ETL Process

Unit 3

Content

Introduction ETL Process (Extract, Transform, Load)
the role of ETL in data warehousing
data extraction methods
Data transformation techniques
data cleaning
loading data into the data warehouse
ETL tools and technologies
case studies of ETL processes in industry

The ETL Process Explained

Extract

Retrieves and verifies data from various sources

Transform

Processes and organizes extracted data so it is usable

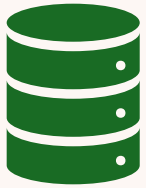
Load

Moves transformed data to a data repository

Technical Architecture of Data Warehouse

Data Acquisition –

The process of extracting and transforming data from various sources.



Data Storage –

The process of storing the transformed data in the data warehouse.

Information Delivery –

The process of making the stored data available for analysis and reporting.

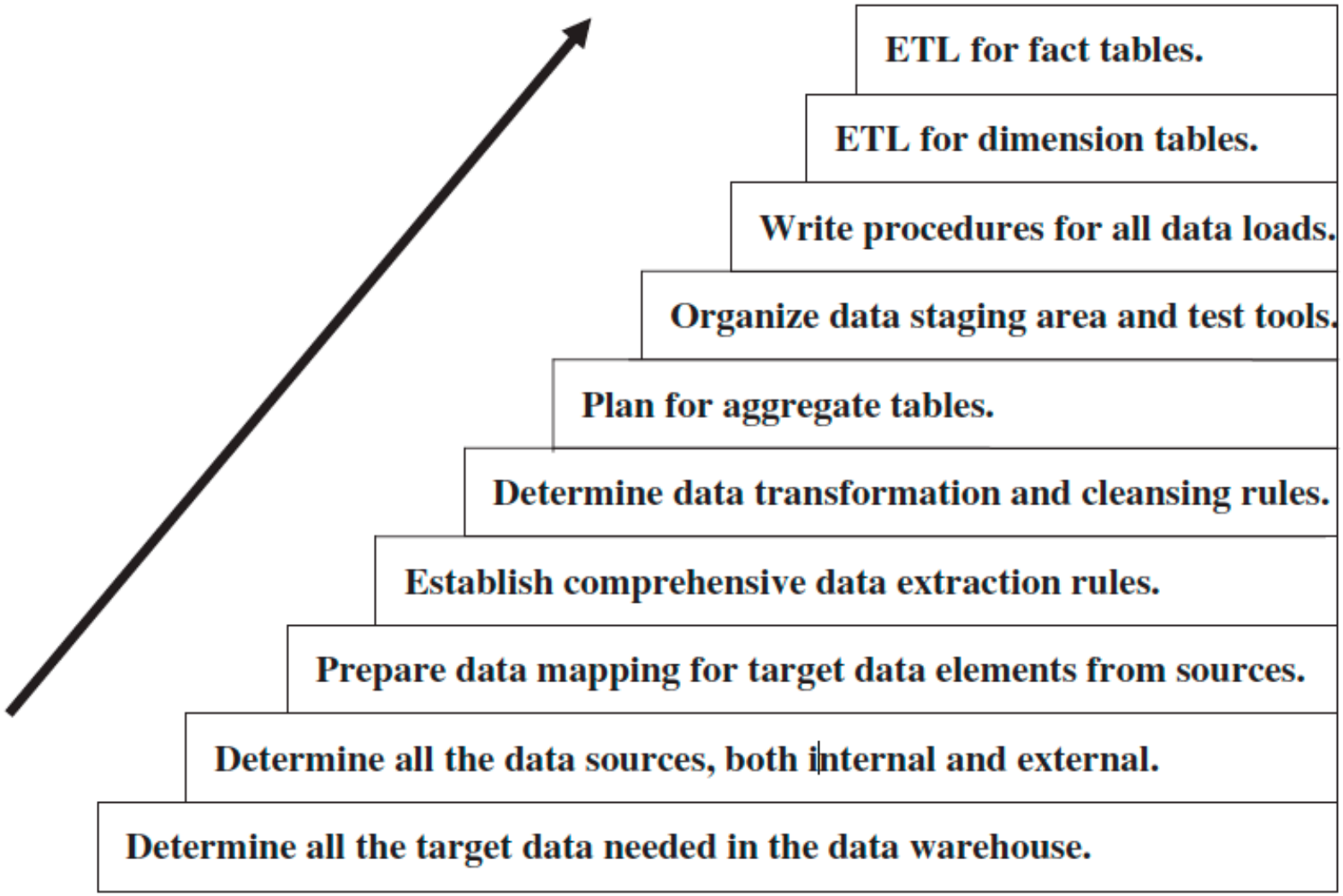


Figure 12-1 Major steps in the ETL process.

Major steps in ETL

Determine all the target data needed in the data warehouse:	Identify the specific data required for storage and analysis.
Determine all the data sources, both internal and external:	Recognize the sources providing the data, including databases, APIs, and external files.
Prepare data mapping for target data elements from sources:	Define how data from different sources will be transformed and mapped to the target warehouse structure.
Establish comprehensive data extraction rules:	Set rules for retrieving data efficiently and accurately from source systems.
Determine data transformation and cleansing rules:	Specify how data should be cleaned, standardized, and transformed to fit the warehouse schema.

Major steps in ETL

Plan for aggregate tables:	Decide on summary tables for efficient query performance and reporting.
Organize data staging area and test tools:	Prepare a temporary storage area for data before final loading, along with validation tools.
Write procedures for all data loads:	Document the steps and processes for loading data systematically into the warehouse.
ETL for dimension tables:	Load and structure dimension tables, which store descriptive data (e.g., customer or product details).
ETL for fact tables:	Load fact tables containing transactional data, often used for analytical reporting.

