DB Performance with Graphics
With Kyle Hailey
Product Manager 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
Launch: Pressure

Midnight before
January 28, 1986
Lives are on the line

Thanks to Edward Tufte
HISTORY OF O-RING DAMAGE ON SRM FIELD JOINTS

CONCLUSIONS:

0 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 73°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

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

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

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

RECOMMENDATIONS:

0 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

0 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 BLOWHOLE IN PUTTY.
Original Engineering data

“damages at the hottest and coldest temperature”
- management

only showed damage
History of O-Ring Damage in Field Joints (Cont)

<table>
<thead>
<tr>
<th>SRM No.</th>
<th>O-Ring Temp (°F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>66</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
</tr>
<tr>
<td>A</td>
<td>69</td>
</tr>
<tr>
<td>B</td>
<td>68</td>
</tr>
<tr>
<td>A</td>
<td>80</td>
</tr>
<tr>
<td>B</td>
<td>67</td>
</tr>
<tr>
<td>A</td>
<td>167</td>
</tr>
<tr>
<td>B</td>
<td>72</td>
</tr>
<tr>
<td>A</td>
<td>73</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
</tr>
<tr>
<td>A</td>
<td>63</td>
</tr>
<tr>
<td>B</td>
<td>78</td>
</tr>
</tbody>
</table>

O-Ring Temp (°F)

<table>
<thead>
<tr>
<th>SRM No.</th>
<th>O-Ring Temp (°F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>75</td>
</tr>
<tr>
<td>14</td>
<td>67</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>67</td>
</tr>
<tr>
<td>17</td>
<td>181</td>
</tr>
<tr>
<td>18</td>
<td>70</td>
</tr>
<tr>
<td>19</td>
<td>76</td>
</tr>
<tr>
<td>20</td>
<td>79</td>
</tr>
<tr>
<td>21</td>
<td>75</td>
</tr>
<tr>
<td>22</td>
<td>76</td>
</tr>
<tr>
<td>23</td>
<td>79</td>
</tr>
<tr>
<td>24</td>
<td>58</td>
</tr>
</tbody>
</table>

*Morton Thiokol, Inc.*

_Wasatch Operations_

*No Erosion*

Information on this page was prepared to support an oral presentation and cannot be considered complete without the oral discussion.
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**.”

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

Cancer Mortality Rates by County (Age-adjusted 1970 US Population)
Lung, Trachea, Bronchus, and Pleura: White Males, 1970-94
“If I can't picture it, I can't understand it”
- Albert Einstein

<table>
<thead>
<tr>
<th>Anscombe's Quartet</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>8.04</td>
<td>10</td>
<td>9.14</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>6.95</td>
<td>8</td>
<td>8.14</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>7.58</td>
<td>13</td>
<td>8.74</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>8.81</td>
<td>9</td>
<td>8.77</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>8.33</td>
<td>11</td>
<td>9.26</td>
<td>11</td>
</tr>
<tr>
<td>14</td>
<td>9.96</td>
<td>14</td>
<td>8.1</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>7.24</td>
<td>6</td>
<td>6.13</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4.26</td>
<td>4</td>
<td>3.1</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>10.84</td>
<td>12</td>
<td>9.13</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>4.82</td>
<td>7</td>
<td>7.26</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>5.68</td>
<td>5</td>
<td>4.74</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>9</td>
<td>7.5</td>
<td>9</td>
<td>7.5</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>3.31</td>
<td>2.03</td>
<td>3.31</td>
<td>2.03</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>1.33</td>
<td>1.33</td>
<td>1.33</td>
<td>1.33</td>
</tr>
</tbody>
</table>
Graphics for Anscombe’s Quartet
What is a day in the life look like for a DBA who has performance issues?

Tuning the Database

Complex Averages

Anscombe's Quartet

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>y</td>
<td>7.5</td>
<td>7.5</td>
<td>7.5</td>
<td>7.5</td>
</tr>
</tbody>
</table>

Average

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>7.5</td>
<td>9</td>
<td>7.5</td>
<td>9</td>
</tr>
</tbody>
</table>

Standard Deviation

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.31</td>
<td>2.03</td>
<td>3.31</td>
<td>2.03</td>
<td>3.31</td>
</tr>
</tbody>
</table>

Linear Regression

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.33</td>
<td>1.33</td>
<td>1.33</td>
<td>1.33</td>
<td>1.33</td>
</tr>
</tbody>
</table>
How Can We Open the Black Box?
How do we get our data?

