Lecture 10

Building the DW

Summary – last week

- OLAP Operations:
  - Roll-up: hierarchical, dimensional
  - Drill-down: You can't drill if you don't have the data
  - Slice, dice, Pivot
- Operations affect data through query languages OLAP Query languages: SQL 99
  - SQL99:
    - Grouping Set, Roll-up, Cube operators

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

This week

Building the DW

1. The DW Project
2. Data Extract/Transform/Load (ETL)
3. Metadata

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

The DW Project

- Building a DW, is a complex IT project
  - A middle size DW-project contains 500-1000 activities
- DW-Project organization
  - Project roles and corresponding tasks, e.g.:

Roles	Tasks
DW-PM	Project Management
DW-Architect	Methods, Concepts, Modeling
DW-Miner	Concepts, Analyze(non-standard)
Domain Expert	Domain Knowledge
DW-System Developer	System- and metadata-management, ETL
DW User	Analyze (standard)

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

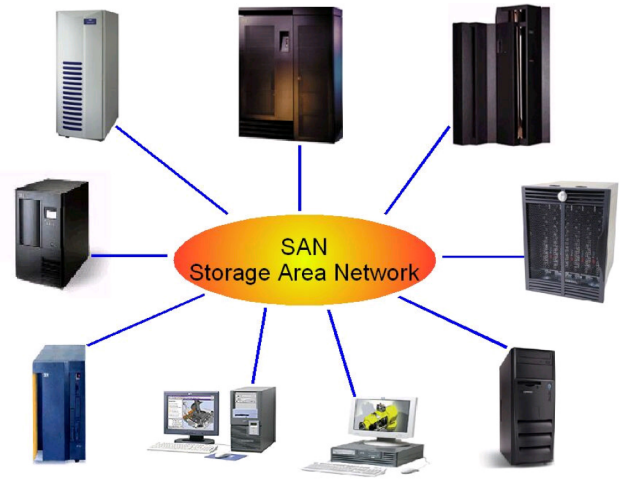
The DW Project (cont'd.)

- Software choice
  - Database system for the DW
    - Usually the choice is to use the same technology provider as for the operational data
    - MDB vs. RDB
  - ETL tools
    - Differentiate by the data cleansing needs
  - Analysis tools
    - Varying from data mining to OLAP products, with a focus on reporting functionality
  - Repository
    - Not very oft used
    - Helpful for metadata management

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

The DW Project (cont'd.)

- Hardware choice
  - Data storage
    - RAID systems, SAN's, NAS's
  - Processing
    - Multi-CPU systems, SMP, Clusters
  - Failure tolerance
    - Data replication, mirroring RAID, backup strategies
  - Other factors
    - Data access times, transfer rates, memory bandwidth, network throughput and latency



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

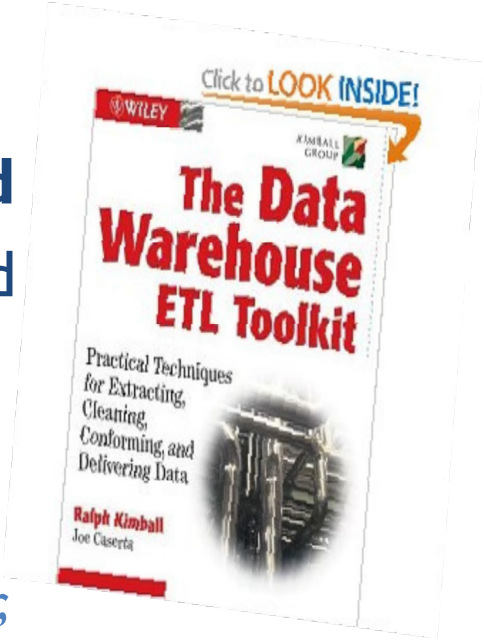
The DW Project (cont'd.)

- Project **Timelines**, depends on the development methodology, but usually
  - Phase I – **Proof of Concept**
    - Establish Technical Infrastructure
    - Prototype Data Extraction/Transformation/Load
    - Prototype Analysis & Reporting
  - Phase II – **Controlled Release**
    - Iterative Process of Building Subject Oriented Data Marts
  - Phase III – **General Availability**
    - On going operations, support and training, maintenance and growth
- The most important part of the DW building project is defining the **ETL** process

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

ETL

- What is ETL?
  - Short for **extract**, **transform**, and **load**
  - Three database functions that are combined into one tool to **pull data out** of productive databases and place it into the DW
    - **Migrate data** from one database to another, to form data marts and data warehouses
    - **Convert** databases from one format or type to another



