

RESEARCH PAPER

Research of LOB Data Compression and Read-Write Efficiency in Oracle Database

Jianjun Wang¹, Yingang Zhao² and Gaochuan Liu³¹ Gansu Provincial Earthquake Administration, Lanzhou, CN² Anqiu Earthquake Station, Weifang, CN³ China Earthquake Networks Center, Beijing, CN

Corresponding author: Yingang Zhao (40857347@qq.com)

Aiming at the problems of huge storage space, low exchange speed and low read-write speed of the current specific oracle database, the read-write speed and exchange speed tests are performed on the compressed and uncompressed Clob and Blob data by three compression algorithms, including Bzip2, Gzip and GzipLO respectively. The read speed test is performed by the direct read, substr read, and substr+threadPool read techniques. The results show that: (1) Blob is superior to Clob in terms of storage, exchange, or read-write speed; (2) For the specific database, Blob+Gzip is the optimal storage structure of the minute and second data. The read-write speed is greatly improved, and the overall capacity of the database is reduced to 7% (or less). The exchange rate of the second data is at least 7.89 times of the present rate, and the station data can be exchanged to the disciplinary center within 2–3 hours (currently 1.5 days); (3) The simplest and most widely used direct read method by software developers has poor database read efficiency, while the substr+threadPool technique shows higher database read efficiency no matter for Clob or Blob, for compressed or uncompressed, which brings a leap-forward improvement in the read speed of LOB data. The results of this paper are of high reference significance to the LOB data storage design and software development.

Keywords: Oracle LOB; compression; database read method; speed test

Introduction

At the end of 2007, the “10th Five-Year” system of a specific network was officially completed and put into operation. The software system is a four-level interconnected distributed system consisting of station, provincial bureau, national center and disciplinary center. In order to facilitate data exchange at all levels, a unified database management system (Oracle10g) and a unified database table structure are adopted nationwide (Zhou Kechang et al, 2009, 2010; Liu Gaochuan 2008). There are two main software systems: management system (B/S architecture, Running on the server) and processing system (C/S architecture, Running on the client PC machine). The former is responsible for daily data collection and storage, while the latter is responsible for daily data preprocessing and product data calculation. The management system exchanges the station data to the provincial bureau, national center, and disciplinary center on daily timing (Liu Gaochuan, 2008).

The national center is the collection center for specific data across the country and is also the largest specific database. As of August 2018, the data outputted by 3328 sets of observation instruments (364 sets of second sampling instruments, 2126 sets of minute-sampling instruments, 838 sets of hourly and daily sampling instruments) is stored in the database. The total database size is about 8000GB, which is still increasing by 800GB every year. In addition, the total data with a time resolution of minutes and seconds accounts for more than 95% (or more) of the total space of the database.

As all the minute and second data is stored in the format of “uncompressed Clob+Ascii”, and the database is caught in problems of huge data storage space, low data exchange speed, low read-write speed, operation

and maintenance difficulty. For example, it takes about 4 minutes for the “processing system” to remotely read the second sampling data of 6 elements in an instrument, and it takes at least 1.5 days to exchange the updated station observation data to the disciplinary center. Besides, 10 days are required to continuously copy the national central database (8000GB) cold backup to another server, during which the database and all services must be shut down. This cold backup method is obviously unrealistic; the hot-backup system (autonomously developed by the specific system) can only correspond to one server due to the software reasons. If there is a problem with both the main database and the backup database, data loss will be catastrophic.

With the development of the information society, people are faced with rapidly growing information, and the pressure on storing, transmitting and processing such massive information is increasing. In this case, data compression is an inevitable choice. In order to save information storage space and improve information transmission efficiency, a large amount of actual data must be effectively compressed. Data compression has been greatly valued as a support technology for solving the storage and transmission of massive information (ZHENG Cui-fang, 2011).

Data compression technique is generally classified into lossy compression and lossless compression. Lossless compression means that the reconstructed compressed data (reduced and decompressed) must be identical to the original data, and is suitable for the cases where the reconstructed signal is required to be identical to the original signal (LI Lei-ding et al, 2009; ZHENG Cui-fang, 2011). Lossless data compression algorithms are mainly divided into two categories according to the compression models: Statistical compression based algorithm and dictionary compression based algorithm. The statistical compression based algorithm mainly includes: Run length coding, Huffman coding, arithmetic coding, etc.; dictionary compression based algorithms mainly include: LZ77, LZ78, LZW, LZSS, etc. (LI Lei-ding et al, 2009; XU Xia et al, 2009; ZHENG Cui-fang, 2011; ZHANG Ai-hua et al, 2017). The compression algorithm must be able to provide a high data compression rate to support the real-time mass data storage characteristics of the database. Both compression and decompression processes must present better speed performance (LIU Hong-xia et al, 2010).

