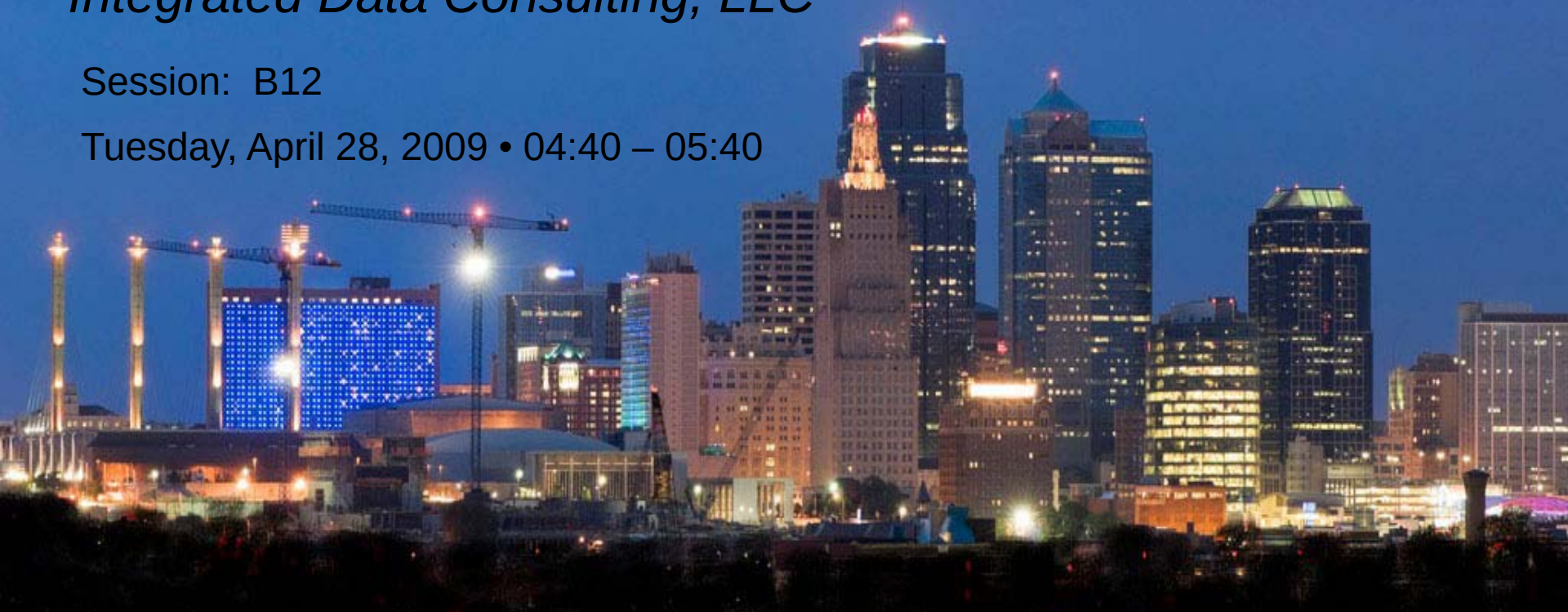
Informix SQL Performance Tuning Tips

Jeff Filippi

Integrated Data Consulting, LLC

Session: B12

Tuesday, April 28, 2009 • 04:40 – 05:40



The Power Conference for Informix Professionals

www.iiug.org

Session: B12

Informix SQL Performance Tuning Tips

Jeff Filippi

Integrated Data Consulting, LLC

jeff.filippi@itdataconsulting.com

Introduction

- 19 years of working with Informix products
- 14 years as an Informix DBA
- Worked for Informix for 5 years 1996 – 2001
- Certified Informix DBA
- Started my own company in 2001 specializing in Informix Database Administration consulting
- IBM Business Partner
- OLTP and Data warehouse systems
- Informix 4.x, 5.x., 7.x, 9.x, 10.x, 11.10, 11.50

Overview

- Identify Problem SQL Statements
- Tracing SQL in Informix 11
- Options Available with Set Explain & Reading sqexplain output with tuning examples
- Methods to use for Improving SQL Performance
- Know your Application in Tuning SQL
- Case Studies

Identify Problem SQL Statements

- First you have to identify what SQL statements are the culprits in causing performance issues
  - Use “onstat –g ntt” to identify the last time read/writes occurred
  - Gather slow SQL statements from onstats, 3rd party tools like Server Studio, Cobrasonic, etc.
  - Look at how many times SQL statements have been executed using SQL Statement Cache (onstat –g ssc)
  - Informix 11 tracing feature (SQLTRACE)
  - Review with developers known problem areas in the application
  - Verify update statistics are current
  - Review what tables/indexes have the most reads

Use SQL Statement Cache

- onstat –g ssc
- Look at SQL's with a large number of executions.
- Saving even a second on a SQL statement that is executed 1 million times can make a difference in performance.

Use SQL Statement Cache

IBM Informix Dynamic Server Version 11.10.HC2X3 -- On-Line -- Up 23 days 23:46:38 -- 2530056 Kbytes

Statement Cache Summary:

#lrus	currsize	maxsize	Poolsize	#hits	nolimit
8	22491472	40960000	11710464	10	0

Statement Cache Entries:

lru	hash	ref_cnt	hits	flag	heap_ptr	database	user
0	140	0	15	-F	bb164020	ntlcom	informix

```
select descr, rowid, seq_nbr from fl_cntrl where uid in ( 'all',
  'cschabel' ) and program_name in ( 'all', 'cr_inv_dl' ) and exc_type     is not null order by seq_nbr
```

lru	hash	ref_cnt	hits	flag	heap_ptr	database	user
0	116	0	1011	-F	aa23bc20	ntlcom	informix

```
update state_tax set row_status = "V", updt_user_id =? where seq_nbr =?   And     ( rec_type = 6 or rec_type = 7)
```

lru	hash	ref_cnt	hits	flag	heap_ptr	database	user
0	207	0	6003	-F	a004f820	ntlcom	informix

```
select count ( *) from invoice_state where cde = "CORR" and seq_nbr =?
```

lru	hash	ref_cnt	hits	flag	heap_ptr	database	user
2	138	0	6244	-F	afa9ec20	ntlcom	informix

```
select int_comm, int_comm2, updt_user_id from invoice_cmnts where seq_nbr     =? and extend ( updt_dte, year to second) =
      ( select max ( extend ( updt_dte, year to second)) from invoice_cmnts where seq_nbr =?)
```

Review number of reads on table/index

- Use the table SYSPTPROF to look at the buffer reads, page reads, sequencial scans.
  - SELECT tabname[1,25], bufreads, pagreads, isreads, seqscans

    FROM sysmaster:sysptprof

    ORDER BY 2 desc

Review number of reads on table/index – Cont'd

tabname	bufreads	pagreads	isreads
trnsit_1	**-2122091061**	**429**	**1630786736**
trnsit_1	**-812314372**	**3162**	**-678524115**
trnsit_1	**-110705308**	**233**	**-390409810**
im_p_route_1	**1806427782**	**247**	**865918944**
ed_rl_event	1749246222	23550	1709746386
loc_sup_data	1479941490	39	1108625557
ed_rl_event_3	1186682507	2668713	789739458
460_4902	1125173161	1003575	373042018
ed_rl_event_4	893520660	25725	886704767
im_mv_event	870108889	30477208	780365364

After adding index – Reduced bufreads

tabname	bufreads	pagreads	isreads
140_409	-845911921	0	1722690950
cntrct_num	1297728722	1	2868878
221_1360	812752007	0	406226191
.....			
trnsit	17215024	22	12007375
trnsit_ix1	15638629	106	12898627
im_p_route_1	23045	347	12904

