Extract, Transform, Load (ETL)

Slides adapted from Torben Bach Pedersen
, Man Lung Yiu and Dimitra Vista

ETL Overview

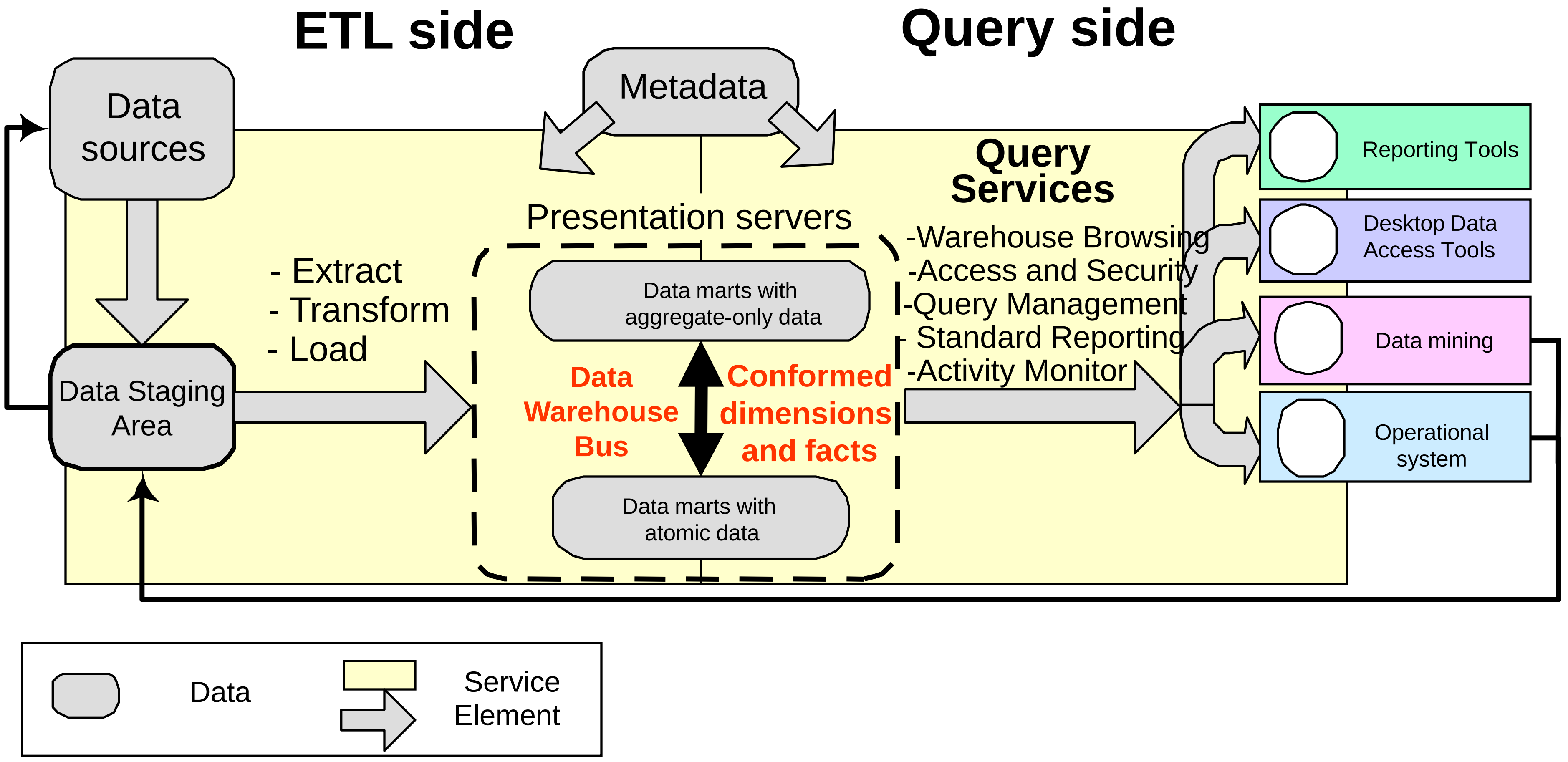
- The ETL Process
- General ETL issues
  - Building dimensions
  - Building fact tables
  - Extract
  - Transformations/cleaning
  - Load
- MS Integration Services

- Reading materials
  - Kimball ch. 10;
  - Jarke ch. 4.1-4.3;
  - Supplementary: Designing and Implementing Packages Tutorials http://msdn.microsoft.com/en-us/library/ms167031.aspx

The ETL Process

- The **most underestimated** process in DW development
- The **most time-consuming** process in DW development
  - 80% of development time is spent on ETL!
- Extract
  - Extract relevant data
- Transform
  - Transform data to DW format
  - Build keys, etc.
  - cleaning of data
- Load
  - Load data into DW
  - Build aggregates, etc.

ETL In The Architecture



Designing the staging area

- The staging area is owned by the ETL team
  - no indexes, no aggregations, no presentation access, no querying, no service level agreements
  - Transformations/cleaning done here
- Users are not allowed in the staging area for any reason
  - staging is a “construction” site
- Reports cannot access data in the staging area
  - tables can be added, or dropped without modifying the user community
- Only ETL processes can read/write the staging area (ETL developers must capture table names, update strategies, load frequency, ETL jobs, expected growth and other details about the staging area)
- The staging area consists of both RDBMS tables and data files (DBMS have become better at this)

