RevistaEconomică
Journal of economic-financial theory and practice
Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

WORKING WITH LARGE OBJECTS IN APPLICATION DEVELOPMENT

Vanya LAZAROVA[1]

University of National and World Economy, Bulgaria, 0000-0003-2849-4818

Abstract:
In this paper are considered the specifics of the manipulation of large objects in the database of an application - recording, storage, processing, transmission over the network. Some literature sources dealing with this topic will be reviewed and after that some specifics of working with large objects will be presented. The paper will also address the advantages of using large objects in application development. Some new software products that assist the storage of large objects will also be reviewed. Large Objects (LOBs) are a set of data types in Oracle DBMS, Microsoft SQL Server etc. LOBs are designed to store large amount of data. Large objects are for huge files such as an audio and video files. LOBs can support a maximum file size ranging from 2 GB to 8 GB. Large objects are suitable for storing semi-structured and unstructured data. Storing data in large objects allows data to be accessed and manipulated efficiently within an application. The data in the application's database, with the accumulation of many audio and video files, will at some point begin to meet the rules of big data, and its processing and storage must be carried out by specialized big data environments. The problems of transferring and storing large objects in specialized big data environments will be discussed. Data transfer between different environments will be addressed. The format and content of the data must be preserved, and this leads to the need to comply with several conditions, which will be considered.
This work has been supported by the project BG05M2OP001-1.002-0002 "Digital Transformation of Economy in Big Data Environment".

Keywords: *Large Objects, LOBs, CLOBs, BLOBs*

JEL classification: *O33*

1. Introduction

The aim of the research is to analyse the problem of converting the multitude of external multimedia files used in information systems into data, entered in databases. This type of database items are large objects. Introducing the large objects is done in order to demonstrate how they can be used in applications.

In the documentation of DBMS Oracle, Microsoft SQL Server, etc. has a description of how the large LOB data types are created and stored. The company documentation provides detailed instructions to programmers on how to create, store, and use LOBs. The term large object (LOB) basically refers to BLOB and CLOB.

Oracle (online) gives the following definitions of BLOB and CLOB:

"BLOB: Variable-length binary large object string that can be up to 2GB long. Primarily intended to hold non-traditional data, such as voice or mixed media. BLOB strings are not associated with a character set."

"CLOB: Variable-length character large object string that can be up to 2GB long. A CLOB can store single-byte character strings or multibyte, character-based data. A CLOB is considered a character string."

In addition to company documentation, many authors consider large objects, as they provide extensive opportunities for system developers.

Some authors (Stojanovic I., Bogdanov M., 2006) consider the LOB data type as a way to store and search images in databases. The authors use an Oracle database to demonstrate searching through QBIC (Query by Image Content).

[1] *vlazarova@unwe.bg*

Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

Other authors (Kudo, T., Yuki I., Serizawa. Y., 2017) compare the capabilities of storing and manipulating data in large LOB objects with the capabilities of MongoDB, where data is stored in document objects.

But there are almost no publications in the literature related to what the technology is, what algorithms to use to convert the external multimedia files into part of the database, how to become large objects.

The processing of LOBs differs from the processing of other data types in the database, such as numbers, strings, or dates. The database cannot understand the contents in LOBs. Therefore, it is impossible to use database functions such as sorting, filtering, and searching for specific content in a column of LOBs. LOB data types have some restrictions. LOB columns cannot be used in the SQL clauses: ORDER BY, GROUP BY, FROM, SELECT DISTINCT, CREATE INDEX.

The content of a BLOB is typically an exact copy of the content of multimedia or text file with a specific format. In BLOBs could be loaded, for example: video files (MP4, MOV), audio files (MP3), images (JPG, PNG, PDF, RAW), formatted documents (PDF, DOC, XLS). The content of a CLOB is typically large text file, with complex structure, for example CVS, XML, JSON.

For very large LOBs there is a technology of Remote Blob Store (RBS). The RBS technology allows the large LOBs to be saved in some remote data storage, instead on the database server. However, the application developer is using the LOBs on way as if they are saved in the database. The maintenance tools perform all actions for support of the remote LOB data. The RBS for MS SQL Server is available as option in the standard package (Microsoft: Binary Large Object, 2023)

In the next sections we consider the cases where the application uses a big number of multimedia and files, of medium size, up to 50MB. The services for larger LOBs are considered in the last section.

