Data Warehousing
Extract, Transform, Load (ETL)

Spring Term 2020
Dr. Andreas Geppert
geppert@acm.org

Outline of the Course

- Introduction
- DWH Architecture
- DWH-Design and multi-dimensional data models
- **Extract, Transform, Load (ETL)**
- Metadata
- Data Quality
- Analytic Applications and Business Intelligence
- Implementation and Performance

Content

1. Motivation and Overview
2. Monitoring
3. Extraction
4. Transformation
5. Loading
6. ETL for Data Marts
7. Data Lineage and Impact Analysis
8. ETL-Tools and Infrastructure

Motivation

- DWH is an "integrated collection of data for analytic needs"
- schemas and data need to be integrated
- pure logical integration (which leaves the data in the sources) does not (always) suffice
- data must be copied into the DWH
- but integration requires homogenization of heterogeneous data
- in addition further DWH-requirements must be met (historization, dimensionality)

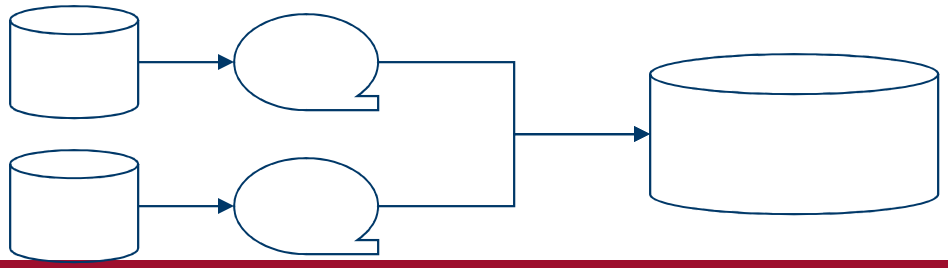
➔Extraction, Transformation, Loading (ETL)

Data Integration

- Precondition for integration of data from sources:
- relevant sources must be identified
- required data must be known
  - which database/tables, files, etc.
- Existence of data sources and their content as well as their semantics must be known
  - might be a problem especially for old, historically grown systems
- existence of up-to-data and complete data models and an overall data architecture is crucial

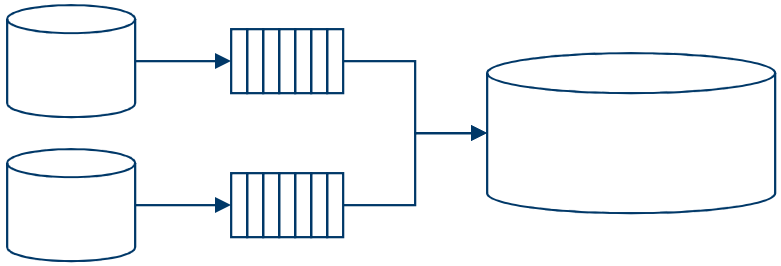
Data Integration Requirements and Variants

- the required timeliness of data in the data warehouse is a major criteria
- in the classical data warehouse, applications need data as of the day before or the end of the last month/quarter/year
- there are at least several hours ("end-of-day processing") or even several days to process the data and to load them into the DWH and the data marts
- data processing is bulk-oriented and done as batch processing



Data Integration Requirements and Variants (2)

- in the "near realtime" data warehouse, applications need (some of their) data very soon after it has been created or modified
- there is not enough time to collect changed data, instead data must be propagated to the DWH immediately
- data processing is event-based and cannot be done as batch processing



Data Integration: Extraction

- a data warehouse needs an initial load of the entire data set from a specific source
- later, full or incremental loads can be applied
- a **full load** loads deletes the whole data set (e.g., all customer master data) and replaces it with the whole extract from the source(s)
  - the DWH then has a data set consistent with the source
  - the full extract will not contain the complete history, so the existing history in the DWH must be reconciled with the full extract from the source(s)
  - for large (and many) data sources, a frequent full load may take too much time
  - a full load may be reasonable for master data, but not for transaction data ("Bewegungsdaten")
- an **incremental load** loads only the changed data since the last load
  - the source must be able to detect its changes
  - applying deltas is much less time consuming than full loads
  - in practice, DWH and sources can diverge. A regular full load (e.g., every quarter) can be used to reconcile sources and DWH

Data Integration : Transformation

- Requirements to DWH data:
  - uniform representation
    - also for data from different sources
    - according to DWH data model
  - resolution of heterogeneity
  - completeness
  - data quality in general, in particular accuracy
  - granularity (in particular in data marts)

➔1:1 copy is usually not sufficient

➔new data need to be **transformed**

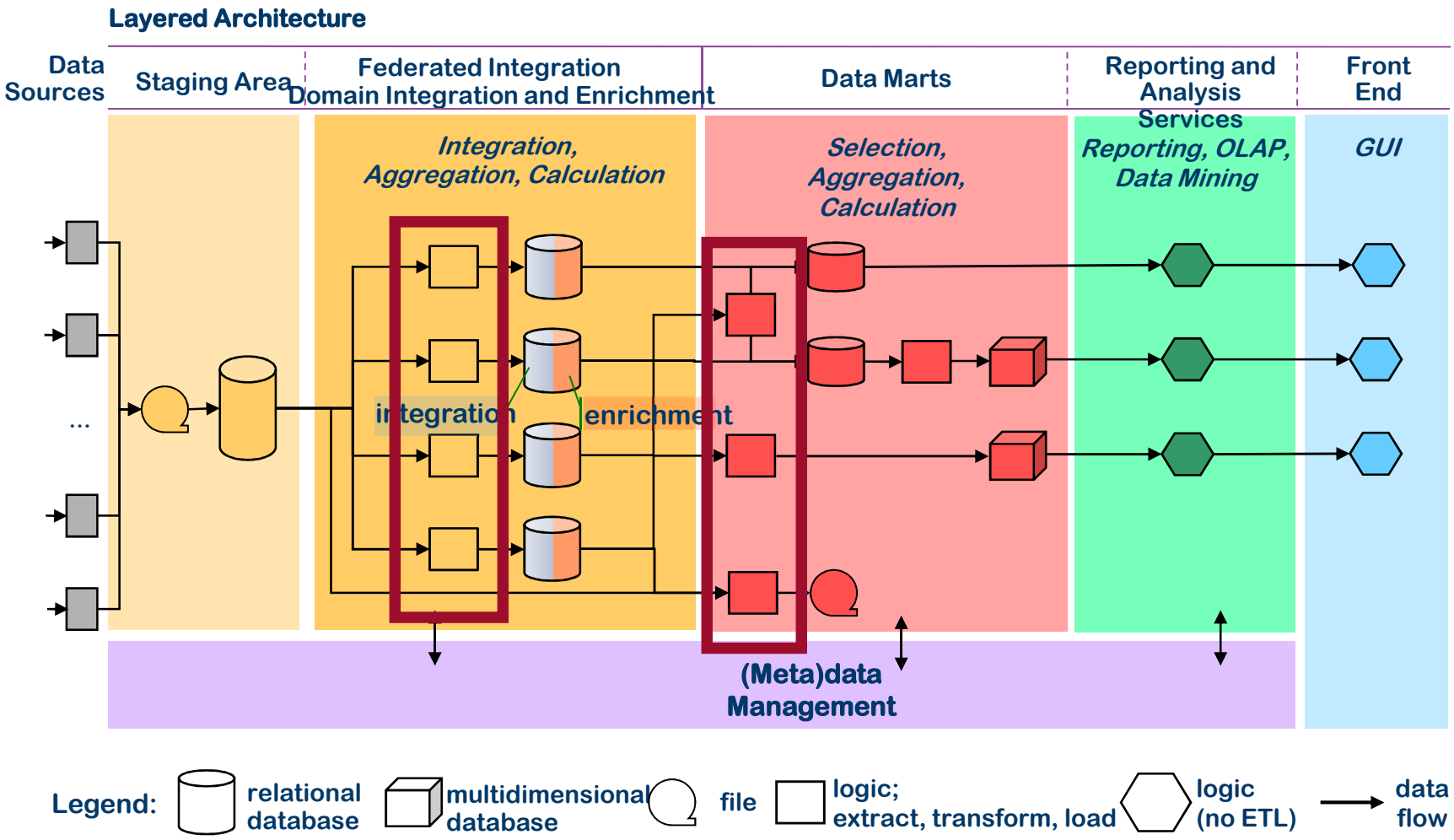
Data Integration: Loading

- loading process proper
  - add new data to DWH or data mart database
  - efficiency is a challenge for large volumes of new data
  - update of metadata and secondary/auxiliary data
    - indexes
    - statistics

➔loading is a separate step in the ETL process

➔special database technology support required

ETL: Position in the DWH Architecture



ETL: DWH and Data Marts

- ETL between sources and DWH (SMAs)
  - purpose is mainly integration and data quality assurrance
- ETL between SMAs and RMDAs (enrichment layer)
  - purpose is mainly calculation and creation of dimensions/measures
- ETL between DWH and data marts
  - purpose is mainly aggregation, selection, calculation
- many commonalities, but also differences