Bzip2 is a data compression algorithm and program developed by Julian Seward and released under the Free Software/Open Source Software Agreement. Seward released Bzip2 0.15 for the first time in July 1996. In the following years, the stability of this compression tool was improved and became more popular. Seward released Version 1.0 and Version 1.0.3 in 2000 and 2007 respectively (J Seward, 2002, 2007). Bzip2 is a lossless compression algorithm based on Burrows-Wheeler Transform (BWT). With its compression rate advantage, it has been more widely applied. BWT is a transform method independent of internal repeatability of data and it can effectively bring together the same characters in data to create conditions for further compression (Li Bing et al, 2015). It is able to compress the common data to 10% to 15%, and offers high compression and decompression efficiency. It is widely used in many versions of UNIX & LINUX, and supports most compression formats, including tar and Gzip. Its main advantages include: Bzip2 open source, free of charge; support for repairing media errors. When it is required to obtain the data in the erroneous compressed file, Bzip2 can still perfectly decompress the unbroken part; it can run on any 32-bit or 64-bit host containing ANSI C compiler (Jeff Gilchrist, 2008; V Pankratius et al, 2009; M Mccool et al, 2012; JS Salazar et al, 2017).

Bzip2 provides two data compression algorithms, including Bzip2 and Gzip, which can be called by dll interface file, ICSharpCode.SharpZipLib.dll. The System.IO.Compression namespace of Microsoft .Net also provides another Gzip compression algorithm, which is referred to as GzipIO in this paper.

In Oracle database, Clob and Blob (Abbreviated LOB) are two typical large object data storage structures, which are widely used in all levels of databases. Clob can only store single-byte character data, and is mostly used to store long text data. Blob is used to store unstructured binary data, mainly including formatted images, videos, audio and Word documents (NIE Hong-mei et al, 2006; ZHANG Jing et al, 2011; ZHANG Hui et al, 2012; XIE Yi et al, 2015).

Based on the Microsoft .Net development platform, this paper uses Bzip2, Gzip, and GzipIO compression algorithms to test and compare the read-write speed, exchange speed of the compressed and uncompressed data of Clob and Blob. The three techniques, including direct read, substr read and substr+threadPool read, are applied for the read speed test. The advantages and disadvantages of each compression algorithm and database read method are summarized in order to test the “optimal” compression algorithm and database read method for the specific database.

1 Test data and research method

1.1 Test data

The test data selected in this paper are 6 elements of minute and second data in 31 days outputted from an instrument from January 1 to January 31, 2009. The instrument adopts second sampling, and each element includes 86400 second sample data per day. The minute data is calculated by the second sample data through Gaussian filtering, and each element includes 1440 minute sample data per day.

All the test results in this paper are completed in the office of Lanzhou, Gansu Province. The local server is located in the information room of the work unit while the remote server is located in the information room of a research institute in Beijing. The test data and table structure of the local and remote are identical. The test software is client software written on the Microsoft .Net development platform (running on an office PC machine).

1.2 Minute and second data table structure

The minute and second data table structure in the specific database is shown in **Table 1**. The observation data is in **Table 1**. The observation data is stored in the format of "uncompressed Clob+Ascii", and one record is required to be made by each set of instruments per day.

1.3 LOB data compression and decompression method

After the interface file ICSharpCode.SharpZipLib.dll provided by Bzip2 is referred to in Microsoft .Net, the BZip2OutputStream and BZip2InputStream method are called through the namespace ICSharpCode.SharpZipLib to complete Bzip2 compression and decompression, the GzipOutputStream and GzipInputStream method are called to complete Gzip compression and decompression, respectively. The process of GzipIO compression and decompression is completed by calling System.IO.Compression.GzipStream method.

1.4 Database connection and LOB read-write method

The Microsoft .Net framework uses ADO.NET to complete access to the database, and OracleConnection for database connection, OracleDataAdapter and DataTable for LOB data reading and temporary storage, and OracleCommand for LOB data writing.

