SSAS - Designing, Development and Deployment Best practices

Amit Bansal

www.PeoplewareIndia.com

www.SQLServerGeeks.com

Microsoft tech·ed India 2011

MARCH 23-25 | BANGALORE

About Amit Bansal

- CTO, eDominer Systems & Peopleware India
- Corporate Trainer/Consultant & Evangelist
- Conducted more than 400 workshops on SQL Server & BI for top notch IT companies world wide
- Microsoft MVP for SQL Server
- Microsoft Certified Trainer Advisory Council member
- Speaker at TechED India, TechED US & TechED Europe
- Technical Reviewer – MSL courses on SQL Server
- SME – SQL Server 2008 certifications
- President – SQLServerGeeks.com

Agenda

- Problem Statement
- Dimension Design
- Cube Design
- Partitioning
- Aggregations
- Summary

Problem Statement

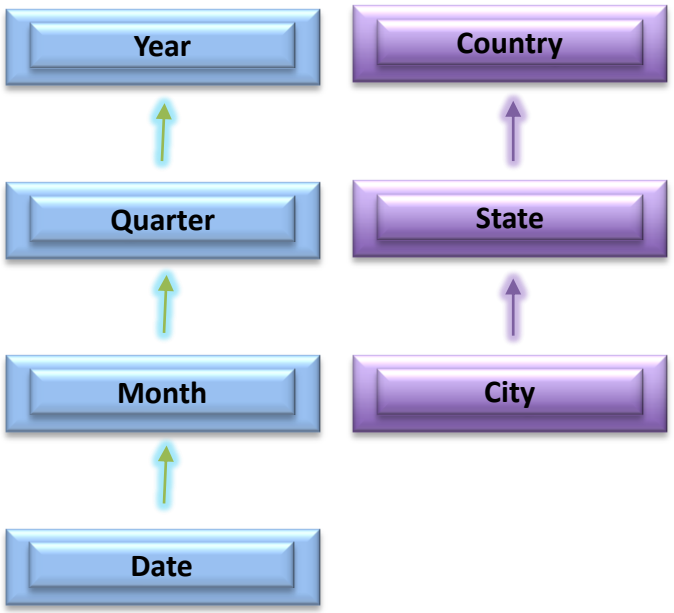
- You want:
  - Faster initial development
  - Easier further development
  - Easier maintenance
  - Agility and scalability in your design
  - Performance, Performance, Performance
- You need to design best, right from start ! (Do you I really need to tell you this ☺)

Agenda

- Problem Statement
- Dimension Design
- Cube Design
- Partitioning
- Aggregations
- Summary

Natural Hierarchies

- A hierarchy is a natural hierarchy when each attribute included in the user-defined hierarchy has a one to many relationship with the attribute immediately below it (every child member has only one parent)
- Server simply “works better”