2. Usage of LOBs in applications with relational DBMS

Any system with database with LOBs containing multimedia, is functional equivalent to a system without LOBs, where in the database are saved links to files with the same multimedia, located in the filesystem. What is the benefit of putting a set of multimedia files in BLOB elements of a database, instead of keeping them in the filesystem?

Some years ago, three authors (Sears, Ingen, Gray, 2006) published research with the markable title “To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem.” In the paper it is analyzed very thoroughly exclusively, in what case the data takes more memory, and in what case the data is retrieved faster.

Nowadays, 17 years later, the focus of the comparison has changed. The applications usually work equally well in both ways. The arguments by making decision “To BLOB or Not To BLOB” are focused mainly about the management of the information system, including such aspects as:

- Easier development process.
- The easy and secure way to make regular back-up of the data.
- The easy and secure way to set-up permission for access to the data.

In the traditional approach, the multimedia files are stored in the filesystem, and these files are hard to protect. Permission to access these files usually requires external authorization, which makes processing difficult.

2.1. BLOBS and CLOBS in MS SQL and Oracle

During the last decades, all popular relational DBMS introduced data types for BLOBs and CLOBs.

Table 1: Data types for BLOBs and CLOBs in different relational DBMS

Relational DBMS	BLOB data type	Max length	CLOB data type	Max length
MS SQL	VARBINARY (MAX)	2 GB	VARCHAR (MAX)	2 GB

Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

IBM DB2	BLOB	2 GB	CLOB	2 GB
Oracle	BLOB	2 GB	CLOB	2 GB
MySQL	MEDIUMBLOB	16 MB	MEDIUMTEXT	16 MB
	LONGBLOB	4 GB	LONGTEXT	4 GB

Source: Own elaboration.

2.2. An example for application using BLOBs with MS SQL

To demonstrate the use of BLOBs with MS SQL, we developed a sample Job Applicant Registration application. The Job Applicant Registration (JAR) is a web-application, realized on Phyton, using MS SQL database. The records of job applicants include a set of standard items as “First Name”, “Last Name”, "Age”, “Address” etc. In addition to these data, for every applicant are stored three files:

1) Curriculum vitae (CV) document, in PDF format.
2) Personal photo, in JPG format.
3) Verbal self-presentation of the applicant, in WAV format (motivational statement).

In the initial version of JAR, the files are saved in three folders in the filesystem. The database records of the applicants include three text fields CV_pdf_path, CV_audio_path, and CV_Photo_path, containing the paths to the corresponding files. The support of these files raises a strong problem about data security. Actually,the files could be illegally copied or harmed, without any alarm in the application.

The new version of JAR is using BLOBs. The transformation of the application has two parts: changes in the database and changes in the application.

On the one side, in the database, the text fields with paths are replaced with BLOBs containing the multimedia data.

Table 2: Columns in the database

Column name	Data type
CV_pdf_BLOB	VARBINARY (max)
CV_audio_BLOB	VARBINARY (max)
CV_Photo_BLOB	VARBINARY (max)

Source: Own elaboration

The load process is extended with SQL Transact scripts, for loading the file contents in the BLOBs. The next (Fig. 1) is an example of the Transact SQL script for loading files into the BLOB columns.

Figure 1: Transact SQL script for loading files in the BLOB columns

```
INSERT INTO JobApplicans(Applicant_ID,  CV_pdf_BLOB, CV_audio_BLOB, and
CV_Photo_BLOB)
SELECT
   “123456” as Applicant_ID,
    (Select  * FROM OPENROWSET(BULK 'cv\123456.pdf ', SINGLE_BLOB))  as CV_pdf_BLOB,
    (Select  * FROM OPENROWSET(BULK 'audio\123456.wav ', SINGLE_BLOB)) as CV_audio_BLOB,
    (Select  * FROM OPENROWSET(BULK 'photo\123456.jpg ', SINGLE_BLOB)) as CV_Photo_BLOB
```

Source: Own elaboration

On the other side, in the application, the modules are changed in order to read the BLOBs instead of the files.

