Using Oracle SQL for Searching

Barry Chase, Information Technology

October 17, 2008

bchase@humana.com

Managing Multiple Parameter Inputs via Dynamic SQL

Not Your Mama's Banana Pudding

Barry Chase, Infor
October 17, 2
bchas

HUMANA
Guidance when you need it most

icial Health Benefits Provider
of the PGA TOUR and Champions Tour

It's All About Me

- Currently employed with Humana, Inc. as an Application Architect within Corporate Systems IT.
- Approx. 10 years working experience with primary background in Oracle
- 1/15th second moment of fame
  - Email and FTP Oracle Package Solutions using UTL_TCP published in the O'Reilly 4th edition of *Oracle PL/SQL Programming* book.
  - Additionally provided content review for chapter on UTL_TCP usage in Oracle DB.

Agenda

- Evolution of Technology : Dynamic SQL
- Scenario Overview
- Solution
  - Variable Parameter Inputs
  - Selective Column Display
  - Pagination
- Future Vision
- Questions

Evolution of Technology : Dynamic SQL

Prior to Dynamic SQL availability, SQL was built and tailored specific to the processes being executed. This method incurred a number of performance issues to include but not limited to :

- Excessive SQL Parsing at the Database level
- Long Term Maintenance
- Etc.

Dynamic SQL execution has offered the ability to build SQL on the fly using provided business rules and input parameters. A number of database features have also been made available by Oracle to enhance performance of the dynamic sql.

- Bind Variables
- Bulk Array Processing
- Etc.

Scenario Overview

Scenario : Project requires provision for client side data entry of criteria to select desired data. Must support inclusion/exclusion logic, must be adaptable to new conditions as the application matures. Also include variable column display of information depending on interface location or criteria conditions.

Objective : Provide a flexible search utility which is extensible to changing criteria and column displays.

Solution : Using REFcursors and associative arrays to build dynamic conditional logic. Providing the following key features :

- Variable Parameter Input
- Selective Column Return
- Optimization (Pagination or Chunking)

This presentation is designed to cover the topics above and provide sample logic that can be easily tested and applied to your project development.

Variable Parameter Input :: API Structure

Variable Parameter Input

- Instead of hardcoded api parameters or limiting to an array containing multiple values for a single parameter type, uses an associative array to build a Criteria Grid of all parameters to applied to SQL execution.
- Number of search parameter inputs become limited only by your imagination and business needs
- No bind variables required or potentially minimized greatly
- Logic to filter out potential SQL Injection routines

BEFORE	AFTER

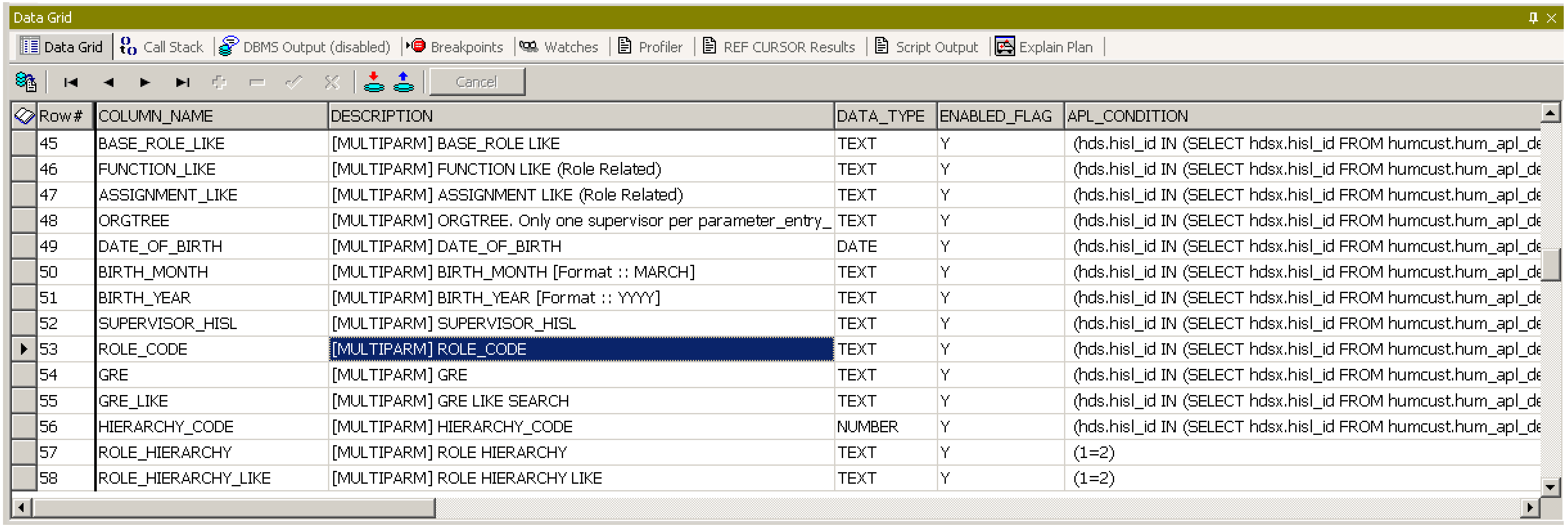
`Get_data_c (`
`  p_person_id in number`
`, p_company in varchar2`
`, p_state_id in varchar2`
`, p_success OUT varchar2`
`, p_sql OUT varchar2`
`, p_cur OUT c_cursor`
`);`
	`Get_data_c (`
	`  p_parameter_list in vc4000_table`
	`, p_success OUT varchar2`
	`, p_sql OUT varchar2`
	`, p_cur OUT c_cursor`
	`);`

