Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

In SQL Server, **table-level compression** refers to the process of applying compression on a table's data to reduce storage space usage and improve I/O performance. This is done through **Row-Level Compression** or **Page-Level Compression**, which are the two types of compression that can be applied at the table level. Here's a breakdown of how **table-level compression** works in SQL Server, along with examples for both **Row-Level Compression** and **Page-Level Compression**.

1. Row-Level Compression

Row-level compression reduces storage by optimizing the way fixed-length data types are stored. This technique is more efficient for tables with a high number of rows, especially when the data doesn't fit well into the typical structure of fixed-length types (e.g., INT, CHAR).

How Row-Level Compression Works:

- **Fixed-Length Data Types:** Fixed-length data types such as CHAR, NCHAR, INT, BIGINT are compressed to variable-length formats.
- **Null Handling:** Null values are handled more efficiently, using a bitmap for each column to track whether the value is null.
- **Variable-Length Data Types:** Data types like VARCHAR and NVARCHAR are compressed by removing unnecessary padding and reducing storage overhead.

Example: Enabling Row-Level Compression on a Table

Let's create a sample table and enable row-level compression on it.

```
-- Create a table without any compression
CREATE TABLE Employees (
    EmployeeID INT,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Salary INT,
    DateOfJoining DATE
);
-- Insert some sample data
INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, DateOfJoining)
VALUES (1, 'John', 'Doe', 50000, '2010-01-01'),
       (2, 'Jane', 'Smith', 60000, '2012-05-15'),
       (3, 'Bob', 'Johnson', 70000, '2014-08-10');
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

```
-- Enabling Row-Level Compression on the Employees table
ALTER TABLE Employees
    REBUILD WITH (DATA_COMPRESSION = ROW);
```

In this example:

- We create a table called Employees and insert some sample data.
- The ALTER TABLE ... REBUILD WITH (DATA_COMPRESSION = ROW) command enables **row-level compression** for the Employees table.

2. Page-Level Compression

Page-level compression is a more advanced compression technique that compresses data at the page level (SQL Server uses 8KB pages to store data). Page-level compression uses multiple techniques like **prefix compression**, **dictionary compression**, and **row-level compression** to achieve better compression ratios compared to row-level compression.

How Page-Level Compression Works:

- **Prefix Compression:** Reduces redundant prefixes across rows in a page, storing the common prefix only once.
- **Dictionary Compression:** Creates a dictionary for common values across rows on the same page, which are replaced with shorter references to the dictionary.
- **Row-Level Compression:** Applied in conjunction with the other techniques to further reduce row size.

Example: Enabling Page-Level Compression on a Table

Now, let's enable **page-level compression** on the same table.

```
-- Enabling Page-Level Compression on the Employees table
ALTER TABLE Employees
    REBUILD WITH (DATA_COMPRESSION = PAGE);
```

In this example:

- The ALTER TABLE ... REBUILD WITH (DATA_COMPRESSION = PAGE) command enables **page-level compression** on the Employees table.

Checking the Compression Type of a Table

Once you have applied compression, you can check the compression type on the table using the sys.partitions system view.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

```
-- Query to check the compression type of the Employees table
SELECT
    t.name AS TableName,
    i.name AS IndexName,
    p.data_compression_desc AS CompressionType
FROM
    sys.tables t
JOIN
    sys.indexes i ON t.object_id = i.object_id
JOIN
    sys.partitions p ON i.object_id = p.object_id AND i.index_id = p.index_id
WHERE
    t.name = 'Employees';
```

This query will show the compression type (either ROW or PAGE) applied to the table or indexes.

Compression Benefits and Considerations:

- **Reduced Storage:** Compression helps save storage space, which is particularly beneficial for large tables or databases with limited disk space.
- **Improved I/O Performance:** With less data being read from disk (due to smaller storage size), I/O performance can be improved.
- **CPU Overhead:** Compression comes with a slight CPU overhead, as the data has to be compressed during write operations and decompressed during reads. The extent of this overhead depends on the workload.
- **Suitability:** Row-level compression works best for tables with a high number of small rows, while page-level compression is better for large tables with many repeated values.

When to Use Row-Level vs Page-Level Compression:

- **Row-Level Compression** is ideal when:
  - There are many small rows with fixed-length data.
  - There is a large number of small transactions or operations with frequent inserts and deletes.
  - The CPU overhead for compressing/decompressing is acceptable.
- **Page-Level Compression** is ideal when:
  - There is a large amount of repetitive data (e.g., in data warehouses or analytical databases).
  - There are long-running queries that scan large portions of a table.
  - The benefit of reducing I/O outweighs the additional CPU overhead.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

Summary:

Table-level compression in SQL Server can be implemented using either **row-level compression** or **page-level compression**. These techniques reduce the amount of storage space required for tables, improve I/O performance, and can help manage large datasets more efficiently. By choosing the appropriate compression method based on your workload, you can achieve significant performance improvements and storage savings.