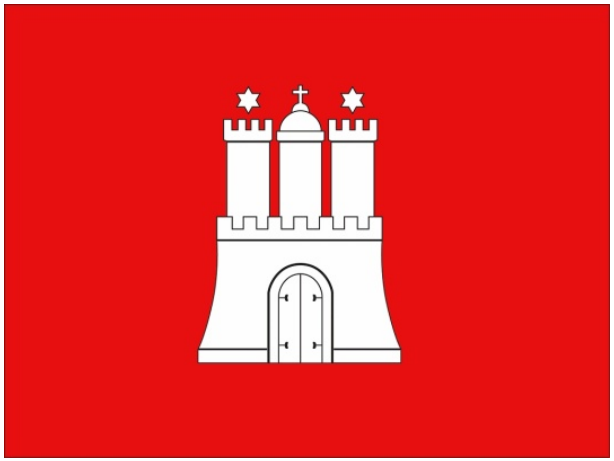
HERZLICH WILLKOMMEN

LOB Komprimierung mit Oracle 11g

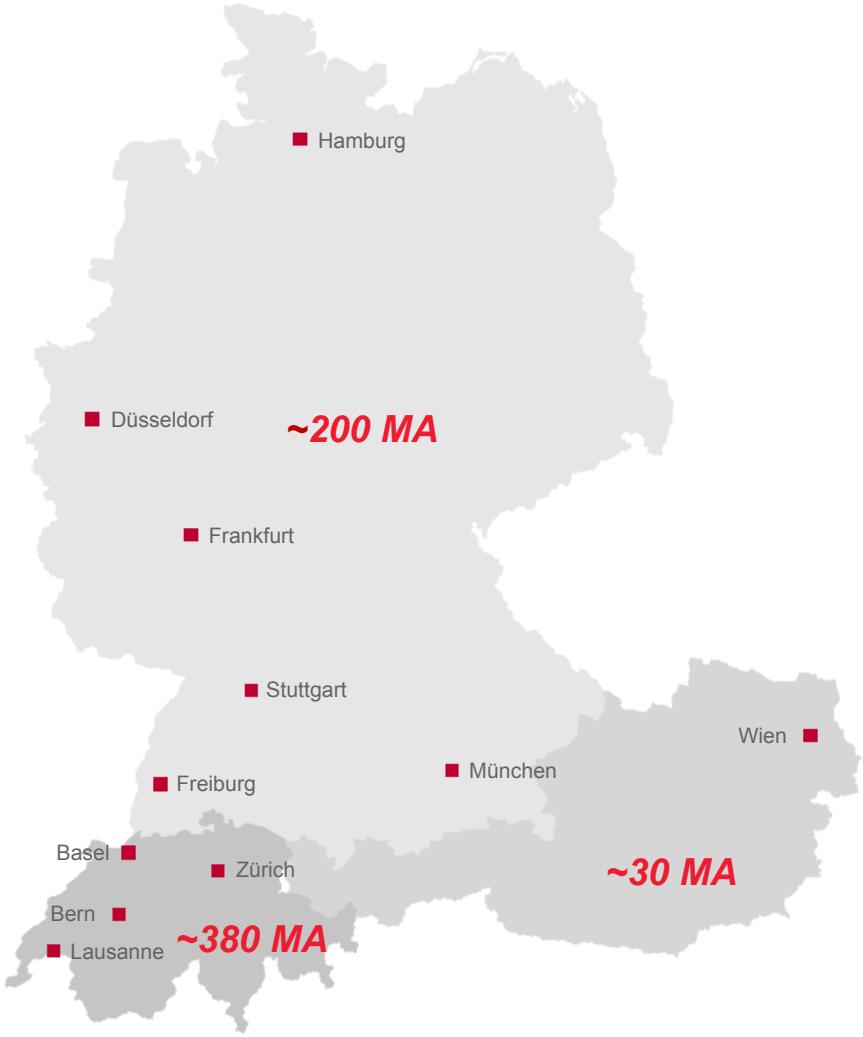
BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

trivadis
makes IT easier.

