SQL Tuning via Toad

QUEST SOFTWARE®

Tips for Optimizing SQL Performance

Bert Scalzo …

Database Expert & Product Architect for Quest Software

Oracle ACE

Oracle Background:
- Worked with Oracle databases for over two decades (starting with version 4)
- Work history includes time at both “Oracle Education” and “Oracle Consulting”

Academic Background:
- Several Oracle Masters certifications
- BS, MS and PhD in Computer Science
- MBA (general business)
- Several insurance industry designations

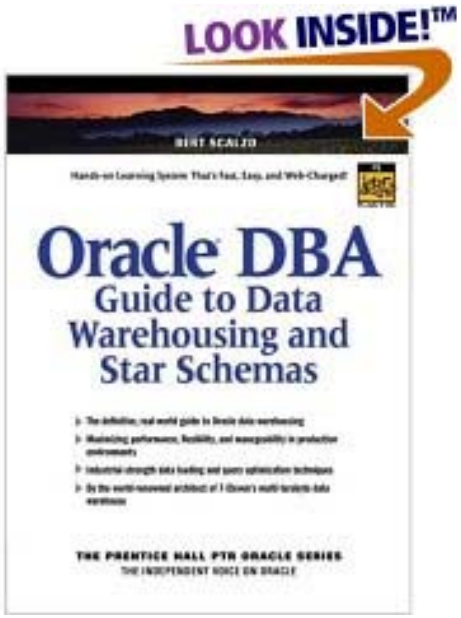
Key Interests:
- Data Modeling
- Database Benchmarking
- Database Tuning & Optimization
- "Star Schema" Data Warehouses
- Oracle on Linux – and specifically: RAC on Linux

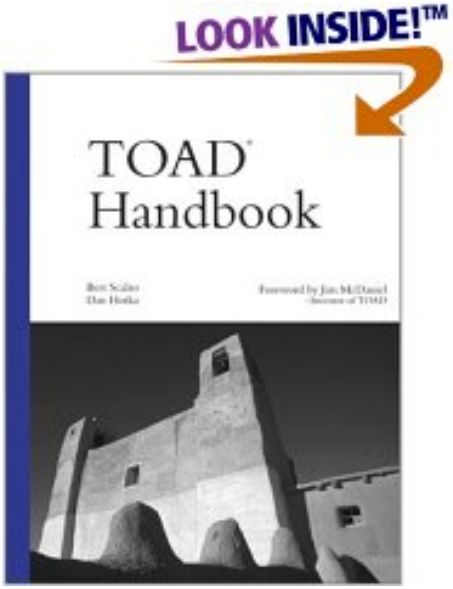
Articles for:
- Oracle’s Technology Network (OTN)
- Oracle Magazine,
- Oracle Informant
- PC Week (eWeek)

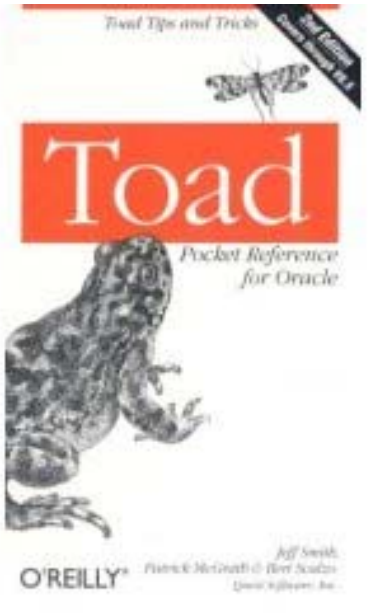
Articles for:
- Dell Power Solutions Magazine
- The Linux Journal
- www.linux.com
- www.orafaq.com

QUEST SOFTWARE®

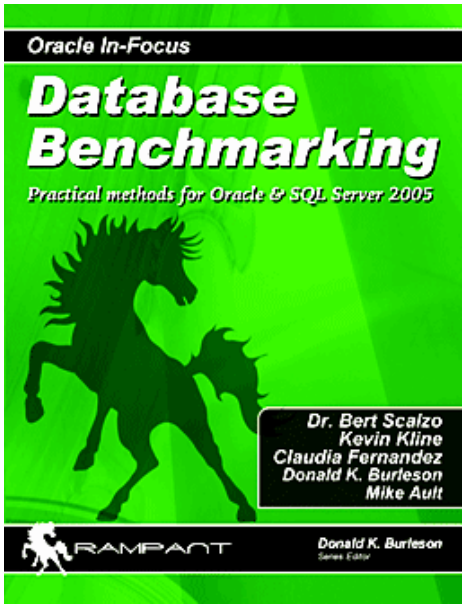
Books by Bert …

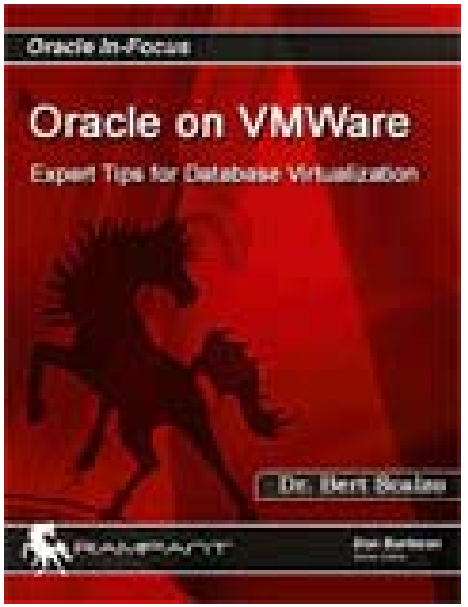


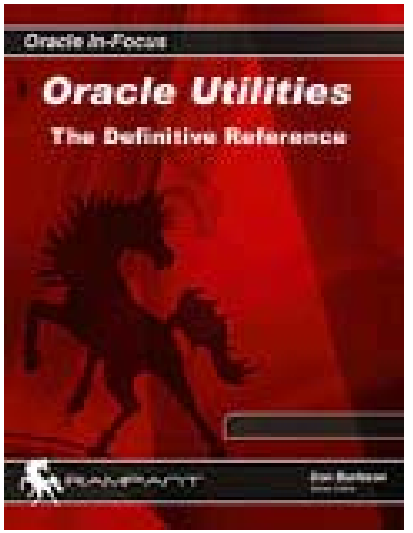




Coming in 2009 …







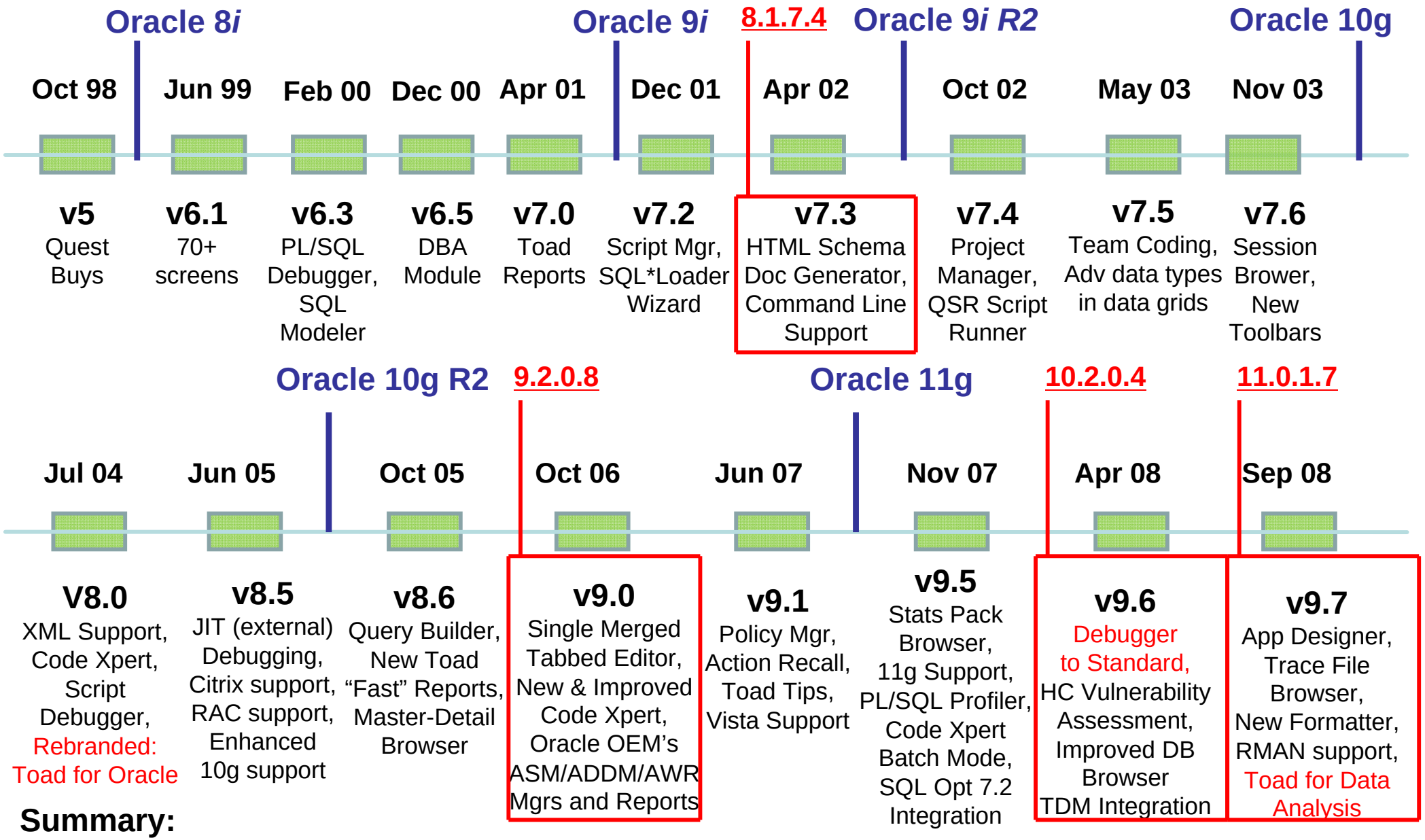
Also: FREE Toad e-Book for Toad 10…

QUEST SOFTWARE®

Topics …

- Pre-Reqs
  - Correct Toad vs. Oracle Database Server version
  - Correct Oracle SQL*Net Client networking version
  - SQL Tuning Approach – much more than just explain plans and run times
- Explain Plans
  - Setup and effective use of the ”Explain Plan”
  - Be careful, Explain Plan costs can sometimes not be the best way to pick the winner - sometimes (auto) trace is required to be 100% sure
  - Some guidelines on how to best or at least more easily read SQL explain plans - which is the general starting point for any SQL tuning attempt
- SQL Tuning Rules
  - Some Guidelines i.e. (“Golden Rules”) – just the tip of the iceberg
    - Efficient and fast selects & sub selects
    - Dealing with large tables
    - Parallel Hints
      Pinning SQL in Memory
    - Efficient SQL queries that use a lot of AND conditionals or sub-queries
    - How to avoid full-table scans
- Is There a Better (i.e. more productive) Way to Tune SQL
  - SQL Optimzier – automate all the above (and much more)

Toad vs. Oracle Product Release History



Summary:

Oracle 9i >= Toad 9.0 **Oracle 10g >= Toad 9.6** **Oracle 11g >= Toad 9.7**

QUEST SOFTWARE

Oracle Client / Server Interoperability Support
(See Metalink Document **207303.1**)

	Server Version										
Client Version	**11.1.0**	**10.2.0**	**10.1.0**	**9.2.0**	**9.0.1**	**8.1.7**	**8.1.6**	**8.1.5**	**8.0.6**	**8.0.5**	**7.3.4**
11.1.0	Yes	Yes #6	Yes #6	ES #5	No	No	No #3	No #3	No #3	No #3	No #3
10.2.0	Yes #6	Yes	Yes	ES #5	No	Was	No #3	No #3	No #3	No #3	No #3
10.1.0(#4)	Yes #6	Yes	Yes	ES	Was	Was #2	No #3	No #3	No #3	No #3	No #3
9.2.0	ES #5	ES #5	ES	ES	Was	Was	No	No	Was	No	No #1
9.0.1	No	No	Was	Was	Was	Was	Was	No	Was	No	Was
8.1.7	No	Was	Was	Was	Was	Was	Was	Was	Was	Was	Was
8.1.6	No	No	No	No	Was	Was	Was	Was	Was	Was	Was
8.1.5	No	No	No	No	No	Was	Was	Was	Was	Was	Was
8.0.6	No	No	No	Was	Was	Was	Was	Was	Was	Was	Was
8.0.5	No	No	No	No	No	Was	Was	Was	Was	Was	Was
7.3.4	No	No	No	Was	Was	Was	Was	Was	Was	Was	Was

Key:

Yes	Supported
ES	Supported but fixes only possible for customers with Extended Support .
Was	Was a supported combination but one of the releases is no longer covered by any of Premier Support , Primary Error Correct support , Extended Support nor Extended Maintenance Support so fixes are no longer possible.
No	Has never been Supported

Toad may work with older client talking to newer databases - but there might be data type issues ☹

