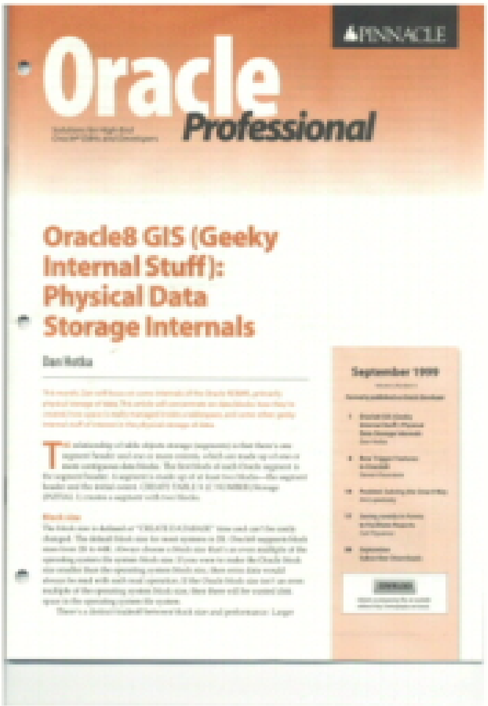
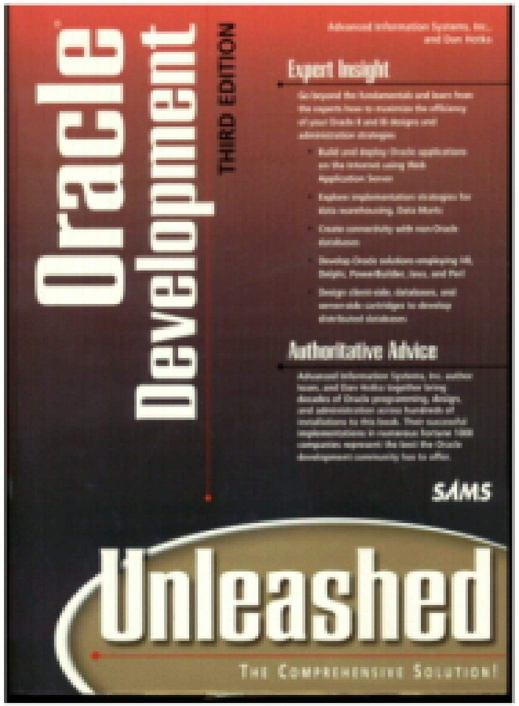
What the SQL is Going On Out Here?

QUEST SOFTWARE

About Dan Hotka:

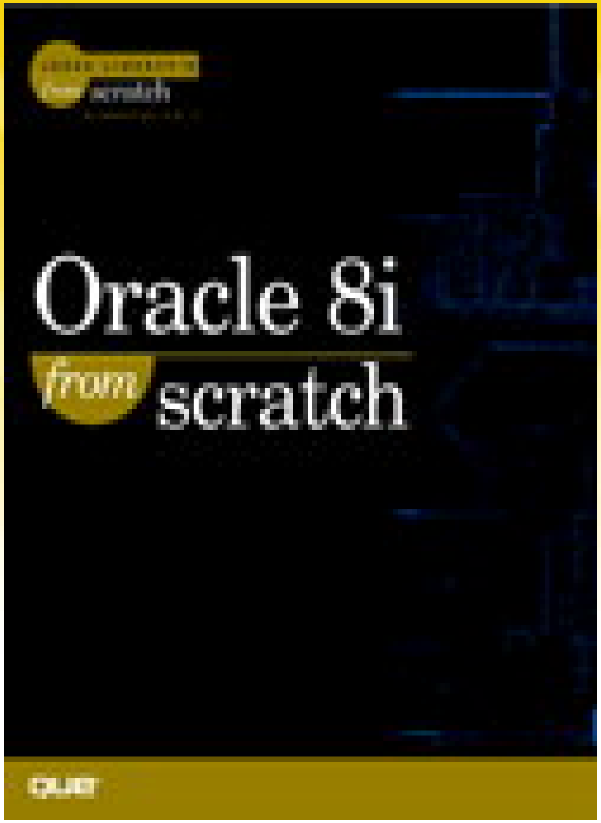
Dan Hotka is a Director of Database Field Operations for Quest Software. He has over 21 years in the computer industry and over 16 years experience with Oracle products. He is an acknowledged Oracle expert with Oracle experience dating back to the Oracle V4.0 days. He has co-authored the popular books Oracle Unleashed, Oracle8 Server Unleashed, Oracle Development Unleashed by SAMS and Special Edition using Oracle8/8i by Que, is frequently published in trade journals, and regularly speaks at Oracle conferences and user groups around the world. Dan can be reached at dhotka@earthlink.net or dhotka@quest.com .


PINNACLE
Oracle Professional
Oracle8 GIS (Geeky Internal Stuff): Physical Data Storage Internals
Dan Hotka
September 1999



Oracle8 Server Unleashed
SAMS
Oracle Second Edition
Concepts • Servers Tools • Products
UNLEASHED
SAMS
ORACLE UNLEASHED
Design, Develop, and Administer Oracle Applications and Databases
SAMS

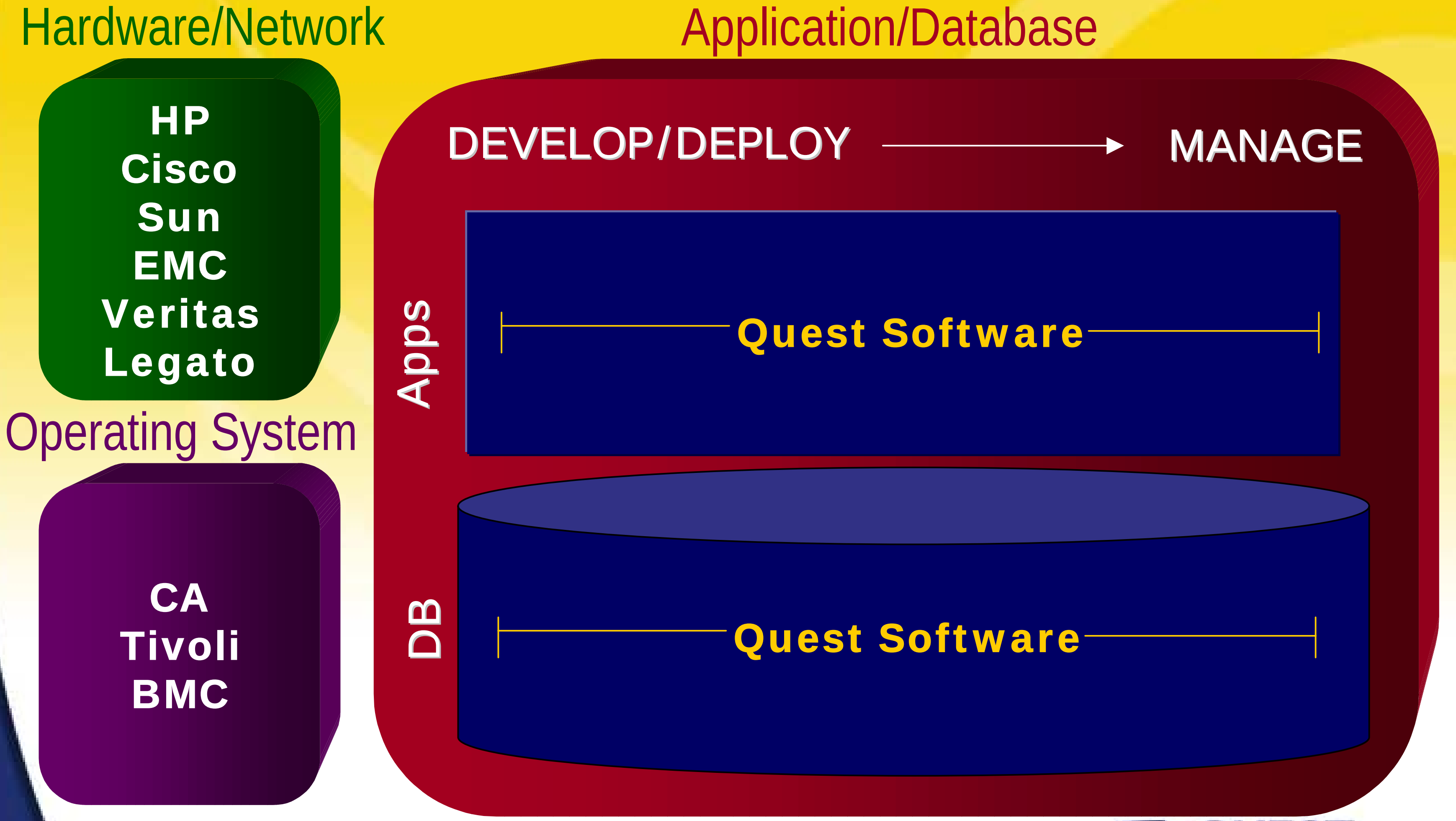
New Book:
ISBN: 0-7897-2369-7
www.amazon.com



Our Mission

To enable today's businesses
to achieve 24x7 operation
of mission-critical applications

The eBusiness Infrastructure Landscape



Hardware/Network

HP
Cisco
Sun
EMC
Veritas
Legato

Operating System

CA
Tivoli
BMC

Application/Database

DEVELOP/DEPLOY → MANAGE

Apps

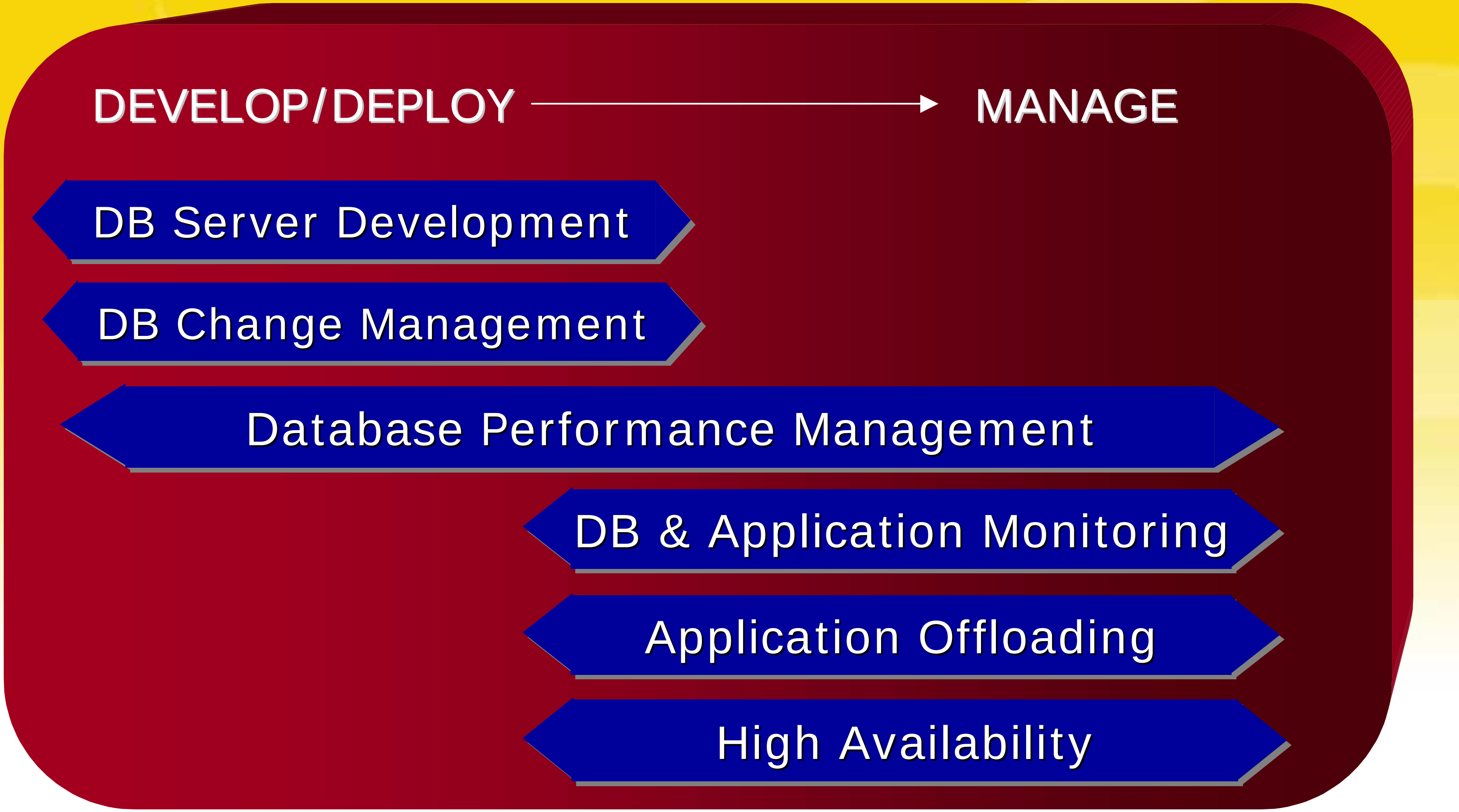
Quest Software

DB

Quest Software

QUEST SOFTWARE

The Complete Quest Solution



DEVELOP/DEPLOY → MANAGE

DB Server Development

DB Change Management

Database Performance Management

DB & Application Monitoring

Application Offloading

High Availability

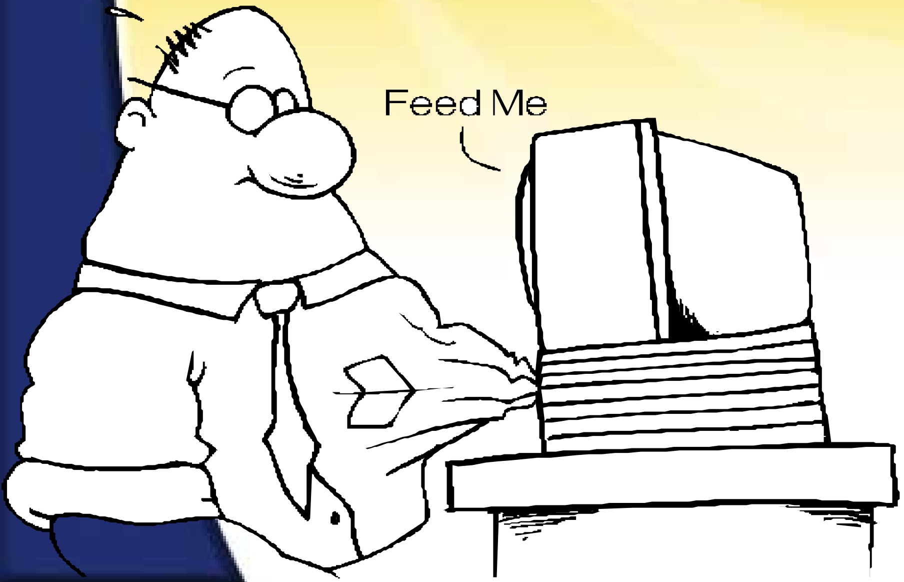
Agenda

- SQL Issues
- Tuning Methodology
- How to find Poorly Performing SQL
- The Oracle Optimizers
- Understanding the Explain Plan
- Oracle Tuning Tools
- SQL Do’s and Don’ts

QUEST SOFTWARE

SQL Issues

- SQL Rage!
  - 20% of SQL is consuming 80% of the resources
  - Poorly performing SQL statements infuriates the staff!



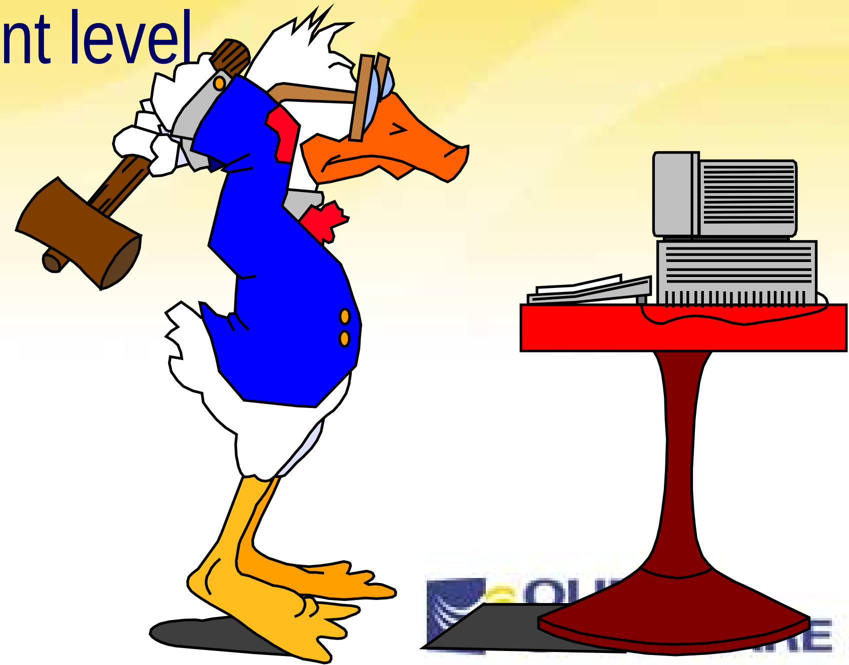
QUEST SOFTWARE

SQL Issues (continued) Where Are These SQL Statements Coming From?

