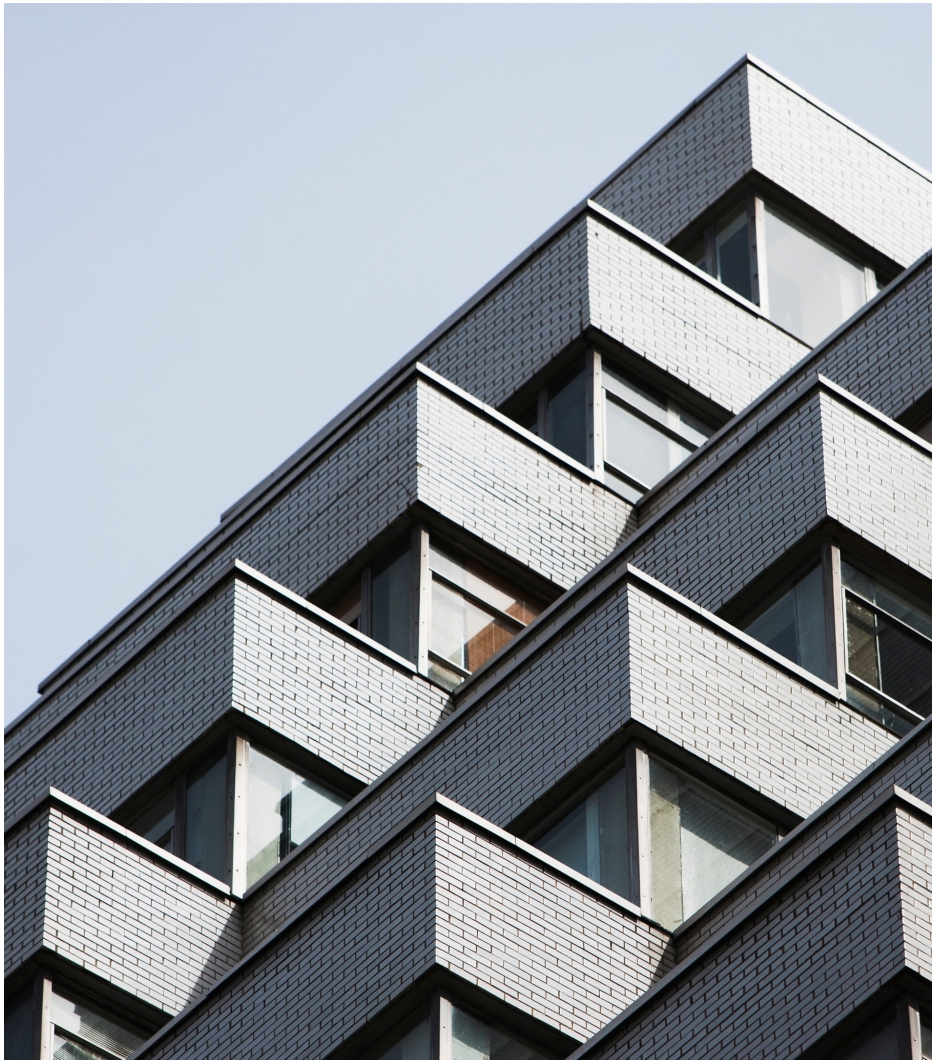
Demystifying LOBs in Db2 LUW

Chris Stojanovski

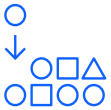


IBM

Objectives of this Presentation

Understand what LOBs are

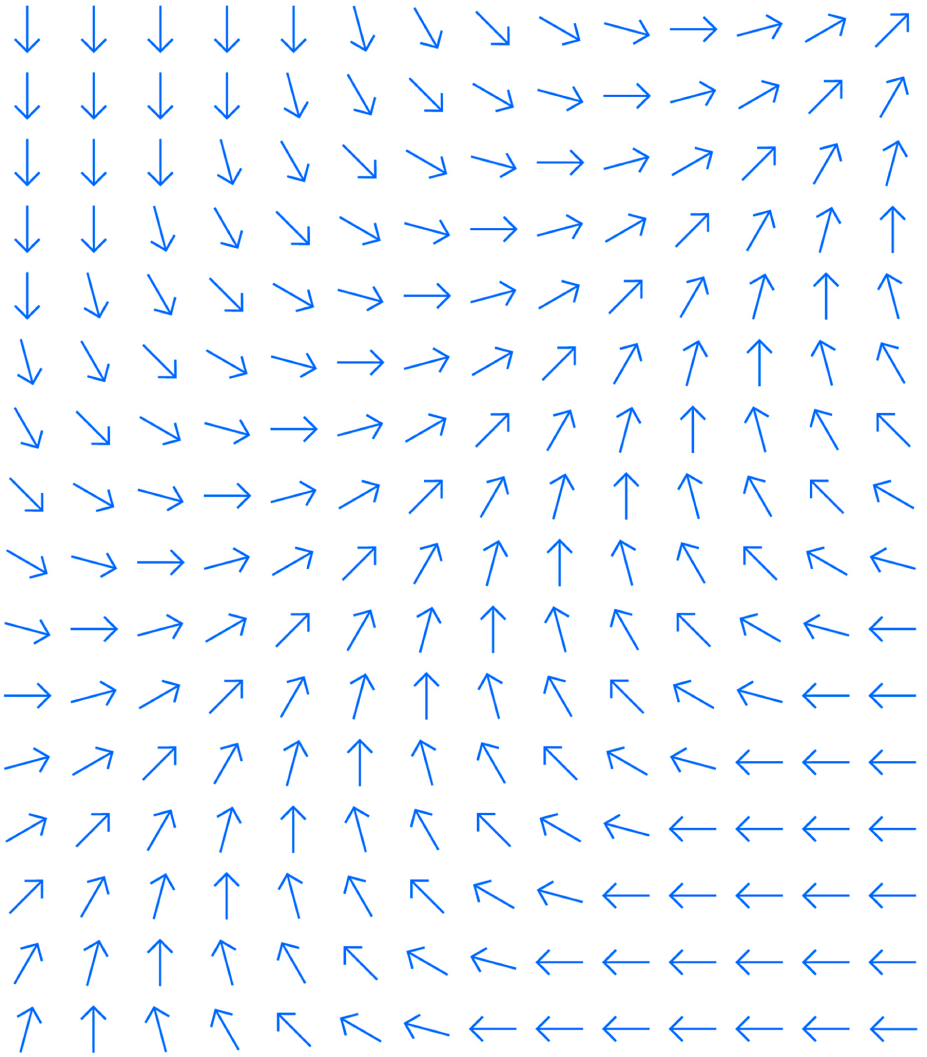
Understand how LOBs are used



Understand how LOBs are stored

Understand how to optimize LOBs

What are LOBs?



IBM

LOB → Large Object

Built-in
LOBs represent a built-in data type capable of storing large amounts of data

Data Store
Can be used to store video, audio, image data or act as a dump bucket

Special Considerations
Store data differently than most standard data types which requires special performance considerations.

What are the types of LOBs?

CLOB

Character Large Object – Character type that is compatible with the CHAR data type

DBCLOB

Double Byte Character Large Object – Character oriented type for large objects compatible with family of GRAPHIC data types

BLOB

Binary Large Objects- Represents a byte-oriented type that does not contain characters

What limitations do LOBs have?

Size

The maximum size of a LOB is 1 byte less than 2GB

Capacity

The total size of a lob object is limited to 4TB per table or per table partition.

Index

A LOB column can not be used as an indexing column

Where can I find LOBs in use?

Explicitly Created

- LOBs can be explicitly created in the create table statement
- LOBs can be added with an alter table

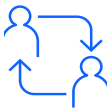
Extended Rows

- If a row is larger than the size of a page and extended row size is enabled
- Variable portion of the row (VARCHAR, VARBINARY, or VARGRAPHIC)

Catalog Tables

- Catalog tables use LOB types extensively to store information
- SYSCAT.TABLES, SYSCAT.INDEXES, SYSCAT.TRIGGERS, etc.