➔we first consider sources → DWH, then DWH → data marts

Content

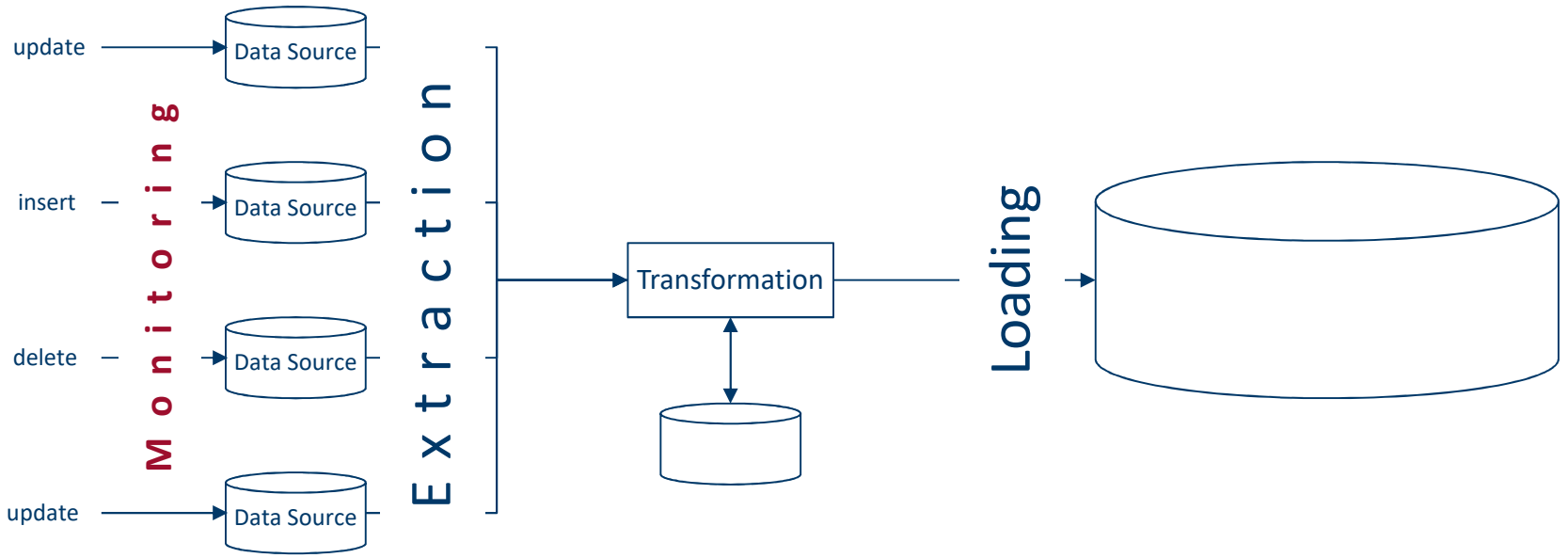
1. Motivation and Overview
2. **Monitoring**
3. Extraction
4. Transformation
5. Loading
6. ETL for Data Marts
7. Data Lineage and Impact Analysis
8. ETL-Tools and Infrastructure

Incremental Refresh

- Add new data from sources into DWH
- mostly new (inserted) data, to a lesser extent deletes and updates
- Problem: detect new/changed data
  - collect changes and send to DWH in regular intervals (daily, etc) or
  - propagate data to DWH immediately

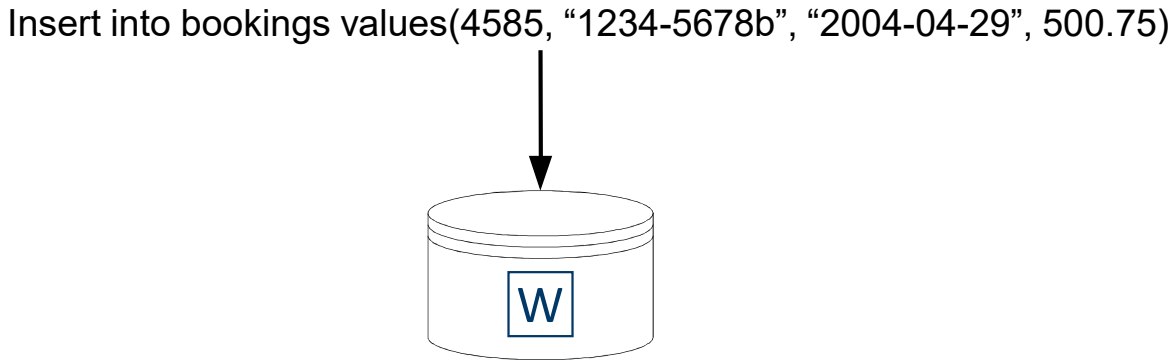
➔Monitoring

Extraction, Transformation, Loading



Monitoring: Classification

- using DBMS-functions
- based on certain attributes and their values
- based on application functionality
- using dedicated monitoring/extraction applications



Monitoring Approaches: Overview

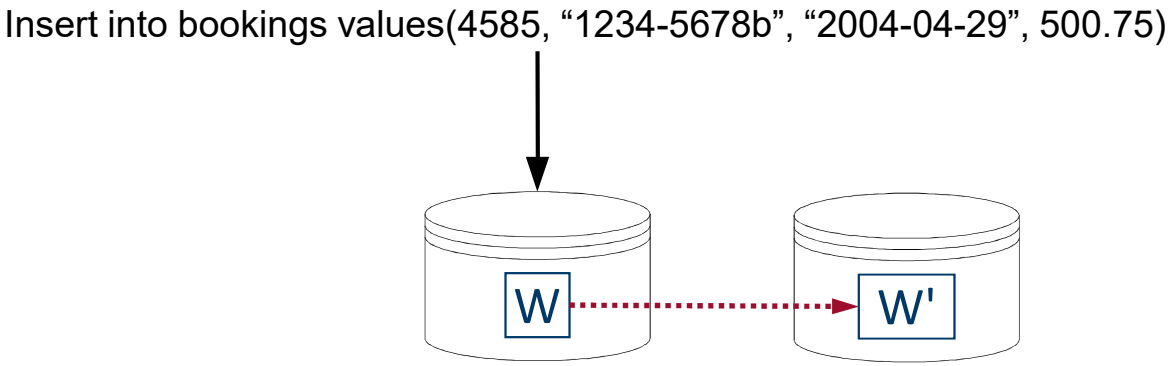
- replication based
- log analysis
- snapshot comparison
- active mechanisms
- timestamp-based
- applicatory

Criteria:

- runtime overhead (for OLTP)
- performance and efficiency
- declarativeness
- vendor-independence
- generality of use
- requirements (required information)
- DBMS-support

Replication-based Monitoring

- DBMS replicates changes into separate, dedicated tables
  - in the same database
  - in another database, possibly managed by a different DBMS
- ETL starts with extraction of the replicas
  - replica is deleted after extraction



Replication-based Monitoring (2)

- Example: Postgres Logical Replication, Oracle GoldenGate

Properties:

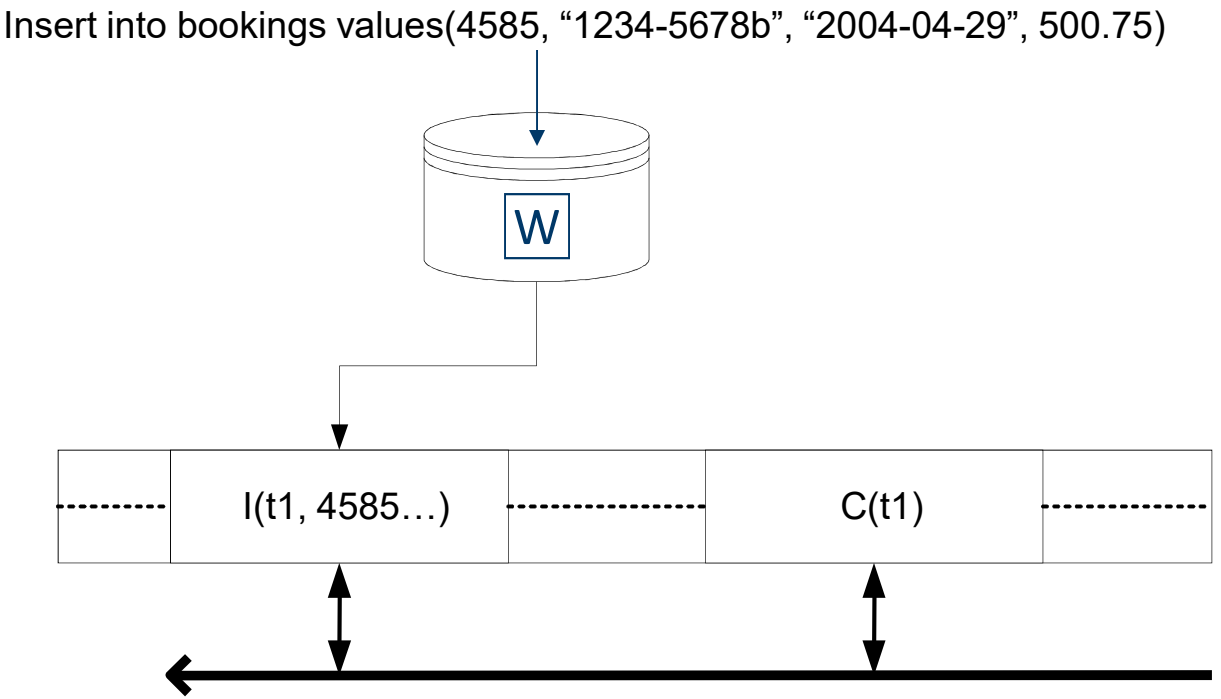
☺ database support

☺ declarative definition which tables to monitor

☺ moderate to medium overhead for OLTP

Log Analysis

- DBMS records changes to data in transaction log (standard functionality for transactional databases)
- Log is analyzed in order to determine changed data to be extracted



