

# Working with Oracle Clinical



## Effect of Procedure Question Group Attributes on Performance of Batch Validation

# Review of Generated Procedures



- Generation of a Procedure defined by completing Procedure Definition Forms creates a PL/SQL program which executes during Batch Validation
- Often, there are several ways to define the procedure and arrive at the same result

**Determination of the most efficient way requires understanding the program structure**

# Structure of the Procedure



- Procedure is a series of nested loops
- Outermost loop fetches the Patient information
  - Next Loop fetches DCM information as defined by the first Procedure Question Group definition
    - Next fetches DCM information for second group
      - Next fetches DCM information for the third group
        - . . . . .
- After fetching the innermost loop, the expression details are evaluated

# Structure of the Procedure

```
LOOP<Patient>
Fetch Patient cursor
  LOOP <DCM1>
  Fetch DCM1 Cursor 1
    LOOP <DCM2>
    Fetch DCM2 Cursor ...
      LOOP <DCMn>
      Fetch DCM Cursor n
        Execute Test Expressions
        Next DCM Cursor n Fetch ...
      Next DCM2 Cursor Fetch
    Next DCM1 Cursor Fetch
  Next Patient Cursor Fetch
```

# What Increases Procedure Execution Performance



- Reduce Number of database fetches
  - Reducing number of fetches with the cursor should provide the most improvement
- Reduce Number of times a detail line is used to perform a check
  - Performing another fetch but not having to execute the details will show some improvement
- Reduce Number of detail lines in the procedure
  - The fewest details per procedure **but** may even be counter-productive for overall batch validation session

# Reduce Database Fetches



- Code generation derives the WHERE clause for each DCM Procedure Question Group defined as an alias (cursor)
- Minimize cursor fetches by controlling the WHERE clause
  - Example: Correlation on Actual Event
    - Two cursors: AE and CM where CM is correlated to AE on Actual Event

# Outer Cursor of Correlated Pair

```
Editor
/*-----*/
create or replace package RXCPD_24901_0 as

/* cursor for getting ADVERSE_EVENT Production */
cursor AE_CUR(
I_PATIENT_POSITION_ID in RECEIVED_DCMS.PATIENT_POSITION_ID%TYPE,
I_BEGIN_VISIT_NUMBER in RECEIVED_DCMS.VISIT_NUMBER%TYPE := RXCPDSTD.C_BEGIN_VISIT_NUMBER,
I_END_VISIT_NUMBER in RECEIVED_DCMS.VISIT_NUMBER%TYPE := RXCPDSTD.C_END_VISIT_NUMBER) is
select /*+ ordered use_nl(RDCM RES)
index(RDCM RECEIVED_DCM_UK2_ID) */
RDCM.RECEIVED_DCM_ID,
RDCM.RECEIVED_DCM_ENTRY_TS,
RDCM.INVESTIGATOR_ID,
RDCM.SITE_ID,
RDCM.DCM_ID,
RDCM.DCM_SUBSET_SN,
RDCM.DCM_DATE,
RDCM.DCM_TIME,
RDCM.ACTUAL_EVENT_ID,
RDCM.LAB_ID,
```

**ACTUAL\_EVENT\_ID is retrieved as part of the standard cursor variables in the outer cursor**

# Inner Cursor of Correlated Pair

Editor

```
cursor CM_CUR(  
I_PATIENT_POSITION_ID in RECEIVED_DCMS.PATIENT_POSITION_ID%TYPE,  
I_BEGIN_VISIT_NUMBER in RECEIVED_DCMS.VISIT_NUMBER%TYPE := RXCPDSTD.C_BEGIN_VISIT_NUMBER,  
I_END_VISIT_NUMBER in RECEIVED_DCMS.VISIT_NUMBER%TYPE := RXCPDSTD.C_END_VISIT_NUMBER,  
I_ACTUAL_EVENT_ID in RECEIVED_DCMS.ACTUAL_EVENT_ID%TYPE) is  
select /*+ ordered use_nl(RDCM RES) */  
index(RDCM RECEIVED_DCM_UK2_IDX) */  
RDCM.RECEIVED_DCM_ID,  
RDCM.RECEIVED_DCM_ENTRY_TS,  
RDCM.INVESTIGATOR_ID,  
RDCM.SITE_ID,  
RDCM.DCM_ID,  
RDCM.DCM_SUBSET_SN,  
RDCM.DCM_DATE,  
RDCM.DCM_TIME,
```