Creating a table with a LOB column:
What are all these options?



LOB Size
What is the maximum size of the LOB?

LONG IN...
Will your long columns be in the same tablespace?

Inline Length
What is the maximum size LOB that should be inlined?

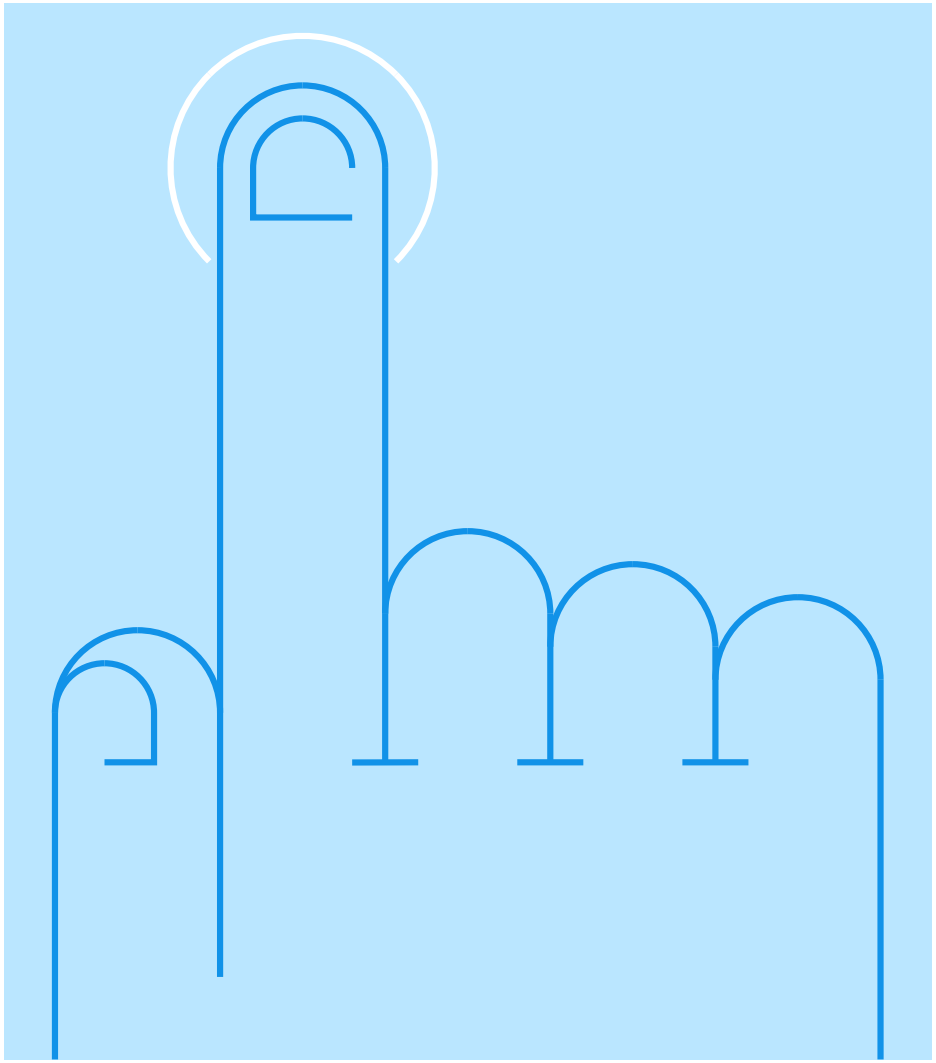
COMPACT
Should values in the LOB column take up minimal disk space?

LOGGED
Will changes that are made to the column be written to the log?

NOT LOGGED
Can data be lost during a restore and roll forward from backup?

IBM client stories

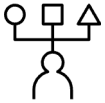
What operations can be performed on LOBs?