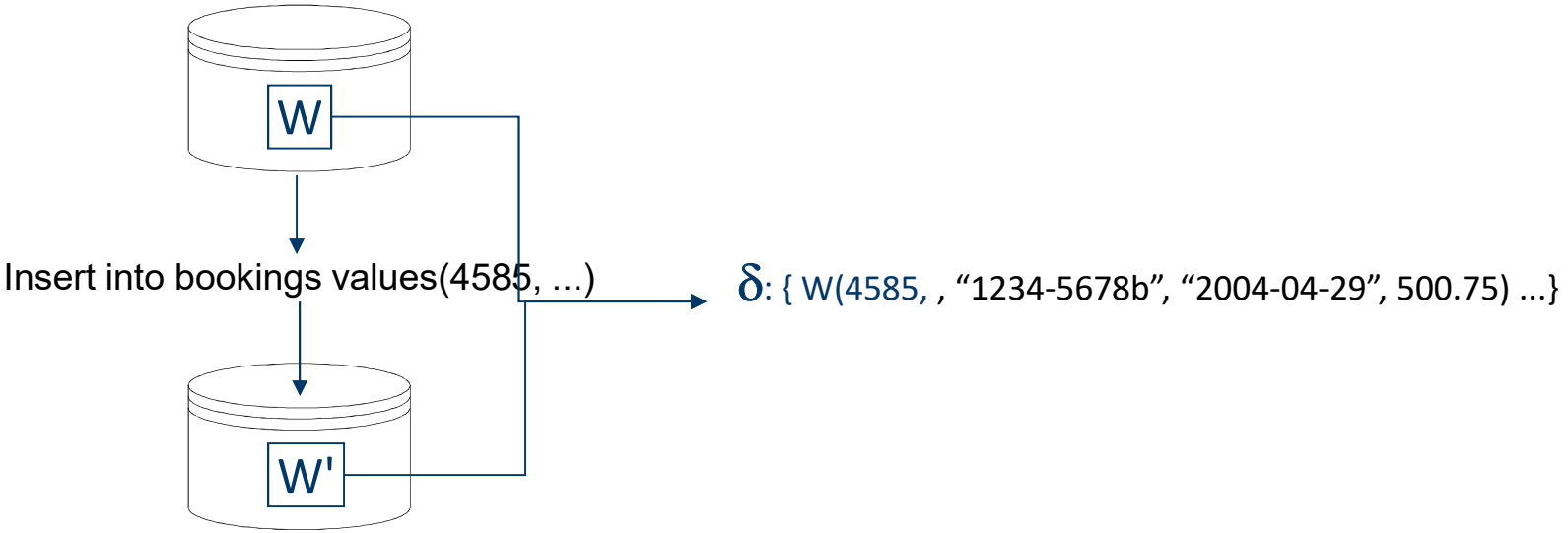
Log Analysis

Properties:

- 😐 DBMS-internal useful usage (e.g., as an implementation mechanism for replication)
- 😐 useful when tool support exists or vendor publishes API (e.g., Oracle's LogMiner)
- ☹ otherwise, requires access to transaction log
- ☹ requires knowledge of the log structure
- ☹ high degree of vendor dependency

Snapshot Comparison

- compute snapshot of interesting part of the database in regular intervals
- compute difference between two subsequent snapshots at points in time (t-1, t) to obtain delta



Snapshot Comparison (2)

Properties:

☹ high storage requirements (for storing snapshots)

☹ tremendous time consumption for snapshot comparison

☹ snapshot generation can impact OLTP operations

➔ only feasible for small databases (or small interesting data sets)

Active Mechanisms

- a trigger has an event, a condition, and an action part
- the event defines when the trigger "fires", in this case the event is one of the "interesting" modifications (insert, update, delete)
- the trigger action records the modification into a separate table (in the same or a different database)
  - "poor man's replication", used by replication products for foreign sources (e.g., by DB2Propagator for Oracle sources)
- ☺ always doable (with modern relational DBMSs)
- ☺ flexible
- ☹ for larger sets of triggers, the overhead incurred on OLTP can be significant (note that each monitored table requires three triggers)

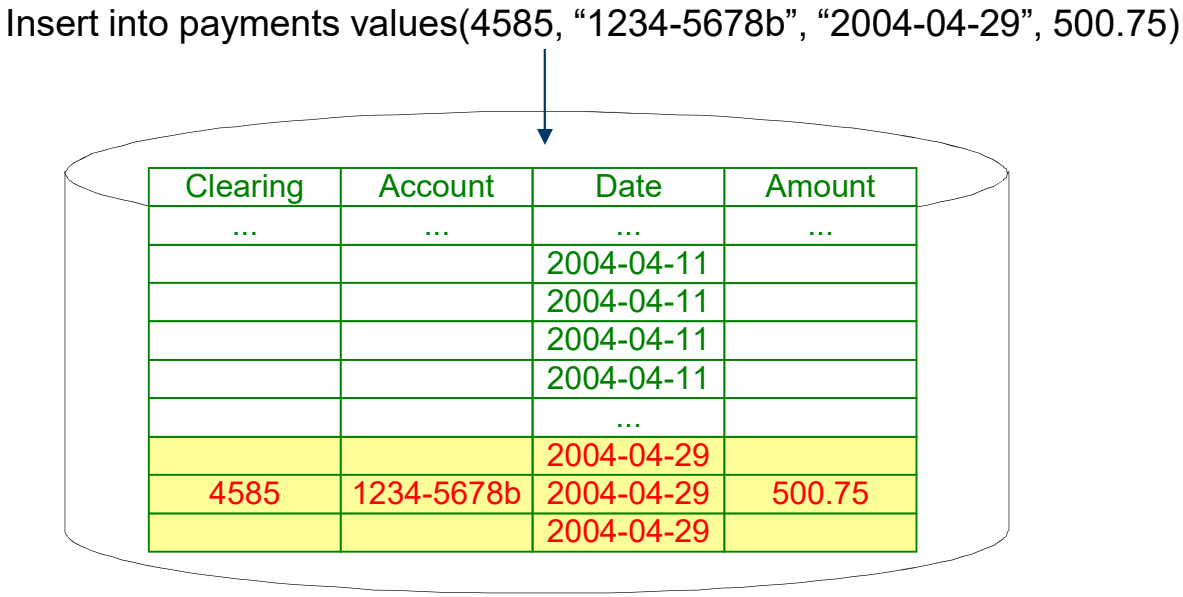
Active Mechanisms (2)

- Example: Trigger for insertions
- action replicate new data

```
create trigger new_payment
   after insert on payments
         referencing new as n
   for each row
         insert into new_ payments
         values (n.clearing...)
```

Timestamp-based Monitoring

- monitored data (table) contains a timestamp that can be used to detect new data
- usage for monitoring: extract all tuples with a timestamp larger (i.e., younger) than that of the last extraction



Timestamp-based Monitoring (2)

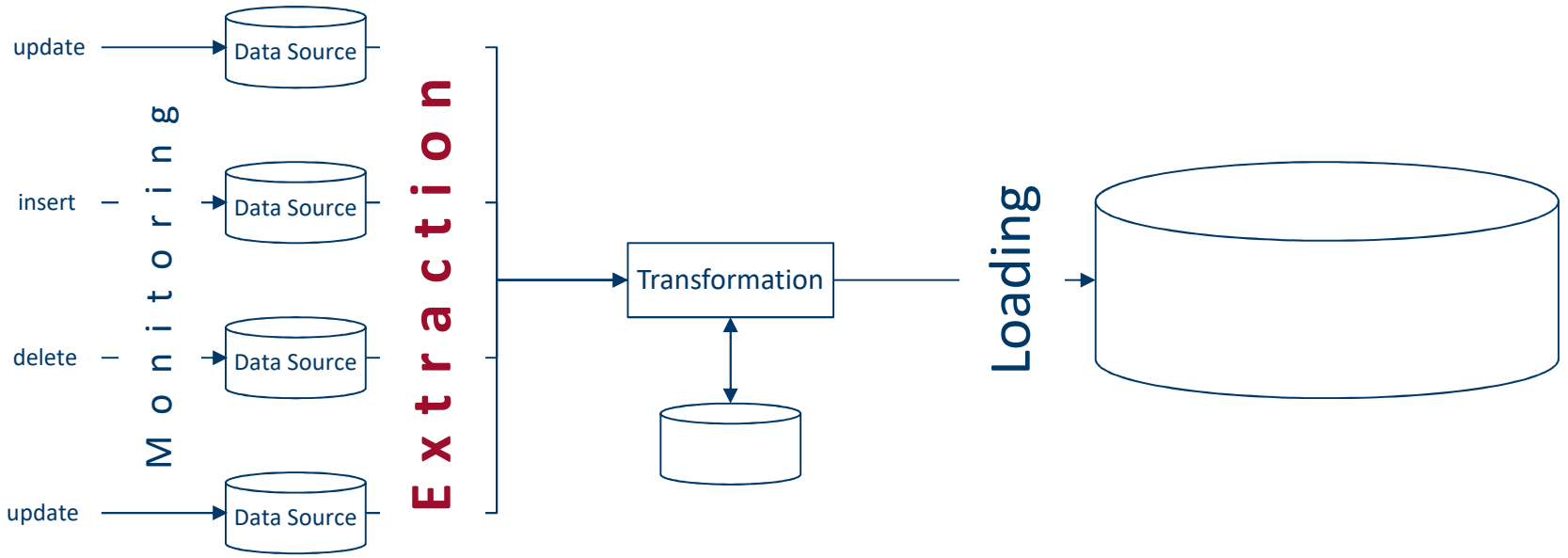
Properties:

