



Case Study Report

Compression of TPC-H Lineitem Table

May 22, 2010

1. OVERVIEW OF METHODOLOGY

This summarizes the methodology used to design compression methods for the TPC-H lineitem table.

1.1. Project Objective

The project objective was to develop the methods needed to compress TPC-H lineitem files to the maximum extent possible without resorting to extraordinary measures; i.e., reverse-engineering or emulating the software process that creates the files.

1.2. Preparation

TPC-H specification documents tpch2.9.0.doc and tpch2.9.0.pdf were downloaded from the website www.tpc.org, as was software in the file tpch_2_8_0.zip. File specifications were reviewed. Database tables for scale factors (SF) of 1, 10, 30, and 100 were generated after modifying source code for DBGEN, their population generator. A small fifth table, called SF0, was constructed from the final 100,000 records of the SF100 table for use in development, testing, and optimization.

1.3. Statistical Analysis

1.3.1. Phase 1: Statistics of field values

A data analyzer was constructed from existing statistical analysis software and assembled with new file-specific parsing software. It was run on all lineitem table files beginning with SF0, and then further developed in tandem with the compression software. The following field-oriented file statistics were measured and eventually retained:

- Field offset (post-parsing)
- Cardinality (maximum 2560)
- Field class
- Alphabet size
- Character entropy
- Minimum field length
- Maximum field length
- Minimum value
- Maximum value
- Mean value (except fields in upper case field class)
- Standard deviation (except fields in upper case field class)
- Largest probability
- Entropy
- Average code length (ACL) of Huffman codes
- Excess bits (redundancy or difference between ACL and entropy)

1.3.2. Phase 2: Statistics of derived functions of field values

A number of functions of field values were defined from review of the field value statistics, and, following enhancements to the statistical analysis software, most of the same statistics were measured and eventually retained. The derived functions were as follows:

- Hard separator codes (for comment field)
- Algebraic combinations of commit date, receipt date, and/or ship date (3 total)

- An algebraic function of extended price and quantity
- Order key value difference from previous record
- Comment field separator count
- Comment field word count
- Tuple 0 (for attribute vector coding of fields 0, 3, 8, 9, & 10)
- Tuple 15 (for comment field)
- Backset (for comment field)
- Overhang (for comment field)
- Part key symbol group number
- Supp key descriptor
- Unit price symbol group number

1.3.3 Phase 3: Frequency domain analysis and attribute vector coding

Code was added to generate .wav files containing integers computed from field values and derived functions. Resulting wave files were analyzed by using standard frequency-domain tools and by listening. From that analysis, the decision was made to compress fields 0, 3, 8, 9, and 10, but no others, using attribute vector coding. To that end, further derived functions were defined and their statistics were measured, and from that information the predictor context, expressors, and coefficients were defined.

1.3.4. Phase 4: Root word probabilities

A 208-by-208-element transition probability matrix was created for the root words in the comment field. Following its review, the decision was made to use first-order word code probabilities despite the fact that the actual generation process was syntactically more complex and therefore of higher order. A major factor was the opportunity to use zero-bit canonical Huffman codes, which provide infinite compression and are fast to decompress. The compression software was enhanced to generate 207 codebook/symbol array sets for root words instead of one set.

1.3.5. Phase 5: Backset and overhang probabilities

An analysis was performed on a sample of records to ascertain the dependence of backset and overhang on word length, and following its review the decision was made to use first-order backset and overhang probabilities conditioned on word length. The compression software was then enhanced to generate 21 additional codebook/symbol array sets.

Following that, a second analysis was performed on a sample of records to ascertain the dependence of backset and overhang on word code, and following its review and further analysis of the word code transition probabilities the decision was made to condition the backset and overhang probabilities on word code instead of word length. The compression software was then enhanced to generate 416 codebook/symbol array sets instead of 23.

