Challenges in a World of Large Databases and Business Intelligence

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

Daniel A. Morgan

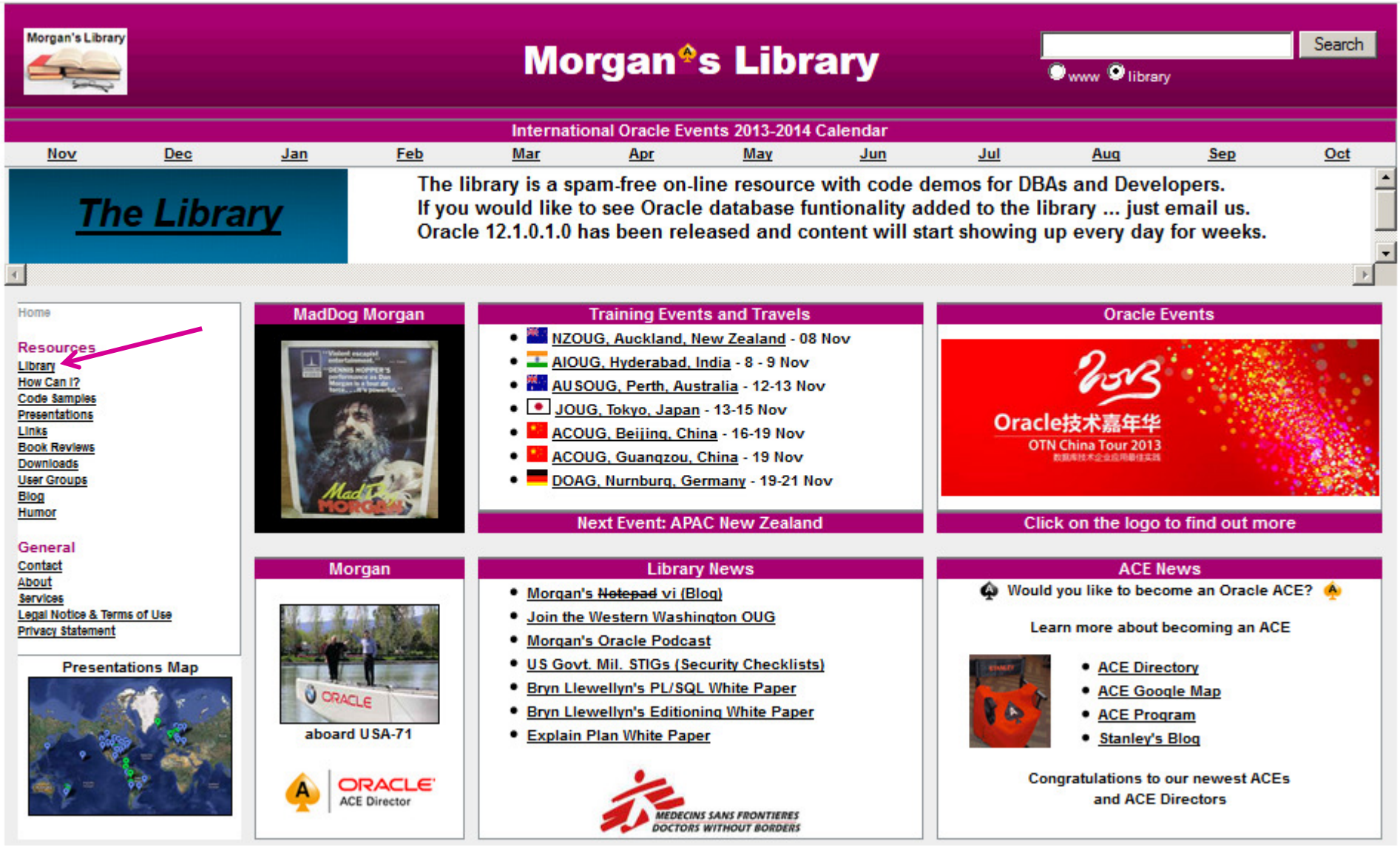
- Oracle ACE Director
- Consultant to Harvard University
- University of Washington Oracle Instructor, ret.
- The Morgan of Morgan's Library on the web
- Board Member: Western Washington OUG, ret.
- Executive Board: Vancouver/Victoria OUGs
  - 10g, 11g, 12c Beta Tester
  - Scheduled Events November
    - Thailand Oracle Users Group
    - New Zealand Oracle Users Group

Official Beta Site ORACLE 12c DATABASE

ORACLE RACSIG

International zSeries Oracle SIG

Morgan's Library: www.morganslibrary.org



Take Notes: Ask Questions

The State of Academia and Industry

- There is a tension between academia and industry
  - You will not get a job where you write a bloom filter
  - You will not get a job where you write a compiler
  - You will not be hired to invent a hashing algorithm
  - It is the mental skills you learn in academia are critical
  - When interviewing one of the first tests I use is to determine whether the candidate can translate a business problem into Boolean logic: Something you can not learn on the job

The State of Industry

- In the 45 years I've been in the industry I have seen some things that are consistent
  - The amount of cpu available increases
  - The amount of memory increases
  - Networks get faster, increase bandwidth, and decrease latency
  - Storage improves in capacity and speed
  - Complexity increases
  - The level of technical expertise goes down
- And some things are constantly in flux and the same patterns repeat every 5-7 years
  - Consolidation vs Deconsolidation
  - Some new programming language is declared better
  - Some new technology is declared newer and better
  - Some new database type is marketed as better

XML versus JSON

```
'<PONumber>1600</PONumber>
 <Special Instructions></Special Instructions>
 <AllowPartialShipment>true</AllowPartialShipment>
 <LineItems><ItemNumber>1</ItemNumber>
              <Part>
                <Description>One Magic Christmas</Description>
                <UnitPrice>19.95</UnitPrice>
                <UPCCode>13131092899</UPCCode>
                <Quantity>9</Quantity>
              </Part>
            <ItemNumber>2</ItemNumber>
              <Part>
                <Description>Cinderella</Description>
                <UnitPrice>19.95</UnitPrice>
                <UPCCode>85391628927</UCCode>
                <Quantity>5</Quantity>
              </Part>
 </LineItmes>'
```

```
'{"PONumber" : 1600,
"Special Instructions" : null,
"AllowPartialShipment" : true,
"LineItems" : [{"ItemNumber" : 1,
                "Part" : {"Description" : "One Magic Christmas",
                          "UnitPrice" : 19.95,
                          "UPCCode" : 13131092899},
                          "Quantity" : 9.0},
               {"ItemNumber" : 2,
                "Part" : {"Description" : "Cinderella",
                          "UnitPrice" : 19.95,
                          "UPCCode" : 85391628927},
                          "Quantity" : 5.0}]}');
```

