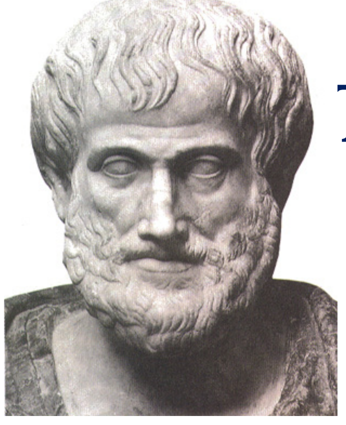
Schema Evolution for Relational Databases

Panos Vassiliadis

joint work with: Apostolos Zarras, Ioannis Skoulis, Petros Manousis, Fanis Giahos, Michael Kolozoff, Athanasios Pappas, Maria Zerva

Department of Computer Science and Engineering
University of Ioannina, Hellas

http://www.cs.uoi.gr/~pvassil/projects/schemaBiographies/



The nature that needs change is vicious; for it is not simple nor good…

Nicomachean Ethics, Book VII, Aristotle

SWEBOK Maintenance

	Correction	Enhancement
Proactive	Preventive	**Perfective**
Reactive	**Corrective**	Adaptive

- **Corrective** maintenance: reactive modification (or repairs) of a software product performed after delivery to **correct discovered problems**.
- **Adaptive** maintenance: modification of a software product performed after delivery **to keep a software product usable in a changed or changing environment.**
- **Perfective** maintenance: modification of a software product after delivery to provide enhancements for users, improvement of program documentation, and recoding **to improve software performance, maintainability, or other software attributes.**
- Preventive maintenance: modification of a software product after delivery to detect and correct latent faults in the software product before they become operational faults.

Database Evolution: why and what

- All software systems and, thus, both the databases themselves and applications built around databases are dynamic environments and can evolve due
  - Changes of requirements
  - Internal restructuring due to performance reasons
  - migration to / integration with another system
  - …
- Database evolution further concerns
  - changes in the **operational environment** of the database
  - changes in the content (**data**) of the databases as time passes by
  - changes in the internal structure, or **schema**, of the database

What evolves in DBMS...

- Data

EMP_ID	SALARY
100	1500

→

EMP_ID	SALARY
100	1650

```
UPDATE EMP
SET SALARY = SALARY *1.10
WHERE...
```

- Metadata – Schemata – Models

```
ALTER TABLE EMP
ADD COLUMN PHONE VARCHAR ...
```

EMP_ID	SALARY
100	1500

→

EMP_ID	SALARY	PHONE
100	1500	210777777

Why is (schema) evolution so important?

- Software and DB **maintenance** makes up for **at least 50% of all resources spent in a project**.
- Changes are more frequent than you think
- **Databases are rarely stand-alone: typically, an entire ecosystem of applications is structured around them =>**
- **Changes in the schema can impact a large (typically, not traced) number of surrounding app's, without explicit identification of the impact**

Embedded queries in the past [Maule+08] …

```
public static IEnumerable<Experiment> Q1(DateTime d){
 DBParams dbParams = new DBParams();
 DBRecordSet queryResult;
 List<Experiment> exps = new List<Experiment>();

 dbParams.Add("@ExpDate", d);

 queryResult = QueryRunner.Run(
  "SELECT Experiments.Name,Experiments.ExperimentId"+
  " FROM Experiments"+
  " WHERE Experiments.Date={@ExpDate}",
  dbParams);

 while (queryResult.MoveNext()){
  exps.Add(new Experiment(queryResult.Record));
 }

 return exps;
}
```

... nowadays, to be complemented with API-based db access (Drupal)

```
function _profile_get_fields($category, $register = FALSE) {
  $query = db_select('profile_field');
  if ($register) {
   $query->condition('register', 1);
  }
  else {
   $query->condition('category', db_like($category), 'LIKE');
  }
  if (!user_access('administer users')) {
   $query->condition('visibility', PROFILE_HIDDEN, '<>');
  }
  return $query
    ->fields('profile_field')
    ->orderBy('category', 'ASC')
    ->orderBy('weight', 'ASC')
    ->execute();
}
```

Evolution taxonomy

- **Schema evolution**, itself, can be addressed at
  - the conceptual level (req's, goals, conc. models, …. evolve)
  - the logical level, where the main constructs of the database structure evolve
    - E.g.,: relations and views in the relational area, classes in the object-oriented database area, or (XML) elements in the XML/semi-structured area),
  - the physical level, involving data placement and partitioning, indexing, compression, archiving etc.

Evolution taxonomy: areas

- Relational databases
- Object Oriented db’s
- Conceptual models
- XML
- Ontologies
- …

- Special case of relational: data warehouses

… To probe further …

- Michael Hartung, James F. Terwilliger, Erhard Rahm: Recent Advances in Schema and Ontology Evolution. In Schema Matching and Mapping (Zohra Bellahsene, Angela Bonifati, Erhard Rahm), 149-190, Springer 2011, ISBN 978-3-642-16517-7

- Matteo Golfarelli, Stefano Rizzi: A Survey on Temporal Data Warehousing. IJDWM 5(1): 1-17 (2009)
- Robert Wrembel: A Survey of Managing the Evolution of Data Warehouses. IJDWM 5(2): 24-56 (2009)

Imagine if we could predict how a schema will evolve over time…

- … we would be able to **“design for evolution”** and **minimize the impact of evolution** to the surrounding applications
  - by applying design patterns
  - by avoiding anti-patterns & complexity increase