Variable Parameter Input

- Variable Parameter input is satisfied by way of a *Criteria Grid*, which is passed as an associative array to the api. The grid is validated and processed to build conditional logic.
- This conditional logic is then appended to the sql being built dynamically. Complex *inclusion* and *exclusion* logic is supported.
- Each *criteria grid* array entry is a concatenated string (pipe separated) of key values, presenting a row of criteria data. Collectively, a set of these then make up conditional logic built dynamically.
- Available criteria parameter name search are to be defined in our QCURSOR_SEARCH_COLUMNS table.

Variable Parameter Input :: Table Preview

- QCURSOR_SEARCH_COLUMNS table


Data Grid
Data Grid | Call Stack | DBMS Output (disabled) | Breakpoints | Watches | Profiler | REF CURSOR Results | Script Output | Explain Plan
Cancel

Row#	COLUMN_NAME	DESCRIPTION	DATA_TYPE	ENABLED_FLAG	APL_CONDITION
45	BASE_ROLE_LIKE	[MULTIPARM] BASE_ROLE LIKE	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
46	FUNCTION_LIKE	[MULTIPARM] FUNCTION LIKE (Role Related)	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
47	ASSIGNMENT_LIKE	[MULTIPARM] ASSIGNMENT LIKE (Role Related)	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
48	ORGTREE	[MULTIPARM] ORGTREE. Only one supervisor per parameter_entry_	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
49	DATE_OF_BIRTH	[MULTIPARM] DATE_OF_BIRTH	DATE	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
50	BIRTH_MONTH	[MULTIPARM] BIRTH_MONTH [Format :: MARCH]	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
51	BIRTH_YEAR	[MULTIPARM] BIRTH_YEAR [Format :: YYYY]	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
52	SUPERVISOR_HISL	[MULTIPARM] SUPERVISOR_HISL	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
53	ROLE_CODE	[MULTIPARM] ROLE_CODE	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
54	GRE	[MULTIPARM] GRE	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
55	GRE_LIKE	[MULTIPARM] GRE LIKE SEARCH	TEXT	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
56	HIERARCHY_CODE	[MULTIPARM] HIERARCHY_CODE	NUMBER	Y	(hds.hisl_id IN (SELECT hdsx.hisl_id FROM humcust.hum_apl_de
57	ROLE_HIERARCHY	[MULTIPARM] ROLE HIERARCHY	TEXT	Y	(1=2)
58	ROLE_HIERARCHY_LIKE	[MULTIPARM] ROLE HIERARCHY LIKE	TEXT	Y	(1=2)

```
 (hds.employee_number IN (SELECT hdsx.employee_number FROM demo_snap hdsx WHERE 1 = 1 AND
EXISTS (SELECT 'x' FROM q_cursor_parms_global_temp WHERE 1 = 1 AND parameter_name =
'BASE_ROLE'|| ' [xxxparameter_entry_idxxx]' AND incl_excl_flag = 'I' AND
UPPER(hdsx.BASE_ROLE) = UPPER(vdata_text))))
```

