https://www.sqldbachamps.com

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

SQL Server offers various compression techniques to reduce the storage footprint of data and improve performance by reducing I/O operations. Here is a detailed explanation of the types of compression in SQL Server, along with examples of how to implement them.

1. Row-Level Compression

Row-level compression reduces the amount of storage required for each row in a table. This technique compresses the data at the row level, optimizing the storage of fixed-length columns.

How Row-Level Compression Works:

- **Fixed-Length Data Types:** Fixed-length data types such as INT, BIGINT, CHAR, and NCHAR are converted to variable-length storage formats to save space.
- **Null Handling:** Null values are stored more efficiently by using a bitmap to track which columns are null.
- **Variable-Length Columns:** Variable-length data types like VARCHAR and NVARCHAR have reduced overhead because unnecessary padding is eliminated.

Example of Enabling Row-Level Compression:

```
-- Create a sample table without compression
CREATE TABLE Employees (
    EmployeeID INT,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    BirthDate DATE
);

-- Enabling row-level compression on the table
ALTER TABLE Employees
    REBUILD WITH (DATA_COMPRESSION = ROW);
```

In this example, the Employees table is compressed using row-level compression. This will save space by compressing the data at the row level, reducing storage for fixed-length data types.

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

2. Page-Level Compression

Page-level compression works by compressing data at the page level (8KB pages in SQL Server). It uses techniques like prefix compression, dictionary compression, and row-level compression to achieve better space savings compared to row-level compression.

How Page-Level Compression Works:

- **Prefix Compression:** Reduces repetitive data by storing the common prefix once and referencing it for each repeated occurrence.
- **Dictionary Compression:** Creates a dictionary for commonly occurring values within a page and replaces repeated values with references to the dictionary.
- **Row-Level Compression:** Applied after prefix and dictionary compression to further reduce the row size.

Example of Enabling Page-Level Compression:

```
-- Enabling page-level compression on a table
ALTER TABLE Employees
   REBUILD WITH (DATA_COMPRESSION = PAGE);
```

In this example, the Employees table is compressed using page-level compression. This is often more effective than row-level compression for large tables where many values are repeated across multiple rows.

3. Columnstore Compression

Columnstore compression is used when creating columnstore indexes, which are highly optimized for analytic workloads (OLAP). It stores data by column instead of row, which allows for better compression and faster query performance for certain types of queries, particularly large scans over many rows.

How Columnstore Compression Works:

- **Delta Encoding:** Stores the difference between consecutive values instead of the values themselves.
- **Run-Length Encoding:** Stores repeated values as a single value with a count of how many times it repeats.
- **Bitmap Compression:** Applies to columns with low cardinality (few distinct values), replacing repeated values with a bitmap.

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

Example of Creating a Columnstore Index:

```
-- Creating a columnstore index on a table
CREATE CLUSTERED COLUMNSTORE INDEX idx_Employees_Columnstore
   ON Employees (EmployeeID, FirstName, LastName, BirthDate);
```

In this example, a clustered columnstore index is created on the Employees table. This allows SQL Server to compress the data significantly and improve query performance, especially for read-heavy analytical queries.

4. Backup Compression

Backup compression reduces the size of the backup files, which helps to save disk space and reduce I/O during backup and restore operations.

How Backup Compression Works:

- SQL Server uses compression algorithms such as zlib to compress data during the backup process, resulting in smaller backup files.
- This compression can be applied to full, differential, and transaction log backups.

Example of Enabling Backup Compression:

```
-- Enabling backup compression for a full database backup
BACKUP DATABASE [MyDatabase]
   TO DISK = 'C:\Backups\MyDatabase.bak'
   WITH COMPRESSION;
```

In this example, the WITH COMPRESSION option is used to create a compressed backup of the MyDatabase database. This reduces the size of the backup and speeds up the process.

5. Transparent Data Encryption (TDE) with Compression

While TDE itself is not a compression technique, it can be used in conjunction with data compression techniques. TDE encrypts data at the storage level, while compression reduces the data's size before encryption. This is especially useful when dealing with large amounts of sensitive data.

Example of Using TDE and Compression Together:

```
-- Enable Transparent Data Encryption (TDE) on a database
CREATE DATABASE MyEncryptedDB
   ENCRYPTION ON;
-- Enabling compression on a table after TDE is enabled
ALTER TABLE Employees
   REBUILD WITH (DATA_COMPRESSION = PAGE);
```

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

In this example, TDE is enabled on the MyEncryptedDB database to encrypt data, and then page-level compression is applied to the Employees table to reduce storage space before the data is encrypted.

Compression and Performance Considerations

- **Storage Savings:** Compression techniques, especially page-level and columnstore compression, can significantly reduce disk space usage, sometimes by up to 90% in certain cases.
- **CPU Overhead:** Compression adds CPU overhead during both read and write operations, but the trade-off in terms of space savings and reduced I/O can make it worthwhile for large tables or heavy read workloads.
- **Workload Type:** OLTP (Online Transaction Processing) systems generally benefit from row-level compression, while OLAP (Online Analytical Processing) systems and data warehouses benefit more from page-level and columnstore compression.

Conclusion

SQL Server provides several powerful compression techniques to help optimize storage and improve performance:

- **Row-Level Compression:** Best suited for OLTP workloads where many small rows are stored.
- **Page-Level Compression:** Ideal for large datasets, especially in data warehouses where rows have repeated values.
- **Columnstore Compression:** Excellent for analytical workloads, as it compresses data by column and significantly speeds up read-heavy operations.
- **Backup Compression:** Reduces the size of backups and improves I/O during backup and restore processes.
- **TDE with Compression:** Encrypts data while also reducing the storage footprint before encryption.

By choosing the appropriate compression method based on your workload, you can significantly improve both storage efficiency and query performance in SQL Server.