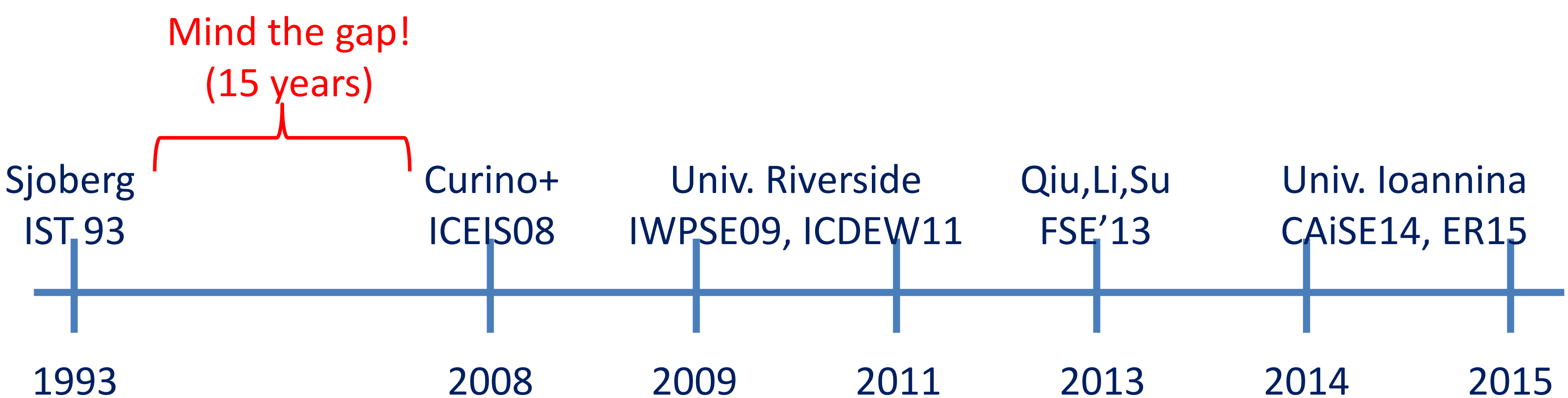
  … **in both the db and the code**
- … we would be able to **plan** administration and perfective maintenance tasks and resources, instead of responding to emergencies

WHAT ARE THE "LAWS" OF DATABASE SCHEMA EVOLUTION?

Why aren't we there yet?

- Historically, nobody from the research community had access + the right to publish to version histories of database schemata

- Open source tools internally hosting databases have changed this landscape &
- We are now presented with the opportunity to study the version histories of such "open source databases"



Our take on the problem

- Collected version histories for the schemata of 8 open-source projects
  - CMS's: MediaWiki, TYPO3, Coppermine, phpBB, OpenCart
  - Physics: ATLAS Trigger --- Bio: Ensemble, BioSQL
- Preprocessed them to be parsable by our **HECATE schema comparison tool** and exported the transitions between each two subsequent versions and measures for them (size, growth, changes)
- Exploratory search where we statistically studied / mined these measures, to **extract patterns & regularities for the lives of tables**
- **Web:**
  http://www.cs.uoi.gr/~pvassil/projects/schemaBiographies/
- **Data available at:**
  https://github.com/DAINTINESS-Group/EvolutionDatasets

Scope of the study

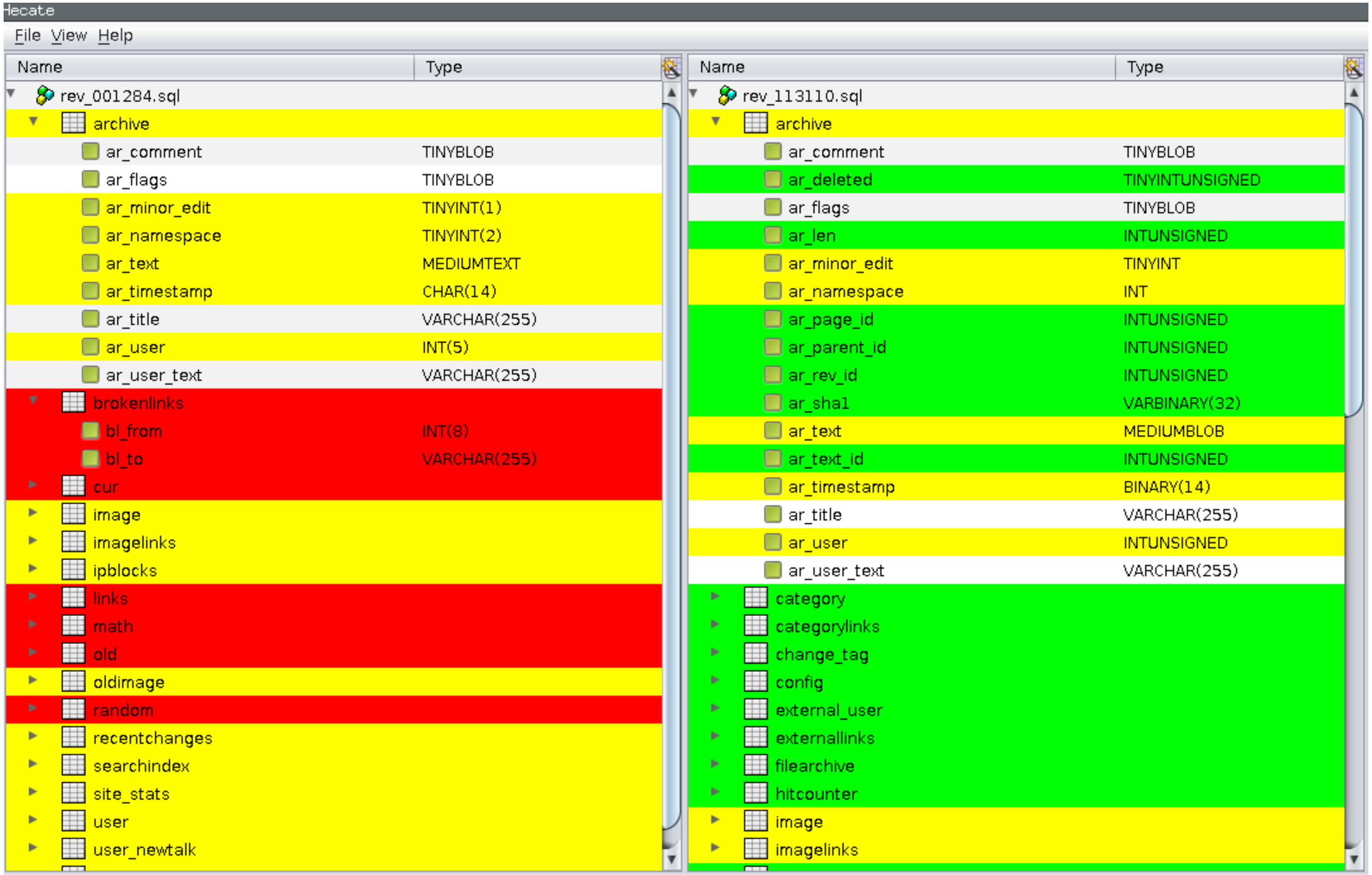
- **Scope**:
  - databases being part of open-source software (and not proprietary ones)
  - long history
  - we work only with changes at the logical schema level (and ignore physical-level changes like index creation or change of storage engine)

