



KIMBALL GROUP
Consulting | Kimball University

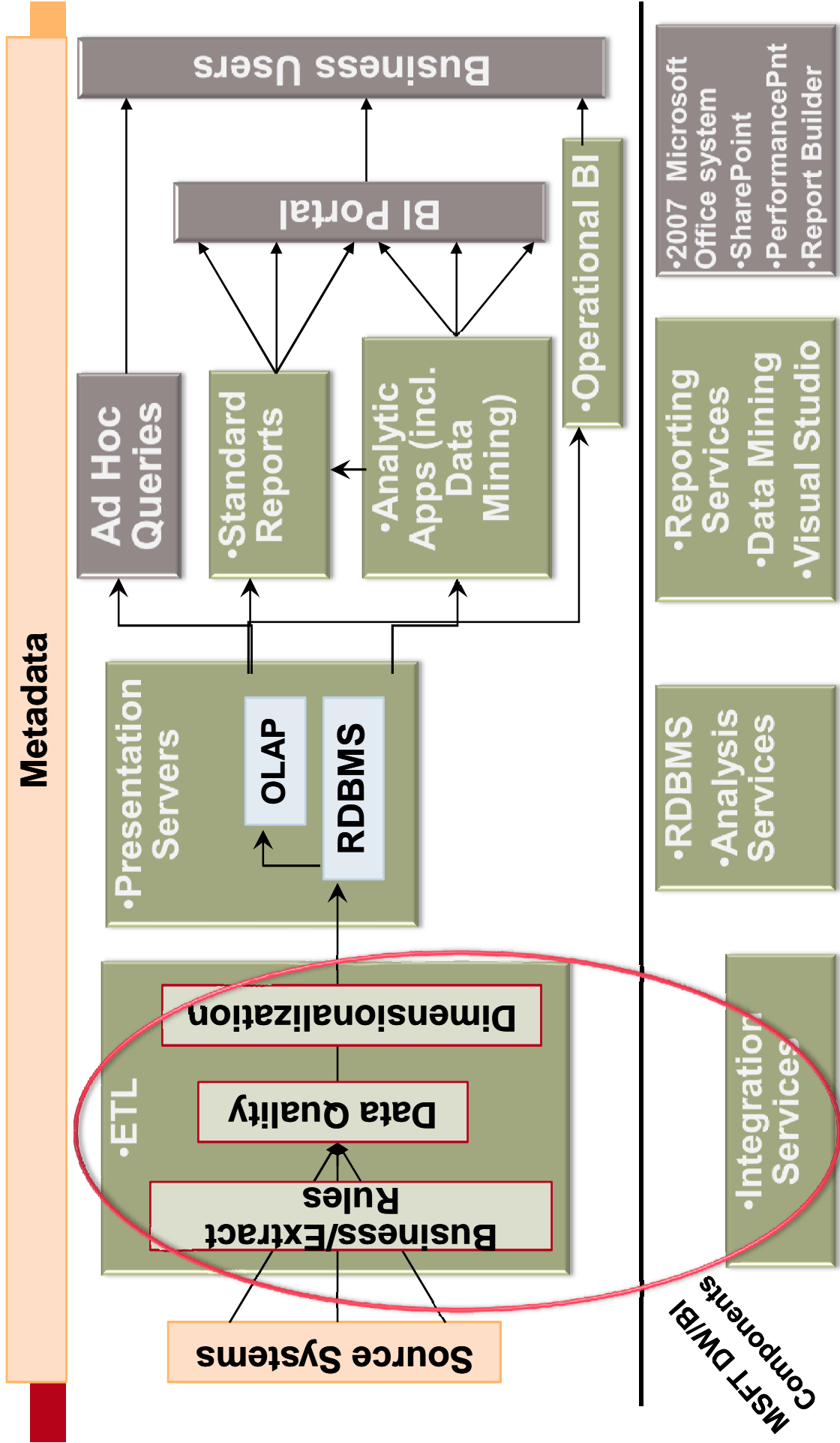
POPULATING A KIMBALL METHOD DATA WAREHOUSE WITH SQL SERVER AND INTEGRATION SERVICES 2008