The State of Industry

- Databases when I started in the business were large when they were hundreds of MB ... the largest database I have worked on in the last couple of years was a DSS system stored on 1.2PB of disk
- Most of what we did in the past was IMS (IBM **I**nformation **M**anagement **S**ystem) which is a hierarchical database
- In 1970 Edwin E.F. Codd wrote his seminal paper on relational algebra: "A Relational Model of Data for Large Shared Data Banks"
- Two different groups used this paper as the basis for creating today's relational databases
  - Michael Stonebreaker and Lawrence Row (UC Berkeley) Project Ingress
  - Ed Oates, Bruce Scott, Bob Miner, and Larry Ellison (IBM / SDL)

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network

A Brief History of Relational Databases

Ingres

Originated as a research project at UC Berkeley: 1970

Several companies used the Ingres source code to produce products. The most successful was a company named Relational Technology, Inc. (RTI) founded in 1980 by MichaelIStonebraker and another Berkeley professor, Lawrence A. Rowe. RTI was renamed Ingres Corporation in the late 1980's.

Informix

Followed the 1995 purchase of Illustra, Informix concentrated on object-relational database (O-R) technology. Illustra was written by ex-Postgres team members and led by database pioneer Michael Stonebraker. Illustra included DataBlades that allowed new data types and features to be included in the basic server as options such as time series, spatial and multimedia data. Informix integrated Illustra's O-R mapping and DataBlades into the 7.x OnLine product, resulting in Informix Universal Server (IUS), or more generally, Version 9.

Sybase

Robert Epstein, the chief programmer on the Ingres project while he was at Berkeley, formed Britton-Lee along with other students from the Ingres Project, Paula Hawthorne and Mike Ubell; they were joined later by Eric Allman. Later, they founded Sybase.

SQL Server

Sybase's product line was licensed to Microsoft in 1992, who re-branded it as Microsoft SQL Server. The relationship soured in the late 1990s, and the product lines were separated. Since then SQL Server has matured and while differentiating itself from Sybase still contains the original Project Ingres transaction architecture.

The State of Industry

- Since then we have seen many attempts to do better in response to the fact that the volume of persistent data continues to increase at a rapid rate
  - Object Databases
  - XML Databases
  - Columnar Databases
  - NoSQL / Schema-less Databases

Where Is All This Data Coming From

- Transactional Data
  - Enterprise Resource Planning (ERP)
  - Finance
  - Human Relations (HR)
- Non-Transactional Data
  - Web Page Clicks
  - Catalogs
  - Content Management (comments)
- Regulatory and Legal Requirements

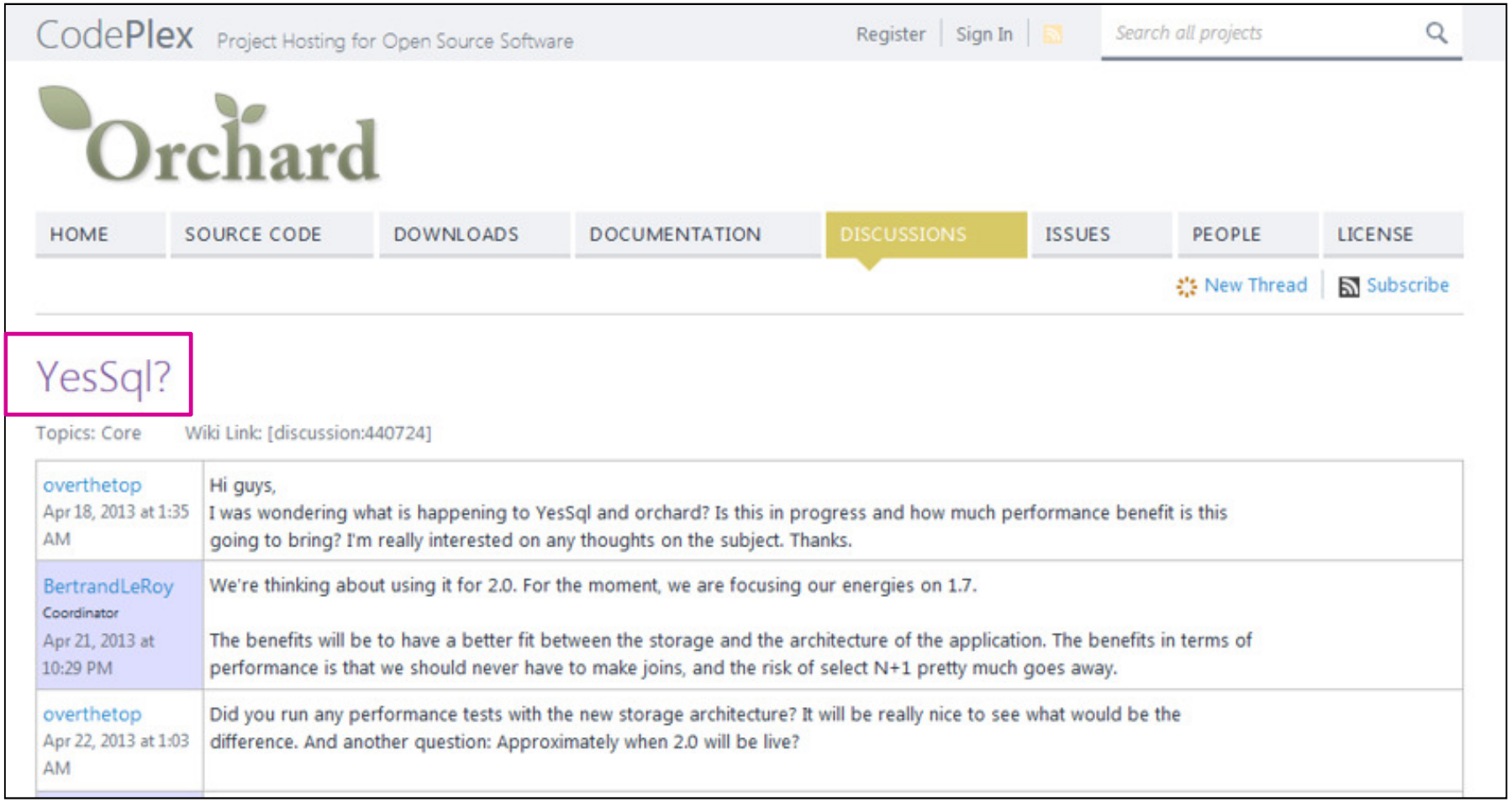
How Important Is Accuracy?