DATA EXTRACTION

DEFINITION :

Data extraction involves retrieving relevant data from operational systems or external sources.

This step precedes all other functions in ETL.

Challenges in Data Extraction

Determining the scope and extent of extraction: Extracting all available data is inefficient and may lead to a "data junkhouse."

User requirement-driven approach: Only extract data relevant to business needs.

Heterogeneous data sources: Operational systems often store data in different formats and structures, requiring reconciliation.

Best Practices for Data Extraction

Define **clear extraction criteria** based on user requirements.

Select data from **reliable and validated sources**.

Avoid unnecessary data duplication.

Operational Systems vs. Data Warehouse Extraction

Operational Systems:

- Data extraction is typically **one-time**, transferring data from legacy systems (e.g., VSAM files) to a new **relational database**.

Data Warehouse:

- Extraction is **ongoing** and must handle data from **multiple disparate sources** while supporting both **initial full loads** and **incremental updates**.

Increased Complexity in Data Warehouse Extraction

Multiple Data Sources:

- Requires consolidating structured (databases) and unstructured (flat files, APIs) data.

Incremental Data Updates:

- Ongoing extraction to capture **only changes** rather than full refreshes.

Third-Party Data Extraction Tools vs. In-House Scripts

Third-Party Tools:

- More expensive but provide **metadata management** and **built-in flexibility**.
- Easier to adapt to changing business conditions by simply modifying input parameters.

In-House Programs:

- More **cost-effective initially** but expensive to **maintain** over time.
- Require frequent updates when **source systems change**.

List of data extraction issues

Source identification	identify source applications and source structures.
Method of extraction	for each data source, define whether the extraction process is manual or tool-based.
Extraction frequency	for each data source, establish how frequently the data extraction must be done: daily, weekly, quarterly, and so on.
Time window	for each data source, denote the time window for the extraction process.
Job sequencing	determine whether the beginning of one job in an extraction job stream must wait until the previous job has finished successfully.
Exception handling	determine how to handle input records that cannot be extracted.

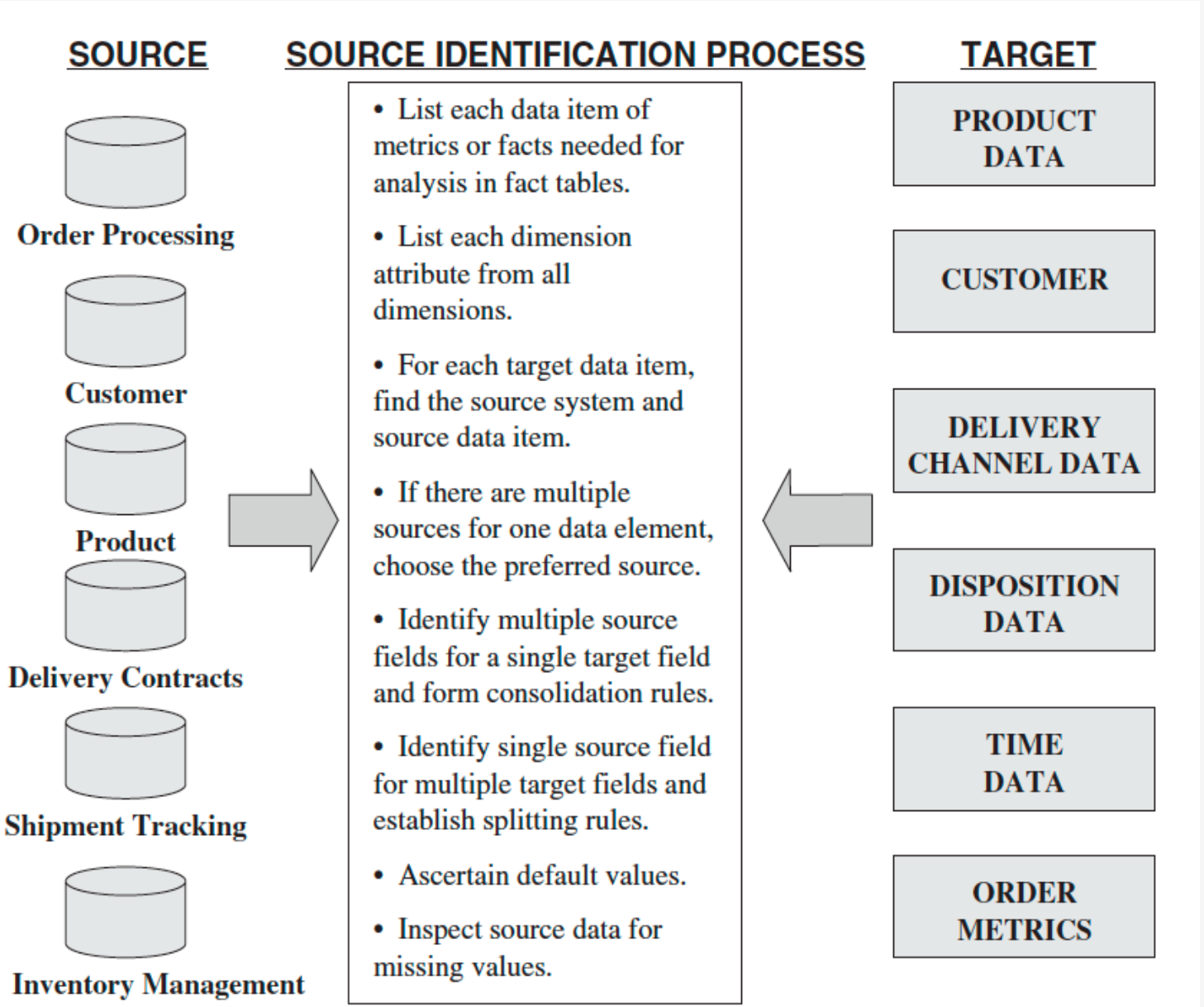


Figure 12-2 Source identification: a stepwise approach.