Database

<table>
<thead>
<tr>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
</tr>
</thead>
</table>

Active Sessions

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)

Waits (v$system_event)

Sessions (v$sesstat)

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

Package
Procedure
Plan
Child #
Module
Action

Sessions
Program
User
Service

SQL

25 34 36 38 45 63 65 87

Sampling V$SESSION

File, block, object
p1, p2, p3

CPU
End: TX - row lock contention
SQL *Net break/reset to client
db file scattered read
db file sequential read

Waits
Statistics

Application
IO

06/05/08
DB Optimizer shows it all
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
Interpreting the Load Chart

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

4 concurrent sessions run:

```sql
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;
```

![Diagram showing SQL optimization and concurrency issues](image_url)
2. Database issue

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

4 concurrent sessions run:

begin
  for i in 1..1000 loop
    insert into foo values ('a');
  end loop;
end;
/
Commit;
3. Machine Undersized
4. SQL needs Tuning
Tuning SQL Complexity

Trace file

Parsing in cursor #2 len=53 dep=0 uid=61 oct=2
id=61 tim=1151519905950403 hv=2296704914
ad='4e50010c'
SELECT 'Hello, world; today is ' || SYSDATE FROM dual
END OF STMT

Parse
#2:c=4000,e=1540,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,tim=1151519905950397

Bins #2:
EXEC
#2:c=0,e=58,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=1151519906034809

Wait #2: nam='SQL*Net message to client' ela=2
id=1650815232 #bytes=1 p3=0 obj#=-1

FETCH
#2:c=0,e=29,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=0,tim=1151519906035165

Wait #2: nam='SQL*Net message from client' ela=215
driver id=1650815232 #bytes=1 p3=0 obj#=-1

Wait #2: nam='SQL*Net message to client' ela=1
driver id=1650815232 #bytes=1 p3=0 obj#=-1

Wait #2: nam='SQL*Net message from client' ela=192
driver id=1650815232 #bytes=1 p3=0 obj#=-1

Wait #2: nam='SQL*Net message to client' ela=1
driver id=1650815232 #bytes=1 p3=0 obj#=-1

Wait #2: nam='SQL*Net message from client' ela=192
driver id=1650815232 #bytes=1 p3=0 obj#=-1

Stat #2 id=1 cnt=1 pid=0 pos=1 obj=0 op='FAST DUAL'
(c=0 pr=0 pw=0 time=3 us)

Execution Plan

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hash Group By</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>*  2</td>
<td>Filter</td>
<td></td>
<td>1</td>
<td></td>
<td>1909</td>
</tr>
<tr>
<td>*  3</td>
<td>Table Access By Index Rowid</td>
<td>PS_RETROPAYPGM_TBL</td>
<td>1</td>
<td>1</td>
<td>3413</td>
</tr>
<tr>
<td>4</td>
<td>Sort Aggregate</td>
<td></td>
<td>1</td>
<td>165</td>
<td>6827</td>
</tr>
<tr>
<td>5</td>
<td>Index Range Scan</td>
<td></td>
<td>1</td>
<td>165</td>
<td>3413</td>
</tr>
<tr>
<td>6</td>
<td>Index Range Scan</td>
<td></td>
<td>1</td>
<td>165</td>
<td>3624</td>
</tr>
<tr>
<td>7</td>
<td>Table Access By Index Rowid</td>
<td>WB_JOB</td>
<td>1</td>
<td>242</td>
<td>2895</td>
</tr>
<tr>
<td>8</td>
<td>Nested Loops</td>
<td></td>
<td>1</td>
<td>233</td>
<td>2897</td>
</tr>
<tr>
<td>9</td>
<td>Table Access By Index Rowid</td>
<td>PS_PAY_CALENDAR</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Index Range Scan</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Index Range Scan</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Table Access Full</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>Table Access Full</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>*  14</td>
<td>Index Range Scan</td>
<td>PS#RETROPAYPGM_TBL</td>
<td>3413</td>
<td>1</td>
<td>3413</td>
</tr>
<tr>
<td>15</td>
<td>Sort Aggregate</td>
<td></td>
<td>1791</td>
<td>1</td>
<td>1791</td>
</tr>
<tr>
<td>16</td>
<td>First Row</td>
<td></td>
<td>1791</td>
<td>1</td>
<td>1579</td>
</tr>
<tr>
<td>*  17</td>
<td>Index Range Scan (Min/Max)</td>
<td>WB_JOB_F</td>
<td>1791</td>
<td>1</td>
<td>1579</td>
</tr>
<tr>
<td>18</td>
<td>Sort Aggregate</td>
<td></td>
<td>1539</td>
<td>1</td>
<td>1539</td>
</tr>
<tr>
<td>19</td>
<td>First Row</td>
<td></td>
<td>1539</td>
<td>1</td>
<td>1539</td>
</tr>
<tr>
<td>*  20</td>
<td>Index Range Scan (Min/Max)</td>
<td>WB_JOB_G</td>
<td>1539</td>
<td>1</td>
<td>1539</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

