embarcadero

heterogeneous database
& developer tools

DB Performance with Graphics

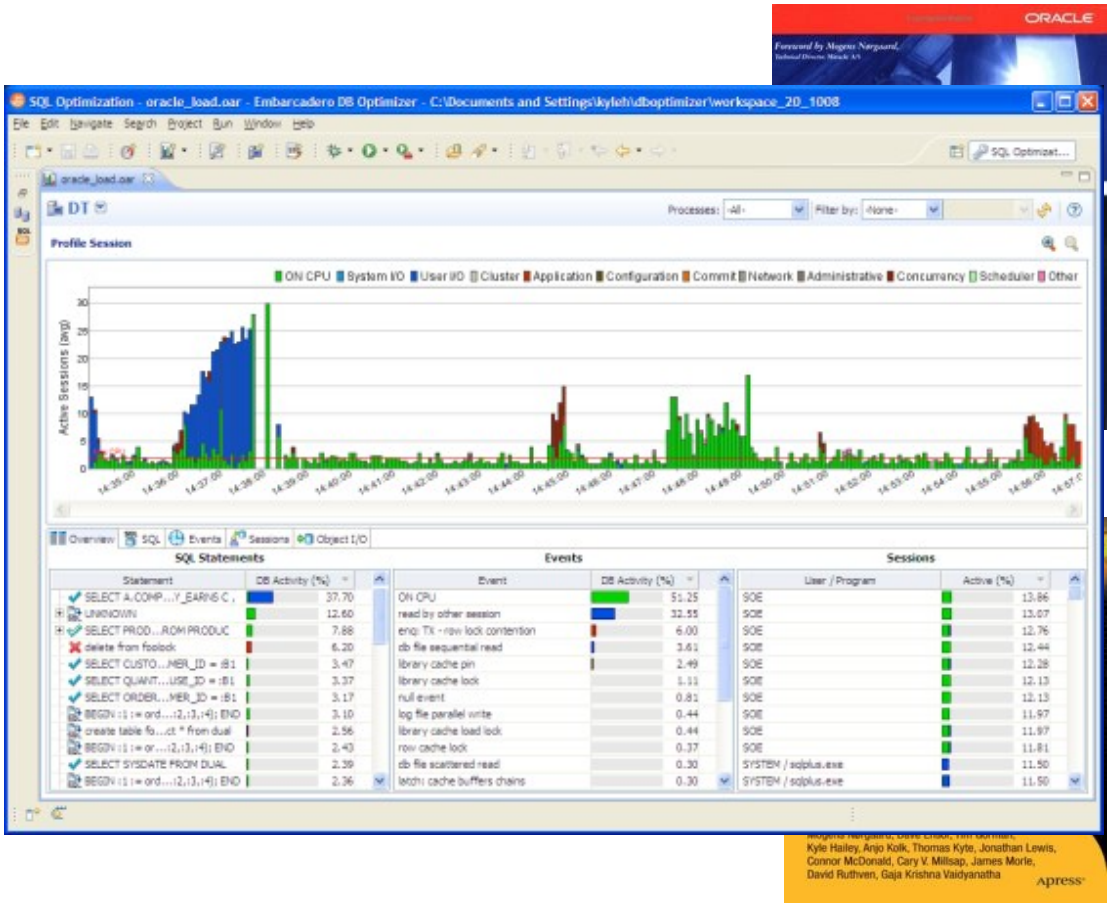
With Kyle Hailey

Product Manger for DB Optimizer

OracleMonitor.com

Who is Kyle Hailey

- 1990 Oracle
  - 90 support
  - 92 Ported v6
  - 93 France
  - 95 Benchmarking
  - 98 ST Real World Performance
- 2000 Dot.Com
- 2001 Quest
- 2002 Oracle OEM 10g
- 2006 Independent
- 2008 Embarcadero
  - DB Optimizer



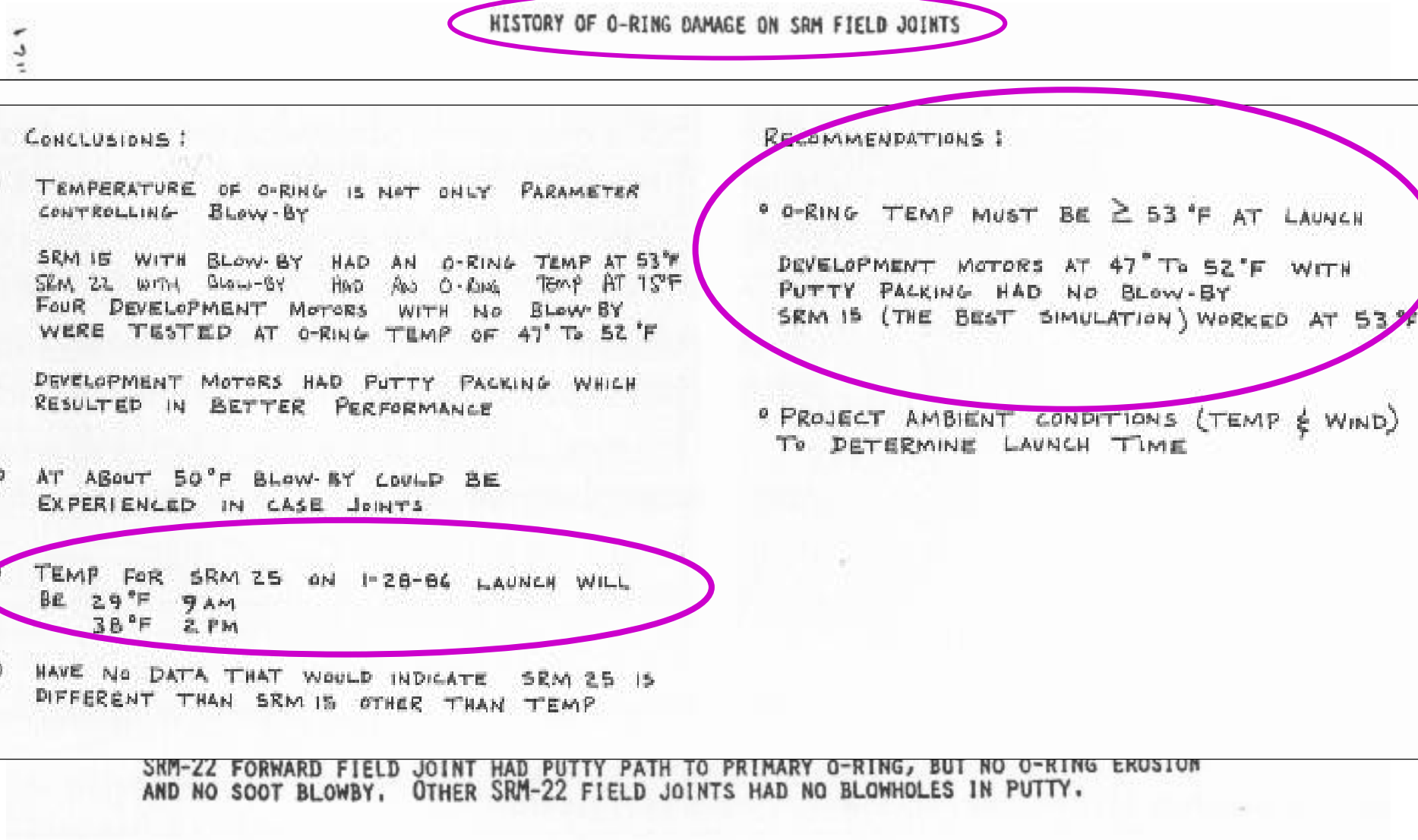
embarcadero

Launch: Pressure

Midnight before
January 28, 1986
Lives are on the line

Thanks to Edward Tufte

13 Pages Faxed



HISTORY OF O-RING DAMAGE ON SRM FIELD JOINTS

CONCLUSIONS:

- TEMPERATURE OF O-RING IS NOT ONLY PARAMETER CONTROLLING BLOW-BY

  SRM 15 WITH BLOW-BY HAD AN O-RING TEMP AT 53°F
  SRM 22 WITH BLOW-BY HAD AN O-RING TEMP AT 75°F
  FOUR DEVELOPMENT MOTORS WITH NO BLOW-BY WERE TESTED AT O-RING TEMP OF 47° TO 52°F

  DEVELOPMENT MOTORS HAD PUTTY PACKING WHICH RESULTED IN BETTER PERFORMANCE

- AT ABOUT 50°F BLOW-BY COULD BE EXPERIENCED IN CASE JOINTS

- TEMP FOR SRM 25 ON 1-28-86 LAUNCH WILL BE 29°F 9 AM
  38°F 2 PM

- HAVE NO DATA THAT WOULD INDICATE SRM 25 IS DIFFERENT THAN SRM 15 OTHER THAN TEMP

RECOMMENDATIONS:

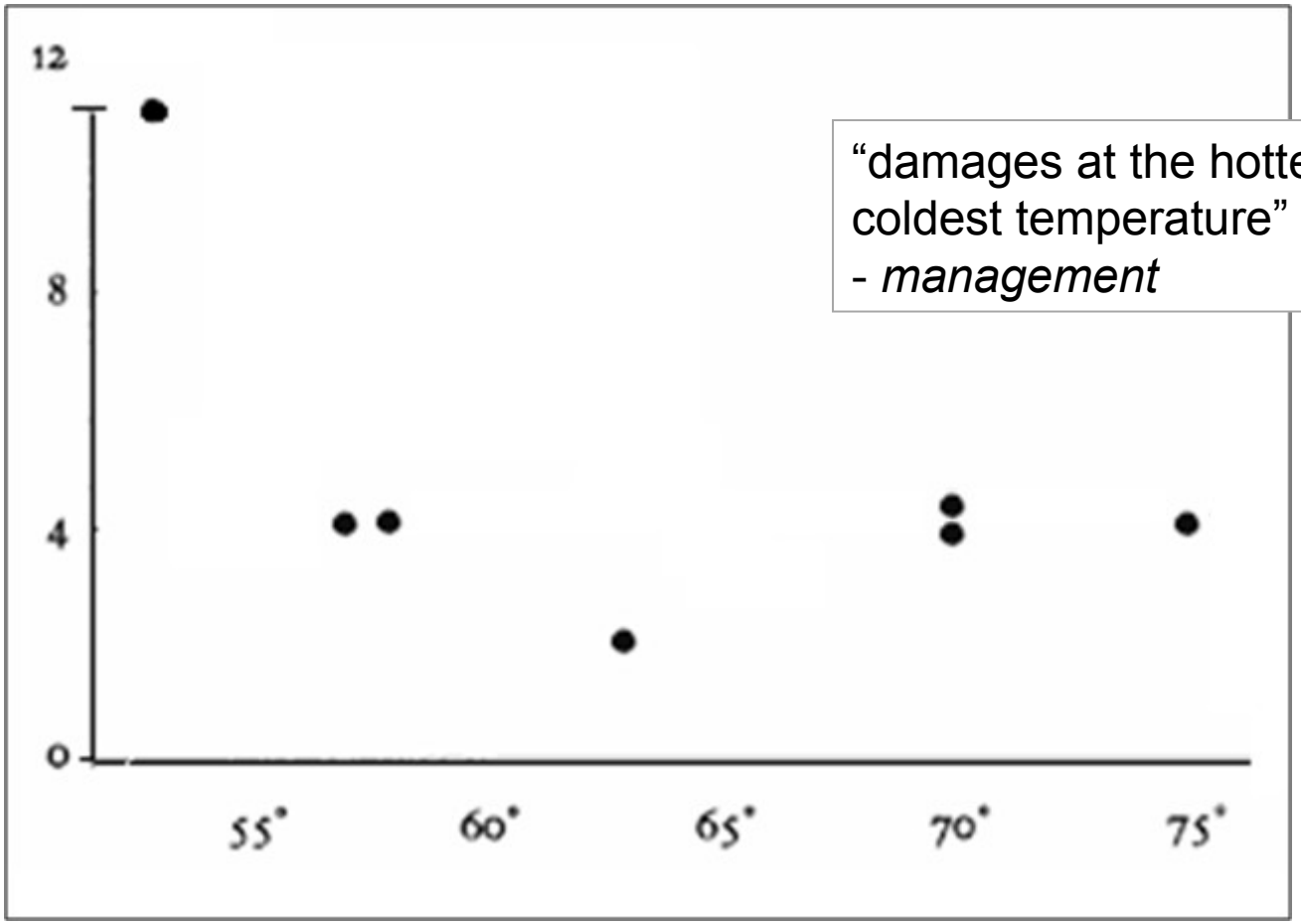
- O-RING TEMP MUST BE ≥ 53°F AT LAUNCH

  DEVELOPMENT MOTORS AT 47° TO 52°F WITH PUTTY PACKING HAD NO BLOW-BY
  SRM 15 (THE BEST SIMULATION) WORKED AT 53°F

- PROJECT AMBIENT CONDITIONS (TEMP & WIND) TO DETERMINE LAUNCH TIME

SRM-22 FORWARD FIELD JOINT HAD PUTTY PATH TO PRIMARY O-RING, BUT NO O-RING EROSION AND NO SOOT BLOWBY. OTHER SRM-22 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY.

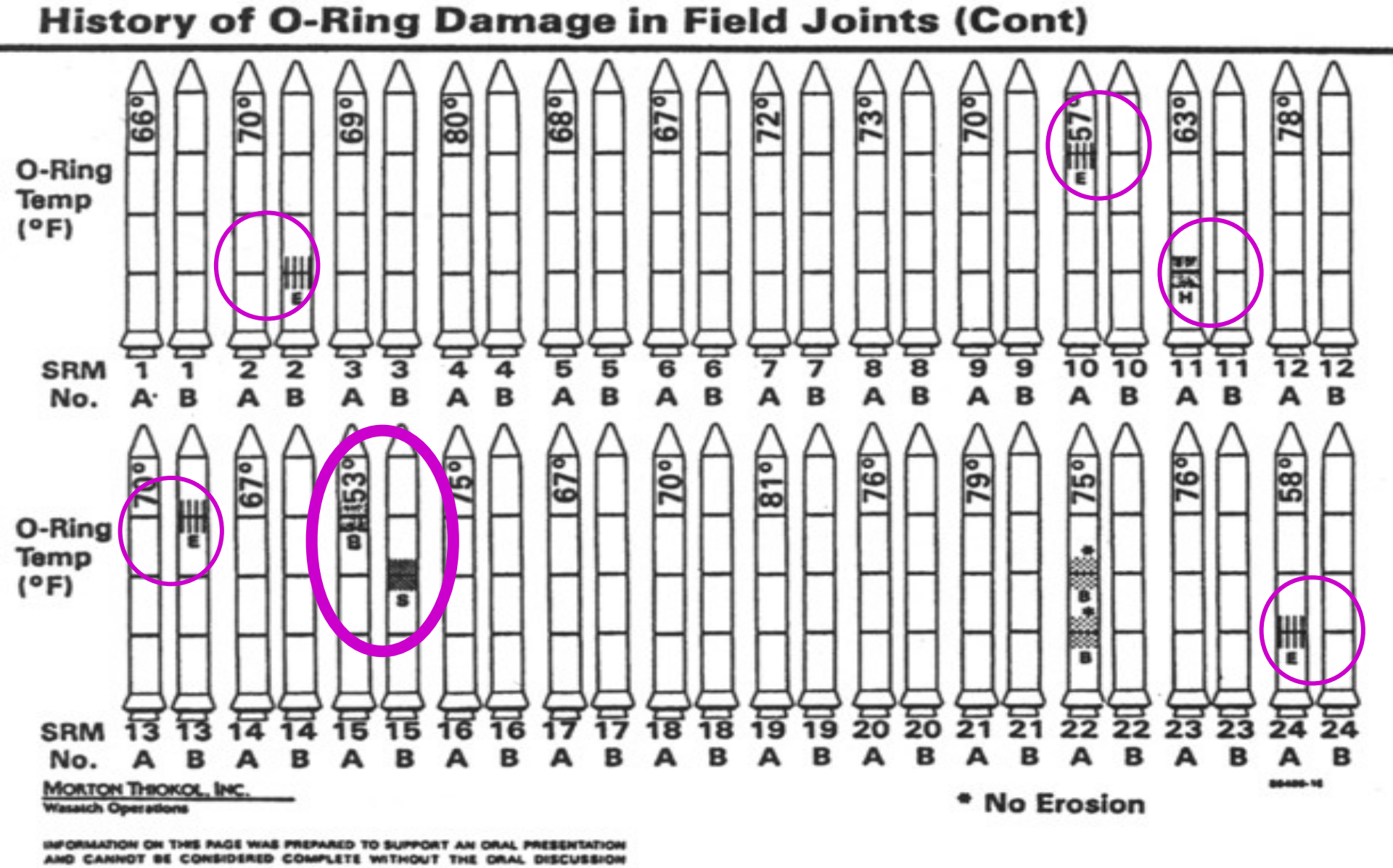
Original Engineering data



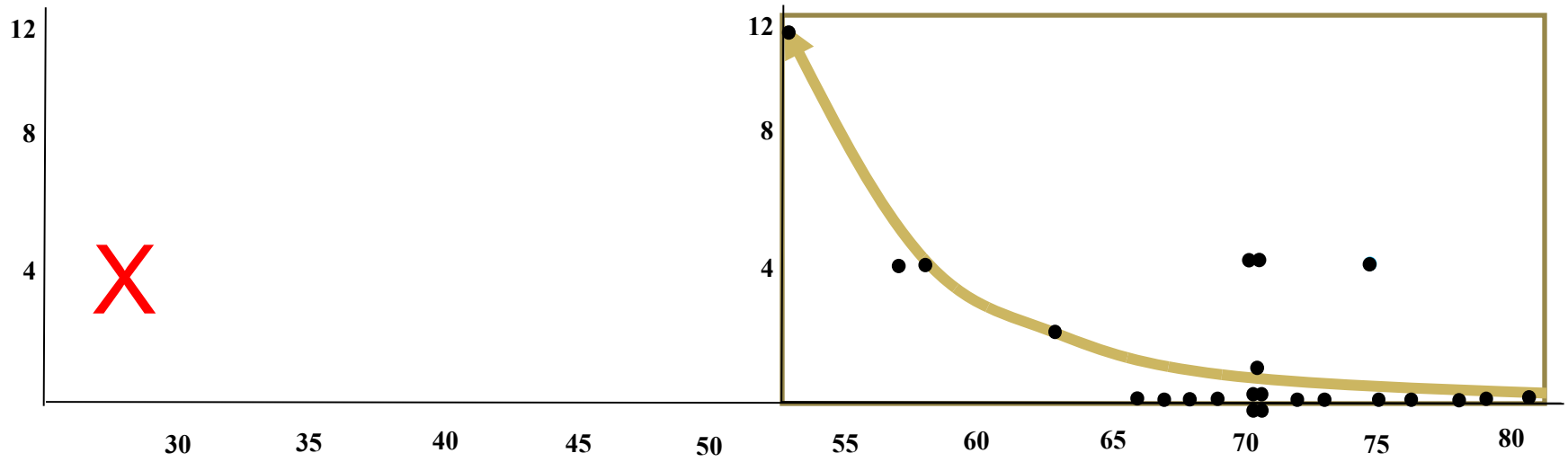
"damages at the hottest and coldest temperature"
- *management*

only showed damage

Congressional Hearings Evidence



Clearer



1. Include successes
2. Mark Differences
3. Normalize same temp
4. Scale known vs unknown

Difficult

- NASA Engineers Fail
- Congressional Investigators Fail
- Data Visualization is Difficult

But …

Lack of Clarity can be devastating

Solutions

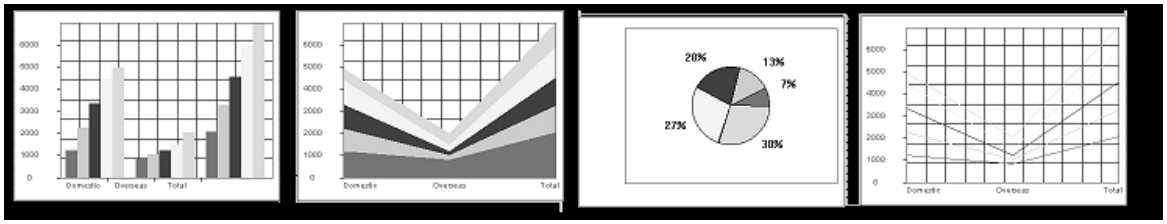
- Clear Identification
  - Know how to identify problems and issues
- Access to details
  - Provide solutions and/or information to address the issues
- Graphics
  - Easy understanding, effective communication and discussion

First Step: Graphics

"The humans … are exceptionally good at parsing ***visual*** information, especially when that information is coded by **color** and/or _____ ."

motion

Knowledge representation in cognitive science. Westbury, C. & Wilensky, U. (1998)



embarcadero

Why Use Graphics

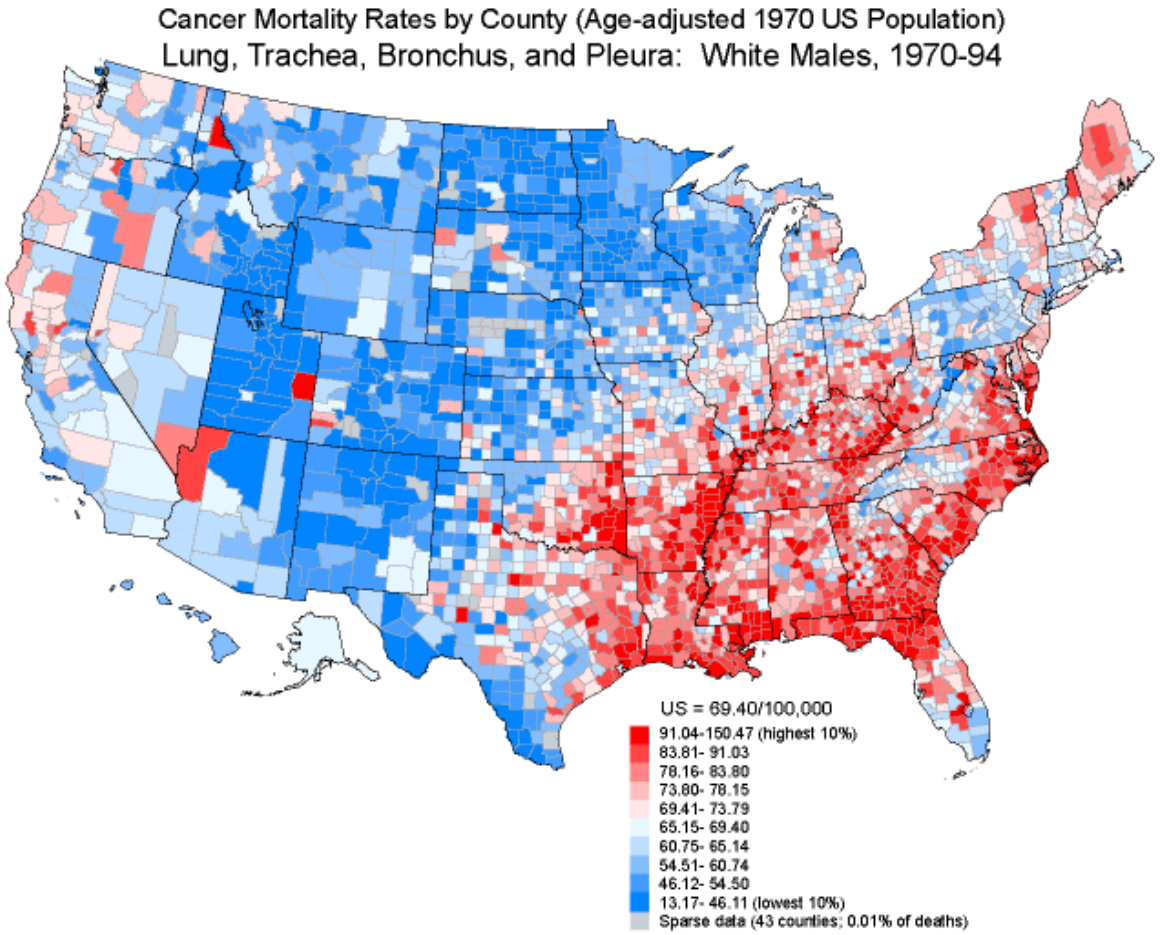
You can't imagine how many times I was told that nobody wanted or would use graphics …

-- *Jef Raskin, the creator of the Macintosh*

Infocus – (overhead projectors) sited a study that humans can parse graphical information 400,000 times faster than textual data

Counties in US

- 3101 Counties in US
- 50 pages

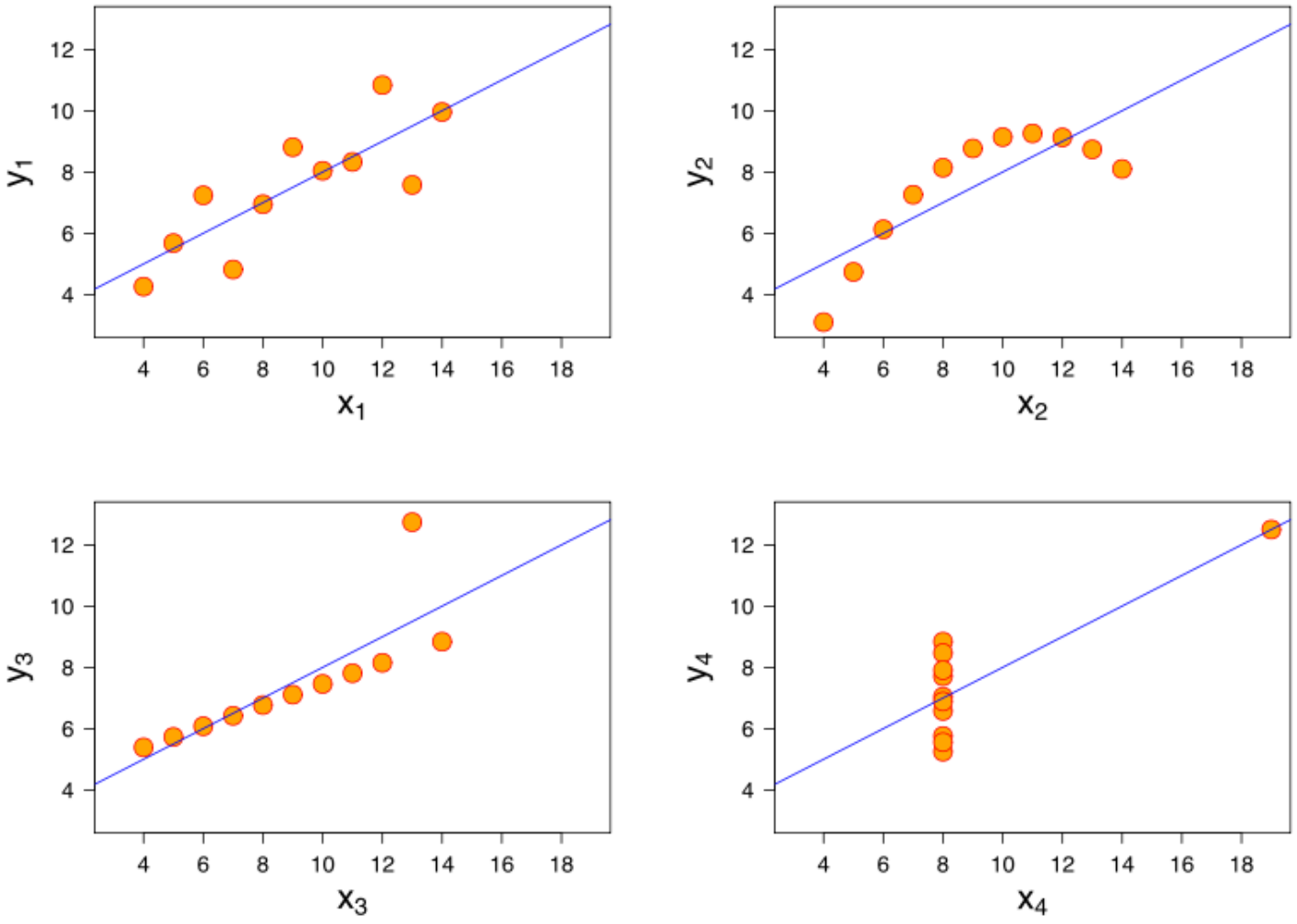


“If I can't picture it, I can't understand it”

- Albert Einstein

	Anscombe's Quartet							
	I		II		III		IV	
	x	y	x	y	x	y	x	y
	10	8.04	10	9.14	10	7.46	8	6.58
	8	6.95	8	8.14	8	6.77	8	5.76
	13	7.58	13	8.74	13	12.74	8	7.71
	9	8.81	9	8.77	9	7.11	8	8.84
	11	8.33	11	9.26	11	7.81	8	8.47
	14	9.96	14	8.1	14	8.84	8	7.04
	6	7.24	6	6.13	6	6.08	8	5.25
	4	4.26	4	3.1	4	5.39	19	12.5
	12	10.84	12	9.13	12	8.15	8	5.56
	7	4.82	7	7.26	7	6.42	8	7.91
	5	5.68	5	4.74	5	5.73	8	6.89
Average	9	7.5	9	7.5	9	7.5	9	7.5
Standard Deviation	3.31	2.03	3.31	2.03	3.31	2.03	3.31	2.03
Linear Regression	1.33		1.33		1.33		1.33	

