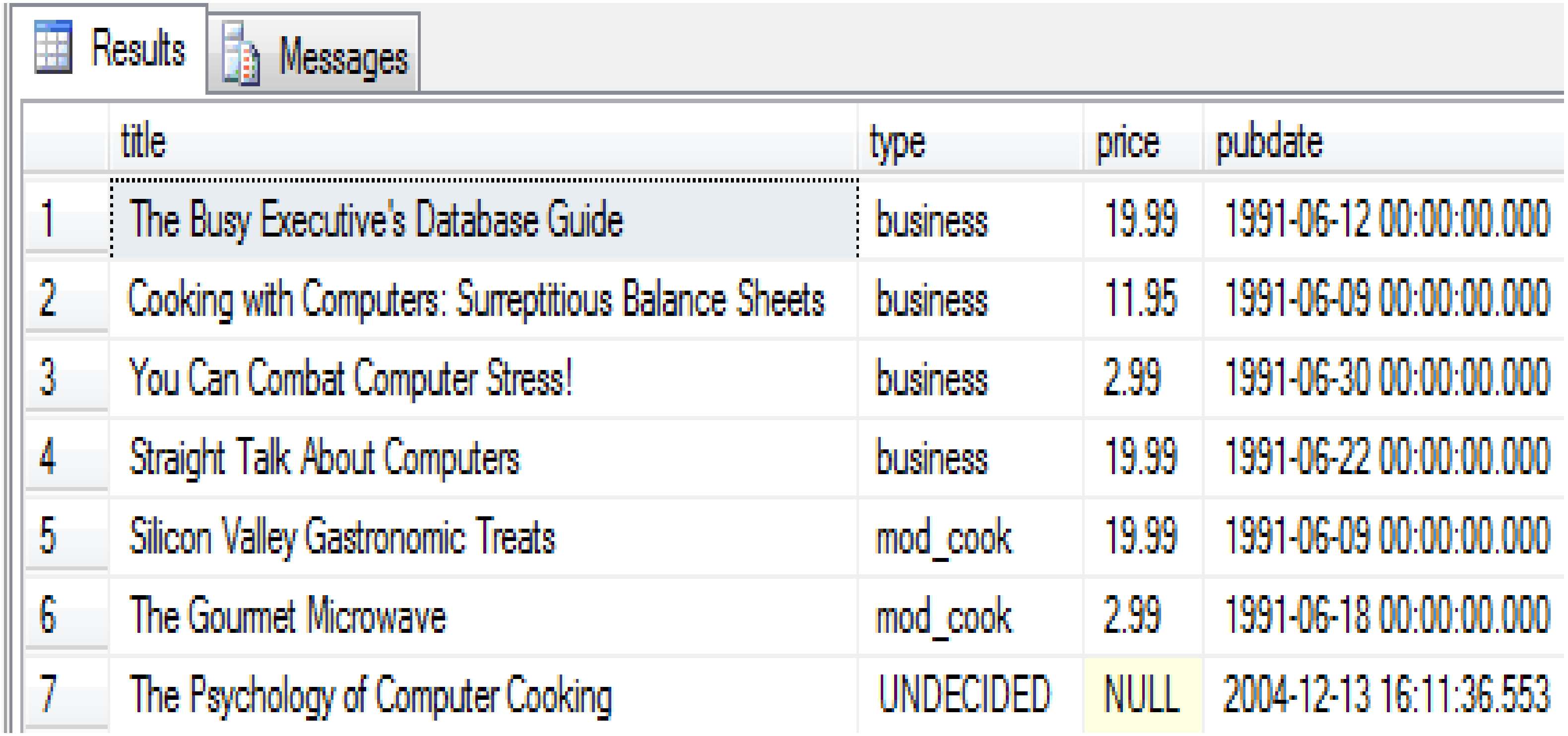
Use a simple CREATE VIEW

- USE **pubs**
- IF EXISTS (SELECT TABLE_NAME
  - FROM INFORMATION_SCHEMA.VIEWS
  - WHERE TABLE_NAME = **'titles_view'**)
  - DROP VIEW **titles_view**

- CREATE VIEW **titles_view**
  - AS
    - SELECT title, type, price, pubdate
    - FROM titles

- SELECT *
  - FROM titles_view

Use a simple CREATE VIEW



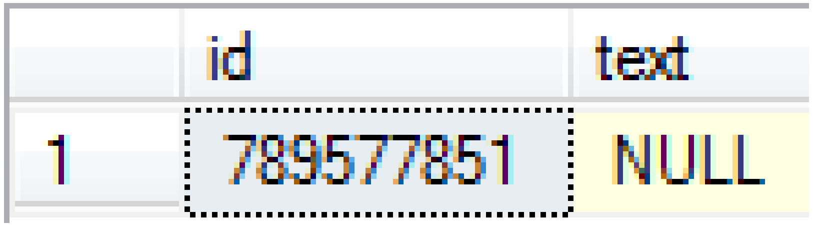
	title	type	price	pubdate
1	The Busy Executive's Database Guide	business	19.99	1991-06-12 00:00:00.000
2	Cooking with Computers: Surreptitious Balance Sheets	business	11.95	1991-06-09 00:00:00.000
3	You Can Combat Computer Stress!	business	2.99	1991-06-30 00:00:00.000
4	Straight Talk About Computers	business	19.99	1991-06-22 00:00:00.000
5	Silicon Valley Gastronomic Treats	mod_cook	19.99	1991-06-09 00:00:00.000
6	The Gourmet Microwave	mod_cook	2.99	1991-06-18 00:00:00.000
7	The Psychology of Computer Cooking	UNDECIDED	NULL	2004-12-13 16:11:36.553

