

# Building an ETL Tool

**Ahimanikya Satapathy**

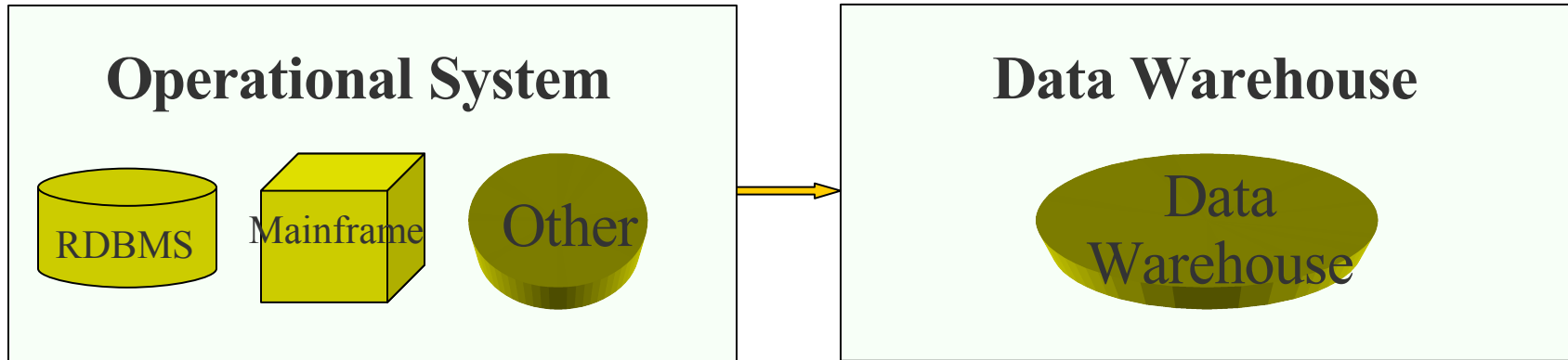
SOA/Business Integration  
Sun Microsystems

# What is ETL?

Short for **E**xtract, **T**ransform, **L**oad; three database functions that are combined into one tool that automates the process to pull data out of one database and place it into another database.

- **Extract** -- the process of reading data from a specified source database and extracting a desired subset of data.
- **Transform** -- the process of converting the extracted/ acquired data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining with other data.
- **Load** -- the process of writing the data into the target database.