- We encompass datasets with different domains ([A]: physics, [B]: biomedical, [C]: CMS's), amount of growth (shade: high, med, low) & schema size

- We should be very careful to not overgeneralize findings to proprietary databases or physical schemata!

FoSS Dataset	Versions	Lifetime	Tables @ Start	Tables @ End
ATLAS Trigger [A]	84	2 Y, 7 M, 2 D	56	73
BioSQL [B]	46	10 Y, 6 M, 19 D	21	28
Coppermine [C]	117	8 Y, 6 M, 2 D	8	22
Ensembl [B]	528	13 Y, 3 M, 15 D	17	75
MediaWiki [C]	322	8 Y, 10 M, 6 D	17	50
OpenCart [C]	164	4 Y, 4 M, 3 D	46	114
phpBB [C]	133	6 Y, 7 M, 10 D	61	65
TYPO3 [C]	97	8 Y, 11 M, 0 D	10	23

Hecate: SQL schema diff extractor



https://github.com/DAINTINESS-Group/Hecate

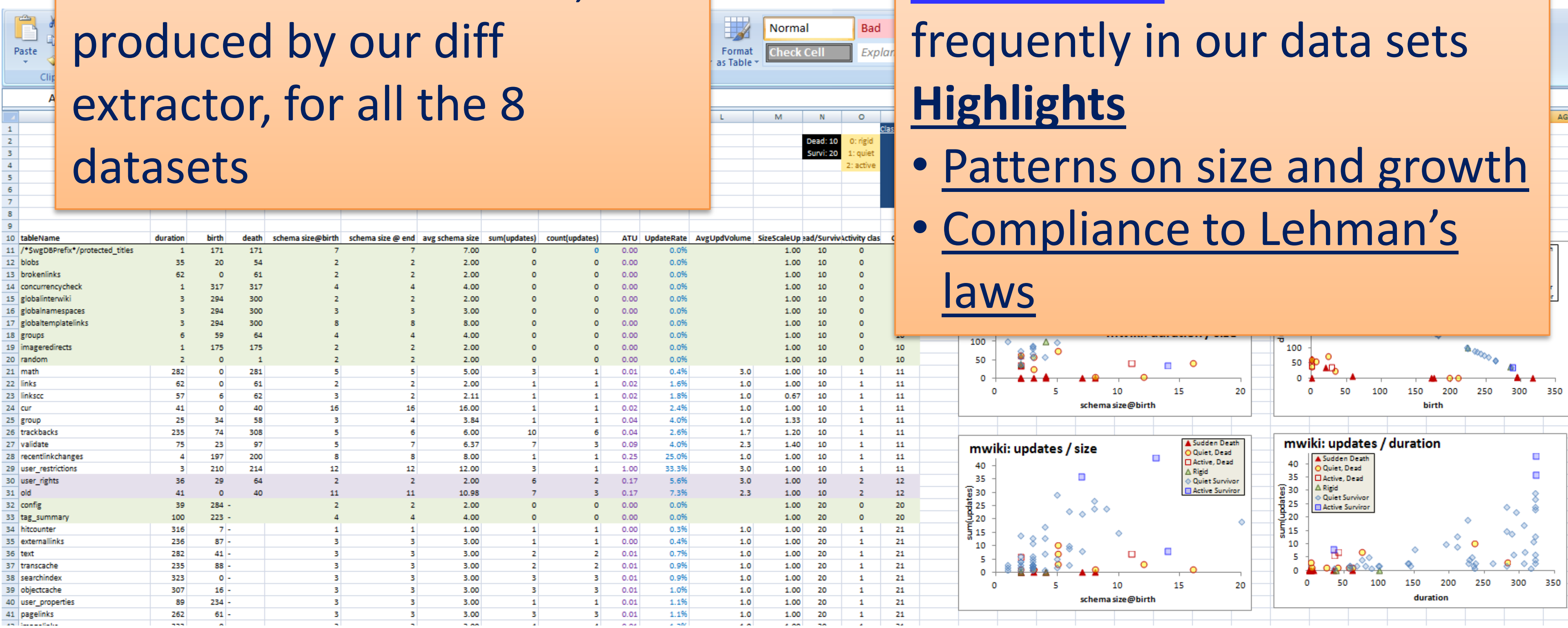
.. What do we see if we observe the evolution of the entire schema?

