Segment Compression with Oracle Database 11g for DBAs and Developers

presentation for:

Oracle Users Group Finland

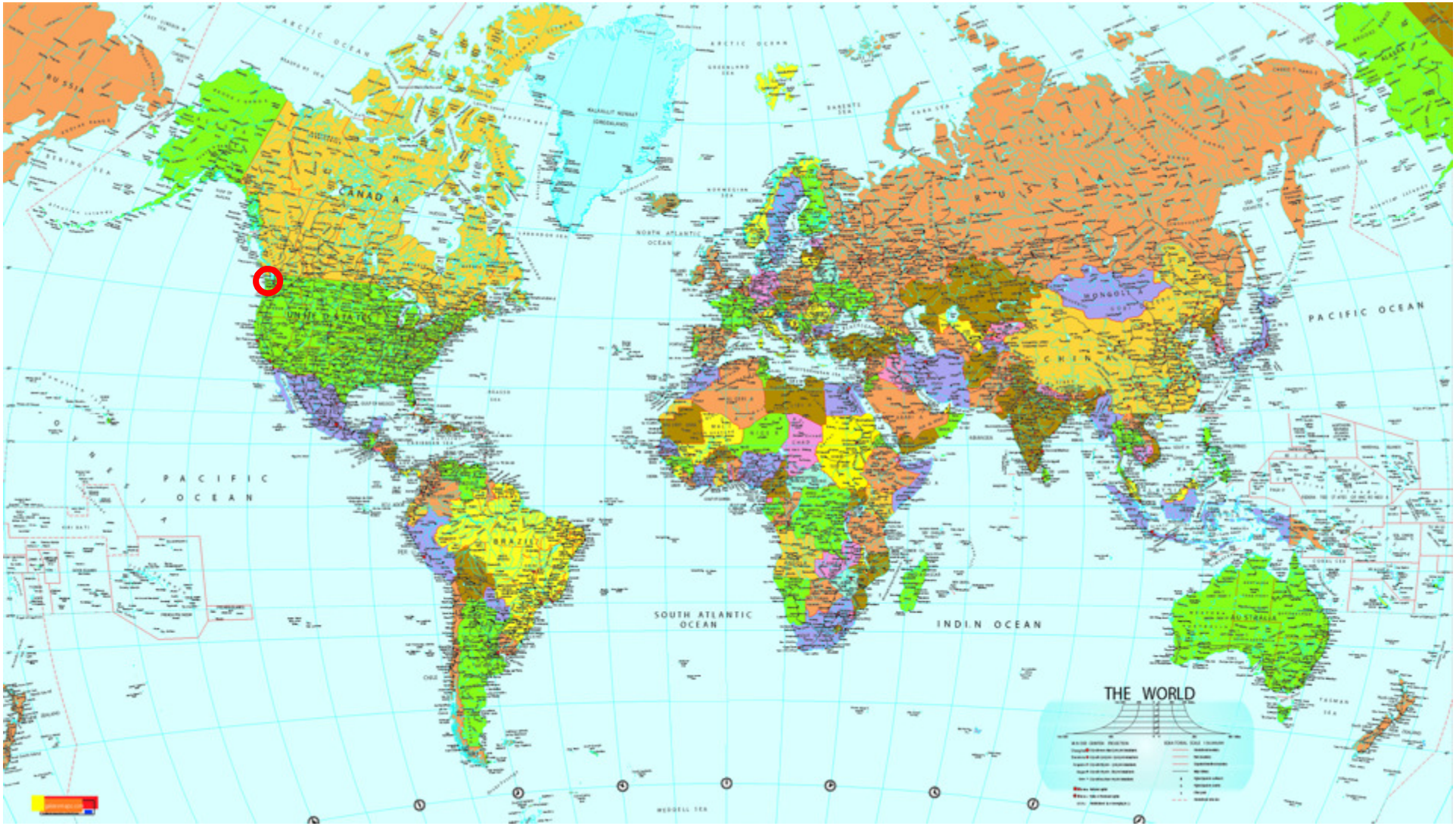
November 2009 Conference

Introduction

- Daniel Morgan – damorgan@u.washington.edu
- Oracle Ace Director
- University of Washington, retired
- The Morgan of Morgan's Library on the web
  - www.morganslibrary.org/library.html
- Member: Western Washington Oracle Users Group
- Member: UK Oracle Users Group
- Member: Oracle Applications Users Group
- Frequent speaker