Variable Parameter Input :: Criteria Setup

- Criteria Grid Elements
  - PARAMETER_ENTRY_ID
    - This identifies the order of the criteria in relation to each other
    - This permits the ability to collapse multiple criteria grid rows into a single conditional entry as in the case of selecting for a list of employee numbers. We call this folding.
  - PARAMETER_NAME
    - This is similar to adding a new parameter input to an API, except that in this array format, I can manage and maintain the desired list of input parameters. Always UPPERCASE
  - VDATA_TEXT
    - This is the parameter varchar2 input value, similar to that of which would be passed on an API parameter input. When not provided the word NULL will be placed in this position.
  - VDATA_DATE
    - This is the parameter date input value, similar to that of which would be passed on an API parameter input. Format should always be represented text as YYYY/MM/DD HH24:MI:SS. When not provided the word NULL will be placed in this position.
  - VDATA_NUMBER
    - This is the parameter number input value, similar to that of which would be passed on an API parameter input. When not provided the word NULL will be placed in this position.
  - INCL_EXCL_FLAG
    - Identifies if this an *exclusion* or *inclusion* criteria entry and represented as an capital E or I.
  - CONDITION_BLOCK_ID, CONDITION_SET_ID, CONDITION_MEMBER_ID
    - These are used to help build the criteria entry into an actual SQL conditions
  - CONDITION_OPERATOR
    - Determines whether criteria entry falls into operator conditions of OR or AND

Variable Parameter Input :: Simple Criteria

Criteria Grid Example

{PARAMETER_ENTRY_ID}|{PARAMETER_NAME}|{VDATA_TEXT}|{VDATA_DATE}|
{VDATA_NUMBER}|{INCL_EXCL_FLAG}|{CONDITION_BLOCK_ID}|
{CONDITION_SET_ID}|{CONDITION_MEMBER_ID}|{CONDITION_OPERATOR}

e.g.

```
l_parameter_list(1) := 1|ORGTREE|NULL|NULL|139245|I|1|1|1|OR
l_parameter_list(2) := 2|EMPLOYEE_NUMBER|NULL|NULL|139245|I|1|1|2|OR
l_parameter_list(3) := 3|BASE_ROLE|Applications Consultant|NULL|NULL|I|2|1|1|AND
```

..

..

and so on.

Variable Parameter Input :: Complex Criteria

- ORGTREE of Supervisor # 45821 or ORGTREE of Supervisor # 53605 or Employee # 45821 or Employee# 53065, but do not include ORGTREE of Supervisor # 3493 who is a direct report to Supervisor 53605.

```
l_parameter_list(1) := 1|ORGTREE|NULL|NULL|45821|I|1|1|1|OR
l_parameter_list(2) := 2|ORGTREE|NULL|NULL|53605|I|1|1|2|OR
l_parameter_list(3) := 3|EMPLOYEE_NUMBER|NULL|NULL|45821|I|1|1|3|OR
l_parameter_list(4) := 3|EMPLOYEE_NUMBER|NULL|NULL|53065|I|1|1|3|OR
l_parameter_list(5) := 4|ORGTREE|NULL|NULL|3493|I|2|1|1|AND
```

- Notice that our parameter_entry_id 3 (EMPLOYEE_NUMBER) is mentioned twice and both rows look identical with exception of the parameter value itself. This is an example of where we are "folding" the criteria entries into a single condition.