- Transactional Data
  - Transactional systems required ACID (Atomicity, Consistency, Isolation, and Durability)
  - If you query a your bank's database for the amount of money in your bank account does it matter if the result is off by $20?
- Non-Transactional Data
  - If you query google for how many hits there are for the word "database" the answer is 122,000,000: Does it matter whether it is correct?
  - If you query Amazon for the number of books on "Computer Science" the answer is 108,330: Does it matter if the result is off by 20 books?

The State of Industry

- Object Databases
  - Gemstone, GBase, VBase, ITASCA, Jasmine
- XML Databases
  - eXtremeDB, Matisse, Objectivity/DB, Orient, Versant
- Columnar
  - Accumulo, Greenplum, MonetDB, Redshift , Vertica
- NoSQL Databases
  - Column: Cassandra, HBase
  - Document: MarkLogic, MongoDB
  - Graph: Allegro, OrientDB
  - Key-Value: Dynamo, FoundationDB
- And NoSQL is already losing its glamour

The State of Industry



CodePlex Project Hosting for Open Source Software

Register | Sign In | Search all projects

Orchard

HOME | SOURCE CODE | DOWNLOADS | DOCUMENTATION | DISCUSSIONS | ISSUES | PEOPLE | LICENSE

New Thread | Subscribe

YesSql?

Topics: Core Wiki Link: [discussion:440724]

overthetop Apr 18, 2013 at 1:35 AM	Hi guys, I was wondering what is happening to YesSql and orchard? Is this in progress and how much performance benefit is this going to bring? I'm really interested on any thoughts on the subject. Thanks.
BertrandLeRoy Coordinator Apr 21, 2013 at 10:29 PM	We're thinking about using it for 2.0. For the moment, we are focusing our energies on 1.7. The benefits will be to have a better fit between the storage and the architecture of the application. The benefits in terms of performance is that we should never have to make joins, and the risk of select N+1 pretty much goes away.
overthetop Apr 22, 2013 at 1:03 AM	Did you run any performance tests with the new storage architecture? It will be really nice to see what would be the difference. And another question: Approximately when 2.0 will be live?

Large Database

- What does Large mean?
  - Storage footprint?
  - Number of bytes read to produce an answer?
  - Number of transactions per second?
  - Number of connected users?
  - Marketing hyperbole?

- I invented the term ILDB for Insanely Large Database which I defined as a database too large to backup and restore

- The most critical thing in working with large databases is minimizing the amount of work performed

Transaction Frequency

SQL ordered by Executions

- Total Executions: 29,717,627
- Captured SQL account for 77.4% of Total

Executions	Rows Processed	Rows per Exec	CPU per Exec (s)	Elap per Exec (s)	SQL Id	SQL Module	SQL Text
10,128,178	2,506,529	0.25	0.00	0.00	932srzg1krc33	ASN_07B_DIP(004110016)	SELECT NE_TIMEZONE FROM CMPM.E...
7,576,759	7,579,197	1.00	0.00	0.00	1h698sb62un99	asci_56_RANAPProtocolStats(01611000E)	SELECT DISTINCT NE_TIMEZONE FR...
3,914,621	3,848,268	0.98	0.00	0.00	5tbzddgguu8cc	asci_56_RANAPProtocolStats(01611000E)	SELECT SYS_VERSION FROM CMPM.T...
311,645	311,604	1.00	0.00	0.00	7qtztzv329wq0		select c.name, u.name from co...
301,428	301,325	1.00	0.00	0.00	36s446f9cnwhw		SELECT C.NAME FROM COL$ C WHER...
200,692	200,669	1.00	0.00	0.00	4vs91dcv7u1p6	OMS	insert into sys.aud$(sessioni...
65,044	65,035	1.00	0.00	0.00	fz9xwpt2cvt0k		SELECT par_type, param_clob, ...
64,949	3,945,482	60.75	0.00	0.00	f5ra7dru5fk5n	XML_P7R_RNC_RCS(003110008)	SELECT NAME, PATH, READ, WR...
64,801	64,807	1.00	0.00	0.00	fhzj09a7fnrnb	XML_V7I_IN_LP_DC(00811000V)	SELECT DBTIMEZONE, LENGTH(DBT...
64,632	64,542	1.00	0.00	0.00	15jnrrb6016nd	XML_V7I_IN_LP_DC(00811000V)	SELECT SESSIONTIMEZONE, LENGT...

```
SELECT /*+ RESULT_CACHE */ srvr_id
FROM (
  SELECT srvr_id, SUM(cnt) SUMCNT
  FROM (
    SELECT DISTINCT srvr_id, 1 AS CNT
    FROM servers
    UNION ALL
    SELECT DISTINCT srvr_id, 1
    FROM serv_inst)
  GROUP BY srvr_id)
WHERE sumcnt = 2;
```