1.3.6. Phase 6: Block codes for selected symbol combinations

Three more derived functions were defined, all with the objectives of increasing compression and reducing decompression time. One (tuple 3) was the combination of domain codes for three flat distributions (fields 7, 13, and 14), the product of whose cardinalities, 252, made the combination suitable for flat 8-bit coding. The second (tuple 2) was a combination of domain codes that was not suitable for flat coding (fields 4 and 6) but nevertheless still suitable for block coding followed by canonical Huffman coding. The third was a combination of two uncorrelated derived values (functions of fields 1 and 5) whose Huffman codes were relatively inefficient; block coding them yielded better compression and faster decompression.

1.4. Final Determination of Coding Algorithms for Individual Fields

Fields 0 (order key), 3 (line number), 8 (return flag), 9 (line status), and 10 (ship date) were compressed by attribute vector coding with 4, 3, 3, 2, and 3 expressors, respectively, and one coefficient for each attribute.

Field 1 (part key) was compressed by integer transformation, followed by splitting by significance, followed by block coding (with field 5), followed by canonical Huffman coding of the block.

Field 2 (supp key) was compressed by character-wise comparison with field 1, followed by integer transformation and decomposition into a descriptor and a residual, followed by residual bit reduction, followed by canonical Huffman coding of the descriptor and bit concatenation with the reduced residual.

Both field 4 (quantity) and field 6 (discount) were compressed by domain coding followed by canonical Huffman coding.

Field 5 (extended price) was compressed by integer transformation, followed by algebraic operation (with field 4), followed by splitting of the result by significance. The most significant part was block-coded with data from field 1; the block was canonical Huffman coded. The least significant part (residual) was flat-coded with 13 bits.

Fields 7 (tax), 13 (ship instructions) and 14 (ship mode) were compressed by block coding the domain codes and flat coding the resulting compound 8-bit symbol.

Both field 11 (commit date) and field 12 (receipt date) were compressed by integer transformation (with field 10), followed by algebraic operation, followed by canonical Huffman coding of the result.

Field 15 was compressed by a new, hybrid method that included the following: parsing; syntactic separation; definition of an entropy-reduced but functionally equivalent generative data model; superimposition of syntactic elements; descriptor decomposition; bit mapping; and multiple instances of canonical Huffman coding.

1.5. Results

Compression ratios of 8.40, 8.56, 8.44, and 8.53 were obtained at scale factors of 1, 10, 30, and 100, respectively.

2. SPECIFICATION OF COMPRESSED BINARY FILE

This describes the result of the methodology described in Section 1: the form and content of the compressed binary file. When specified, bit lengths of canonical Huffman codes apply to the SF100 file.

2.1. First Level of Organization

The compressed binary file consists of four byte-concatenated parts: a 5,330-byte file header, a variable-byte-length codebook area, a variable-byte-length symbol area, and a variable-bit-length record area. They are stored in that order from lower to higher memory addresses.

2.1.1 File header

The file header contains 11 fixed-length fields and is stored at the beginning of the file.

2.1.2 Codebook area

The codebook area contains 650 or fewer variable-byte-length codebook data structures, each identified by a codebook number in the range 0 to 649 (inclusive) and stored in ascending codebook number order.

2.1.3 Symbol area

The symbol area contains 650 or fewer variable-byte-length symbol array data structures, each 1 or 2 bytes in width, identified by a codebook number in the range 0 to 649, associated with the codebook having the same codebook number, and stored in ascending codebook number order.

2.1.4 Record area

The record area contains one variable-bit-length compressed binary record for each uncompressed table record in the uncompressed table file from which the file was made, in the same order. Individual records are bit concatenated. Padding to the next byte boundary follows the final record in the file.

2.2. Second Level of Organization

2.2.1. The file header data structure

The file header contains the following fields stored in order of increasing memory address:

1. File size, a 64-bit unsigned integer specifying the number of bytes in the file.
2. Original file size, a 64-bit unsigned integer specifying the number of bytes in the uncompressed table file from which the file was made.
3. Record count, a 64-bit unsigned integer specifying the number of compressed binary records in the file.
4. Code mode, an 8-bit unsigned integer used for code development whose value is undefined but nominally zero.
5. Root word cardinality, an 8-bit unsigned integer specifying the number of root words in the lexicon.
6. Codebook area offset, a 32-bit unsigned integer specifying the byte address of the first codebook

relative to the beginning of the file.