- ☺ no additional effort and mechanism necessary
- ☺ simple extraction using SQL
- ☹ partial solution only, because not all source tables carry required timestamps

Application-based Monitoring

- applications record their changes (typically in files)
- 😐 solution if all else fails
- 😐 overhead for application, not for source (database)
- ☹ requires extension of applications

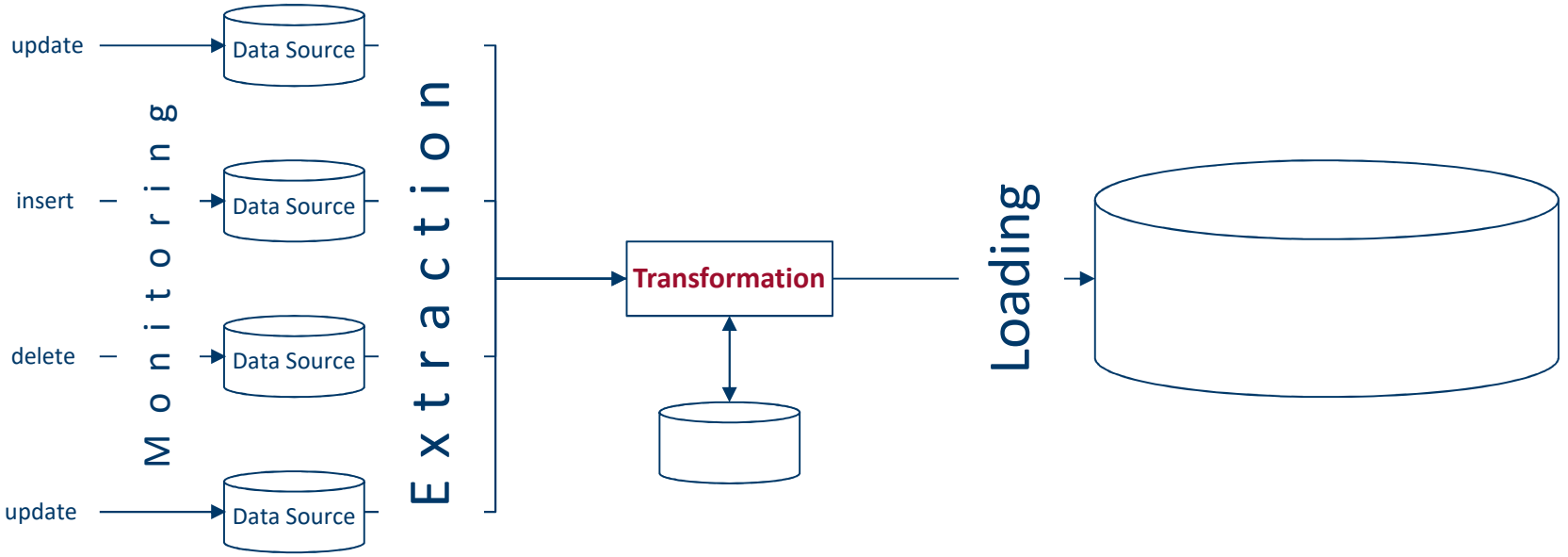
Extraction, Transformation, Loading



Extraction

- extraction of new/modified/deleted data from sources
- insertion into staging area (or directly into data warehouse)
- variants (depending on monitoring strategy):
- direct access from source
- indirectly via data transfer (file or message)
  - export of data from source
  - transport
  - import of data into staging area

Extraction, Transformation, Loading



Transformation

- "Transformation" converts data as obtained from sources into structures conforming to the common, unified data model
- this is the step where integration is performed and heterogeneities are resolved

Possible transformation steps per record:

- Decomposing of data items into parts (*elementizing*)
- Standardization of elements
- Plausibility checks (*verification*)
- Reconciliation with existing data records (*matching*)
- Historization
- Key Generation and Mapping
- Grouping

Transformation (2)

- Decomposing of data items into parts (*elementizing*)
- typically done when composite data are represented as Strings
- Address Data, Names, Dates
- Example:
  - [name: „Joe Cool“¨] → [firstname: „Joe“, lastname: „Cool“]
  - [date: „12.12.2002“] → [day: 12, month: 12, year: 2002]
  - [address: „4300 The Woods Dr, San Jose CA-95127“]
    → [street: „The Woods Dr“, number: „4300“,
    city: „San Jose“, zip: 95127, state: „CA“]

Transformation (3)

- Standardization of elements
- data elements of the same type or from the same domain will be represented the same
- Strings, Dates, Addresses, Names, Codes
- Replacement of abbreviations
- Examples:
  - 0/1, m/f, m/w, M/F, ... → m/f
  - „The Woods Dr “ → „The Woods Drive“

Transformation (4)

- Plausibility checks
- Checks plausibility and correctness of data
- incorrect data may or may not be corrected
- Example (street name does not exist and is replaced with the "closest" one):
  - „The Wolves Dr ive “ -> „The Woods Drive“

➔very depending on domain knowledge

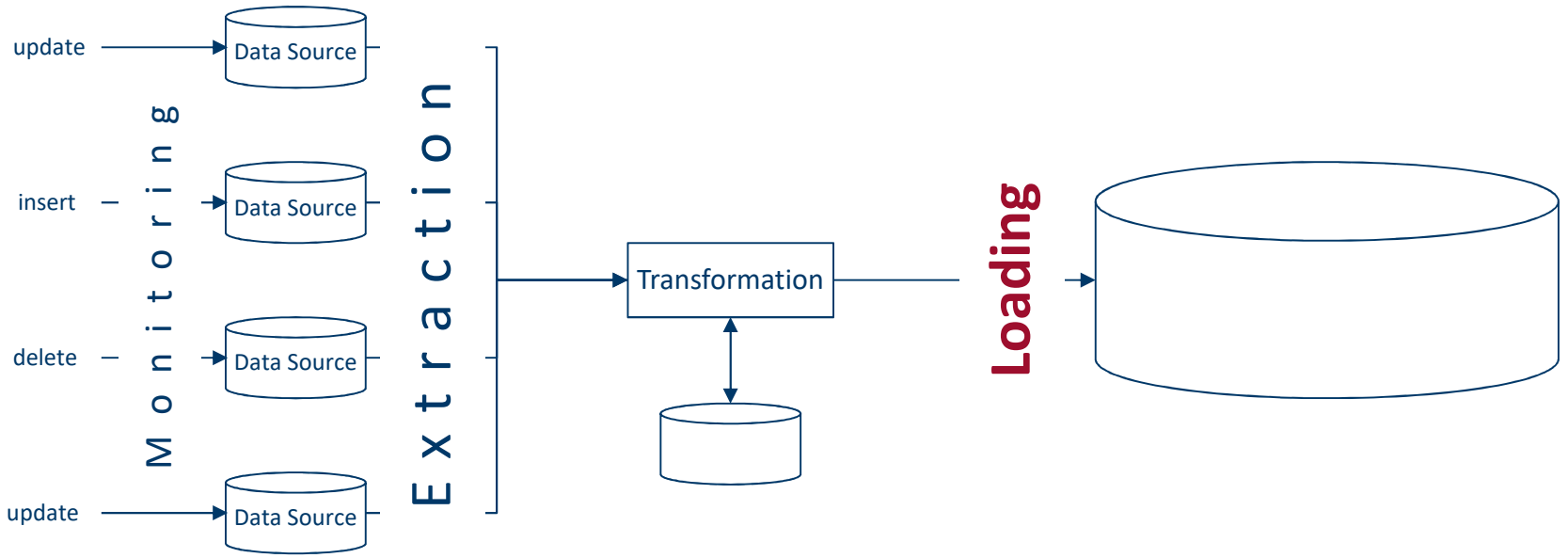
➔done as part of data quality assurrance and data cleansing (s. separate lecture)

Transformation (5)

- Reconciliation with existing data records (*matching*)
- possibly includes update to existing records
- determine whether data record is new or refers to an already existing record
- (surrogate) primary key determination
  - identify existing primary key or....
  - generate new primary key
- depending on historization scheme in place, perform insert, update, or combination (in case of validity time or bi temp)
- duplicate elimination
  - determine whether differing records represent the same real-world entity
  - Example: „Joe Cool“, „J. Cool“, „Snoopy“→ „Joe“ „Cool“

