Database Tuning for Better Performance

by Rafi Balbirsky

In association with:

PerformanceArt

Passionate about Performance

Webinar Targets

- Present the challenges of managing performance in Multi-DB environments

  (Oracle, SQL Server, MySQL, DB2)
- Focus on the common performance issues of all DB applications and establish a common language
- Introduce a new methodology for tuning applications across different databases

Multi-DB Challenges

- Business processes and applications are spanned across different DBs
- Each database will have its own set of performance tools and guidelines for best practice – and its own team of internal and external advisors

Multi-DB Challenges

Business Performance information is fragmented over different databases

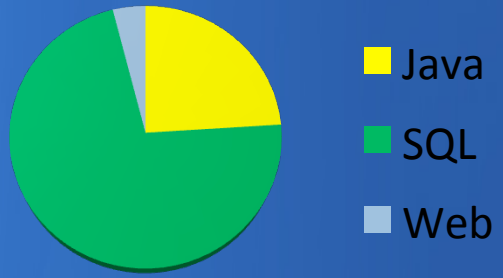
Performance of DB Applications

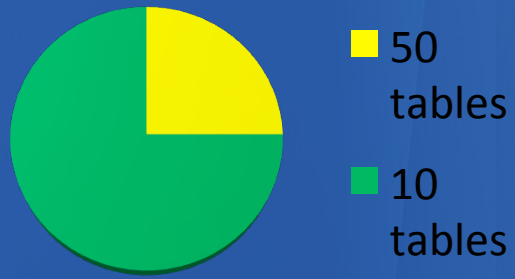
- 80% of application performance problems lie in the application
- Dealing with them by DB, storage or hardware upgrades is costly, not scalable and has very poor ROI compared to application tuning

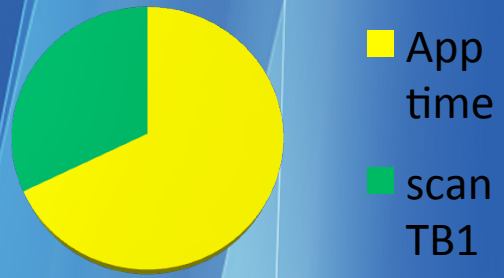
Pareto Principle in SQL Tuning

- 80% of application time is spent in 20% of the application code (SQL calls)
- 80% of application DB time is spent in 20% of the SQL statements it issues
- 80% of SQL time is spent in 20% of its Execution Plan steps

Application time breakdown







Different Databases – Same Issues

- Oracle, SQL server, MySQL and DB2 are different in many aspects but they use B-tree indices in basically the same way.
- They all suffer in the same way from sort on disk, heavy joins, full scans in accessing the data and index overhead in updating the data

Application Time in DB Breakdown – Typical Appearance

Resource Pie – Optimal Appearance

Common Performance Issues

Application layer

- Rerunning SQL
- Reparsing SQL

The solution is in the application.

Common Performance Issues

DB Layer

- Full scans
- Index range
- Index overhead
- Sorts
- Joins

New Approach

- A methodology for tuning objects by schema which is applicable to all DB types
- Enables a unified view of performance across all databases

Key Metrics of Object Performance

- Access time
- Access type (full scan, index range, sort, join etc.)
- Access wait (IO, CPU, Lock, DB engine)
- Statistics (Rows#, distinct values etc.)