- However, note that the array position still increments from position 3 to 4 since there can still only one entry per array index

Variable Parameter Input :: Generated SQL

```
AND ( ( /* ORGTREE */    ( hds.employee_number IN (
                                 SELECT     hdsx. employee_number
                                       FROM demo_snap hdsx
                                      WHERE 1 = 1 AND LEVEL > 1
                                 CONNECT BY PRIOR hdsx. employee_number =
                                                        hdsx.supervisor_number
                                 START WITH hdsx.hisl_id IN (
                                               SELECT vdata_text
                                                 FROMq_cursor_parms_global_temp
                                                WHERE 1 = 1
                                                  AND parameter_name =
                                                           'ORGTREE' || ' [1]'
                                                  AND incl_excl_flag = 'I' ))   )
                         OR /* ORGTREE */ ( hds. employee_number IN (
                                               SELECT     hdsx. employee_number
                                                     FROM demo_snap hdsx
                                                    WHERE 1 = 1 AND LEVEL > 1
                                               CONNECT BY PRIOR hdsx. employee_number =
                                                              hdsx.supervisor_number
                                               START WITH hdsx. employee_number IN (
                                                              SELECT vdata_text
                                                                FROMq_cursor_parms_global_temp
                                                               WHERE 1 = 1
                                                                 AND parameter_name =
                                                                           'ORGTREE'
                                                                        || ' [2]'
                                                                 AND incl_excl_flag =
                                                                            'I' )) )
                         OR /* EMPLOYEE_NUMBER */ ( hds. employee_number IN (
                                               SELECT hdsx. employee_number
                                                 FROM demo_snap hdsx
                                                WHERE 1 = 1
                                                  AND EXISTS (
                                                         SELECT 'x'
                                                           FROMq_cursor_parms_global_temp
                                                          WHERE 1 = 1
                                                            AND parameter_name =
                                                                      'EMPLOYEE_NUMBER'
                                                                   || ' [3]'
                                                            AND incl_excl_flag =
```

Variable Parameter Input :: Criteria Layout

As you can see you can see we build a construct that contains blocks which contain sets which contain members. Although we permit dynamic parameter selections, we enforce a series of rules around this logic.

```
AND (BLOCK 1
   |
   |__>>> SET 1
    |   |____>> MEMBER 1
    |   |____>> MEMBER 2
    |
    |_>>> SET 2
           |
           |___>> MEMBER 1
  )
AND (BLOCK 2
   |
   |__>>> SET 1
    |   |____>> MEMBER 1
    |   |____>> MEMBER 2
    |
    |_>>> SET 2
           |
           |___>> MEMBER 1
           |___>> MEMBER 2
  )
```

Variable Parameter Input :: Criteria Rules

The following rules permit highly flexible queries to be generated dynamically.

The VALIDATE_DEMO_PG_CRITERIA API call will be used to enforce these rules.

- NON RULE VIOLATION ENCOUNTERED, BUT CAUSED FAILURE IN VALIDATION ROUTINE (e.g. Parameters invalid)
- CONDITION MEMBERS ARE SEPARATED BY CONDITION OPERATORS OF 'OR' and 'AND' WHEN MORE THAN ONE CONDITION MEMBER.
- WHEN ONE CONDITION MEMBER ONLY, THEN CONDITION OPERATOR SHOULD BE 'AND'.
- CONDITION OPERATORS APPLY THE CONDITION SET AS A WHOLE AND USED AS A SEPARATOR BETWEEN CONDITION MEMBERS.
- CONDITION SETS CAN ONLY CONTAIN ONE TYPE OF CONDITION OPERATOR. NO MIXING OF OPERATORS.
- ANY GIVEN CONDITION SET CAN CONTAIN ANY NUMBER OF ONE OR MORE CONDITION MEMBERS.

