Business Intelligence Solution Providers

Specialized in creating talent resource pool

Oracle SQL Tuning Basics Tips and Tricks

Description:

BISP is committed to provide BEST learning material to the beginners and advance learners. In the same series, we have prepared a complete end-to end Hands-on Guide SQL optimization tips. The document focuses on top 21 tics for SQL optimization. See our youtube collections for more details. Join our professional training program and learn from experts.

Description Change	Author	Publish Date
Initial Draft	Kuldeep Mishra	12th Aug
		-
Review#1	Amit Sharma	18 th Aug 2011
	Description Change Initial Draft Review#1	Description Change Author Initial Draft Kuldeep Mishra Review#1 Amit Sharma

Contents

Contents
Tuning Tips#13
Tuning Tips#24
Tuning Tips#34
Tuning Tips#45
Tuning Tips#56
Tuning Tips#6
Tuning Tips#7
Tuning Tips#811
Tuning Tips#912
Tuning Tips#1013
Tuning Tips#1115
Tuning Tips#1216
Tuning Tips#1317
Tuning Tips#1417
Tuning Tips#15
Tuning Tips#16
Tuning Tips#1719
Tuning Tips#1819
Tuning Tips#1920
Tuning Tips#2021

Objective: SQL Tuning top 20 Rules for Oracle SQL beginners and intermediate learners.

Oracle SQL Tuning Tips

Consideration when writing an SQL statement is that it returns a correct result. The second is that it be the most efficient for a given situation. You can use many different SQL statements to achieve the same result. It is often the case that only one statement will be the most efficient choice in a given situation.

Remember that processing SQL is a sequence of Parse (syntax check and object resolution), Execution (required reads and writes), and Fetch (row results retrieved, listed, sorted, and returned). SQL "tuning" consists, quite simply, of reducing one or more of them.

Note: generally Parse is the greatest time and resource hog. Parse overhead can be minimized by the use of Procedures, Functions, Packages, Views, etc.

Inadequate performance can have a significant cost impact on your business. A poor performing system and application can result in customer dissatisfaction, reduced productivity, and high costs. It is absolutely critical that the system's performance is operating at its peak levels.

Following are some general tips that often increase SQL statement efficiency. Being general they may not apply to a particular scenario.

SQL Tuning Top 20 Tics Tuning Tips#1

1) The sql query becomes faster if you use the actual columns names in SELECT statement instead of than '*'.

SELECT Employee_code,First_Name,Last_Name,Hire_Date_Hired, from Employee;

	🗃 🥔 0.5522998 second	ds	
SELECT employee_code	,first_name,last_nam	e,date_hired from employee;	
<			·
	🗑 Explain 📓 Autotrace 🗔		
Image: A state of the state	<u> </u>		
EMPLOYEE_CODE	FIRST_NAME	LAST_NAME	DATE_HIRED
 10054	Alan	Wilcox	22-MAY-97
10055	Pierre	Lavoie	02-JUL-03
10056	Rhonda	Cummings	13-AUG-02
10057	Samantha	Floyd	02-MAY-00
10058	Ashley	McCormick	05-0CT-01
10059	Audrey	Lastman	19-JUN-00
10060	Bart	Scott	12-FEB-01
10061	Julie	Olsen	15-AUG-00
10062	Ken	Chambers	17-AUG-99
10063	Chad	Michaels	16-APR-03
10064	Maureen	Hoffman	26-JUN-01
10065	Corey	Wright	22-JAN-02
nstead of :			
FLECT * from Emplo	WAAS'		
	, y c c 3,		

2) HAVING clause is used to filter the rows after all the rows are selected. It is just like a filter. Do not use HAVING clause for any other purposes. For Example: Write the query as

SELECT employee_code,COUNT(employee_code),SUM(salary),AVG(salary) FROM employee_summary WHERE employee_code!=10014 AND employee_code!=10021 GROUP BY employee_code;

📵 gs_or_modify.sql 👘 🕪	admin >> gosales_connection	gosaleshr_connecti	on 🕞 gosalesmr_connection
۱ 🖉 ا 🗞 🗐 🛃 🖌	対 🥔 0.5390892 seconds		
SELECT employee_cod	de,COUNT(employee_code),S e!=10014 AND employee_cod	OM(salary),AVG(salary de!=10021 GROUP BY emp	<pre>r) FROM employee_summary loyee_code;</pre>
Results Script Output	BExplain Autotrace RDB		
			1
EMPLOYEE_CODE	COUNT (EMPLOYEE_CODE)	SUM (SALARY)	AVG(SALARY)
10015	4	 154000	38500
10006	15	418846.14	27923.076
10029	43	2034653.87	47317.53186
10045	43	2930576.93	68152.95186
10048	43	2959192.31	68818.42581
10064	43	2709500	63011.62790
10065	43	3113000.06	72395.35023
10078	43	2959192.31	68818.42581
10091	31	1131999.94	36516.12709
10051	43	1826730.67	42482.10860
10095	43	2292346.2	53310.37674
10098	43	2368807.76	55088.55255
10100	•	115500	00500

Instead of:

SELECT employee_code,COUNT(employee_code),SUM(salary),AVG(salary) FROM employee_summary GROUP BY employee_code Having employee_code!=10014 AND employee_code!=10021 ;

Tuning Tips#3

Sometimes you may have more than one subqueries in your main query. Try to minimize the number of subquery block in your query.

For Example: Write the query as

SELECT employee_code,First_name, last_name, date_hired from EMPLOYEE WHERE (employee_code)=(SELECT MAX(employee_code) from EMPLOYEE) AND termination_code=150;

<pre>gs_or_modify.sql</pre>	▶ admin ▶ gosales_connec	tion b gosaleshr_connection	gosalesmr_connection	EMPLOYEE_EXPENSE_DETAIL	
🕨 📃 🖗 🖪 🛞 I '	🗃 📓 🥔 0.96617919 seco	onds			gosaleshr_connection \star
SELECT employee		name, date_hired from EMPLO	YEE WHERE (employee_cod	de)=(SELECT MAX(employee_cod	le) from EMPLOYEE)
AND termination_	code=150;				
<					>
Results Script Ou	tput 🗃 Explain 📓 Autotrace 🕻	DBMS Output			
🥔 🗄 📇					
EMPLOYEE_CODE	FIRST_NAME	LAST_NAME	DATE_HIRED		
 10803	Xiedong	Wu	02-FEB-04		
l rows selected					

Instead of:

SELECT employee_code,first_name, last_name, date_hired from EMPLOYEE WHERE employee_code =(SELECT MAX(employee_code) from EMPLOYEE) AND termination_code=150;

Tuning Tips#4

Use operator EXISTS, IN and table joins appropriately in your query.

- a) Usually IN has the slowest performance.
- b) IN is efficient when most of the filter criteria is in the sub-query.
- c) EXISTS is efficient when most of the filter criteria is in the main query.

For Example: Write the query as

SELECT * from EMPLOYEE A WHERE EXISTS (SELECT * from Employee_Expense_Detail B WHERE b.employee_code= a.employee_code);

gs_or_modify.sql	⊳admin ⊳gosales_connec	tion > gosaleshr_connection	osalesmr_connection	EMPLOYEE	Ð
۱ 💿 🛃 🖏 📃 ۱	🗃 😹 🥔 0.59942144 seco	onds			gosaleshr_connection 🔶
SELECT * FROM E	MPLOYEE & WHERE EXISTS	(SELECT * from Employee_Expe	nse_Detail B WHERE b.employ	<pre>code= a.employee_code);</pre>	
<					>
Results Script Out	put 👸 Explain 🎇 Autotrace 🕻	🐌 DBMS Output 🌏 OWA Output			
🧼 🗖 📇		· · · · · · · · · · · · · · · · · · ·			
EMPLOYEE_CODE	FIRST_NAME	FIRST_NAME_MB	LAST_NAME	LAST_NAME_MB	DATE_HIRED
10004	Denis	Denis	Pagé	Pagé	11-DEC-01
10005	Élizabeth	Élizabeth	Michel	Michel	24-N0V-03
10006	Émile	Émile	Clermont	Clermont	10-MAY-06
10007	Étienne	Étienne	Jauvin	Jauvin	09-0CT-03
10012	Elsbeth	Elsbeth	Wiesinger	Wiesinger	22-MAR-05
10013	Else	Else	Mörike	Mörike	07-NOV-00
10014	Frank	Frank	Fuhlroth	Fuhlroth	12-MAY-03
10015	Gunter	Gunter	Erler	Erler	18-APR-07
10016	Björn	Björn	Winkler	Winkler	13-JUL-04
10017	Fritz	Fritz	Hirsch	Hirsch	20-DEC-04
10018	Jörg	Jörg	Kunze	Kunze	18-FEB-03
10019	Silvano	Silvano	Allessori	Allessori	27-JUN-06
Instead			of		:

SELECT * from EMPLOYEE A WHERE employee_code IN(SELECT employee_code from Employee_Expense_Detail);

Use EXISTS instead of DISTINCT when using joins which involves tables having one-tomany relationship.