Traditional Application Tuning

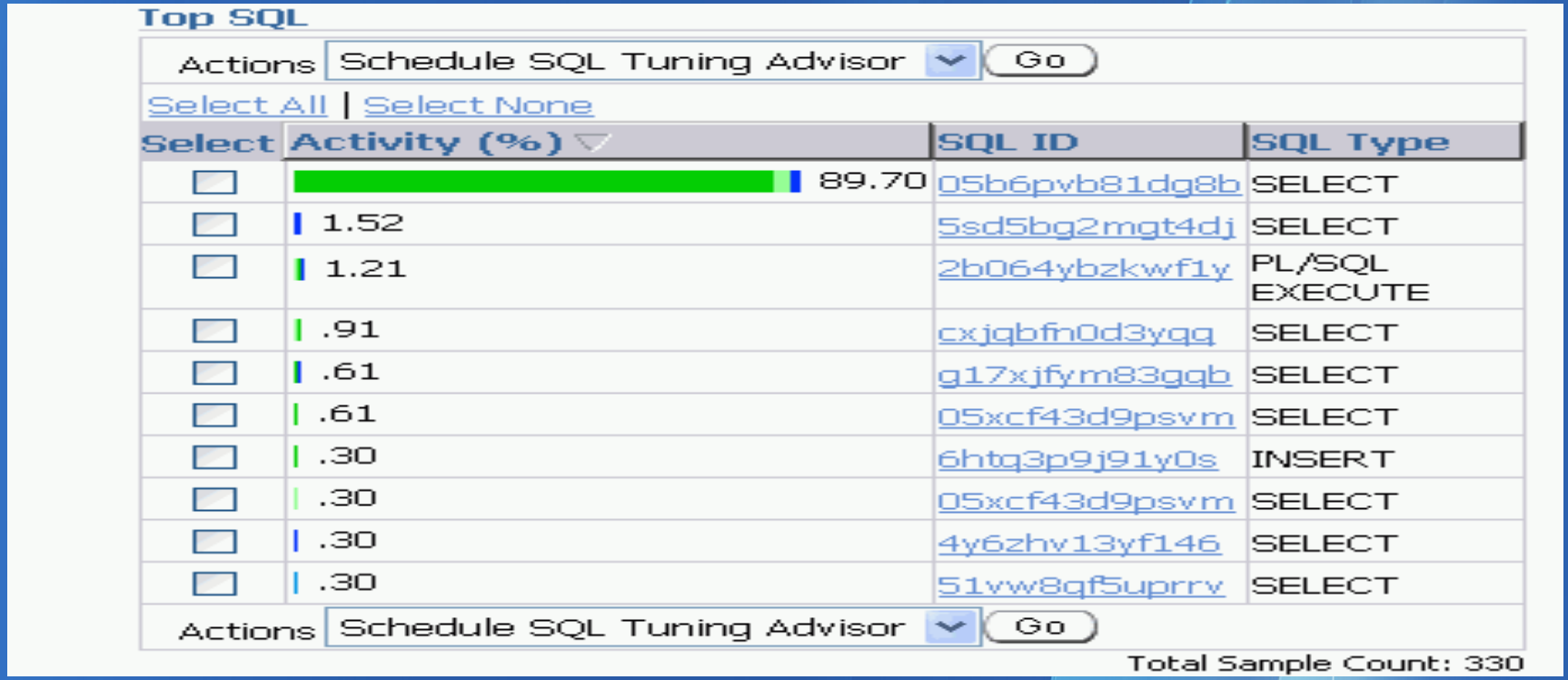
Currently, DB tuning and monitoring efforts use two methodologies as a starting point for the tuning process:

- Instance level - Instance stats
- SQL statement level – top statements

What's Wrong with That?

- Poor link between schema / business elements and performance information
- Poor root cause analysis
- Poor visibility to actual usage of schema objects
- Missing highly beneficial tuning opportunities

Top SQL is Misleading



Top SQL

Actions Schedule SQL Tuning Advisor Go

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
☐	89.70	05b6pvb81dg8b	SELECT
☐	1.52	5sd5bg2mgt4dj	SELECT
☐	1.21	2b064ybzkwf1y	PL/SQL EXECUTE
☐	.91	cxjqbfn0d3yqq	SELECT
☐	.61	g17xjfym83gqb	SELECT
☐	.61	05xcf43d9psvm	SELECT
☐	.30	6htq3p9j91y0s	INSERT
☐	.30	05xcf43d9psvm	SELECT
☐	.30	4y6zhv13yf146	SELECT
☐	.30	51vw8qf5uprrv	SELECT

Actions Schedule SQL Tuning Advisor Go

Total Sample Count: 330

Real Application Bottlenecks

Many statements use the same access patterns on the same objects but won't show up in the top SQL.

