LOB
The Unknown Creature

Dierk Lenz

Herrmann & Lenz Services GmbH

DOAG Konferenz 2018

Herrmann & Lenz Services GmbH
Herrmann & Lenz Solutions GmbH

Herrmann & Lenz Services

Herrmann & Lenz Solutions

- Founded in 1996
- Located in Burscheid (near Leverkusen)
- Consulting, Education and Remote Operations concerning Oracle and SQL Server Databases
- Focussed on high availability, performance tuning, migrations, troubleshooting and monitoring
- Herrmann & Lenz Solutions GmbH
  - Products: Monitoring Module, Taskzone
  - Booth at level 2 (236)

Why speak about LOBs?

- LOBs commonly used
- Applications often use LOBs „just like strings“
- Admins struggling with that ever growing database
- LOBs often stand for poor performance

- So: It‘s time for a general overview!
- Use it as a starting point!
- (40 minutes are not enough for everything you need to know!)

Contents

- Why use LOBs?
- How LOBs are implemented
- Using LOBs in Applications
- CLOBs and Unicode
- Conclusion

Why use LOBs?

Easy Answer

- Values too large for standard datatype columns, i.e. VARCHAR2
- *Large Object*: LOB
- Data model:

```
...
large_col CLOB,
...
```

- Some tests with the application, …
- Looking good, done!

More detailed answer

- LOBs not just *a little bit* bigger than standard data types
- Maximum size per LOB: between 8 TB and 128 TB (depending on block size)
- Everything is possible: text documents, images, raw data, movies, …
- Multiple LOB columns per table allowed

…and what about LONGs?

- LONGs and LONG RAWs available in versions before Oracle8i
- Strong limitations:
  - Only one LONG columns per table
  - „Only“ 2 GB per value
  - Stored with the rest of the row
- But also
  - Nearly 250 LONG columns in the data dictionary (SYS schema of a fresh 18c XE)
  - Still used in a lot of applications

How LOBs are implemented

LOB types

- Internal LOBs
  - Stored inside the database
  - Database mechanisms inplace: transactions, recovery, …
  - [N]CLOBs ([national] character LOBs): anything based on characters (and conversion of character sets)
  - BLOBs (binary LOBs): binary or raw data, no conversion
- External LOBs
  - BFILEs: LOB contents stored in files outside the database
    - Database contains path to files
    - External files not part of transactions, recovery, …

BasicFiles LOBs

- First LOB implementation (starting with Oracle8i)
- Still supported
- No new features
- Slower than…

SecureFiles LOBs

- New LOB implementation (starting with Oracle 11g)
- Optionally compressed, deduplicated, encrypted
  (if Advanced Compression and/or Advanced Security options of EE are available)
- Faster
- Same access, APIs, …

LOB segments

- LOB values not stored with the rest of the row
  - Performance: full table scan does not read LOB data!
- Exception: LOB value small (4000 Bytes) and `ENABLE STORAGE IN ROW` set
- LOB segment and LOB index segment per LOB column
- LOB values stored in LOB segment
- One value: one or more blocks
- Different values go into different blocks
- LOB index: storage details for the values

LOBs and block size

- Bigger block sizes more efficient for large LOBs
- Overhead for small LOBs:
  - Block size = 16 KB
  - LOB size 5 KB
  - 11 KB of unused space per block
- LOB design:
  - Use LOB tablespaces, optionally with non-standard block size
  - Small block size for small LOBs
  - Big block size for large LOBs
- BasicFiles LOBs: chunk size (multiple of block size) could be definied, no longer used for SecureFiles

LOBs and caching

- Historical note:
  - LOBs introduced with Oracle8i
  - RAM sizes of hundredths of GB **not available**
  - LOBs in the buffer cache would have destroyed database performance
- Option per LOB: `NOCACHE` | `CACHE` | `CACHE READS`
- `NOCACHE`
  - LOB data does not hit the buffer cache on DML and SELECT
  - direct path read/write events on LOB operations
- `CACHE`: LOB operations use buffer cache
- `CACHE READS`: only LOB SELECTs use buffer cache

LOBs and caching (cont.)

