THE KEYS TO BREAKTHROUGH APPLICATIONS

Show Plan to Generated COS code

Brendan Bannon

9/11/2013

INTERSYSTEMS

Introduction

- **In this class we will look at Show Plans for Queries and then the generated code and try to see how the two relate to one another.**
  - Match up phases in the Show Plan with part of the COS
  - Show what is missing from the Show Plan
  - Show what can be misleading in the Show Plan
  - Identify what the different line tag of the COS mean

Basic Query Plan

Simple Query looping over 2 tables.

```
SELECT T.ID, T.Name, T.Title, A.ChildSub, A.Name,
    A.HireDate, A.DaysWorked

FROM WITS.TeamLeaders T LEFT OUTER JOIN
    WITS.Advisors A ON T.ID = A.ParentPointer

WHERE T.Title LIKE 'Senior%'
```

Basic Show Plan

- **Relative cost = 433318**
- Read master map WITS.TeamLeaders.IDKEY, looping on ID.
- For each row:
  - Read master map WITS.Advisors.IDKEY, using the given ParentPointer, and looping on childsub,
    generating a row padded with nulls if none found.
- For each row:
  Output the row.

What is missing from the Show Plan?

- WHERE T.Title LIKE 'Senior%'
- When we get the fields
- When we execute the compute code

Embedded SQL

- The INTO list is used as variables in the generated code so it will be a little easier to read.

```
SQL1     ;

    #SQLCOMPILE SELECT=ODBC

        &SQL(DECLARE cur CURSOR FOR
        SELECT T.ID, T.Name, T.Title, A.ChildSub, A.Name, A.HireDate, A.DaysWorked
         INTO    :TID, :TName, :TTitle, :AChildSub, :AName, :AHireDate, :ADaysWorked
        FROM WITS.TeamLeaders T
    LEFT OUTER JOIN WITS.Advisors A ON T.ID = A.ParentPointer
        WHERE T.Title LIKE 'Senior%')

        &SQL(OPEN cur)
        f  &SQL(FETCH cur) QUIT:SQLCODE'=0
        &SQL(CLOSE cur)
```

Show Plan to COS

- Read master map WITS.TeamLeaders.IDKEY, looping on ID.

```
; asl MOD# 2

 s TID="“
%0AmBk1 s TID=$o(^WITS.TeamLeadersD(TID),1)

 i TID="" g %0AmBdun
```

Show Plan to COS

- Read master map WITS.Advisors.IDKEY, using the given ParentPointer, and looping on childsub,

```
    s AChildSub="“
%0AmDk1 i %cur035322p(4)=2 g %0AmDdun

     s AChildSub=$o(^WITS.TeamLeadersD(%cur035322d(15)
    ,"ChildPointer",AChildSub),1)

     i AChildSub="" g %0AmDdun:%cur035322p(4)=1
    g %0AmD0pad
```

Cursor Based Tags

- **Tags related to the cursor commands use the cursor name in the tag name.**
- **For this example the cursor name is cur so we have**

%cur0o	for the OPEN
%cur0f	for the FETCH
%cur0c	for the CLOSE
%cur0E	for the error trap

Looping tags

- The real work start at the tag with first in the name, if there is only one query in the routine the tag will be
  - %0Afirst
- Tags with a lower case k and then a number at the end are looping tags
  - %0AmBk1 First loop on first global
  - %0AmDk1 First loop on second global
- If the global had multiple subscripts to loop on you would have multiple tags: %0AmBk1, %0AmBk2, %0AmBk3

Read Committed Code

- **If running in Read Committed Mode we need to make sure we can lock the row and then we double check the values have not changed.**
  - g:$zu(115,2)=0 %0AmBuncommitted…..
  - %0AmBuncommitted ;

Conversion Code and Compute Code

- This code is generated just before the OPEN code
- Conversion code tags have a lower case s in the tag name.
  - %0AmBs1 called from %0AmBk1+2
  - %0AmDs1 called from %0AmDk1+3
  - %0AmDs2 called from %0AmDk1+3
- Compute code tags have a lower case r in the tag
  - %0AmDr3 called from %0AmDk1+5

