USING AND DEFINING VIEWS

By using SQL queries with any RDBMS, it is also possible to create virtual tables, or dynamic views, whose contents materialize when referenced. These views may often be manipulated in the same way as a base table can be manipulated, through SQL SELECT queries. Materialized views, which are stored physically on a disk and refreshed at appropriate intervals or events, may also be used.

The often-stated purpose of a view is to simplify query commands, but a view may also improve data security and significantly enhance programming consistency and productivity for a database.

Essentially what a view helps us with is putting sql queries that will often be used to create certain outputted tables of data from base tables into containers that we can constantly call upon instead of writing large sql queries with multiple tables each time'

A view, Invoice_V, is defined by specifying an SQL query (SELECT . . . FROM . . . WHERE) that has the view as its result.

Query: What are the data elements necessary to create an invoice for a customer? Save this query as a view named Invoice_V.

```
CREATE VIEW Invoice_V AS
  SELECT Customer_T.CustomerID, CustomerAddress, Order_T.OrderID,
  Product_T.ProductID,ProductStandardPrice,
  OrderedQuantity, and other columns as required
    FROM Customer_T, Order_T, OrderLine_T, Product_T
      WHERE Customer_T.CustomerID = Order_T.CustomerID
        AND Order_T.OrderID = OrderLine_T.OrderD
        AND Product_T.ProductID = OrderLine_T.ProductID;
```

- The SELECT clause specifies, or projects, what data elements (columns) are to be included in the view table. The FROM clause lists the tables and views involved in the view development. The WHERE clause specifies the names of the common columns used to join Customer_T to Order_T to OrderLine_T to Product_T.

- You can see the power of such a view when building a query to generate an invoice for order number 1004. Rather than specify the joining of four tables, you can have the query include all relevant data elements from the view table, Invoice_V.

Query: What are the data elements necessary to create an invoice for order number 1004?

```
SELECT CustomerID, CustomerAddress, ProductID,
  OrderedQuantity, and other columns as required
    FROM Invoice_V
      WHERE OrderID = 1004;
```

Note:
- Very easy to write a query which would otherwise require large amounts of code writing

Pros and cons of views:

TABLE 6-1 Pros and Cons of Using Dynamic Views

Positive Aspects	Negative Aspects
Simplify query commands	Use processing time re-creating the view each time it is referenced
Help provide data security and confidentiality	May or may not be directly updateable
Improve programmer productivity	
Contain most current base table data	
Use little storage space	
Provide a customized view for a user	
Establish physical data independence	

Another example of the usage of a view:

A view may join together multiple tables or views and may contain derived (or virtual) columns. For example, if a user of the Pine Valley Furniture database only wants to know the total value of orders placed for each furniture product, a view for this can be created from Invoice_V. The following example illustrates how this is done with Oracle, although this can be done with any RDBMS that supports views.

Query: What is the total value of orders placed for each furniture product?

```
CREATE VIEW OrderTotals_V AS
  SELECT ProductID Product, SUM (ProductStandardPrice*OrderedQuantity)
  Total
    FROM Invoice_V
    GROUP BY ProductID;
```

Views offer security too:

- Views can also help establish security. Tables and columns that are not included will not be obvious to the user of the view. Restricting access to a view with GRANT and REVOKE statements adds another layer of security. For example, granting some users access rig hts to aggregated data, such as averages, in a view but denying them access to detailed base table data will not allow them to displ ay the base table data.

You can also update (using the commands INSERT, DELETE, UPDATE) a base table from a view as long as it is completely clear what data from the base table must change.

- If the view definition includes the WITH CHECK OPTION clause, attempts to insert data through the view will be rejected when the data values do not meet the specifications of WITH CHECK OPTION.
  - Specifically, when the CREATE VIEW statement contains any of the following situations, that view may not be used to update th e data:
    - 1. The SELECT clause includes the key word DISTINCT.
      2. The SELECT clause contains expressions, including derived columns, aggregates, statistical functions, and so on.
      3. The FROM clause, a subquery, or a UNION clause references more than one table.
      4. The FROM clause or a subquery references another view that is not updateable.
      5. The CREATE VIEW command contains a GROUP BY or HAVING clause.
  - Example:
    - Let's create a view named ExpensiveStuff_V, which lists all furniture products that have a StandardPrice over $300. That view will include ProductID 5, a writer's desk, which has a unit price of $325.
    - If you update data using Expensive_Stuff_V and reduce the unit price of the writer's desk to $295, then the writer's desk will no longer appear in the ExpensiveStuff_V virtual table because its unit price is now less than $300.
    - In Oracle, if you want to track all merchandise with an original price over $300, include a WITH CHECK OPTION clause after the SELECT clause in the CREATE VIEW command.
    - WITH CHECK OPTION will cause UPDATE or INSERT statements on that view to be rejected when those statements would cause updated or inserted rows to be removed from the view.
      - This option can be used only with updateable views.
    - Here is the CREATE VIEW statement for ExpensiveStuff_V.

Query: List all furniture products that have ever had a standard price over $300.

```
CREATE VIEW ExpensiveStuff_V
  AS
    SELECT ProductID, ProductDescription, ProductStandardPrice
      FROM Product_T
        WHERE ProductStandardPrice > 300
        WITH CHECK OPTION;
```

When attempting to update the unit price of the writer's desk to $295 using the Oracle syntax

```
UPDATE ExpensiveStuff_V
SET ProductStandardPrice = 295
  WHERE ProductID = 5;
```

Oracle gives the following error message: ERROR at line 1: ORA-01402: view WITH CHECK OPTION where-clause violation

- But if you wanted to increase the price to say 350 it would allow it since it doesn't violate the condition

So far we have called everything views but really they are called dynamic views which are a little different from materialized views:

Materialized Views

Like dynamic views, materialized views can be constructed in different ways for various purposes. Tables may be replicated in whole or in part and refreshed on a predetermined time interval or triggered when the table needs to be accessed. Materialized views can be based on queries from one or more tables. It is possible to create summary tables based on aggregations of data. Copies of remote data that use distributed data may be stored locally as materialized views. Maintenance overhead will be incurred to keep the local view synchronized with the remote base tables or data warehouse, but the use of materialized views may improve the performance of distributed queries, especially if the data in the materialized view are relatively static and do not have to be refreshed very often.