2 - filter(('B''EFFDT''= AND 'B''EFFSEQ''=')
3 - filter('E''OFF_CYCLE''='A''PAY_OFF_CYCLE_CAL')
5 - access('D''RETROPAY_SEQ_NO''='C''RETROPAY_SEQ_NO')
6 - access('C''EMPLID''='B''EMPLID'' AND 'C''EMPL_RCD#''='B''EMPL_RCD#')
10 - access('A''RUN_ID''='PD2' AND 'A''PAY_CONFIRM_RUN''='N')
11 - access('B''COMPANY''='A''COMPANY'' AND 'B''PAYGROUP''='A''PAYGROUP')
12 - filter(('C''RETROPAY_PRCS_FLAG''='C' AND 'C''RETROPAY_LOAD_SW''='Y'))
14 - access('E''RETROPAY_PGM_ID''='D''RETROPAY_PGM_ID')
17 - access('F''EMPLID''=:B1 AND 'F''EMPL_RCD#''=:B2 AND 'F''EFFDT''<=:B3')
20 - access('G''EMPLID''=:B1 AND 'G''EMPL_RCD#''=:B2 AND 'G''EFFDT''=:B3')
Visual SQL Tuning

SELECT ct.action,
c.client_id,
i.investment_unit,
it.investment_type_name
FROM client_transaction ct,
client c,
investment_type it,
investment i
WHERE ct.client_id = c.client_id AND ct.investment_id = i.investment_id AND i.investment_type_id = it.investment_type_id
ct.broker_commission = 100
Tuning SQL

- Indexes
  - Missing indexes

- Table Stats
  - Stale statistics

- Histograms
  - Recommendations

- Execution Plan
  - DB Optimizer automatically finds better plans

- Visual SQL Tuning diagrams
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_Line1, 
A.Street_AddressLine2, 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, Shipments S, 
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.Order_Detail_ID;

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 
- OT.Code_Type = 'ORDER_STATUS' 
- OD.Status_Code = ODT.Code 
- ODT.Code_Type = 'ORDER_DETAIL_STATUS' 
- O.Order_Date > :Now - 366

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'
Layout tables and connections

### 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
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

No index or non-unique

Primary Key (unique index)

Detail

Master

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
VST – filters and best path

- Filters help determine best path

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

```sql
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)
VST – best path

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

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

Predicate Filter

Now what?

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