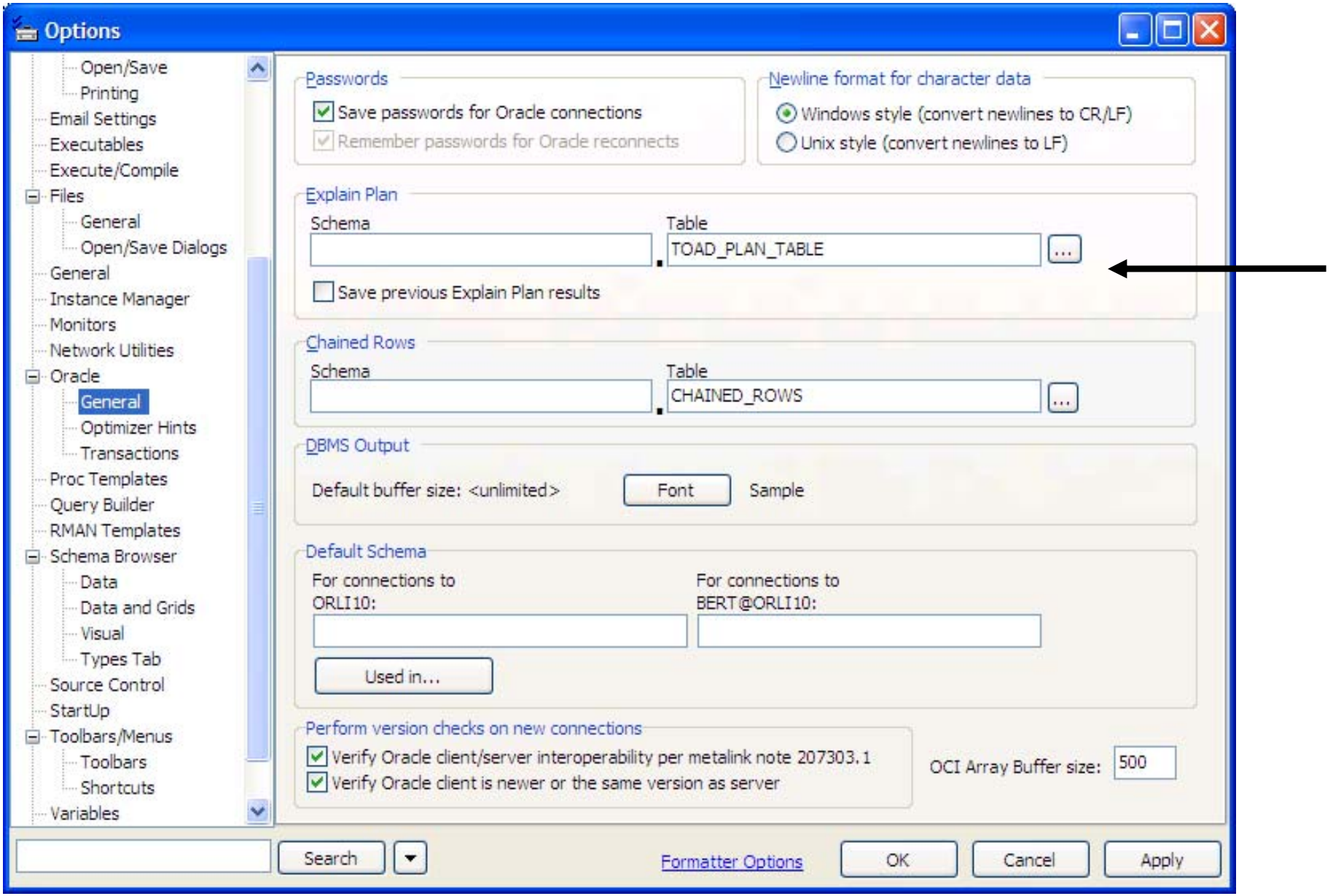
QUEST SOFTWARE

Seven Steps for SQL Tuning Success

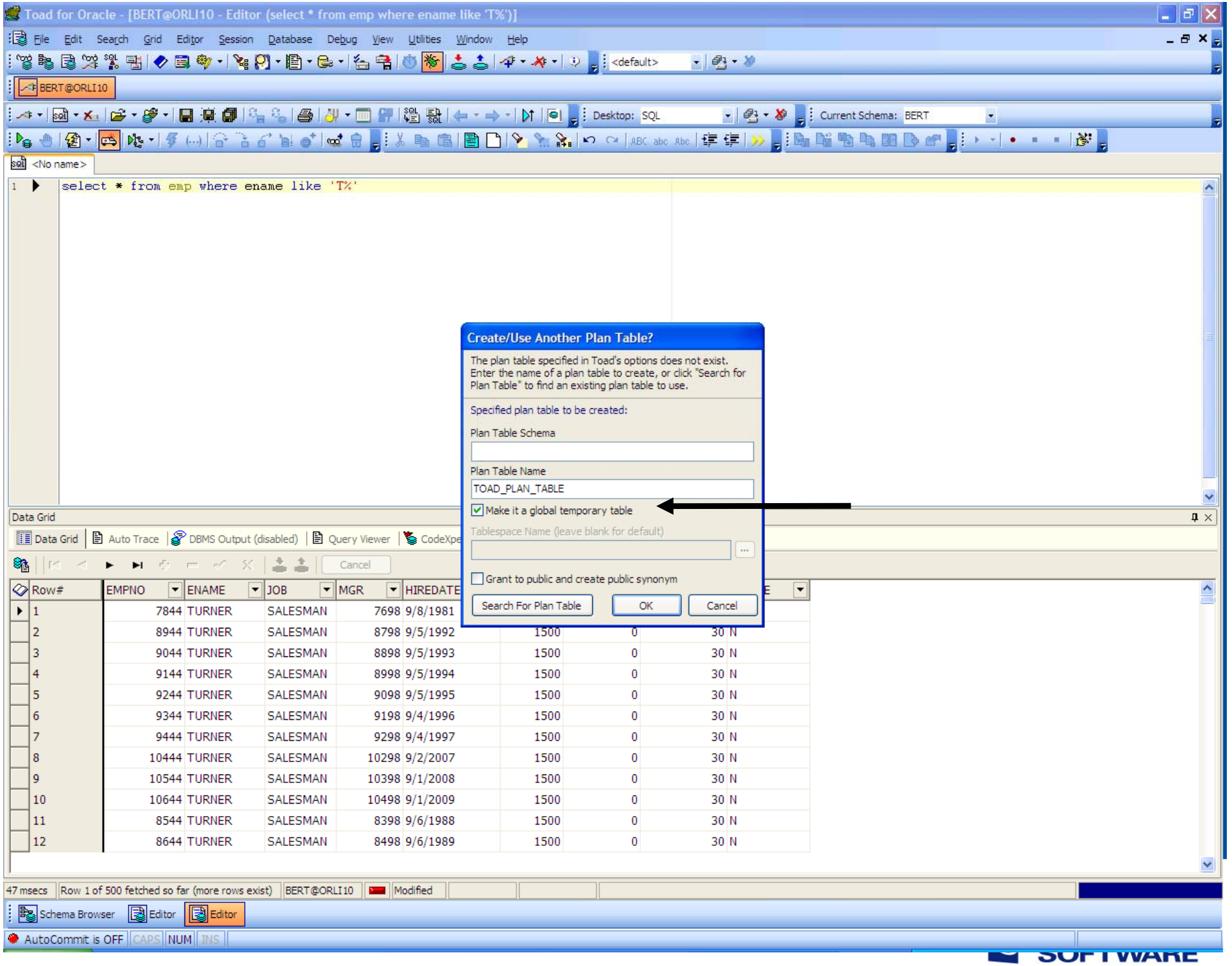
1. Always start by knowing (i.e. being able to say in English) what the query does
2. For queries involving more than 2 tables, a data model can be a handy road map
3. Explain plan costs alone may well lead you astray – sometimes the costs can lie
4. Sometimes equal execution times don't necessarily equate to equivalent solutions
5. You should always include (auto) trace information to divine among all the above
6. Sole reliance on automatic SQL optimization and tuning tools can be suboptimal
7. You must add human intuition and insight to the optimization process for success

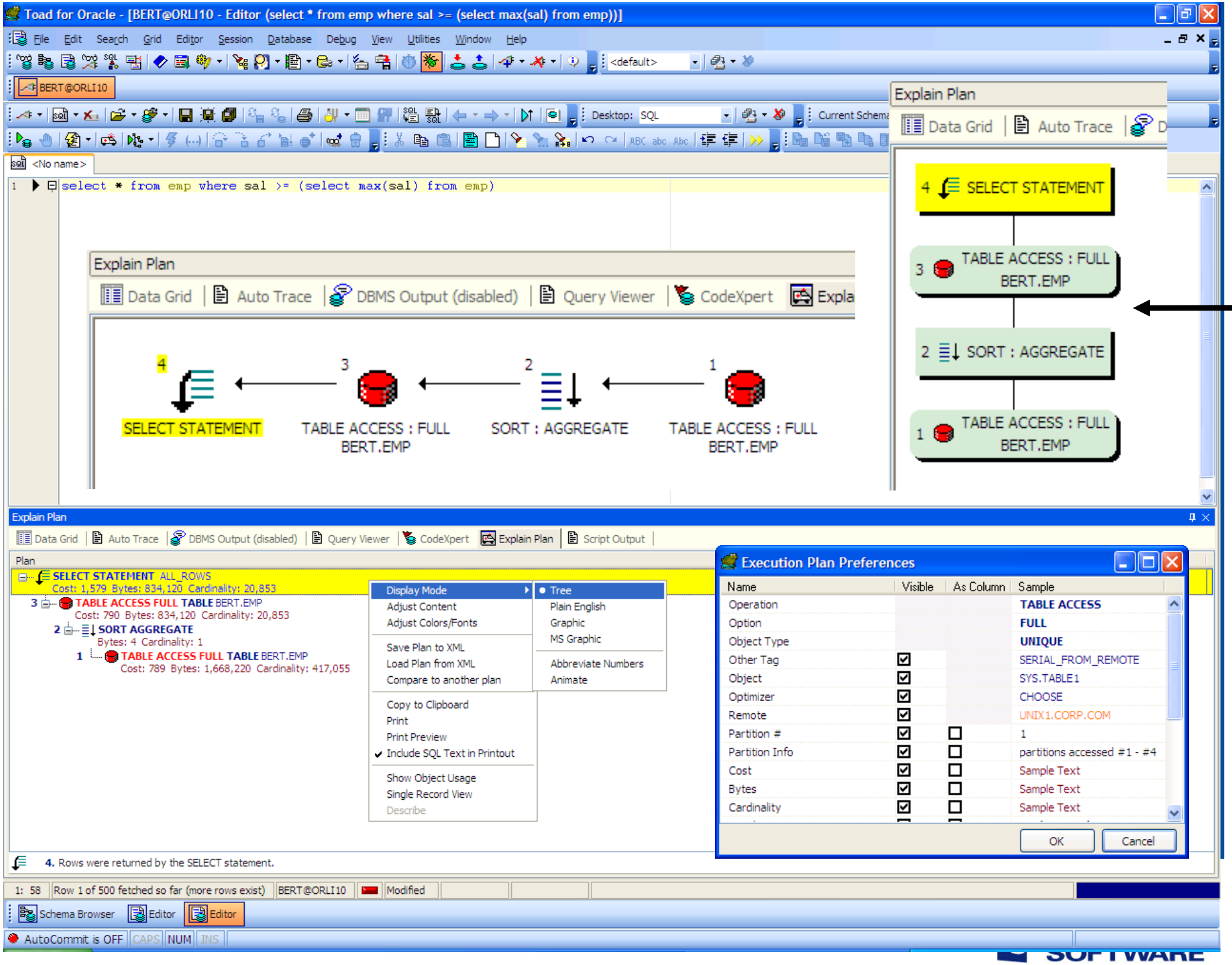
Explain Plans

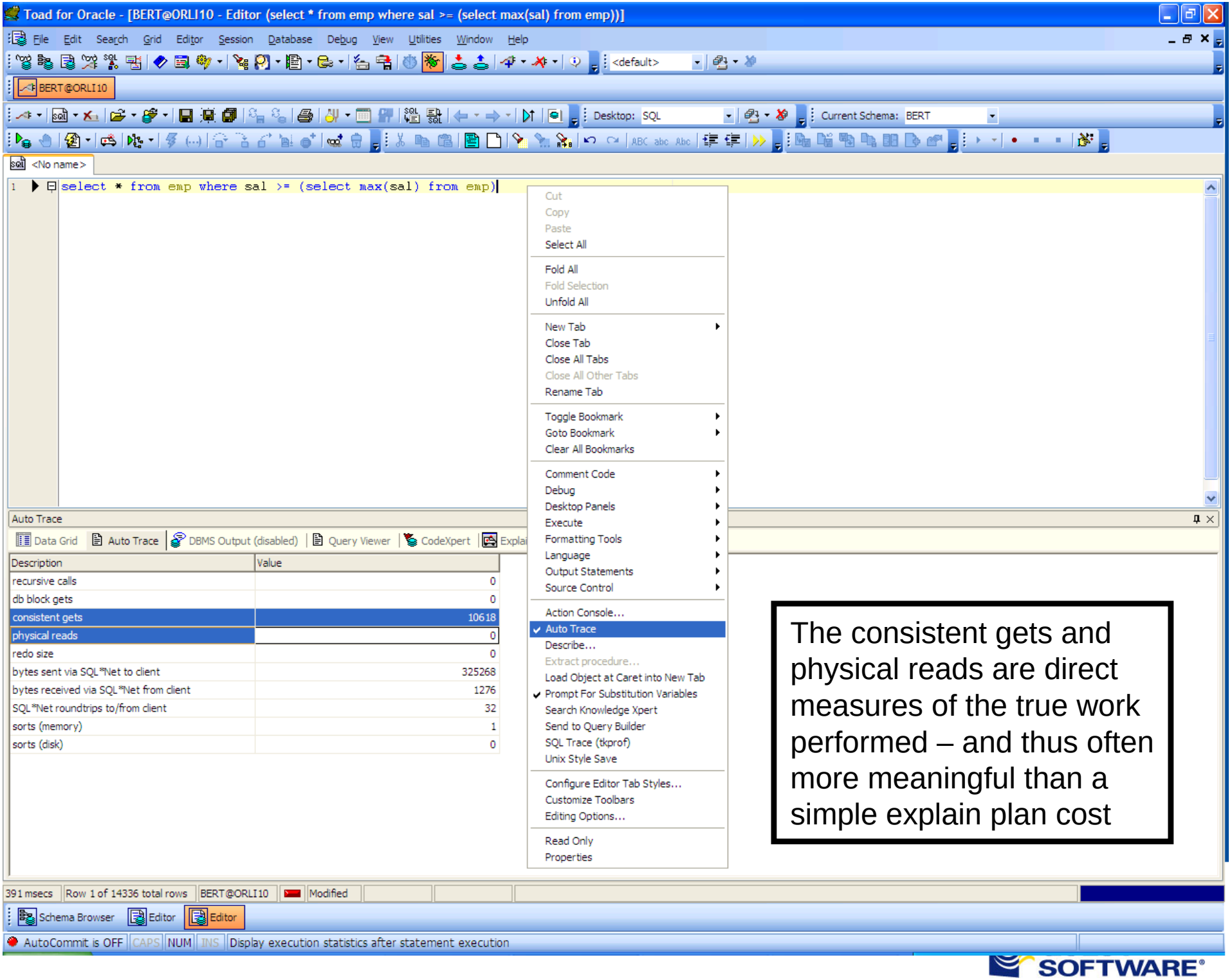
- Explain Plans are the standard Oracle mechanism to peek into the possible “internal algorithm” the database engine might execute for the query (think of it as sort of like program pseudo-code)
- Explain Plans generally require an Oracle “plan table” to hold the explain plan intermediate results
  - Three Options here:
    - Central “plan table” for all users to share – managed by DBA
    - “Plan table” per schema – but be careful if users all login the same
    - “Plan table” per session -
- When doing explain plans manually
  - Method #1
    - EXPLAIN FOR SELECT * FROM emp;
    - SELECT … FROM plan_table WHERE … (fairly complex SQL)
  - Method #2
    - EXPLAIN FOR SELECT * FROM emp;
    - SELECT * FROM table(DBMS_XPLAN.DISPLAY(PLAN_TABLE));