Tracing SQL in Informix 11

- There are more ways to find information to tune in Informix 11 utilizing the tracing of SQL
  - onconfig parameter: SQLTRACE
  - SQL Admin API
  - Ability to trace by database (11.50XC3)
  - Open Admin Tool (OAT)

Tracing SQL in Informix 11

- There are a couple ways to turn tracing on in Informix 11
  - Onconfig parameter: SQLTRACE
    - level = [off,low,med,high]
    - ntraces = [# of traces]
    - size = [size of each trace buffer in kb]
    - mode = [global,user]
    - Example:
      - SQLTRACE level=low,ntraces=1000,size=2,mode= global

        (This allows me to trace the last 1000 sql statements of the instance)
  - Open Admin Tool (OAT)

Tracing SQL in Informix 11.50

- Improved SQL tracing with the SQL Admin API in Informix 11.50FC3
  - You can use these new commands to manage SQL tracing by databases.
    - SET SQL TRACING DATABASE ADD {Database}
      CLEAR
      LIST
      REMOVE {Database}
  - You can also suspend and resume all tracing at the server without deallocating any resources.
    - SET SQL TRACING SUSPEND/RESUME

Tracing SQL in Informix 11 (cont'd)

- Here is how you enable tracing thru the "sysadmin" database by running the following command:
  - EXECUTE FUNCTION task("set sql tracing on",1000,2,"low","global")

- To validate that tracing is turned on by:
  - onstat –g his
  - This option prints information about the SQLTRACE configuration parameter.

Tracing SQL in Informix 11 (cont'd)

onstat –g his

IBM Informix Dynamic Server Version 11.10.HC2X3 -- On-Line -- Up 25 days 23:44:15 -- 2530056 Kbytes

Statement history:

Trace Level	Low
Trace Mode	Global
Number of traces	3000
Current Stmt ID	939412632
Trace Buffer size	2024
Duration of buffer	8241 Seconds
Trace Flags	0x00001611
Control Block	9df53018

Tracing SQL in Informix 11 (cont'd)

Statement # 94653656: @ 9df68cb0

Database: 0x1700002
Statement text:
SELECT * FROM invc_state WHERE seq_nbr = ?

Iterator/Explain
==========

ID	Left	Right	Est Cost	Est Rows	Num Rows	Type
1	0	0	26	4	6	Index Scan

Statement information:

Sess_id	User_id	Stmt Type	Finish Time	Run Time
7640	1001	SELECT	18:44:20	0.0006

Tracing SQL in Informix 11 (cont'd)

Statement Statistics:

Page Read	Buffer Read	Read % Cache	Buffer IDX Read	Page Write	Buffer Write	Write % Cache
0	17	100.00	0	0	0	0.00

Lock Requests	Lock Waits	LK Wait Time (S)	Log Space	Num Sorts	Disk Sorts	Memory Sorts
13	0	0.0000	0.000 B	0	0	0

Total Executions	Total Time (S)	Avg Time (S)	Max Time (S)	Avg IO Wait	I/O Wait Time (S)	Avg Rows Per Sec
7294	6.6040	0.0009	0.0015	0.000000	0.000000	10869.5652

Estimated Cost	Estimated Rows	,	Actual Rows	SQL Error	ISAM Error	Isolation Level	SQL Memory
26	4		6	0	0	LC	13416

Tracing SQL in Informix 11 (cont'd)

- You can also get information on the tracing thru the **"syssqltrace"** table in the sysmaster database.
  - Ex. {# of queries that ran > 2 seconds)

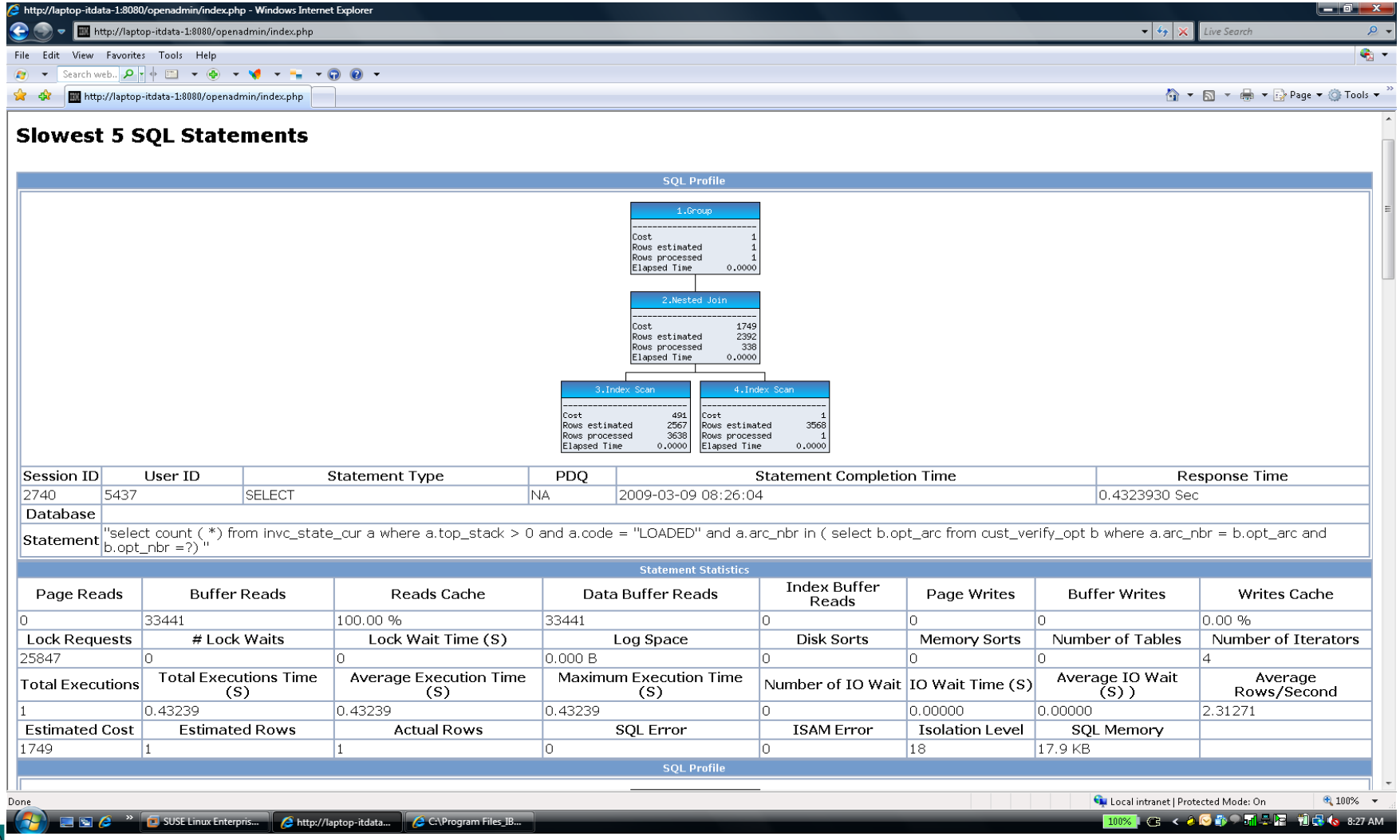
```
SELECT count(*)
FROM syssqltrace
WHERE sql_totaltime > 2;
```

- Another useful table is the **"syssqltrace_iter"** which gives information in the form of an iteration tree for each SQL. It allows you to identify which part of the query plan took the most time to run.