http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

- Skoulis, Vassiliadis, Zarras. Open-Source Databases: Within, Outside, or Beyond Lehman's Laws of Software Evolution? **CAiSE 2014**
- Growing up with stability: How open-source relational databases evolve. **Information Systems, Volume 53**, **October–November 2015**

SCHEMA EVOLUTION FOR O/S DB'S AT THE "MACRO" LEVEL

Exploratory search of the schema histories for patterns



Input: schema histories from github/sourceforge/…

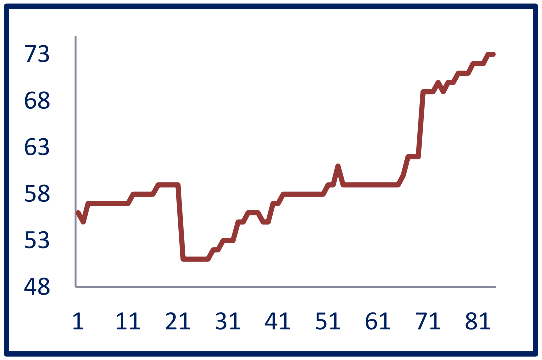
Raw material: details and stats on each table's life, as produced by our diff extractor, for all the 8 datasets

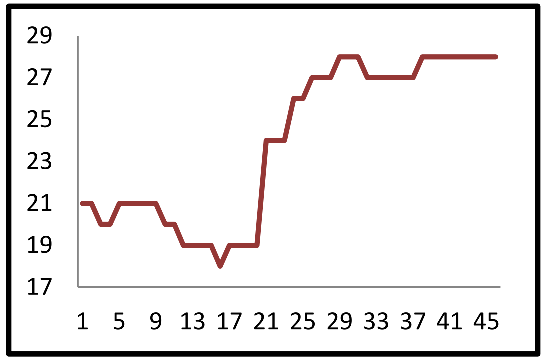
Output: properties & patterns on **schema properties (size, growth, changes, …)** that occur frequently in our data sets

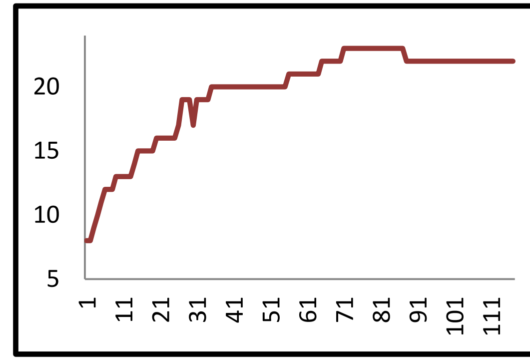
Highlights

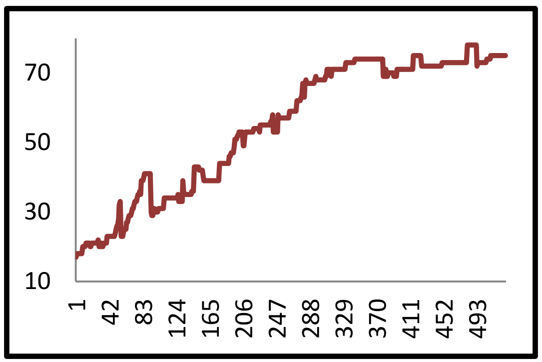
- Patterns on size and growth
- Compliance to Lehman's laws

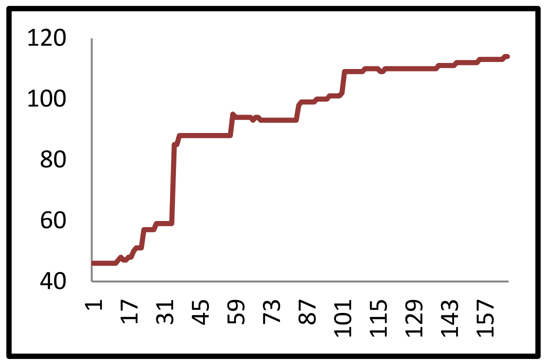
Schema Size (relations)