➔also subject of data quality assurrance and data cleansing (s. separate lecture)

Extraction, Transformation, Loading



Staging Area

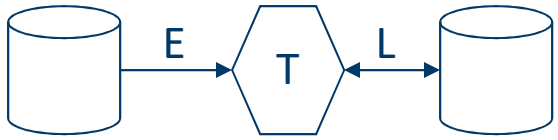
- area dedicated to data processing/transformation
- using database for the staging area is reasonable
- data from flat files are turned into database structures when loading into the staging area. "Semantic" transformations are done when data are moved into the DWH (SMAs)

Loading into Staging Area and DWH

- Technical aspects
- load in bulk mode instead of single inserts
- no logging
  - avoids log overflow and logging overhead
- No checking of integrity constraints
  - primary keys, referential integrity
- Firing of triggers is suppressed
- locking entire tables
- direct path load

ETL vs ELT

- "classical" approaches consists of the sequence extraction – transformation – loading
- Transformations are performed by a ETL-tools on top of the database, often on a separate server than the database



- The E-L-T approach switches the loading and the transformation phase
- data are loaded into the database (without major transformations)
- data are transformed within the database, using SQL and possibly stored procedures as generated by the ELT-tool

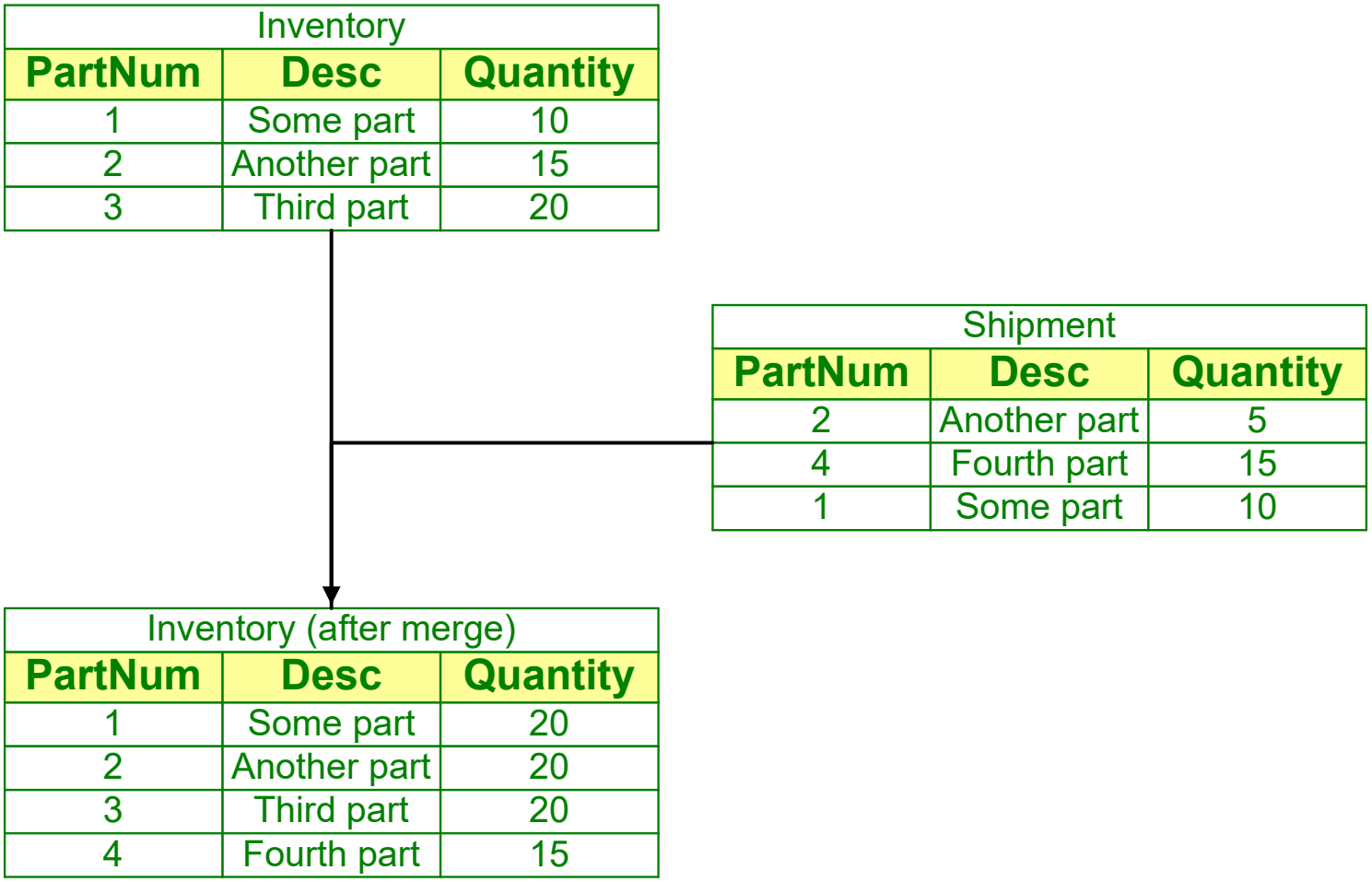
Loading: Merge

- when loading entire data sets (into a table), some of the element records may or may not refer to existing records
- two different operations need to be carried out, depending on whether a specific record is new or not
- addressed by the merge operation:
  - new in SQL:2003
  - defines a conditional branch
  - tests whether records are new
  - inserts new data
  - updates existing data

Loading: Merge (2)

```
merge into inventory as inv
using (select partnum, desc, quantity from shipment) as sh
on (inv.partnum = sh.partnum)      -- test
when matched
   then update set quantity = ...   -- old → update
when not matched
   then insert ...                  -- new → insert
```

Merge: Example



Inventory

PartNum	Desc	Quantity
1	Some part	10
2	Another part	15
3	Third part	20

Shipment

PartNum	Desc	Quantity
2	Another part	5
4	Fourth part	15
1	Some part	10

Inventory (after merge)

PartNum	Desc	Quantity
1	Some part	20
2	Another part	20
3	Third part	20
4	Fourth part	15

Loading: Alternative Syntax for Merge

```
insert into inventory (partnum, desc, quantity)
      (select partnum, desc, quantity from shipment)
on conflict (partnum)
do update set quantity = inventory.quantity +
   excluded.quantity;
```

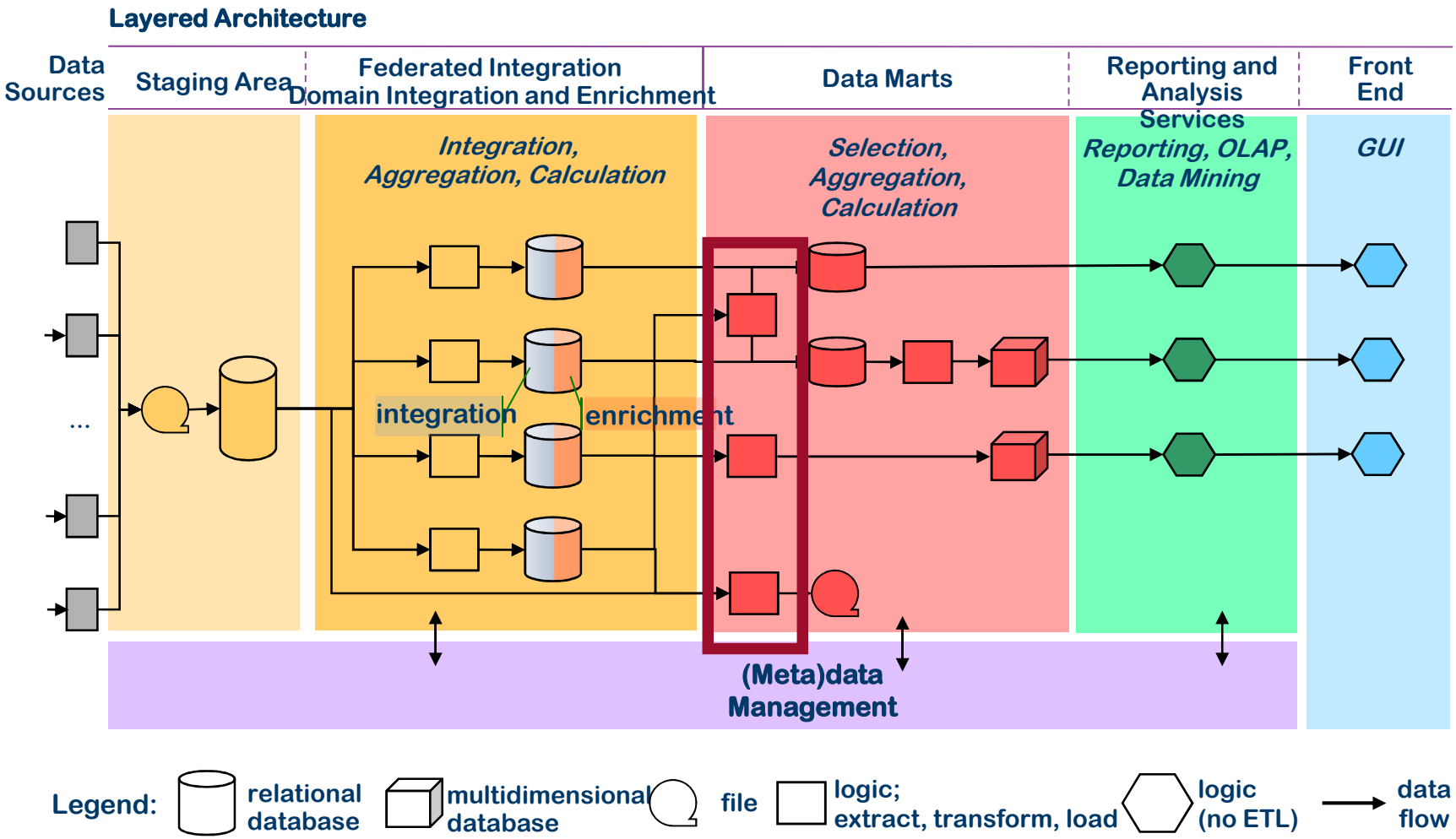
Content

1. Motivation and Overview
2. Monitoring
3. Extraction
4. Transformation
5. Loading
6. **ETL for Data Marts**
7. Data Lineage and Impact Analysis
8. ETL-Tools and Infrastructure