Joy Mundy
Kimball Group

Agenda

- *Setting the stage*
 - *The DW / BI technical architecture*
 - *Vocabulary*
- Before you write a single line of code
- E=Extract
- T=Transform
- L=Load

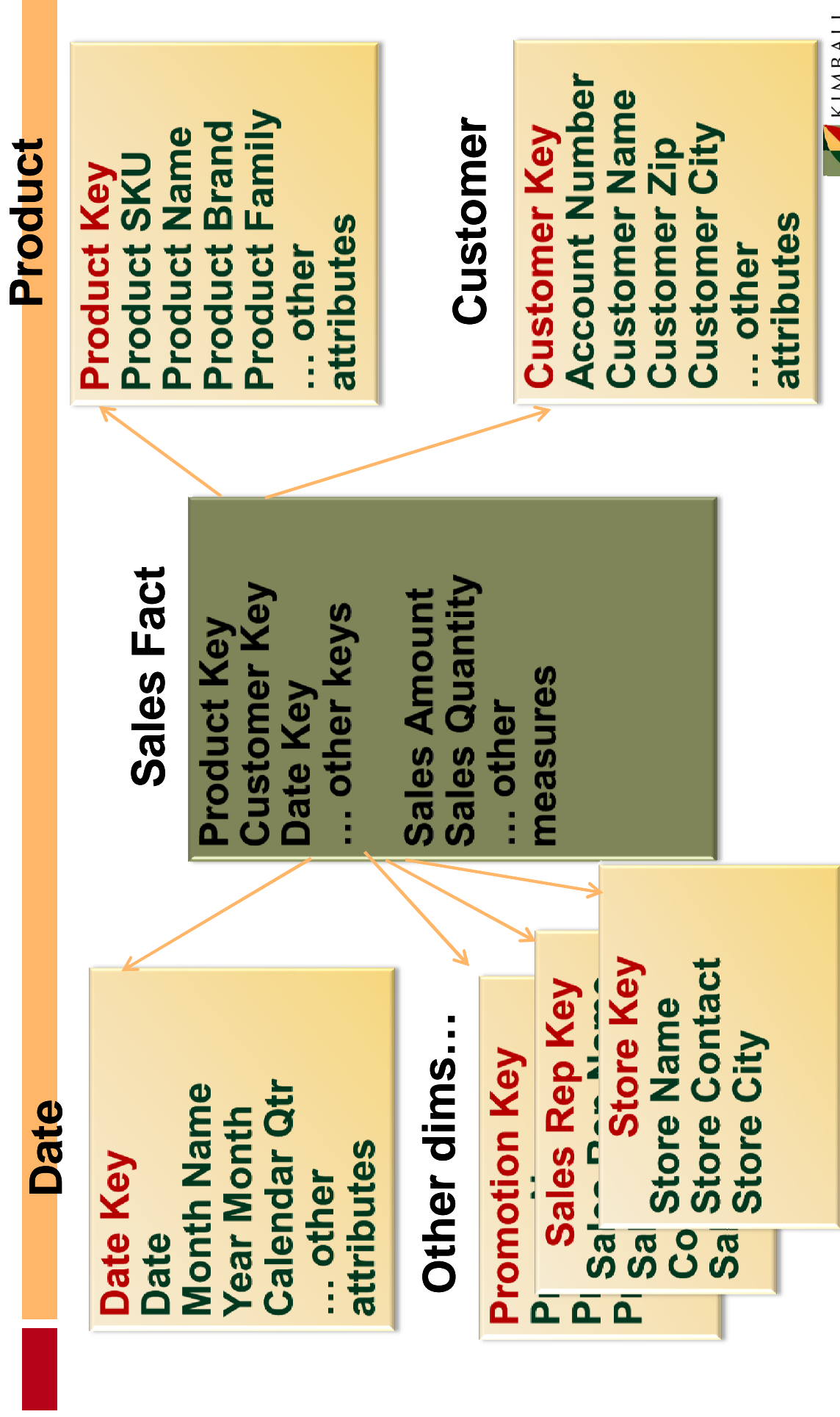
The DW/BI Technical Architecture



Kimball Method Fundamentals

- Focus on the business
- Build a **dimensional** data warehouse / business intelligence system
 - Dimension tables such as Customer contain descriptive information
 - Fact tables such as Sales contain detailed transactions, and link to the dimensions
 - Dimension attribute changes are managed and **conformed** across the enterprise
- Excellent user experience is paramount

