

see.
learn.
share.

HOW.

Creating Better OLAP Queries in Cognos 8

Rick Blackwell

Agenda

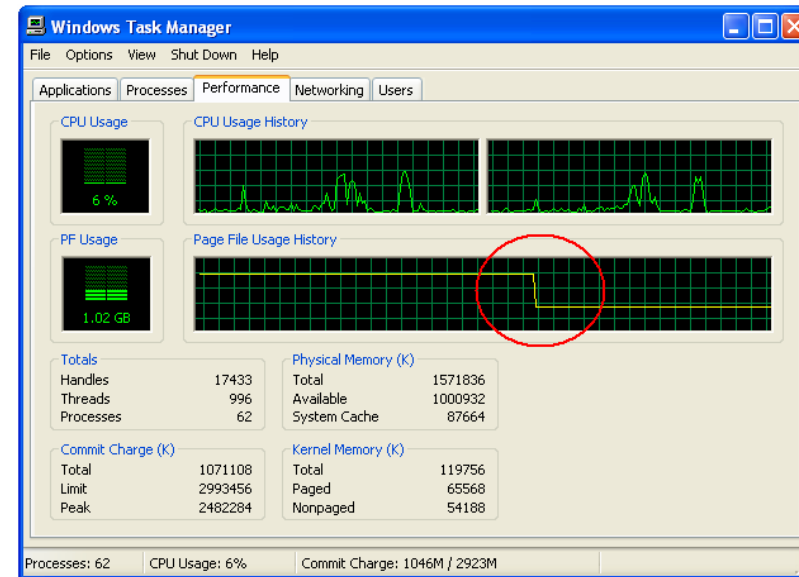
- Introduction
- Understanding OLAP Queries
- Creating Efficient OLAP Queries
- Customer Use Case
- Scalar Functions in OLAP Expressions
- Detail filters
- Conclusion

Introduction

- Like SQL, OLAP queries must be well-designed for correct results and performance
- This presentation focuses on data retrieval and filtering to create efficient queries

How this Paper Came to Exist

- Reports show symptoms such as:
 - Report fails after exceeding 64 million cell limit on crosstab cube engine.
 - Report runs a very, very long time for no apparent reason.
 - Report fails with an RSV-BBP-0022 The absolute affinity request 'asynchWait_Request' failed



Understanding OLAP Queries

- Relational queries are driven off the data
 - Sparse reports are unusual
 - Outer joins can cause sparse data but tend to be limited in use
- OLAP queries are driven off the metadata
 - Sparse reports are often the norm when mixing hierarchies

Crosstab Example

- Ex. Crosstab report with and Retailer name and Order method
- Crosstab

Revenue		<#Order year#>	<#Order year#>
<#Retailer name#>	<#Order method#>	<#1234#>	<#1234#>
	<#Order method#>	<#1234#>	<#1234#>
<#Retailer name#>	<#Order method#>	<#1234#>	<#1234#>
	<#Order method#>	<#1234#>	<#1234#>

Crosstab Behavior with RDBMS

- RDBMS
 - We see only Retailers and Order methods for which there is data

Advanced Climbing Ltd	Mail	\$5,338.80		
	E-mail		\$11,042.98	
	Telephone		\$5,346.56	
	Web		\$11,793.26	

Crosstab Behavior with OLAP

- OLAP
 - We see all members regardless if there is data

Advanced Climbing Ltd	Fax	\$0.00	\$0.00	\$0.00
	Telephone	\$0.00	\$5,346.56	\$0.00
	Mail	\$5,338.80	\$0.00	\$0.00
	E-mail	\$0.00	\$11,042.98	\$0.00
	Web	\$0.00	\$11,793.26	\$0.00
	Sales visit	\$0.00	\$0.00	\$0.00
	Special	\$0.00	\$0.00	\$0.00

- Because ... Members from different hierarchies are cross joined ...
 - All members from one hierarchy are paired with all members of the other hierarchy

Crosstab Behavior with OLAP

Consider this example

Revenue				<#Year#>	<#Year#>
<#Product line#>	<#Staff name#>	<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
		<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
	<#Staff name#>	<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
		<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
<#Product line#>	<#Staff name#>	<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
		<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
	<#Staff name#>	<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
		<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>

115 Product names * 104 Staff names * 114 Retailer names * 7 Order methods
 = 12,847,800 rows

Crosstab Behavior with OLAP

Consider this example

Revenue				<#Year#>	<#Year#>
<#Product line#>	<#Staff name#>	<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>
		<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
There are only 11,000 rows of source data					
			<#Order Method1#>	<#1234#>	<#1234#>
		<#Retailer name#>	<#Order Method1#>	<#1234#>	<#1234#>
			<#Order Method1#>	<#1234#>	<#1234#>