Use Open Admin Tool (OAT)

- Use the Open Admin Tool to find slow SQL statements.
  - Under “System Reports” there is an option to show the 5 slowest SQL’s.

Use Open Admin Tool (OAT) – cont'd



Session ID	User ID	Statement Type	PDQ	Statement Completion Time	Response Time
2740	5437	SELECT	NA	2009-03-09 08:26:04	0.4323930 Sec
Database					
Statement	"select count (*) from invc_state_cur a where a.top_stack > 0 and a.code = "LOADED" and a.arc_nbr in (select b.opt_arc from cust_verify_opt b where a.arc_nbr = b.opt_arc and b.opt_nbr =?) "				

Statement Statistics

Page Reads	Buffer Reads	Reads Cache	Data Buffer Reads	Index Buffer Reads	Page Writes	Buffer Writes	Writes Cache
0	33441	100.00 %	33441	0	0	0	0.00 %
Lock Requests	**# Lock Waits**	**Lock Wait Time (S)**	**Log Space**	**Disk Sorts**	**Memory Sorts**	**Number of Tables**	**Number of Iterators**
25847	0	0	0.000 B	0	0	0	4
Total Executions	**Total Executions Time (S)**	**Average Execution Time (S)**	**Maximum Execution Time (S)**	**Number of IO Wait**	**IO Wait Time (S)**	**Average IO Wait (S))**	**Average Rows/Second**
1	0.43239	0.43239	0.43239	0	0.00000	0.00000	2.31271
Estimated Cost	**Estimated Rows**	**Actual Rows**	**SQL Error**	**ISAM Error**	**Isolation Level**	**SQL Memory**	
1749	1	1	0	0	18	17.9 KB	

SQL Profile

Options Available with Set Explain

- Optimizer Directives – AVOID_EXECUTE
  - Introduced in Informix 9.30
  - Generate query plan without executing SQL, useful for getting query plans for inserts, updates and delete where data is manipulated, but you do not want to change data
  - Example:
    - set explain on AVOID_EXECUTE;
    - SQL Statement

Options Available with Set Explain

- Set Explain Enhancements

  - Another improvement with Informix 11.10 is that you can turn on/off explain statistics thru the onconfig parameter “EXPLAIN_STAT”.
    - 0 – Disables the display of query statistics
    - 1 – Enables the display of query statistics
  - FYI, this is an undocumented feature in Informix 10.
  - You can also set it with the following statement:
    - SET EXPLAIN STATISTICS
  - When this is enabled, the inclusion of the “Query Statistics” section in the explain output file. It shows the query plan’s estimated number of rows and the actual number of rows returned.

Options Available with Set Explain - Query Statistics

```
QUERY:
------
select *
from partsupp
where ps_partkey >= 1 and ps_partkey <= 100 and ps_suppkey >= 0 and ps_suppkey <= 100000 and ps_availqty >= 1000 and ps_availqty <= 1000000

Estimated Cost: 49
Estimated # of Rows Returned: 360

 1) informix.partsupp: INDEX PATH
  (1) Index Keys: ps_partkey ps_suppkey ps_availqty   (Key-First)  (Serial, fragments: ALL)
      Lower Index Filter: informix.partsupp.ps_partkey >= 1 AND (informix.partsupp.ps_availqty >= 1000 ) AND (informix.partsupp.ps_suppkey >= 0 )
      Upper Index Filter: informix.partsupp.ps_partkey <= 100 AND (informix.partsupp.ps_availqty <= 1000000 ) AND (informix.partsupp.ps_suppkey <=100000
        )
      Index Key Filters:  (informix.partsupp.ps_availqty >= 1000 ) AND   (informix.partsupp.ps_availqty <= 1000000 ) AND
                  (informix.partsupp.ps_suppkey <= 100000 ) AND  (informix.partsupp.ps_suppkey >= 0 )

Query statistics:
-----------------
 Table map :
 ----------------------------
 Internal name    Table name
 ----------------------------------------
 t1                   partsupp

 type    table  rows_prod  est_rows  rows_scan    time      est_cost
 ------------------------------------------------------------------------------------------------
 scan    t1          26         360          26         00:00:00      49
```

Dynamic Set Explain

- Dynamically set explain on for a session
  (Introduced in 9.40)
  - Onmode –Y {session id} {0|1} (0 – Off/1 – On)
  - Output is to a file “sqexplain.out.{session id}
  - With Informix 11 there are a couple changes:
    - An additional value “2” (explain without statistics for session, displays query plan only)
    - Also you can specify the file name and directory that you want the explain output to be sent:
      - Onmode –Y {session id} {0|1|2} {filename}

- This is a great feature to allow you to see the SQL statements executed and the explain plan for each SQL statement.
  - **NOTE**: make sure that you only have this turned on for a short period of time, it creates a large file.

Set Explain Output

- Add “set explain on” before the statement you want to examine
- Starting in Informix 11.10 you can specify the directory that you want the file to go:
  - Set explain file to “filename”
- Review the “set explain” output:
  - UNIX: The file “sqexplain.out” will be generated in the directory that you run the query from
  - Windows: Look for a file “username.out” in the directory on the UNIX server %INFORMIXDIR%\sqexpln

Set explain output

- **Query** – Displays the executed query and indicates whether “set optimization” was set to high or low
- **Directives Followed** – Lists any directives used for the query
- **Estimated Cost** – An estimated of the amount of work for the query. The number does not translate into time. Only compare to same query not others.
- **Estimated number of rows returned** – An estimate of the number of rows returned, number based on information from system catalog tables

Set explain output – cont'd

- **Numbered List** – The order in which tables are accessed, followed by the access method (index or sequential scan)
- **Index Keys** – The columns used as filters or indexes
- **Query Statistics** – Enabled by onconfig parameter "EXPLAIN_STAT"

Examples of Explain Plans

- The following slides will show tuning of SQL based on the following scenarios:
  - Functions causing index to not be used
  - Criteria from views causing sequential scans
  - Use of Directives
  - Use of substrings in queries
  - Use of functions in queries
  - Using a better index (Creation of new index)

Function causes index to not be used

```
QUERY:
------
SELECT DISTINCT BUSINESS_UNIT, VOUCHER_ID, INVOICE_ID, GROSS_AMT,
   INVOICE_DT, VENDOR_NAME_SHORT, VENDOR_ID, NAME1, VOUCHER_STYLE,
   ENTRY_STATUS_SRH
FROM PS_VOUCHER_SRCH_VW
WHERE BUSINESS_UNIT='GH'
AND UPPER(INVOICE_ID) LIKE UPPER('KURT') || '%' ESCAPE '\'
ORDER BY INVOICE_ID, BUSINESS_UNIT, VOUCHER_ID DESC FOR READ ONLY


Estimated Cost: 55943
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By

  1) sysadm.ps_vendor: SEQUENTIAL SCAN

  2) sysadm.ps_voucher: INDEX PATH

       Filters: (sysadm.ps_voucher.entry_status IN ('P' , 'R' , 'T' )AND UPPER(sysadm.ps_voucher.invoice_id ) LIKE
       'KURT%' ESCAPE '\' )

    (1) Index Keys: vendor_id vendor_setid business_unit   (Serial, fragments: ALL)
        Lower Index Filter: ((sysadm.ps_voucher.vendor_id = sysadm.ps_vendor.vendor_id AND
        sysadm.ps_voucher.vendor_setid = sysadm.ps_vendor.setid ) AND sysadm.ps_voucher.business_unit = 'GH' )
NESTED LOOP JOIN
```