Calling a Sub Module

- Sometimes are part of the processing of data we need to call out to a different block of code to prep some part of the data.

```
SELECT A.Name, I.Status, I.OpenDate
FROM WITS.Advisors A JOIN WITS.Issues I ON A.ID =
    I.Owner
WHERE A.HireDate = ?
```

Query Plan

- Read master map WITS.Issues.IDKEY, looping on ID.
- For each row:
- Call module D, which populates temp-file A.
  Read temp-file A, using the given ID.
  For each row:
  Output the row.

- module D
- Read index map WITS.Advisors.HireDateIndex, using the given HireDate, and looping on ParentPointer and childsub.
- For each row:
- Read master map WITS.Advisors.IDKEY, using the given idkey value.
  Add a row to temp-file A, subscripted by ID,
  with node data of Name.

How Many Times is Module D called?

- [stats] Time in Module D = 0.000 Module Execution Count = 1 Global References = 0 Commands Executed = 51
- Read index map WITS.Advisors.HireDateIndex, using the given HireDate, and looping on ParentPointer and childsub.
- For each row:
- Read master map WITS.Advisors.IDKEY, using the given idkey value.
  Add a row to temp-file A, subscripted by ID,
  with node data of Name.

Sub Query Example

```
select Unit, Ten_Status, Officer_Name, Date_Time, Action
from OPD_CADCOPY.Unit_Log as A
where (Date_Time = (select MAX(B.Date_Time)
from OPD_CADCOPY.Unit_Log as B
where A.Unit = B.Unit
and Date_Time > '1999-01-01'
and (A.Action = 'OnDuty'
or A.Action = 'OffDuty'
or A.Action = 'OffDuty Mobile'
or A.Action = 'In Service'
or A.Action = 'Out Of Service'
or A.Action = 'Out Of Service Mobile')))
and (A.Action = 'In Service')
order by Unit
```

Sub Query Plan does not look so bad

- subquery
- Call module E.
  Determine subquery result.

- module E
- Call module G, which populates bitmap temp-file B.
- Generate a stream of idkey values using the multi-index combination:
- ((bitmap index OPD_CADCopy.Unit_Log.Unit) INTERSECT (bitmap temp-file B))
  For each idkey value:
- Read master map OPD_CADCopy.Unit_Log.IDKEY, using the given idkey value.
  Accumulate the max(Date_Time).

- module G
- Read index map OPD_CADCopy.Unit_Log.DateTime, looping on Date_Time (with a range condition) and ID.
- For each row:
- Add ID bit to bitmap temp-file B.

BUT

- module G
- [stats] Time in Module G = 26.041 Module Execution Count = 115 Global References = 28,521,035 Commands Executed = 70,491,690
- Read index map OPD_CADCopy.Unit_Log.DateTime, looping on Date_Time (with a range condition) and ID.
- For each row:
- Add ID bit to bitmap temp-file B.

Problem

- Even though we are using an index this module is taking 26 or the 28 seconds of the query run.
- The range condition is very big
  - Date_Time > '1999-01-01'
- We are calling the module 115 times because it is based on a value from the outer query
  - where A.Unit = B.Unit

Solution

- **We need to add an index to the class based on Unit and Date_Time.**
  - Index WITSIndex On (Unit, DateTime);
- **Before**
  - [stats] Time in Module MAIN = 28.036 Module Execution Count = 3 Global References = 28,671,996 Commands Executed = 73,427,556 Number of Rows = 2
- **After**
  - [stats] Time in Module MAIN = 0.049 Module Execution Count = 3 Global References = 22,067 Commands Executed = 168,096 Number of Rows = 2

THE KEYS TO BREAKTHROUGH APPLICATIONS

Show Plan to Generated COS code

Brendan Bannon

INTERSYSTEMS