ETL for Data Marts

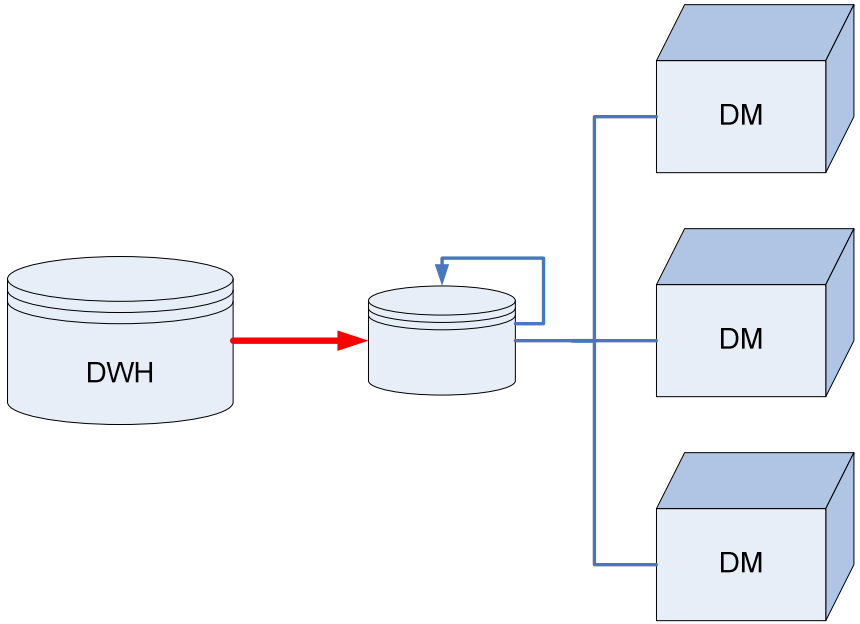
- propagation of extracted and transformed data from the DWH to data marts
- similar questions as in ETL for DWH
- but data mart specifics need to be taken into account
- ETL for data marts may not rely on file transfer, because data exchange occurs within the same system and platform. Data transport can be implemented by the ETL-tool instead

ETL for Data Marts



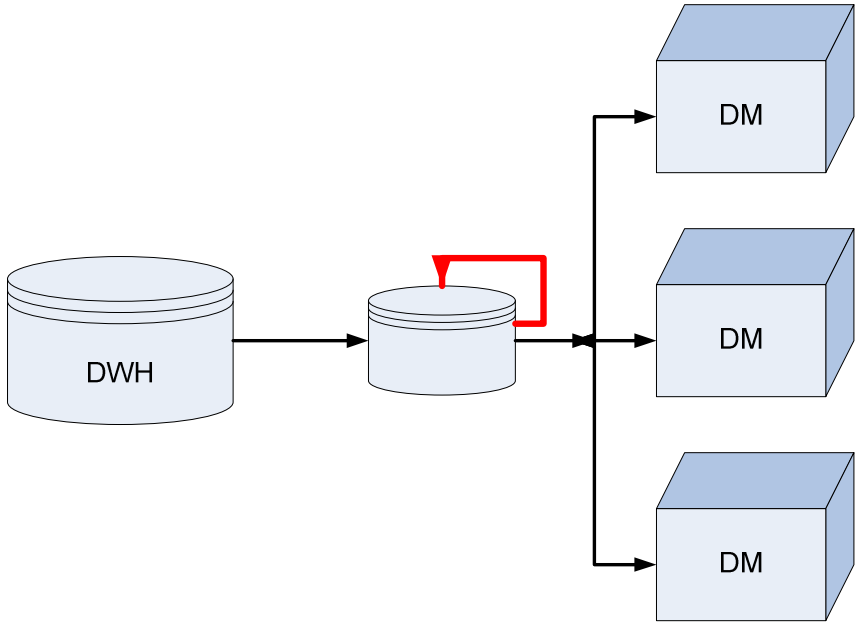
ETL for Data Marts: Extraction

- ETL-process for data marts can start, as soon as loading phase for the DWH (SMAs) has finished
- Detection of changes not a problem
- DWH system "knows" when refresh of the DWH is finished
  - end of the loading phase
  - no monitoring necessary



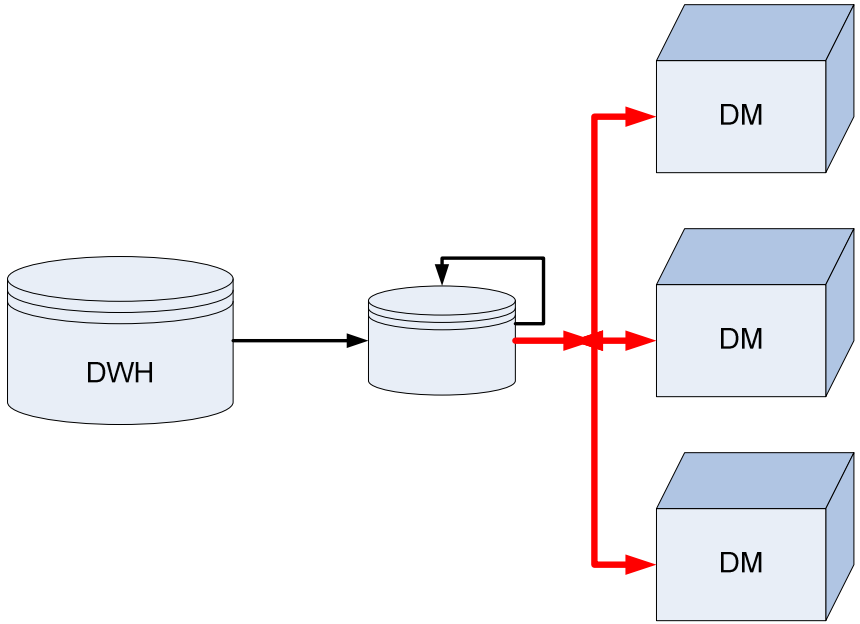
ETL for Data Marts: Transformation

- Homogenization, duplicate elimination etc. not necessary
- transformation into analytic structures
- historization (of dimensions), in particular slowly-changing dimensions
- computation: calculation of non-trivial derived measures
- aggregation: data mart needs a coarser granularity than available in the DWH