- SQL, while easy to understand conceptually, can be difficult to grasp as it relates to performance.
- Biggest problem, non-trained application programmers and/or end-users are expected to deliver highly tuned SQL.
- High demand for new applications & RAD techniques causes sloppiness.
- Not enough time or resources to examine what the SQL is really doing.

Tuning Methodologies
Top-down & Bottom-up

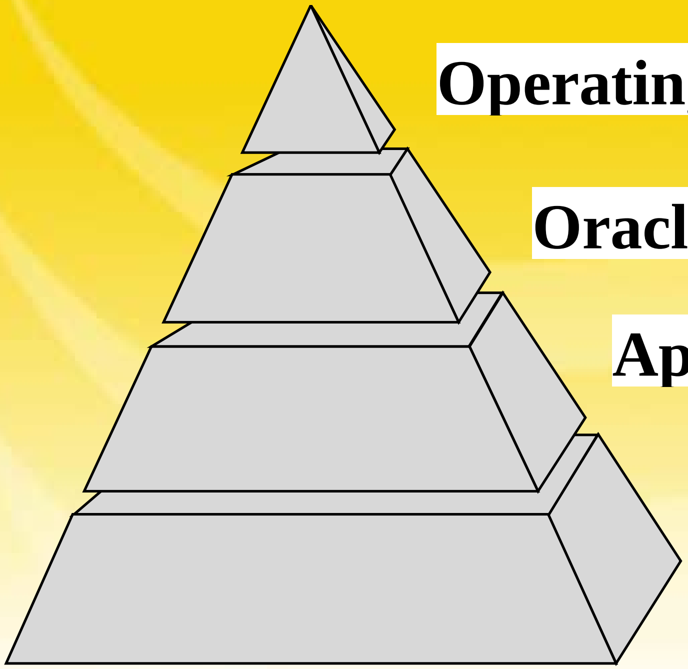
- Top Down
  - Reacting to problem SQL
  - End users using 3rd party tools
  - No tuning at the development level
  - No review of code



Tuning Methodologies Top-down & Bottom-up

- Bottom-up
  - Build it right the first time
  - Know the data and design
  - Quickly evaluate alternatives - rule, first, or all
  - Add hints based on database stats
  - Execute & compare
  - Determine which plans are affected by a database change

Tuning Methodology



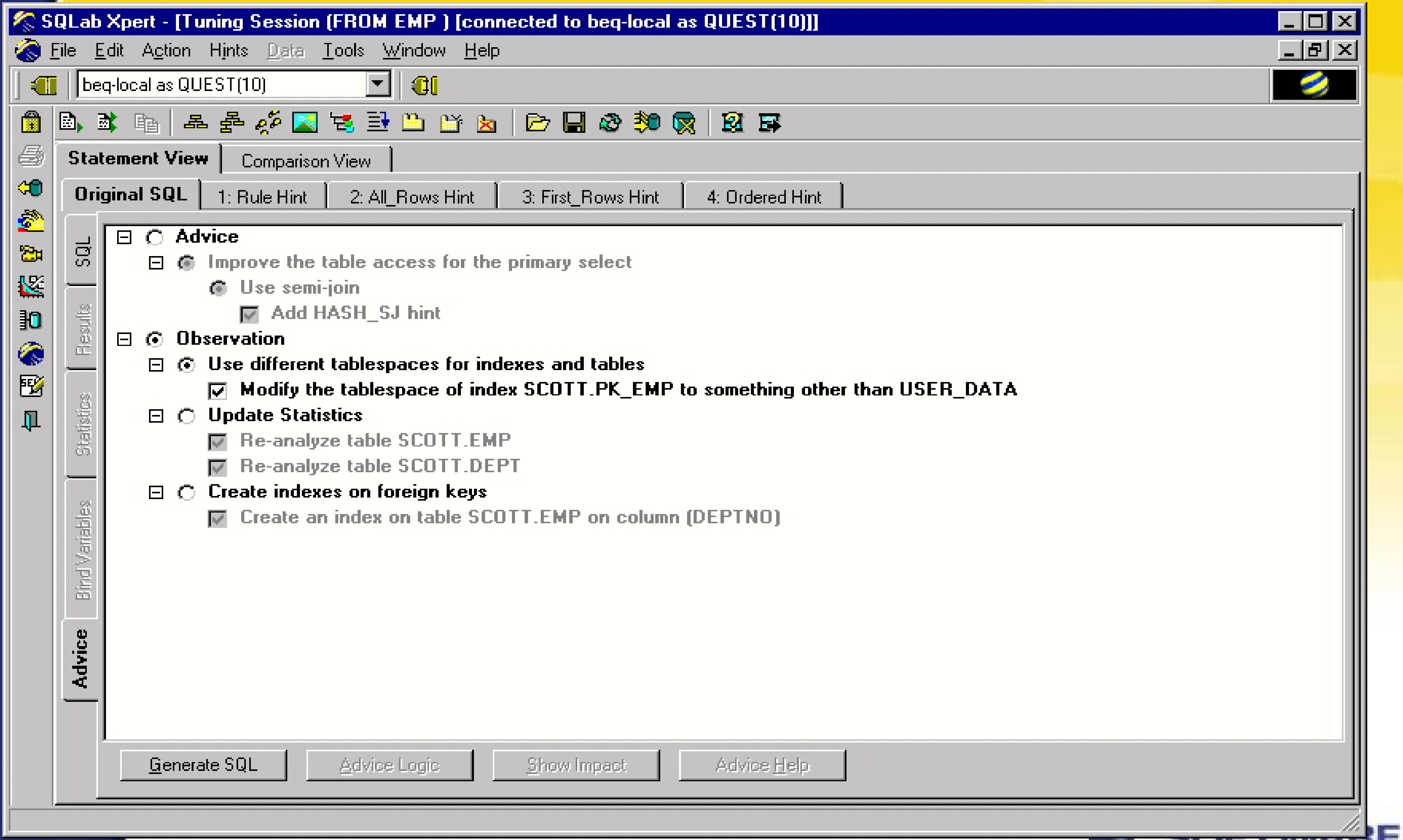
Operating System

Oracle Database

App Design

SQL Code

Tuning Methodology In Action

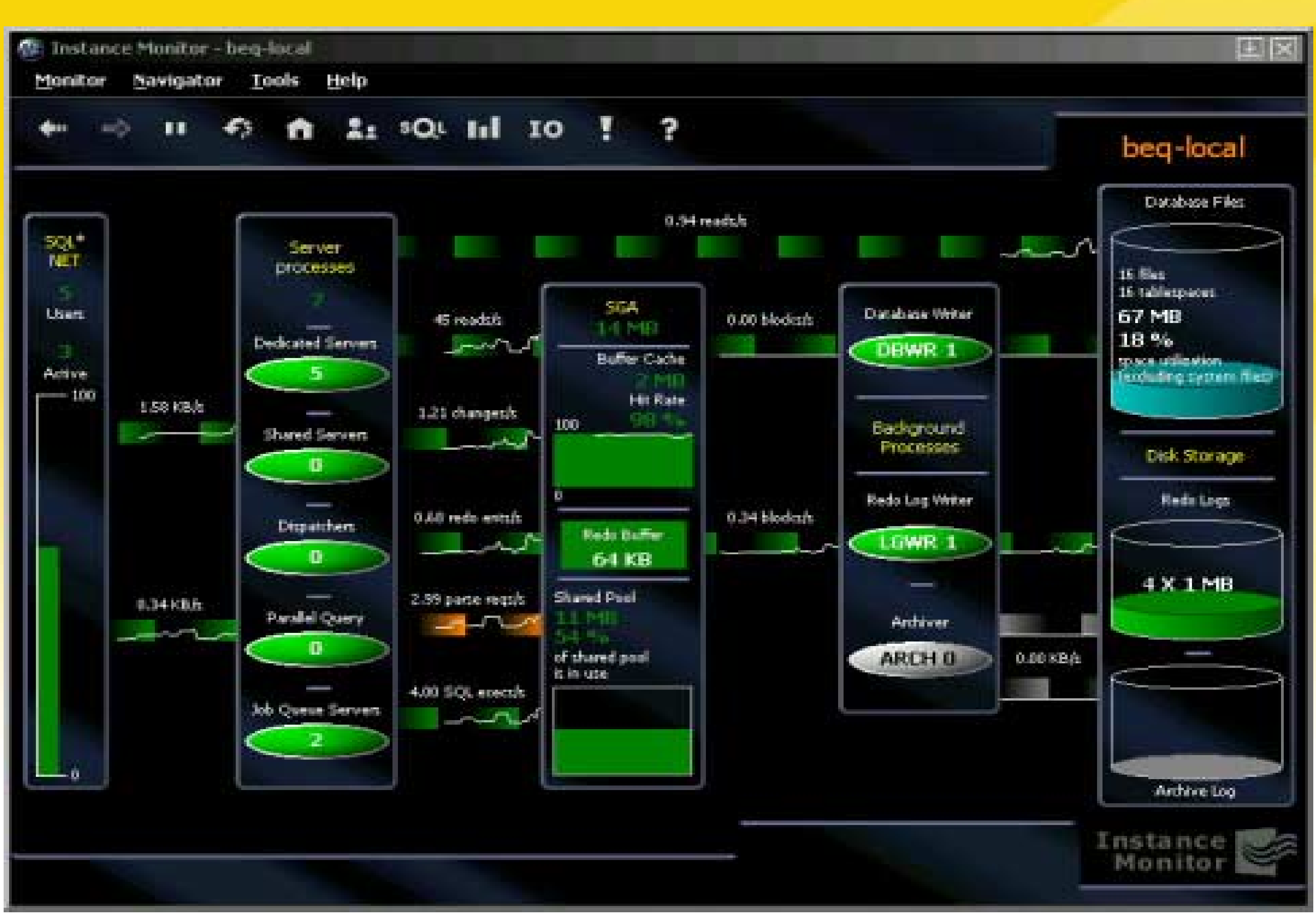


- ○ Advice
  - ● Improve the table access for the primary select
    - ● Use semi-join
      - ☑ Add HASH_SJ hint
- ● Observation
  - ● Use different tablespaces for indexes and tables
    - ☑ Modify the tablespace of index SCOTT.PK_EMP to something other than USER_DATA
  - ○ Update Statistics
    - ☑ Re-analyze table SCOTT.EMP
    - ☑ Re-analyze table SCOTT.DEPT
  - ○ Create indexes on foreign keys
    - ☑ Create an index on table SCOTT.EMP on column (DEPTNO)

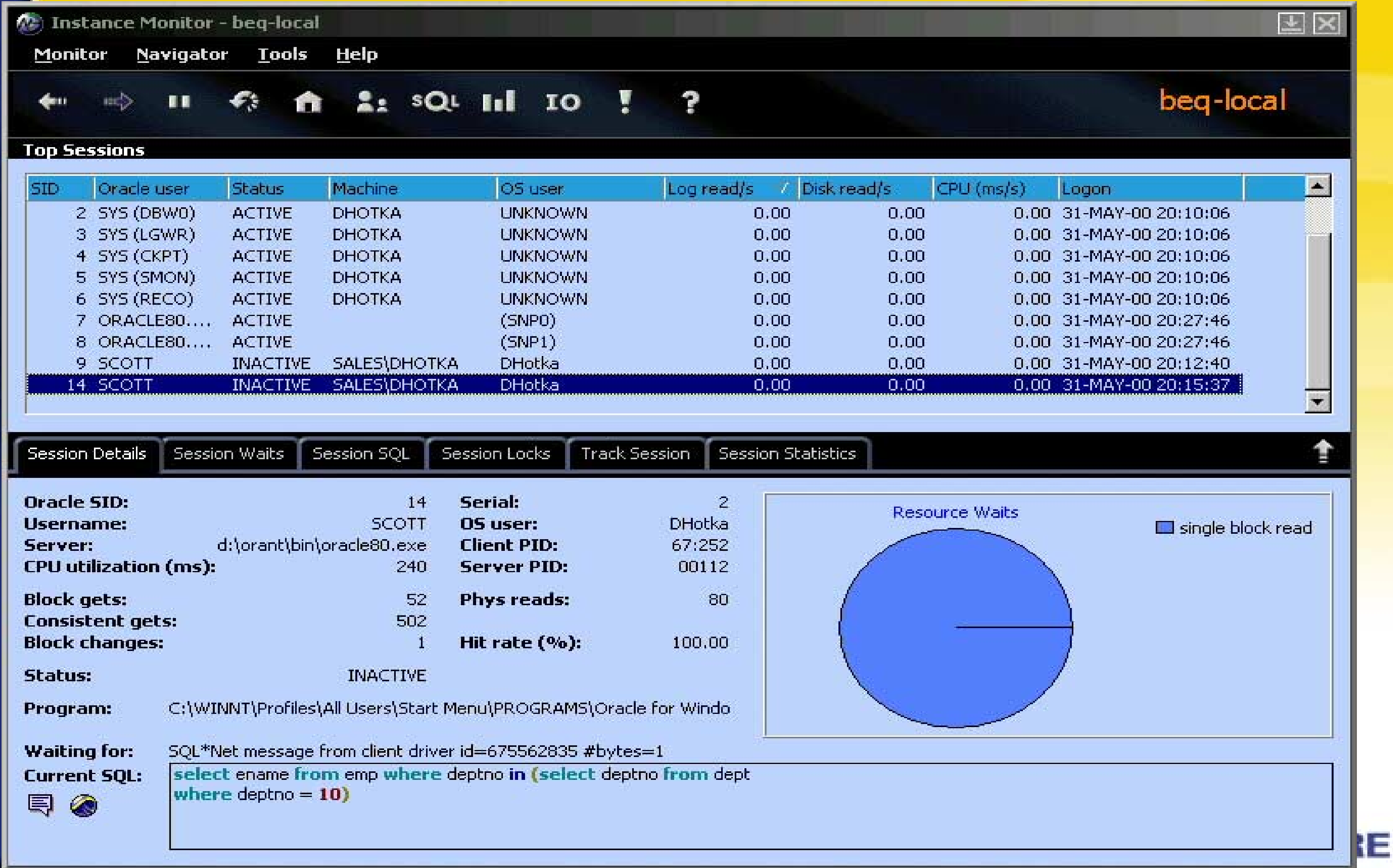
How to find Poorly Performing SQL

- Monitoring
- Scripts
  - Tim Gorman www.sagelogix.com
  - Top Offensive SQL Statements
    - Oracle Professional June 2000
- Tools
- Luck…

How to find Poorly Performing SQL



How to find Poorly Performing SQL



SID	Oracle user	Status	Machine	OS user	Log read/s	Disk read/s	CPU (ms/s)	Logon
2	SYS (DBW0)	ACTIVE	DHOTKA	UNKNOWN	0.00	0.00	0.00	31-MAY-00 20:10:06
3	SYS (LGWR)	ACTIVE	DHOTKA	UNKNOWN	0.00	0.00	0.00	31-MAY-00 20:10:06
4	SYS (CKPT)	ACTIVE	DHOTKA	UNKNOWN	0.00	0.00	0.00	31-MAY-00 20:10:06
5	SYS (SMON)	ACTIVE	DHOTKA	UNKNOWN	0.00	0.00	0.00	31-MAY-00 20:10:06
6	SYS (RECO)	ACTIVE	DHOTKA	UNKNOWN	0.00	0.00	0.00	31-MAY-00 20:10:06
7	ORACLE80....	ACTIVE		(SNP0)	0.00	0.00	0.00	31-MAY-00 20:27:46
8	ORACLE80....	ACTIVE		(SNP1)	0.00	0.00	0.00	31-MAY-00 20:27:46
9	SCOTT	INACTIVE	SALES\DHOTKA	DHotka	0.00	0.00	0.00	31-MAY-00 20:12:40
14	SCOTT	INACTIVE	SALES\DHOTKA	DHotka	0.00	0.00	0.00	31-MAY-00 20:15:37

