Compression

Data compression

- Compression is used to reduce the storage used by the tables.
- Compression happens at *column* level.
- Teradata offers some Technique to compress the data:
    - Multi Value Compression(MVC)
    - Algorithmic Compression(ALC)
    - Block Level Compression(BLC)

Data you can compress

- Any numeric data type
- Nulls, zeros, blanks
- `DATE`
- Up to 255 `char` / `varchar` distinct values.

Data you can't compress

- Primary index column(s)
- Identity columns
- Volatile tables
- Derived tables
- BLOB, CLOB

MVC

- MVC is a logical data compression form and is lossless.
- It can compress up to 255 distinct values including `NULL`.
- Can be added at table creation using `CREATE TABLE` or after table creation using `ALTER TABLE`.
- When compression is applied on a column, the values for this column are not stored with the row. Instead the values are stored in the Table header in each AMP and only presence bits are added to the row to indicate the value.
- **No overhead**

MVC

- Usually the best cost/benefit ratio compared to other methods.
- It requires minimal resources to uncompress the data during query processing, you can use MVC for hot (frequently used) data without compromising query/load performance.
- MVC is also considered the easiest to implement of all the compression methods.
- Besides storage capacity and disk I/O size improvements, MVC has the following performance impacts:
    - Improves table scan response times for most configurations and workloads
    - Provides moderate to little CPU savings

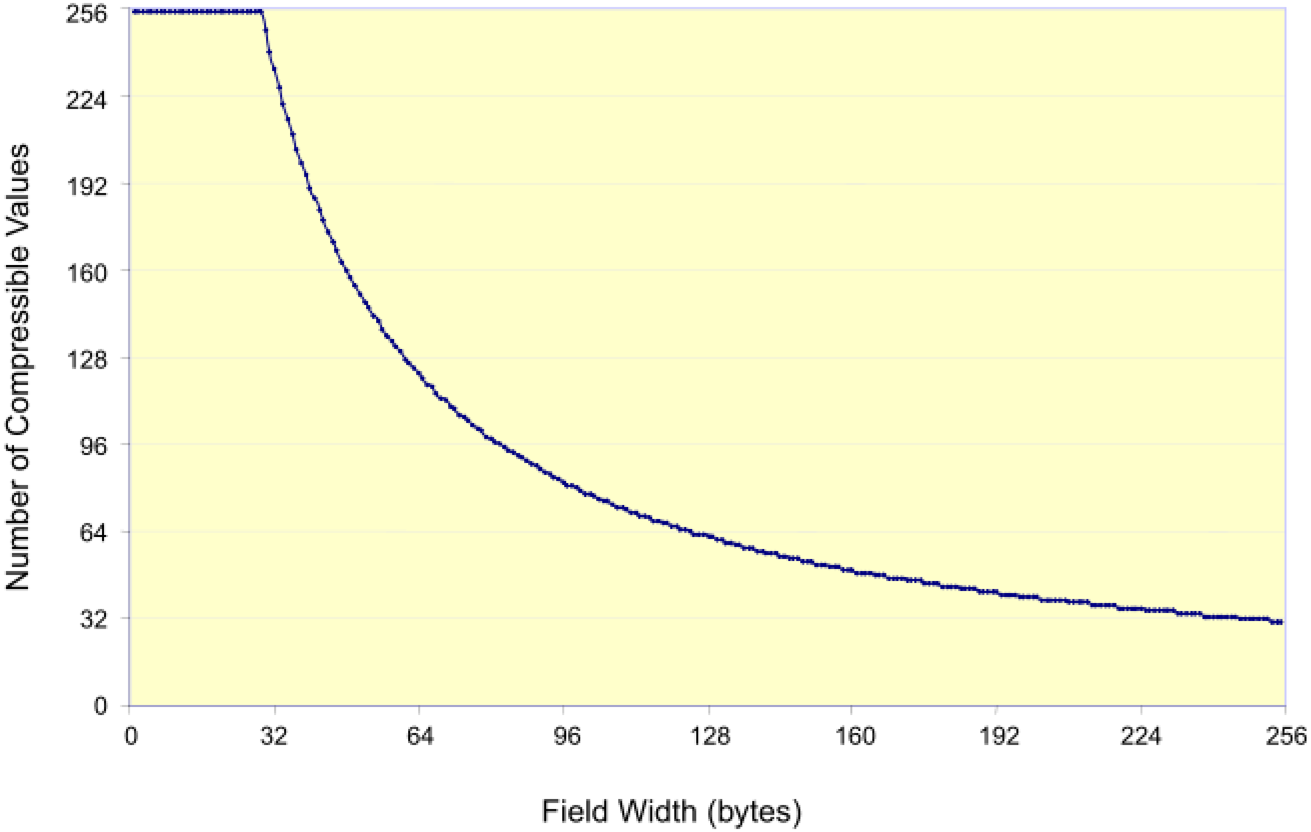
Example

```
CREATE TABLE employee (
   employee_number INTEGER
   ...
   jobtitle        CHARACTER(30) COMPRESS (’cashier’,
                   ’manager’, ’programmer’)
   ...
   );
```

- **Trivia:** how much space is saved (per row)?