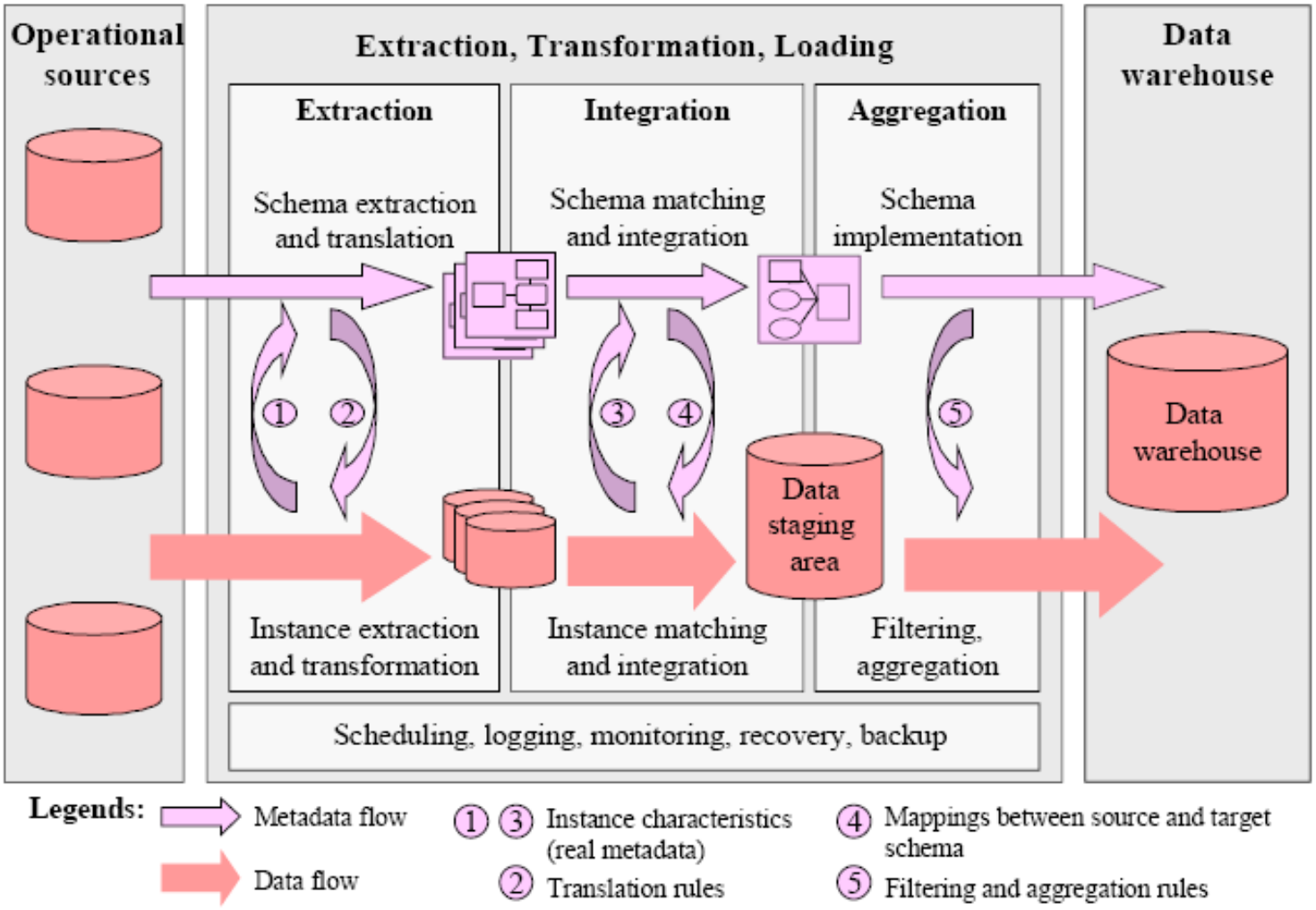
Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

ETL (cont'd.)

- **When** should we ETL?
  - Periodically (e.g., every night, every week) or after significant events
  - Refresh policy set by administrator based on user needs and traffic
  - Possibly different policies for different sources
  - Rarely, on every update (real-time DW)
    - Not warranted unless warehouse data require current data (up to the minute stock quotes)

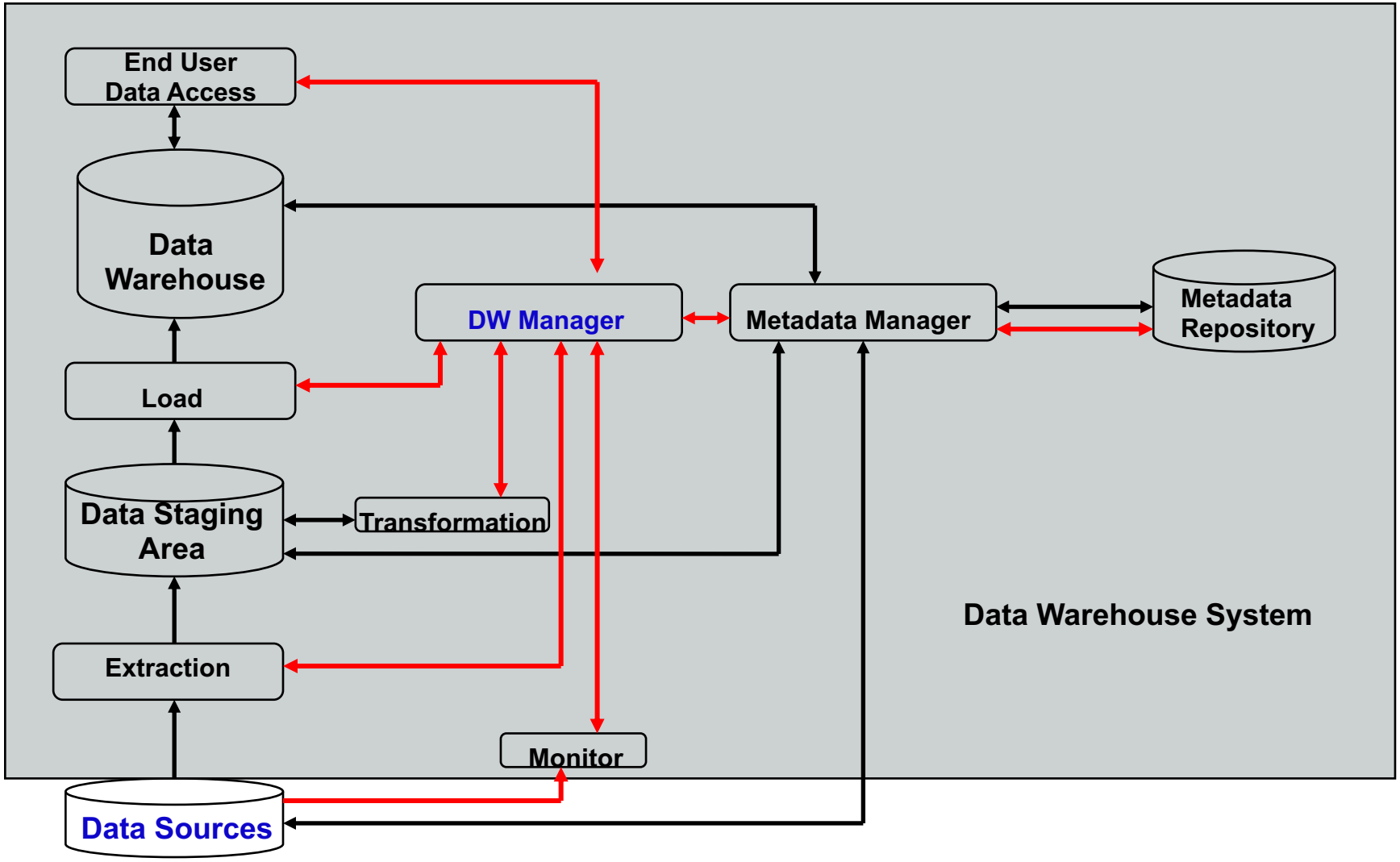
Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

ETL (cont'd.)