1.5 Three LOB read methods

- 1) **Direct read:** For software developers, the simplest and most common database read method for LOB data is to read directly using Select LobName.
- 2) **Substr read:** Both Clob and Blob can use the substr function in Oracle's own DBMS_LOB package to read data segmentally, namely Select DBMS_LOB.substr(lobName, n, pos), where lobName is Lob field name, n is the number of bytes read, and pos is the starting position of the read. The maximum length that the Clob can read is 4000 bytes at a time, while the maximum length that the Blob can read is 2000 bytes at a time. Therefore, the substr read must be cyclically executed, and the starting position (pos) of the read must be reset every time. After the cycle, the data of the same date should be spliced in order.

Table 1: Minute and second data table structure of specific database.

Field name	Primary key	Field type	Example of value	Description of value
startDate	√	date	2006-02-23	One record per day
stationID	√	Char(5)	44010	National unified 5-digit station ID
pointID	√	Char(1)	4	Positioned to each set of instrument
itemID	√	Char(4)	3125	D, H, Z, F, I, X, Y
sampleRate		Char(2)	01	01: Minute data, 02: Second data
obsValue		Clob	86.73 86.23 ... 89.12	The data is separated by a space character, 1440 minute data and 86400 second data are observed per day.

3) Substr+threadPool read: ThreadPool class provides thread pool management in Microsoft .NET. The SQL statement from substr read is placed in the thread pool in turn, which can execute the “substr read” in parallel (multiple threads read at the same time). Besides, a separate sub-thread class is required, which needs to create a new database connection to execute the SQL statement of substr read; after all the tasks are added to the thread pool, a while loop is required to exit the cycle and perform subsequent operations after all threads have been executed.

2 LOB data compression and exchange speed test

2.1 LOB data compression test

Bzip2, Gzip and GzipIO compression algorithms are used to test minute and second data of an instrument in January 2009 outputted by an instrument. **Table 2** shows the Average compression rate of every record. For both minute and second data, the compression rate of Bzip2 is the highest, followed by Gzip and GzipIO successively; the capacity of compressed record in Bzip2 is the smallest, which means that the compressed record occupies smaller storage space. However, Bzip2 takes the longest compression and decompression time, far higher than that of other two algorithm. This means that the database read (decompression) and write (compression) consume a longer time. The compression time of Gzip is about 2.5 times of that of GzipIO but the difference in the decompression time of both algorithms is very small. But the binary compression rate of minute and second data is improved by 5% and 3% respectively.

2.2 LOB data exchange speed test

Currently, the specific management system software adopts “dbLink+Insert” technique in data exchange. The core command of the data exchange is “insert into XX from XX@dbLinkName”, where dbLinkName is the dbLink of the remote database. The statement directly insert the data from a remote table into the same table at the local database (The local data records are firstly deleted and then inserted when data records exist). According to the current specific data exchange mechanism, there is no need to parse the LOB data during the exchange process. Therefore, the compression and decompression efficiency has no effect on the exchange speed, and only the capacity of each record can be affected by the exchange speed. After logging into the remote (Beijing) database, the command is run directly, and its execution time is taken as the actual exchange time, which refers to the average time for each record to be transmitted from the local (Lanzhou) to the remote (Beijing). The data exchange speed test results of an instrument in Lanzhou in January 2009 are shown in **Table 3**. The estimated exchange rate = Capacity of “uncompressed Clob”/other capacity, and the actual exchange rate = Exchange time of “uncompressed Clob”/other exchange time.

Table 3 shows the Average data exchange speed test of every record. For both Clob and Blob, the actual exchange rate of the three compression algorithm for minute and second data is not as good as the estimated exchange rate, and the actual exchange rate of the second data Blob compression is improved by 7–9 times, but the actual exchange rate of the minute data is only slightly increased; regarding the Blob and Clob uncompressed algorithm for second data, the actual exchange rate is improved by 1.84 times in the case of the same storage capacity.

Table 2: Average compression rate of every record.