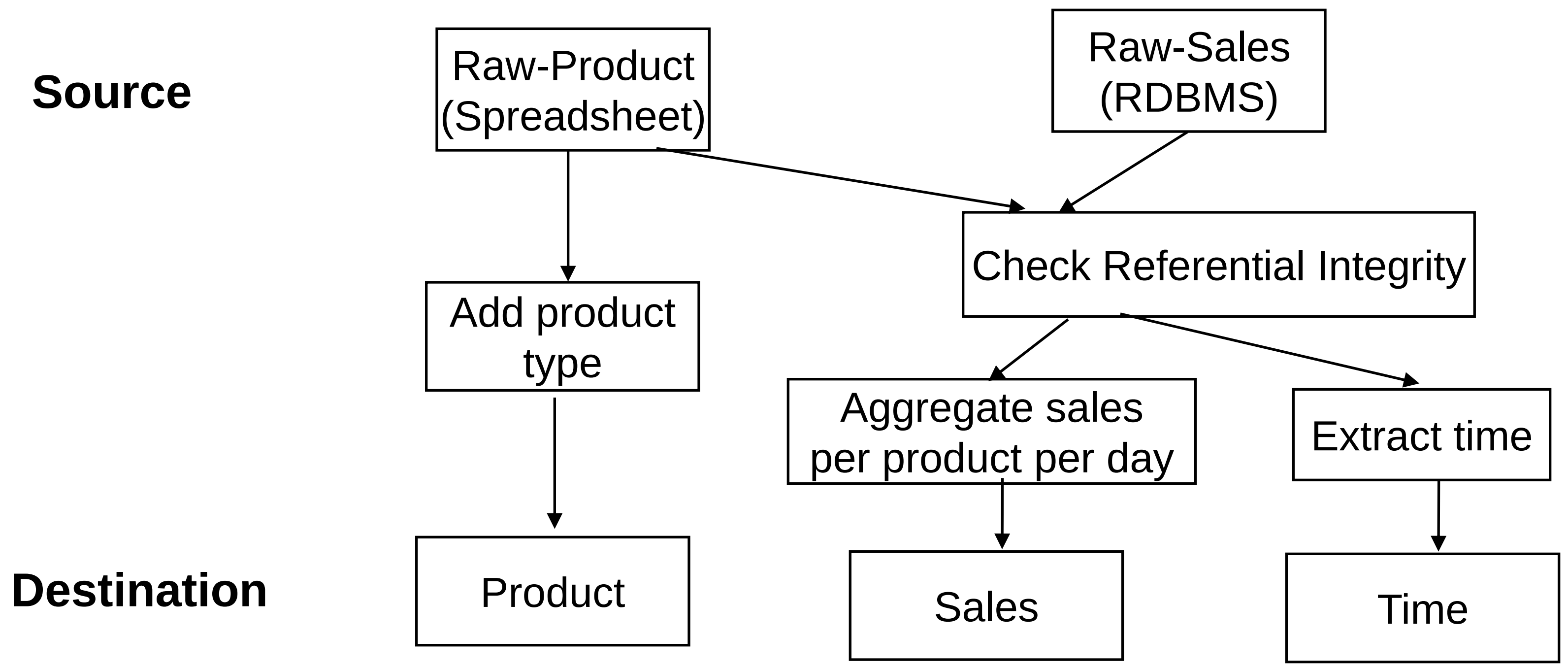
ETL Construction Process

- Plan
  1) Make high-level diagram of source-destination flow
  2) Test, choose and implement ETL tool
  3) Develop default strategies for common activities, e.g. extraction, dimension management, etc
  4) Drill down by target table, each column
- Develop One-time historic load process
  4) Build and test the historic dimension table loads
  5) Build and test the historic fact table loads
- Construction of fact tables and automation
  7) Build and test the dimension table incremental load process
  8) Build and test the fact table incremental load process
  9) Build and test aggregate table / OLAP (you do this later)
  10) Design, construct, and test ETL automation

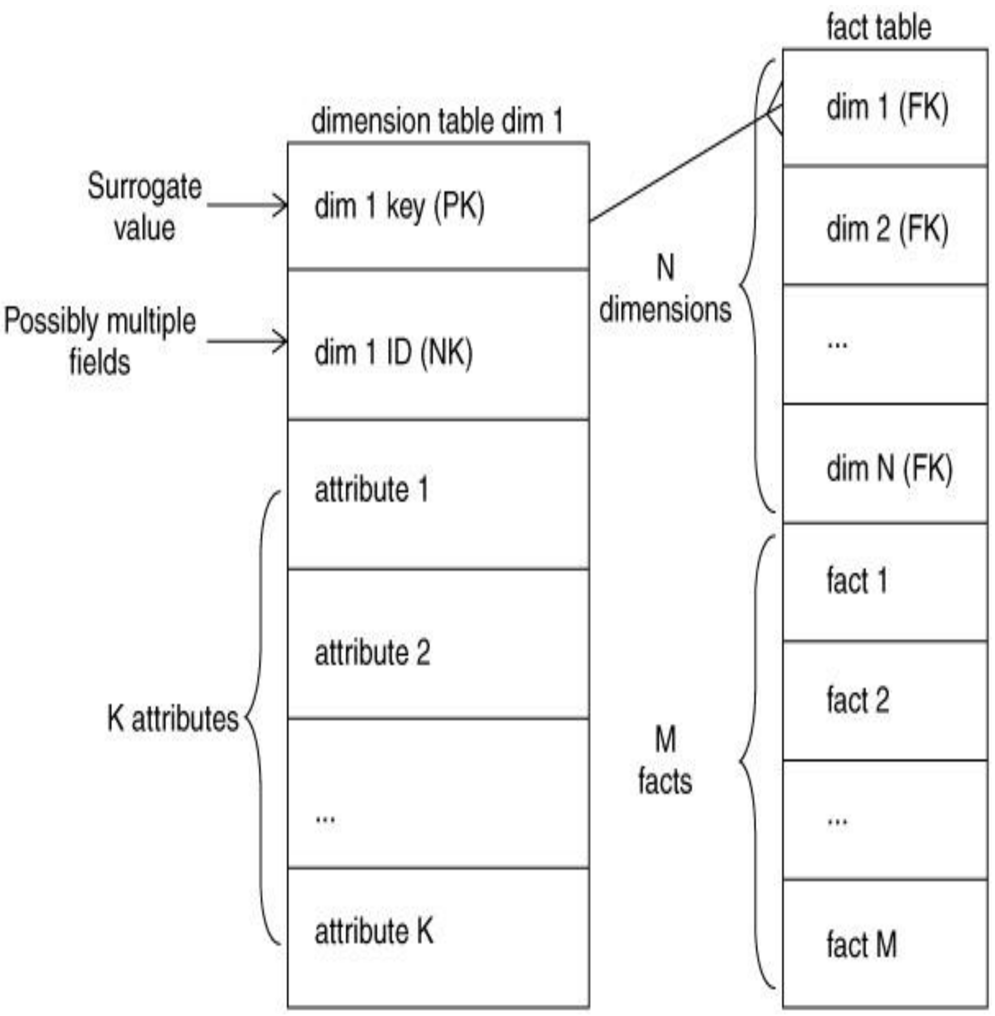
Why we consider dimensions before fact tables?