Their aggregated impact is missed, together with their tuning opportunities.

```
4   top statements per object
```

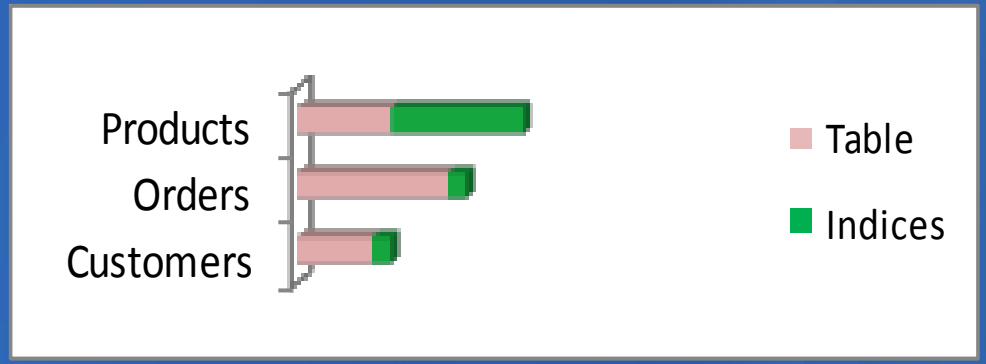
utput | Explain | Autotrace | DBMS Output | OWA Output