Data	Compression algorithm	Pre-compression capacity (KB)	Compressed clob capacity (KB)	Compressed blob capacity (KB)	Compression time (s)	Decompression time (s)	Clob compression rate (%)	Blob compression rate (%)
Second data	Bzip2	588.51	37.462	28.096	0.18909	0.02016	6.24	4.68
	Gzip	588.51	63.376	47.531	0.03370	0.00692	10.45	7.84
	GzipIO	588.51	86.906	65.179	0.01328	0.00633	14.36	10.77
Minute data	Bzip2	8.842	0.961	0.720	0.00458	0.00113	10.89	8.15
	Gzip	8.842	1.399	1.048	0.00100	0.00034	15.76	11.81
	GzipIO	8.842	2.008	1.505	0.00035	0.00027	22.62	16.95

Note: The black bold characters refer to smallest compressed capacity, shortest compression and decompression time, and highest compression rate.

Table 3: Average data exchange speed test of every record.

Field type	Compression algorithm	Second data				Minute data			
		Capacity of every record (KB)	Estimated exchange rate	Actual exchange time (s)	Actual exchange rate	Capacity of every record (KB)	Estimated exchange rate	Actual exchange time (s)	Actual exchange rate
Blob (Binary)	Bzip2	28.09	20.9	0.051	8.97	0.72	12.3	0.043	1.13
	Gzip	47.53	12.4	0.058	7.89	1.05	8.4	0.044	1.11
	GzipIO	65.18	9.0	0.064	7.14	1.50	5.9	0.044	1.09
	Uncompressed	588.51	1.0	0.248	1.84	8.84	1.0	0.045	1.06
Clob (Ascii)	Bzip2	37.46	15.7	0.068	6.75	0.96	9.2	0.045	1.06
	Gzip	63.38	9.3	0.083	5.49	1.40	6.3	0.045	1.08
	GzipIO	86.91	6.8	0.106	4.30	2.01	4.4	0.046	1.05
	Uncompressed	588.51		0.456		8.84		0.048	

3 LOB data read-write speed test

3.1 Read-write speed test for direct read method

The direct read method are used to test the read-write speed of the four storage structures (three compressed structures + one uncompressed structure). **Table 4** shows the read-write speed test result of four storage structures. (1) The database write speed of GzipIO is the highest for both Clob and Blob, closely followed by Gzip, difference between the two is very small. As Bzip2 requires a long compression time, the local database write speed is much lower than that of others. (2) The database read speed in Bzip2 for Clob and Gzip for Blob is the highest. Even if it is sometimes slower than other methods, the difference between the read speed and the highest speed is the smallest. (3) For the same compressed or uncompressed structure, the database write speed of the two LOB types is basically the same, but the database read speed of Blob is much higher than that of Clob.

3.2 Read speed test for three LOB read methods

The three LOB read methods are used to test the read speed of the compressed and uncompressed structures. **Table 5** shows the read speed test result of three LOB read methods. (1) For the direct read method, it has the worst read efficiency in the uncompressed Clob, and its read speed is much lower than that of other two methods; in the Blob, except for its high local read speed of one-day second data, the other read efficiencies are almost the worst, and as the number of read days increases, its read speed gap with the substr+threadPool method is increasingly. (2) For the substr read method, its read speed in uncompressed Clob is significantly higher than that of the direct read method, but it is unstable in the Blob second data read, and it is common that its read time is much longer than that of other methods. (3) For the substr+threadPool method, its read speed is the highest in the uncompressed Clob, far superior to the other two methods. Even if it is sometimes slower than other methods, the difference between its read speed and the highest speed is the smallest. (4) No matter what read methods, the read speed of Gzip is higher than that of GzipIO.

In relative terms, the substr+threadPool method can display the highest database read efficiency compared with the other two methods no matter for Clob or Blob, for compressed or uncompressed. Especially, its read speed of the uncompressed Clob is greatly higher than that of other two methods. The storage structure of "Blob+Gzip" combined with the method of "Substr+threadPool read" can make the reading performance of specific database to be "optimal".

4 Discussion and conclusion

4.1 Discussion

The tests show that Blob is superior to Clob in storage performance, but the Clob has advantages in improving the retrieval speed of long text data (Zhang Jing et al, 2011). The above test results once again verify the conclusion of Zhang Jing et al. Blob is superior to Clob in terms of storage, exchange or read-write speed, but

Table 4: Read-write speed test result of four storage structures.