Source Identification

- **Source identification** is the **first step** in data extraction and plays a crucial role in ensuring that the right data is collected from **reliable and relevant sources**

Key Aspects of Source Identification

Beyond Just Identifying Sources

It is not enough to just list data sources; each source must be **examined and verified** to ensure it provides valuable, high-quality data for the data warehouse.

Example: Order Fulfilment Data Mart

Suppose you are building a **data mart** to analyse order fulfilment.

You need **historical and current data** on fulfilled and pending orders.

You must consider factors such as:

- **Delivery channels** (multiple shipping methods)
- **Order status tracking** (as orders move through the process)

Fact and Dimension Tables in Source Identification

The **fact table** may include attributes like:

- Total order amount
- Discounts and commissions
- Expected vs. actual delivery time
- Various timestamps in the fulfilment process

Key Aspects of Source Identification

The **dimension tables** may include:

Product details

Order disposition (order status tracking)

Delivery channels

Customer information

Steps in Source Identification

Determine available source systems for the required data.

Identify the correct source for each data element in the data mart.

Verify that the selected sources are accurate, complete, and useful.

Challenges in Source Identification

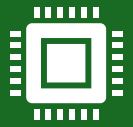
Complexity: It requires deep analysis and validation for every data point.

Time-consuming: Thoroughness is necessary to avoid incorrect or inconsistent data.

Multiple sources: Data may come from **disparate systems**, requiring integration.

Nature of the source data and how the extracted data will be used

Source data is in constant flux due to ongoing business transactions.



Most source systems store only the current value of an attribute, reflecting its state at that moment in time.

However, data warehouses require historical data to analyse trends and patterns over time

EXAMPLES OF ATTRIBUTES

VALUES OF ATTRIBUTES AS STORED IN OPERATIONAL SYSTEMS AT DIFFERENT DATES

Storing Current Value

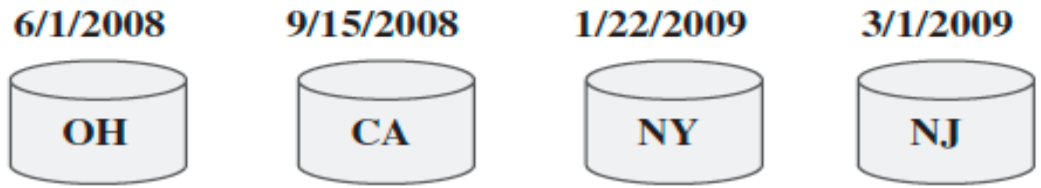
Attribute: Customer's State of Residence

6/1/2008 Value: OH

9/15/2008 Changed to CA

1/22/2009 Changed to NY

3/1/2009 Changed to NJ



Storing Periodic Status

Attribute: Status of Property consigned to an auction house for sale.

6/1/2008 Value: RE (property receipted)

9/15/2008 Changed to ES (value estimated)

1/22/2009 Changed to AS (assigned to auction)

3/1/2009 Changed to SL (property sold)

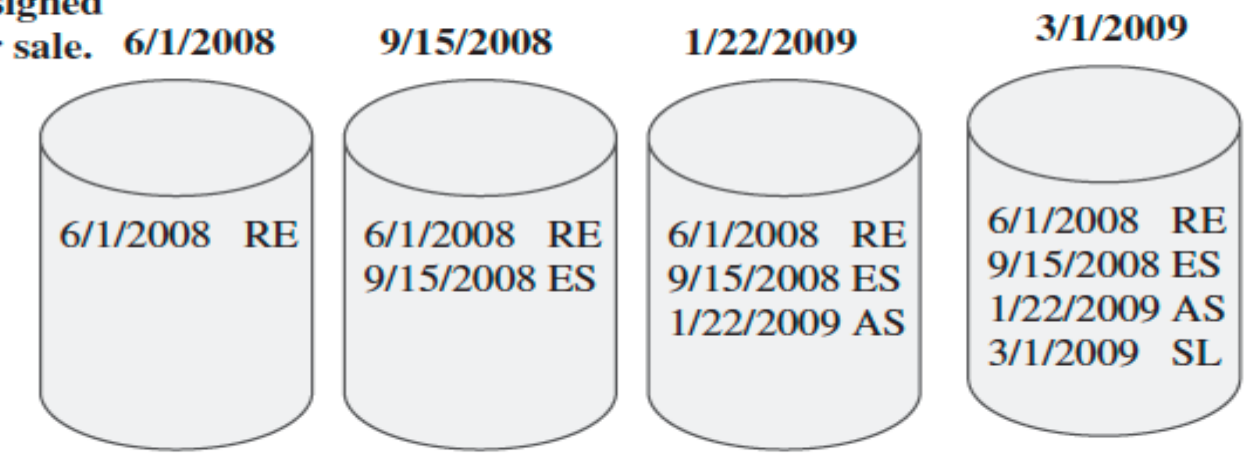


Figure 12-3 Data in operational systems.