For Example: Write the query as

SELECT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E WHERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code);

Ops_or_modify.sd Deades_connection Opsalestr_connection Opsalestr_connection EMPLOYEE SELECT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E Interference EMPLOYEE Employee_code=.employee_code]; Presult Sorth Output Employee Code=.e.employee_code=.e.employee_code]; Presult Sorth Output Employee Code=.e.employee_code]; Presult Sorth Output Employee OBMS Output OWA Output Presult Sorth Output Employee OBMS Output OWA Output Presult Sorth Output Employee OBMS Output OWA Output Presult Sorth Output Employee OWA Output OWA Output Presult Sorth Output Employee OWA Output OWA Output Presult Sorth Output Denis Pagé 11-PEC-01 10005 Élizabeth Michel 24-MOY-03 10017 10012 Else Morike 07-MOY-00 10014 Prank Publiroth 12-MAY-03 10013 Else Morike 13-UI-04 10017 Fritz Hirech				
Indestruction Indestruction Indestruction	🗐 gs or modify.sql D	admin ∣⊳ gosales connection	> gosaleshr connection	osalesmr connection
SELECT e.employee_code,e.first_name,e.last_name,e.date_hired from ENPLOYEE E NHERE EXISTS (SELECT expense_total from ENPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code); Results Script Output Terplain Autotrace DDBMS Output Results Script Output Terplain Terpla) 📓 🥜 1.05615687 seconds		
INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code); INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code); INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=code; INTERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code from product Union All Select product_color_code from product Color_lookup;	SELECT e.employee	code.e.first name.e.last na	ame.e.date bired from El	MPLOYEE E
Results Script Output Results Script Output Results OWA Output PRESURE CODE FIRST_NAME LAST_NAME Pagé 11-DEC-01 10004 Denis Pagé 11-DEC-01 10005 10005 Élizabeth Michel 24-NOV-03 10006 Émile Clermont 10-MAY-06 10007 Étienne Jauvin 09-0CT-03 10012 Elsbeth Wiesinger 22-MAR-05 10013 Else Norike 07-NOV-00 10014 Frank Publiroth 12-MAY-03 10015 Gunter Erler 18-APR-07 10016 Björn Winkler 13-JUL-04 10017 Pritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-PED-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code.e.first_name.e.last_name.e.date_hired from EMPLOYEE E.EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code;	WHERE EXISTS (SELE	CT expense_total from EMPL	OYEE_EXPENSE_DETAIL D W	<pre>ERE d.employee_code=e.employee_code);</pre>
Results Script Output Mutotrace DBMS Output OWA Output Image: Code FIRST_NAME LAST_NAME DATE_HIRED 10004 Denis Pagé 11-DEC-01 10005 Élizabeth Michel 24-MOV-03 10006 Émile Clermont 10-MAY-06 10007 Étienne Jauvin 09-0CT-03 10012 Elabeth Wiesinger 22-MAR-05 10013 Else Morike 07-NOV-00 10014 Frank Puhlroth 12-MAY-03 10015 Gunter Erler 18-APR-07 10016 Björn Winkler 13-UU-04 10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-FEB-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E.EMPLOYEE EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; Try to use UNION ALL in place of UNION. For Example				
Results Script Output Mutotrace DOMS Output Presults First_NAME LAST_NAME DATE_HIRED 10004 Denis Pagé 11-DEC-01 10005 Élizabeth Michel 24-807-03 10006 Émile Clermont 10-MAY-06 10012 Elsbeth Wiesinger 22-MAR-05 10013 Else Morike 07-807-03 10014 Frank Puhlroth 12-hAP-05 10015 Guuter Else Norike 10016 Björn Winkler 13-JUL-04 10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 16-FEE-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E.EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; . SELECT D WHERE SELECT product_color_code from product Union All Select product_color_code from product Loion _lo				
ENPLOYEE_CODE FIRST_NAME LAST_NAME PATE_HIRED 10004 Denis Pagé 11-DEC-01 10005 Élizabeth Michel 24-N0V-03 10006 Émile Clermont 10-MAY-06 10012 Elsbeth Wiceinger 22-MAR-05 10013 Else Nörike 07-N0V-00 10014 Frank Fuhlroth 12-MAY-03 10015 Gunter Eller 18-AFR-07 10016 Björn Winkler 13-JUL-04 10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-AFR-07 10018 Björn Winkler 13-JUL-04 10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-AFR-07 10018 Silvano Allessori 27-JUL-06 Instead of: SELECT DSTINCT eemployee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; D <td< td=""><td>🕞 Results 📃 Script Outpu</td><td>Jt 📷 Explain 📓 Autotrace 🗔 DBMS</td><td>i Output 💽 OWA Output</td><td></td></td<>	🕞 Results 📃 Script Outpu	Jt 📷 Explain 📓 Autotrace 🗔 DBMS	i Output 💽 OWA Output	
EMPLOYEE_CODE FIRST_NAME LAST_NAME DATE_HIRED 10004 Denis Pagé 11-DEC-01 10005 Élizabeth Michel 24-N07-03 10006 Émile Clermont 10-NAY-06 10007 Étienne Jauvin 09-OCT-03 10012 Elsbeth Wiesinger 22-MAR-05 10013 Else Mörike 07-N0V-00 10014 Frank Fuhlroth 12-MAY-03 10015 Gunter Erler 18-APR-07 10016 Björn Winkler 13-JUL-04 10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-FEB-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code; Eirst_name,e.last_name,e.date_hired from EMPLOYEE E_EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; D WHERE SELECT product_color_code from product Union All Select product_color_code from product Union All Select product_color	🥔 🗔 📇			
10004DenisPagé11-DEC-0110005ÉlizabethMichel24-N0V-0310006ÉmileClermont10-MAT-0610007ÉtienneJauvin09-0CT-0310012ElsbethWiesinger22-MAR-0510013ElseMörike07-N0V-0010014FrankFullroth12-MAY-0310015GunterErler18-AFR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hiredfromEMPLOYEEE.EMPLOYEE_EXPENSE_DETAILDWHEREd.employee_code=e.employee_code;DWHERE5522222d.employee_code=c.employee_code;Frankthe query asSELECT product_color_code from product Union All Select product_color_code from522222orduct_color_lookup;Silect product_color_code from	EMPLOYEE_CODE	FIRST_NAME	LAST_NAME	DATE_HIRED
10005ÉlizabethMichel24-N0V-0310006ÉmileClermont10-MAY-0610007ÉtienneJauvin09-0CT-0310012ElsbethWiesinger22-MAR-0510013ElseMörike07-N0V-0010014FrankFuhlroth12-MAY-0310015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hiredfromEMPLOYEEE,EMPLOYEE_EXPENSE_DETAILDWHEREd.employee_code=e.employee_code;Try to use UNION ALL in place of UNION.ForExample:WritetheForExample:WritethequeryasSELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	10004	Denis	Pagé	11-DEC-01
10006ÉmileClermontIn-MAY-0610007ÉtienneJauvin09-0CT-0310012ElsbethWiesinger22-MAR-0510013ElseMörike07-NOV-0010014FrankFuhlroth12-MAY-0310015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code.e.first_name,e.last_name,e.date_hiredfromEMPLOYEE EXPENSE_DETAILDWHEREd.employee_code.e.first_name,e.last_name,e.date_hiredfromESLECT DISTINCT e.employee_code.e.first_name,e.last_name,e.date_hiredfromEMPLOYEE EXPENSE_DETAILDWHEREd.employee_code.e.first_name,e.last_name,e.date_hiredfromFry to use UNION ALL in place of UNION.For Example: Write the query asSELECT product_color_code from product Union All Select product_color_code fromproduct_color_lookup;	10005	Élizabeth	Michel	24-NOV-03
10007ÉtienneJauvin09-0CT-0310012ElsbethWiesinger22-MAR-0510013ElseMörike07-N0V-0010014FrankFuhroth12-MAY-0310015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code.e.first_name,e.last_name,e.date_hiredfromEMPLOYEE_EXPENSE_DETAILDWHEREd.employee_code:Tuning Tips#6Try to use UNION ALL in place of UNION.For Example:Writethe query asSELECT product_color_code from product Union All Select product_color_code from product Loion_lookup;	10006	Émile	Clermont	10-MAY-06
10012ElsbethWiesinger22-MAR-0510013ElseMörike07-NOV-0010014FrankFuhroth12-MAY-0310015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hiredEMPLOYEEE,EMPLOYEE_EXPENSE_DETAILDWHEREd.employee_code=e.employee_code;WitethequeryasSELECT product_color_code from product Union All Select product_color_code fromForExample:WritethequeryasSELECT plookup;Select product_color_code fromSelect product_color_code fromSelect product_color_code from	10007	Étienne	Jauvin	09-0CT-03
10013ElseMörike07-N0V-0010014FrankFuhlroth12-MAY-0310015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hiredfromEMPLOYEE EXPENSE_DETAILDWHEREd.employee_code;Tuning Tips#6Try to use UNION ALL in place of UNION.For Example: Write the query asSELECT product_color_code from product Union All Select product_color_code from product Union All Select product_color_code from product_color_lookup;	10012	Elsbeth	Wiesinger	22-MAR-05
10014FrankFuhlroth12-MAY-0310015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hiredMPLOYEEE,EMPLOYEE_EXPENSE_DETAILDWHEREd.employee_code=e.employee_code;Tuning Tips#6Try to use UNION ALL in place of UNION.For Example: Write the query asSELECT product_color_code from product Union All Select product_color_code from product Union All Select product_color_code from product_color_lookup;	10013	Else	Mörike	07-NOV-00
10015GunterErler18-APR-0710016BjörnWinkler13-JUL-0410017FritzHirsch20-DEC-0410018JörgKunze18-FEB-0310019SilvanoAllessori27-JUN-06Instead of:SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hiredd.employee_code=e.employee_code;DWHERETuning Tips#6Try to use UNION ALL in place of UNION.For Example: Write the query asSELECT product_color_code from product Union All Select product_color_code fromproduct_color_lookup;	10014	Frank	Fuhlroth	12-MAY-03
10016 Björn Winkler 13-JUL-04 10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-FEB-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup; Select product_color_lookup; Select product_color_lookup;	10015	Gunter	Erler	18-APR-07
10017 Fritz Hirsch 20-DEC-04 10018 Jörg Kunze 18-FEB-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup; Select product_color_code from product_color_code from	10016	Björn	Winkler	13-JUL-04
10018 Jörg Kunze 18-FEB-03 10019 Silvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; WHERE D WHERE Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product_color_code from product_color_lookup; union All Select product_color_code from	10017	Fritz	Hirsch	20-DEC-04
10019 \$ilvano Allessori 27-JUN-06 Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; WHERE D WHERE Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup; product_color_lookup;	10018	Jörg	Kunze	18-FEB-03
Instead of: SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; Tuning Tips#6 Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	10019	Silvano	Allessori	27-JUN-06
SELECT DISTINCT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; Tuning Tips#6 Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	Instead of:			
EMPLOYEE E,EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code; Tuning Tips#6 Try to use UNION ALL in place of UNION.	SELECT DISTING	CT e.employee_code,e	<mark>e.fir</mark> st_name,e.last_na	ame,e.date_hired from
d.employee_code=e.employee_code; Tuning Tips#6 Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	EMPLOYEE	E,EMPLOYEE_EXPEI	<mark>VSE</mark> _DETAIL	D WHERE
Tuning Tips#6 Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	d.employee_code=	=e.employee_code;		
Tuning Tips#6 Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;				
Try to use UNION ALL in place of UNION. For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	Tunina Tips#	6		
For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;				
For Example: Write the query as SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;				
SELECT product_color_code from product Union All Select product_color_code from product_color_lookup;	For Exam	ipie: write	the qu	ery as
product_color_lookup;	SELECT product_c	color_code from produc	t Union All Select p	product_color_code from
	product_color_look	<up;< td=""><td></td><td></td></up;<>		
www.bispsolutions.com www.hyperionguru.com www.bisptrainings.com Page 6	www.bispsolutions Page 6	s.com www.hyperior	nguru.com www.k	oisptrainings.com