Resolution: Function causes index to not be used

```
QUERY:
------
SELECT DISTINCT BUSINESS_UNIT, VOUCHER_ID, INVOICE_ID, GROSS_AMT,
   INVOICE_DT, VENDOR_NAME_SHORT, VENDOR_ID, NAME1, VOUCHER_STYLE,
   ENTRY_STATUS_SRH
FROM PS_VOUCHER_SRCH_VW
WHERE BUSINESS_UNIT='GH'
AND INVOICE_ID LIKE 'KURT' || '%' ESCAPE '\'
ORDER BY INVOICE_ID, BUSINESS_UNIT, VOUCHER_ID DESC FOR READ ONLY

Estimated Cost: 35009
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By

  1) sysadm.ps_voucher: INDEX PATH

       Filters: sysadm.ps_voucher.entry_status IN ('P' , 'R' , 'T' )

    (1) Index Keys: business_unit invoice_id   (Serial, fragments: ALL)
       Lower Index Filter: (sysadm.ps_voucher.business_unit = 'GH' AND sysadm.ps_voucher.invoice_id LIKE 'KURT%'
       ESCAPE '\' )

  2) sysadm.ps_vendor: INDEX PATH

    (1) Index Keys: vendor_id setid   (Serial, fragments: ALL)
       Lower Index Filter: (sysadm.ps_voucher.vendor_id = sysadm.ps_vendor.vendor_id AND
       sysadm.ps_voucher.vendor_setid = sysadm.ps_vendor.setid )
NESTED LOOP JOIN
```

Resolution: Function causes index to not be used

- Another way is to add a function which converts a character to all upper case and change the index to include the use of the function.

```
CREATE FUNCTION upper_idx(char_up char(20))
    RETURNING char(20) WITH (not variant);
    DEFINE char_out char(20);
    LET char_out = upper(char_up);
    RETURN char_out;
END FUNCTION;
```

```
CREATE INDEX upper_idx on ps_vendor(business_unit, (upper_idx(invoice_id))
    USING btree;
```

Criteria used to select from view causes sequencial scans

```
QUERY:
------
SELECT BUSINESS_UNIT,INV_ITEM_ID,CM_BOOK,DT_TIMESTAMP,SEQ_NBR,
   CM_DT_TIMESTAMP_A,CM_SEQ_NBR_A,CM_ORIG_TRANS_DATE,CONSIGNED_FLAG,
   STORAGE_AREA,INV_LOT_ID,SERIAL_ID,CM_RECEIPT_QTY,CM_DEPLETE_QTY, CM_ONHAND_QTY
FROM PS_CM_ONHAND_VW
WHERE BUSINESS_UNIT = 'RPRO'
AND INV_ITEM_ID = '05-04-CVC-6-KINS'
AND CM_BOOK = 'FIN'
AND CONSIGNED_FLAG = 'N'
AND CM_ONHAND_QTY > 0
ORDER BY CM_ORIG_TRANS_DATE, CM_DT_TIMESTAMP_A, CM_SEQ_NBR_A FOR READ ONLY

Estimated Cost: 8425
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By  Group By

  1) sysadm.ps_cm_deplete: SEQUENTIAL SCAN

  2) sysadm.ps_cm_receipts: INDEX PATH

    (1) Index Keys: business_unit inv_item_id cm_book dt_timestamp seq_nbr cm_dt_timestamp_a cm_seq_nbr_a   (Serial,
        fragments: ALL)
       Lower Index Filter: ((((((sysadm.ps_cm_receipts.dt_timestamp = sysadm.ps_cm_deplete.cm_dt_timestamp AND
        sysadm.ps_cm_receipts.cm_dt_timestamp_a = sysadm.ps_cm_deplete.cm_dt_timestamp_a ) AND
        sysadm.ps_cm_receipts.inv_item_id = sysadm.ps_cm_deplete.inv_item_id ) AND sysadm.ps_cm_receipts.seq_nbr =
        sysadm.ps_cm_deplete.cm_seq_nbr ) AND sysadm.ps_cm_receipts.cm_seq_nbr_a =
        sysadm.ps_cm_deplete.cm_seq_nbr_a ) AND sysadm.ps_cm_receipts.business_unit =
        sysadm.ps_cm_deplete.business_unit ) AND sysadm.ps_cm_receipts.cm_book = sysadm.ps_cm_deplete.cm_book )
NESTED LOOP JOIN
```

Resolution to Criteria used for view causes sequential scans

```
QUERY:
------
SELECT BUSINESS_UNIT,INV_ITEM_ID,CM_BOOK,DT_TIMESTAMP,SEQ_NBR,
      CM_DT_TIMESTAMP_A,CM_SEQ_NBR_A,CM_ORIG_TRANS_DATE,CONSIGNED_FLAG,
      STORAGE_AREA,INV_LOT_ID,SERIAL_ID,CM_RECEIPT_QTY,CM_DEPLETE_QTY,
      CM_ONHAND_QTY
FROM PS_CM_ONHAND_VW
WHERE BUSINESS_UNIT = 'RPRO'
AND INV_ITEM_ID = '04X35-X-042'
AND CM_BOOK = 'FIN'
AND CONSIGNED_FLAG = 'N'
--AND CM_ONHAND_QTY > 0
ORDER BY CM_ORIG_TRANS_DATE, CM_DT_TIMESTAMP_A, CM_SEQ_NBR_A FOR READ ONLY

Estimated Cost: 10
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By  Group By

  1) sysadm.ps_cm_deplete: INDEX PATH

    (1) Index Keys: business_unit inv_item_id cm_book dt_timestamp seq_nbr cm_dt_timestamp cm_seq_nbr cm_dt_timestamp_a cm_seq_nbr_a   (Serial,
        fragments: ALL)
       Lower Index Filter: ((sysadm.ps_cm_deplete.inv_item_id = '04X35-X-042' AND sysadm.ps_cm_deplete.business_unit = 'RPRO' ) AND
        sysadm.ps_cm_deplete.cm_book = 'FIN' )

  2) sysadm.ps_cm_receipts: INDEX PATH

       Filters: sysadm.ps_cm_receipts.consigned_flag = 'N'

    (1) Index Keys: business_unit inv_item_id cm_book dt_timestamp seq_nbr cm_dt_timestamp_a cm_seq_nbr_a   (Serial, fragments: ALL)
       Lower Index Filter: ((((((sysadm.ps_cm_receipts.inv_item_id = sysadm.ps_cm_deplete.inv_item_id AND sysadm.ps_cm_receipts.dt_timestamp =
        sysadm.ps_cm_deplete.cm_dt_timestamp ) AND sysadm.ps_cm_receipts.seq_nbr = sysadm.ps_cm_deplete.cm_seq_nbr ) AND
        sysadm.ps_cm_receipts.cm_dt_timestamp_a = sysadm.ps_cm_deplete.cm_dt_timestamp_a ) AND sysadm.ps_cm_receipts.cm_seq_nbr_a =
        sysadm.ps_cm_deplete.cm_seq_nbr_a ) AND sysadm.ps_cm_receipts.business_unit = sysadm.ps_cm_deplete.business_unit ) AND
        sysadm.ps_cm_receipts.cm_book = sysadm.ps_cm_deplete.cm_book )
NESTED LOOP JOIN
```