Transaction Size (1:3)

```
SELECT DISTINCT E1_2.OBJECT_ID
   FROM PMCM.ELEMENT_DETAIL E1_1, PMCM.ELEMENT_DETAIL E1_2, PMCM.MARK_NETW_HIERARCHY H1,
        PMCM.ELEMENT_DETAIL E2_1, PMCM.ELEMENT_DETAIL E2_2, PMCM.MARK_NETW_HIERARCHY H2
   WHERE E1_1.OBJECT_ID = H1.PARENT_ID
     AND E1_2.OBJECT_ID = H1.OBJECT_ID
     AND E2_1.OBJECT_ID = H2.PARENT_ID
     AND E2_2.OBJECT_ID = H2.OBJECT_ID
     AND E1_1.CURRENT_IND = 'Y' AND E2_1.CURRENT_IND = 'Y'
     AND E2_1.CURRENT_IND = 'Y' AND E2_2.CURRENT_IND = 'Y'
     AND H1.CURRENT_IND = 'Y' AND H2.CURRENT_IND = 'Y'
     AND H1.HIERARCHY_TYPE = 'NETWORK' AND H2.HIERARCHY_TYPE = 'NETWORK'
     AND H1.PARENT_TYPE IN ('BSC','RNC')  AND H2.PARENT_TYPE IN ('BSC','RNC')
     AND E2_2.ELEMENT_TYPE = 'CELL' AND E1_2.ELEMENT_TYPE = 'CELL'
     AND H1.PARENT_TYPE IN ('BSC','RNC')
     AND E1_1.ELEMENT_NAME = E2_1.ELEMENT_NAME
     AND E1_1.ELEMENT_ID = E2_1.ELEMENT_ID
     AND E1_2.ELEMENT_NAME = E2_2.ELEMENT_NAME
     AND E1_2.ELEMENT_ID = E2_2.ELEMENT_ID
     AND E1_2.USEID LIKE '*%' AND E2_2.USEID NOT LIKE '*%';

| Id  | Operation                    | Name                          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
|   0 | SELECT STATEMENT             |                               |     1 |    78 |       |    74M (40)| 50:54:42 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION   |                               |       |       |       |             |          |       |       |
|   2 |   LOAD AS SELECT             |                               |       |       |       |             |          |       |       |
|   3 |    PARTITION RANGE ALL       |                               |    22M|  1111M|       | 38153   (11)| 00:01:34 |     1 |    29 |
|*  4 |     TABLE ACCESS FULL        | ELEMENT_DETAIL                |    22M|  1111M|       | 38153   (11)| 00:01:34 |       |       |
|   5 |   LOAD AS SELECT             |                               |       |       |       |             |          |       |       |
|   6 |    PARTITION HASH ALL        |                               |   337K|  9231K|       |  3514   (15)| 00:00:09 |     1 |    16 |
|*  7 |     TABLE ACCESS FULL        | MARK_NETW_HIERARCHY           |   337K|  9231K|       |  3514   (15)| 00:00:09 |       |       |
|   8 |   SORT AGGREGATE             |                               |     1 |    78 |       |             |          |       |       |
|*  9 |    HASH JOIN                 |                               |   927G|   65T|   534M|    74M (40)| 50:53:00 |       |       |
|  10 |     VIEW                     |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  11 |      TABLE ACCESS FULL       | SYS_TEMP_0FDA7485F_6A66C42E   |    22M|  1111M|       | 16808   (12)| 00       |       |       |
|* 12 |     HASH JOIN                |                               |    21G|  1272G|   534M|  1616K (43)| 01:06:04 |       |       |
|  13 |      VIEW                    |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  14 |       TABLE ACCESS FULL      | SYS_TEMP_0FDA7485F_6A66C42E   |    22M|  1111M|       | 16808   (12)| 0        |       |       |
|* 15 |      HASH JOIN               |                               |   476M|    23G|   524M| 97327   (22)| 00:03:59 |       |       |
|* 16 |       HASH JOIN              |                               |    10M|   401M|  8704K| 34520   (10)| 00:01:25 |       |       |
|* 17 |        HASH JOIN             |                               |   234K|  5948K|  8256K|   783   (10)| 00:00:02 |       |       |
|  18 |         VIEW                 |                               |   337K|  4286K|       |   142   (14)| 00:00:01 |       |       |
|  19 |          TABLE ACCESS FULL   | SYS_TEMP_0FDA74860_6A66C42E   |   337K|  3956K|       |   142   (14)| 00:00:01 |       |       |
|  20 |         VIEW                 |                               |   337K|  4286K|       |   142   (14)| 00:00:01 |       |       |
|  21 |          TABLE ACCESS FULL   | SYS_TEMP_0FDA74860_6A66C42E   |   337K|  3956K|       |   142   (14)| 00:00:01 |       |       |
|  22 |        VIEW                  |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  23 |         TABLE ACCESS FULL    | SYS_TEMP_0FDA7485F_6A66C42E   |    22M|  1111M|       | 16808   (12)| 00:00:42 |       |       |
|  24 |       VIEW                   |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  25 |        TABLE ACCESS FULL     | SYS_TEMP_0FDA7485F_6A66C42E   |    22M|  1111M|       | 16808   (12)| 0        |       |       |
```

Transaction Size (2:3)

```
-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name                          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                               |     1 |    78 |       |    14T(100)|999:59:59 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION |                               |       |       |       |            |          |       |       |
|   2 |   LOAD AS SELECT           |                               |       |       |       |            |          |       |       |
|   3 |    PARTITION RANGE ALL     |                               |    22M|  1111M|       | 38153   (11)| 00:01:34 |     1 |    29 |
|*  4 |     TABLE ACCESS FULL      | ELEMENT_DETAIL                |    22M|  1111M|       | 38153   (11)| 00:01:34 |       |       |
|   5 |   LOAD AS SELECT           |                               |       |       |       |            |          |       |       |
|   6 |    PARTITION HASH ALL      |                               |   337K|  9231K|       |  3514   (15)| 00:00:09 |     1 |    16 |
|*  7 |     TABLE ACCESS FULL      | MARK_NETW_HIERARCHY           |   337K|  9231K|       |  3514   (15)| 00:00:09 |       |       |
|   8 |   SORT AGGREGATE           |                               |     1 |    78 |       |            |          |       |       |
|   9 |    MERGE JOIN              |                               |   471P|   15E|       |    14T(100)|999:59:59 |       |       |
|  10 |     MERGE JOIN             |                               |    10P|   616P|       |   694G (81)|999:59:59 |       |       |
|  11 |      MERGE JOIN            |                               |   231T|    10P|       |   377G (64)|999:59:59 |       |       |
|  12 |       SORT JOIN            |                               |   334T|    11P|   28P|   377G (64)|999:59:59 |       |       |
|  13 |        MERGE JOIN CARTESIAN|                               |   334T|    11P|       |   140G (14)|999:59:59 |       |       |
|* 14 |         HASH JOIN          |                               |   989M|    23G|   534M| 96010   (38)| 00:03:56 |       |       |
|  15 |          VIEW              |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  16 |           TABLE ACCESS FULL| SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808   (12)| 00:00:42 |       |       |
|  17 |          VIEW              |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  18 |           TABLE ACCESS FULL| SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808   (12)| 00:00:42 |       |       |
|  19 |         BUFFER SORT        |                               |   337K|  4286K|       |   140G (14)|999:59:59 |       |       |
|  20 |          VIEW              |                               |   337K|  4286K|       |   142   (14)| 00:00:01 |       |       |
|  21 |           TABLE ACCESS FULL| SYS_TEMP_0FDA7485C_6A66C42E   |   337K|  3956K|       |   142   (14)| 00:00:01 |       |       |
|* 22 |       SORT JOIN            |                               |   337K|  4286K|    12M|   844   (14)| 00:00:03 |       |       |
|  23 |        VIEW                |                               |   337K|  4286K|       |   142   (14)| 00:00:01 |       |       |
|  24 |         TABLE ACCESS FULL  | SYS_TEMP_0FDA7485C_6A66C42E   |   337K|  3956K|       |   142   (14)| 00:00:01 |       |       |
|* 25 |      SORT JOIN             |                               |    22M|   277M|   855M| 65084   (16)| 00:02:40 |       |       |
|  26 |       VIEW                 |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  27 |        TABLE ACCESS FULL   | SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808   (12)| 0        |       |       |
|* 28 |     SORT JOIN              |                               |    22M|   277M|   855M| 65084   (16)| 00:02:40 |       |       |
|  29 |      VIEW                  |                               |    22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  30 |       TABLE ACCESS FULL    | SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808   (12)| 0        |       |       |
-----------------------------------------------------------------------------------------------------------------------------------
```