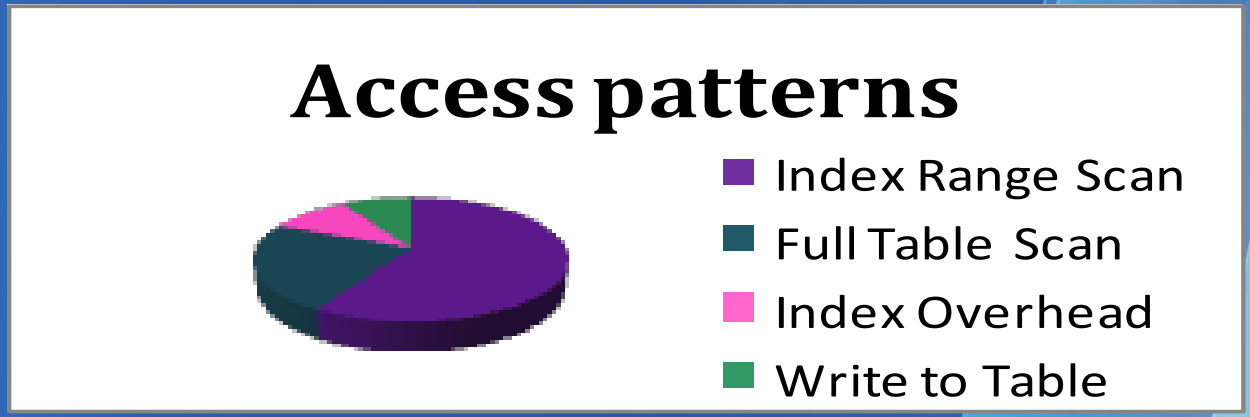
AME	SQL_ID	CPU	WAIT	IO	TOTAL	SQL_TEXT
	gya73as76zut0	235	3	577	815	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	g9rh4b2h9x1nj	251	2	562	815	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	9a1zvvg3kqama	251	3	559	813	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	6vnuy4xcabq9u	242	2	569	813	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	amjnjpgr2zypk	243	3	564	810	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	6hy0c8y1d45y5	1	0	643	644	SELECT PAYMENT_CODE FROM PAYMENTS WHERE PHONE = TRIM(:B2) AND PHONE_PRE = TRIM(:B1) A
	1s4666kcuzhdb	127	0	0	127	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	c5fzf1whpf7rd	10	1	87	98	SELECT ROWID,PAYMENT_CODE,PAYMENT_TIMESTAMP,PAYMENT_BY,CUST,PAYMENT_TYPE,FORM_N
	86cr03g23p46x	0	0	64	64	SELECT PAYMENT_CODE,S.CUST,PAYMENT_TYPE, GET_CUST_NAME(S.CUST) CUST_NAME, REPLACE(
	5zs50mjha9n74	0	0	27	27	SELECT PAYMENT_CODE FROM PAYMENTS WHERE ID_NUMBER = TRIM(:B1) AND PAYMENT_TIMESTAM
	7w78fpu7ntcp8	3	0	13	16	SELECT PAYMENT_CODE FROM PAYMENTS WHERE PAYMENT_CODE NOT IN (SELECT PAYMENT_CODE
	bwqps49vt024h	1	0	15	16	SELECT ROWID,PAYMENT_CODE,PAYMENT_TIMESTAMP,PAYMENT_BY,CUST,PAYMENT_TYPE,FORM_N
	7h5v67mc6fzj0	1	0	10	11	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE PAYMENT_TIMESTAMP BETWEEN TO_DATE(:b1
	bg8x29vhbn9cs	11	0	0	11	SELECT SUM(CALLS_ANSWERED) FROM CTI.CTI_ICDNSTAT WHERE CDN= :B2 AND TRUNC(TIMESTAMP)
	9nnzz1jn5f41s	0	5	0	5	DECLARE job BINARY_INTEGER := :job; next_date DATE := :mydate; broken BOOLEAN := FALSE; BEGIN d
	86mm6ythqfbqz	4	0	0	4	select p.cust,count(*), (select count(*) from payments p1 where TRUNC(P1.PAYMENT_TIMESTAMP) =TRU
	8szmwam7fysa3	1	0	3	4	insert into wri$_adv_objspace_trend_data select timepoint, space_usage, space_alloc, quality from table
	2n6xk782a8ray	4	0	0	4	SELECT NVL(TRUNC(SUM(CALL_LENGTH) / 60 ,2),0) FROM PAYMNT.PAYMENTS WHERE PAYMENT_TI
	7wgks43wrjtrz	0	0	2	2	SELECT U.SPACE_USED, U.SPACE_ALLOCATED FROM TABLE(DBMS_SPACE.OBJECT_SPACE_USAGE_
	8ay8h4m2wcbdj	0	0	2	2	SELECT ROWID,PAYMENT_CODE,PAYMENT_TIMESTAMP,PAYMENT_BY,CUST,PAYMENT_TYPE,FORM_N
	06dnm2r3vkbqv	0	0	2	2	insert into paymnt.payments (payment_code,payment_timestamp,payment_by,payment_at,cust,payment

Line 54 Column 45 | Insert | Modified | Windows: CR/... Ed

C:\WINDOWS\... | C:\Documents ... | EditPlus - [C:\D... | סילבוס לטלא... | Findings.pptx | 11:39 A

Aggregate by Application Objects





Object Tuning Methodology

Anchor Point – Top Objects

- Start with the central and most used tables (scope can be instance or application)
- Table ***total*** access time includes all the access time of its indices

Create Tuning Sets from Explain Plan and Performance Information

A tuning set consists of SQL statements grouped by:

- Application identifiers
- Application objects
- Access types

Object Tuning Methodology

Applies for all relational DB:

- Full table scan has different names in different DB but means the same
- Range scans on B-tree indices in all DBs have the same inefficiencies
- Adding indices to a table will always degrade update activities
- Sorts in all DBs use temporary space on disk when the sort result set is too big to be handled in memory

Aggregate Full Scan Time on a Table

- See how much time the ***application*** is spending on full scans
- Look at the top SQL in this group and identify possible indices