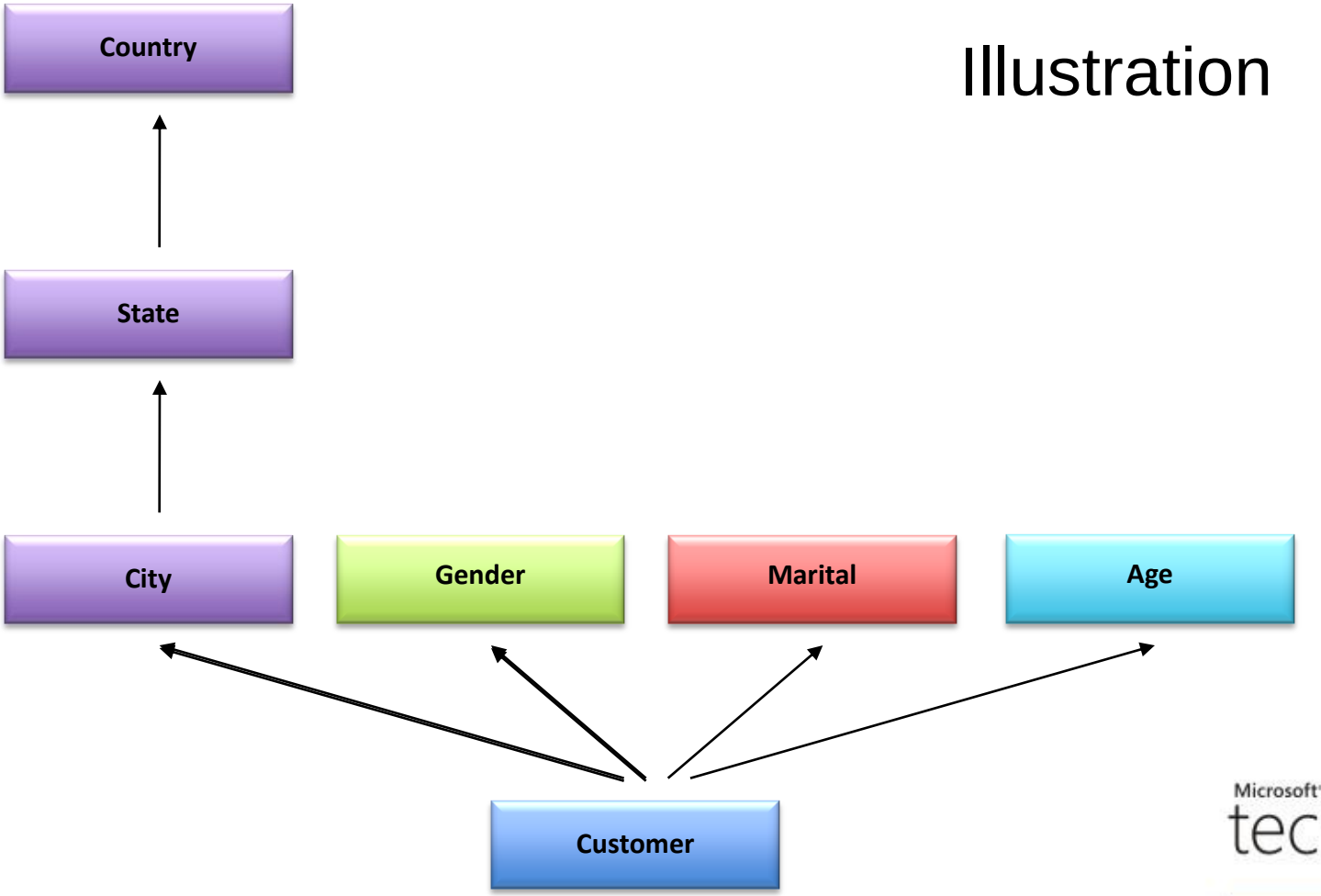


Natural Hierarchies

- Performance implications
  - Only natural hierarchies are materialized on disk during processing
  - Unnatural hierarchies are built on the fly during queries (and cached in memory)
  - Server internally decomposes unnatural hierarchies into natural components
  - Essentially operates like ad hoc navigation path (but somewhat better)
  - Aggregation designer favors user defined hierarchies

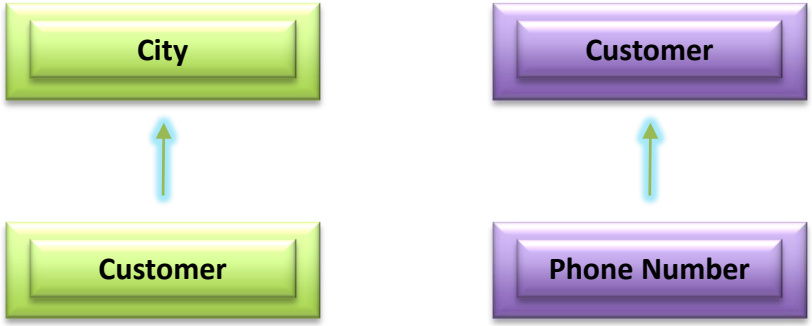
Attribute Relationships

Illustration

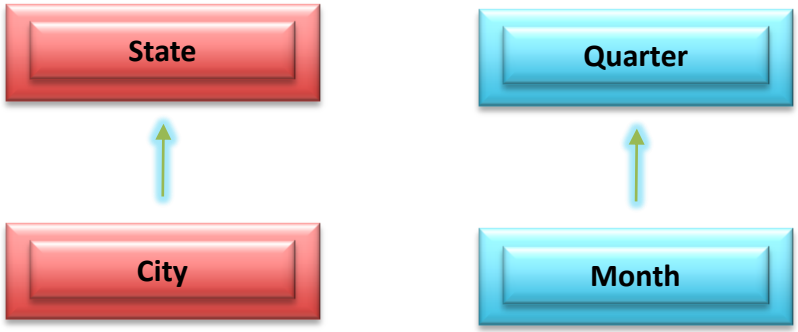


Attribute Relationships

- Flexible relationships can change



- Rigid relationships do not change



Attribute Relationships

- Where are they used?
  - Storage
    - Query performance
      - Greatly improved effectiveness of in-memory caching
      - Materialized hierarchies when present
    - Processing performance: Fewer, smaller hash tables result in faster, less memory intensive processing
    - Aggregation design: Algorithm needs relationships in order to design effective aggregations
    - Member properties: Attribute relationships identify member properties on levels