So, as a result, no markable change in the performance of the application is detected.

The main benefit of using BLOBS in the application is that the development team must not care about thousands of small files; all the multimedia data is saved in one table of the database.

When a lot of data accumulates and becomes Big, it is good to use specialized Big Data Storage.

3. Usage of LOBs in applications connected to Big Data Storage

It is now considered as a notoriously well-known fact that big data is data that conforms to the principles of the four Vs. The 4Vs of Big Data are:

Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

- Variety: Structured data; Semi-structured data; unstructured data
- Velocity: Speed of generation; streaming data; rate of analysis
- Volume: Terabytes
- Value: Valuable information

3.1. Big Data Storages

Apache Hadoop is an open source, java-based framework which involves some of the big data principles.

It implements Hadoop Distributed File System (HDFS) which allows the storage of different variety of data. Hadoop is used for cluster resource management, parallel processing, and for data storage.

The most popular SQL engines for Hadoop are Apache Spark SQL and IBM Big SQL (Table 3). Both systems get access to a variety of data sources including HDFS, RDBMS, NoSQL databases. As seen in the table, Hadoop does not support BLOBs.

Table 3: Most popular SQL engines for Hadoop

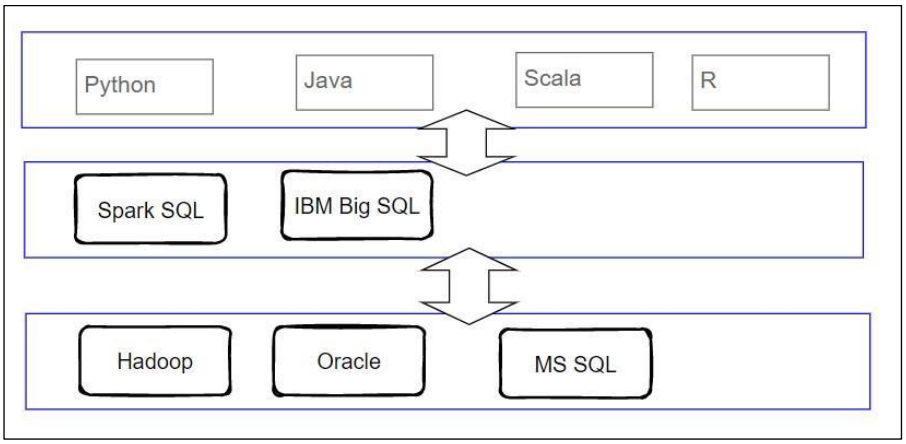
SQL engine for HDFS	Not-large binary data type	Max length	CLOB data type	Max length
Spark SQL	VARBINARY	32 KB	STRING	2 GB
IBM Big SQL	VARBINARY	32 KB	CLOB	2 GB

Source: Own elaboration.

To develop the JAR sample application, it is necessary to build a multi-tier architecture. We need such a structure in which to combine the capabilities of programming languages with storage in Big Data Storages (Fig 2).

- Layer of application development environments for different programming languages, e.g. Python, Java, Scala.
- Layer of Database Interface Engines – Interface systems for connection of the applications to different databases, some of them supporting SQL - Apache Spark SQL and IBM Big SQL.
- Layer of Big Data Storages, e.g. Hadoop, Apache HBase as well as RDBS (MS SQL, Oracle).

Figure 2: Multi-layered Architecture of Application systems using Big Data



Source: Own elaboration

Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

3.2. Transforming BLOBs into Char64 CLOBs

As it was noted above, the SQL engines, connected to Big Data storages, like Hadoop Distributed File System (HDFS), do support CLOBs but not BLOBs.

The content of a CLOB is, by default, interpreted as string of symbols in UTF-8 coding. Because of this default interpretation, the database servers apply algorithms for compression of the strings. Such "optimization" of the strings has fatal consequences when we use CLOB datatype to save binary data. In order to use CLOB datatype for saving binary data, we need to grant that no single bit of the content will be changed.

