Jenga and the art of data-intensive ecosystems maintenance

Panos Vassiliadis

in collaboration with

G. Papastefanatos, P. Manousis, A. Simitsis, Y. Vassiliou

University of Ioannina, Greece
Software Evolution and ETL

• Software evolution causes at least as much as 60% of the costs for the entire software lifecycle

• ETL is not the exception:
  – Source database change their internal structure
  – Users require new structure and contents for their reports (and therefore for the collected DW data)
  – DBA and development teams do not synch well all the time
  – ...

Evolving data-intensive ecosystem

CREATE VIEW V_COURSE AS
SELECT S.S_ID, S.S_DESCR, CS.ID, CS.C_NAME, C.ID
FROM Semester S >- CourseStd CS >- Course C

CREATE VIEW V_TR AS
SELECT V.*, T.STUDENT_ID, T.GRADE
FROM V_Course V >- Transcript T

SELECT V1.STUDENT_ID, V1.C_NAME, V1.GRADE,
V2.C_NAME, V2.GRADE
FROM V_TR V1 >- V_TR V2 ON STUDENT_ID
WHERE V1.C_NAME='DBI'
AND V2.C_NAME='DBII'

SELECT V1.STUDENT_ID, S.STUDENT_NAME,
AVG(V.GRADE) AS GPA
FROM V_TR V >- STUDENT S ON STUDENT_ID
GROUP BY V1.STUDENT_ID, S.STUDENT_NAME
The impact can be syntactical (causing crashes), semantic (causing info loss or inconsistencies) and related to the performance.
The impact of changes & a wish-list

• **Syntactic**: scripts & reports simply crash
• **Semantic**: views and applications can become inconsistent or information losing
• **Performance**: can vary a lot

We would like: *evolution predictability*

i.e., control of *what will be affected*

*before* changes happen

- Learn what changes & how
- Find ways to quarantine effects
Research goals

• Part I: a case study for ETL evolution
  – Can we study ETL evolution and see in what ways do DW/ETL environments evolve?
  – Can we predict evolution in some way?
• Part II: regulating the evolution
  – Can we regulate the evolution?
  – Can we forbid unwanted changes?
  – Can we suggest adaptation to the code when the structure of data changes?
Zen and the Art of Motorcycle Maintenance, R. Pirsig

"Solve Problem: What is wrong with cycle?"

... By asking the right questions and choosing the right tests and drawing the right conclusions the mechanic works his way down the echelons of the motorcycle hierarchy until he has found the exact specific cause or causes of the engine failure, and then he changes them so that they no longer cause the failure...
Metrics for the Prediction of Evolution Impact in ETL Ecosystems: A Case Study

George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, Yannis Vassiliou

Journal on Data Semantics, August 2012, Volume 1, Issue 2, pp 75-97

Work conducted in the context of the "EICOS: foundations for perSOnalized Cooperative Information Ecosystems" project of the "Thales" Programme. The only source of funding for this research comes from the European Social Fund (ESF) -European Union (EU) and National Resources of the Greek State under the Operational Programme "Education and Lifelong Learning (EdLL).
Main goals of this effort

• We present a real-world case study of data warehouse evolution for exploring the behavior of a set of metrics that
  – monitor the vulnerability of warehouse modules to future changes and
  – assess the quality of various ETL designs with respect to their maintainability.

• We model the DW ecosystem as a graph and we employ a set of graph-theoretic metrics to see which ones fit best the series of actual evolution events

• We have used, Hecataeus, a publicly available, software tool, which allows us to monitor evolution and perform evolution scenarios in database-centric environments

Nothing is possible without a model

Architecture Graphs: the graph-based Model for data-Intensive ecosystems
Graph modeling of a data-intensive ecosystem

• The entire data-intensive ecosystem, comprising databases and their internals, as well as applications and their data-intensive parts, is modeled via a graph that we call Architecture Graph

• Why Graph modeling?
  – Completeness: graphs can model everything
  – Uniformity: we would like to module everything uniform manner
  – Detail and Grand-View: we would like to capture parts and dependencies at the very finest level; at same time, we would like to have the ability to zoom-out at higher levels of abstraction
  – Exploit graph management techniques and toolkits
CREATE TABLE EMP (EMP#  INTEGER  PRIMARY KEY,
NAME  VARCHAR(25) NOT NULL,
TITLE  VARCHAR(10),
SAL  INTEGER NOT NULL);
Q: SELECT EMP.Emp# as Emp#, 
    Sum(WORKS.Hours) as T_Hours
FROM EMP, WORKS
WHERE EMP.Emp# = WORKS.Emp#
    AND EMP.SAL > 50K
GROUP BY EMP.Emp#
CREATE VIEW V AS
SELECT Emp#, Hours
FROM EMP E, WORKS W
WHERE E.Emp# = W.Emp#
AND E.Sal >= 50K

SELECT Emp#, SUM(Hours) as T_HOURS
FROM V
GROUP BY Emp#
Zooming out to top-level nodes (modules)

CREATE VIEW V AS
SELECT Emp#, Hours
FROM EMP E, WORKS W
WHERE E.Emp# = W.Emp#
AND E.Sal >= 50K

SELECT Emp#, SUM(Hours) as T_HOURS
FROM V
GROUP BY Emp#
Can we relate graph-theoretic properties of nodes & modules to the probability of sustaining change?

Metrics
Node Degree

**Simple metrics:**
in-degree, out-degree, degree

**EMP.Emp#** is the most important attribute of **EMP.SAL**, if one considers how many nodes depend on it.

Edge direction: from dependant to depended upon
Transitive Node Degree

Observe that there is both a view and a query with nodes dependent upon attribute \(EMP.Emp\#\).

