

Designing Cubes for Performance

Stacia Misner, MVP, MCTS, MCITP
Author, Educator, Consultant

smisner@datainspirations.com
blog.datainspirations.com
Twitter: @Stacia Misner





Agenda

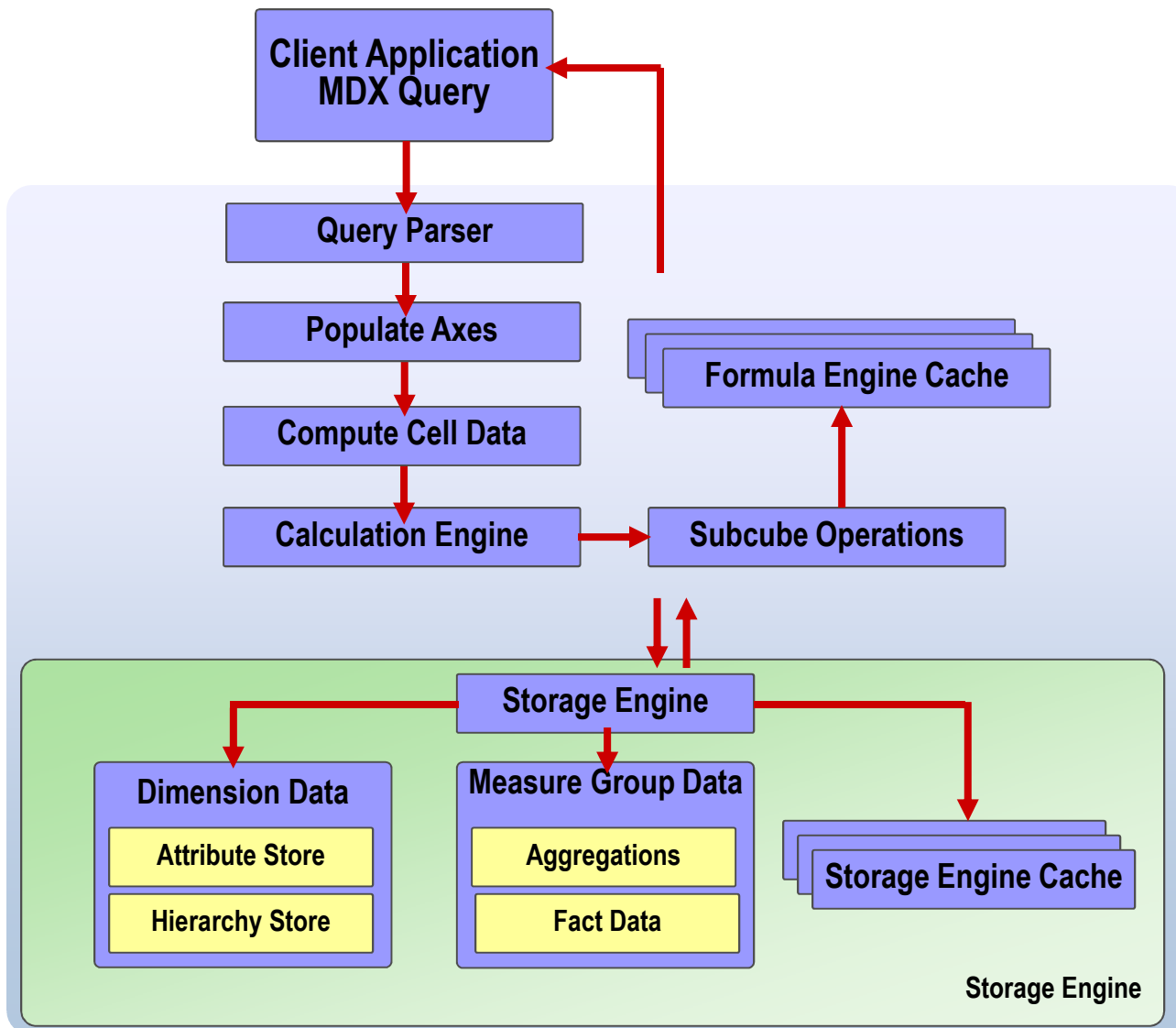
- Introduction
- Designing for Query Performance
- Designing for Processing Performance