```sql
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_MONTHS (MAX (planted_date),
        FROM
        FOO.B B1
    WHERE
        pears = 'M'
    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.pears,
    A4.zuchinis,
    A4.brocoli
FROM
    FOO.A A4
WHERE
    A4.planted_date > '01-OCT-08' AND
    A4.planted_date < '03-OCT-08' AND
    A4.pears = 'D' AND
    A4.green_beans = '1'
AND
    NOT EXISTS (SELECT
        *
    FROM
        FOO.A A5
    WHERE
        pears = 'M' AND
        planted_date = (SELECT
            MAX (planted_date)
        FROM
            FOO.B B3
        WHERE
            pears = 'M'
    ) AND
    A4.zuchinis = A5.zuchinis AND
    A4.brocoli = A5.brocoli)
WHERE
    A0.planted_date > '01-OCT-08' AND
    A0.planted_date < '03-OCT-08' AND
    A0.pears = 'D' AND
    A0.green_beans = '1' AND
    A0.zuchinis = b.zuchinis AND
    A0.brocoli = b.brocoli AND
    A0.planted_date = C0.planted_date AND
    A0.pears = C0.pears AND
    A0.zuchinis = C0.zuchinis AND
    A0.brocoli = C0.brocoli AND
    A0.planted_date = D0.planted_date AND
    A0.pears = D0.pears AND
    A0.harvest_size = D0.harvest_size AND
    C0.Oranges = D0.Oranges AND
    C0.apples = D0.apples AND
    (D0.lemons = 0 OR
    D0.lemons IS NULL) AND
    A0.planted_date = E0.planted_date AND
    A0.pears = E0.pears AND
    A0.harvest_size = E0.harvest_size AND
    C0.Oranges = E0.Oranges AND
    C0.apples = E0.apples AND
    (E0.lemons = 0 OR
    E0.lemons IS NULL)
ORDER BY
    A0.zuchinis, A0.brocoli;
```
Default vs Tuned

Default

Tuned

5180

198301422

3891976

3921080

C (C0)

146305244

3497

99.86%

85757

1795545

3891976

642552

1671232

2.31%

E (E0)

A (A0)

D (D0)

B

99.54%
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 
(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(+)
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

select

682348
845
825
12
Q2

The subqueries in the select clause look like

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

and should be merged into the query like:

```sql
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.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' )
```
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?

<table>
<thead>
<tr>
<th>SRM No.</th>
<th>Depth (in.)</th>
<th>Affected Perimeter (deg)</th>
<th>Nominal Dia. (in.)</th>
<th>Length Of Max Erosion (in.)</th>
<th>Total Heat Affected Length (in.)</th>
<th>Clocking Location (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>61A LH Center Field**</td>
<td>0.019</td>
<td>154.0</td>
<td>0.280</td>
<td>4.25</td>
<td>5.25</td>
<td>163</td>
</tr>
<tr>
<td>61A LH Center Field**</td>
<td>0.038</td>
<td>130.0</td>
<td>0.280</td>
<td>12.50</td>
<td>58.75</td>
<td>354</td>
</tr>
<tr>
<td>51C LH Forward Field**</td>
<td>None</td>
<td>45.0</td>
<td>0.280</td>
<td>None</td>
<td>None</td>
<td>354</td>
</tr>
<tr>
<td>51C RH Center Field (prim)**</td>
<td>None</td>
<td>110.0</td>
<td>0.280</td>
<td>3.00</td>
<td>None</td>
<td>275</td>
</tr>
<tr>
<td>51C RH Center Field (sec)**</td>
<td>None</td>
<td>217.0</td>
<td>0.280</td>
<td>3.00</td>
<td>14.30</td>
<td>351</td>
</tr>
<tr>
<td>41D RH Forward Field</td>
<td>0.053</td>
<td>116.0</td>
<td>0.280</td>
<td>--</td>
<td>--</td>
<td>90</td>
</tr>
</tbody>
</table>

*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 erosions and no soot blowby. Other SRM-22 field joints had no blowholes in putty.
Do You Want?

Pretty Pictures

History of O-Ring Damage in Field Joints (Cont)

O-Ring Temp (°F)

<table>
<thead>
<tr>
<th>SRM No.</th>
<th>O-Ring</th>
<th>Temp</th>
<th>O-Ring</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-12</td>
<td>A</td>
<td>66°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>69°</td>
<td>B</td>
<td>69°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>69°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>66°</td>
<td>B</td>
<td>68°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>72°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>67°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>72°</td>
<td>A</td>
<td>72°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>73°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>66°</td>
<td>A</td>
<td>68°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>76°</td>
<td>B</td>
<td>68°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>68°</td>
<td>A</td>
<td>70°</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>68°</td>
<td>B</td>
<td>70°</td>
</tr>
</tbody>
</table>

* No Erosion

Clocking location of leak chase

---

Morton Thiokol Inc.

Information on this page was prepared to support an oral presentation and cannot be considered complete without the oral discussion.

Copyright 2006
Kyle Hailey
Do You Want?

Clean and Clear

History of O-Ring Damage in Field Joints (Cont)

[Graph showing data with X mark]
Imagine Trying to Drive your Car

Would you want your dashboard to look like:

And is updated once and hour

Or would you like it to look …
Or This

![Graph showing active sessions and SQL statements with column headers for Statement and DB Activity, Event and DB Activity, User/Program and Active (as examples).]
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
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)