Transaction Size (3:3)

```
WITH ed AS (SELECT object_id, element_id, element_name, element_type, useid
            FROM pmcm.element_detail
            WHERE element_type = 'CELL'
            AND current_ind = 'Y'),
     mnh AS (SELECT parent_id, object_id
             FROM pmcm.mark_netw_hierarchy
             WHERE current_ind = 'Y'
             AND hierarchy_type = 'NETWORK'
             AND parent_type IN ('BSC', 'RNC'))
SELECT COUNT(*)
FROM ed e1_1, ed e1_2, ed e2_1, ed e2_2, mnh h1, mnh h2
WHERE e1_1.object_id = h1.parent_id AND   e1_2.object_id = h1.object_id
AND   e2_1.object_id = h2.parent_id AND   e2_2.object_id = h2.object_id
AND   e1_1.element_name = e2_1.element_name
AND   e1_1.element_id = e2_1.element_id
AND   e1_2.element_name = e2_2.element_name
AND   e1_2.element_id = e2_2.element_id
AND   e1_2.useid LIKE '*%'
AND   e2_2.useid NOT LIKE '*%';

| Id  | Operation                              | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                     |     1 |   214 |       |   100K  (6)| 00:04:08 |
|   1 |  HASH UNIQUE                           |                     |     1 |   214 |       |   100K  (6)| 00:04:08 |
|*  2 |   HASH JOIN                            |                     |     1 |   214 |    12M|   100K  (6)| 00:04:08 |
|   3 |    PARTITION HASH ALL                  |                     |   337K|  9231K|       |  3514  (15)| 00:00:09 |
|*  4 |     TABLE ACCESS FULL                  | MARK_NETW_HIERARCHY |   337K|  9231K|       |  3514  (15)| 00:00:00 |
|*  5 |    HASH JOIN                           |                     |   207K|    36M|    22M| 95860   (6)| 00:03:56 |
|   6 |     PARTITION RANGE ALL                |                     |   586K|    15M|       | 16233   (2)| 00:00:40 |
|   7 |      TABLE ACCESS BY LOCAL INDEX ROWID | ELEMENT_DETAIL      |   586K|    15M|       | 16233      | ??:??:?? |
|*  8 |       INDEX SKIP SCAN                  | ED_ET_TECH_CI       |   586K|       |       | 12791   (1)| 00:00:3? |
|*  9 |     HASH JOIN                          |                     |   207K|    31M|    22M| 77982   (7)| 00:03:12 |
|  10 |      PARTITION RANGE ALL               |                     |   586K|    15M|       | 16233   (2)| 00:00:40 |
|  11 |       TABLE ACCESS BY LOCAL INDEX ROWID | ELEMENT_DETAIL      |   586K|    15M|       | 16233      | ??:??:?? |
|* 12 |        INDEX SKIP SCAN                 | ED_ET_TECH_CI       |   586K|       |       | 12791   (1)| 00:00:?? |
|* 13 |      HASH JOIN                         |                     |   179K|    22M|    12M| 60372   (8)| 00:02:29 |
|  14 |       PARTITION HASH ALL               |                     |   337K|  9231K|       |  3514  (15)| 00:00:09 |
|* 15 |        TABLE ACCESS FULL               | MARK_NETW_HIERARCHY |   337K|  9231K|       |  3514  (15)| 00:00:?? |
|* 16 |       HASH JOIN                        |                     |   184K|    17M|    10M| 55886   (8)| 00:02:18 |
|  17 |        PARTITION RANGE ALL             |                     |   184K|  9008K|       | 37137   (8)| 00:01:32 |
|* 18 |         TABLE ACCESS FULL              | ELEMENT_DETAIL      |   184K|  9008K|       | 37137   (8)| 00:01:32 |
|  19 |        PARTITION RANGE ALL             |                     |   576K|    28M|       | 17383   (8)| 00:00:43 |
|* 20 |         TABLE ACCESS BY LOCAL INDEX ROWID| ELEMENT_DETAIL      |   576K|    28M|       | 17383   (8)| ??:??:?? |
|* 21 |          INDEX SKIP SCAN               | ED_ET_TECH_CI       |   583K|       |       | 13939   (9)| 00:00:35 |
-------------------------------------------------------------------------------------------------------------
```

Big Queries

How Big Is Your SQL? (1:5)