Transitive Metrics: in-degree, out-degree, degree
Strength: Zooming out to modules

A zoomed out graph highlights the dependence between modules (relations, queries, views), incorporating the detailed dependencies as the weight of the edges.

Again, for modules, we can have both:
- Simple **strength**
- Transitive **strength**
Node Entropy

The probability a node $v$ being affected by an evolution event on node $y_i$:

$$P(v|y_k) = \frac{\text{paths}(v, y_k)}{\sum_{y_i \in V} \text{paths}(v, y_i)}, \text{ for all nodes } y_i \in V.$$ 

**Examples**

- $P(Q/V) = 1/4,$
- $P(Q/EMP) = 2/4,$
- $P(V/WORKS) = 1/3$

**Entropy of a node $v$** : How sensitive the node $v$ is by an arbitrary event on the graph.

$$H(v) = -\sum_{y_i \in V} P(v | y_i) \log_2 P(v | y_i), \text{ for all nodes } y_i \in V.$$
A case study

Experimental Assessment
Context of the Study

• We have studied a data warehouse scenario from a Greek public sector’s data warehouse maintaining information for farming and agricultural statistics.

• The warehouse maintains statistical information collected from surveys, held once per year via questionnaires.

• Our study is based on the evolution of the source tables and their accompanying ETL flows, which has happened in the context of maintenance due to the change of requirements at the real world.

• Practically this is due to the update of the questionnaires from year to year
Internals of the monitored scenario

• The environment involves a set of 7 ETL workflows:
  – 7 source tables, (S1 to S7)
  – 3 lookup tables (L1 to L3),
  – 7 target tables, (T1 to T7), stored in the data warehouse.
  – 7 temporary tables (each target table has a temporary replica) for keeping data in the data staging area,
  – 58 ETL activities in total for all the 7 workflows.
PL/SQL to graph transformation

• All ETL scenarios were source coded as PL\SQL stored procedures in the data warehouse.
  – We extracted embedded SQL code (e.g., cursor definitions, DML statements, SQL queries) from activity stored procedures
  – Each activity was represented in our graph model as a view defined over the previous activities
  – Table definitions were represented as relation graphs.
Method of assessment

• We have represented the ETL workflows in our graph model
• We have recorded evolution events on the nodes of the source, lookup and temporary tables.
• We have applied each event sequentially on the graph and monitored the impact of the change towards the rest of the graph by recording the times that a node has been affected by each change
Macroscopic view

<table>
<thead>
<tr>
<th>Event Type</th>
<th># tables affected</th>
<th>Occurrences</th>
<th>pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Attribute</td>
<td>8</td>
<td>122</td>
<td>29%</td>
</tr>
<tr>
<td>Add Constraint</td>
<td>1</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>Drop Attribute Count</td>
<td>5</td>
<td>34</td>
<td>8%</td>
</tr>
<tr>
<td>Modify Attribute</td>
<td>9</td>
<td>16</td>
<td>4%</td>
</tr>
<tr>
<td>Rename Attribute</td>
<td>5</td>
<td>236</td>
<td>57%</td>
</tr>
<tr>
<td>Rename Table</td>
<td>7</td>
<td>7</td>
<td>2%</td>
</tr>
</tbody>
</table>

**ATTN: change of requirements at the real world determines pct breakdown!!**

Breakdown per event type

- Add Attribute: 29%
- Add Constraint: 0%
- Drop Attribute Count: 8%
- Modify Attribute: 4%
- Rename Attribute: 57%
- Rename Table: 2%
Workflow of the first ETL scenario, ETL1
Out – degree
- Schema size for tables
- Output schema size for activities
Pretty good job for tables
Decent job for filters and joins
Not so good for projection activities
Strength out did not work so well -- esp. for tables, it is too bad
Strength-total works the other way around
Workflows of the second & third ETL scenarios, ETL2 – ETL3
ETL 4
ETL4 -- Actual # affected nodes vs Graph Metrics

- Filter
- Join
- Project
Suddenly everything is underestimated

Pretty good job in the left part
ETL4 -- Actual # affected nodes vs Graph Metrics

Transitive metrics to the rescue
ETL4 -- Actual # affected nodes vs Graph Metrics

Entropy too
ETL 5,6,7
Lessons Learned
Schema size and module complexity as predictors for the vulnerability of a system

- The **size of the schemas** involved in an ETL design significantly affects the design vulnerability to evolution events.
  - For example, source or intermediate tables with many attributes are more vulnerable to changes at the attribute level.
  - The *out-degree* captures the projected attributes by an activity, whereas the *out-strength* captures the total number of dependencies between an activity and its sources.

- The **internal structure of an ETL activity** plays a significant role for the impact of evolution events on it.
  - Activities with high out-degree and out-strengths tend to be more vulnerable to evolution.
  - Activities performing *attribute reduction* (e.g., through either a group-by or a projection operation) are in general, less vulnerable to evolution events.
  - Transitive degree and entropy metrics capture the dependencies of a module with its various non-adjacent sources. Useful for activities which act as “hubs” of various different paths from sources in complex workflows.