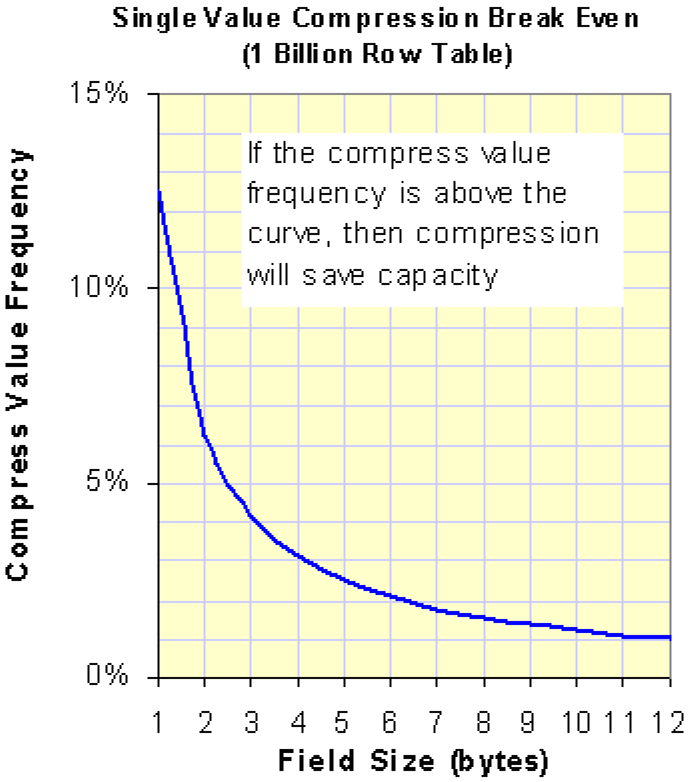
Gotcha

- Table header size is limited to 1MB: compressing many wide columns eats up the space fast.



When is compression worth it?

- Break-even curve



ALC

- Compression using UDFs instead of simple one-hot encoding.
- When column values are mostly unique, algorithmic compression (ALC) may provide better compression results than MVC.
- ALC and MVC can be used concurrently on the same column, but not on the same values.
- There is overhead involved to compress and decompress data in ALC.
- Recommended for large character columns which are not accessed often.

Example

```
CREATE TABLE Student
    (Roll_No      INTEGER,
    Student_Name VARCHAR(50),
    Student_Address CHAR(200) CHARACTER SET UNICODE
    COMPRESS USING TransUnicodeToUTF8
    DECOMPRESS USING TransUTF8ToUnicode)
UNIQUE PRIMARY INDEX(Roll_No);
```

BLC

- Compress the table by blocks of rows.
- To access a row, the entire block needs to be decompressed.
- Does not carry over across sessions.
- In case of MVC and ALC values need to be define in `CREATE TABLE` statement, but BLC is activated outside of table definition.
- More space reduction compared to MVC and ALC (up to 60%).
- Recommended mostly for *cold data* as there is overhead in compressing/decompressing.

Example

BLC can be applied in two ways:

- For an empty table query band can be used to apply BLC.

```
/* Turn on BLC */
SET QUERY_BAND = 'BLOCKCOMPRESSION=YES;' FOR SESSION;
/* Insert data into empty table */
INSERT INTO STUDENT_MVC AS SELECT * FROM STUDENT;
/* Turn off BLC */
SET QUERY_BAND = 'BLOCKCOMPRESSION=NO;' FOR SESSION;
```

- For a non-empty table, **Ferret** utility can be used to either compress all the data block in its or to decompress it.

Compression and Query Performance

- The optimizer evaluates the relative cost of many potential execution plans and picks a **low cost** plan.
- One of the costs considered is the number of estimated I/O operations needed to execute a plan.
- The Optimizer will take advantage of the compressed structure.
- **Great use of freed space**: add indices or pre-joined aggregate summaries to speed-up queries.

Useful queries

- You can find all compressed columns:

```
SELECT * FROM dbc.ColumnsV
WHERE CompressValueList IS NOT NULL
```

- Uncompressed tables

```
SELECT * FROM dbc.TablesV
WHERE TableKind IN ('T', 'O') -- both PI and NoPI tables
AND (DatabaseName, TableName) NOT IN
 (
   SELECT DatabaseName, TableName
   FROM dbc.columnsv
   WHERE CompressValueList IS NOT NULL
 )
```

Exercise

- In the `listings` table, apply compression to the `neighbourhood_group` and `room_type` columns. Compare the table sizes before and after.
- Apply compression on the exercise of lecture `02 Indexes`. Does this help to improve your query?