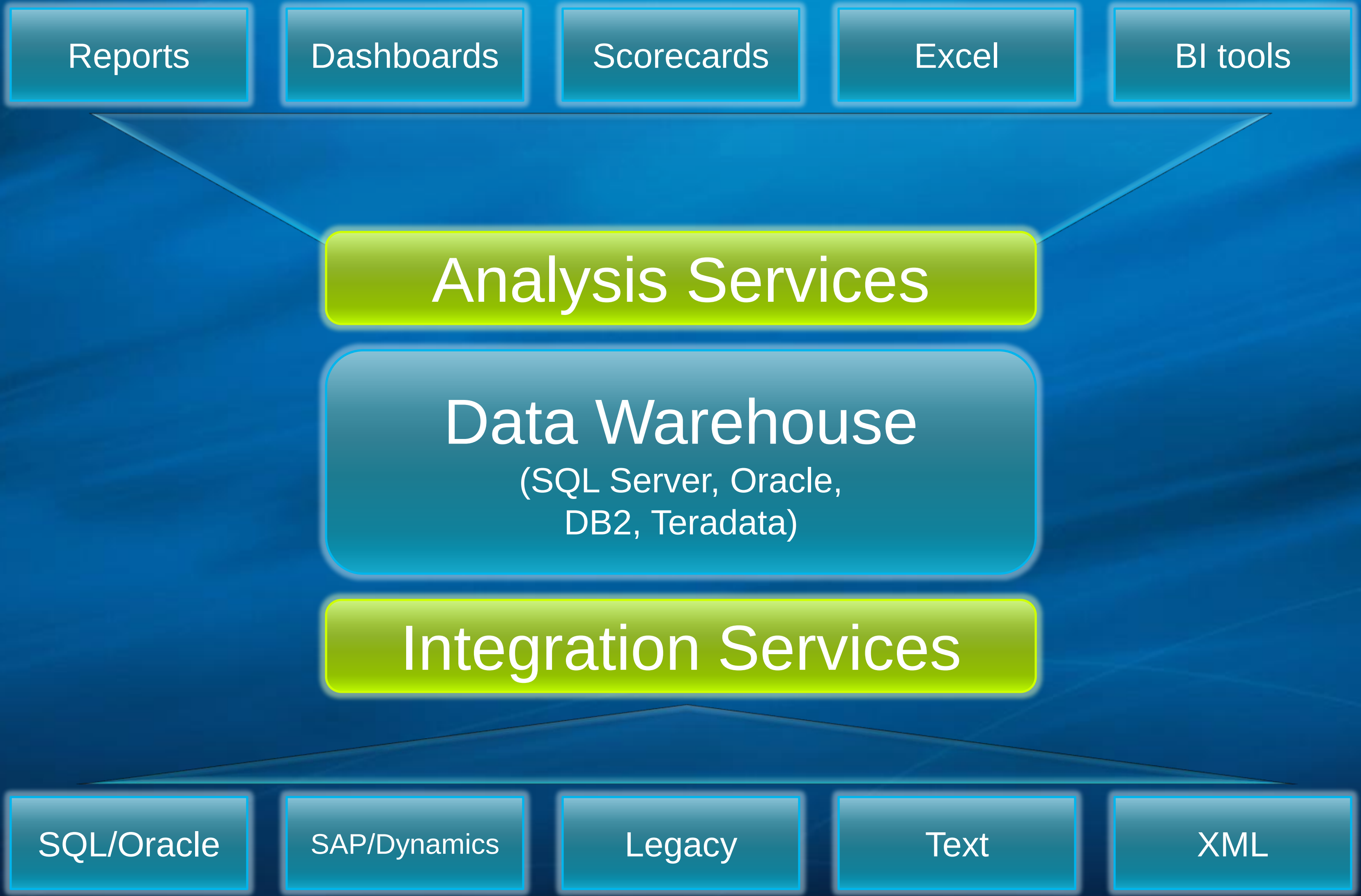
# High impact Data Warehousing with SQL Server Integration Services and Analysis Services

Ashvini Sharma (ashvinis@microsoft.com)
Senior Program Manager/Development Lead
SQL Server
Microsoft

# A DW architecture



Reports

Dashboards

Scorecards

Excel

BI tools

Analysis Services

Data Warehouse

(SQL Server, Oracle,
DB2, Teradata)

Integration Services

SQL/Oracle

SAP/Dynamics

Legacy

Text

XML

# Session Objectives

- Assumptions
  - Experience with SSIS and SSAS
- Goals
  - Discuss design, performance, and scalability for building ETL packages and cubes (UDMs)
  - Best practices
  - Common mistakes

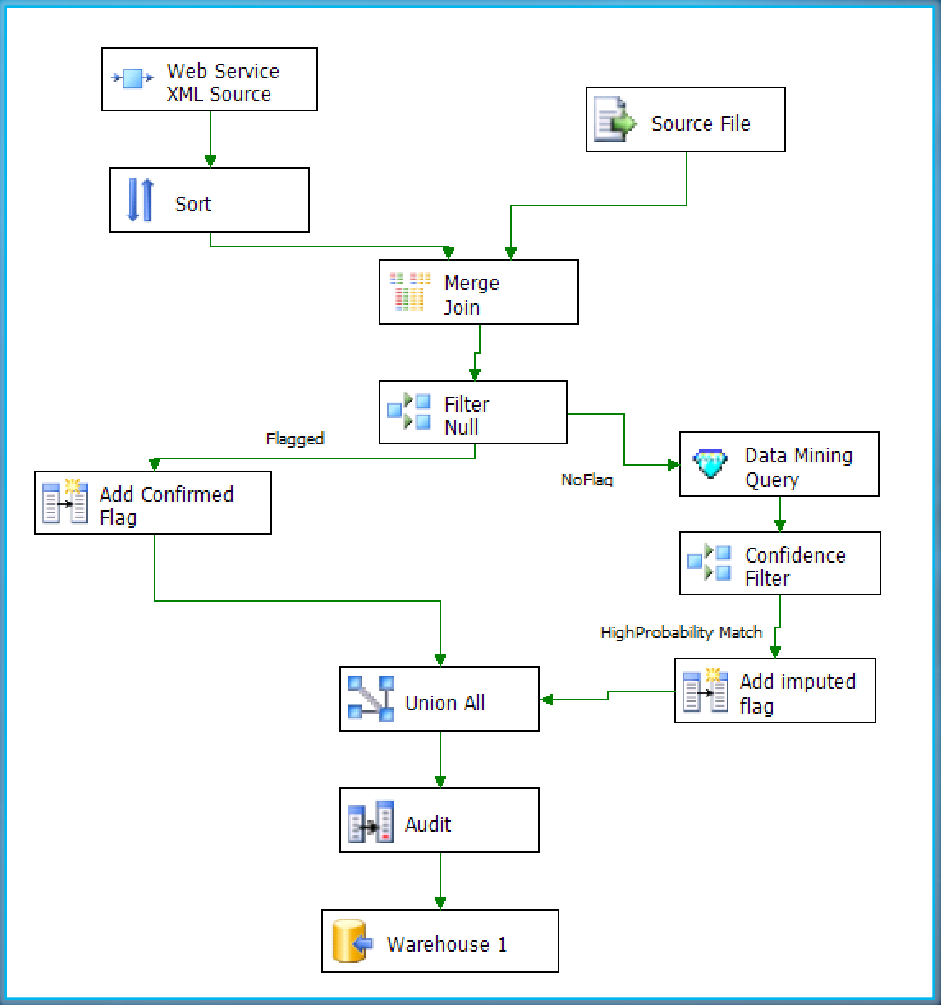
# SQL Server 2005 BPA availability!

- BPA = Best Practice Analyzer
- Utility that scans your SQL Server metadata and recommends best practices
- Best practices from dev team and Customer Support Services
- What’s new:
  - Support for SQL Server 2005
  - Support for Analysis Services and Integration Services
  - Scan scheduling
  - Auto update framework
  - CTP available now, RTM April
- http://www.microsoft.com/downloads/details.aspx?FamilyId=DA0531E4-E94C-4991-82FA-F0E3FBD05E63&displaylang=en