Attribute Relationships

- Where are they used?
  - Semantics
    - MDX overwrite semantics: City.Seattle State. WA | State.OR City.All
    - Non-key granularity (Aggregation Paths)
    - Dimension security: DeniedSet = {State.WA}

DEMO

Attribute relationships & Natural hierarchies

Dealing with Large Dimensions

- Optimizing Processing
  - Use natural hierarchies
    - Good attribute/hierarchy relationships forces the AS engine to build smaller DISTINCT queries versus one large and expensive query
    - Consider size of other properties/attributes
  - Dimension SQL queries are in the form of

```
select distinct Key1, Key2, Name, …,
RelKey1, RelKey2, …
from [DimensionTable]
```

Dealing with Large Dimensions

- Important to tune your SQL statements
  - Indexes to underlying tables
  - Create a separate table for dimensions
  - Avoid OPENROWSET queries
  - Use Views to create your own version of “query binding”
- Size limitations for string stores and effect on dimension size
  - 4 GB, stored in Unicode, 6 byte per-string overhead.
  - E.g. 50-character name: 4*1024*1024*1024 / (6+50*2) = 40.5 million members

Dimension Processing

- ByAttribute vs ByTable
  - This is a `ProcessingGroup` property
    - Default = `ByAttribute`
  - Advantages of `ByTable`
    - Entire set of dimension data loaded into memory
    - Theoretically processes data faster
    - But BEWARE
      - Bypasses normal checks
      - Assumes there is enough memory to process all attributes concurrently
      - If this is not true…

Dimension Processing

- ByAttribute vs ByTable
  - 2 dimensions
    - Each >25M members with 8-10 attributes
  - `ByTable`
    - Took 80% of available memory
    - 25.6 / 32 GB
    - Never completed
  - `ByAttribute`
    - Only 28% of available memory
    - 9 / 32 GB
    - Process completed

Agenda

- Problem Statement
- Dimension Design
- Cube Design
- Partitioning
- Aggregations
- Summary

Cube Dimensions

- Dimensions
  - Consolidate multiple hierarchies into single dimension (unless they are related via fact table)
  - Use role-playing dimensions (e.g., OrderDate, BillDate, ShipDate)—avoids multiple physical copies
  - Use parent-child dimensions prudently
    - No aggregation support
  - Set Materialized = true on reference dimensions

Cube Dimensions

- Dimensions
  - Use many-to-many dimensions prudently
    - Slower than regular dimensions, but faster than calculations
    - Intermediate measure group must be “small” relative to primary measure group
    - Consider creating aggregations on the shared common attributes of the intermediate measure group

Measure Groups

- Common questions
  - At what point do you split from a single cube and create one or more additional cubes?
  - How many is too many?
- Why is this important?
  - New measure groups adding new dimensions result in an expansion of the cube space
  - Larger calculation space = more work for the engine when evaluating calculations

Measure Groups

- Guidance
  - Look at increase in dimensionality. If significant, and overlap with other measure groups is minimal, consider a separate cube
  - Will users want to analyze measures together?
  - Will calculations need to reference unified measures collection?

DEMO

Cube Design Best Practices

Agenda

- Problem Statement
- Dimension Design
- Cube Design
- Partitioning
- Aggregations
- Summary

Why Partition?

- Breaks large cubes into manageable chunks
- For measure groups, not dimensions
- Fact rows are distributed by a partitioning scheme
  - Managed by DBA
  - By Time: Sales for 2001, 2002, 2003, …
  - By Geography: Sales for North America, Europe, Asia, …
- Why?
  - For Manageability, Performance, Scalability

Benefits of Partitioning

- Partitions can be added, processed, deleted independently
  - Update to last month's data does not affect prior months' partitions
  - Sliding window scenario easy to implement
  - e.g., 24 month window → add June 2006 partition and delete June 2004
- Partitions can have different storage settings
  - Storage mode (MOLAP, ROLAP, HOLAP)
  - Aggregation design
  - Alternate disk drive
  - Remote server