7. Symbol area offset, a 16-bit unsigned integer specifying the byte address of the first symbol array relative to the beginning of the file.
8. Record area offset, a 32-bit unsigned integer specifying the byte address of the first compressed binary record relative to the beginning of the file.
9. The word length array, an array of 208 8-bit unsigned integers. Each array location except the last corresponds to a word code and a lexicon entry, and contains the character length of the root word stored in the entry. The last location is set to zero.
10. The lexicon, an array of 207 12-byte entries, each corresponding to a word code and containing one root word. Entries are space-padded to the right and stored in order of ascending word code, which corresponds to descending root word frequency.
11. The offset table, an array of 651 4-byte locations, each identified by a codebook number in the range 0 to 650, and stored in ascending codebook number order. Each array location except the last is associated with the codebook having the same codebook number. Each table location consists of 2 subfields: a 2-byte codebook offset field, and a 2-byte symbol array offset field. The codebook offset field is stored at the end of the location toward lower memory addresses.
 - A. The codebook offset field contains a 16-bit unsigned integer specifying the byte address of the codebook, if present, for the codebook number relative to the beginning of the codebook area.
 - B. The symbol array offset field contains a 16-bit unsigned integer specifying the byte address of the symbol array, if present, for the codebook number relative to the beginning of the symbol array area.

2.2.2. The codebook data structure

Each codebook contains 3 or 5 fields:

1. Minimum bit length, a 4- or 8-bit unsigned integer specifying the bit length of the shortest assigned codeword.
2. Maximum bit length, a 4- or 8-bit unsigned integer specifying the bit length of the longest assigned codeword.
3. Cardinality, an 8- or 16-bit unsigned integer specifying the number of symbols in the associated symbol array.
4. In codebooks for which there are two or more codeword lengths, the codeword value base array, an array of 0 to 26 unsigned 8-, 16-, or 32-bit integers, is present. Each array location corresponds to one codeword length and holds the zero-extension of the numerically lowest codeword for the length. The first array location corresponds to a codeword length one bit longer than minimum bit length; the last array location corresponds to maximum bit length. Codebooks for which maximum bit length equals minimum bit length contain no base array.
5. In codebooks for which there are two or more codeword lengths, the symbol ordinal base array, an array of 0 to 26 unsigned 8- or 16-bit integers, is present. Each array location corresponds to one codeword length and holds the numerically lowest symbol ordinal for the length.

2.2.3. The symbol array data structure

Each symbol array is an array of zero or more 8- or 16-bit locations. Each list except the first corresponds to the codebook having the same codebook number. Each list location holds a symbol that corresponds to the symbol ordinal with which it is indexed. The first location in the structure is an exception; it has no corresponding codebook and contains the 16-bit symbols for tuple code 3.

2.2.4. The compressed binary record data structure

The compressed binary record is the bit-concatenation of the following elements, each associated with one or more record fields, which are numbered 0-15:

1. Tuple code 0, a 1- to 17-bit canonical Huffman code holding partial or full information for record fields 0, 3, 8, 9, and 10.
2. Tuple code 1, a 6- to 11-bit canonical Huffman code holding partial information for record fields 1 and 5.
3. Tuple code 2, a 9- to 10-bit canonical Huffman code holding all information for record fields 4 and 6.
4. Tuple code 3, an 8-bit code holding all information for record fields 7, 13, and 14.
5. Field 0, containing a variable number of bits related to tuple code 0 and record field 0. That number may be zero.
6. Field 1, containing 21 bits related to tuple code 1 and record field 1.
7. Field 2, containing a 2- to 23-bit canonical Huffman code followed by a variable number of bits related to record field 2. That number may be zero.
8. Field 3, containing a variable number of bits related to tuple code 0 and record field 3. That number may be zero.
9. Field 5, containing 13 bits related to tuple code 1 and record field 5.
10. Field 8, containing a variable number of bits related to tuple code 0 and record field 8. That number may be zero.
11. Field 9, containing a variable number of bits related to tuple code 0 and record field 9. That number may be zero.
12. Field 10, containing a 7- to 24-bit canonical Huffman code related to tuple code 0 and record field 10.
13. Field 11, containing a 7- to 13-bit canonical Huffman code related to record field 11.
14. Field 12, containing a 4- to 5-bit canonical Huffman code related to record field 12.
15. Field 15, containing a variable number of bits related to record field 15.