Oracle SID:	14	**Serial:**	2
Username:	SCOTT	**OS user:**	DHotka
Server:	d:\orant\bin\oracle80.exe	**Client PID:**	67:252
CPU utilization (ms):	240	**Server PID:**	00112
Block gets:	52	**Phys reads:**	80
Consistent gets:	502		
Block changes:	1	**Hit rate (%):**	100.00
Status:	INACTIVE		

Program: C:\WINNT\Profiles\All Users\Start Menu\PROGRAMS\Oracle for Windo

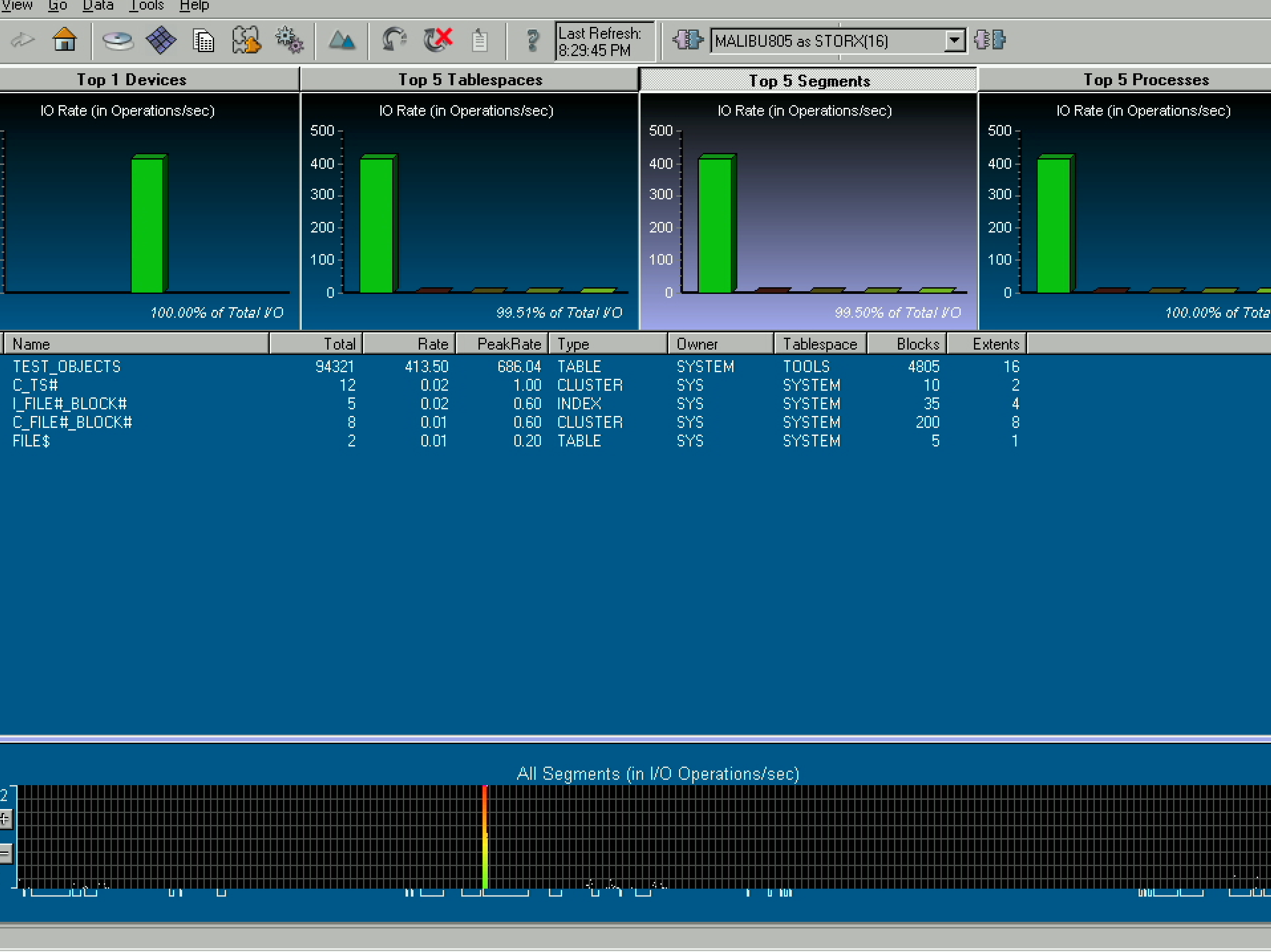
Waiting for: SQL*Net message from client driver id=675562835 #bytes=1

Current SQL:

```
select ename from emp where deptno in (select deptno from dept
where deptno = 10)
```

How to find Poorly Performing SQL

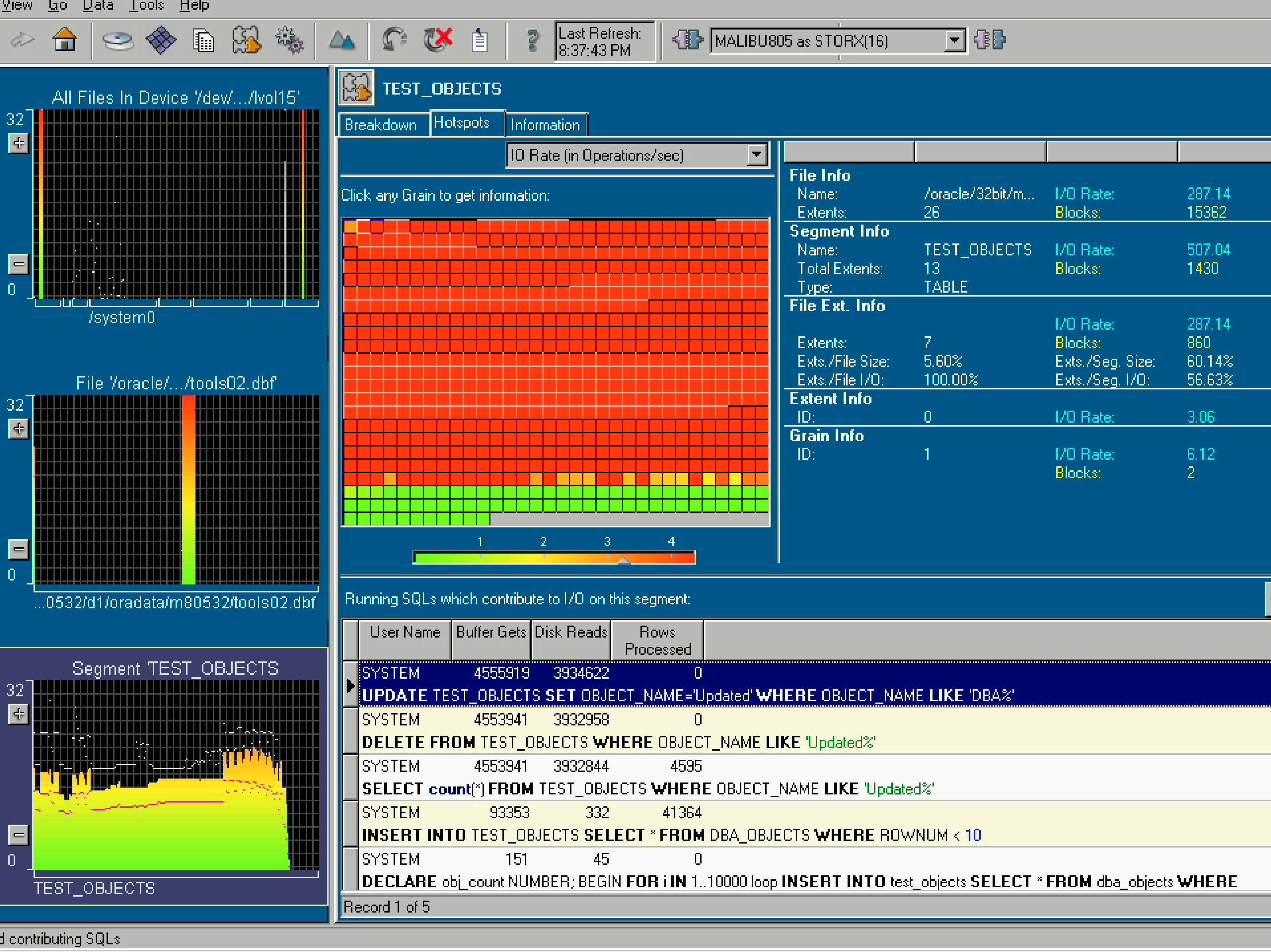
- Diagnoses Source of I/O Bottlenecks:
  - Poor SQL
  - Poor Layout
- Quickly identify disk hot spots
- Drill down to actual SQL statements
- Identify frequently accessed objects
- Soon: History, I/O Trends, Capacity Planning



Last Refresh: 8:29:45 PM

MALIBU805 as STORX(16)

Name	Total	Rate	PeakRate	Type	Owner	Tablespace	Blocks	Extents
TEST_OBJECTS	94321	413.50	686.04	TABLE	SYSTEM	TOOLS	4805	16
C_TS#	12	0.02	1.00	CLUSTER	SYS	SYSTEM	10	2
I_FILE#_BLOCK#	5	0.02	0.60	INDEX	SYS	SYSTEM	35	4
C_FILE#_BLOCK#	8	0.01	0.60	CLUSTER	SYS	SYSTEM	200	8
FILE$	2	0.01	0.20	TABLE	SYS	SYSTEM	5	1



Click any Grain to get information:

File Info

Name:	/oracle/32bit/m...	I/O Rate:	287.14
Extents:	26	Blocks:	15362

Segment Info

Name:	TEST_OBJECTS	I/O Rate:	507.04
Total Extents:	13	Blocks:	1430
Type:	TABLE		

File Ext. Info

		I/O Rate:	287.14
Extents:	7	Blocks:	860
Exts./File Size:	5.60%	Exts./Seg. Size:	60.14%
Exts./File I/O:	100.00%	Exts./Seg. I/O:	56.63%

Extent Info

ID:	0	I/O Rate:	3.06

Grain Info

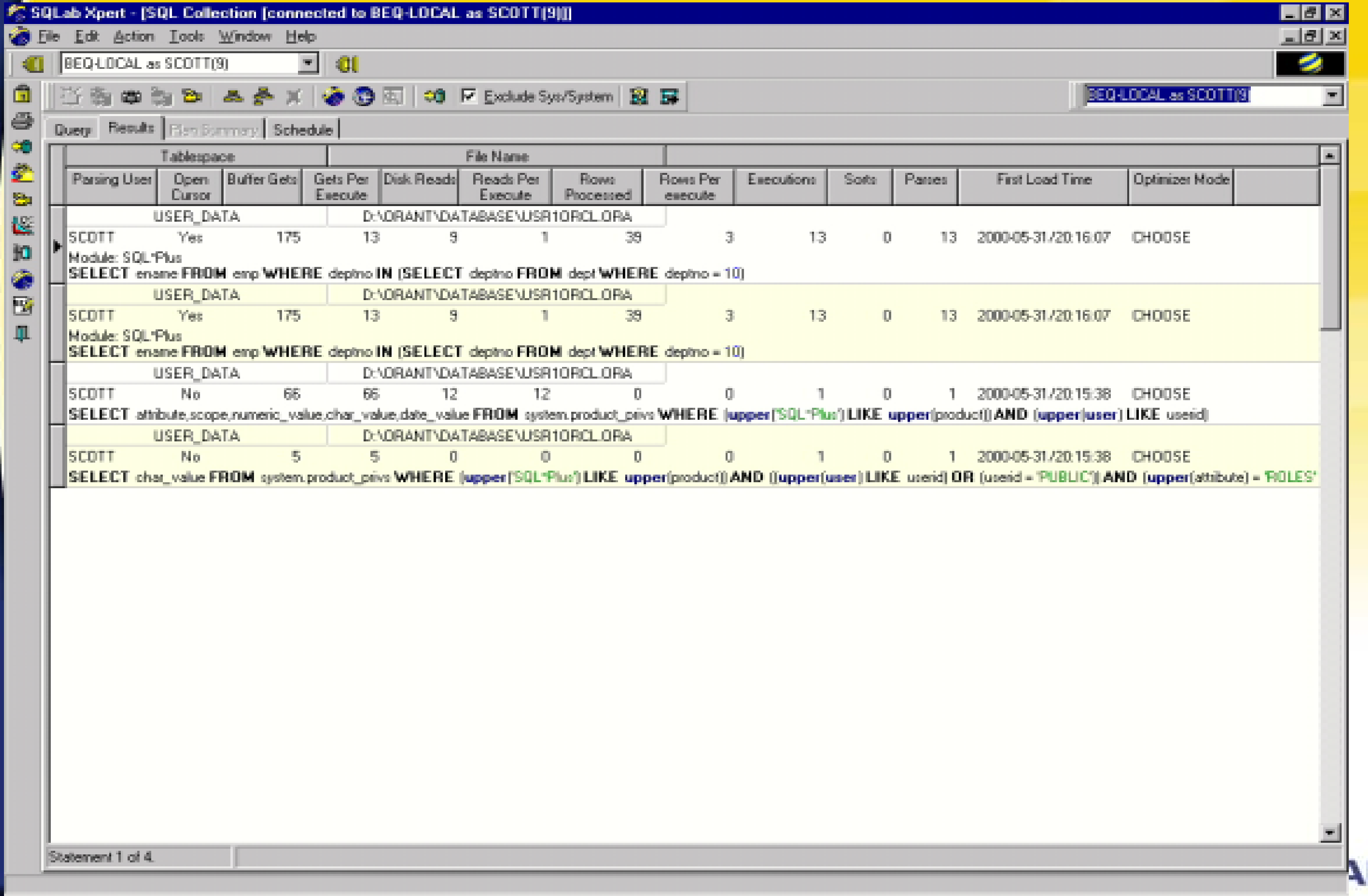
ID:	1	I/O Rate:	6.12
		Blocks:	2

Running SQLs which contribute to I/O on this segment:

User Name	Buffer Gets	Disk Reads	Rows Processed
SYSTEM	4555919	3934622	0
UPDATE TEST_OBJECTS SET OBJECT_NAME='Updated' WHERE OBJECT_NAME LIKE 'DBA%'			
SYSTEM	4553941	3932958	0
DELETE FROM TEST_OBJECTS WHERE OBJECT_NAME LIKE 'Updated%'			
SYSTEM	4553941	3932844	4595
SELECT count(*) FROM TEST_OBJECTS WHERE OBJECT_NAME LIKE 'Updated%'			
SYSTEM	93353	332	41364
INSERT INTO TEST_OBJECTS SELECT * FROM DBA_OBJECTS WHERE ROWNUM < 10			
SYSTEM	151	45	0
DECLARE obj_count NUMBER; BEGIN FOR i IN 1..10000 loop INSERT INTO test_objects SELECT * FROM dba_objects WHERE			

Record 1 of 5

How to find Poorly Performing SQL



Tablespace			File Name									
Parsing User	Open Cursor	Buffer Gets	Gets Per Execute	Disk Reads	Reads Per Execute	Rows Processed	Rows Per execute	Executions	Sorts	Parses	First Load Time	Optimizer Mode
USER_DATA			D:\ORANT\DATABASE\USR1ORCL.ORA									
SCOTT	Yes	175	13	9	1	39	3	13	0	13	2000-05-31/20:16:07	CHOOSE
Module: SQL*Plus												
SELECT ename FROM emp WHERE deptno IN (SELECT deptno FROM dept WHERE deptno = 10)												
USER_DATA			D:\ORANT\DATABASE\USR1ORCL.ORA									
SCOTT	Yes	175	13	9	1	39	3	13	0	13	2000-05-31/20:16:07	CHOOSE
Module: SQL*Plus												
SELECT ename FROM emp WHERE deptno IN (SELECT deptno FROM dept WHERE deptno = 10)												
USER_DATA			D:\ORANT\DATABASE\USR1ORCL.ORA									
SCOTT	No	66	66	12	12	0	0	1	0	1	2000-05-31/20:15:38	CHOOSE
SELECT attribute,scope,numeric_value,char_value,date_value FROM system.product_privs WHERE (upper('SQL*Plus') LIKE upper(product)) AND (upper(user) LIKE userid)												
USER_DATA			D:\ORANT\DATABASE\USR1ORCL.ORA									
SCOTT	No	5	5	0	0	0	0	1	0	1	2000-05-31/20:15:38	CHOOSE
SELECT char_value FROM system.product_privs WHERE (upper('SQL*Plus') LIKE upper(product)) AND ((upper(user) LIKE userid) OR (userid = 'PUBLIC')) AND (upper(attribute) = 'ROLES'												

Statement 1 of 4.