Graphics for Anscombe's Quartet



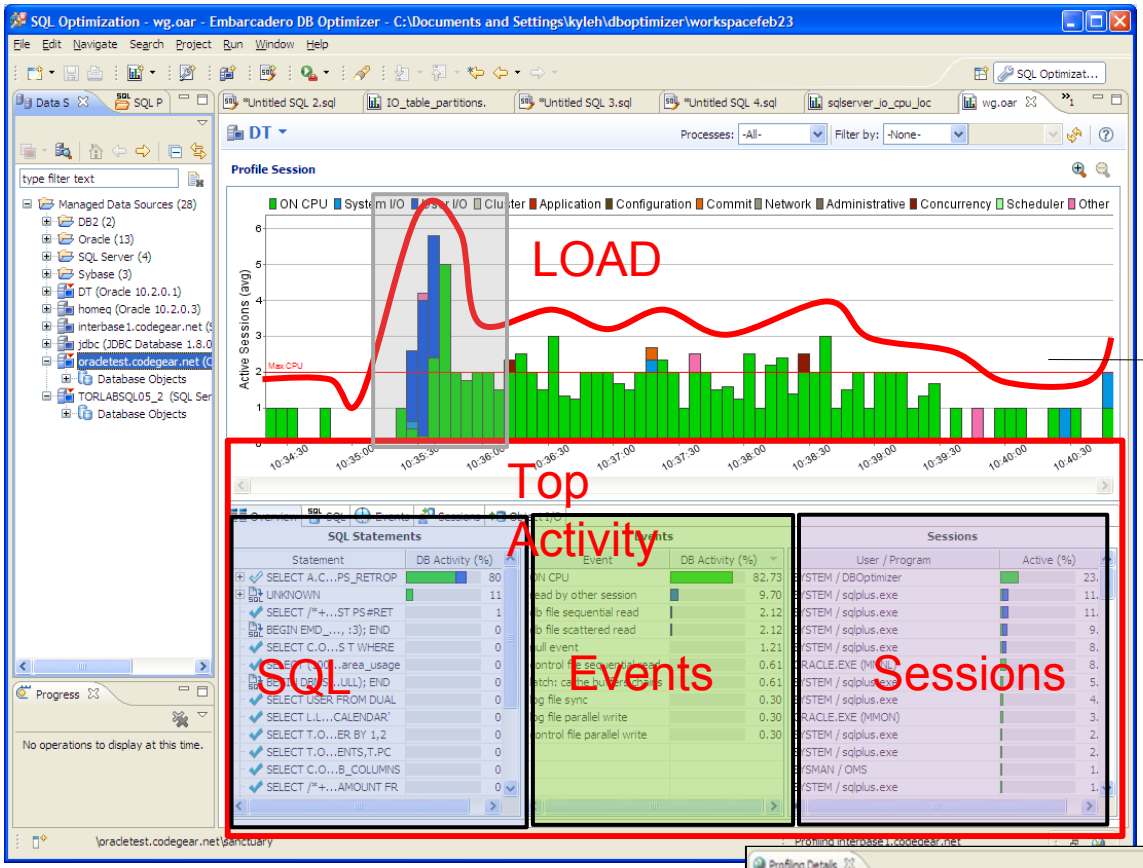
Tuning the Database

Complex

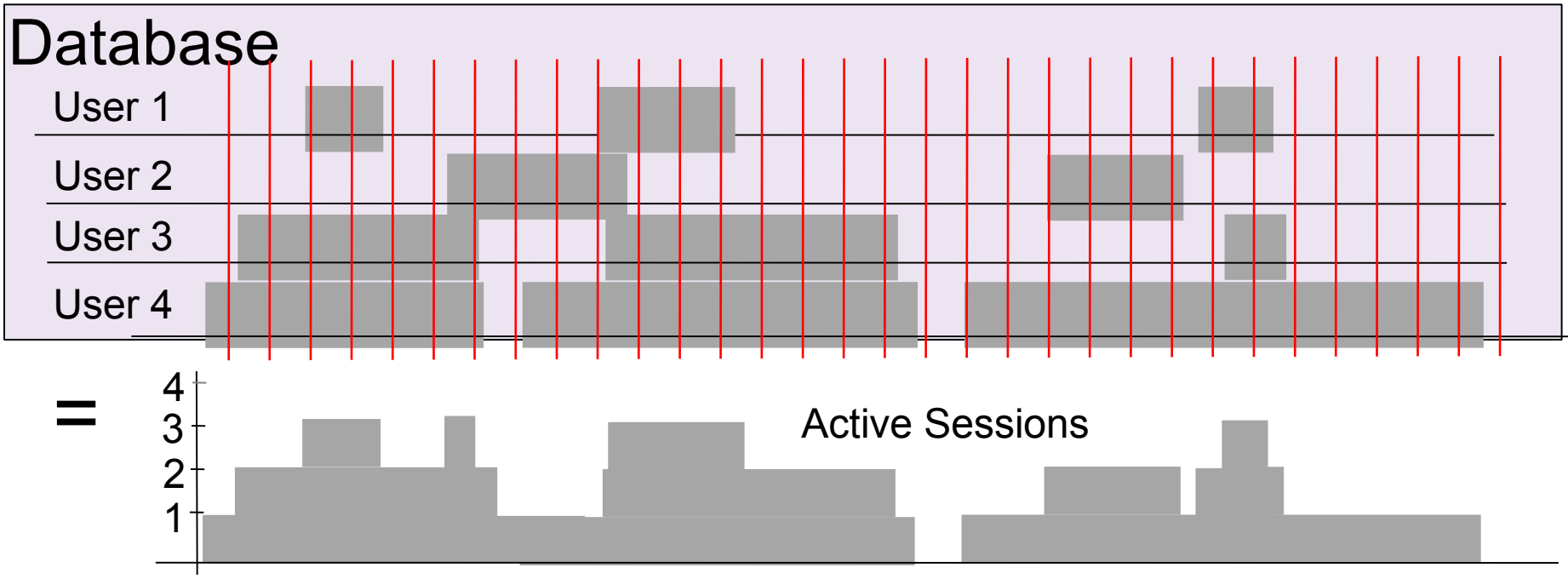
Averages

	Anscombe's Quartet							
	I		II		III		IV	
	x	y	x	y	x	y	x	y
Average	9	7.5	9	7.5	9	7.5	9	7.5
Standard Deviation	3.31	2.03	3.31	2.03	3.31	2.03	3.31	2.03
Linear Regression	1.33		1.33		1.33		1.33	

How Can We Open the Black Box?



How do we get our data?



Graph represents # of sessions active, but also represents amount of time active in the database
For Every Active Session there is a user (or application) waiting

Sample v$session : fast and light weight
We collect

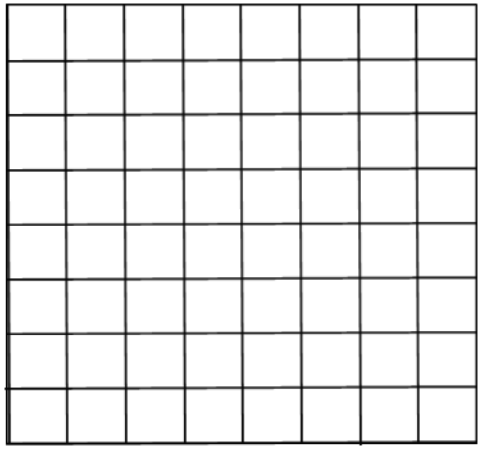
- session
- SQL
- State (CPU, IO, LOCK, Other Waits)
- and many other columns

Multi-dimensional fine grain data. Difficult to use w/o graphics

AWR collects

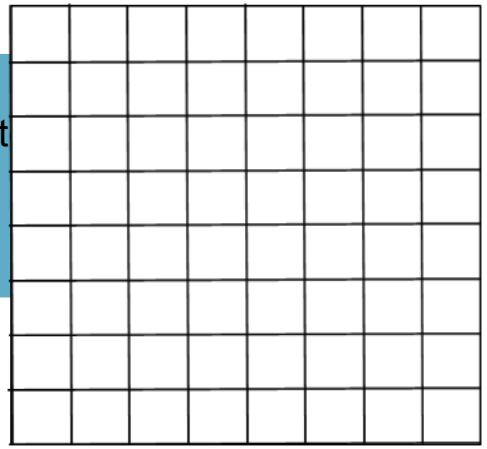
SQL (v$sqlstats)

```
F1qcyh20550cf
fj6gjgsshtxyx
0cjsxw5ndqdbc
8t8as9usk11qw
dr1rkrznhh95b
10dkqv3kr8xa5
38zhkf4jdyff4
298wmz1kxjs1m
```



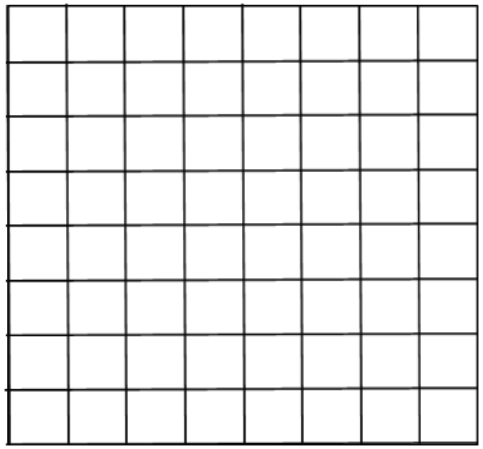
Waits (v$system_event)

CPU
Enq: TX – row lock cont
SQL*Net break/reset to
db file scattered read
db file sequential read



Sessions (v$sesstat)

25
34
36
38
45
63



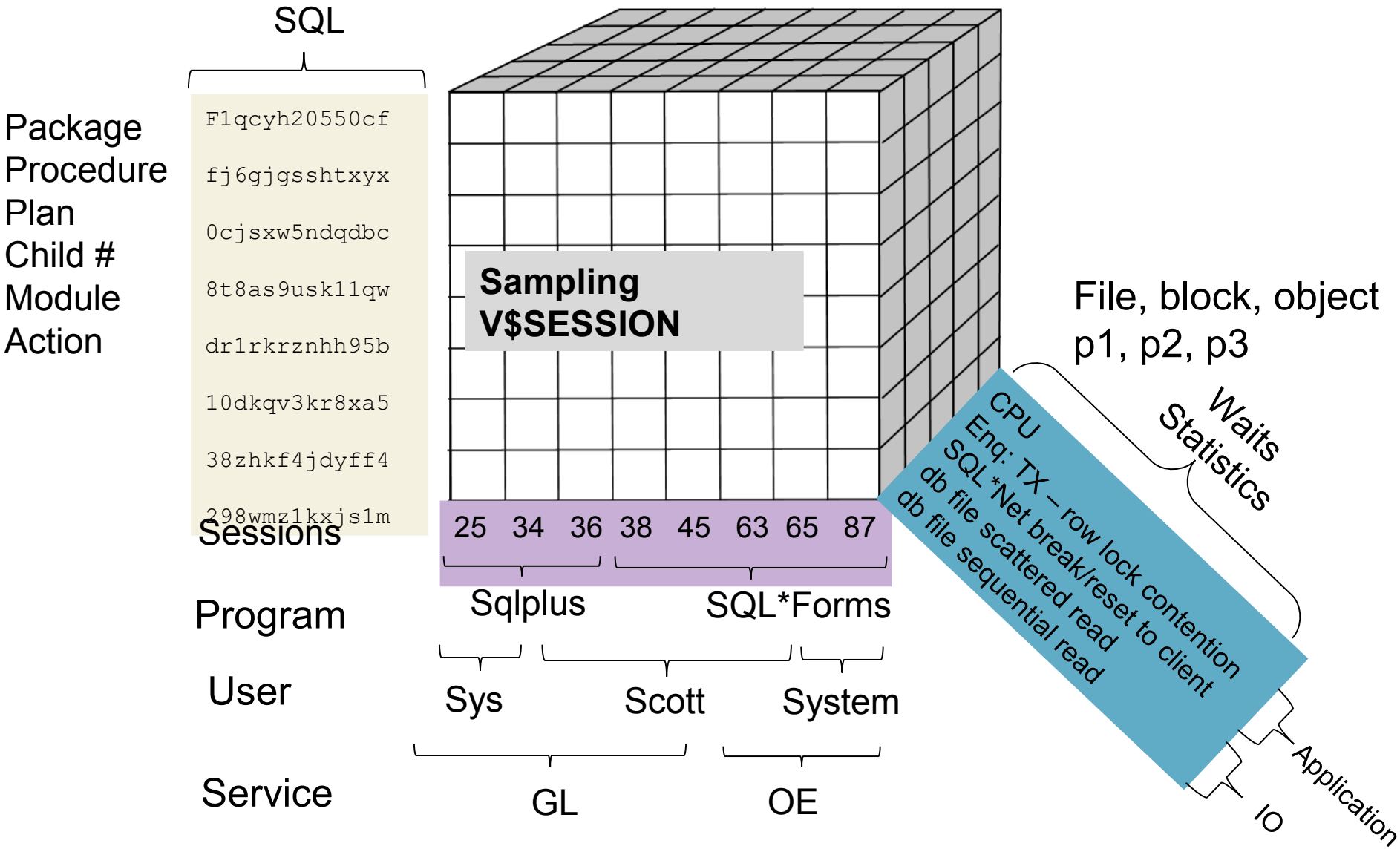
Which session executed which SQL?

Which SQL was block on a wait event ?

AWR (and Statspack)
Can't answer!

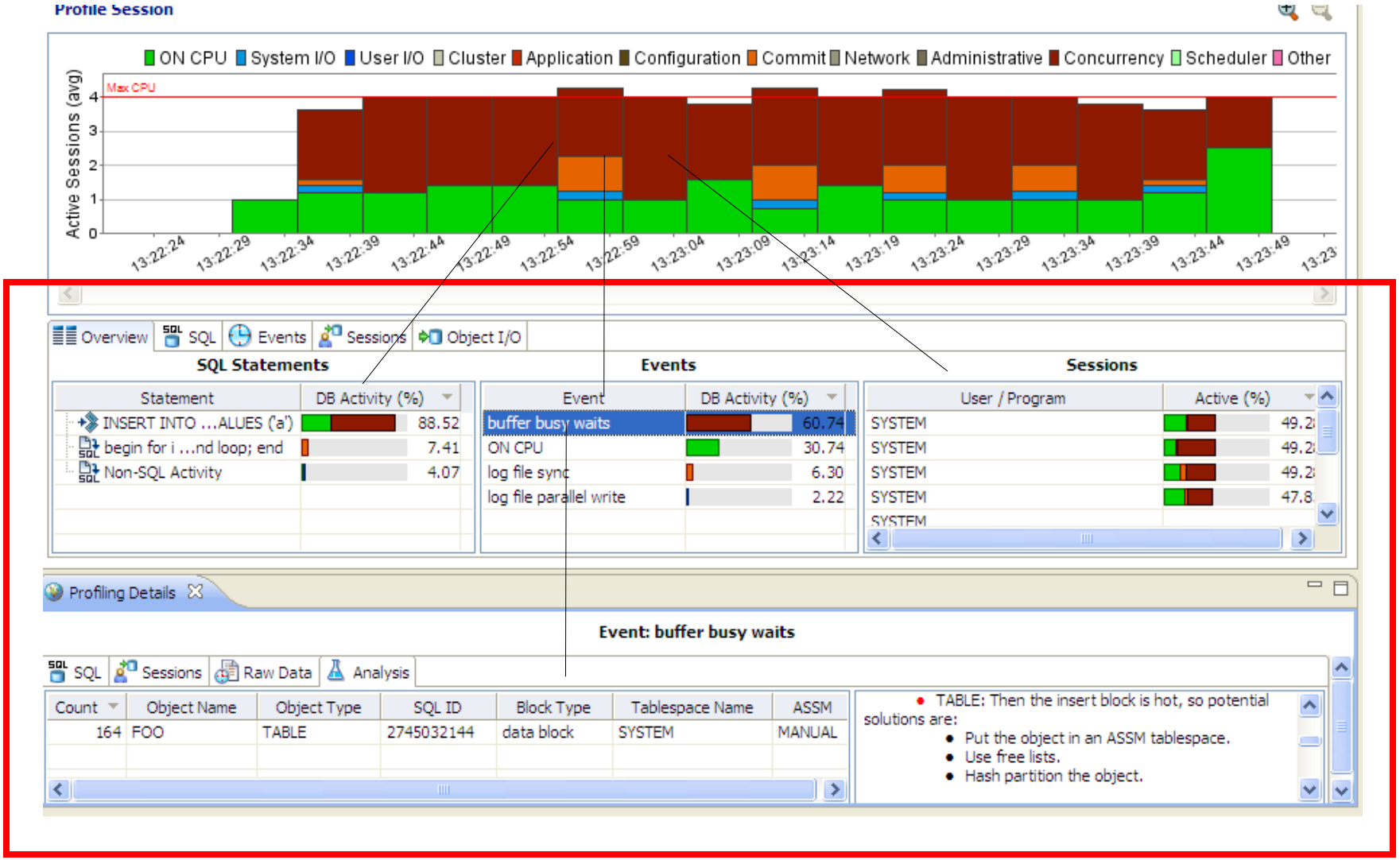
Old Method

After sampling Multi-dimensional



embarcadero

DB Optimizer shows it all

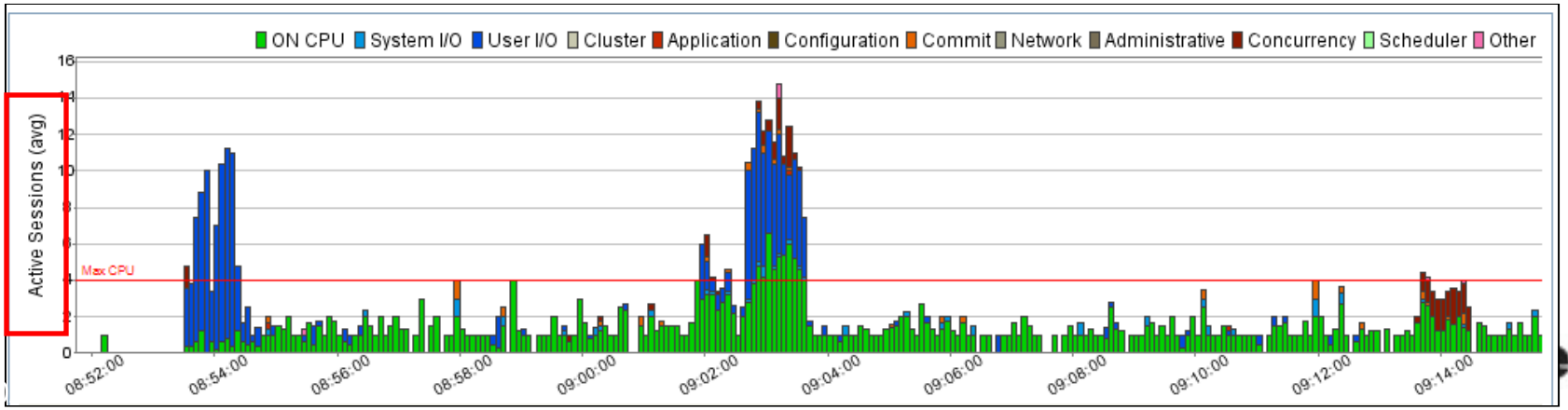


embarcadero

Average Active Sessions (AAS)

Use CPU count as yardstick:

- ✓ AAS < 1
  - Database is not blocked
- ✓ AAS ~= 0
  - Database basically idle
  - Problems are in the APP not DB
- ❖ AAS >> # of CPUS
  - There is a bottleneck



21

Interpreting the Load Chart

1. Application
   - Code inefficient?
1. Database
   - Configured correctly?
1. Machine
   - Is the machine undersized?
1. SQL
   - Inefficient SQL?

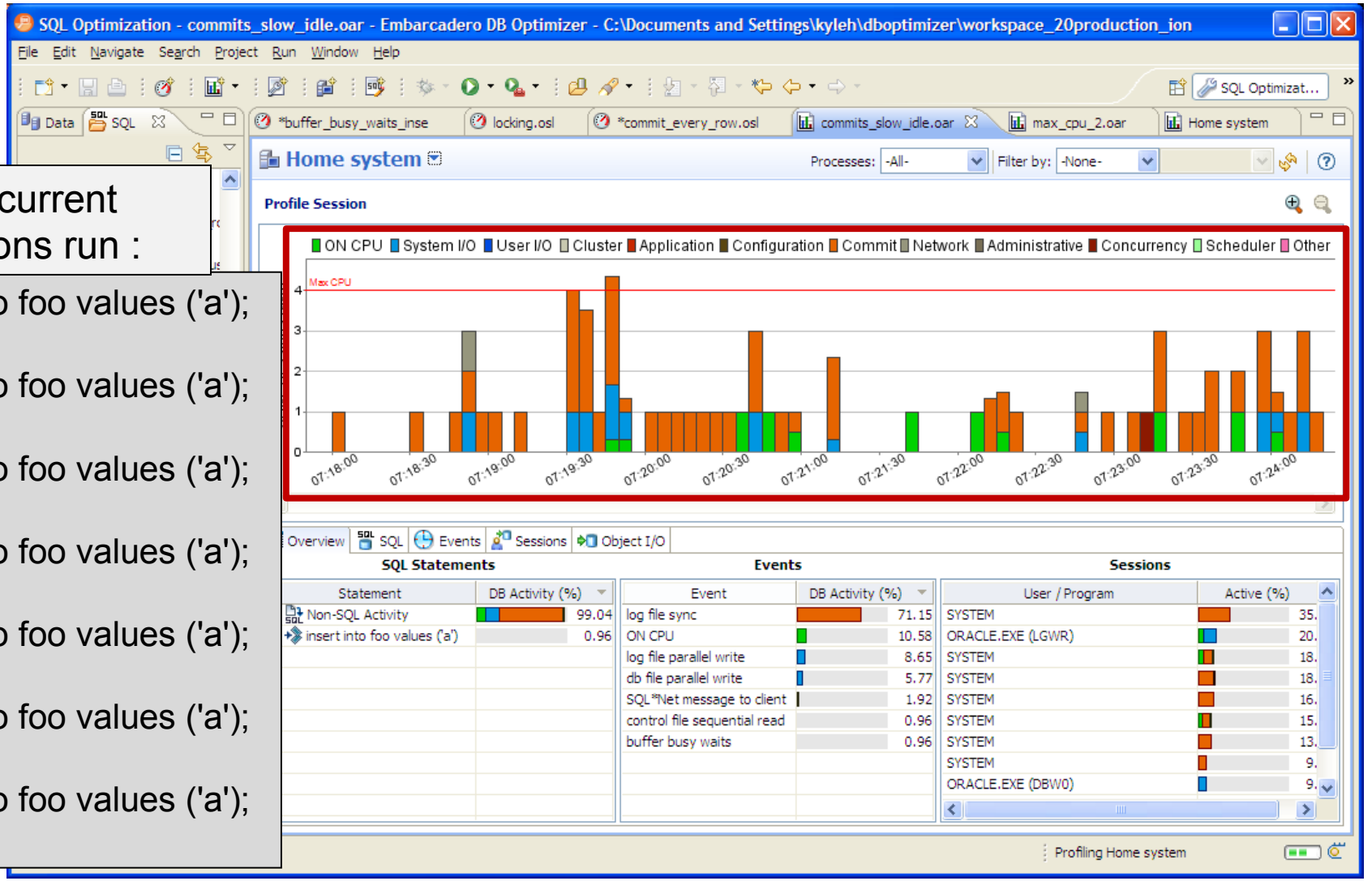
Who's problem is it ?

Chart answers this question and assists with solutions

1. Application Issues

4 concurrent sessions run :

```
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
insert into foo values ('a');
commit;
```

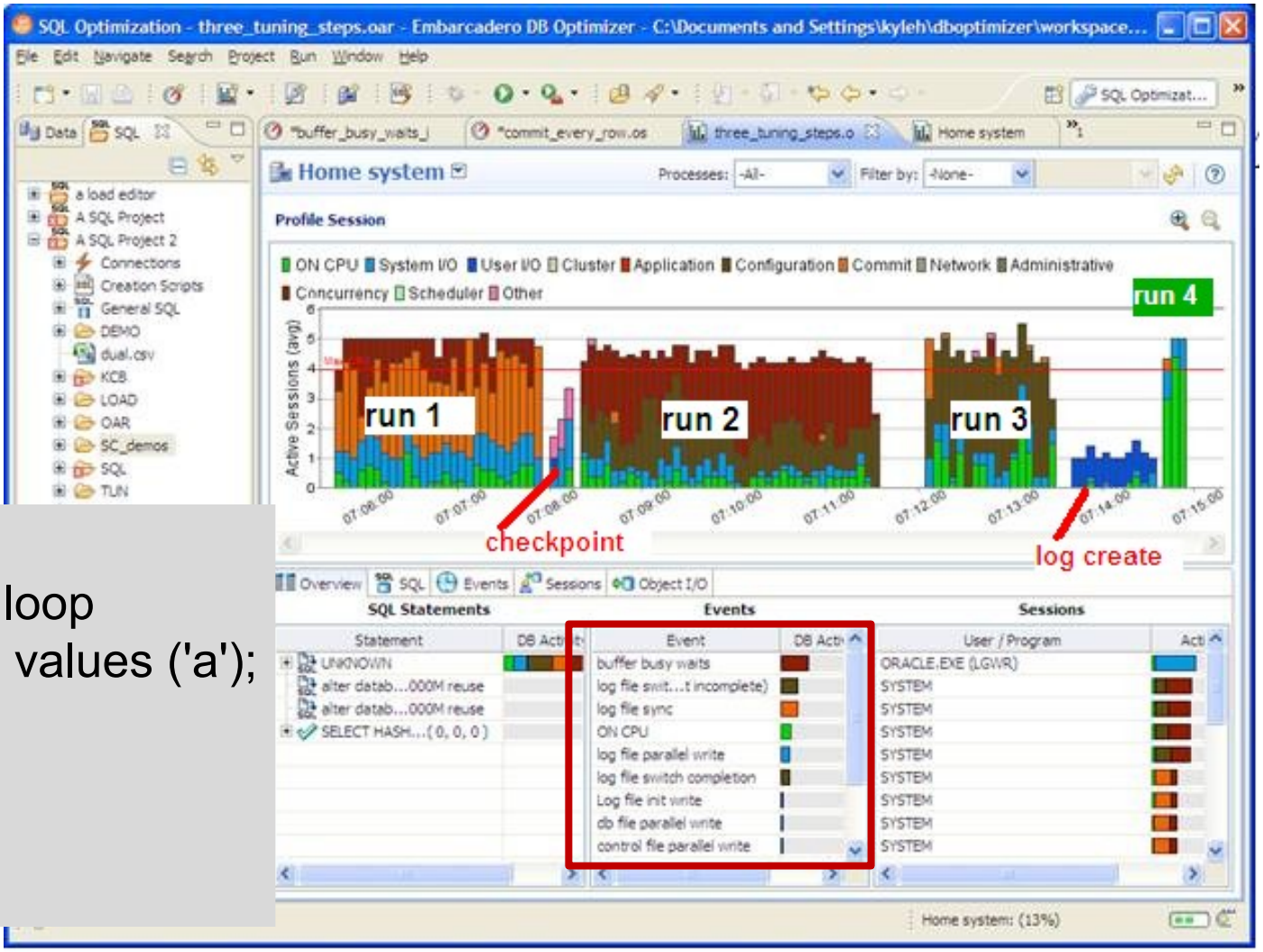


embarcadero

2. Database issue

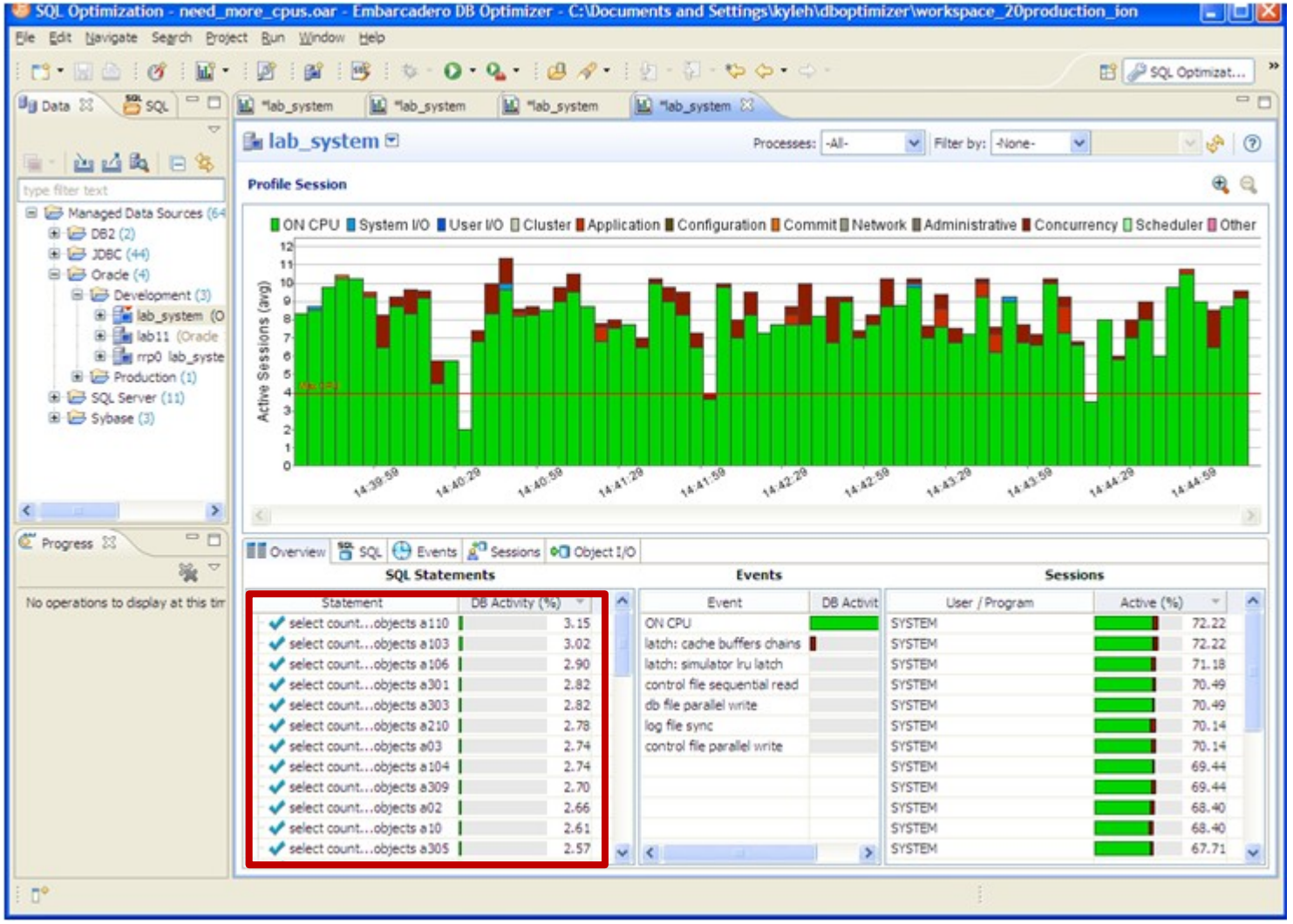
4 concurrent sessions run :

```
begin
  for i in 1..1000 loop
    insert into foo values ('a');
  end loop;
end;
/
Commit;
```



