Near Real-Time Data Integration

Zero-Latency Data Warehousing

Robert M. Bruckner
bruckner@ifs.tuwien.ac.at

Institut für Softwaretechnik & Interaktive Systeme
Technische Universität Wien

Scientific Viewpoint: Zero-Latency DWH

- Support for various data freshness requirements
  - Minimize propagation delays for particular data

- Continuous data integration
  - Near real-time capturing and loading of data

- Timeliness of data / late-arriving data
  - Represent the history as accurately as possible
  - Deal with late-arriving data in tactical decision support

- Active decision engine
  - Notifications, event-condition-action

- High availability, scalability, and performance
Zero-Latency Data Warehouse Architecture

Information Validity / Overlapping Periods
Overlapping Validity Periods

Data Integration Approaches

- Batch load / update windows
- Continuous data integration enhanced by database triggers
- Near real-time data integration enhanced by integrated event-handling
Weaknesses Teradata-Prototype

- Data retrieval capability from the Teradata RDBMS via SELECT statements is not allowed.
- Arithmetic functions are not supported.
- Limited data transformations
- Limited to Teradata RDBMS

J2EE ETL Environment with ETLets

Diagram showing the J2EE ETL Environment with ETLets, including external data sources, JMX Managed Resources, J2EE Connectors, Resource Adapter, EJB Container, ETL Container, and Java Services.
J2EE-based Prototype

Purpose of the ETL Container

- Event cleansing
- Coordination of ETLet components for the processing/transformation
- Transaction-oriented processing of the events
- Persisting events and business metrics
- Calculation of business metrics (KPIs)
- Runtime (re-)configuration of ETLets
- Redeployment at runtime
Event Processing

ETLets → Metric Publisher → Metric Handler → Metric Evaluators

Binding between ETLets and Metric Evaluators are defined in the deployment descriptor
Application Scenario

![Diagram showing the process flow of Near Real-Time Data Integration]

Evaluation: Strengths (1)

- Support of complex transformations / processing
- Lightweight, near-realtime ETL processing (Threads)
- Flexible evaluation capabilities of (complex) calculated business metrics

- Usage of existing J2EE infrastructure
- Container managed, optimized, reusable ETL components
- Existing Java IDEs can be used for development
- Java exception handling
Evaluation: Strengths (2)

- Clean separation of
  - extraction logic (EventProcessor)
  - transformation logic (ETLets)
  - evaluation logic (MetricEvaluators)

- Event handling is (partly) separated from the data warehouse

- Event driven actions based on near real-time data integration

ETLContainer Opportunities

- Development methodology very similar to servlets
- ETL Java solution templates for specific DWH solutions (e.g. Process Warehouse)
- Seamless integration of operational Java Application Servers
- Existing Java tooling can be used/extended
- Reuse of ETL components (EventProcessors, ETLets, Evaluators) in multiple projects
  - Reuse of ETL components with different configurations
  - Extending ETL components by simple class inheritance
Active behavior will become an important mechanisms for data warehousing.

Typical update patterns and data propagation delays make tracing and monitoring of active decisions difficult.

Timely consistent representation is very important.

Zero-Latency Data Warehousing:
- Continuous / Near-Real Time Data Integration
- Temporal Consistency
- Active decision engine

Thank You!

Questions?