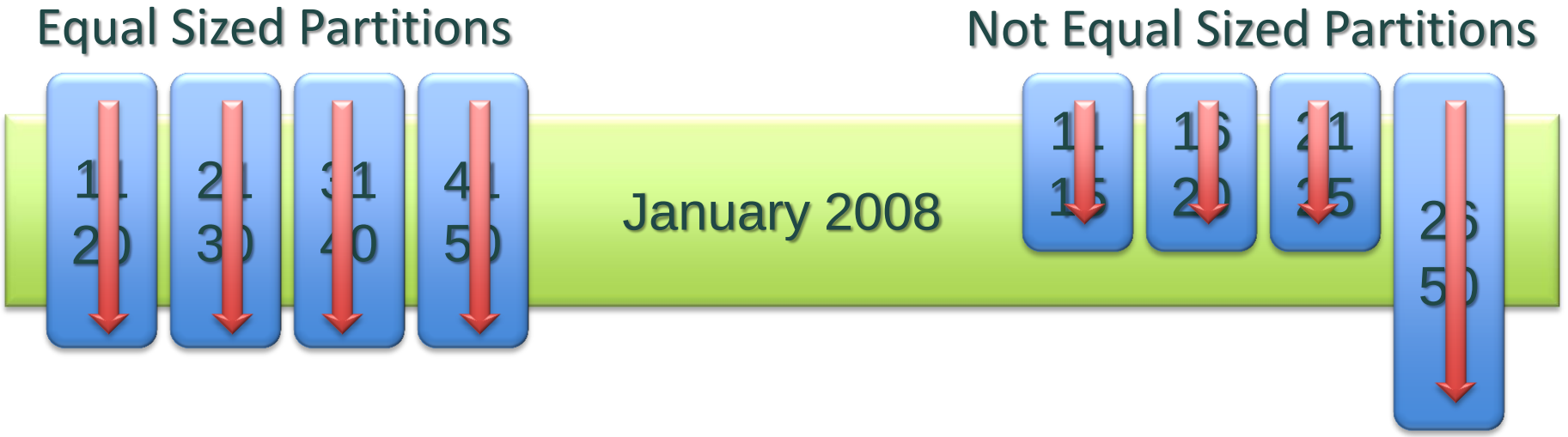
Benefits of Partitioning

- Partitions can be processed and queried in parallel
  - Better utilization of server resources
  - Reduced data warehouse load times
- Queries are isolated to relevant partitions → less data to scan
  - SELECT … FROM… WHERE [Time].[Year].[2006]
  - Queries only 2006 partitions
- Bottom line → partitions enable
  - Manageability, Performance & Scalability

Best Practices for Partitions

- General guidance: 20M rows per partition
  - Use judgment, e.g., perhaps better to have 500 partitions with 40 million rows than 1000 20 million row partitions
  - Standard tools unable to manage thousands of partitions
- More partitions means more files
  - E.g. one 10GB cube with ~250,000 files (design issues)
  - Deletion of database took ~25min to complete
- Partition by time plus another dimension e.g. Geography
  - Limits amount of reprocessing
  - Use query patterns to pick another partitioning attribute
- When data changes
  - All data cache for the measure group is discarded
  - Separate cube or measure groups by “static” and “real-time” analysis

Best Practices for Partitions



DEMO

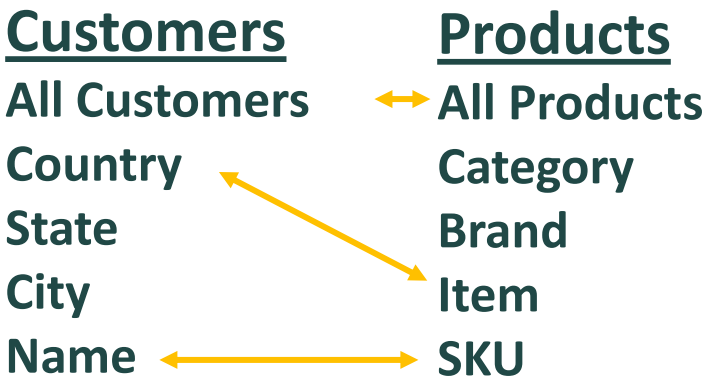
Partitioning

Agenda

- Problem Statement
- Dimension Design
- Cube Design
- Partitioning
- Aggregations
- Summary

Aggregations for query performance

- A subtotal of partition data
  - based on a set of attributes from each dimension