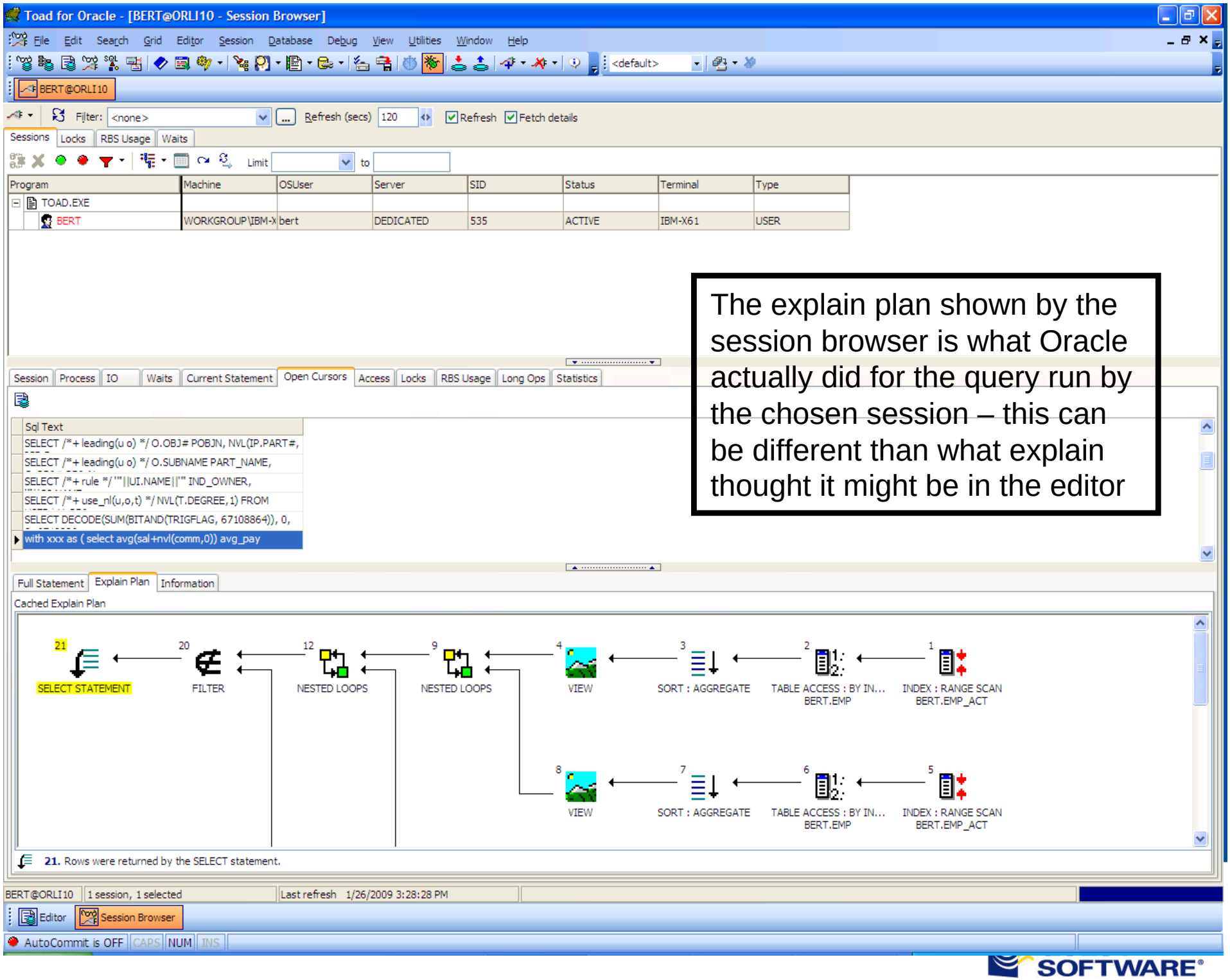
QUEST SOFTWARE®


Toad for Oracle - [BERT@ORLI10 - Editor (select * from emp where ename like 'T%')]
File Edit Search Grid Editor Session Database Debug View Utilities Window Help
BERT@ORLI10
Desktop: SQL
Current Schema: BERT
<No name>
select * from emp where ename like 'T%'
Create/Use Another Plan Table?
The plan table specified in Toad's options does not exist. Enter the name of a plan table to create, or click "Search for Plan Table" to find an existing plan table to use.
Specified plan table to be created:
Plan Table Schema
Plan Table Name
TOAD_PLAN_TABLE
Make it a global temporary table
Tablespace Name (leave blank for default)
Grant to public and create public synonym
Search For Plan Table
OK
Cancel
Data Grid
Auto Trace
DBMS Output (disabled)
Query Viewer
Cancel
Row# EMPNO ENAME JOB MGR HIREDATE
1 7844 TURNER SALESMAN 7698 9/8/1981
2 8944 TURNER SALESMAN 8798 9/5/1992 1500 0 30 N
3 9044 TURNER SALESMAN 8898 9/5/1993 1500 0 30 N
4 9144 TURNER SALESMAN 8998 9/5/1994 1500 0 30 N
5 9244 TURNER SALESMAN 9098 9/5/1995 1500 0 30 N
6 9344 TURNER SALESMAN 9198 9/4/1996 1500 0 30 N
7 9444 TURNER SALESMAN 9298 9/4/1997 1500 0 30 N
8 10444 TURNER SALESMAN 10298 9/2/2007 1500 0 30 N
9 10544 TURNER SALESMAN 10398 9/1/2008 1500 0 30 N
10 10644 TURNER SALESMAN 10498 9/1/2009 1500 0 30 N
11 8544 TURNER SALESMAN 8398 9/6/1988 1500 0 30 N
12 8644 TURNER SALESMAN 8498 9/6/1989 1500 0 30 N
47 msecs
Row 1 of 500 fetched so far (more rows exist)
BERT@ORLI10
Modified
Schema Browser
Editor
Editor
AutoCommit is OFF
CAPS
NUM
INS


Toad for Oracle - [BERT@ORLI10 - Editor (select * from emp where sal >= (select max(sal) from emp))]
File Edit Search Grid Editor Session Database Debug View Utilities Window Help
BERT@ORLI10
select * from emp where sal >= (select max(sal) from emp)
Explain Plan
Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert
4 SELECT STATEMENT
3 TABLE ACCESS : FULL BERT.EMP
2 SORT : AGGREGATE
1 TABLE ACCESS : FULL BERT.EMP
Plan
SELECT STATEMENT ALL_ROWS
Cost: 1,579 Bytes: 834,120 Cardinality: 20,853
3 TABLE ACCESS FULL TABLE BERT.EMP
Cost: 790 Bytes: 834,120 Cardinality: 20,853
2 SORT AGGREGATE
Bytes: 4 Cardinality: 1
1 TABLE ACCESS FULL TABLE BERT.EMP
Cost: 789 Bytes: 1,668,220 Cardinality: 417,055
Display Mode
Adjust Content
Adjust Colors/Fonts
Save Plan to XML
Load Plan from XML
Compare to another plan
Copy to Clipboard
Print
Print Preview
Include SQL Text in Printout
Show Object Usage
Single Record View
Describe
Tree
Plain English
Graphic
MS Graphic
Abbreviate Numbers
Animate
Execution Plan Preferences
Name Visible As Column Sample
Operation TABLE ACCESS
Option FULL
Object Type UNIQUE
Other Tag SERIAL_FROM_REMOTE
Object SYS.TABLE1
Optimizer CHOOSE
Remote UNIX1.CORP.COM
Partition # 1
Partition Info partitions accessed #1 - #4
Cost Sample Text
Bytes Sample Text
Cardinality Sample Text
OK
Cancel
4. Rows were returned by the SELECT statement.
1: 58 Row 1 of 500 fetched so far (more rows exist) BERT@ORLI10 Modified
Schema Browser Editor Editor
AutoCommit is OFF CAPS NUM INS



The consistent gets and physical reads are direct measures of the true work performed – and thus often more meaningful than a simple explain plan cost



The explain plan shown by the session browser is what Oracle actually did for the query run by the chosen session – this can be different than what explain thought it might be in the editor

SQL Guidelines

Rule #1: Watch Indexed WHERE Conditions

Assume address index has columns (city, state)

•non-leading index column references may not use indexes	
•where state = 'TX'	[Depends Oracle on Version]
•where city = 'DALLAS'	**[Index Used]**
•where state = 'TX' **and** city = 'DALLAS'	**[Index Used]**
•NOT, != and <> disable index use	
•where state not in ('TX', 'FL','OH')	[Index Not used]
•where state != 'TX'	[Index Not used]
•NULL value references almost never use indexes (one exception - bitmaps)	
•where state IS NULL	[Index Not used]
•where state IS NOT NULL	[Index Not used]
•expression references can never use indexes	
•where substr(city,1,3) = 'DAL'	[Index Not used]
•where city like 'DAL%'	**[Index Used]**
•where city || state = 'DALLASTX'	[Index Not used]
•where city = 'DALLAS' **and** state = 'TX‘	**[Index Used]**
•where salary * 12 >= 24000	[Index Not used]
•where salary >= 2000	**[Index Used]**

SQL Guidelines

Rule #2: Watch Non-Indexed WHERE Conditions

•Oracle evaluates Non-Indexed conditions linked by **"AND"** bottom up

- **Bad:** select * from address where
  areacode = 972 **and**
  type_nr = (select seq_nr from code_table where type = 'HOME')

- **Good:** select * from address where
  type_nr = (select seq_nr from code_table where type = 'HOME') **and**
  areacode = 972

•Oracle evaluates Non-Indexed conditions linked by **"OR"** top down

- **Bad:** select * from address where
  type_nr = (select seq_nr from code_table where type = 'HOME') **or**
  areacode = 972

- **Good:** select * from address where
  areacode = 972 **or**
  type_nr = (select seq_nr from code_table where type = 'HOME')

SQL Guidelines

Rule #3: Order Table in the FROM Clause (pre-10g)

- important under rule based optimizer, and won't hurt under cost based optimizer
- order FROM clauses in descending order of table sizes based upon row counts
- for example
  - select * from larger table, smaller table
  - select * from larger table, smaller table, smallest table
  - select * from larger table, smaller table, associative table

Note – rule based optimizer only (pre-10g)

SQL Guidelines

Rule #4: Consider IN or UNION in place of OR

- if columns are not indexed, stick with OR
- if columns are indexed, use IN or UNION in place of OR
  - IN example
    - **Bad:** select * from address where
      state = 'TX' **or**
      state = 'FL' **or**
      state = 'OH'
    - **Good:** select * from address where
      state **in** ('TX','FL','OH')
  - UNION example
    - **Bad:** select * from address where
      state = 'TX' **or**
      areacode = 972
    - **Good:** select * from address where
      state = 'TX'

      union

      select * from address where
      areacode = 972

SQL Guidelines

Rule #5: Weigh JOIN versus EXISTS Sub-Query

•use table JOIN instead of EXISTS sub-query

•when the percentage of rows returned from the outer sub-query is high

```
select  e.name, e.phone, e.mailstop
from    employee e, department d
where e.deptno = d.deptno
    and d.status = 'ACTIVE'
```

•use EXISTS sub-query instead of table JOIN

•when the percentage of rows returned from the outer sub-query is low

```
select e.name, e.phone, e.mailstop
from employee e
where e.deptno in (select  d.deptno
                   from   department d
                   where d.status != 'ACTIVE')
```

SQL Guidelines

Rule #6: Consider EXISTS in place of DISTINCT

•avoid joins that use DISTINCT, use EXISTS sub-query instead

•**Bad:** select **distinct** deptno, deptname from emp, dept where
emp.deptno = dept.deptno

•**Good:** select deptno, deptname from dept where
exists (select 'X' from emp where
emp.deptno = dept.deptno)

Note – only has to find one match

SQL Guidelines

Rule #7: Consider NOT EXISTS in place of NOT IN

•avoid sub-queries that use NOT IN, use NOT EXISTS instead

•**Bad:** select * from emp where
deptno **not in** (select deptno from dept where
deptstatus = 'A')

•**Good:** select * from emp where
not exists (select 'X' from dept where
deptstatus = 'A' **and**
dept.deptno = emp.deptno)

Note – only has to find one non-match

SQL Guidelines

Rule #8: Ordering Via the WHERE Clause

- a dummy WHERE clause referencing an indexed column will
  - retrieve all records in ascending order (descending for 8i descending index)
  - not perform a costly sort operation
- **Bad:** select * from address
  order by city
- **Good:** select * from address where
  city > ‘’

SQL Guidelines

Rule #9: Use PL/SQL to reduce network traffic

•Utilize PL/SQL to group related SQL commands and thereby reduce network traffic

•**Bad:**

```
select city_name, state_code
   into :v_city, :v_sate
   from zip_codes where zip_code = '75022';

insert into customer ('Bert Scalzo','75022', :v_city, v_state);
```

•**Good:**

```
begin
   select city_name, state_code
      into :v_city, :v_sate
      from zip_codes where zip_code = '75022';
   insert into customer ('Bert Scalzo','75022', :v_city, v_state);
end;
/
```

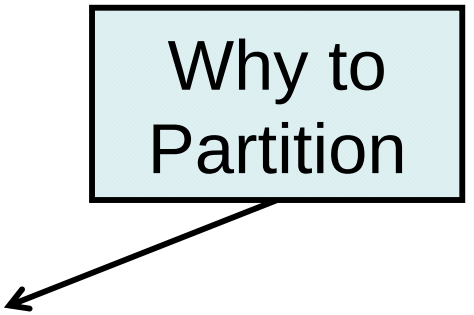
SQL Guidelines

Rule #10: Partition Large Tables and Indexes

- **Partition Elimination**
- **Partition-Wise Join (requires Parallel too)**
- NOTE: Do not expect that merely partitioning will solve some major performance problem, it should merely make an incremental improvement to a non-partitioned explain plan. Read that as partitioning can make an already good explain plan even better.

Partitioning Benefits: Opinion (Mine)

- Manageability 40%
- Availability 20%
- Capacity Management 20%
- **Performance** **20%**



- Don’t over-sell/over-expect the performance aspect
- Need to experiment for best approach for a database
- Better to take longer at the start to get right, because very often it’s far too expensive to change afterwards

Partition Pruning (Restriction Based) ☆

- From Docs: In partition pruning, the optimizer analyzes FROM and WHERE clauses in SQL statements to eliminate unneeded partitions when building the partition access list. This enables Oracle Database to perform operations only on those partitions that are relevant ...

- "Divide and Conquer" for performance
  - Sometimes can yield order of magnitude improvement
  - But once again, best not to oversell and/or over-expect

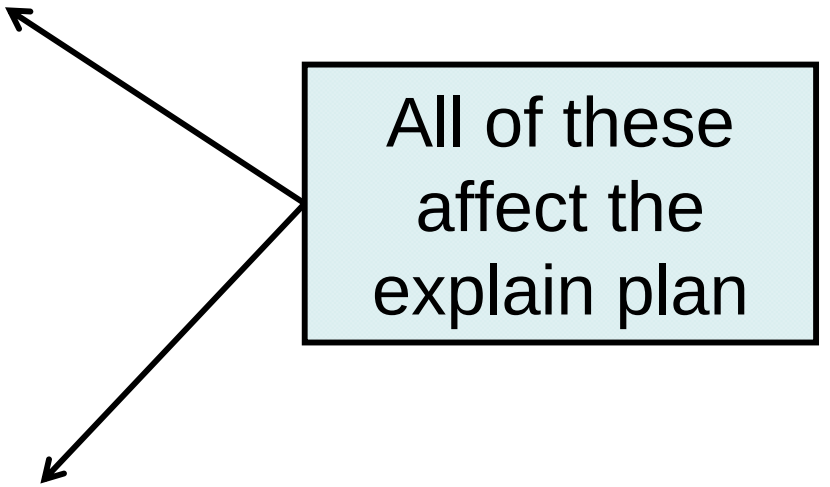
- Some Potential Issues to be aware of:
  - SQL*Plus Auto-Trace can sometimes miss partition pruning
  - "Old Style" Explain Plans via simple SELECT has issues too
  - Best to always use **DBMS_XPLAN** and/or **SQL_TRACE** ←

Partition-Wise Join (Multi-Object Based)

- From Docs: Partition-wise joins reduce query response time by minimizing the amount of data exchanged among parallel execution servers when joins execute in parallel. This significantly reduces response time & improves the use of both CPU & memory resources.