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

ETL (cont'd.)



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Extraction

- **Data Extraction**
  - Data needs to be taken from a data source so that it can be put into the DW
    - Internal scripts/tools at the data source, which **export** the data to be used
    - External programs, which **extract** the data from the source
  - If the data is exported, it is typically exported into a **text file** that can then be brought into an intermediary database
  - If the data is extracted from the source, it is typically transferred directly into an **intermediary database**

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Extraction (cont'd.)

- **Steps** in data extraction
  - Initial extraction
    - Preparing the **logical map**
    - First time data extraction
  - Ongoing extraction
    - Just **new** data
    - **Changed** data
    - Or even **deleted** data

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Extraction (cont'd.)

- **Logical map** connects the original source data to the final data
- **Building the logical map**: first identify the data sources
  - Data discovery phase
    - Collecting and documenting source systems: databases, tables, relations, cardinality, keys, data types, etc.
  - Anomaly detection phase
    - NULL values can destroy any ETL process, e.g., if a foreign key is NULL, joining tables on a NULL column results in data loss, because in RDB NULL ≠ NULL in DW

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Extraction (cont'd.)

- Example of solving NULL anomalies
  - If NULL on **foreign key** then use **outer joins**
  - If NULL on other columns then create a **business rule** to replace NULLs while loading data in DW
- Ongoing Extraction
  - Data needs to be **maintained** in the DW also after the initial load
    - Extraction is performed on a regular basis
    - Only changes are extracted after the first time
- **How do we detect changes?**

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Ongoing Extraction

- Detecting changes (new/changed data)
  - Using audit columns
  - Database log scraping or sniffing
  - Process of elimination
- Using **audit columns**
  - Store the date and time a record has been added or modified at
  - Detect changes based on **date stamps higher than the last extraction date**

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Detecting Changes

- **Log scraping**
  - Takes a snapshot of the database **redo log** at a certain time (e.g., midnight) and finds the transactions affecting the tables ETL is interested in
  - It can be problematic when the redo log gets full and is emptied by the DBA
- **Log Sniffing**
  - Pooling the redo log at small time granularity, capturing the transactions on the fly
  - The better choice: suitable also for real-time ETL

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Detecting Changes

- **Process of Elimination**
  - Preserves exactly one copy of each previous extraction
  - During next run, it compares the entire source tables against the extraction copy
  - Only differences are sent to the DW
  - Advantages
    - Because the process makes row by row comparisons, it is **impossible to miss data**
    - **It can also detect deleted rows**

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

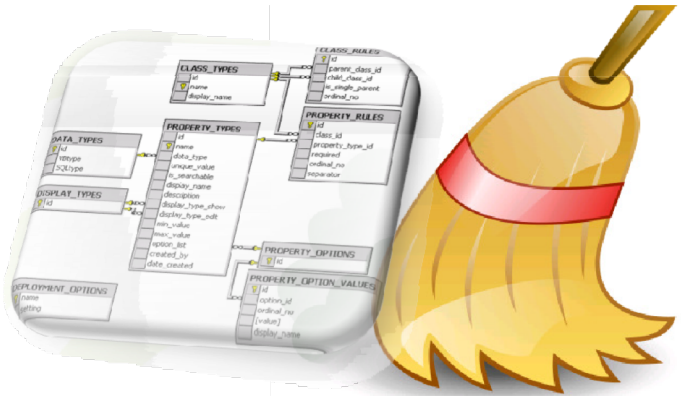
Detecting Changes

- Detecting deleted or overwritten fact records
  - If records or incorrect values are inserted by mistake, records in ODS get deleted or overwritten
  - If the mistakes have already been loaded in the DW, corrections have to be made
  - In such cases the solution is not to modify or delete data in the DW, but inserting an additional record which **corrects** or even **cancels** the mistake by negating it

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Transformation

- **Data transformation**
  - Uses **rules**, **lookup tables**, or combinations with other data (master data), to convert source data to the desired state
- 2 major steps
  - Data Cleaning
    - Mostly involves manual work
    - Assisted by artificial intelligence algorithms and pattern recognition
  - Data Integration
    - Also involves manual work



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Cleaning

- Extracted data can be dirty. How does clean data look like?
- **Data Quality** characteristics:
  - **Correct**: values and descriptions in data represent their associated objects truthfully
    - E.g., if the city in which store A is located is Braunschweig, then the address should not report Paris
  - **Unambiguous**: the values and descriptions in data can be taken to have only one meaning

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

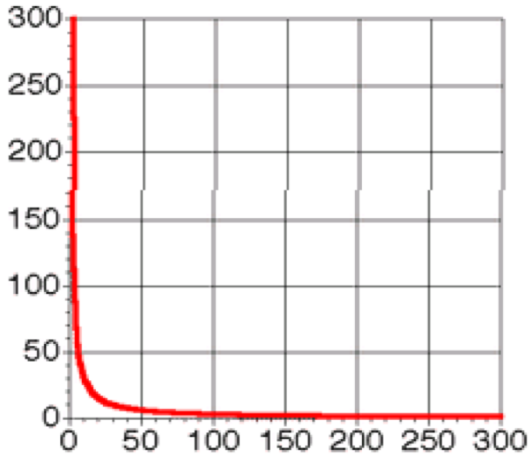
Cleaning Engine

- **Core** of the data cleaning engine
  - Break data into atomic units
    - E.g., breaking the address into street, number, city, zip and country
  - Standardizing
    - E.g., encoding of the sex: 0/1, M/F, m/f, male/female
  - Verification
    - E.g., does zip code 38106 belong to Braunschweig?
  - Matching

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Quality (cont'd.)