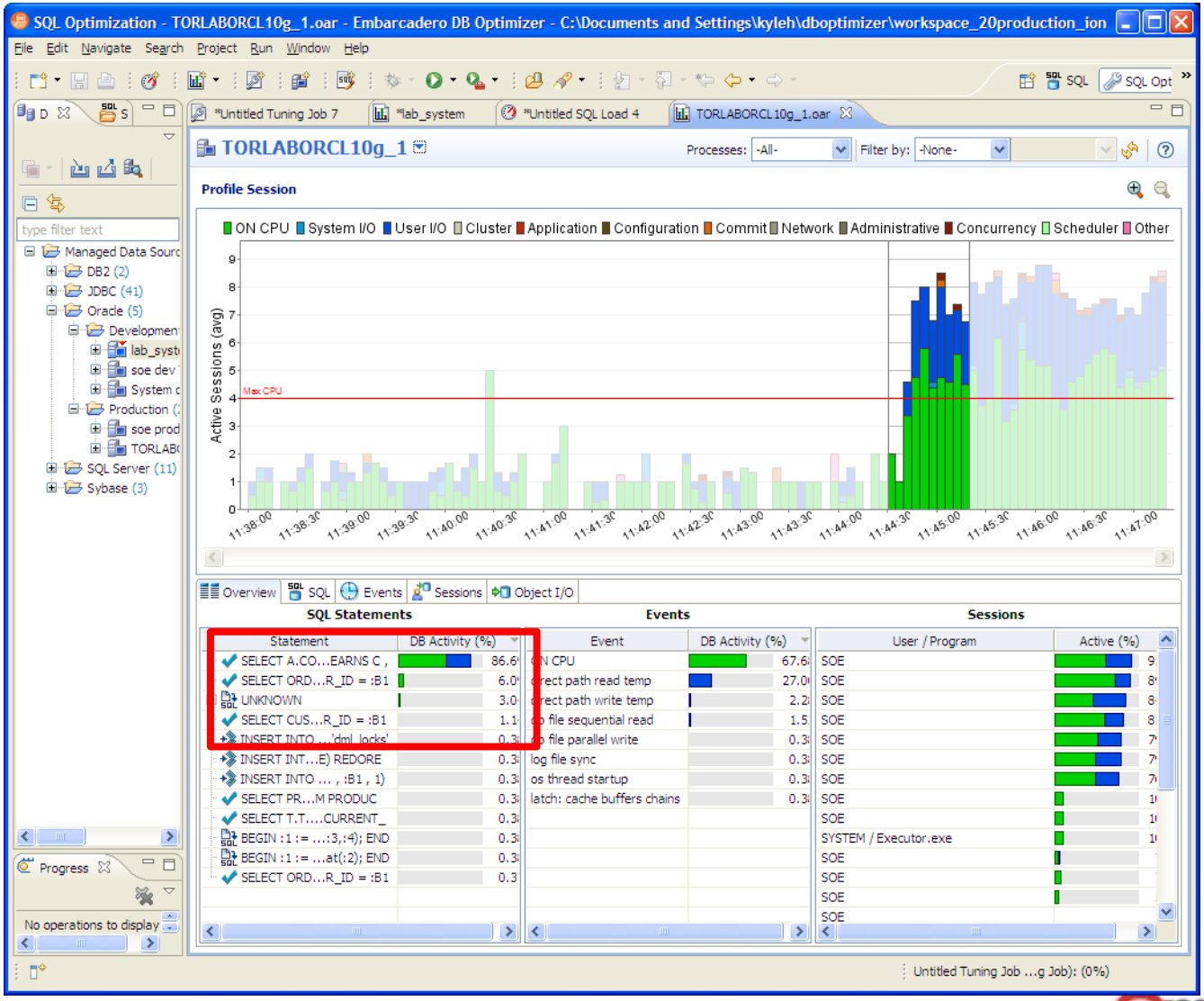
embarcadero

3. Machine Undersized



embarcadero

4. SQL needs Tuning



Tuning SQL Complexity

Trace file

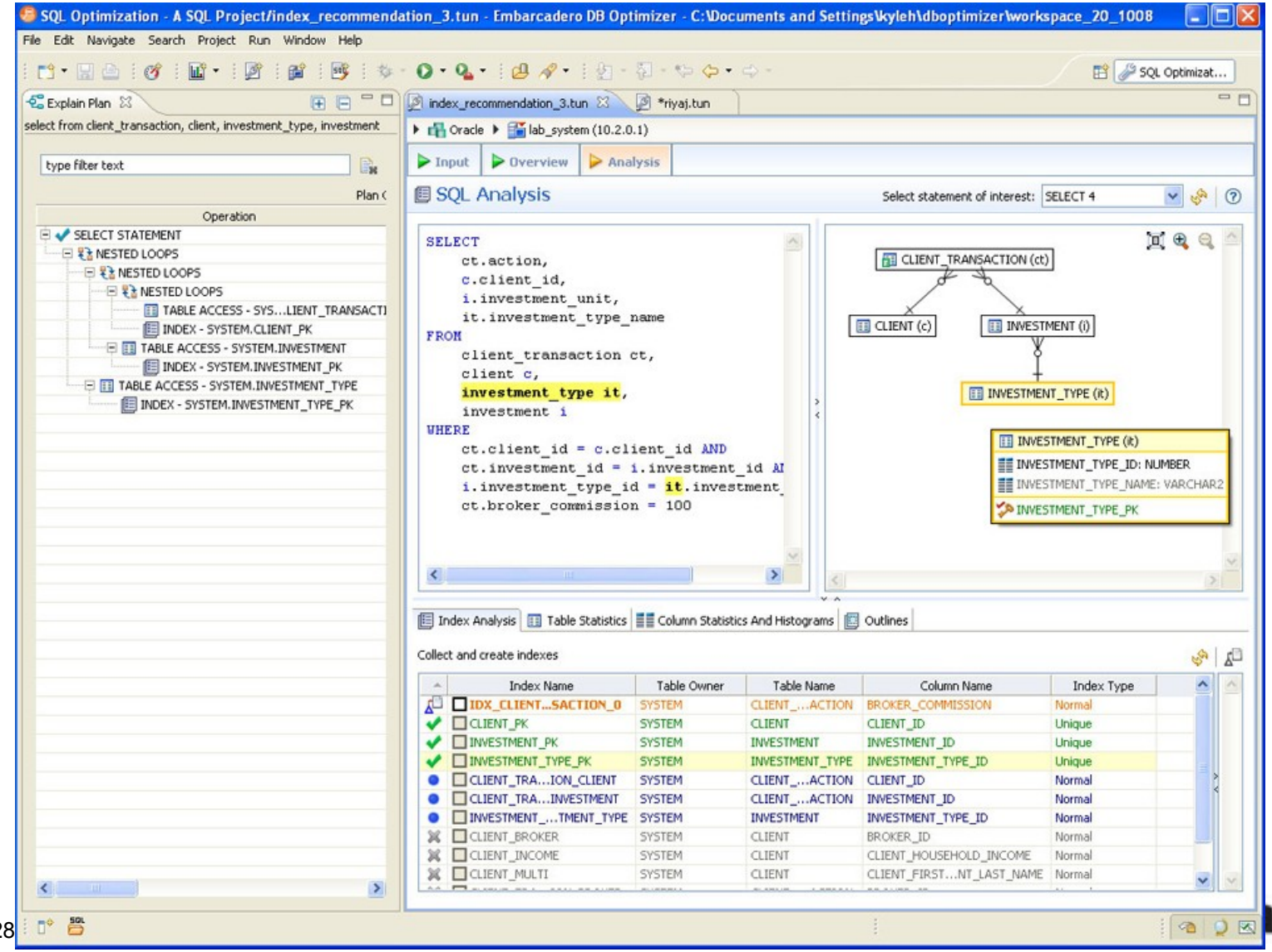
```
PARSING IN CURSOR #2 len=53 dep=0 uid=61 oct=3
lid=61 tim=1151519905950403 hv=2296704914
ad='4e50010c'
SELECT 'Hello, world; today is ' || SYSDATE FROM
END OF STMT
PARSE
#2:c=4000,e=1540,p=0,cr=0,cu=0,mis=1,r=0,dep=0,
tim=1151519905950397
BINDS #2:
EXEC
#2:c=0,e=58,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,ti
151519906034782
WAIT #2: nam='SQL*Net message to client' ela= 2
id=1650815232 #bytes=1 p3=0 obj#=-1
tim=1151519906034809
FETCH
#2:c=0,e=29,p=0,cr=0,cu=0
151519906034864
WAIT #2: nam='
driver id=16
tim=1151519
FETCH
#2:c=0,e=1,p=0,
51519906035165
WAIT #2: nam='SQ
id=1650815232 #by
tim=1151519906035
WAIT #2: nam='SQL*Net message from client' ela=
driver id=1650815232 #bytes=1 p3=0 obj#=-1
tim=1151519906035400
STAT #2 id=1 cnt=1 pid=0 pos=1 obj=0 op='FAST D
(cr=0 pr=0 pw=0 time=3 us)
```

Execution Plan

Id	Operation	Name	Starts	E-Rows	A-Rows
1	HASH GROUP BY		1	1	1
* 2	FILTER		1		1909
* 3	TABLE A		1	1	3413
4			1	165	6827
			1	165	3413
			1	165	3624
	ROWID	WB_JOB	1	242	2895
			1	233	2897
	BLE ACCESS BY INDEX ROWID	PS_PAY_CALENDAR	1	1	1
	INDEX RANGE SCAN	PS0PAY_CALENDAR	1	1	1
11	INDEX RANGE SCAN	WBBJOB_B	1	286	2895
* 12	TABLE ACCESS FULL	WB_RETROPAY_EARNS	1	27456	122K
13	TABLE ACCESS FULL	PS_RETROPAY_RQST	1	13679	13679
* 14	INDEX RANGE SCAN	PS#RETROPAYPGM_TBL	3413	1	3413
15	SORT AGGREGATE		1791	1	1791
16	FIRST ROW		1791	1	1579
* 17	INDEX RANGE SCAN (MIN/MAX)	WB_JOB_F	1791	1	1579
18	SORT AGGREGATE		1539	1	1539
19	FIRST ROW		1539	1	1539
* 20	INDEX RANGE SCAN (MIN/MAX)	WB_JOB_G	1539	1	1539

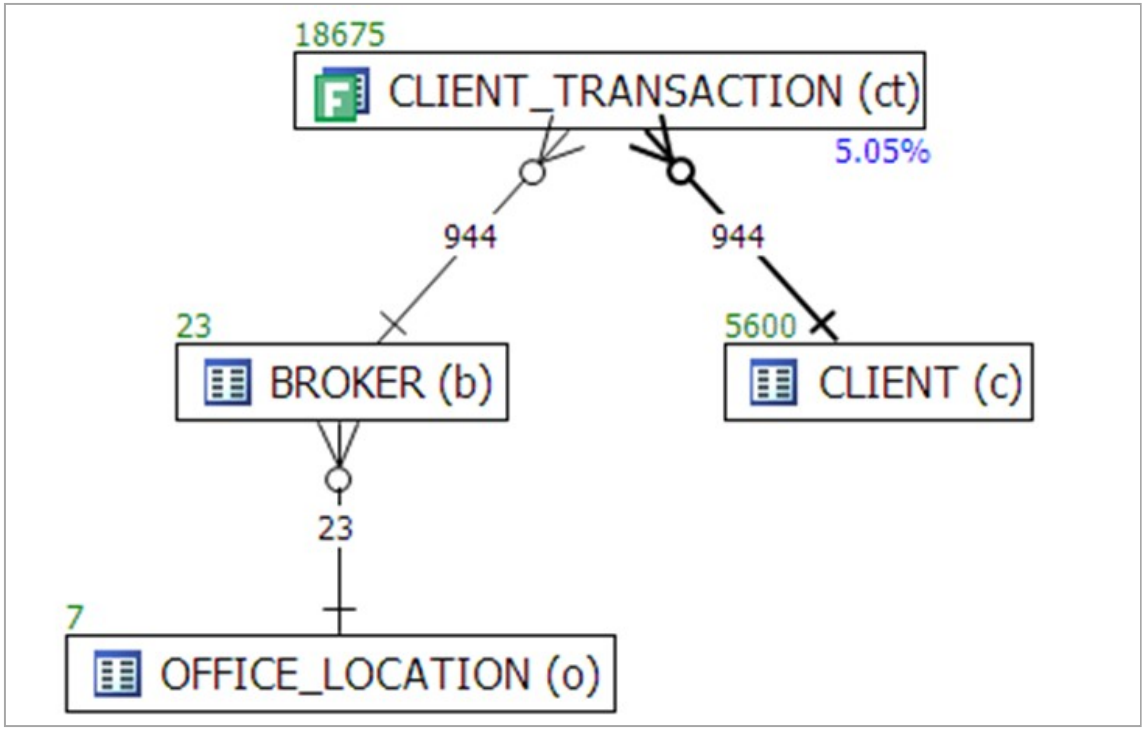
Complexity !

Visual SQL Tuning



Tuning SQL

- Indexes
  - Missing indexes
- Table Stats
  - Stale statistics
- Histograms
  - Recommendations
- Execution Plan
  - DB Optimizer automatically finds better plans
- Visual SQL Tuning diagrams



embarcadero

VST Steps

1. Tables
   - drawn as nodes
1. Joins
   - drawn as connector lines
1. Filters
   - mark on each table with filter in where clause

How to VST: Tables and Joins

```
SELECT C.Phone_Number, C.Honorific, C.First_Name, C.Last_Name,
       C.Suffix, C.Address_ID, A.Address_ID, A.Street_Address_Li
       A.Street_Address_Line2, A.City_Name, A.State_Abbreviation
       A.ZIP_Code, OD.Deferred_Shipment_Date, OD.Item_Count,
       ODT.Text, OT.Text, P.Product_Description, S.Shipment_Date
FROM Orders O, Order_Details OD, Products P, Customers C, Shipme
     Addresses A, Code_Translations ODT, Code_Translations OT
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND OD.Order_ID = O.Order_ID
  AND O.Customer_ID = C.Customer_ID
  AND OD.Product_ID = P.Product_ID(+)
  AND OD.Shipment_ID = S.Shipment_ID(+)
  AND S.Address_ID = A.Address_ID(+)
  AND O.Status_Code = OT.Code
  AND OT.Code_Type = 'ORDER_STATUS'
  AND OD.Status_Code = ODT.Code
  AND ODT.Code_Type = 'ORDER_DETAIL_STATUS'
  AND O.Order_Date > :Now - 366
ORDER BY C.Customer_ID, O.Order_ID DESC, S.Shipment_ID, OD.Orde
```

Tables

```
Orders O,
Order_Details OD,
Products P,
Customers C,
Shipments S,
Addresses A,
Code_Translations ODT,
Code_Translations OT
```

Joins

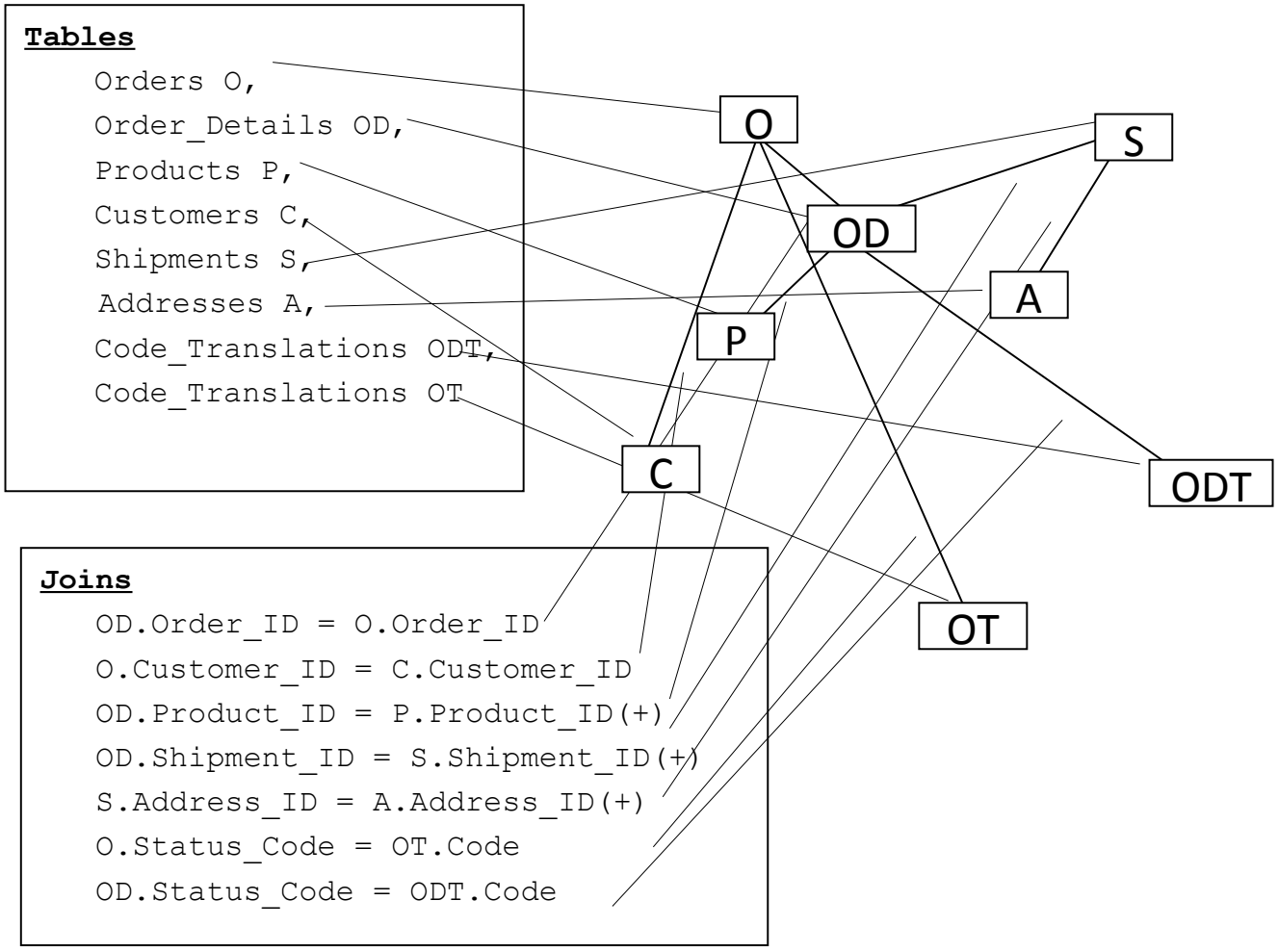
```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID =
 S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

Filters

```
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND OT.Code_Type = 'ORDER_STATUS'
  AND O.Order_Date > :Now - 366
  AND ODT.Code_Type = 'ORDER_DETAIL_STATUS'
```

Dan Tow – SQL TUNING

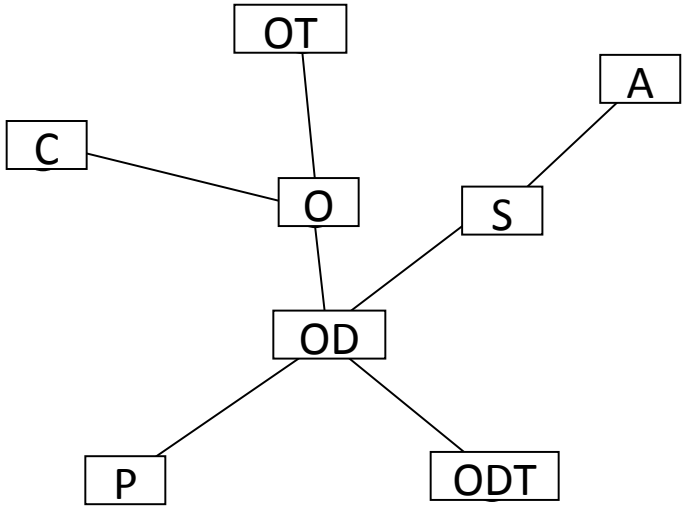
Layout tables and connections



Unstructured

Joins

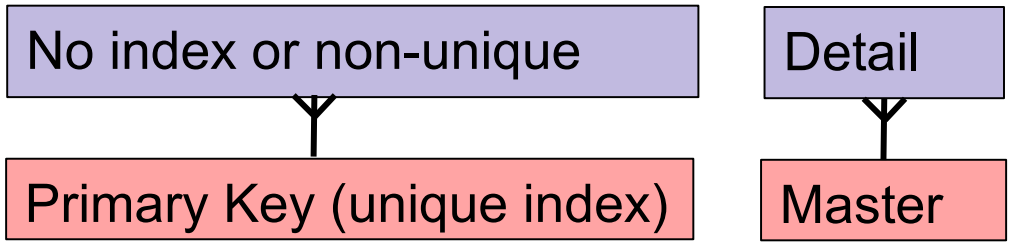
```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID = S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

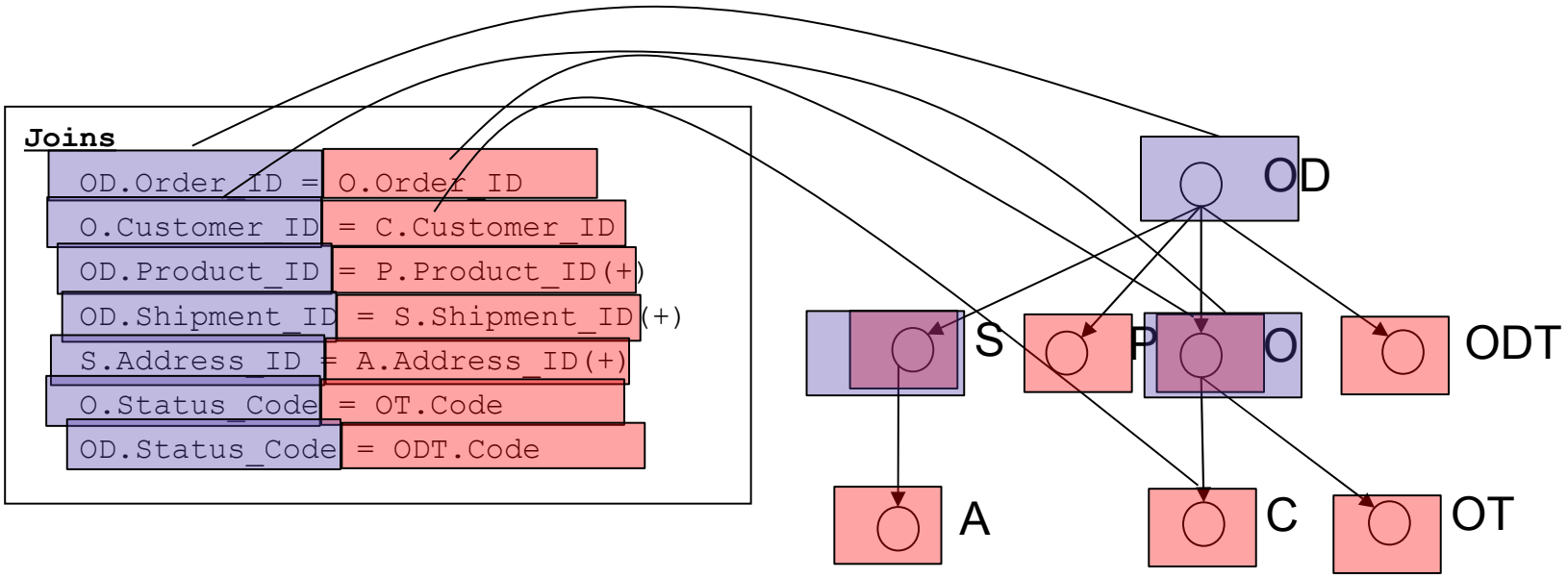


Neater, but can you do anything with it?
What's the optimal execution path?