OLAP Requirements

- Fast queries
 - User-facing application
 - First benchmark for measuring success
- Fresh and accurate data
 - Acceptable latency for application
 - Consistent results

Architecture Overview





Design Tools in SSAS

- Dimension Wizard
 - Does create attributes
 - ...but not attribute relationships and hierarchies
- Cube Wizard
 - Does create one partition
- Aggregation Wizard
 - Does create aggregations
 - ... but not necessarily aggregations satisfying queries

Analysis Services 2008+
includes
Best Practices Warnings
to guide you
to implementation of
optimal design

Aggregations and Queries

| Year | Category | Measure |
|------|----------|---------|
| 2008 | Bikes | xxx |
| 2009 | Bikes | xxx |
| 2010 | Bikes | xxx |
| 2008 | Clothing | xxx |
| 2009 | Clothing | xxx |
| 2010 | Clothing | xxx |

- Aggregations above good for Year-Category queries
- But what about...
 - Quarter-Category
 - Year-Subcategory



Agenda

- Introduction
- Designing for Query Performance
- Designing for Processing Performance

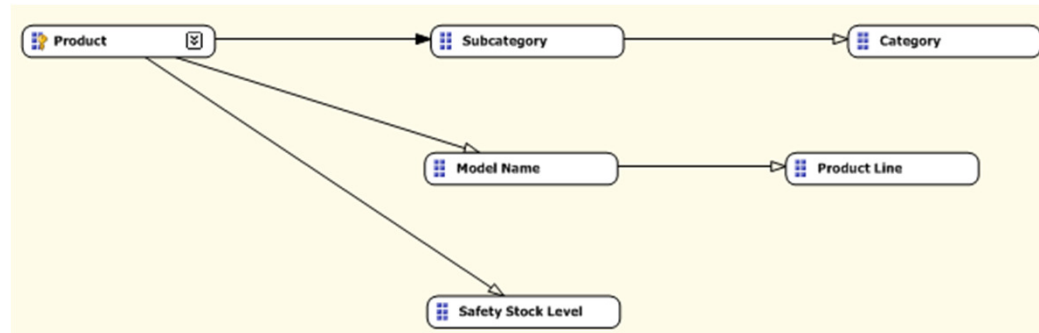
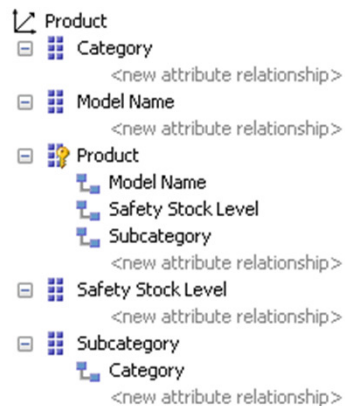