- Different Flavors:
  - Full – Single to Single
  - Full – Composite to Single
  - Full – Composite to Composite
  - Partial – Single
  - Partial – Composite

- Indexing Strategy Counts
  - Local Prefixed/Non-Prefixed
  - Global



Picture Worth 1000 Words (from Docs)

Simple Mantra: Subdivide the work into equally paired chunks, then perform all that work using many parallel processes

Figure 4-1 Parallel Execution of a Full Partition-wise Join

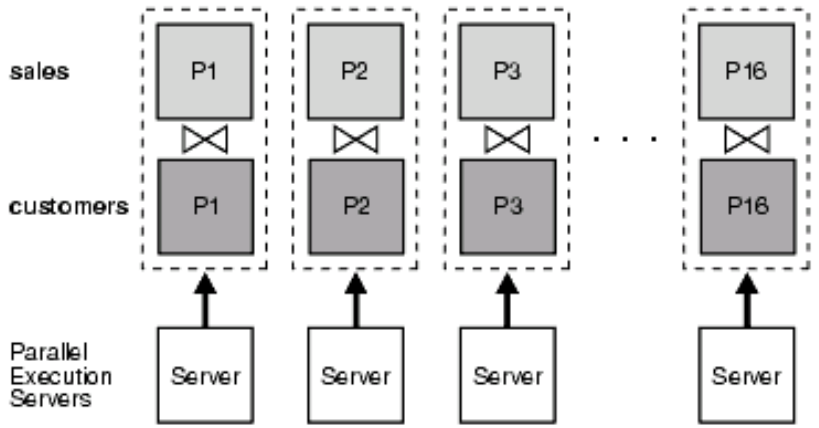
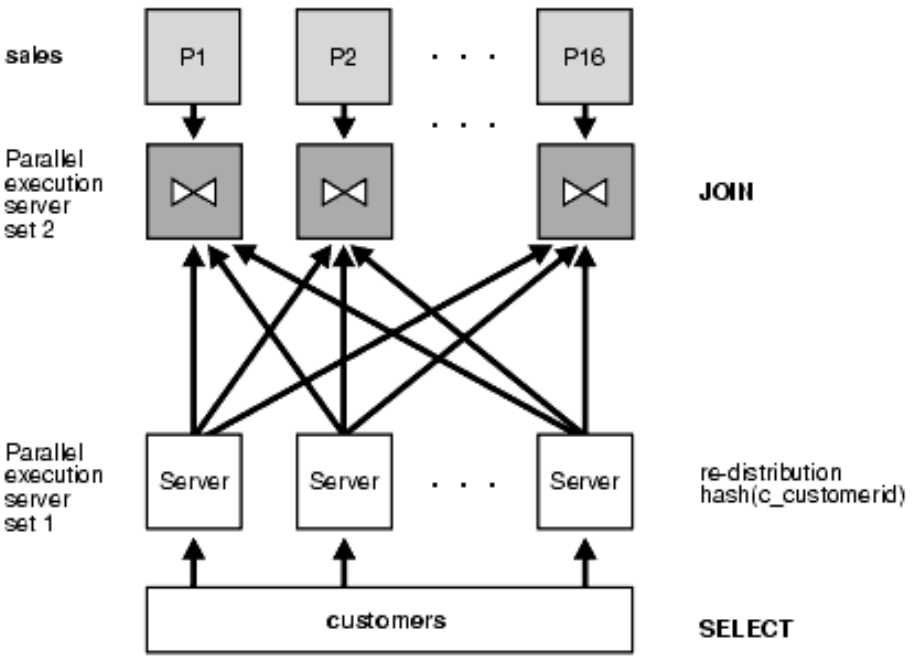


Figure 4-3 Partial Partition-Wise Join



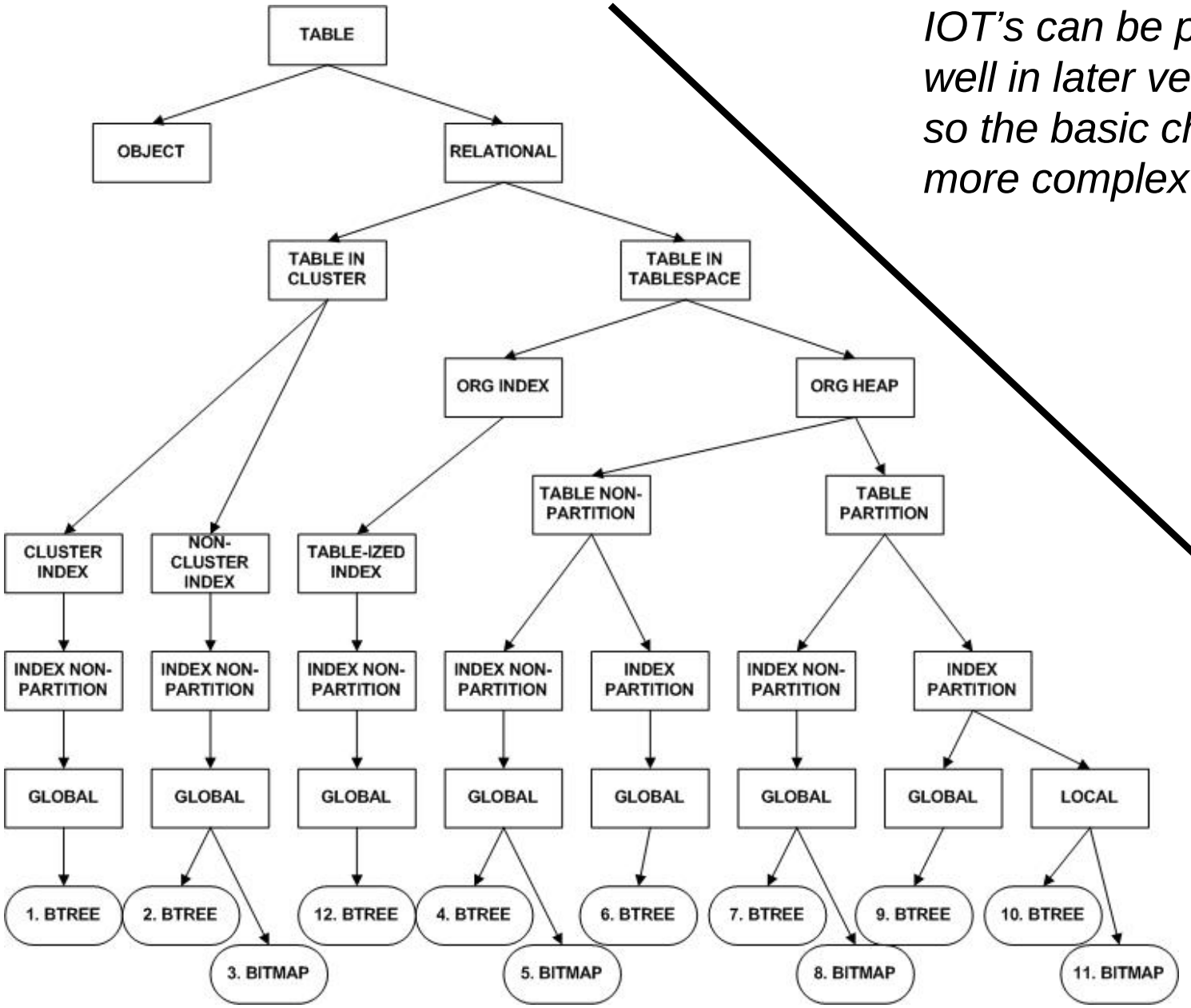
Make sure not to over-allocate CPU's – remember there will also be concurrent workload

Partitioning History (from Oracle 11G training+)

Oracle 5	Before Tablespaces – we had partitions ☺
Oracle 7	Partition Views – really more of a cheat ☹

	Core functionality	Performance	Manageability
Oracle8	Range partitioning Global range indexes	"Static" partition pruning	Basic maintenance operations: add, drop, exchange
Oracle8*i*	Hash and composite range-hash partitioning	Partition-wise joins "Dynamic" pruning	Merge operation
Oracle9*i*	List partitioning		Global index maintenance
Oracle9*i* R2	Composite range-list partitioning	Fast partition split	
Oracle10*g*	Global hash indexes		Local Index maintenance
Oracle10*g* R2	1M partitions per table	"Multi-dimensional" pruning	Fast drop table
Oracle Database 11*g*	More composite choices REF Partitioning Virtual Column		Interval Partitioning Partition Advisor

Partitioning Options – Part 1



IOT's can be partitioned as well in later versions of Oracle, so the basic choices are even more complex than this…

Partitioning Options – Part 2

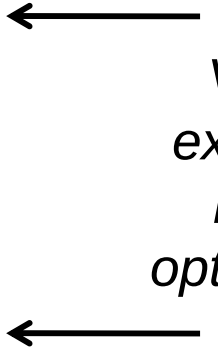
Prior to 11G: Oracle White Paper: 2007 Partitioning in Oracle Database 11g

Partitioning Strategy	Data Distribution	Sample Business Case
Range Partitioning	Based on consecutive ranges of values.	• Orders table range partitioned by order_date
List Partitioning	Based on unordered lists of values.	• Orders table list partitioned by country
Hash Partitioning	Based on a hash algorithm.	• Orders table hash partitioned by customer_id
Composite Partitioning • Range-Range • Range-List • Range-Hash • List-List • List-Range • List-Hash	Based on a combination of two of the above-mentioned basic techniques of Range, List, Hash, and Interval Partitioning	• Orders table is range partitioned by order_date and sub-partitioned by hash on customer_id • Orders table is range partitioned by order_date and sub-partitioned by range on shipment_date

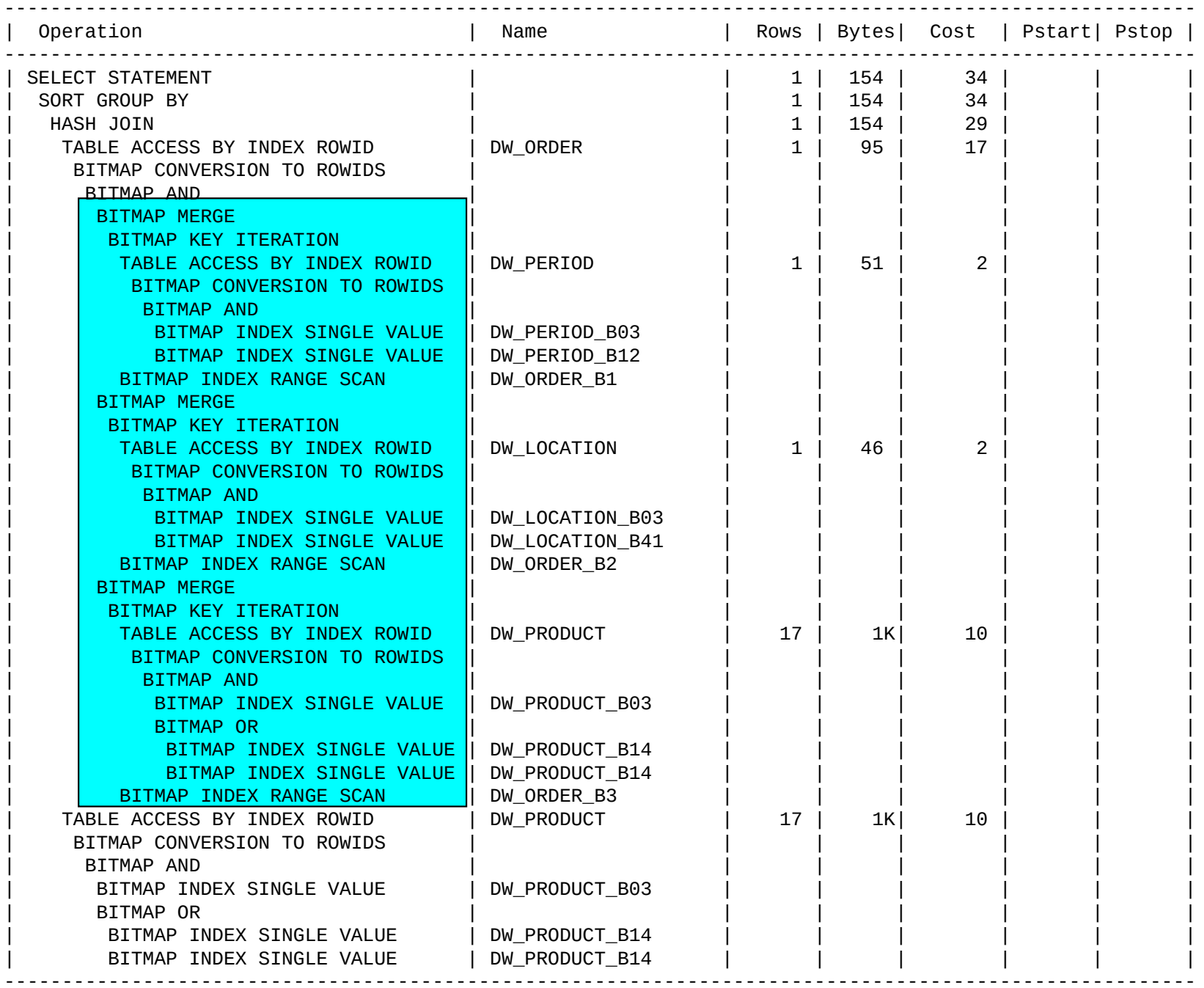
Partitioning Options – Part 3

Post 11G: Oracle White Paper: 2007 Partitioning in Oracle Database 11g

Partitioning Extension	Partitioning Key	Sample Business Case
Interval Partitioning • Interval • Interval-Range • Interval-List • Interval-Hash	An extension to Range Partition. Defined by an interval, providing equi-width ranges. With the exception of the first partition all partitions are automatically created on-demand when matching data arrives.	• Orders table partitioned by order_date with a predefined daily interval, starting with '01-Jan-2007'
REF Partitioning	Partitioning for a child table is inherited from the parent table through a primary key – foreign key relationship. The partitioning keys are not stored in actual columns in the child table.	• (Parent) Orders table range partitioned by order_date and inherits the partitioning technique to (child) order lines table. Column order_date is only present in the parent orders table
Virtual column based Partitioning	Defined by one of the above-mentioned partition techniques and the partitioning key is based on a virtual column. Virtual columns are not stored on disk and only exist as metadata.	• Orders table has a virtual column that derives the sales region based on the first three digits of the customer account number. The orders table is then list partitioned by sales region.



Very exciting new options…



Operation	Name	Rows	Bytes	Cost	Pstart	Pstop
SELECT STATEMENT		1	154	34		
SORT GROUP BY		1	154	34		
HASH JOIN		1	154	29		
TABLE ACCESS BY INDEX ROWID	DW_ORDER	1	95	17		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP MERGE						
BITMAP KEY ITERATION						
TABLE ACCESS BY INDEX ROWID	DW_PERIOD	1	51	2		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PERIOD_B03					
BITMAP INDEX SINGLE VALUE	DW_PERIOD_B12					
BITMAP INDEX RANGE SCAN	DW_ORDER_B1					
BITMAP MERGE						
BITMAP KEY ITERATION						
TABLE ACCESS BY INDEX ROWID	DW_LOCATION	1	46	2		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_LOCATION_B03					
BITMAP INDEX SINGLE VALUE	DW_LOCATION_B41					
BITMAP INDEX RANGE SCAN	DW_ORDER_B2					
BITMAP MERGE						
BITMAP KEY ITERATION						
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03					
BITMAP OR						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX RANGE SCAN	DW_ORDER_B3					
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03					
BITMAP OR						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					

Non-Partitioned, Non-Parallel explain plan

Operation	Name	Rows	Bytes	Cost	Pstart	Pstop
SELECT STATEMENT		1	154	35		
SORT GROUP BY		1	154	35		
SORT GROUP BY		1	154	35		
HASH JOIN		1	154	30		
PARTITION RANGE ALL					1	10
TABLE ACCESS BY LOCAL INDEX ROWID	DW_ORDER_PART	0	20	18	1	10
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP MERGE						
BITMAP KEY ITERATION						
SORT BUFFER						
TABLE ACCESS BY INDEX ROWID	DW_PERIOD	1	51	2		
BITMAP CONVERSION TO ROWID						
BITMAP AND						
BITMAP INDEX SINGLE VALU	DW_PERIOD_B03					
BITMAP INDEX SINGLE VALU	DW_PERIOD_B12					
BITMAP INDEX RANGE SCAN	DW_ORDER_PART_B1				1	10
BITMAP MERGE						
BITMAP KEY ITERATION						
SORT BUFFER						
TABLE ACCESS BY INDEX ROWID	DW_LOCATION	1	46	2		
BITMAP CONVERSION TO ROWID						
BITMAP AND						
BITMAP INDEX SINGLE VALU	DW_LOCATION_B03					
BITMAP INDEX SINGLE VALU	DW_LOCATION_B41					
BITMAP INDEX RANGE SCAN	DW_ORDER_PART_B2				1	10
BITMAP MERGE						
BITMAP KEY ITERATION						
SORT BUFFER						
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10		
BITMAP CONVERSION TO ROWID						
BITMAP AND						
BITMAP INDEX SINGLE VALU	DW_PRODUCT_B03					
BITMAP OR						
BITMAP INDEX SINGLE VAL	DW_PRODUCT_B14					
BITMAP INDEX SINGLE VAL	DW_PRODUCT_B14					
BITMAP INDEX RANGE SCAN	DW_ORDER_PART_B3				1	10
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03					
BITMAP OR						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					

