Oracle Advanced Compression

in Database 11g Rel. 2: Value/Performance

Hybrid Columnar Compression

in Database 11g Rel. 2: Value/Performance on Exadata V2

Daniel A. Morgan | damorgan12c@gmail.com | www.morganslibrary.org

Daniel A. Morgan

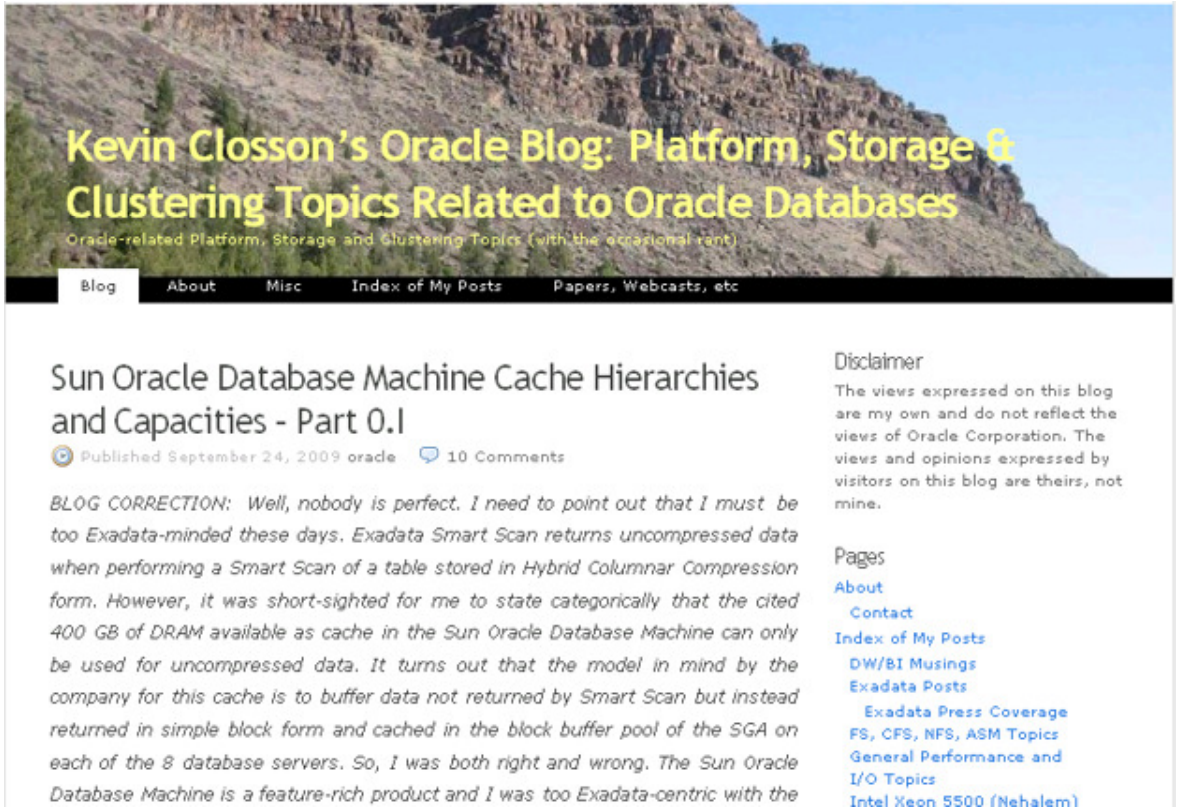
- Oracle ACE Director
- Consultant to Harvard University
- University of Washington Oracle Instructor, ret.
- The Morgan of Morgan's Library on the web
- Board Member: Western Washington OUG
- Upcoming Presentations
  - Mar 28: Vancouver B.C.
  - Apr 24: Benelux Connect (Netherlands)
  - May 30-31: Harmony Finland
  - Jun 1: Harmony Latvia

Official Beta Site ORACLE DATABASE 11g ORACLE RACSIG International zSeries Oracle SIG

Kevin Closson

http://kevinclosson.wordpress.com

- kevin.closson@oracle.com
- Oak Table Network
- Oracle Employee Ace
- Performance Architect Exadata Development



