Informix SQL Performance Tuning Tips

Jeff Filippi
Integrated Data Consulting, LLC
jeff.filippi@itdataconsulting.com

Session: B01
Monday April 28 – 10 am
Introduction

• 24 years of working with Informix products
• 20 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, 5, 7, 9, 10, 11.10, 11.50, 11.70, 12.10
Overview

• Identify Problem SQL Statements

• Tracing SQL in Informix

• Options Available with Set Explain & Reading sqexplain output with tuning examples

• Methods to use for Improving SQL Performance

• Subquery Support for Update/Delete

• External Tables

• Informix 12 SQL Features
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, OAT and 3rd party tools like Server Studio.
  – Look at how many times SQL statements have been executed using SQL Statement Cache (onstat –g ssc)
  – Informix 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 11.70.FC7 -- 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:

<table>
<thead>
<tr>
<th>lru</th>
<th>hash</th>
<th>ref_cnt</th>
<th>hits</th>
<th>flag</th>
<th>heap_ptr</th>
<th>database</th>
<th>user</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>140</td>
<td>0</td>
<td>15</td>
<td>-F</td>
<td>bb164020</td>
<td>ntlcom</td>
<td>informix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>116</td>
<td>0</td>
<td>1011</td>
<td>-F</td>
<td>aa23bc20</td>
<td>ntlcom</td>
<td>informix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>207</td>
<td>0</td>
<td>6003</td>
<td>-F</td>
<td>a004f820</td>
<td>ntlcom</td>
<td>informix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>138</td>
<td>0</td>
<td>6244</td>
<td>-F</td>
<td>afa9ec20</td>
<td>ntlcom</td>
<td>informix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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, sequential scans.

  – SELECT tabname[1,25], bufreads, pagreads, isreads, seqscans
    FROM sysmaster:sysptprof
    ORDER BY 2 desc
Example - Reads

<table>
<thead>
<tr>
<th>tabname</th>
<th>bufreads</th>
<th>pagreads</th>
<th>isreads</th>
</tr>
</thead>
<tbody>
<tr>
<td>trnsit_1</td>
<td>-2122091061</td>
<td>429</td>
<td>1630786736</td>
</tr>
<tr>
<td>trnsit_1</td>
<td>-812314372</td>
<td>3162</td>
<td>-678524115</td>
</tr>
<tr>
<td>trnsit_1</td>
<td>-110705308</td>
<td>233</td>
<td>-390409810</td>
</tr>
<tr>
<td>im_p_route_1</td>
<td>1806427782</td>
<td>247</td>
<td>865918944</td>
</tr>
<tr>
<td>ed_rl_event</td>
<td>1749246222</td>
<td>23550</td>
<td>1709746386</td>
</tr>
<tr>
<td>loc_sup_data</td>
<td>1479941490</td>
<td>39</td>
<td>1108625557</td>
</tr>
<tr>
<td>ed_rl_event_3</td>
<td>1186682507</td>
<td>2668713</td>
<td>789739458</td>
</tr>
<tr>
<td>460_4902</td>
<td>1125173161</td>
<td>1003575</td>
<td>373042018</td>
</tr>
<tr>
<td>ed_rl_event_4</td>
<td>893520660</td>
<td>25725</td>
<td>886704767</td>
</tr>
<tr>
<td>im_mv_event</td>
<td>870108889</td>
<td>30477208</td>
<td>780365364</td>
</tr>
</tbody>
</table>
# New Index Added

<table>
<thead>
<tr>
<th>tabname</th>
<th>bufreads</th>
<th>pagreads</th>
<th>isreads</th>
</tr>
</thead>
<tbody>
<tr>
<td>140_409</td>
<td>-845911921</td>
<td>0</td>
<td>1722690950</td>
</tr>
<tr>
<td>cntrct_num</td>
<td>1297728722</td>
<td>1</td>
<td>2868878</td>
</tr>
<tr>
<td>221_1360</td>
<td>812752007</td>
<td>0</td>
<td>406226191</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>......................</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trnsit</td>
<td>17215024</td>
<td>22</td>
<td>12007375</td>
</tr>
<tr>
<td>trnsit_ix1</td>
<td>15638629</td>
<td>106</td>
<td>12898627</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>im_p_route_1</td>
<td>23045</td>
<td>347</td>
<td>12904</td>
</tr>
</tbody>
</table>
Tracing SQL in Informix