Highest-Level Aggregation

Customer	Product	Units Sold	Sales
All	All	347814123	$345,212,301.30

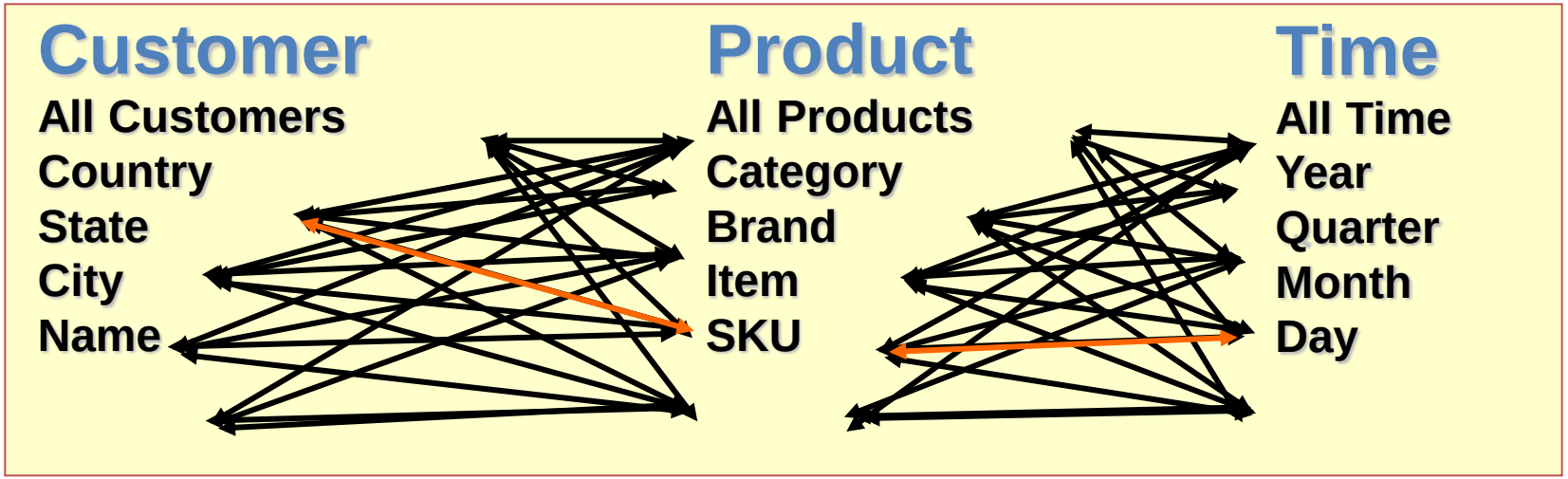
Intermediate Aggregation

countryCode	productID	Units Sold	Sales
Can	sd452	9456	$23,914.30
US	yu678	4623	$57,931.45
…			

Facts

custID	SKU	Units Sold	Sales
345-23	135123	2	$45.67
563-01	451236	34	$67.32
…			

How many Aggregations



- 125 possible combinations (just for user-defined dimensions)
  - 5 customer levels, 5 product levels, 5 time levels
- Imagine a cube with ten dimensions, five levels each
  - 9,765,625 combinations! Then you add attribute hierarchies to the mix
- General rule: multiply the number of attributes in each dimension
- **Goal should be to find the best subset of this potentially huge number of possibilities**
  - **Tradeoff between query performance and processing/storage overhead**