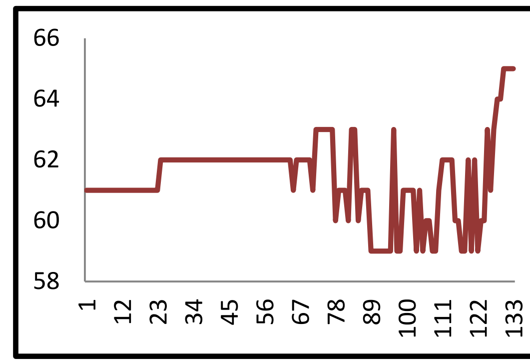


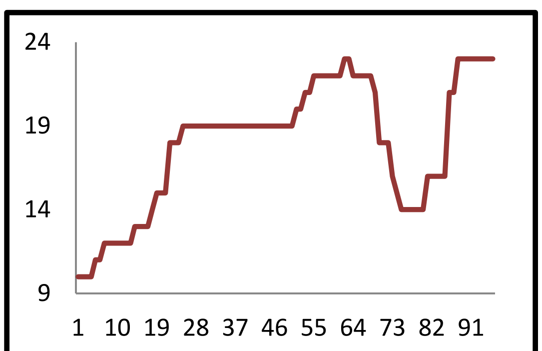


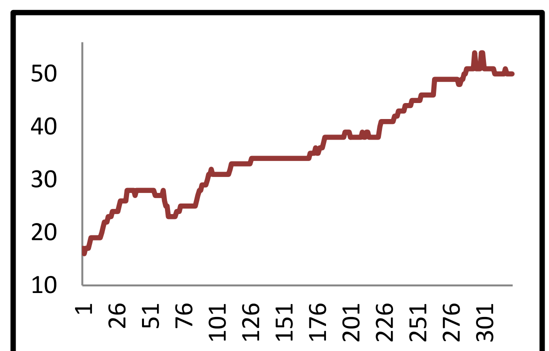






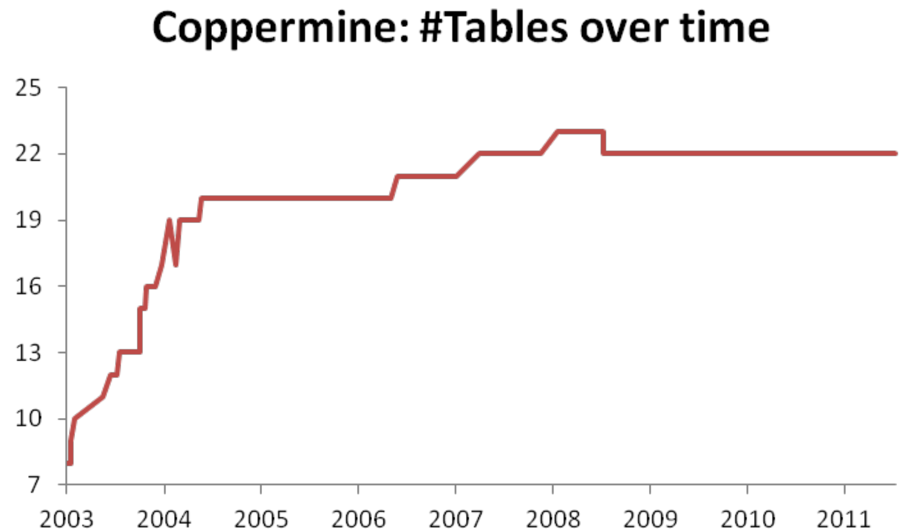




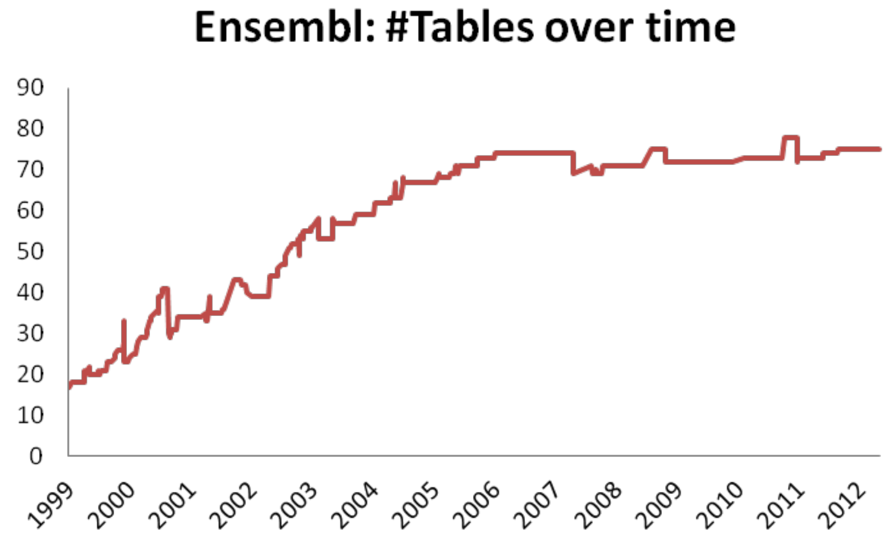


Schema Size

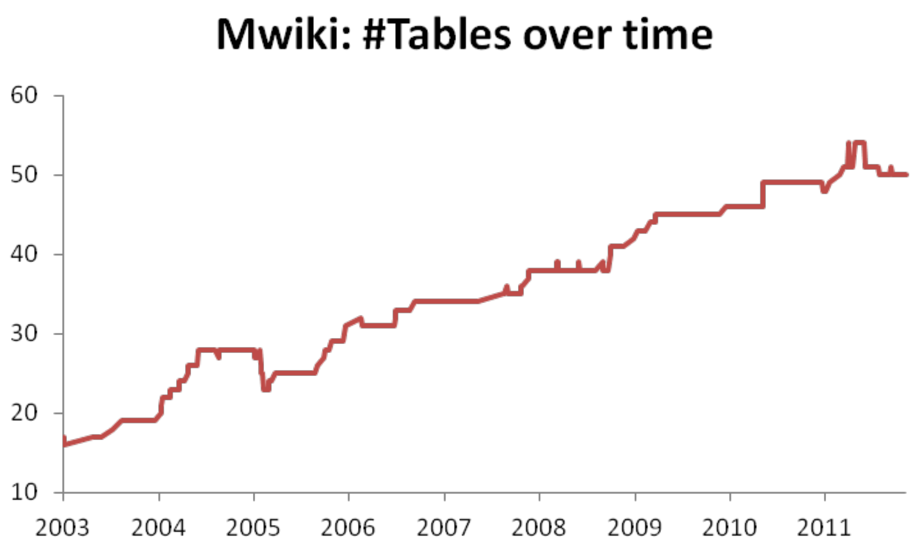
- **Overall increase in size**
- Periods of **increase**, esp. at beginning and after large drops -> positive feedback
- **Drops**: sudden and steep (in short duration) -> negative feedback
- **Large periods of stability!**
  - Unlike traditional S/W, db's are dependency magnets…