- The **module-level design** of an ETL flow also affects the overall evolution impact on the flow.
  - For example, it might be worthy to place schema reduction activities early in an ETL flow to restrain the flooding of evolution events.
## Summary & Guidelines

<table>
<thead>
<tr>
<th>ETL Construct</th>
<th>Most suitable Metric</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Tables</td>
<td>out-degree</td>
<td>Retain small schema size</td>
</tr>
<tr>
<td>Intermediate &amp; Target Tables</td>
<td>out-degree</td>
<td>Retain small schema size in intermediate tables</td>
</tr>
<tr>
<td>Filtering activities</td>
<td>out-degree, out-strength</td>
<td>Retain small number of conditions</td>
</tr>
<tr>
<td>Join Activities</td>
<td>out-degree, out-strength, trans. out-degree, trans. out-strength, entropy</td>
<td>Move to early stages of the workflow</td>
</tr>
<tr>
<td>Project Activities</td>
<td>out-degree, out-strength, trans. out-degree, trans. out-strength, entropy</td>
<td>Move attribute reduction activities to early stages of the workflow and attribute increase activities to later stages</td>
</tr>
</tbody>
</table>
This was just a first step

• … we need many more studies to establish a firm knowledge of the mechanics of evolution
• … and we have not answered yet the core question:

*Are we helpless in managing evolution with predictability?*
Automating the adaptation of evolving data-intensive ecosystems

Petros Manousis, Panos Vassiliadis, and George Papastefanatos

Mainly based on the work of the MSc P. Manousis, currently under submission
Evolving data-intensive ecosystem

Which parts are affected, how exactly are they affected and, how can we intervene and predetermine their reaction?
Policies to predetermine reactions

Remove CS.C_NAME
Add exam year

Policies to predetermine the modules’ reaction to a hypothetical event?

DATABASE: ON ADD_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON ADD_CONDITION TO RELATION THEN PROPAGATE;
DATABASE: ON ADD_RELATION TO RELATION THEN PROPAGATE;
DATABASE: ON DELETE_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON DELETE_CONDITION TO RELATION THEN PROPAGATE;
DATABASE: ON DELETE_RELATION TO RELATION THEN PROPAGATE;
DATABASE: ON MODIFYDOMAIN_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON RENAME_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON MODIFY_CONDITION TO RELATION THEN PROPAGATE;
DATABASE: ON RENAME_RELATION TO RELATION THEN PROPAGATE;

DATABASE: ON ADD_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON ADD_CONDITION TO VIEW THEN BLOCK;
DATABASE: ON ADD_RELATION TO VIEW THEN BLOCK;
DATABASE: ON DELETE_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON DELETE_CONDITION TO VIEW THEN BLOCK;
DATABASE: ON DELETE_RELATION TO VIEW THEN BLOCK;
DATABASE: ON MODIFYDOMAIN_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON RENAME_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON MODIFY_CONDITION TO VIEW THEN BLOCK;
DATABASE: ON RENAME_RELATION TO VIEW THEN BLOCK;

DATABASE: ON ADD_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON ADD_CONDITION TO QUERY THEN BLOCK;
DATABASE: ON ADD_RELATION TO QUERY THEN BLOCK;
DATABASE: ON DELETE_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON DELETE_CONDITION TO QUERY THEN BLOCK;
DATABASE: ON DELETE_RELATION TO QUERY THEN BLOCK;
DATABASE: ON MODIFYDOMAIN_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON RENAME_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON MODIFY_CONDITION TO QUERY THEN BLOCK;
DATABASE: ON RENAME_RELATION TO QUERY THEN BLOCK;
Overview of solution

- **Architecture Graphs**: graph with the data flow between modules (i.e., relations, views or queries) at the detailed (attribute) level; module internals are also modeled as subgraphs of the Architecture Graph.

- **Policies**, that annotate a module with a reaction for each possible event that it can withstand, in one of two possible modes:
  - (a) *block*, to veto the event and demand that the module retains its previous structure and semantics, or,
  - (b) *propagate*, to allow the event and adapt the module to a new internal structure.

- **Given a potential change in the ecosystem**
  - we identify which parts of the ecosystem are affected via a “change propagation” algorithm
  - we rewrite the ecosystem to reflect the new version in the parts that are affected and do not veto the change via a rewriting algorithm
    - Within this task, we resolve conflicts (different modules dictate conflicting reactions) via a conflict resolution algorithm
Status Determination: who is affected and how
How do we guarantee that when a change occurs at the source nodes of the AG, this is correctly propagated to the end nodes of the graph?

- We notify exactly the nodes that should be notified
- The status of a node is determined independently of how messages arrive at the node
- Without infinite looping – i.e., termination
Method at a glance

1. Topologically sort the graph

2. Visit affected modules with its topological order and process its incoming messages for it.

3. Principle of locality: process locally the incoming messages and make sure that within each module
   – Affected internal nodes are appropriately highlighted
   – The reaction to the event is determined correctly
   – If the final status is not a veto, notify appropriately the next modules
Propagation mechanism

• Modules communicate with each other via a single means: the schema of a provider module notifies the input schema of a consumer module when this is necessary

• Two levels of propagation:
  • Graph level: At the module level, we need to determine the order and mechanism to visit each module
  • Intra-module level: within each module, we need to determine the order and mechanism to visit the module’s components and decide who is affected and how it reacts + notify consumers
Algorithm 2 Status determination algorithm

Input: A topologically sorted architecture graph summary $G_s(V_s, E_s)$ (output of algorithm 1),
a global queue $Q$ that facilitates the exchange of messages between modules

Output: A list of modules $Affected\ Modules \subseteq V_s$ that were affected by the event and acquire a status other than $NO\_STATUS$

1: function SetStatus(Module, Messages)
2:     Consumers Messages = ∅;
3:     for all Message $\in$ Messages do
4:         decide status of Module;
5:         put messages for Module's consumers in Consumers Messages;
6:     end for
7: end function

8: Begin
9:     for all node $\in G_s(V_s, E_s)$ do
10:         node.status = $NO\_STATUS$;
11:     end for
12:     while size($Q$) $>$ 0 do
13:         visit module (node) in head of $Q$;
14:         insert node in $Affected\ Modules$ list;
15:         get all messages, Messages, that refer to node;
16:         SetStatus(node, Messages);
17:         if node.status $==$ PROPAGATE then
18:             insert node.Consumers Messages to the Q;
19:         end if
20:     end while
21:     return $Affected\ Modules$;
22: End
Theoretical Guarantees