Variable Parameter Input :: Criteria Rules

- ANY GIVEN CONDITION BLOCK CAN CONTAIN ANY NUMBER OF ONE OR MORE CONDITION SETS.
- MUST AT LEAST CONTAIN ONE CONDITION BLOCK.
- CONDITIONS SETS WITHIN A BLOCK ARE ALWAYS SEPARATED BY 'OR' OPERATORS. MEANING ANY ONE OF THE CONDITION SETS CAN BE TRUE, AND DOES REQUIRE ALL CONDITION SETS TO BE TRUE TO SATISFY THE QUERY CONDITION LOGIC.
- CONDITION BLOCKS ARE ALWAYS SEPARATED BY 'AND' OPERATORS. MEANING THAT EACH BLOCK MUST BE TRUE TO SATISFY QUERY CONDITION LOGIC.
- ANY NUMBER OF ONE OR MORE CONDITION BLOCKS CAN BE DEFINED.
- CONDITION BLOCK ID's are UNIQUE. CONDITION SET IDs ARE UNIQUE WITHIN A GIVEN CONDITION BLOCK. CONDITION MEMBER IDs ARE UNIQUE WITHIN A GIVEN CONDITION SET.
- ONLY ONE ORGTREE SEARCH PER PARAMETER_ENTRY_ID.
- KEYWORDS NOT AUTHORIZED IN PARAMETER VALUES (SQL INJECTION RULES)

- DBA	- DBMS_SQL	- GRANT	- CREATE
- ALTER	- REVOKE	- EXECUTE IMMEDIATE	- SELECT

Selective Column Return

Another area of improvement is gained via creating a few, but very flexible, APIs for other systems to leverage. However, each client request clearly has different data needs that not only include what the search conditions are but also what columns of data that should be returned.

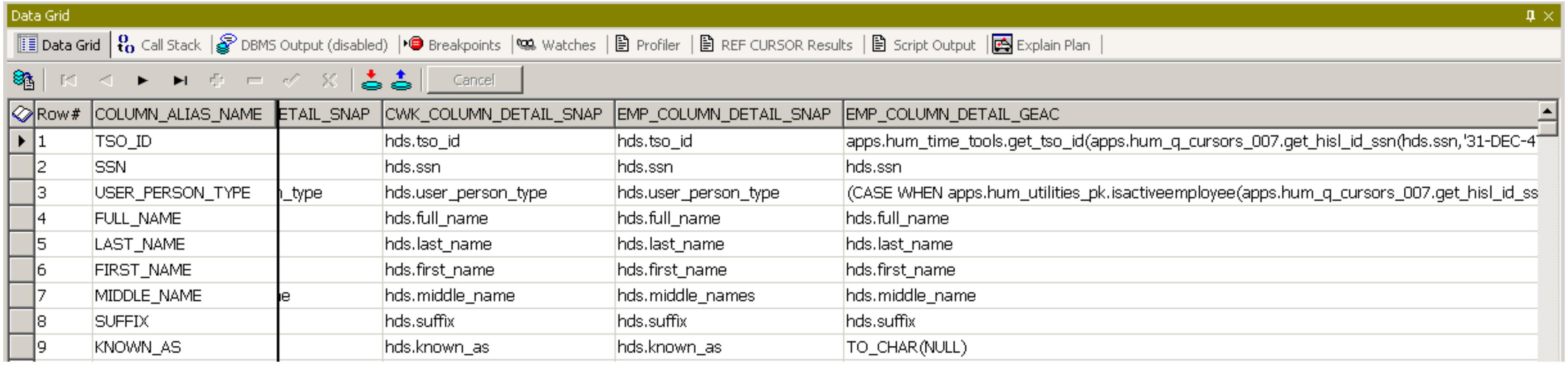
By creating master table, called QCURSOR_DATASET_COLUMN_MSTR, we can then track and deploy a variety of columns to our common api.

To ensure that each client can have their own representative set of column assignments, we leverage the use of PROCESS_DESC and SUB_PROCESS_DESC identifiers. Then populate a QCURSOR_DATASET_COLUMN_DETAIL with the columns desired for that specific feed.

Optionally, we can declare the columns desired on the fly.