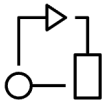
IBM

SQL Operations with LOBs

Insert



Update



Delete

Scalar Operations

Search

User Defined Function

What operations can be performed on LOBs?

LOB Operation → FETCH

Not Buffered

LOB pages differ from data pages as they are not buffered and therefore can not be fetched from the buffer pool.

LOBs that are in lined will be buffered like regular row data.

Direct Fetching

When a LOB needs to be read it this is done through direct disk I/O.

LOB reads are therefore slower, and performance considerations must be made.

Monitoring

Direct I/O activity can be observed through MON_GET_TABLESPACE by looking at:

- DIRECT_READS
- DIRECT_READ_REQS
- DIRECT_READ_TIME
- FS_CACHING

LOB Operation → UPDATE

Delete then Insert

Non-concatenation operations are treated as a DELETE followed by an INSERT

Concatenation

LOB manager supports concatenation in place – this is treated as a traditional UPDATE

Twice the Disk Space

Old data is left intact on disk (with protection) and new data is inserted elsewhere for an UPDATE.

This uses twice the disk space until deleted entries are cleaned up.

LOB Operation → Delete

Pending Delete

Marked pending deleted but left intact on disk with protection to not be over written or cleared.

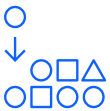
Shadowing

Deletions are not logged explicitly but we use a form of shadowing. What is logged is the marking of buddy segments as pending deleted.

Cleaning up

Subsequent inserts will review pending deleted buddy segments if they can be safely reused.

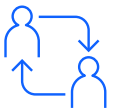
LOB support a number of data utilities including...



IMPORT

LOAD

EXPORT



REORG

IMPORT

Data File Limitations

- When LOB data is stored in the main input data file the size is limited to 32KB

LOBSINFILE

- Used to separate LOB data from main data file and avoid truncation

LOB Location Specifier

- Indicates where LOB data can be found by specifying the file name, offset, and length

What operations can be performed on LOBs?

LOAD

Similar Syntax to Import

– LOAD follows syntax like IMPORT using LOBSINFILE modifier

Efficient Data Handling

– Better able to handle large amounts of data efficiently

Bypassing Checks

– Able to bypass trigger firing, logging, and constraint checking

What operations can be performed on LOBs?

EXPORT

Data File Limitation

– LOBs exported to default data file are limited to 32 KB

LOBSINFILE

– Used to separate LOB data from main data file and avoid truncation

Single or Individual Files

– LOB data can be outputted to a single file using an LLS or individual files

What operations can be performed on LOBs?