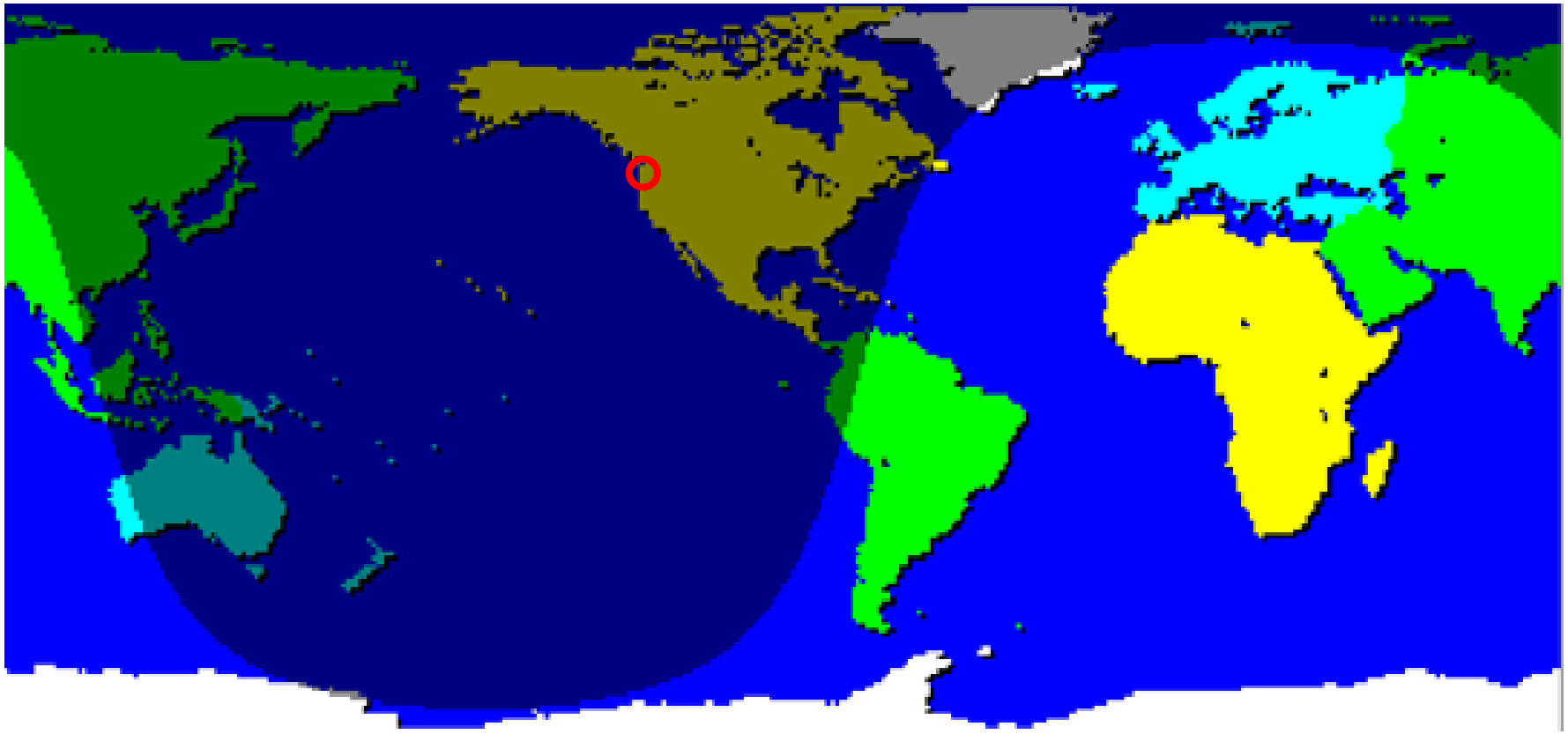
- Oracle since version 6
- 11g beta test site