Best Practice Dimension Design for Queries

- Attribute Relationships
- Hierarchies
- Parent-child Dimensions

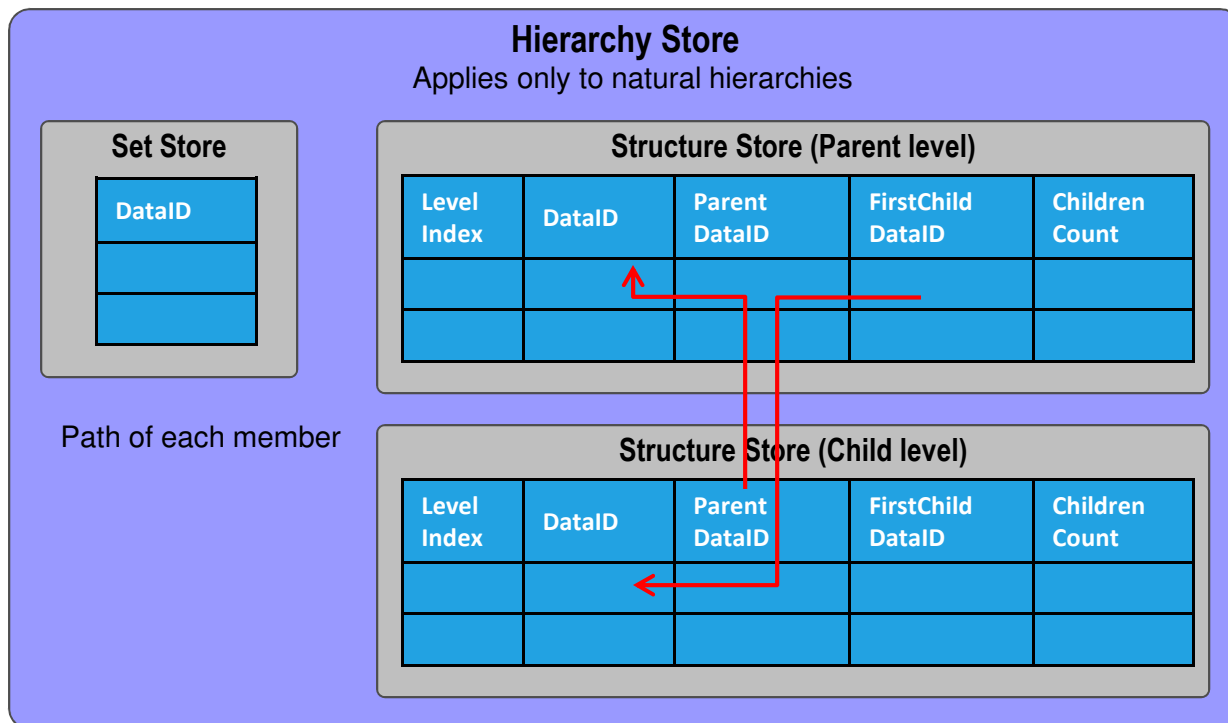
Attribute Relationships

- Define cascading attribute relationships for natural hierarchies (many-to-one)
 - Identifies aggregation candidates
- Remove redundant relationships
 - Helps formula execution engine generate best query plan



Hierarchies

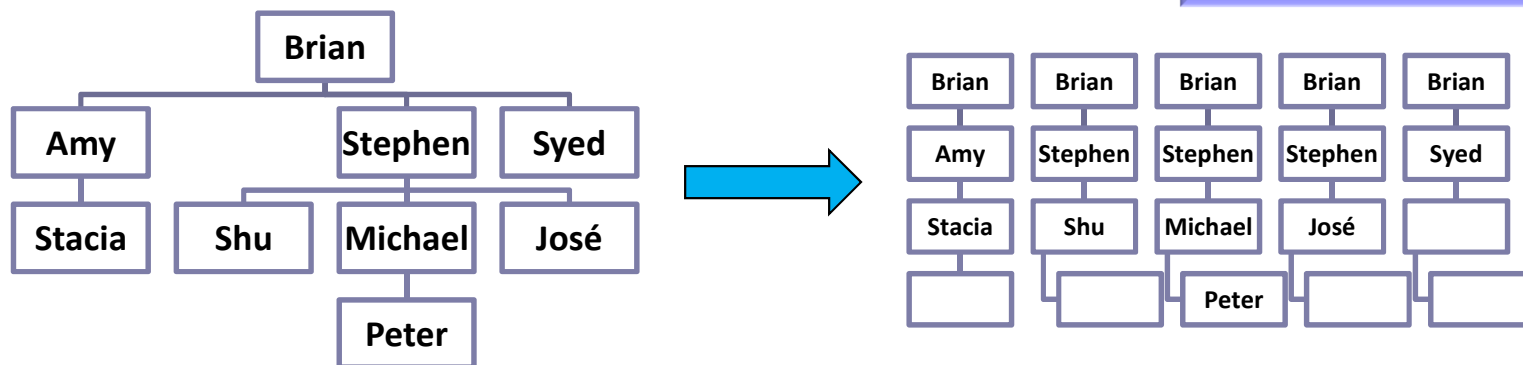
- Definition of hierarchy defines path from member to member
 - Materializes on disk for faster query access