115 Product names * 104 Staff names * 114 Retailer names * 7 Order methods
= 12,847,800 rows

Crosstab Behavior with OLAP

- OLAP
 - Members from same hierarchy
 - Parent-child relationships are used rather than cross joins

Revenue		2004	2005	2006
Camping Equipment	Cooking Gear	\$1,188,011.28	\$1,871,272.24	\$2,341,848.56
	Sleeping Bags	\$2,606,764.70	\$4,094,981.02	\$4,987,198.60
	Packs	\$3,259,319.58	\$5,095,093.26	\$6,273,583.42
	Tents	\$11,025,593.26	\$16,570,227.26	\$19,860,897.58
	Lanterns	\$2,391,640.06	\$3,741,719.36	\$4,405,840.74
Golf Equipment	Irons	\$1,910,593.72	\$3,254,333.98	\$3,636,537.28
	Putters	\$482,401.34	\$880,927.82	\$947,038.08
	Woods	\$3,030,709.00	\$5,128,543.30	\$5,822,864.34
	Golf Accessories	\$174,276.80	\$317,153.60	\$320,086.32

Reduce the number of members prior to the cross joins

- Considerations
 - Dimensional expressions
 - Slicers
 - Detail filters
 - Filtering on members
 - Filtering on measures

Filter Projected Member Sets for Good Performance

- Hierarchy reference –all members in the hierarchy
- Level reference - all members in the level
- Individual members
- Dimension functions
 - set([Golf Equipment], [Camping Equipment])
 - children([Camping Equipment])
 - descendants(([Golf Equipment], 2)
- Detail filter using members
 - [Product line] = [Camping Equipment]

Filter Un-projected Member Sets

- Slicer using member sets
- Detail filter using members
 - [Order Method1] = [Fax]

Filter Member Attributes

- Generally not fast performers
- OLAP providers generally are not well optimized
 - Data source dependant
 - Get to know your OLAP provider
- Generally similar to SQL table scans
- Examples
 - roleValue() attributes
 - intrinsic member attributes – Caption, Long name, Short name
 - user defined attributes

Filter Measures

- Reduce members on edges by testing measure value
- Detail filters
 - Test measures at crosstab cell level
 - Remove leaf level members from both level row and column edges

Revenue	2004	2005	2006
Camping Equipment	\$20,471,328.88	\$31,373,293.14	\$37,869,368.90
Golf Equipment	\$5,597,980.86	\$9,580,958.70	\$10,726,526.02
Mountaineering Equipment	\$0.00	\$9,642,674.54	\$11,248,676.06
Outdoor Protection	\$1,536,456.24	\$988,230.64	\$646,428.04
Personal Accessories	\$7,144,797.52	\$10,955,708.04	\$13,793,960.30



Revenue	2005	2006
Camping Equipment	\$31,373,293.14	\$37,869,368.90

[Revenue] > 25000000

Note: If the filter results in all child members being removed the parent member will also be removed

Creating Efficient OLAP Queries

- filter() function
- Can be applied to any level
- Affects only the item it is applied to

Revenue	2004	2005	2006
Camping Equipment	\$20,471,328.88	\$31,373,293.14	\$37,869,368.90
Golf Equipment	\$5,597,980.86	\$9,580,958.70	\$10,726,526.02
Mountaineering Equipment	\$0.00	\$9,642,674.54	\$11,248,676.06
Outdoor Protection	\$1,536,456.24	\$988,230.64	\$646,428.04
Personal Accessories	\$7,144,797.52	\$10,955,708.04	\$13,793,960.30