> gosales_connection	
gs_or_modify.sgl > admin > gosaleshr_connection > ge	sal
Image: A state of the state	
	_
SELECT product_color_code from product	
Union all Select product color code FDOM product color lockup	
Select product_color_code rRow product_color_lookup	<i>.</i>
> Results 📃 Script Output 📓 Explain 📓 Autotrace 🗔 DBMS Output	Q
	~
PRODUCT_COLOR_CODE	
902	
903	
918	
919	
922	
923	
925 002	
903 003	
906	

Instead of:

SELECT product_color_code from product Union Select product_color_code from product_color_lookup;

For Example: Write the query as

SELECT product_brand_code ,product_brand_en AS product_brand_color_en FROM PRODUCT_BRAND UNION All SELECT product_color_code ,product_color_en AS product_brand_color_en from PRODUCT_COLOR_LOOKUP;

gosales_connection	
	San I 🖉 I
📵 gs_or_modify.sql 🛛 🗼	admin > gosaleshr_connection >> gosales_connection~1
> 📃 🔯 🔩 🕘 関	🛐 🥔 0.52049714 seconds
SELECT PRODUCT_BRAN FROM PRODUCT_BRAND UNION All SELECT PRODUCT_COLOR FROM PRODUCT_COLOR	ND_CODE ,PRODUCT_BRAND_EN AS product_brand_color_en DR_CODE ,PRODUCT_COLOR_EN AS product_brand_color_en LOOKUP;
<	
Peculta Script Output	MExplain Mutotrace RDBMS Output
Image: A start and a start	
PRODUCT_BRAND_CODE	PRODUCT_BRAND_COLOR_EN
719	Hailstorm
720	Relief
752	Alpha
753	Antoni
754	Epoch
755	Navigator
756	Relax
757	Trakker
758	

Instead of:

SELECT product_brand_code ,product_brand_en AS product_brand_color_en FROM PRODUCT_BRAND UNION SELECT product_color_code ,product_color_en AS product_brand_color_en from PRODUCT_COLOR_LOOKUP;

Tuning Tips#7

Beware of WHERE clauses which do not use indexes at all. Even if there is an index over a column that is referenced by a WHERE clause included in this section, Oracle will ignore the index. All these WHERE clause can be re-written to use an index while returning the same values. In other words, Do not perform operations on database objects referenced in the WHERE clause:

SELECT order_number,product_number,unit_sale_price from order_details WHERE unit_sale_price>1000;