My Home

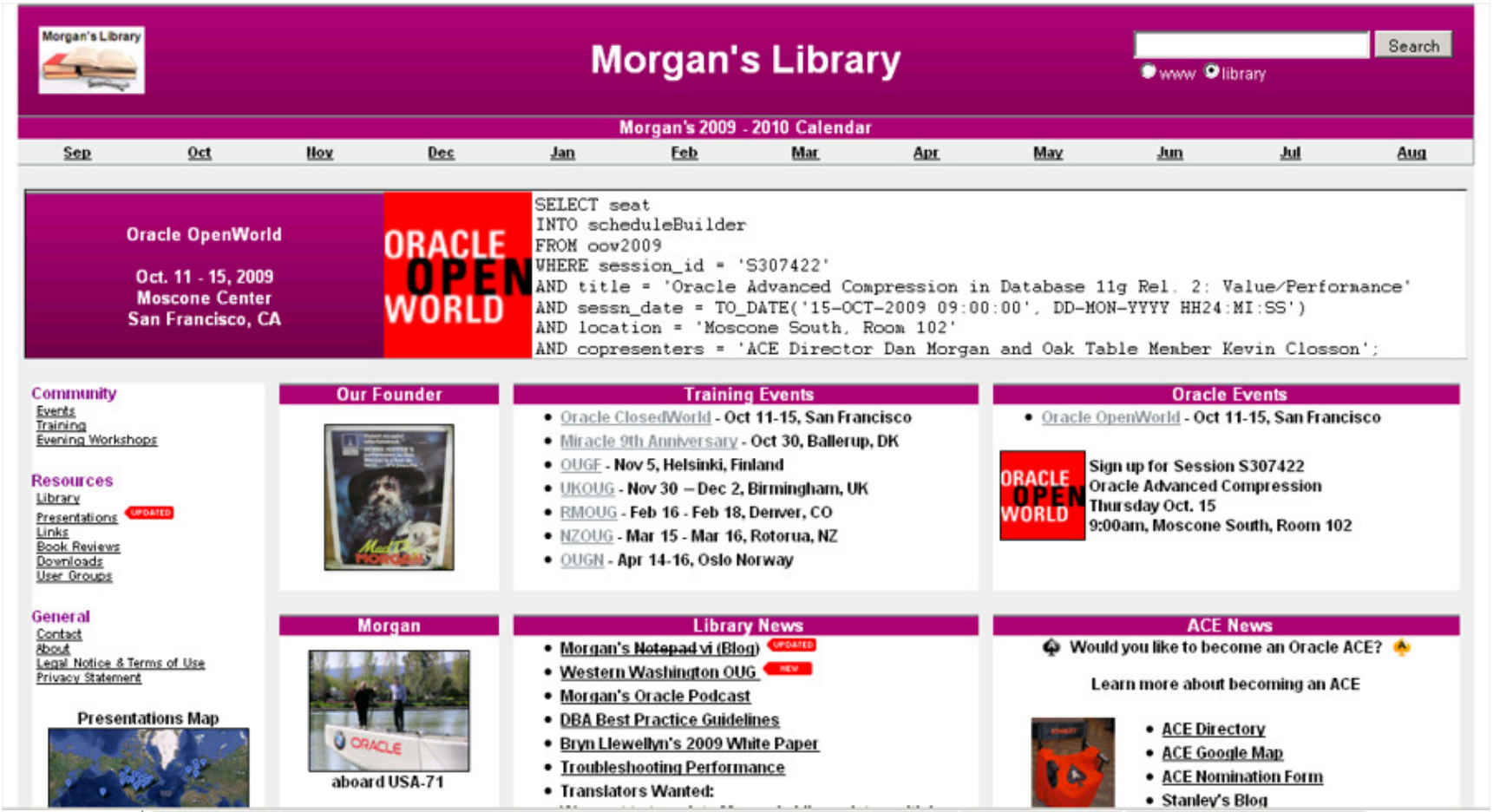

ARCTIC OCEAN
RUSSIA
CANADA
NORTH ATLANTIC OCEAN
PACIFIC OCEAN
PACIFIC OCEAN
BRAZIL
SOUTH ATLANTIC OCEAN
INDIAN OCEAN
THE WORLD

My Home

My Clock



Morgan's Library: www.morganslibrary.org



Hybrid Columnar Compression	11gR2	10-Oct-2009
IF Statements	11gR2	01-Sep-2009
Import	11gR2	04-Aug-2009
Indexes	11gR2	28-Sep-2009
Index Organized Tables (IOT)	11gR2	15-Sep-2009

How Can I?

A Guide To Meeting Challenges in Oracle Database 11g		
Guarantee case insensitive uniqueness	Click Here	28-Oct-2009
Read a list of operating system files into a PL/SQL array (without using C or Java)	Click Here	28-Oct-2009
Identify the columns and data types in a weakly typed ref cursor	Click Here	29-Oct-2009
Write a CLOB to a file	Click Here	01-Nov-2009
Track the parts of my application that are in use	Click Here	02-Nov-2009
Verify that a string passed into my function or procedure does not contain malicious code (SQL Injection)	Click Here	Soon
Create a single check constraint across multiple columns guaranteeing only one is used	Click Here	Soon
Work effectively with the deprecated LONG data type?	Click Here	Soon

This is the start of a new Morgan's Library feature.
For the next couple of years, while we learn more about 11gR2,
we will be focusing our efforts here.
Please join us while we grow this section
and learn how to index it so you can find what you need.

Can we code it?

Yes we can!

Since I was here last year

Date	Presenter	Topic
OpenWorld 10/09	Daniel Morgan, Morgan's Library Kevin Closson, Oracle	• What's New In Eleven Dot Two • Hybrid Columnar Compress and Exadata • Ken's Exadata V2 Slides
NWOUG 09/09	Daniel Morgan Morgan's Library	• Oracle ACE Program • What's New in 11 dot 2?
NoCOUG 08/09	Daniel Morgan Morgan's Library	• The Latest Oracle 11g Gems
ODTUG 07/09	Daniel Morgan Morgan's Library	• Oracle Audit Vault: Trust but Verify for Enterprise Databases
CLOUG 04/09	Daniel Morgan University of Washington	• Oracle Gems for DBAs and Developers • Oracle ACE Program
OUGN 03/09	Daniel Morgan University of Washington	• Oracle Gems for DBAs and Developers
UKOUG 12/08	Daniel Morgan University of Washington	• Breaking Oracle
PSOUG 11/08	Kevin Closson, Oracle Daniel Morgan, UW Mogens Norgaard, Miracle A/S	• Exadata • Disaster Avoidance with Oracle 11g