Selective Column Return : Table Preview

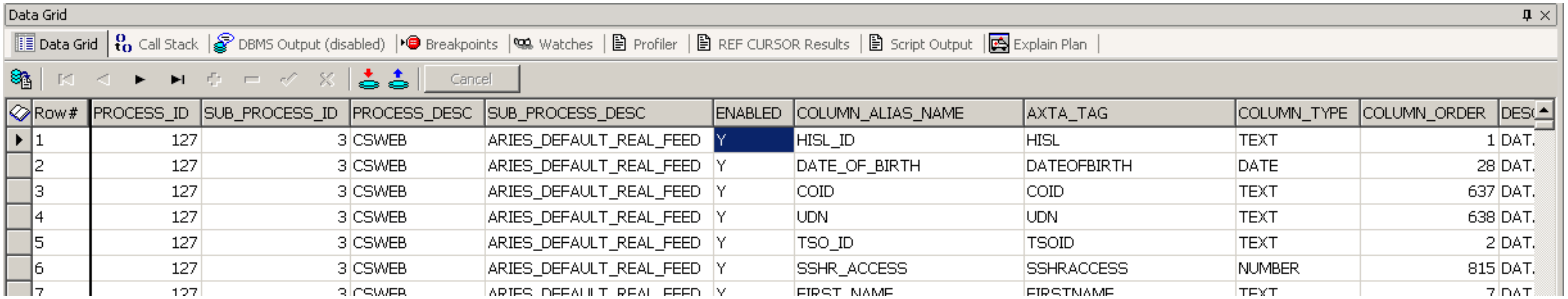
- QCURSOR_DATASET_COLUMN_MSTR



Data Grid

Data Grid | Call Stack | DBMS Output (disabled) | Breakpoints | Watches | Profiler | REF CURSOR Results | Script Output | Explain Plan

Row#	COLUMN_ALIAS_NAME	ETAIL_SNAP	CWK_COLUMN_DETAIL_SNAP	EMP_COLUMN_DETAIL_SNAP	EMP_COLUMN_DETAIL_GEAC
1	TSO_ID		hds.tso_id	hds.tso_id	apps.hum_time_tools.get_tso_id(apps.hum_q_cursors_007.get_hisl_id_ssn(hds.ssn,'31-DEC-4
2	SSN		hds.ssn	hds.ssn	hds.ssn
3	USER_PERSON_TYPE	_type	hds.user_person_type	hds.user_person_type	(CASE WHEN apps.hum_utilities_pk.isactiveemployee(apps.hum_q_cursors_007.get_hisl_id_ss
4	FULL_NAME		hds.full_name	hds.full_name	hds.full_name
5	LAST_NAME		hds.last_name	hds.last_name	hds.last_name
6	FIRST_NAME		hds.first_name	hds.first_name	hds.first_name
7	MIDDLE_NAME	e	hds.middle_name	hds.middle_names	hds.middle_name
8	SUFFIX		hds.suffix	hds.suffix	hds.suffix
9	KNOWN_AS		hds.known_as	hds.known_as	TO_CHAR(NULL)



Data Grid

Data Grid | Call Stack | DBMS Output (disabled) | Breakpoints | Watches | Profiler | REF CURSOR Results | Script Output | Explain Plan

Row#	PROCESS_ID	SUB_PROCESS_ID	PROCESS_DESC	SUB_PROCESS_DESC	ENABLED	COLUMN_ALIAS_NAME	AXTA_TAG	COLUMN_TYPE	COLUMN_ORDER	DES
1	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	HISL_ID	HISL	TEXT	1	DAT.
2	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	DATE_OF_BIRTH	DATEOFBIRTH	DATE	28	DAT.
3	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	COID	COID	TEXT	637	DAT.
4	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	UDN	UDN	TEXT	638	DAT.
5	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	TSO_ID	TSOID	TEXT	2	DAT.
6	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	SSHR_ACCESS	SSHRACCESS	NUMBER	815	DAT.
7	127	3	CSWEB	ARIES_DEFAULT_REAL_FEED	Y	FIRST_NAME	FIRSTNAME	TEXT	7	DAT.

Optimization (Pagination/Chunking)

Final feature to the described solution is the ability to improve performance via options to grab chunks of data, page by page.

- Define page/chunk size dynamically
- Very few bind variables required, and same regardless of call
- Collect a page/chunk at a time
- Improves Web Page performance
- Decreases impact of data pulls on overall system performance
- Permits data-stream processing

Future Vision

Where do we go from here…

- – Self-Service Feed Definition
- – Feed Personalization
- – Approval Cycle
- – Setup Migration
- – ??

HUMANA®

Guidance when you need it most