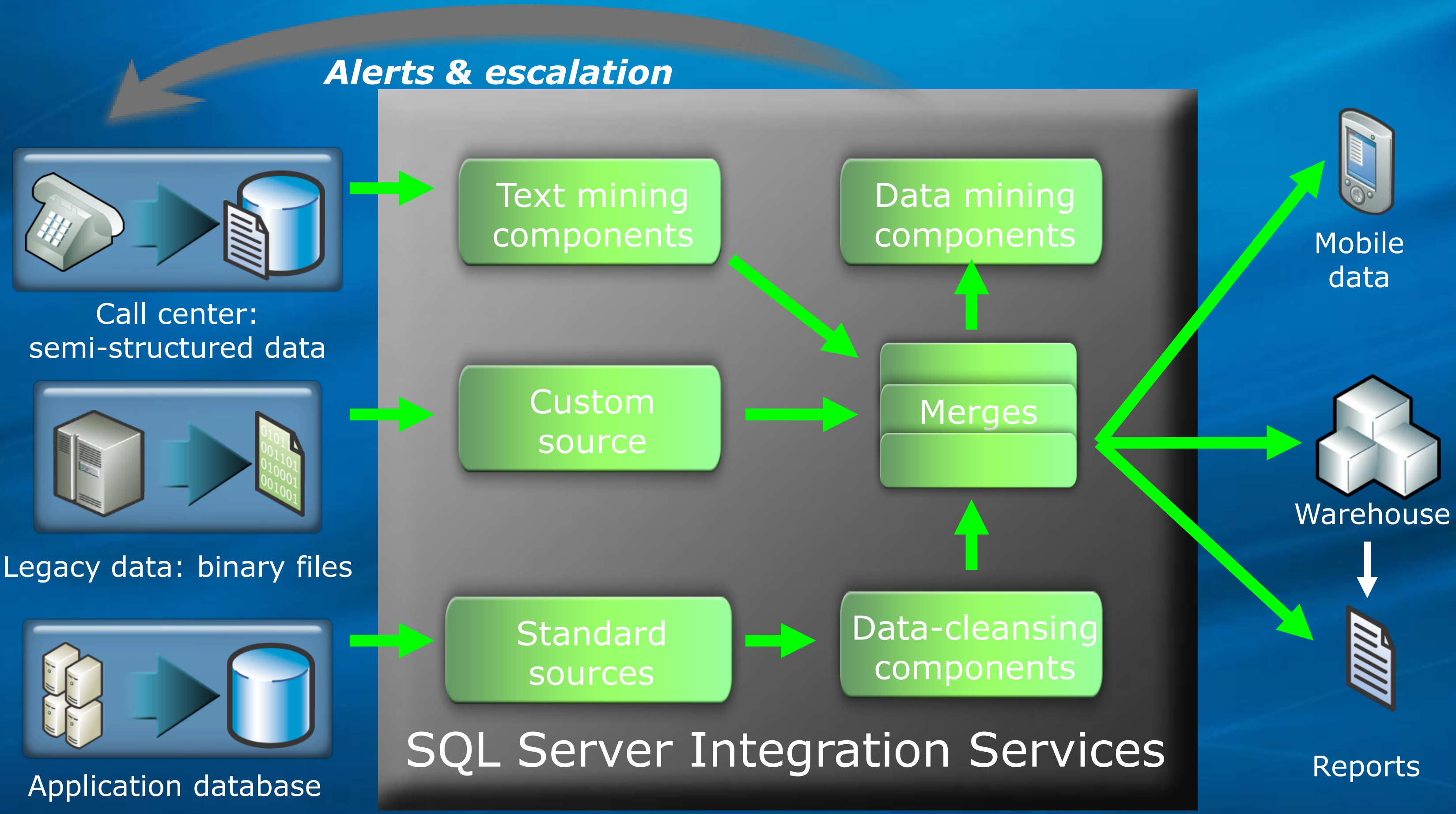
# Integration Services

# What is SQL Server Integration Services?

- Introduced in SQL Server 2005
- The successor to Data Transformation Services
- The platform for a new generation of high-performance data integration technologies



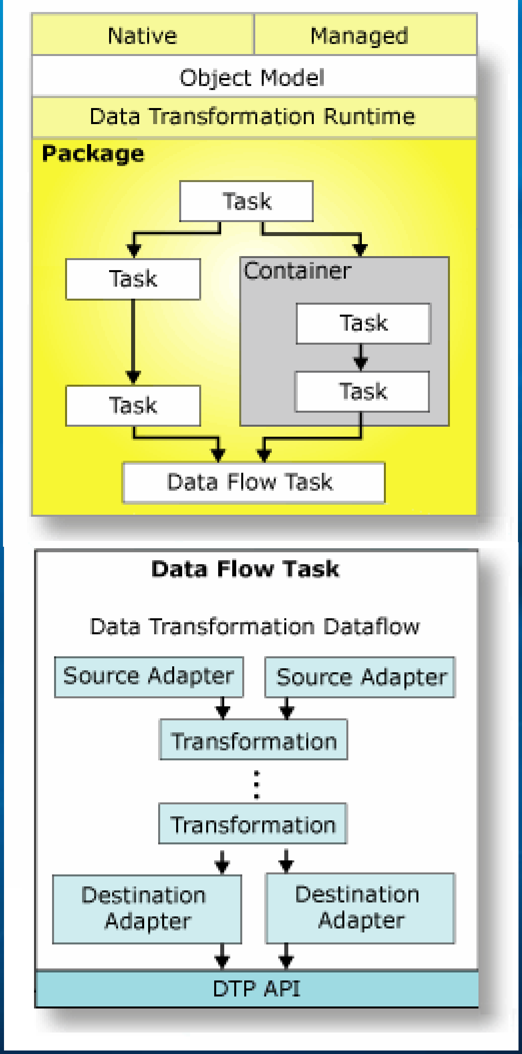
# Changing the Game with SSIS



- Integration and warehousing are a seamless, manageable operation.
- Source, prepare, and load data in a single, auditable process.
- Reporting and escalation can be parallelized with the warehouse load.
- Scales to handle heavy and complex data requirements.

# SSIS Architecture

- Control Flow (Runtime)
  - A parallel workflow engine
  - Executes containers and tasks
- Data Flow ("Pipeline")
  - A special runtime task
  - A high-performance data pipeline
  - Applies graphs of components to data movement
  - Component can be sources, transformations or destinations
  - Highly parallel operations possible



# Agenda

- Overview of Integration Services
- **Principles of Good Package Design**
- Component Drilldown
- Performance Tuning

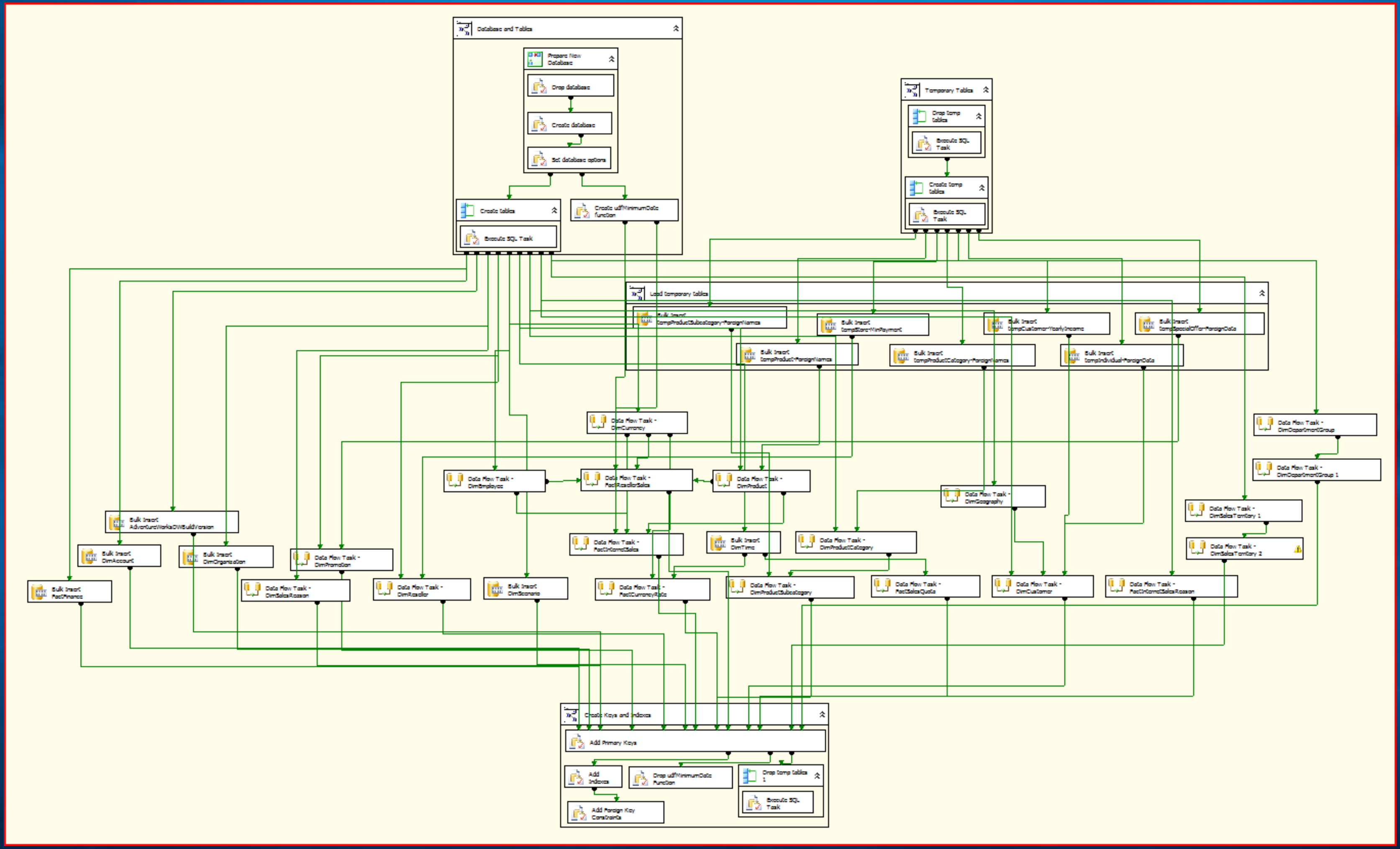
# Principles of Good Package Design - General

- Follow Microsoft Development Guidelines
  - Iterative design, development & testing
- Understand the Business
  - Understanding the people & processes are critical for success
  - Kimball's "Data Warehouse ETL Toolkit" book is an excellent reference
- Get the big picture
  - Resource contention, processing windows, …
  - SSIS does not forgive bad database design
  - Old principles still apply – e.g. load with/without indexes?
- Platform considerations
  - Will this run on IA64 / X64?
    - No BIDS on IA64 – how will I debug?
    - Is OLE-DB driver XXX available on IA64?
  - Memory and resource usage on different platforms
  - SSIS connectivity options at http://ssis.wik.is/Connectivity_Libraries