Operational systems store data in two main categories:

Current Value

- Most attributes in source systems fall into this category, where the stored value represents the attribute's current state.
- Examples include customer addresses, bank account balances, and outstanding order amounts.
- Since these values change unpredictably, capturing historical changes for the data warehouse can be challenging.

Periodic Status

- This category preserves the value of an attribute at each change event.
- Status data is recorded with timestamps, making it easier to track historical changes.
- Examples include insurance policy records and claim processing steps.

Understanding these categories helps in selecting the appropriate data extraction method.

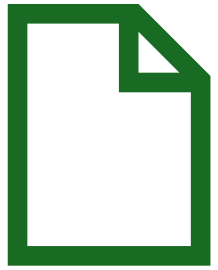
Types of Data Extraction

Static Data Capture (“As-Is” Extraction):

Captures a snapshot of data at a specific point in time.

Used for initial data loads and full refreshes of tables.

Commonly applied when source system data undergoes major structural changes.



Incremental Data Capture (Revisions Extraction):

Captures only the changes (updates, inserts, deletes) since the last extraction.

Used for maintaining an up-to-date data warehouse without duplicating data.

Particularly useful for tracking changes in historical data.

Immediate Data Extraction Methods

Transaction Log Capture:

Uses database transaction logs to extract committed changes.

Efficient and incurs minimal overhead since logging is inherent in database operations.

Not applicable for non-database source systems.

Ensures all transactions are extracted before log files are refreshed.

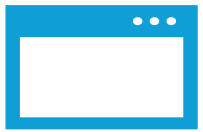
Database Triggers:

Uses triggers to capture and log changes in database tables.

Can capture both before and after values of an attribute.

Increases system overhead and complexity due to additional trigger execution.

Only applicable to database-driven source systems.



Application-Assisted Capture:

Modifies source applications to log changes separately during data entry.

Provides flexibility but requires development effort.

Suitable for cases where transaction logs and triggers are not viable.

Immediate Data Extraction Methods

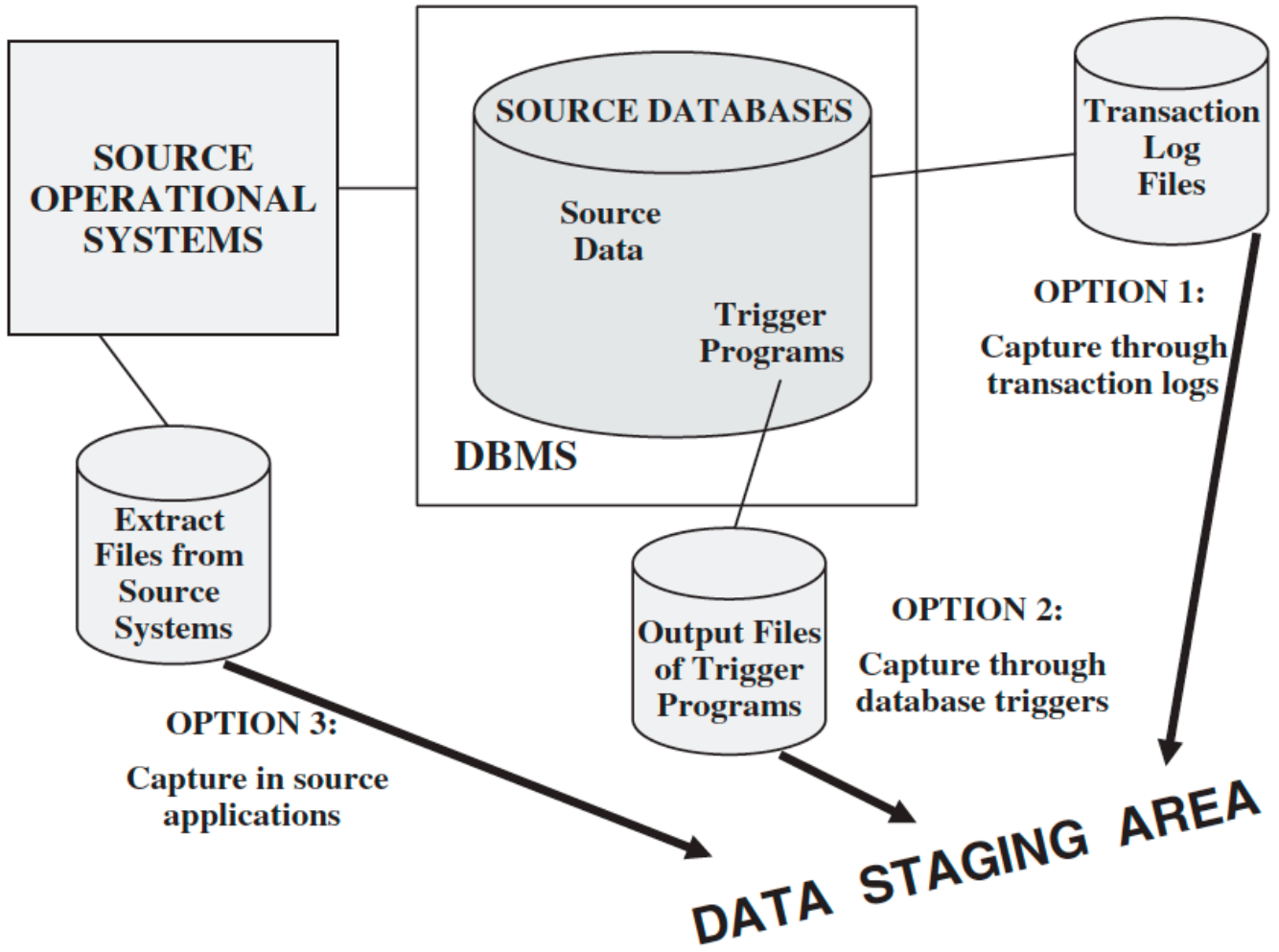


Figure 12-4 Options for immediate data extraction.