View used in query

```
CREATE VIEW "sysadm".ps_cm_onhand_vw
  (business_unit,inv_item_id,cm_book,dt_timestamp,seq_nbr,cm_dt_timestamp_a, …….

cm_onhand_qty) AS

SELECT x1.business_unit ,x1.inv_item_id ,x1.cm_book ,x1.cm_dt_timestamp, …….

 (x0.qty_base - sum(x1.qty_base) )

FROM "sysadm".ps_cm_receipts x0 ,"sysadm".ps_cm_deplete x1
WHERE (((((((x0.business_unit = x1.business_unit )
AND (x0.inv_item_id = x1.inv_item_id ) )
AND (x0.cm_book = x1.cm_book) )
AND (x0.dt_timestamp = x1.cm_dt_timestamp ) )
AND (x0.seq_nbr = x1.cm_seq_nbr ) )
AND (x0.cm_dt_timestamp_a = x1.cm_dt_timestamp_a) )
AND (x0.cm_seq_nbr_a = x1.cm_seq_nbr_a ) )
GROUP BY x1.business_unit ,x1.inv_item_id ,x1.cm_book ,x1.cm_dt_timestamp ,
        x1.cm_seq_nbr,x0.cm_dt_timestamp_a ,x0.cm_seq_nbr_a ,
        x0.cm_orig_trans_date,x0.consigned_flag ,x0.storage_area ,
        x0.inv_lot_id ,x0.serial_id,x0.qty_base ;
```

Use of directives for Queries

```
QUERY:
------
SELECT D.BUSINESS_UNIT, D.VENDOR_SETID, E.VENDOR_ID, E.NAME1, E.NAME2, VNDR_LOC
FROM PS_PAYMENT_TBL A, PS_PYMNT_VCHR_XREF B, PS_VOUCHER_LINE C,
          PS_VOUCHER D, PS_VENDOR E, PS_VENDOR_LOC F
WHERE A.BANK_SETID = B.BANK_SETID
    AND A.BANK_CD = B.BANK_CD
    AND A.BANK_ACCT_KEY = B.BANK_ACCT_KEY
    AND A.PYMNT_ID = B.PYMNT_ID
    AND B.BUSINESS_UNIT = C.BUSINESS_UNIT
    AND B.VOUCHER_ID = C.VOUCHER_ID
    AND C.BUSINESS_UNIT = D.BUSINESS_UNIT
    AND C.VOUCHER_ID = D.VOUCHER_ID
    AND E.VENDOR_ID = D.VENDOR_ID
    AND A.PYMNT_STATUS = 'P'
    AND A.PYMNT_DT BETWEEN '01-01-2003' AND '12-31-2003'
    AND D.BUSINESS_UNIT IN ('CAT','SNCPY')
    AND E.SETID = F.SETID
    AND E.VENDOR_ID = F.VENDOR_ID
    AND C.WTHD_CD <> F.WTHD_CD

Estimated Cost: 57005
Estimated # of Rows Returned: 1
```

Use of Directive for Queries – cont'd

1) informix.f: INDEX PATH
   Filters: informix.f.effdt = <subquery>
  (1) Index Keys: setid vendor_id vndr_loc effdt (desc) eff_status (Serial, fragments: ALL)
2) informix.e: INDEX PATH
  (1) Index Keys: vendor_id setid (Serial, fragments: ALL)
   Lower Index Filter: (informix.e.vendor_id = informix.f.vendor_id AND informix.e.setid = informix.f.setid)
   NESTED LOOP JOIN
3) informix.d: INDEX PATH
   Filters: informix.d.business_unit IN ('CAT' , 'SNCPY')
  (1) Index Keys: vendor_id vendor_setid entry_status (Serial, fragments: ALL)
   Lower Index Filter: informix.d.vendor_id = informix.f.vendor_id NESTED LOOP JOIN
4) informix.c: INDEX PATH
   Filters: informix.c.wthd_cd != informix.f.wthd_cd
  (1) Index Keys: business_unit voucher_id (desc) voucher_line_num (Serial, fragments: ALL)
   Lower Index Filter: (informix.c.voucher_id = informix.d.voucher_id AND informix.c.business_unit =
   informix.d.business_unit) NESTED LOOP JOIN
5) informix.b: INDEX PATH
  (1) Index Keys: business_unit voucher_id (desc) pymnt_id bank_cd bank_acct_key (Serial, fragments: ALL)
   Lower Index Filter: (informix.b.voucher_id = informix.c.voucher_id AND informix.b.business_unit =
   informix.d.business_unit)
NESTED LOOP JOIN
6) informix.a: INDEX PATH
   Filters: ((informix.a.pymnt_dt >= 01/01/2003 AND informix.a.pymnt_status = 'P') AND informix.a.pymnt_dt <=
   12/31/2003)

  (1) Index Keys: pymnt_id (desc) bank_acct_key bank_cd bank_setid (Serial, fragments: ALL)
   Lower Index Filter: (((informix.a.pymnt_id = informix.b.pymnt_id AND informix.a.bank_acct_key =
   informix.b.bank_acct_key) AND informix.a.bank_cd = informix.b.bank_cd) AND informix.a.bank_setid =
   informix.b.bank_setid) NESTED LOOP JOIN

Use of Directive for Queries – cont'd

```
QUERY:
------
SELECT --+ORDERED
D.BUSINESS_UNIT, D.VENDOR_SETID, E.VENDOR_ID, E.NAME1, E.NAME2,   B.VNDR_LOC
FROM PS_PAYMENT_TBL A, PS_PYMNT_VCHR_XREF B, PS_VOUCHER_LINE C,
    PS_VOUCHER D, PS_VENDOR E, PS_VENDOR_LOC F
WHERE A.BANK_SETID = B.BANK_SETID
    AND A.BANK_CD = B.BANK_CD
    AND A.BANK_ACCT_KEY = B.BANK_ACCT_KEY
    AND A.PYMNT_ID = B.PYMNT_ID
    AND B.BUSINESS_UNIT = C.BUSINESS_UNIT
    AND B.VOUCHER_ID = C.VOUCHER_ID
    AND C.BUSINESS_UNIT = D.BUSINESS_UNIT
    AND C.VOUCHER_ID = D.VOUCHER_ID
    AND E.VENDOR_ID = D.VENDOR_ID
    AND A.PYMNT_STATUS = 'P'
    AND A.PYMNT_DT BETWEEN '01-01-2003' AND '12-31-2003'
    AND D.BUSINESS_UNIT IN ('CAT','SNCPY')
    AND E.SETID = F.SETID
    AND E.VENDOR_ID = F.VENDOR_ID
    AND C.WTHD_CD <> F.WTHD_CD

DIRECTIVES FOLLOWED:
ORDERED
DIRECTIVES NOT FOLLOWED:

Estimated Cost: 70888                    (Cost of Original Query: 57005)
Estimated # of Rows Returned: 1
```