Christoph Jansen
Consultant

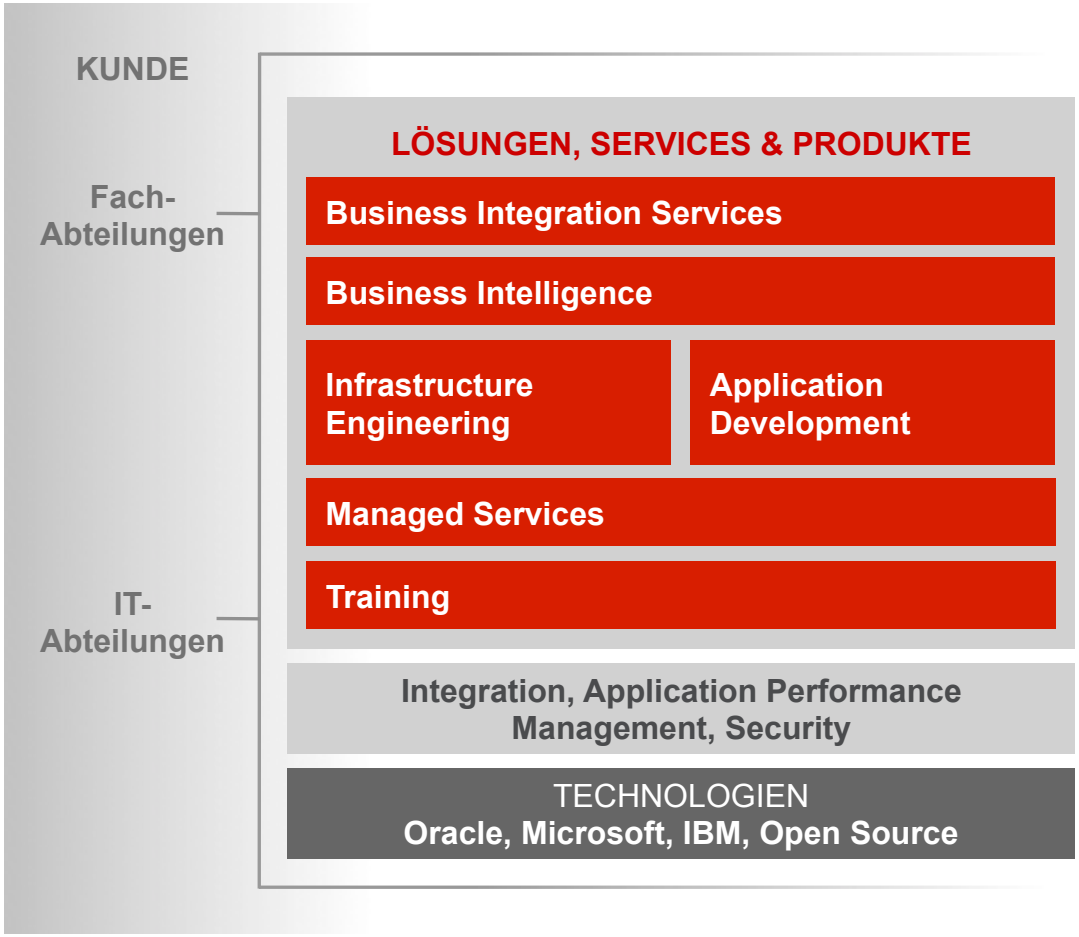


Trivadis Facts & Figures



trivadis
makes IT easier.

Trivadis Lösungsportfolio



Agenda

1. Secure Files
2. Komprimierung
3. Performance und Empfehlungen

Large Objects - LOBs

- Textdateien, Worddokumente, Bilder usw.
- CMS / DMS
- Die private Video-Datenbank

Secure Files neu in 11g

- Performance wie Dateisystem
- Transparent
- Unterstützt: NCLOB, CLOB, BLOB

```
CREATE TABLE t (
     x INT PRIMARY KEY,
     y BLOB
     )
LOB (y) STORE AS SECUREFILE (TABLESPACE ts1)
```

Secure Files neu in 11g

- Verschlüsselung (ASO)
- Deplulizierung (ACO)
- Komprimierung (ACO)
- FILESYSTEM_LIKE_LOGGING

BasicFile vs. SecureFile

Hard-ware	Lesen, Vergleich von 2 LOBs mittels dbms_lob. compare (2 x 100MB)		Lesen eines LOBs 100 MB mittels JDBC thin		Schreiben des LOBs mittels dbms_lob. loadfromfile		Lesen und Schreiben, Kopie eines LOBs innerhalb des selben Segments mittels insert into … select …	
	BF	SF	BF	SF	BF	SF	BF	SF
A1 IO	1,8 s	1,2 s	3,6 s	3,2 s	4,2 s	1,5 s	5,2 s	3,6 s
A2	17,9 s	5,1 s	12,2 s	4,9 s	4,7 s	4,1 s	15,2 s	6,3 s
B1	4,0 s	11,5 s	2,9 s	3,6 s	22,1 s	79,0 s	12,5 s	79,7 s
B2 IO	2,6 s	1,5 s	2,9 s	3,5 s	8,3 s	75,3 s	6,7 s	76,3 s