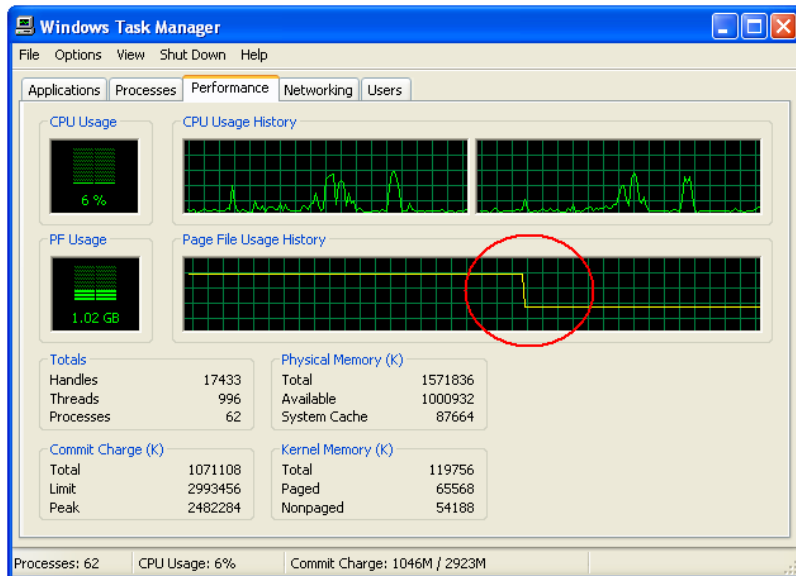
Revenue	2004	2005	2006
Camping Equipment	\$20,471,328.88	\$31,373,293.14	\$37,869,368.90
Golf Equipment	\$5,597,980.86	\$9,580,958.70	\$10,726,526.02
Personal Accessories	\$7,144,797.52	\$10,955,708.04	\$13,793,960.30

filter([Product line], [Revenue] > 25000000)

Note: If the filter results in all child members being removed the parent member will also be removed

Customer Use Case

- Report ran 'forever' and then Report Server failed



- PowerCube as data source

Dimension	Levels	Leaf Members
Customers	9	1,500
Products	7	15,000
Item	7	19,000
Supplier	9	1,500
Time	4	2,500
Customer Organization	1	20
Currency	1	200
Event Type	1	5
Event Start Date	1	1,900
Event End Date	1	1,900
Delay Reason	1	10
Reorder Status	1	10
Delay Cause	1	35
Item Deleted Flag	1	2
Etc for 25 dimensions		
Measures	1	Event Count

Report Definition

- Crosstab Rows
 - Customer Location: The Customer Location - Long Name attribute
 - Event Type: The Event Type - Long Name attribute
 - Product Line: The Product Line- Long Name attribute
 - Product: The Product - Long Name attribute
 - Supplier Site: The Supplier Site - Long Name attribute
 - Delay Reason: The Delay Reason - Long Name attribute
 - Delay Cause: The Delay Root Cause - Long Name attribute
- Crosstab Columns
 - Last 7 Days member
- Crosstab Cells
 - Event Count measure

Report Definition

- Detail Filters
 - Customer Location in ('Central Europe', 'Northern Europe', 'Southern Europe')
 - Event Type in ('Forecast Out of Stock', 'Out of Stock')
 - [Item Deleted Flag] = 'No'
 - [Event End Date] = 'Open'
 - [Event Count] > 0
 - This acts as a zero suppression filter

Diagnosis

- Diagnosis discussion

Diagnosis

- Crosstab Rows
 - Long name attributes – extraneous complexity
- Crosstab Columns
 - A single member poses no problem.
- Crosstab Cells
 - A single measure poses no problem.

Customer Use Case

- Detail filters
 - Filters long name attributes results in slow filtering operations
 - The Event Count filter is applied after the intermediate row set has been created.
 - = Customers * Event Types * Products * Reason * Item * Supplier * Root Cause
 - = 1,500 * 5 * 15,000 * 10 * 19,000 * 1,500 * 35
 - = 1,122,187,500,000,000,000 rows
 - = 1.1 Quintillion rows

Solution

- Crosstab Rows
 - Replace long name attribute references with level references to improve report maintainability.
- Crosstab Columns
 - No change
- Crosstab Cells
 - No change

Solution

- Detail filter - Customer Location
- Move to Customer Location data item
 - set([Central Europe], [Northern Europe], [Southern Europe])
- Or use a member based detail filter
 - Customer Location in ([Central Europe], [Northern Europe], [Southern Europe])

Solution

- Detail filter - Event Type
- Move to Event Type data item
 - set([Forecast Out of Stock], [Out of Stock])
- Or a member based detail filter:
 - [Event Type] in ([Forecast Out of Stock], [Out of Stock])

Solution

- Detail filter - Item Deleted Flag and Event End Date
 - Filter un-projected members
 - Member based detail filters
 - [Item Deleted Flag] = [No]
 - [Event End Date] = [Open]
 - Convert them to slicers
 - Move them into the row edge filter expressions:
 - filter([Customer Locations],
tuple(currentMember([Customers]), [N]) <> 0)
 - where [N] is a member in the Item Deleted Flag