	mber, Product_	Number, unit_s	sale_price from or	der_details	MULKE UII	rd_sale_prices.
tesults 📃 Script C	utput 📷 Explain	🔊 Autotrace 🗔	DBMS Output 💽 OWA 🤇	Dutput		
lts:						
ORDER_N	JMBER 🔋 PRODU	JCT_NUMBER	UNIT_SALE_PRICE			
1	100035	107110	1264.54			
2	100035	105110	1080.16			
4	100043	105110	1182.1			
5	100047	105110	1182.1			
6	100005	107110	1264.54			
7	100018	105110	1182.1			
8	100054	105110	1182.1			
9	100056	106110	1080.16			
10	100056	107110	1264.54			
'Al%';	D gosalesbr	annection	aosales connection			-
'Al%'; s_or_modify.sql S & & LECT First_Na		connection	<pre>gosales_connection ds ail from employee</pre>	MHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql S_ & & & I LECT First_Na	∫ > gosaleshr_c	connection	≥ gosales_connection ds ail from employee	MERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql S & & & I		connection	≥ gosales_connection ds mail from employee	MHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql CECT First_Na esults Script O	J > gosaleshr_d 3 3 1 2 1 me , Last_Name , utput 3 Explain	connection 0.05036031 second Date_Hired,Em	≥ gosales_connection ds mail from employee DBMS Output	WHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql ECT First_Na esults Script O ts:	J gosaleshr_c gosaleshr_c gosaleshr_c me, Last_Name, utput gerain	Connection 0.05036031 second Date_Hired,Em	≥ gosales_connection ds mail from employee DBMS Output , OWA C	MERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql E & & ILECT First_Na esults Script O ts: FIRST_NA	J gosaleshr_d gosaleshr_d me, Last_Name, utput g Explain	E DATE_HIRE	> gosales_connection ds ail from employee DBMS Output ○ ② EMAIL	WHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql LECT First_Na esults Script O ts: FIRST_NAL 1 Alan	Desideshr_c gosaleshr_c me,Last_Name, utput BExplain ME LAST_NAMI Wilcox	E DATE_HIREI	gosales_connection ds wail from employee DBMS Output O EMAIL AWilcox@grtd123.com	WHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql ECT First_Na LECT First_Na LECT First_Na Script O ts: FIRST_NA 1 Alan 2 Alice	J gosaleshr_(gosaleshr_(me, Last_Name, utput ge xplain (E ge LAST_NAM Wilcox Walter	E DATE_HIRED	gosales_connection ds dail from employee DBMS Output O EMAIL AWilcox@grtd123.com	WHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql s_or_modify.sql ECT_First_Na ESUITS Script O ts: FIRST_NAL 1 Alan 2 Alice 3 Alessandra	J gosaleshr_(gosaleshr_(me,Last_Name, utput BExplain ME LAST_NAMI Wilcox Walter Torta	Connection 0.05036031 second Date_Hired,Em Autotrace E DATE_HIRED 22-MAY-97 25-FEB-05 15-DEC-03	gosales_connection ds ail from employee DBMS Output O EMAIL AWilcox@grtd123.com AWalter@grtd123.com ATorta@grtd123.com	WHERE first	t_name LIK	E 'A1%';
'Al%'; c_or_modify.sql Cor_	VE LAST_NAMI Wilcox Walter Torta Rodriguez	connection 0.05036031 second Date_Hired,Em Mutotrace Image: Constraint of the second secon	gosales_connection ds tail from employee DBMS Output O EMAIL AWilcox@grtd123.com ATorta@grtd123.com ARodriguez@grtd123.com	MHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql s_or_modify.sq	Image: gosaleshr_q I	Connection 0 0.05036031 second Date_Hired,Em Date_Hired,Em Email: Em	gosales_connection ds ail from employee DBMS Output O EMAIL AWilcox@grtd123.com AWalter@grtd123.com ATorta@grtd123.com ARodriguez@grtd123.com ALehrer@grtd123.com	WHERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql s_@ @ @ 0 LECT First_Na esults Script O ts: FIRST_NAL 1 Alan 2 Alice 3 Alessandra 4 Alex 5 Albrecht 6 Almut	Image: gosaleshr_c I	E DATE_HIRE 22-MAY-97 25-FEB-05 15-DEC-03 08-SEP-00 21-JUN-99 17-JUL-00	gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWalter@grtd123.com AWalter@grtd123.com ARodriguez@grtd123.com ALehrer@grtd123.com ADiederichsen@grtd123.com	Dutput	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql s_or_modify.sq	Image: gosaleshr_c I	Connection 0.05036031 second Date_Hired,Em Date_Constant Datem Datem<	gosales_connection ds dail from employee DBMS Output DBMS Output Q 2 EMAIL AWilcox@grtd123.com AWalter@grtd123.com ATorta@grtd123.com ARodriguez@grtd123.com ALehrer@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com	MHERE first	t_name LIK	E 'A1%';
'Al%'; <u>s_or_modify.sql</u> s_or_modify.sql LECT First_Na LECT First_Na LECT First_Na Alan 2 Alice 3 Alessandra 4 Alex 5 Albrecht 6 Almut 7 Alejandrino 8 Alexander	Image: gosaleshr_q I	Connection Image: Connection 0.05036031 second Date_Hired,Em Image: Connection	gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWilcox@grtd123.com AWalter@grtd123.com ARodriguez@grtd123.com ALehrer@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com	MERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql s_or_modify.sql LECT First_Na esults Script O ts: FIRST_NAL 1 Alan 2 Alice 3 Alessandra 4 Alex 5 Albrecht 6 Almut 7 Alejandrino 8 Alexander 9 Alice	gosaleshr_(gosaleshr	Connection Image: Connection Date_Hired,Em Date_Hired,Em Image: Connection Image: Connection <td>gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWilcox@grtd123.com AWalter@grtd123.com ARodriguez@grtd123.com ARodriguez@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ASchmuker@grtd123.com</td> <td>Dutput</td> <td>t_name LIK</td> <td>E 'A1%';</td>	gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWilcox@grtd123.com AWalter@grtd123.com ARodriguez@grtd123.com ARodriguez@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ASchmuker@grtd123.com	Dutput	t_name LIK	E 'A1%';
 'Al%'; s_or_modify.sql s_or_modify.sql s_or_modify.sql s_or_modify.sql sql Script O ts: FIRST_NAL Alan Alan Alan Alex Alexandra Alex Albrecht Almut Alexander Alice Alexander Alice Alexander Alice Alexander Alice 	Image: gosaleshr_c I	Autotrace 22-MAY-97 25-FEB-05 15-DEC-03 08-SEP-00 21-JUN-99 17-JUL-00 26-NOV-01 16-MAY-01 14-DEC-05 18-JAN-99	gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWilcox@grtd123.com AWalter@grtd123.com ATorta@grtd123.com ARodriguez@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ASchmuker@grtd123.com ASchmuker@grtd123.com AMartin@grtd123.com AMartin@grtd123.com	MERE first	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql s_or_modify.sql IECT First_Na esults Script O ts: FIRST_NAL 1 Alan 2 Alice 3 Alessandra 4 Alex 5 Albrecht 6 Almut 7 Alejandrino 8 Alexander 9 Alice 10 Allen	Image: gosaleshr_e I	Connection Image: Connection Date_Hired,Em Date_Hired,Em Image: Connection Image: Connection Date_Hired,Em Image: Connection Image: Connet Connection Image: Connection<	gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWalter@grtd123.com AWalter@grtd123.com ARodriguez@grtd123.com ALehrer@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ASchmuker@grtd123.com AMartin@grtd123.com AMiller@grtd123.com	Dutput	t_name LIK	E 'A1%';
'Al%'; s_or_modify.sql ■ ② ③ ③ ③ 1 LECT First_Na ■ Example 1 Alan 2 Alice 3 Alessandra 4 Alex 5 Albrecht 6 Almut 7 Alejandrino 8 Alexander 9 Alice 10 Allen	Image: gosaleshr_q I	Date_Hired,Ex Autotrace 22-MAY-97 25-FEB-05 15-DEC-03 08-SEP-00 21-JUN-99 17-JUL-00 26-NOV-01 16-MAY-01 14-DEC-05 18-JAN-99	gosales_connection ds ail from employee DBMS Output DBMS Output Q 2 EMAIL AWilcox@grtd123.com AWalter@grtd123.com ATorta@grtd123.com ARodriguez@grtd123.com ADiederichsen@grtd123.com ADiederichsen@grtd123.com ASchmuker@grtd123.com ASchmuker@grtd123.com AMartin@grtd123.com	MERE first		E 'A1%';

Rather SELECT First_ SUBSTR(first_nar USE: SELECT sales ye	_Name,Last_Name,Dat ne,1,2)='Al'; ar,branch code,unit c	te_Hired,Email from employee	Than, • WHERE it cost >5;
gs_or_modify.sql	▶ gosales_connection	gosaleshr_connection > gosalesmr_connection	
🕨 📃 🐼 🖪 🚳 I	দ্বি 🎽 । 🏈 । 0.55229419 se	conds	
SELECT sales ve	ar branch code unit cos	t from product forecast WHEDE unit cos	->.5.
biliter bures_re	ar,pranon_00ac,anro_00p	o from produco_rorcodso militar anto-cos	
Results Script O	utput 🕲 Explain 🔊 Autotrace	😺 DBMS Output 💽 OWA Output	
SALES_YEAR	BRANCH_CODE	UNIT_COST	
2004	15	5.51	
2004	15	9.41	
2004	15	232.97	
2004	15	447.4 385.43	
2004	15	368.28	
2004	15	551.99	
2004	15	62.26	
2004	15	134.35	
2004	15	64.34	
2004	15	21.58	
2004	15	10.15	
Instead		of	:
SELECT sales_y	ear,branch_code,unit_	cost from product_for <mark>ecast</mark> WHEF	E unit_cost
NOT = 5;			
USE:			
SELECT sales ve	ear.ret <mark>ailer nam</mark> e.prod	duct brand code sales target from	sales target
WHERE sales tar	aet<2000:		ballob_tallget
WHERE SUCS_CU	get <2000,		