Parents and Children

Structure the tree

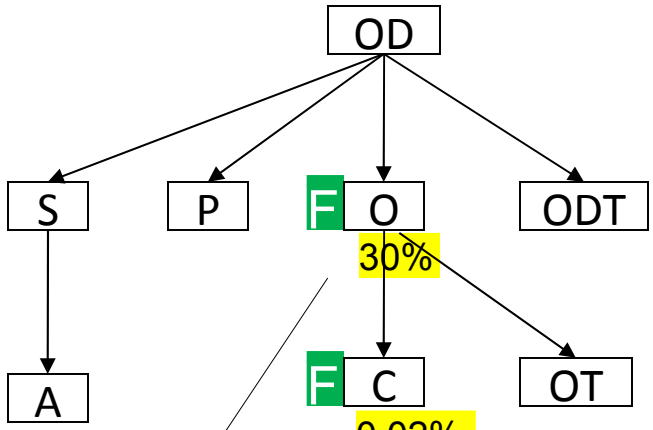




VST – filters and best path

- Filters help determine best path

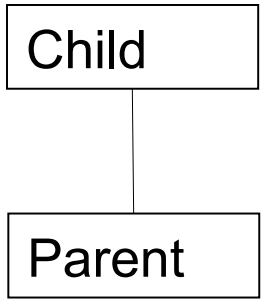
```
Filters
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND OT.Code_Type = 'ORDER_STATUS'
  AND ODT.Code_Type = 'ORDER_DETAIL_STATUS'
  AND O.Order_Date > :Now - 366
```

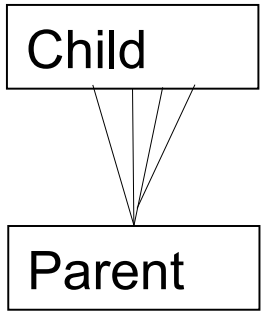



```
100% * (select count(*) from TAB where condition)
--------------------------
(select count(*) from Tab)
```

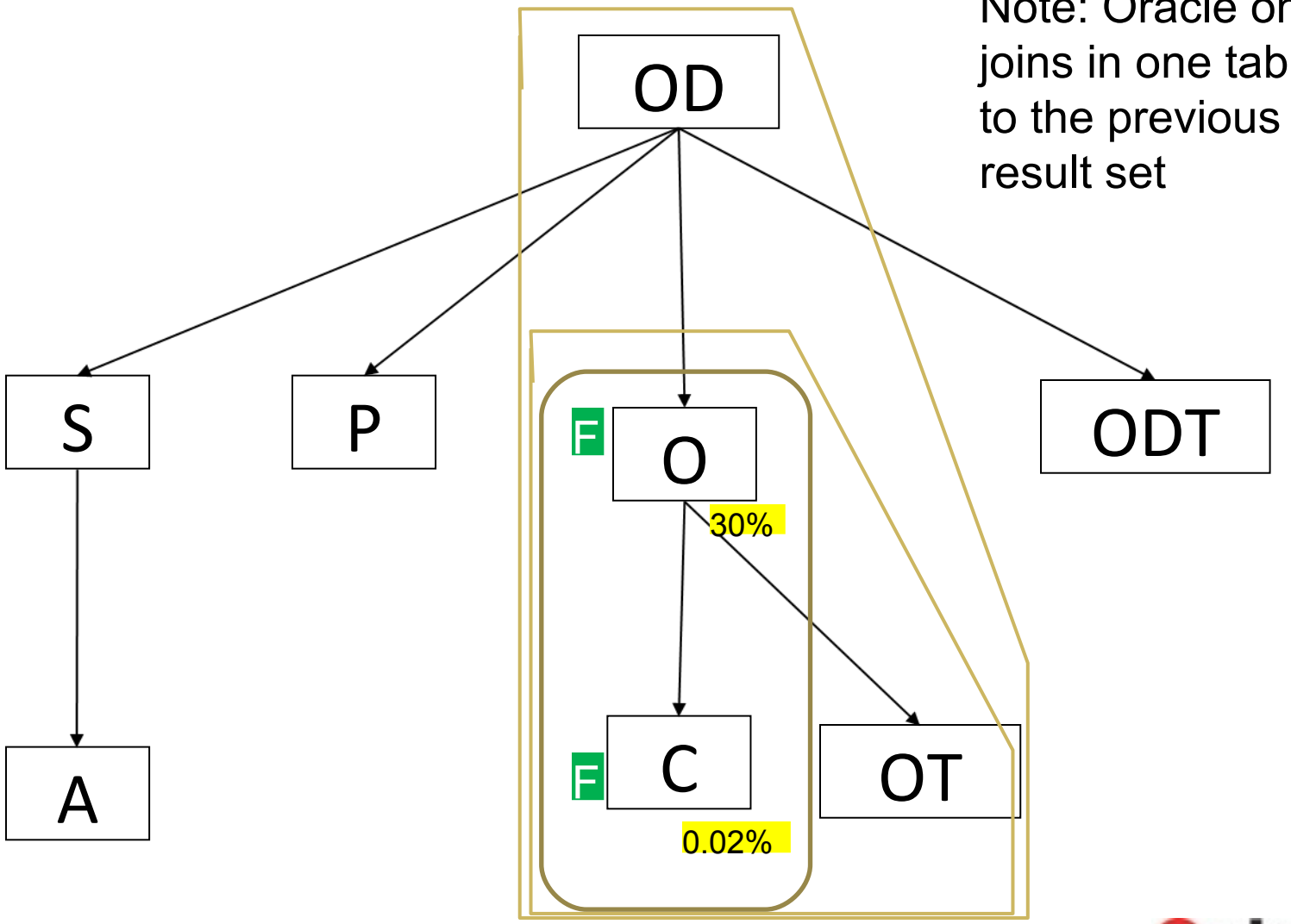
Concept:

1. Start at most selective filter
2. Join down first, before joining upwards





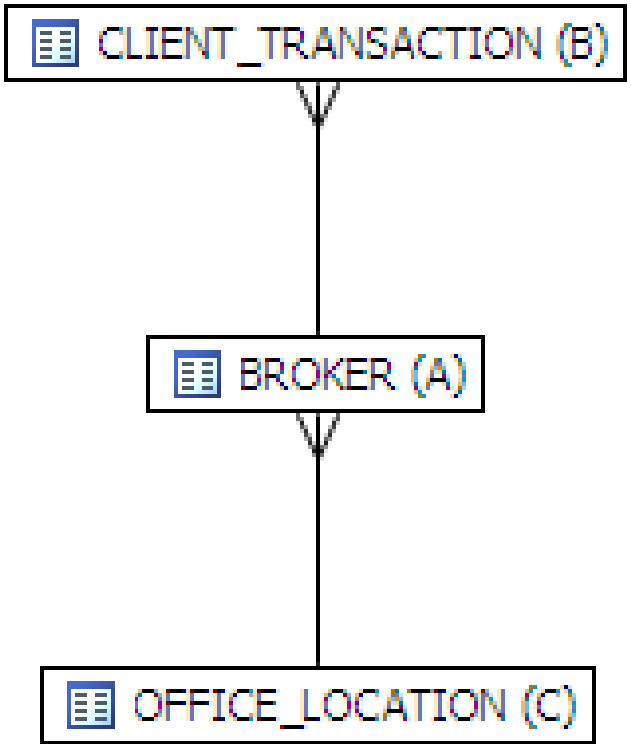
VST – best path



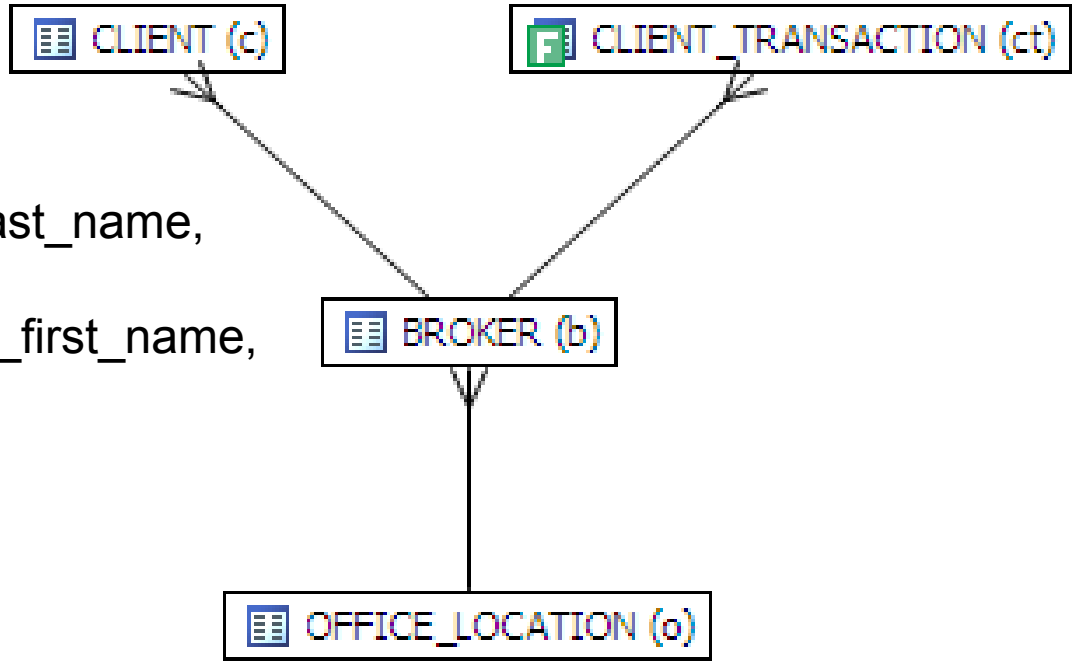
Note: Oracle only joins in one table to the previous result set

Cartesian

```
SELECT
    A.BROKER_ID BROKER_ID,
    A.BROKER_LAST_NAME BROKER_LAST_NAME,
    A.BROKER_FIRST_NAME BROKER_FIRST_NAME,
    A.YEARS_WITH_FIRM YEARS_WITH_FIRM,
    C.OFFICE_NAME OFFICE_NAME,
    SUM (B.BROKER_COMMISSION)
TOTAL_COMMISSIONS
FROM
    BROKER A,
    CLIENT_TRANSACTION B,
    OFFICE_LOCATION C,
    INVESTMENT I
WHERE
    A.BROKER_ID = B.BROKER_ID AND
    A.OFFICE_LOCATION_ID =
C.OFFICE_LOCATION_ID
GROUP BY
    A.BROKER_ID,
    A.BROKER_LAST_NAME,
    A.BROKER_FIRST_NAME,
    A.YEARS_WITH_FIRM,
    C.OFFICE_NAME;
```



Implied Cartesian




```
select
     c.client_first_name, c.client_last_name,
     ct.action, ct.price,
     b.broker_last_name, b.broker_first_name,
     o.office_name
from
      client_transaction ct,
      client c,
      broker b,
      office_location o
where
      ct.price > 100
  and b.broker_id=ct.broker_id
  and c.broker_id = b.broker_id
  and o.office_location_id = b.office_location_id
```

Diagram work for Many to One

One to many

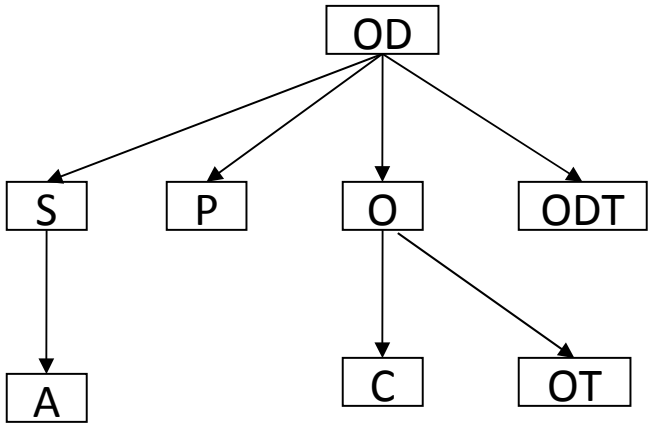
Many to many

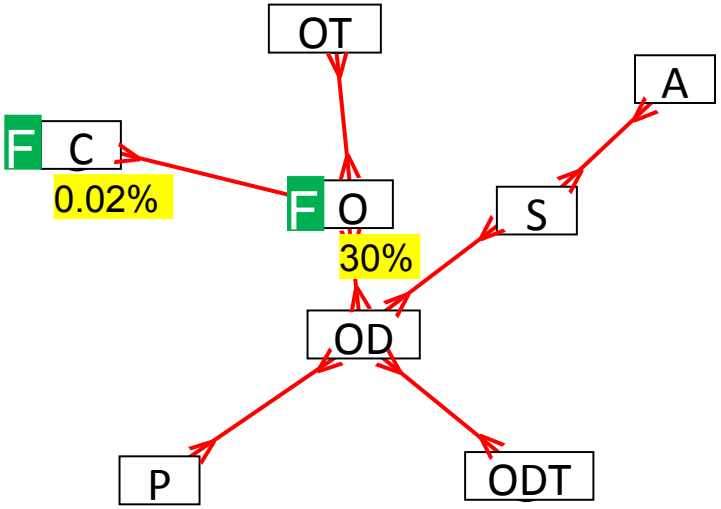
What about many to many?

Unstructured

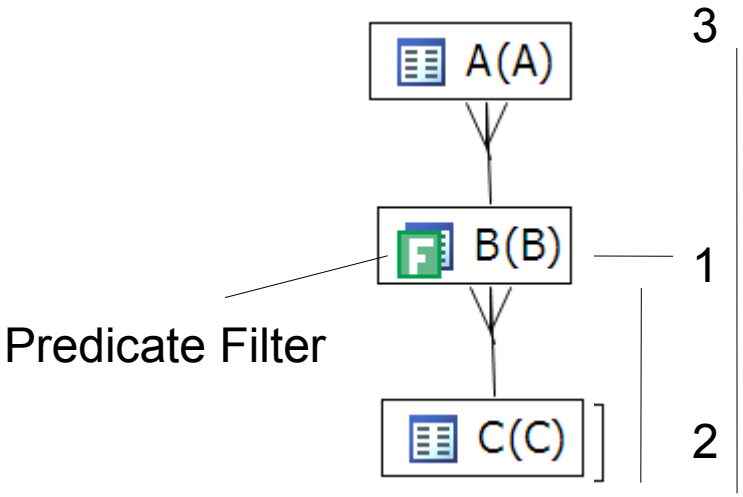
Joins

```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID = S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

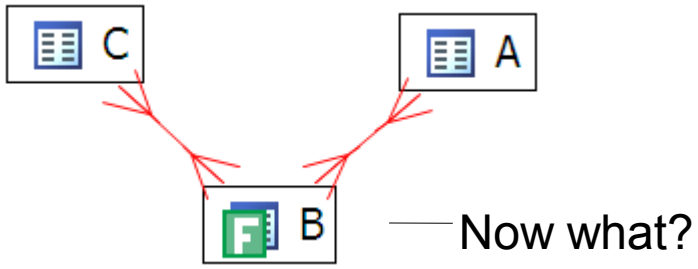




Many-to-One vs Many-to-Many



B -> C -> A

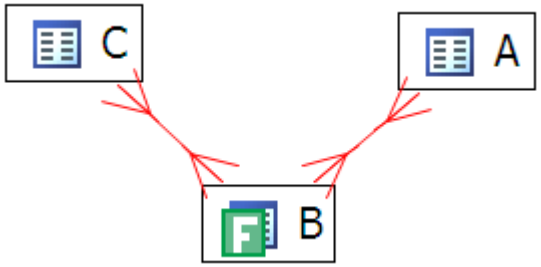


go to A or C?

Adding Constraints

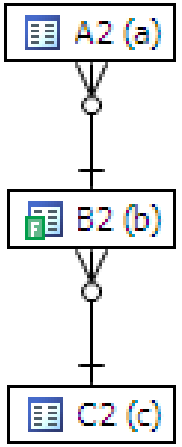
```
SELECT COUNT (*)
FROM
    b,
    c,
    a
WHERE
    b.val2 = 100 AND
    a.val1 = b.id AND
    b.val1 = c.id;
```

58 logical reads

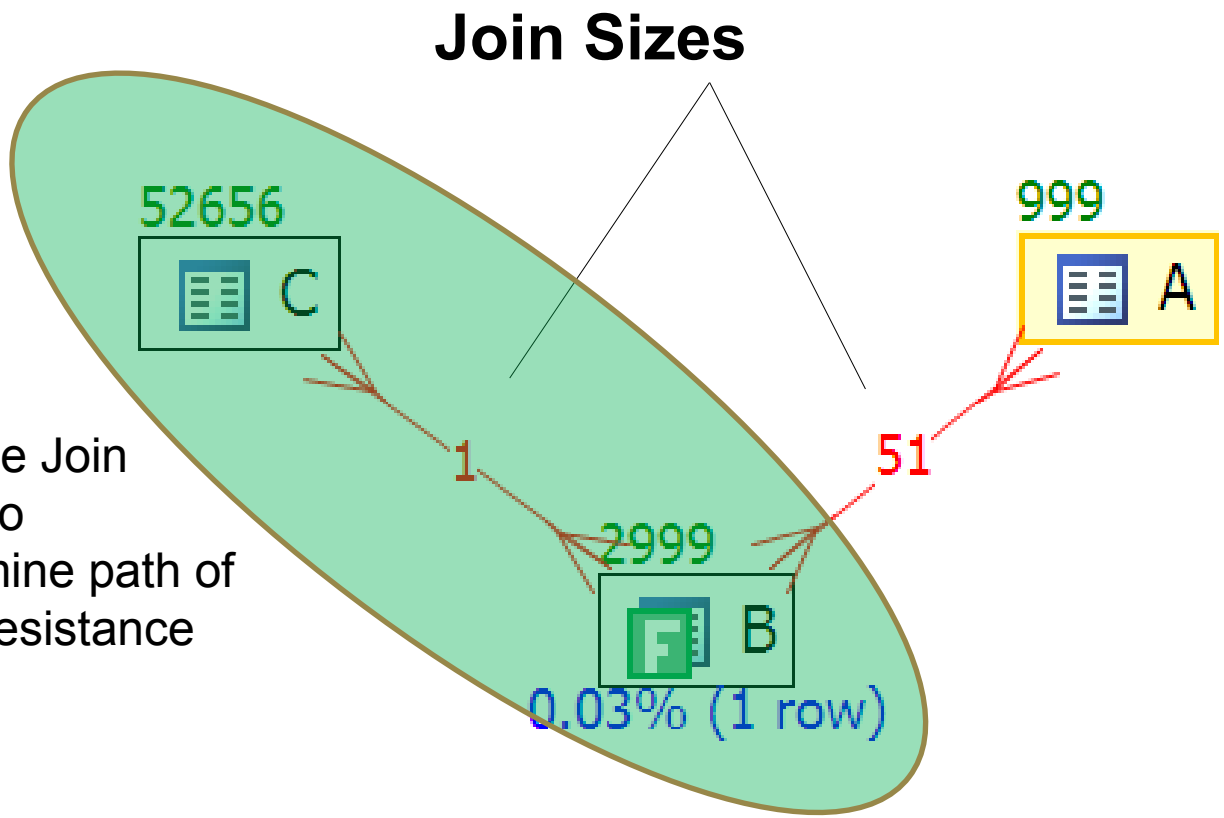



```
alter table c add constraint c_pk unique (id);
alter table b add constraint b_pk unique (id);
```

7 logical reads



Join sizes



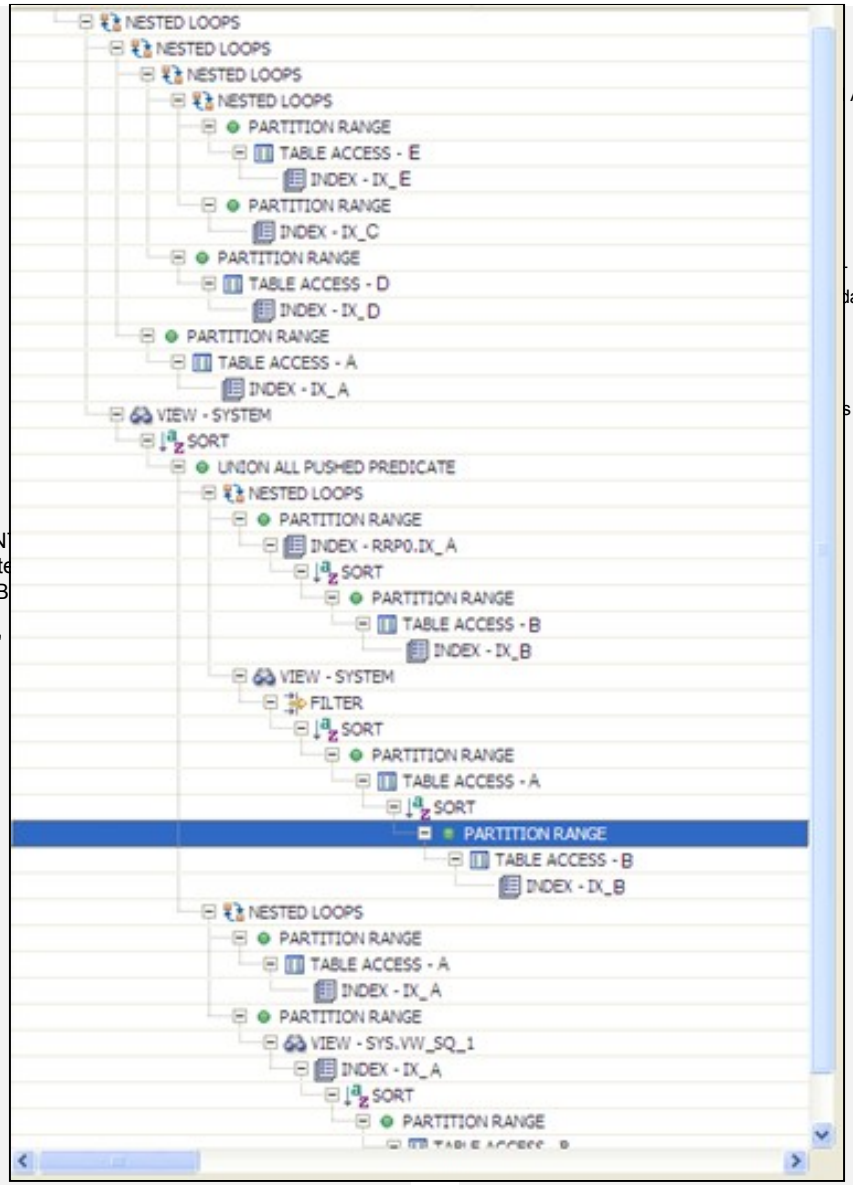
Use the Join sizes to determine path of least resistance

Look at 3 queries

- Query 1 runs more than 24 hours
- Query 2 outer joins and scalar subqueries
- Query 3 create path not available to Oracle

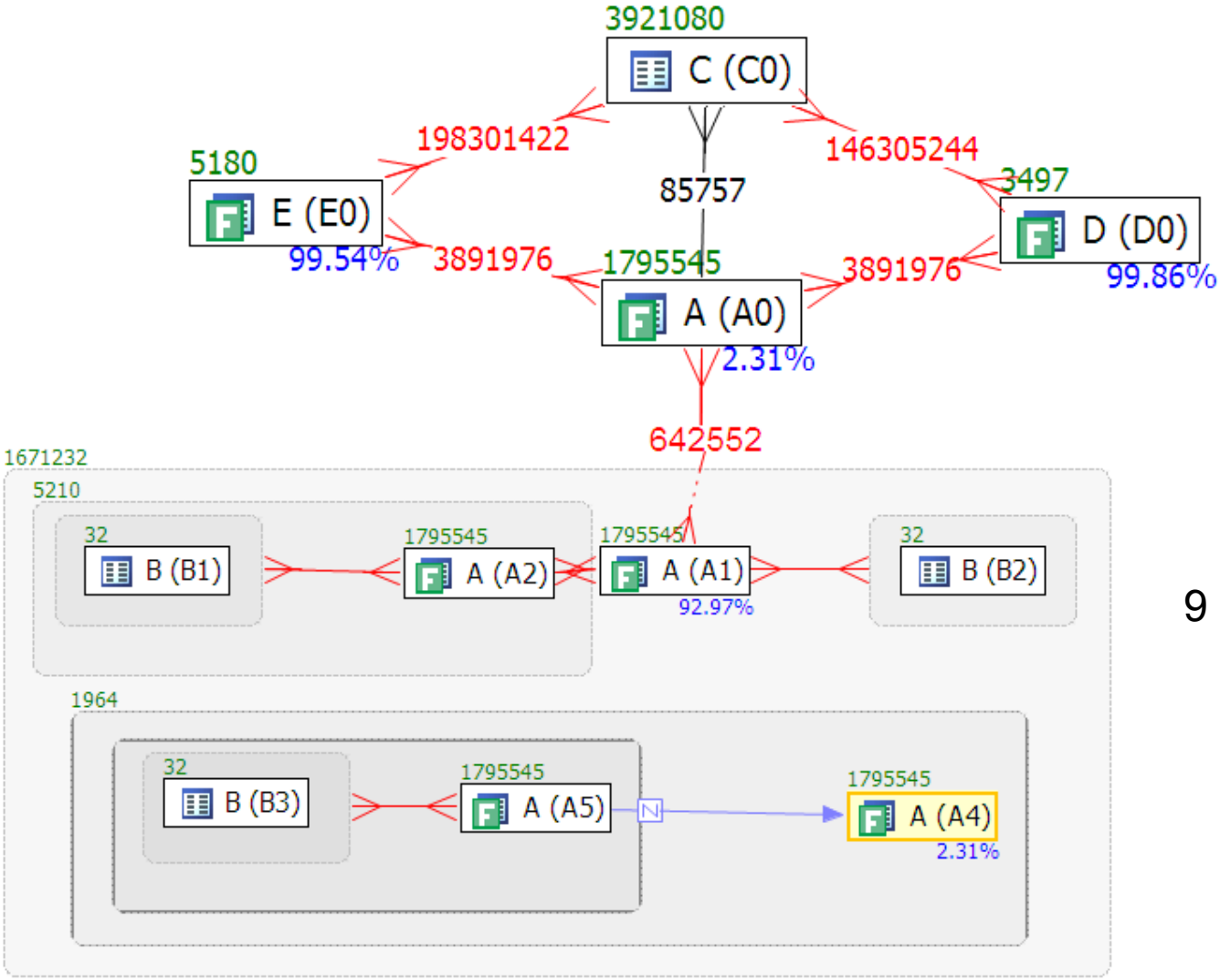
Query 1 : Over 24 hours to run

```
SELECT
   A0.zuchinis,
   A0.brocoli,
   C0.Oranges
FROM
   (
      SELECT
         A1.planted_date,
         A1.pears,
         A1.zuchinis,
         A1.brocoli
      FROM
         FOO.A A1,
         (
            SELECT
               zuchinis,
               brocoli
            FROM FOO.A A2
            WHERE
               pears = 'M' AND
               planted_date + 0 >= ADD_MON
                                   MAX (plante
                                FROM FOO.B
                                WHERE
                                   pears = 'M'
         ),
                                - 11)
            GROUP BY
               zuchinis,
               brocoli
            HAVING COUNT (*) = 12
         )
         i2
      WHERE
         A1.planted_date = (SELECT
                     MAX (planted_date)
                  FROM FOO.B B2
                  WHERE
                     pears = 'M'
      ) AND
         A1.pears = 'M' AND
         A1.zuchinis = i2.zuchinis (+) AND
         A1.brocoli = i2.brocoli (+)
 UNION
         SELECT
            A4.planted_date,
```