- Data profiling
  - Pay closer look to strange values
  - Observe data distribution pattern
- Data distribution
  - Flat distribution
    - Identifiers distributions (keys)
  - Zipfian distribution
    - Some values appear more often than others
      - In sales, more cheap goods are sold than expensive ones



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

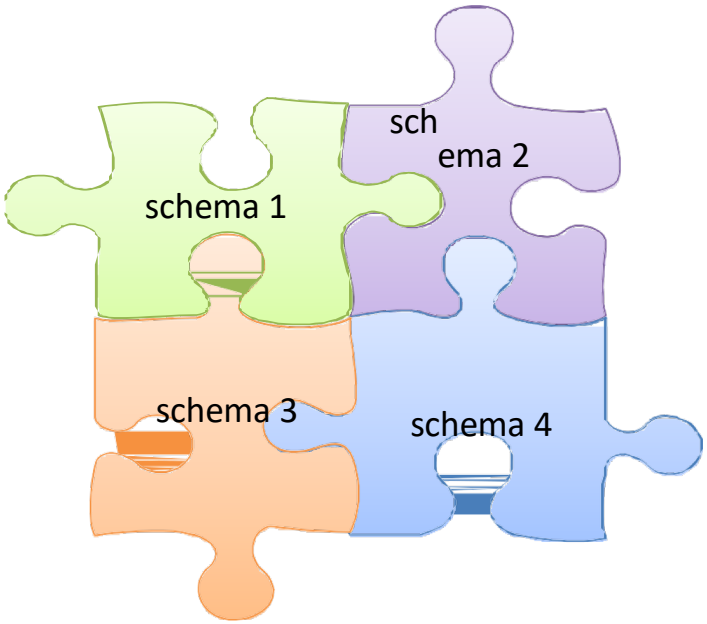
Data Quality Checks

- Types of enforcement
  - Column property enforcement
    - Ensures that incoming data contains expected values
    - No NULL values in required columns
    - No numeric values outside the expected high/low ranges
    - No columns whose lengths are unexpectedly short or long
    - No columns that contain values outside of valid value sets
    - Spell-checker rejects

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Integration

- Data integration
  - Several conceptual schemas need to be combined into a unified global schema
  - All differences in perspective and terminology have to be resolved
  - All redundancy has to be removed



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Data Integration (cont'd.)

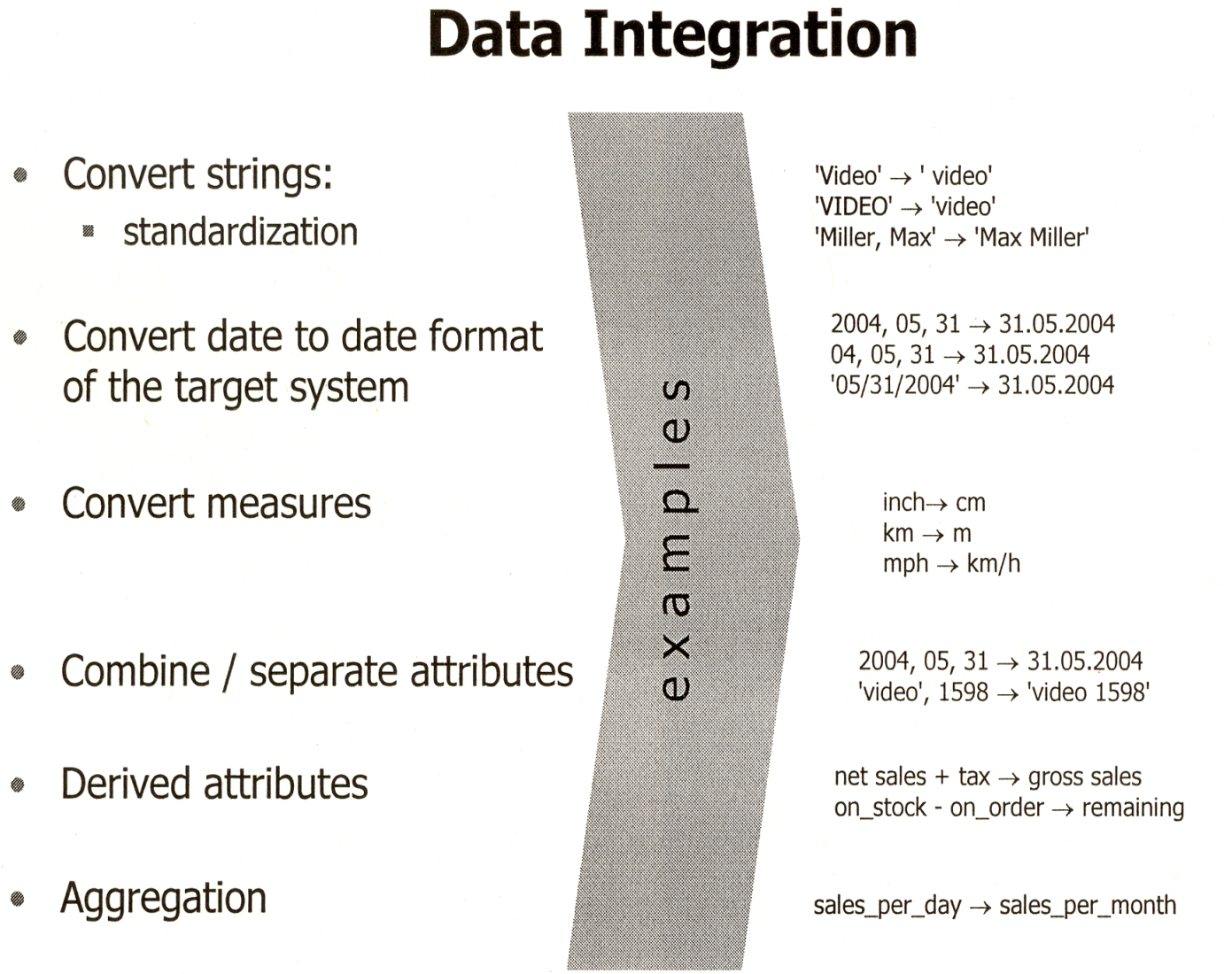
customer

system	local key	global key
1	107	5400345
1	109	5401340
2	107	4900342
2	214	5401340

character→ date
character → integer
'MM-DD-YYYY' → 'DD.MM.YYYY'