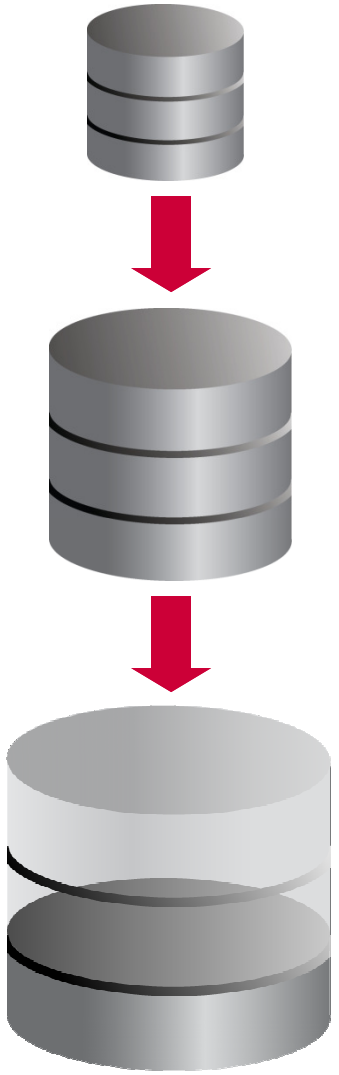
Agenda

- Why so much interest in compression?
- A brief history of Oracle Database Compression
  - Index Compression
  - Data Segment Compression
  - LOB Compression
  - Advanced Compression in 11gR1
- Advanced Compression in 11gR2
- Hybrid Columnar Compression & Exadata V2

My Favorite Customers

Why Compress Segments?

- Explosion in Data Volumes
  - Regulatory and audit requirements
  - Online content
- As data volume expands performance often declines
- Disk costs money
- Powerful and efficient compression is key



What Is Traditional Compression?

- A trade-off between CPU and Disk I/O
  - The use of spare CPU cycles to decrease the bytes written and read
- First introduced in Oracle 9.2.0.1
- Transparent to applications, SQL, and PL/SQL
- Can improve performance by requiring the transfer of fewer bytes from disk through the network, into the CPU, to be stored in the buffer cache
- Increase the amount of data stored on existing disk

How Traditional Compression Works

- A grossly oversimplified "how it works"
  1. Oracle examines full blocks for duplicates
  2. Creates a symbol that is stored in the block header
  3. Rewrites the block substituting the symbol for the values it represents
- Compression is performed at the block level
  not table level as in DB2

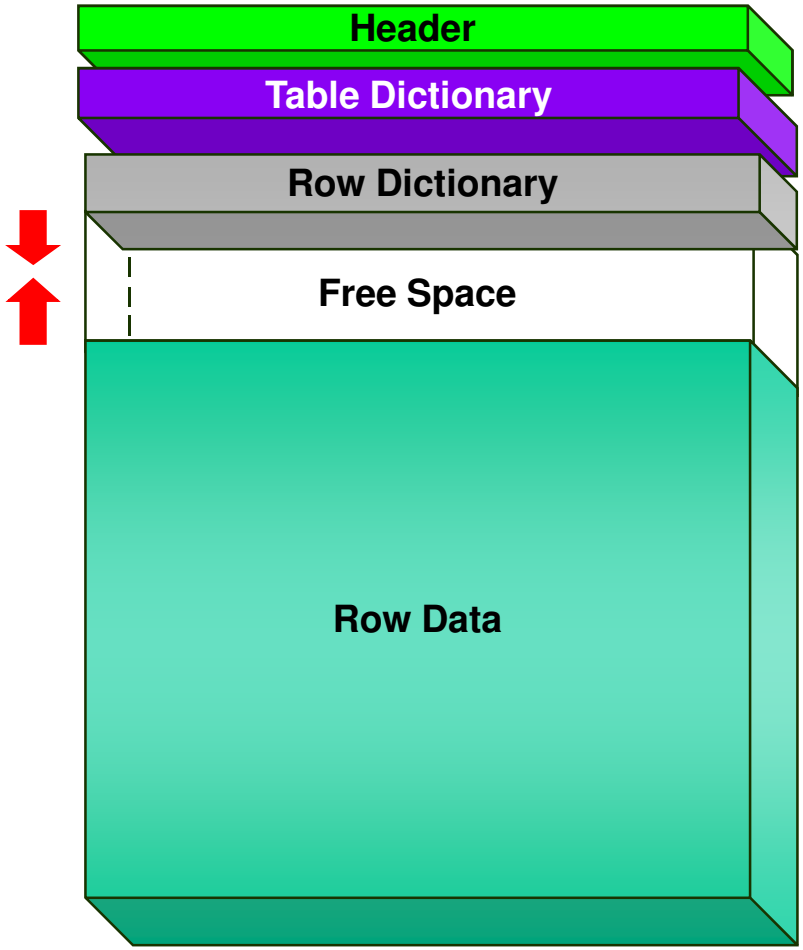
City	State	Postal Code
Hot Springs National Park	AR	71901
Hot Springs National Park	AR	71902
Hot Springs National Park	AR	71903
Hot Springs National Park	AR	71913