# Migrating from one database to another



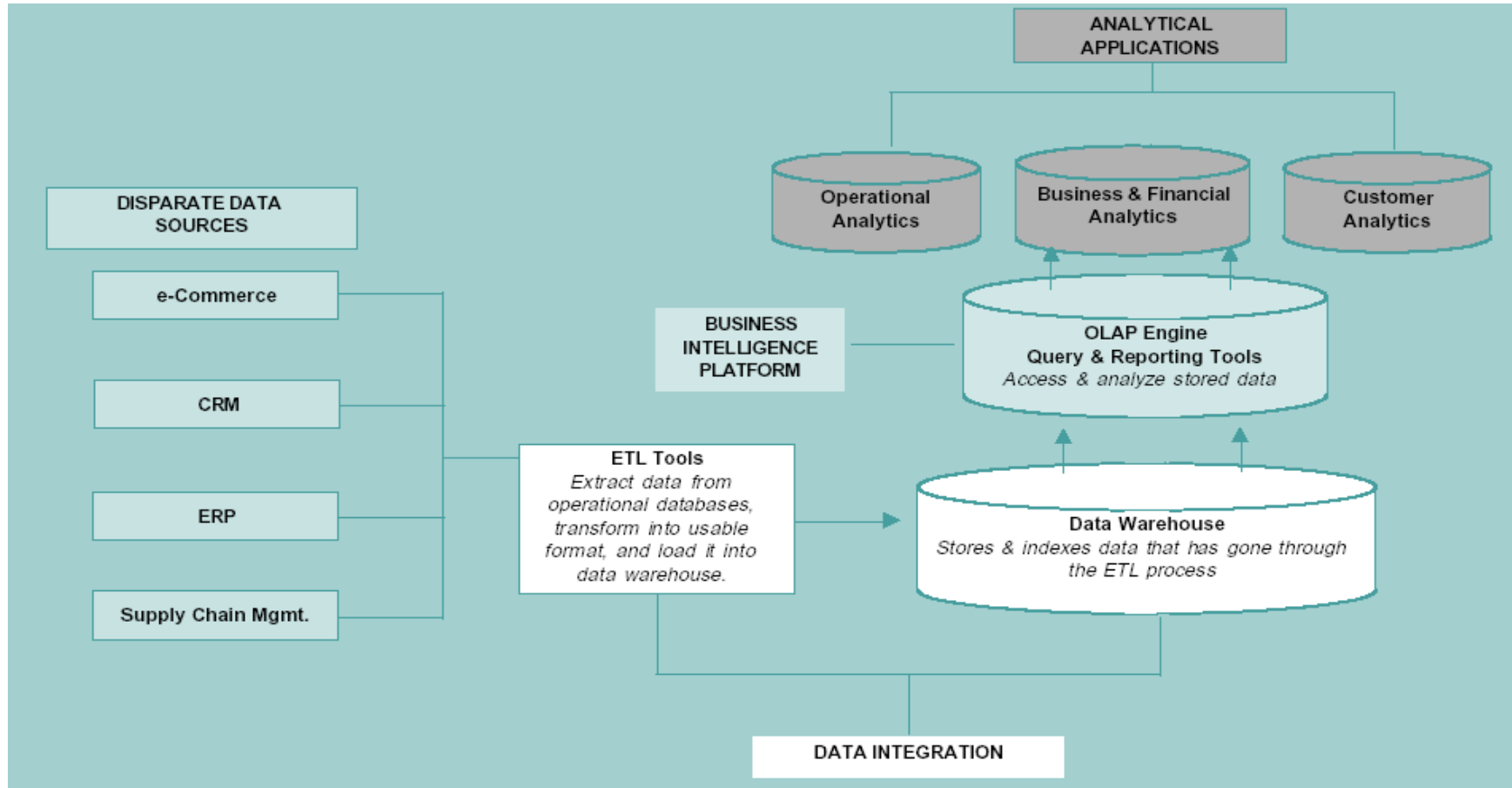
- Transaction level data
  - Optimized for Transaction Response Time
  - Current
  - Normalized or De-Normalized data
- Aggregated Data
  - Optimized for Query Response Time
  - Historical
  - Normalized or De-Normalized data