Relational Dimensional Model



Surrogate Keys

- Dimension PKs should be surrogate (meaningless) keys
 - Managed by the DW
 - Usually an integer type
 - Usually populated via IDENTITY keyword in dimension table definition
- Why?
 - Small (int) keys are vital for performance
 - The source system **WILL** screw you if you don't manage the keys yourself
 - Enables dimension attribute change tracking

Surrogate Keys and ETL

- Dimensions
 - Carry source system key(s) as non-key attributes in the dimension
 - New rows automatically get a new surrogate key
- Facts
 - Fact table usually does not contain source system keys
 - Final step of fact processing is to exchange the source system keys for DW surrogate keys
 - Lookup to dimension tables based on source key, returning surrogate key

Conformed Dimensions

- ❑ One master dimension table that all fact tables subscribe to
- ❑ Get agreement organization-wide on:
 - What the dimensions are called
 - Which hierarchies you have
 - Similar-but-different attributes and hierarchies have different names
 - Which attributes are managed by restating history and which by tracking history
 - Create two sets of attributes if you need it both ways
- ❑ Why?
 - Single version of the truth
 - Flexibility of basic design

Dimensional Modeling Myths and Misconceptions

- Dimensional means summary
- Dimensional models are built to support specific applications (or departments)
- The dimensional model is less flexible than a third normal form model in DW/BI systems
- The dimensional approach is not Enterprise-oriented

FALSE

Agenda

- Setting the stage
- *Before you write a single line of code*
 - *Gather business requirements*
 - *Profile data early and often*
 - *Develop an excellent dimensional model*
 - *Plan the ETL application*
- **E=Extract**
- **T=Transform**
- **L=Load**

Gather Business Requirements

- Gather detailed requirements *from the business users*
 - This means you have to talk to them
 - Document requirements, or you'll have to do it again
 - Talk to IT (data experts) too
- Multi-step process
 - [Sometimes] Overview of the landscape. Where should we begin? Overview of data realities
 - Detailed requirements and data realities

Profile the Data



- Early and often
- Does the data exist to support the required analysis?
- Where are the problems affecting ETL design
 - Primary keys
 - Referential integrity
 - NULL values
 - Junk values
 - The dreaded “Notes” field

Demo: Data Profiling

- SSIS Data Profiling Task
- Kimball Data Profiling Reports

Design the Dimensional Model

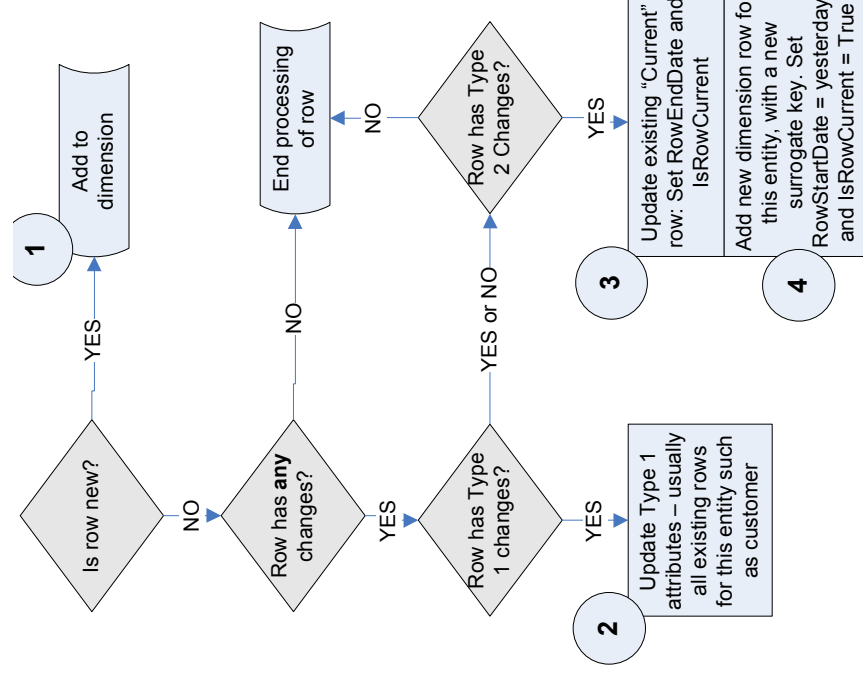
- Dimensions are the key UI element to your organization's information
- Dimension grain
 - What does a row in the dimension table represent?
- Dimension hierarchies
 - Real vs. navigational
- Dimension attribute changes
 - Type 1: Restate history
 - Type 2: Track history