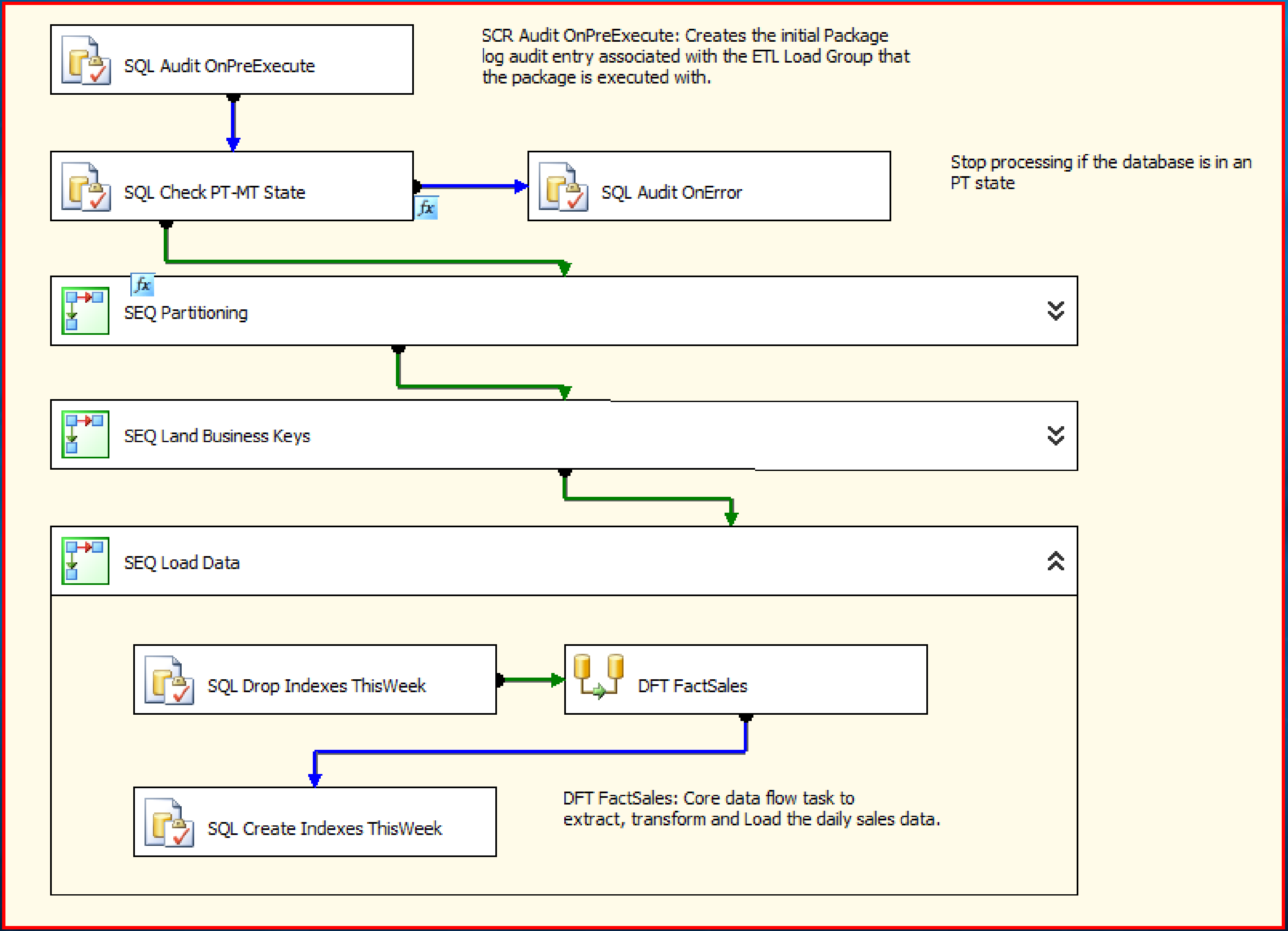
# Principles of Good Package Design - Architecture

- Process Modularity
  - Break complex ETL into logically distinct packages (vs. monolithic design)
  - Improves development & debug experience
- Package Modularity
  - …

# Bad Package Modularity


Database and Tables
Prepare New Database
Drop database
Create database
Set database options
Create tables
Execute SQL Task
Create udfMinimumDate function
Temporary Tables
Drop temp tables
Execute SQL Task
Create temp tables
Execute SQL Task
Load temporary tables
Bulk Insert tempProductSubcategory-ForeignNames
Bulk Insert tempStore-MinPayment
Bulk Insert tempCustomer-YearlyIncome
Bulk Insert tempSpecialOffer-ForeignData
Bulk Insert tempProduct-ForeignNames
Bulk Insert tempProductCategory-ForeignNames
Bulk Insert tempIndividual-ForeignData
Data Flow Task - DimCurrency
Data Flow Task - DimDepartmentGroup
Data Flow Task - DimDepartmentGroup 1
Data Flow Task - DimEmployee
Data Flow Task - FactResellerSales
Data Flow Task - DimProduct
Data Flow Task - DimGeography
Data Flow Task - DimSalesTerritory 1
Data Flow Task - DimSalesTerritory 2
Bulk Insert AdventureWorksDWBuildVersion
Bulk Insert DimAccount
Bulk Insert DimOrganization
Data Flow Task - DimPromotion
Data Flow Task - FactInternetSales
Bulk Insert DimTime
Data Flow Task - DimProductCategory
Bulk Insert FactFinance
Data Flow Task - DimSalesReason
Data Flow Task - DimReseller
Bulk Insert DimScenario
Data Flow Task - FactCurrencyRate
Data Flow Task - DimProductSubcategory
Data Flow Task - FactSalesQuota
Data Flow Task - DimCustomer
Data Flow Task - FactInternetSalesReason
Create Keys and Indexes
Add Primary Keys
Add Indexes
Drop udfMinimumDate Function
Drop temp tables 1
Execute SQL Task
Add Foreign Key Constraints

# Good Package Modularity



SCR Audit OnPreExecute: Creates the initial Package log audit entry associated with the ETL Load Group that the package is executed with.

Stop processing if the database is in an PT state

DFT FactSales: Core data flow task to extract, transform and Load the daily sales data.

# Principles of Good Package Design – Architecture (continued)

- Package Modularity
  - Separate sub-processes within package into separate Containers
  - More elegant, easier to develop
  - Simple to disable whole Containers when debugging
- Component Modularity
  - Use Script Task/Transform for one-off problems
  - Build custom components for maximum re-use

# Principles of Good Package Design - Infrastructure

- Use Package Configurations
  - Build it in from the start
    - Will make things easier later on
  - Simplify deployment Dev → QA → Production
- Use Package Logging
  - Performance & debugging
- Build in Security from the start
  - Credentials and other sensitive info
  - Package & Process IP
  - Configurations & Parameters

# Component Drilldown - Tasks & Transforms

- Maximize Parallelism
  - Allocate enough threads
  - EngineThreads property on DataFlow Task
  - “Rule of thumb” - # of datasources + # of async components
- Minimize blocking
  - Synchronous vs. Asynchronous components
  - Memcopy is expensive – reduce the number of asynchronous components in a flow if possible – example coming up later
- Minimize ancillary data
  - For example, minimize data retrieved by LookupTx

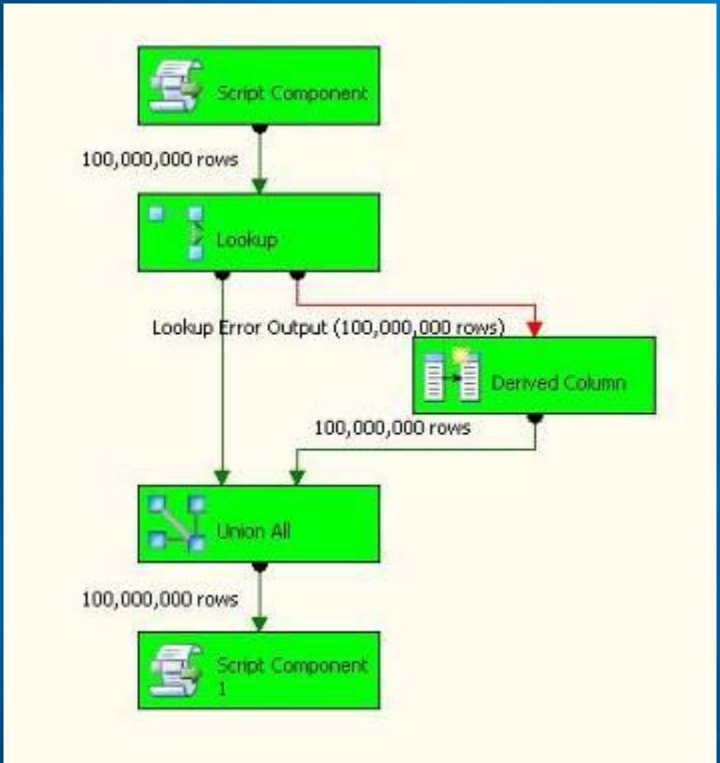
# Debugging & Performance Tuning - Volume