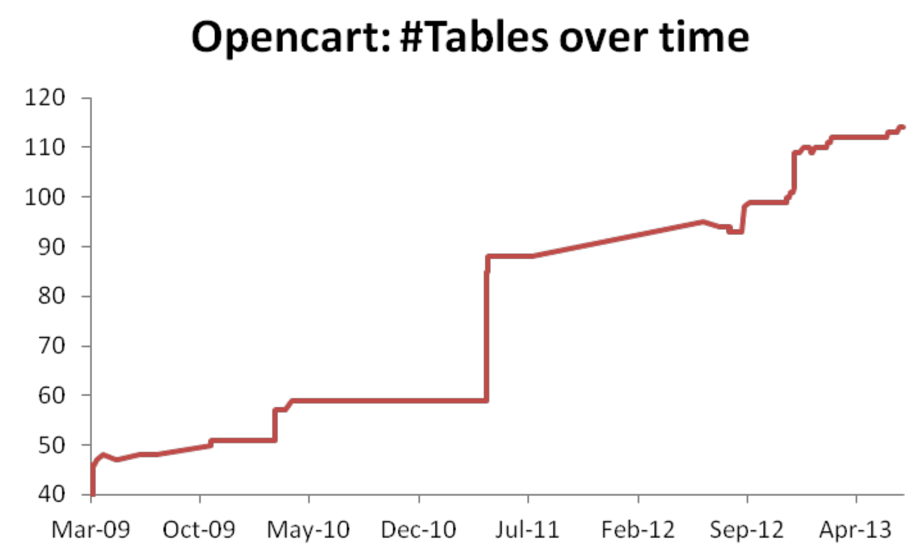
Coppermine: #Tables over time
25
22
19
16
13
10
7
2003
2004
2005
2006
2007
2008
2009
2010
2011



Ensembl: #Tables over time
90
80
70
60
50
40
30
20
10
0
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012

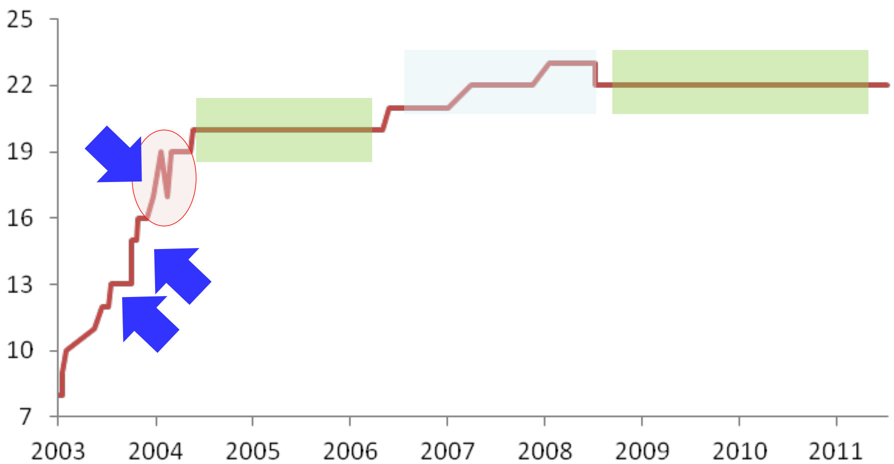


Mwiki: #Tables over time
60
50
40
30
20
10
2003
2004
2005
2006
2007
2008
2009
2010
2011

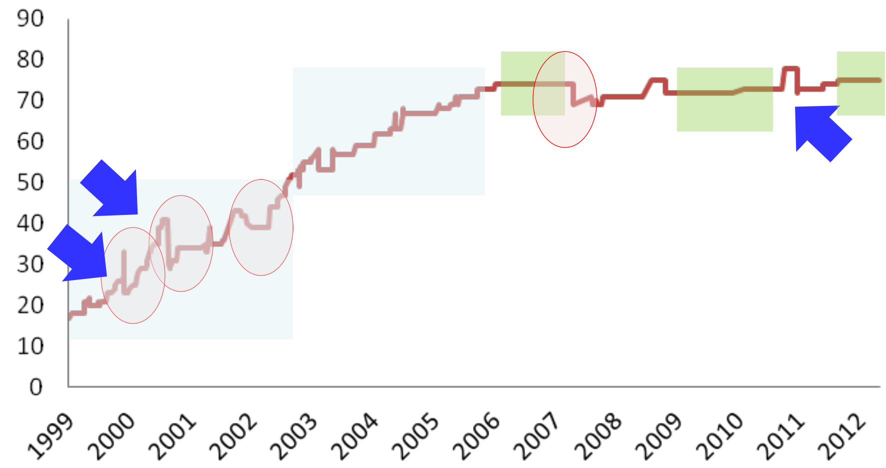


Opencart: #Tables over time
120
110
100
90
80
70
60
50
40
Mar-09
Oct-09
May-10
Dec-10
Jul-11
Feb-12
Sep-12
Apr-13