128 bytes

City	State	Postal Code
Hot Springs National Park	AR	71901
"	"	"02
"	"	"03
"	"	"13

38 bytes

Database Block Anatomy



General block information

Table info in clusters

Row info in block (2 bytes/row)

Used when a new row is inserted or an existing row is updated

Segment data (table or index)

9.2 Index Compression

- Most often used with multi-column indexes to compress duplicates in leading columns

```
CREATE INDEX ix_serv_inst
ON serv_inst (srvr_id, custacct_id);

ANALYZE INDEX ix_serv_inst VALIDATE STRUCTURE;

SELECT opt_cmpr_count, opt_cmpr_pctsave
FROM index_stats;

SELECT sum(bytes)
FROM user_segments
WHERE segment_name = 'IX_PCODES';


OPT_CMPR_COUNT OPT_CMPR_PCTSAVE
-------------- ----------------
             1               10
```

9.2 Data Segment Compression

- Heap Organized Tables
- Materialized Views

```
CREATE TABLE reg_tab AS
SELECT *
FROM dba_tables;

CREATE TABLE COMPRESS comp_tab AS
SELECT *
FROM dba_tables;

exec dbms_stats.gather_table_stats(USER, 'REG_TAB');
exec dbms_stats.gather_table_stats(USER, 'COMP_TAB');

SELECT table_name, blocks
FROM user_tables
WHERE table_name LIKE '%TAB';

SELECT table_name, blocks FROM user_tables WHERE table_name LIKE '%TAB';

TABLE_NAME                       BLOCKS
------------------------------ ----------
REG_TAB                             109
COMP_TAB                             20
```

10.1 LOB Compression

- UTL_COMPRESS Built-in Package

```
DECLARE
 b      BLOB;
 r      RAW(32);
 handle BINARY_INTEGER;
BEGIN
  SELECT iblob
  INTO b
  FROM test
  WHERE fname = 'Uncompressed'
  FOR UPDATE;

  handle := utl_compress.lz_compress_open(b);

  IF NOT utl_compress.isopen(handle) THEN
    RAISE NO_DATA_FOUND;
  END IF;

  r := utl_raw.cast_to_raw('ABC');
  utl_compress.lz_compress_add(handle, b, r);
  utl_compress.lz_compress_close(handle, b);
END;
/
```