An instrument (6 elements) Read and Write times(second)		Second data						Minute data					
		1 day		31 days		31 days		1 day		31 days		31 days	
		Local	Remote	Local	Remote	Local	Remote	Local	Remote	Local	Remote	Local	Remote
Field type	Compression algorithm	Read(s)	Write(s)	Read(s)	Write(s)	Read(s)	Write(s)	Read(s)	Write(s)	Read(s)	Write(s)	Read(s)	Write(s)
Blob (Binary)	Bzip2	0.20	1.21	1.73	2.92	5.36	44.43	39.98	96.32	1.74	2.32	26.76	51.93
	Gzip	0.11	0.23	1.64	2.06	2.91	6.88	39.52	63.91	1.48	1.37	26.71	49.96
	GzipIO	0.12	0.14	1.87	2.04	3.10	4.11	45.82	63.02	1.56	1.23	26.75	49.95
	Uncompressed	0.36	0.37	4.63	6.43	11.02	12.28	194.44	200.60	0.97	1.40	25.83	50.39
Clob (Ascii)	Bzip2	1.03	1.23	17.15	3.01	28.54	44.72	556	105.63	3.07	2.14	43.42	51.53
	Gzip	1.47	0.24	28.24	2.13	42.52	7.48	944	76.76	2.83	1.41	46.67	50.38
	GzipIO	1.95	0.16	38.21	2.22	57.29	4.97	1278	76.71	2.69	1.33	55.21	50.14
	Uncompressed	11.94	0.39	243.70	6.58	372.52	12.58	7568	205.56	6.25	1.58	139.15	50.88

Note: Read time = Database read time + Decompression time, Write time = Compression time + Database write time. The black bold characters refer to the highest read-write speed in the four storage structures.

Table 5: Read speed test result of three LOB read methods.

Field type	Compression algorithm	Database read method	An instrument (6 elements) Read times(second)			Second data			Minute data					
			1 day	10 days	31 days	1 day	10 days	31 days	1 day	10 days	31 days			
BLOB (Binary)	Bzip2	direct read	0.19	1.80	1.75	14.72	4.97	36.56	0.057	1.374	0.526	10.51	1.59	26.84
		substr read	0.52	6.85	1.83	11.33	4.83	19.39	0.040	0.954	0.124	1.32	0.25	1.37
		substr+threadPool	0.21	1.33	1.45	5.49	4.18	10.14	0.039	0.937	0.122	1.13	0.28	1.44
Gzip	Gzip	direct read	0.11	1.74	1.03	16.26	2.88	41.49	0.058	1.425	0.476	10.16	1.52	26.83
		substr read	0.66	11.25	1.41	16.63	3.18	27.22	0.035	0.945	0.062	1.15	0.11	1.31
		substr+threadPool	0.14	1.64	0.81	5.37	2.11	8.57	0.041	0.940	0.061	1.11	0.09	1.23
GzipIO	GzipIO	direct read	0.12	1.91	1.09	17.49	3.08	45.69	0.065	1.460	0.534	10.24	1.67	27.23
		substr read	0.88	14.79	1.71	22.03	3.79	36.63	0.049	1.110	0.074	1.36	0.11	1.60
		substr+threadPool	0.17	2.60	0.92	5.79	2.38	9.09	0.043	0.956	0.068	1.18	0.10	1.48
Uncompressed	Uncompressed	direct read	0.38	6.34	3.66	70.01	10.89	198.20	0.045	1.414	0.332	9.77	1.06	26.15
		substr read	6.69	127.46	12.58	185.94	25.41	300.55	0.131	2.860	0.199	3.49	0.32	5.14
		substr+threadPool	0.79	6.84	4.80	20.48	10.76	40.09	0.049	0.992	0.099	2.39	0.20	3.49
Clob (Ascii)	Uncompressed	direct read	12.62	245.94	121.31	2654.53	384.05	7579.19	0.228	5.681	2.177	54.24	6.69	140.84
		substr read	4.19	65.97	13.69	186.07	23.33	353.76	0.094	1.944	0.227	3.94	0.44	6.65
		substr+threadPool	0.56	5.85	4.37	18.41	10.87	39.34	0.052	0.958	0.116	2.09	0.26	3.89

Note: Read time = Database read time + Decompression time. The black bold characters refer to the highest database read speed in the three LOB read methods, and the black bold boxed characters refer to the higher speed in Clob and Blob.

the “uncompressed Clob+Ascii” format can use the `DBMS_LOB.substr` function to read partial data (Obtain the starting position of each data by separator), and the read speed is much better than the overall read speed, which Blob cannot achieve because of binary storage. For a specific database, there are very few cases of reading partial of the data, and a large number of practical applications require overall read (data processing, drawing, downloading, etc.).