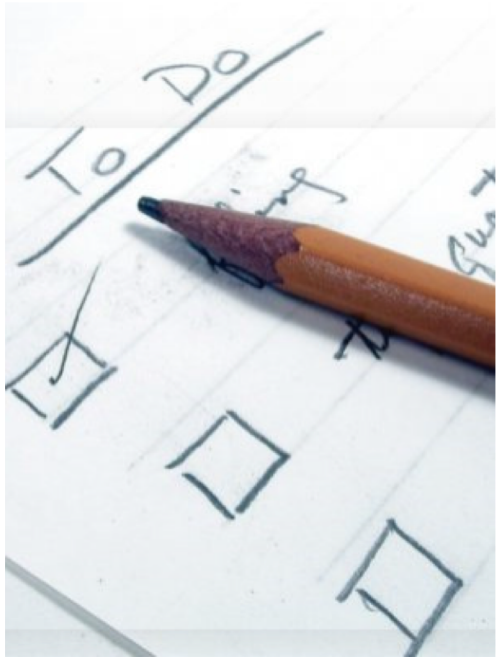
gross sales → 1
net sales → 2
3 → price
2 → GS

Data Integration (cont'd.)



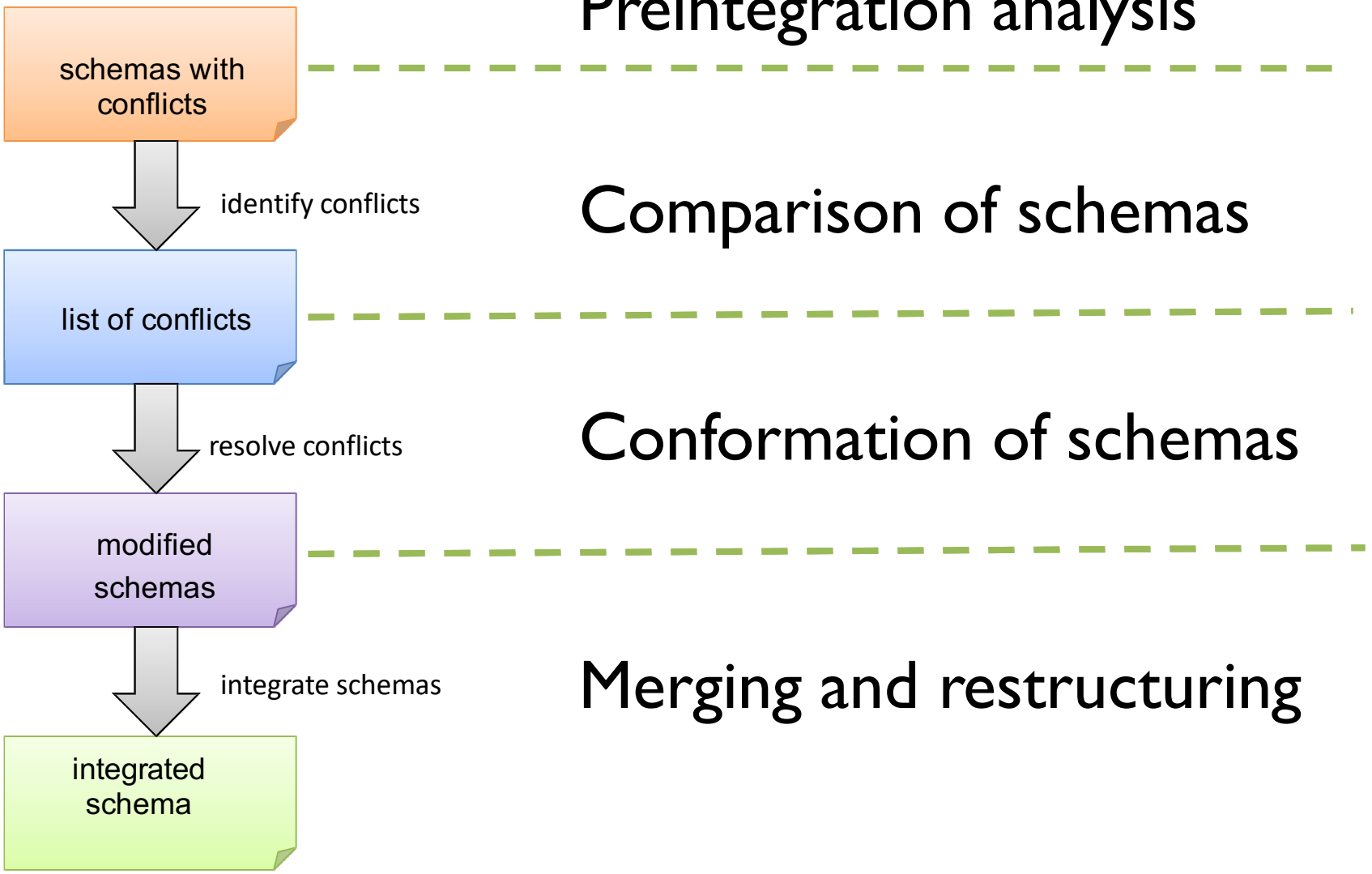
Schema Integration

- There are **four basic steps** needed for conceptual schema integration
  1. Preintegration analysis
  2. Comparison of schemas
  3. Conformation of schemas
  4. Merging and restructuring of schemas
- The integration process needs continual refinement and reevaluation



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Schema Integration (cont'd.)

