Live Data Integration

Fritz Laux  
Prof. emeritus  
Reutlingen University  
Dept. of Informatics  
Reutlingen, Germany

fritz.laux@reutlingen-university.de
My short CV

- **Education:** MSc (Diplom) and PhD (Dr. rer. nat.) in Mathematics
- **Working as** SW-analyst, designer and architect of commercial information systems for ZF, Porsche, Bosch/ Junkers, Telekurs, and Swiss PTT
- **Full Professor for Database and Information Systems at Reutlingen University. Dean of Studies. Supervised >200 Bachelor and Master students, and 3 Ph.D. students**
- **Cofounded DBTechNet (www.dbtechnet.org)**
- **Research activities in Database Modelling, Transaction Processing, Data Warehousing, and Data Mining.**
- **Research Award, IARIA fellow.**

Once upon a time …
~40 years ago
Outline

- **Motivation for Live Data Integration**
- **Requirements & Challenges**
- **Framework for Integration**
- **Live Data using Views and REST-Service**
- **Data Preparation** Precondition for Data Quality
- **Data Integration using the Typed Graph Model illustrated by Example**
- **ReadCheck for Transaction support**
Motivation for Live Data Integration

There is a need to integrate heterogeneous data sources to...

- gain added value (knowledge, insights) for decision support, predictive analysis, performance management, etc.
- coordinate complex processes in (near) real-time with transaction support (e.g. traffic control, industry 4.0, fight epidemic)
Need for Live Data Integration (Example)

**Covid-19 Pandemic Analysis and Control**

- Collected from 196 states under the International Health Regulations (IHR 2005)
  - Case numbers differ due to collection methods and actuality of sources
- National authorities like RKI (Germany), CDC (USA), DGS (Portugal), … collect the case data on county or city level, others collect on district level (e.g. SPF (France))
  - Data need to be adjusted to the same granularity for transnational analysis
- ETL is not sufficient because of periodic updates as can be seen from the Covid-19 data below for Portugal (all from the same day 18.09.20)

Live (permanent) data integration is necessary for up-to-date information and epidemic control.
Requirements and Challenges

❖ Combine data from heterogeneous sources
   ❗ Challenge: transform data to be compatible for integration

❖ Integrate latest data (even real-time data)
   ❗ Challenge: get data on the fly, increased network traffic

❖ Ensure high data quality
   ❗ Challenge: prepare and improve data quality

❖ Transaction support
   ❗ Challenge: distributed transactions for data management
Framework Architecture for Live Data Integration

Outline
Motivation
Framework
Live Data Preparation
Integration
TGM
Example
Transactions
Conclusion
References

Virtual integrated Database

Match & Merge & Map & Transform & Group

Apps requesting data, write through if permission is granted

Receives live data on request

mediated schema

REST access

Provided by data owner

Data preparation

Views \( V_1 \)
Views \( V_2 \)
Views \( V_{n-1} \)
Views \( V_n \)

WWW

Data source
Data source
Data source
Data source

...
Heterogeneous sources with different data structures need to be integrated.

**Issues**

- Find quality sources with up-to-date (live) data suitable for the application purpose
- Disclose (hidden) semantics of data require cooperation with the data owners
  - Data owners should provide mediated data views and semantic information for integration
- If two sources contain overlapping (redundant) data they usually do not match
  - Different granularity, actuality, semantics
  - Examples: Covid-19 Pandemic data vary between WHO, JHU, ECDC (European and national authorities (e.g. RKI, DGS) due to different collection and registration
Ensure that only latest data (even real-time data) is collected

Issues

- get data on the fly using mediated views on the live data
  - This needs cooperation with the source owner
- Reduce network traffic using ReadCheck [Crowe2017] validation and caching
- ReadCheck is a validator for freshness that checks if the requested data is (partially) in the cache and still up-to-date.
- ReadCheck combines ideas from Etag (Fielding and Reschke RFC 7232) and RVV [Laiho2010]
Model for Virtual Data Integration

Requester \( R \)  

Owners/Contractors \( C_i \)  

Global View \( V \)  

Views \( V_i \)  

Data Sources \( D_{ij} \)  

Figure taken from [Crowe2017, p.30] with modifications
To reduce network traffic and ensure freshness of data

(1) ReadCheck for a single row is the row-version validator
The RVVs for joined rows can be concatenated
For several specific rows these can be combined

(2) For data selected by range query need a table version
Compare time of latest change to the table with readCheck

(3) For data from multiple tables or databases, concatenate of build a vector of readCheck

(4) Data selected or aggregated from (3) is no fresher

Table $T_1$

<table>
<thead>
<tr>
<th>Key</th>
<th>Attributes</th>
<th>RVV</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>k</td>
<td>r(d)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table $T_2$