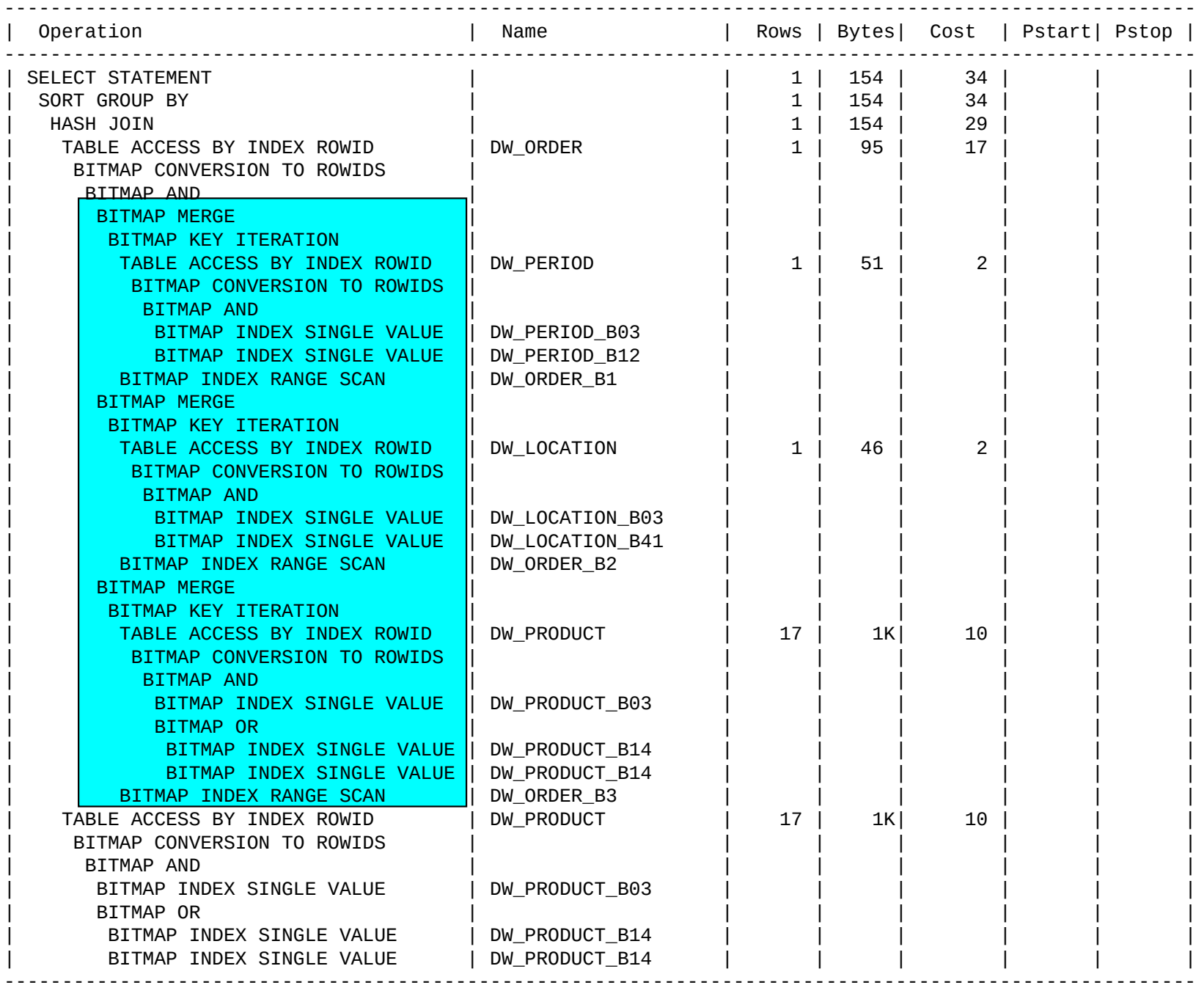
Partitioned, Non-Parallel explain plan

SQL Guidelines

Rule #11: Serial Explain Plans, then Parallel (maybe)

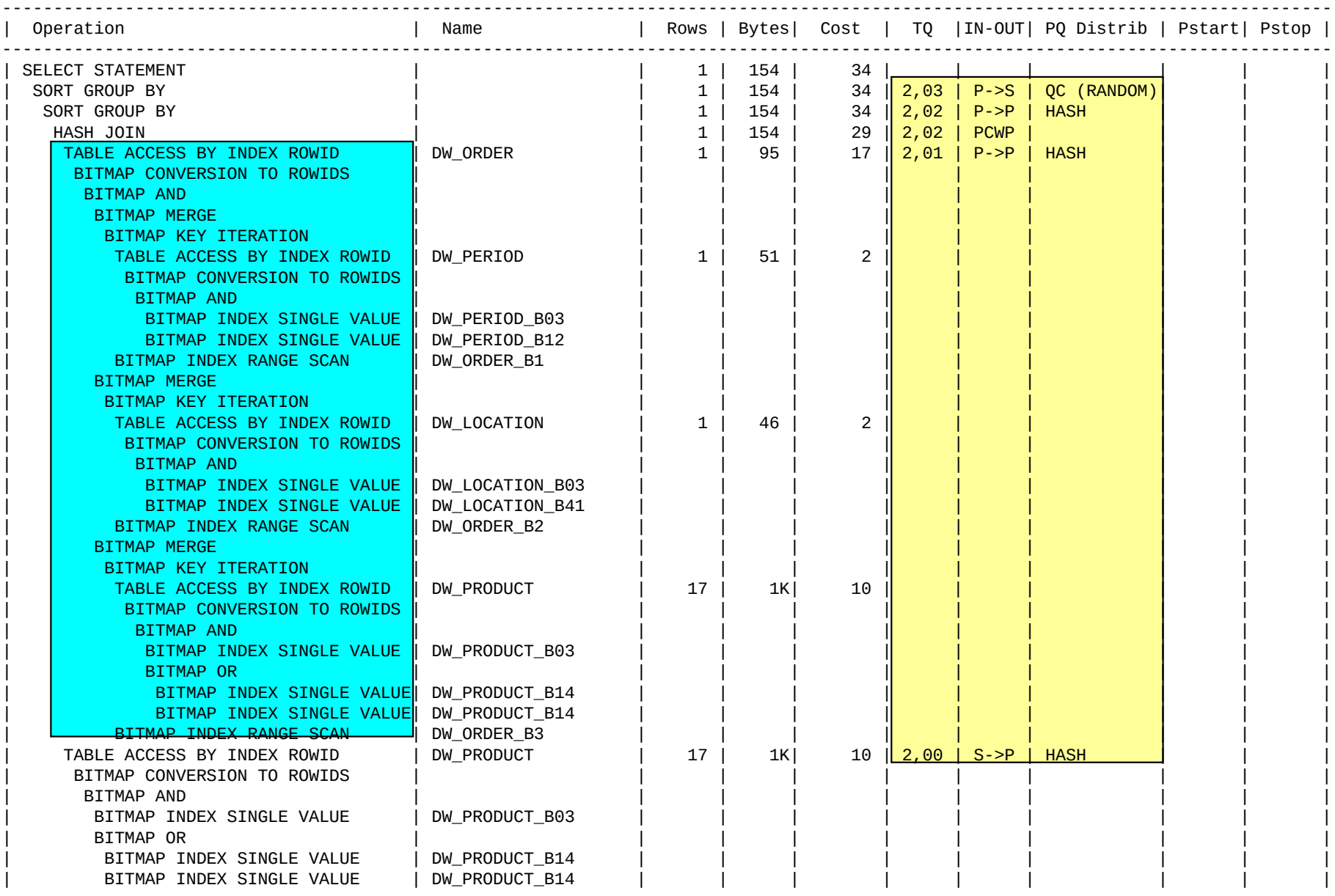
- **Parallel Full Table Scan**
- **Parallel Index Scan**
- **Parallel Fast Full Scan (FFS Index Scan)**
- NOTE: Do not expect that merely parallelizing will solve some major performance problem, it should merely make an incremental improvement to a non-paralell (i.e. serial) explain plan. Read that as parallel can make an already good explain plan even better.

- Parallel processing is controlled as follows:
  - Query has /*+parallel*/ hint
    - Some shops do NOT favor hints
      - What if database version changes
      - What happens if statistics change
      - Other questionable future scenarios
    - Cannot add hints to pre-canned applications
  - Object (table or index) has parallel degree
  - Database instance parameter for parallel
- For RAC, parallel can also span the RAC nodes too



Operation	Name	Rows	Bytes	Cost	Pstart	Pstop
SELECT STATEMENT		1	154	34		
SORT GROUP BY		1	154	34		
HASH JOIN		1	154	29		
TABLE ACCESS BY INDEX ROWID	DW_ORDER	1	95	17		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP MERGE						
BITMAP KEY ITERATION						
TABLE ACCESS BY INDEX ROWID	DW_PERIOD	1	51	2		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PERIOD_B03					
BITMAP INDEX SINGLE VALUE	DW_PERIOD_B12					
BITMAP INDEX RANGE SCAN	DW_ORDER_B1					
BITMAP MERGE						
BITMAP KEY ITERATION						
TABLE ACCESS BY INDEX ROWID	DW_LOCATION	1	46	2		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_LOCATION_B03					
BITMAP INDEX SINGLE VALUE	DW_LOCATION_B41					
BITMAP INDEX RANGE SCAN	DW_ORDER_B2					
BITMAP MERGE						
BITMAP KEY ITERATION						
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03					
BITMAP OR						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX RANGE SCAN	DW_ORDER_B3					
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10		
BITMAP CONVERSION TO ROWIDS						
BITMAP AND						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03					
BITMAP OR						
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14					

Non-Partitioned, Non-Parallel explain plan



Operation	Name	Rows	Bytes	Cost	TQ	IN-OUT	PQ Distrib	Pstart	Pstop
SELECT STATEMENT		1	154	34					
SORT GROUP BY		1	154	34	2,03	P->S	QC (RANDOM)		
SORT GROUP BY		1	154	34	2,02	P->P	HASH		
HASH JOIN		1	154	29	2,02	PCWP			
TABLE ACCESS BY INDEX ROWID	DW_ORDER	1	95	17	2,01	P->P	HASH		
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP MERGE									
BITMAP KEY ITERATION									
TABLE ACCESS BY INDEX ROWID	DW_PERIOD	1	51	2					
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP INDEX SINGLE VALUE	DW_PERIOD_B03								
BITMAP INDEX SINGLE VALUE	DW_PERIOD_B12								
BITMAP INDEX RANGE SCAN	DW_ORDER_B1								
BITMAP MERGE									
BITMAP KEY ITERATION									
TABLE ACCESS BY INDEX ROWID	DW_LOCATION	1	46	2					
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP INDEX SINGLE VALUE	DW_LOCATION_B03								
BITMAP INDEX SINGLE VALUE	DW_LOCATION_B41								
BITMAP INDEX RANGE SCAN	DW_ORDER_B2								
BITMAP MERGE									
BITMAP KEY ITERATION									
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10					
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03								
BITMAP OR									
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14								
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14								
BITMAP INDEX RANGE SCAN	DW_ORDER_B3								
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10	2,00	S->P	HASH		
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03								
BITMAP OR									
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14								
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14								

Non-Partitioned, Parallel explain plan

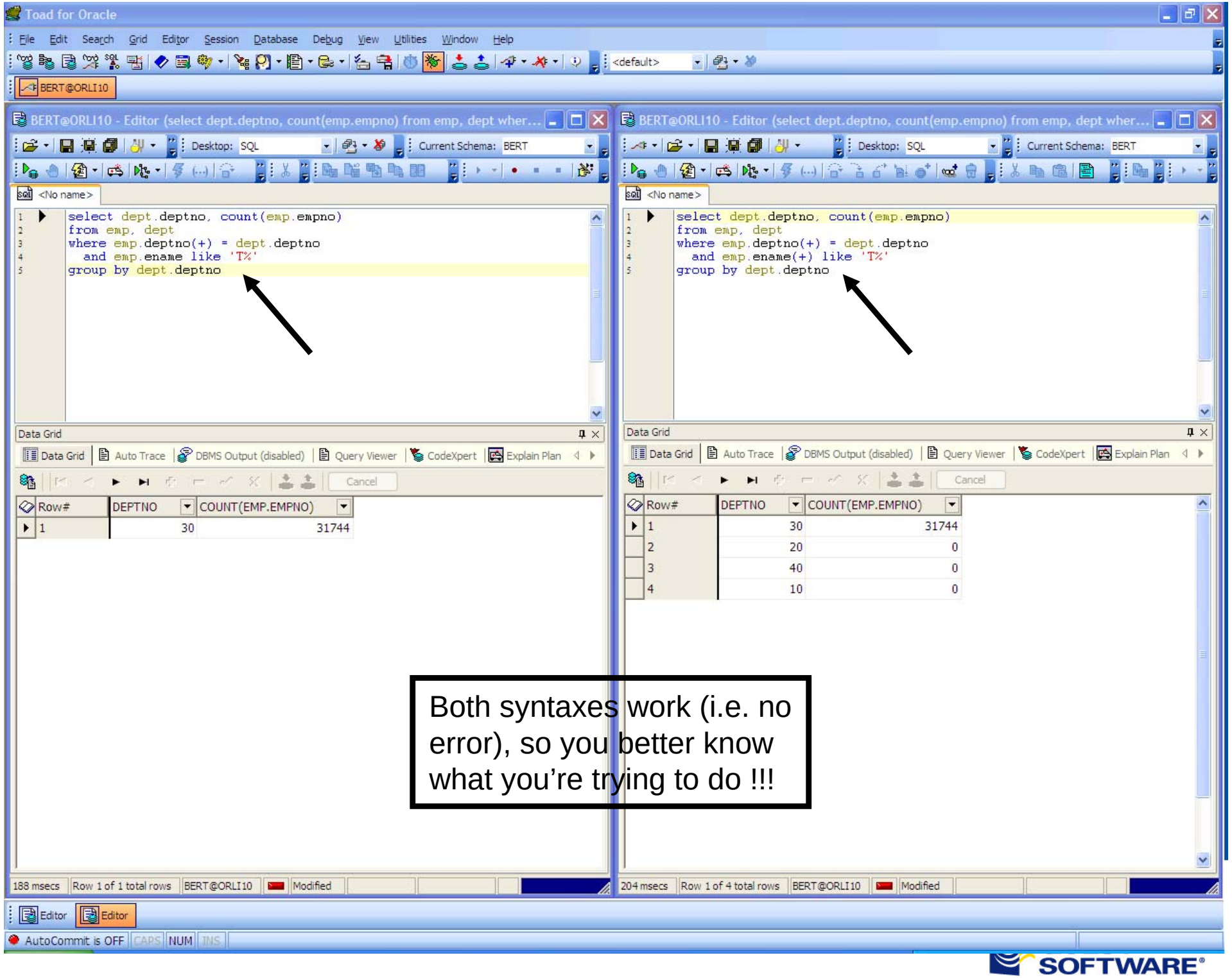
Operation	Name	Rows	Bytes	Cost	TQ	IN-OUT	PQ Distrib	Pstart	Pstop
SELECT STATEMENT		1	154	34					
SORT GROUP BY		1	154	34	5,03	P->S	QC (RANDOM)		
SORT GROUP BY		1	154	34	5,02	P->P	HASH		
HASH JOIN		1	154	29	5,02	PCWP			
PARTITION RANGE ALL					5,02	PCWP		1	10
TABLE ACCESS BY LOCAL INDEX ROWID	DW_ORDER_PART	0	20	18	5,01	P->P	HASH	1	10
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP MERGE									
BITMAP KEY ITERATION									
SORT BUFFER									
TABLE ACCESS BY INDEX ROWID	DW_PERIOD	1	51	2					
BITMAP CONVERSION TO ROWID									
BITMAP AND									
BITMAP INDEX SINGLE VALU	DW_PERIOD_B03								
BITMAP INDEX SINGLE VALU	DW_PERIOD_B12								
BITMAP INDEX RANGE SCAN	DW_ORDER_PART_B1							1	10
BITMAP MERGE									
BITMAP KEY ITERATION									
SORT BUFFER									
TABLE ACCESS BY INDEX ROWID	DW_LOCATION	1	46	2					
BITMAP CONVERSION TO ROWID									
BITMAP AND									
BITMAP INDEX SINGLE VALU	DW_LOCATION_B03								
BITMAP INDEX SINGLE VALU	DW_LOCATION_B41								
BITMAP INDEX RANGE SCAN	DW_ORDER_PART_B2							1	10
BITMAP MERGE									
BITMAP KEY ITERATION									
SORT BUFFER									
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10					
BITMAP CONVERSION TO ROWID									
BITMAP AND									
BITMAP INDEX SINGLE VALU	DW_PRODUCT_B03								
BITMAP OR									
BITMAP INDEX SINGLE VAL	DW_PRODUCT_B14								
BITMAP INDEX SINGLE VAL	DW_PRODUCT_B14								
BITMAP INDEX RANGE SCAN	DW_ORDER_PART_B3							1	10
TABLE ACCESS BY INDEX ROWID	DW_PRODUCT	17	1K	10	5,00	S->P	HASH		
BITMAP CONVERSION TO ROWIDS									
BITMAP AND									
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B03								
BITMAP OR									
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14								
BITMAP INDEX SINGLE VALUE	DW_PRODUCT_B14								