SELECT sales_year, retailer_name, product_brand_code, sales_target from sales_target basks sales_target Results Solpt Output Explain ALES_TEAR RETAILER_NAME SALES_TEAR The Caddy's Cornet	Image: Section 1.1	` [™] [™] [™] [™] [™] [™] [™] [™] [™] [™]		
Result Sorbet Output Explain Autotrace DBMS Output OWA Output SALES_TEAR RETAILER_NAME PRODUCT_EPAND_CODE SALES_ D004 The Caddy's Connet 757 1800 D004 The Caddy's Connet 753 1800 D004 The Caddy's Connet 753 1800 D004 The Caddy's Connet 753 1800 D004 The Sports Factory 720 400 D004 Husky Outfitters 752 1700 D004 Bullinger's Power Golf 754 1900 D004 Sallinger's Power Golf 756 1000 D004 Sallinger's Power Golf 756 1000 D004 Sallinger's Power Golf 756 1000 D004 Sallinger's Power Golf 757 1800 D	SELECT sales ye	ar,retailer name,product brand code,s	sales target from sales target WHERE sal	es target<2000
Result Sorph Output Result Result<				
Results Sorpe Output PRODUCT_BRAND_CODE SALES SALES_TEAR RETAILER_NAME PRODUCT_BRAND_CODE SALES SALES_TEAR RETAILER_NAME PRODUCT_BRAND_CODE SALES 0004 The Caddy's Cornet 756 1700 0004 The Caddy's Cornet 757 1800 0004 The Caddy's Cornet 733 1900 0004 The Caddy's Cornet 733 1900 0004 Busky Outfitters 720 400 0004 Busky Outfitters 754 1900 0004 Sallinget's Power Golf 757 1800 0004 Sallinget's Power Golf 757 1900 0000 Sallinget's Power Golf	<			
ALES_TEAR PETAILER_NAME PRODUCT_BRAND_CODE SALES_ BALES_TEAR The Caddy's Corner 756 1700 2004 The Caddy's Corner 757 1800 2004 The Caddy's Corner 757 1800 2004 The Gaddy's Corner 753 1800 2004 The Caddy's Corner 753 1800 2004 The Caddy's Corner 753 1900 2004 Busky Outfittes 720 400 2004 Busky Outfittes 752 1700 2004 Sallinget's Power Colf 754 1900 2004 Sallinget's Power Colf 757 1000 2005 Course_code,course_days,course_cost	Nesults Script C	Dutput CExplain CEADBMS Output		
ALES_TEAR RETAILER_NAME PRODUCT_BRAND_CODE SALES 2004 The Caddy's Corner 756 1700 2004 The Caddy's Corner 753 1800 2004 The Caddy's Corner 753 1800 2004 The Caddy's Corner 753 1800 2004 The Sports Pactory 720 400 2004 Rusty Outfitters 720 400 2004 Builinger's Power Colf 754 1900 2004 Sallinger's Power Colf 757 1800 2005 Course_retailer_name_product_brand_code	🥢 🗔 📇		×	
2004 The Caddy's Corner 756 1700 2004 The Caddy's Corner 757 1800 2004 The Sports Factory 720 400 2004 The Sports Factory 720 400 2004 The Sports Factory 720 400 2004 Hasky Outfitters 720 400 2004 Busky Outfitters 754 1900 2004 Sallinger's Power Colf 755 1400 2004 Sallinger's Power Colf 756 1400 2004 Sallinger's Power Colf 756 1600 2004 Sallinger's Power Colf 756 1000 2004 Sallinger's Power Colf 757 1800 2004 Sallinger's Power Colf 757 1000 Stander Of : : : ELECT course_code.course_days.course_cost from training WHERE inger.on Ourse_name_en=="Time Management'AND course_name_id="Manajemen Waktu"; : : : Ourse_code.course_code.course_cost from training WHERE course_name_id="Manajemen Waktu"; : :	SALES_YEAR	RETAILER_NAME	PRODUCT_BRAND_CODE	SALES_TAR
2004 The Caddy's Corner 757 1000 2004 The Caddy's Corner 757 1000 2004 The Caddy's Corner 753 1000 2004 The Caddy's Corner 753 1000 2004 The Caddy's Corner 753 1000 2004 Husky Outfitters 720 400 2004 Bulky Outfitters 754 1000 2004 Bulky Outfitters 755 1000 2004 Bulky Outfitters 755 1000 2004 Sallinger's Power Golf 755 1000 2004 Sallinger's Power Golf 757 1000 2005 ELECT course_code,course_days, course_cost				
The Caday's Corner 753 1000 2004 The Society's Corner 753 1000 2004 The Society's Corner 720 400 2004 The Society's Corner 720 400 2004 Extreme Outdoors 720 400 2004 Extreme Outdoors 720 400 2004 Husky Outfitters 752 1700 2004 Sallinger's Power Golf 754 1900 2004 Sallinger's Power Golf 758 1000 2004 Sallinger's Power Golf 758 1000 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1900 2005 Course_code.course_dosc.course_cost from training WHERE 2006 Course_code.course_dosc.course_cost from training W	2004 2004	The Caddy's Corner The Caddy's Corner	756	1700
Intersport Course Factory 720 4000 2004 Husky Outfitters 720 600 2004 Husky Outfitters 720 600 2004 Husky Outfitters 752 1700 2004 Husky Outfitters 752 1700 2004 Sallinger's Power Golf 754 1900 2004 Sallinger's Power Golf 756 1400 2004 Sallinger's Power Golf 756 1400 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1900 2005 Selector Selector Selector Selector 2005 Course_code,course_days, course_cost from training WHERE course_name_en [I Selector Stead	2004	The Caddy's Corner The Caddy's Corner	753	1800
2004 Husky Outfitters 720 500 2004 Extreme Outdoors 720 400 2004 Extreme Outdoors 720 400 2004 Husky Outfitters 752 1700 2004 Husky Outfitters 754 1900 2004 Sallinger's Power Golf 756 1400 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1900 2005 ELECT course_code.course_days.course_cost from training WHERE 2005 Course_code.course_days.course_cost from training WHERE 2006 Course_code.cours	2004	The Sports Factory	720	400
2004 Extreme Outdoors 720 400 2004 Husky Outfitters 752 1700 2004 Salinger's Power Golf 754 1900 2004 Salinger's Power Golf 754 1900 2004 Salinger's Power Golf 755 1800 2004 Salinger's Power Golf 755 1800 2004 Salinger's Power Golf 755 1800 2004 Salinger's Power Golf 755 1900 2004 Salinger's Power Golf 755 1900 2004 Salinger's Power Golf 755 1900 2004 Salinger's Power Golf 755 1000 2004 Salinger's Power Golf 755 1000 2005 1000 1000 1000 1000 1000 1000 1000	2004	Husky Outfitters	720	500
2004 Husky Outfitters 752 1700 2004 Husky Outfitters 754 1900 2004 Salinget's Power Golf 754 1900 2004 Salinget's Power Golf 755 1400 2004 Salinget's Power Golf 755 1400 2004 Salinget's Power Golf 756 1400 2004 Salinget's Power Golf 757 1900 2004 Taland Shorrs 756 1000 Stead of : : ELECT course_code, course_days, course_cost from training WHERE Dourse_name_en='Time Management' AND course_name_id='Manajemen Waktu'; gosdetr_course_code, course_code, course_cost from training WHERE Outse_code_course_code, course_cost from training WHERE course_name_id='Wanajemen Waktu'; gosdetr_course_code, course_code, course_cost from training WHERE course_name_en Outse_code course_code, course_cost from training WHERE course_name_en pourse_cost	2004	Extreme Outdoors	720	400
2004 Husky Dutfitters 754 1900 2004 Sallinger's Power Golf 754 1900 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1900 2005 Setted of : : ELECT sales year, retailer_name, product_brand_code, sales_target from sales_target ////////////////////////////////////	2004	Husky Outfitters	752	1700
2004 Sallinger's Power Golt 754 1900 2004 Sallinger's Power Golt 756 1400 2004 Sallinger's Power Golt 757 1800 2004 Sallinger's Power Golt 757 1900 2004 Sallinger's Power Golt 756 1000 2005 Setted 0f : : ELECT course_code,course_days,course_cost from training WHERE : 2004	2004	Husky Outfitters	754	1900
2004 Sallinger's Power Golf 756 1400 2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 757 1900 Stead of : . . ELECT sales_year, retailer_name, product_brand_code, sales_target from sales_target . . . ELECT course_code, course_days, course_cost from training WHERE pourse_name_en="Time Management" AND course_name_id="Manajemen Waktu'; . . . gs.gr.mdHy.st @saletr_cometon @saletr_cometon @saletr_cometon . . wisses_course_code, course_cost from training MARM course_name_en="Time Management" AND course_name_id="Time Management" M	2004	Sallinger's Power Golf	754	1900
2004 Sallinger's Power Golf 757 1800 2004 Sallinger's Power Golf 758 1000 2004 Sallinger's Power Golf 757 1900 2005 Setad 1 1000 1000 Stead Course_code, course_code, course_cost from training WHERE 1000 2007.001/sdl Power Golf Course_code, course_cost from training TRANNAS 000000000000000000000000000000000000	2004	Sallinger's Power Golf	756	1400
2004 Sallinger's Power Golf 758 1000 Sallinger's Power Golf 757 1900 Interest Source 0f : ELECT sales_year, retailer_name, product_brand_code, sales_target from sales_target . HERE sales_target + 3000<5000;	2004	Sallinger's Power Golf	757	1800
2004 Salinger's Power Golf 757 1900 instead of : ELECT sales_year, retailer_name, product_brand_code, sales_target from sales_target : HERE sales_target + 3000<5000;	2004	Sallinger's Power Golf	758	1000
20104 Teland Shorts 756 1000 Instead of : : ELECT sales_year,retailer_name,product_brand_code,sales_target from sales_target : WHERE sales_target + 3000<5000;	2004	Sallinger's Power Golf	757	1900
stead of : ELECT sales_year,retailer_name,product_brand_code,sales_target from sales_target Hard sales_target HERE sales_target + 3000<5000;	2004	Island Snorts	756	1000
Image: Control of the second secon]gs_or_modify.sql	Hes_connection gosaleshr_connection gosalesmr_connection		
SELECT course_code,course_days,course_cost from training WERE course_name_en='Time Management' NO course_name_id='Manajemen Waktu'; Results Stript Output Explain Autotrace ODEMS Output OWA Output OURSE_CODE COURSE_DAYS OURSE_CODE COURSE_DAYS OURSE_CODE COURSE_DAYS OURSE_CODE COURSE_DAYS OURSE_CODE COURSE_DAYS OURSE_CODE COURSE_COST 4501 1 250 250 rows selected of Instead of ELECT course_code,course_days,course_cost from training WHERE course_name_en ourse_name_id= 'Time Management Manajemen Waktu'; Funing Tips#8 ron't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL Statement are equally QL statement are	> 📃 🐼 🔩 🛞 🛙 🕅	↓ ↓ 0.51737773 seconds		gosaleshr_connecti
Results Script Output Results Script Output Results Re	SELECT course_code,cou	urse_days,course_cost from training WHERE course_name	e_en='Time Management' AND course_name_id='Manajemen Wa	ktu';
Results Script Output Mutotrace DBMS Output OUTRSE_COST Image: COURSE_CODE COURSE_DAYS COURSE_COST Image: Course code Image: Course code Image: Course code Image: Image: Course code Image: Course code Image: Course code Image: Image: Image: Course code Image: Course code Image: Course course code Image: Image: Image: Image: Course code Image: Course course code Image: Course course course course code Image: Image: Image: Image: Image: Course	<			
Results Softworder Course_Davis Course_cost 1 250 4501 1 250 rows selected				
COURSE_CODE COURSE_DAYS COURSE_COST 4501 1 250 1 250 1 250 1 250 1 250 1 250 1 250 1 250 1 250 1 250 1 250 1 250 1 0f 1 0f 1 1 1 0f	Results Script Output	Explain Mutotrace UDBMS Output MOWA Output		
NOURSE_CODE COURSE_DAYS COURSE_COST				
4501 1 250 A rows selected of : Instead of : ELECT course_code,course_days,course_cost from training WHERE course_name_en : ourse_name_id= 'Time Management Manajemen Waktu'; : Funing Tips#8 : on't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL statement are equally applicable to views.	COURSE_CODE	COURSE_DAYS COURSE_COST		
Instead of : ELECT course_code,course_days,course_cost from training WHERE course_name_en ourse_name_id= 'Time Management Manajemen Waktu'; Tuning Tips#8 on't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL QL statement are equally applicable to views.	.4501	1 250		
instead of : iELECT course_code,course_days,course_cost from training WHERE course_name_en ourse_name_id= 'Time Management Manajemen Waktu'; intermediate 'Time Management Manajemen Waktu'; intermediate 'Time Management Manajemen Waktu'; intermediate 'ELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL intermediate are equally applicable to views.	rows selected			
ELECT course_code, course_days, course_cost from training WHERE course_name_en ourse_name_id= 'Time Management Manajemen Waktu'; Funing Tips#8 oon't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL QL statement are equally applicable to views.	nstead	ot	f	:
ourse_name_id= 'Time Management Manajemen Waktu'; Tuning Tips#8 bon't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL statement are equally applicable to views.	ELECT course	code,course days,course cost fi	rom training WHERE course name	en
Funing Tips#8 oon't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL statement are equally applicable to views.	ourse name i	d= 'Time Management Manajeme	en Waktu';	
on't forget to tune views. Views are SELECT statements and can be tuned in just the ame way as any other type of SLECT statement can be. All tuning applicable to any QL statement are equally applicable to views.	unina Tip	s#8		
ame way as any other type of SLECT statement can be. All tuning applicable to any QL statement are equally applicable to views.	an't forget to	tupo viewa Viewa era CELECT -	tatomonto and can be tuned in it.	ct tha
ame way as any other type of SLECT statement can be. All tuning applicable to any QL statement are equally applicable to views.	for a lorger to	Lune views. views are SELECT S	catements and can be tuned in ju	scule
QL statement are equally applicable to views.	ame way as a	iny other type of SLECT stateme	ent can be. All tuning applicable t	o any
· · · · · · · · · · · · · · · · · · ·	QL stat	ement are equally	applicable to v	views.
void including a HAV/ING clause in SELECT statements. The HAV/ING clause filters	void including	1 a HAVING days in SELECT	statements. The HAV/ING clause	filtors
volumentum a navino clause in select statements, the navino clause lillers		A A HAVING CIAUSE III SELECT S		