• At the inter-module level
  • Theorem 1 (termination). The message propagation at the inter-module level terminates.
  • Theorem 2 (unique status). Each module in the graph will assume a unique status once the message propagation terminates.
  • Theorem 3 (correctness). Messages are correctly propagated to the modules of the graph

• At the intra-module level
  • Theorem 4 (termination and correctness). The message propagation at the intra-module level terminates and each node assumes a status.
Path Check: handling policy conflicts
Conflicts: what they are and how to handle them

- View0 initiates a change
- View1 and View 2 accept the change
- Query2 rejects the change
- Query1 accepts the change

BEFORE

- The path to Query2 is left intact, so that it retains it semantics
- View1 and Query1 are adapted
- View0 and View2 are adapted too, however, we need two version for each: one to serve Query2 and another to serve View1 and Query1
Path Check algorithm

Algorithm 3 Path check algorithm

Input: A summary of an architecture graph $G_s(V_s, E_s)$, a list of modules Affected modules, that were affected by the event (output of algorithm 2)

Output: Annotation of the modules of Affected modules on the action needed to take, and specifically whether we have to make a new version of it, or, implement the change the user asked on the current version

1: function CheckModule(Module, Affected modules)
2:     if Module has been marked then
3:         return; ▷ notified by previous block path
4:     end if
5:     mark Module to keep current version and apply the change on a clone;
6:     for all New module ∈ Affected modules feeding Module do
7:         CheckModule(New module, Affected modules); ▷ notify path
8:     end for
9:     end function
10: Begin
11:     for all Module ∈ Affected modules do
12:         if Module.status == BLOCK then
13:             CheckModule(Module, Affected modules);
14:             mark Module not to change; ▷ blockers keep only current version
15:         end if
16:     end for
17:     End
Path Check

- If there exists any Block Module we travel in reverse the Architecture Graph from blocker node to initiator of change
- In each step we inform the Module to keep current version and produce a new one adapting to the change
- We inform the blocker node that it should not change at all.
Path Check

Relation R

View0

View1

View2

Query1

Query2
Query2 starts Path Check algorithm Searching which of his providers sent him the message and notify him that he does not want to change
Path Check

View2 is notified to keep current version for Query2 and produce new version for Query1
Path Check

View0 is notified
To keep current version for Query2 and
Produce new version for Query1
We make sure that Query2 will not change since it is the blocker
Rewriting: once we identified affected parts and resolved conflicts, how will the ecosystem look like?
Rewriting algorithm

Algorithm 4 Rewriting algorithm

Input: A list of modules \textit{Affected modules}, knowing the number of versions they have to retain (output of algorithm 3), initial messages of \textit{Affected modules}

Output: Architecture graph after the implementation of the change the user asked

1: Begin
2: \textbf{if} any of \textit{Affected modules} has status BLOCK then
3: \hspace{1em} \textbf{if} initial message started from Relation module type then
4: \hspace{2em} return ; \hspace{1em} \triangledown \text{Relations do not change at all}
5: \hspace{1em} \textbf{else}
6: \hspace{1em} \hspace{1em} \textbf{for all} Module $\in$ \textit{Affected modules} do
7: \hspace{2em} \hspace{1em} \textbf{if} Module needs only new version then
8: \hspace{3em} proceed with rewriting of Module;
9: \hspace{3em} connect Module to new providers; \hspace{1em} \triangledown \text{new version goes to new path}
10: \hspace{2em} \hspace{1em} \textbf{else}
11: \hspace{2em} \hspace{2em} clone Module; \hspace{1em} \triangledown \text{clone module, to keep both versions}
12: \hspace{2em} \hspace{2em} connect cloned Module to new providers; \hspace{1em} \triangledown \text{clone is the new version}
13: \hspace{2em} \hspace{2em} proceed with rewriting of cloned Module;
14: \hspace{2em} \hspace{1em} \textbf{end if}
15: \hspace{1em} \textbf{end for}
16: \hspace{1em} \textbf{end if}
17: \hspace{1em} \textbf{else}
18: \hspace{2em} \textbf{for all} Module $\in$ \textit{Affected modules} do
19: \hspace{3em} proceed with rewriting of Module \hspace{1em} \triangledown \text{no blocker node;}
20: \hspace{2em} \textbf{end for}
21: \hspace{1em} \textbf{end if}
22: End
Rewriting

• If there is no Block, we perform the rewriting.

• If there is Block
  • If the change initiator is a relation we stop further processing.
  • Otherwise:
    • We clone the Modules that are part of a block path and were informed by Path Check and we perform the rewrite on the clones
    • We perform the rewrite on the Module if it is not part of a block path.

• Within each module, all its internals are appropriately adjusted (attribute / selection conditions / ... additions and removals)
Rewriting

- Relation R
- Keep current & produce new
- View0
- View2
- Query2
- Keep only current

- Relation R
- View0
- View2
- Query2
- Query1

- View1
- View2
- Query2
Experiments and results
Experimental setup

• University database ecosystem (the one we used in previous slides, consisting of 5 relations, 2 views and 2 queries)
• TPC-DS ecosystem (consisting of 15 relations, 5 views and 27 queries) where we used two workloads of events
  – WL1 with changes mainly at tables
  – WL2 with changes mainly at views

• Policies used (for both ecosystems):
  – propagate all policy and
  – mixture policy (20% blockers)

