Advanced Data Management Technologies

Unit 8 — Extract, Transform, Load

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE

Acknowledgements: I am indebted to M. Böhlen for providing me the lecture notes.

Outline

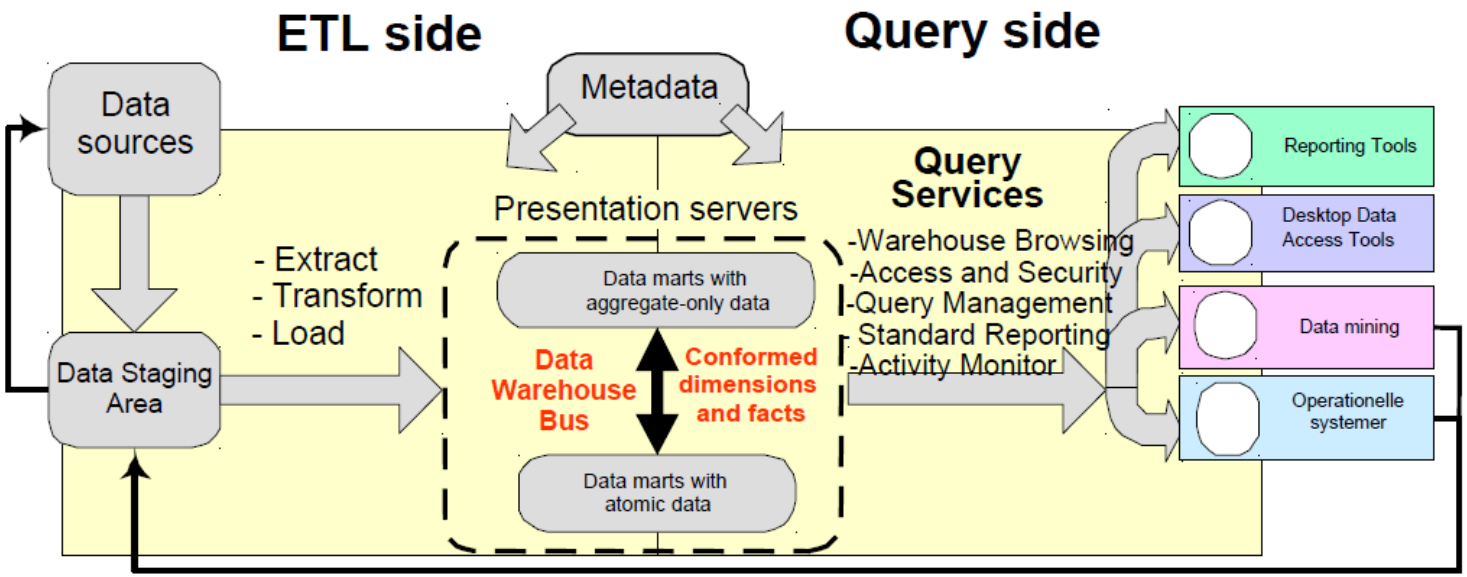
Outline

The ETL Process

- **Extract**
  - Extract relevant data
- **Transform**
  - Transform data to DW format
  - Build keys, etc.
  - Cleansing of data
- **Load**
  - Load data into DW
  - Build aggregates, etc.

- The ETL system is the foundation of the DW/BI project
  - its success makes or breaks the data warehouse.
- The most underestimated and time-consuming process in DW development
  - Often, 80% of development time is spent on ETL

ETL – Big Picture



Data Staging Area

- **Data staging area** (DSA) is a transit storage for data underway in the ETL process.
  - Transformations/cleansing done here
- No user queries (some do it)
- Sequential operations (few) on large data volumes
  - Performed by central ETL logic
  - Easily restarted
  - No need for locking, logging, etc.
  - RDBMS or flat files? (DBMSs have become better at this)
- Finished dimension and fact data copied from DSA to relevant data marts.