High-level diagram

1) Make high-level diagram of source-destination flow
   - Mainly used for communication purpose
   - One page only, highlight sources and destinations
   - Steps: extract, transform, load


Source
Raw-Product (Spreadsheet)
Raw-Sales (RDBMS)
Check Referential Integrity
Add product type
Aggregate sales per product per day
Extract time
Destination
Product
Sales
Time

The basic structure of a dimension



- Primary key (PK)
  - Meaningless, unique integer
  - Aka as surrogate key
  - Joins to Fact Tables
  - Is a Foreign Key to Fact Tables
- Natural key (NK)
  - Meaningful key extracted from source systems
  - 1-to-1 relationship to the PK for static dimensions
  - 1-to-many relationship to the PK for slowly changing dimensions, tracks history of changes to the dimension
- Descriptive Attributes
  - Primary textual but numbers legitimate but not numbers that are measured quantities
  - 100 such attributes normal
  - Static or slow changing only
  - Product price -- either fact or dimension attribute

Generating surrogate keys for Dimensions

- Via triggers in the DBMS
  - Read the latest surrogate key, generate the next value, create the record
  - Disadvantages: severe performance bottlenecks
- Via the ETL process, an ETL tool or a 3-rd party application generate the unique numbers
  - A surrogate key counter per dimension
  - Maintain consistency of surrogate keys between dev, test and production
- Using Smart Keys
  - Concatenate the natural key of the dimension in the source(s) with the timestamp of the record in the source or the Data Warehouse.
  - Tempting but wrong

Building Dimensions

- Static dimension table
  - DW key assignment: production keys to DW keys (surrogate) using table
  - Combination of data sources: find common key?
  - Check one-one and one-many relationships using sorting
- Handling dimension changes
  - Described in last lecture
  - Find the **newest** DW key for a given production key
  - Table for mapping production keys to DW keys must be maintained and updated
- Load of dimension changes
  - Small dimensions: replace
  - Large dimensions: load only changes

Product table

Type	**Product**	**……**
Flask	Tuborg	
Flask	Carlsberg	
Milk	Skim-Milk	
Milk	Cacao-Milk	
Vand	Sodavand	
……	……	

Product dimension of FClub vs.
Product dimension of a supermarket

How to check 1 to 1 and 1 to many

- Sorting
- 1 to many, given a table product, check if product_sku and product_model are 1: many

```
SELECT  product_sku
Count[*]  as row count
Count (distinct product_model) as model_count
FROM staging_database.product
GROUP BY product_sku
HAVING count (distinct product_model)  > 1
```

The grain of a dimension

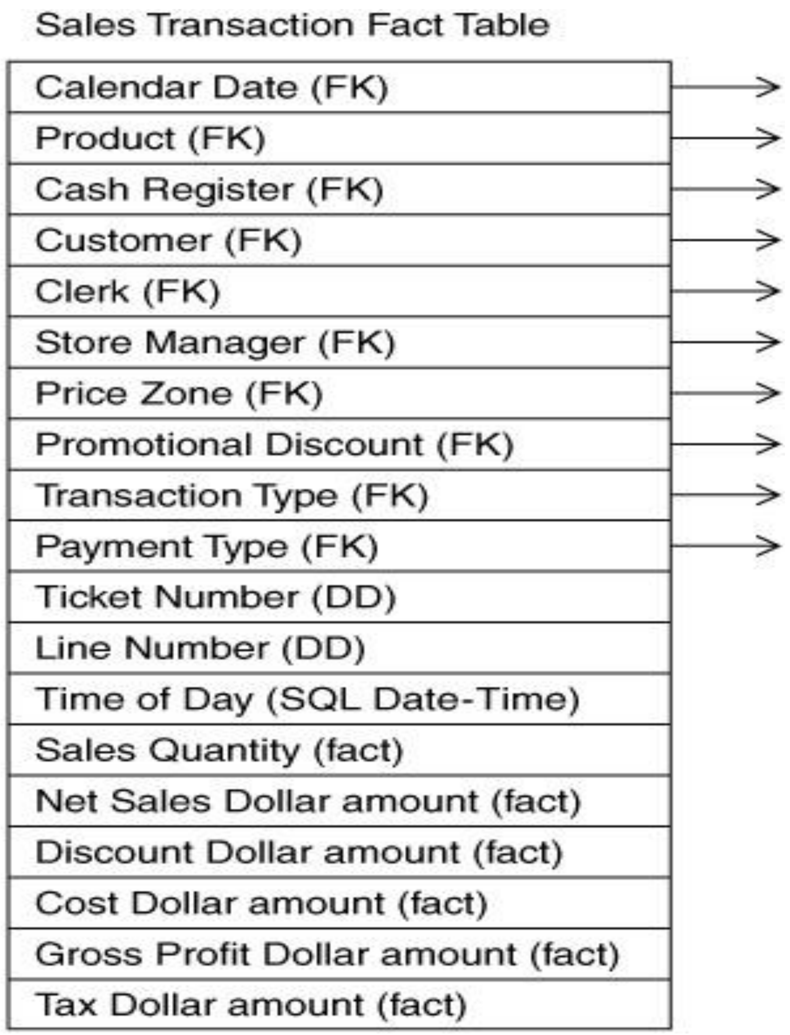
- The definition of the key of the dimension in business terms, what does the dimension represent
- Challenge: analyze the source systems so that a particular set of fields in that source corresponds to the grain of the dimension
- Verify that a given source (file) implements the intended grain
  - Nothing should be returned by this from the source system/file
  - If something is returned by this, the fields A, B and C do not represent the grain of the dimension

```
select A, B, C, count(*)
from DimensionTableSource
group by A, B, C
having count(*) > 1
```

The basic load plan for a dimension

- Simple Case: the dimension is loaded as a lookup table
- Typical Case
  - Data cleaning
    - Validate the data, apply business rules to make the data consistent, column validity enforcement, cross-column value checking, row de-duplication
  - Data conforming
    - Align the content of some or all of the fields in the dimension with fields in similar or identical dimensions in other parts of the data warehouse
      - Fact tables: billing transactions, customer support calls
      - IF they use the same dimensions, then the dimensions are conformed
  - Data Delivery
    - All the steps required to deal with slow-changing dimensions
    - Write the dimension to the physical table
    - Creating and assigning the surrogate key, making sure the natural key is correct, etc.