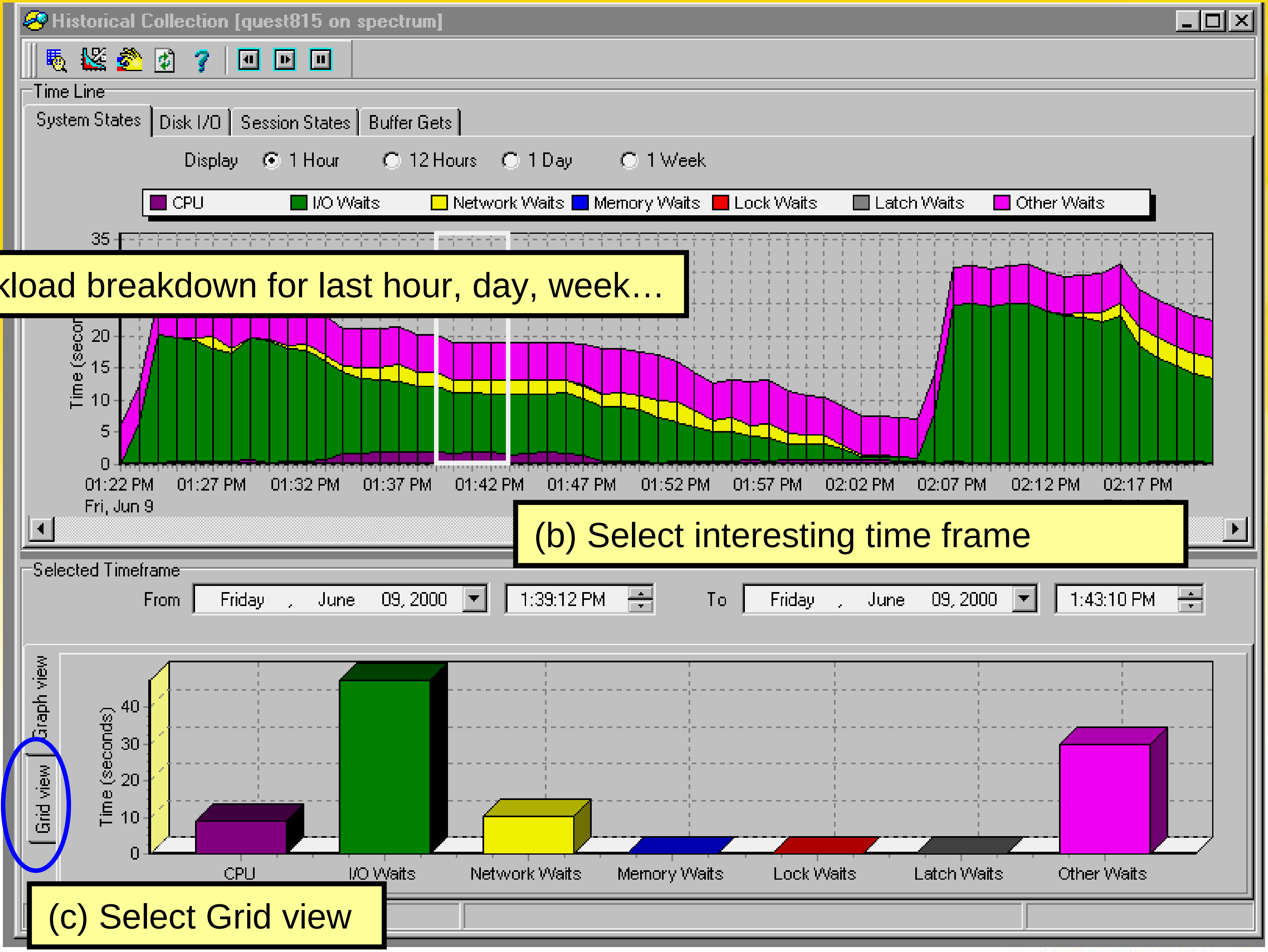
How to find Poorly Performing SQL

```
/**********************************************************************
  * File:top_stmt2.sql
  * Type:SQL*Plus script
  * Author:    Tim Gorman (SageLogix, Inc.)
  * Date:04-Oct-99
  *
  * Description:
  *DDL script to create the TOP_STMT2 stored procedure.
  *
  * Modifications:
 *********************************************************************/

create or replace procedure top_stmt2
(
  in_top_count in number default 20,
  in_max_disk_reads in number default 10000,
  in_max_buffer_gets in number default 100000
) is
  --
  cursor get_top_stmts(in_dr in number, in_bg in number)
  is
  select/*+ rule */
         substr(sql_text, 1, 60) sql_text,
         min(address) address,
         sum(abs(disk_reads)) disk_reads,
         sum(abs(buffer_gets)) buffer_gets,
         sum(abs(sorts)) sorts,
         sum(abs(executions)) executions,
         sum(abs(loads)) loads,
         count(*) cnt,
        ((sum(abs(disk_reads))*100)+sum(abs(buffer_gets)))/1000
factor
  from  sys.v_$sqlarea
  group by substr(sql_text, 1, 60)
  having sum(abs(disk_reads)) > in_dr
  and    sum(abs(buffer_gets)) > in_bg
  order by factor desc;
```

Tuning in Production: Recon

- Historical 24x7 information
  - From a minute to a full week--intuitive navigation
  - Detailed info on SQL, users, programs and clients
  - Detailed resource consumption and resource waits
  - Analysis graphs and direct entry into SQL tuning
- Live Diagnostics
  - What sessions are active/inactive and what are they doing NOW
  - Trace session with full SQL history and detailed stats per SQL
- Zero Oracle overhead, negligible server overhead

Historical Analysis and Diagnostics



Historical Collection [quest815 on spectrum]

Time Line

Display ● 1 Hour ○ 12 Hours ○ 1 Day ○ 1 Week

Selected Timeframe

From Friday , June 09, 2000 1:39:12 PM To Friday , June 09, 2000 1:43:10 PM

(a) View workload breakdown for last hour, day, week…

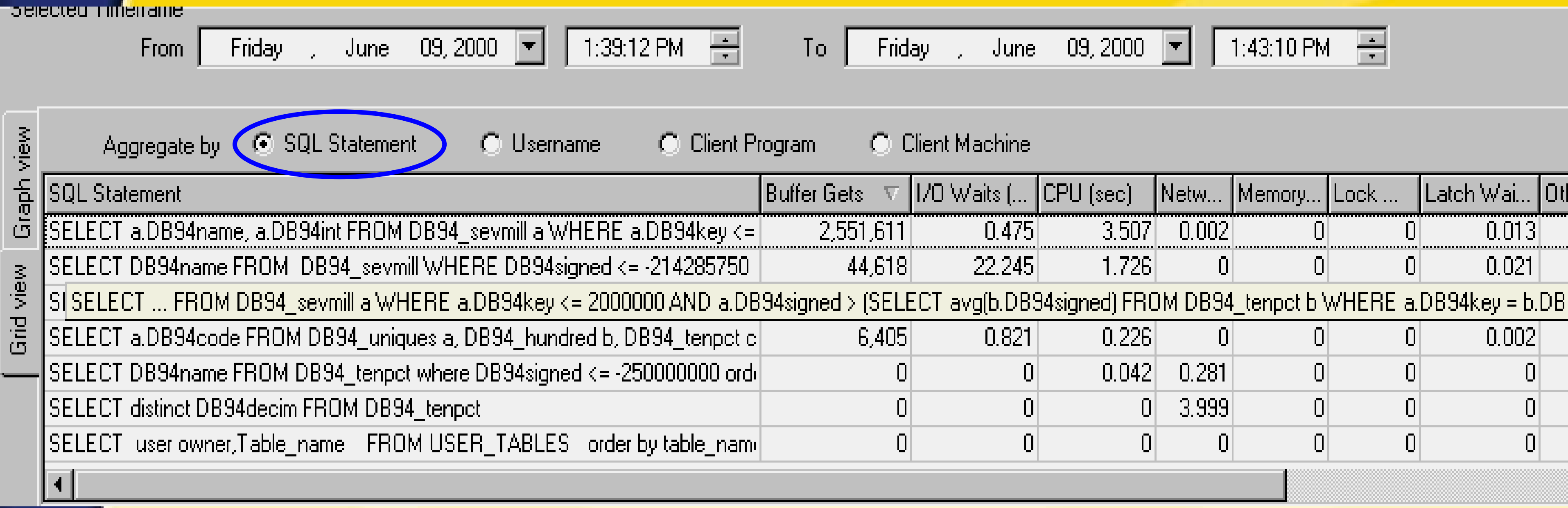
(b) Select interesting time frame

(c) Select Grid view

QUEST SOFTWARE

Historical Analysis and Diagnostics

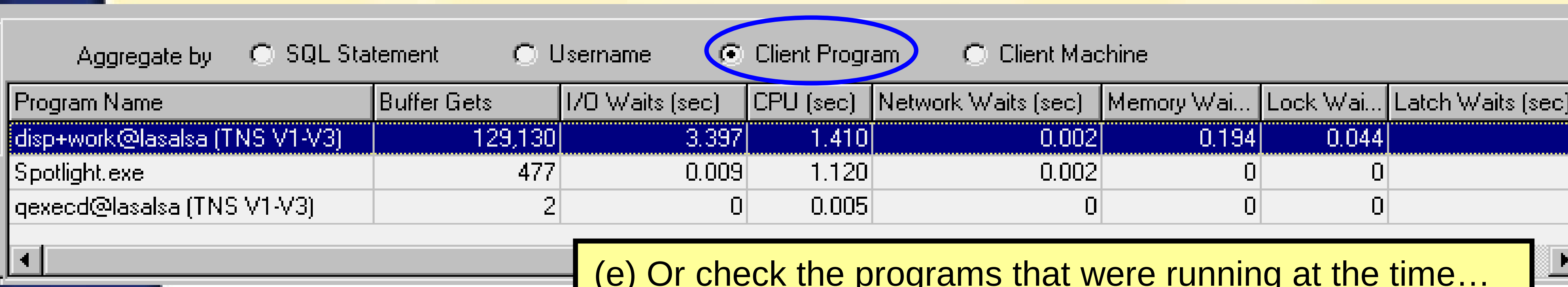
(d) View the SQL statements using resources at 1:39PM…



From Friday , June 09, 2000 ▼ 1:39:12 PM To Friday , June 09, 2000 ▼ 1:43:10 PM

Aggregate by ● SQL Statement ○ Username ○ Client Program ○ Client Machine

SQL Statement	Buffer Gets ▽	I/O Waits (...	CPU (sec)	Netw...	Memory...	Lock ...	Latch Wai...	Oth
SELECT a.DB94name, a.DB94int FROM DB94_sevmill a WHERE a.DB94key <=	2,551,611	0.475	3.507	0.002	0	0	0.013	
SELECT DB94name FROM DB94_sevmill WHERE DB94signed <= -214285750	44,618	22.245	1.726	0	0	0	0.021	
SI SELECT ... FROM DB94_sevmill a WHERE a.DB94key <= 2000000 AND a.DB94signed > (SELECT avg(b.DB94signed) FROM DB94_tenpct b WHERE a.DB94key = b.DB								
SELECT a.DB94code FROM DB94_uniques a, DB94_hundred b, DB94_tenpct c	6,405	0.821	0.226	0	0	0	0.002	
SELECT DB94name FROM DB94_tenpct where DB94signed <= -250000000 ord	0	0	0.042	0.281	0	0	0	
SELECT distinct DB94decim FROM DB94_tenpct	0	0	0	3.999	0	0	0	
SELECT user owner,Table_name FROM USER_TABLES order by table_nam	0	0	0	0	0	0	0	



Aggregate by ○ SQL Statement ○ Username ● Client Program ○ Client Machine

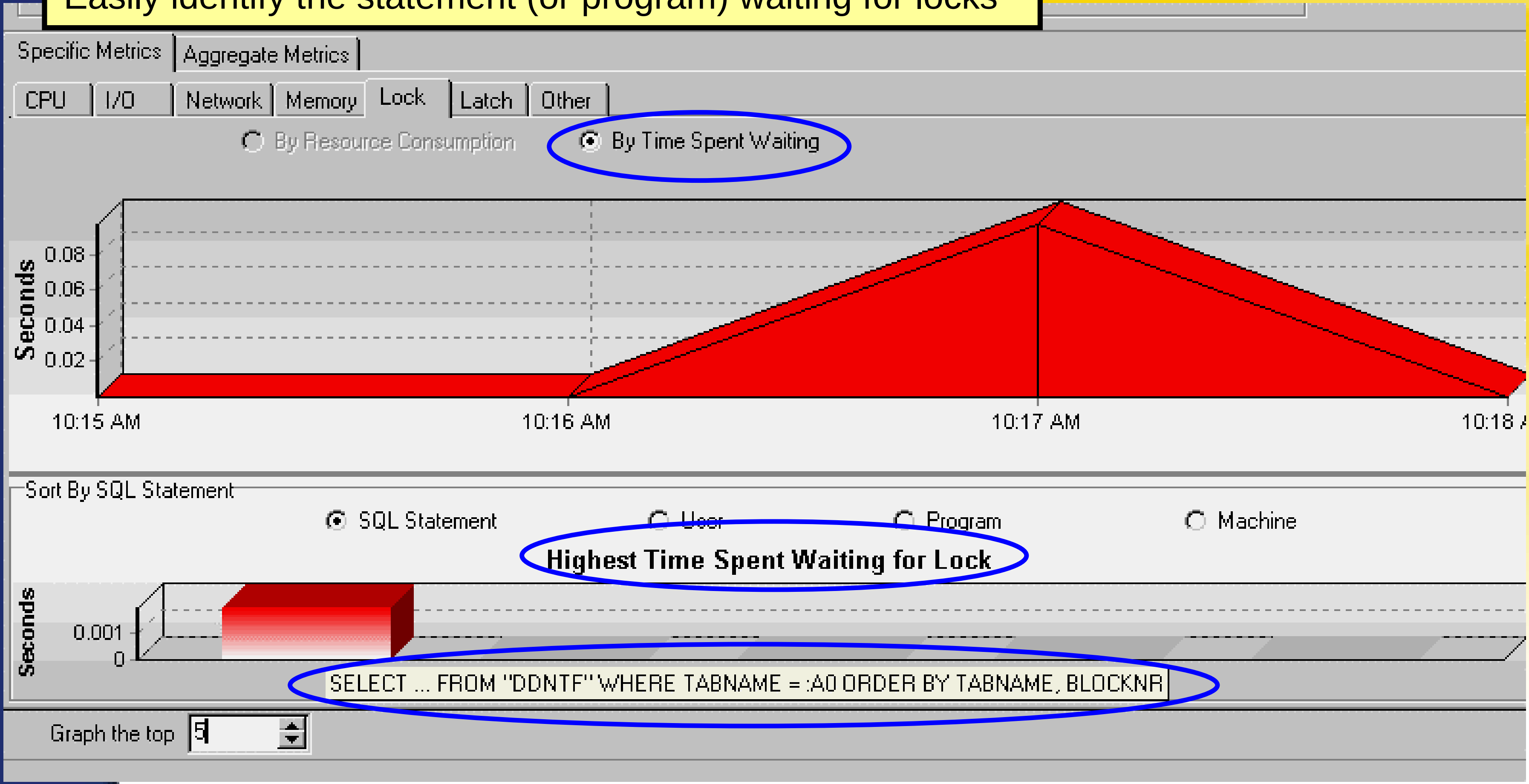
Program Name	Buffer Gets	I/O Waits (sec)	CPU (sec)	Network Waits (sec)	Memory Wai...	Lock Wai...	Latch Waits (sec)
disp+work@lasalsa (TNS V1-V3)	129,130	3.397	1.410	0.002	0.194	0.044	
Spotlight.exe	477	0.009	1.120	0.002	0	0	
qexecd@lasalsa (TNS V1-V3)	2	0	0.005	0	0	0	