The 34 ETL Subsystems

- Kimball et al. report 34 subsystems to compose the ETL system
  - Extracting
    - Gathering raw data from the source systems and usually writing it to disk in the ETL environment.
    - 3 subsystems
  - Cleaning and conforming
    - Sending source data through a series of processing steps in the ETL system to improve the quality of the data and merging data.
    - 4 subsystems
  - Delivering
    - Physically structuring and loading the data into the dimensional model.
    - 13 subsystems
  - Managing (usually considered a separate component)
    - Managing the related systems and processes of the ETL environment.
    - 13 subsystems

ETL Construction Process

- Plan
  - Make high-level diagram of source-destination flow.
  - Test, choose and implement ETL tool.
  - Outline complex transformations, key generation and job sequence for every destination table.
- Construction of dimensions
  - Construct and test building static dimension.
  - Construct and test change mechanisms for one dimension.
  - Construct and test remaining dimension builds.
- Construction of fact tables and automation
  - Construct and test initial fact table build.
  - Construct and test incremental update.
  - Construct and test aggregate build (will be done later).
  - Design, construct, and test ETL automation.

Building Dimension Tables

- Static dimension table
  - Relatively easy
  - Assignment of production keys to DW keys using a mapping table.
  - Combination of data sources: find common key.
  - Check one-one and one-many relationships (using sorting).
- Handling dimension changes
  - Find newest DW key for a given production key.
  - Table for mapping production keys to DW keys must be updated.
- Load of dimensions
  - Small dimensions: replace
  - Large dimensions: load only changes

Building Fact Tables

- Two types of load are distinguished
  - Initial load
    - ETL for all data up till now.
    - Done when DW is started the first time.
    - Often problematic to get correct historical data.
    - Very heavy – large data volumes
  - Incremental update
    - Move only changes since last load.
    - Done periodically (. . . /month/week/day/hour/. . .) 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.

Outline

Extract

- Goal: fast extract of relevant data
  - Extract from source systems can take a long time.
- Different types of extraction methods
  - Extract applications (SQL): co-existence with other applications.
  - DB unload tools: much faster than SQL-based extracts.
  - Extract applications sometimes the only solution.
- Often too time consuming to ETL all data at each load.
  - Extracts can take days/weeks.
  - Drain on the operational systems.
  - Drain on DW systems.
- Alternative: Extract/ETL only changes since last load (delta).

Computing Deltas

- Much faster to only "ETL" changes since last load
  - A number of methods can be used.
- Store sorted total extracts in DSA
  - Delta can easily be computed from current+last extract

  \+ Always possible
  \+ Handles deletions
  – Does not reduce extract time
- Put update timestamp on all rows to reduce extract time
  - Updated by DB trigger
  - Extract only where "timestamp $>$ time for last extract"

  \+ Reduces extract time
  \+ Less operational overhead
  – Deletions cannot be handled (alone)
  – Source system must be changed, e.g., change of DB schema

Capturing Changed Data

- Messages
  - Applications insert messages in a "queue" at updates
  - \+ Works for all types of updates and systems
  - – Operational applications must be changed+operational overhead
- 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 (SQL Server, Oracle, DB2 can do it)

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.

Outline

Data Transformations

- Data type conversions
  - EBCDIC → ASCII/UniCode
  - String manipulations
  - Date/time format conversions
- Normalization/denormalization
  - To the desired DW format
  - Depending on source format
- Building keys
  - Table that maps production keys to surrogate DW keys.
  - Observe correct handling of history – especially for total reload.