Parent-Child Dimensions

- Avoid using parent-child dimensions if possible
 - Supports aggregations only at leaf level and All level
- Consider flattening the structure to fixed number of levels
 - Enables aggregation design at all levels

Trade-off: Flat structure imposes limit on number of levels supported in dimension





Best Practice Aggregation Design

- Design aggregations as part of initial cube design
 - Potentially optimizes queries
- Add Usage-Based Optimization
 - Tunes aggregations based on actual query patterns

AggregationUsage Configuration

- Set this property to define aggregation candidates in dimension

| Value | Result | Default Rule |
|--------------|--|---|
| Full | Every agg must include this attribute or a lower-level attribute | Non-aggregatable attributes |
| None | No agg contains this attribute | Attributes in many-to-many dimension Non-materialized dimensions Data mining dimensions All other attributes not covered elsewhere |
| Unrestricted | Attribute is agg candidate | Key attribute “All” attribute Attributes in natural hierarchies |



Best Practice Measure Group Design

- Design fact table to include all measures queried together
 - Reduces number of requests to storage engine
- Design separate fact tables for measures not queried together
 - Optimizes cache storage

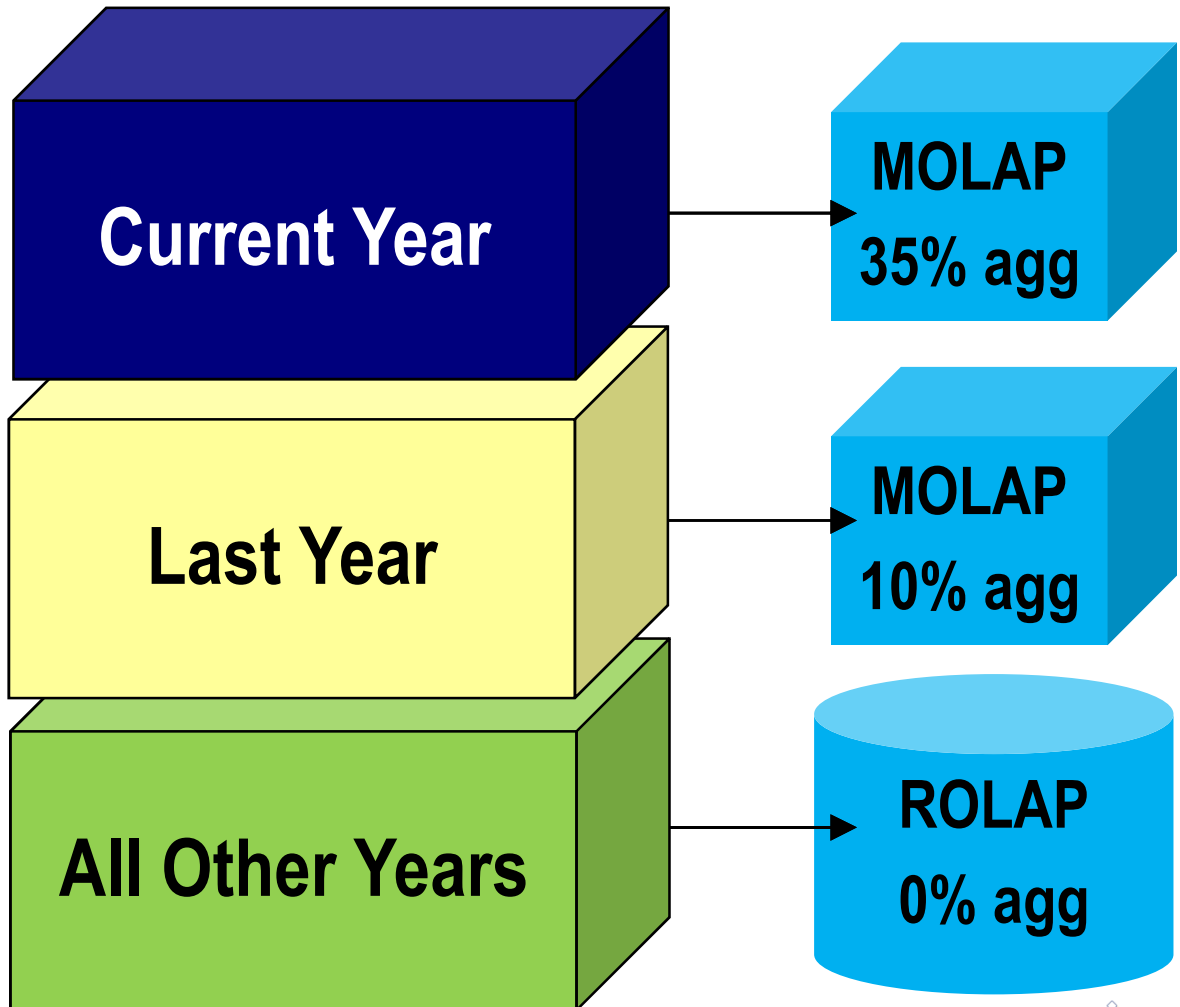


Partitions

- Physical storage of cube data
 - Associated with one measure group only
 - Transparent to users and client applications
 - Multiple partitions supported in Enterprise Edition only
- Performance benefits
 - Better cube processing due to smaller increments of data
 - Better query performance due to smart queries and parallelism

Partitioning Strategies

- Every OLAP cube has at least one partition
- Multiple partitions can be managed with different storage, aggregations, and data slices





Creating a Partition

- Tools
 - Use BIDS to create manually on Partitions tab of Cube Designer
 - Use XMLA script to create in SSMS query or SSIS package
- Technique
 - One fact table per partition
 - One fact table for all partitions using WHERE clause to define rows assigned to each partition



Best Practice Partition Design

- Partition Sizing