0cjpvvqy854kh	SELECT 'ERICSSON_CSCORE.EOS' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from ERICSSON_CSCORE.EOS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(s ysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_CSCORE.EOS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_CSCORE.MSCSTATS' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from ERICSSON_CSCORE.MSCSTATS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sumday, count(dist inct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_CSCORE.MSCSTATS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_CSCORE.MTRAFTYPE' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count , count (*) num_of_rows from ERICSSON_CSCORE.MTRAFTYPE where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_CSCORE.MTRAFTYPE_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_GERAN.CELLGPRS' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(d atetime_ins, 'mi') > trunc(sysdate)+3/24 then BSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then BSC end) ontime_count, count (*) num_of_rows from ERICSSON_GERAN.CELLGPRS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then BSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then BSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_GERAN.CELLGPRS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_GERAN.CELLSTATS' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_ count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then BSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then BSC end) ontime_count, count (*) num_of_rows from ERICSSON_GERAN.CELLSTATS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then BSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then BSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_GERAN.CELLSTATS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_PSCORE.MM' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count,

How Big Is Your SQL? (2:5)

How Big Is Your SQL? (3:5)

How Big Is Your SQL? (4:5)

late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.CCS7 where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.CCS7_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.LOCATIONAREACODE' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cut off, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.LOCATIONAREACODE where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.LOCATIONAREACODE_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.MSCCP_CP' AS TABLENAME, sumd ay, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.MSCCP_CP where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_c ount, sum(entries) num_of_sumrows from NORTEL_CSCORE.MSCCP_CP_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.MSCCP' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.MSCCP where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(s ysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.MSCCP_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.TRKGRP' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.TRKGRP where dateti me between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.TRKGRP_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.VLR6' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(d istinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.VLR6 where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case

How Big Is Your SQL? (5:5)

Large Database Challenges

- All of these different technologies are marvelous
  - They require organizations to spend huge sums of money on what is marketed as new and better
  - They demand that IT professionals be flexible and learn new skills
- But yet the challenges from the 1970s and 80s still exist unchanged
  - Is it stable?
  - Is it secure?
  - Can you back it up and recover to a point-in-time?
  - Can you guarantee business continuity?
- The conflict is between new technology whose benefits are theoretical and the uncertainty deploying it creates

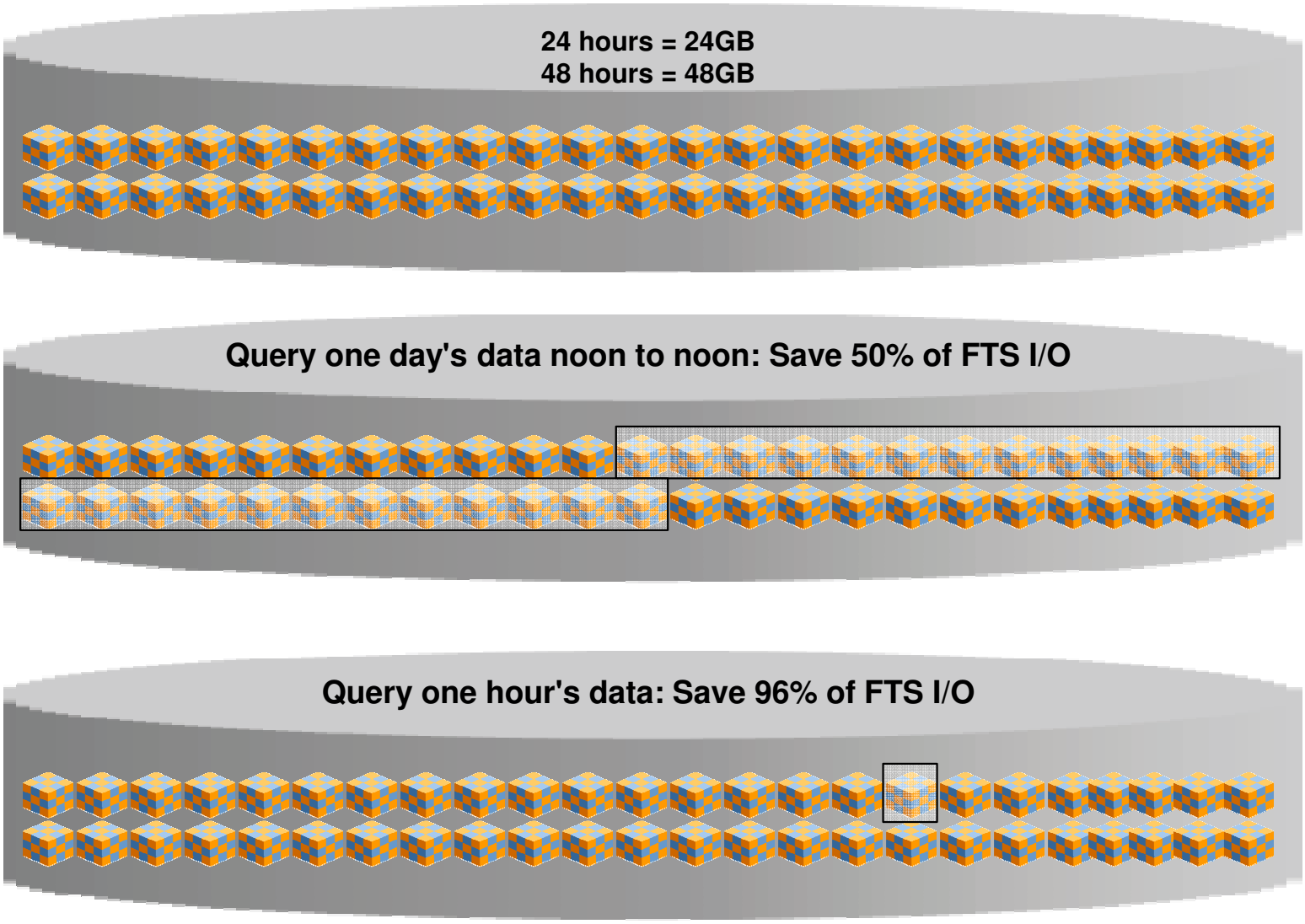
Large Database Challenges

Most of the interest in the new schema-less databases comes from developers that have no interest in learning about databases and have no interest in operations.

Large Database Challenges

- Is there a pool of people that can design for it?
- Is there a pool of people that can develop for it?
- Is there a pool of people that can run it in operations?
- Is there a support organization that knows it?
- How stable is the company selling it?
- How many years of experience does anyone have with it?
- How secure is it? Can it meet regulatory and governance requirements?