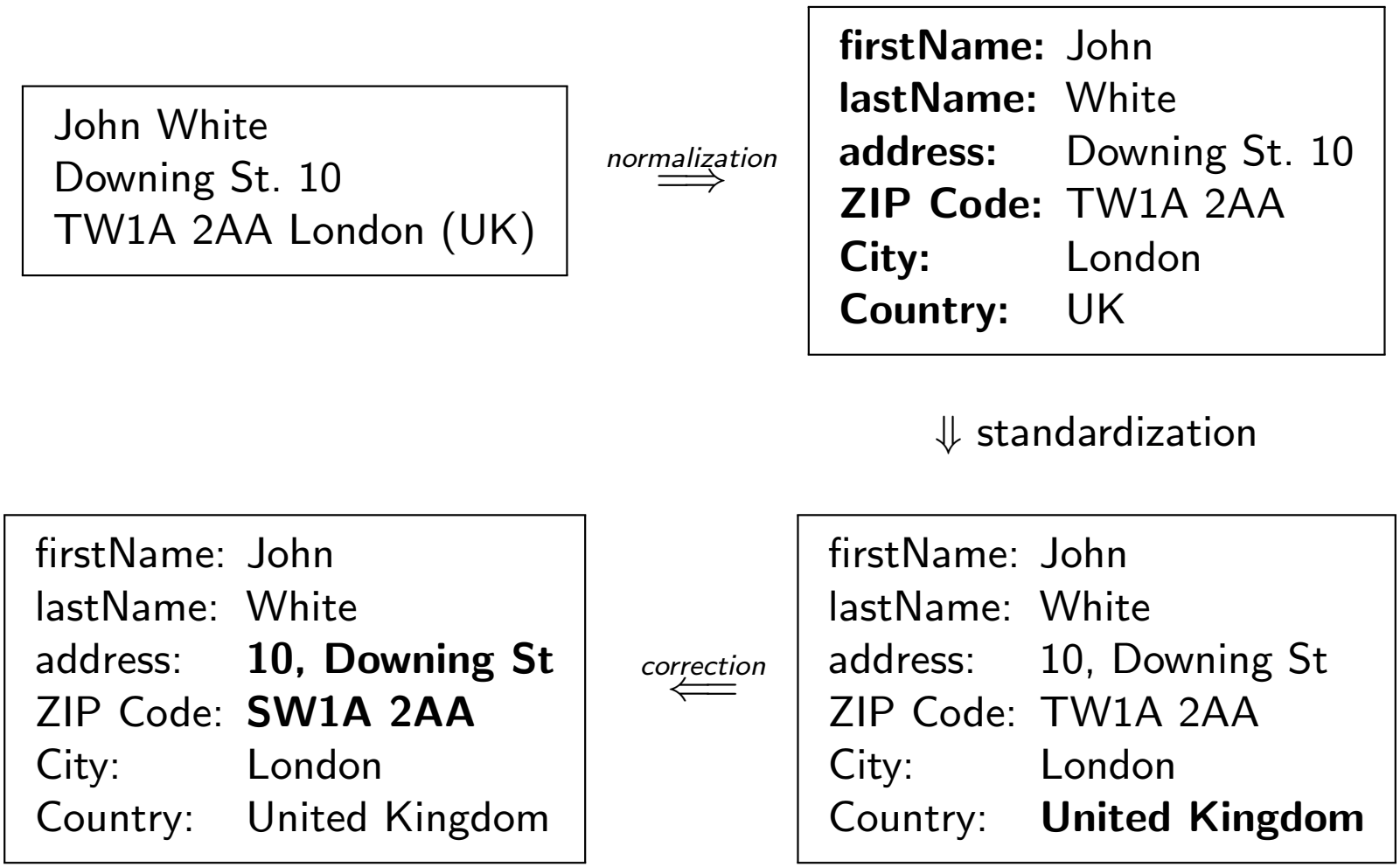
Cleansing/1

- BI does not work on "raw" data.
  - Pre-processing necessary for good results.
  - Can "disturb" BI analyses if not handled (e.g., duplicates).
- Handle inconsistent data formats.
  - Spellings, codings, ...
- Remove unneeded attributes
  - Production keys, comments, ...
- Replace codes with text
  - City name instead of ZIP code, 0/1 by Yes/No, ...
- Combine data from multiple sources with common key
  - Customer data, ...
- Find attributes with several uses
  - Extra values used for "programmer hacks".