 - Between 2 million and 20 million records per partition
 - Between 50 MB and 250 MB partition size
 - But...fewer than 2,000 partitions per measure group
- Partition Content
 - Partition “slice” matches popular queries (e.g. by time period)



Agenda

- Introduction
- Designing for Query Performance
- Designing for Processing Performance



Dimension Source Query Design Issues

- Use OLE DB providers instead of .NET providers for better performance
- Set cascading attribute relationships for multiple data sources to avoid OPENROWSET queries
- Ensure **KeyColumn** property uniquely identifies each attribute member
- Keep default **ProcessingGroup** property – ByAttribute for better performance



Dimension Design for Processing

- Remove unnecessary attributes and define cascading relationships for better performance
- Use a numeric value for **KeyColumn** property to reduce storage and processing
- Set **AttributeHierarchyOptimizedState** property = Not Optimized for attributes for member properties to disable bitmap indexing
- Set **AttributeHierarchyEnabled** = False for member properties to disable bitmap indexing and aggregations
- Define rigid attribute relationships (unless parent-child relationships change frequently) to keep aggs in place



Partition Processing Fundamentals

- Analysis Services processing is a balancing act
 - How much data to process?
 - How much time available for processing?
 - What type of storage is required for cube data?
- A good partitioning strategy can help you
 - Limit processing to specific partitions
 - Minimize the amount of data to process per partition
 - Optimize the storage mode for frequently processed partitions



Partition Processing Options

- Process Full
 - Does full rebuild
- Process Clear
 - Empties object
- Process Default
 - Builds only unprocessed objects
- Process Data
 - Discards contents and reloads data (no indexes)
- Process Incremental
 - Loads data using filter
- Process Indexes
 - Preserves data and appends indexes
- Process Update
 - Affects dimensions only
 - Preserves data
 - Performs incremental update (new, changed, or deleted members)
- Process Add
 - Affects dimensions only
 - Adds new members only
- Process Script Cache
 - Evaluates and persists MDX script in cube