**Inner cursor has actual\_event\_id as an input cursor variable**

```
max(decode(RES.DCM_QUESTION_ID,74301,RES.RESPONSE_ID,NULL)) INDICATION$RESP_ID,  
max(decode(RES.DCM_QUESTION_ID,74301,RES.RESPONSE_ENTRY_TS,NULL)) INDICATION$ENT_TS  
from RECEIVED_DCMS RDCM,  
RESPONSES RES  
where RDCM.PATIENT_POSITION_ID = I_PATIENT_POSITION_ID  
and RDCM.DCM_ID = 54901  
and RDCM.END_TS = TO_DATE(3000000,'J')  
and RDCM.RECEIVED_DCM_ID+0 = RES.RECEIVED_DCM_ID  
and RES.END_TS = TO_DATE(3000000,'J')  
and RES.DCM_QUESTION_GROUP_ID = 61001 and RES.CLINICAL_STUDY_ID = 10101  
and RDCM.ACCESSIBLE_TS <= SYSDATE  
and RDCM.VISIT_NUMBER between I_BEGIN_VISIT_NUMBER and I_END_VISIT_NUMBER  
and RDCM.ACTUAL_EVENT_ID = I_ACTUAL_EVENT_ID  
and res.dcm_question_id in (  
73901, 74301)  
group by RDCM.RECEIVED_DCM_ID,  
RDCM.RECEIVED_DCM_ENTRY_TS,
```

**which is used in the cursor's where clause**

# Call to Open Cursor for the Inner Loop

Editor

```
CM$HAS_DATA := 'N';  
CM$FIRST_REPEAT := true;  
if I_MODE = 'P'  
  then open CM_CUR(RXCPDSTD.PATIENTS_REC.PATIENT_POSITION_ID, CM$BEGIN_SEQNUM, CM$END_SEQNUM, AE.actual_event_id);  
  else open CM_CURT(RXCPDSTD.PATIENTS_REC.PATIENT_POSITION_ID, CM$BEGIN_SEQNUM, CM$END_SEQNUM, AE.actual_event_id);  
end if;
```

**Call to open inner CM cursor uses the actual\_event\_id retrieved for the outer\_cursor**

# Reduce Number of Detail Checks



- Checks are made after the cursors have fetched data but before the details are evaluated
  - Example: Qualifying Expression to test only those records where the Medication on the AE CRF is not null

# Testing after Fetch

```
Editor
begin
  RXCPDSTD.V_CODE_LOCATION := 'Post Patient';
  /***** Post Patient *****/

  /***** Post Patient *****/
  AE$FIRST_REPEAT := true;
  if I_MODE = 'P'
    then open AE_CUR(RXCPDSTD.PATIENTS_REC.PATIENT_POSITION_ID, AE$BEGIN_SEQNUM, AE$END_SEQNUM);
    else open AE_CURT(RXCPDSTD.PATIENTS_REC.PATIENT_POSITION_ID, AE$BEGIN_SEQNUM, AE$END_SEQNUM);
  end if;
  <<AE_cursor>>
  loop << fetch_AE_cur>>
    RXCPDSTD.V_CODE_LOCATION := 'Fetching AE data';
    <<AE_fetch>>
    loop
      if I_MODE = 'P'
        then fetch AE_CUR into AE;
          exit AE_cursor when AE_CUR%notfound;
        else fetch AE_CURT into AE;
          exit AE_cursor when AE_CURT%notfound;
        end if;
        exit AE_fetch when AE.MEDICATION is not null;
      end loop AE_fetch;
    if I_MODE = 'P'
      then exit when AE_CUR%notfound;
      else exit when AE_CURT%notfound;
    end if;
  end loop;
```

**Loop until there are no more received\_dcms OR until an AE record is found where Medication is not null**

# Effecting Performance Summary



**Different Attributes are handled  
differently in the Generated  
PL/SQL Program**

# Effecting Performance Summary

<u>Parameter</u>	<u>Where Implemented</u>	<u>Effect on Performance</u>
Sort Order	In DCM Cursors	Negligible
Event Range	In DCM Cursors	Large
First/Last Event	Test after DCM Fetch	Small
Correlation		
Event	In DCM Cursor	Large
Qualifying Value	In DCM Cursor	Large
Correlating Questions	In DCM Cursor	Large
Single Repeat Only	Test after DCM Fetch	Medium
Where Clause Extension	In DCM Cursor	Large
Qualifying Expression	Test after DCM Fetch	Small

**Note: In many cases, parameters must be changed so procedure performs correctly**