BMW
ORACLE

Agenda

- Why the 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

We all have our favorite customers: This is mine

Store More Data

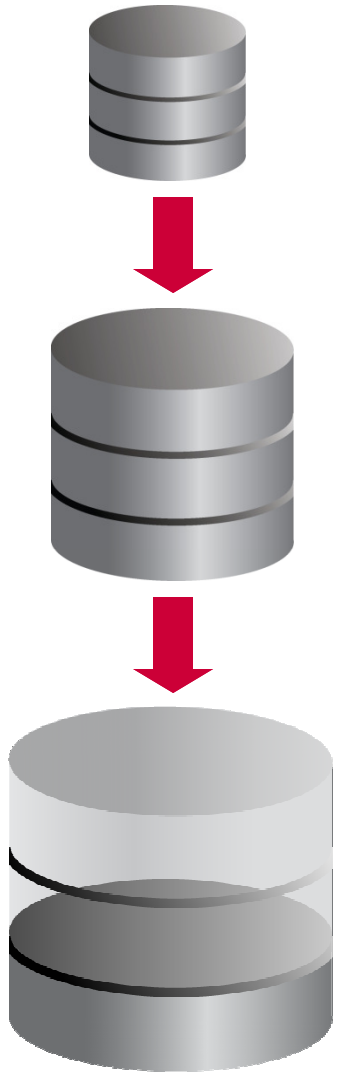
Maintain Performance

Honor the same Service Level Agreement

GB, TB, EB What's the big deal?

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
  - Use 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
- May 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 the table like 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 btyes

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

38 btyes

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);

  r := utl_raw.cast_to_raw('ABC');

  utl_compress.lz_compress_add(handle, b, r);

  utl_compress.lz_compress_close(handle, b);
END;
/
```

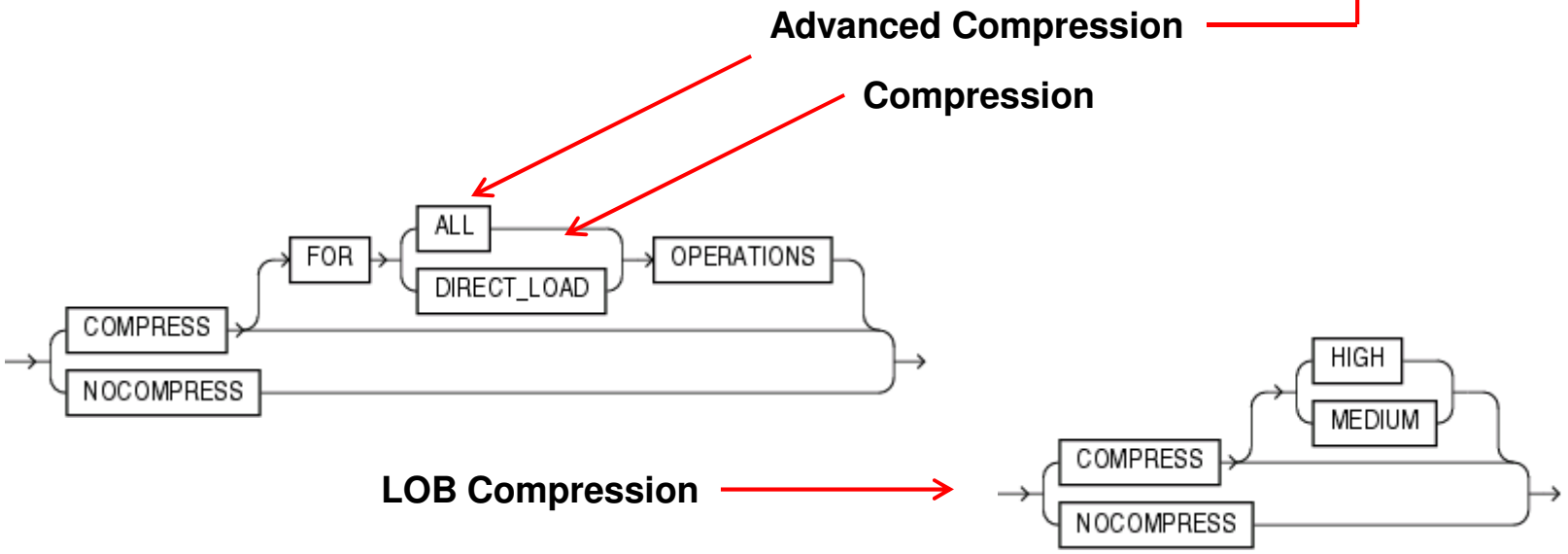
- No significant changes in 10gR2

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 sec_tab_kd (
rid  NUMBER(5),
bcol BLOB)
LOB (bcol)
STORE AS SECUREFILE bcol (
  TABLESPACE securefiletbs
  RETENTION MIN 3600
  KEEP_DUPLICATES
  NOCOMPRESS
  ECRYPT
  CACHE READS)
TABLESPACE uwdata;
```

```
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;
```

- What Oracle now calls DBFS (Database File System)

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)