The optimal compression algorithm should have the highest compression rate, the highest compression and decompression speed, which, however, is difficult to achieve in practice. Bzip2 has the highest compression rate but longer compression and decompression time. Gzip and GzipIO have shorter compression and decompression time but slightly lower compression rate. Both compression and decompression require good speed performance. The solutions to these two problems are contradictory. The study of the compression algorithm is to find the balance between the two and achieve optimal performance (LIU Hong-xia et al, 2010). Gzip and GzipIO are better than Bzip2 if only the data read-write speed is considered. Compared with GzipIO, Gzip is superior in read speed, and GzipIO is superior in write speed. The difference between the two in terms of read-write speed is very small, but the compression rates for minute and second data of Gzip are 5% and 3% higher than those of GzipIO, which saves more storage space for disks and provides faster data exchange.

For the specific database, if the Blob+Gzip storage structure is adopted, the overall capacity of the database is reduced to 7% (or lower), and the data read-write speed is greatly improved. The second data exchange rate is at least 7.89 times of the present rate, and the station data can be exchanged to the disciplinary center in the shortest time, thus improving the time efficiency of the specific data. At present, it takes 1.5 days to exchange the data from the station to the disciplinary center, and 4 times exchanges per day are generally performed. After the compressed structure is adopted, more than 24 exchanges per day can be performed (once per hour), and it will take less than 2–3 hours to exchange the station data to the disciplinary center.

The direct read method is the simplest and most widely used database read method for software developers, but it is less efficient. The `substr` read method can read a maximum length of 4000 bytes in Clob and can read a maximum length of 2000 bytes in Blob. In the case of the same record capacity, the number of Blob cycles is twice that of Clob, which will lead to reduced read efficiency of Blob. This should be the root cause for the unstable performance and frequent longer read time than that of other two methods in the Blob second data read. The `substr+threadPool` method adopts a multi-thread parallel read technique, which just makes up for this deficiency, and shows high read efficiency in both Clob and Blob, compressed and uncompressed.

The disadvantage of the `substr+threadPool` method is that a large number of database connections are consumed during reading, and there must be enough connections (`Open_Cursors`) in the database. Thread pool management in NET has a default limit of up to 25 threads per available processor, and the maximum number of concurrent threads we monitored so far is only 19. That is to say, although the total number of threads opened at the time of LOB reading may be as high as 200 to 300, but, in fact, only 25 threads can be read concurrently, while the other threads are all in the waiting state. The total number of `Open_Cursors` (the national specific oracle database) is set to 30000, so 1200 users can be supported to read data simultaneously according to this method, and the database access must be under the specific industry network. This configuration is sufficient to support `substr+threadPool` method within the specific system.

4.2 Conclusion

Aiming at the problems of huge storage space, low exchange speed and low read-write speed of the current specific oracle database, the read-write speed and exchange speed tests are performed on the compressed and uncompressed Clob and Blob data by three compression algorithms, including Bzip2, Gzip and GzipIO respectively. The read speed test is performed by the direct read, `substr` read, and `substr+threadPool` read techniques. The results show that:

- (1) Blob is superior to Clob in terms of storage, exchange, or read-write speed.
- (2) For the specific database, Blob+Gzip is the optimal storage structure of the minute and second data. The read-write speed is greatly improved, and the overall capacity of the database is reduced to 7% (or less). The exchange rate of the second data is at least 7.89 times of the present rate, and the station data can be exchanged to the disciplinary center within 2–3 hours (currently 1.5 days).
- (3) The simplest and most widely used direct read method by software developers has poor database read efficiency, while the `substr+threadPool` technique shows higher database read efficiency no matter for Clob or Blob, for compressed or uncompressed, which brings a leap-forward improvement in the read speed of LOB data.

Acknowledgements

All the test data in this paper are from the China National Geomagnetic Network Center. We would like to express our sincere gratitude!

Funding Information

Jointly funded by the Seismic Science and Technology “Spark Program” Specific Project (XH17038) of China Earthquake Administration and the Basic Scientific Research Project of Gansu Provincial Earthquake Administration (2013IESLZ02).