Partitioning



Repeating Issue: Unannounced Loads

ASHPCE1D

MMDD	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0804	0	0	0	0	0	0	0	0	0	0	0	0	5	32	18	65	91	13	12	20	84	9	14	9
0805	137	112	26	27	141	17	21	9	85	13	21	17	96	23	23	24	91	13	11	21	86	11	14	9
0806	151	111	21	24	96	41	50	14	84	22	20	22	91	18	17	18	92	24	10	11	83	9	14	20
0807	139	100	32	30	99	43	49	19	105	17	31	14	76	23	27	25	111	20	15	18	86	13	13	10
0808	145	99	29	30	109	52	48	11	102	25	47	24	101	23	20	23	117	31	30	16	91	12	11	9
0809	123	83	65	37	93	17	25	10	102	23	44	25	111	37	24	29	98	19	29	16	92	16	15	9
0810	169	120	52	32	125	58	38	9	109	17	26	14	104	13	17	15	93	13	16	11	61	10	10	9
0811	107	82	51	34	85	17	22	10	73	10	12	11	92	32	13	69	65	11	11	10	60	9	12	9
0812	149	121	26	15	70	16	24	11	95	34	15	18	34	67	21	21	87	11	13	9	77	9	14	9
0813	115	76	55	56	27	9	9	9	11	9	9	0	0	0	0	0	0	0	0	0	0	0	0	0

What patterns do you see in this data?

Repeating Issue: Unannounced Loads

ASHPCE1D

MMDD	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0804	0	0	0	0	0	0	0	0	0	0	0	0	5	32	18	65	91	13	12	20	84	9	14	9
0805	137	112	26	27	141	17	21	9	85	13	21	17	96	23	23	24	91	13	11	21	86	11	14	9
0806	151	111	21	24	96	41	50	14	84	22	20	22	91	18	17	18	92	24	10	11	83	9	14	20
0807	139	100	32	30	99	43	49	19	105	17	31	14	76	23	27	25	111	20	15	18	86	13	13	10
0808	145	99	29	30	109	52	48	11	102	25	47	24	101	23	20	23	117	31	30	16	91	12	11	9
0809	123	83	65	37	93	17	25	10	102	23	44	25	111	37	24	29	98	19	29	16	92	16	15	9
0810	169	120	52	32	125	58	38	9	109	17	26	14	104	13	17	15	93	13	16	11	61	10	10	9
0811	107	82	51	34	85	17	22	10	73	10	12	11	92	32	13	69	65	11	11	10	60	9	12	9
0812	149	121	26	15	70	16	24	11	95	34	15	18	34	67	21	21	87	11	13	9	77	9	14	9
0813	115	76	55	56	27	9	9	9	11	9	9	0	0	0	0	0	0	0	0	0	0	0	0	0

What patterns do you see in this data?

Computers Are Not Humans (1:4)

- Computers are not humans and tables are not paper forms
- This table will be accessed by person_id or state: No one will ever put the address2 column into the WHERE clause as a filter

Common Design

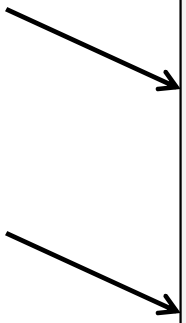
```
CREATE TABLE customers (
person_id   NUMBER,
first_name  VARCHAR2(30) NOT NULL,
middle_init VARCHAR2(2),
last_name   VARCHAR2(30) NOT NULL,
address1    VARCHAR2(30),
address2    VARCHAR2(30),
city        VARCHAR2(30),
state       VARCHAR2(2));
```

Optimized Design

```
CREATE TABLE customers (
person_id   NUMBER,
last_name   VARCHAR2(30) NOT NULL,
state       VARCHAR2(2)  NOT NULL,
city        VARCHAR2(30) NOT NULL,
first_name  VARCHAR2(30) NOT NULL,
address1    VARCHAR2(30),
address2    VARCHAR2(30),
middle_init VARCHAR2(2));
```

Computers Are Not Humans (2:4)

- Proof that column order matters



```
CREATE TABLE read_test AS
SELECT *
FROM apex_040200.wwv_flow_page_plugs
WHERE rownum = 1;

SQL> explain plan for
  2  select * from read_test;

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |     1 |   214K|     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| READ_TEST |     1 |   214K|     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------

-- fetch value from column 1
Final cost for query block SEL$1 (#0) - All Rows Plan:
  Best join order: 1
  Cost: 2.0002  Degree: 1  Card: 1.0000  Bytes: 13
  Resc: 2.0002  Resc_io: 2.0000  Resc_cpu: 7271
  Resp: 2.0002  Resp_io: 2.0000  Resc_cpu: 7271

-- fetch value from column 193
Final cost for query block SEL$1 (#0) - All Rows Plan:
  Best join order: 1
  Cost: 2.0003  Degree: 1  Card: 1.0000  Bytes: 2002
  Resc: 2.0003  Resc_io: 2.0000  Resc_cpu: 11111
  Resp: 2.0003  Resp_io: 2.0000  Resc_cpu: 11111
```

Computers Are Not Humans (3:4)

- And what is seemingly simple for a human can be extremely expensive for a computer
- Consider what is involved in deleting a table
  - Determine if the table exists
  - Determine if the user has permission to delete the table
    The permission could be granted in multiple ways
  - Determine if there are indexes on the table
  - Determine if there are constraints on the table
  - Determine if there are triggers on the table
  - Determine if there are view based on the table
  - Determine if there are synonyms that reference the table
  - Determine whether any programs reference the table
  - Determine whether any object types reference the table
  - Determine whether there are any locks on the table
  - Determine whether any jobs reference the table
  - Determine whether a table column is a LOB (Large Object)

Computers Are Not Humans (4:4)

- Consider what is involved in deleting a table (cont.)
  - Determine whether an operator references the table
  - Determine whether the table is part of a queue
  - Determine whether the table has an identity column
  - Determine whether the table is partitioned
  - Determine whether auditing is being performed on the table
  - Determine whether the table is in a cluster and which one
  - Determine whether a table column is stored in a SecureFile
  - Determine what tablespace space the table is in

- Now go into the data dictionary and, in the proper sequence lock resources, perform updates and deletes, and finally unlock locked resources

What is Business Intelligence?