```
SELECT order_line_date
FROM
    dbo.CUSTOMERS cus
INNER JOIN
da.ORDERS ord
ON ord.id_customer = cus.id
INNER JOIN
da.ORDER_LINES cpl
ON cpl.id_order = ord.id
INNER JOIN
da.PRODUCTS prd1
ON prd1.id = cpl.id_product
INNER JOIN
da.SUPPLIERS sup1
ON sup1.id = prd1.id_supplier
WHERE
    cus.location = 'LONDON' AND
    ord.date_placed BETWEEN '04-JUN-10' AND
    cus.location = 'LONDON' AND
    EXISTS (SELECT NULL
            FROM
                dbo.ALTERNATIVES alt
            INNER JOIN
da.PRODUCTS prd2
ON prd2.id = alt.id_product
INNER JOIN
da.SUPPLIERS sup2
```

Industry Exclusive
DB Optimizer XE Key Features

PROFILE
DB Optimizer XE Key Features

Profile

Tune
DB Optimizer XE Key Features

Profile
Tune
Load Test
DB Optimizer XE Key Features

Profile

Tune

Load Test

SQL IDE
Review of Functionality

Profile
Tune
Load Test
SQL IDE

Only tool on the market with these features integrated
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
One to one
One to many
Many to many
Cartesian

Dangerous

Safe

Cartesian
Simple queries and sub-queries

Simple join

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

Non-correlated sub-query

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

Special case: non correlated sub-query returns one row

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

Diagram:

- Unique index on b.f1
- No unique indexes
- Unique index on Both b.f1 and a.f1

Diagram:

- A
- B
- A
- B
- A
- B

Diagram:

- A
- B

Diagram:

- A
- B

Outer Joins

The diagram illustrates different types of outer joins in SQL, specifically ANSI and ANSI 89 (Oracle) standards. The table and the accompanying code snippets detail how each join type is implemented in SQL:

- **Outer Joins**

  - **Inner Join**
    - ANSI: `english INNER JOIN french using (ordinal_id)`
    - ANSI 89 (Oracle): `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:

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

Scalar Subqueries

```sql
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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>SELECT STATEMENT</td>
</tr>
<tr>
<td>TABLE ACCESS - SOE.ORDER_ITEMS</td>
<td>TABLE ACCESS - SOE.ORDER_ITEMS</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td>NESTED LOOPS</td>
</tr>
<tr>
<td>PARTITION HASH</td>
<td>PARTITION HASH</td>
</tr>
<tr>
<td>TABLE ACCESS - SOE.CUSTOMERS</td>
<td>TABLE ACCESS - SOE.CUSTOMERS</td>
</tr>
<tr>
<td>INDEX - SOE.CUSTOMERS_PK</td>
<td>INDEX - SOE.CUSTOMERS_PK</td>
</tr>
<tr>
<td>TABLE ACCESS - SOE.ORDER_ITEMS</td>
<td>TABLE ACCESS - SOE.ORDER_ITEMS</td>
</tr>
<tr>
<td>INDEX - SOE.ORDER_ITEMS_PK</td>
<td>INDEX - SOE.ORDER_ITEMS_PK</td>
</tr>
</tbody>
</table>
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
When to Tune

1. Machine
   a) CPU
      • Response times skewed
      • 100% CPU might be fine
      • Users wait in queue (run queue) => machine underpowered
   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
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!
1. Include successes
2. Remove Irrelevant
3. Normalize same temp
4. Scale known vs unknown