Competing Interests

The authors have no competing interests to declare.

Author Informations

Wang Jianjun, male, born in 1975, senior engineer, Master's degree, engaged in seismic electromagnetic observation method research and technical management.

Zhao Yingang, male, engineer, Bachelor's degree, engaged in seismic monitoring and seismic software research and development.


References

- Gilchrist, J.** 2008. Parallel data compression with BZip2[J]. *Parallel & Distributed Computing & Systems*.
- Li, B, Long, B-J and Liu, Y.** 2015. A Fast Algorithm for Burrows-Wheeler Transform Using Suffix Sorting. *Journal of Electronics & Information Technology*, 37(2): 504–508.
- Li, L-D, Ma, T-H and You, W-B.** 2009. Analysis of common lossless compression algorithm. *Electronic Design Engineering*, 17(1): 49–50.
- Liu, G.** 2008. Earthquake Precursory Data Exchange System Design[D]. Beijing: Institute of Geophysics, China Earthquake Administration, 1–85.
- Liu, H-X and Niu, F-L.** 2010. Research and Improvement of Data Compression Algorithm in Real-time Database. *Control and Instruments in Chemical Industr*, 37(6): 72–75.
- Mccool, M, Robison, AD and Reinders, J.** 2012. Chapter 12 – Bzip2 Data Compression. In: *Structured Parallel Programming[M]*. Elsevier Inc, 291–297. DOI: <https://doi.org/10.1016/B978-0-12-415993-8.00012-8>
- Nie, H-M and Zhao, J-M.** 2006. Research of Optimum Query Technology on Clob Big Segment in Oracle Database. *Computer Technology and Development*, 16(8): 97–99.
- Pankratius, V, Jannesari, A and Tichy, WF.** 2009. Parallelizing Bzip2: A Case Study in Multicore Software Engineering[J]. *IEEE Software*, 26(6): 70–77. DOI: <https://doi.org/10.1109/MS.2009.183>
- Salazar, JS and Sánchez, EA.** 2017. Enhanced Parallel bzip2 Compression with Lock-Free Queue[J]. *Uniciencia*, 31: 37–49.
- Seward, J.** 2002. The bzip2 and libbzip2 official homepage (<http://sources.redhat.com/bzip2/>).
- Seward, J.** 2007. bzip2 and libbzip2, version 1.0.3, a program and library for data compression.
- Xie, Y, Wang, H, Liu, X-H, et al.** 2015. Research on Data Reading Techniques Based on Big Data Environment. *Computer Technology and Development*, 25(2): 113–116.
- Xu, X, Ma, G-S and Yu, T.** 2009. Research and Improvement on LZW Lossless Compression Algorithm. *Computer Technology and Development*, 19(4): 125–127.
- Zhang, A-H, He, Y-H and Zhang, J.** 2017. Image Compression Coding Algorithm Based on Wavelet and Fractal Theory. *Computer Technology and Development*, 27(6): 46–50.
- Zhang, H, Zhao, Y-L, Xu, J, et al.** 2012. Query Optimization Research on Mass of Data Based on Oracle Database. *Computer Technology and Development*, 22(2): 165–167.
- Zhang, J and Wang, Y-M.** 2011. Research on Application Technology of LOB in Database Application System. *Computer Technology and Development*, 21(2): 166–169.
- Zheng, C-F.** 2011. Research of Several Common Lossless Data Compression Algorithms. *Computer Technology and Development*, 21(9): 73–76.
- Zhou, K, Zhang, C, Ji, S, et al.** 2009. Discussion on the Problems of the Earthquake Precursory Observation Networks of China. *Seismological and Geomagnetic Observation and Research*, 30(1): 76–80.
- Zhou, K, Jiang, C-H, Ji, S-W, et al.** 2010. On the Design of Earthquake Precursor Observation Database System. *Earthquake*, 30(2): 143–151.

How to cite this article: Wang, J, Zhao, Y and Liu, G. 2019. Research of LOB Data Compression and Read-Write Efficiency in Oracle Database. *Data Science Journal*, 18: 8, pp. 1–10. DOI: <https://doi.org/10.5334/dsj-2019-008>

Submitted: 12 September 2018 **Accepted:** 28 January 2019 **Published:** 08 February 2019

Copyright: © 2019 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See <http://creativecommons.org/licenses/by/4.0/>.

 *Data Science Journal* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS 