

Base de Données II

Chap3: Vues / Views

Outline

- 1 View overview
- 2 Create views in TSQL
- 3 The with clause
- 4 Updatable views

Views



- In some cases, it is not desirable for all users to see the **entire logical model** (that is, all the actual relations stored in the database).
Example: need to know an instructor's name and his department, but not his salary;
- A view provides a mechanism to **hide certain data from the view of certain users**;



Views



- In some cases, it is not desirable for all users to see the **entire logical model** (that is, all the actual relations stored in the database).
Example: need to know an instructor's name and his department, but not his salary;
- A view provides a mechanism to **hide certain data from the view of certain users**;
- A view is a named **virtual table** that is defined by a query and used as a table;
- Unlike permanent tables, a view has no physical representation of its data, unless you create an index on it.



Outline



- 1 View overview
- 2 Create views in TSQL**
- 3 The with clause
- 4 Updatable views



View definition

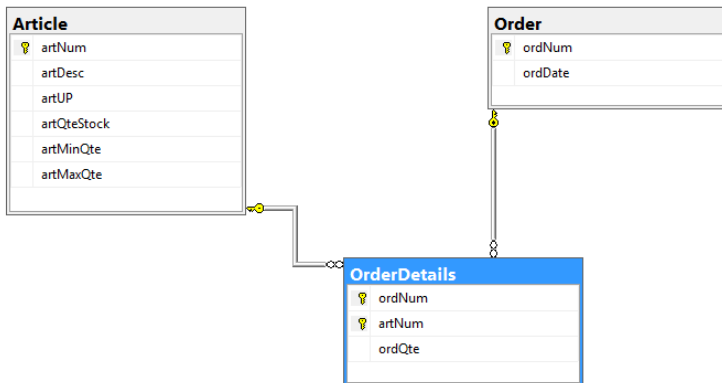
- A **view** is defined using the **create view** statement which has the form:

```
1 create view viewName as <query expression>
```

where <query expression> is any legal SQL expression.

- Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.

Database to use





Example 1

A view consisting of **articles and their order dates**.

```
1 create view articlesOrderDates as
2 select artNum, artDesc, ordDate
3 from Article, OrderDetails, Order
4 where Article.artNum=OrderDetails.artNum and OrderDetails.ordNum=Order.ordNum
```

Find all articles that have been ordered in 2009.

```
1 select artNum, artDesc
2 from articlesOrderedDate
3 where ordDate between '01/01/2009' and '31/12/2009'
```





Example 2

A view consisting of **no ordered articles**.

```
1 create view NoOrderedArticles as
2 select artNum, artDesc
3 from Article
4 where artNum not in (select artNum from OrderDetails)
```

Find the number of **no ordered articles**.

```
1 select count(*)
2 from NoOrderedArticles
```





Example 3

This creates a "summary line" for each article, just listing their number along with the **ordered quantity** and the **number of times** this article has been ordered.

```
1 create view articleRecord as
2 select artNum, artDesc, Sum(ordQte) as orderedQte, count(*) as orderedNb
3 from Article, OrderDetails
4 where Article.artNum = OrderDetails.artNum
5 group by artNum
```

```
1 select * from articleRecord
```



Example 4



This creates a "summary line" for each order, just listing their number along with the **total price** and the **number of ordered articles**.

```
1 create view orderRecord as
2 select ordNum, Sum(artUP * ordQte) as totalPrice, SUM(ordQte) as totalQte
3 from Article, OrderDetails
4 where Article.artNum = OrderDetails.artNum
5 group by ordNum
```

```
1 select * from orderRecord
```



A new table



A view can do pretty much anything a normal table can:

```
1 SELECT ordDate, ordNum, totalPrice  
2 FROM Order inner join orderRecord on Order.ordNum=orderRecord.ordNum
```



Outline



- 1 View overview
- 2 Create views in TSQL
- 3 The with clause**
- 4 Updatable views



With Clause



- The **with clause** provides a way of defining a **temporary view** whose definition is available only to the query in which the with clause occurs.
- Find all **latest orders**

```
1 with latest_order(last_order) as
2     (select max(ordDate)
3      from Order)
4
5 select ordNum
6 from Order, latest_order
7 where Order.ordDate = latest_order.last_order
```



Complex Queries using With Clause



Find all orders where the **total ordered quantity is greater than the average of the total ordered quantity.**





Complex Queries using With Clause

Find all orders where the **total ordered quantity is greater than the average of the total ordered quantity**.

```
1 with order_total (ordNum, sum_qte) as (  
2     select ordNum, sum(ordQte)  
3     from OrderDetails  
4     group by ordNum),  
5 order_total_avg (avg_qte) as (  
6     select avg(sum_qte)  
7     from order_total)  
9 select ordNum  
10 from order_total, order_total_avg  
11 where order_total.sum_qte >= order_total_avg.avg_qte
```



Views: remarks



```
1 create table T(col1 int, col2 int);
2 insert into T(col1, col2) values (1, 2);

5 create view V
6 as
7 select * from T;

9 select * from V
```

```
1 col1  col2
2 ----  ----
3 1      2
```



Views: remarks



```
1 alter table T add col3 int;  
2 go  
4 select * from V
```

Still get only two columns in the output

```
1 col1    col2  
2 -----  
3 1        2
```



Views: remarks



To refresh the view's meta-data information, run the **sp_refreshview** stored procedure against V:

```
1 EXEC sp_refreshview 'V';
```

```
1 col1  col2  col3
2 ----  ----  ----
3 1      2    NULL
```



Outline



- 1 View overview
- 2 Create views in TSQL
- 3 The with clause
- 4 Updatable views**



More view terminology



■ Updatable views:

A view is updatable if you can use it to **alter the underlying tables**.

■ Materialized views:

A view is materialized if the results of the query it represents have actually **been stored in the database**.



Updatable views



A view is updatable if you can use DELECT, INSERT and UPDATE statements on the view and have the underlying tables altered in such a way that the view is changed in a manner consistent with the operation.

```
1 create view articlePrice as  
2 select artNum, artUP  
3 from Article
```

This just pulls **two fields** out of the "Article" table.



Updatable views



What should this statement do?

```
1 delete from articlePrice where artNum=12
```



Updatable views



What should this statement do?

```
1 delete from articlePrice where artNum=12
```

The only way to change the underlying tables so that the view no longer "sees" the article 12, is to delete those rows from article table and so this is what happens.



Not Updatable views

```
1 create view priceOrder as
2 select ordNum, SUM(artUP * ordQte) as totalprice
3 from OrderDetails, Article
4 where OrderDetails.artNum=Article.artNum
5 group by ordNum
```

```
1 update priceOrder set totalPrice = 1000 where ordNum=100
```

cannot be interpreted: there is **no unique way to change the unit price** for articles in order to get a **sum of 1000**.

Updatable views

Not all views can be updated. View is read-only if:

- DISTINCT is in the SELECT statement
- Statement contains expressions (averages, totals, etc.)
- Statement references views that are not updatable
- Statement has GROUP BY or HAVING clauses