Table Compression in Oracle9*i* Release2

An Oracle White Paper
May 2002

Table Compression in Oracle9*i* Release2

Table Compression in Oracle9*i* Release2

EXECUTIVE OVERVIEW

Data stored in relational databases keep growing as a result of businesses requirements for more information. A big portion of the cost of keeping large amounts of data is in the cost of disk systems, and the resources utilized in managing that data. Oracle9*i* Release2 Enterprise Edition introduces a unique way to deal with this cost by compressing data stored in relational tables with virtually no negative impact on query time against that data, thereby enabling substantial cost savings.

INTRODUCTION

Commercially available relational database systems have not heavily utilized compression techniques on data stored in relational tables. One reason is that the trade-off between time and space for compression is not always attractive for relational databases. A typical compression technique may offer space savings, but only at a cost of much increased query time against the data. Furthermore, many of the standard techniques do not even guarantee that data size does not increase after compression.

Oracle9*i* Release2 Enterprise Edition introduces a unique compression technique that is very attractive for large data warehouses. It is unique in many ways. Its reduction of disk space can be significantly higher than standard compression algorithms, because it is optimized for relational data. It has virtually no negative impact on the performance of queries against compressed data; in fact, it may have a significant positive impact on queries accessing large amounts of data, as well as on data management operations like backup and recovery. It ensures that compressed data is never larger than uncompressed data.

HOW IT WORKS

Oracle9*i* Release2 compresses data by eliminating duplicate values in a database block. Compressed data stored in a database block (a.k.a. disk page) is self - contained. That is, all the information needed to recreate the uncompressed data in a block is available within that block. Duplicate values in all the rows and columns in a block are stored once at the beginning of the block, in what is called a symbol table for that block. All occurrences of such values are replaced with a short reference to the symbol table.

With the exception of a symbol table at the beginning, compressed database blocks look very much like regular database blocks. Program modifications done in the server to allow for compression were very localized. Only the portions of the program dealing with formatting the block, and accessing rows and columns needed to be modified. As a result, all database features and functions that work on regular database blocks also work on compressed database blocks.

What can be compressed

Database objects that can be compressed in Oracle9*i* Release2 include tables and materialized views. For partitioned tables, it is possible to choose to compress some or all partitions.

Compression attribute can be declared for a tablespace, a table, or a partition of a table. If declared at the tablespace level, all tables created in that tablespace will be compressed by default.

It is possible to alter the compression attribute for a table (or a partition or tablespace) and the change will only apply to new data going into that table. As a result, a single table or partition may contain some compressed blocks and some regular blocks. The fact that a table may contain mixed blocks is utilized to guarantee that data size will not increase as a result of compression; in cases where compression could increase the size of a block, it is simply not applied to that block.

Compression occurs while data is being bulk inserted or bulk loaded. These operations include:

- Direct Path SQL*Loader
- CREATE TABLE … AS SELECT statement
- Parallel INSERT (or serial INSERT with an APPEND hint) statement

Existing data in the database can also be compressed by moving it into compressed form through ALTER TABLE…MOVE statement. This operation takes an exclusive lock on the table, and therefore prevents any updates, and loads until it completes. If this is not desirable, Oracle9*i*'s online redefinition utility (dbms_redefinition plsql package) can be used to overcome this problem.

Data compression works for all data types except all variants of LOBs and data types derived from LOBs, such as VARRAYs stored out of line or the XML data type stored in a CLOB.

Releases prior to Oracle9*i* Release2 had the ability to compress indexes, both Bitmap and Btree, as well as Index Organized Tables. That remains the case in Oracle9*i* Release2.

COST AND BENEFIT ANALYSIS

The main benefit of compression is the space savings achieved. The ratio of the size of uncompressed data to compressed data is often referred to as the *compression ratio*. For example, a compression ratio of 2 indicates that uncompressed data takes twice as much disk space as compressed data.

When a high compression ratio can be achieved for large amounts of data, there is a direct benefit of using much less disk space. This often translates into indirect benefits when accessing that data. For example, if the access involved having to scan a table, that could be done much faster because compression made the table much smaller. In addition, more data can be kept in the database cache in the compressed form. Therefore, even when compressed tables are accessed through an index, there may be some performance advantage of compression as it increases the chance of finding more of the table data in the cache.

Cost of compression

As a result of Oracle's unique compression technique, there is no expensive decompression operation needed to access compressed table data. This means that the decision as to when to apply compression does not need to take a possible negative impact on queries into account.

Compression is done as part of bulk loading data into the database. The overhead associated with compression is most visible at that time. This is the primary trade-off that needs to be taken into account when considering compression. If rolling window partitioning techniques are used for loading data, increased load time may be less of an issue, because the impact of load on other data warehouse workload is minimal.

Compressed tables or partitions can be modified just like any other Oracle tables or partitions. Data can be modified using INSERT, UPATE, and DELETE commands, for example. However, data, which is modified without using bulk insertion or bulk loading techniques will not be compressed. Deleting compressed data is as fast as deleting uncompressed data. Inserting new data is also as fast, because data is not compressed in case of conventional insert; it is compressed only doing bulk load. Updating compressed data may be somewhat slower in some cases.

It is possible to cause fragmentation and waste disk space when modifying compressed data. For example, if a row is deleted, the space occupied by that row becomes free in that block, but since a conventional insert does not go through compression, a future row to be inserted is likely not to fit in that space released by a compressed row. With high numbers of modifications, it is possible to waste a lot more space in this manner than can be saved through compression. In such a case, it would be necessary to recompress the data.