• 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)
  – Ability to save trace buffer (11.50XC4)
  – Open Admin Tool (OAT)
Tracing SQL in Informix

• There are a couple ways to turn tracing on in Informix
  – 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

• 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}
    • SET SQL TRACING DATABASE CLEAR
    • SET SQL TRACING DATABASE LIST
    • SET SQL TRACING DATABASE REMOVE {Database}
  – You can also suspend and resume all tracing at the server without de-allocating any resources.
    • SET SQL TRACING SUSPEND/RESUME
Tracing SQL in Informix

• 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

`onstat -g his`

IBM Informix Dynamic Server Version 11.70.FC7   -- 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

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

### Statement Statistics:

<table>
<thead>
<tr>
<th>Page</th>
<th>Buffer</th>
<th>Read</th>
<th>% Cache</th>
<th>IDX Read</th>
<th>Write</th>
<th>Write</th>
<th>% Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17</td>
<td>100.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lock</th>
<th>Lock</th>
<th>LK Wait</th>
<th>Log</th>
<th>Num</th>
<th>Disk</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>Waits</td>
<td>Time (S)</td>
<td>Space</td>
<td>Sorts</td>
<td>Sorts</td>
<td>Sorts</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0.0000</td>
<td>0.000 B</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total</th>
<th>Total</th>
<th>Avg</th>
<th>Max</th>
<th>Avg</th>
<th>I/O Wait</th>
<th>Avg Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executions</td>
<td>Time (S)</td>
<td>Time (S)</td>
<td>Time (S)</td>
<td>IO Wait</td>
<td>Time (S)</td>
<td>Per Sec</td>
</tr>
<tr>
<td>7294</td>
<td>6.6040</td>
<td>0.0009</td>
<td>0.0015</td>
<td>0.000000</td>
<td>0.000000</td>
<td>10869.5652</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimated</th>
<th>Estimated</th>
<th>Actual</th>
<th>SQL</th>
<th>ISAM</th>
<th>Isolation</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Rows</td>
<td>,</td>
<td>Rows</td>
<td>Error</td>
<td>Error</td>
<td>Level</td>
</tr>
<tr>
<td>26</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LC</td>
</tr>
</tbody>
</table>
Tracing in Informix

• You can also get information on the tracing thru the “syssqltrace” table in the sysmaster database.
  – Ex. (# of queries that ran > 2 seconds)
    
    ```sql
    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.
Tracing in Informix

• You can run the following SQL statement to get the SQL for the ones that have a higher run time.

```sql
select sql_runtime, sql_statement
from sysmaster:syssqltrace
where sql_runtime > .5
order by 1 desc
```
Tracing in Informix

- Here is what the “syssqltrace” table looks like:

  sql_id          9894804
  sql_address     13746521704
  sql_sid         26646
  sql_uid         668
  sql_stmttype    2
  sql_stmtname    SELECT
  sql_finishtime  1396011341
  sql_begintxtime 301590260
  sql_runtime     4.7849315e-05
  sql_pgreads     0
  sql_bfreads     5
  sql_rdcache     100.0000000000
  sql_bfidxreads  0
### Tracing in Informix

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sql_pgwrites</td>
<td>0</td>
</tr>
<tr>
<td>sql_bfwrites</td>
<td>0</td>
</tr>
<tr>
<td>sql_wrcache</td>
<td>0.00</td>
</tr>
<tr>
<td>sql_lockreq</td>
<td>1</td>
</tr>
<tr>
<td>sql_lockwaits</td>
<td>0</td>
</tr>
<tr>
<td>sql_lockwttime</td>
<td>0.00</td>
</tr>
<tr>
<td>sql_logspace</td>
<td>0</td>
</tr>
<tr>
<td>sql_sorttotal</td>
<td>0</td>
</tr>
<tr>
<td>sql_sortdisk</td>
<td>0</td>
</tr>
<tr>
<td>sql_sortmem</td>
<td>0</td>
</tr>
<tr>
<td>sql_executions</td>
<td>1</td>
</tr>
<tr>
<td>sql_totaltime</td>
<td>4.7849315e-05</td>
</tr>
<tr>
<td>sql_avgtime</td>
<td>4.7849315e-05</td>
</tr>
<tr>
<td>sql_maxtime</td>
<td>4.7849315e-05</td>
</tr>
<tr>
<td>sql_numiowaits</td>
<td>0</td>
</tr>
<tr>
<td>sql_avgiowaits</td>
<td>0.00</td>
</tr>
<tr>
<td>sql_totaliowaits</td>
<td>0.00</td>
</tr>
<tr>
<td>sql_rowspersec</td>
<td>20898.94078138</td>
</tr>
</tbody>
</table>
Tracing in Informix