www.bispsolutions.com | www.hyperionguru.com | www.bisptrainings.com | Page 11 reduce overheads in sorting, summing, etc. HAVING clauses should only be used when columns with summary operations applied to them are restricted by the clause. USE:

SELECT retailer_name from order_header WHERE retailer_name!='Grapevine Golf AND retailer_name!='Backcountry Discovery' GROUP BY retailer_name;



Instead of :

SELECT retailer_name from order_header GROUP BY retailer_name HAVING retailer_name!='Grapevine Golf' AND retailer_name!='Backcountry Discovery';

Tuning Tips#9

Minimize the number of table lookups (subquery blocks) in queries, particularly if our statements include subquery SELECTs or multicolumn UPDATEs. USE:

SELECTinventory_year,unit_costfrominventory_levelsWHEREproduct_number=(SELECTMAX(product_number)fromproduct)ANDaverage_unit_cost>8.5

gs_or_modify.sql ↓ [▶ ad	Imin > gosales_connection > gosaleshr_connection > gosalesmr_connection = IIIINVENTORY_LEVELS
۵ 🕲 😓 🖓 📃 🖉	1 0.98897684 seconds
SELECT inventory_yes	r,unit_cost from inventory_levels WHERE product_number=(SELECT MAX(product_number)from product
May average_unic_cos	
<	
Results Script Output	Secolain Autotrace BDBMS Output
INVENTORY_YEAR	UNIT_COST
2004	67.68
2004	67.68
2004	67.52
2004	67.42
2004	67.34
2004	67.36
2005	67.8
2005	67.48
2005	67.68
2005	67.6
2005	67.76
2005	67.82
2006	

When writing a sub-query (a SELECT statement within the WHERE or HAVING clause of another SQL statement):

i) Use a correlated (refers to at least one value from the outer query) sub-query when the return is relatively small and/or other criteria are efficient i.e. if the tables within the sub-query have efficient indexes.

ii) Use a non-correlated (does not refer to the outer query) sub-query when dealing with large tables from which you expect a large return (many rows) and/or if the tables within the sub-query do not have efficient indexes.

iii) Ensure that multiple sub-queries are in the most efficient order.

iv) Remember that rewriting a sub-query as a join can sometimes increase efficiency.

v) When doing multiple table joins consider the benefits/costs for each of EXISTS,

IN, and table joins. Depending on your data one or another may be faster. Note: IN is usually the slowest.

For Example: Write the query as

SELECT * from EMPLOYEE A WHERE EXISTS (SELECT * from Employee_Expense_Detail B WHERE b.employee_code= a.employee_code);

gs_or_modify.sql	> admin	tion b gosaleshr_connection	gosalesmr_connection 🛛 🔛	EMPLOYEE	
🕨 📃 🔯 🖪 🚳 🗆	🝓 🎉 🥜 0.59942144 sec	onds			gosaleshr_connection 🔻
SELECT * FROM E	MPLOYEE A WHERE EXISTS	(SELECT * from Employee_Expe	ense_Detail B MERE b.employ	<pre>yee_code= a.employee_code);</pre>	<u> </u>
					× .
					>
🕞 Results 📃 Script Ou	tput 📷Explain 📓Autotrace 🕻	3DBMS Output			
		· · · · · · · · · · · · · · · · · · ·			
EMPLOYEE_CODE	FIRST_NAME	FIRST_NAME_MB	LAST_NAME	LAST_NAME_MB	DATE_HIRED
10004	Denis	Denis	Pagé	 Pagé	11-DEC-01
10005	Élizabeth	Élizabeth	Michel	Michel	24-N0V-03
10006	Émile	Émile	Clermont	Clermont	10-MAY-06
10007	Étienne	Étienne	Jauvin	Jauvin	09-0CT-03
10012	Elsbeth	Elsbeth	Wiesinger	Wiesinger	22-MAR-05
10013	Else	Else	Mörike	Mörike	07-NOV-00
10014	Frank	Frank	Fuhlroth	Fuhlroth	12-MAY-03
10015	Gunter	Gunter	Erler	Erler	18-APR-07
10016	Björn	Björn	Winkler	Winkler	13-JUL-04
10017	Fritz	Fritz	Hirsch	Hirsch	20-DEC-04
10018	Jörg	Jörg	Kunze	Kunze	18-FEB-03
10019	Silvano	Silvano	Allessori	Allessori	27-JUN-06
<					>

Instead

of

SELECT * from EMPLOYEE A WHERE employee_code IN(SELECT employee_code from Employee_Expense_Detail);

vi)Use EXISTS instead of DISTINCT when using joins which involves tables having oneto-many relationship.