- On large caches: LOB caching possible
- Non-standard block size for LOB tablespaces: extra sizeable LOB cache!
- CACHE for LOBs:
  - DBWR does the writes!
  - Use multiple DBWR processes

LOBs and Data Pump

- LOBs slow when exported / imported
- Reasons:
  - No Array Interface used
  - (No-)Caching effects
- Special solution: HLTablePump (multi-threaded, high-speed)

LOBs and transactions

- LOB data does not go into undo segments
- Old versions of LOBs stay in LOB segment
- Frequent modifications of LOBs may **dramatically increase** LOB segment size!

Performance Example

- Testing CACHE/NOCACHE and BasicFiles/SecureFiles LOBs: 1000 LOB INSERTs
- Laptop, Oracle Linux on Virtualbox VM, SSD

	NOCACHE	CACHE
SecureFiles	4,01sec	0,24sec
BasicFiles	6,74sec	0,25sec

- Oracle Database Appliance X6-2 S, NVMe SSDs

	NOCACHE	CACHE
SecureFiles	0,15sec	0,11sec
BasicFiles	0,25sec	0,16sec

Using LOBs in Applications

LOB Restrictions

- Restrictions apply for
  - SQL functions
  - Remote operations
- LOB columns cannot be indexed

The Data Interface

- The standard bind variable interface may be used for LOBs
- Applications may use same interfaces used for LONGs
- Advantage: easy to use
- Disadvantage: not well suited for large LOBs, piecewise manipulation, …

The Locator Interface

- Supported by most programming environments
- For example: PL/SQL, OCI, OCCI, JDBC, ODP.NET
- Piecewise operations (i.e. manipulating a small part of a huge LOB)
- LOB locator: reference to the LOB itself
- Dereferencing is implicit
- DBMS_LOB package

Read the Documentation!

- Database SecureFiles and Large Objects Developer‘s Guide
- And also other books like
  - Database Globalization Support Guide
  - Database PL/SQL Packages and Types Reference

CLOBs and Unicode

Standard database character set: AL32UTF8

- AL32UTF8 as Oracle‘s up-to-date UTF-8 implementation
- UTF-8:
  - Variable length Unicode implementation
  - From 1 Byte (7 Bit ASCII characters) to 5 Bytes (special characters)
- „Variable length“ does not fit with „piecewise LOB manipulation“ („replace characters from position n to m with …“)
- CLOBs in a Unicode database use UCS-2!
- UCS-2: 2 Byte, fixed length
- Result: Moving from single byte character set to Unicode ***doubles*** space used for CLOBs!

Using Extended String Size

- Introduced with Oracle 12c: optionally increased limit of VARCHAR2s from 4000 Bytes to 32 KB (also for NVARCHAR2 and RAW)
- Database needs modification:
  - Set max_string_size = EXTENDED
  - Run $ORACLE_HOME/rdbms/admin/utl32k.sql in UPGRADE mode
  - Not possible to go back! (PLEASE TEST!!!)
- VARCHAR2 strings larger than 4000 Bytes use LOB technology
  - All implicit, no LOB parameters, no extra tablespace
  - Database characterset used, also when AL32UTF8!
  - Indexes possible!
  - When „ORA-01450: maximum key length exceeded“ happens: move indexes to tablespace with larger block size

Conclusion

- Use Securefiles LOBs (with current database releases!)
- When using LOBs: know the options!
- Developers: know the APIs!
- DBAs and Developers: work together!
- Test! Test! Test!

Questions & Contact

- Mail: dierk.lenz@hl-services.de
- Web: www.hl-services.de
- Blog: blog.hl-services.de
- Twitter: @ora1578
- Live: Booth 236, level 2 (yellow)

Thank you for listening!

Also attend:

Endlich erwachsen! Ein SQL Server DBA wechselt die Seiten
Sebastian Röhrig, 20.11.2018, 14:00h, Seoul