Use of Directive for Queries – cont'd

1) informix.a: INDEX PATH
    Filters: informix.a.pymnt_status = 'P'
  (1) Index Keys: pymnt_dt name1 remit_setid currency_pymnt (Serial, fragments: ALL)
    Lower Index Filter: informix.a.pymnt_dt >= 01/01/2003
    Upper Index Filter: informix.a.pymnt_dt <= 12/31/2003
2) informix.b: INDEX PATH
    Filters: informix.b.business_unit IN ('CAT' , 'SNCPY')
  (1) Index Keys: bank_setid bank_cd bank_acct_key pymnt_id (Serial, fragments: ALL)
    Lower Index Filter: (((informix.a.pymnt_id = informix.b.pymnt_id AND informix.a.bank_acct_key = informix.b.bank_acct_key) AND informix.a.bank_cd = informix.b.bank_cd) AND informix.a.bank_setid = informix.b.bank_setid) NESTED LOOP JOIN
3) informix.c: INDEX PATH
  (1) Index Keys: business_unit voucher_id (desc) voucher_line_num (Serial, fragments: ALL)
    Lower Index Filter: (informix.b.voucher_id = informix.c.voucher_id AND informix.b.business_unit = informix.c.business_unit) NESTED LOOP JOIN
4) informix.d: INDEX PATH
  (1) Index Keys: voucher_id (desc) business_unit invoice_id (Serial, fragments: ALL)
    Lower Index Filter: (informix.b.voucher_id = informix.d.voucher_id AND informix.b.business_unit = informix.d.business_unit) NESTED LOOP JOIN
5) informix.e: INDEX PATH
  (1) Index Keys: vendor_id setid (Serial, fragments: ALL)
    Lower Index Filter: informix.e.vendor_id = informix.d.vendor_id NESTED LOOP JOIN
6) informix.f: INDEX PATH
    Filters: (informix.c.wthd_cd != informix.f.wthd_cd AND informix.f.effdt = <subquery>)
  (1) Index Keys: setid vendor_id vndr_loc effdt (desc) eff_status (Serial, fragments: ALL)
    Lower Index Filter: (informix.e.vendor_id = informix.f.vendor_id AND informix.e.setid = informix.f.setid) NESTED LOOP JOIN

Use of substrings (Best index not being used)

```
QUERY:
------
SELECT ACLNL.MONETARY_AMOUNT
FROM PS_CM_ACCTG_LINE ACLNL
WHERE ACLNL.BUSINESS_UNIT = ‘ABCDE'
AND ACLNL.PRODUCTION_ID = '12334'
AND SUBSTR(ACLNL.ACCOUNT,1,3) IN ( '085' , '334', '072' )

Estimated Cost: 49722
Estimated # of Rows Returned: 1

  1) informix.aclnl: INDEX PATH

      Filters: (informix.aclnl.production_id = '12334' AND SUBSTR
     (informix.aclnl.account , 1 , 3 ) IN ('085' , '334' , '072' ))

    (1) Index Keys: business_unit cm_book gl_distrib_status budget_hdr_status
      cm_iu_status   (Serial, fragments: ALL)
       Lower Index Filter: informix.aclnl.business_unit = ‘ABCDE'
```

Resolution for Use of substrings

QUERY:

SELECT ACLNL.MONETARY_AMOUNT
FROM PS_CM_ACCTG_LINE ACLNL
WHERE ACLNL.BUSINESS_UNIT = ‘ABCDE'
AND ACLNL.PRODUCTION_ID = '12334'
AND (ACLNL.ACCOUNT matches '085*'
OR ACLNL.ACCOUNT matches '334*'
OR ACLNL.ACCOUNT matches '072*')

Estimated Cost: 3
Estimated # of Rows Returned: 1

1) informix.aclnl: INDEX PATH

(1) Index Keys: business_unit production_id account (Key-First) (Serial, fragments: ALL)
Lower Index Filter: (informix.aclnl.production_id = '12334' AND informix.aclnl.business_unit = ‘ABCDE')
Key-First Filters: (((informix.aclnl.account MATCHES '085*' OR informix.aclnl.account MATCHES '334*') OR informix.aclnl.account MATCHES '072*'))

Use of Functions in Queries cause specific Indexes not be used

QUERY:

```
SELECT od.order_id AS order_id,od.club_model_id AS club_model_id,od.purchase_type_cd
    AS purchase_type_cd,od.order_status_cd AS order_status_cd,
    EXTEND(od.create_ts, YEAR TO DAY) AS create_ts,price AS price,
    shipping_amt AS shipping_amt,od.session_id AS session_id
FROM order_detail od, order_header oh
WHERE oh.source_id != -1
AND oh.source_id IS NOT NULL
AND oh.source_id != 23150010

AND EXTEND(od.create_ts, YEAR TO DAY) = '2004-05-17'

AND od.order_id = oh.order_id
AND club_model_id = 10
AND (purchase_type_cd = 'CLUB' OR purchase_type_cd = 'SEYMOS'
OR purchase_type_cd = 'DCSSORC' OR purchase_type_cd = 'SHVSSORC'
OR purchase_type_cd = 'DDVSSORC' OR purchase_type_cd = 'HSACNUF')
```

Estimated Cost: 546168
Estimated # of Rows Returned: 67774

Use of Functions in Queries causes specific Indexes not be used

```
1) informix.od: INDEX PATH
   Filters: (EXTEND (informix.od.create_ts ,year to day) =  datetime(2004-05-17) year to day

                AND (((((informix.od.purchase_type_cd = 'CLUB'
                OR informix.od.purchase_type_cd = 'SEYMOS' )
                OR informix.od.purchase_type_cd = 'DCSSORC' )
                OR informix.od.purchase_type_cd = 'SHVSSORC' )
                OR informix.od.purchase_type_cd = 'DDVSSORC' )
                OR informix.od.purchase_type_cd = 'HSACNUF' ) )

   (1) Index Keys: club_model_id   (Serial, fragments: ALL)
       Lower Index Filter: informix.od.club_model_id = 10

2) informix.oh: INDEX PATH
   Filters: (informix.oh.source_id != -1 AND (informix.oh.source_id IS NOT NULL
       AND informix.oh.source_id != 23150010 ) )

   (1) Index Keys: order_id   (Serial, fragments: ALL)
       Lower Index Filter: informix.oh.order_id = informix.od.order_id
NESTED LOOP JOIN
```

Resolution to Use of Functions in Queries

```
QUERY:
------
SELECT od.order_id AS order_id,od.club_model_id AS club_model_id,
      od.purchase_type_cd AS purchase_type_cd,od.order_status_cd AS order_status_cd,
      EXTEND(od.create_ts, YEAR TO DAY) AS create_ts,price AS price, shipping_amt
      ASshipping_amt,od.session_id AS session_id
FROM order_detail od, order_header oh
WHERE oh.source_id != -1
AND oh.source_id IS NOT NULL
AND oh.source_id != 23150010

AND (od.create_ts >= '2004-05-17 00:00:00.000‘ AND od.create_ts <= '2004-05-17 23:59:59.999')

AND od.order_id = oh.order_id
AND club_model_id = 10
AND (purchase_type_cd = 'CLUB' OR purchase_type_cd = 'SEYMOS'
OR purchase_type_cd = 'DCSSORC' OR purchase_type_cd = 'SHVSSORC'
OR purchase_type_cd = 'DDVSSORC' OR purchase_type_cd = 'HSACNUF')

Estimated Cost: 2                    (Original Query Cost: 546168)
Estimated # of Rows Returned: 1
```

Resolution to Use of Functions in Queries

1) informix.od: INDEX PATH

(1) Index Keys: create_ts purchase_type_cd order_status_cd club_model_id (Key-First) (Serial, fragments: ALL)
Lower Index Filter: informix.od.create_ts >= datetime(2004-05-17 00:00:00.000) year to fraction(3)
Upper Index Filter: informix.od.create_ts <= datetime(2004-05-17 23:59:59.999) year to fraction(3)
Key-First Filters: ((((((informix.od.purchase_type_cd = 'CLUB'
OR informix.od.purchase_type_cd = 'SEYMOS')
OR informix.od.purchase_type_cd = 'DCSSORC')
OR informix.od.purchase_type_cd = 'SHVSSORC')
OR informix.od.purchase_type_cd = 'DDVSSORC')
OR informix.od.purchase_type_cd = 'HSACNUF'))
AND (informix.od.club_model_id = 10)