Coppermine: #Tables over time



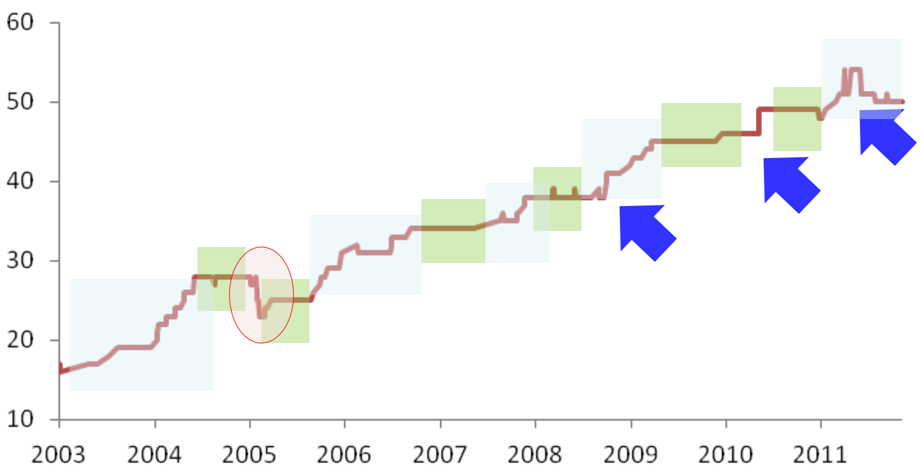
Ensembl: #Tables over time



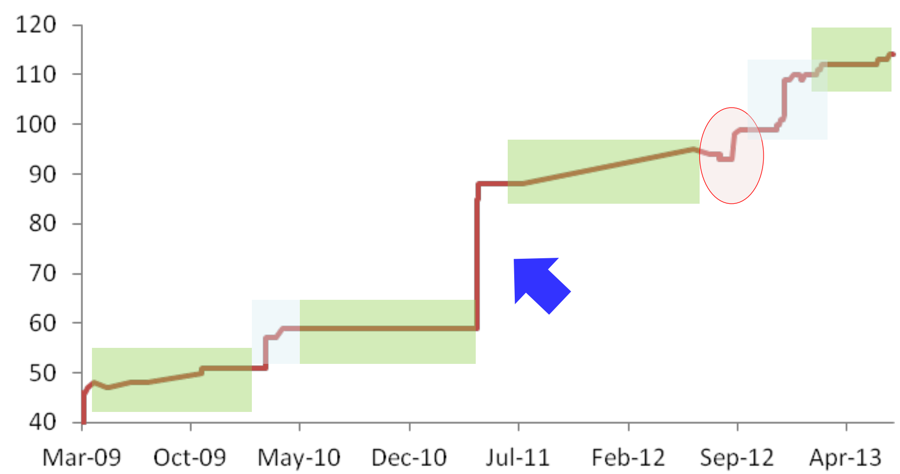
Growth over time
Calmness periods

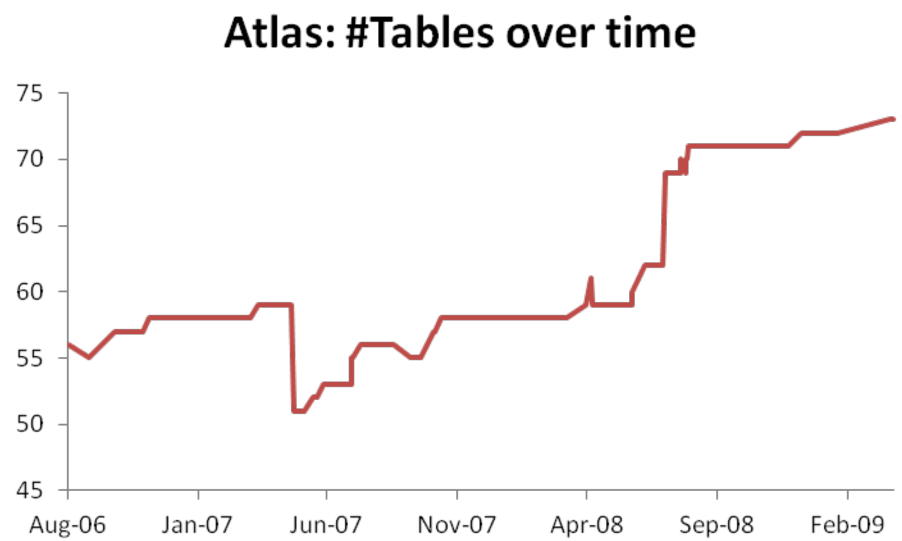
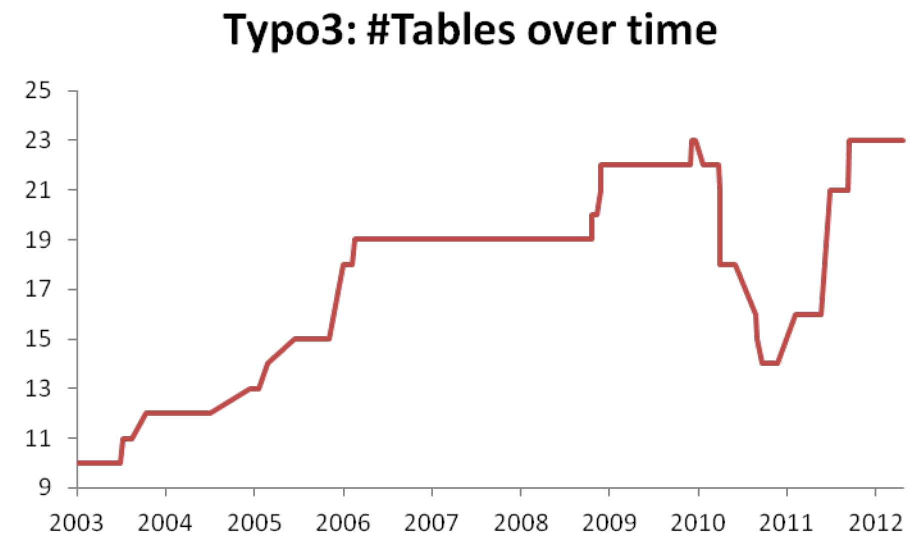
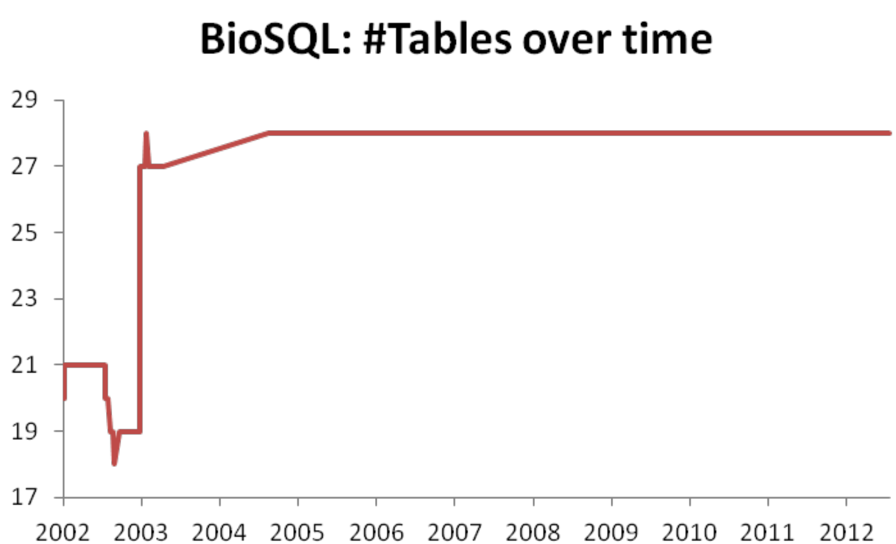