sql_estcost       23
sql_estrows       46
sql_actualrows    1
sql_sqlerror      0
sql_isamerror     0
sql_isollevel     1
sql_sqlmemory     24200
sql_numiterators  1
sql_database      ir_live2
sql_numtables     4
sql_tablelist     systables
sql_statement     select owner,tabname,tabtype,tabid from informix.systables
sql_stmtlen       117
sql_stmthash      206750675
sql_pdq           0
sql_num_hvars     0
sql_dbspartnum    10488544
sql_aqt           None
sql_aqtinfo       -26500
Tracing in Informix

- Starting in 11.50XC4 you can now save the history of the SQL tracing information. **NOTE**: Caution when using this, it can use a lot of space very quickly. I had a customer fill a 20 gig dbspace in 24 hours.

- In the Scheduler there is a new task “Save SQL Trace”.

- Information is saved in the following tables in the sysadmin database:
  - `mon_syssqltrace` (SQL Statement text and profile info)
  - `mon_syssqltrace_info` (SQL Statement tracing setup info)
  - `mon_sqltrace_iter` (SQL Statement iterators)
Tracing in Informix

• Here is an alternative to saving ALL tracing information.

• Create a task which saves SQL trace information for SQL’s that have a run time of greater than 10 seconds.

• This allows you to still trace SQL statements, but only show you the really long running SQL statements
Tracing in Informix

• Create a Table to save the SQL Trace information
• Create a new dbspace to put the table in so that if it does fill up a dbspace it does not affect any other processes.

CREATE RAW TABLE "informix".save_sqltrace
  (date_time DATETIME YEAR TO SECOND,
   sql_id INT8,
   sql_runtime FLOAT,
   sql_sid INT8,
   sql_uid INT8,
   sql_statement CHAR(11000),
   sql_database CHAR(30))
in sqltrace extent size 100000 next size 50000 lock mode row;

CREATE INDEX "informix".idx_savesql1 ON "informix".save_sqltrace (date_time) IN sqltrace;
CREATE INDEX "informix".idx_savesql2 ON "informix".save_sqltrace (sql_runtime) IN sqltrace;
CREATE INDEX "informix".idx_savesql3 ON "informix".save_sqltrace (sql_id) IN sqltrace;
Tracing in Informix

• Here is the information that needs to be inserted into the “ph_task” table to activate the task.
• In my case I was running it every minute.
• You will want to see what the shortest time that your trace buffer is and make the frequency the task runs smaller than that.

0|save_trace|Saves SQL Trace when run time greater than set value.|TASK|9251||sysadmin|insert into save_sqltrace select current, sql_id, sql_runtime, sql_sid, sql_uid,sql_statement,sql_database from sysmaster:syssqltrace where sql_runtime > 10 and sql_id > (select max(sql_id) from save_sqltrace)| 30 00:00:00|00:00:00| 0 00:01:00|2014-03-27 14:54:17|9237|t|t|t|t|t|t|400|PERFORMANCE|t|0|
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 Slowest SQL Statements.
Options Available with SQEXPLAIN

• 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 )

Query statistics:
-----------------
Table map :

<table>
<thead>
<tr>
<th>Internal name</th>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>partsupp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>type</th>
<th>table</th>
<th>rows_prod</th>
<th>est_rows</th>
<th>rows_scan</th>
<th>time</th>
<th>est_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>scan</td>
<td>t1</td>
<td>26</td>
<td>360</td>
<td>26</td>
<td>00:00:00</td>
<td>49</td>
</tr>
</tbody>
</table>
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%\sqexplain
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