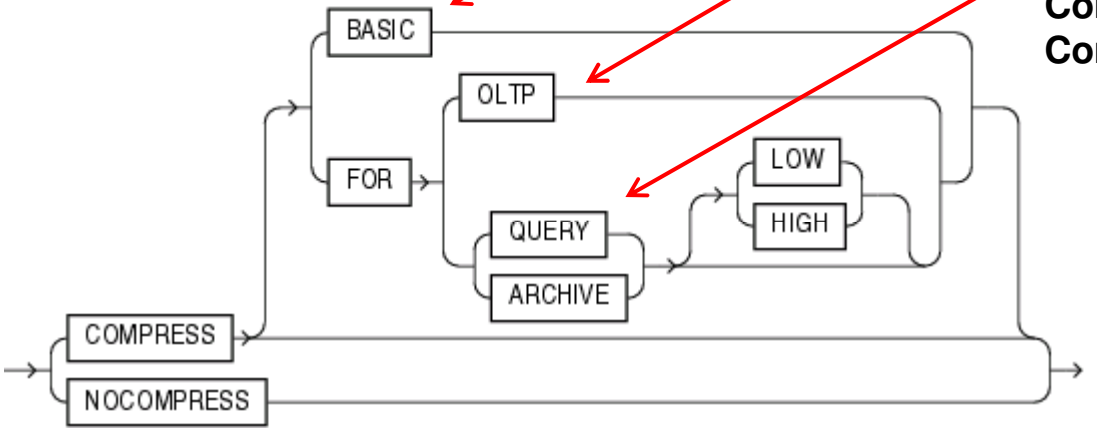
11.2 Segment Compression Changes

- Compressed Tables

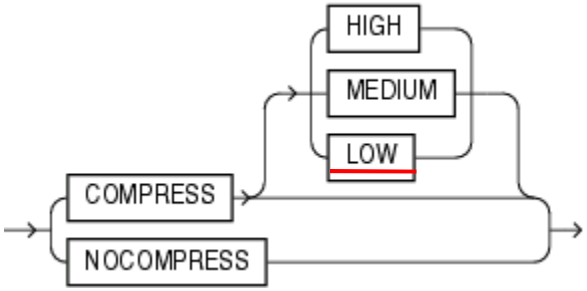
Compression
BASIC replaces DIRECT_LOAD

Advanced Compression
OLTP replaces ALL OPERATIONS

Hybrid Columnar Compression
Compress For Query
Compress For Archive



LOB Compression



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

Hybrid Columnar Compression and Performance

- Is compression a cost/space/power/cooling savings feature …
- or a performance feature?
  - Yes

Hybrid Columnar Compression and Performance

- Oracle Database doesn't decompress data for the sheer fun of it. Decompression is a part of query processing
- Query processing starts with I/O and continues on to filtration and column projection
- Compression ratio is also the physical I/O savings ratio
- Decompression is inserted between I/O and SQL processing. Strictly speaking, it is a tax
- Spending CPU cycles to uncompress data takes CPU from query processing
- Decompression needs to produce a lot of output to pay for itself

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

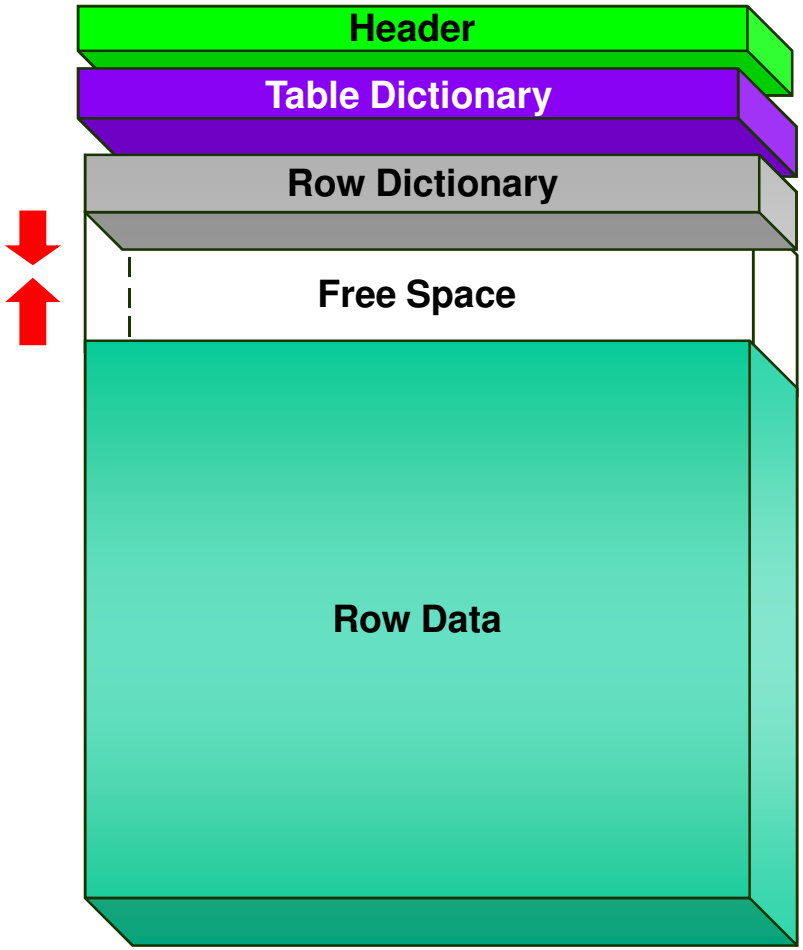
Optimized for Space

Completely 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
  - Compression changes I/O to CPU balance considerations