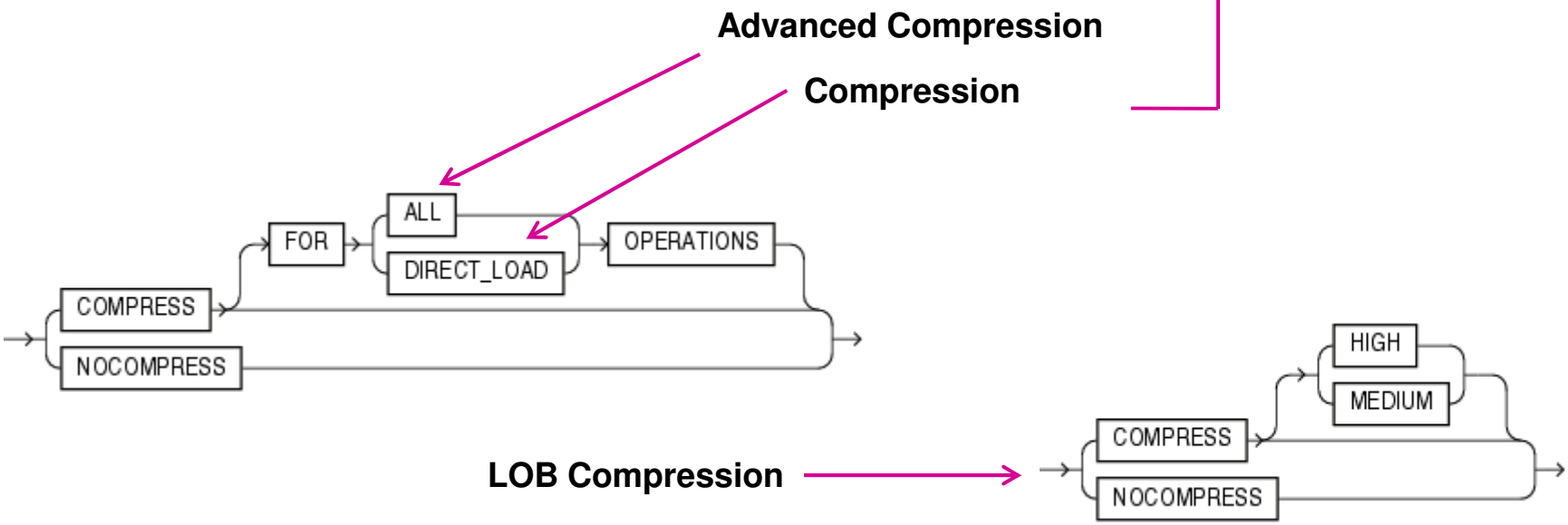
- No significant changes in 10gR2

11.1 Compression

- Index and Segment Compression
- The Advanced Compression Option includes
  - Data Guard Network Compression
  - Data Pump Compression
  - Fast RMAN Compression
  - OLTP Table Compression
  - SecureFile Compression and Deduplication
    - Leveraged in 11gR2 DBFS (DataBase File System)

11.1 Many Options

- Compressed Tablespaces
- Segment Compression
  - COMPRESS
  - COMPRESS FOR DIRECT_LOAD [OPERATIONS]
  - COMPRESS FOR ALL [OPERATIONS]
- user_tablespaces.compress_for column



SecureFiles

- Part of the Advanced Compression option

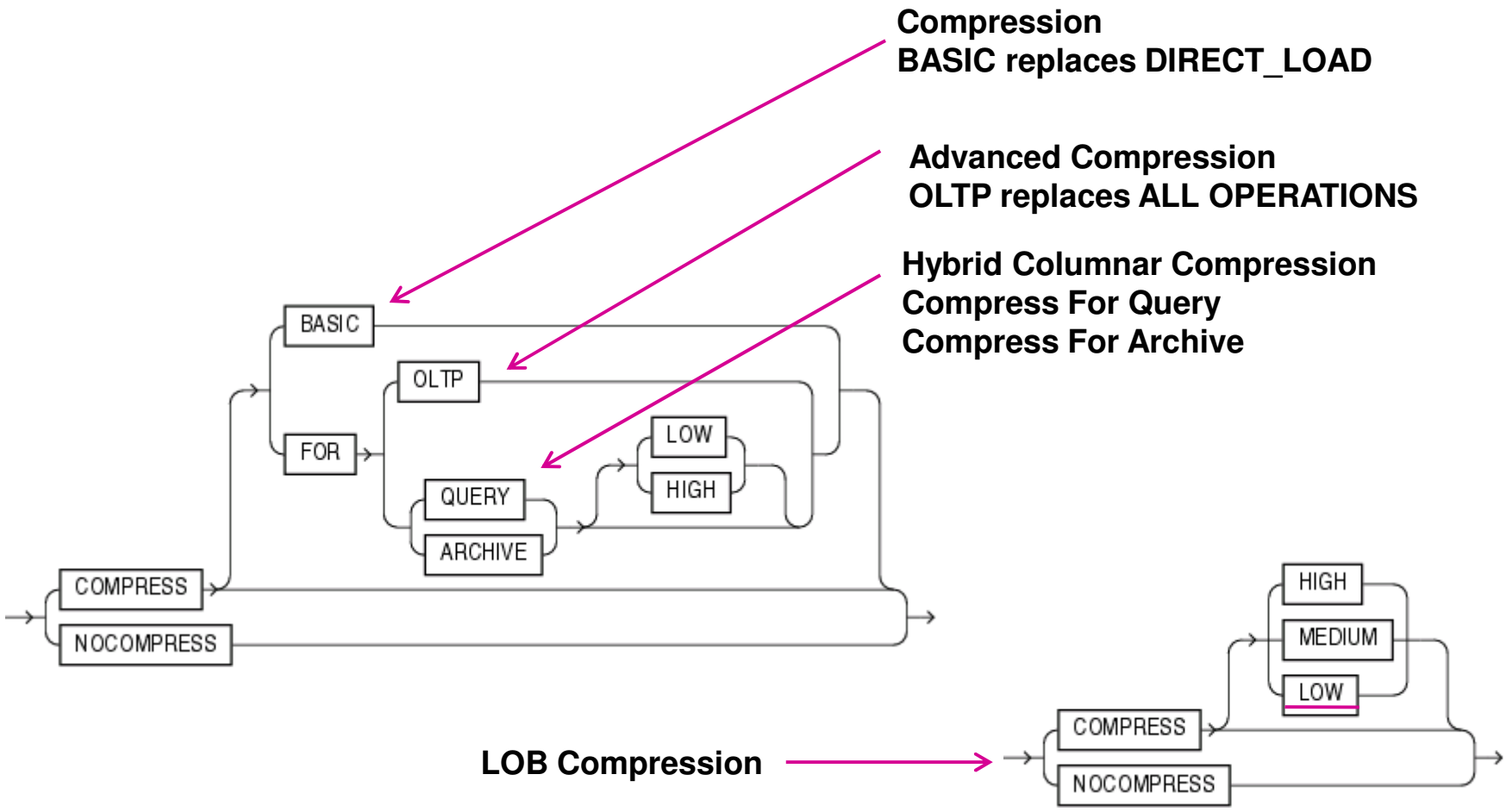
```
CREATE TABLE secfile_table (
rid  NUMBER(5),
bcol BLOB)
LOB (bcol)
STORE AS SECUREFILE bcol2 (
  TABLESPACE securefiletbs
  RETENTION MIN 3600
  COMPRESS ENCRYPT CACHE READS)
TABLESPACE uwdata;
```