# Why use an ETL Tool?

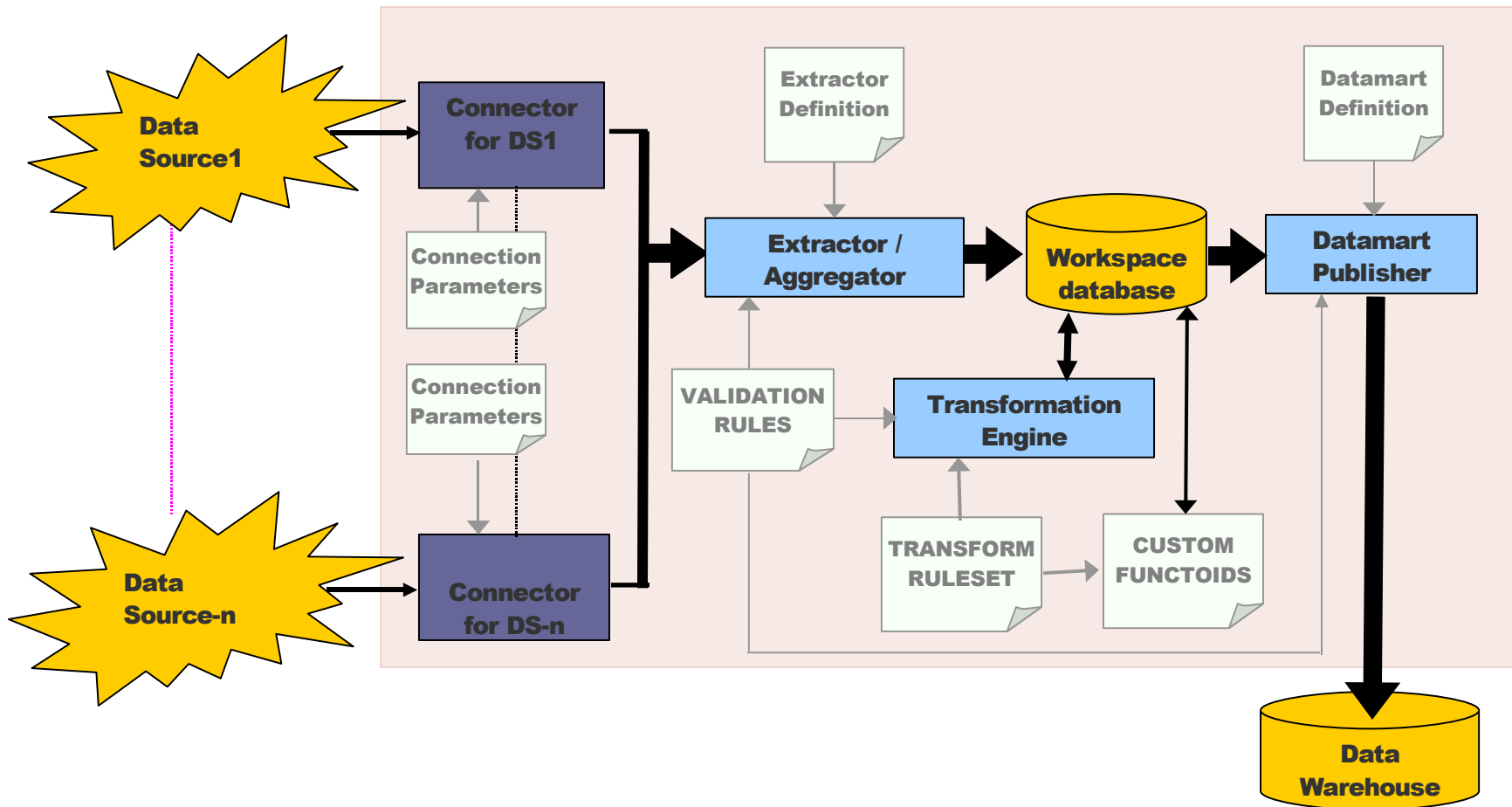
Use an ETL tool to:

- Simplify the process of migrating data
- Standardize the method of data migration
- Store all data transformation logic/rules as Meta data
- Enable users, managers and architects to understand, review, and modify the various interfaces
- Reduce cost and effort associated with building interfaces (custom coding in conventional languages could cost 2 - 5 times as much)

# Data Integration Segmentation



# ETL Dataflow Diagram



# Problems with hand-coded ETL

A common mistake is to write custom programs to perform the extraction, transformation, and load functions. Writing an ETL program by hand may seem to be a viable option because the program does not appear to be too complex and programmers are available. However, there are serious problems with hand-coded ETL programs.

## Problems with hand-coded ETL (Contd.)

- Unlike OLTP applications, the functions to be supported by individual data marts cannot be predicted in advance. In a typical data mart, over 50% of the required functionality is defined by end users after the data mart goes into production. To keep up with the high volume of changes initiated by end users, hand-written ETL programs have to be constantly modified and in many cases rewritten. The effort required to maintain these programs often becomes a major burden for the project
- Metadata is not generated automatically by hand-generated ETL programs. Metadata is the key to integrating data marts across business units. If metadata is not available, it is difficult to avoid the development of “stovepipe” data marts that satisfy the needs of individual business units, but cannot be integrated across the enterprise
- Hand-coded ETL programs are likely to have a slower speed of execution, compared with directly executable code generated by off-the-shelf ETL tools. Hand-generated programs are typically single-threaded, while modern ETL tools generate multi-threaded, directly executable code that can run on parallel, high-speed engines

# Off-The-Shelf ETL Tools

Off-the-shelf ETL tools are increasingly being used to extract, cleanse, transform, and load data into target databases. An important function of these tools is to generate and maintain centralized metadata.