0.32335591 seconds

```
----  top objects

select * from (
select
   obj.object_name ,
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	OBJECT_NAME	CPU	WAIT	IO	TOTAL
1	PAYMENTS	1396	14	3729	5139
2	DEBIT	52	61	2140	2253
3	INBOUND_CM	104	1788	91	1983
4	ONLINE_MESSAGE	1100	0	0	1100
5	TARGETS_DATA	379	2	706	1087
6	REPORT_CUST	5	789	0	794
7	PRODS_Q_LOG__PK_PROD	691	0	4	695
8	CTI_IAGENTPERFORMANCESTAT	233	1	187	421
9	NU_TARGETS	355	1	25	381
10	PRODS_Q_LOG	311	0	2	313
11	DEBIT_ACCOUNT	283	0	20	303
12	EMP_EVENTS	277	0	2	279
13	INBOUND_CM__SK_CUST_DATE	228	0	33	261
14	PAYMENT_SOURCE_CUST_ACC_PT	178	0	83	261
15	WORK_CLOCK__SK_ACTIVE	233	0	20	253
16	PAYM_CARD_AYTH	2	0	241	243
17	CUST_INPUT_PAY	1	0	184	185
18	ONLINE_MESSAGE__PK_CODE	175	0	0	175
19	CTI_IDNISSTAT	99	0	73	172
20	TARGETS_DATA__PK_TARGET_1	42	0	118	160

Line 52 Column 21 | Insert | Modif

Oracle SQL De... | C:\WINDOWS\... | C:\Documents ... | EditPlus - [C:\D... | סילבוס לטלאו... | Findings.pptx

0.23328803 seconds

```
---- top operations for object

select * from (
select
   obj.object_name ,
   obj.owner ,
 --  ash.SQL_ID
    ash.SQL_PLAN_OPERATION||' '|| ash.SQL_PLAN_OPTIONS access_type,
  --    ash.SQL_PLAN_HASH_VALUE Plan_hash,
     sum(decode(ash.session_state,'ON CPU',1,0))      "CPU",
     sum(decode(ash.session_state,'WAITING',1,0))     -
     sum(decode(ash.session_state,'WAITING', decode(wait_class, 'User I/O',1,0),0))    "WAIT" ,
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	OBJECT_NAME	OWNER	ACCESS_TYPE	CPU	WAIT	IO	TOTAL
1	PAYMENTS	PAYMNT	TABLE ACCESS FULL	1252	14	2958	4224
2	PAYMENTS	PAYMNT	TABLE ACCESS BY INDEX ROWID	5	0	765	770
3	PAYMENTS	PAYMNT	INDEX FAST FULL SCAN	131	0	0	131
4	PAYMENTS	PAYMNT	COLLECTION ITERATOR PICKLER FETCH	1	0	5	6
5	PAYMENTS	PAYMNT	INDEX RANGE SCAN	5	0	0	5
6	PAYMENTS	PAYMNT	HASH UNIQUE	1	0	0	1
7	PAYMENTS	PAYMNT	HASH JOIN	1	0	0	1

Line 114 Column 1 | Insert

Oracle SQL De... | C:\WINDOWS\... | C:\Documents ... | EditPlus - [C:\D... | סילבוס לטלאו... | Findings.pptx

```
4   top statements per object
```

