Partitioned views allow the data in a large table to be split into smaller member tables. The data is partitioned between the member tables based on ranges of data values in one of the columns. The data ranges for each member table are defined in a CHECK constraint specified on the partitioning column. A view that uses UNION ALL to combine selects of all the member tables into a single result set is then defined. When SELECT statements referencing the view specify a search condition on the partition column, the query optimizer uses the CHECK constraint definitions to determine which member table contains the rows.

|  |
| --- |
| **ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/note.gifNote:** |
| The preferred method for partitioning data local to one server is through partitioned tables. For more information, see Partitioned Tables and Indexes. |

For example, a sales table that records sales for 1998 has been partitioned into 12 member tables, one for each month. Each member table has a constraint defined on the OrderMonth column:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| CREATE TABLE May1998sales  (OrderID INT,  CustomerID INT NOT NULL,  OrderDate DATETIME NULL  CHECK (DATEPART(yy, OrderDate) = 1998),  OrderMonth INT  CHECK (OrderMonth = 5),  DeliveryDate DATETIME NULL  CHECK(DATEPART(mm, DeliveryDate) = 5)  CONSTRAINT OrderIDMonth PRIMARY KEY(OrderID, OrderMonth)  ) | |

The application that populates May1998sales must make sure that all rows have 5 in the OrderMonth column and that the order date specifies a date in May 1998. This is enforced by the constraints defined on the table.

A view is then defined that uses UNION ALL to select the data from all 12 member tables as a single result set:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| CREATE VIEW Year1998Sales  AS  SELECT \* FROM Jan1998Sales  UNION ALL  SELECT \* FROM Feb1998Sales  UNION ALL  SELECT \* FROM Mar1998Sales  UNION ALL  SELECT \* FROM Apr1998Sales  UNION ALL  SELECT \* FROM May1998Sales  UNION ALL  SELECT \* FROM Jun1998Sales  UNION ALL  SELECT \* FROM Jul1998Sales  UNION ALL  SELECT \* FROM Aug1998Sales  UNION ALL  SELECT \* FROM Sep1998Sales  UNION ALL  SELECT \* FROM Oct1998Sales  UNION ALL  SELECT \* FROM Nov1998Sales  UNION ALL  SELECT \* FROM Dec1998Sales | |

For example, the following SELECT statement queries for information about specific months.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| SELECT \*  FROM Year1998Sales  WHERE OrderMonth IN (5,6) AND CustomerID = 64892 | |

The SQL Server query optimizer recognizes that the search condition in this SELECT statement references only rows in the May1998Sales and Jun1998Sales tables. Therefore, it limits its search to those tables.

To perform updates on a partitioned view, the partitioning column must be a part of the primary key of the base table. If a view is not updatable, you can create an INSTEAD OF trigger on the view that allows updates. You should design error handling into the trigger to make sure that no duplicate rows are inserted. For an example of an INSTEAD OF trigger designed on a view, see Designing INSTEAD OF Triggers.

CHECK constraints are not needed for the partitioned view to return the correct results. However, if the CHECK constraints have not been defined, the query optimizer must search all the tables instead of only those that cover the search condition on the partitioning column. Without the CHECK constraints, the view operates like any other view with UNION ALL. The query optimizer cannot make any assumptions about the values stored in different tables and it cannot skip searching the tables that participate in the view definition.

If all the member tables referenced by a partitioned view are on the same server, the view is a local partitioned view. If the member tables are on multiple servers, the view is a distributed partitioned view. Distributed partitioned views can be used to spread the database processing load of a system across a group of servers. For more information, see Federated Database Servers.

Partitioned views make it easier to maintain the member tables independently. For example, you can do the following at the end of a period:

* The definition of the partitioned view for current results can be changed to add the newest period and drop the oldest period.
* The definition of the partitioned view for past results can be changed to add the period just dropped from the current results view. The past results view can also be updated to remove and archive the oldest period it covers.

When you insert data into the partitioned views, the **sp\_executesql** system stored procedure can be used to create INSERT statements with execution plans that have a significant chance of being reused in systems with many concurrent users.