Steps for using replication to capture

Identify the source system database table
Identify and define target files in the staging area
Create mapping between the source table and target files
Define the replication mode
Schedule the replication process
Capture the changes from the transaction logs
Transfer captured data from logs to target files
Verify transfer of data changes
Confirm success or failure of replication
In metadata, document the outcome of replication
Maintain definitions of sources, targets, and mappings

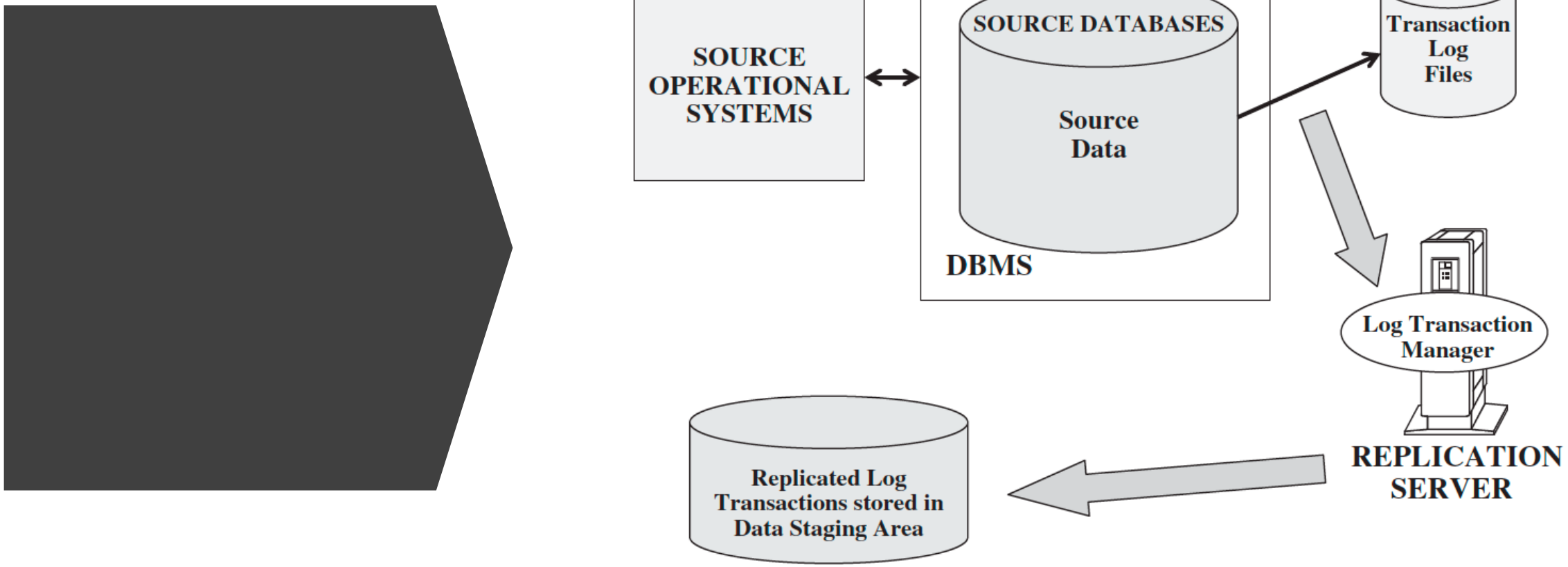


Figure 12-5 Data extraction using replication technology.

Deferred Data Extraction

Capture Based on Date and Time Stamp

- Each source record is marked with a timestamp upon creation or update.
- Data extraction occurs later by selecting records with timestamps after the last extraction time.
- Effective when the number of revised records is small.
- Requires all source records to have date and time stamps.
- Captures only the latest state; intermediate changes between extractions are lost.
- Deleted records are not detected unless explicitly marked before extraction.

Capture by Comparing Files (Snapshot Differential Technique)

- Compares two snapshots of source data (e.g., today's and yesterday's copies).
- Identifies inserts, updates, and deletes by comparing record keys.
- Requires storing previous versions of the source data.
- Simple but inefficient for large files.
- Useful for legacy systems that lack transaction logs or timestamps.

