Extract, Transform, Load (ETL)
ETL Overview

• The ETL Process
• General ETL issues
  ■ Building dimensions
  ■ Building fact tables
  ■ Extract
  ■ Transformations/cleansing
  ■ Load
• MS Integration Services
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.
  - Cleansing of data
- Load
  - Load data into DW
  - Build aggregates, etc.
ETL In The Architecture

ETL side

Data sources

- Extract
- Transform
- Load

Data Staging Area

Presentation servers

Data Warehouse Bus

- Conformed dimensions and facts

Data marts with aggregate-only data

Data marts with atomic data

Metadata

Query side

Query Services

- Warehouse Browsing
- Access and Security
- Query Management
- Standard Reporting
- Activity Monitor

Reporting Tools

Desktop Data Access Tools

Data mining

Operational system
Data Staging Area (DSA)

- Transit storage for data in the ETL process
  - Transformations/cleansing done here
- No user queries
- Sequential operations on large data volumes
  - Performed by central ETL logic
  - No need for locking, logging, etc.
  - RDBMS or flat files? (DBMS have become better at this)
- Finished dimensions copied from DSA to relevant marts
- Allows centralized backup/recovery
  - Backup/recovery facilities needed
  - Better to do this centrally in DSA than in all data marts
- Do we need a plan for ETL?
ETL Construction Process

- **Plan**
  1. Make high-level diagram of source-destination flow
  2. Test, choose and implement ETL tool
  3. Outline complex transformations, DW key generation and job sequence for every destination table

- **Construction of dimensions**
  4. Construct and test building static dimension
  5. Construct and test change mechanisms for one dimension
  6. Construct and test remaining dimension builds

- **Construction of fact tables and automation**
  7. Construct and test initial fact table build
  8. Construct and test incremental update
  9. Construct and test aggregate build (**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)
  - Add product type
  - Aggregate sales per product per day

Destination

- Raw-Sales (RDBMS)
  - Check R.I.
  - Extract time

- Product
- Sales
- Time
Building Dimensions

• **Static dimension table**
  - DW key assignment: production keys to DW keys 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 dimensions**
  - Small dimensions: replace
  - Large dimensions: load only changes

Product table

<table>
<thead>
<tr>
<th>Type</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flask</td>
<td>Tuborg</td>
</tr>
<tr>
<td>Flask</td>
<td>Carlsberg</td>
</tr>
<tr>
<td>Milk</td>
<td>Skim-Milk</td>
</tr>
<tr>
<td>Milk</td>
<td>Cacao-Milk</td>
</tr>
<tr>
<td>Vand</td>
<td>Sodavand</td>
</tr>
</tbody>
</table>

Product dimension of FClub vs. Product dimension of a supermarket
Building Fact Tables

- Two types of load
  - Initial load
    - ETL for all data up till now
    - Done when DW is started the first time
    - Very heavy - large data volumes
  - Incremental update
    - Move only changes since last load
    - Done periodically (e.g., month or week) after DW start
    - 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
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)
Computing Deltas

- Delta = changes since last load
- Store sorted total extracts in DSA
  - Delta can easily be computed from current+last extract
  - + Always possible
  - + Handles deletions
  - - High extraction time
- Put update timestamp on all rows (in sources)
  - Updated by DB trigger
  - Extract only where “timestamp > time for last extract”
  - + Reduces extract time
  - - Cannot (alone) handle deletions. **WHY?**
  - - Source system must be changed, operational overhead
Changed Data Capture

• Messages
  - Applications insert messages in a “queue” at updates
  - Works for all types of updates and systems
  - Operational applications must be changed

• DB triggers
  - Triggers execute actions on INSERT/UPDATE/DELETE
  - Operational applications need not be changed
  - Enables real-time update of DW
  - Operational overhead

• Replication based on DB log
  - Find changes directly in DB log which is written anyway
  - Operational applications need not be changed
  - No operational overhead
  - Not possible in some DBMS
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
Cleansing

- Why cleansing? 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 Cleansing

- **Conversion and normalization**
  - Most common type of cleansing
  - Text coding, date formats
    - e.g., 3/2/2008 means 3rd February or 2nd March?

- **Special-purpose cleansing**
  - Look-up tables, dictionaries to find valid data, synonyms, abbreviations
  - Normalize spellings of names, addresses, etc.
  - Remove duplicates, e.g., duplicate customers

- **Domain-independent cleansing**
  - 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 cleansing**
  - User-specified rules: if-then style
  - Automatic rules: use data mining to find patterns in data
    - Guess missing sales person based on customer and item
Cleansing

- 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
Issues

- **Pipes**
  - Redirect output from one process to input of another process
    
    ```sh
    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 for:
  - Sources, Destinations
  - Connections
  - Control flow
  - Tasks, Workflows
  - Transformations
  - ......
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
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**
Transformations

• Business intelligence transformations
  - Term Extraction - extract terms from text
  - Term Lookup – look up terms and find term counts

• Row Transformations
  - Character Map - applies string functions to character data
  - Derived Column – populates columns using expressions

• Rowset Transformations \((\text{rowset} = \text{tabular data})\)
  - Aggregate - performs aggregations
  - Sort - sorts data
  - Percentage Sampling - creates sample data set by setting %

• Split and Join Transformations
  - Conditional Split - routes data rows to different outputs
  - Merge - merges two sorted data sets
  - Lookup Transformation - looks up ref values by exact match

• Other Transformations
  - Export Column - inserts data from a data flow into a file
  - Import Column - reads data from a file and adds it to a data flow
  - Slowly Changing Dimension - configures update of a SCD
ETL Demo. Session Today

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

\[\text{Product} \rightarrow \text{CP\_Product} \rightarrow \text{CP\_Member} \rightarrow \text{CP\_Sale} \rightarrow \text{NewSale}\]
ETL Demo. Session Today

- Method 1: **Copy** a source table into the destination table, or
- Sample SQL for creating the Member dimension table:
  ```sql
  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:
  ```sql
  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
  ```
Demo. and the Mini-Project

• The demo. session focuses on the construction of DW keys, and the initial load of the DW

• More to do in the mini-project:
  - Transformation of attribute value (e.g., price)
  - Create the time dimension
    - Determine the min/max timestamp and then populate the time dimension, OR
    - Perform a SQL query to extract all timestamps from the sales table
  - Create derived values in time hierarchy, product hierarchy, ......
  - ......
  - You name it
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 cleansing

• **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 cleansing
  - Perform error handling in the ETL flow
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/cleansing
  ■ Load
• MS Integration Services