REORG

Only for Space Reclaim

– LOB data is reorganized to reclaim space but can not be clustered

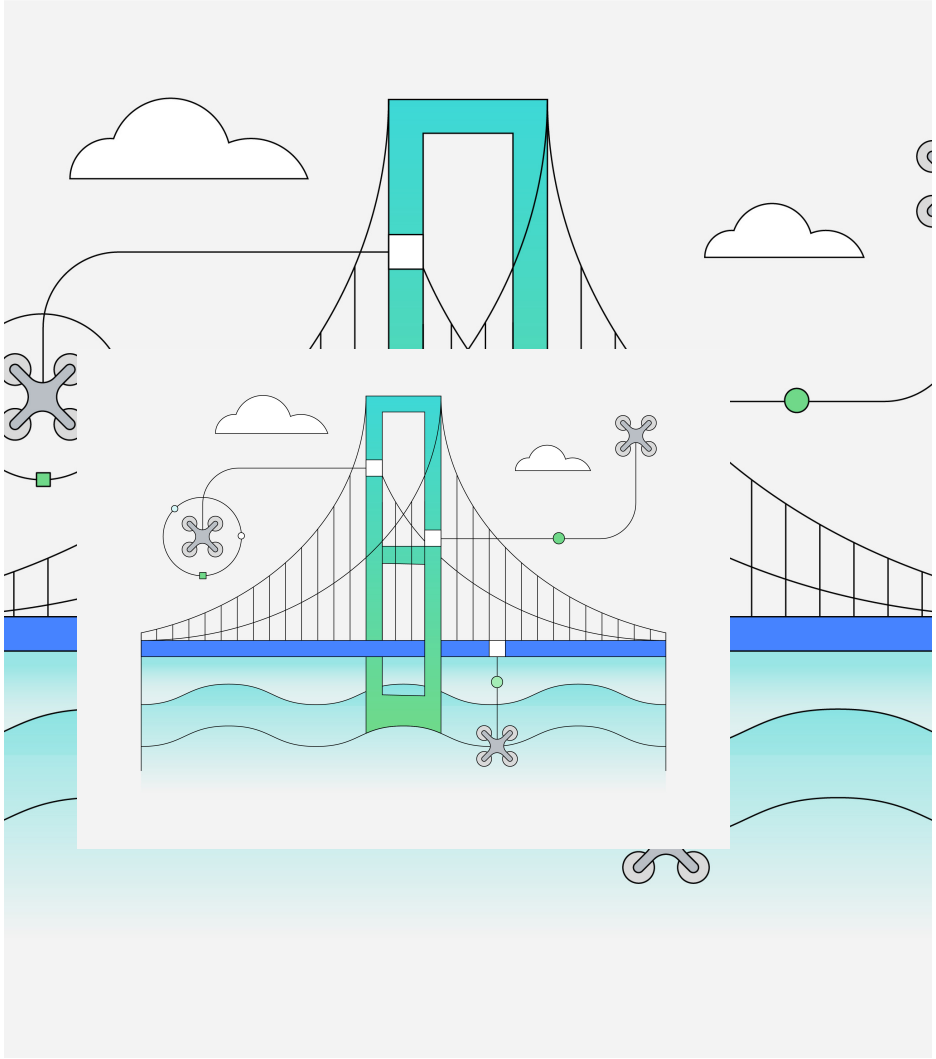
LONGLOBDATA

– LOB data will not be reorganized by default and must be explicitly requested

Only Offline REORG

– LOB data will only be reorganized during an offline REORG

How are LOBs stored?



IBM

The Basics

LOB Descriptor –
Pointer to LOB data

LB Object –
Raw user data

LBA Object –
Space allocation
structures

LOB Descriptor

Pointer

Pointer to locations of the LOB data in the LB object

Buffered

Stored within the formatted data row and therefore on data pages

Logged

All changes to LOB descriptor are logged regardless of LOGGED or NOT LOGGED

LB Object

Raw Data

Stores raw user data within blocks known as buddy segments

Not Buffered

Does not contain any BPS page header and does not flow through buffer pool

Partially Logged

Logging is dependent on LOB column setting

LBA Object

Allocation Structures

Stores information regarding the space usage within the LB object

Buffered

Allocation pages flow through buffer pool similarly to data pages

Logged

Completely logged as allocations change

Buddy Space

Block of Storage

Storage mechanism used by the LB object made up of blocks of at least 1KB known as buddy segments