The basic structure of a fact table



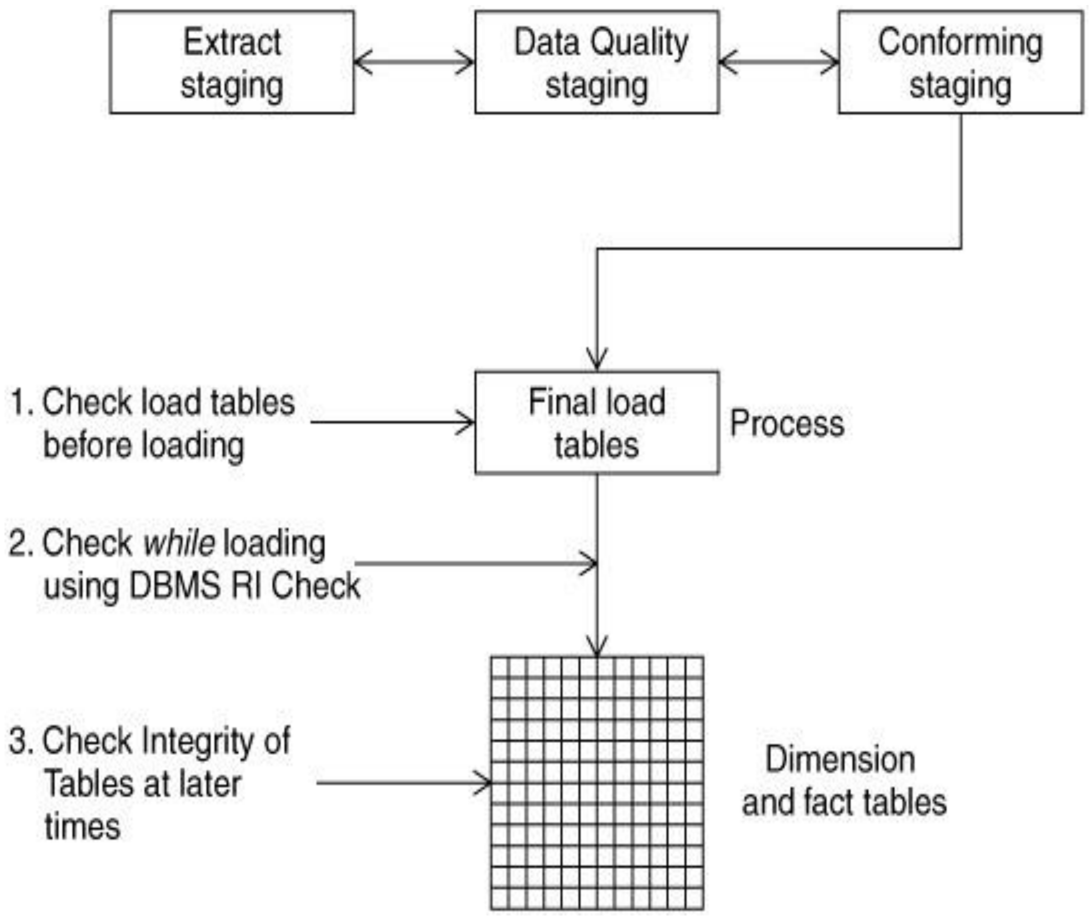
Sales Transaction Fact Table
Calendar Date (FK)
Product (FK)
Cash Register (FK)
Customer (FK)
Clerk (FK)
Store Manager (FK)
Price Zone (FK)
Promotional Discount (FK)
Transaction Type (FK)
Payment Type (FK)
Ticket Number (DD)
Line Number (DD)
Time of Day (SQL Date-Time)
Sales Quantity (fact)
Net Sales Dollar amount (fact)
Discount Dollar amount (fact)
Cost Dollar amount (fact)
Gross Profit Dollar amount (fact)
Tax Dollar amount (fact)

- Every table defined by its grain
  - in business terms
  - in terms of the dimension foreign keys and other fields
- A set of foreign keys (FK)
  - context for the fact
  - Join to Dimension Tables
- Degenerate Dimensions
  - Part of the key
  - Not a foreign key to a Dimension table
- Primary Key
  - a subset of the FKs
  - must be defined in the table
- Fact Attributes
  - measurements

Building Fact Tables

- Two types of load
- Initial load of historic data
  - ETL for all data up till now
  - Done when DW is started the first time , human
  - Very heavy - large data volumes
- Incremental update
  - Move only changes since last load
  - Done periodically (e.g., month or week) after DW start, automatically
  - Less heavy - smaller data volumes
- Dimensions must be updated **before** facts
  - The relevant dimension rows for new facts must be in place
  - Special key considerations if initial load must be performed again

Guaranteeing Referential Integrity



Extract staging
Data Quality staging
Conforming staging
1. Check load tables before loading
Final load tables
Process
2. Check *while* loading using DBMS RI Check
3. Check Integrity of Tables at later times
Dimension and fact tables

1. **Check Before Loading**
   - Check before you add fact records
   - Check before you delete dimension records
   - Best approach
2. **Check While Loading**
   - DBMS enforces RI
   - Elegant but typically SLOW
   - Exception: Red Brick database system is capable of loading 100 million records an hour into a fact table where it is checking referential integrity on all the dimensions simultaneously!
3. **Check After Loading**
   - No RI in the DBMS
   - Periodic checks for invalid foreign keys looking for invalid data
   - Ridiculously slow

Types of Data Sources

- Non-cooperative sources
  - Snapshot sources – provides only full copy of source, e.g., files
  - Specific sources – each is different, e.g., legacy systems
  - Logged sources – writes change log, e.g., DB log
  - Queryable sources – provides query interface, e.g., RDBMS
- Cooperative sources
  - Replicated sources – publish/subscribe mechanism
  - Call back sources – calls external code (ETL) when changes occur
  - Internal action sources – only internal actions when changes occur
    - DB triggers is an example
- Extract strategy depends on the source types

Extract

- Goal: fast extract of relevant data
  - Extract from source systems can take **long** time
- Types of extracts:
  - Extract applications (SQL): co-existence with other applications
  - DB unload tools: faster than SQL-based extracts
    - e.g., MS SQL Export Wizard, MySQL DB dump
- Extract applications the only solution in some scenarios
- **Too** time consuming to ETL all data at each load
  - Extraction can take days/weeks
  - Drain on the operational systems and DW systems
- Extract/ETL only changes since last load (delta)

Common Transformations

- Data type conversions
  - EBCDIC → ASCII/UniCode
  - String manipulations
  - Date/time format conversions
    - E.g., unix time 1201928400 = what time?
- Normalization/denormalization
  - To the desired DW format
  - Depending on source format
- Building keys
  - Table matches production keys to surrogate DW keys
  - Correct handling of history - especially for total reload

Data Quality

- Data almost **never** has decent quality
- Data in DW must be:
  - Precise
    - DW data must match known numbers - or explanation needed
  - Complete
    - DW has all relevant data and the users know
  - Consistent
    - No contradictory data: aggregates fit with detail data
  - Unique
    - The same things is called the same and has the same key (customers)
  - Timely
    - Data is updated ”frequently enough” and the users know when

The high cost of low quality data

- Wrong price data in retail databases may cost American consumers as much as $2.5 billion in overcharges annually. Data audits show 4 out of 5 errors in prices are overcharges. (Information Week Sept 1992)
- The Gartner group estimates for the worldwide costs to modify software and change databases to fix the Y2K problem was $400-$600 billion. T.Capers Jones says this estimate is low, it should be $1.5 trillion. The cost to fix this single pervasive error is one eighth of the US federal deficit ($8 trillion Oct 2005).
- Another way to look at it. The 50 most profitable companies in the world earned a combined $178 billion in profits in 1996. If the entire profit of these companies was used to fix the problem, it would only fix about 12% of the problem
- And MS Excell, in year 2000, still regards 1900 as a leap year (which is not).