There are quite a few algorithms for converting binary code to text, such as Base91, Base122, but currently the most widely used is Base64, probably because of its simplicity and strictness.

The most popular algorithm to convert binary data into plain text is Base64. The Base64 alphabet is defined in RFC 4648 (2006) "Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string."

It is often used to embed binary data into text documents such as HTML, CSS, JavaScript, or XML.

Base64 is supported by various programming languages. (Further information in Base64.Guru (online) for transformation algorithms.)

We will give a small example. Let there be an image whose content in binary is:

010101000000011010000110 ... We'll convert the image to text using the Base64 algorithm (Fig. 3).

Figure 3: Converting image to text using the Base64 algorithm

Bit Pattern	0101010000000110100001 10 ...			
Split in group of 6 bits	010101	000000	011010	000110
Base64 encoded	V	A	a	G

Source: Own elaboration

The converted file is larger than the original about 33 %. It is because to any 6 bits from the original bit string corresponds 8 bits in the resulting string (8/6 = 1.33).

To recall, here we consider the usage of LOBs where the application needs to work with a big number of multimedia files, of medium size, up to 50MB. In such cases the grow-up of the items, in result of the Base64 coding does not rise essential problems.

3.3. An example for application using CLOBs with Spark SQL

Let us consider how we could transform the JAR application, if we want to move the database from MS SQL server to a public Big Data storage (Hadoop).

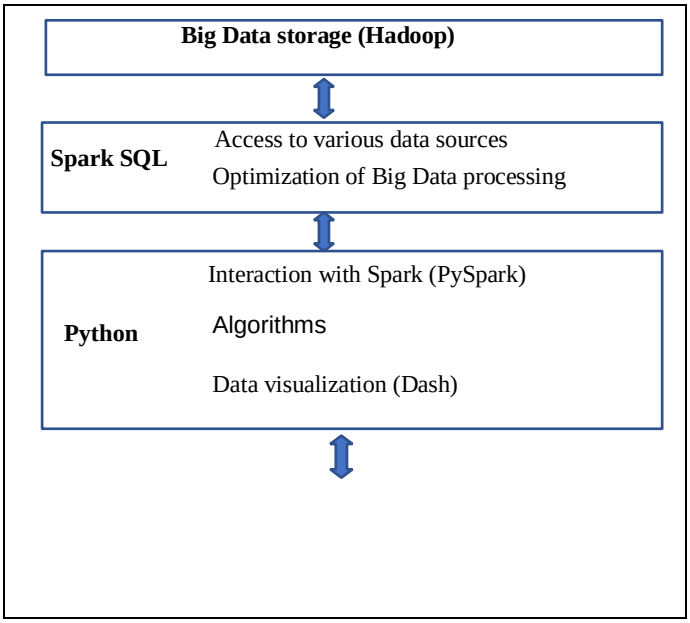
In the Hadoop database, the columns CV_pdf_BLOB, CV_audio_BLOB, and CV_Photo_BLOB are replaced with CLOBs in base64, labeled respectively CV_pdf_base64, CV_audio_base64, and CV_Photo_base64.

The connection of the Python application to Spark SQL is realized using the PySpark package.

The algorisms in the application do not need any changes. All we need is to modify the data visualization (Fig. 4). In this case we use the package Dash for Python, which generates visualization in HTML5.

Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

Figure 4: Connection between Python application, Spark SQL and Hadoop



Source: Own elaboration

We must write down in the program, that the input data for the images, audio records, and pdf files are not binary data, but data in base64 coding. These changes are easy, because the tags for visualization in HTML5 support an option for input in base64 code.

The changed code in the Python module looks like that (Fig. 5).

Figure 5: Python code

```
…
html.Audio( src='data:audio/ wav;base64,{}'.format(CV_audio_base64) )
…
html.Img( src='{}'.format(CV_photo_base64) )
…
html.Iframe( src='data:application/pdf;base64,{}'.format(CV_pdf_base64)
)
```

Source: Own elaboration

3.4. Services for very large LOBS - Azure Blob Storage

The described way of working - storing large files as a BLOBs is effective when the files are no more than 2 gigabytes in size. If they become too large, then another technological solution must be sought. Microsoft offers such a solution for very large LOBs called Azure Blob Storage (Azure Blob Storage documentation online).