```
A4.planted_date <'03-OCT-08' AND

date)

s AND
```

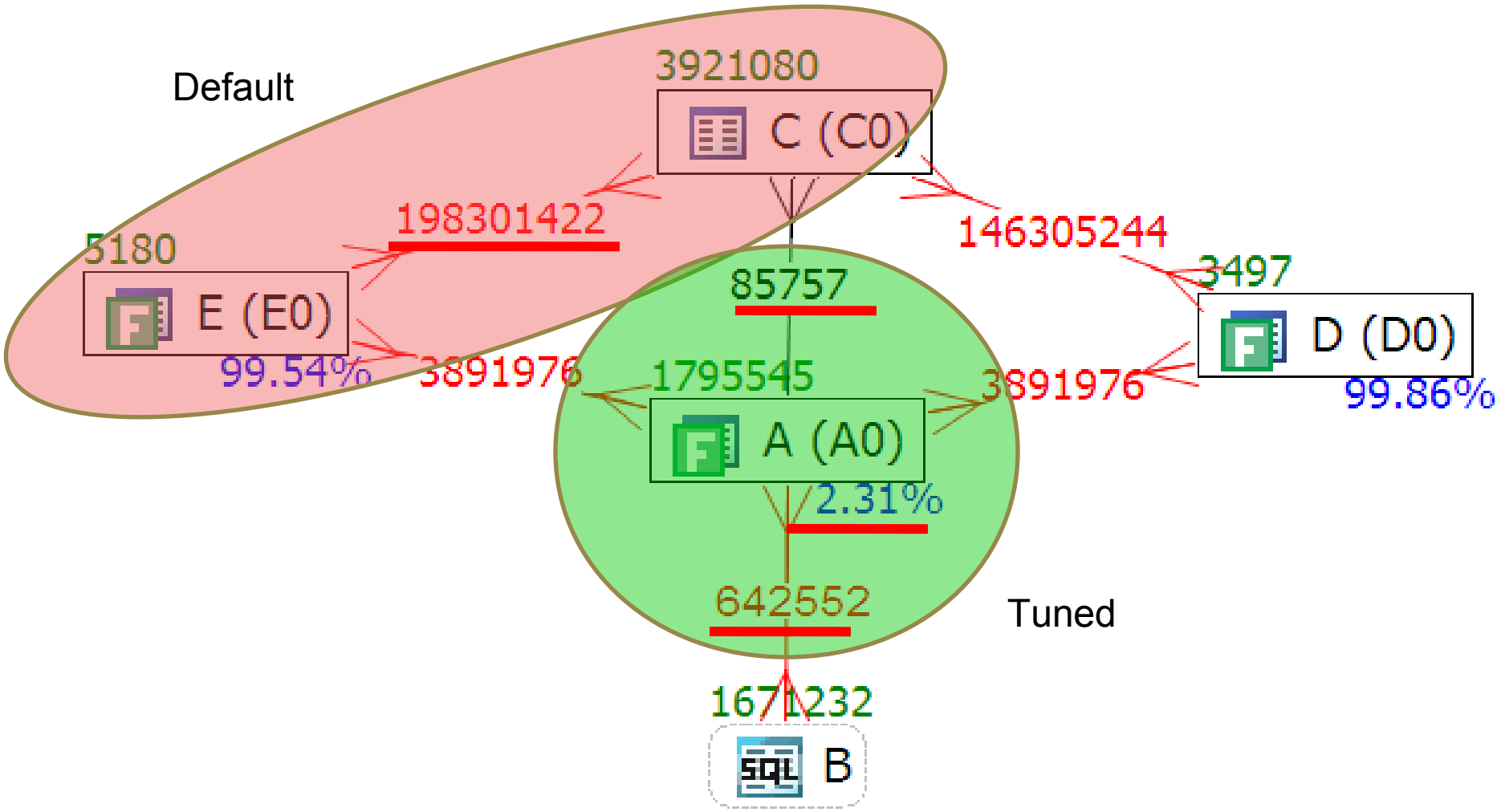
Visual SQL Diagram



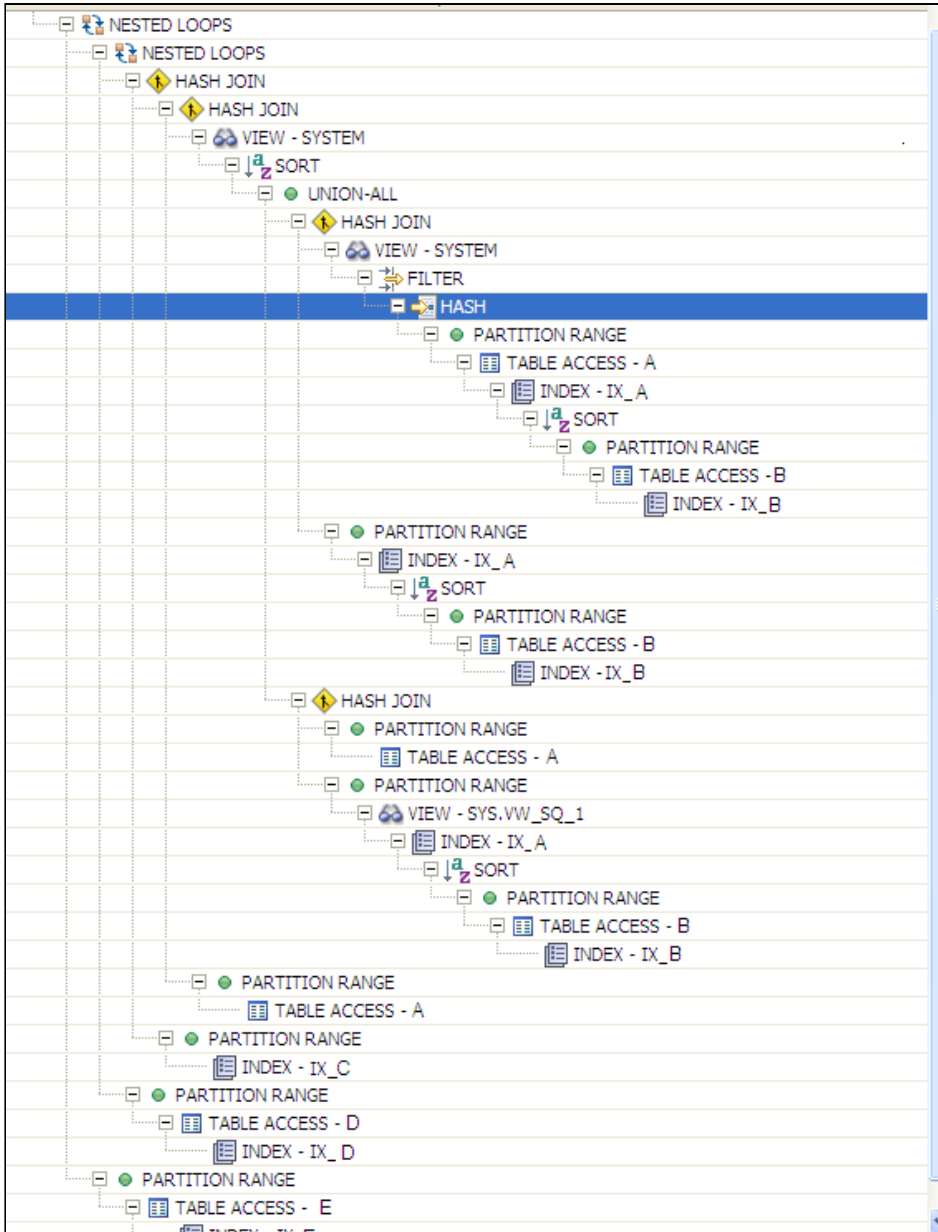
9 secs

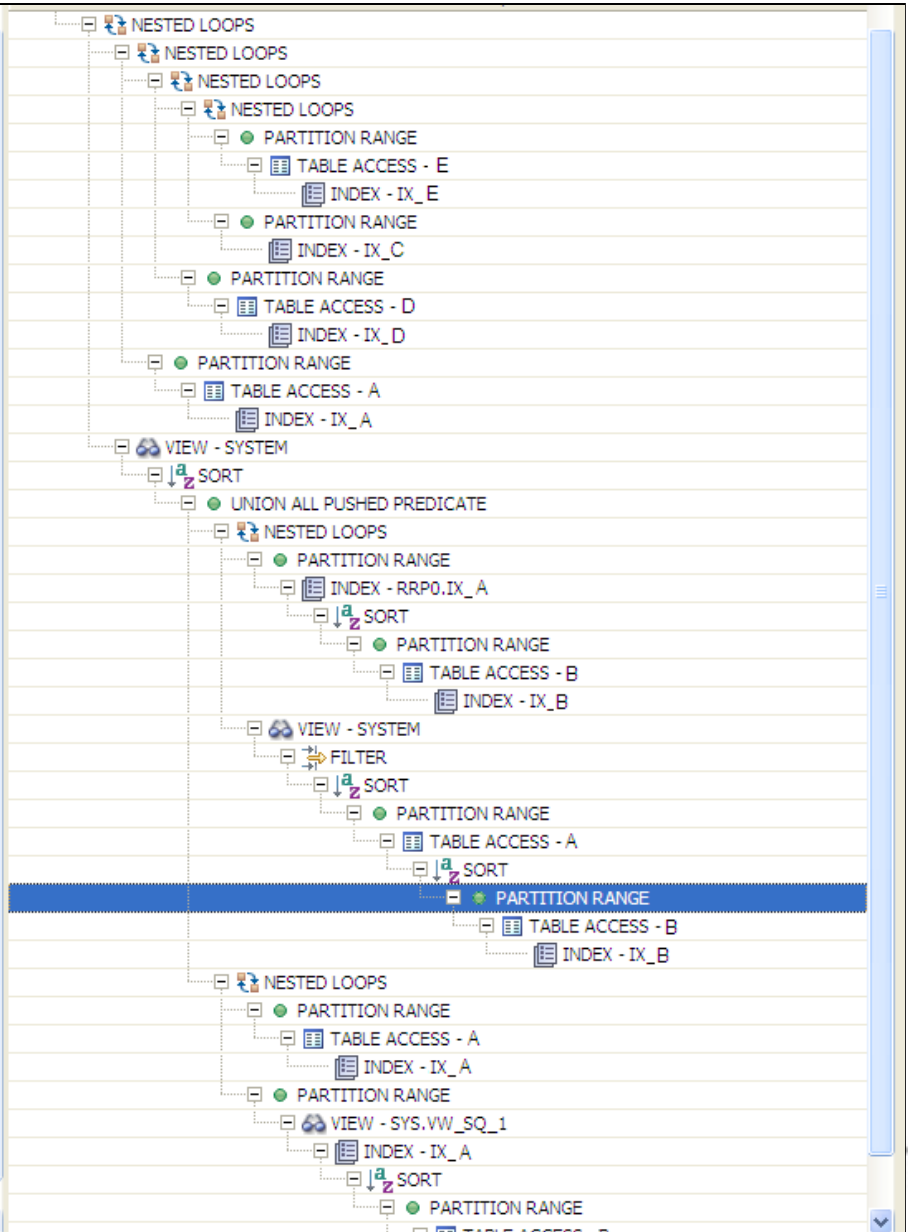
embarcadero

Default vs Tuned



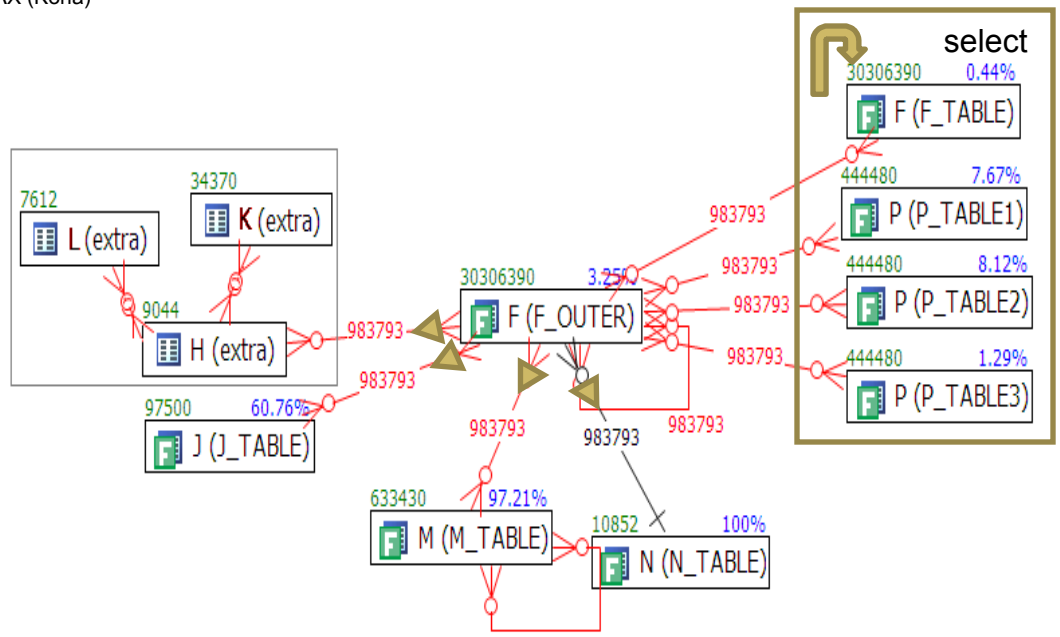
Comparing Plans : 24 hours to 5 mins





Q2

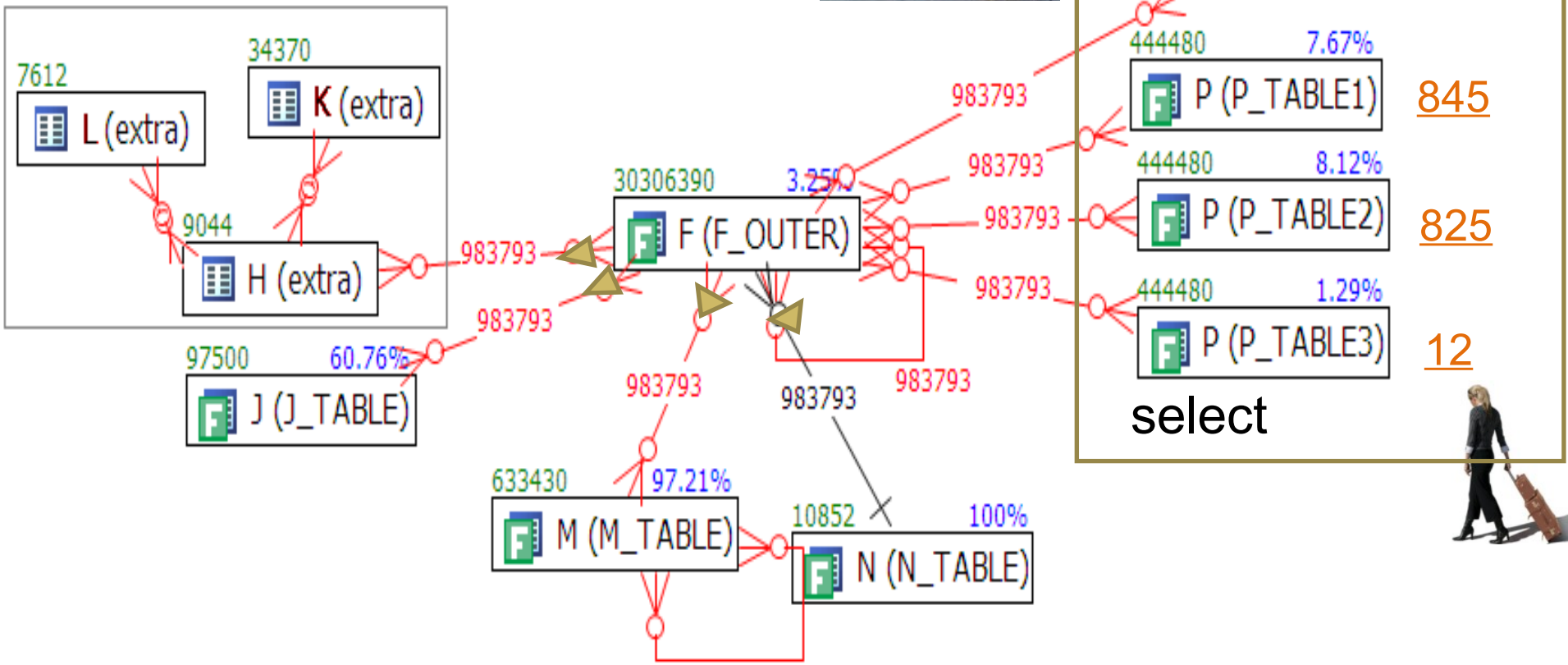
```
SELECT CASE WHEN M.NYC IS NULL THEN (SELECT /*+ qb_name(qb1) */ MAX (Kona)
                FROM foo.F
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                  AND Argentina = TRIM ('D') AND Norway = F_OUTER.Norway
            ELSE M.NYC END AS NYC,
        CASE WHEN F_OUTER.Perth IS NULL THEN NULL
            ELSE (SELECT /*+ qb_name(qb2) */ Georgia FROM foo.P
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                  AND Argentina = TRIM ('D') AND Paris = F_OUTER.Perth)
            END AS richard,
        CASE WHEN F_OUTER.Aruba IS NULL THEN NULL
            ELSE (SELECT /*+ qb_name(qb3) */ Georgia FROM foo.P
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                  AND Argentina = TRIM ('D') AND Paris = F_OUTER.Aruba)
            END AS Jody,
         CASE WHEN F_OUTER.Portland IS NULL THEN NULL
            ELSE (SELECT /*+ qb_name(qb4) */ Georgia FROM foo.P
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                  AND Argentina = TRIM ('D') AND Paris = F_OUTER.Portland)
            END AS Tom
      FROM foo.F F_OUTER, foo.M , foo.J , foo.N ,
(SELECT /*+ qb_name(qb5) */ H.SF, Oregon, H.Haiti, K.Bermuda, L.Denmark
          FROM (foo.H  LEFT OUTER JOIN foo.K
                    ON H.harvest_date = K.harvest_date
                   AND H.Argentina = K.Argentina AND H.SF = K.SF
                   AND K.Dallas = '001')
                    LEFT OUTER JOIN FOo.L
                        ON H.harvest_date = L.harvest_date
                       AND H.Argentina = L.Argentina AND H.SF = L.SF
                    WHERE H.harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                      AND H.Argentina = TRIM ('D')) extra
        WHERE F_OUTER.harvest_date = M.harvest_date(+)
          AND F_OUTER.Argentina = M.Argentina(+)
          AND F_OUTER.Norway = M.Norway(+)
          AND M.Norway(+) = M.Texas(+)
          AND F_OUTER.harvest_date = to_date('08/10/2008','dd/mm/yyyy')
          AND F_OUTER.Argentina = TRIM ('D')
          AND M.harvest_date(+) = to_date('08/10/2008','dd/mm/yyyy')
          AND M.Argentina(+) = TRIM ('D')
          AND F_OUTER.Norway = F_OUTER.Hawaii
          AND F_OUTER.harvest_date = J.harvest_date(+)
          AND F_OUTER.Argentina = J.Argentina(+)
          AND F_OUTER.Norway = J.Texas(+)
          AND J.harvest_date(+) = to_date('08/10/2008','dd/mm/yyyy')
          AND J.Argentina(+) = TRIM ('D')
          AND F_OUTER.Iraq = extra.SF(+)
          AND F_OUTER.harvest_date = N.harvest_date(+)
```



Two important qualities:

- All outer joins to F_OUTER
- 4 subqueries in select

Q2


30306390
0.44%
F (F_TABLE)
682348
444480
7.67%
P (P_TABLE1)
845
444480
8.12%
P (P_TABLE2)
825
444480
1.29%
P (P_TABLE3)
12
select
7612
L (extra)
34370
K (extra)
9044
H (extra)
97500
60.76%
J (J_TABLE)
30306390
3.25%
F (F_OUTER)
633430
97.21%
M (M_TABLE)
10852
100%
N (N_TABLE)
983793

Q2

The subqueries in the select clause look like

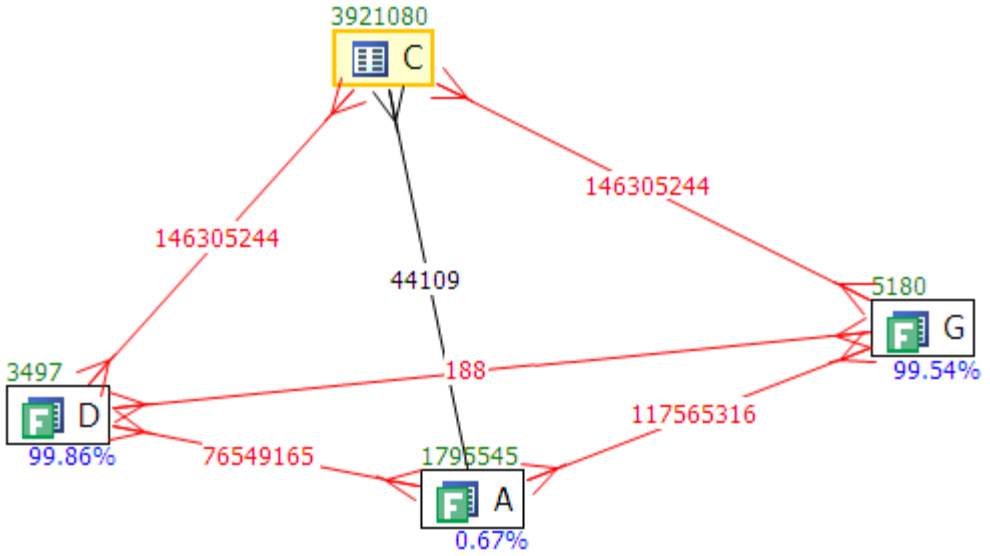
```
select CASE WHEN F.f1 IS NULL
            THEN NULL
            ELSE (SELECT X.f2
                  FROM X
                  WHERE code_vl = F.f2)
            END AS f0
from F;
```

and should be merged into the query like:

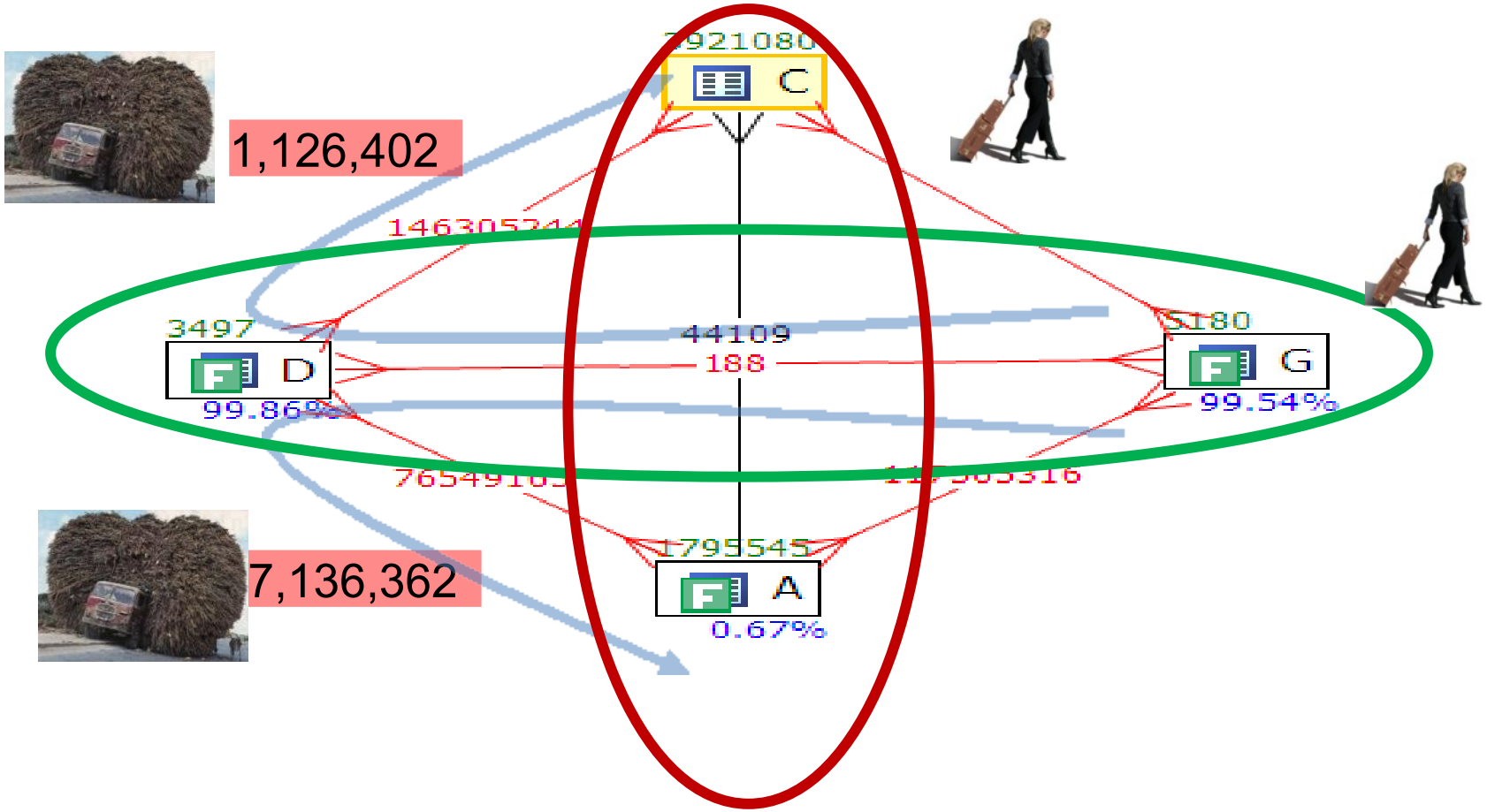
```
 select CASE WHEN F.f1 IS NULL
             THEN NULL
             ELSE ( X.f2)
             END AS f0
 from F , X
where code_vl(+) = F.f1;
```

Q3

```
SELECT DISTINCT *
FROM
   FOO.a a, FOO.c c, FOO.d d, FOO.g g
WHERE
   a.planted_date > '01-OCT-08' AND
   a.planted_date < '03-OCT-08' AND
   a.pears = 'D' AND a.green_beans = '1' AND
   a.planted_date = c.planted_date AND
   a.pears = c.pears AND
   a.zuchinis = c.zuchinis AND
   a.brocoli = c.brocoli AND
   a.planted_date = d.planted_date AND
   a.pears = d.pears AND
   a.harvest_size = d.harvest_size AND
   c.oranges = d.oranges AND
   c.apples = d.apples AND
   (d.lemons = 0 OR d.lemons IS NULL) AND
   a.planted_date = g.planted_date AND
   a.pears = g.pears AND
   a.harvest_size = g.harvest_size AND
   c.oranges = g.oranges AND
   c.apples = g.apples AND
   (g.lemons = 0 OR g.lemons IS NULL) AND
   a.zuchinis = '0236' AND
   d.apples = g.apples AND
   d.oranges = g.oranges
ORDER BY a.zuchinis, a.brocoli;
```

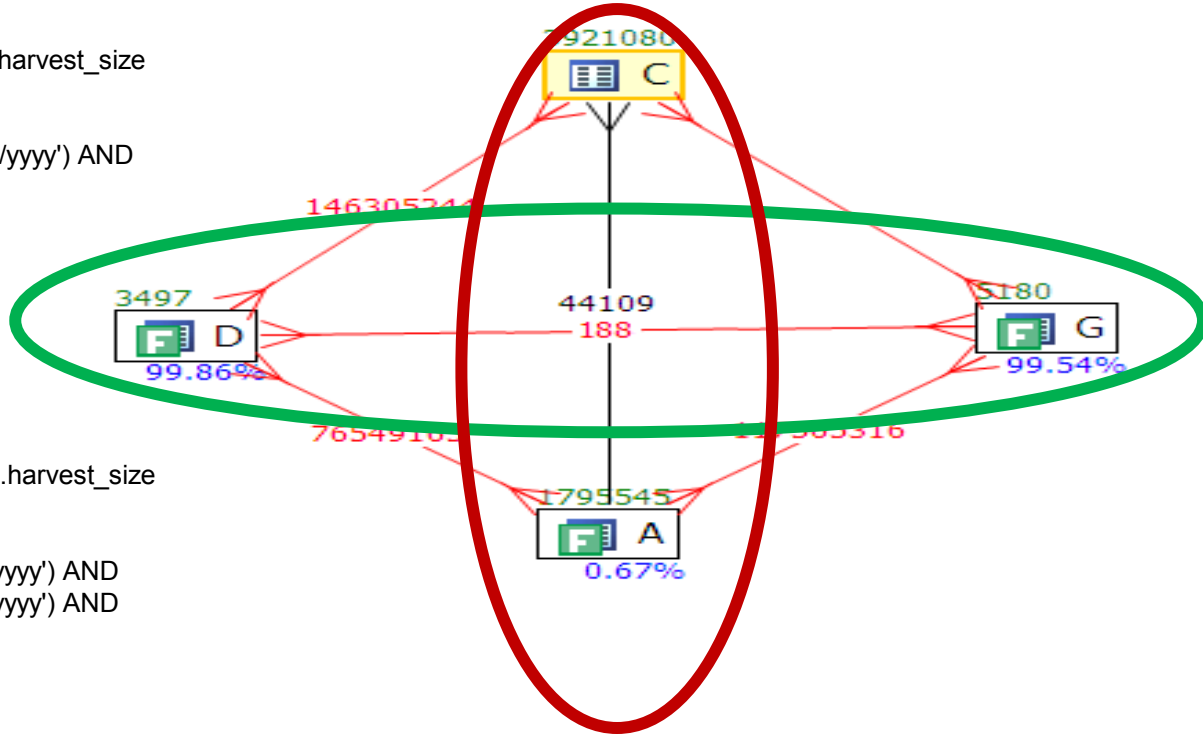


Q3: Transitivity



Q3