• Measurements: effectiveness & cost
# Impact & adaptation assessment for TPC-DS

<table>
<thead>
<tr>
<th>Event:Node</th>
<th>AM</th>
<th>% AM</th>
<th>AI</th>
<th>% AI</th>
<th>NM</th>
<th>ERM</th>
<th>RM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS:WEB_SALES</td>
<td>0</td>
<td>100</td>
<td>6</td>
<td>99.59</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RS:CUSTOMER_DEMOGRAPHICS.CD_DEMO_SK</td>
<td>3</td>
<td>90.63</td>
<td>8</td>
<td>99.46</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RS:VIEW38.C_LAST_NAME</td>
<td>2</td>
<td>93.75</td>
<td>4</td>
<td>99.73</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>RS:CUSTOMER_TOTAL_RET.CTR_TOTAL_RETURN</td>
<td>2</td>
<td>93.94</td>
<td>4</td>
<td>99.73</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>RS:CUSTOMER_TOTAL_RETRN.CTR_TOTAL_RETURN</td>
<td>2</td>
<td>94.12</td>
<td>6</td>
<td>99.6</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>AS:VIEW38</td>
<td>2</td>
<td>94.29</td>
<td>2</td>
<td>99.87</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>AS:CUSTOMER_TOTAL_RET</td>
<td>3</td>
<td>91.67</td>
<td>3</td>
<td>99.81</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>AS:CUSTOMER_TOTAL_RETRN</td>
<td>3</td>
<td>91.89</td>
<td>3</td>
<td>99.81</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>AA:VIEW38</td>
<td>2</td>
<td>94.74</td>
<td>4</td>
<td>99.75</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>AA:Q18</td>
<td>1</td>
<td>97.44</td>
<td>1</td>
<td>99.94</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DS:Q18</td>
<td>1</td>
<td>97.44</td>
<td>3</td>
<td>99.82</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DS:CUSTOMER_DEMOGRAPHICS</td>
<td>3</td>
<td>92.11</td>
<td>34</td>
<td>97.9</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RS:ITEM</td>
<td>10</td>
<td>73.68</td>
<td>11</td>
<td>99.32</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RS:PROMOTION</td>
<td>2</td>
<td>94.74</td>
<td>3</td>
<td>99.81</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
## Impact & adaptation assessment

<table>
<thead>
<tr>
<th>University ecosystem propagate all</th>
<th>TPC-DS workload 1 propagate all</th>
<th>TPC-DS workload 2 propagate</th>
<th>TPC-DS workload 2 mixture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum AM</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Maximum AM</td>
<td>4</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td>Average AM</td>
<td>2.79</td>
<td>30.36</td>
<td>87.51</td>
</tr>
<tr>
<td>Minimum AI</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Maximum AI</td>
<td>12.14</td>
<td>91.47</td>
<td>12.63</td>
</tr>
<tr>
<td>Average AI</td>
<td>66.25</td>
<td>98.37</td>
<td>97.68</td>
</tr>
<tr>
<td>Adaptation assessment NM</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Adaptation assessment ERM</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Adaptation assessment RM</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

### University ecosystem mixture

| Minimum AM                        | 0                                | 0                           | 0                         |
| Maximum AM                        | 7                                | 100                         | 24                        |
| Average AM                        | 3.86                             | 28.01                       | 87.51                     |
| Minimum AI                        | 1                                | 22.58                       | 0                         |
| Maximum AI                        | 15.86                            | 94.44                       | 80                        |
| Average AI                        | 90.19                            | 94.44                       | 99.94                     |
| Adaptation assessment NM          | 0.21                             | 0                           | 0                         |
| Adaptation assessment ERM         | 1.64                             | 1                           | 25                        |
| Adaptation assessment RM          | 1.86                             | 1                           | 4                         |

### TPC-DS workload 1 mixture

| Minimum AM                        | 0                                | 21.21                       | 0                         |
| Maximum AM                        | 26                               | 100                         | 100                       |
| Average AM                        | 3.92                             | 88.22                       | 99.94                     |
| Minimum AI                        | 1                                | 97.9                        | 0                         |
| Maximum AI                        | 12.63                            | 99.95                       | 100                       |
| Average AI                        | 97.68                            | 99.55                       | 99.94                     |
| Adaptation assessment NM          | 0.13                             | 0                           | 1                         |
| Adaptation assessment ERM         | 1.02                             | 1                           | 4                         |
| Adaptation assessment RM          | 1.15                             | 1                           | 4                         |

### TPC-DS workload 2 mixture

| Minimum AM                        | 0                                | 73.68                       | 0                         |
| Maximum AM                        | 10                               | 100                         | 100                       |
| Average AM                        | 2.57                             | 91.86                       | 92.89                     |
| Minimum AI                        | 1                                | 97.9                        | 0                         |
| Maximum AI                        | 6.57                             | 99.95                       | 6.57                      |
| Average AI                        | 97.9                             | 99.58                       | 99.58                     |
| Adaptation assessment NM          | 0.5                              | 1                           | 0.5                       |
| Adaptation assessment ERM         | 1.64                             | 4                           | 1.64                      |
| Adaptation assessment RM          | 2.14                             | 4                           | 2.14                      |
Cost analysis

- The results of TPC-DS ecosystem in workload 1
- Path check nearly no cost at all, but in 20% blockers doubled its value

<table>
<thead>
<tr>
<th></th>
<th>Status Determination</th>
<th>Path Check</th>
<th>Rewriting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagate all</td>
<td>358161</td>
<td>4947</td>
<td>367071</td>
<td>730179</td>
</tr>
<tr>
<td>Mixture</td>
<td>327488</td>
<td>18340</td>
<td>341735</td>
<td>687563</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status Determination</th>
<th>Percentage Breakdown</th>
<th>Path Check</th>
<th>Rewriting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagate all</td>
<td>49%</td>
<td>1%</td>
<td>50%</td>
</tr>
<tr>
<td>Mixture</td>
<td>48%</td>
<td>2%</td>
<td>50%</td>
</tr>
</tbody>
</table>
Status Determination Cost

Blue line: time
Red line: affected nodes

Slightly slower time in mixture mode due to blockers.
Rewrite Cost

Blue line: time
Red line: affected nodes

Due to blockers and workload containing mostly relation changes, we have no rewrites in mixture mode in a set of events.
Rewrite time comparison

- Peaks of red are due to cloning of modules.
- Valleys of red are due to blockers at a relation related event.
Lessons Learned #1

• Users gain up to 90% of effort.