"The Azure Storage platform is Microsoft's cloud storage solution for modern data storage scenarios. Azure Storage offers highly available, massively scalable, durable, and secure storage for a variety of data objects in the cloud. Azure Storage data objects are accessible from anywhere in the world over HTTP or HTTPS via a REST API. Azure Storage also offers client libraries for developers building applications or services with .NET, Java, Python, JavaScript, C++". (Azure Blob Storage Introduction, 2023). So far, this storage for large BLOBs is the best developed and provided with proprietary processing algorithms from our point of view.

Vol. 75, issue 2 Year 2023
DOI: 10.56043/reveco-2023-0020

4. Conclusions

The most popular contemporary database systems, MS SQL, IBM DB2, Oracle, support special datatypes for very large data elements, named LOBs, with maximal size up to 2GB, or even larger in some systems (My SQL). The usage of LOBs is very effective in applications, where a large number of multimedia data is used. Saving the multimedia data as BLOBs in the database, instead of as separate files in the file system has three important benefits: (i) easier development process (ii) easier performers of backup / restore operations; (iii) easier security management.

Nowadays the technology of BLOBs gives us reason to answer Sears, R., Ingen C., Gray J. (2006) questions "To BLOB or Not to BLOB?" to be "Do BLOB, whenever we can".

The usage of LOBs is possible also in cases when the applications use Big Data Storage, (like Hadoop) The SQL interface engines, Spark SQL and IBM BigData SQL support, provide effective connection to LOBs in Big Data Storages.

References

- Base64: Guru. *Base64 Encode Algorithm*. Available at https://base64.guru/learn/base64-algorithm/encode, last accessed 2023/18/5.
- Handling Large Object (LOB) *Parameters in the CLR*. Available at https://learn.microsoft.com/en-us/sql/relational-databases/clr-integration-database-objects-types-net-framework/handling-large-object-lob-parameters-in-the-clr?view=sql-server-ver16, last accessed 2023/18/5.
- *IBM Support*. Available at https://www.ibm.com/support/pages/db2-version-115-linux-unix-and-windows-english-manuals , last accessed 2023/18/5.
- Kudo, Tsukasa, Ito Yuki, Yuki Serizawa: *An Application of MongoDB to Enterprise System Manipulating Enormous Data*. International Journal of Informatics Society, VOL.9, NO.3 (2017) 97-108. Available at http://www.infsoc.org/journal/vol09/IJIS_09_3_097-108.pdf, last accessed 2023/18/5.
- *Microsoft: Azure Blob Storage Introduction*. (2023) Available at https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction, last accessed 2023/18/5.
- Microsoft: *Binary Large Object* (2023) Available at https://learn.microsoft.com/en-us/sql/relational-databases/blob/binary-large-object-blob-data-sql-server?view=sql-server-ver16, last accessed 2023/18/5.
- *Oracle Database SecureFiles and Large Objects Developer's Guide*. Available at https://docs.oracle.com/cd/E18283_01/appdev.112/e18294/adlob_intro.htm , last accessed 2023/18/5.
- Parsian, Mahmoud: *Data Algorithms with Spark Recipes and Design Patterns for Scaling Up using PySpark.* (2022) Reilly Media.
- RFC 4648. (2006) https://datatracker.ietf.org/doc/html/rfc4648#section-4
- Sears, R., Ingen C., Gray J.: *To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?* Technical Report MSR-TR-2006-45. Microsoft Research, Microsoft Corporation. (2006). Available at https://www.microsoft.com/en-us/research/wp-content/uploads/2006/04/tr-2006-45.pdf , last accessed 2023/18/5.
- Stojanovic I., Bogdanov M., Bogdanova S.: *Searching of images stored in a database using content and pixel based methods*. 14th Telecommunications Forum TELFOR Belgrade, 2006. Available at https://eprints.ugd.edu.mk/871/1/Conf6_Telfor2006_IgorStojanovic.pdf, last accessed 2023/18/5.
- *SQL Server technical documentation.* Available at https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16, last accessed 2023/18/5.