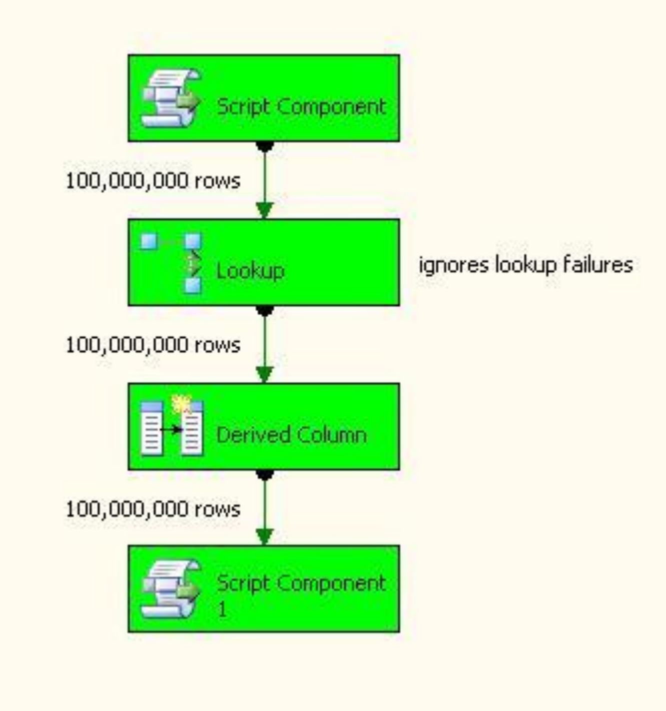
- SSIS is just another client for the data sources
- Remove redundant columns
  - Use SELECT statements as opposed to tables
  - SELECT * is your enemy
  - Also remove redundant columns after every async component!
- Filter rows
  - WHERE clause is your friend
  - Conditional Split in SSIS
  - Concatenate or re-route unneeded columns
- Parallel loading
  - Source system split source data into multiple chunks
    - Flat Files – multiple files
    - Relational – via key fields and indexes
  - Multiple Destination components all loading same table

# Case Study - Patterns

Use Error Output for handling Lookup miss

Ignore lookup errors and check for null looked up values in Derived Column





105 seconds

83 seconds

See Project Real for examples of patterns:
http://www.microsoft.com/sql/solutions/bi/projectreal.mspx

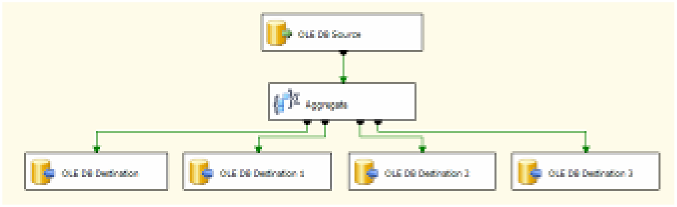
# Debugging & Performance Tuning – A methodology

- Optimize and Stabilize the basics
  - Minimize staging (else use RawFiles if possible)
  - Make sure you have enough Memory
  - Windows, Disk, Network, …
  - SQL FileGroups, Indexing, Partitioning
- Get Baseline
  - Replace destinations with RowCount
  - Source->RowCount throughput
  - Source->Destination throughput
- Incrementally add/change components to see effect
  - This could include the DB layer
  - Use source code control!
- Optimize slow components for resources available

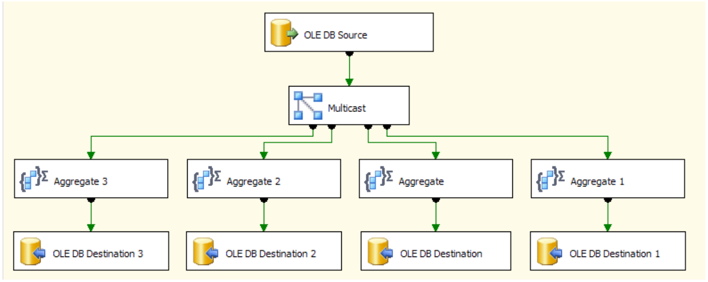
# Case Study - Parallelism

- Focus on critical path
- Utilize available resources

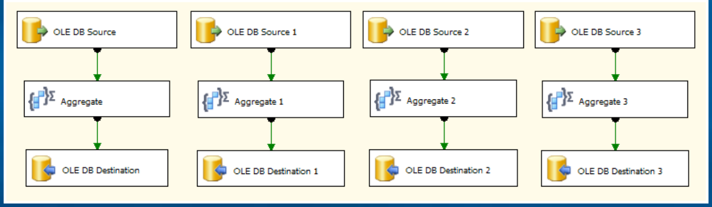
## Memory Constrained



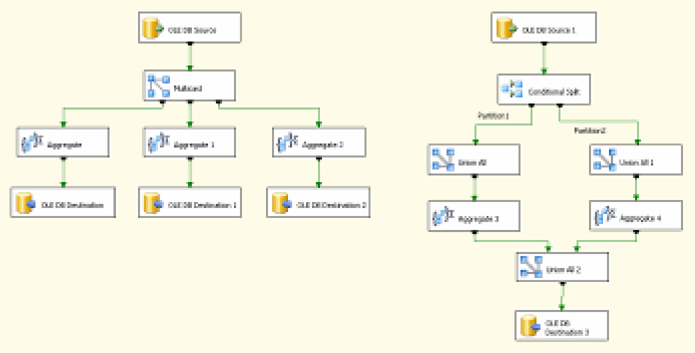
## Reader and CPU Constrained



## Let it rip!



## Optimize the slowest



- Tip: Use Throughput Component in Project REAL to identify slow paths
- Tip:  SSIS Performance whitepaper for more details on CPU, memory usage

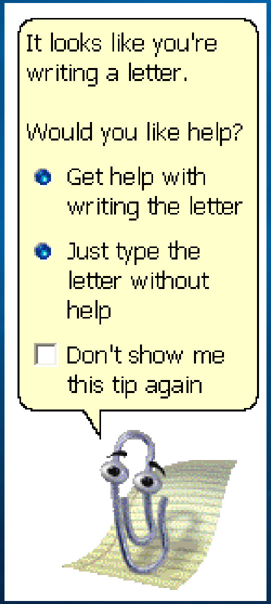
# Summary

- Follow best practice development methods
- Understand how SSIS architecture influences performance
  - Buffers, component types
  - Design Patterns
- Learn the new features
  - But do not forget the existing principles
- Use the native functionality
  - But do not be afraid to extend
- Measure performance
  - Focus on the bottlenecks
- Maximize parallelism and memory use where appropriate
  - Be aware of different platforms capabilities (64bit RAM)
- Discipline is key

# Analysis Services

# Announcing…

Clippy® for Business Intelligence!

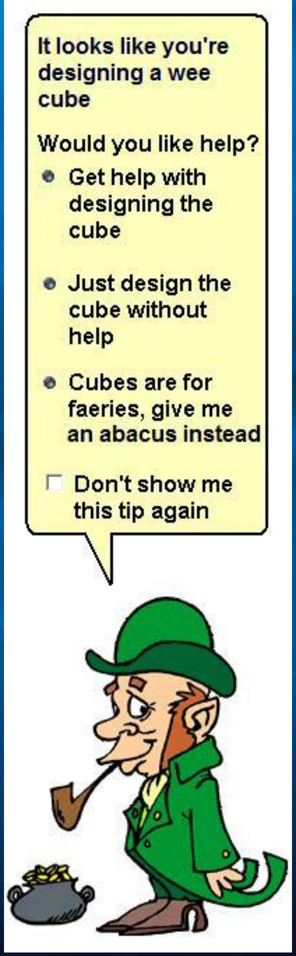


It looks like you're writing a letter.

Would you like help?

- ● Get help with writing the letter
- ● Just type the letter without help

□ Don't show me this tip again



It looks like you're designing a wee cube

Would you like help?

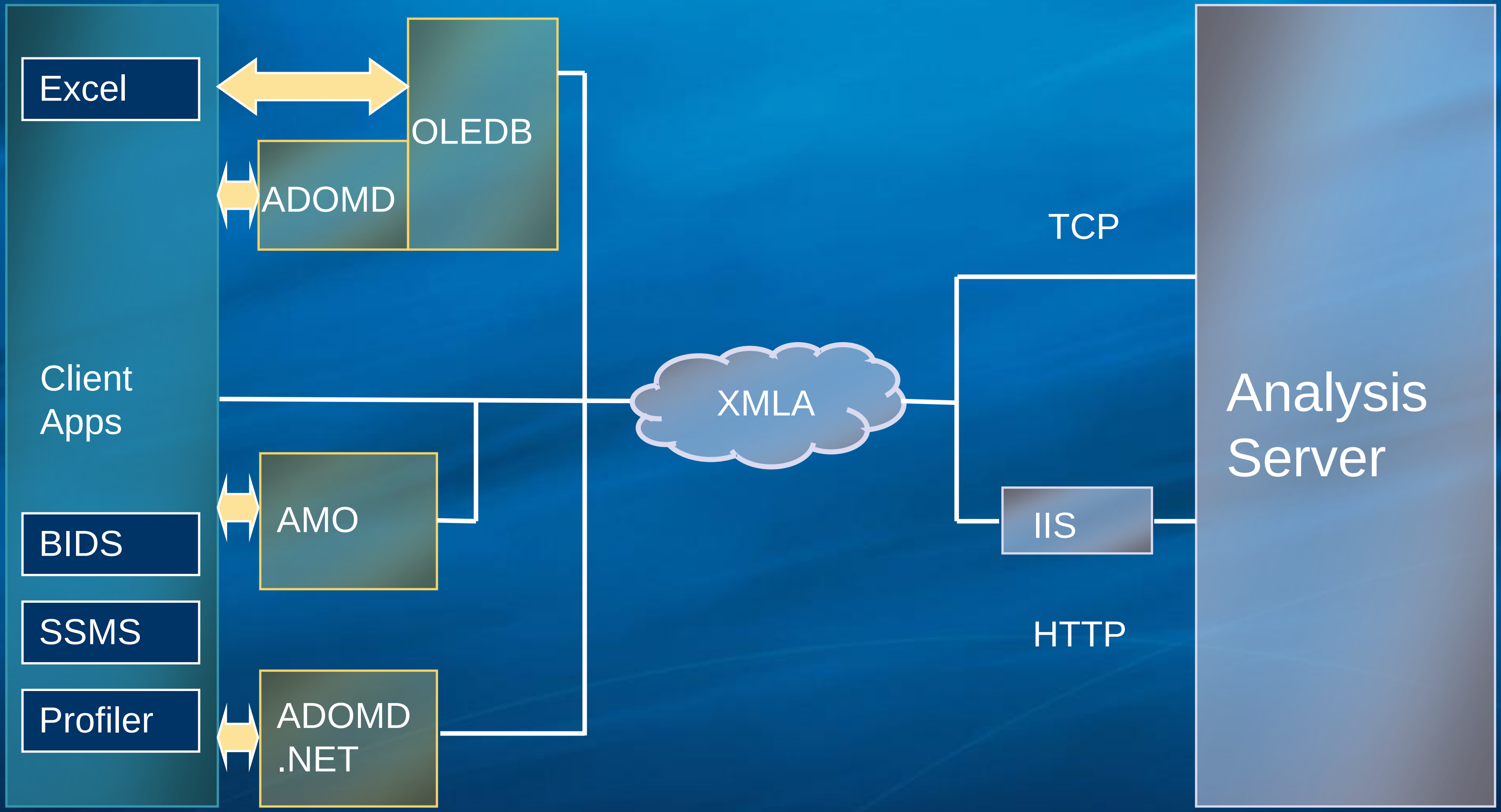
- ● Get help with designing the cube
- ● Just design the cube without help
- ● Cubes are for faeries, give me an abacus instead