• **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
  – Using “in” causes sequential scans
  – Use of Directives
  – Use of substrings in queries
  – Use of functions in queries
  – Using a better index (Creation of new index)
Functions cause 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 views 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 = '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_deplete.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
                (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, cm_seq_nbr
                (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
Using “in” causes sequential scans

QUERY:
---------------------
SELECT inv3_invno
FROM inv3
WHERE 448050 IN (inv3_physcode, inv3_altphys1, inv3_altphys2)

Estimated Cost: 157852
Estimated # of Rows Returned: 566880

1) informix.cus3: SEQUENTIAL SCAN
   Filters: 448050 IN (informix.inv3.inv3_physcode , informix.inv3.inv3_altphys1 , informix.inv3.inv3_altphys2)

Query statistics:
---------------------
Table map:
---------------------
<table>
<thead>
<tr>
<th>Internal name</th>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>inv3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>type</th>
<th>table</th>
<th>rows_prod</th>
<th>est_rows</th>
<th>rows_scan</th>
<th>time</th>
<th>est_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>scan</td>
<td>t1</td>
<td>21</td>
<td>566880</td>
<td>2096652</td>
<td>00:01:26</td>
<td>157852</td>
</tr>
</tbody>
</table>
Resolution to Criteria used for Using “in” causes sequential scans

• First I create two new indexes
  – create index ixinv3_altphys1 on inv3(inv3_altphys1)
  – create index ixinv3_altphys2 on inv3(inv3_altphys2)

• Then I changed the query to use “or” instead of “in”
Resolution to Criteria used for Using “in” causes sequential scans

QUERY:
-------
SELECT inv3_invno
FROM inv3
WHERE (inv3_physcode = 448050
or inv3_altphys1 = 448050
or inv3_altphys2 = 448050)

Estimated Cost: 13
Estimated # of Rows Returned: 9

1) informix.inv3: INDEX PATH

(1) Index Name: informix.ixinv3_physcode
   Index Keys: inv3_physcode  (Serial, fragments: ALL)
   Lower Index Filter: informix.inv3.inv3_physcode = 448050

(2) Index Name: informix.ixinv3_altphys2
   Index Keys: inv3_altphys2  (Serial, fragments: ALL)
   Lower Index Filter: informix.inv3.inv3_altphys2 = 448050

(3) Index Name: informix.ixinv3_altphys1
   Index Keys: inv3_altphys1  (Serial, fragments: ALL)
   Lower Index Filter: informix.inv3.inv3_altphys1 = 448050
Resolution to Criteria used for Using “in” causes sequential scans

Query statistics:
-----------------

<table>
<thead>
<tr>
<th>type</th>
<th>table</th>
<th>rows_prod</th>
<th>est_rows</th>
<th>rows_scan</th>
<th>time</th>
<th>est_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>scan</td>
<td>t1</td>
<td>21</td>
<td>9</td>
<td>21</td>
<td>00:00.24</td>
<td>13</td>
</tr>
</tbody>
</table>

Table map:
-------------------
<table>
<thead>
<tr>
<th>Internal name</th>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>inv3</td>
</tr>
</tbody>
</table>
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, ........

\text{cm\_onhand\_qty}) \text{ AS}

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

(x0.qty_base - \text{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 Directives in Queries

1) informix.f: INDEX PATH
   Filters: informix.f.effdt = <subquery>
   (1) Index Keys: setid vendor_id vnrdLoc 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 Directives in Queries