Changing Attributes

- Each attribute (column) in the dimension table is subject to change
 - **What do the business users want?**
 - Sometimes they want it both ways
- Type 1 (Update)
 - Easiest for base ETL, simply update column in place
 - Potentially problematic for aggregate management
- Type 2 (Track history)
 - Add a new row for each new set of attributes
 - Facts tie in to the set of attributes in effect at txn time
 - Track row effective date range, row is current, potentially row change reason

Overview of SCD Processing

- SCD = Slowly Changing Dimension
- Most dims include both SCD-1 (restate history) and SCD-2 (track history) attributes
- Ideally, differentiate by attribute & reason
 - Eg, “fix an error” vs. “change of status”



Plan the ETL Application

ETL System Specification

- Detailed dimensional data model
- Architectures and basic (default) strategies
 - Dimension attribute change management
 - Fact table surrogate key lookups
 - Default methods for extracting data
 - ...
- Table-level details
 - Table physical design
 - Historical load range and sources
 - Incremental load frequency, volume, source, changed data capture
 - Dimension attribute change strategy (SCD1 or 2)
 - Detailed source-to-target mapping
 - Pseudocode transformations
 - ...
- Summary of flow

The ETL Plan: Get Real

- ❑ OK, that's a lot of work
- ❑ You *must* think these issues through before you go into production
- ❑ At the very least, before you write a single package, pull together
 - Relational database physical design
 - Detailed source-to-target map
 - Data profiling reports
 - High level ETL plan
 - 5 pages explaining basically what you're planning to do

Agenda



- Setting the stage
- Before you write a single line of code
 - *E=Extract*
 - *Extract and archive*
 - *Compute measures of data validity*
 - *Identify changed data*
- T=Transform
- L=Load

Data Extraction Best Practices

- ❑ Minimize impact on the source system
 - Keep source queries simple
 - Often, stage to file or table rather than ETL in stream
 - Avoid transforming data upon extract
- ❑ Save a copy of the untransformed data
 - For archival purposes (Internal Audit will love you, insofar as they are capable of emotion)
 - Design packages with a restart point here
- ❑ Often makes sense to separate E and TL into separate packages

Extracting Data

- Relational sources
 - Usually *pull* from sources (use a query)
 - Keep the source query simple
 - Implement source query in the SSIS Data Flow task, Data Reader Source
- Non-relational sources
 - Usually *pushed* from source system
 - Flat files common
 - Third-party connectors can allow pull from within SSIS

Compute Data Validity Measures

- Compute and check measures of data validity as early in the process as possible
 - Rowcounts
 - “Reasonableness” counts
 - Income=Expense
 - Sold more than N products to X customers
- Store data validity measures in processing metadata
- Evaluate data validity before moving forward

Changed Data Capture

- ❑ The only sensible place to do this is on the source system
 - Changed Data Capture feature in SQL2008
 - Replication
 - Triggers
 - Some other application logic
- ❑ If you have to do it in ETL
 - Use caution relying on LastModifyDT in source system
 - Easiest: Do comparison on clean, fully constructed dimension rows
 - Best performance: Stage all tables, compare immediately after extract