Solution

- Detail filter - Event Count
 - Will use to suppress zero value members at all levels on row edge
 - Reduce the number of member prior to the cross join
 - Customer Locations?
 - There are only 4 Customer Locations included in the report. This does not seem to be the highest priority.
 - Event Type?
 - There are only 2 Event Types included in the report. This does not seem to be the highest priority.
 - Product Line
 - There are over 800 products lines. Removing as many as possible early in the processing would be very beneficial. We will change the expression to:
 - `filter([Product Line], [Event Count]) <> 0)`

Solution

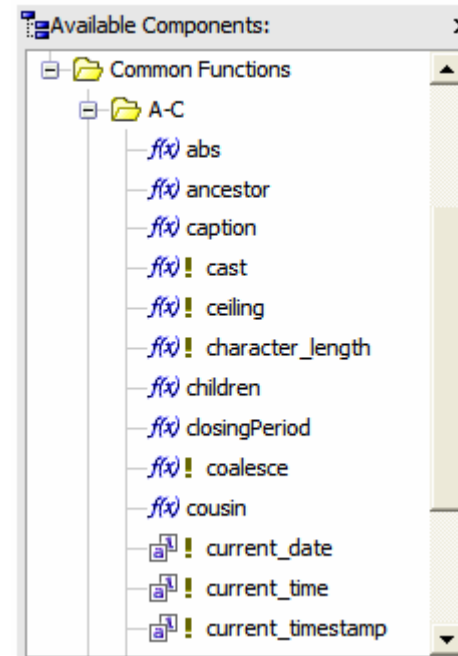
- Product
 - There are over 19,000 products. Removing as many as possible early in the processing would be very beneficial. We will change the expression to:
 - `filter([Product], [Event Count]) <> 0)`
- Supplier Site
 - There are over 1,500 sites. Removing as many as possible early in the processing would be very beneficial.
 - `filter([Supplier Site], [Event Count]) <> 0)`
- Delay Reason
 - While there are only 12 reasons, we'll err on the side of caution.
 - `filter([Delay Reason], [Event Count]) <> 0)`
- Delay Cause
 - As this is the lowest level edge item, we must filter on it to provide the same functionality as the original detail filter:

Real Life Results

- Conclusion
 - Report executes ~2 minutes 10 seconds
 - Report returns ~145 rows of data
 - Good test of our theories
 - Test case used in the development of this document.

Scalar Functions in OLAP Expressions

- Quality of Service indicators
- ! indicates expensive operation
- OLAP – locale execution



Step-by-Step Query Creation

- Focus on a performant final result
- Add one member set at time
- Filter as you go
- Build performance into the report, don't try to correct after-the-fact
- Build and test with realistic data volumes
 - No surprises going into production

Scalar Functions in OLAP Expressions

- In the case of OLAP data, this means:
 - Data will be fetched from the underlying OLAP data provider
 - The data will be streamed through the Cognos data engine where the scalar functions will be applied.
 - The data will be transformed into an in-memory PowerCube
 - The PowerCube will be queried to create the final report
- In other words, there is a lot of overhead to using scalar functions with OLAP.
- Use them wisely and judiciously, be aware of the resource and performance implications

Detail and Summary Filters

- Relational construct not ideally suited to OLAP queries
 - OLAP queries processed in the cube
 - Detail filters are processed ‘in the report’
 - Timing issues can cause unexpected results
- Should only be used with a completely SQL approach
 - Level definitions or query item on edges
 - No members sets or other OLAP constructs
- Best replaced with OLAP constructs
 - Expressions, slicers, etc.

Detail and Summary Filters

- Query Studio uses detail and summary filters
 - Query Studio uses only SQL constructs even on OLAP
 - Safe approach
- Analysis Studio uses only OLAP constructs
 - Analysis Studio does not use detail or summary filters

Conclusion

- Just as developed habits for writing performant SQL queries, we need to develop habits for writing performant OLAP queries.

Additional Forum Resources For You:

Breakout Sessions

- Check the Onsite Guide for other sessions within this conference track
-



Performance World

- Meet one-on-one with Cognos experts at one of the Cognos exhibits in the Solutions Zone
 - See a 30-minute demo of new features and products in one of two Demo Theaters in the Solutions Zone
-

Latest Forum Updates

- Check Cognos Central for the latest developments at Forum
-

Additional Resources For You:

Continue your learning

- Full curriculum of Cognos 8 BI courses, including “What’s New” courses
-

Get implementation assistance

- Download the Cognos Solutions Implementation Methodology & roadmaps for BI, Migration & Conversion
 - Ensure success with services from Cognos Consulting
-

Leverage additional customer resources

- Proven Practices & other valuable Support resources
-

<http://support.cognos.com>