Chain of Blocks

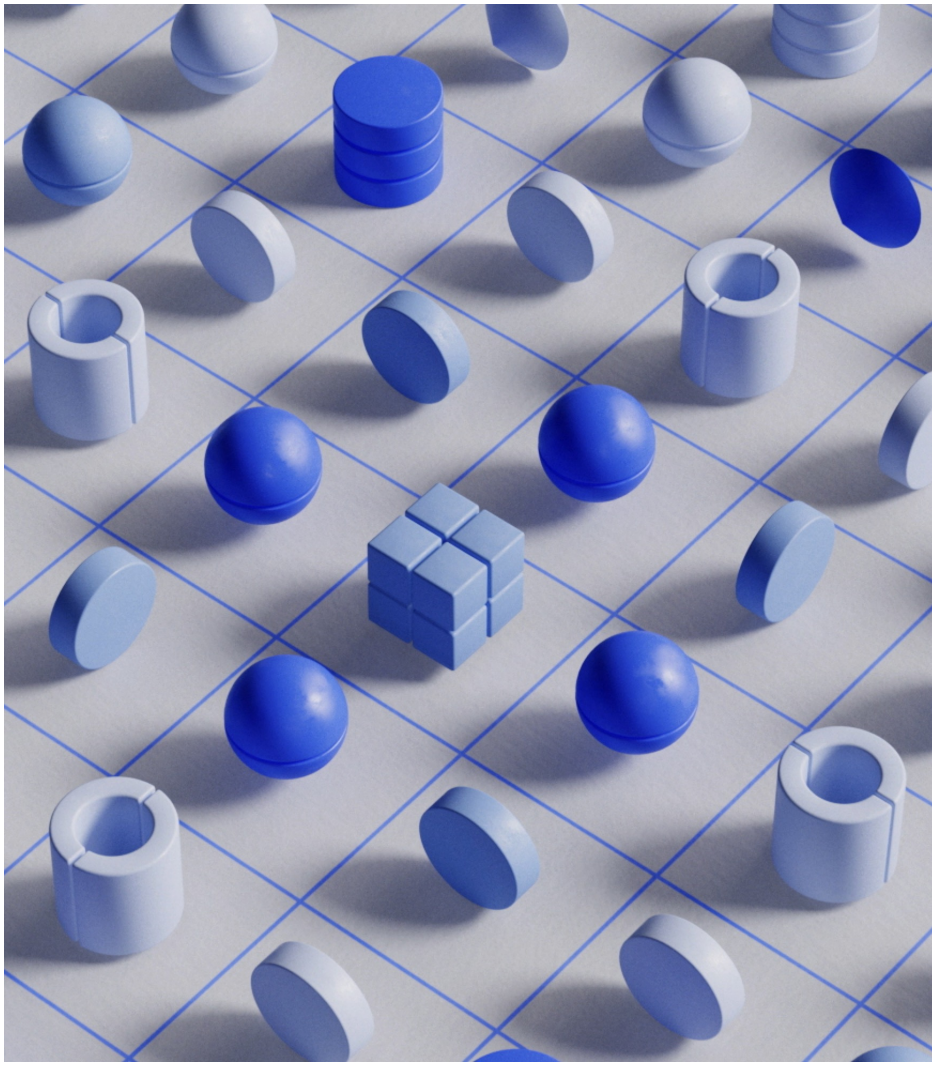
Data is stored in a chain of blocks where each block is 2 times larger than the previous if COMPACT is specified

Not Contiguous

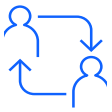
Blocks can be on different pages even after a REORG

IBM client stories

How to optimize LOB usage?



Creating a table with a LOB column:
How can we make best use of these options?



LOB Size
What is the maximum size of the LOB?

LONG IN...
Will your long columns be in the same tablespace?

Inline Length
What is the maximum size LOB that should be inlined?

COMPACT
Should values in the LOB column take up minimal disk space?

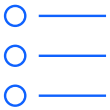
LOGGED
Will changes that are made to the column be written to the log?

NOT LOGGED
Can data be lost during a restore and roll forward from backup?

LOB Size

What does this specify?

- Maximum possible size of a LOB in the column
- This size can be up to 2G
- Default is 1M



What implications does this have?

- The larger the LOB the larger the LOB descriptor

Maximum LOB Length	Maximum LOB Descriptor Size
1,024	68
8,192	92
65,536	116
524,000	140
4,190,000	164
13,400,000	196
536,000,000	220
1,070,000,000	252
1,470,000,000	276
2,147,483,647	312

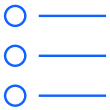
Can this value be changed?

- LOB size can be modified using an ALTER table
- This is limited to **increasing** the LOB size only

LONG IN...

Why specify a different table space?

- A LOB is made up of two objects an LBA and LB)
  - LBA – A structure used for buddy space allocation
  - LB – Stores the actual LOBs
- LBs are limited to 4TB regardless of page size



What if there are multiple LOBs on the table?

- All LOBs will be stored in the same table space
- There is one LBA / LB per table space regardless of the number of LOBs

What if data extends beyond 4TB?