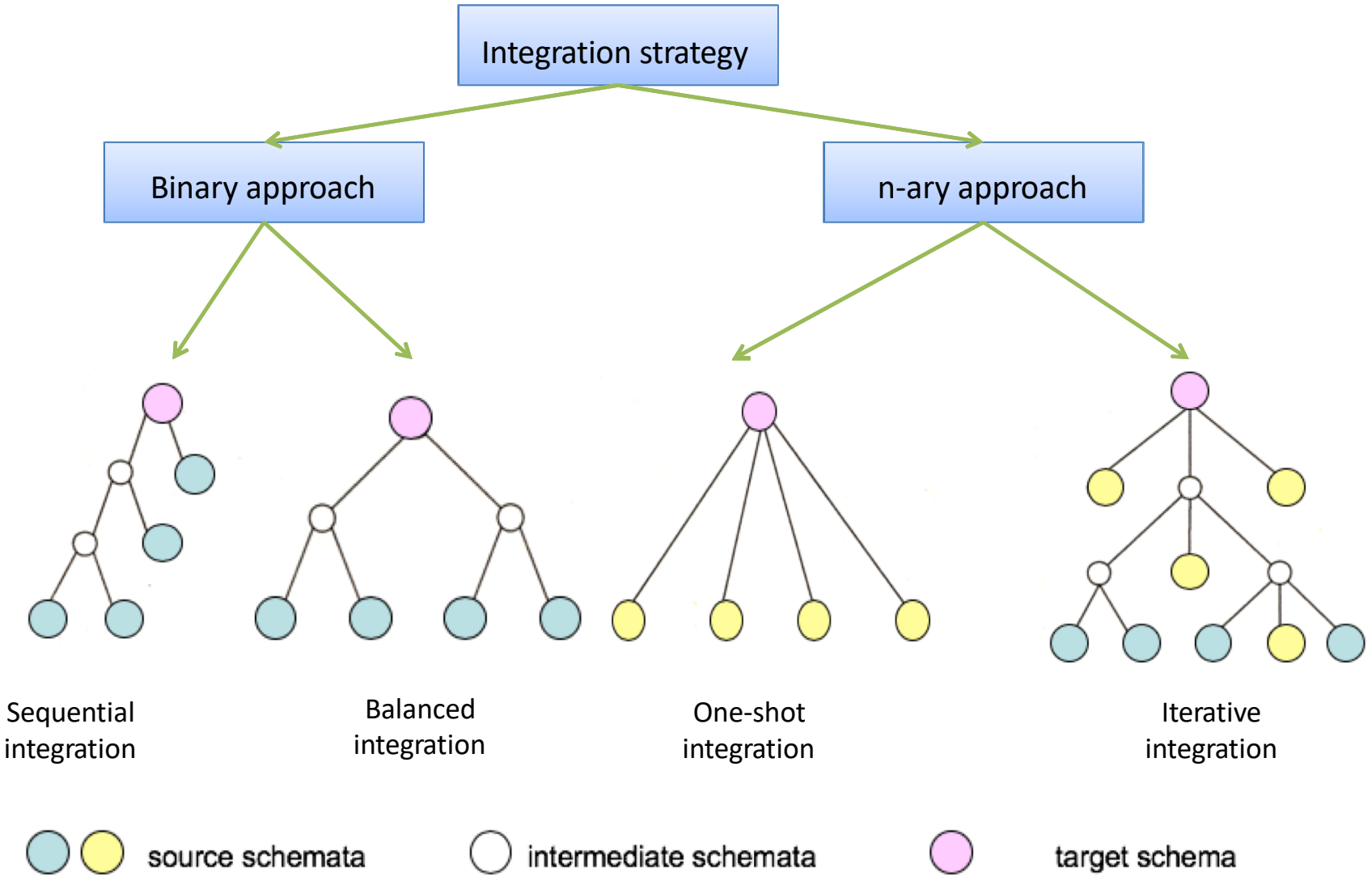


Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Schema Integration (cont'd.)

- **Preintegration analysis** needs a close look on the individual conceptual schemas to decide for an **adequate integration strategy**
  - Is it really sensible/possible to integrate all schemas? e.g. no need to load attribute TransactionID into DW.

Schema Integration (cont'd.)



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Schema Integration (cont'd.)

- **Schema conflicts**
  - Table/Table conflicts
    - Table naming e.g., synonyms
    - Structure conflicts e.g., missing attributes
    - Conflicting integrity conditions
  - Attribute/Attribute conflicts
    - Naming e.g., synonyms
    - Default value conflicts
    - Conflicting integrity conditions e.g., different data types or boundary limitations
  - Table/Attribute conflicts

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

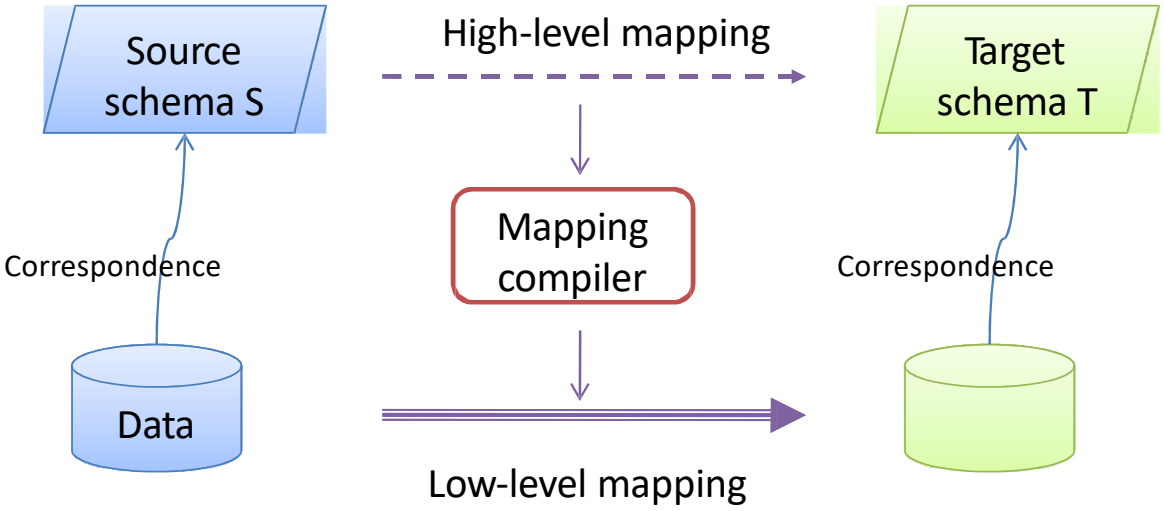
Schema Integration (cont'd.)