Database Block Anatomy



General block information

Table info in clusters

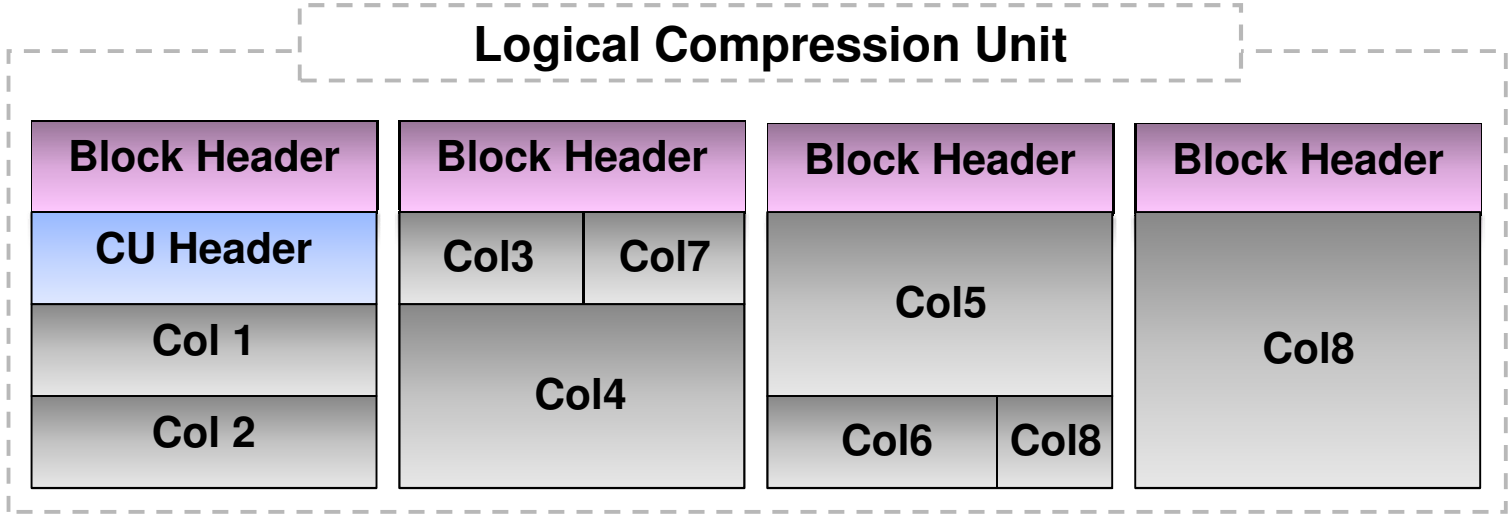
Row info in block (2 bytes/row)

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

Segment data (table or index)