Finding Changed Data

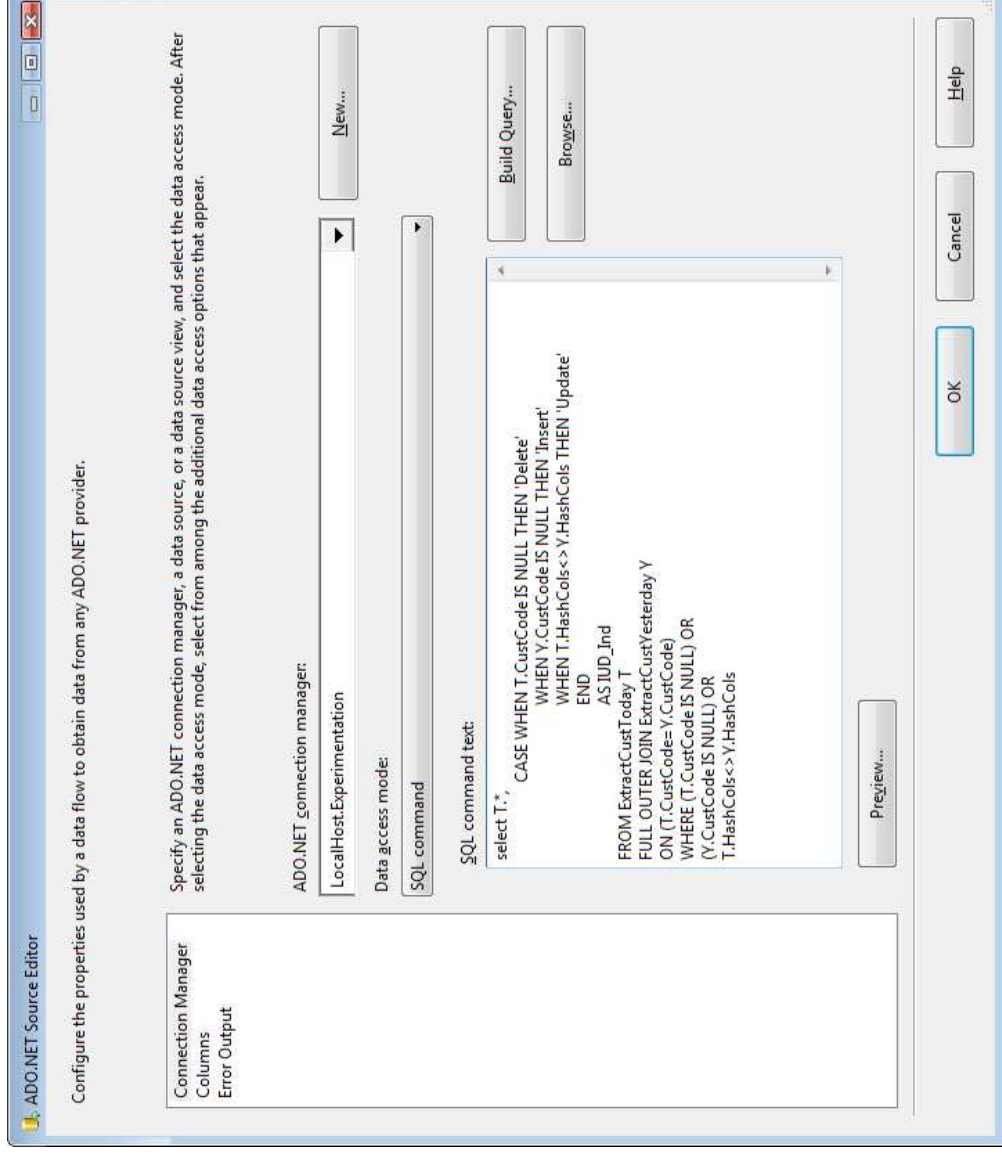
As data is extracted

- ❑ Begin with two identically structured staging tables
 - Today and Yesterday (rename daily)
 - Extract all rows and columns to Today
 - You want a PK index
 - One additional column contains a checksum (or HashBytes) of all non-key attributes
- ❑ Transformation step begins with a FULL outer join between Yesterday and Today
 - Not practical for huge data
 - Fine for most dimensions, some fact tables