```
SELECT * FROM
(
   SELECT /*+  NO_MERGE */ c.apples, c.oranges, a.harvest_size
   FROM a,   c
   WHERE
      a.planted_date = TO_DATE ('02/10/2008', 'dd/mm/yyyy') AND
      a.pears = 'D' AND
      a.green_beans = '1' AND
      a.planted_date = c.planted_date AND
      a.pears = c.pears AND
      a.zuchinis = c.zuchinis AND
      a.brocoli = c.brocoli AND
      a.zuchinis = '0236'
) X,
(
    SELECT /*+  NO_MERGE */ d.apples, d.oranges, d.harvest_size
    FROM  d,  g
    WHERE
    d.planted_date = TO_DATE ('02/10/2008', 'dd/mm/yyyy') AND
    g.planted_date = TO_DATE ('02/10/2008', 'dd/mm/yyyy') AND
    g.apples = d.apples AND
    d.oranges = g.oranges AND
    d.pears = 'D' AND
    g.pears = 'D' AND
    g.pears = d.pears AND
    g.harvest_size = d.harvest_size AND
    (d.lemons = 0 OR d.lemons IS NULL) AND
    (g.lemons = 0 OR  g.lemons IS NULL)
) Y
WHERE
X.oranges = Y.oranges AND
X.apples = Y.apples AND
X.harvest_size = Y.harvest_size;
```



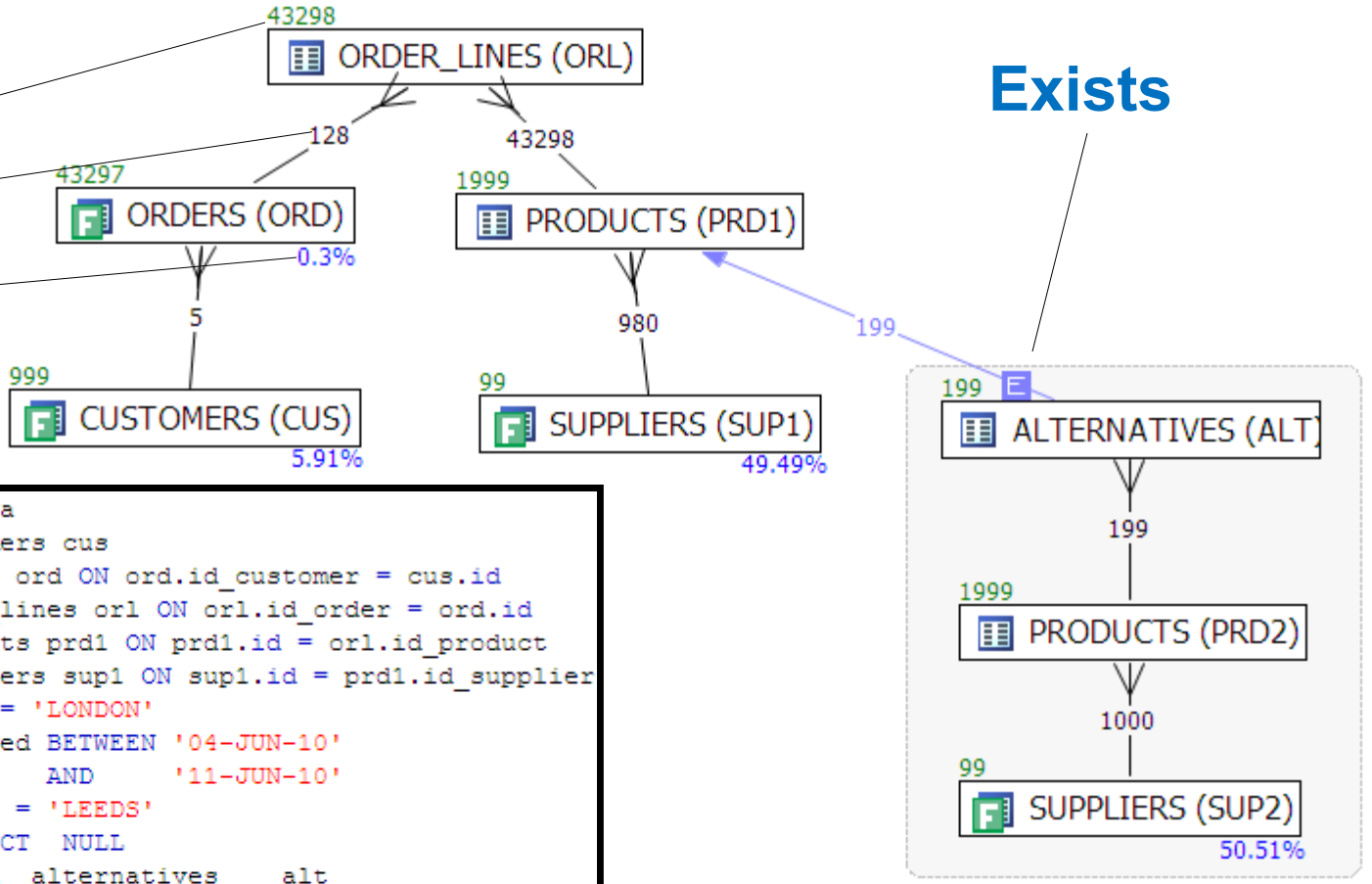
This final version runs in
elapsed **0.33 secs** and 12K logical reads
down from an original
elapsed **4.5 secs** and 1M logical reads

What tools to use to create VST diagrams?

- Paper
  - Modifications messy and difficult
- PowerPoint
  - Can move things around
  - Sticky connectors
  - Easy to modify
- DB Optimizer
  - Automatic and still modifiable

Visual SQL Tuning (VST)

Table Sizes
Join Sizes
Filter Ratios

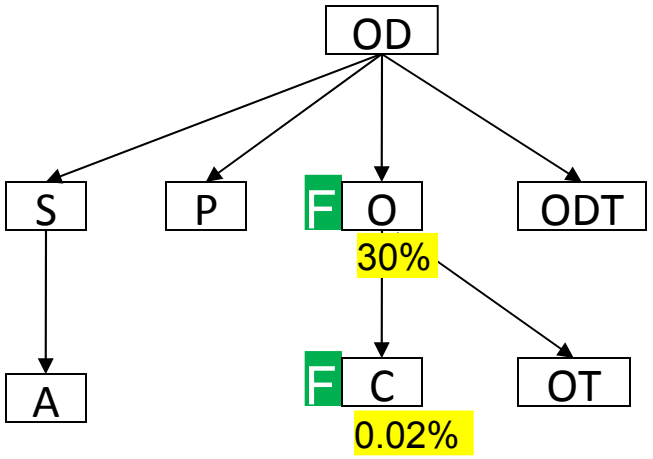



```
SELECT order_line_data
FROM           customers cus
    INNER JOIN orders ord ON ord.id_customer = cus.id
    INNER JOIN order_lines orl ON orl.id_order = ord.id
    INNER JOIN products prd1 ON prd1.id = orl.id_product
    INNER JOIN suppliers sup1 ON sup1.id = prd1.id_supplier
WHERE   cus.location = 'LONDON'
    AND ord.date_placed BETWEEN '04-JUN-10'
                        AND     '11-JUN-10'
    AND sup1.location = 'LEEDS'
    AND EXISTS ( SELECT  NULL
                 FROM  alternatives    alt
                       INNER JOIN     products prd2
                         ON prd2.id = alt.id_product_sub
                       INNER JOIN     suppliers sup2
                         ON sup2.id = prd2.id_supplier
                 WHERE    alt.id_product = prd1.id
                       AND sup2.location != 'LEEDS' )
```


embarcadero

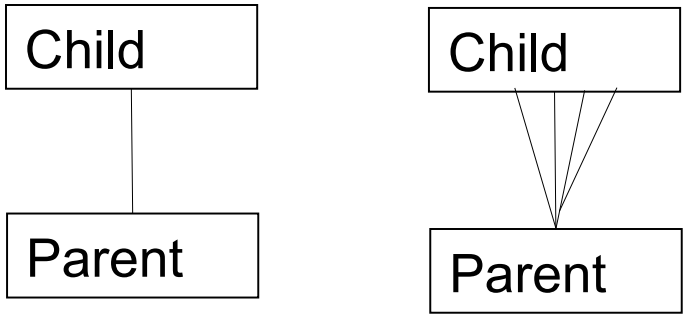
VST Steps Summary

1. Diagram tables
2. Draw connectors for each join
3. Calculate filter ratios
4. Find the table sizes
5. Calculate two table join sizes

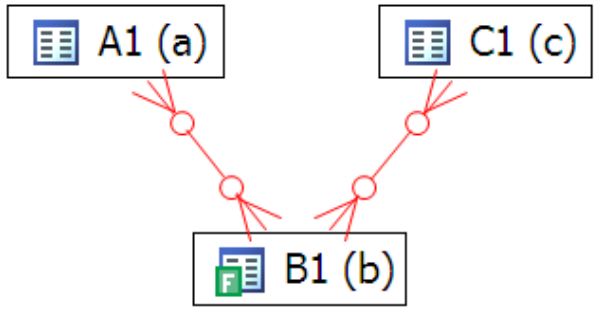


Execution Path

- Start at the most selective join filter
- Join to keep the running result set size small



Many to Many relationships = problems



Do You Want?

Engineering Data?

HISTORY OF O-RING DAMAGE ON SRM FIELD JOINTS

	SRM No.	Cross Sectional View: Erosion Depth (in.)	Cross Sectional View: Perimeter Affected (deg)	Cross Sectional View: Nominal Dia. (in.)	Top View: Length Of Max Erosion (in.)	Top View: Total Heat Affected Length (in.)	Clocking Location (deg)
61A LH Center Field**	22A	None	None	0.280	None	None	36° --66°
61A LH ~~CENTER~~ FIELD**	22A	NONE	NONE	0.280	NONE	NONE	338°-18°
51C LH Forward Field**	15A	0.010	154.0	0.280	4.25	5.25	163
51C RH Center Field (prim)***	15B	0.038	130.0	0.280	12.50	58.75	354
51C RH Center Field (sec)***	15B	None	45.0	0.280	None	29.50	354
41D RH Forward Field	13B	0.028	110.0	0.280	3.00	None	275
41C LH Aft Field*	11A	None	None	0.280	None	None	--
41B LH Forward Field	10A	0.040	217.0	0.280	3.00	14.50	351
STS-2 RH Aft Field	2B	0.053	116.0	0.280	--	--	90

*Hot gas path detected in putty. Indication of heat on O-ring, but no damage.
**Soot behind primary O-ring.
***Soot behind primary O-ring, heat affected secondary O-ring.

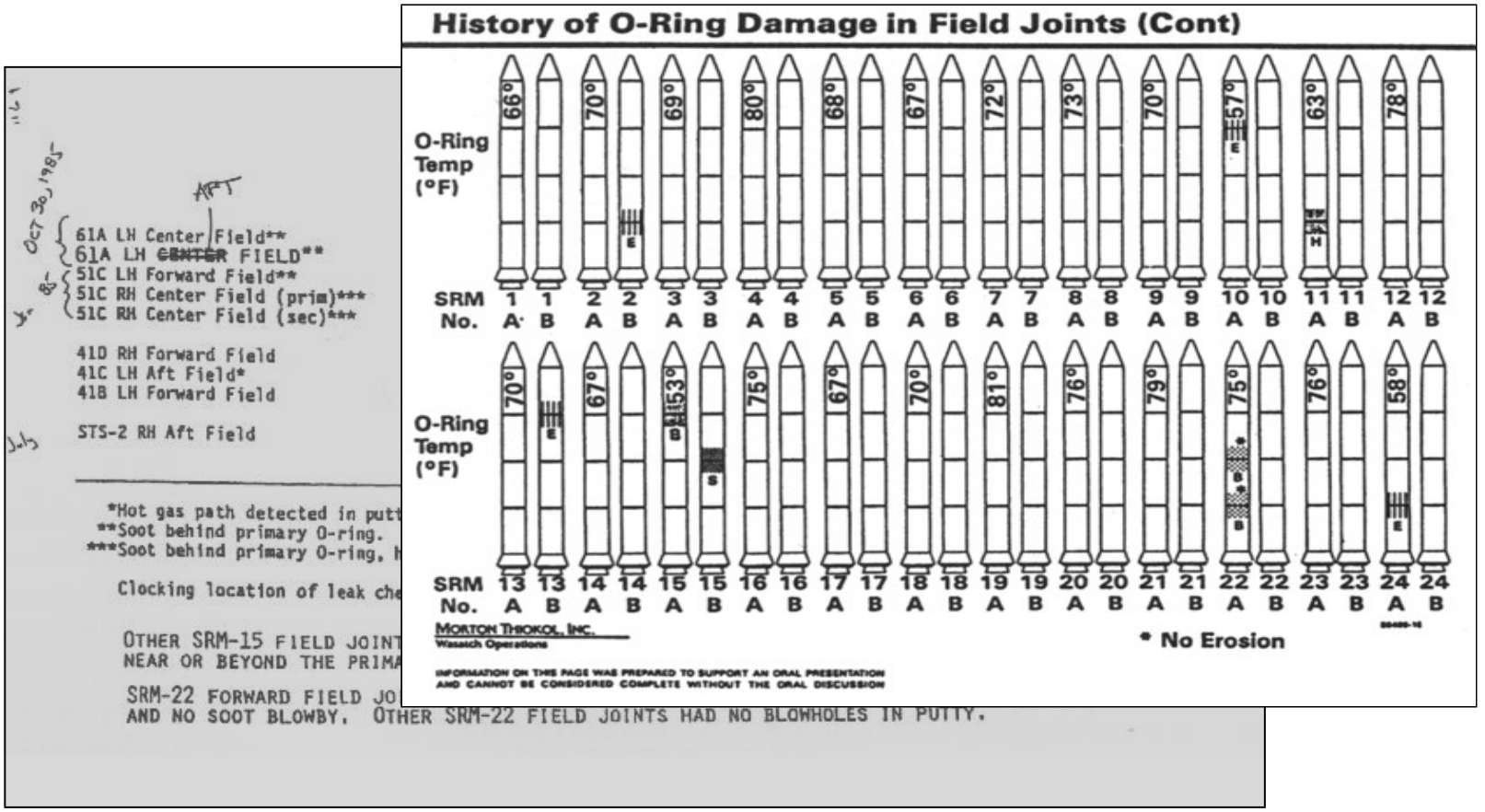
Clocking location of leak check port - 0 deg.

OTHER SRM-15 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY AND NO SOOT NEAR OR BEYOND THE PRIMARY O-RING.

SRM-22 FORWARD FIELD JOINT HAD PUTTY PATH TO PRIMARY O-RING, BUT NO O-RING EROSION AND NO SOOT BLOWBY. OTHER SRM-22 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY.

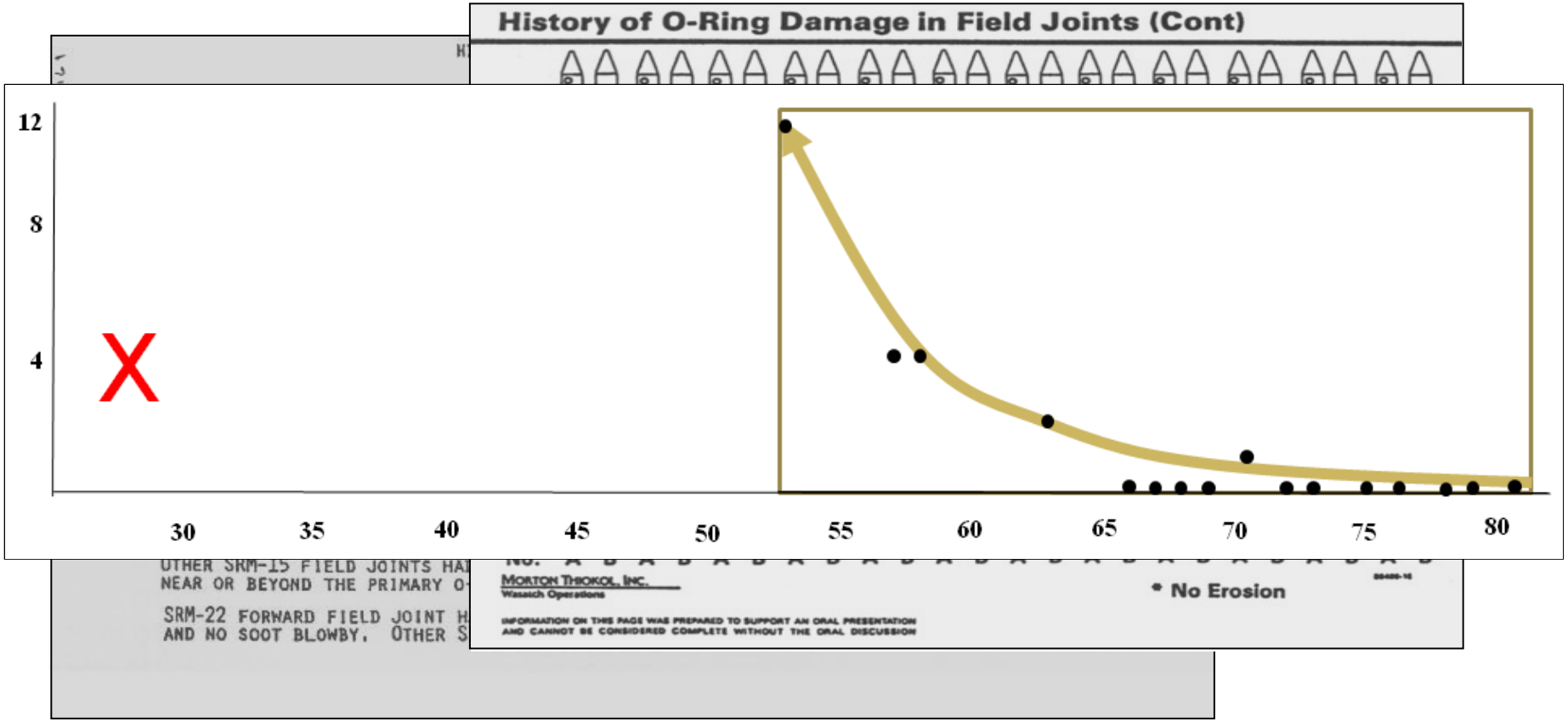
Do You Want?

Pretty Pictures

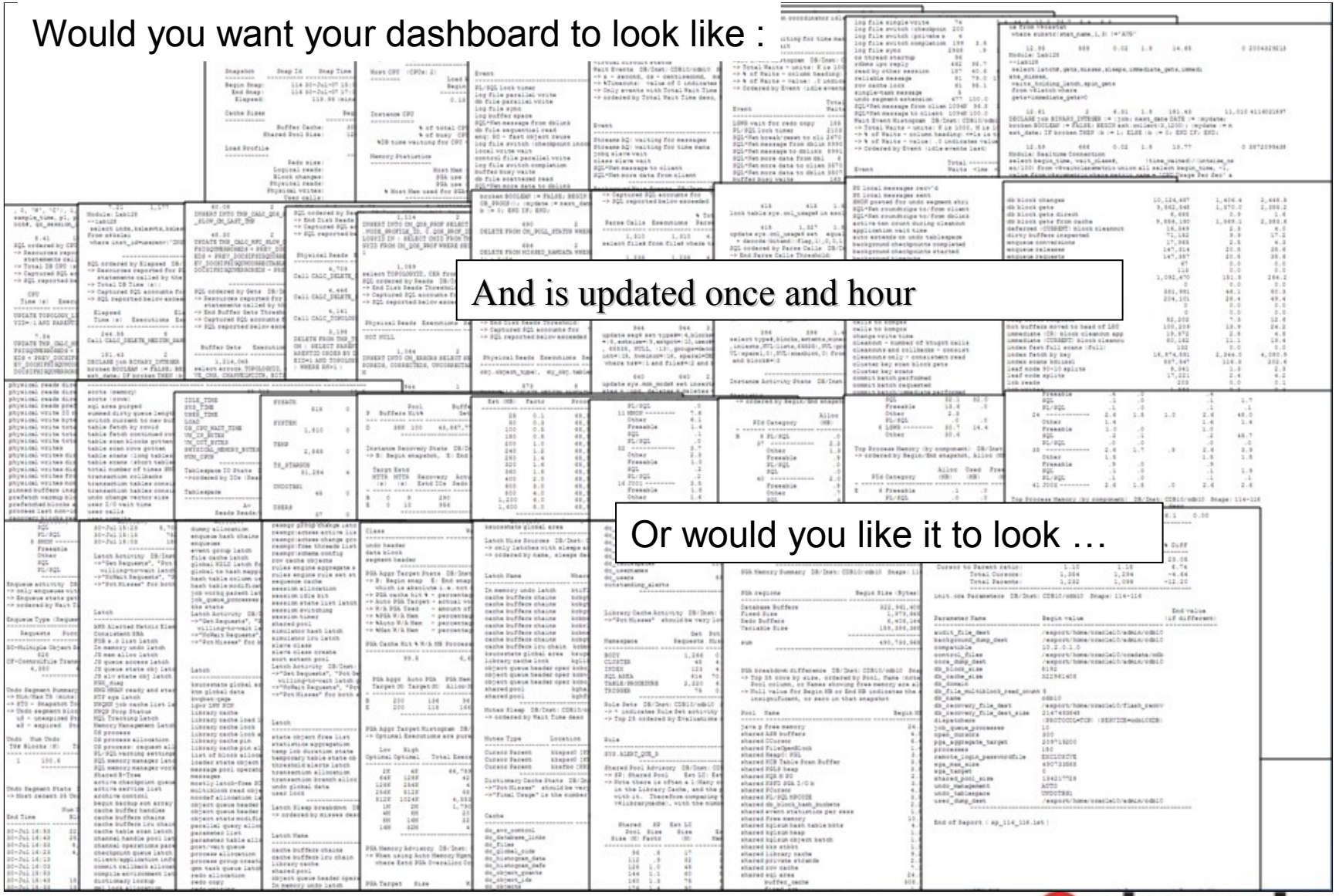


Do You Want?

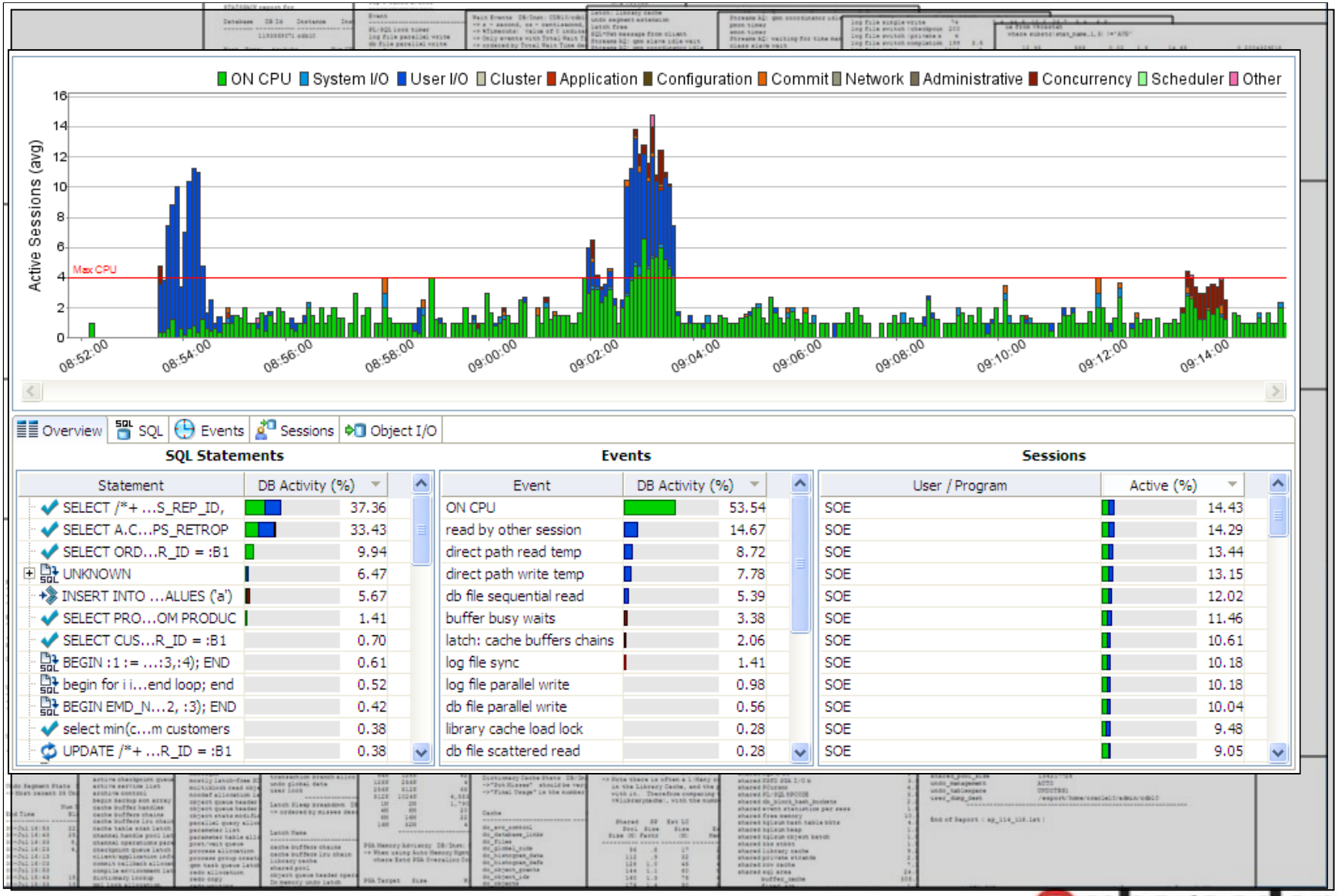
Clean and Clear



Imagine Trying to Drive your Car



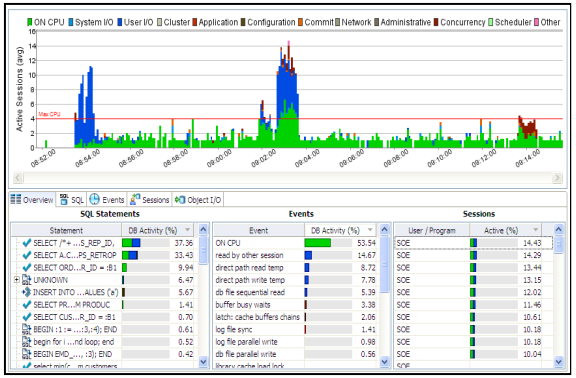
Or This



Summary

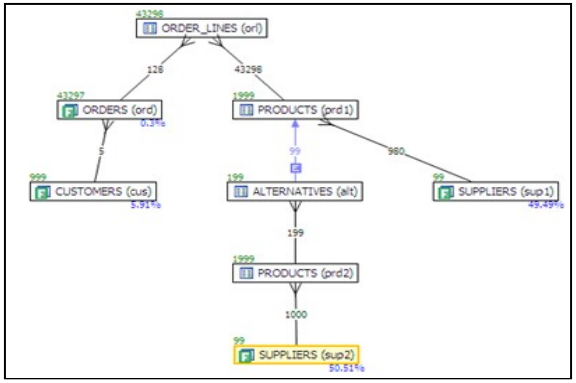
1. Database - AAS

- Profile database
- Use wait interface and graphics
- Identify machine, application, database or SQL



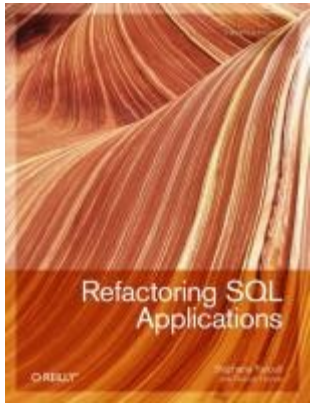
1. SQL - VST

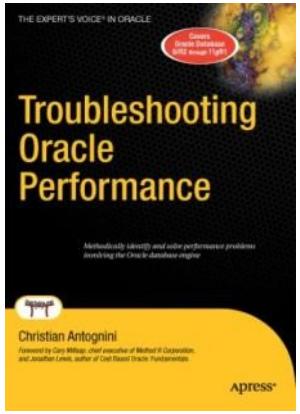
- Indexes, stats, execution path
- Visual SQL Tuning

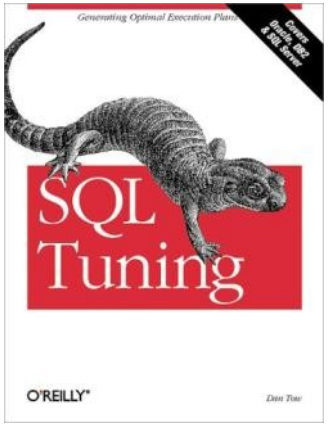


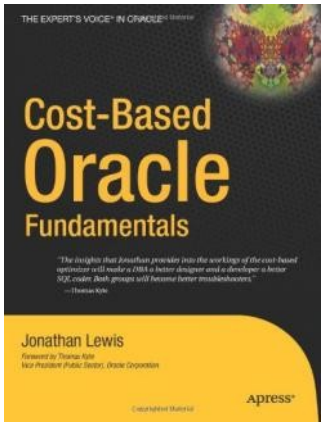
embarcadero

Bibliography



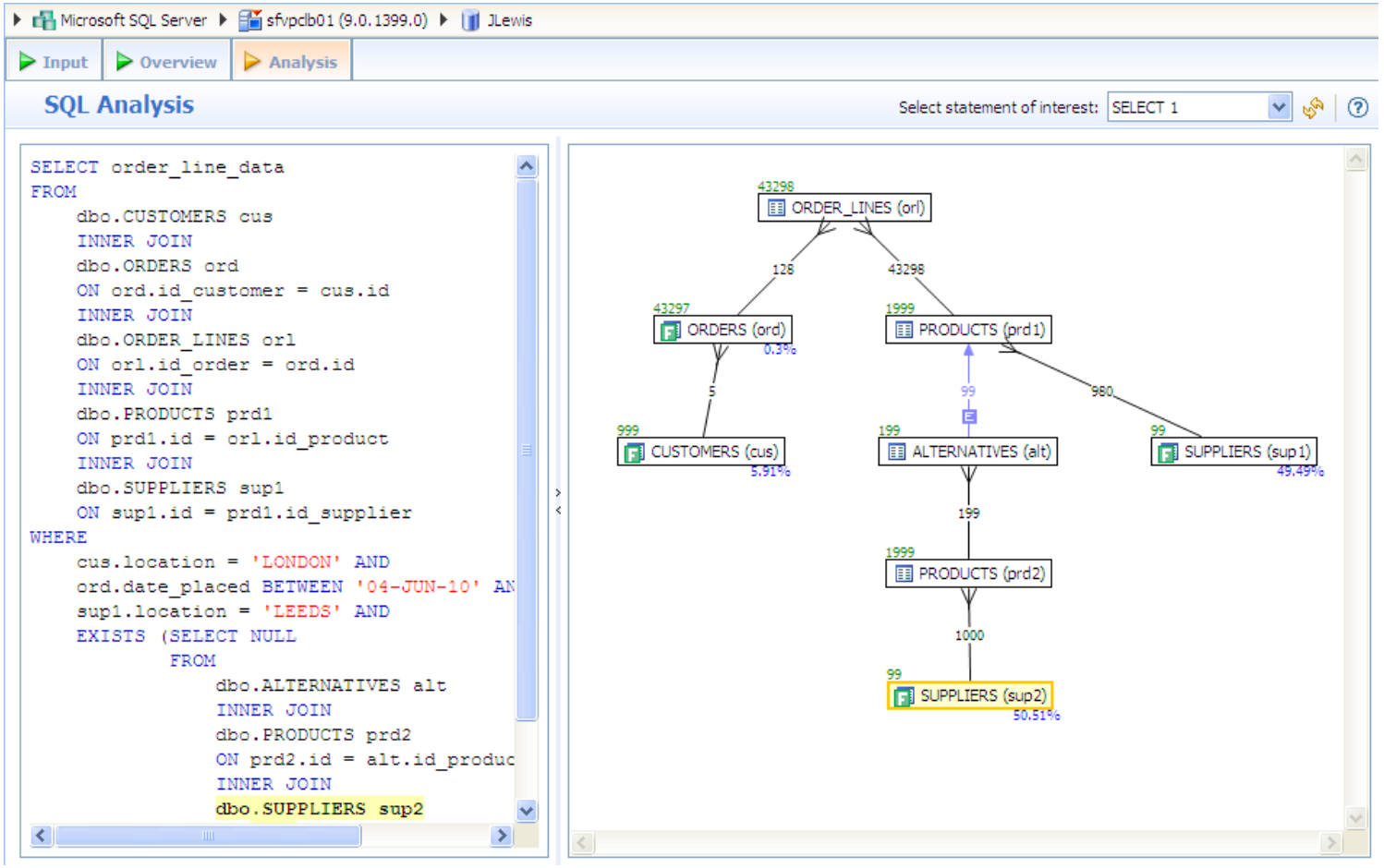






Refactoring SQL Applications – Stephane Faroult
Troubleshooting Oracle Performance – Christian Antognini
SQL Tuning – Dan Tow
Cost-Based Oracle Fundamentals – Jonathan Lewis
http://www.simple-talk.com/sql/performance/designing-efficient-sql-a-visual-approach/

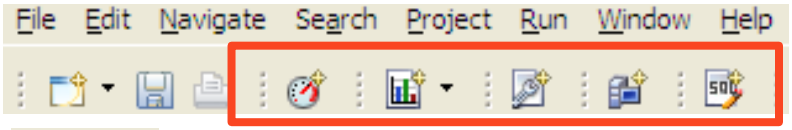
Visual SQL Tuning (VST)

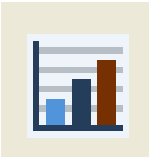


Industry Exclusive

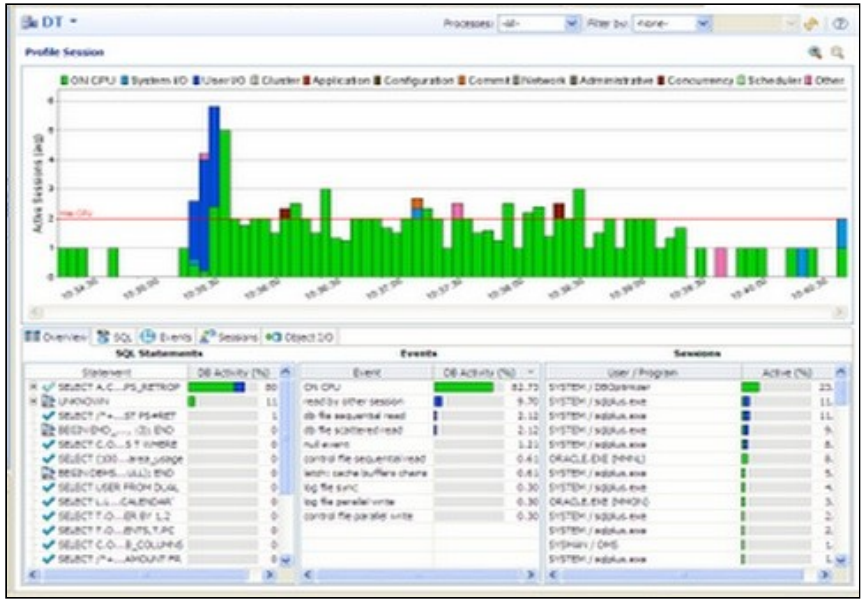
embarcadero

DB Optimizer XE Key Features



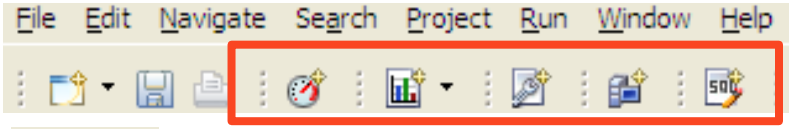


Profile



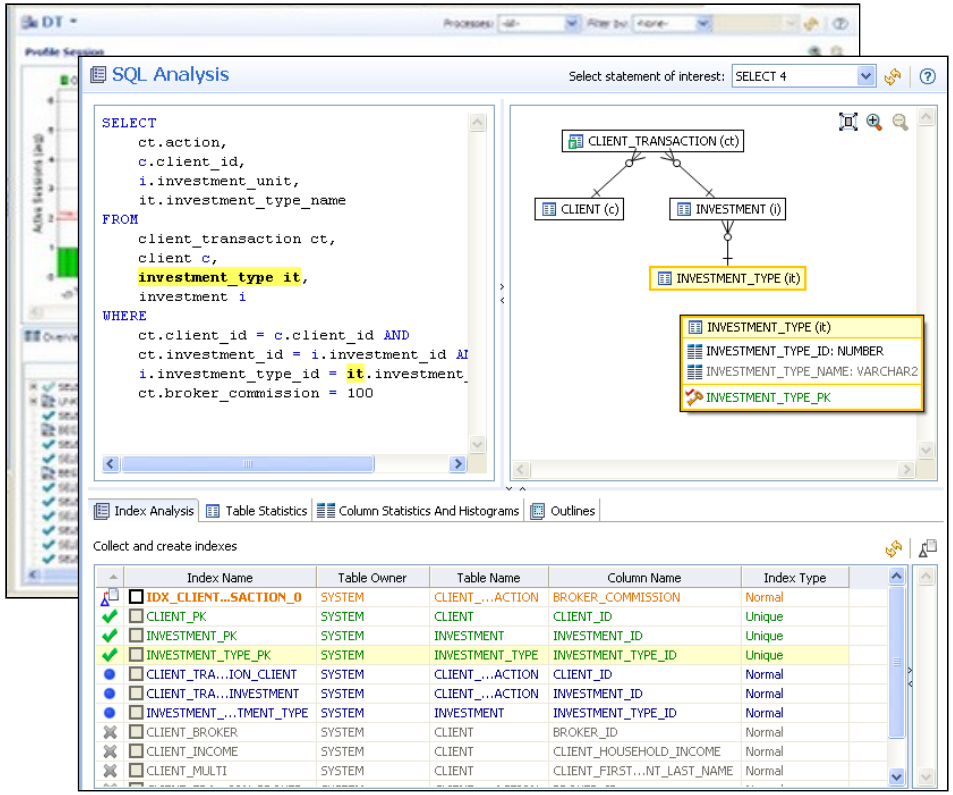
embarcadero

DB Optimizer XE Key Features



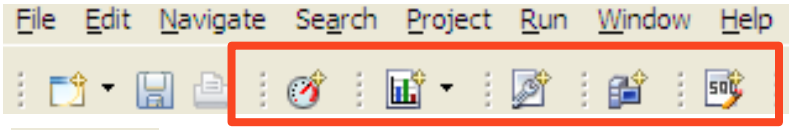
Profile

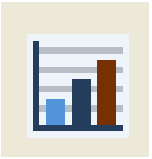
Tune



embarcadero

DB Optimizer XE Key Features

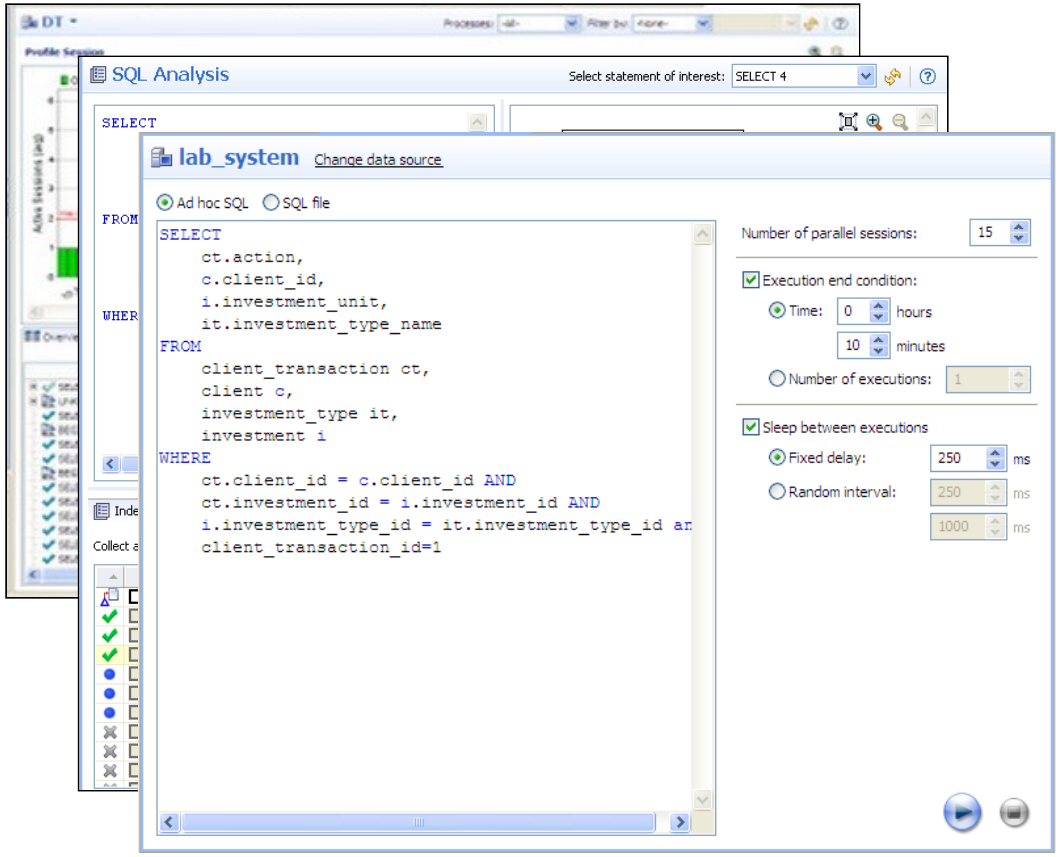




Profile

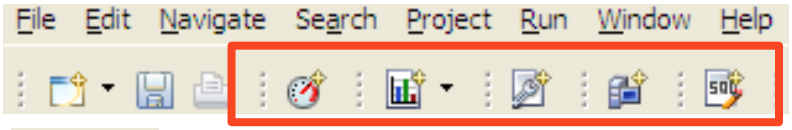
Tune

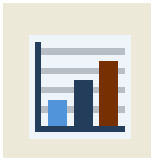
Load Test



embarcadero

DB Optimizer XE Key Features



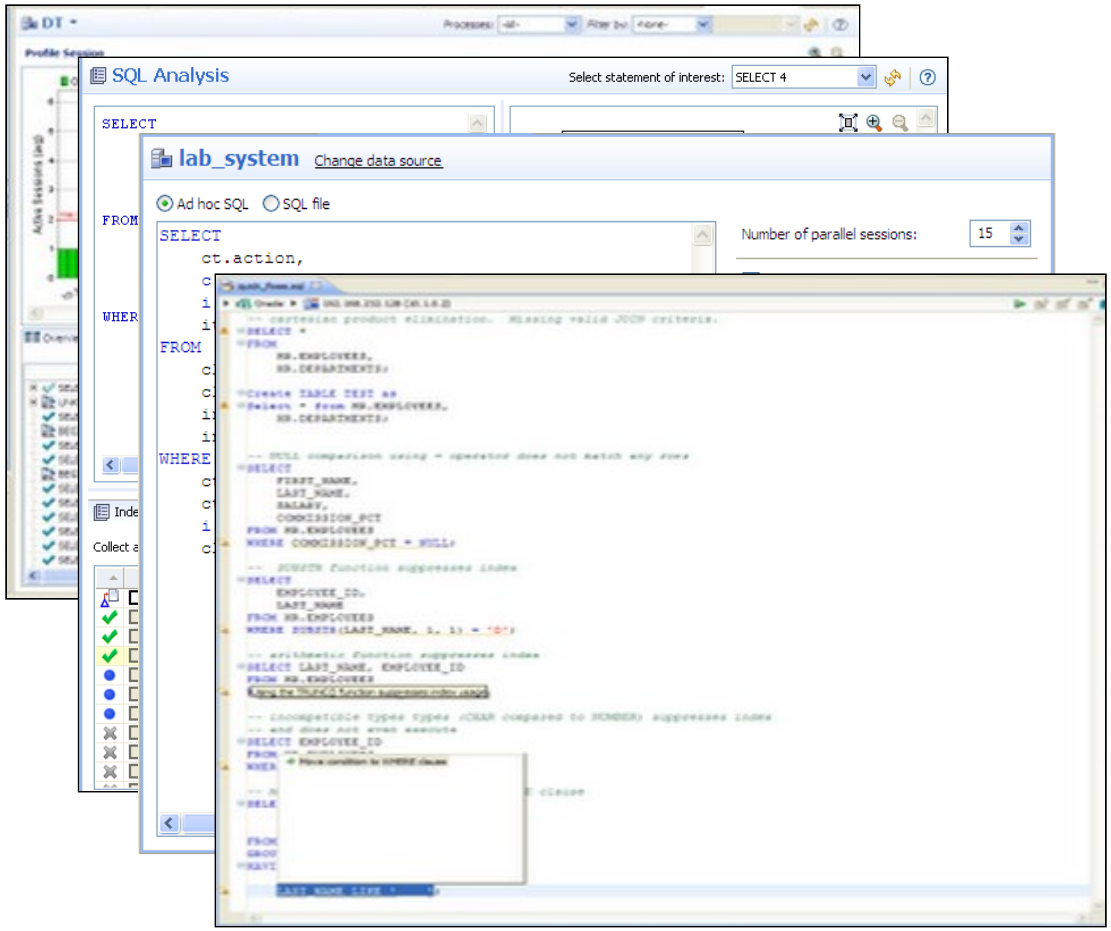


Profile

Tune

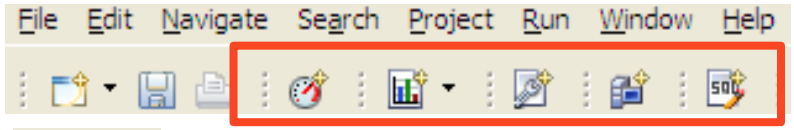
Load Test

SQL IDE



embarcadero

Review of Functionality



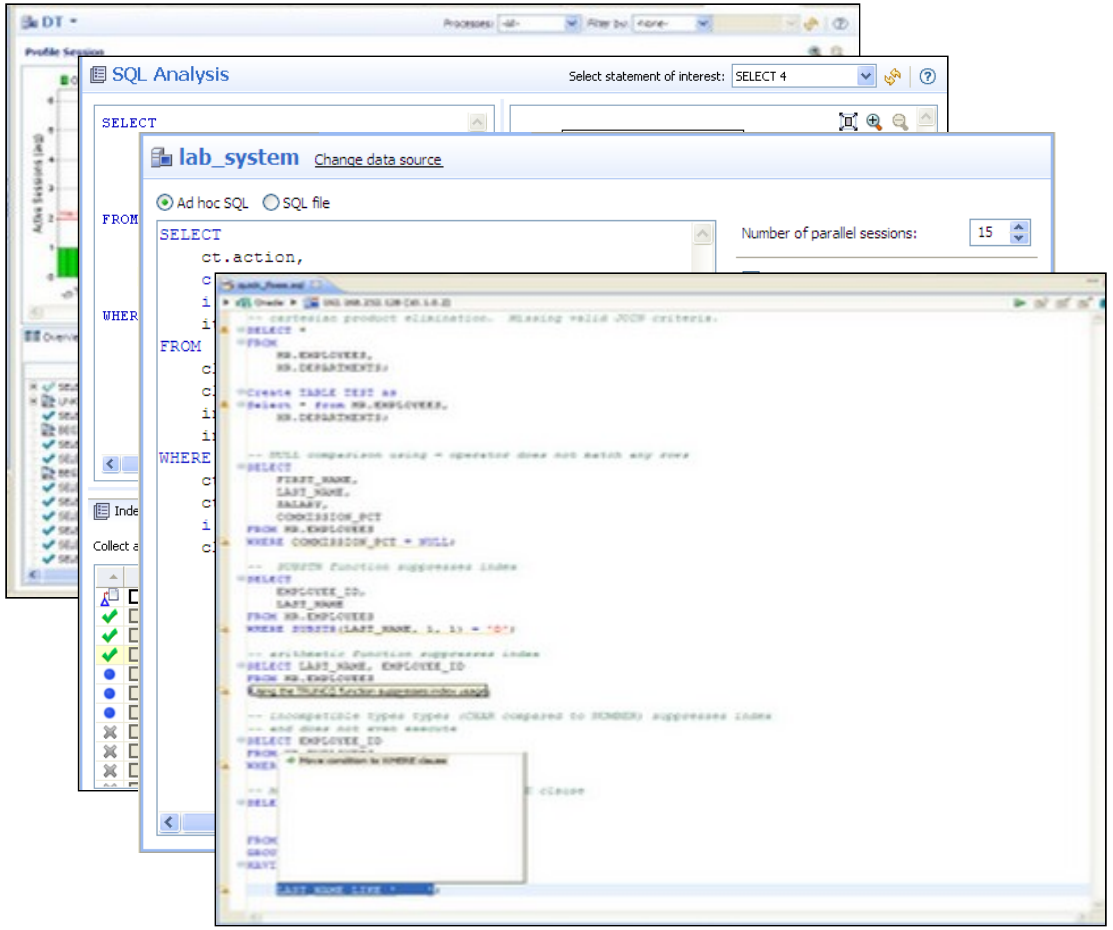
Profile

Tune

Load Test

SQL IDE

Only tool on the market with these features integrated



embarcadero

Thank You

Kyle Hailey
http://oraclemonitor.com

Appendix

- Diagramming
  - Simple joins and inline views
  - Outer joins
  - Exists/not exists (in/not in)
  - Correlated aggregate sub-queries
- Hints
  - LEADING
  - USE_NL
  - USE_HASH
  - INDEX
  - NO_MERGE
- Execution Order
- Machine Health
- VST method with Statistics

Basic Joins

Many to single value

M

One to one

One to many

Safe

Dangerous

Many to many

Cartesian

Cartesian

Simple queries and sub-queries

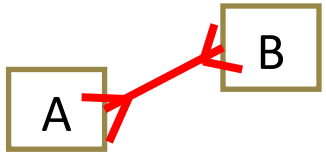
Simple join