For Example: Write the query as

SELECT e.employee_code,e.first_name,e.last_name,e.date_hired from EMPLOYEE E WHERE EXISTS (SELECT expense_total from EMPLOYEE_EXPENSE_DETAIL D WHERE d.employee_code=e.employee_code);

gs_or_modify.sql	▶ admin	tion b gosaleshr_connection	Social gosalesmr_connection
> 📃 🖗 🖪 🐵	🗃 対 🥔 1.05615687 seco	nds	
SELECT e.emplo	oyee_code,e.first_name,e.l:	ast_name,e.date_hired from E	MPLOYEE E
WHERE EXISTS ((SELECT expense_total from	EMPLOYEE_EXPENSE_DETAIL D W	HERE d.employee_code=e.employee_code);
🕨 Results 📃 Script	Output 📷 Explain 📓 Autotrace 🕻	BDBMS Output	
🥔 🗟 🔒 🛛			
EMPLOYEE_CODE	FIRST_NAME	LAST_NAME	DATE_HIRED
10004	Denis	 Pagé	11-DEC-01
10005	Élizabeth	Michel	24-NOV-03
10006	Émile	Clermont	10-MAY-06
10007	Étienne	Jauvin	09-0CT-03
10012	Elsbeth	Wiesinger	22-MAR-05
10013	Else	Mörike	07-NOV-00
10014	Frank	Fuhlroth	12-MAY-03
10015	Gunter	Erler	18-APR-07
10016	Björn	Winkler	13-JUL-04
10017	Fritz	Hirsch	20-DEC-04
10018	Jörg	Kunze	18-FEB-03
10019	Silvano	Allessori	27-JUN-06

Instead of:

SELECTDISTINCTe.employee_code,e.first_name,e.last_name,e.date_hiredfromEMPLOYEEE,EMPLOYEE_EXPENSE_DETAILDWHERE

d.employee_code=e.employee_code;

Tuning Tips**#11**

Consider using DECODE to avoid having to scan the same rows repetitively or join the same table repetitively. Note, DECODE is not necessarily faster as it depends on your data and the complexity of the resulting query. Also, using DECODE requires you t change your code when new values are allowed in the field.

SELECT product_brand_code,DECODE(Product_Brand_Code,700,'Ramond' ,701,'Allen Solly' ,702,'Club Fox',703,'Peter England') result from Product_Brand;

gs_or_modify.sql	gosales_connection	gosaleshr_connection	a 🕞 gosalesmr_con	nection 🛛 🕪 a	admin IIIIPRODUCT
Image: A state of the state	🗃 📓 🥜 0.52270108 s	seconds			
SELECT product_b	rand_code,DECODE(Prod	uct_Brand_Code,700	,'Ramond'		
		,701	,'Allen Solly'		
		,702	,'Club Fox'		
		,703	<pre>,'Peter England')</pre>	result from	m Product_Brand;
AT					
🕞 Results 😼 Script Ou	tput 👸 Explain 📓 Autotrace	e 🗔 DBMS Output 🛛 🚷 O	WA Output		
🤣 🗟 昌					
PRODUCT_BRAND_CODI	E RESULT				
700	Ramond				
701	Allen Solly				
702	Club Fox				
703	Peter England				
704					

Oracle automatically performs simple column type conversions(or casting) when it compares columns of different types. Depending on the type of conversion, indexes may not be used. Make sure you declare your program variables as the same type as your Oracle columns, if the type is supported in the programming language you are using. USE:

SELECT order_detail_code,order_number,ship_date,product_number,unit_sale_price from order_details WHERE order_detail_code='1000644';

🛙 gs_or_modify.sql 🛛 🕨	gosales_connection	> gosaleshr_connection	gosalesmr_connection	ORDER_DETAILS	admin (111) ORE
> 📃 🖗 🖪 🔕 🛯) 📓 🥔 0.54705554 s	econds			
SELECT order_detai	l_code,order_number.	.ship_date,product_numbe	er,unit_sale_price f <mark>rom</mark> orde	r_details WHERE order_detai	l_code='1000644';
<					
Results Script Outpu	It MExplain Mutotrace		nut		
		Telephine output I would output			
ORDER_DETAIL_CODE	ORDER_NUMBER	SHIP_DATE	PRODUCT_NUMBER	UNIT_SALE_PRICE	
			74110		-
1000644	100034	21-0AN-04	/4110	10.14	
l rows selected					
HERE if order	_detail_code i	ndexed numeric,	then after implicit	conversion query w	vill
be:					
and the last second second				atustata a sama	

www.bispsolutions.com | www.hyperionguru.com | www.bisptrainings.com | Page 16

SELECT order_detail_code,order_number,ship_date,product_number,unit_sale_price from order_details WHERE order_detail_code=To_Number('1000644');

	-	-			. ,,			
🗐 gs_or_modify.sql	gosales_connection	> gosaleshr_connection	> gosalesmr_connection	⊳ admin	ORDER_DETAILS			E
🕨 📃 🔯 🗟 🍥 I	1 0.64229614 🔮	seconds					gosales_connectio	on 🔻
SELECT order_de	tail_code,order_number	,ship_date,product_n	umber,unit_sale_price	from order	_details WHERE order_d	etail_code=To_Number('1000644');	^
								>
AT.								
Results Script O	utput 🕲Explain 📓Autotrac	e 🗔 DBMS Output 💽 OWA	4 Output					
🥔 🗟 📇								
ORDER_DETAIL_CODE	CORDER_NUMBER	SHIP_DATE	PRODUCT_	NUMBER	UNIT_SALE_PRICE			
1000644	100034	21-JAN-04	74110		16.14			
l rows selected								
Thus, index	is used in thi	s case.						

Instead of:

SELECTorder_detail_code,order_number,ship_date,product_number,unit_sale_pricefromorder_detailsWHEREorder_type=121;HERE if order_type is indexed varchar2, then after implicit conversion query will be:SELECT emp_no, emp_name, sal FROM order_details WHERE TO_NUMBER(order_type)=121;

Thus, index will not be used in this case.

Tuning Tips#13

The most efficient method for storing large binary objects, i.e. multimedia objects, is to place them in the file system and place a pointer in the DB.

Tuning Tips#14

B-Tree Indexes do not store entries for NULL, so IS NULL is not indexable, but IS NOT NULL is indexable and thus if a huge table contains very few not null values then you should go for B-Tree indexes. On the other hand bitmap indexes support IS NULL condition.

SELECT product_number,base_product_number,product_type_code, introduction_date, discontinued_date from product WHERE discontinued_date IS NOT NULL;

🗐 gs_or_modify.sql 🛛 🚺 🔊	admin [> gosales_connect	ion 📄 🕞 gosaleshr_connectio	on > gosalesmr_connection	PRODUCT_FORECAST	G
ی او	📔 🥔 0.55313534 seconds				gosales_connection
SELECT product_num	ber,base_product_number,	product_type_code, int	roduction_date, discontin	nued_date from product WHERE	discontinued_date IS NOT NULL;
<					× *
	• ¹ ***********************************				
	Calexplain Salantocrace 🤯 DE	IMS Output 💓 OWA Output			
🥜 🖯 📇					
PRODUCT_NUMBER	BASE_PRODUCT_NUMBER	PRODUCT_TYPE_CODE	INTRODUCTION_DATE	DISCONTINUED_DATE	f
145150	145	960	01-JAN-03		
145160	145	960	01-JAN-03	31-MAR-06	
154110	154	960	01-JAN-03	31-JUL-06	
154130	154	960	01-JAN-03	31-DEC-06	
122120	122	961	01-JAN-03	31-JUL-06	
123120	123	961	01-JAN-03	28-FEB-06	
123160	123	961	01-JAN-03	30-JUN-05	
124130	124	961	01-JAN-03	28-FEB-06	
124140	124	961	01-JAN-03	31-0CT-06	
125120	125	960	01-JAN-03	31-MAR-07	

Avoid using functions on indexed columns unless a function-based index is created; as it leads to full table scan even though index exists on the column.