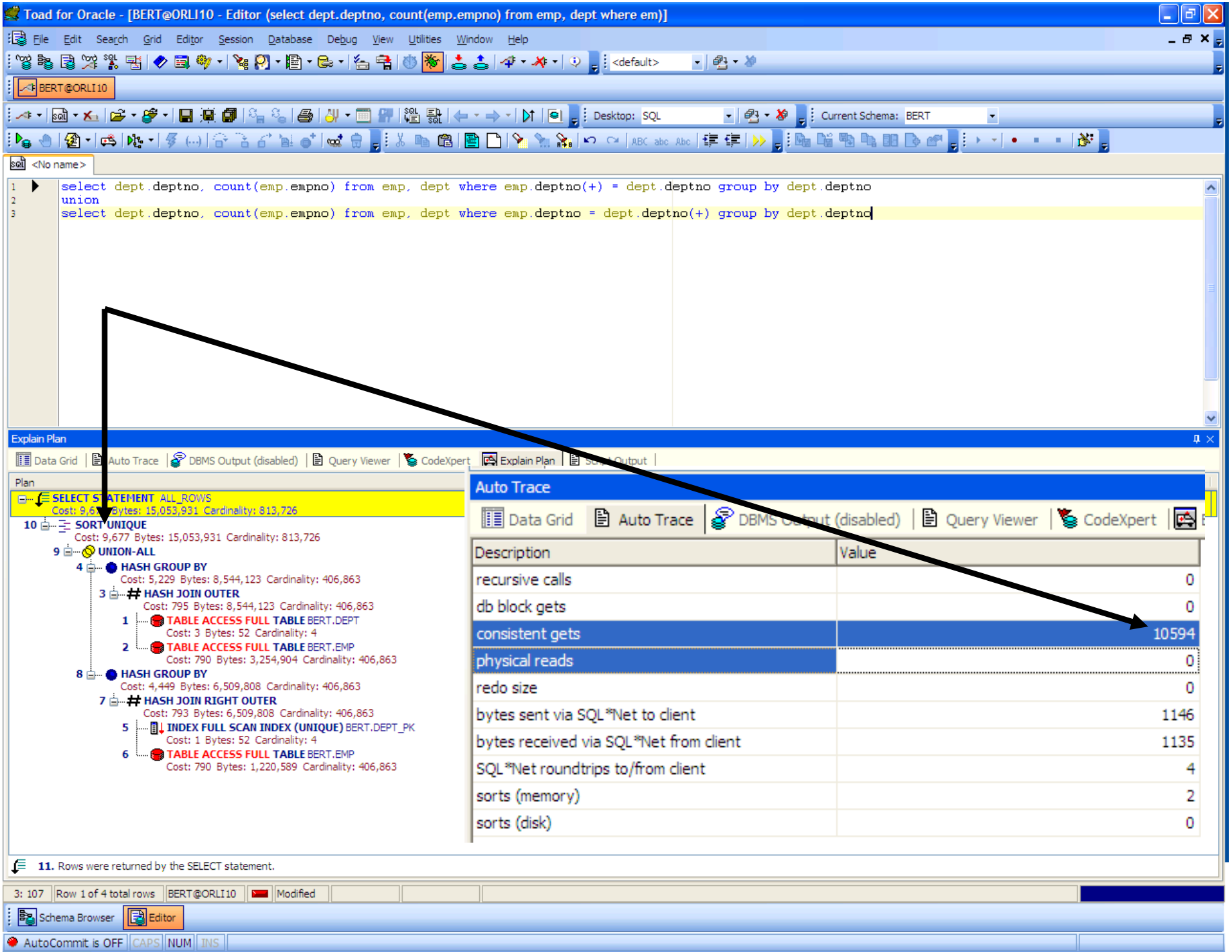
Partitioned, Parallel explain plan

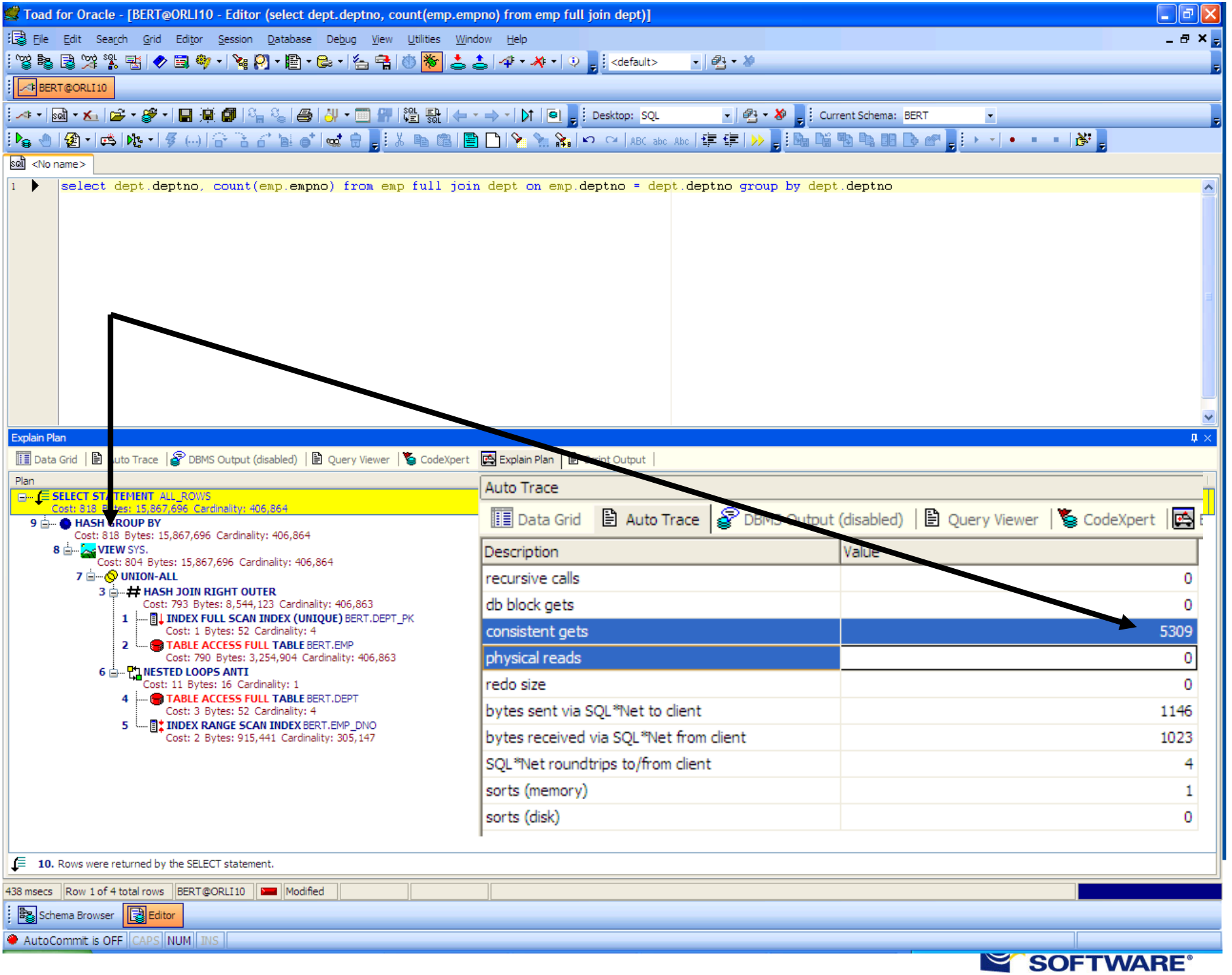
SQL Guidelines

Rule #12: Use ANSI 99 JOIN Syntax – ALWAYS !!!

- Oracle proprietary (+) syntax has problems:
  - Cannot do a FULL JOIN efficiently
    - See slides that follow the next
  - Outer JOIN syntax prone to user error
    - You must specify (+) in the WHERE clause for both
      - The JOIN condition(s)
      - All other references to that table (source of many mistakes)


Toad for Oracle
File Edit Search Grid Editor Session Database Debug View Utilities Window Help
BERT@ORLI10
BERT@ORLI10 - Editor (select dept.deptno, count(emp.empno) from emp, dept wher...
Desktop: SQL
Current Schema: BERT
<No name>
select dept.deptno, count(emp.empno)
from emp, dept
where emp.deptno(+) = dept.deptno
and emp.ename like 'T%'
group by dept.deptno
Data Grid
Auto Trace
DBMS Output (disabled)
Query Viewer
CodeXpert
Explain Plan
Cancel
Row# DEPTNO COUNT(EMP.EMPNO)
1 30 31744
188 msecs
Row 1 of 1 total rows
BERT@ORLI10
Modified
BERT@ORLI10 - Editor (select dept.deptno, count(emp.empno) from emp, dept wher...
Desktop: SQL
Current Schema: BERT
<No name>
select dept.deptno, count(emp.empno)
from emp, dept
where emp.deptno(+) = dept.deptno
and emp.ename(+) like 'T%'
group by dept.deptno
Data Grid
Auto Trace
DBMS Output (disabled)
Query Viewer
CodeXpert
Explain Plan
Cancel
Row# DEPTNO COUNT(EMP.EMPNO)
1 30 31744
2 20 0
3 40 0
4 10 0
204 msecs
Row 1 of 4 total rows
BERT@ORLI10
Modified
Editor
Editor
AutoCommit is OFF
CAPS
NUM
INS
Both syntaxes work (i.e. no error), so you better know what you're trying to do !!!


Toad for Oracle - [BERT@ORLI10 - Editor (select dept.deptno, count(emp.empno) from emp, dept where em)]
File Edit Search Grid Editor Session Database Debug View Utilities Window Help
BERT@ORLI10
Desktop: SQL
Current Schema: BERT
<No name>
select dept.deptno, count(emp.empno) from emp, dept where emp.deptno(+) = dept.deptno group by dept.deptno
union
select dept.deptno, count(emp.empno) from emp, dept where emp.deptno = dept.deptno(+) group by dept.deptno
Explain Plan
Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert | Explain Plan
Plan
SELECT STATEMENT ALL_ROWS
Cost: 9,6 Bytes: 15,053,931 Cardinality: 813,726
10 SORT UNIQUE
Cost: 9,677 Bytes: 15,053,931 Cardinality: 813,726
9 UNION-ALL
4 HASH GROUP BY
Cost: 5,229 Bytes: 8,544,123 Cardinality: 406,863
3 HASH JOIN OUTER
Cost: 795 Bytes: 8,544,123 Cardinality: 406,863
1 TABLE ACCESS FULL TABLE BERT.DEPT
Cost: 3 Bytes: 52 Cardinality: 4
2 TABLE ACCESS FULL TABLE BERT.EMP
Cost: 790 Bytes: 3,254,904 Cardinality: 406,863
8 HASH GROUP BY
Cost: 4,449 Bytes: 6,509,808 Cardinality: 406,863
7 HASH JOIN RIGHT OUTER
Cost: 793 Bytes: 6,509,808 Cardinality: 406,863
5 INDEX FULL SCAN INDEX (UNIQUE) BERT.DEPT_PK
Cost: 1 Bytes: 52 Cardinality: 4
6 TABLE ACCESS FULL TABLE BERT.EMP
Cost: 790 Bytes: 1,220,589 Cardinality: 406,863
Auto Trace
Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert
Description | Value
recursive calls | 0
db block gets | 0
consistent gets | 10594
physical reads | 0
redo size | 0
bytes sent via SQL*Net to client | 1146
bytes received via SQL*Net from client | 1135
SQL*Net roundtrips to/from client | 4
sorts (memory) | 2
sorts (disk) | 0
11. Rows were returned by the SELECT statement.
3: 107 | Row 1 of 4 total rows | BERT@ORLI10 | Modified
Schema Browser | Editor
AutoCommit is OFF | CAPS | NUM | INS


Toad for Oracle - [BERT@ORLI10 - Editor (select dept.deptno, count(emp.empno) from emp full join dept)]
File Edit Search Grid Editor Session Database Debug View Utilities Window Help
BERT@ORLI10
Desktop: SQL
Current Schema: BERT
<No name>
select dept.deptno, count(emp.empno) from emp full join dept on emp.deptno = dept.deptno group by dept.deptno
Explain Plan
Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert | Explain Plan
Plan
SELECT STATEMENT ALL_ROWS
Cost: 818 Bytes: 15,867,696 Cardinality: 406,864
9 HASH GROUP BY
Cost: 818 Bytes: 15,867,696 Cardinality: 406,864
8 VIEW SYS.
Cost: 804 Bytes: 15,867,696 Cardinality: 406,864
7 UNION-ALL
3 HASH JOIN RIGHT OUTER
Cost: 793 Bytes: 8,544,123 Cardinality: 406,863
1 INDEX FULL SCAN INDEX (UNIQUE) BERT.DEPT_PK
Cost: 1 Bytes: 52 Cardinality: 4
2 TABLE ACCESS FULL TABLE BERT.EMP
Cost: 790 Bytes: 3,254,904 Cardinality: 406,863
6 NESTED LOOPS ANTI
Cost: 11 Bytes: 16 Cardinality: 1
4 TABLE ACCESS FULL TABLE BERT.DEPT
Cost: 3 Bytes: 52 Cardinality: 4
5 INDEX RANGE SCAN INDEX BERT.EMP_DNO
Cost: 2 Bytes: 915,441 Cardinality: 305,147
Auto Trace
Data Grid | Auto Trace | DBMS Output (disabled) | Query Viewer | CodeXpert
Description
Value
recursive calls 0
db block gets 0
consistent gets 5309
physical reads 0
redo size 0
bytes sent via SQL*Net to client 1146
bytes received via SQL*Net from client 1023
SQL*Net roundtrips to/from client 4
sorts (memory) 1
sorts (disk) 0
10. Rows were returned by the SELECT statement.
438 msecs
Row 1 of 4 total rows
BERT@ORLI10
Modified
Schema Browser
Editor
AutoCommit is OFF
CAPS
NUM
INS

Wow – this is becoming overwhelming

I could go on and list probably another two dozen or so "Best Practices" SQL Tuning and Optimization rules, but you should already be seeing my point – there is a lot of tuning stuff to remember while trying to get your job done.