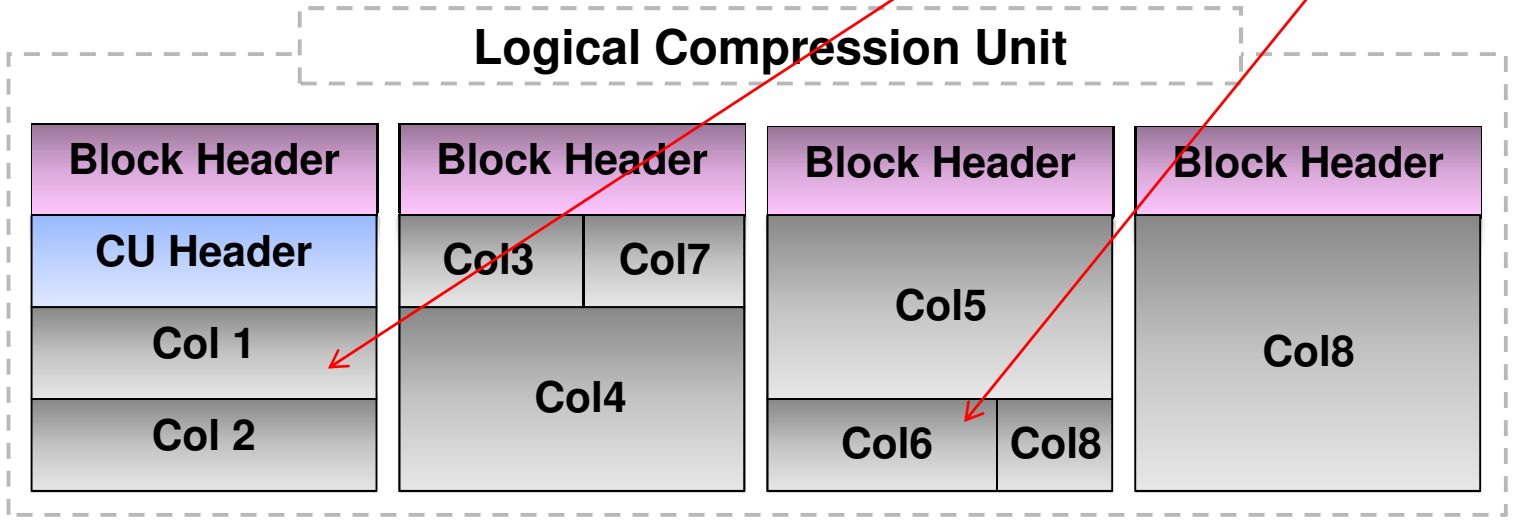
Logical Compression Unit

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



Logical Compression Unit

```
CREATE TABLE demo (
person_id      NUMBER(10),
first_name     VARCHAR2(20),
mid_initial    VARCHAR2(4),
last_name      VARCHAR2(35),
date_of_birth  DATE,
hire_date      DATE,
status         VARCHAR2(5),
comments       VARCHAR2(500));
```



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 applications 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