- In the Paleolithic period we called it "Report Writing"
- In the Neolithic period we called it "Data Mining"
- Now we call it "Business Intelligence" because customers will pay more for BI than for a report
- If you join our industry and wait 5 to 7 years not much will change ... except the name ... and an entirely new name will make it fresh, exciting, and more expensive

Business Intelligence Challenges

- Integrating relational and non-relational data
- Integrating data from different vendors
- Producing actionable results

Our Biggest Challenges Have Not Changed

```
Aug  8 12:56:04 orap1n1 ntpd[1339]: ntpd exiting on signal 15
Aug  8 12:57:27 orap1n1 ntpdate[12406]: step time server 10.2.255.254 offset 82.262906 sec
Aug  8 12:57:27 orap1n1 ntpd[12408]: ntpd 4.2.2p1@1.1570-o Fri Jul 22 18:07:53 UTC 2011 (1)
Aug  8 12:57:27 orap1n1 ntpd[12409]: precision = 1.000 usec
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface wildcard, 0.0.0.0#123 Disabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface wildcard, ::#123 Disabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond2, fe80::217:a4ff:fe77:fc18#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface lo, ::1#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond0, fe80::217:a4ff:fe77:fc10#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface eth2, fe80::217:a4ff:fe77:fc14#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface eth3, fe80::217:a4ff:fe77:fc16#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface lo, 127.0.0.1#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface eth2, 192.168.0.11#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface eth2:1, 169.254.34.21#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface eth3, 192.168.0.12#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface eth3:1, 169.254.139.181#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond0, 10.2.78.11#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond0:1, 10.2.78.10#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond0:3, 10.2.78.102#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond0:4, 10.2.78.100#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: Listening on interface bond2, 10.2.2.14#123 Enabled
Aug  8 12:57:27 orap1n1 ntpd[12409]: kernel time sync status 0040
Aug  8 12:57:27 orap1n1 ntpd[12409]: frequency initialized 0.000 PPM from /var/lib/ntp/drift
```

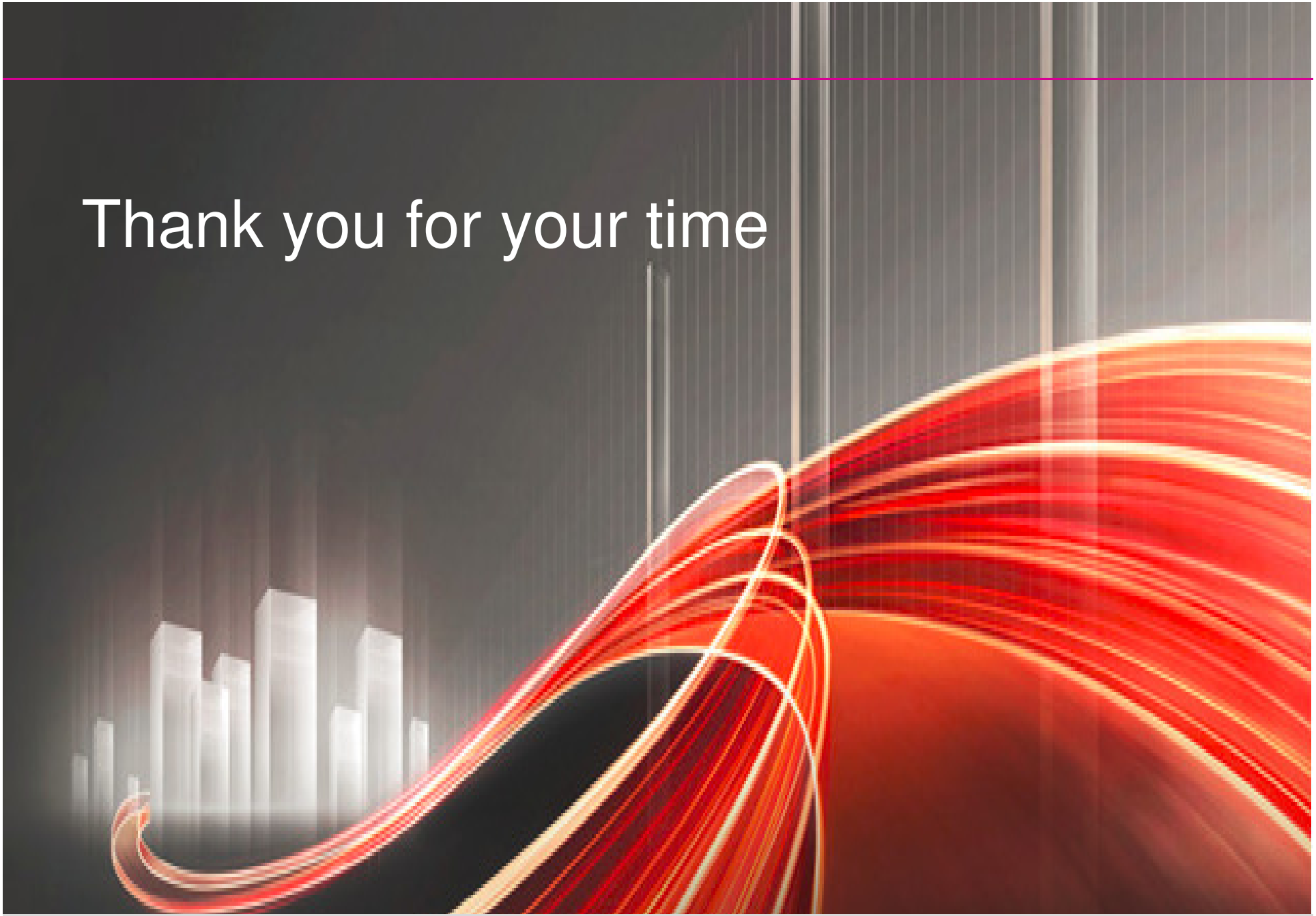
Your Biggest Challenges Will Not Change

- "Above all do no harm"
- You expect your doctor to keep current
- How are you going to keep your skills current?

Conclusion

- Never make an argument based on technology when you can make an argument based on money
- Green is good and Red is bad
- Learn to use PowerPoint well

Thank you for your time



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