□ Don't show me this tip again

# Agenda

- Server architecture and UDM basics
- Optimizing the cube design
- Partitioning and Aggregations
- Processing
- Conclusion

# Client Server Architecture



Excel

OLEDB

ADOMD

Client Apps

BIDS

SSMS

Profiler

AMO

ADOMD .NET

XMLA

TCP

IIS

HTTP

Analysis Server

# Dimension

- An entity on which analysis is to be performed (e.g. Customers)
- Consists of:
  - Attributes that describe the entity
  - Hierarchies that organize dimension members in meaningful ways

| Customer ID | First Name | Last Name | State | City | Marital Status | Gender | … | Age |
|---|---|---|---|---|---|---|---|---|
| 123 | John | Doe | WA | Seattle | Married | Male | … | 42 |
| 456 | Lance | Smith | WA | Redmond | Unmarried | Male | … | 34 |
| 789 | Jill | Thompson | OR | Portland | Married | Female | … | 21 |

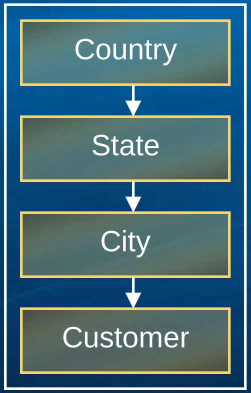
# Attribute

- Containers of dimension members.
- Completely define the dimensional space.
- Enable slicing and grouping the dimensional space in interesting ways.
  - Customers in **Dubai** and **age > 50**
  - Customers who are **married** and **eat Shawarma**
- Typically have one-many relationships
  - City → State, State → Country, etc.
  - All attributes implicitly related to the key

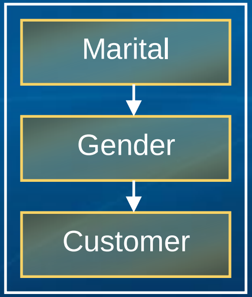
# Hierarchy

- Ordered collection of attributes into levels
- Navigation path through dimensional space
- User defined hierarchies – typically multiple levels
- Attribute hierarchies – implicitly created for each  attribute – single level