(e) Or check the programs that were running at the time…

QUEST SOFTWARE

Historical Analysis and Diagnostics: Drilldowns

Easily identify the statement (or program) waiting for locks



Specific Metrics | Aggregate Metrics

CPU | I/O | Network | Memory | Lock | Latch | Other

○ By Resource Consumption ● By Time Spent Waiting

Sort By SQL Statement

● SQL Statement ○ User ○ Program ○ Machine

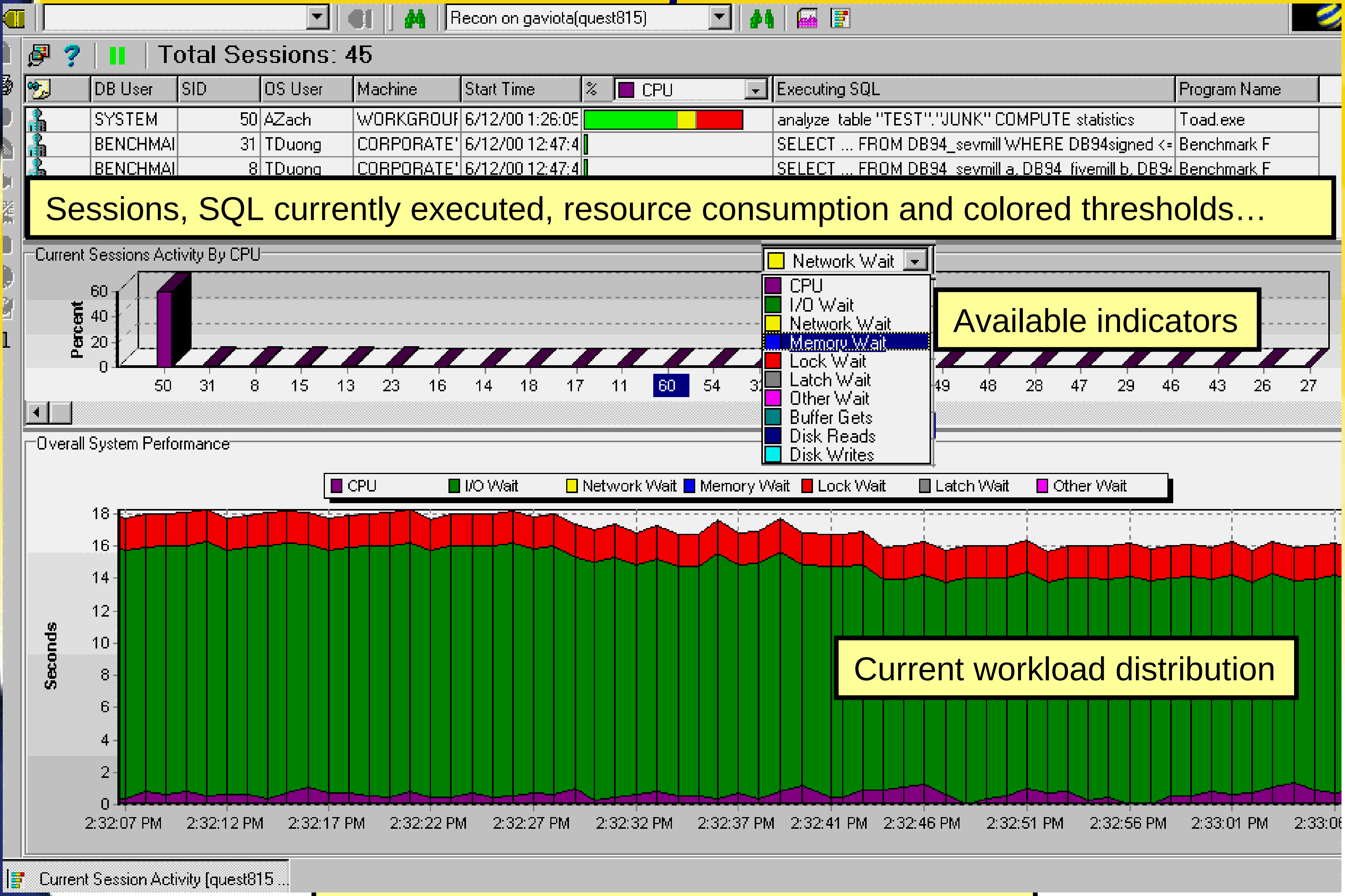
Highest Time Spent Waiting for Lock

SELECT ... FROM "DDNTF" WHERE TABNAME = :A0 ORDER BY TABNAME, BLOCKNR

Graph the top 5

The same is available for CPU, I/O, Network, Latches and Other Waits

Live Diagnostics



Recon on gaviota[quest815]

Total Sessions: 45

	DB User	SID	OS User	Machine	Start Time	%	CPU	Executing SQL	Program Name
	SYSTEM	50	AZach	WORKGROUI	6/12/00 1:26:05			analyze table "TEST"."JUNK" COMPUTE statistics	Toad.exe
	BENCHMAI	31	TDuong	CORPORATE'	6/12/00 12:47:4			SELECT ... FROM DB94_sevmill WHERE DB94signed <=	Benchmark F
	BENCHMAI	8	TDuong	CORPORATE'	6/12/00 12:47:4			SELECT ... FROM DB94_sevmill a, DB94_fivemill b, DB9	Benchmark F

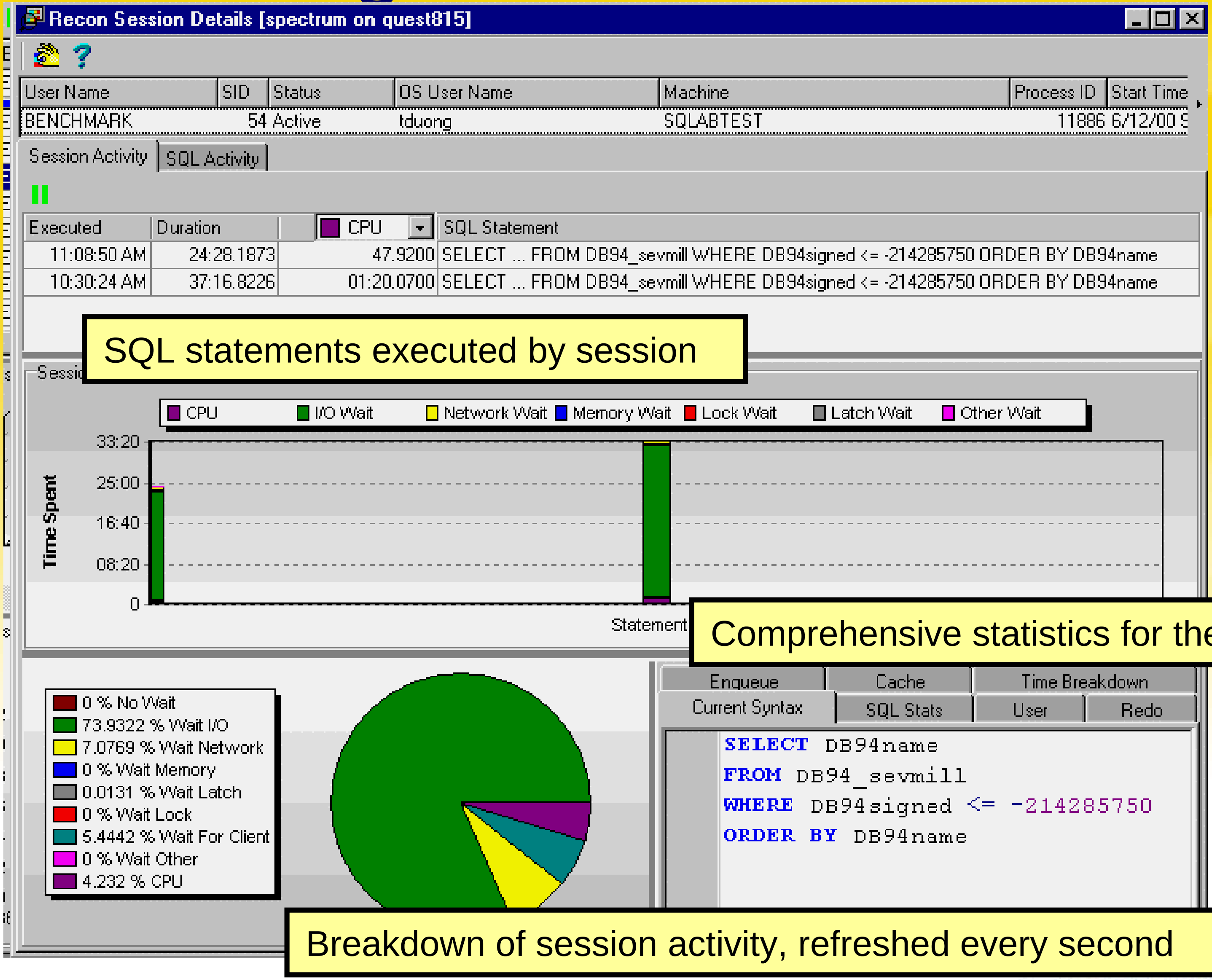
Sessions, SQL currently executed, resource consumption and colored thresholds…

Available indicators

Current workload distribution

No Oracle connection, Zero Overhead

Live Diagnostics Drilldowns



Recon Session Details [spectrum on quest815]

User Name	SID	Status	OS User Name	Machine	Process ID	Start Time
BENCHMARK	54	Active	tduong	SQLABTEST	11886	6/12/00 9

Executed	Duration	CPU	SQL Statement
11:08:50 AM	24:28.1873	47.9200	SELECT ... FROM DB94_sevmill WHERE DB94signed <= -214285750 ORDER BY DB94name
10:30:24 AM	37:16.8226	01:20.0700	SELECT ... FROM DB94_sevmill WHERE DB94signed <= -214285750 ORDER BY DB94name

SQL statements executed by session

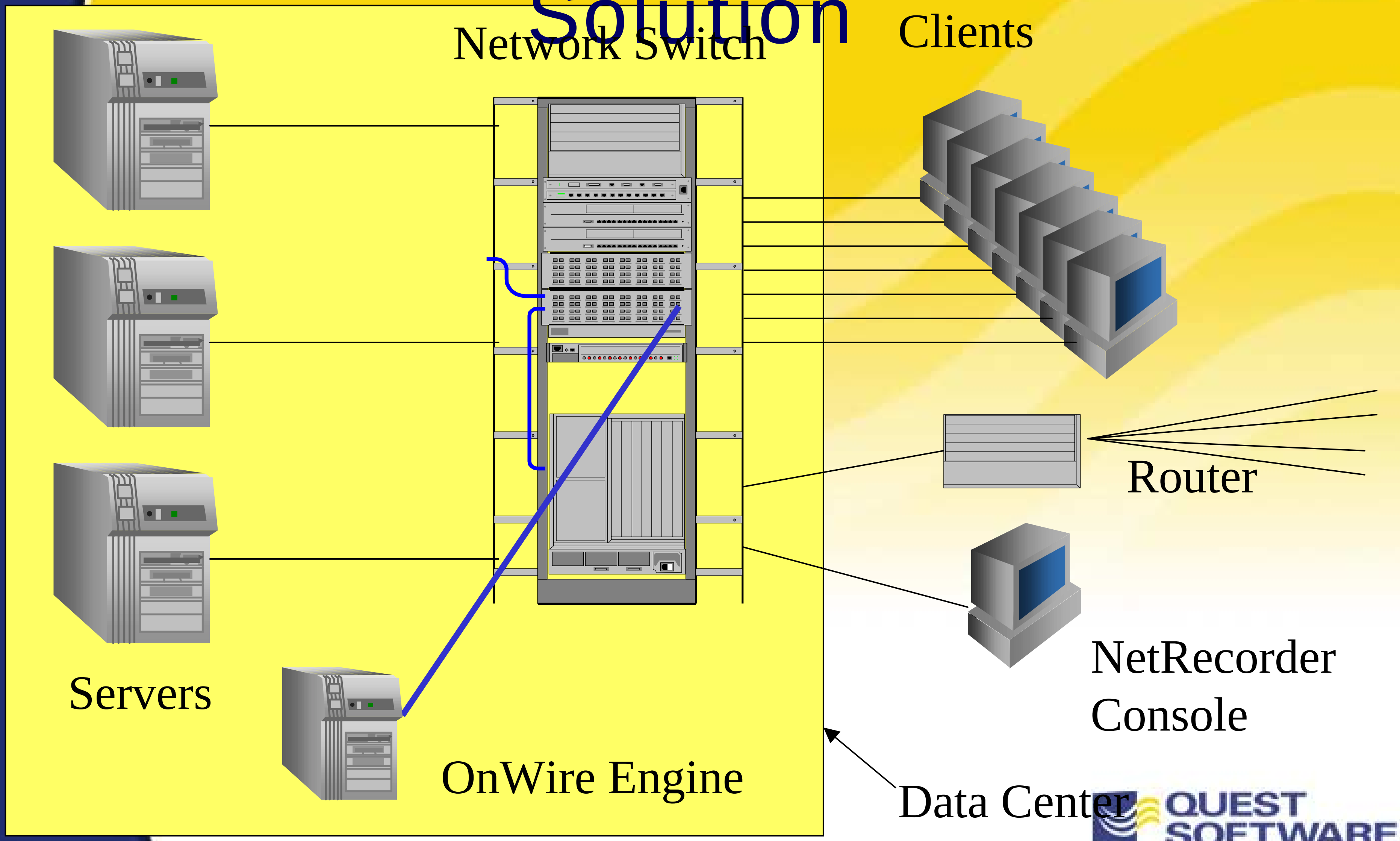
Comprehensive statistics for the session

🟥 0 % No Wait
🟩 73.9322 % Wait I/O
🟨 7.0769 % Wait Network
🟦 0 % Wait Memory
⬜ 0.0131 % Wait Latch
🟥 0 % Wait Lock
▨ 5.4442 % Wait For Client
🟪 0 % Wait Other
🟪 4.232 % CPU

```
SELECT DB94name
FROM DB94_sevmill
WHERE DB94signed <= -214285750
ORDER BY DB94name
```

Breakdown of session activity, refreshed every second

OnWire's Non-Intrusive Solution



HostRecorder Collector

- Implemented as a Kernel Streams driver
  - NO Overhead
    - No context switching
    - No expensive User mode processing
    - No polling
    - No contention
    - No agents
  - 10 minute install (< 25K size)
  - No network overhead

OnWire 3.1 Features

- HostRecorder
- Built-in Forms End-User Transactions
- Concurrent Manager Support
- Advanced Locking Diagnostics
- Support for Web and PLSQL Table Arrays
- Normalized Performance Metrics
- Identify and classify resource utilization
- Full Lifecycle Tuning Support w/SQLAB

The Oracle Optimizers

- Oracle Optimizer gives you choices:
  - Rule-based
    - **Based on a set of rules (Index existence, SQL coding)**
    - **Does NOT consider object statistics**
  - Cost-based
    - **Uses object statistics (from ANALYZE command)**
    - **User has more control in tuning**
    - **2 goals:**
      - First row: response time for interactive apps (OLTP)
      - All rows: throughput for batch processing (DSS)
    - **Makes Assumptions**
  - Hints
    - **Cost Based 'suggestions', not always used by Oracle!**
    - **Can be specified in combinations**

The Oracle Optimizers
Selecting a Mode

- Database Level
  - init.ora OPTIMIZER_MODE parameter
  - RULE, COST, or CHOOSE* (default)
- Session Level
  - ALTER SESSION SET OPTIMIZER_GOAL= < >
    - < > = RULE, FIRST_ROWS, ALL_ROWS, CHOOSE
- SQL Level
  - HINT RULE, FIRST_ROWS, ALL_ROWS

The Oracle Optimizers Rule-based Optimizer

Rank	Where Clause Rule
1	ROWID = constant
2	unique indexed column = constant
3	entire unique concatenated index = constant
4	entire cluster key = cluster key of object in same cluster
5	entire cluster key = constant
6	entire nonunique concatenated index = constant
7	nonunique index = constant
8	entire noncompressed concatenated index >= constant
9	entire compressed concatenated index >= constant
10	partial but leading columns of noncompressed concatenated index
11	partial but leading columns of compressed concatenated index
12	unique indexed column using the SQL statement BETWEEN or LIKE options
13	nonunique indexed column using the SQL statement BETWEEN or LIKE options
14	unique indexed column < or > constant
15	nonunique indexed column < or > constant
16	sort/merge
17	MAX or MIN SQL statement functions on indexed column
18	ORDER BY entire index
19	full table scans

The Oracle Optimizers
Cost-based Optimizer

Oracle hints include:

/*+ALL_ROWS*/	**Optimize SQL for best throughput**
/*+AND_EQUAL*/	**Use index merging on specified tables**
/*+CLUSTER*/	**Use a cluster scan for a specified table**
/*+COST*/	**Use cost-based optimizer always**
/*+FIRST_ROWS*/	**Optimize SQL for best response times**
/*+FULL*/	**Use a full-table scan**
/*+HASH*/	**Use a hash-search method**
/*+INDEX*/	**Force the use of a specified index**
/*+STAR*/	**Force Star join, between a large table with concatenated keys and smaller tables**
/*+ORDERED*/	**Use the from clause join sequence**
/*+ROWID*/	**Use ROWID access method**
/*+USE_MERGE*/	**Use sort merge join technique**
/*+USE_NL*/	**Use nested loop join technique**
/*+USE_NOCACHE*/	**Don’t put the data in the buffers**
/*+USE_HASH*/	**Use a hash join**
/*+USE_CONCAT*/	**Use multiple indexes for or conditions**