2) informix.oh: INDEX PATH

Filters: (informix.oh.source_id != -1 AND (informix.oh.source_id IS NOT NULL AND informix.oh.source_id != 23150010))

(1) Index Keys: order_id (Serial, fragments: ALL)
Lower Index Filter: informix.oh.order_id = informix.od.order_id
NESTED LOOP JOIN

Using a better index

QUERY:
------
select club_model_id, order_status_cd, count(distinct order_id) as
order_count
from order_detail
where create_ts >= '2004-09-30 00:00:00.000‘ and create_ts < '2004-10-02 00:00:00.000'
and purchase_type_cd = 'CASH'
and order_status_cd not in ('REJ', ‘ACCP')
group by 1,2
order by 1,2

Estimated Cost: 407
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By Group By

1) informix.order_detail: INDEX PATH

Filters: (informix.order_detail.create_ts >= datetime(2004-09-30 00:00:00.000) year to fraction(3) AND
(informix.order_detail.create_ts < datetime(2004-10-02 00:00:00.000) year to fraction(3) AND
informix.order_detail.order_status_cd NOT IN (‘REJ' , ‘ACCP')))

(1) **Index Keys: purchase_type_cd** (Serial, fragments: ALL)
Lower Index Filter: informix.order_detail.purchase_type_cd = 'CASH'

Resolution to Using a better index

QUERY: (AFTER ADDING A NEW INDEX)
------
select club_model_id, order_status_cd, count(distinct order_id) as
order_count
from order_detail
where create_ts >= '2004-09-30 00:00:00.000'
and create_ts < '2004-10-02 00:00:00.000'
and purchase_type_cd = 'CASH'
and order_status_cd not in ('REJ', 'ACCP')
group by 1,2
order by 1,2

Estimated Cost: 3
Estimated # of Rows Returned: 1
Temporary Files Required For: Order By Group By

1) informix.order_detail: INDEX PATH

(1) Index Keys: create_ts purchase_type_cd order_status_cd club_model_id (Key-First) (Serial, fragments: ALL)
Lower Index Filter: informix.order_detail.create_ts >= datetime(2004-09-30 00:00:00.000) year to fraction(3)
Upper Index Filter: informix.order_detail.create_ts < datetime(2004-10-02 00:00:00.000) year to fraction(3)
Key-First Filters: (informix.order_detail.order_status_cd NOT IN ('REJ' , 'ACCP')) AND
(informix.order_detail.purchase_type_cd = 'CASH'

Methods to use for Improving SQL Performance

- Utilize “UNIONS” when you have “OR” in where clause
- Utilize temp tables in optimizing queries by splitting the query into multiple queries
- Utilize PDQPRIORITY
- Utilize DS_NONPDQ_QUERY_MEM (V 9.40/10.00)
- Fragment tables (Understand the use of the data) to eliminate fragments from selection of the data
- Utilize external directives (V 10.00)

Utilize Unions

```
SELECT a.email_template_id, b.description, a.club_model_id, a.email_log_id, efd.field_id
FROM email_log a, email_template b, email_field_data efd
WHERE a.email_template_id = b.email_template_id
AND a.email_template_id = efd.email_template_id
AND efd.email_log_id = a.email_log_id
AND efd.field_id in (561, 558)
AND a.club_model_id in ('1', '2')
AND a.email_template_id in ('275','128')
UNION
SELECT a.email_template_id, b.description, a.club_model_id, 0 as email_log_id, 0 as field_id
FROM email_log a, email_template b
WHERE a.email_template_id = b.email_template_id
AND a.club_model_id in ('1', '2')
AND a.email_template_id in ('125','2171')
UNION
SELECT a.email_template_id, b.description, a.club_model_id, 1 as email_log_id, 1 as field_id
FROM email_log a, email_template b
WHERE a.email_template_id = b.email_template_id
AND a.club_model_id in (‘3', ‘4')
AND a.email_template_id = '2152';
```

Utilize Temp Tables

```
SET PDQPRIORITY 100;
SELECT acct_n, gender
FROM v_master
WHERE acct_n MATCHES '90*' AND mbr_phase_cde IN ('E','F','M','R')
INTO TEMP tmp_v_master WITH NO LOG;
```

NOTE: With V11.10, you can globally set the onconfig parameter "TEMPTAB_NOLOG"
0 – Off (Enable logical logging on temp tables)
1 – On (Disable logical logging on temp tables)

```
CREATE INDEX idx_vidmaster ON tmp_v_master(acct_n);
UPDATE STATISTICS LOW FOR TABLE tmp_v_master;
```

NOTE: With V11.10 you no longer need to run update statistics on a temp table

```
SELECT pull, equip, type_equip
FROM cat_pull
WHERE equip in ('S','W','T','M')
INTO TEMP temp_cat_pull WITH NO LOG;

CREATE INDEX idx_cat_pull ON temp_cat_pull(pull, equip);
UPDATE STATISTICS LOW FOR TABLE temp_cat_pull;

SELECT --+ORDERED
      account, m.gender, c.type_equip,
FROM v_trans v, tmp_v_master m, temp_cat_pull c
WHERE cntrl_num >= 118265 AND cntrl_num < 118786
AND m.acct_n = v.account
AND (v.selection = c.pulland v.equip = c.equip)
AND (uimm > 0 OR upos > 0 OR udis > 0 OR ubon > 0 OR udoc > 0 OR ugaf > 0
       OR uxdoc > 0 OR ues > 0 OR ufso > 0 OR urain > 0 OR ufree > 0)
INTO TEMP tst WITH NO LOG;
```

Utilize PDQPRIORITY

```
SET PDQPRIORITY 100;

SELECT accnt, pc, cntrl_wd, mfree, uauto, salestype
FROM vid_tran
WHERE substr(accnt,11,1) = '7'
AND (magz <> '' OR magz IS NOT NULL)
AND (pc LIKE 'BV1%' OR pc LIKE 'DA2%'
        OR pc LIKE 'DVM%' )
AND (cntrl_wd BETWEEN 118530 AND 119335)
```

Estimated Cost: 2485223
Estimated # of Rows Returned: 6067559

Maximum Threads: 3

Utilize DS_NONPDQ_QUERY_MEM

DS_NONPDQ_QUERY_MEM = 50,000

```
session                                #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
324499   indprod  -        27952    prodtu   1        2367488    2328592    off

tid      name     rstcb            flags    curstk   status
25339601 sqlexec  c0000002b2c9ac18 ---PR--  1842583336 sleeping(Forever)

Memory pools    count 2
name         class addr             totalsize  freesize   #allocfrag #freefrag
324499       V     c0000002b60be040 2306048    34848      4315       157
324499_SORT  V     c0000002b59a3040 61440      4048       7          1

name          free       used          name          free       used
sort          0          34144         sqscb         0          41344
sql           0          80            srtmembuf     0          20384

sqscb info
scb              sqscb            optofc   pdqpriority sqlstats optcompind  directives
c0000002b5be61d0 c0000002cb9d7030 0        0           0        0           1

Sess  SQL           Current           Iso Lock       SQL  ISAM F.E.
Id    Stmt type     Database          Lvl Mode       ERR  ERR  Vers Explain
324499 SELECT       elstest           CR  Wait 600   0    0    9.03 Off

Current SQL statement :
  select  unique b.* from tmp_fids a, name_init b where a.fid = b.fid
```