• Even in really cohesive environments, users gain at least 25% of effort.

• When all modules propagate changes, on average there are 3.5 modules that rewrite themselves.
Lessons Learned #2

• “Popular” modules need more time to process compared to unpopular ones.

• Module-cloning costs more than other tasks.

• But since the time is measured in nanoseconds this is not a big deal.
Wrapping things up
In a nutshell

- **Studying the evolution of ecosystems is important**
  - Not just the database; the surrounding applications too!
  - Case studies are important (and very rare!!)
  - Reducing unnecessary schema elements can help us reduce the impact of maintaining applications in the presence of changes

- **Managing the evolution of ecosystems is possible**
  - We need to model the ecosystem and annotate it with evolution management techniques that dictate its reaction to future events
  - We can highlight who is impacted and if there is a veto or not.
  - We can handle conflicts, suggest automated rewritings and guarantee correctness
  - We can do it fast and gain effort for all involved stakeholders
Selected readings

• Matteo Golfarelli, Stefano Rizzi: A Survey on Temporal Data Warehousing. International Journal of Data Warehousing and Mining, Volume 5, Number 1, 2009, p. 1-17


Some thoughts for future work

• Vision: come up with laws (i.e., recurring patterns) that govern the evolution of data-intensive ecosystems
  – More (a lot more) case studies needed!

• Visualization: graph modeling results in large graphs that are really hard to use interactively

• Coupling applications with the underlying databases (e.g., via plugging A.G. + policies inside db’s or other repositories)
  – Useful to avoid unexpected crashes
  – Not without problems (too much coupling can hurt)
  – Data warehouses pose a nice opportunity
Merci bien pour votre attention!

Commentaires, questions, ...?
Auxiliary slides
Detailed experimental results
Fig. 4 Results for degree metrics for ETL1

Fig. 5 Results for strength metrics for ETL1

Fig. 6 Results for transitive degrees and entropy metrics for ETL1

Fig. 7 Results for transitive strength metrics for ETL1
Modeling tools
How to regulate evolution of ecosystems

• **Impact Analysis**
  – We employ evolution events to model how data-intensive ecosystems change
  – We apply a hypothetical event and propagate it over the Architecture Graph to assess which modules are affected by it, and how (i.e., in which parts of their internal structure)
  – This way, we can visualize and measure the impact of a potential change to the entire ecosystem

• **Impact Regulation**
  – We employ evolution policies to pre-determine how modules should react to incoming events
  – Whenever a notification on an event “arrives” at a module, the module knows what to do: adapt to the incoming event, or block it and require to retain its previous structure and semantics
  – This Blocking restricts the flooding of events to the entire Architecture Graph and can allow developers “fix contracts” with the underlying database
Evolving data-centric ecosystem

Policies to \textit{predetermine the modules’ reaction to a hypothetical event}?
A language for policies

Policies to predetermine the modules’ reaction to a hypothetical event?

DATABASE: ON ADD_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON ADD_CONDITION TO RELATION THEN PROPAGATE;
DATABASE: ON ADD_RELATION TO RELATION THEN PROPAGATE;
DATABASE: ON DELETE_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON DELETE_RELATION TO RELATION THEN PROPAGATE;
DATABASE: ON MODIFYDOMAIN_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON RENAME_ATTRIBUTE TO RELATION THEN PROPAGATE;
DATABASE: ON MODIFY_CONDITION TO RELATION THEN PROPAGATE;
DATABASE: ON RENAME_RELATION TO RELATION THEN PROPAGATE;

DATABASE: ON ADD_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON ADD_CONDITION TO VIEW THEN BLOCK;
DATABASE: ON ADD_RELATION TO VIEW THEN BLOCK;
DATABASE: ON DELETE_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON DELETE_RELATION TO VIEW THEN BLOCK;
DATABASE: ON MODIFYDOMAIN_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON RENAME_ATTRIBUTE TO VIEW THEN BLOCK;
DATABASE: ON MODIFY_CONDITION TO VIEW THEN BLOCK;
DATABASE: ON RENAME_RELATION TO VIEW THEN BLOCK;

DATABASE: ON ADD_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON ADD_CONDITION TO QUERY THEN BLOCK;
DATABASE: ON ADD_RELATION TO QUERY THEN BLOCK;
DATABASE: ON DELETE_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON DELETE_RELATION TO QUERY THEN BLOCK;
DATABASE: ON MODIFYDOMAIN_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON RENAME_ATTRIBUTE TO QUERY THEN BLOCK;
DATABASE: ON MODIFY_CONDITION TO QUERY THEN BLOCK;
DATABASE: ON RENAME_RELATION TO QUERY THEN BLOCK;
A language for policies

- For all possible events, we define **rules**

```
DATABASE: ON <event> TO <module type> THEN <reaction policy>
```

- Module types: relations, views, queries

- Events:
  
  `{add, delete, rename} \times \{\text{module internals}\}`

- Policies:
  - **Propagate**: adapt to the incoming notification for change, willing to modify structure and semantics
  - **Block**: resist change; require to retain the previous structure and semantics
  - **Prompt**: indecisive for the moment, prompt the user at runtime (never implemented)
A language for policies

- Language requirements:
  - Completeness
  - Conciseness
  - Customizability

- We can override default policies, to allow module parts to differentiate their behavior from the default

<table>
<thead>
<tr>
<th>DATABASE:</th>
<th>ON</th>
<th>DELETE_ATTRIBUTE</th>
<th>TO</th>
<th>RELATION</th>
<th>THEN</th>
<th>PROPAGATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSCRIPT:</td>
<td>ON</td>
<td>DELETE_ATTRIBUTE</td>
<td></td>
<td></td>
<td>THEN</td>
<td>BLOCK</td>
</tr>
<tr>
<td>TRANSCRIPT.STUDENT_ID:</td>
<td>ON</td>
<td>DELETE_ATTRIBUTE</td>
<td></td>
<td></td>
<td>THEN</td>
<td>PROPAGATE</td>
</tr>
</tbody>
</table>

The two first req’s are covered by a complete set of default policies that have to be defined for the entire ecosystem.
For the metrics part

Other Useful: metrics
Evolution Variants

• Data Evolution: INS/DEL/UPD of data without affecting the structure of the db.