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 Directives in Queries

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 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 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 to 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 cause 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 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 (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 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 Unions

```sql
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 PDQ_PRIORITY 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 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 PDQ Priority

SET PDQPRIORITY 100;

SELECT acct, 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

<table>
<thead>
<tr>
<th>session</th>
<th>#RSAM</th>
<th>total</th>
<th>used</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>user</td>
<td>tty</td>
<td>pid</td>
<td>hostname</td>
</tr>
<tr>
<td>324499</td>
<td>indprod</td>
<td>-</td>
<td>27952</td>
<td>prodtu</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tid</th>
<th>name</th>
<th>rstcb</th>
<th>flags</th>
<th>curstk</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>25339601</td>
<td>sqlexec</td>
<td>c0000002b2c9ac18</td>
<td>---PR--</td>
<td>1842583336 sleeping(Forever)</td>
<td></td>
</tr>
</tbody>
</table>

Memory pools count 2

<table>
<thead>
<tr>
<th>name</th>
<th>class</th>
<th>addr</th>
<th>totalsize</th>
<th>freesize</th>
<th>#allocfrag</th>
<th>#freefrag</th>
</tr>
</thead>
<tbody>
<tr>
<td>324499</td>
<td>V</td>
<td>c0000002b60be040</td>
<td>2306048</td>
<td>4315</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>324499_SORT</td>
<td>V</td>
<td>c0000002b59a3040</td>
<td>61440</td>
<td>4048</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>free</th>
<th>used</th>
<th>name</th>
<th>free</th>
<th>used</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort</td>
<td>0</td>
<td>34144</td>
<td>sqscb</td>
<td>0</td>
<td>41344</td>
</tr>
<tr>
<td>sql</td>
<td>0</td>
<td>80</td>
<td>srtmembuf</td>
<td>0</td>
<td>20384</td>
</tr>
</tbody>
</table>

sqscb info

<table>
<thead>
<tr>
<th>scb</th>
<th>sqscb</th>
<th>optofc</th>
<th>pdqpriority</th>
<th>sqlstats</th>
<th>optcompind</th>
<th>directives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c0000002b5be61d0</td>
<td>c0000002cb9d7030</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Sess SQL Current Iso Lock SQL ISAM F.E.
<table>
<thead>
<tr>
<th>Id</th>
<th>Stmt type</th>
<th>Database</th>
<th>Lvl Mode</th>
<th>ERR</th>
<th>ERR</th>
<th>Vers Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>324499</td>
<td>SELECT</td>
<td>elstest</td>
<td>CR Wait</td>
<td>600</td>
<td>0</td>
<td>0 9.03 Off</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>session</th>
<th>#RSAM</th>
<th>total</th>
<th>used</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>16354</td>
<td>vijays</td>
<td>-</td>
<td>16777</td>
<td>prodtu 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2490368</td>
<td>2458128</td>
</tr>
</tbody>
</table>

**Memory pools count 2**

<table>
<thead>
<tr>
<th>name</th>
<th>class</th>
<th>addr</th>
<th>totalsize</th>
<th>freesize</th>
<th>#allocfrag</th>
<th>#freefrag</th>
</tr>
</thead>
<tbody>
<tr>
<td>16354</td>
<td>V</td>
<td>c0000001a12a4040</td>
<td>2244608</td>
<td>27536</td>
<td>4211</td>
<td>82</td>
</tr>
<tr>
<td>16354_SORT</td>
<td>V</td>
<td>c0000001b3df5040</td>
<td>245760</td>
<td>4704</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>free</th>
<th>used</th>
<th>name</th>
<th>free</th>
<th>used</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort</td>
<td>0</td>
<td>34128</td>
<td>sqscb</td>
<td>0</td>
<td>56080</td>
</tr>
<tr>
<td>sql</td>
<td>0</td>
<td>80</td>
<td>srtmembuf</td>
<td>0</td>
<td>204064</td>
</tr>
</tbody>
</table>

**sqscb info**

<table>
<thead>
<tr>
<th>scb</th>
<th>sqscb</th>
<th>optofc</th>
<th>pdqpriority</th>
<th>sqlstats</th>
<th>optcompind</th>
<th>directives</th>
</tr>
</thead>
<tbody>
<tr>
<td>c0000001ad54a8c0</td>
<td>c0000001a12af030</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Sess SQL Current Iso Lock SQL ISAM F.E.**

<table>
<thead>
<tr>
<th>Id</th>
<th>Stmt type</th>
<th>Database</th>
<th>Lvl Mode</th>
<th>ERR</th>
<th>ERR</th>
<th>Vers</th>
<th>Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td>16354</td>
<td>SELECT</td>
<td>elstest</td>
<td>CR Wait 600</td>
<td>0</td>
<td>0</td>
<td>9.03</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Current SQL statement:**

```
select unique b.* from tmp_fids a, name_init b where a.fid = b.fid and a.fid > 0
```
Fragment Tables

```sql
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 )
Subquery Support (Update/Delete)

- The FROM clause of a subquery in the WHERE clause of the UPDATE statement can specify as a data source the same table or view that the Table Options clause of the UPDATE statement specifies.