11.2 Compression

- Segment Compression
- The Advanced Compression Option includes
  - Data Guard Network Compression
  - Data Pump Compression
  - Fast RMAN Compression
  - OLTP Table Compression
  - SecureFile Compression and Deduplication
- Hybrid Columnar Compression
  - Warehouse Compression (Query)
  - Archival Compression (Archive)

11.2 Segment Compression Changes

- Compressed Tables



11.2 Table Segment Compression

- Compress for OLTP

```
CREATE TABLE ct1
COMPRESS FOR OLTP AS
SELECT * FROM dba_objects;
```

- Compress for Query

```
CREATE TABLE ct2
COMPRESS FOR QUERY HIGH AS
SELECT * FROM dba_objects;
```

- Compress for Archive

```
CREATE TABLE ct3
COMPRESS FOR ARCHIVE LOW AS
SELECT * FROM dba_objects;
```

Hybrid Columnar Compression

Two New Features in Exadata V2

Warehouse Compression

- 10x average storage savings
- 10x reduction in Scan IO

Optimized for Speed

Archive Compression

- 15x average storage savings
  - Up to 70x on some data
- Some access overhead
- For cold or historical data

Optimized for Space

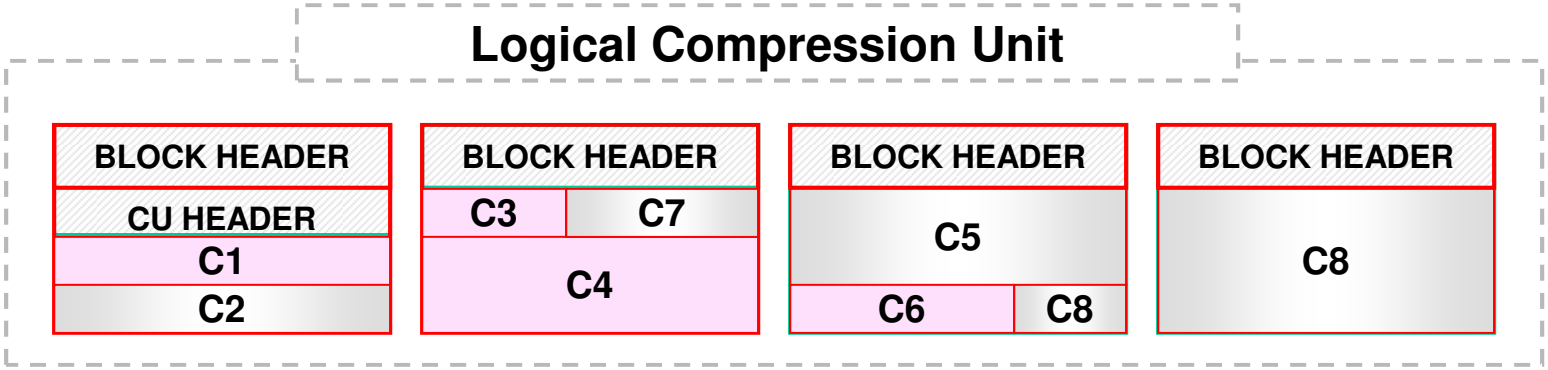
Application transparent

How It Works

- New technology in Oracle Exadata V2
  - New method for organizing data in a database block
  - A second columnar generation technology combining the best of columnar and row organization
  - Columnar Organization
    - Transparently organizes and stores table data by column
    - Improves analytic and aggregate query performance
    - 93% of the performance of full columnar w/o the drawbacks
  - Row Organization
    - The best storage for workloads with updates or trickle feeds
    - A row is self-contained within a ‘compression unit’
    - Minimal I/O to retrieve entire row
    - Efficient index lookups, updates, and deletes