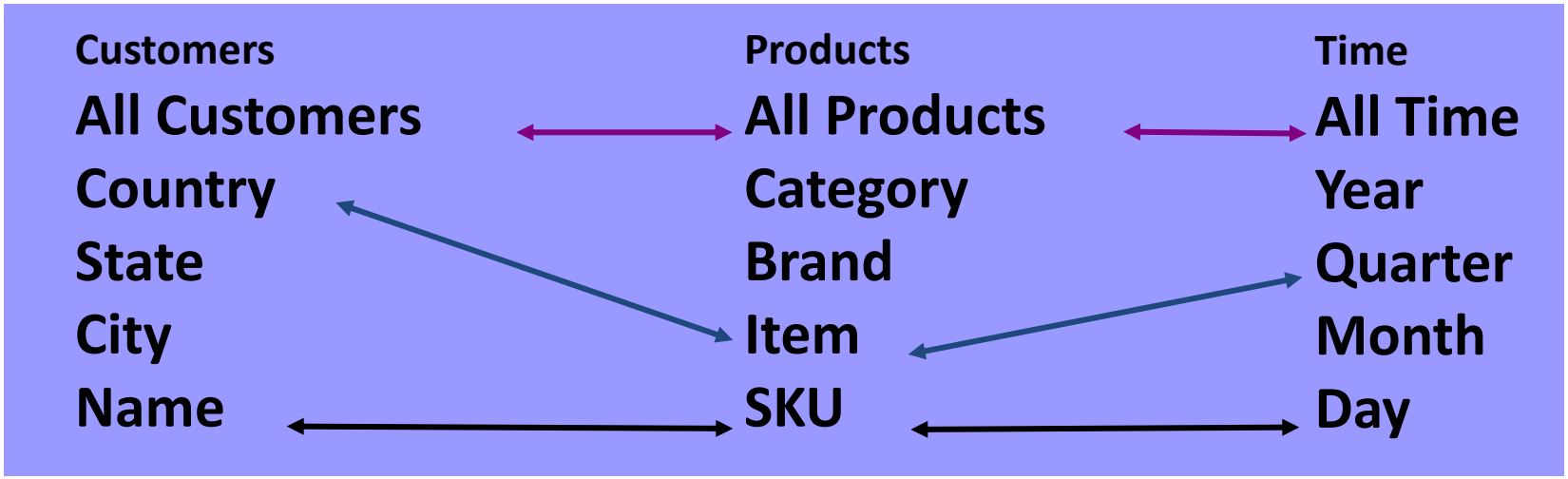
Aggregations for query performance

- Aggregations at lower levels have more possible rows…

(All, All, All)	1 x 1 x 1	= 1
(Country, Item, Quarter)	3 x 7621 x 12	= 274,356
(Name, SKU, Day)	3811 x 8211 x 1095	= 34,264,872,495

- Actual number of rows depends on the data sparsity
- Size also depends on the number of measures

Aggregations for query performance



Query levels	Aggregation used	Max Cells
(All, All, All)	(All, All, All)	1
(Country, Item, Quarter)	(Country, Item, Quarter)	274,356
(Country, Brand, Quarter)	(Country, Item, Quarter)	274,356
(Country, Category, All)	(Country, Item, Quarter)	274,356
(State, Item, Quarter)	(Name, SKU, Day)	34,264,872,495
(City, Category, Year)	(Name, SKU, Day)	34,264,872,495

Using a higher-level aggregation means fewer cells to consider

Best Practices for Aggregations

- Define all possible attribute relationships
- Set accurate attribute member counts and fact table counts
- Set `AggregationUsage`
  - Set rarely queried attributes to `None`
  - Commonly queried attributes to `Unrestricted`

Best Practices for Aggregations

- Not *too* many
  - In the 100s, not 1000s!
- Do not build aggregations
  > 30% of fact table size

Best Practices for Aggregations

1. Use Storage Design Wizard for the initial aggregations (~20% perf gain)
2. Enable query log
3. Run pilot workload with limited users
4. Refine with Usage Based Optimization Wizard
5. Use a larger perf gain (70+%)
6. Reprocess partitions for new aggregations to take effect
7. Periodically use UBO to refine aggregations

DEMO

Aggregations

Agenda

- Problem Statement
- Dimension Design
- Cube Design
- Partitioning
- Aggregations
- Summary

Summary

- Design for performance and scalability from the start
- Some fundamental principles carry through from SQL 7.0
  - Dimension design
  - Partitioning
  - Aggregations
- Critical to properly implement/utilize modeling capabilities introduced in SSAS 2005 and carried forward in 2008
  - Attribute relationships, natural hierarchies
  - Design alternatives: role-playing, many-to-many, reference dimensions, semi-additive measures
  - Flexible processing options
- SSAS 2008 development tools have been redesigned and enhanced to better assist in development of high performance cubes

Resources

- Analysis Services 2005 Processing Architecture
  http://msdn.microsoft.com/en-US/library/ms345142(v=SQL.90).aspx
- Many-to-Many Dimensions in Analysis Services
  http://msdn.microsoft.com/en-US/library/ms345139(v=SQL.90).aspx
- Analysis Services Query Performance Top 10 Best Practices
  http://msdn.microsoft.com/en-US/library/cc966527.aspx
- SQL Server 2008 Analysis Services Performance Guide
  http://msdn.microsoft.com/en-us/library/dd542635(v=SQL.100).aspx

Resources

Software Application Developers

msdn®

http://msdn.microsoft.com/

msdnindia

@msdnindia

Infrastructure Professionals

Microsoft® TechNet

http://technet.microsoft.com/

technetindia

@technetindia

Microsoft® tech·ed India | 2011

MARCH 23-25 | BANGALORE

Microsoft®

Be what's next.™