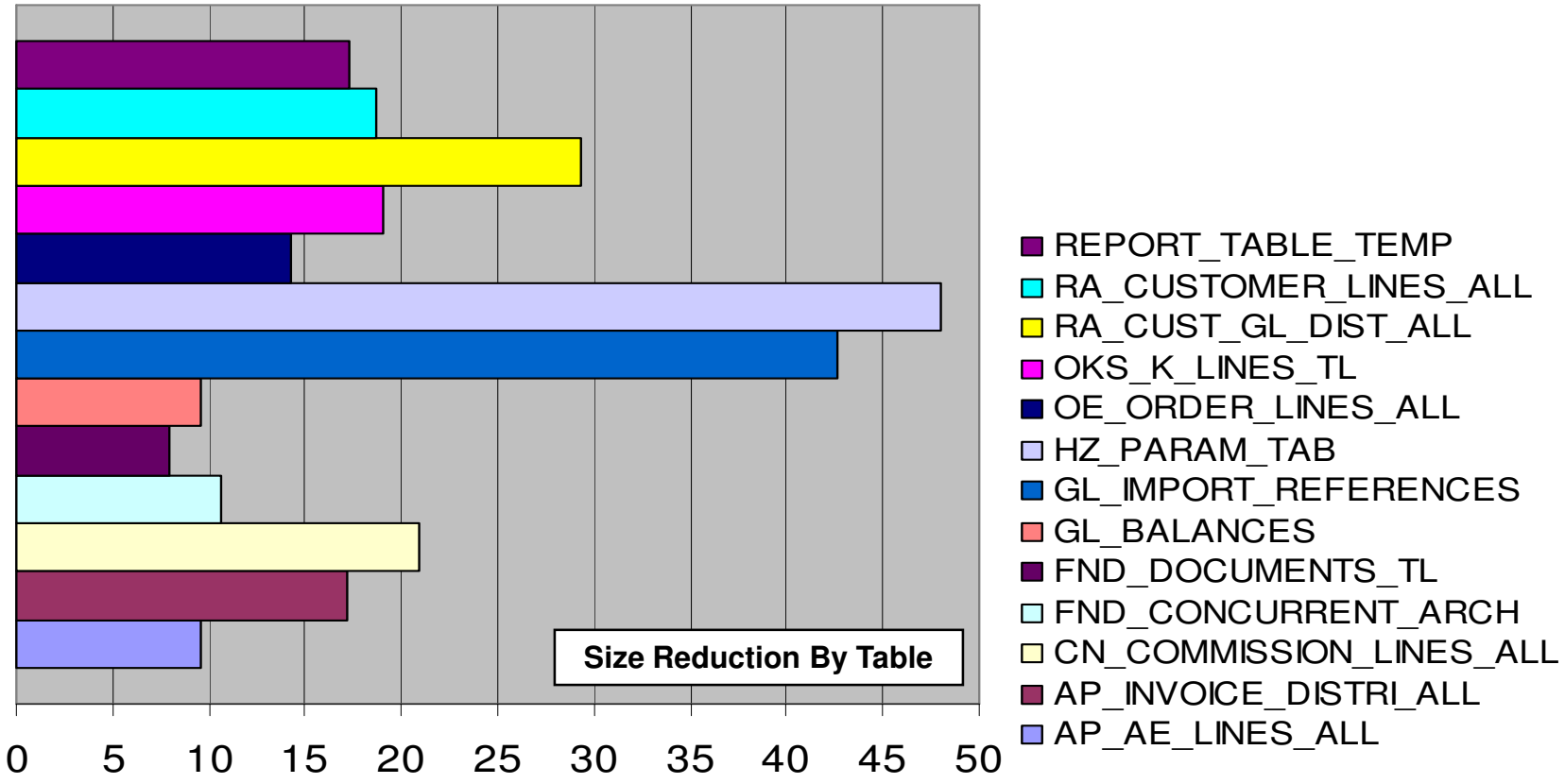
Business as Usual

- 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

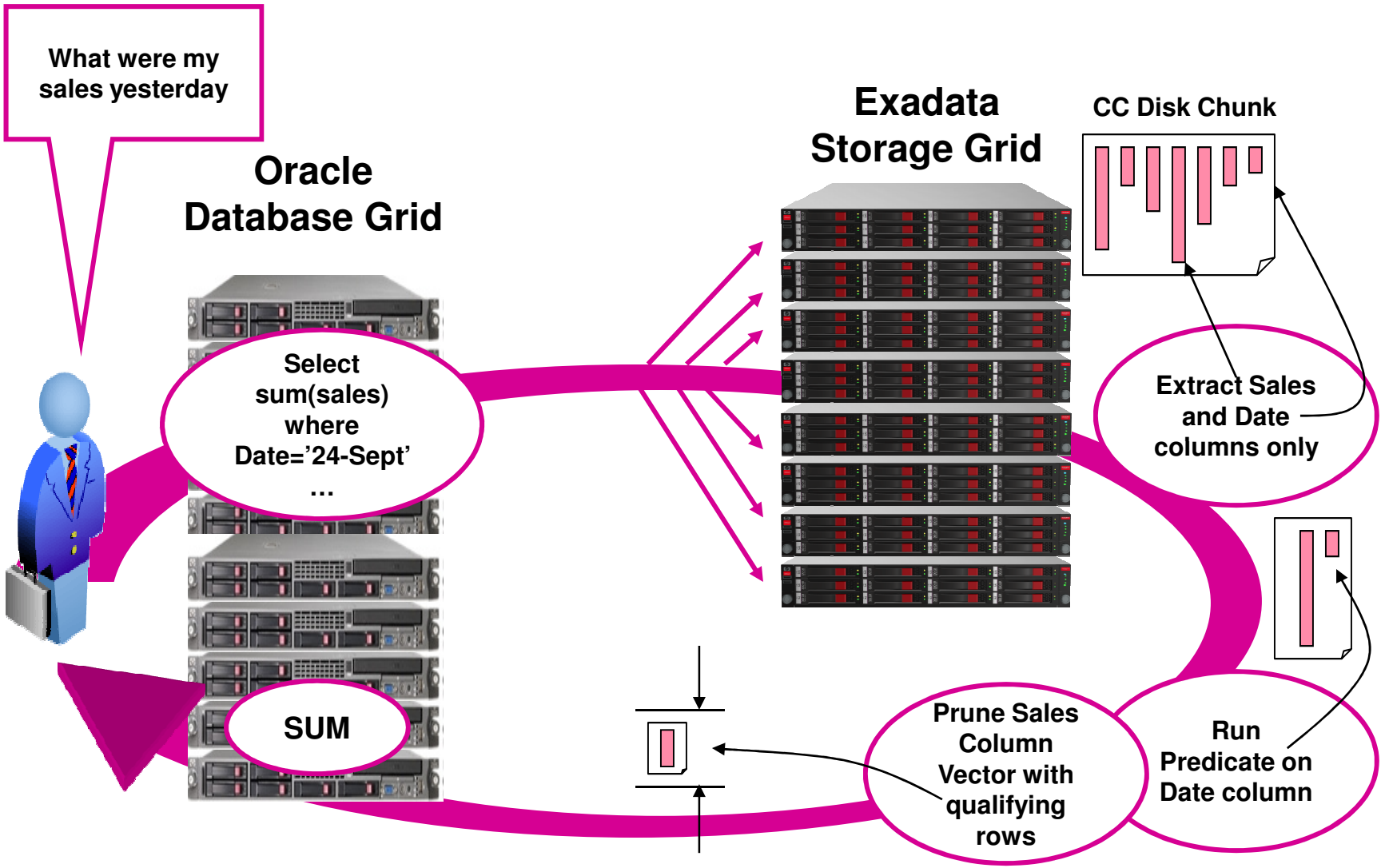
HCC's Performance Characteristics

	GB/s Physical	4 GFC HBAs	EHCC Reinflated Data Flow (GB/s)	Required Cores for "Primary Processing
Generic System *	4	10	40	6.8
Database Machine	21 (disk)	N/A	210	35.7
	50 (flash)	N/A	500	85

- Division of work: Sun Oracle Database Machine has 112 processor cores for decompression and filtration
- A hypothetical equivalent non-Exadata system would need on the order of 10 cluster nodes just for decompression and filtration if provisioned 21 GB/s I/O

* Hypothetical as HCC is an Exadata Feature

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;

  n := dbms_compression.get_compression_type(USER, 'COMP_TEST2', rid);
  dbms_output.put_line(n);
END;
/
```

owner table name rowid

Summary

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

We did not come here to fear the future

Questions

damorgan@u.washington.edu

demo code: www.morganslibrary.org

Thank you.