A1: Linux x86, 2x Intel Xeon CPU 2.40GHz, I/O Durchsatz mit dd: 80 MB/s
A2: wie A1, Datenfiles jetzt auf nfs, I/O Durchsatz nun 40 MB/s
B1: Solaris 10, Sun Fire V210, 2x UltraSPARC-IIIi 1336 MHz, I/O Durchsatz mit dd: 48 MB/s.
B2: wie B1, Datenfiles jetzt auf einer Ramdisk

Agenda

1. Secure Files
2. Komprimierung
3. Performance und Empfehlungen

SecureFiles – LOB Komprimierung

```
LOB (y) STORE AS SECUREFILE
([NOCOMPRESS | COMPRESS [LOW | MEDIUM | HIGH]])
```

- Komprimierung in kleinen Chunks (256KB)
  - Ermöglicht Random Reads
- Algorithmus basiert auf ZLIB
  - Oracle Kernel Funktion: kgcczlibdo
- LOW Komprimierung basiert auf LZO
  - Oracle Kernel Funktion: kgcclzodo

Underscores zur Steuerung

- _kdlxp_lobcompress = TRUE | FALSE
- _kdlxp_lobcmplevel = 1 | 2 | 3
- _kdlxp_cmp_subunit_size
- _kdlxp_lobdeduplicate = TRUE | FALSE

Ermittlung Platzbedarf

100 MB Text File

```
# ls -al
-rw-r--r-- 1 oracle dba 105000720 Jul 14 11:57 large_txt_file.txt
-rw-r--r-- 1 oracle dba 27063480 Jul 15 16:15 large_txt_file.zip
```

als LOB

```
SELECT segment_name, bytes FROM
user_segments
WHERE segment_type = 'LOBSEGMENT';

SEGMENT_NAME                   BYTES
------------------------------ ----------
DATABLOB_BF                    109051904
DATABLOB_SF                    189005824
DATABLOB_COMP_MED              36962304
DATABLOB_COMP_HIGH             31719424
```

Ermittlung Platzbedarf

100 MB Text File

```
# ls -al
-rw-r--r-- 1 oracle dba 105000720 Jul 14 11:57 large_txt_file.txt
-rw-r--r-- 1 oracle dba 27063480 Jul 15 16:15 large_txt_file.zip
```

als LOB

```
SELECT segment_name, bytes FROM
user_segments
WHERE segment_type = 'LOBSEGMENT';

SEGMENT_NAME                   BYTES
------------------------------ ----------
DATABLOB_BF                    109051904
DATABLOB_SF                    189005824
DATABLOB_COMP_MED              36962304
DATABLOB_COMP_HIGH             31719424
```

Starke Überallozierung bei SecureFiles

Echter Platzbedarf?

Ermittlung Platzbedarf: DBMS_SPACE

```
CREATE OR REPLACE PROCEDURE lobsegmentsize(lobsegment_name VARCHAR2)
AS
  segment_size_blocks NUMBER;
  segment_size_bytes NUMBER;
  used_blocks NUMBER;
  used_bytes NUMBER;
  expired_blocks NUMBER;
  expired_bytes NUMBER;
  unexpired_blocks NUMBER;
  unexpired_bytes NUMBER;
BEGIN
  dbms_space.space_usage(user, lobsegment_name, 'LOB',
  segment_size_blocks, segment_size_bytes, used_blocks, used_bytes,
  expired_blocks, expired_bytes, unexpired_blocks, unexpired_bytes);
  dbms_output.put_line(lobsegment_name);
  dbms_output.put_line('------------------');
  dbms_output.put_line('segment_size_bytes = ' || segment_size_bytes);
  dbms_output.put_line('used_bytes = ' || used_bytes);
END;
/
```

Ermittlung Platzbedarf: DBMS_SPACE

```
SQL> EXEC lobsegmentsize('DATABLOB_SF');
DATABLOB_SF
-----------------
segment_size_bytes = 189005824
used_bytes = 106741760

PL/SQL procedure successfully completed.

SQL> EXEC lobsegmentsize('DATABLOB_COMP_MED');
DATABLOB_COMP_MED
-----------------
segment_size_bytes = 36962304
used_bytes = 32235520

PL/SQL procedure successfully completed.

SQL> EXEC lobsegmentsize('DATABLOB_COMP_HIGH');
DATABLOB_COMP_HIGH
-----------------
segment_size_bytes = 31719424
used_bytes = 27574272

PL/SQL procedure successfully completed.
```