utput | Explain | Autotrace | DBMS Output | OWA Output

AME	SQL_ID	CPU	WAIT	IO	TOTAL	SQL_TEXT
	gya73as76zut0	235	3	577	815	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	g9rh4b2h9x1nj	251	2	562	815	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	9a1zvvg3kqama	251	3	559	813	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	6vnuy4xcabq9u	242	2	569	813	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	amjnjpgr2zypk	243	3	564	810	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	6hy0c8y1d45y5	1	0	643	644	SELECT PAYMENT_CODE FROM PAYMENTS WHERE PHONE = TRIM(:B2) AND PHONE_PRE = TRIM(:B1) A
	1s4666kcuzhdb	127	0	0	127	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE TRUNC(PAYMENT_TIMESTAMP) = TRUNC(:b1) A
	c5fzf1whpf7rd	10	1	87	98	SELECT ROWID,PAYMENT_CODE,PAYMENT_TIMESTAMP,PAYMENT_BY,CUST,PAYMENT_TYPE,FORM_N
	86cr03g23p46x	0	0	64	64	SELECT PAYMENT_CODE,S.CUST,PAYMENT_TYPE, GET_CUST_NAME(S.CUST) CUST_NAME, REPLACE(
	5zs50mjha9n74	0	0	27	27	SELECT PAYMENT_CODE FROM PAYMENTS WHERE ID_NUMBER = TRIM(:B1) AND PAYMENT_TIMESTAM
	7w78fpu7ntcp8	3	0	13	16	SELECT PAYMENT_CODE FROM PAYMENTS WHERE PAYMENT_CODE NOT IN (SELECT PAYMENT_CODE
	bwqps49vt024h	1	0	15	16	SELECT ROWID,PAYMENT_CODE,PAYMENT_TIMESTAMP,PAYMENT_BY,CUST,PAYMENT_TYPE,FORM_N
	7h5v67mc6fzj0	1	0	10	11	SELECT COUNT(*) FROM PAYMNT.PAYMENTS WHERE PAYMENT_TIMESTAMP BETWEEN TO_DATE(:b1
	bg8x29vhbn9cs	11	0	0	11	SELECT SUM(CALLS_ANSWERED) FROM CTI.CTI_ICDNSTAT WHERE CDN= :B2 AND TRUNC(TIMESTAMP)
	9nnzz1jn5f41s	0	5	0	5	DECLARE job BINARY_INTEGER := :job; next_date DATE := :mydate; broken BOOLEAN := FALSE; BEGIN d
	86mm6ythqfbqz	4	0	0	4	select p.cust,count(*), (select count(*) from payments p1 where TRUNC(P1.PAYMENT_TIMESTAMP) =TRU
	8szmwam7fysa3	1	0	3	4	insert into wri$_adv_objspace_trend_data select timepoint, space_usage, space_alloc, quality from table
	2n6xk782a8ray	4	0	0	4	SELECT NVL(TRUNC(SUM(CALL_LENGTH) / 60 ,2),0) FROM PAYMNT.PAYMENTS WHERE PAYMENT_TI
	7wgks43wrjtrz	0	0	2	2	SELECT U.SPACE_USED, U.SPACE_ALLOCATED FROM TABLE(DBMS_SPACE.OBJECT_SPACE_USAGE_
	8ay8h4m2wcbdj	0	0	2	2	SELECT ROWID,PAYMENT_CODE,PAYMENT_TIMESTAMP,PAYMENT_BY,CUST,PAYMENT_TYPE,FORM_N
	06dnm2r3vkbqv	0	0	2	2	insert into paymnt.payments (payment_code,payment_timestamp,payment_by,payment_at,cust,payment

Line 54 Column 45 | Insert | Modified | Windows: CR/... Ed

C:\WINDOWS\... | C:\Documents ... | EditPlus - [C:\D... | סילבוס לטלא... | Findings.pptx | 11:39 A

5.49952841 seconds

teleall

```
object_name,
SQL_PLAN_OPERATION||' '|| ash.SQL_PLAN_OPTIONS,
access_predicates,
filter_predicates
L_PLAN_HASH_VALUE
r by sum(decode(session_state,'ON CPU',1,1))   desc
ere  rownum < 50

 top predicates per table

ct * from (
ct
   ash.sql_id ,
   ash.SQL_PLAN_HASH_VALUE Plan_hash,
```