For this reason, compression is more suitable for Data Warehousing applications than OLTP applications. Data should be organized such that read only or infrequently changing portions of the data (e.g. historical data) should be kept compressed.

Compression ratio

Oracle has tested compression on real world customer data from several customers in different industries. The typical compression ratio for large data-warehouse tables ranges from 2:1 to 4:1. Higher compression ratios have been observed. For example, call detail data from a major telecom company produced 12:1 compression. That is, the uncompressed data was 12 times larger than compressed data. Among the customer test results, this was the highest compression ratio achieved. In other tests, 5:1 compression ratio was achieved on aggregated sales data from different customers in different industries.

Since Oracle's compression algorithm is based upon eliminating duplicate values in each block, there are additional techniques, which can improve the compression ratios in most situations. First, the data can be loaded in order to maximize the duplication of values within a database block. The easiest mechanism for doing this is by sorting the data before it is loaded. If Oracle9*i*'s external table mechanism is used for loading, then an ORDER BY clause can be added to the SELECT statement of the external table to achieve the desired ordering.

It is for this reason that materialized views containing summarized data are ideal candidates for compression. The GROUP BY clause in such a materialized view definition has the side effect of generating partially sorted data.

Also, larger block sizes may yield better compression, in general. This may happen for two reasons: First, there is an increased probability of duplicate values in a larger amount of data. Second, space taken by the symbol table in each compressed block is amortized over more data fitting in a larger block.

Storage attributes of a table affects compression ratio. For example, large PCTFREE will lead to low compression ratios. Since frequent updates are not expected on compressed tables, setting PCTFREE to 0 is recommended for all tables storing compressed data; PCTFREE is automatically set to 0 for all tables created with the COMPRESS attribute.

Compression ratio for existing tables can be estimated in an inexpensive manner with a simple program outlined in Appendix A.

Performance impact on loads and DML

Compressing data has a performance impact on loads, DML statements, and queries. Oracle has run numerous experiments to measure the performance characteristics of compression, and this section summarizes the results.

Compression overhead is most visible at bulk load time. For example, for simple loads, compressing data may cause twice the CPU usage on average. If run on a system with unlimited IO bandwidth, this may translate into doubling the load time. However, bulk loads are IO-bound on many systems. In those cases, since compression reduces the amount of data to be written, there would be some benefit in terms of elapsed load time to offset the cost of additional CPU usage. Furthermore, in cases where loading data involves complex transformations, which may take significant amount of time by themselves, the overhead of compression as a percentage of the entire load process would be even less. Without any application specific measurements you should expect an average performance impact for bulk loads of approximately 50% or more.

There is no measurable difference in the performance of non-bulk INSERT operations on compressed and uncompressed data. The reason is that a conventional INSERT operation does not go through compression. New rows are inserted uncompressed. However, bulk INSERT operations, such as parallel INSERT or CREATE TABLE … AS SELECT operations, and INSERT with an APPEND hint (a.k.a. *direct path INSERT*) go through compression, and are subject to the same bulk load performance characteristics as outlined above.

DELETE operations are 10% faster for compressed tables. The benefit comes from the fact that the compressed rows are smaller, so that there is less data to be logged. No extra work, such as cleaning up of symbol tables when appropriate, is currently done during this operation.

UPDATE operations are 10-20% slower for compressed tables on average, mainly due to some complex optimizations that have been implemented for uncompressed tables, and not yet implemented for compressed tables. These may be implemented in a future release of Oracle.

Querying compressed data is virtually as fast as querying uncompressed data in most cases. For IO-bound queries, accessing compressed data may be significantly faster. For example, a simple scan of a table that is not in the buffer cache may be 3-4 times faster if the compression ratio is 4:1 or more.

CONCLUSION

Cost of disk systems can be a very large portion of building and maintaining large data warehouses. Oracle9*i* Release2 helps reduce this cost by compressing the data stored in an Oracle database, and it does so without the typical trade-offs of space savings versus access time to data.

APPENDIX A

Following function is given as an example of how to estimate the compression ratio for an existing table. It takes the table name as an argument, and returns the compression ratio. During its processing it creates and drops a couple of tables.

```
create function compression_ratio (tabname varchar2)
return number is
   -- sample percentage
   pct number := 0.000099;
   -- original block count (should be less than 10k)
   blkcnt number := 0;
   -- compressed block count
   blkcntc number;
begin
  execute immediate ' create table TEMP_UNCOMPRESSED pctfree 0
                      as select * from ' || tabname ||
                    ' where rownum < 1';
  while ((pct < 100) and (blkcnt < 1000)) loop
    execute immediate 'truncate table TEMP_UNCOMPRESSED';
    execute immediate 'insert into TEMP_UNCOMPRESSED select *
from ' ||
                       tabname ||  ' sample block (' || pct ||
',10)';
    execute immediate 'select

count(distinct(dbms_rowid.rowid_block_number(rowid)))
            from TEMP_UNCOMPRESSED' into blkcnt;
    pct := pct * 10;
  end loop;

  execute immediate 'create table TEMP_COMPRESSED compress as
                     select * from TEMP_UNCOMPRESSED';
  execute immediate 'select
          count(distinct(dbms_rowid.rowid_block_number(rowid)))
          from TEMP_COMPRESSED' into blkcntc;
  execute immediate 'drop table TEMP_COMPRESSED';
  execute immediate 'drop table TEMP_UNCOMPRESSED';

  return (blkcnt/blkcntc);
end;
/
```

White Paper Table Compression in Oracle9i Release
[May] 2002
Author: Çetin Özbütün
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com