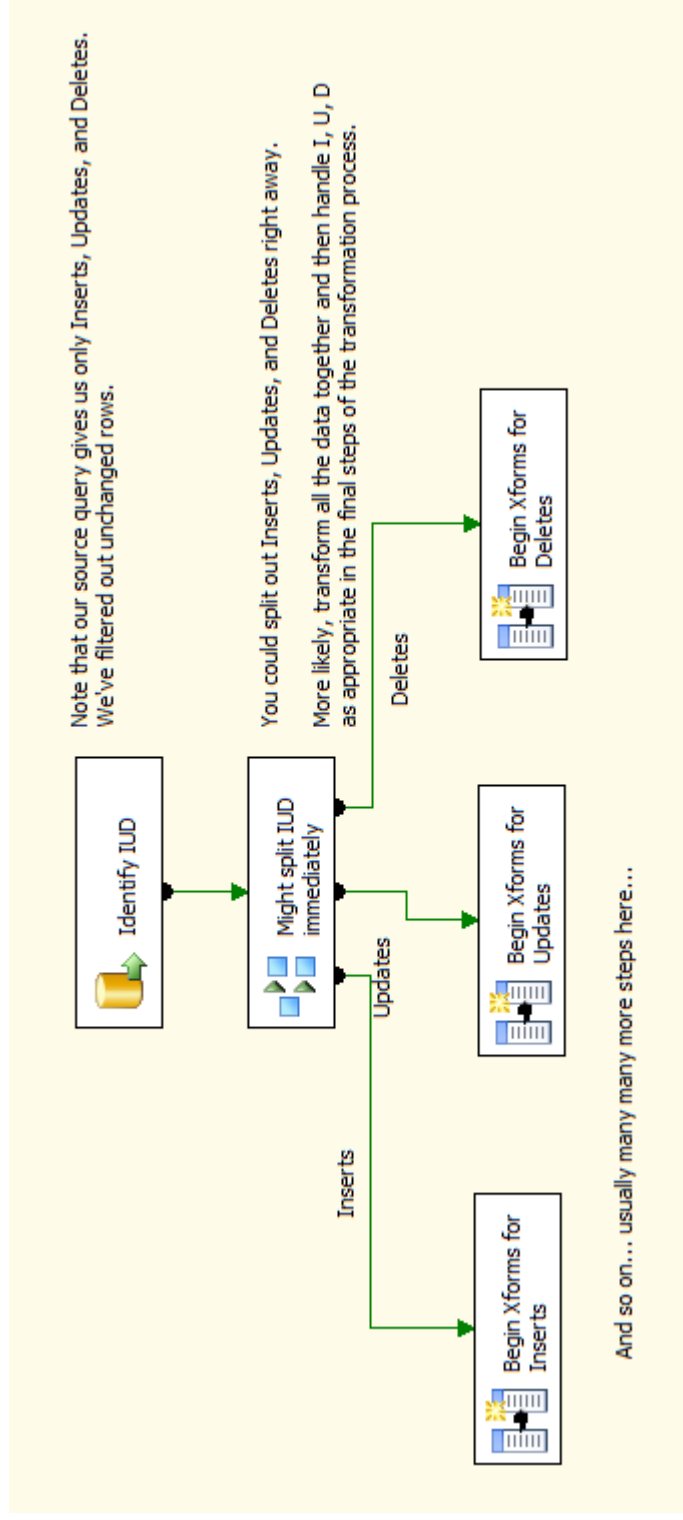
Finding Changed Data (cont)

- ❑ Full outer join with Today and Yesterday
- ❑ INSERTS (new rows)
 - Left part of join: Here today, not yesterday
- ❑ UPDATES
 - “Inner join” part of the result set
 - WHERE Today.Checksum <> Yesterday.Checksum
- ❑ DELETES
 - Right part of join: Here yesterday, not today
- ❑ Compute IUD_Indicator in the same query
- ❑ Filter out unchanged rows
 - Inner join where the checksums are the same

Changed Data: Ugly Outer Join as Source to Xform Data Flow



Changed Data: The Xform Data Flow



Agenda

- Setting the stage
- Before you write a single line of code
- E=Extract
- T=Transform
 - General best practices
 - Dimension table hierarchy management
 - Exceptional case: Loading dimensions as you populate facts
- L=Load

Transformations: Best Practices

- Use a template package to enforce (encourage) best practices
- For most of us, the main reason to use SSIS is to create comprehensible “code”
 - Your DW will change; your ETL system will change
- Always rename every object in your package
 - Data Flow 2 == Fail!
- Resist the temptation to hide transformation logic in SQL