cleaning

- Why cleaning? Garbage In Garbage Out
- BI does not work on “raw” data
  - Pre-processing necessary for BI analysis
- Handle inconsistent data formats
  - Spellings, codings, …
- Remove unnecessary attributes
  - Production keys, comments,…
- Replace codes with text **(Why?)**
  - City name instead of ZIP code, e.g., Aalborg Centrum vs. DK-9000
- Combine data from multiple sources with common key
  - E.g., customer data from customer address, customer name, …

Types of cleaning

- Conversion and normalization
  - Most common type of cleaning
  - Text coding, date formats
    - e.g., 3/2/2008 means 3rd February or 2nd March?
- Special-purpose cleaning
  - Look-up tables, dictionaries to find valid data, synonyms, abbreviations
  - Normalize spellings of names, addresses, etc.
  - Remove duplicates, e.g., duplicate customers
- Domain-independent cleaning
  - Approximate, "fuzzy" joins on records from different sources
  - E.g., two customers are regarded as the same if their respective values match for most of the attributes (e.g., address, phone number)
- Rule-based cleaning
  - User-specifed rules: if-then style
  - Automatic rules: use data mining to find patterns in data
    - Guess missing sales person based on customer and item

cleaning

- Should a “special” value (e.g., 0, -1) be used in your data?
  - Why this issue is relevant to query/analysis operations?
- Mark facts with Data Status dimension
  - Normal, abnormal, outside bounds, impossible,…
  - Facts can be taken in/out of analyses
- Uniform treatment of NULL
  - Use explicit NULL value rather than “special” value (0,-1,…)
  - Use NULLs only for measure values (estimates instead?)
  - Use special dimension key (i.e., surrogate key value) for NULL dimension values
    - E.g., for the time dimension, instead of NULL, use special key values to represent “Date not known”, “Soon to happen”
    - Avoid problems in joins, since NULL is not equal to NULL
- Mark facts with changed status
  - New customer, Customer about to cancel contract, ……

Improving Data Quality

- Appoint “data quality administrator”
  - Responsibility for data quality
  - Includes manual inspections and corrections!
- Source-controlled improvements
  - The optimal?
- Construct programs that check data quality
  - Are totals as expected?
  - Do results agree with alternative source?
  - Number of NULL values?
- Do not fix **all** problems with data quality
  - Allow management to see “weird” data in their reports?
  - Such data may be meaningful for them? (e.g., fraud detection)

Load

- Goal: fast loading into DW
  - Loading deltas is much faster than total load
- SQL-based update is **slow**
  - Large overhead (optimization, locking, etc.) for every SQL call
  - DB load tools are much faster
- Index on tables **slows** load a lot
  - Drop index and rebuild after load
  - Can be done per index partition
- Parallellization
  - Dimensions can be loaded concurrently
  - Fact tables can be loaded concurrently
  - Partitions can be loaded concurrently

Load

- Relationships in the data
  - Referential integrity and data consistency must be ensured before loading **(Why?)**
  - Can be done by loader
- Aggregates
  - Can be built and loaded at the same time as the detail data
- Load tuning
  - Load without log
  - Sort load file first
  - Make only simple transformations in loader
  - Use loader facilities for building aggregates

ETL Tools

- ETL tools from the big vendors
  - Oracle Warehouse Builder
  - IBM DB2 Warehouse Manager
  - Microsoft Integration Services
- Offers much functionality at a reasonable price
  - Data modeling
  - ETL code generation
  - Scheduling DW jobs
- The “best” tool does not exist
  - Choose based on your own needs

No.	List of ETL Tools	Version	ETL Vendors
1.	Oracle Warehouse Builder (OWB)	11gR1	Oracle new!
2.	Data Integrator & Services (BODI)	XI 3.0	Business Objects, SAP
3.	IBM Information Server (Ascential)	8.0.1	IBM
4.	SAS Data Integration Studio	4.2	SAS Institute new!
5.	PowerCenter	8.5.1	Informatica new!
6.	Elixir Repertoire	7.2.2	Elixir new!
7.	Data Migrator	7.6	Information Builders
8.	Integration Services (SSIS)	2005	Microsoft
9.	Talend Open Studio	1.1	Talend
10.	DataFlow Manager	6	Group 1 Software (Sagent)
11.	Data Integrator	8.12	Pervasive
12.	Transformation Server	5.4	IBM DataMirror
13.	Transformation Manager	5.2.2	ETL Solutions Ltd.
14.	Data Manager/Decision Stream	8.2	IBM Cognos
15.	DT/Studio	3.1	Embarcadero Technologies
16.	ETL4ALL	4.2	IKAN
17.	DB2 Warehouse Edition	9.1	IBM
18.	Pentaho Data Integration	3.0	Pentaho new!

Vendor ETL tools

- DataStage IBM
  - big player, high ability to execute, gets good results from Gartner, visionary
- Informatica
  - another big player, gets good results from Gartner, visionary with high ability to execute
  - most expensive
- SAS ETL Server
  - fast becoming a major player, very positive results from Gartner
  - low exposure as an ETL tool (SAS a significant statistical analysis vendor)
- Information Builder's Data Migrator/ETL Manager tool suite
  - part of Enterprise Focus/WebFocus
  - not a major player but industrial strength language, data connectors, etc
- Sunopsis
  - cheap
  - relies on native RDBMS functionality
  - CIGNA people exposed to it at conferences liked it

Issues

- Pipes
  - Redirect output from one process to input of another process

    ```
    ls | grep 'a' | sort -r
    ```
- Files versus streams/pipes
  - Streams/pipes: no disk overhead, fast throughput
  - Files: easier restart, often only possibility
- Use ETL tool or write ETL code
  - Code: easy start, co-existence with IT infrastructure
  - Tool: better productivity on subsequent projects
- Load frequency
  - ETL time dependent of data volumes
  - Daily load is much faster than monthly
  - Applies to all steps in the ETL process

MS Integration Services

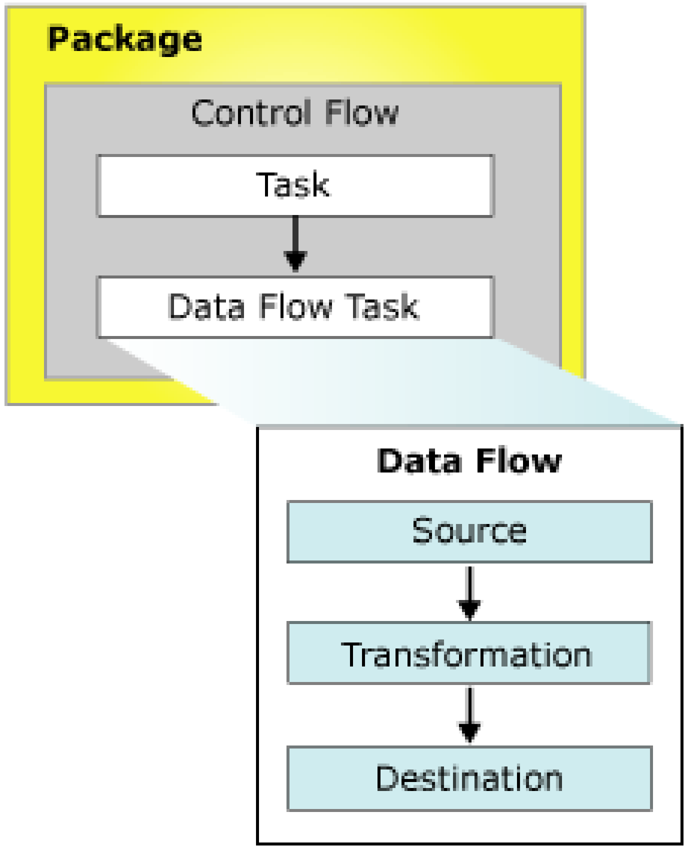
- A concrete ETL tool
- Example ETL flow
- Demo