```
select * from A, B where b.f1=a.f1
```

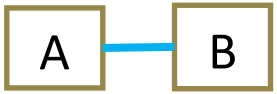
Non-correlated sub-query

```
select * from A,(select b.f1 from B) c
where c.f1=a.f1
```

Unique index on b.f1



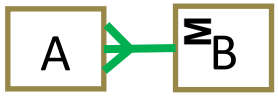
No unique indexes



Unique index on
Both b.f1 and a.f1

Special case: non correlated sub-query returns one row

```
select * from A where a.f1 =
                (select max(b.f1) from B )
```



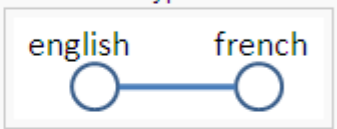
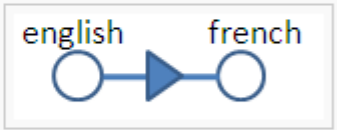
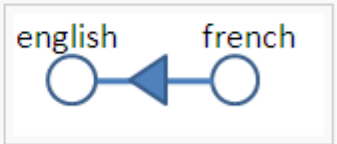
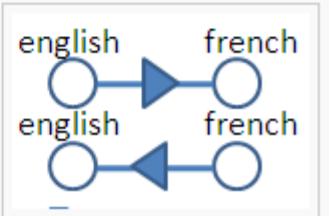
74

Outer Joins

ENGLISH_TEXT	ORDINAL_ID
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

ORDINAL_ID	FRENCH_TEXT
1	Un
3	Trois
4	Quatre
5	Cinq
6	Six
7	Sept
8	Huit

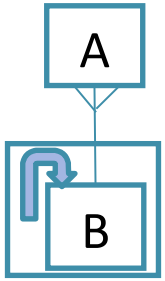
Outer Joins

type	ANSI	ANSI 89 (Oracle)	type	type
inner join	english INNER JOIN french using (ordinal_id)	english e, french f where e.ordinal_id=f.ordinal_id		
left outer join	english LEFT JOIN french using (ordinal_id)	english e, french f where e.ordinal_id=f.ordinal_id(+)		
right outer join	english RIGHT JOIN french using (ordinal_id)	english e, french f where e.ordinal_id(+)=f.ordinal_id		
full join	english FULL JOIN french using (ordinal_id)	english e, french f where e.ordinal_id=f.ordinal_id(+) UNION english e, french f where e.ordinal_id(+)=f.ordinal_id		

Correlated scalar sub-queries

Correlated aggregate subquery:

```
select ename from emp a where a.sal >
          (select avg(sal)
           from emp b
           where a.deptno=b.deptno)
```

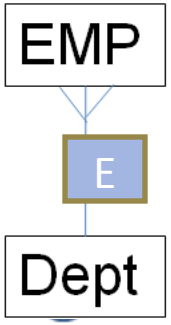


Scalar Subqueries

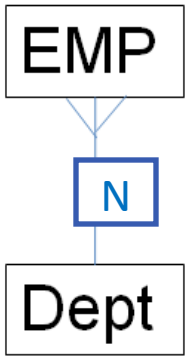
```
select ename, (select avg(sal)
           from emp b
           where a.deptno=b.deptno)
 from emp a;
```

Exists and Not In

```
SELECT d.*
FROM dept d  WHERE exists (
 SELECT null FROM emp e  WHERE e.deptno=d.deptno);

SELECT d.*
FROM dept d  WHERE d.deptno in (
SELECT  deptno FROM emp e  );

select distinct d.* from dept1 d ,emp e
where e.deptno = d.deptno;
```




```
SELECT  d.*
FROM dept d  WHERE not exists (
SELECT null FROM emp e  WHERE e.deptno=d.deptno);

SELECT  d.*
FROM dept d  WHERE d.deptno not in (
SELECT deptno FROM emp e where e.deptno is not null )
or d.deptno is null;