Partition Processing Optimization

- Use OLE DB providers instead of .NET providers for better query performance
- Design partition to support refresh strategy
 - Inserts: Use **ProcessAdd** or New partition + **ProcessFull**
 - Updates: Add delta values for data changes + **ProcessAdd**



Aggregation Processing Optimization

- Reprocess flexible aggregations following incremental update of dimension
 - Potentially improves query performance
- Set **ProcessingMode** = LazyAggregations (cube, measure group, partition)
 - Makes cube available for queries as processing continues
- Set **LazyProcessing\MaxObjectsInParallel** server property
 - Reduces time required for processing through parallelization



Key Points to Remember

- Design of database objects has consequences
 - Storage engine's query performance
 - Time required to process
- Focus first on design strategies to improve query performance
 - OLAP is more useful to users when it's fast
 - Monitor query patterns periodically to match aggregation design to current usage

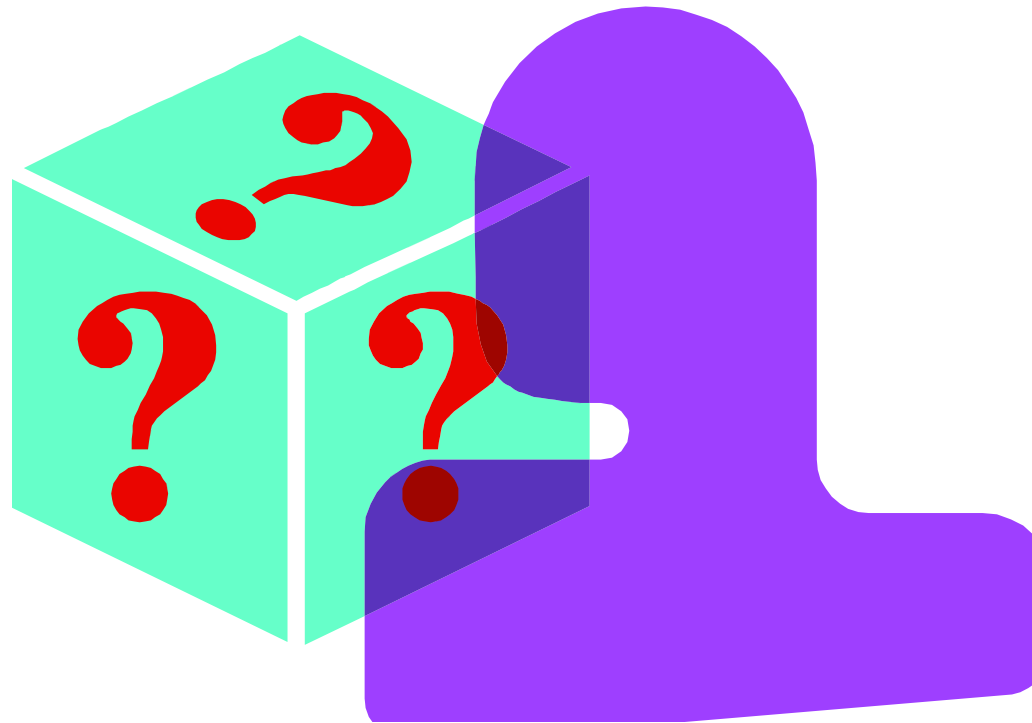


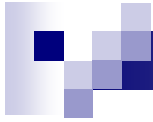
For More Information

- Analysis Services 2008 Performance Guide
 - <http://tinyurl.com/266zwgt>
- Microsoft SQL Server 2008 Analysis Services Unleashed by Irina Gorbach, Alexander Berger, and Edward Melomed
 - <http://tinyurl.com/3usd6hh>



Q & A





Data Inspirations

Inspiring Intelligence From Information

Stacia Misner
smisner@datainspirations.com
(702) 561-7118

Twitter: @StaciaMisner
blog.datainspirations.com