Integration Services (IS)

- Microsoft’s ETL tool
  - Part of SQL Server 2005
- Tools
  - Import/export wizard - simple transformations
  - BI Development Studio - advanced development
    - New/Open an Integration Services Project
- Functionality available in several ways
  - Through GUI - basic functionality
  - Programming - advanced functionality

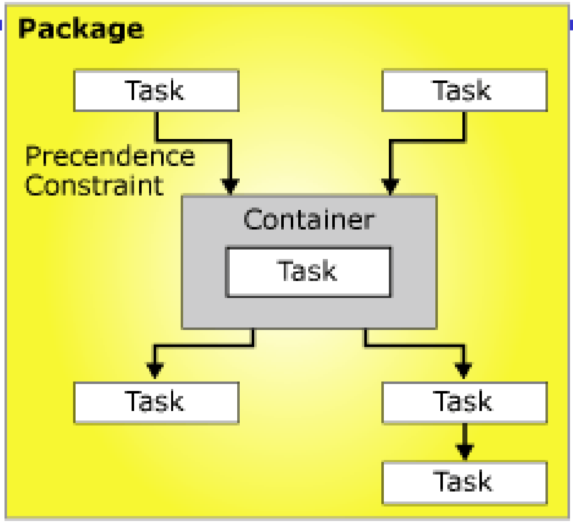
Packages

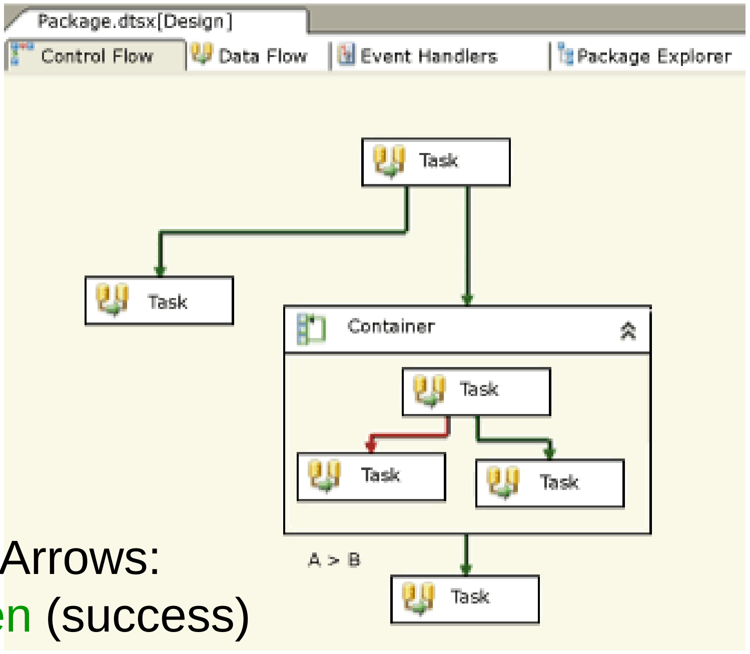
- A package as a "dtsx" file
- Encapsulation, reuse
- The central concept in IS
- Package a collection of:
  - Data flow: Sources, Destinations, Transformations
  - Connections
  - Control flow: Tasks, Workflows
  - Variables
  - ……



Package Control Flow

- "Containers" provide
  - Structure to packages
  - Services to tasks
- Control flow
  - Foreach loop container
    - Repeat tasks by using an enumerator
  - For loop container
    - Repeat tasks by testing a condition
  - Sequence container
    - Groups tasks and containers into control flows that are subsets of the package control flow
- Task host container
  - Provides services to a single task

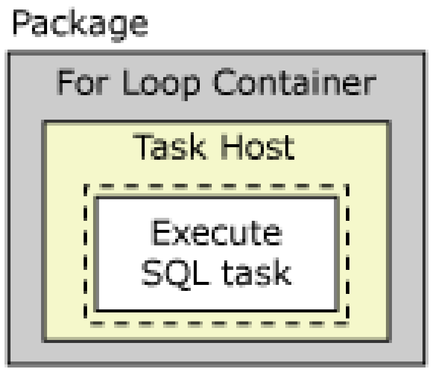




Arrows:
green (success)
red (failure)