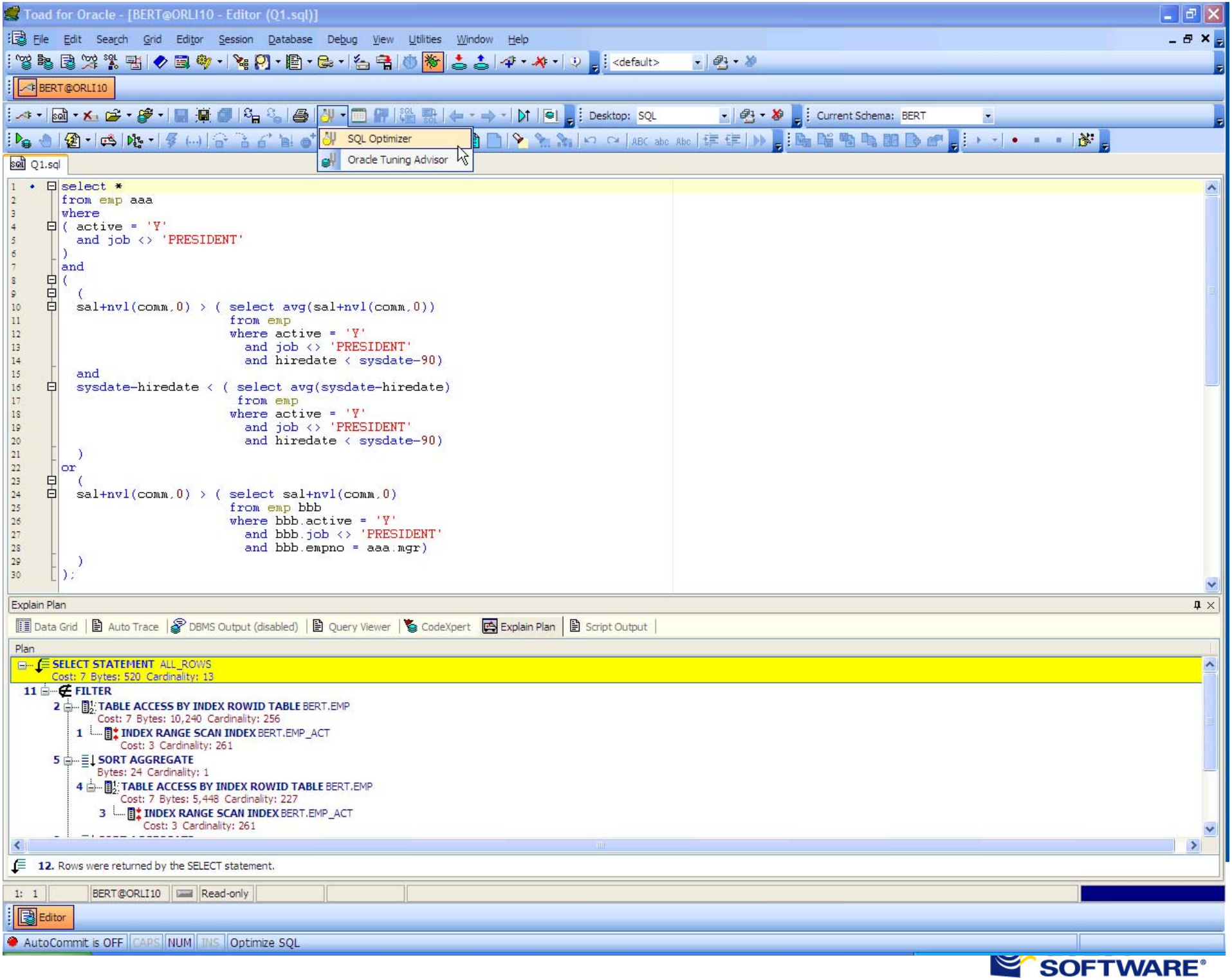
You should focus on being **effective** – i.e. the SQL does what the business and/or user requirements mandate.

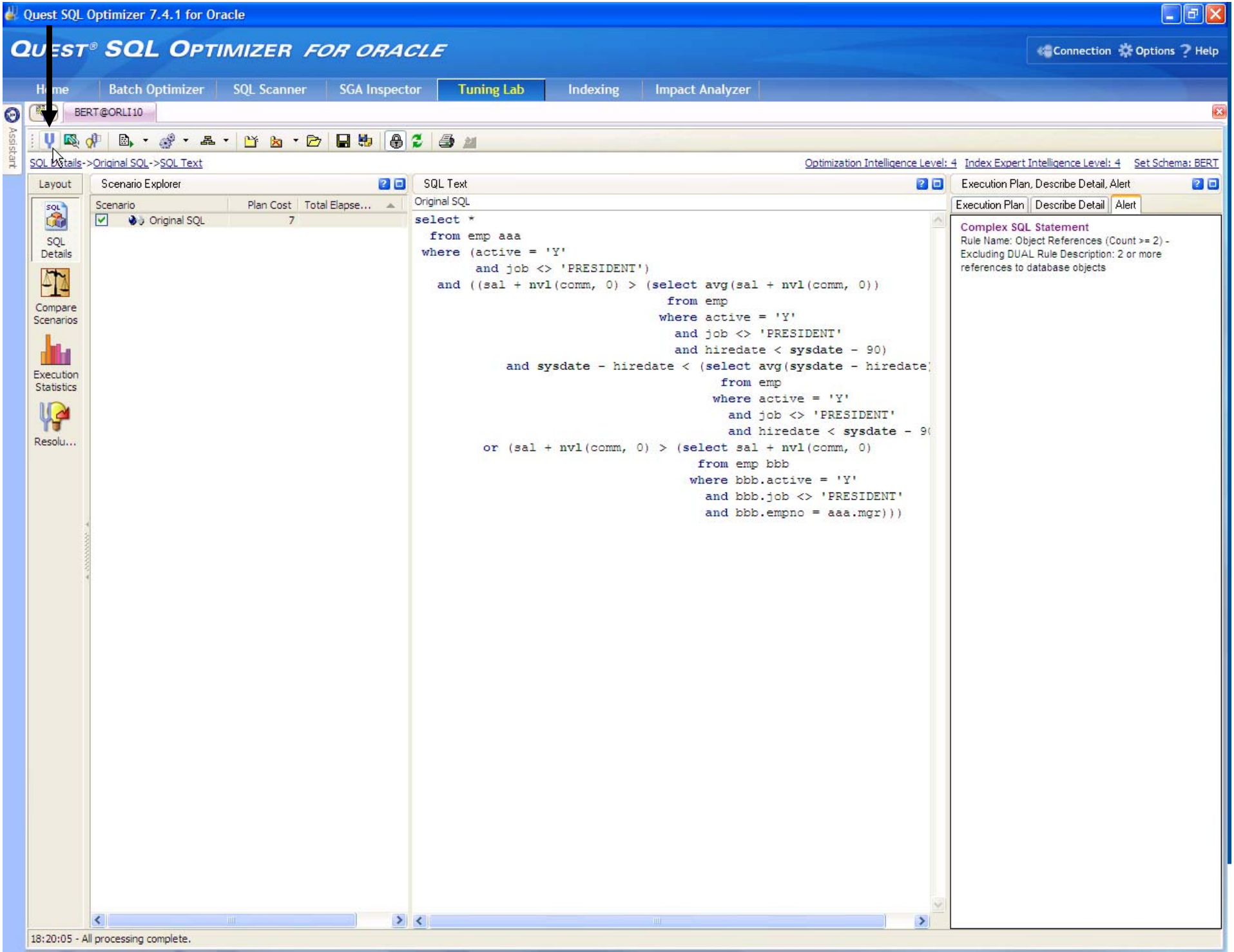
You should let Toad handle making you SQL **efficient** !!!

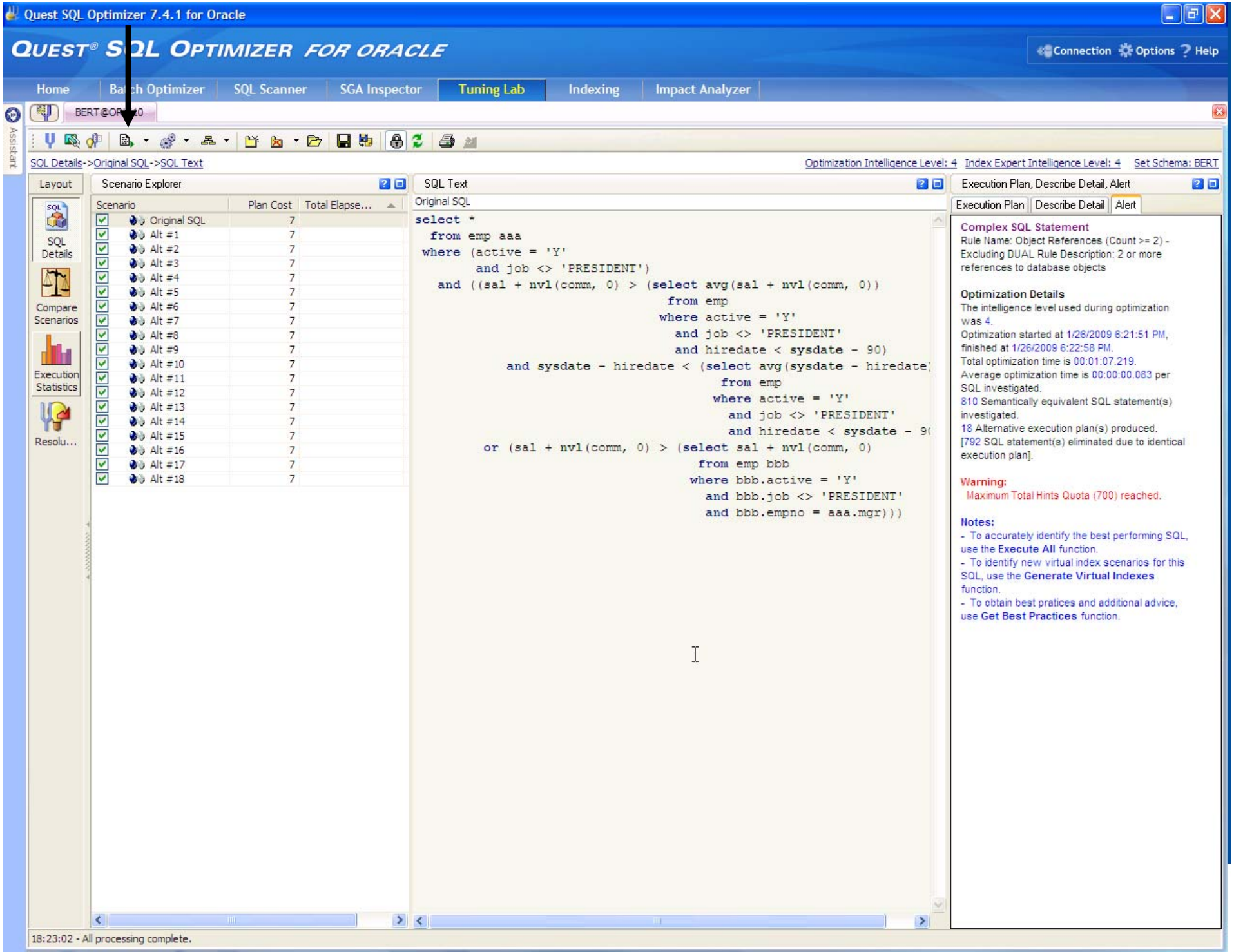
SQL Optimizer knows all this and much, much more:

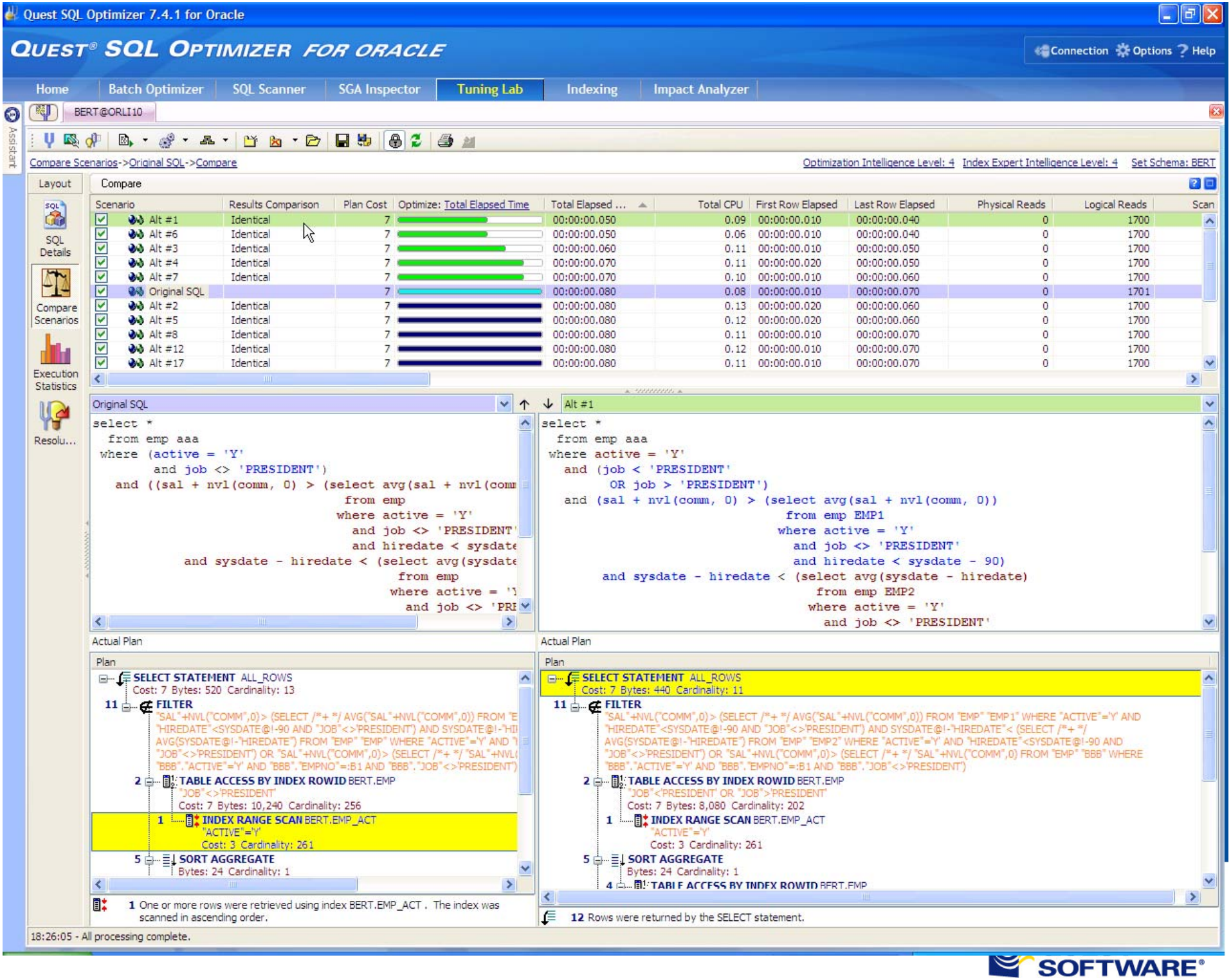
developers can press just two buttons to get their SQL statements automatically and 100% fully tuned!