## Off-The-Shelf ETL Tools (Contd.)

- The ETL tool provides coordinated access to multiple data sources. Functions supported by ETL tools include extraction of data from multiple source environments, data cleansing, reorganization, transformation, aggregation, calculation, automatic loading of data into the target database, and automatic generation of executable code to perform parallel processing of transformations on multiple engines
- ETL tools are used to generate and maintain a central metadata repository. The metadata repository provides a “single version of the truth” that can be used to define enterprise-wide source data definitions, data models for target databases, and transformation rules that convert source data into target data. A metadata exchange architecture is used to synchronize central business rules with local business rules, maintained as local metadata by end-user BI tools

## Off-The-Shelf ETL Tools (Contd.)

- The ETL tool also addresses the dirty data problem - data from source files can be cleansed and inconsistencies in the data resolved as part of the extraction and transformation process, using procedural data cleansing techniques. Name and address correction, de-duping, and house-holding functions require use of an external data cleansing tool
- Analysts define source-to-target mappings, data cleansing rules, and data transformation rules using a graphical point-and-click interface provided by the ETL tool. When all mappings and transformations have been specified, the ETL tool automatically generates the data extract/transformation/load programs, which typically run in batch mode.

# Extraction

- Multiple sources
- Multiple extract types
  - Full extract (refresh)
  - Incremental Extract
    - CRC codes
    - Window algorithm
    - Log-based extraction
    - Snapshot-based extraction
    - Trigger-Based extraction
    - Etc.
- Extract in files or databases

# Extract from Operational System

- Design Time
  - Create/Import Data Sources definition
  - Define Stage or Work Areas
  - Validate Connectivity
  - Preview/Analyze Sources
  - Define Extraction Scheduling
    - Determine Extract Windows for source system
    - Batch Extract (Overnight, weekly, monthly)
    - Continuous extracts (Trigger on Source Table)
- Run Time
  - Connect to the predefined Data Sources as scheduled
  - Get raw data save locally in workspace DB

# Transformation

- Eliminate inconsistencies in the data from multiple sources
- Convert data into a consistent, standardized form
- Fold/Unfold
- Cleanse
- Merge/Purge
- Aggregate
- Calculate
- Data type conversion
- Data content audit
- Null value handling
- Customized transformation

# Data Content Example

- Domain value redundancy – Unit of Measure
  - Dozen, Doz. , Dz. , 12
- Non Standard Data Format
  - Phone Numbers
    - 18005553434, 800-555-3434
  - Tax Payer Id Number
    - 123-44-6789, 123446789, 12-3446789

# Transformation

- Design Time
  - Specify Criteria/Filter for aggregation
  - Define operators (Mostly Set/SQL based)
  - Map columns using operators/Lookups
  - Define other transformation rules
  - Define mappings and/or add new fields
- Run Time
  - Transform (Cleanse, consolidate, Apply Business Rule, De-Normalize/Normalize) Extracted Data by applying the operators mapped in design time.
  - Aggregate (create & populate raw table)
  - Create & populate Staging table

# Load to Data Warehouse

- Design Time
  - Design Warehouse
  - Map Staging Data to fact or dimension table attributes
- Run Time
  - Publish Staging data to Data mart (update dimension tables along with the fact tables)

# Real Time ETL

**Bringing real-time data into your strategic business apps is becoming a necessity. But where do you begin?**

**Nightly loads are no longer frequent enough for many just-in-time manufacturing, supply chain, and CRM applications. Although achieving the Holy Grail of a continuously loaded warehouse is challenging with today's technologies, now is the time to get started.**

# Real Time ETL - Challenges

## **Challenge 1: Real-Time ETL**

Moving from nightly to real-time warehouse loading brings about a host of new challenges.