<table>
<thead>
<tr>
<th>Key</th>
<th>Attributes</th>
<th>RVV</th>
</tr>
</thead>
<tbody>
<tr>
<td>k_1</td>
<td>...</td>
<td>r(d_1)</td>
</tr>
<tr>
<td>k_2</td>
<td>...</td>
<td>r(d_2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

$Q = \bigcup_i Q_i$ on View $V$

$\bar{r}(Q) = X \bar{r}(Q_i)$

$\bar{r}(Q_1) = \left( \frac{r(d)}{r(T_2)} \right)$

$\text{res}(Q_1)$

Calculations for $Q_1$ on Views $V_1$

$C_i$ executes

Select … where key = k

result = row d

(1) RVV = r(d)

Select … where <predicate>

result = {all rows d, with <predicate>}

(2) $r(T_2) = r(d_j) | d_j \in T_2 \& r(d_j) \text{ last changed}$

$Q_i$ on Views $V_i$

$\text{res}(Q_i), \bar{r}(Q_i)$

$Q_1$ on Views $V_1$

$\text{res}(Q_1), \bar{r}(Q_1)$

$Q = \bigcup_i Q_i$ on View $V$

$\bar{r}(Q) = X \bar{r}(Q_i)$
Prepare data to meet highest quality for its purpose

Preparation steps [Sim2005] [Kemp2010] [Caf2009]

1. Select data
2. Adjust measurement units
3. Harmonize semantics
4. Group and classify
5. Correct and amend data

Steps 1 – 3 are mostly application neutral and should be realized by specific views
Steps 4 and 5 depend on the application and can be provided by the integration mapping

Issues

Requires semantic knowledge (coding, type, granularity, etc.) for all steps
Processing costs for step 4
Human decisions for step 5 required
Data Preparation Example

Step 1 (Covid-19 data from Web page)
- Identify and retrieve data

Step 2
- Get units and other meta information from HTML page for unit adjustments

Step 3
- Synonyms: apart from national language differences, the English names: cases, positive cases, reported cases, hospitalized, etc. could mean all the same or could mean different things.

Step 4
- Group patients according cost factors (ABC analysis)
- For time series bin data into equidistant intervals

Step 5
- Apparently incorrect: age < 0 or age > 130, interpolate missing data in time series.
**Match and map data with compatible semantics**

**Solution**

- The Typed Graph Model (TGM) [Laux2020] can help to identify, visualize, and map the data correctly.
- TGM is flexible to support various data structures and visualizes the integration process.
- TGM provides clear quality criteria for the data mapping. [Laux2017]

**Issues**

- The matching and mapping task is manual.
- Choosing the best quality (freshness, reliability, precision) data is the task of the integration schema designer.
Typed Graph Model (TGM)

Graph Model Schema

Graph Instance

Homomorphism $\Phi$ guarantees type and structural integrity of the graph instance
Graph Mapping Patterns

**Match** used in preparation steps 2 and 3

Given schema S and G. A 1:1- or renaming mapping is called a Match. The mapping preserves the semantics.

**Merge** used in preparation steps 1 - 4

Given mapping $M_{12}: S_1 \rightarrow S_2$, $G = S_1 \cup S_2$
Important Graph Mapping Types

**Isomorphism** (*Edge preserving injection*) used for steps 2 and 3

Given two graphs \( S=(V_1,E_1) \) and \( G=(V_2,E_2) \)

\[ f: (V_1) \rightarrow (V_2) \] is injection and

\[ \forall (v_1,v_2) \in E_1 \iff (f(v_1),f(v_2)) \in E_2 \]

**Homomorphism** (*Edge preserving map*) used for step 4

\[ f: (V_1) \rightarrow (V_2) \] is mapping and

\[ \forall (v_1,v_2) \in E_1 \Rightarrow (f(v_1),f(v_2)) \in E_2 \]
Commutative Mappings

A function chain is called commutative if and only if
\[ f_2 \circ f_1 = f_1 \circ f_2 \], i.e. \[ f_2(f_1(x)) = f_1(f_2(x)) \ \forall x \in \text{dom}(f_1) \].

Example: \( g \circ \text{id} = \text{id} \circ g \) (and more general \( \hat{g} \circ \text{iso}_1 = \text{iso}_2 \circ g \)).

For a consistent mapping from patient to rCode it is irrelevant if the projection g to region is done first or the isomorphic mapping \( \text{iso}_1 \) to patientCode.

Desirable Mappings

Projection \( \pi \), Homomorphism \( \text{hom} \), and Isomorphism \( \text{iso} \) are good candidates for commutative mappings.
(e.g. \( \pi \circ \text{iso} = \text{iso} \circ \pi \)).
Bipartite Graph

Let $G = (V,E)$ with $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$. If there are no edges within $V_1$ and $V_2$ then $G$ is bipartite.