Use WITH ENCRYPTION

- This example uses the **WITH ENCRYPTION** option and shows computed columns, renamed columns, and multiple columns.
- CREATE VIEW **accounts** (title, advance, amt_due)
  - **WITH ENCRYPTION**
  - AS
  - SELECT title, advance, price * royalty * ytd_sales
  - FROM titles
  - WHERE price > $5
- GO

Use WITH ENCRYPTION

- EXEC sp_helptext '**accounts**‘
- **Result**
  - *The text for object 'accounts' is encrypted.*
- SELECT c.id, c.text
  - FROM syscomments c, sysobjects o
  - WHERE c.id = o.id and o.name = '**accounts**'
- GO
- **Result**



	id	text
1	789577851	NULL

Use WITH CHECK OPTION

- This example shows a view named **CAonly** that allows data modifications to apply only to authors within the state of California.
- USE **pubs**
- IF EXISTS (SELECT TABLE_NAME
  - FROM INFORMATION_SCHEMA.VIEWS
  - WHERE TABLE_NAME = **'CAonly'**)
  - DROP VIEW **CAonly**
- GO
- CREATE VIEW **CAonly**
  - AS
  - SELECT au_lname, au_fname, city, state
  - FROM authors
  - WHERE state = 'CA'
  - **WITH CHECK OPTION**
- GO

Use WITH CHECK OPTION

- update **CAonly**
  - set **state='WA'**
  - where au_lname='White‘
- Error
  - Msg 550, Level 16, State 1, Line 1
  - The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION and one or more rows resulting from the operation did not qualify under the CHECK OPTION constraint.
  - The statement has been terminated.

Use built-in functions within a view

- This example shows a view definition that includes a built-in function. When you use functions, the derived column must include a column name in the CREATE VIEW statement.
- USE **pubs**
- IF EXISTS (SELECT TABLE_NAME
  - FROM INFORMATION_SCHEMA.VIEWS
  - WHERE TABLE_NAME = '**categories**')
  - DROP VIEW **categories**
- CREATE VIEW **categories** (category, average_price)
  - AS
    - SELECT type, AVG(price)
    - FROM titles
    - GROUP BY type

SCHEMABINDING

- Binds the view to the schema of the underlying table or tables.
- When SCHEMABINDING is specified, the base table or tables cannot be modified in a way that would affect the view definition.
- The view definition itself must first be modified or dropped to remove dependencies on the table that is to be modified.

Use partitioned data

- This example uses tables named **SUPPLY1**, **SUPPLY2**, **SUPPLY3**, and **SUPPLY4**, which correspond to the supplier tables from four offices, located in different countries/regions.
- --create the tables and insert the values
  - CREATE TABLE **SUPPLY1** (
    - supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 1 and 150), supplier CHAR(50)
    -)
  - CREATE TABLE **SUPPLY2** (
    - supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 151 and 300), supplier CHAR(50))

Use partitioned data

- CREATE TABLE **SUPPLY3** (
  - supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 301 and 450), supplier CHAR(50))
- CREATE TABLE **SUPPLY4** (
  - supplyID INT PRIMARY KEY CHECK (supplyID BETWEEN 451 and 600), supplier CHAR(50))
- INSERT SUPPLY1 VALUES ('1', 'CaliforniaCorp')
- INSERT SUPPLY1 VALUES ('5', 'BraziliaLtd')
- INSERT SUPPLY2 VALUES ('231', 'FarEast')
- INSERT SUPPLY2 VALUES ('280', 'NZ')
- INSERT SUPPLY3 VALUES ('321', 'EuroGroup')
- INSERT SUPPLY3 VALUES ('442', 'UKArchip')
- INSERT SUPPLY4 VALUES ('475', 'India')
- INSERT SUPPLY4 VALUES ('521', 'Afrique')
- GO

Use partitioned data

- --create the view that combines all supplier tables
- CREATE VIEW **all_supplier_view**
  - AS
    - SELECT *
      - FROM **SUPPLY1**
    - UNION ALL
    - SELECT *
      - FROM **SUPPLY2**
    - UNION ALL
    - SELECT *
      - FROM **SUPPLY3**
    - UNION ALL
    - SELECT *
      - FROM **SUPPLY4**

Thank You!