Deferred Data Extraction

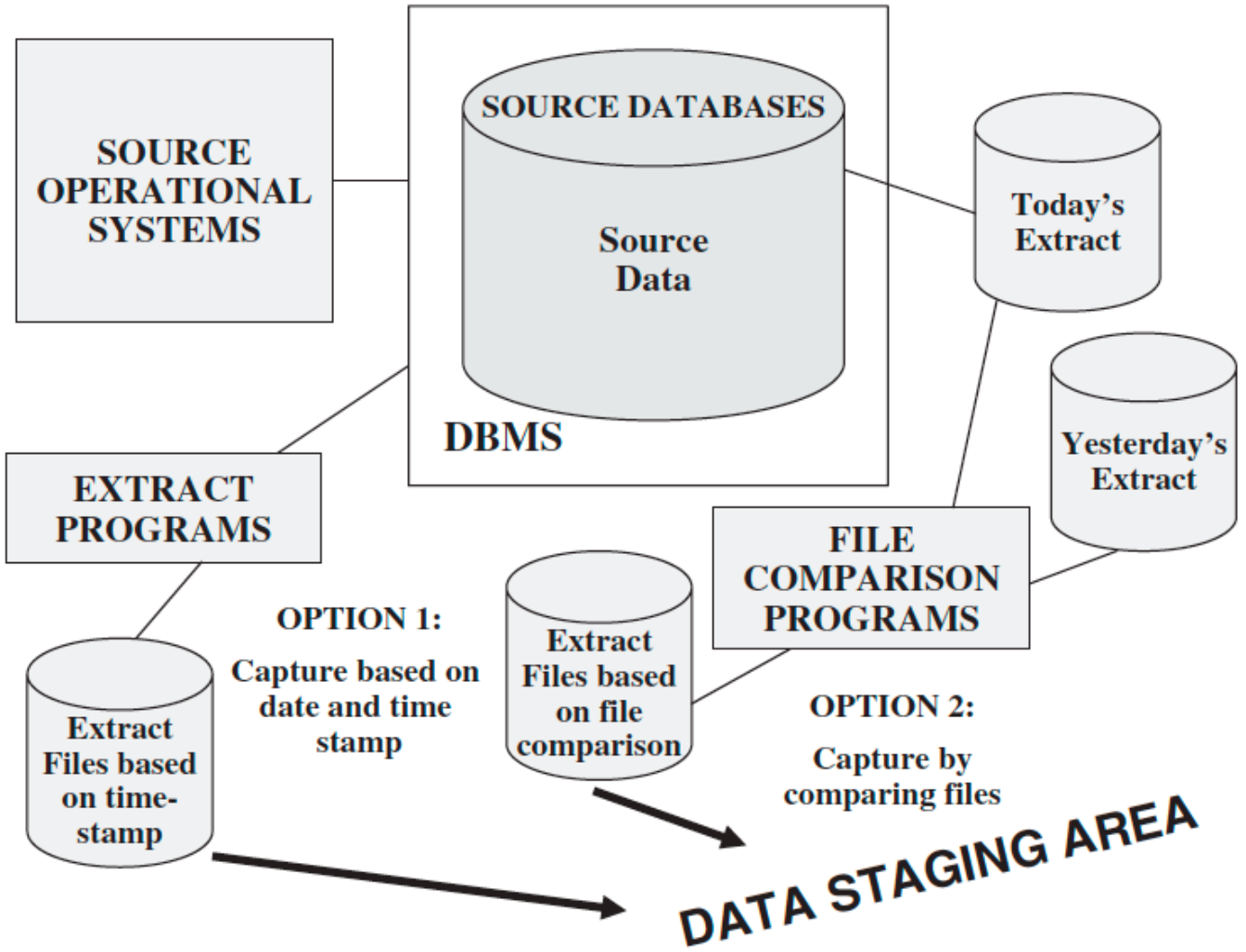


Figure 12-6 Options for deferred data extraction.

Capture of static data

Good flexibility for capture specifications.
Performance of source systems not affected.
No revisions to existing applications.
Can be used on legacy systems.
Can be used on file-oriented systems.
Vendor products are used. No internal costs.

Capture in source applications

Good flexibility for capture specifications.
Performance of source systems affected a bit.
Major revisions to existing applications.
Can be used on most legacy systems.
Can be used on file-oriented systems.
High internal costs because of in-house work.

Capture through transaction logs

Not much flexibility for capture specifications.
Performance of source systems not affected.
No revisions to existing applications.
Can be used on most legacy systems.
Cannot be used on file-oriented systems.
Vendor products are used. No internal costs.

Capture based on date and time stamp

Good flexibility for capture specifications.
Performance of source systems not affected.
Major revisions to existing applications likely.
Cannot be used on most legacy systems.
Can be used on file-oriented systems.
Vendor products may be used.

Capture through database triggers

Not much flexibility for capture specifications.
Performance of source systems affected a bit.
No revisions to existing applications.
Cannot be used on most legacy systems.
Cannot be used on file-oriented systems.
Vendor products are used. No internal costs.

Capture by comparing files

Good flexibility for capture specifications.
Performance of source systems not affected.
No revisions to existing applications.
May be used on legacy systems.
May be used on file-oriented systems.
Vendor products are used. No internal costs.

Figure 12-7 Data capture techniques: advantages and disadvantages.

Data Transformation for Data Warehousing

Raw Data Processing

- Extracted data from operational systems is raw and cannot be directly applied to a data warehouse.
- Data must be enriched and improved for strategic decision-making.

Challenges of Operational Data

- Data from legacy systems often lacks quality and consistency.
- Data must be standardized to avoid business rule violations.

Need for Data Transformation

- Ensures compatibility among data from different sources.
- Converts extracted data into usable information.
- Prevents structural and semantic inconsistencies in the data warehouse.

Data Transformation for Data Warehousing

Complexity of Data Transformation

- Organizations often underestimate the complexity of transformation.
- Simple transformations involve field conversions and reformatting.
- Complex transformations involve multiple operations across different systems.

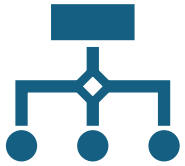
Classification of Transformations

- Simple transformations: Basic field conversions and formatting.
- Complex transformations: Integration, mapping, and preprocessing.