2.3. Third Level of Organization (for Comment field)

The compressed binary record's comment field, field 15, has further structure. It consists of the bit-concatenation of 3, 4, or 5 variable-bit-length subfields, as described below.

2.3.1. Tuple code 15

Tuple code 15 is the 3- to 24-bit maximum-length-constrained modified canonical Huffman code for tuple 15, a 16-bit symbol with 5 fixed bit subfields:

1. Word count: bits 0-3, which directly specify the number of word codes in the range 1 to 10.
2. Separator first flag: bit 4, which when set to 1 indicates the first field character is part of a separator sequence rather than part of a word.
3. Separator last flag: bit 5, which when set to 1 indicates the last field character is part of a separator sequence rather than part of a word.
4. Not periods only flag: bit 6, which is set to 1 if at least one hard separator is present in the field that is not the period.
5. Separator bit map: bits 7-15, each of which corresponds to a separator sequence position beginning at the left end of the field and progressing toward higher memory addresses. Each bit is set to 1 if the separator sequence in the corresponding position contains a hard separator, and to 0 if it is a space.

2.3.2. Hard separator code(s)

When tuple 15's not periods only flag is set to 1, one or more hard separator codes are bit concatenated in the order of their corresponding separator sequences. Each hard separator code is the canonical Huffman code for one of eight 2- or 3-character sequences:

Hard separator sequence 1:	Exclamation point followed by space character
Hard separator sequence 2:	Colon followed by space character
Hard separator sequence 3:	Semicolon followed by space character
Hard separator sequence 4:	Comma followed by space character
Hard separator sequence 5:	Hyphen followed by space character
Hard separator sequence 6:	Period followed by space character
Hard separator sequence 7:	Question mark followed by space character
Hard separator sequence 8:	Double hyphen followed by space character

2.3.3. Word code(s)

One to ten 0- to 22-bit canonical Huffman codes are bit concatenated in the order their corresponding root words appear in the field. Each specifies one lexicon entry and root word.

2.3.4. Backset code

When tuple 15's separator first flag is set to 0, a 0- to 4-bit canonical Huffman code is encoded. It specifies, indirectly through the corresponding symbol array, the number of leading characters to be removed from the first word in the field.

2.3.5. Overhang code

When tuple 15's separator last flag is set to 1, a 1-bit code is encoded; otherwise, a 0- to 4-bit canonical Huffman code is encoded. The overhang code specifies, indirectly through the corresponding symbol array, the number of trailing characters to be removed from the last separator or word in the field.

3. SELECTED FILE STATISTICS

3.1 Statistics of Field Values in Uncompressed Table File for SF100

Field #	Field Name	Field Offset	Cardinality	Field Class	Alphabet Size	Char. Entropy	Lengths Min	Lengths Max
0	Order key	0	2600	Integer	10	3.3	1	9
1	Part key	9	2600	Integer	10	3.3	1	8
2	Supp key	17	2600	Integer	10	3.3	1	7
3	Line number	24	7	Integer	7	2.6	1	1
4	Quantity	25	50	Integer	10	3.1	1	2
5	Extended price	27	2600	Decimal	11	3.4	6	9
6	Discount	36	11	Decimal	11	2.2	4	4
7	Tax	40	9	Decimal	10	2.1	4	4
8	Return flag	44	3	Upper case	3	1.5	1	1
9	Line status	45	2	Upper case	2	1.0	1	1
10	Ship date	46	2526	Date	11	3.0	10	10
11	Commit date	56	2466	Date	11	3.0	10	10
12	Receipt date	66	2555	Date	11	3.0	10	10
13	Ship instruct	76	4	Upper case	17	3.8	4	17
14	Ship mode	93	7	Upper case	18	3.8	3	7
15	Comment	100	2600	General	35	4.3	10	43