& MORE

Helpful INIT.ORA Parameters

- Database File Multi Block Read Count
  - Both rule and cost
- Hash Multi Block IO Count
- Hash Join Enabled
- Hash Area Size
  - Cost only
- Sort Area Size
  - Both rule and cost

Helpful 8i INIT.ORA settings

- Optimizer Index Caching
- Optimizer Index Cost Adjustment
  - adjusts for low hit ratio assumption

Index Usage

- Rule Optimizer
  - Follows rules
  - Lower clustering factor, better rule selection
  - No functions in index
  - Use a function to NOT use an index

- Cost Optimizer
  - Follows stats
  - 8i supports function-based indexes
  - can use HINTS

Where comm * 1.1 > 1000 vs

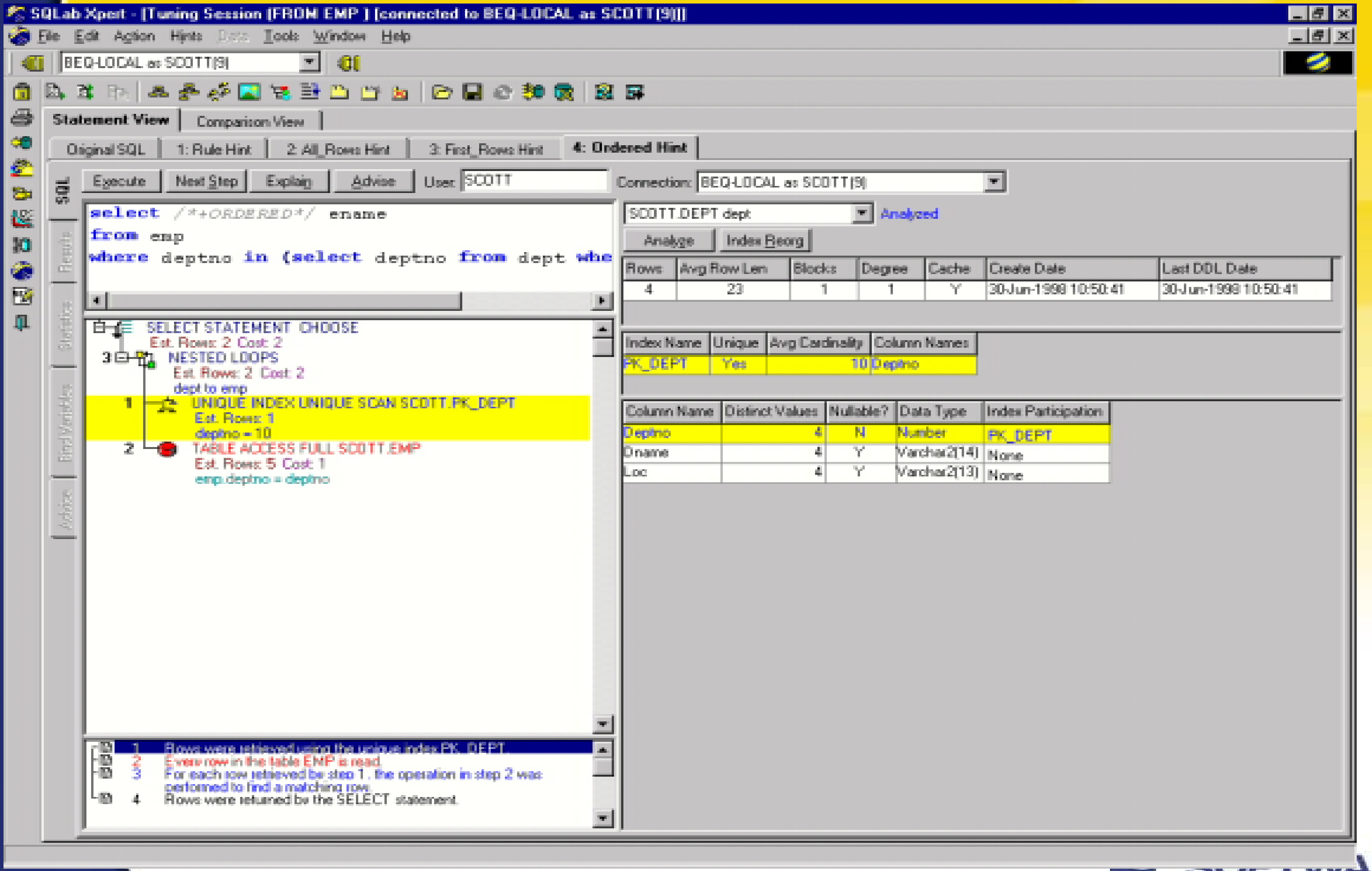
Where comm > 1000/1.1

Clustering Factor

- Is the relationship between the number of index blocks vs the number of related data blocks
- Low is good.
- Lower Clustering Factors:
  - Sort data into index order prior to loading
  - Use primary key (and in sorted order again)

QUEST SOFTWARE

The Oracle Optimizers
Cost-based Optimizer



User: SCOTT

Connection: BEQ-LOCAL as SCOTT(9)

```
select /*+ORDERED*/ ename
from emp
where deptno in (select deptno from dept whe
```

```
SELECT STATEMENT  CHOOSE
  Est. Rows: 2 Cost: 2
3    NESTED LOOPS
       Est. Rows: 2  Cost: 2
       dept to emp
1        UNIQUE INDEX UNIQUE SCAN SCOTT.PK_DEPT
           Est. Rows: 1
           deptno = 10
2        TABLE ACCESS FULL SCOTT.EMP
           Est. Rows: 5 Cost: 1
           emp.deptno = deptno
```

SCOTT.DEPT dept

Rows	Avg Row Len	Blocks	Degree	Cache	Create Date	Last DDL Date
4	23	1	1	Y	30-Jun-1998 10:50:41	30-Jun-1998 10:50:41

Index Name	Unique	Avg Cardinality	Column Names
PK_DEPT	Yes	10	Deptno

Column Name	Distinct Values	Nullable?	Data Type	Index Participation
Deptno	4	N	Number	PK_DEPT
Dname	4	Y	Varchar2(14)	None
Loc	4	Y	Varchar2(13)	None

1 Rows were retrieved using the unique index PK_DEPT.

2 Every row in the table EMP is read.

3 For each row retrieved by step 1, the operation in step 2 was performed to find a matching row.

4 Rows were returned by the SELECT statement.

Cost-based Assumtions

- Can use Cost Optimizer even if no stats
- Assumes even data distribution
- Assumes certain row counts
- Assumes low buffer hit ratio
- Assumes lots of users

Cost Optimizer Statistics

- Analyze <object> compute statistics
  - Collects:
    - Number of blocks
    - Number of rows
    - Indexes:
      - Granularity of Indexes (clustering factor)
      - Distinct values
      - Number of blocks per leaf
    - Histograms - if uneven data distribution

Understanding Explain Plans

- Understanding Explain Plans
  - Is a Necessity to Tuning SQL Statements
  - Shows you the choices made by either optimizer
  - Can be difficult to interpret
  - Indenting and tools greatly aid!

Understanding Explain Plans
Driving Table

- Rule
  - Based on existence of indexes or LAST table in FROM clause or first NESTED SELECT
- COST - FIRST ROWS
  - Based on unique indexes
  - Tries to avoid Full Scans and Sorts
- COST - ALL ROWS
  - Based on total rows returned
- Cost - Ordered Hint

Understanding Explain Plans Syntax

```
SQL>
SQL> EXPLAIN PLAN FOR
  2  select ename
  3  from emp
  4  where deptno in (select deptno from dept where deptno = 10);

Explained.

SQL> SELECT operation, options, object_name, id, parent_id
  2  from plan_table;

OPERATION                      OPTIONS              OBJECT_NAME                     ID        PARENT_ID
------------------------------ -------------------- ------------------------------ --------- ---------
SELECT STATEMENT                                                                    0
NESTED LOOPS                                                                        1         0
 INDEX                         UNIQUE SCAN                  PK_DEPT                 2         1
 TABLE ACCESS                       FULL                    EMP                     3         1
```

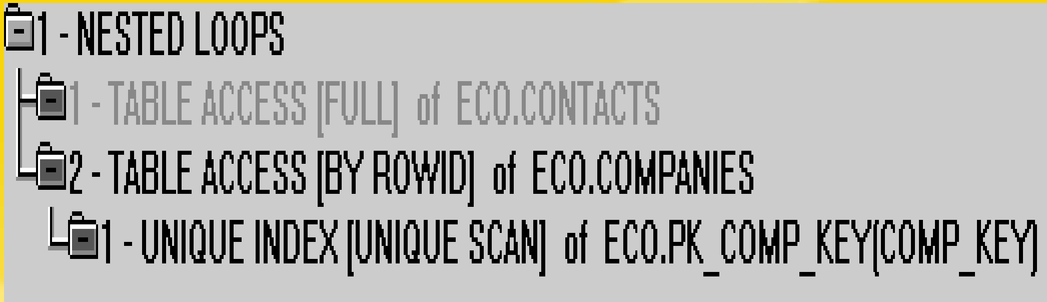
Understanding Explain Plans Syntax

Explain Symbol	**Description**
AND-EQUAL	Index values will be used to join rows.
CONCATENATION	SQL statement UNION command.
FILTER	FILTERs apply 'other criteria' in the query to further qualify the matching rows. The 'other criteria' include correlated subqueries, and HAVING clause.
FIRST ROW	SQL statement will be processed via a cursor.
FOR UPDATE	SQL statement clause ‘for update of’ placed row level locks on affected rows.
INDEX (UNIQUE)	SQL statement utilized a unique index to search for a specific value.
INDEX (RANGE SCAN)	SQL statement contains a nonequality or BETWEEN condition.
HASH JOIN	SQL statement initiated a hash-join operation.
MERGE JOIN	SQL statement references two or more tables, sorting the two result sets being joined over the join columns and then merging the results via the join columns.
NESTED LOOPS	This operation is one form of joining tables. One row is retrieved from the row source identified by the first (inner) operation, and then joined to all matching rows in the other table (outer).
NONUNIQUE INDEX (RANGE SCAN)	The RANGE SCAN option indicates that ORACLE expects to return multiple matches (ROWIDs) from the index search
PARTITION (CONCATTENATED)	SQL statement will access a partitioned object and merge the retrieved rows from the accessed partitions.
PARTITION (SINGLE)	SQL statement will access a single partition.
PARTITION (EMPTY)	The SQL statement makes reference to an empty partition.
SORT (ORDER BY)	SQL statement contains an ORDER BY SQL command.
SORT (AGREGATE)	SQL statement initiated a sort to resolve a MIN or MAX function.
SORT (GROUP BY)	SQL statement contains a GROUP BY SQL command.
TABLE ACCESS (FULL)	All rows are retrieved from the table without using an index.
TABLE ACCESS (BY ROWID)	A row is retrieved based on ROWID
TABLE ACCESS (CLUSTER)	A row is retrieved from a table that is part of a cluster.
UNION	SQL statement contains a DISTINCT SQL command.

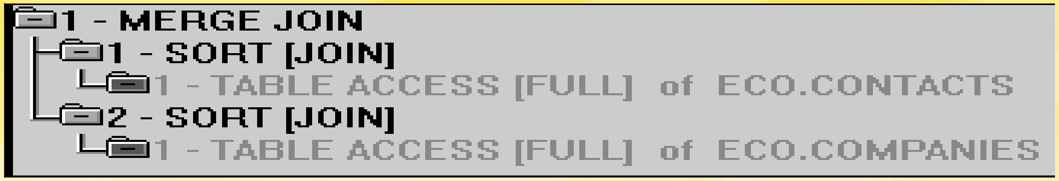
QUEST SOFTWARE

Understanding Explain Plans Syntax

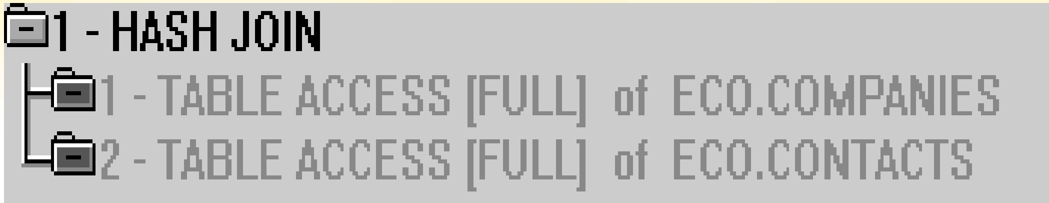
- Nested Loop Join
  - driving table
  - Default order(rule)
- Merge Scan Join
  - sort & match
- Hash Join (7.3)
  - Full scans with no sorts
  - Join column to row address



```
1 - NESTED LOOPS
├1 - TABLE ACCESS [FULL] of ECO.CONTACTS
└2 - TABLE ACCESS [BY ROWID] of ECO.COMPANIES
  └1 - UNIQUE INDEX [UNIQUE SCAN] of ECO.PK_COMP_KEY(COMP_KEY)
```



```
1 - MERGE JOIN
├1 - SORT [JOIN]
│ └1 - TABLE ACCESS [FULL] of ECO.CONTACTS
└2 - SORT [JOIN]
  └1 - TABLE ACCESS [FULL] of ECO.COMPANIES
```



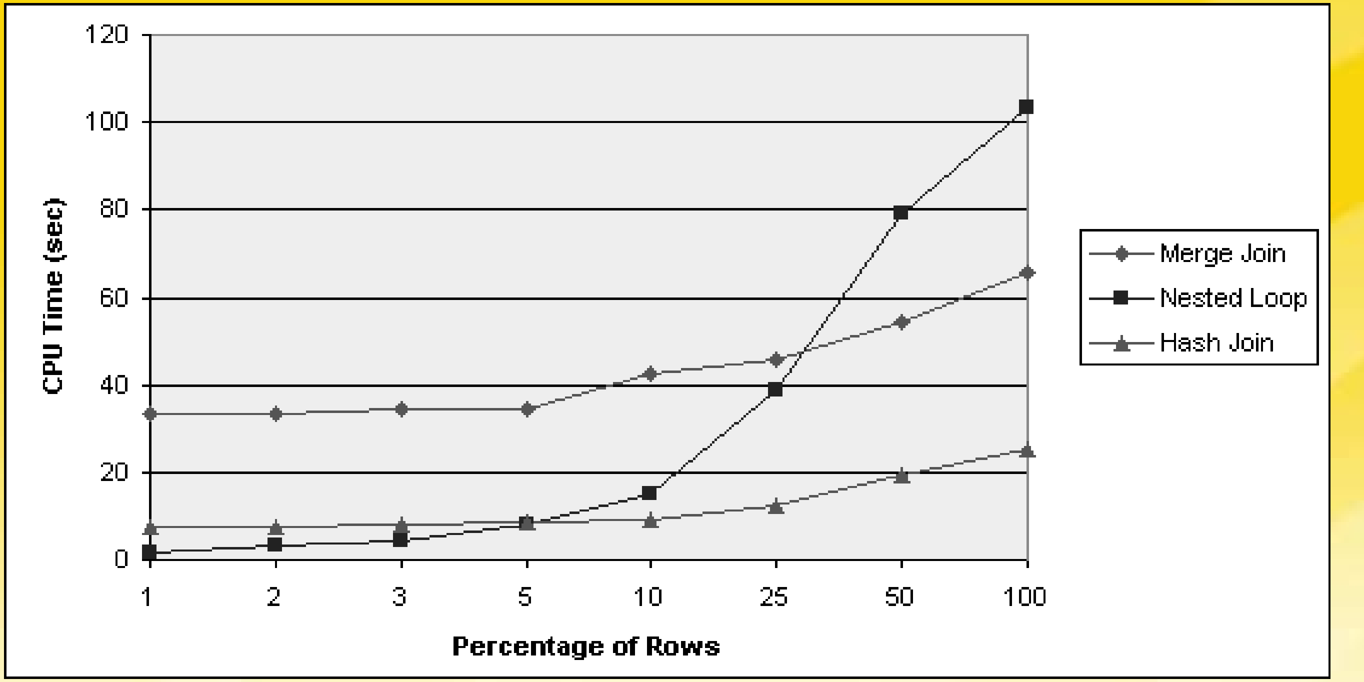
```
1 - HASH JOIN
├1 - TABLE ACCESS [FULL] of ECO.COMPANIES
└2 - TABLE ACCESS [FULL] of ECO.CONTACTS
```

QUEST SOFTWARE

Understanding Explain Plans

- Nested Loop Join
  - small portion accessed from a large table & joined from a small portion of the second table
- Merge Scan Join
  - large portion of rows are being joined
- Hash Join
  - large portion with a lot of memory