Utilize DS_NONPDQ_QUERY_MEM

DS_NONPDQ_QUERY_MEM 500,000

```
session                                  #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
16354    vijays   -        16777    prodtu   1        2490368    2458128    off

Memory pools    count 2
name         class addr             totalsize  freesize   #allocfrag #freefrag
16354        V     c0000001a12a4040 2244608    27536      4211       82
16354_SORT_  V     c0000001b3df5040 245760     4704       7          2

name          free      used          name          free      used
sort          0         34128         sqscb         0         56080
sql           0         80            srtmembuf     0         204064          (vs 20384 with 50,000 DS_NONPDQ_QUERY_MEM)


sqscb info
scb              sqscb            optofc   pdqpriority sqlstats optcompind  directives
c0000001ad54a8c0 c0000001a12af030 0        0           0        0           1

Sess  SQL            Current            Iso Lock       SQL  ISAM F.E.
Id    Stmt type      Database           Lvl Mode       ERR  ERR  Vers Explain
16354 SELECT         elstest            CR  Wait 600   0    0    9.03 Off


Current SQL statement :
  select  unique b.* from tmp_fids a, name_init b where a.fid = b.fid and      a.fid > 0
```

Fragment Tables

```
SELECT UNIQUE fid, serial_num, total_nos
FROM addridx b
WHERE name  = “JOHN"
   AND b.state = 27
   AND value IN (12345, 98765)
   AND total_nos = 1;

Estimated Cost: 1
Estimated # of Rows Returned: 1

  1) informix.b: INDEX PATH
    (1) Index Keys: state value name total_nos serial_num fid
                (Key-Only)  (Serial, fragments: 26)
        Lower Index Filter: (((informix.b.name = ‘JOHN'  AND informix.b.value = 12345 )
      AND informix.b.state = 27) AND informix.b.total_nos = 1 )

    (2) Index Keys: state value name total_nos serial_num fid
                (Key-Only)  (Serial, fragments: 26)
        Lower Index Filter: (((informix.b.name = ‘JOHN' AND informix.b.value = 98765 )
      AND informix.b.state = 27) AND informix.b.total_nos = 1 )
```

Using External Directives

- External Directives allow you to use directives on SQL statements that cannot be changed.
  - For example, you have an application that you cannot changed the SQL statements in it, but are having an issue with the performance of a specific SQL statement. With the use of external directives, you can override the SQL statement by forcing it to use directives.

Using External Directives – cont'd

NOTE CAUTION:

- The purpose of external directives is to improve the performance of queries that match the *query* string.
- The use of such directives can potentially slow other queries, if the query optimizer must compare the *query* strings of a large number of active external directives with the text of every SELECT statement.
- For this reason, it is recommended that the DBA not allow the **sysdirectives** table to accumulate more than a few ACTIVE rows.

Use of External Directives – cont'd

- Syntax:

```
SAVE EXTERNAL DIRECTIVES /*+ AVOID_INDEX
 (table1 index1)*/, /*+ FULL(table1) */
        ACTIVE FOR
        SELECT /*+ INDEX( table1 index1 ) */ col1, col2
        FROM table1, table2
        WHERE table1.col1 = table2.col1
```

Use of External Directives – cont'd

- How do we enable the use of External Directives?
  - ONCONFIG:
    - EX_DIRECTIVES (0 – OFF, 1 – ON, 2 – ON)
  - Individual sessions external directives can be enabled with the following, all other combinations will have external directives OFF:
    - IFX_EXTDIRECTIVES
      - NOT SET/EX_DIRECTIVES = 2
      - 1 / EX_DIRECTIVES = 1 or 2
      - 0 "NO" External directives no matter what EX_DIRECTIVES is set to

Know Your Applications

- One of the biggest things that I see is that DBA's do not understand the business of the systems they are supporting to effectively support the systems.
- The most important item is that you understand the business of the system that you are trying to support and tune.
- Get involved early in the design, work with the developers in designing the systems.

Know your Application – cont'd

ISSUE

- Client where DBA's did not work with development.
- Developers designed tables.
- DBA's just maintained production.

RESULTS

- Poor Performance.
- DBA's did not understand how the application worked.
- There was finger pointing on who's problem it was, no accountability.

Know your Application – cont'd

RECOMENDATIONS

- Involved the DBA's to work with the developers from the beginning of the projects.
- Tables created jointly between developers and DBA's during development.

RESULTS

- DBA's now had a handle on enforcing what was approved for production.
- The number of issues that occurred after a project launch were reduced dramatically.

Know your Applications – cont'd

Key Points

- Be involved early in the process.
- Go to development meetings, understand current/upcoming projects and how they impact the system.
- Create a data model of the database. Understand relationship of tables.
- Identify potential tables that could have scheduled archiving of data. Reduce the amount of data that needs to be searched.
- Mentor developers on coding tips for efficient SQL programming. Host "Lunch & Learn" sessions to teach developers on best practices for SQL coding.

Case Studies – Case 1

Case Study 1

- Review the whole business logic, do not just review the specific SQL statements.

- Example:
  - In reviewing a client's performance issue, I saw that the specific SQL statement was written correctly.
  - The issues were the following:
    - Two tables had 20 times more reads than any other table
    - 87 extents on one table, 98 extents on the other

Case Studies – Case 1

SOLUTION

- Reorganize the tables into single extent.
- Purged data that was no longer needed, which reduced table size by 70%.
- Created monthly job to purge records that were not needed.
- Rebuilt indexes that were last created 5 years ago.
- Added new indexes for improved selection.

Case Studies – Case 1

RESULTS

- Performance Improved Dramatically!!!!!
- Number of buffer reads on the two tables were reduced. They went from being the top two tables in number of reads by 20 times the next table to not even in the top 5.

Case Studies – Case 2

Case Study 2

- Development was changing a process.
- How can DBA's help in development?
  - Probe them on how the new process will work.
  - Investigate how to improve the process further.

Case Studies – Case 2

- During analysis of the process that was changing, take a step back and look at the whole process, not just the piece that is changing.
- Review how the change may help or hurt performance and what other changes may need to be made.
- Review other areas in the application where data is being selected and see if there are improvements that can be made.

Case Studies – Case 2

- The change was to write the data to the cart and cart_item table in real time instead of batch mode.
- After looking at the tables, I wondered why would the cart table have more records than the child table cart_items?
- I questioned the developers, should a cart exist with no items, the answer was no there should not be.

Case Studies – Case 2

After reviewing the new process, I found some improvements to be made.

- Reduced one table (CART) row count by 90% from 85 million to 4 million by purging cart records with no items.

Implementation:

- Since the system was 24X7, I could not take an outage long enough to rebuild the CART table which would have been optimal.
- I ended up writing a stored procedure to delete the records and commit after every 100 deletes to keep the transactions small and to not cause any locking issues for the users on the system.
- Rebuild indexes after purging the data.

Case Studies – Case 3

Client was having performance issues when load on the system increased.

- Investigated the index with the highest buffer reads.
- After reviewing the SQL statements being processed using that index, I identified that it was not the best index on the table being used for the query.

Case Studies – Case 3

Resolution

- Ran update statistics high on a column in the index which previously had "medium" statistics on it.
- Dropped an index that was a duplicate of another index.

RESULTS

- Total number of buffer reads on the system decreased by 50%.
- During their peak times, there were no performance issues, the system performed flawlessly.

Summary

- Identify Problem SQL Statements
- Tracing SQL in Informix 11
- Options Available with Set Explain & Reading sqexplain output with tuning examples
- Methods to use for Improving SQL performance
- Know your Application in Tuning SQL
- Case Studies

QUESTIONS

Informix SQL Performance Tuning Tips
Session: B12

Jeff Filippi

Integrated Data Consulting, LLC
jeff.filippi@itdataconsulting.com
www.itdataconsulting.com
630-842-3608