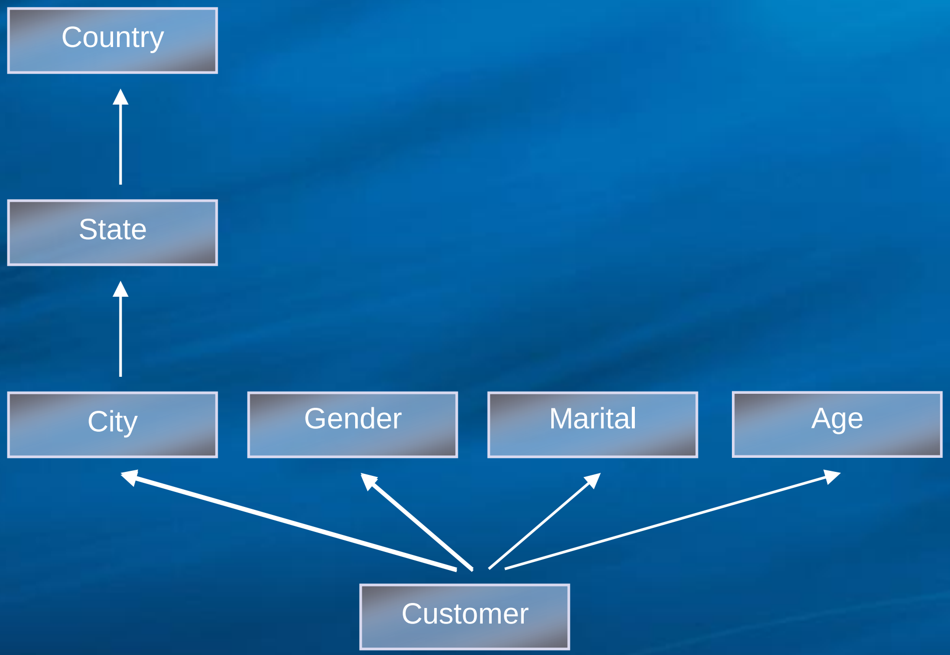
Customers by Geography



Customers by Demographics



# Dimension Model



Country

State

City

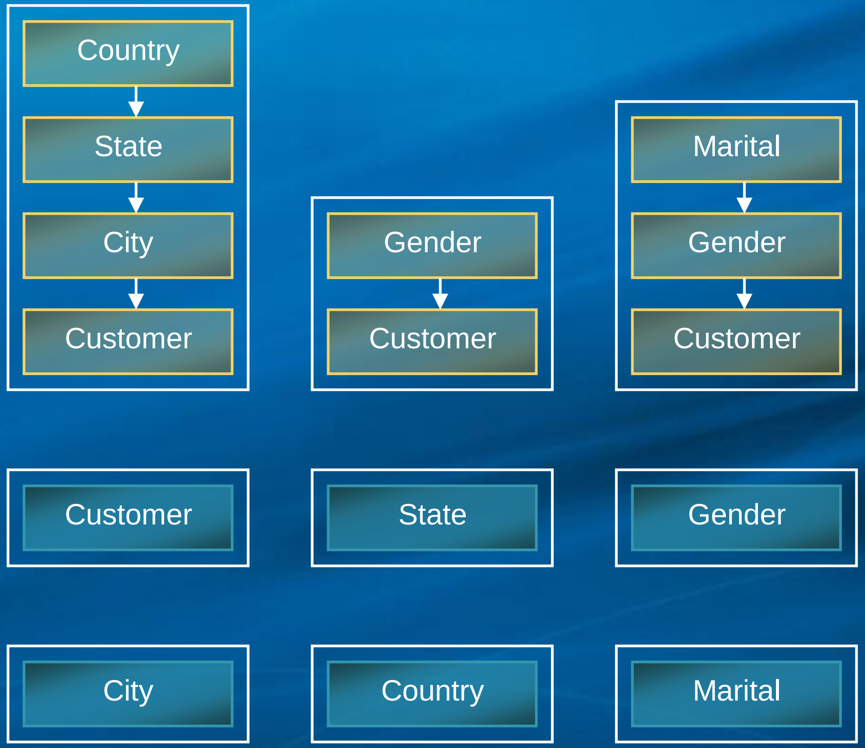
Gender

Marital

Age

Customer

Attributes



Country

State

City

Customer

Gender

Customer

Marital

Gender

Customer

Customer

State

Gender

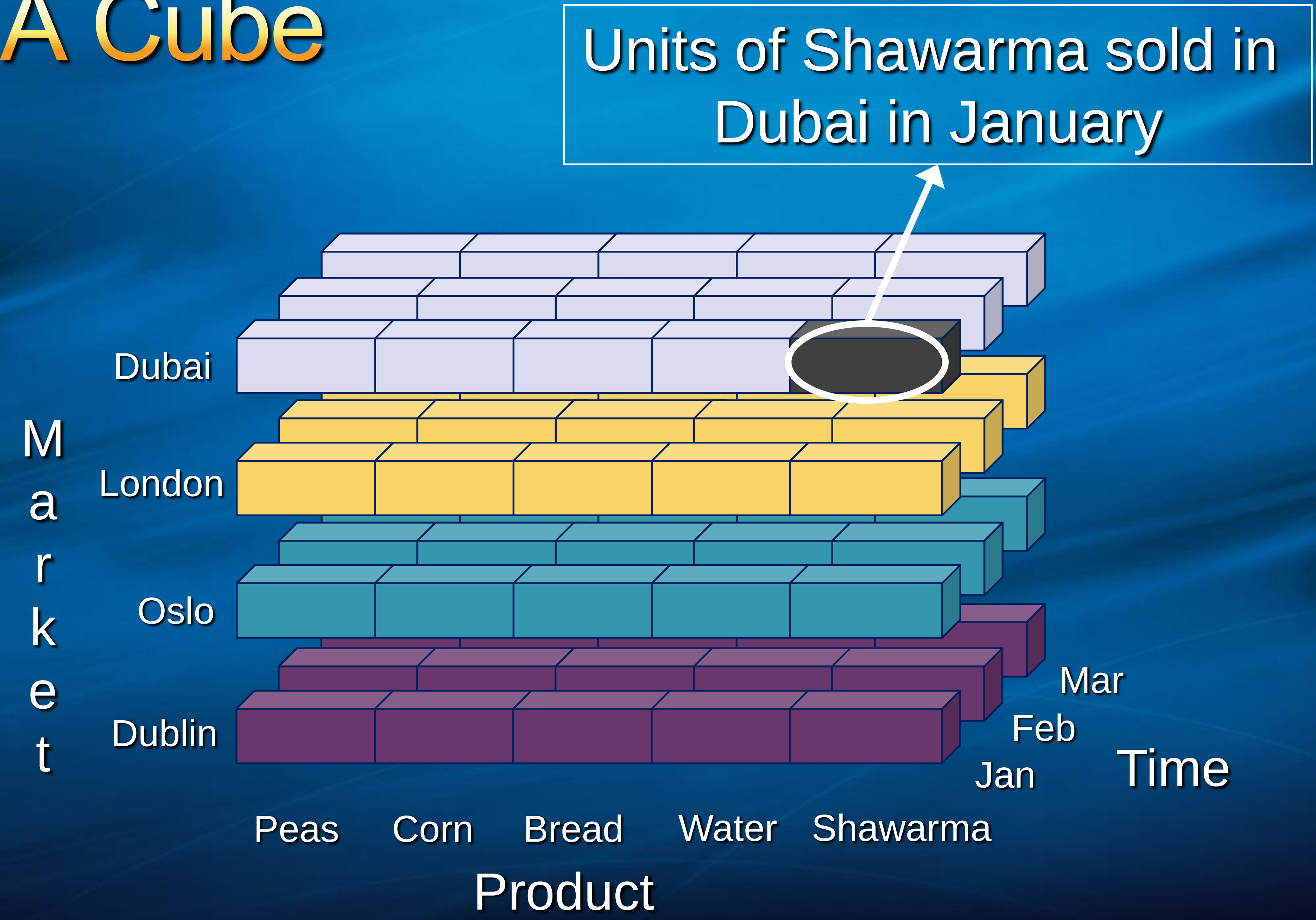
City

Country

Marital

Hierarchies

# A Cube



Units of Shawarma sold in Dubai in January

# Cube

- Collection of dimensions and measures
- Measure → numeric data associated with a set of dimensions (e.g. Qty Sold, Sales Amount, Cost)
- Multi-dimensional space
  - Defined by dimensions and measures
    - E.g. (Customers, Products, Time, Measures)
  - Intersection of dimension members and measures is
    a cell (U.A.E., Shawarma, 2006, Sales Amount) = Dhs. 10,523,374.83

# Measure Group

Measure Group

Dimension

| | Sales | Inventory | Finance |
|---|---|---|---|
| Customers | X | | |
| Products | X | X | |
| Time | X | X | X |
| Promotions | X | | |
| Warehouse | | X | |
| Department | | | X |
| Account | | | X |
| Scenario | | | X |

# Measure Group

- Group of measures with same dimensionality
- Analogous to fact table
- Cube can contain more than one measure group
  - E.g. Sales, Inventory, Finance
- Multi-dimensional space
  - Subset of dimensions and measures in the cube
- AS2000 comparison
  - Virtual Cube → Cube
  - Cube → Measure Group

# Agenda

- Server architecture and UDM Basics
- Optimizing the cube design
- Partitioning and Aggregations
- Processing
- Conclusion

# SQL Server 2005 SP2

- Consider moving to SP2 as soon as possible
- Very important release for SSAS
  - Significant number of performance optimizations
  - Excel 2007 features
  - Responses to top customer feedback
  - Etc.

- http://www.microsoft.com/downloads/details.aspx?familyid=D07219B2-1E23-49C8-8F0C-63FA18F26D3A&displaylang=en

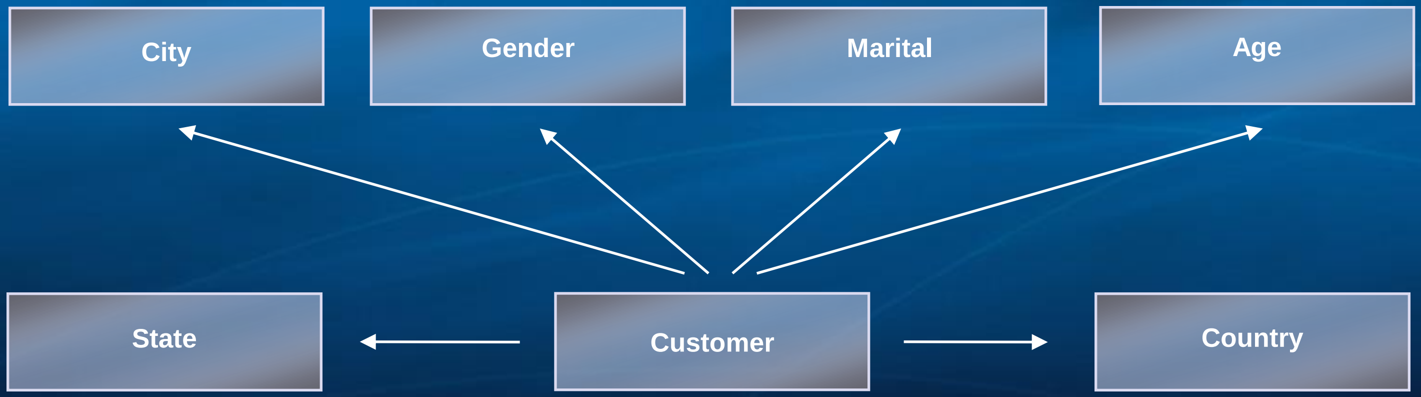
# Top 3 Tenets of Good Cube Design

- Attribute relationships
- Attribute relationships
- Attribute relationships

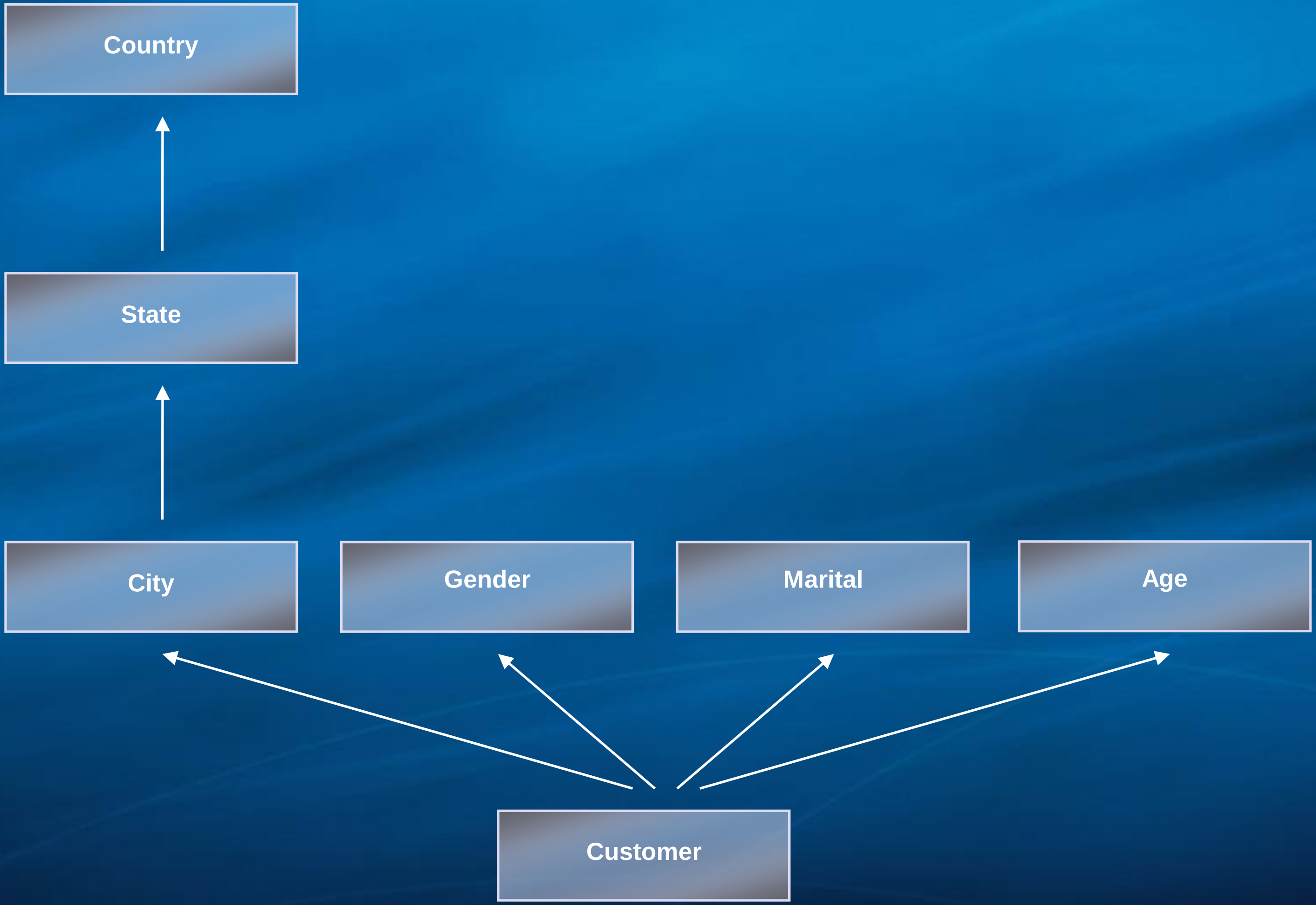
# Attribute Relationships

- One-to-many relationships between attributes
- Examples:
  - City → State, State → Country
  - Day → Month, Month → Quarter, Quarter → Year
  - Product Subcategory → Product Category
- Rigid v/s flexible relationships (default is flexible)
  - Customer → City, Customer → PhoneNo are flexible
  - Customer → BirthDate, City → State are rigid
- Server simply “works better” if you define them where applicable
- All attributes implicitly related to key attribute

# Attribute Relationships (continued)



# Attribute Relationships (continued)



# Attribute Relationships

## Where are they used?

- MDX Semantics
  - Tells the formula engine how to roll up measure values
  - If the grain of the measure group is different from the key attribute (e.g. Sales by Month)
    - Attribute relationships from grain to other attributes required (e.g. Month → Quarter, Quarter → Year)
    - Otherwise no data (NULL) returned for Quarter and Year

MDX Semantics explained in detail at:
http://www.sqlserveranalysisservices.com/OLAPPapers/AttributeRelationships.htm

# Attribute Relationships

## Where are they used?

- Aggregation design
  - Enables aggregation design algorithm to produce effective set of aggregations
- Dimension security
  - DeniedSet = {Country.U.A.E.} should deny cities and customers in U.A.E. – requires attribute relationships

# Attribute Relationships

## How to set them up?

- Creating an attribute relationship is easy, but …

# Attribute Relationships

## How to set them up?

- Creating an attribute relationship is easy, but …
  - Pay careful attention to the key columns!
  - Make sure every attribute has unique key columns (add composite keys as needed)
- There must be a 1:M relation between the key columns of the two attributes
- Invalid key columns cause a member to have multiple parents
  - Dimension processing picks one parent arbitrarily and succeeds
  - Hierarchy looks wrong!