• Schema Evolution: the structure of the db changes, without loss of existing data, but without retaining historical information on previous snapshots of the db.

• Schema versioning: schema evolution + the ability to answer historical queries (practically being able to restore a snapshot of the db at a given timepoint).
Node Entropy

Entropy of a node $v$: How sensitive the node $v$ is by an arbitrary event on the graph.

$$H(v) = -\sum_{y_i \in V} P(v \mid y_i) \log_2 P(v \mid y_i), \text{ for all nodes } y_i \in V.$$

High values of entropy are assigned to the nodes of the graph with high level of dependence to other nodes, either directly or transitively.
Who is likely to undergo change?

• Schema size is (quite expectedly) the most important factor for a relation’s vulnerability to change

• The same holds for activities, too!
Most accurate predictors: \textit{out-degree} and \textit{out-strength}\hfill\hfill105
Internal structure of activities

• Activities with high *out-degree* and *out-strength* tend to be more vulnerable to evolution. The *out-degree* captures the projected attributes by an activity, whereas the *out-strength* captures the total number of dependencies between an activity and its sources.

• Activities with joins between many sources tend to be more affected than activities sourced by only one provider, but still, the most decisive factor seems to be the activity size.
  – Thus, activities that perform an attribute reduction on the workflow through either a group-by operation or a projection of a small number of attributes are in general, less vulnerable to evolution events and propagate the impact of evolution further away on the workflow (e.g., Q4 in ETL1 or Q2 – Q10 in ETL4).
  – In contrast, activities that perform join and selection operations on many sources and result in attribute preservation or generation on the workflow have a higher potential to be affected by evolution events (e.g., observe the activities ETL1_ACT10 - ETL1_ACT12 or the activity ETL4_ACT5).
Transitive degrees

• Transitive degree metrics capture the dependencies of a module with its various non-adjacent sources.

• Useful for activities, which act as “hubs” of various different paths from sources in complex workflows.

• For cases where the out-degree metrics do not provide a clear view of the evolution potential of two or more modules, the out-transitive degree and entropy metrics may offer a more adequate prediction (as for example ETL4_Q3 and ETL4_Q2).
Context and internals of evolution

• As already mentioned, source S1 stores the constant data of the surveys and did not change a lot. The rest of the source tables (S2-S7), on the other hand, sustained maintenance.

• The recorded changes in these tables mainly involve restructuring, additions and renaming of the questions comprising each survey, which are furthermore captured as changes in the source attributes names and types.

• The set of evolution events includes renaming of relations and attributes, deletion of attributes, modification of their domain, and lastly addition of primary key constraints. We have recorded a total number of 416 evolution events (see next table for a breakdown).

• The majority of evolution changes concerns attribute renaming and attribute additions.
## Some numbers

### ETL scenarios configuration

<table>
<thead>
<tr>
<th>Scenario</th>
<th># Activ.</th>
<th>Sources</th>
<th>Tmp Tables</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETL 1</td>
<td>16</td>
<td>L1,L2,L3,S1,S4</td>
<td>T1_Tmp, T2_Tmp, T3_Tmp</td>
<td>T1, T2, T3</td>
</tr>
<tr>
<td>ETL 2</td>
<td>6</td>
<td>L1,S2</td>
<td>T1_Tmp, T3_Tmp</td>
<td>T3</td>
</tr>
<tr>
<td>ETL 3</td>
<td>6</td>
<td>L1,S3</td>
<td>T1_Tmp, T3_Tmp</td>
<td>T3</td>
</tr>
<tr>
<td>ETL 4</td>
<td>15</td>
<td>L1,S4</td>
<td>T1_Tmp, T3_Tmp, T4_Tmp</td>
<td>T3, T4</td>
</tr>
<tr>
<td>ETL 5</td>
<td>5</td>
<td>S5</td>
<td>T1_Tmp, T5_Tmp</td>
<td>T5</td>
</tr>
<tr>
<td>ETL 6</td>
<td>5</td>
<td>S6</td>
<td>T1_Tmp, T6_Tmp</td>
<td>T6</td>
</tr>
<tr>
<td>ETL 7</td>
<td>5</td>
<td>S7</td>
<td>T1_Tmp, T7_Tmp</td>
<td>T7</td>
</tr>
<tr>
<td>Total</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Number of Attributes in ETL Source Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
</tr>
</thead>
<tbody>
<tr>
<td># Attributes</td>
<td>59</td>
<td>160</td>
<td>82</td>
<td>111</td>
<td>13</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Source</td>
<td>Change Type</td>
<td>Occurrence</td>
<td>Affected ETL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------------------</td>
<td>------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>Add Attribute</td>
<td>1</td>
<td>ETL 1, 2, 3, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>Add Constraint</td>
<td>1</td>
<td>ETL 1, 2, 3, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>Add Attribute</td>
<td>3</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>Add Attribute</td>
<td>1</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Add Attribute</td>
<td>14</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Drop Attribute</td>
<td>2</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Modify Attribute</td>
<td>3</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Rename Attribute</td>
<td>3</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>Add Attribute</td>
<td>15</td>
<td>ETL 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>Drop Attribute</td>
<td>4</td>
<td>ETL 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>Rename Attribute</td>
<td>121</td>
<td>ETL 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>Rename Attribute</td>
<td>80</td>
<td>ETL 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>Add Attribute</td>
<td>58</td>
<td>ETL 1, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>Drop Attribute</td>
<td>26</td>
<td>ETL 1, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>Modify Attribute</td>
<td>1</td>
<td>ETL 1, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>Rename Attribute</td>
<td>27</td>
<td>ETL 1, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 1, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>Modify Attribute</td>
<td>2</td>
<td>ETL 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7</td>
<td>Rename Attribute</td>
<td>5</td>
<td>ETL 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7</td>
<td>Rename Table</td>
<td>1</td>
<td>ETL 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>Drop Attribute</td>
<td>1</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>Modify Attribute</td>
<td>1</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1_tmp</td>
<td>Drop Attribute</td>
<td>1</td>
<td>ETL1-7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1_tmp</td>
<td>Modify Attribute</td>
<td>1</td>
<td>ETL1-7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>Add Attribute</td>
<td>15</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>Modify Attribute</td>
<td>2</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2_tmp</td>
<td>Add Attribute</td>
<td>15</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2_tmp</td>
<td>Modify Attribute</td>
<td>2</td>
<td>ETL 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>Modify Attribute</td>
<td>2</td>
<td>ETL 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5_tmp</td>
<td>Modify Attribute</td>
<td>2</td>
<td>ETL 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total 416