Tuning Tips#15

Avoid using the following: i)Boolean operators >, <, >=, <=, IS NULL, IS NOT NULL

ii)NOT IN, !=

iii)Like '%pattern', not exists

iv)Calculations on unindexed columns or (use union instead)

v)Having (use a WHERE clause instead when appropriate)

Tuning Tips#16

Do use the following:

i)Enable aliases to prefix all columns

ii)Place indexed columns higher in the WHERE clause

iii)Use SQL Joins instead of using sub-queries

iv)Make the table with the least number of rows, the driving table, by making it first in the FROM clause

Other important points for SQL Tuning i)Establish a tuning environment that reflects your production database ii)Establish performance expectations before you begin iii)Always Design and develop with performance in mind iv)Create Indexes to support selective WHERE clauses and join conditions v)Use concatenated indexes where appropriate vi)Consider indexing more than you think you should, to avoid table lookups vii)Pick the best join method viii)Nested loops joins are best for indexed joins of subsets ix)Hash joins are usually the best choice for "big" joins x)Pick the best join order xi)Pick the best "driving" table xii)Eliminate rows as early as possible in the join order xiii)Use bind variables. Bind variables are key to application scalability xiv)Use Oracle hints where appropriate xv)Compare performance between alternative syntax for your SQL statement xvi)Consider utilizing PL/SQL to overcome difficult SQL tuning issues xvii)Consider using third party tools to make the job of SQL tuning easier

Performing these steps is easy and provides a tremendous benefit and performance boost. Follow these simple steps and you can increase your system performance.

Tuning Tips#18

Using Function-based Indexes (FBI)

In almost all cases, the use of a built-in function like to_char, decode, substr, etc. in an SQL query may cause a full-table scan of the target table. To avoid this problem, many Oracle DBAs will create corresponding indexes that make use of function-based indexes. If a corresponding function-based index matches the built-in function of the query, Oracle will be able to service the query with an index range scan thereby avoiding a potentially expensive full-table scan.

The following is a simple example. Suppose the DBA has identified a SQL statement with hundreds of full-table scans against a large table with a built-in function (BIF) in the WHERE clause of the query. After examining the SQL, it is simple to see that it is accessing a customer by converting the customer name to uppercase using the upper BIF.

SELECT Upper(p.product_name),o.unit_sale_price from product_name_lookup p,order_details o WHERE p.product_number=o.product_number;

as or modify sal	\bigcirc gosaleshr connection \bigcirc gosaleshr connection~1	Dogalesm				
$ = g_{\text{control}} - g$						
SELECT UPPER(p.product_name), o.unit_s	ale_price from product_name_lookup p,order	details o				
where p.produce_namper-o.produce_nam	DEL,					
🕞 Results 📃 Script Output 📷 Explain 📓 Autotrace	e 🗔 DBMS Output 🛛 💽 OWA Output					
UPPER(P.PRODUCT_NAME)	UNIT_SALE_PRICE					
มีดพบ DOUBLE EDGE	16.14					
	16.14					
DUBBEL FICKKNIV	16.14					
	16.14					
нож "двойное лезвие"	16.14					
LASER DUO	16.14					
TVEEGGEN	16.14					
INCISIE DUBBEL ZAKMES	16.14					
DOUBLE EDGE	16.14					
	16.14					
1						

The table access full product option confirms that this BIF not using the existing index on the product_name column. Since a matching function-based index may change the execution plan, a function-based index can be added on upper(product_name).

Create index upper_product_name on product_name_lookup(upper(product_name)) pctfree 10 storage (initial 128k next 128k maxextents 2147483645 pctincrease 0);

💷 gs_or_modify.sql	gosales_connection	PRODUCT_NAME_LOOKUP	> gosaleshr_conn
> 📃 🕏 🖪 🧶	🝓 🎉 🥔 1.0592078 s	econds	
create index up pctfree 10 sto	pper_product_name on pp prage	roduct_name_lookup(upper	(product_name))
(initial 128k p	next 128k maxextents 23	147483645 pctincrease 0)	;
<			
Results Script C)utput 📷 Explain 🕅 Autotrac	:e 🗔 DBMS Output 💽 OWA Out	put
🥔 🗟 📇			

create index succeeded.

It can be risky to add indexes to a table because the execution plans of many queries may change as a result. This is not a problem with a function-based index because Oracle will only use this type of index when the query uses a matching BIF.

Tuning Tips#19

Using Temporary Tables

The prudent use of temporary tables can dramatically improve Oracle SQL performance. The following example from the DBA world can be used to illustrate this concept. The query could be formulated as an anti-join with a noncorrelated subquery as shown here:

🗊 gs_or_modify.sql 🛛	> gosales_connection == === EMPLO	YEE_HISTORY > gosaleshr_com	nection	> gosaleshr_connectior	∾1	> gosalesmr_connection	🕨 gosalesn	nr_connection^
🕨 🥃 🔯 🛃 🚳 I 1	👸 📓 🥔 0.55495685 seconds		_					gosaleshr_c
SELECT employee	_code, record_start_date, br	anch_code FROM EMPLOYEE_HIST	FORY WHERE	employee_code N	от п	N (SELECT terminatio	on_code FRO	EMPLOYEE
WHERE terminatio	on_code=150);							
AT								
问 Results 😼 Script Out	tput 🐚 Explain 🚵 Autotrace 🗔 DB	MS Output 🛛 🏹 OWA Output						
🥢 🖪 🚇 🛛								
EMPLOYEE_CODE	RECORD_START_DATE	BRANCH_CODE						
10074	13-JUL-99	21						
10075	26-JAN-04	23						
10075	23-FEB-05	23						
10076	07-MAR-05	14						
10077	26-JUN-02	23						
10077	26-JAN-04	23						
10078	13-JAN-03	6						
10078	23-FEB-05	6						
10079	08-SEP-03	24						
10080	16-APR-04	24						
10080	21-FEB-07	24						
10081	30-JUL-03	21						
10000								

Removing full-table scans with Oracle Text

One serious SQL performance problem occurs when the SQL LIKE operator is used to find a string within a large Oracle table column such as VARCHAR(2000), CLOB, or BLOB:

□ gs_or_modify.sql	SOREV >> gosaleshr_connection >> gosaleshr_connection
🕨 📃 🗟 🗟 🍥 🝓 🎉 🥔 0.54220998 seconds	
SELECT go obj name.go obj parent type from xgorev WH	ERE go change description LIKE '%10,000%';
🕞 Results 🔄 Script Output 📓 Explain 📓 Autotrace 🗔 DBMS Output	OWA Output
	~
GO_OBJ_NAME	GO_OBJ_PARENT_TYPE
EMPLOYEE_CODE	TABLE
EMPLOYEE_CODE	TABLE
EMPLOYEE_CODE	TABLE
MANAGER_CODE	TABLE
EMPLOYEE_CODE	TABLE
EMPLOYEE_CODE	TABLE
EMPLOYEE_CODE	TABLE
SUCCESSOR_EMPLOYEE_CODE	TABLE
FWAFOAFF CODF	TABLE

Since standard Oracle cannot index into a large column, their LIKE queries cause fulltable scans, and Oracle must examine every row in the table, even when the result set is very small. The following problems can be caused by unnecessary full-table scans:

i) Large-table full-table scans increase the load on the disk I/O sub-system ii)Small-table full-table scans(in the data buffer cause high consistent gets and drive up CPU consumption

The Oracle*Text utility, also called Oracle ConText and Oracle Intermedia, allows parsing through a large text column and index on the words within the column. Unlike ordinary b-tree or bitmap indexes, Oracle context ctxcat and ctxrule indexes are not updated as content is changed. Since most standard Oracle databases will use the ctxcat index with standard relational tables, the DBA must decide on a refresh interval. As a result, Oracle Text indexes are only useful for removing full-table scans when the tables are largely read-only and/or the end-users do not mind not having 100% search recall:

i) The target table is relatively static (e.g. nightly batch updates)

ii) The end-users would not mind missing the latest row data

Oracle Text works with traditional data columns as well as with MS-Word docs and Adobe PDF files that are stored within Oracle. Oracle Text has several index types:

iii) CTXCAT Indexes: A CTXCAT index is best for smaller text fragments that must be indexed along with other standard relational data (VARCHAR2). WHERE CATSEARCH(text_column, 'ipod')> 0;

iv) CONTEXT Indexes: The CONTEXT index type is used to index large amounts of text such as Word, PDF, XML, HTML or plain text documents.

WHERE CONTAINS(test_column, 'ipod', 1) > 0

v)CTXRULE Indexes: A CTXRULE index can be used to build document classification applications.

These types of indexes allow users to replace the old-fashioned SQL LIKE syntax with CONTAINS or CATSEARCH SQL syntax.

When the query is executed with the new index, the full-table scan is replaced with a index scan, thereby greatly reducing execution speed and improving hardware stress: Execution Plan

- 0 SELECT STATEMENT Optimizer=FIRST_ROWS
- 1 0 SORT (ORDER BY)
- 2 1 TABLE ACCESS (BY INDEX ROWID) OF 'BIGTAB'
- 3 2 DOMAIN INDEX OF 'TEXT-COLUMN_IDX'