Cleansing/2

- Mark facts with data status dimension
  - Normal, abnormal, outside bounds, impossible,
  - Facts can be taken in/out of analyses
- Recognize random or noise values
  - Can disturb BI tools
  - Replace with NULLs (or estimates)
- Uniform treatment of NULL
  - Use explicit NULL value rather than "normal" value (0, -1, . . .)
  - Use NULLs only for measure values (estimates instead?)
  - Use special dimension keys for NULL dimension values
- Aggregate fact data?
  - Performance
  - Statistical significance only for higher levels

Data Cleansing Example



Data Mining Transformations

- Can often **not** be done generally for the whole DW.
- Divide data into training, test, and evaluation sets.
  - Training: used to train model.
  - Test: used to check the model's generality (overfitting).
  - Evaluation: model uses evaluation set to find "real" clusters, ...
- Add computed attributes as inputs or "targets"
  - Derived attributes often more interesting for mining (profit,...)
  - Means more possibilities for data mining
- Mappings
  - Continuous values to intervals, textual values to numerical
- Normalization of values between 0 and 1
  - Demanded by some tools (neural nets)
- Emphasizing the rare case
  - Duplicate rare elements in training set

Improving Data Quality

- Appoint "data stewards" – responsible for data quality
  - A given steward has the responsibility for certain tables
  - Includes manual inspections and corrections!
- DW-controlled improvement
  - Default values
  - "Not yet assigned 157" note to data steward
- Source-controlled improvements
  - The optimal solution, but not feasible in most cases.
- Construct programs to automatically check data quality
  - Are totals as expected?
  - Do results agree with alternative source?
- Do not fix **all** problems with data quality
  - Allow management to see "weird" data in their reports.

Outline

Load/1

- 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.
  - Some load tools can also perform UPDATEs.
- Index on tables slows load a lot.
  - Drop index and rebuild after load.
  - Can be done per partition.
- Parallelization
  - Dimensions can be loaded concurrently.
  - Fact tables can be loaded concurrently.
  - Partitions can be loaded concurrently.

Load/2

- Relationships in the data
  - Referential integrity must be ensured.
  - Can be done by loader.
- Aggregates
  - Must be built and loaded at the same time as the detail data.
  - Today, RDBMSs can often do this.
- Load tuning
  - Load without log
  - Sort load file first
  - Make only simple transformations in loader
  - Use loader facilities for building aggregates
  - Use loader within the same database
- Should DW be on-line 24*7?
  - Use partitions or several sets of tables

Outline

ETL Tools

- ETL tools from the big vendors
  - Oracle Warehouse Builder/Oracle Data Integrator
  - IBM DB2 Warehouse Manager
  - Microsoft Data Transformation Services
- Offer much functionality at a reasonable price
  - Data modeling
  - ETL code generation
  - Scheduling DW jobs
  - ...
- Many others
  - Hundreds of tools
  - Often specialized tools for certain jobs (insurance cleansing, ...)
- The "best" tool does not exist.
  - Choose based on your own needs.
  - Check first if the "standard tools" from the big vendors are ok.

Other ETL Issues

- Files versus streams/pipes?
  - Streams/pipes: no disk overhead, fast throughput
  - Files: easier restart, often only possibility
- ETL tool or not?
  - Code: easy start, co-existence with IT infrastructure
  - Tool: better productivity on subsequent projects
- Load frequency?
  - ETL time depends on data volumes
  - Daily load is much faster than monthly
  - Applies to all steps in the ETL process

A Few Hints on ETL Design

- Do **not** try to implement all transformations in one step!
- Do **one** (or just a few) thing(s) at the time
  - Copy source data one-one to 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

- ETL is very time consuming (80% of entire DW project).
  - Not a single-step process, but needs to be implemented as a sequence of many small steps.
  - Data quality plays a crucial role in ETL.
  - Data has almost never good quality, which need to be fixed in ETL.
- Extraction of data from source systems might be very time consuming.
  - Incremental approach is suggested.
- Transformation into DW format includes many steps, such as
  - building key, cleansing the data, handle inconsistent/duplicate data, etc.
- Load includes the loading of the data in the DW, updating indexes and pre-aggregates, etc.
- ETL tools are available from big vendors, but also many other (open source) tools
  - Offer good functionality at a reasonable price
  - "Best tool" does not exist