Distribution of events at the ETL tables
How to design a scenario

- When persistent data stores are involved, the generic guideline is to retain their schema as small as possible.
- Since the schema size affects a lot the propagation of evolution events, it is advisable to reduce schema sizes across the ETL flow, so activities that help in that direction should be considered first.
- Since attribute reduction activities (e.g., projections, group by queries) are less likely to be affected by evolution actions than other activities that retain or increase the number of attributes in the workflow (many projections with joins), the ETL designer should attempt placing the attribute reduction activities in the early stages of the workflow in order to restrain the flooding of evolution events.
Evolving data-intensive ecosystem
Evolving data-intensive ecosystem
Evolving data-intensive ecosystem

- Add exam year
- Remove CS.C_NAME
Evolving data-intensive ecosystem

- Add exam year
- Remove CS.C_NAME

View for Courses
CREATE VIEW V_COURSE AS
SELECT S.S_ID, S.S_DESCR, CS.ID, CS.C_NAME, C.ID
FROM Semester S INNER JOIN CourseStd CS ON S.S_ID = CS.S_ID
INNER JOIN Course C ON CS.ID = C.ID

Report on DBI, DBII Grades
FROM V_TR V1 INNER JOIN V_TR V2 ON V1.STUDENT_ID = V2.STUDENT_ID
WHERE V1.C_NAME = 'DBI'
AND V2.C_NAME = 'DBII'

View for Student Transcripts
CREATE VIEW V_TR AS
SELECT V.*, T.STUDENT_ID, T.GRADE
FROM V_Course V INNER JOIN Transcript T ON V.COURSE_ID = T.COURSE_ID

Report on Average Grade
SELECT V.STUDENT_ID, S.STUDENT_NAME,
AVG(V.GRADE) AS GPA
FROM V_TR V INNER JOIN STUDENT S ON V.STUDENT_ID = S.ID
GROUP BY V.STUDENT_ID, S.STUDENT_NAME

Which parts are affected ... ?
Evolving data-intensive ecosystem

Which parts are affected, and how exactly are they affected ...?
For the regulation part

**Other Useful: Regulation**
Problem definition

• Changes on a database schema may cause syntactic or semantic inconsistency in its surrounding applications; is there a way to regulate the evolution of the database in a way that application needs are taken into account?

• If there are conflicts between the applications’ needs on the acceptance or rejection of a change in the database, is there a possibility of satisfying all the different constraints?

• If conflicts are eventually resolved and, for every affected module we know whether to accept or reject a change, how can we rewrite the ecosystem to reflect the new status?
Observe the input and output schemata!!

```
SELECT V.STUDENT_ID, S.STUDENT_NAME, AVG(V.TGRADE) AS GPA
FROM V_TR V |><| STUDENT S ON STUDENT_ID
WHERE V.TGRADE > 4 / 10
GROUP BY V.STUDENT_ID, S.STUDENT_NAME
```
Annotation with Policies

On attribute addition

Then propagate

On attribute deletion

Then block

Status Determination
Module Level Propagation

Add Exam Year
Module Level Propagation
Module Level Propagation

Add Exam Year

Student
- SID
- SName

Course
- CID
- CSID
- CSName
- CSPt

Course Std
- CID
- CSID
- CSName
- CSPt

Semester
- SM
- MID
- MDescr

V_Course
- MID
- MDescr
- CSID
- CSName
- CID

Outs
- MID
- MDescr
- CSID
- CSName
- CID

AND

Module Level Propagation
Message initiation

• The Message is initiated in one of the following schemata:
  – Output schema and its attributes if the user wants to change the output of a module (add / delete / rename attribute).
  – Semantics schema if the user wants to change the semantics tree of the module.
Intra-module processing

• When a Message arrives at a module via the propagation mechanism, these steps describe the module's way of handling it:

1) Input schema and its attributes if applicable, are probed.
2) If the parameter of the Message has any kind of connection with the semantics tree, then the Semantics schema is probed.
3) Likewise if the parameter of the Message has any kind of connection with the output schema, then the Output schema and its attributes (if applicable) is probed.

• Finally, Messages are produced within the module for its consumers.