Tasks

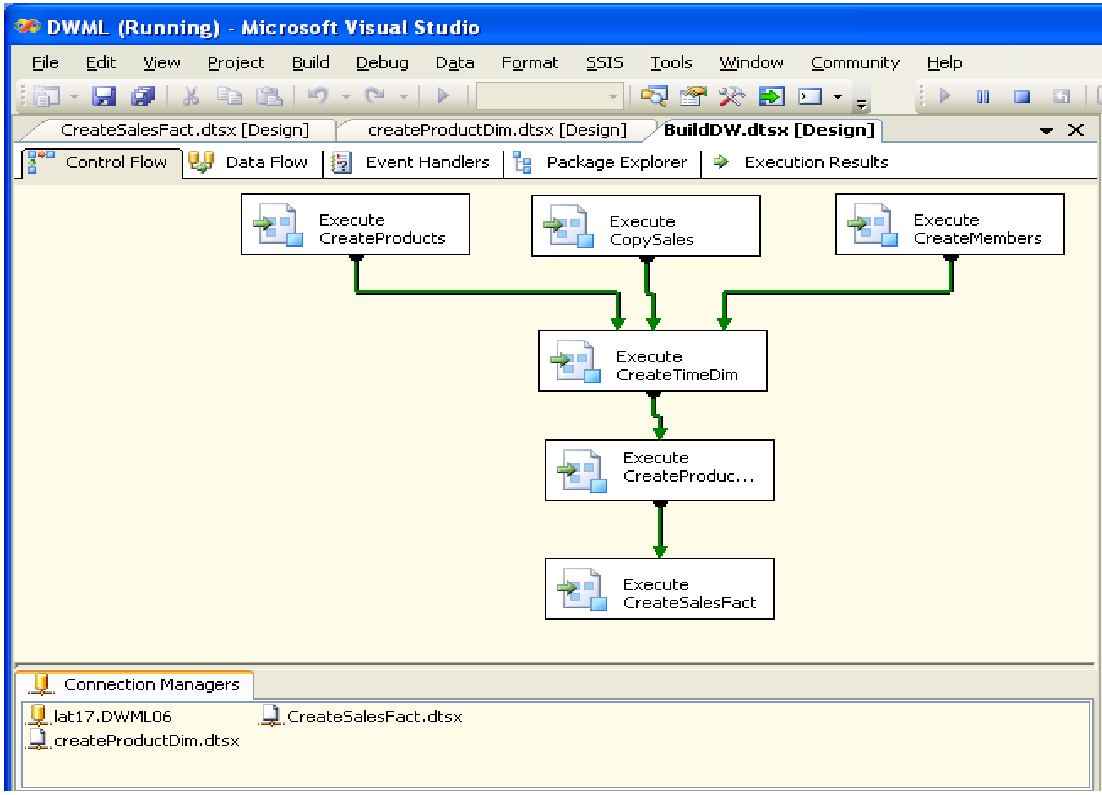
- Workflow Tasks
  - Execute package – execute other IS packages, good for structure!
  - Execute Process – run external application/batch file
- SQL Servers Tasks
  - Bulk insert – fast load of data
  - Execute SQL – execute any SQL query
- Data Flow – runs data flows
- Data Preparation Tasks
  - File System – operations on files
  - FTP – up/down-load data
- Scripting Tasks
  - Script – execute VN .NET code
- Maintenance Tasks – DB maintenance



Why there are so many different tasks?

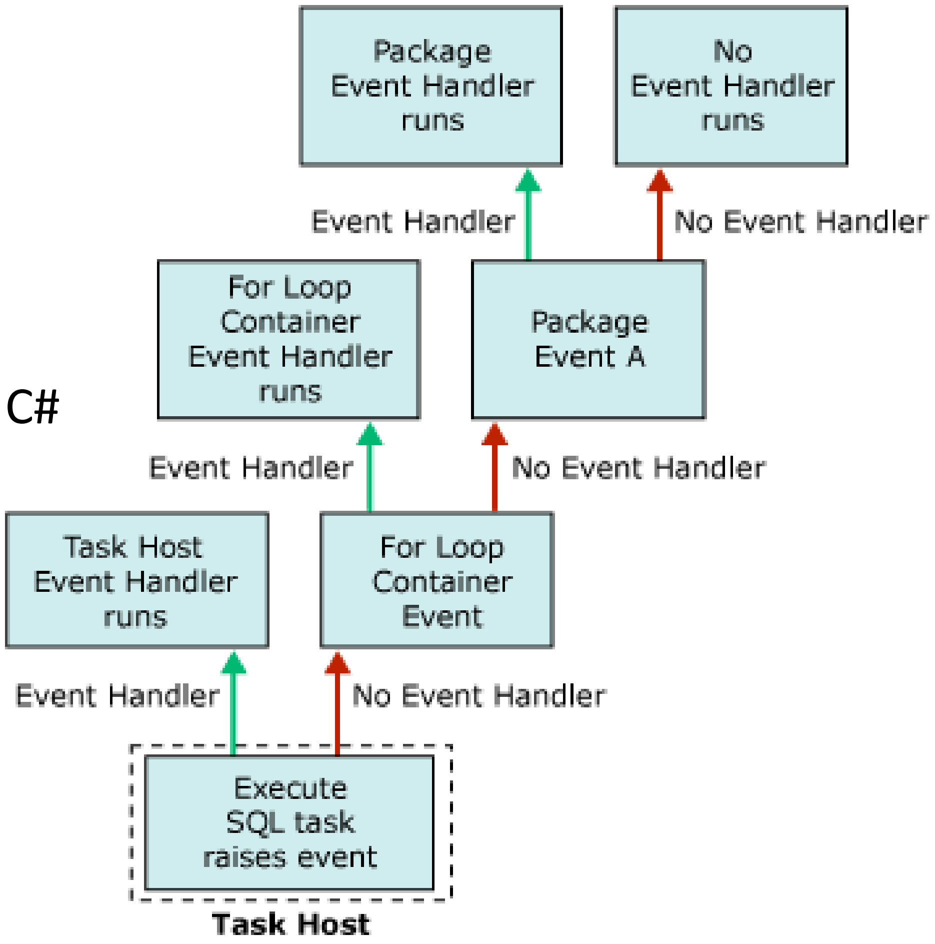
A Simple IS Case

- Use BI Dev Studio/Import Wizard to copy FClub tables
- Look at package structure
  - Available from mini-project web page
- Look at package parts
  - DROP, CREATE, source, transformation, destination
- Execute package
  - Error messages?
- Steps execute in parallel
  - Dependencies can be set up