## **Challenge 2: Data Modeling for Real Time**

Real-time intraday data can be stored in a real-time partition for easy data modeling.

## **Challenge 3: Query Tools vs. Real-Time Data**

Today's query and OLAP tools, not having been designed with real-time warehousing in mind, can produce unanticipated results.

## **Challenge 4: Scalability**

Real-time warehousing introduces new scalability challenges, including contention among concurrent warehouse updates and reads.

## **Challenge 5: Real-Time Alerting**

When properly implemented with real-time data, "narrowcast" alerting applications begin to really shine.

# Feature List

Support for data extraction, cleansing, aggregation, reorganization, transformation, calculation, and load operations, including the following functions:– Access data from multiple operational data sources:– Access data from multiple operational data sources – Re-map source data into a common format

## Feature List (Contd.)

- Standardize data to enable load to conformed target databases
- Filter data, convert codes, perform table lookups, calculate derived values
- Automated, slowly changing dimension support (Type I, Type II, Type III)
- Incremental aggregation - computation of aggregates by the ETL tool in one pass of the source data
- Support for Unicode - multi-byte character sets localized for Japanese and other languages
- Support graphical job sequencer, re-usable containers, and nesting of sessions  
Validate data to check content and range of field values
- Perform procedural data cleansing functions

## Feature List (Contd.)

- Support complete development environment, including versioning and run-time debugger
- Load cleansed data to the target data mart or central DW
- Produce audit and operational reports for each data load
- Automatic generation of centralized metadata
- Automatic generation of data extract programs (Pre-profiling)
- Native interfaces to legacy files, relational databases, ERP sources (e.g., SAP R/3 and PeopleSoft), e-Business applications, Web log files, IBM MQ-Series, XML sources, etc.
- **Support for near real-time click-stream data warehousing**
- Support for an Enterprise e-Business environment, including integration at the metadata level with BI tools, ERP applications, CRM applications, analytic applications, corporate portals, etc.

## Feature List (Contd.)

- Platform independence and scalability to enterprise data warehousing applications
- Directly executable, in-memory, multi-threaded processing for fast, parallel operation
- No requirement to generate and compile source code
- No requirement for intermediate disc files
- Support for concurrent processing of multiple source data streams, without writing procedural code
- Specification of ETL functions using pre-packaged transformation objects, accessible via an intuitive graphical user interface
- Extensible transformation objects at a high level of significance
- Ability to specify complex transformations using only built-in transformation objects. The goal is to specify transformations without writing any procedural code

## Feature List (Contd.)

- Support for change management functions or "versioning".
- Automatic generation of central metadata, including source data definitions, transformation objects, target data models, and operational statistics
- Metadata exchange architecture that supports automatic synchronization of central metadata with local metadata for multiple end-user BI tools
- Central management of distributed ETL engines and metadata using a central console and a global metadata repository
- End-user access to central metadata repository via a right-mouse click
- Metadata exchange API compliant with COM, UML, and XML
- Support of metadata standards, including OLE DB for OLAP

## Feature List (Contd.)

- Ability to schedule ETL sessions on time or the occurrence of a specified event, including support for command-line scheduling using external scheduling programs
- Ability to schedule FTP sessions on time or event
- Integration with data cleansing tools (e.g. eView, Vality etc)
- Import of complete data models from external data modeling tools
- Strong data warehouse administration functions
- Support for the analysis of transformations that failed to be accepted by the ETL process
- Extensive reporting of the results of an ETL session, including automatic notification of significant failures of the ETL process

**Thank You!**

**Ahimanikya Satapathy**

Ahimanikya.Satapathy@Sun.Com