Field #	Field Name	Minimum Value	Maximum Value	Mean Value
0	Order key	000000001	600000000	300008342
1	Part key	00000001	20000000	10000074
2	Supp key	0000001	1000000	500003
3	Line number	1	7	3
4	Quantity	01	50	25
5	Extended price	000900.05	104948.50	38236.70
6	Discount	0.00	0.10	0.05
7	Tax	0.00	0.08	0.04
8	Return flag	A	R	
9	Line status	F	0	
10	Ship date	1992-01-02	1998-12-01	1995-06-17
11	Commit date	1992-01-31	1998-10-31	1995-06-16
12	Receipt date	1992-01-03	1998-12-31	1995-07-03
13	Ship instruct	COLLECT COD	TAKE BACK RETURN	
14	Ship mode	AIR	TRUCK	
15	Comment			

Field #	Field Name	Standard Deviation	Largest Prob.	0-Order Entropy	Huffman ACL	Excess Bits	Codebook Number
0	Order key	0.0	0.00	14.13	0.00	0.00	0 *
1	Part key	0.0	0.00	13.49	0.00	0.00	0 *
2	Supp key	0.0	0.00	14.05	0.00	0.00	0 *
3	Line number	1.7	0.25	2.61	2.64	0.03	1
4	Quantity	14.4	0.02	5.65	5.72	0.07	2
5	Extended price	0.0	0.00	15.62	0.00	0.00	0 *
6	Discount	0.0	0.09	3.46	3.55	0.09	3
7	Tax	0.0	0.11	3.17	3.22	0.05	4
8	Return flag		0.51	1.49	1.49	0.00	5
9	Line status		0.50	1.00	1.00	0.00	6
10	Ship date	695.4	0.00	11.27	11.33	0.06	7
11	Commit date	694.8	0.00	11.25	11.31	0.06	8
12	Receipt date	695.5	0.00	11.27	11.33	0.06	9
13	Ship instruct		0.25	2.00	2.00	0.00	10
14	Ship mode		0.14	2.81	2.86	0.05	11
15	Comment						

* Due to excess cardinality, entropy was estimated and standard deviation, largest probability, Huffman ACL, and excess bits were not computed

3.2 Field-by-Field Before-and-After Bit Breakdown for SF100

Field Name	Uncompressed		Compressed		Compression Factor & Ratio	
	Field Total	Bits Per Record	Field Total	Bits Per Record	Factor	Ratio
Order key	47114232792	78.5	301641928	0.5	99%	156.2:1
Part key	40535914792	67.6	14590234976	24.3	64%	2.8:1
Supp key	33068765736	55.1	7172432439	12.0	78%	4.6:1
Line number	9600606432	16.0	258366322	0.4	97%	37.2:1
Quantity	13536794328	22.6	3399444467	5.7	75%	4.0:1
Extended price	42583444600	71.0	10069898528	16.8	76%	4.2:1
Discount	24001516080	40.0	2083707560	3.5	91%	11.5:1
Tax	24001516080	40.0	1907494052	3.2	92%	12.6:1
Return flag	9600606432	16.0	546085800	0.9	94%	17.6:1
Line status	9600606432	16.0	266570145	0.4	97%	36.0:1
Ship date	52803335376	88.0	5392444568	9.0	90%	9.8:1
Commit date	52803335376	88.0	4377954873	7.3	92%	12.1:1
Receipt date	52803335376	88.0	2960171703	4.9	94%	17.8:1
Ship instruct	62402998512	104.0	1203494751	2.0	98%	51.9:1
Ship mode	25372800848	42.3	1689318461	2.8	93%	15.0:1
Comment	132007444040	220.0	18952193799	31.6	86%	7.0:1
Other	9600606432	16.0	1268	0.0	-	-
Grand totals	641437859664	1069.0	75171455640	125.3	88%	8.5:1

3.3 Contents of Compressed Binary File Header for SF100

Name of Field or Structure	Size	Offset	Value
File size (bytes)	8	0	9396431955
Original file size (bytes)	8	8	80179732458
Record count	8	16	600037902
Code mode	1	24	0
Root word cardinality	1	25	207
Codebook area offset	2	26	5330
Symbol area offset	2	28	21146
Record area offset	4	30	63798
Word length array	208	34	-
Lexicon	2484	242	-
Offset table	2596	2726	-