Understanding Explain Plans



CPU Time (sec) - Parse & Execute

Percentage	Nested Loop	Merge Join	Hash Join
100	103.26	65.63	25.06
50	79.26	54.34	19.16
25	39.01	45.85	12.40
10	15.22	42.26	8.99
5	8.10	34.33	8.50
3	4.54	34.25	7.95
2	3.15	33.47	7.63
1	1.74	33.59	7.60

Understanding Explain Plans Tips

- Use bind variables
  - – Helps SQL parsing/processing
- Merging Indexes
  - – Oracle will process up to 5
  - – Use AND-EQUAL hint to select and limit
- Nested Selects vs Joins vs UNION ALL
  - – Use UNION ALL syntax where possible
  - – Cost optimizer might change NS to Joins

Oracle Tuning Tools

- Explain Table
- TKPROF
- GUI Tools
  - SQL Navigator
  - TOAD
  - OEM SQL Analyze
  - SQLab

Oracle Tuning Tools
Explain Plan

```
SQL>
SQL> EXPLAIN PLAN FOR
  2  select ename
  3  from emp
  4  where deptno in (select deptno from dept where deptno = 10);

Explained.

SQL> SELECT operation, options, object_name, id, parent_id
  2  from plan_table;

OPERATION                     OPTIONS              OBJECT_NAME                    ID         PARENT_ID
----------------------------- -------------------- ------------------------------ ---------  ---------
SELECT STATEMENT                                                                  0
NESTED LOOPS                                                                      1          0
 INDEX                        UNIQUE SCAN                  PK_DEPT                2          1
 TABLE ACCESS                 FULL                         EMP                    3          1
```

Oracle Tuning Tools
TKPROF

```
select * from tutorial.cur_emp_status

    call  count  cpu    elapsed  disk   query  current   rows

  ------- ------  -------- ---------- ---------- ---------- ----------  ----------

Parse         1      0.02       0.00        0          0          0          0

Execute     1      0.00       0.00        0          0          0          0

Fetch        2      0.23       0.25       11    1051          3        15

   ------- ------  -------- ---------- ---------- ---------- ----------  ----------

total        4      0.25       0.27       11    1051          3        15
```

QUEST SOFTWARE

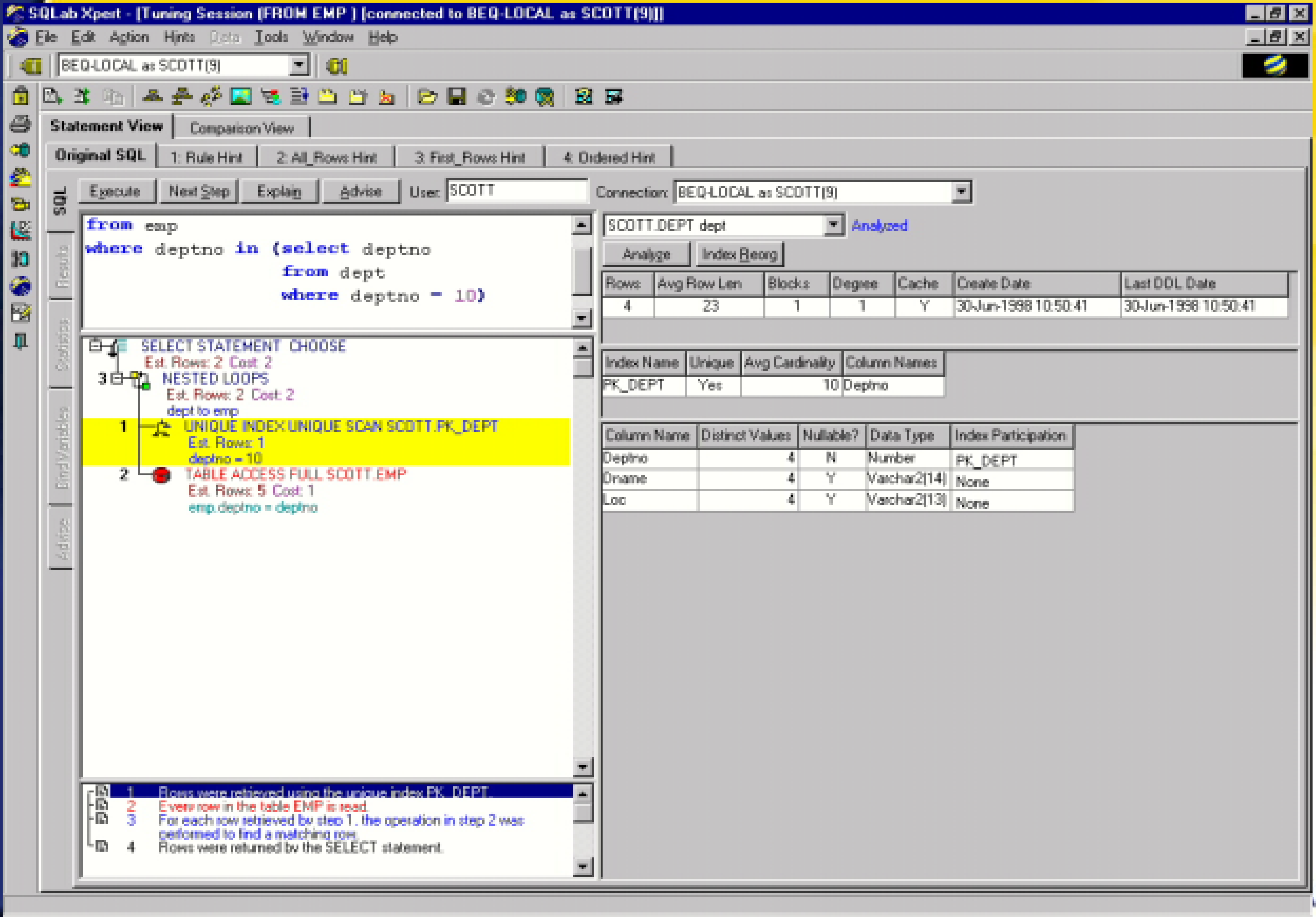
Oracle Tuning Tools -TKPROF

```
Rows    Execution Plan

-------  ---------------------------------------------------

   0  SELECT STATEMENT   HINT: CHOOSE

 137   FILTER

 137    NESTED LOOPS

 137     NESTED LOOPS

 137      NESTED LOOPS

  49      NESTED LOOPS

  23       NESTED LOOPS

  15        TABLE ACCESS   HINT: ANALYZED (FULL)
    OF 'EMPLOYEES'

  23        TABLE ACCESS   HINT: ANALYZED
    (CLUSTER) OF

             'SAL_HISTORY'
```

```
Rows      Execution Plan

-------     -------------------------------------------------

   137     TABLE ACCESS   HINT: ANALYZED (BY ROWID) OF 'JOB_CODES'

   137      INDEX   HINT: ANALYZED (UNIQUE SCAN) OF 'I_JOBS' (UNIQUE)


   137     TABLE ACCESS   HINT: ANALYZED (BY ROWID) OF 'DEPARTMENTS'

   137      INDEX   HINT: ANALYZED (UNIQUE SCAN) OF 'I_DEPTS' (UNIQUE)


    23    SORT (AGGREGATE)

    23     TABLE ACCESS   HINT: ANALYZED (CLUSTER) OF 'DEPT_HISTORY'

    15      INDEX   HINT: ANALYZED (UNIQUE SCAN) OF 'I_EMP_EMPNO'

              (CLUSTER)

    23    SORT (AGGREGATE)

    23     TABLE ACCESS   HINT: ANALYZED (CLUSTER) OF 'JOB_HISTORY'

    15      INDEX   HINT: ANALYZED (UNIQUE SCAN) OF 'I_EMP_EMPNO'

              (CLUSTER)
```

QUEST SOFTWARE

Oracle Tools - SQLab XPert



User: SCOTT

Connection: BEQ-LOCAL as SCOTT(9)

```
from emp
where deptno in (select deptno
                 from dept
                 where deptno = 10)
```

```
SELECT STATEMENT  CHOOSE
  Est. Rows: 2  Cost: 2
3   NESTED LOOPS
      Est. Rows: 2  Cost: 2
      dept to emp
1       UNIQUE INDEX UNIQUE SCAN SCOTT.PK_DEPT
          Est. Rows: 1
          deptno = 10
2       TABLE ACCESS FULL SCOTT.EMP
          Est. Rows: 5  Cost: 1
          emp.deptno = deptno
```

SCOTT.DEPT dept

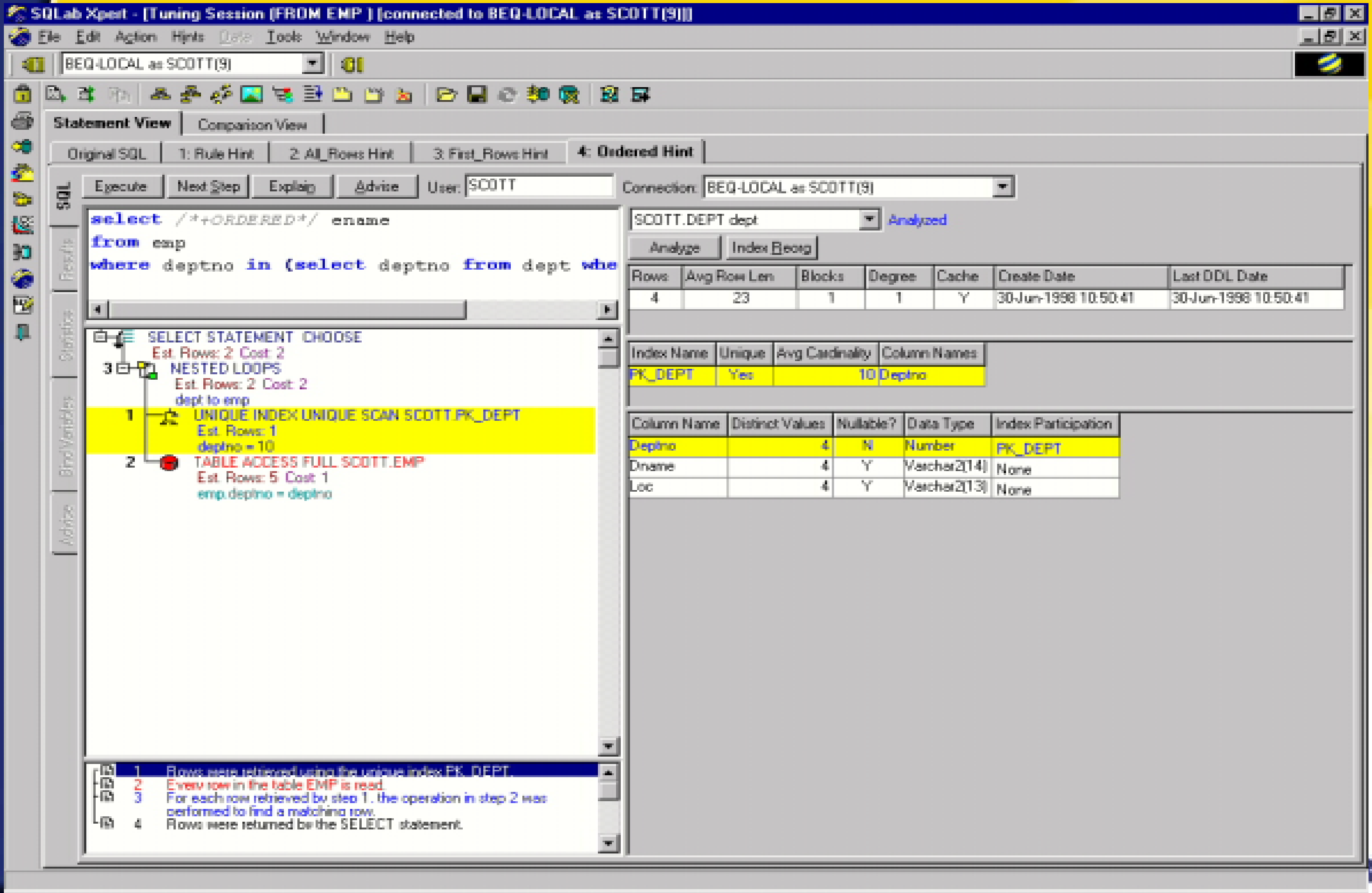
Rows	Avg Row Len	Blocks	Degree	Cache	Create Date	Last DDL Date
4	23	1	1	Y	30-Jun-1998 10:50:41	30-Jun-1998 10:50:41

Index Name	Unique	Avg Cardinality	Column Names
PK_DEPT	Yes	10	Deptno

Column Name	Distinct Values	Nullable?	Data Type	Index Participation
Deptno	4	N	Number	PK_DEPT
Dname	4	Y	Varchar2(14)	None
Loc	4	Y	Varchar2(13)	None

1 Rows were retrieved using the unique index PK_DEPT.
2 Every row in the table EMP is read.
3 For each row retrieved by step 1, the operation in step 2 was performed to find a matching row.
4 Rows were returned by the SELECT statement.

Oracle Tools - SQLab XPert



User: SCOTT

Connection: BEQ-LOCAL as SCOTT(9)

```
select /*+ORDERED*/ ename
from emp
where deptno in (select deptno from dept whe
```

```
SELECT STATEMENT  CHOOSE
 Est. Rows: 2  Cost: 2
3  NESTED LOOPS
    Est. Rows: 2  Cost: 2
    dept to emp
  1  UNIQUE INDEX UNIQUE SCAN SCOTT.PK_DEPT
      Est. Rows: 1
      deptno = 10
  2  TABLE ACCESS FULL SCOTT.EMP
      Est. Rows: 5  Cost: 1
      emp.deptno = deptno
```

```
1   Rows were retrieved using the unique index PK_DEPT.
2   Every row in the table EMP is read.
3   For each row retrieved by step 1, the operation in step 2 was
    performed to find a matching row.
4   Rows were returned by the SELECT statement.
```

SCOTT.DEPT dept

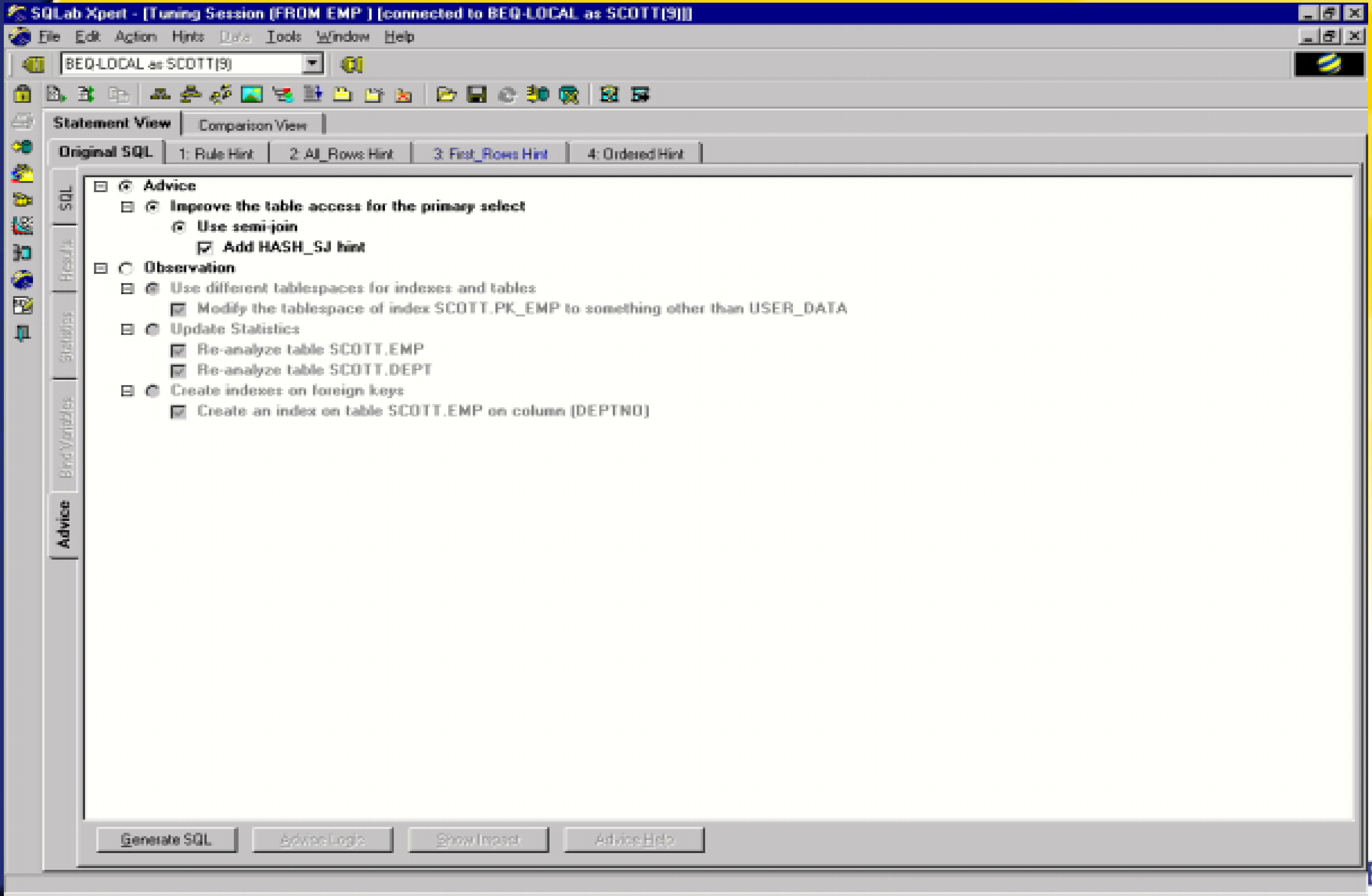
Rows	Avg Row Len	Blocks	Degree	Cache	Create Date	Last DDL Date
4	23	1	1	Y	30-Jun-1998 10:50:41	30-Jun-1998 10:50:41