- UPDATE operations with subqueries that reference the same table object are supported only if all of the following conditions are true:
  - The subquery either returns a single row,
  - Has no correlated column references.

- No SPL routine in the subquery can reference the same table that UPDATE is modifying.
Subquery Support (Update/Delete)

- Example: Updates the stock table by reducing the unit_price value by 5% for a subset of prices. The WHERE clause specifies which prices to reduce by applying the IN operator to the rows returned by a subquery that selects only the rows of the stock table where the unit_price value is greater than 50:

```
UPDATE stock
SET unit_price = unit_price * 0.95
WHERE unit_price IN (SELECT unit_price
 FROM stock
 WHERE unit_price > 50);
```
External Tables

• Ways to use External Tables
  – Creating external tables
    » Create table ext_xyz (col1 integer, col2 char(20))
      USING (DATAFILES (“DISK:/data/xyz.unl”),
              FORMAT “DELIMITED”)
    » Create table ext_xyz SAMEAS xyz
      USING (DATAFILES (“DISK:/data/xyz.unl”),
              FORMAT “DELIMITED”)

External Tables

• Ways to use External Tables
  – Load data from external table into database table
    » insert into xyz select * from ext_xyz;
  – Unload data to external table
    » insert into ext_xyz select * from xyz;
External Tables

• Using PIPE feature
  – When you do not want to unload to a flat file, but pipe the data to another server/table you can use the PIPE feature.

  – Here is what the create table statement looks like:
    » Create external table ext_xyz SAMEAS xyz
      USING ( DATAFILES("PIPE:/data/ext_xyz1"))

  – To improve performance using PIPES make sure that there are enough FIFO VP’s defined. The database server uses one FIFO VP for each named pipe that specify in the DATAFILES clause.
    » To add a FIFO VP – onmode –p + {# to add} FIFO
New Features – Optimizer Directives

– 11.70xC1 – NEW OPTIMIZER DIRECTIVES

• Query optimizer support for star-schema and snowflake-schema queries. A primary key column in each dimension table must correspond to a foreign key in the fact table.

– New optimizer directives have been added:
  » STAR_JOIN, FACT, AVOID_STAR_JOIN & AVOID_FACT
  » Can also enable with SET OPTIMIZATION
New Features – Optimizer Directives

– 11.70xC1 – NEW OPTIMIZER DIRECTIVES

• Syntax support for DDL statement with IF [NOT] EXISTS:
  – Can include “IF [NOT] EXISTS” condition to SQL statements that create a database object or a database.
New Features – Optimizer Directives

- 11.70xC2
  
  • Table and column Aliases in DML statements.

  - SELECT – statements and subqueries can declare an alias in the projection clause for columns in the select list.

  - DELETE / UPDATE – can declare an alias for a local or remote target table.
Informix 12 SQL Features

• Faster ANSI join queries
  – ANSI outer join queries that have equality joins can run faster because the Informix optimizer now uses either a hash join or a nested loop on a cost basis. In earlier releases, Informix used only nested loop joins in ANSI outer joins.

• Temporary table projection optimization for views and derived tables
  – Applications and analytic tools can define a query in which a derived table contains multiple views joined with base tables, potentially including hundreds of columns. The database server materializes this query in a system-generated temporary table. The parent query, however, might project only a few columns.
  – The database server creates internally generated temporary tables that include only the columns that are specified in the Projection list, the WHERE clause, the ORDER BY clause, and in other clauses of the immediate parent query. By excluding unnecessary columns from the temporary table, the database server uses storage resources efficiently and avoids I/O operations on the columns that do not contribute to the query result.
Summary

• Identify Problem SQL Statements

• Tracing SQL in Informix

• Options Available with Set Explain & Reading sqexplain output with tuning examples

• Methods to use for Improving SQL performance

• Subquery Support for Update/Delete

• External Tables

• Informix 12 SQL Features
Questions?

Jeff Filippi
jeff.filippi@itdataconsulting.com