- The basic goal is to make schemas **compatible** for integration
- Conformation usually needs **manual** interaction
  - Conflicts need to be **resolved semantically**
  - Rename entities/attributes
  - Convert differing types, e.g., convert an entity to an attribute or a relationship
  - Align cardinalities/functionalities
  - Align different datatypes

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Schema Integration (cont'd.)

- Mapping from one or more source schemas to a target schema



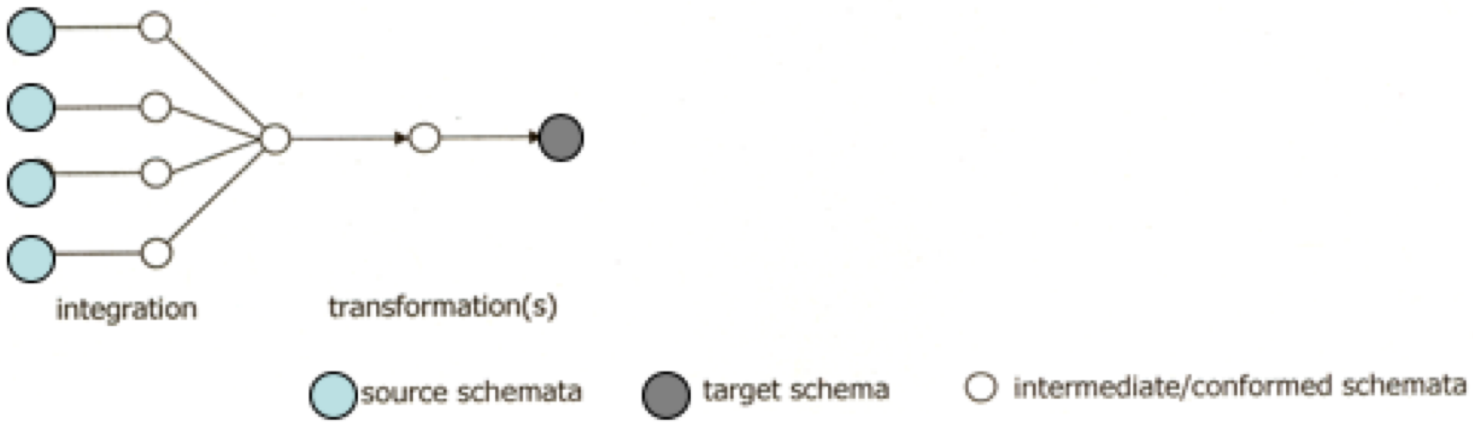
Schema Integration (cont'd.)

- Schema Mapping
  - Abstracting from the actual labels, regarding element types, depths in hierarchies, number and type of relationships, etc.

Product
ID: Decimal
Product: VARCHAR(50)
GroupID:Decimal

→

Product
ProdID: Decimal
Product: VARCHAR(50)
Group: VARCHAR(50)
Categ: VARCHAR(50)



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Loading

- The **loading** process can be broken down into 2 different types:
  - Initial load
  - Continuous load (loading over time)

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

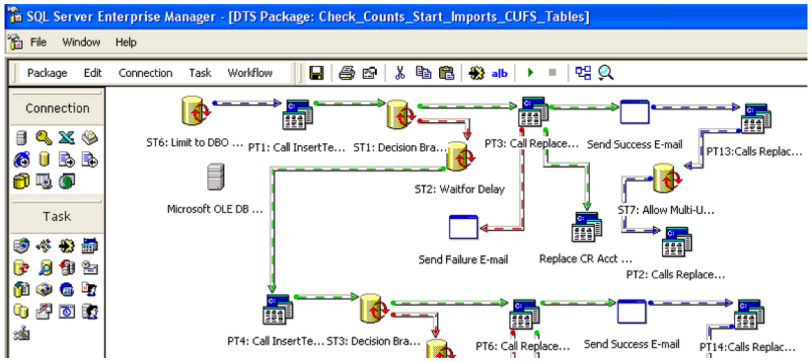
Loading (cont'd.)

- **Issues**
  - Huge volumes of data to be loaded
  - Small time window available when warehouse can be taken off line (usually nights)
  - When to build index and aggregated tables
  - Allow system administrators to monitor, cancel, resume, change load rates
  - **Recover gracefully** - restart after failure from where you were and without loss of data integrity

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Loading (cont'd.)

- **Initial Load**
  - Deliver dimensions tables
    - Create and assign surrogate **keys**, each time a new cleaned and conformed dimension record has to be loaded
    - Write dimensions to disk as physical tables, in the **proper dimensional format**
  - Deliver fact tables
    - Utilize bulk-load utilities
    - Load in parallel
  - Tools
    - DTS – DataTransformation Services (set of tools)
    - bcp utility – batch copy
    - SQL* Loader



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Loading (cont'd.)

- **Continuous load** (loading over time)
  - Must be scheduled and processed in a specific order to maintain integrity, completeness, and a satisfactory level of trust.
  - Should be the most carefully planned step in data warehousing or can lead to:
    - Error duplication
    - Exaggeration of inconsistencies in data

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Loading (cont'd.)