Example 1:

Graph Matching quality criteria

Let $G$ be a bipartite Graph. A matching is a subset of edges where no two edges share an endpoint (node).

- Maximum matching = maximum number of vertices are matched
- Perfect matching = all vertices are matched (not merged)

Example 2:
### Theorem of Hall (Marriage Theorem)

Let $G = (V_1 \cup V_2, E)$ be a bipartite Graph. In $G$ exist a perfect matching if 
\[ \forall U_1 \subseteq V_1: d(U_1) \geq |U_1|. \]
\[ d(U_1) := |\{v \in V_2 | u \in U_1 \land (u,v) \in E\}| \]

General criteria for data integration coverage/completeness

### Example 3 (perfect match possible)

All subsets $U_1$ of $V_1$ have $d(U_1) \geq |U_1|$. 

(a1,a4), (a2,a5), (a3,a6) is a (the only) possible perfect matching.

### Example 4 (no perfect match possible)

Subset $U_1=\{a2, a3\}$ has $d(U_1)=|\{a5\}|=1$, but $|U_1| = 2$. 

#### Outline
- Motivation
- Framework
- Live Data Preparation
- Integration
  - TGM
  - Example
  - Transactions
- Conclusion
- References
Patient stats (semi-structured)

For privacy reasons the hospital agrees to provide only the following aggregated Patient statistics:
- region string
- numPatients int
- admissDate date
- Diagnosis text
- Treatment text

Admin office (hierarchical)

Population in hierarchically organized administrative areas:
- code | state | #inhabitants
- code | province | #persons
- code | district | #persons
- code | county | #persons
- code | quarter | #persons

Mediated Schema (relational)

ICD10_classifier
- Icd10_char(6)
- description text

Patient statistics
- regionCode char(8)
- patients int
- admissDate date
- ICD10 Code
- Treatment text

Population
- regionCode char(8)
- Area_name string
- #inhabitants int
Data Integration Example Solution (1st part)

- Identify matching schema metadata
- Merge diagnosis & description to lookup ICD10 classifier
- If node can be mapped by multiple path, check for commutative mappings
Do the same concat & merge with rural hierarchy

match 1:1, using concat code of area hierarchy

Do the same concat & merge with rural hierarchy

concat code hierarchy & merge

concat code of area hierarchy

Match with area hierarchy

Patients stats

includes

regionCode ID

Population includes

regionCode ID

Area_name

hospital patient stats

includes

region

treatment

diagnosis

admissDate

numPatients

Statistics office

areas hierarchy

related to

rural areas hierarchy

code, state, #inhabitants

code, district, #pers

code, province, #pers

code, city, #pers

code, county, #pers

code, quarter, #pers

urban areas hierarchy

code, district, #pers

code, province, #pers

code, city, #pers

code, county, #pers

hospital patient stats

includes

region

treatment

diagnosis

admissDate

numPatients

Statistics office

areas hierarchy

related to

rural areas hierarchy

code, state, #inhabitants

code, district, #pers

code, province, #pers

code, city, #pers

code, county, #pers

code, quarter, #pers

urban areas hierarchy

code, district, #pers

code, province, #pers

code, city, #pers

code, county, #pers

hospital patient stats

includes

region

treatment

diagnosis

admissDate

numPatients

Statistics office

areas hierarchy

related to

rural areas hierarchy

code, state, #inhabitants

code, district, #pers

code, province, #pers

code, city, #pers

code, county, #pers

code, quarter, #pers

principal area hierarchy

code, state, #inhabitants

code, district, #pers

code, province, #pers

code, city, #pers

code, county, #pers

code, quarter, #pers

principal area hierarchy
Some data require transactional guarantees for manual updates, corrections, and processing

These Apps need permission to write data

Solution

ReadCheck make distributed transactions feasible in loosely coupled environments

ReadCheck can be used for an Optimistic Concurrency Control (OCC) like Row Version Verifying (RVV) [Laiho2010]

Issues

Update and insert transaction must operate on the source data

only possible when data that is passed one-to-one to an App

It does not work on aggregated data
Lessons learned

⚠️ **Live Data Integration is possible with high quality and freshness if**

- Sources provide Live Views of data
- A mediated data schema is used for integration
- TGM helps to match, map, and merge data for integration
- Data is prepared in 5 steps for quality improvement

⚠️ **Key points for successful integration**

- Cooperation of data sources is necessary
- Careful semantic analysis and preparation of data is required to ensure quality data
- The integration process is iterative as most aspects are interwoven
References


[Laux2020] F. Laux, “The Typed Graph Model”, DBKDA 2020, Lisbon, Portugal,