Logical Compression Unit

- Tables are organized into Compression Unit
  - Logical structure spanning multiple database blocks
  - Data organized by column during data load
  - Each column compressed separately
  - Column organization brings similar values close together
  - Typically 32K (4 blocks x 8k block size)



Hybrid Columnar Compression

- DML with Hybrid Columnar Compression
  - Direct Load operations result in Hybrid Columnar Compression
    - Parallel DML, INSERT /*+ APPEND */, Direct Path SQL*LDR
    - Data is transformed into columnar format and compressed during load
  - Conventional INSERT results in OLTP Compression
  - Updated rows automatically migrate to OLTP Compression
- Queries with Hybrid Columnar Compression
  - Only decompress necessary columns to satisfy query
  - Data can remain compressed in the buffer cache
- Optimized algorithm avoids or greatly reduces overhead of decompression during queries

Warehouse Compression

- Built on HCC technology
- Compression algorithm optimized for query performance
- Reduces storage and I/O payload requirements
- Optimal workload characteristics for Warehouse Compression
  - Data loaded with Direct Load operations
  - Scan oriented access
  - Minimal update activity

Optimized for Query Performance

Archival Compression

- Built on HCC technology
- Compression algorithm optimized for maximum storage savings
- Benefits any application with data retention requirements
- Best approach for ILM and data archival
  - Minimum storage footprint
  - No need to move data to tape or less expensive disks
  - Data is always online and always accessible
    - Run queries against historical data (without recovering from tape)
    - Update historical data
  - Supports schema evolution (add/drop columns)

Optimized for Space Utilization

Online Archival Compression

- Optimal workload characteristics for Online Archival Compression
  - Any application (OLTP, Data Warehouse)
  - Cold or Historical Data
  - Data loaded with Direct Load operations
  - Minimal access and update requirements
- 15x average storage savings
  - 1 TB Database compresses to 67 GB
  - Keep historical data online forever
  - Up to 40x savings seen on production customer data

Compression & Partitioning

- OLTP Applications
  - Table Partitioning
  - Heavily accessed data
    - Partitions using OLTP Table Compression
  - Cold or historical data
    - Partitions using Online Archival Compression
- Data Warehouses
  - Table Partitioning
  - Heavily accessed data
    - Partitions using Warehouse Compression
  - Cold or historical data
    - Partitions using Online Archival Compression