Index Name	Unique	Avg Cardinality	Column Names
PK_DEPT	Yes	10	Deptno

Column Name	Distinct Values	Nullable?	Data Type	Index Participation
Deptno	4	N	Number	PK_DEPT
Dname	4	Y	Varchar2(14)	None
Loc	4	Y	Varchar2(13)	None

RE

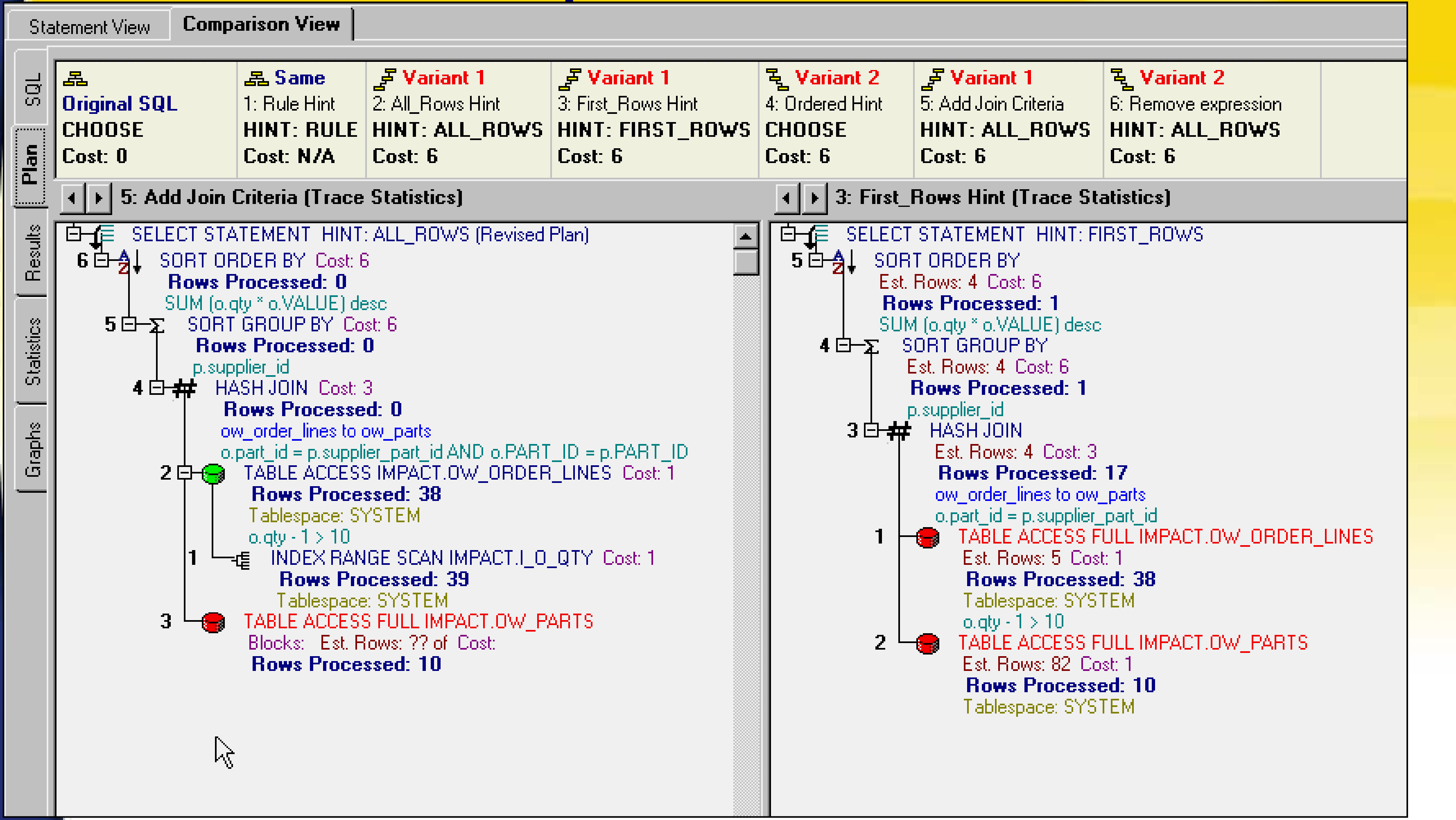
Oracle Tools - SQLab XPert



⊟ ● Advice
- ⊟ ● Improve the table access for the primary select
  - ● Use semi-join
    - ☑ Add HASH_SJ hint

⊟ ○ Observation
- ⊟ ● Use different tablespaces for indexes and tables
  - ☑ Modify the tablespace of index SCOTT.PK_EMP to something other than USER_DATA
- ⊟ ● Update Statistics
  - ☑ Re-analyze table SCOTT.EMP
  - ☑ Re-analyze table SCOTT.DEPT
- ⊟ ● Create indexes on foreign keys
  - ☑ Create an index on table SCOTT.EMP on column (DEPTNO)

Oracle Tools - SQLab XPert Compare Plans



	Same	Variant 1	Variant 1	Variant 2	Variant 1	Variant 2
Original SQL	1: Rule Hint	2: All_Rows Hint	3: First_Rows Hint	4: Ordered Hint	5: Add Join Criteria	6: Remove expression
CHOOSE	HINT: RULE	HINT: ALL_ROWS	HINT: FIRST_ROWS	CHOOSE	HINT: ALL_ROWS	HINT: ALL_ROWS
Cost: 0	Cost: N/A	Cost: 6	Cost: 6	Cost: 6	Cost: 6	Cost: 6

5: Add Join Criteria (Trace Statistics)

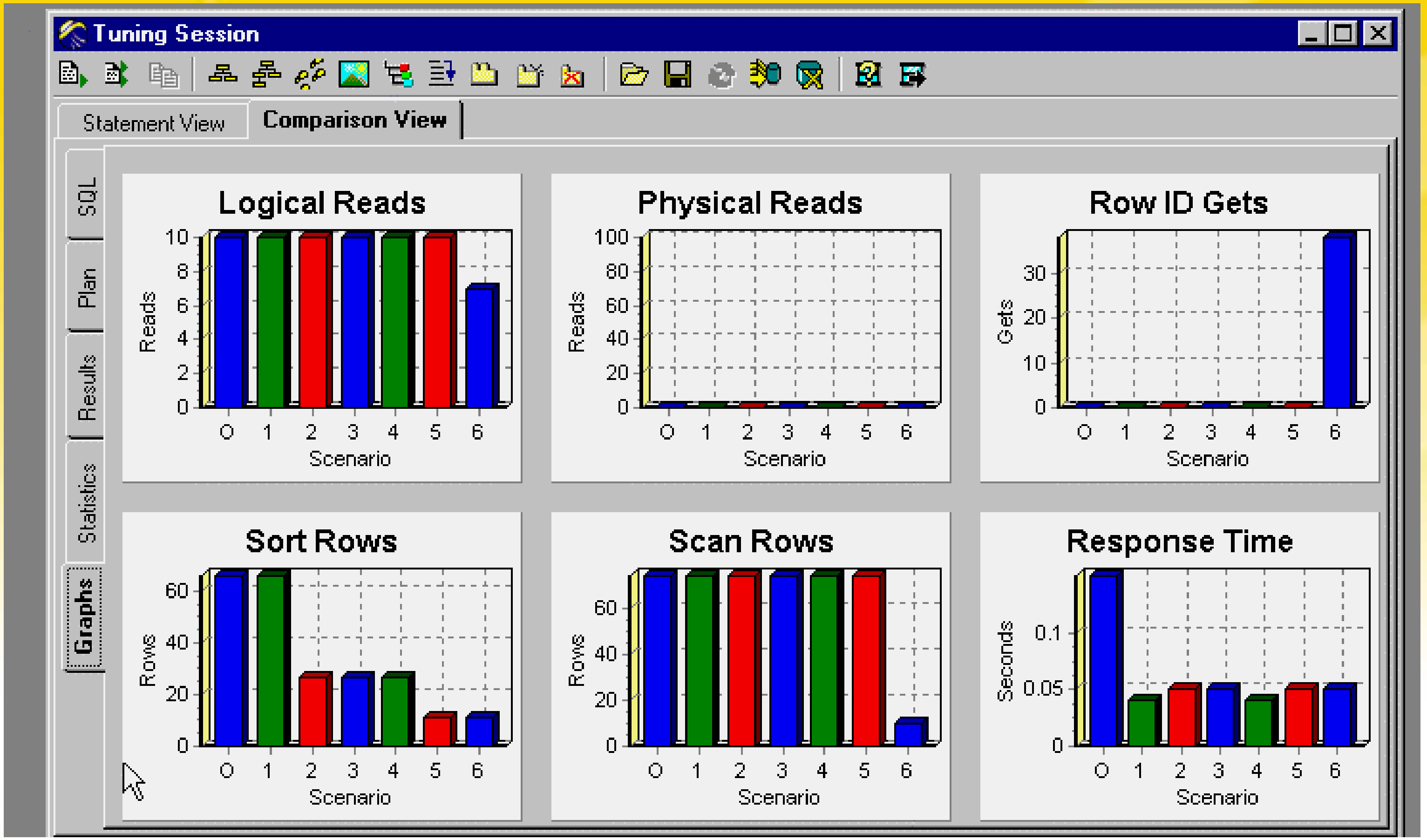
```
SELECT STATEMENT  HINT: ALL_ROWS (Revised Plan)
6  SORT ORDER BY  Cost: 6
     Rows Processed: 0
     SUM (o.qty * o.VALUE) desc
   5  SORT GROUP BY  Cost: 6
        Rows Processed: 0
        p.supplier_id
      4  HASH JOIN  Cost: 3
           Rows Processed: 0
           ow_order_lines to ow_parts
           o.part_id = p.supplier_part_id AND o.PART_ID = p.PART_ID
         2  TABLE ACCESS IMPACT.OW_ORDER_LINES  Cost: 1
              Rows Processed: 38
              Tablespace: SYSTEM
              o.qty - 1 > 10
            1  INDEX RANGE SCAN IMPACT.I_O_QTY  Cost: 1
                 Rows Processed: 39
                 Tablespace: SYSTEM
         3  TABLE ACCESS FULL IMPACT.OW_PARTS
              Blocks:   Est. Rows: ?? of  Cost:
              Rows Processed: 10
```

3: First_Rows Hint (Trace Statistics)

```
SELECT STATEMENT  HINT: FIRST_ROWS
5  SORT ORDER BY
     Est. Rows: 4  Cost: 6
     Rows Processed: 1
     SUM (o.qty * o.VALUE) desc
   4  SORT GROUP BY
        Est. Rows: 4  Cost: 6
        Rows Processed: 1
        p.supplier_id
      3  HASH JOIN
           Est. Rows: 4  Cost: 3
           Rows Processed: 17
           ow_order_lines to ow_parts
           o.part_id = p.supplier_part_id
         1  TABLE ACCESS FULL IMPACT.OW_ORDER_LINES
              Est. Rows: 5  Cost: 1
              Rows Processed: 38
              Tablespace: SYSTEM
              o.qty - 1 > 10
         2  TABLE ACCESS FULL IMPACT.OW_PARTS
              Est. Rows: 82  Cost: 1
              Rows Processed: 10
              Tablespace: SYSTEM
```

QUEST SOFTWARE

Oracle Tools - SQLab XPert Compare Performance



Oracle Tools - Plan Stability

- Stored Outlines
  - Is a method to guarantee that a certain Execution Plan will be used for a particular SQL statement
  - Oracle can specifically create Stored Outlines -or- it can create one for each SQL statement presented
    - CREATE OR REPLACE <outline> FOR CATEGORY <category> ON <sql statement>;
    - system/session parameter CREATE_STORED_OUTLINES = <TRUE, FALSE, ‘category name’ NOOVERIDE>
  - Must have ‘CREATE ANY OUTLINE’ permissions

Oracle Tools - Plan Stability

- Oracle will use a Stored Outline unless:
  - system/session parameter USE_STORED_OUTLINES = False
  - mismatch on SQL text including Hints
- How it works:
  - Oracle uses OL$ and OL$HINTS tables
    - see also USER_OUTLINES, USER_OUTLINE_HINTS
  - Stored indefinitely unless explicitly removed
  - system/session parameter USE_STORED_OUTLINES = TRUE or <category>

Oracle Tools - Plan Stability

- Stored Outline Management
  - Packages DROP_UNUSED, DROP_BY_CAT, UPDATE_BY_CAT
  - Moving Outlines:
    - EXP OUTLN/OUTLN FILE = <file name> TABLES = ‘OL$’ ‘OL$HINTS’ SILENT=Y [WHERE CATEGORY=<category>]
    - IMP OUTLN/OUTLN FILE=<file name> TABLES = ‘OL$’ ‘OL$HINT’ IGNORE=Y SILENT=Y
    - see p 7-32 Oracle8i Tuning Guide

SQL Do's and Don'ts

- Avoid using the HAVING clause.
  - Use WHERE clause
  - The HAVING statement filters selected rows only after all of the rows have been retrieved.
- Use the NOT EXISTS statement in place of a NOT IN statement.
- Use joins in place of EXISTS.
- Use EXISTS in place of DISTINCT.

SQL Do's and Don'ts

- AVOID doing calculations on indexed WHERE columns
  - the optimizer will use a full-table scan
    - Oracle8i has new function-based index feature
- Depending on the types of SQL statement issued, think about using a concatenated index.
- Avoid using NOT on indexed columns(precludes using a index).
- Use WHERE instead of ORDER BY when an index is used.
- Bind variables Vs. constants

Summary

- Try to use good database design throughout all applications
- Monitor
- Understand your options
  - Optimizer Modes
  - Join conditions
- Use GUI Tuning Tools
- Don't Let *SQL RAGE* Happen to you!

QUEST SOFTWARE

ORACLE8
Advanced Tuning & Administration
Oracle Press
Oracle SQL
HIGH-PERFORMANCE TUNING
SUN PERFORMANCE AND TUNING
Oracle
THIRD EDITION
Expert Insight
Authoritative Advice
SAMS
Unleashed
Oracle Performance Tuning
O'Reilly & Associates, Inc.
Oracle Desk Reference
THE PROFESSIONAL'S COMPANION
Software Engineering with Oracle
Best Practices for Mission-Critical Systems
Oracle 8i from scratch
QUE

Our Experts Wrote the Books...

QUEST SOFTWARE

Questions?

Dan Hotka is a Director of Database Field Operations for Quest Software. He has over 21 years in the computer industry and over 16 years experience with Oracle products. He is an acknowledged Oracle expert with Oracle experience dating back to the Oracle V4.0 days. He has just completed Oracle8i from Scratch by Que and has co-authored the popular books Oracle Unleashed, Oracle8 Server Unleashed, Oracle Development Unleashed by SAMS and Special Edition using Oracle8/8i by Que, is frequently published in Oracle Professional by Pinnacle Publications, and regularly speaks at Oracle conferences and user groups around the world. Dan can be reached at dhotka@earthlink.net or dhotka@quest.com .

Bibliography:

Tim Gorman, www.sagelogix.com

Top Offensive SQL Statements, Oracle Professional June 2000; A Pinnacle Publication

Oracle8i from Scratch by Dan Hotka; A Que Publication

Oracle8i SQL Tuning Guide (Oracle Doc Set)

QUEST SOFTWARE

www.quest.com