Importance of Data Quality

- Includes filling in missing values and ensuring accuracy.
- Poor data quality can lead to incorrect strategic decisions.
- Requires careful attention and dedicated effort.

Data Transformation: Basic Tasks

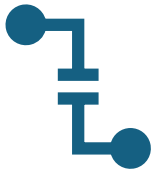


Selection

Performed at the beginning of data transformation.

Involves selecting whole records or specific parts from source systems.

May be done during extraction or separately in transformation.



Splitting/Joining

Splitting: Dividing selected parts into smaller components (less common).

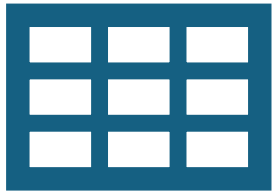
Joining: Combining data from multiple sources (more common in data warehousing).

Conversion

Standardizes data from different sources.

Makes fields more usable and understandable for users.

Data Transformation: Basic Tasks



Summarization

Reduces data granularity when detailed data is unnecessary.

Example: Instead of storing every sales transaction, aggregate sales by product/store/day.

Enrichment

Enhances data quality and usability.

Rearranges and simplifies fields to provide a better data structure.

May combine multiple fields from different records into a single useful field.

Major Transformation Types

Transformation Type	Description	Example
Format Revisions	Changes data types and field lengths for consistency.	Standardizing product package types from numeric to text.
Decoding of Fields	Converts cryptic codes into meaningful values.	Changing gender codes (1,2 → M,F) or business status codes (AC → Active).
Calculated and Derived Values	Computes new fields based on existing data.	Calculating profit margin using sales and cost data.
Splitting of Single Fields	Divides large text fields into separate components.	Splitting "John A. Smith" into first, middle, and last name.
Merging of Information	Combines data from multiple sources into a unified dataset.	Integrating product code, description, and cost from different sources.
Character Set Conversion	Standardizes character encoding for compatibility.	Converting EBCDIC (mainframe) to ASCII (PC-based system).

Major Transformation Types

Transformation Type	Description	Example
Conversion of Units of Measurement	Standardizes measurement units across global datasets.	Converting metric units (kilograms) to imperial units (pounds).
Date/Time Conversion	Standardizes date formats across systems.	Converting US format (MM/DD/YYYY) to International format (DD MON YYYY).
Summarization	Stores aggregated data instead of raw granular data.	Storing daily credit card transactions instead of individual transactions.
Key Restructuring	Replaces business-dependent keys with generic system-generated keys. Fig:12.8	Avoiding product codes with warehouse locations as primary keys.
Deduplication	Removes duplicate records to maintain a single entity per record.	Merging duplicate customer records into a single master record.