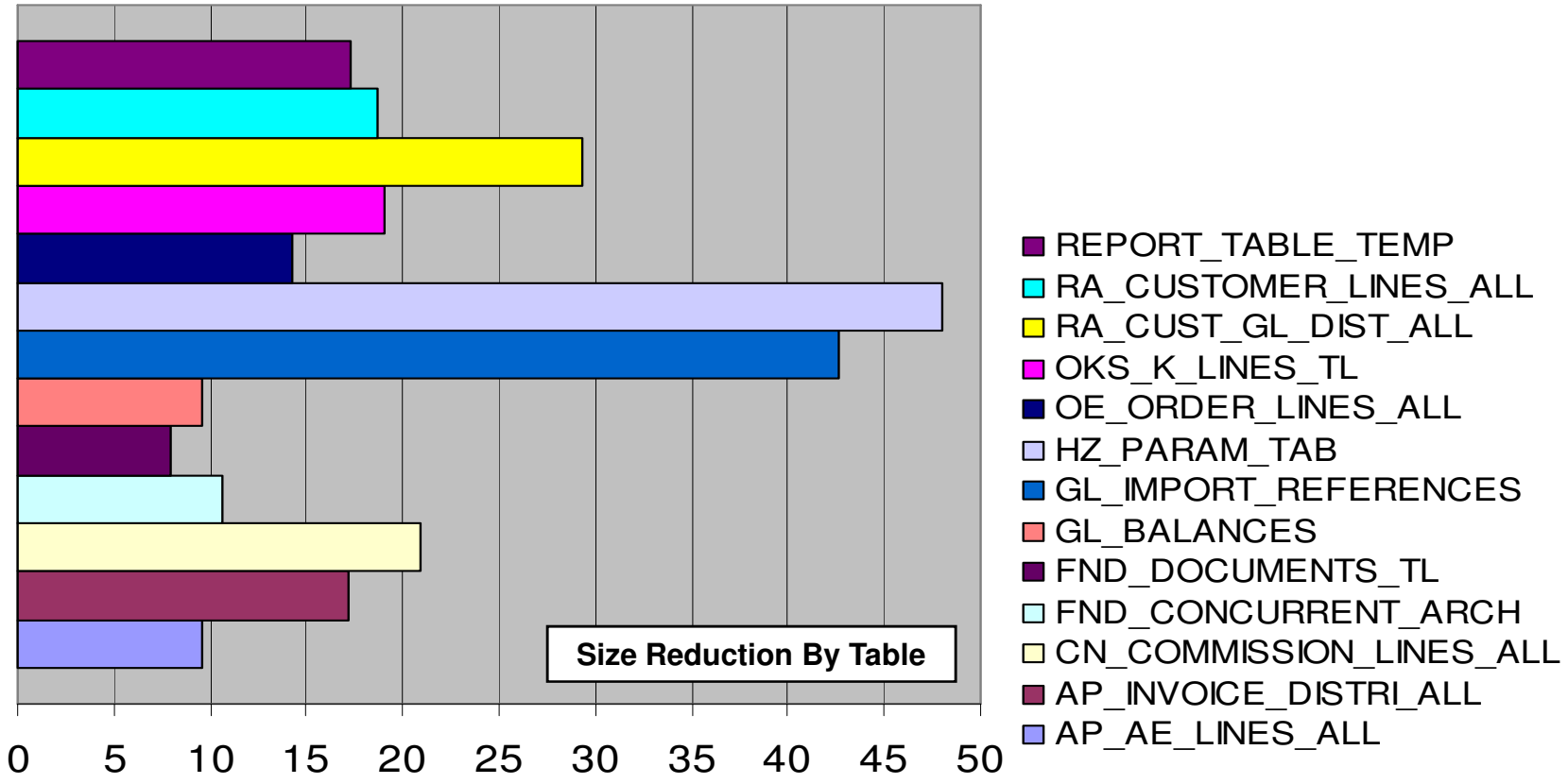
Business as Usual

- Fully supported with…
  - B-Tree, Bitmap Indexes, Text indexes
  - Materialized Views
  - Exadata Server and Cells
  - Partitioning
  - Parallel Query, PDML, PDDL
  - Schema Evolution support, online, metadata-only add/drop columns
  - Data Guard Physical Standby Support
- Will be supported in a future release
  - Logical Standby
  - Streams

Things to Consider …

- When a row is updated
  - It is automatically migrated to OLTP Table Compression
  - Table size will increase moderately
  - All rows in the compression unit are locked
- When tables are queried
  - Table scans are faster due to less I/O
  - Index lookups are usually slower
    - Need to decompress the compression unit to read entire row