ults | Script Output | Explain | Autotrace | DBMS Output | OWA Output

FILTER_PREDICATES	ACCESS_PREDICATES
TRUNC(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"))=TRUNC(SYSDATE@!-TO_NUMBER(:D))	"P1"."PAYMENT_SOURCE"=3 AND "P1"."CUST"=:B1
TRUNC(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"))=TRUNC(SYSDATE@!-TO_NUMBER(:D))	"P1"."PAYMENT_SOURCE"=1 AND "P1"."CUST"=:B1
TRUNC(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"))=TRUNC(SYSDATE@!-TO_NUMBER(:D))	"P1"."PAYMENT_SOURCE"=2 AND "P1"."CUST"=:B1
(TO_CHAR(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"),'HH24')=:B1 AND TRUNC(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"))=TRUNC(SYSDATE@!-TO_NUMBER(:D)))	"P1"."PAYMENT_SOURCE"=4
TRUNC(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"))=TRUNC(SYSDATE@!-TO_NUMBER(:D))	"P1"."PAYMENT_SOURCE"=4 AND "P1"."CUST"=:B1
TRUNC(INTERNAL_FUNCTION("P1"."PAYMENT_TIMESTAMP"))=TRUNC(SYSDATE@!-TO_NUMBER(:D))	"P1"."PAYMENT_SOURCE"=5 AND "P1"."CUST"=:B1

ned: 6 | Line 212 Column 21 | Insert | Modified | Windows: CR/... Edi

Oracle SQL ... | C:\WINDO... | C:\Docume... | EditPlus - [... | סילבוס ל... | Application ... | Presentation1 | 11:23 A

B-tree Indices

- Most applications rely heavily on B-tree indices across all DBs
- Indices on large and central tables can be large and consume high amounts of I/O, CPU and memory
- Avoiding full scans by using indices can create new performance issues, slowing down application responsiveness

Aggregate Index Time to Table

Now you can assess:

- Total time of access to a table
- Index usage
- Index misusage
- Index overhead

Now you can optimize your index structure

Object Issues – Bad Index Structure

Common index related problems:

- Matching level / Clustering Factor
- Index overhead
- Change to index structure can boost performance
- Tuning gain can be calculated

Index Only / Covering Index Access

- No table blocks are read
- Clustering factor has no effect
- Sort can be avoided

Index Overhead

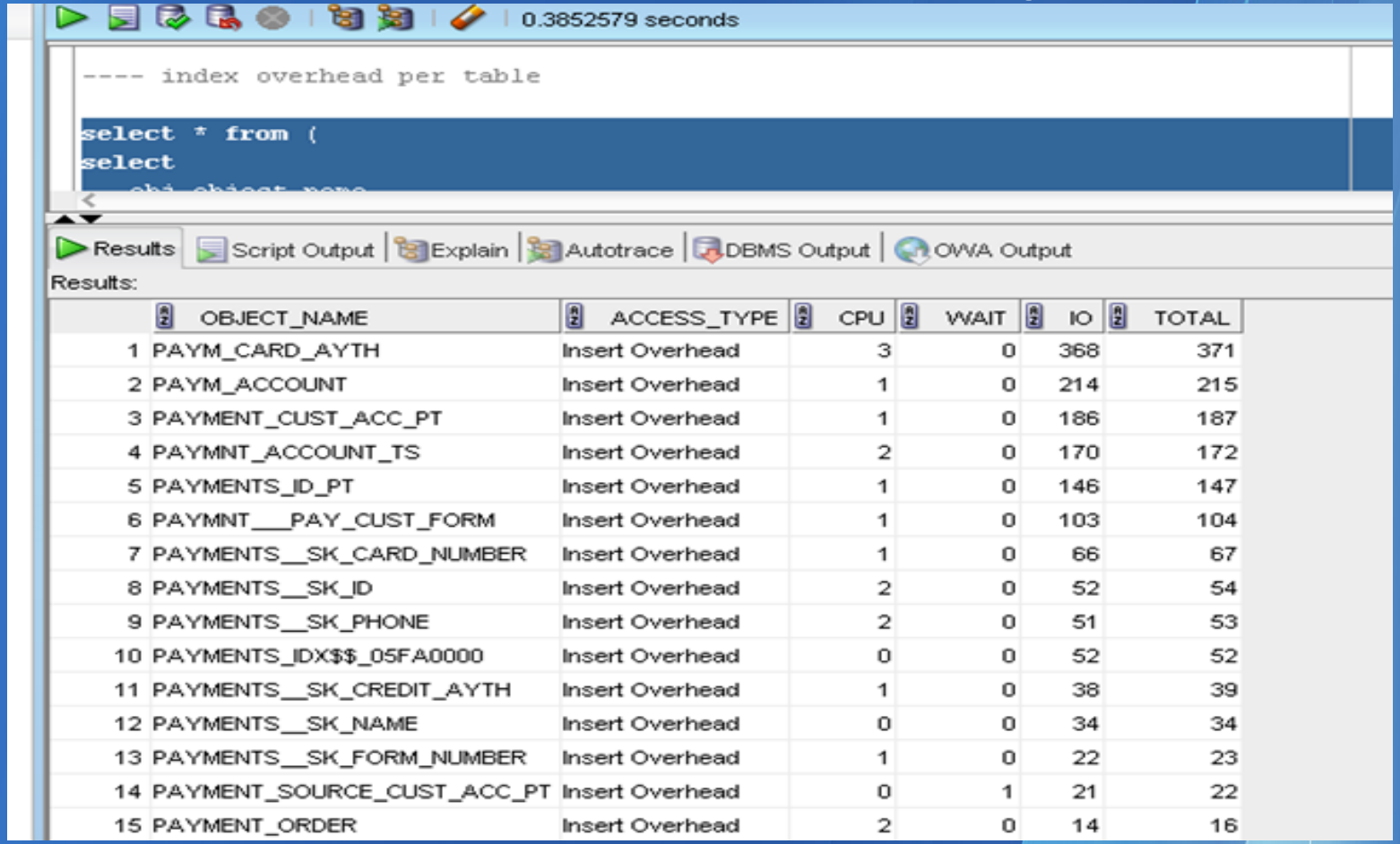
Finally get an answer for this important question