Dimension Hierarchy Management

- Get religious about hierarchies
- A true hierarchy has referential integrity between levels
 - Anything else is simply a reporting relationship
 - Do you need (or want) a key for hierarchy levels?
 - The payoff comes to users, especially in SSAS
- The ETL system is not the right place to manage true hierarchies
 - People need to be involved
 - Do it before ETL
 - This is a job for Master Data Management or correctly designed source systems

Dimension Loading during Fact Processing

- Normal case
 - Process dimensions first, then facts
 - Fact table transformation ends with surrogate key pipeline
 - Look up each source system key for its DW surrogate key
 - Join to populated dimension tables on source system key
 - Return surrogate key
- Fairly common exception case
 - Load dimension row as observed in fact stream
 - Early arriving facts
 - Many-to-many dimensions

Demo: Surrogate Key Pipeline

- Referential Integrity error handling

Agenda



- Setting the stage
- Before you write a single line of code
- E=Extract
- T=Transform
- *L=Load*
 - *Loading Data Fast*
 - *Handling Updates and Deletes*

Loading Data Fast

- Bulk load techniques include bcp, BULK INSERT, SELECT...INTO, and SSIS OLE DB or SQL destination adapters
- Bulk load is a necessary, but not sufficient, condition for fast load. In addition,
 - Table must be empty (in which case it can have indexes), or
 - Table can have data or no indexes
- Fast loads are 10-100 **times** faster than row-by-row

Data Loading Best Practices

- ❑ Use large(-ish) batch size for best performance
- ❑ Always use Error Flow on load
 - Add a Data Viewer for the first time you try to load data
- ❑ Fast load restrictions limit incremental fast loads to partitioned tables in most scenarios
- ❑ Updates and deletes:
 - Good design usually forbids actual deletes, instead set flag
 - Ledger transaction-grain facts, even if your source system does not
 - OLEDB Command transform
 - MERGE SQL statement
 - Set-based UPDATE SQL statement

Demo: Loading Data

- Set up to load batches of data

MERGE Statement

- New SQL 2008 MERGE statement
 - Insert, Update, and Delete in a single statement
 - Each row can only be I/U/D'd once
 - Syntax is straightforward
 - Works on tables, not SSIS flows
- IF you have a mixed I/U/D load, and no xforms, use MERGE rather than SSIS
 - Not a very common situation

MERGE and SSIS

1. Create a Data Flow task

- Use SSIS for **inserts** if you can separate them easily and cheaply (Conditional Split is cheap; Lookup is not).
- De-duplicate the flow!
- Flow [I]/U/D candidates into a **staging table**



2. In the Control Flow, create an Execute SQL Task. Set up the MERGE statement

- Careful of transaction size



Places MERGE Doesn't Work Well (but feels like it should)

- SCD handling
 - Cannot do SCD handling in a single MERGE statement
 - BUT: See recent Kimball Design Tip (Nov-2008) for walkthrough of how to do it with MERGE embedded inside an INSERT
- Ledgering fact rows from an update-in-place source
 - You only get one INSERT flow on a Merge; we'd need two (one for real inserts, one for the ledgered rows)

Overall Design Considerations

- ETL application must understand Analysis Services' requirements
 - The only way to process a Fact's update or delete in SSAS is to fully process the partition containing that fact
 - ETL needs to know which SSAS partitions to process
- ETL developers aren't allowed to compromise business user requirements without a cost/benefit analysis and signoff

Conclusion

- ETL system is harder to design and develop than most people think
- SQL Server Integration Services contains functionality you need to build an enterprise-class ETL system
 - Must add some scripting
 - It's not hard!
- Some problems are intractably difficult, and no tool will make them magically go away
- Don't skimp on ETL! Great ETL makes Analysis Services & Reporting be ezmode

Additional Resources

- Books
 - **The Microsoft Data Warehouse Toolkit**, J. Mundy and W. Thornthwaite, Wiley (2006)
 - **The Data Warehouse ETL Toolkit**, R. Kimball and J. Caserta, Wiley (2005)
 - **The Data Warehouse Toolkit**, 2nd Edition, R. Kimball and M. Ross, Wiley (2002)

Even More Resources

- Websites
 - See fourteen years of free articles at www.KimballGroup.com
- Classes
 - Microsoft Data Warehouse in Depth, a 4-day class in the SQL Server product set at <http://www.kimballgroup.com/html/kucourseMDWD.html>