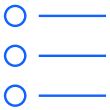
- Partitioned tables can be leveraged to extend past the 4TB limit
- Each partition can have a LB object of size 4TB

Table Space Considerations

Regular DMS Tablespace Limits

– Regular DMS tablespaces are limited to 64 GB with 4K pages

– This limit increases to 512 GB for 32K pages



Large DMS table space

– Large DMS tablespaces are limited to 8TB with 4k pages

- LOB data is still limited to 4TB even if DMS tablespace can store more

File System Caching

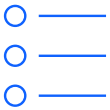
– LOB data is not cached in the buffer pool and is retrieved from disk on each subsequent read

– If LOB data is stored in SMS or DMS file containers file system caching might provide buffering

Inline Length

What does this specify?

- Maximum LOB size that will be in lined
- This data will be stored in the row and therefore buffered



What are the benefits of inline?

- Data pages are cached in the buffer pool
- LOBs can be inserted and retrieved with less disk I/O
- Can have a significant improvement on storage due to row compression

What is implicit versus explicit inlining?

- The minimum size for inlining is dependent on the size of the LOB descriptor and the overhead associated
- If an inline length is not chosen one will be implicitly set

Inline Length Illustrated

How was storage improved?

- Size before inlining : 110 GB
- Size after inlining and compression: 10 GB

How was performance improved?

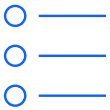
- Performance before inlining: 10 minutes to run audit report
- Performance after inlining: 2.5 minutes to run audit report (cold buffer pool) and 20 seconds (primed buffer pool)

Lob Size	Row Count
0	71,365,117
1 - 160	70,002,386
161 - 1000	28,219
1001 - 32000	1,332,581
32001+	1,931

Inline Length Best Practices

What are good candidate lengths?

- Choose an inline length that will fit within a specified data page size
- The size must consider other column sizes in the row



What LOBs should be inlined?

- LOBs are frequently accessed
- LOBs are not already compressed

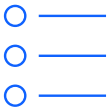
What admin table functions can help you?

- ADMIN_EST_INLINE_LENGTH – estimates the inline length required to inline the data
- ADMIN_IS_INLINED – reports if the LOB/XML documents in a column are inlined

COMPACT

How is data stored within the LB?

- Data is broken up into buddy segments
- There are regular buddy spaces (RBS) and super buddy spaces (SBS)
- Buddy segments come in 17 different sizes ranging from 1K to 64M



Why choose COMPACT?

- Compact uses less disk space
- When inserting a 11K LOB it will use 11K of space
- Can have higher I/O cost as blocks are not guaranteed to be contiguous

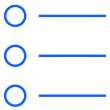
Why choose NOT COMPACT?

- Looks for one buddy segment that covers the space requirement
- Fewer buddy segments can mean fewer disk I/Os
- When inserting a 11K LOB it will use 16K of space
- The above example wastes 5k or roughly 30%

LOGGED

How does logging work?

- LOB columns are not logged in a single log record but broken into 32KB chunks
- All changes made to the LBA are logged regardless of setting
- User data log records are ignored during crash recovery regardless of setting



How does NOT LOGGED work?

- A small log record is written that includes the size and offset of the LOB data
- During roll forward recovery NOT LOGGED columns do not recover the original LOB column value
- Data is replaced by binary-0's equal to the size of the LOB

What happens in HADR?

- All logged LOB columns are replicated
- For non-logged LOB columns binary-0's will be written instead of the LOB data

Best Practices to Follow

01

LOB Size

Choose the minimum LOB size that works for your data

02

Inline Size

Tune the inline size for the best possible performance

03

Not Compact

To reduce fragmentation and if concatenation is a common use case

04

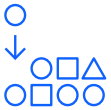
Partitioning

Can be used to extend beyond 4TB of LOBs per table

Objectives of this Presentation

Understand what LOBs are

Understand how LOBs are used



Understand how LOBs are stored

Understand how to optimize LOBs

Questions