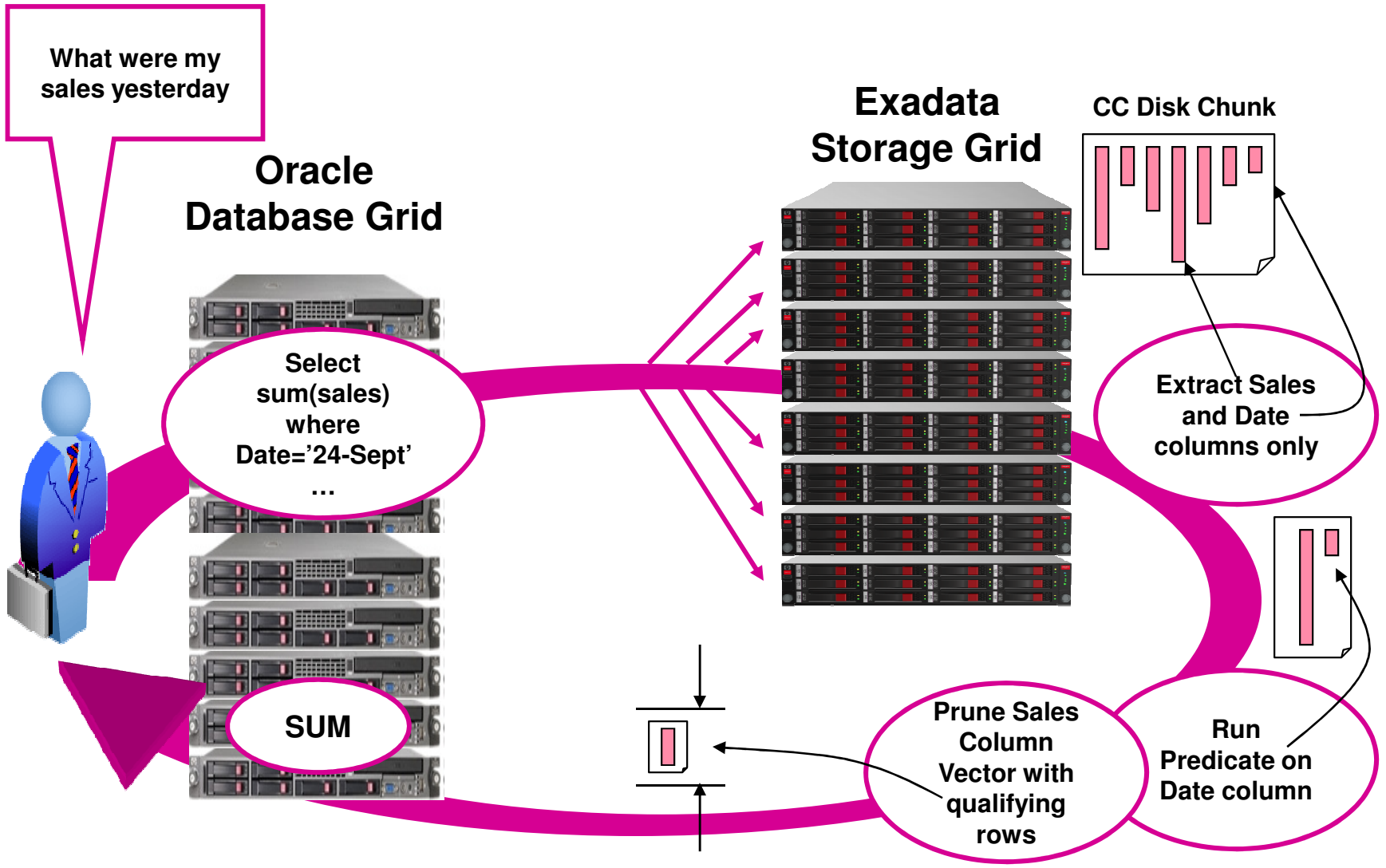
Oracle Production E-Business Suite Data



Archive Compression 8x to 48x - Reduction Average 20x

Big Banks achieved 30X average, Major Telcos 9X average

Smart Scans of Columnar Compressed Tables



New Compression Advisor

- DBMS_COMPRESSION built-in package
  - GET_COMPRESSION_RATIO
    Returns the possible compression ratio for an uncompressed table or materialized view and estimates achievable compression
  - GET_COMPRESSION_TYPE
    Inspects data and reports what compression type is in use by row
- Enterprise Manager Segment Advisor
  - Estimates OLTP Table Compression automatically
  - Advises tables that will benefit from OLTP Compression

GET_COMPRESSION_RATIO

```
CREATE TABLE comp_test1 AS
SELECT * FROM dba_objects;

set serveroutput on


DECLARE
  blkcnt_comp PLS_INTEGER;
  blkcnt_uncm PLS_INTEGER;
  row_comp    PLS_INTEGER;
  row_uncm    PLS_INTEGER;
  comp_ratio  PLS_INTEGER;
  comp_type   VARCHAR2(30);
BEGIN
  dbms_compression.get_compression_ratio('UWDATA', 'UWCLASS', 'COMP_TEST1', NULL,
  dbms_compression.comp_for_oltp, blkcnt_cmp, blkcnt_uncmp, row_comp, row_uncm,
  comp_ratio, comp_type);

  dbms_output.put_line('Block Count Compressed:   ' || TO_CHAR(blkcnt_comp));
  dbms_output.put_line('Block Count UnCompressed: ' || TO_CHAR(blkcnt_uncm));
  dbms_output.put_line('Row Count Compressed:     ' || TO_CHAR(row_comp));
  dbms_output.put_line('Row Count UnCompressed:   ' || TO_CHAR(row_uncm));
  dbms_output.put_line('Block Count Compressed:   ' || TO_CHAR(comp_ratio));
  dbms_output.put_line('Compression Type:         ' || comp_type;
END;
/
```

tablespace owner table name partition

GET_COMPRESSION_TYPE

```
CREATE TABLE comp_test2
COMPRESS FOR OLTP AS
SELECT * FROM dba_objects;

set serveroutput on

DECLARE
 rid ROWID;
 n   NUMBER;
BEGIN
  SELECT MAX(rowid)
  INTO rid
  FROM comp_test2;
                                                    owner     table name     rowid
  n := dbms_compression.get_compression_type(USER, 'COMP_TEST2', rid);
  dbms_output.put_line(n);
END;
/
```

Summary

- If you can move to Exadata V2 … you will better serve your customers
- If you can not then don't choose a single technology … leverage them in combination
  - ASM
  - Real Application Clusters
  - Advanced Compression
  - Partitioning

We did not come here to fear the future

Questions

ERROR at line 1:

ORA-00028: your session has been killed

All demos at morganslibrary.org

- **Library**
- **How Can I?**

Thank you