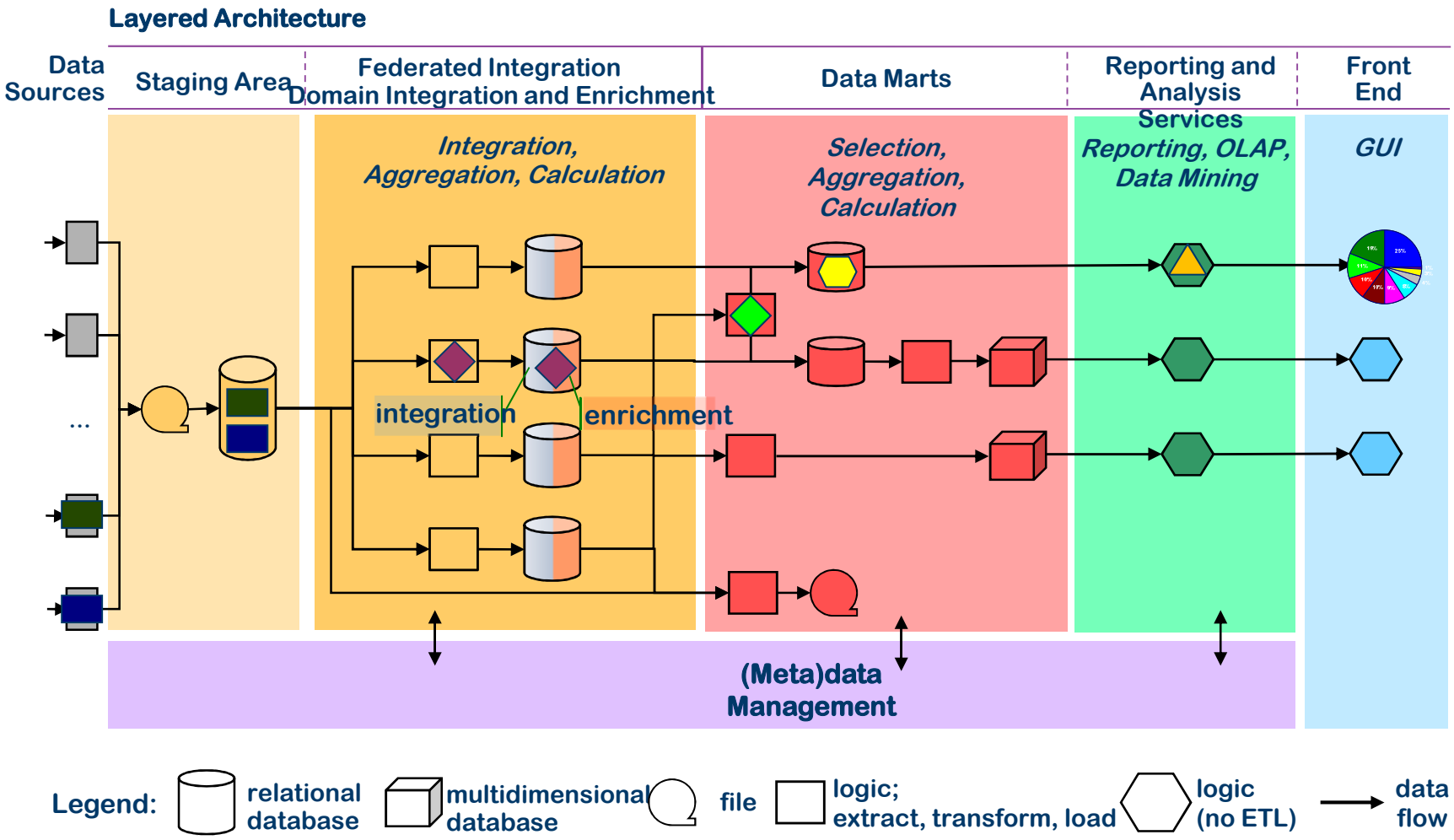


ETL for Data Marts: Loading

- refresh of the data in the data marts
- Cubes
  - refreshing cubes
  - including compression
- Star/Snowflake
  - load dimension and fact tables
  - possibly refresh materialized views (see below)
- update secondary and meta data
  - statistics and indexes



ETL: Summary


Layered Architecture
Data Sources
Staging Area
Federated Integration
Domain Integration and Enrichment
Data Marts
Reporting and Analysis Services
Front End
Integration, Aggregation, Calculation
Selection, Aggregation, Calculation
Reporting, OLAP, Data Mining
GUI
integration
enrichment
(Meta)data Management
Legend:
relational database
multidimensional database
file
logic; extract, transform, load
logic (no ETL)
data flow

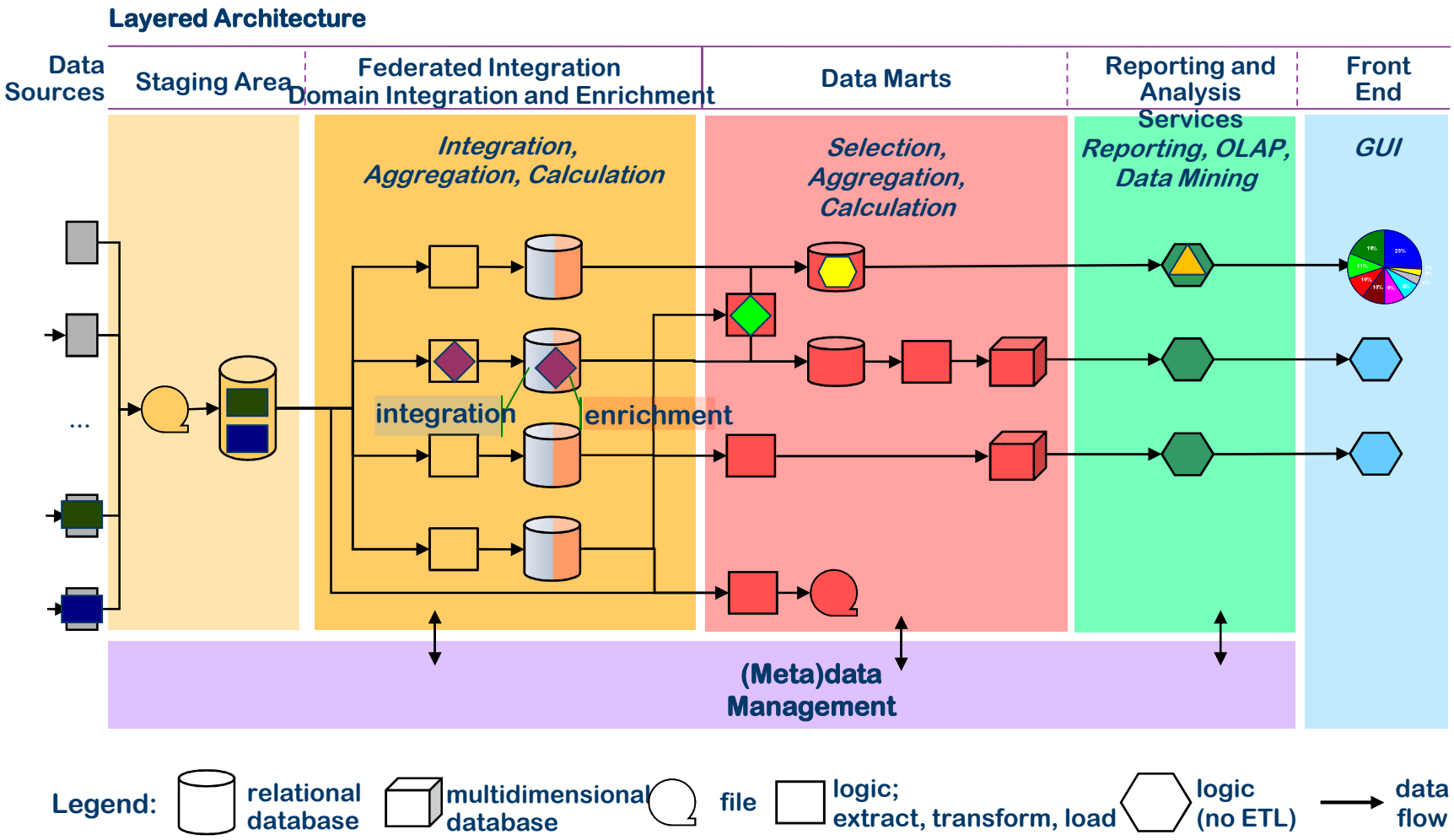
Content

1. Motivation and Overview
2. Monitoring
3. Extraction
4. Transformation
5. Loading
6. ETL for Data Marts
7. **Data Lineage and Impact Analysis**
8. ETL-Tools and Infrastructure

ETL: Data Lineage

- "reversal" of the ETL process for metadata
  - which transformations on data from which sources led to the current structures in the data marts?
- "reversal" of the ETL process for instance data
  - trace back single data elements
    - in data marts, reports, analyses, ...
  - which source data elements have been transformed in which way to result in the data shown in the current analysis?

ETL: Data Lineage



Content

1. Motivation and Overview
2. Monitoring
3. Extraction
4. Transformation
5. Loading
6. ETL for Data Marts
7. Data Lineage and Impact Analysis
8. **ETL-Tools and Infrastructure**

ETL-Tools

- ETL-process handle many sources, multiple targets, many and complex transformations
- there are two general strategies (not mutually exclusive) how to implement ETL:
  - using ETL-tools for all three phases
  - implementing transformations inside the database, using ETL-tools only for transport

ETL: Requirements

- ETL-software and development phase
  - often the most complex and expensive part of a DWH-project

➔ Requirements for design component:

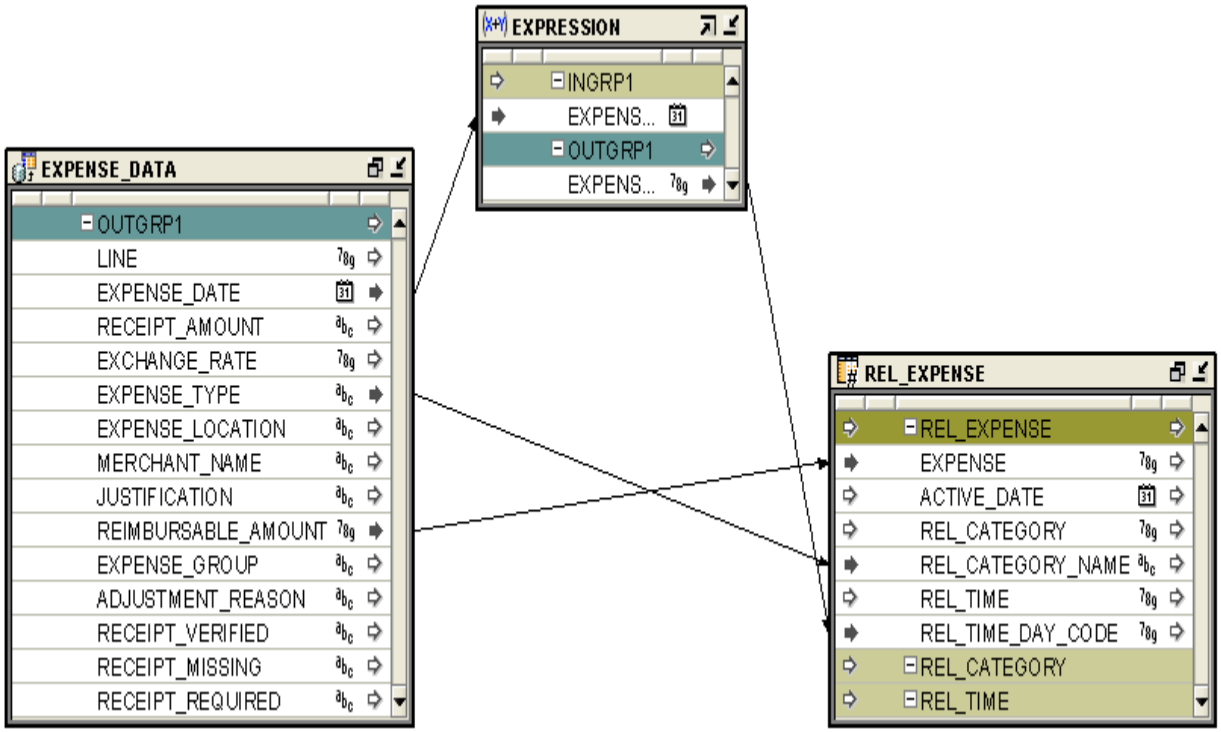
- reusability
- maintainability
- understandability: impact analysis, data lineage
- extensibility and flexibility
- declarative transformation rule design
- modeling and design of ETL-processes (cf. data modeling levels)

➔ Requirements for runtime component :

- execution of ETL-processes
- monitoring
- data quality assurance
- integration with metadata management
- performance
- scalability

ETL-Tools: Example

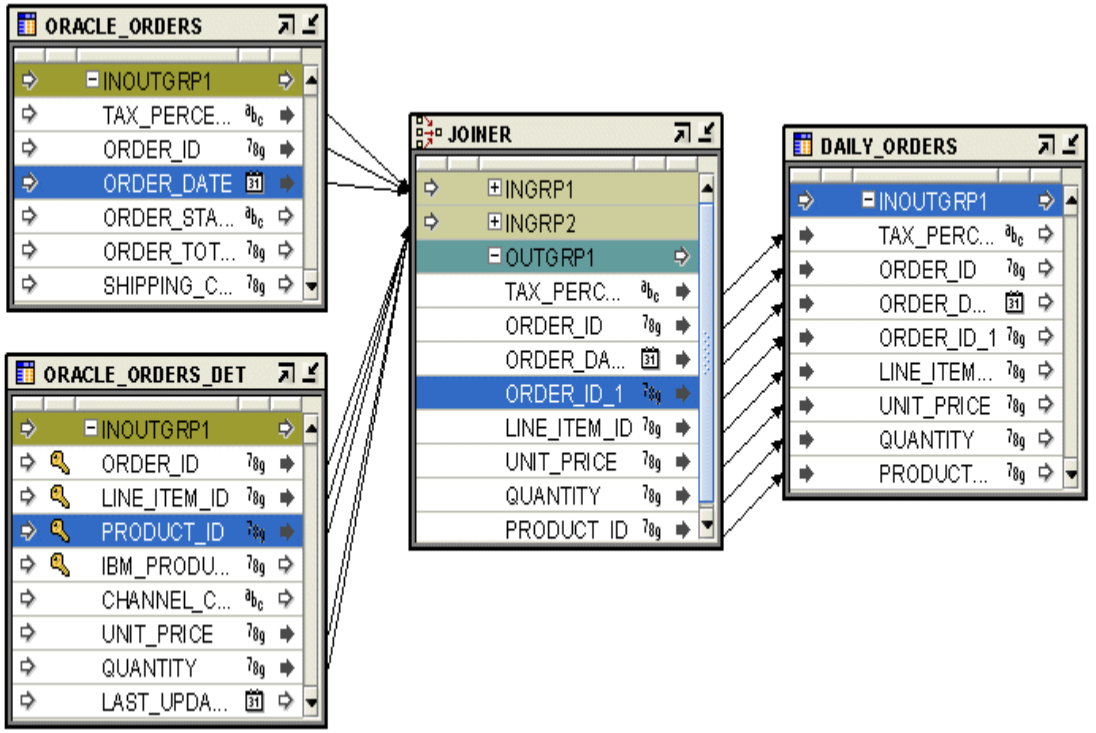
- Mappings connect sources with targets:
  - files (external tables)
  - tables, views
  - dimensions
  - cubes
  - constants
  - sequences



ETL-Tools: Example (2)

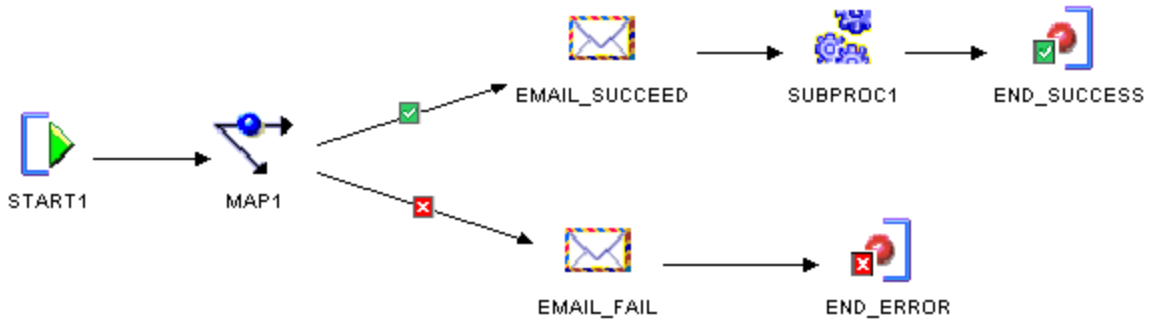
- data flow operators

  transform data:
  - aggregation
  - filter
  - sorting
  - joiner
  - splitter
  - key lookup
  - match/merge
  - de-duplication



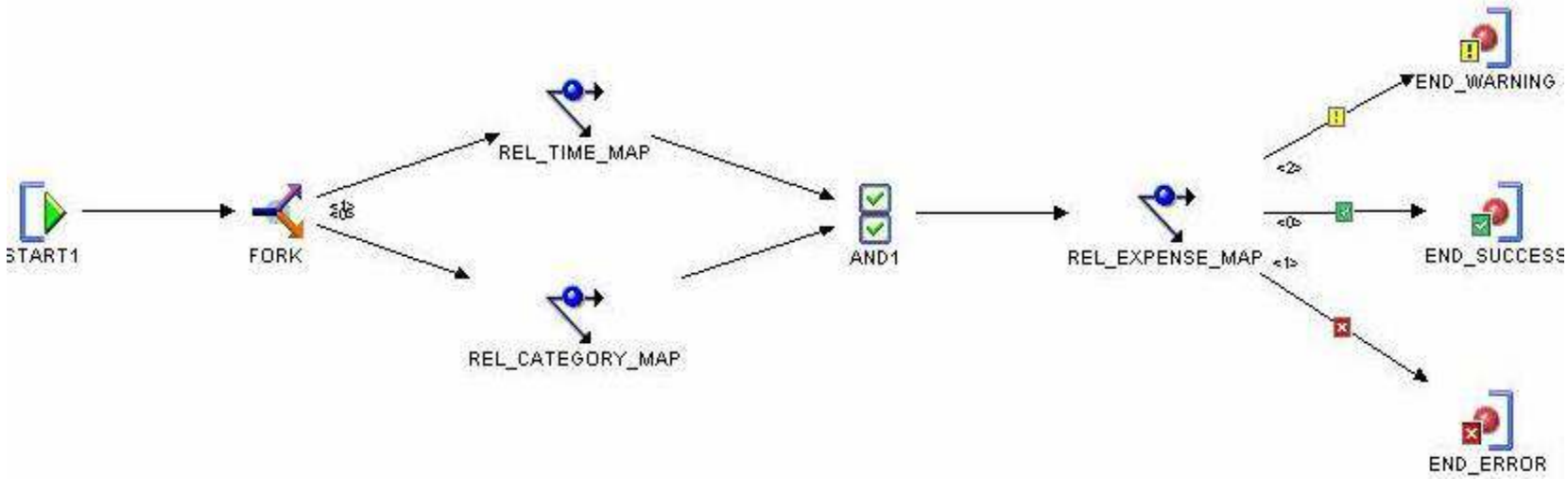
ETL-Tools: Example (3)

- further activities:
  - file operations (transfer/FTP, etc.)
  - notification, Email
  - data quality operators



ETL-Tools: Example (4)

- control flow operators connect activities and form processes:
  - conjunction (AND)
  - disjunction (OR)
  - fork
  - loops
  - end (successful, with warnings, with errors)



Summary

- ETL-phases:
  - extraction (based on monitoring)
  - transformation
  - loading
- normally batch-oriented
- leads to integrated, homogeneous DWH
- because of integration (and source heterogeneity) typically very complex and expensive
- tool and infrastructure support