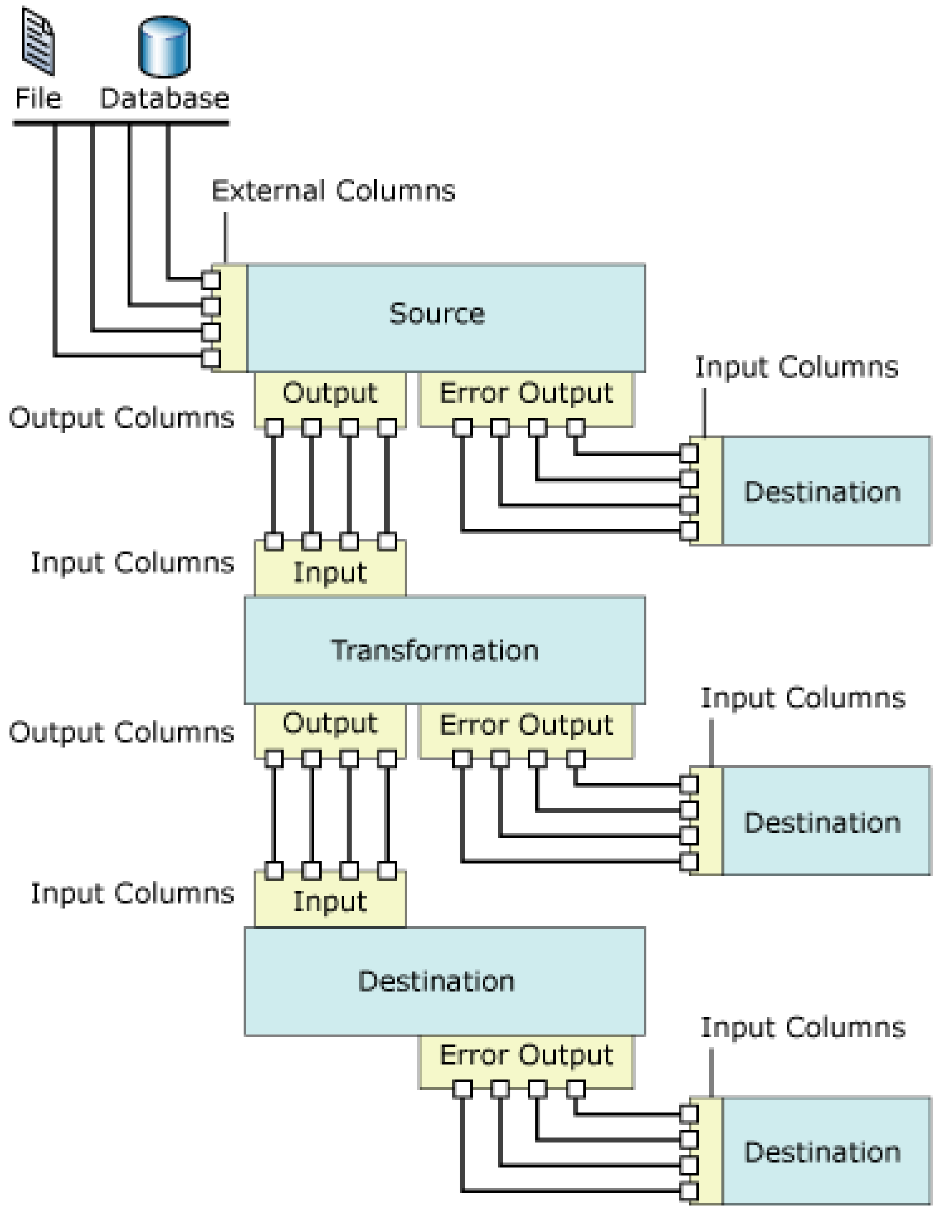
Event Handlers

- Executables (packages, containers) can raise events
- Event handlers manage the events
- Similar to those in languages JAVA, C#



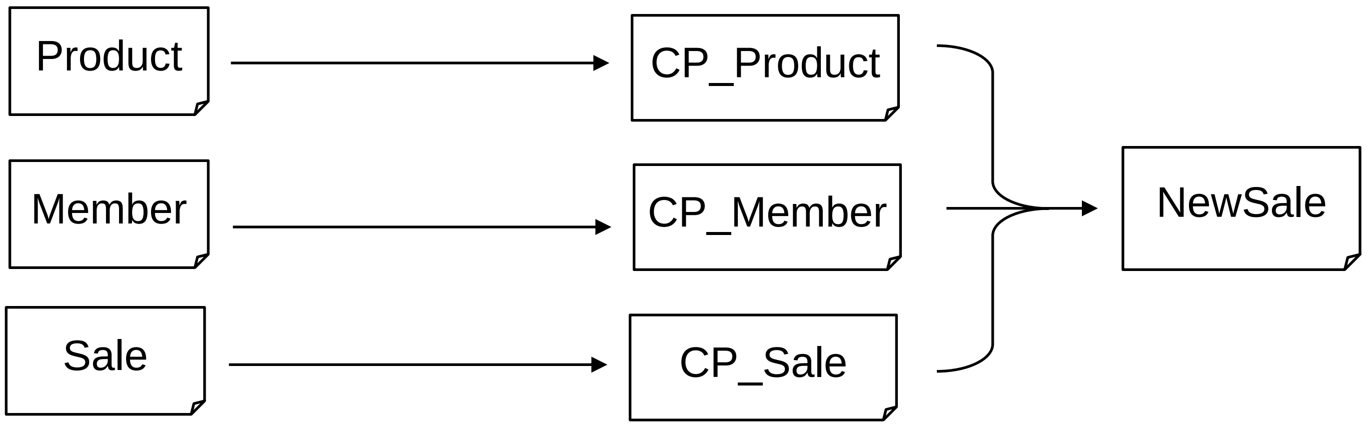
Data Flow Elements

- Sources
  - Makes external data available
  - All ODBC/OLE DB data sources: RDBMS, Excel, Text files, ……
- Transformations
  - Update, summarize, cleanse, merge, distribute
- Destinations
  - Write data to specific store
  - Create in-memory data set
- Input, Output, Error output



ETL Demo. Session

1. Load data into the Product dimension table
    1. Use the SSIS Import and Export Wizard
    2. Construct the **DW key** for the table by using “IDENTITY”
    3. Copy data to the Product dimension table
2. Load data into the Member dimension table
3. Copy the “raw” Sale fact table (for convenience only)
4. Load data into the NewSale fact table
    1. Join “raw” sales table with other tables to get DW keys for each sales record
    2. Output of the query written into the fact table
5. Create a StartUp package that organizes the 4 packages above



ETL Demo. Session Today

- Method 1: **Copy** a source table into the destination table, or
- Sample SQL for creating the Member dimension table:

```
CREATE TABLE CP_Member (
        [dw_member_id] int IDENTITY,
        [id] int NOT NULL,
        [balance] int NOT NULL
        ……
)
```

- Method 2: **Write** into the destination table the result of a SQL query
- Sample SQL for inserting data into the NewSale fact table:

```
select
        CP_Member.dw_member_id, CP_Product.dw_product_id,
        sum(CP_Sale.price) as sales
from
        CP_Sale, CP_Member, CP_Product
where
        CP_Sale.member_id=CP_Member.id and
        CP_Sale.product_id=CP_Product.id
group by
        CP_Member.dw_member_id, CP_Product.dw_product_id
```

A Few Hints on ETL Design

- **Don't** implement all transformations in one step!
  - Build first step and check that result is as expected
  - Add second step and execute both, check result (How to check?)
  - Add third step ……
- Test SQL statements before putting into IS
- Do **one** thing at the time
  - Copy source data one-by-one to the data staging area (DSA)
  - Compute deltas
    - Only if doing incremental load
  - Handle versions and DW keys
    - Versions only if handling slowly changing dimensions
  - Implement complex transformations
  - Load dimensions
  - Load facts

Summary

- The ETL Process
- General ETL issues
  - Building dimensions
  - Building fact tables
  - Extract
  - Transformations/cleaning
  - Load
- MS Integration Services

ETL Part of Mini Project

- *Core:*
  - Build an ETL flow using MS DTS that can do an initial (first-time) load of the data warehouse
  - Include logic for generating special DW surrogate integer keys for the tables
  - Discuss and implement basic transformations/data cleaning
- *Extensions:*
  - Extend the ETL flow to handle incremental loads, i.e., updates to the DW, both for dimensions and facts
  - Extend the DW design and the ETL logic to handle slowly changing dimensions of Type 2
  - Implement more advanced transformations/data cleaning
  - Perform error handling in the ETL flow