Key Restructuring

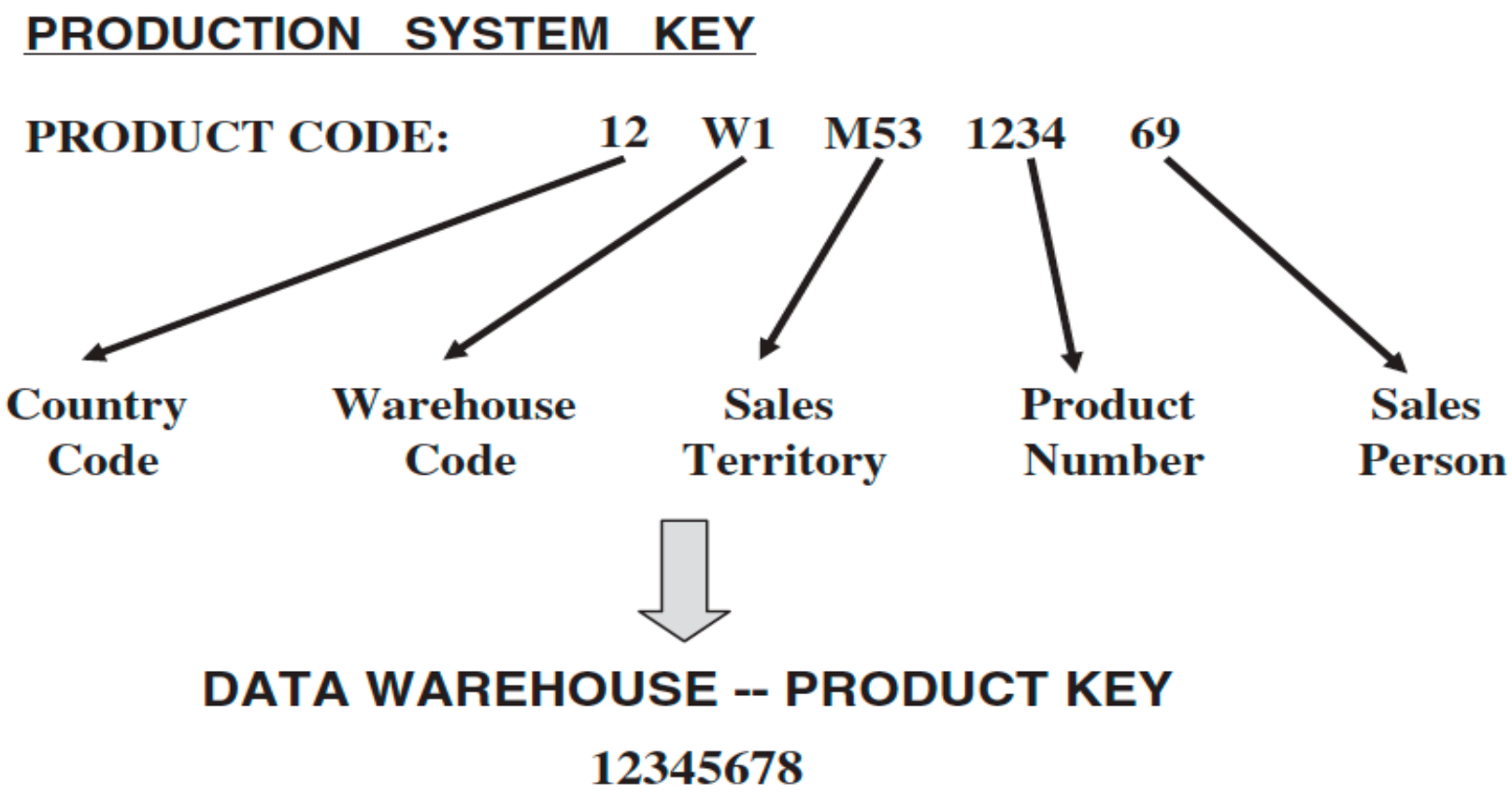


Figure 12-8 Data transformation: key restructuring.

Data Integration and Consolidation

- The real challenge of ETL functions is the pulling together of all the source data from many disparate, dissimilar source systems.
- Many data warehouses get data extracted from a combination of legacy mainframe systems and old minicomputer applications, in addition to newer client/server systems.
- Most of these source systems do not conform to the same set of business rules.
- Very often they follow different naming conventions and varied standards for data representation.
- Figure 12-9 shows a typical data source environment.

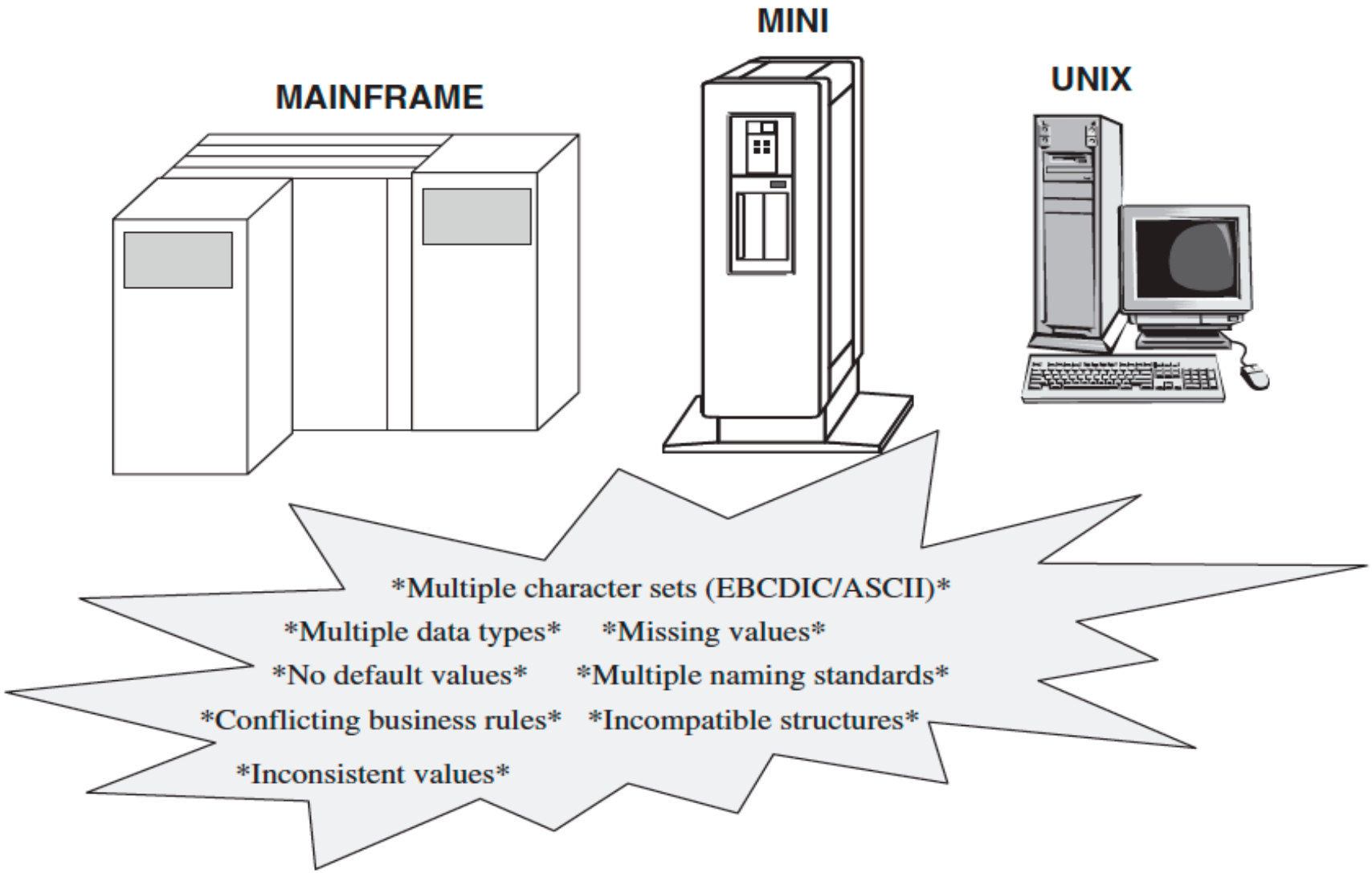


Figure 12-9 Typical data source environment.