- Continuous load of **facts**
  - Separate updates from inserts
  - Drop any indexes not required to support updates
  - Load updates
  - Drop all remaining indexes
  - Load inserts through bulk loaders
  - Rebuild indexes

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Loading (cont'd.)

MERGE INTO

- Transaction table cust_napier contains updates to existing rows in DW and/or new rows that should be inserted
- MERGE statement of SQL 2003 allows us to:
  - Update (address in) rows that have a matching counterpart in the target table
  - Insert rows that do not have a matching counterpart in the target table

```
MERGE INTO customer as c1
USING (  SELECT key, name, addr, …
         FROM cust_napier
         WHERE …) as c2
ON (c1.key = c2.key)
WHEN MATCHED THEN
         UPDATE SET c1.addr = c2.addr
WHEN NOT MATCHED THEN
         INSERT (key, name addr, …
         VALUES (key, name, addr, …)
```

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Loading (cont'd.)

Multi-table INSERT

- Insert rows into several target tables
- Insert the same row into several target tables

Conditional Insert

- Can define conditions to select target table

Unconditional INSERT

```
INSERT ALL
    INTO customer
    VALUES (key, name, addr,…)
    INTO location
        VALUES (addr, 'Napier')
SELECT * FROM cust_napier
WHERE …
```

Conditional INSERT

```
INSERT ALL
WHEN KEY < 100
    INTO customer
    VALUES (key, name, addr,…)
WHEN KEY < 1000
    INTO location
     VALUES (addr, 'Napier')
SELECT * FROM cust_napier
WHERE …
```

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

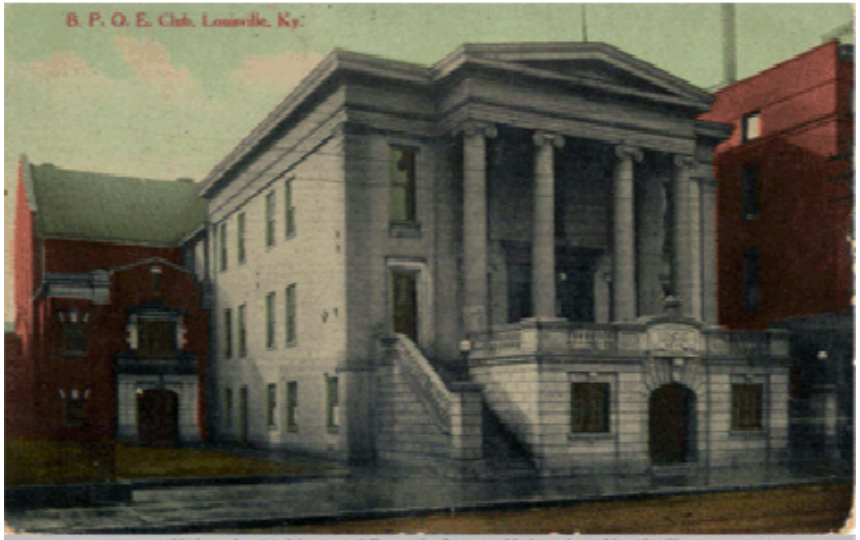
Metadata

- **Metadata** - data about data
  - In DW, metadata describe the contents of a data warehouse and how to use it
    - What information exists in a data warehouse, what the information means, how it was derived, from what source systems it comes, when it was created, what pre-built reports and analyses exist for manipulating the information, etc.

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Metadata (Cont'd.)

- **Types** of metadata in DW
  - Source system metadata
  - Data staging metadata
  - DBMS metadata



Title	B.P.O.E. Club, Louisville, Ky.
Description	Benevolent and Protective Order of Elks hall in Louisville, Kentucky, three-quarters view. Published by Bagby-Howe Drug Co. Postmarked 1910 (on verso).
Subject	Fraternal lodges Elks (Fraternal order). Lodge No. 8 (Louisville, Ky.)
Location Depicted	Louisville (Ky.)
Date Original	1910?
Object Type	Postcards
Source	3x5 color lithograph postcard published by Bagby-Howe Drug Co. Item no. 008.028 in the Morton Owen Postcard Collection, University of Louisville Archives and Records Center
Collection	Morton Owen Postcard Collection, University of Louisville Archives and Records Center
Digital Publisher	University of Louisville Archives and Records Center
Format	image/jpeg
Image Number	ULVA.OOP.028

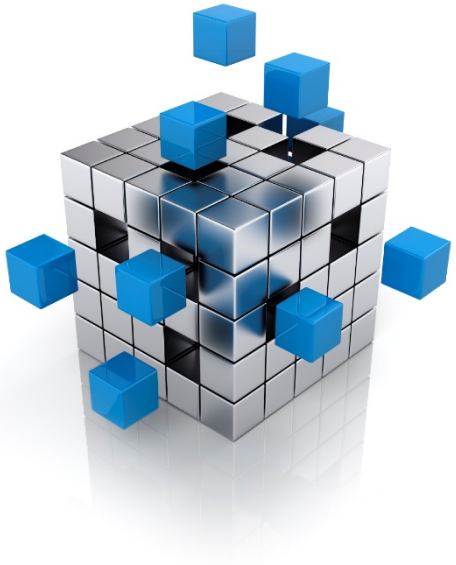
Summary

- How to build a DW
  - The DW Project: usual tasks, hardware, software, timeline (phases)
  - Data Extract/Transform/Load (ETL):
    - Data storage structures, extraction strategies (e.g., scraping, sniffing)
    - Transformation: data quality, integration
    - Loading: issues, and strategies, (bulk loading for fact data is a must)
  - Metadata:
    - Describes the contents of a DW, comprises all the intermediate products of ETL,
    - Helps for understanding how to use the DW

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Next Lecture

- Real-time Data Warehousing
  - Real-time Data Processing Challenges
  - Real-time Join Algorithms



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides