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

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Chapter 10 - Data Replication and Materialized Integration

Motivation

- Replication: controlled redundancy of data to
  - improve performance (query response time)
  - increase availability
- 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
  - synchronous/asynchronous 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 drawback
  - potential for stale (out-of-date) data

Challenges

- Integration of replication with transaction processing
- Enforcing consistency of replica in the presence of updates
  - possible model: update of replicated data becomes a distributed transaction that updates all copies
- Detecting and resolving conflicts
  - reconciling "versions" of replicated data elements that have evolved independently (e.g., because of a communication failure)

Eager Replication

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

TA	ER1	ER2
writeA		
	writeA	
		writeA
writeB		
	writeB	
		writeB
writeC		
	writeC	
		writeC
commit		
	commit	
		commit

Lazy Replication

- Changes introduced at one site are propagated (as separate transactions) to other sites only after commit
  - 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  - stale (out-of-date) data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 can cause inconsistencies between the copies
    - concept for detecting inconsistencies
    - reconcile conflicting transactions
      - rollback of (committed) transactions not possible
  - ⇨ potential for "system delusion" (Gray, Reuter)
    - inconsistent database, with no obvious way to repair it

TA	LR1	LR2
writeA		
writeB		
writeC		
commit		
	writeA	
	writeB	
	writeC	
	commit	
		writeA
		writeB
		writeC
		commit

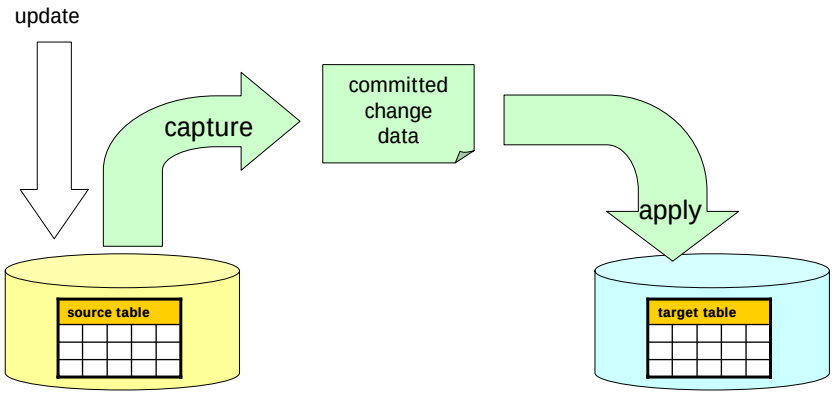
Replication Middleware

- 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

Change-Capture Replication



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
    - 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

Replication Conflicts

- 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

Ownership of Replicas

- Group
  - "update anywhere" update model
  - any node/site with a replica can update it
    - may cause conflicts, need for reconciliation
- Master
  - "primary copy" update model
  - each object has a master node
  - only the master can update the primary copy
    - other replicas are read-only
    - other nodes request the master node to perform an update (e.g., using RPC)
  - eliminates reconciliation failures with lazy replication
    - results in waits, potential deadlocks for updates initiated by non-master node
  - does not work for mobile, disconnected databases

Reconciliation

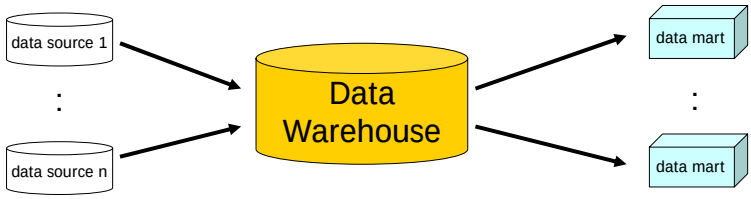
- Approaches
  - automatically, based on rules
    - site, time or value priority, merging of updates, ...
  - manually
    - conflict situations are reported in a conflicts table or queue
- Non-transactional replication schemes
  - abandon serializability for convergence
    - all nodes converge to 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

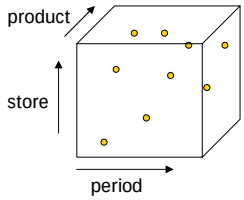
Data Warehousing

- 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



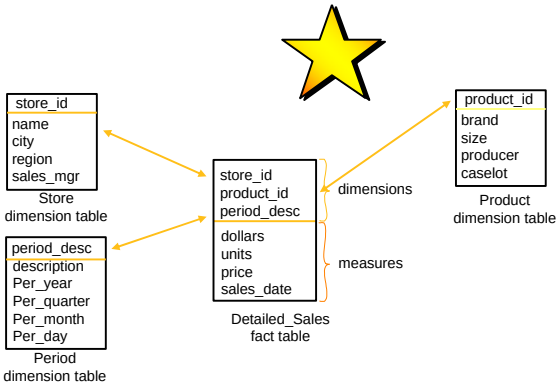
Multidimensional View of OLAP Data

- 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

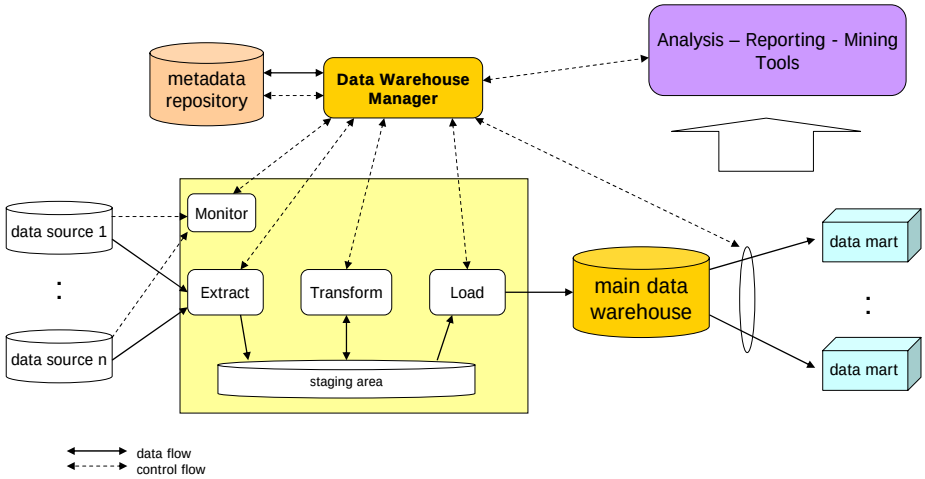


(Relational) OLAP Schema

- 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



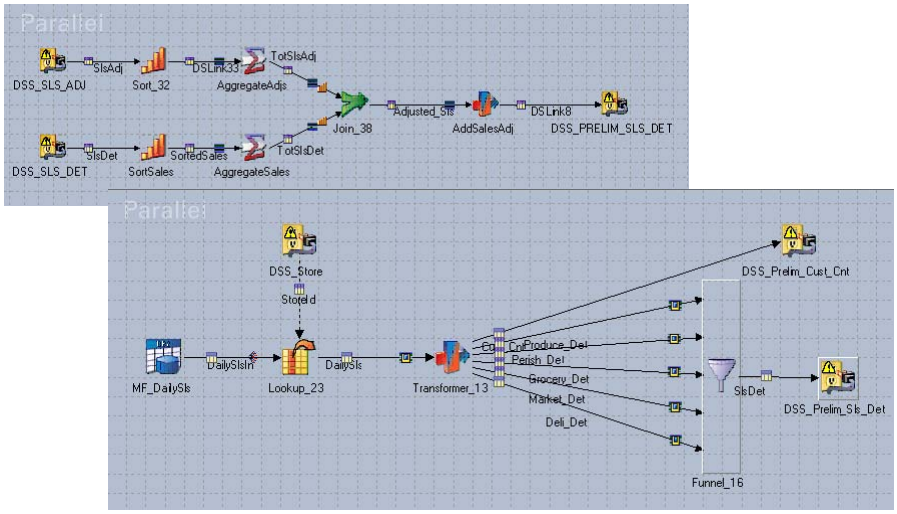
Data Warehouse Manager

- 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
    - replication-based (staging tables may be directly used by extractors)
    - timestamp-based
      - intermediate changes may be lost
  - **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!
      - 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

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