Platzersparnis durch LOB Komprimierung

- Komprimierungsrate abhängig von
  - Komprimierungsfaktor
  - Komprimierbarkeit der „Rohdaten“

Agenda

1. Secure Files
2. Komprimierung
3. Performance und Empfehlungen

Medium Komprimierte LOBs – 2 x 100 MB

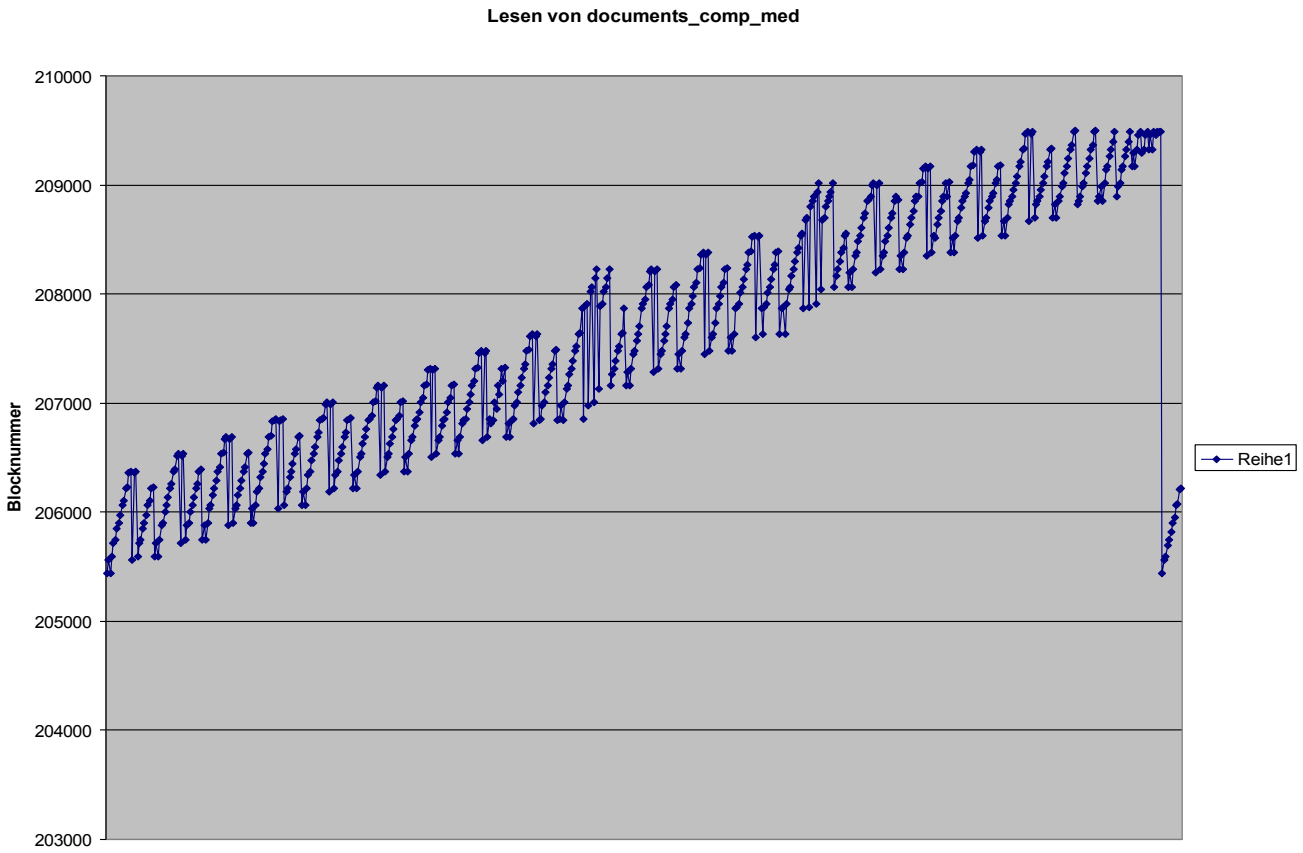
```
DECLARE
  lob_1 BLOB;  lob_2 BLOB;  retval INTEGER;
BEGIN
  SELECT data INTO lob_1 FROM documents
  WHERE id=1 AND version=1;
  SELECT data INTO lob_2 FROM documents
  WHERE id=1 AND version=2;
  retval := dbms_lob.compare(lob_1, lob_2);
  IF retval = 0 THEN
    dbms_output.put_line('equal');
  ELSE
    dbms_output.put_line('not equal');
  END IF;
END;
/

equal
PL/SQL procedure successfully completed.
Elapsed: 00:00:47.76

SQL> REM Wieviel wurde gelesen?
SQL> SELECT name, value FROM v$mystat NATURAL JOIN v$statname WHERE
     name = 'physical read bytes';
NAME                             VALUE
-------------------------------- ----------------
physical read bytes              11993120768
```

> 11 GB

SecureFiles – Lesen von Komprimierten LOBs



- Größe des LOBs unkomprimiert: 100 MB
- Größe des LOBs im Segment: 32 MB
- physical Reads: 158 MB

Derselbe Test mit verschiedenen Komprimierungen mit und ohne Caching

		NOCACHE		CACHE READS			
LOB Einstellungen	LOB Größe in MB	Physical Reads in MB	Zeit in s	Physical Reads in MB	Zeit in s	I/O Einsparung in Prozent	Zeit Einsparung in Prozent
NOCOMPRESS KEEP_DUPLICATES	204	204	1,3	204	2,9	0	-123
COMPRESS MEDIUM KEEP_DUPLICATES	61	11438	53,0	61	19,7	99,5	62,8
COMPRESS HIGH KEEP_DUPLICATES	53	10553	48,7	53	18,6	99,5	61,8
NOCOMPRESS DEDUPLICATE	102	12424	38,5	102	2,2	99,2	94,3
COMPRESS MEDIUM DEDUPLICATE	31	882	6,5	31	4,9	96,5	24,7
COMPRESS HIGH DEDUPLICATE	26	791	7,0	26	4,4	96,7	37,1

Schreibperformance Messungen

```
SQL> REM neu connecten um v$mystat zu "initialisieren"
SQL> conn scott/tiger
Connected.
SQL> INSERT INTO documents_comp_med SELECT * FROM documents;

2 rows created.

Elapsed: 00:00:26.32
SQL> commit;

Commit complete.

Elapsed: 00:00:00.11
SQL> SELECT name, value FROM v$mystat NATURAL JOIN v$statname
2    WHERE name IN ('physical read bytes', 'physical write bytes', 'redo size');
NAME                           VALUE
------------------------------ ----------
physical read bytes             213950464
physical write bytes             64438272
redo size                          240432
```

Schreiben mit verschiedenen Komprimierungen mit und ohne Caching

		NOCACHE			CACHE		
LOB Einstellungen Zieltabelle	LOB Segment Größe in MB	Physical Writes in MB	Redo Size in MB	Ø Zeit in s	Physical Writes in MB	Redo Size in MB	Ø Zeit in s
NOCOMPRESS KEEP_DUPLICATES	204	204	0,6	8,6	0	206	66,6
COMPRESS MEDIUM KEEP_DUPLICATES	61	61	0,3	24,0	0	62	34,2
COMPRESS HIGH KEEP_DUPLICATES	53	53	0,2	70,4	0	53	66,3
NOCOMPRESS DEDUPLICATE	102	102	0,3	7,9	0	103	36,5
COMPRESS MEDIUM DEDUPLICATE	31	31	0,1	26,9	0	31	22,9
COMPRESS HIGH DEDUPLICATE	26	26	0,1	63,6	0	26	58,2

Schreibperformance – Erkenntnisse

- Komprimierung kostet CPU und Zeit
- Caching
  - Keine Phyiscal Writes
  - Abhängig von LGWR Performance (multiplexed Redo Log Members)
  - Caching von Medium und Uncompressed nicht sinnvoll
  - Caching bei High Compression und Dedup sinnvoll

Empfehlungen beim Setup von komprimierten LOBs

- Nutzung von leistungsfähiger Hardware
- In-Line oder Out-Line?
  - Majorität der Queries
- Eigener Tablespace für große LOBs mit großer Blocksize (32 KB)
- CACHE READ für Komprimierte LOBs
- FILESYSTEM_LIKE_LOGGING
- Testen und Verifizieren für beste Performance

Referenz

http://www.trivadis.com/uploads/tx_cabagdownloadarea/LOB_Komprimierung_mit_Oracle_11g.pdf

VIELEN DANK.

Trivadis GmbH

Christoph Jansen

Paul-Dessau-Straße 6
Hamburg

Tel. +49-40-24 85 91 30
Fax +49-40-24 85 91 59

christoph.jansen@trivadis.com
www.trivadis.com

BASEL BERN LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

trivadis
makes IT easier.