select d.*  from dept1 d  left outer join emp e
on e.deptno = d.deptno  where e.deptno is null;
```



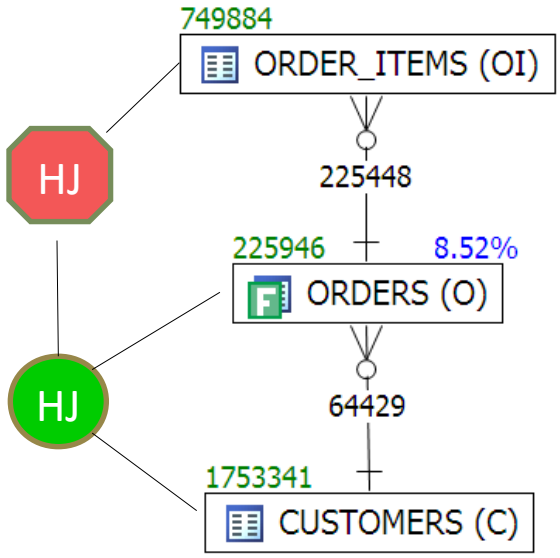
HINTS

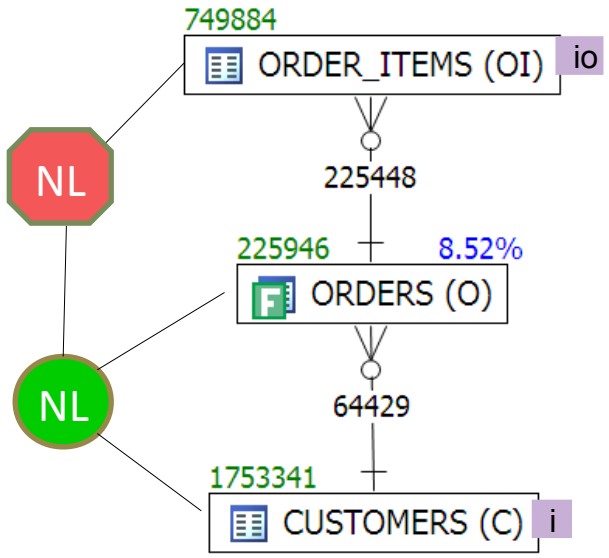
- ORDERED - good on 9i
- Leading(tab_alias , table_alias …) – 10g format
- USE_NL(table_alias) – Inner Table (not driving)
- USE_HASH(table_alias) – 2cd table, probe into
- INDEX(tab_alias index_name)
- NO_MERGE

Oracle first decides join order then join type

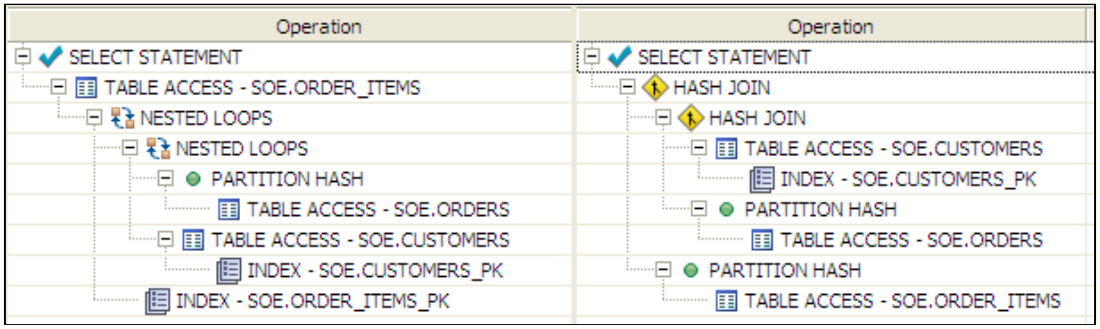
(example http://www.adp-gmbh.ch/blog/2008/01/17.php)

Visual SQL Tuning (VST) diagrams





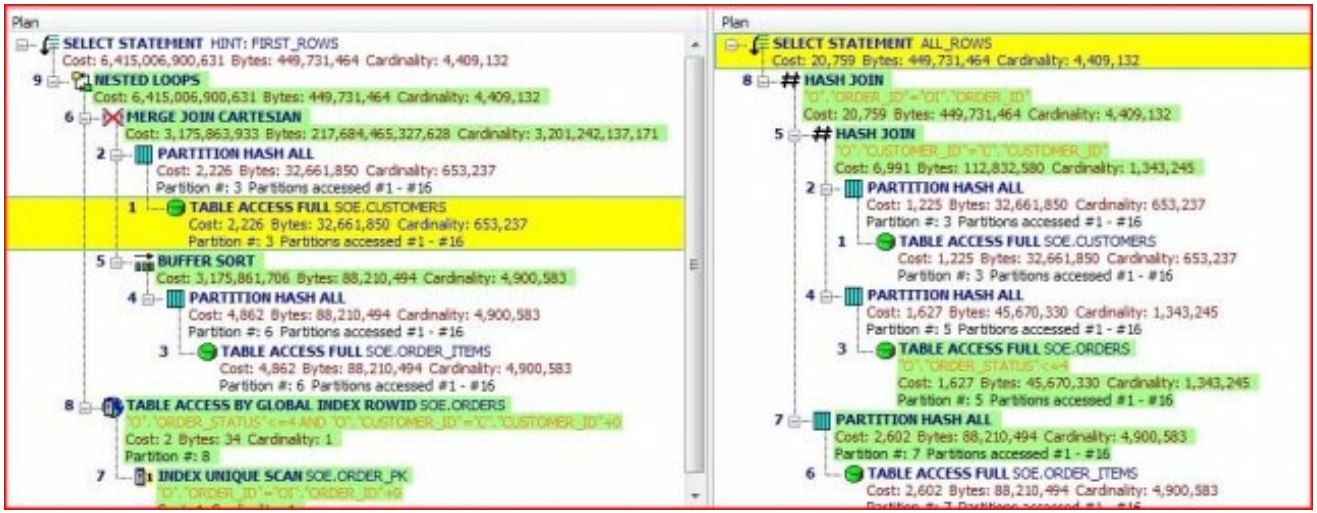
Makes comparison of execution plans easy
Better yet, it will help us find the best execution path

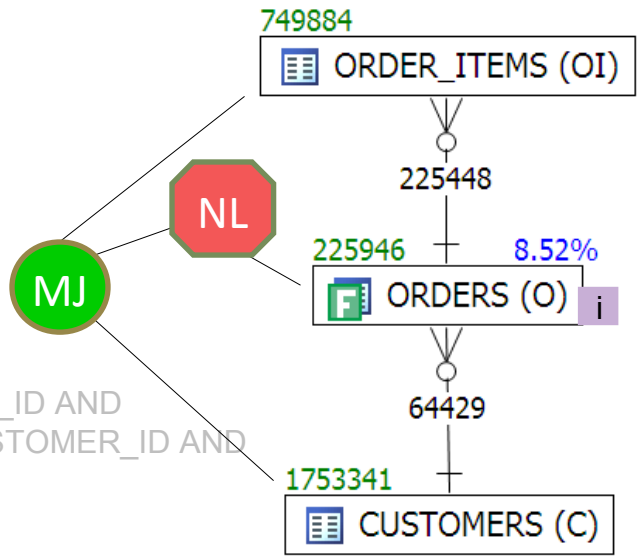


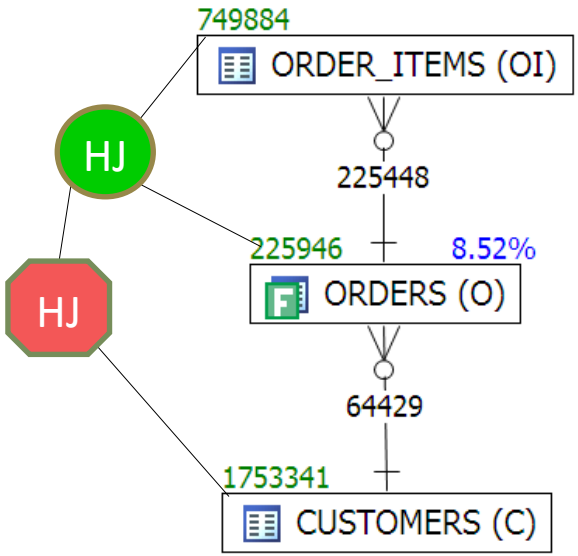
79

VST vs Explain Plan

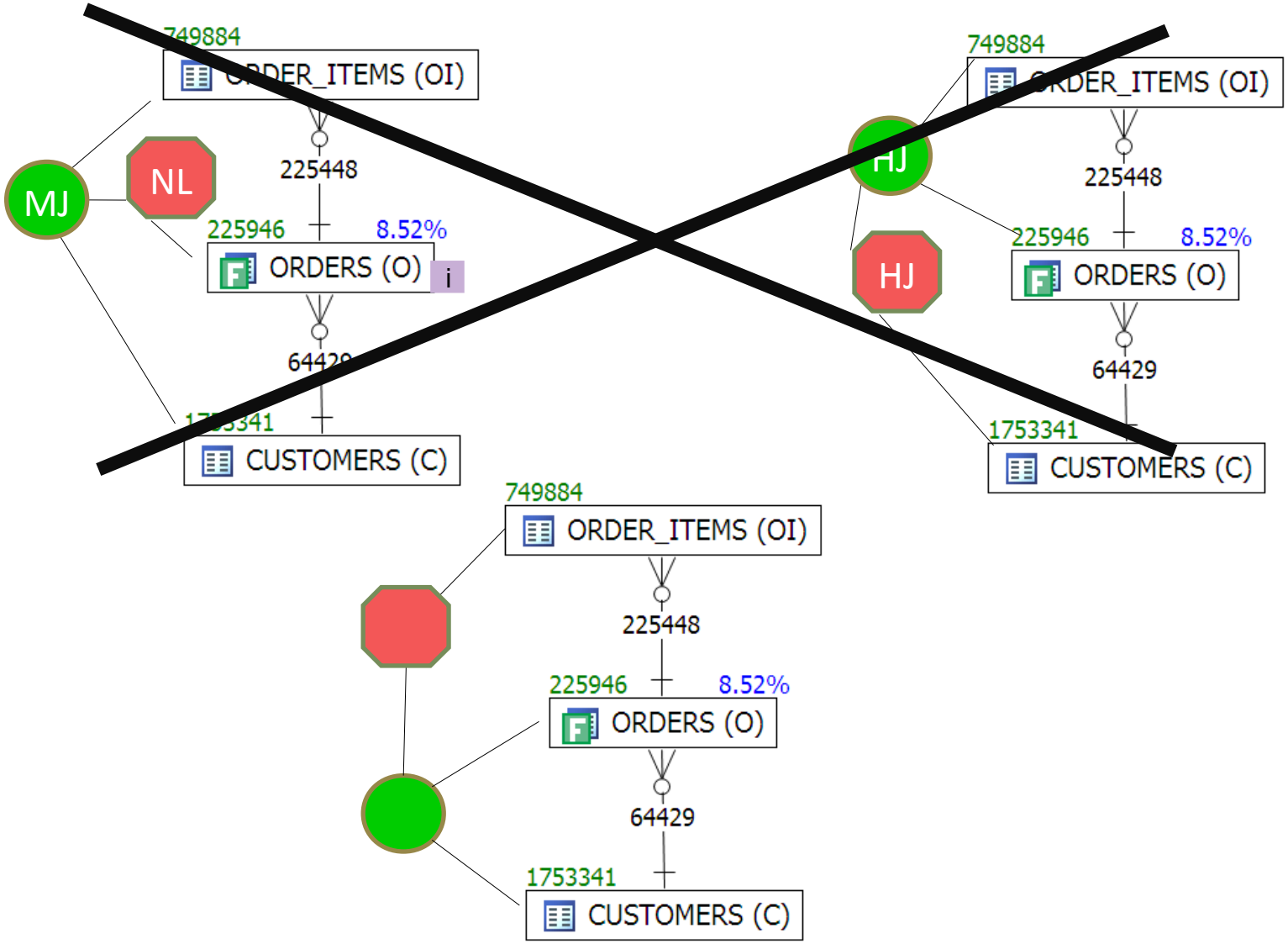
```
SELECT
   O.ORDER_ID,
   LINE_ITEM_ID,
   PRODUCT_ID,
   UNIT_PRICE,
   QUANTITY,
   ORDER_MODE,
   ORDER_STATUS,
   ORDER_TOTAL,
   SALES_REP_ID,
   PROMOTION_ID,
   C.CUSTOMER_ID,
   CUST_FIRST_NAME,
   CUST_LAST_NAME,
   CREDIT_LIMIT,
   CUST_EMAIL,
   ORDER_DATE
FROM
   ORDERS O,
   ORDER_ITEMS OI,
   CUSTOMERS C
WHERE
   O.ORDER_ID = OI.ORDER_ID AND
   O.CUSTOMER_ID = C.CUSTOMER_ID AND
   O.ORDER_STATUS <= 4
```



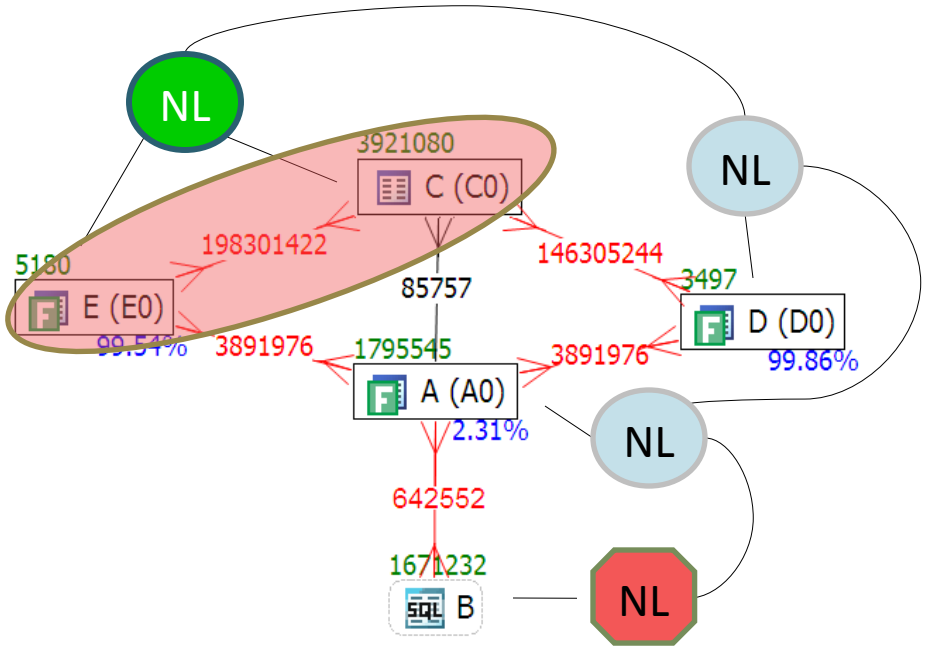


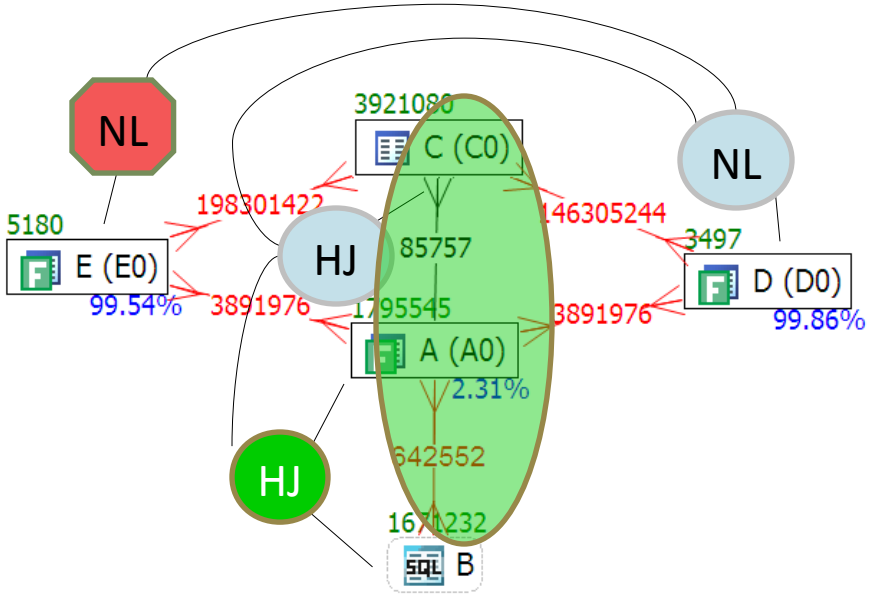


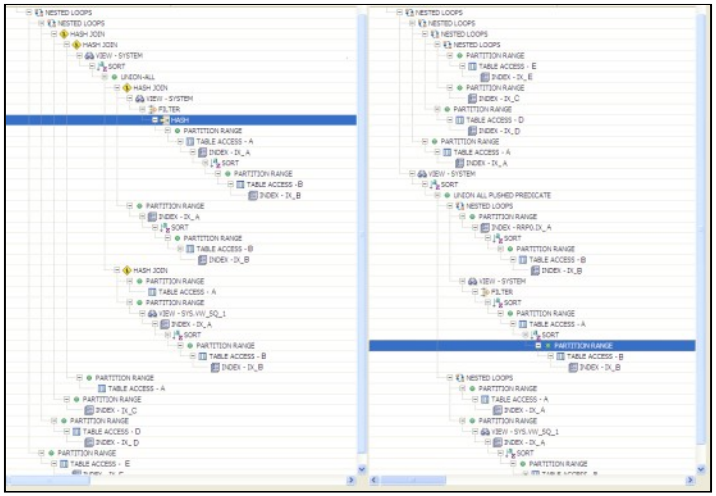
Bad Plans vs Good Plan



Comparing Plans : 24 hours to 5 mins



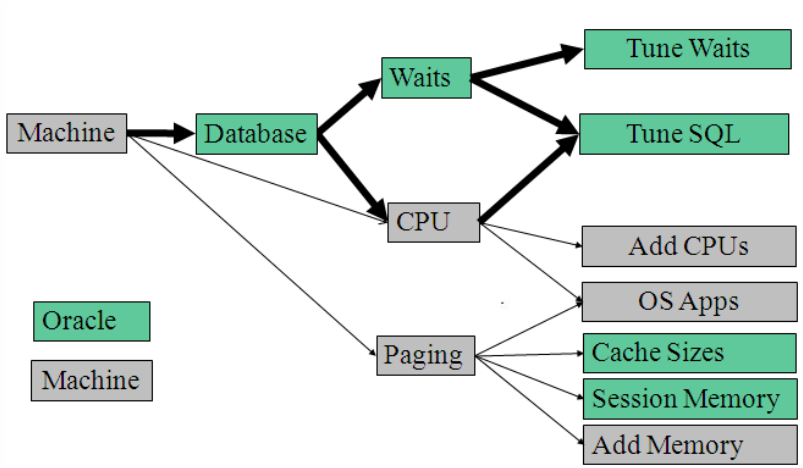




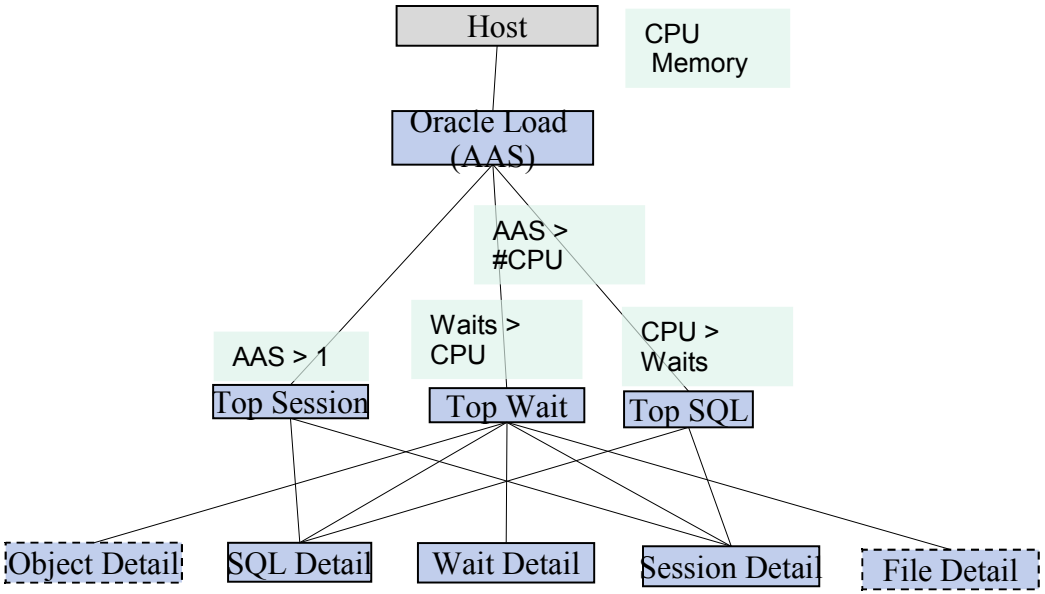
embarcadero

When to Tune

1. Machine
   a) CPU
      - Response times skewed
      - 100% CPU might be fine
      - Users wait in queue (run queue) => machine under
   a) Memory
      - Paging
      - Wait times skewed (ex : latch free)
      - Erratic response times (ex : ls)



1. Oracle
   1) Waits > CPU ?
      - tune waits
   1) CPU > 100% ?
      - tune top CPU SQL
   1) Else
      - It's the application

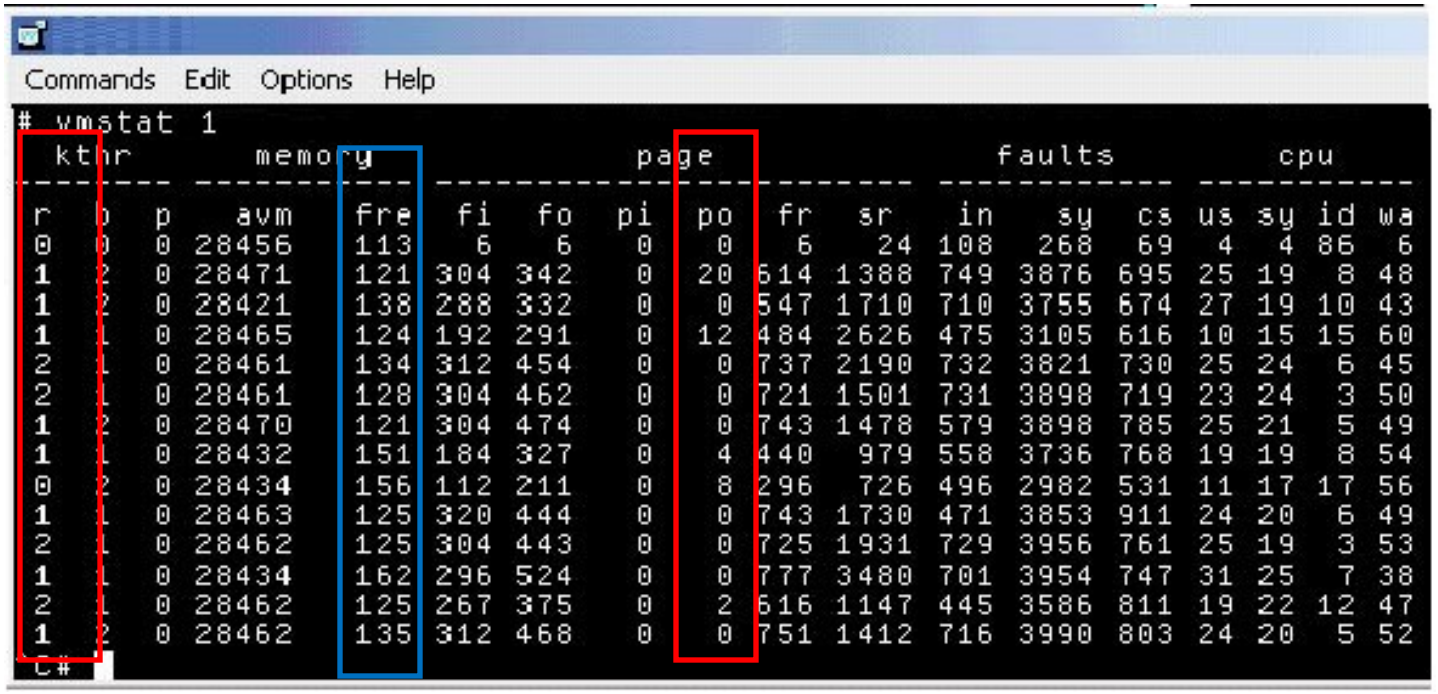


Machine

Make sure the machine is healthy before tuning Oracle

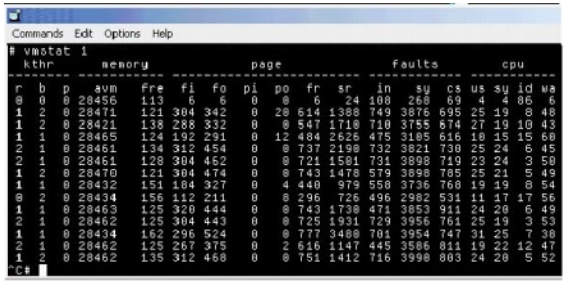
- CPU => use run queue, < 2 * #CPU
- Memory => page out

VMSTAT

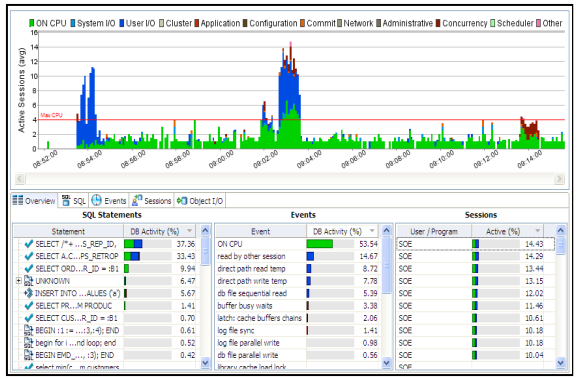


Summary

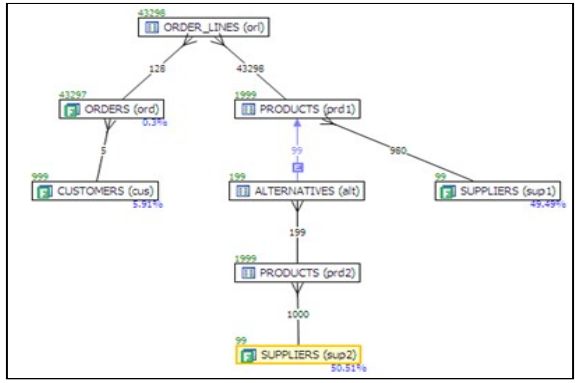
1. Machine - vmstat
   - Memory, CPU (we can see IO response in Oracle)



1. Database - AAS
   - Use wait interface and graphics
   - Identify machine, application, database or SQL



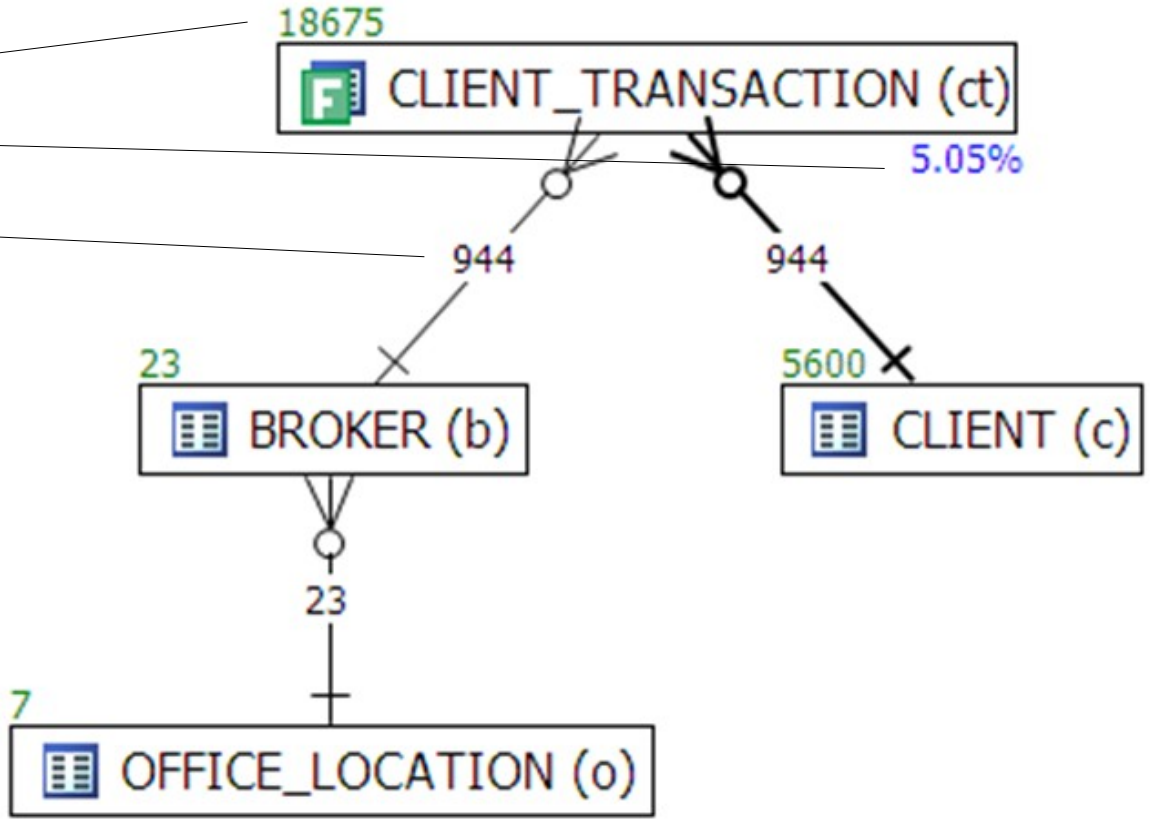
1. SQL - VST
   - Indexes, stats, execution path
   - Visual SQL Tuning



embarcadero

Solution to Many to Many

Table Sizes
Filter Ratios
Join Sizes



VST Steps

Objects:

1. Tables : drawn as nodes
2. Joins : drawn as connector lines
3. Filters : mark on each table with filter in where clause

Statistics:

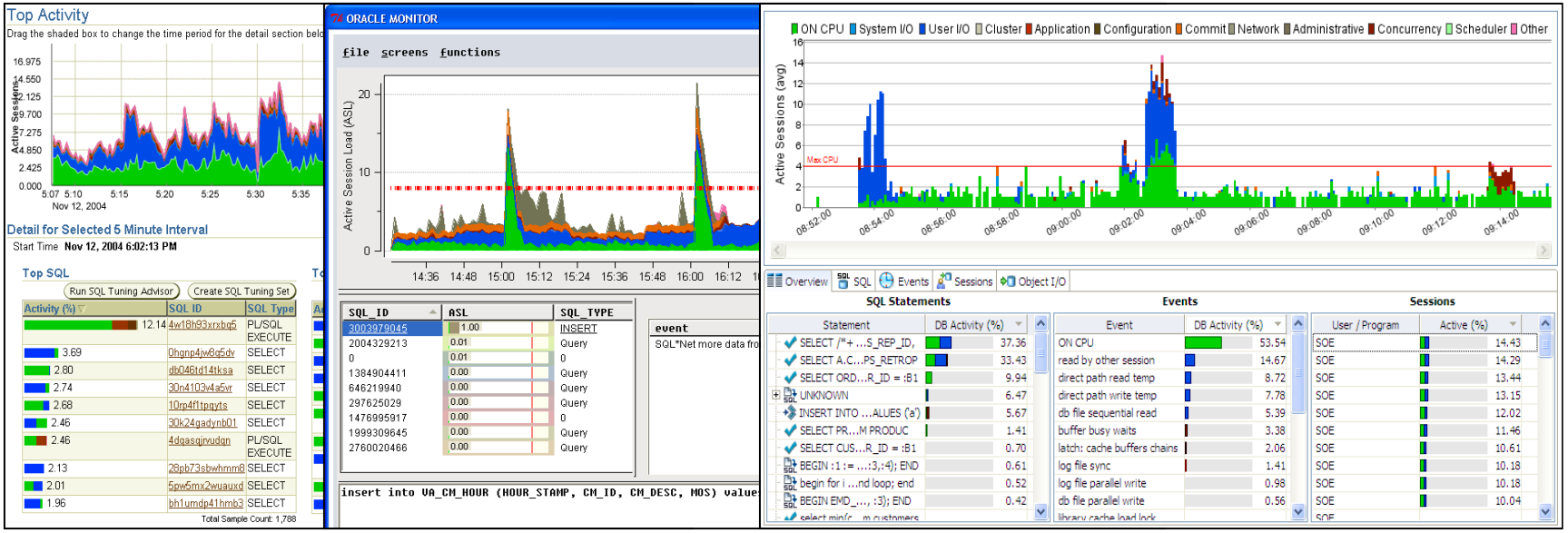
1. Table sizes
2. Join sizes
3. Calculate filter percentages
   - filter ratio = number of rows returned with filter / number of rows

Google Directions

- Compare two sets of directions
- (optionally show graphical set of directions)
- Show map
- Show map with traffic flow, read, yellow, green

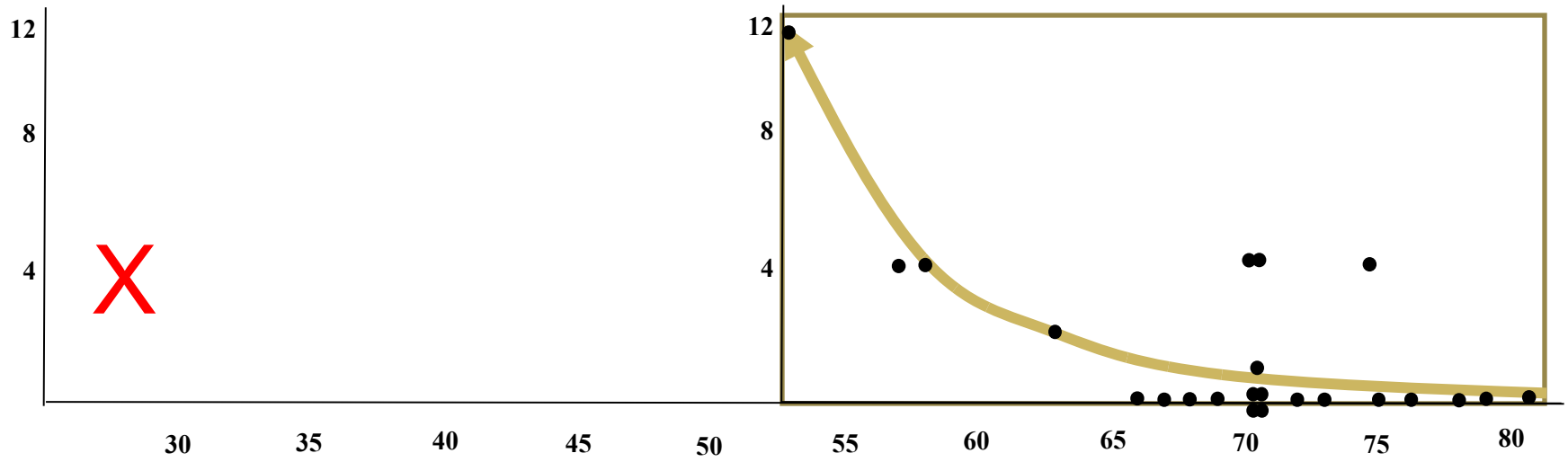
How Can We Open the Black Box?

OEM ASHMON/SASH DB Optimizer



- **Powerful** - Identifies issues quickly and powerfully
- **Interactive** - Allows exploring the data
- **Easy** - Understandable by everyone, DBA, Dev and Managers !

Clearer



1. Include successes
2. Remove Irrelevant
3. Normalize same temp
4. Scale known vs unknown