How much do I pay for my indices?

Index Overhead Solution

- Detect unused indices
- Detect subset indices
- Detect unused columns that inflate index size

Index Overhead – Real Example



0.3852579 seconds

```
---- index overhead per table

select * from (
select
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	OBJECT_NAME	ACCESS_TYPE	CPU	WAIT	IO	TOTAL
1	PAYM_CARD_AYTH	Insert Overhead	3	0	368	371
2	PAYM_ACCOUNT	Insert Overhead	1	0	214	215
3	PAYMENT_CUST_ACC_PT	Insert Overhead	1	0	186	187
4	PAYMNT_ACCOUNT_TS	Insert Overhead	2	0	170	172
5	PAYMENTS_ID_PT	Insert Overhead	1	0	146	147
6	PAYMNT___PAY_CUST_FORM	Insert Overhead	1	0	103	104
7	PAYMENTS__SK_CARD_NUMBER	Insert Overhead	1	0	66	67
8	PAYMENTS__SK_ID	Insert Overhead	2	0	52	54
9	PAYMENTS__SK_PHONE	Insert Overhead	2	0	51	53
10	PAYMENTS_IDX$$_05FA0000	Insert Overhead	0	0	52	52
11	PAYMENTS__SK_CREDIT_AYTH	Insert Overhead	1	0	38	39
12	PAYMENTS__SK_NAME	Insert Overhead	0	0	34	34
13	PAYMENTS__SK_FORM_NUMBER	Insert Overhead	1	0	22	23
14	PAYMENT_SOURCE_CUST_ACC_PT	Insert Overhead	0	1	21	22
15	PAYMENT_ORDER	Insert Overhead	2	0	14	16

Schema Analysis - Local

- Often more effective than SQL tuning
- Solid entity for DBA. Fits well in solution space
- Helps in optimizing the match between application design and usage
- Exposes hidden performance bottlenecks
- Reveals usage of schema objects
- Enables effective usage of DB advisors

Schema Analysis - Global

- Enables continuous and full visibility of DB layer impact on business performance
- Cuts down the time and cost of performance management
- Increases company's scalability and its flexibility in answering business needs

Thank you for your attention

Questions?

Passionate about Performance