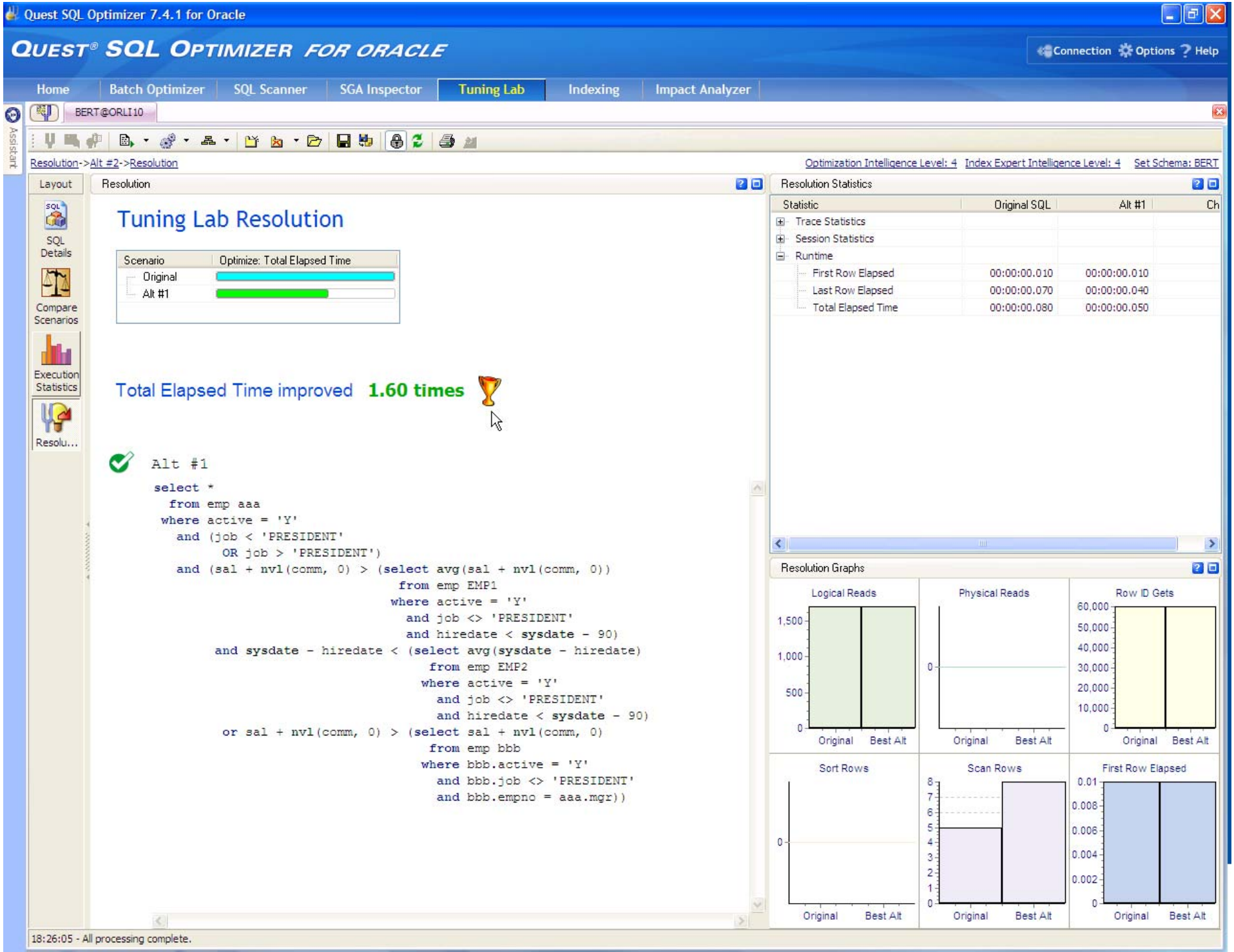
QUEST SOFTWARE®


Toad for Oracle - [BERT@ORLI10 - Editor (Q1.sql)]
File Edit Search Grid Editor Session Database Debug View Utilities Window Help
<default>
BERT@ORLI10
Desktop: SQL
Current Schema: BERT
SQL Optimizer
Oracle Tuning Advisor
Q1.sql
select *
from emp aaa
where
(active = 'Y'
and job <> 'PRESIDENT'
)
and
(
(
sal+nvl(comm,0) > (select avg(sal+nvl(comm,0))
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate-90)
and
sysdate-hiredate < (select avg(sysdate-hiredate)
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate-90)
)
or
(
sal+nvl(comm,0) > (select sal+nvl(comm,0)
from emp bbb
where bbb.active = 'Y'
and bbb.job <> 'PRESIDENT'
and bbb.empno = aaa.mgr)
)
);
Explain Plan
Data Grid
Auto Trace
DBMS Output (disabled)
Query Viewer
CodeXpert
Explain Plan
Script Output
Plan
SELECT STATEMENT ALL_ROWS
Cost: 7 Bytes: 520 Cardinality: 13
11 FILTER
2 TABLE ACCESS BY INDEX ROWID TABLE BERT.EMP
Cost: 7 Bytes: 10,240 Cardinality: 256
1 INDEX RANGE SCAN INDEX BERT.EMP_ACT
Cost: 3 Cardinality: 261
5 SORT AGGREGATE
Bytes: 24 Cardinality: 1
4 TABLE ACCESS BY INDEX ROWID TABLE BERT.EMP
Cost: 7 Bytes: 5,448 Cardinality: 227
3 INDEX RANGE SCAN INDEX BERT.EMP_ACT
Cost: 3 Cardinality: 261
12. Rows were returned by the SELECT statement.
1: 1
BERT@ORLI10
Read-only
Editor
AutoCommit is OFF
CAPS
NUM
INS
Optimize SQL


Quest SQL Optimizer 7.4.1 for Oracle
QUEST® SQL OPTIMIZER FOR ORACLE
Connection
Options
Help
Home
Batch Optimizer
SQL Scanner
SGA Inspector
Tuning Lab
Indexing
Impact Analyzer
BERT@ORLI10
Assistant
SQL Details->Original SQL->SQL Text
Optimization Intelligence Level: 4
Index Expert Intelligence Level: 4
Set Schema: BERT
Layout
SQL Details
Compare Scenarios
Execution Statistics
Resolu...
Scenario Explorer
Scenario
Plan Cost
Total Elapse...
Original SQL
7
SQL Text
Original SQL
select *
from emp aaa
where (active = 'Y'
and job <> 'PRESIDENT')
and ((sal + nvl(comm, 0) > (select avg(sal + nvl(comm, 0))
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 90)
and sysdate - hiredate < (select avg(sysdate - hiredate)
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 9
or (sal + nvl(comm, 0) > (select sal + nvl(comm, 0)
from emp bbb
where bbb.active = 'Y'
and bbb.job <> 'PRESIDENT'
and bbb.empno = aaa.mgr)))
Execution Plan, Describe Detail, Alert
Execution Plan
Describe Detail
Alert
Complex SQL Statement
Rule Name: Object References (Count >= 2) - Excluding DUAL Rule Description: 2 or more references to database objects
18:20:05 - All processing complete.


Quest SQL Optimizer 7.4.1 for Oracle
QUEST® SQL OPTIMIZER FOR ORACLE
Connection
Options
Help
Home
Batch Optimizer
SQL Scanner
SGA Inspector
Tuning Lab
Indexing
Impact Analyzer
BERT@OR
SQL Details->Original SQL->SQL Text
Optimization Intelligence Level: 4
Index Expert Intelligence Level: 4
Set Schema: BERT
Layout
SQL Details
Compare Scenarios
Execution Statistics
Resolu...
Scenario Explorer
Scenario
Plan Cost
Total Elapse...
Original SQL 7
Alt #1 7
Alt #2 7
Alt #3 7
Alt #4 7
Alt #5 7
Alt #6 7
Alt #7 7
Alt #8 7
Alt #9 7
Alt #10 7
Alt #11 7
Alt #12 7
Alt #13 7
Alt #14 7
Alt #15 7
Alt #16 7
Alt #17 7
Alt #18 7
SQL Text
Original SQL
select *
from emp aaa
where (active = 'Y'
and job <> 'PRESIDENT')
and ((sal + nvl(comm, 0) > (select avg(sal + nvl(comm, 0))
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 90)
and sysdate - hiredate < (select avg(sysdate - hiredate)
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 9
or (sal + nvl(comm, 0) > (select sal + nvl(comm, 0)
from emp bbb
where bbb.active = 'Y'
and bbb.job <> 'PRESIDENT'
and bbb.empno = aaa.mgr)))
Execution Plan, Describe Detail, Alert
Execution Plan
Describe Detail
Alert
Complex SQL Statement
Rule Name: Object References (Count >= 2) - Excluding DUAL Rule Description: 2 or more references to database objects
Optimization Details
The intelligence level used during optimization was 4.
Optimization started at 1/26/2009 6:21:51 PM, finished at 1/26/2009 6:22:58 PM.
Total optimization time is 00:01:07.219.
Average optimization time is 00:00:00.083 per SQL investigated.
810 Semantically equivalent SQL statement(s) investigated.
18 Alternative execution plan(s) produced.
[792 SQL statement(s) eliminated due to identical execution plan].
Warning:
Maximum Total Hints Quota (700) reached.
Notes:
- To accurately identify the best performing SQL, use the Execute All function.
- To identify new virtual index scenarios for this SQL, use the Generate Virtual Indexes function.
- To obtain best pratices and additional advice, use Get Best Practices function.
18:23:02 - All processing complete.


Quest SQL Optimizer 7.4.1 for Oracle
QUEST® SQL OPTIMIZER FOR ORACLE
Connection Options Help
Home Batch Optimizer SQL Scanner SGA Inspector Tuning Lab Indexing Impact Analyzer
BERT@ORLI10
Compare Scenarios->Original SQL->Compare
Optimization Intelligence Level: 4 Index Expert Intelligence Level: 4 Set Schema: BERT
Layout
Compare
SQL Details
Compare Scenarios
Execution Statistics
Resolu...
Scenario | Results Comparison | Plan Cost | Optimize: Total Elapsed Time | Total Elapsed ... | Total CPU | First Row Elapsed | Last Row Elapsed | Physical Reads | Logical Reads | Scan
Alt #1 | Identical | 7 | 00:00:00.050 | 0.09 | 00:00:00.010 | 00:00:00.040 | 0 | 1700
Alt #6 | Identical | 7 | 00:00:00.050 | 0.06 | 00:00:00.010 | 00:00:00.040 | 0 | 1700
Alt #3 | Identical | 7 | 00:00:00.060 | 0.11 | 00:00:00.010 | 00:00:00.050 | 0 | 1700
Alt #4 | Identical | 7 | 00:00:00.070 | 0.11 | 00:00:00.020 | 00:00:00.050 | 0 | 1700
Alt #7 | Identical | 7 | 00:00:00.070 | 0.10 | 00:00:00.010 | 00:00:00.060 | 0 | 1700
Original SQL | | 7 | 00:00:00.080 | 0.08 | 00:00:00.010 | 00:00:00.070 | 0 | 1701
Alt #2 | Identical | 7 | 00:00:00.080 | 0.13 | 00:00:00.020 | 00:00:00.060 | 0 | 1700
Alt #5 | Identical | 7 | 00:00:00.080 | 0.12 | 00:00:00.020 | 00:00:00.060 | 0 | 1700
Alt #8 | Identical | 7 | 00:00:00.080 | 0.11 | 00:00:00.010 | 00:00:00.070 | 0 | 1700
Alt #12 | Identical | 7 | 00:00:00.080 | 0.12 | 00:00:00.010 | 00:00:00.070 | 0 | 1700
Alt #17 | Identical | 7 | 00:00:00.080 | 0.11 | 00:00:00.010 | 00:00:00.070 | 0 | 1700
Original SQL
select *
from emp aaa
where (active = 'Y'
and job <> 'PRESIDENT')
and ((sal + nvl(comm, 0) > (select avg(sal + nvl(comm
from emp
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate
and sysdate - hiredate < (select avg(sysdate
from emp
where active = 'Y'
and job <> 'PRE
Alt #1
select *
from emp aaa
where active = 'Y'
and (job < 'PRESIDENT'
OR job > 'PRESIDENT')
and (sal + nvl(comm, 0) > (select avg(sal + nvl(comm, 0))
from emp EMP1
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 90)
and sysdate - hiredate < (select avg(sysdate - hiredate)
from emp EMP2
where active = 'Y'
and job <> 'PRESIDENT'
Actual Plan
Plan
SELECT STATEMENT ALL_ROWS
Cost: 7 Bytes: 520 Cardinality: 13
11 FILTER
2 TABLE ACCESS BY INDEX ROWID BERT.EMP
"JOB"<>'PRESIDENT'
Cost: 7 Bytes: 10,240 Cardinality: 256
1 INDEX RANGE SCAN BERT.EMP_ACT
"ACTIVE"='Y'
Cost: 3 Cardinality: 261
5 SORT AGGREGATE
Bytes: 24 Cardinality: 1
1 One or more rows were retrieved using index BERT.EMP_ACT . The index was scanned in ascending order.
Actual Plan
Plan
SELECT STATEMENT ALL_ROWS
Cost: 7 Bytes: 440 Cardinality: 11
11 FILTER
2 TABLE ACCESS BY INDEX ROWID BERT.EMP
"JOB"<'PRESIDENT' OR "JOB">'PRESIDENT'
Cost: 7 Bytes: 8,080 Cardinality: 202
1 INDEX RANGE SCAN BERT.EMP_ACT
"ACTIVE"='Y'
Cost: 3 Cardinality: 261
5 SORT AGGREGATE
Bytes: 24 Cardinality: 1
12 Rows were returned by the SELECT statement.
18:26:05 - All processing complete.


Quest SQL Optimizer 7.4.1 for Oracle
QUEST® SQL OPTIMIZER FOR ORACLE
Connection
Options
Help
Home
Batch Optimizer
SQL Scanner
SGA Inspector
Tuning Lab
Indexing
Impact Analyzer
BERT@ORLI10
Assistant
Resolution->Alt #2->Resolution
Optimization Intelligence Level: 4
Index Expert Intelligence Level: 4
Set Schema: BERT
Layout
Resolution
SQL Details
Compare Scenarios
Execution Statistics
Resolu...
Tuning Lab Resolution
Scenario
Optimize: Total Elapsed Time
Original
Alt #1
Total Elapsed Time improved 1.60 times
Alt #1
select *
from emp aaa
where active = 'Y'
and (job < 'PRESIDENT'
OR job > 'PRESIDENT')
and (sal + nvl(comm, 0) > (select avg(sal + nvl(comm, 0))
from emp EMP1
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 90)
and sysdate - hiredate < (select avg(sysdate - hiredate)
from emp EMP2
where active = 'Y'
and job <> 'PRESIDENT'
and hiredate < sysdate - 90)
or sal + nvl(comm, 0) > (select sal + nvl(comm, 0)
from emp bbb
where bbb.active = 'Y'
and bbb.job <> 'PRESIDENT'
and bbb.empno = aaa.mgr))
Resolution Statistics
Statistic
Original SQL
Alt #1
Ch
Trace Statistics
Session Statistics
Runtime
First Row Elapsed 00:00:00.010 00:00:00.010
Last Row Elapsed 00:00:00.070 00:00:00.040
Total Elapsed Time 00:00:00.080 00:00:00.050
Resolution Graphs
Logical Reads
Physical Reads
Row ID Gets
Sort Rows
Scan Rows
First Row Elapsed
Original
Best Alt
18:26:05 - All processing complete.