Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

Chapter 14 - Data Replication and Materialized Integration

Motivation

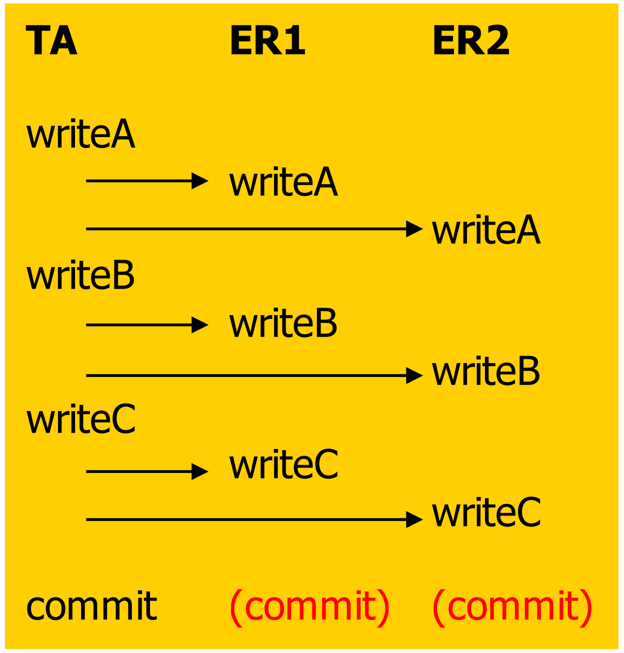
- Replication: controlled redundancy of data to
  - increase availability
  - improve performance (query response time)
- Replication is a common concept in
  - (homogeneous) distributed DBMS
  - centralized DBMS in the form of materialized views
  - mobile DBMS environments
  - data/information integration middleware
- Materialized integration: data warehouses
  - replication of data from multiple sources into a central data warehouse
    - avoid performance problems of virtual integration solutions
    - diverts query load away from operational data sources
    - enables complex and powerful data analysis (business intelligence)
  - major problem
    - potential for stale (out-of-date) data

Challenges

- Integration of replication with transaction processing
- Enforcing consistency of replica in the presence of updates
  - goal: one-copy serializability
    Execution has the same effect as a serial execution on a one-copy database!
  - problem example:
    $H = r_1[x_A]w_1[x_A]\ w_1[x_B]c_1\ r_2[x_B]w_2[x_B]c_2\ r_3[x_A]w_3[x_A]w_3[x_B]c_3$
    - here: T_2 does not write X_A before it is read by T_3!
  - each transaction that updates X should update all copies of X
    - not always possible (replica may be unavailable)
  - each transaction that reads X should read a copy of X that was written by the most recent committed transaction that updated any copy of X
    - requires careful synchronization

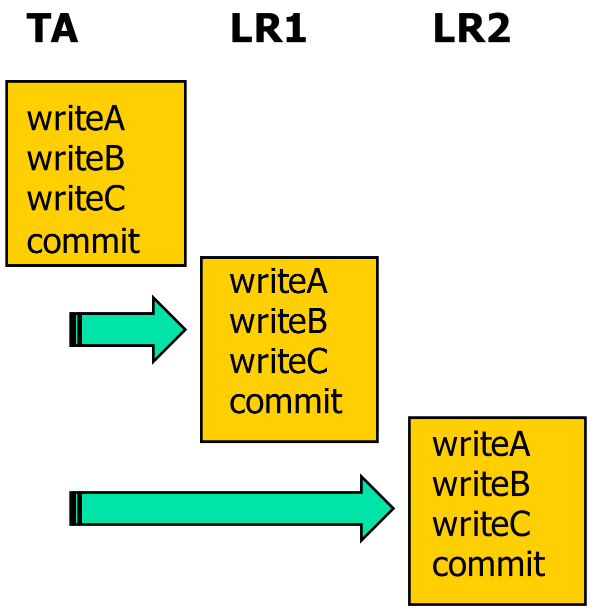
Eager (Synchronous) Replication

- Transaction synchronizes with copies of replicated elements before commit
  - guarantees one-copy serializable execution
    - locking
  - avoids inconsistencies
- Potential problems
  - update overhead
    - reduced update performance (2PC)
    - increased transaction response time
  - deadlocks
  - lack of scalability
  - cannot be used if nodes are disconnected (e.g., mobile databases) or unavailable



Lazy (Asynchronous) Replication

- Changes introduced at one site are propagated (as separate transactions) to other sites only after commit
  - updates are applied in the same order at all replicas
- Advantages
  - minimal update overhead (i.e., improved response times over eager replication)
  - works also if sites are not connected (e.g., mobile environments) or temporarily unavailable
- Potential problems
  - stale (out-of-date) data
    - update of a completed transaction is "in-transit", i.e., has not been reflected in all replicas
    - new transaction operating on a replica sees an old version of the data
  - conflicting updates on different replicas can cause inconsistencies between the copies
    - ⇨ potential for "system delusion" (Gray, Reuter)
      - inconsistent database, with no obvious way to repair it
    - need to detect inconsistencies and reconcile conflicting transactions
      - rollback of (committed) transactions not possible



Single-Master Primary-Copy Replication

- Designate one replica as the primary copy
  - update transactions are only permitted on the primary copy
    - other nodes request the master node to perform an update (e.g., using RPC)
  - updates on the primary are propagated other replicas (i.e., secondaries) and applied in the order in which they executed at the primary
    - both eager and (most often) lazy replication can be used for that
  - avoids replication conflicts
- Applications
  - database mirroring (hot backup for high availability)
  - queryable copies of (parts of) a database (e.g., for decision support)
- Not suitable for mobile/disconnected environments
  - because update capability depending on connection to primary

Failures and Recovery

- Secondary fails
  - recovers and catches up processing stream of updates from the primary
  - may get a fresh DB copy, if downtime is too long
- Primary fails
  - Alternative 1: disallow updates until primary recovers
    - may be the only option, e.g. for queryable copies of DBs
  - Alternative 2 (for availability): a secondary must take over as primary
    - only one secondary: watchdog process may detect (type of) failure and direct the secondary to become the new primary, if needed
    - multiple secondaries: majority and quorum consensus algorithms for determining the (new) primary
- Does the new primary have the latest state?
  - cannot be guaranteed with lazy replication
  - tradeoff between performance and reliability

Multi-Master Replication

- Group Ownership
  - "update anywhere" model
  - any node/site with a replica can update it
    - may cause conflicts, need for reconciliation
- Two nodes may concurrently update replicas of the same object
  - "race" each other to propagate the update to all the other nodes
  - potential for lost updates
- Detecting conflicts
  - usually based on timestamps (or before-image data)
    - object carries timestamp of most recent update
    - replica update carries new value and old object timestamp
  - each node compares old object timestamp of incoming replica updates with its own object timestamp
    - if timestamps are the same, then the update is accepted
    - if not, then the incoming update transaction is rejected, submitted for reconciliation
      - rollback of transaction is not possible anymore, has been committed at the originating site
  - wait situations in eager replication ↔ reconciliation in lazy replication

Reconciliation

- Approaches
  - automatically, based on rules
    - site, time or value priority, merging of updates, …
  - manually
    - conflict situations are reported in a conflicts table or queue
- Non-transactional replication schemes
  - abandon serializability for convergence
    - all nodes converge to the same replicated state, which may not correspond to a serial transaction execution
  - tolerate lost updates

Alternatives for Conflict Detection, Avoidance

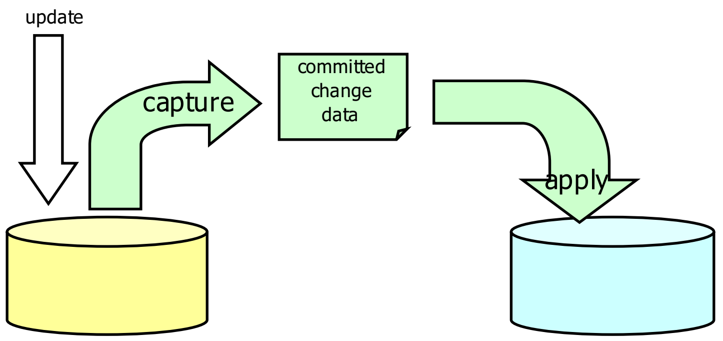
- Semantic synchronization
  - permit commutative transactions
    - requires capturing update semantics at a logical level
    - performing the transaction update may yield different results, but still be semantically correct
      - example: processing checks at a bank
  - provide acceptance criteria for detecting conflicts
    - incoming replica transaction updates are tentatively accepted and performed, but need to pass the acceptance test
    - replaces/augments the generic conflict detection mechanisms
- Avoid conflicts by implementing update strategies in the application
  - fragmentation by key
    - a site can update only rows whose keys are in a fixed range
    - no range overlaps
  - fragmentation by time
    - disjoint "time windows" for each site to perform updates

Replication Middleware

- Source table data is replicated to a target table
- Scenarios and uses
  - data distribution
    - data from one source table is replicated to more than one (read-only) target table
  - data consolidation/integration
    - data from more than one source table is replicated to a single target table (union view)
  - bidirectional replication allows updates on target tables to be replicated back to the source table
    - master/slave replication: all updates flow back to a designated master, are then distributed to other targets
    - peer-to-peer replication: each location exchanges data with all other locations
  - combination of the above
- Multi-tier replication
  - introduction of staging area(s)
    - changes are copied to another system
    - then copied from staging area to multiple targets
  - minimizes impact on source systems
- Lazy replication techniques are widely used

Replication methods

- Target table refresh
  - at intervals, target table is replaced by a fresh copy of the source table
  - no requirement to capture individual changes
  - only makes sense if
    - uni-directional replication is used (i.e., updates only occur on the source table)
    - target table is small or replication occurs infrequently
- Change-capture replication
  - committed changes to source table are captured and replicated to the target table
  - replication activity
    - continuous (near-real-time)
    - interval-based
      - e.g., during off-peak hours
    - triggered by DB-events
    - one-time snapshot
      - need to compare snapshots of tables to determine the updates



Capturing Changes

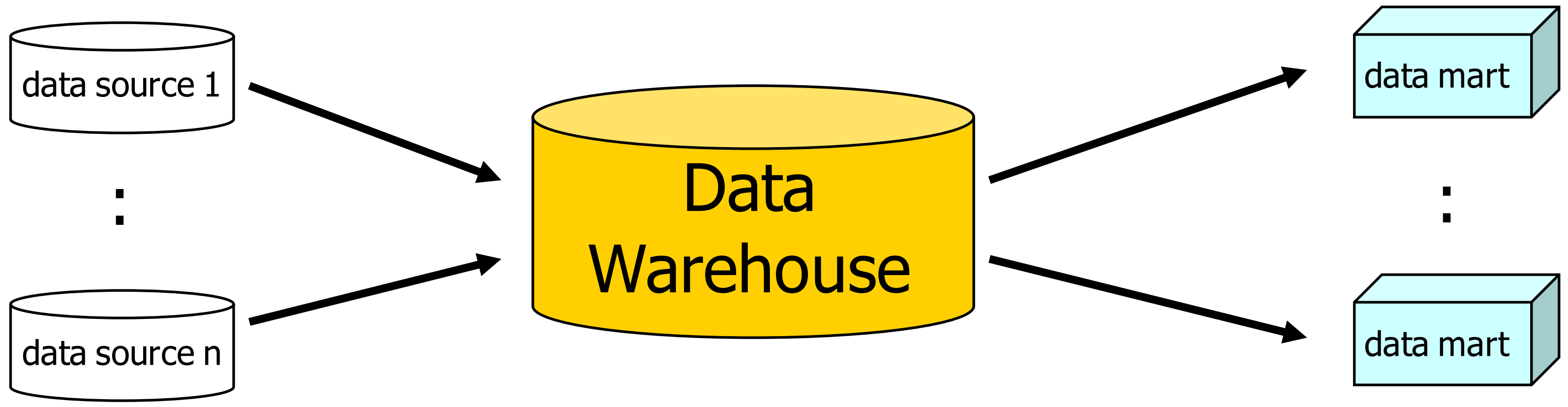
- Source table registration to define
  - which parts of the table changes should be captured
  - when replication should occur
- Capture logic
  - responsible for detecting the changes to the source table
  - make committed change data available to the apply logic
  - realization approaches
    - capture program analyzes the database log files, or
    - use database triggers
- Committed Change Data
  - needs to (at least) include
    - type of change (insert, update, delete)
    - new values of updated data items, plus data item identifier (keys)
    - (optional) before-image information
  - can be provided using
    - (relational) staging table at the source location
      - each change is reflected as a row in the staging table
    - message-oriented middleware
      - changes are provided as message on a queue

Applying Changes

- Apply logic is responsible for
  - initializing target table from source table
  - applying captured changes to the target table
    - preserve order of dependent transactions
- Needs access to
  - the captured changes stored in staging tables or change message queues
  - the target table
  - (the source table)
- Enhanced capabilities
  - filtering, joins, aggregation, transformation of data for the target

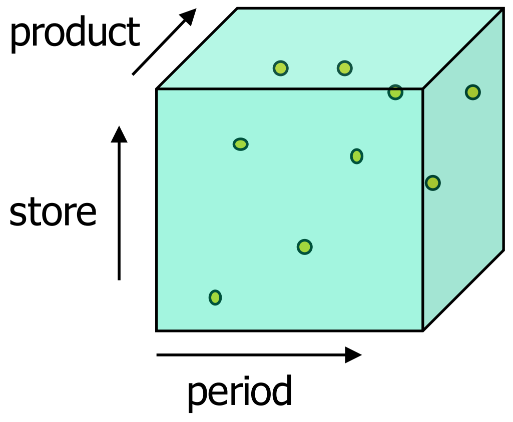
Data Warehousing

- Main goal: materialized integration of data from numerous heterogeneous sources to enable powerful strategic data analysis
  - OLAP – online analytical processing
  - data mining
  - often provided through (application-specific) data marts
    - data derived from a data warehouse through pre-aggregation
    - usually employ materialized views
- High-level architecture



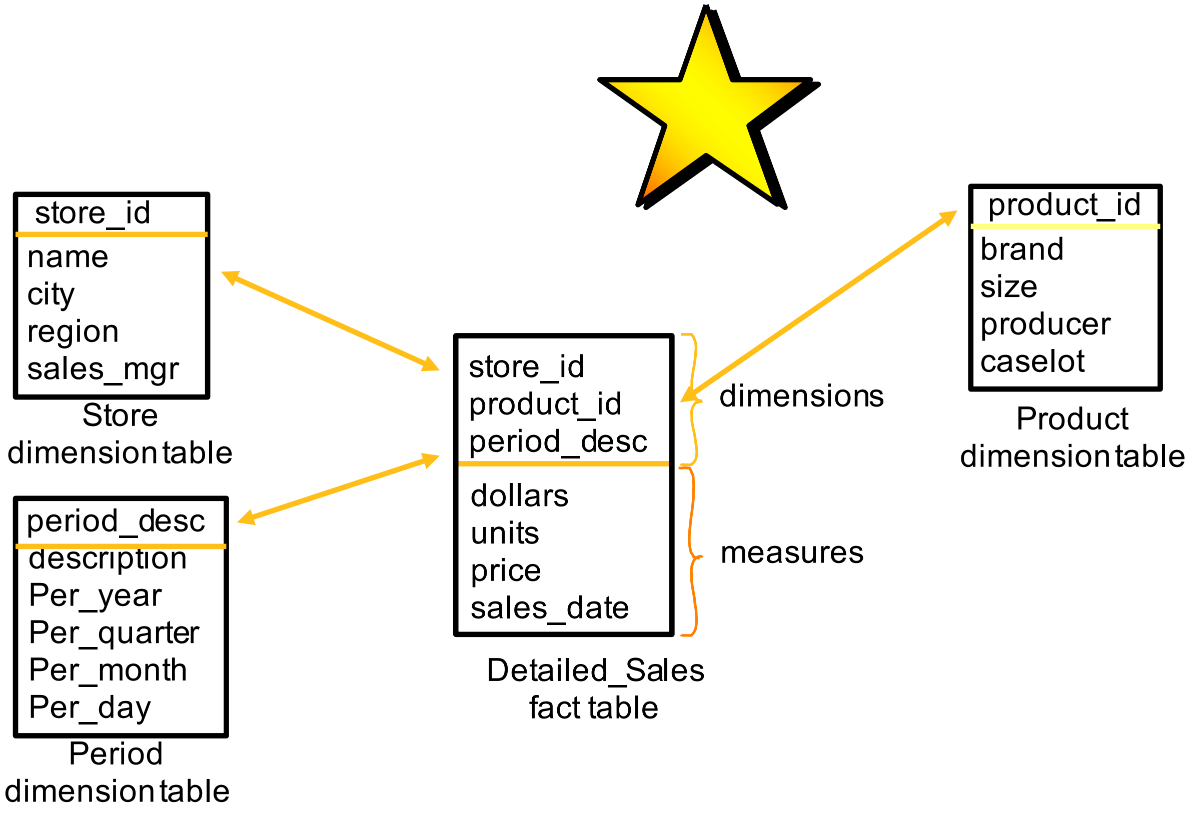
Multidimensional View of OLAP Data

- Facts
  - central relation or collection of data in an OLAP application
  - represents events or objects of interest
    - e.g., a sales event, with information about the product sold, the store, the sales date and price
- Dimensions
  - objects can often be thought of as arranged in a multi-dimensional space, or **cube**
    - e.g., sales events have store, product, and time period dimensions
    - each point is a single sales event, dimensions represent properties of the sale
  - hierarchical nature of dimensions
    - time: year, quarter, month, week, day
    - store: country, state, region, city

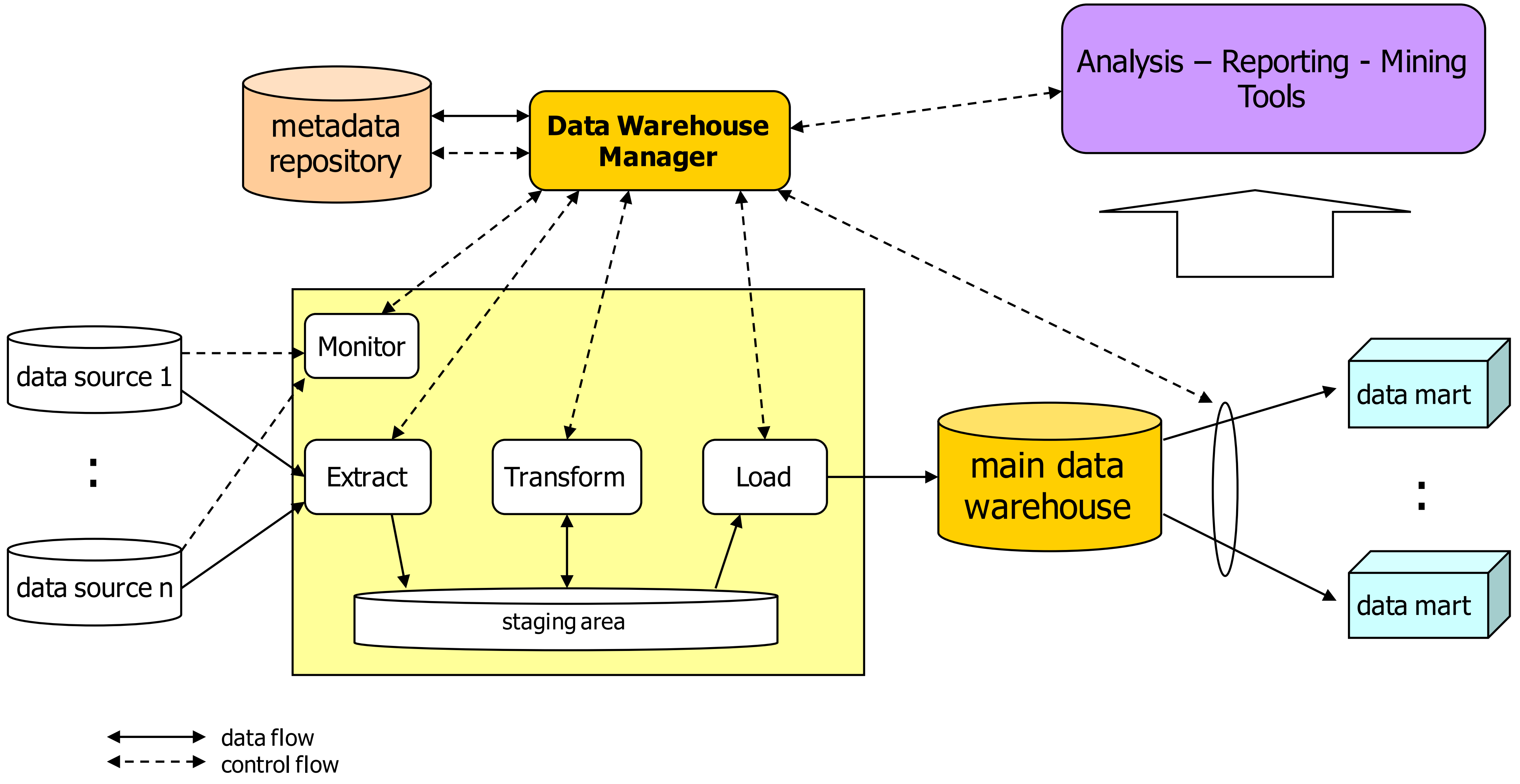


(Relational) OLAP Schema

- Typically uses a Star structure
  - Dimension tables (linked to fact table) tend to be small
  - Fact table tends to be huge
  - Measures (dependent attributes)
- Snowflake schema
  - "normalized" dimensions
  - multiple tables to avoid redundancy
  - requires additional joins for OLAP queries
- OLAP queries usually
  - GROUP BY the dimensions
  - compute aggregate values of measures



Data Warehousing Architecture



Data Warehouse Manager

- Central component of the architecture
- Responsible for controlling and supervising the overall process
  - initiate data preparation, loading
  - implement error recovery routines
  - manage ETL scripts or process descriptions
  - schedule and control analysis actions
- Directs data warehouse refresh
  - full load vs. incremental load
  - periodic (e.g., every night, weekend), driven by source updates (e.g., after n changes), or on request
- Utilizes metadata repository
- Monitors the overall DW environment

Data Preparation Components

- Data preparation steps (ETL) are conducted in a staging area
  - Monitor discovers and reports changes in data sources
    - e.g., replication-based (staging tables may be directly used by extractors)
  - **E**xtractors select and transport data from data sources into the staging area
    - DBMS or file system for managing the staging area
  - **T**ransformers perform standardization and integration of data
    - responsible for "implementing" an integrated schema
    - integrated data requires data quality! (see next chapter)
      - data migration, data cleaning
      - entity identification, duplicate elimination
    - may happen SQL-based or based on external data processing operators
  - **L**oaders insert the data from the staging area into the main warehouse
    - usually employ bulk load utilities of DBMS for performance reasons

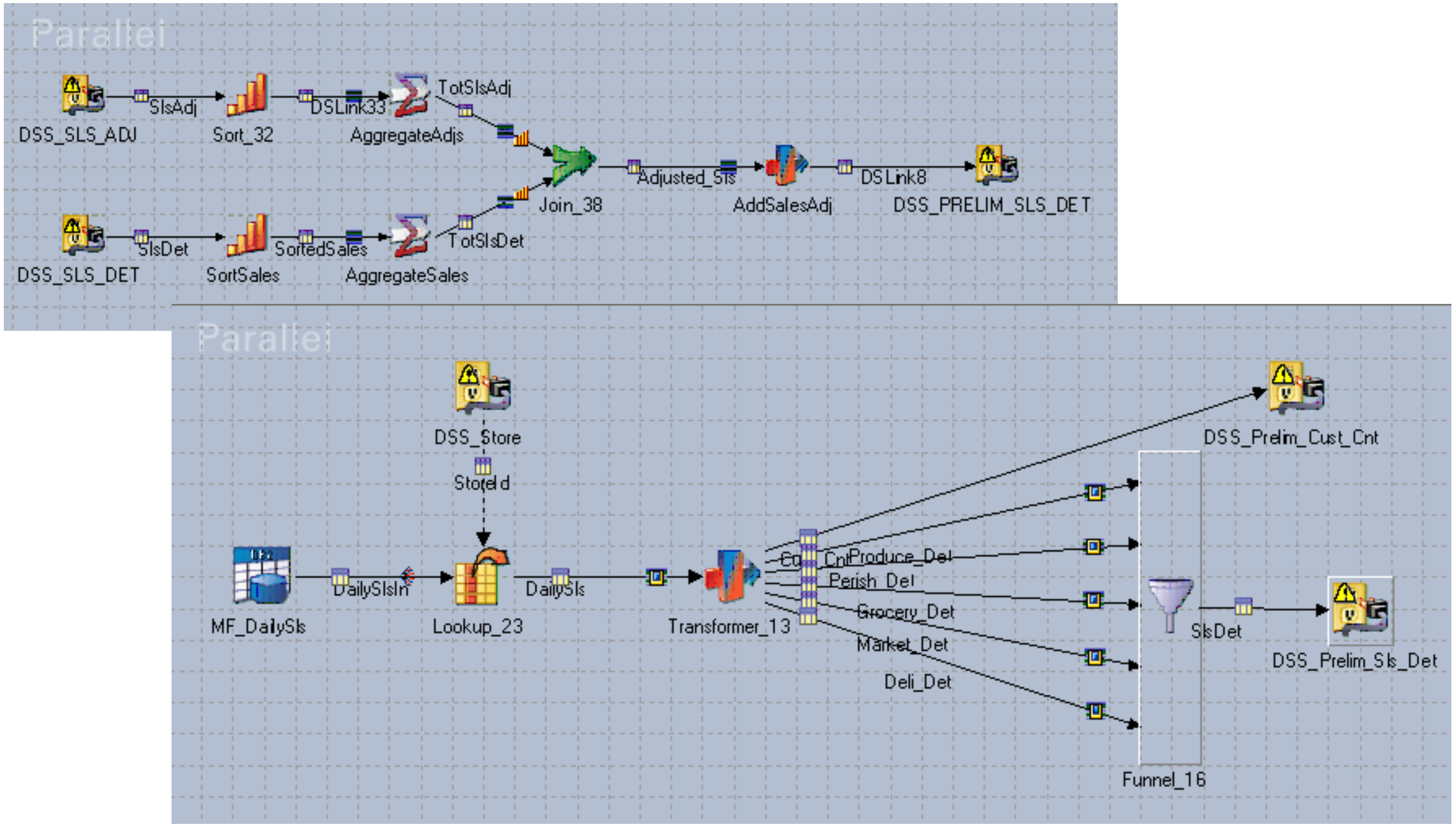
Monitoring and Change Data Capture

- Approaches
  - log-based
    - DBMS writes information about updates into its transaction log
    - Logs as analyzed to extract the change data
  - trigger-based
    - DB triggers (user-defined or internal) are used to gather change data
  - using replication middleware
    - may again use the approaches above
  - audit columns (application-based)
    - application records information about changes in an additional audit column per table
    - timestamp-based
      - intermediate changes may be lost
  - snapshot differentials
    - requires a full snapshot taken at previous extraction step
    - compares current state of data source with the snapshot
    - expensive, but may be the only option for legacy data sources
- Some approaches have limitations
  - example: audit columns don't support deletion, don't distinguish update and insert

Transformation

- Typical transformation operations
  - transformation into (de-)normalized warehouse schema
  - generation of global identifiers (surrogate keys)
    - keys from original sources may carry semantic information, may not be globally unique
  - data type conversion
  - data encoding (adjustments)
    - e.g., California → CA
  - standardization of character-based representations
    - e.g., <first name> <last name> (instead of <last name>, <first name>)
  - date/time, unit of measure standardization
  - combination/separation of attribute value(s)
  - derived values
  - aggregation
- Transformation Languages
  - (E)SQL
  - product-specific operators in data flow graphs

Sample ETL Processes (IBM DataStage)



Summary

- Replication middleware
  - usages
    - data distribution and consolidation
    - improve performance, availability
    - materialized information integration
  - architecture
    - capture and apply
    - committed change data
  - change propagation and ownership strategies
    - eager vs. lazy
    - group vs. master
  - conflict detection and reconciliation approaches are required for lazy group replication!

- Data Warehousing
  - materialized integration approach
    - avoids problems and restrictions of virtual integration architectures
  - integrated schema for mult-dimensional data analysis, OLAP
    - facts, dimensions, (hyper-)cubes
    - star and snowflake schema
  - architectures
    - central role of data warehouse manager
    - extract/transform/load (ETL) for data preparation
    - transformation implements schema and data integration logic
    - data quality requirements
  - potential problem: stale data
    - requires real-time data warehousing