# User Defined Hierarchies

- Pre-defined navigation paths thru dimensional space defined by attributes
- Attribute hierarchies enable ad hoc navigation
- Why create user defined hierarchies then?
  - Guide end users to interesting navigation paths
  - Existing client tools are not “attribute aware”
  - Performance
    - Optimize navigation path at processing time
    - Materialization of hierarchy tree on disk
    - Aggregation designer favors user defined hierarchies

# Best Practices for Cube Design

- Attributes
  - Define all possible attribute relationships!
  - Mark attribute relationships as rigid where appropriate
  - Use integer (or numeric) key columns
  - Set AttributeHierarchyEnabled to false for attributes not used for navigation (e.g. Phone#, Address)
  - Set AttributeHierarchyOptimizedState to NotOptimized for infrequently used attributes
  - Set AttributeHierarchyOrdered to false if the order of members returned by queries is not important
- Hierarchies
  - Use natural hierarchies where possible

# Agenda

- Server architecture and UDM Basics
- Optimizing the cube design
- Partitioning and Aggregations
- Processing
- Conclusion

# Partitioning

- Mechanism to break up large cube into manageable chunks
- Partitions can be added, processed, deleted independently
  - 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
  - MOLAP: store aggregations and facts in OLAP
  - ROLAP: store aggregations and facts in Relational
  - HOLAP: aggregations in MOLAP, leave facts in relational

***Partitions require Enterprise Edition!***

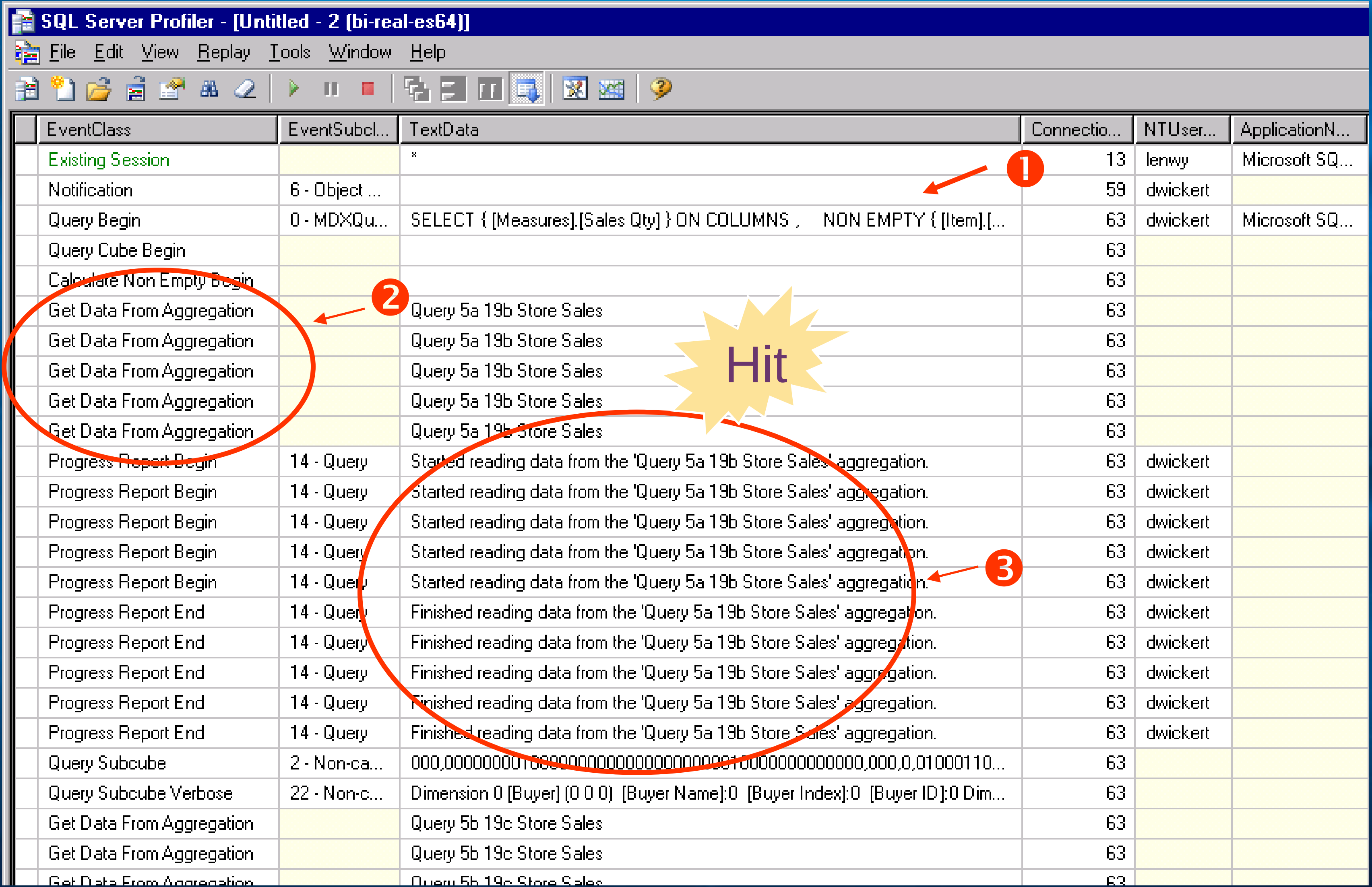
# Benefits of Partitioning

- 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 Partitioning

- No more than 100s of partitions, less than 20M rows per partition
- Specify partition slice
  - Optional for MOLAP – server auto-detects the slice and validates against user specified slice (if any)
  - Must be specified for ROLAP
- Manage storage settings by usage patterns
  - Frequently queried → MOLAP with lots of aggs
  - Periodically queried → MOLAP with less or no aggs
  - Historical → ROLAP with no aggs

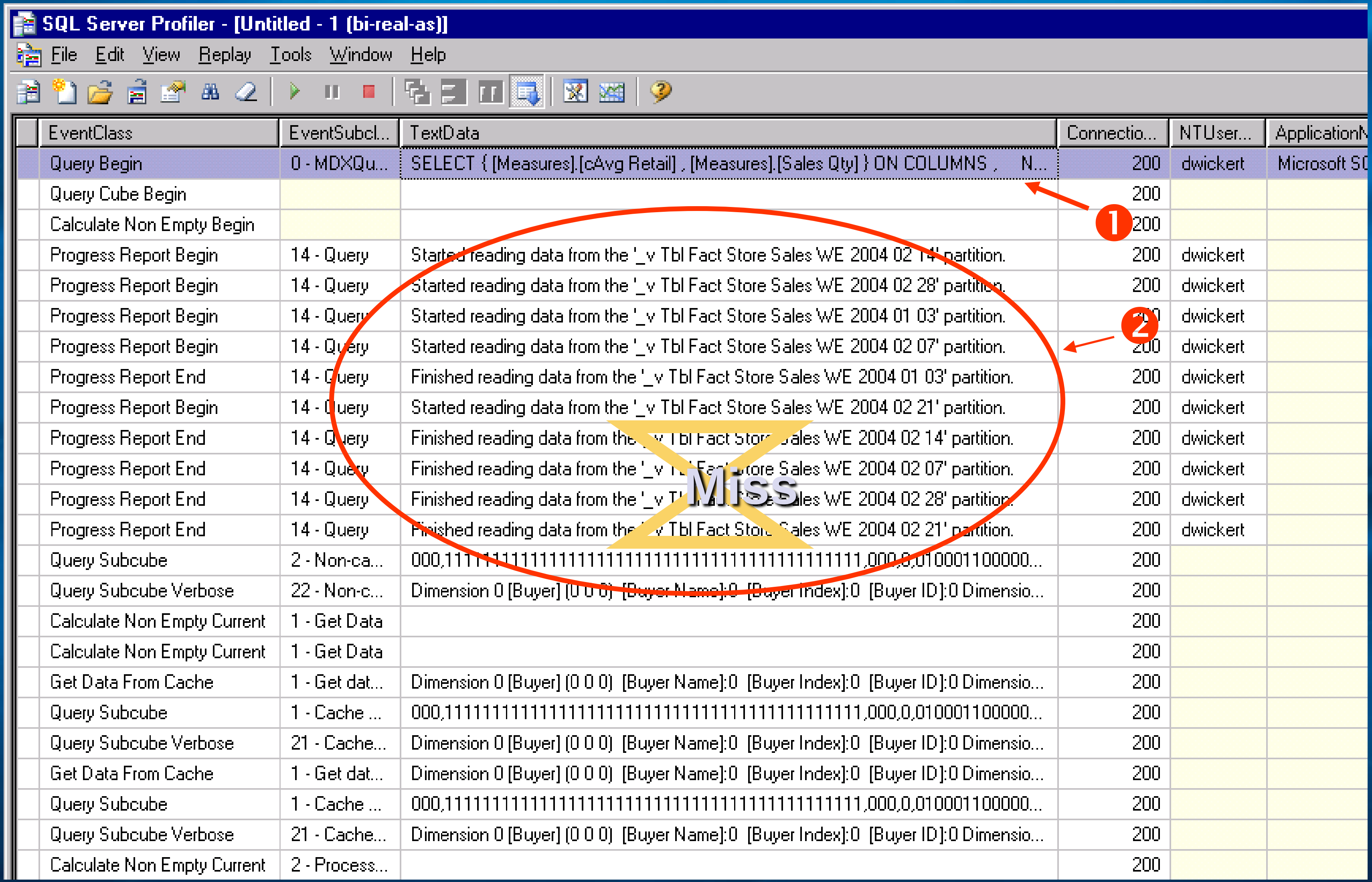
# How to Monitor Aggregation Usage?


SQL Server Profiler - [Untitled - 2 (bi-real-es64)]
File Edit View Replay Tools Window Help
❶
❷
❸
Hit

| EventClass | EventSubcl... | TextData | Connectio... | NTUser... | ApplicationN... |
|---|---|---|---|---|---|
| Existing Session | | * | 13 | lenwy | Microsoft SQ... |
| Notification | 6 - Object ... | | 59 | dwickert | |
| Query Begin | 0 - MDXQu... | SELECT { [Measures].[Sales Qty] } ON COLUMNS , NON EMPTY { [Item].[... | 63 | dwickert | Microsoft SQ... |
| Query Cube Begin | | | 63 | | |
| Calculate Non Empty Begin | | | 63 | | |
| Get Data From Aggregation | | Query 5a 19b Store Sales | 63 | | |
| Get Data From Aggregation | | Query 5a 19b Store Sales | 63 | | |
| Get Data From Aggregation | | Query 5a 19b Store Sales | 63 | | |
| Get Data From Aggregation | | Query 5a 19b Store Sales | 63 | | |
| Get Data From Aggregation | | Query 5a 19b Store Sales | 63 | | |
| Progress Report Begin | 14 - Query | Started reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the 'Query 5a 19b Store Sales' aggregation. | 63 | dwickert | |
| Query Subcube | 2 - Non-ca... | 000,0000000100000000000000000001000000000000,000,0,01000110... | 63 | | |
| Query Subcube Verbose | 22 - Non-c... | Dimension 0 [Buyer] (0 0 0) [Buyer Name]:0 [Buyer Index]:0 [Buyer ID]:0 Dim... | 63 | | |
| Get Data From Aggregation | | Query 5b 19c Store Sales | 63 | | |
| Get Data From Aggregation | | Query 5b 19c Store Sales | 63 | | |
| Get Data From Aggregation | | Query 5b 19c Store Sales | 63 | | |

Profiler

# How to Monitor Aggregation Usage?



| EventClass | EventSubcl... | TextData | Connectio... | NTUser... | ApplicationN |
|---|---|---|---|---|---|
| Query Begin | 0 - MDXQu... | SELECT { [Measures].[cAvg Retail] , [Measures].[Sales Qty] } ON COLUMNS , N... | 200 | dwickert | Microsoft SC |
| Query Cube Begin | | | 200 | | |
| Calculate Non Empty Begin | | | 200 | | |
| Progress Report Begin | 14 - Query | Started reading data from the '_v Tbl Fact Store Sales WE 2004 02 14' partition. | 200 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the '_v Tbl Fact Store Sales WE 2004 02 28' partition. | 200 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the '_v Tbl Fact Store Sales WE 2004 01 03' partition. | 200 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the '_v Tbl Fact Store Sales WE 2004 02 07' partition. | 200 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the '_v Tbl Fact Store Sales WE 2004 01 03' partition. | 200 | dwickert | |
| Progress Report Begin | 14 - Query | Started reading data from the '_v Tbl Fact Store Sales WE 2004 02 21' partition. | 200 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the '_v Tbl Fact Store Sales WE 2004 02 14' partition. | 200 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the '_v Tbl Fact Store Sales WE 2004 02 07' partition. | 200 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the '_v Tbl Fact Store Sales WE 2004 02 28' partition. | 200 | dwickert | |
| Progress Report End | 14 - Query | Finished reading data from the '_v Tbl Fact Store Sales WE 2004 02 21' partition. | 200 | dwickert | |
| Query Subcube | 2 - Non-ca... | 000,1111111111111111111111111111111111111111,000,0,010001100000... | 200 | | |
| Query Subcube Verbose | 22 - Non-c... | Dimension 0 [Buyer] (0 0 0) [Buyer Name]:0 [Buyer Index]:0 [Buyer ID]:0 Dimensio... | 200 | | |
| Calculate Non Empty Current | 1 - Get Data | | 200 | | |
| Calculate Non Empty Current | 1 - Get Data | | 200 | | |
| Get Data From Cache | 1 - Get dat... | Dimension 0 [Buyer] (0 0 0) [Buyer Name]:0 [Buyer Index]:0 [Buyer ID]:0 Dimensio... | 200 | | |
| Query Subcube | 1 - Cache ... | 000,1111111111111111111111111111111111111111,000,0,010001100000... | 200 | | |
| Query Subcube Verbose | 21 - Cache... | Dimension 0 [Buyer] (0 0 0) [Buyer Name]:0 [Buyer Index]:0 [Buyer ID]:0 Dimensio... | 200 | | |
| Get Data From Cache | 1 - Get dat... | Dimension 0 [Buyer] (0 0 0) [Buyer Name]:0 [Buyer Index]:0 [Buyer ID]:0 Dimensio... | 200 | | |
| Query Subcube | 1 - Cache ... | 000,1111111111111111111111111111111111111111,000,0,010001100000... | 200 | | |
| Query Subcube Verbose | 21 - Cache... | Dimension 0 [Buyer] (0 0 0) [Buyer Name]:0 [Buyer Index]:0 [Buyer ID]:0 Dimensio... | 200 | | |
| Calculate Non Empty Current | 2 - Process... | | 200 | | |

**Tip: SP2 includes aggregation utility for fine tuning aggregations**

# Best Practices for Aggregations

- Aggregation design cycle
  - Use Storage Design Wizard (~20% perf gain) to design initial set of aggregations
  - Enable query log and run pilot workload (beta test with limited set of users)
  - Use Usage Based Optimization (UBO) Wizard to refine aggregations
  - Use larger perf gain (70-80%)
  - Reprocess partitions for new aggregations to take effect
  - Periodically use UBO to refine aggregations

# Agenda

- Server architecture and UDM Basics
- Optimizing the cube design
- Partitioning and Aggregations
- Processing
- Conclusion

# Improving Processing

- SQL Server Performance Tuning
  - Improve the queries that are used for extracting data from SQL Server
    - Check for proper plans and indexing
    - Conduct regular SQL performance tuning process
- AS Processing Improvements
  - Use SP2 !!
    - Processing 20 partitions: SP1 1:56, SP2: 1:06
  - Don’t let UI default for parallel processing
    - Go into advanced processing tab and change it
  - Monitor the values:
    - Maximum number of data source connections
    - MaxParallel – How many partitions processed in parallel, don’t let the server decide on its own.
  - Use INT for keys, if possible.

***Parallel processing requires Enterprise Edition!***

# Improving Processing

- For best performance use ASCMD.EXE and XMLA
  - Use <Parallel> </Parallel> to group processing tasks together until Server is using maximum resources
  - Proper use of <Transaction> </Transaction>
- ProcessFact and ProcessIndex separately instead of ProcessFull (for large partitions)
  - Can help isolate performance problems & provide predictable data source usage
- ProcessClearIndexes deletes existing indexes and ProcessIndexes generates or reprocesses existing ones.

# Best Practices for Processing

- Partition processing
  - Monitor aggregation processing spilling to disk (perfmon counters for temp file usage)
    - Add memory, turn on /3GB, move to x64/ia64
  - Fully process partitions periodically
    - Achieves better compression over repeated incremental processing
- Data sources
  - Avoid using .NET data sources – OLEDB is faster for processing, .NET data sources may have cause ~30% performance hit

# Conclusion

- AS2005 is major re-architecture from AS2000
- Design for perf & scalability from the start
- Many principles carry through from AS2000
  - Dimensional design, Partitioning, Aggregations
- Many new principles in AS2005
  - Attribute relationships, natural hierarchies
  - New design alternatives – role playing, many-to-many, reference dimensions, semi-additive measures
  - Flexible processing options
  - MDX scripts, scopes
- Use Analysis Services Enterprise Edition and SP2 to get max performance and scale

# Resources

- SSIS
  - SQL Server Integration Services site – links to blogs, training, partners, etc.: http://msdn.microsoft.com/SQL/sqlwarehouse/SSIS/default.aspx
  - SSIS MSDN Forum: http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=80&SiteID=1
  - SSIS MVP community site: http://www.sqlis.com
- SSAS
  - BLOGS: http://blogs.msdn.com/sqlcat
  - PROJECT REAL-Business Intelligence in Practice
  - Analysis Services Performance Guide
  - TechNet: Analysis Services for IT Professionals
- Microsoft BI
  - SQL Server Business Intelligence public site: http://www.microsoft.com/sql/evaluation/bi/default.asp
  - http://www.microsoft.com/bi