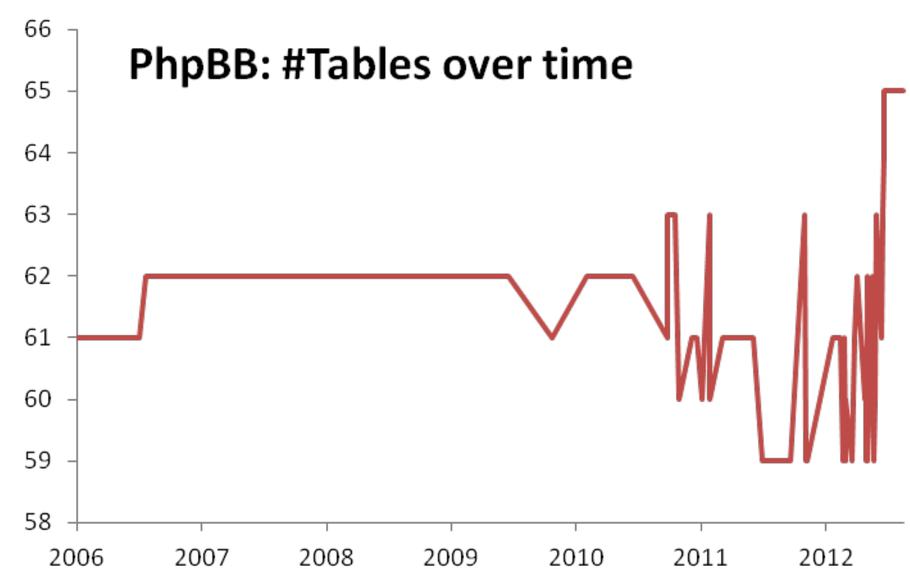
Increase both slow (mostly) and **abrupt**
Occasional abrupt drops (maintenance)

Mwiki: #Tables over time

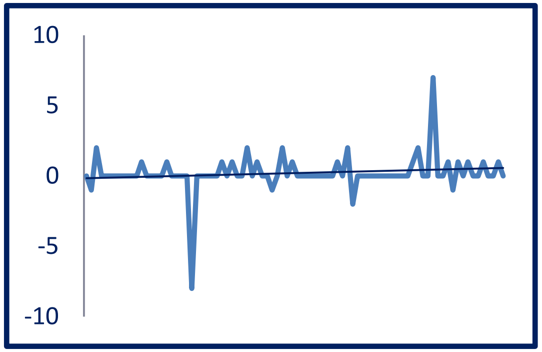


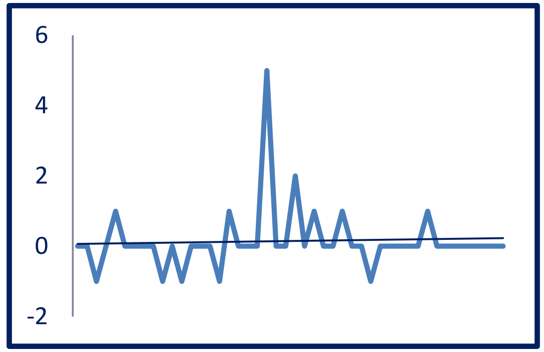
Opencart: #Tables over time

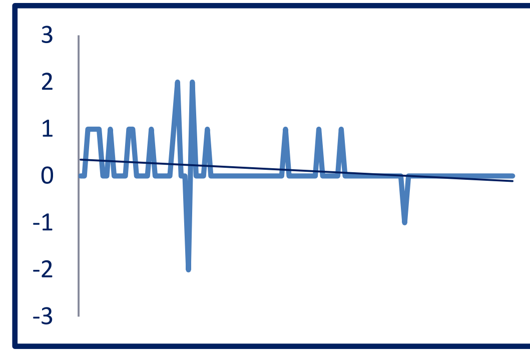


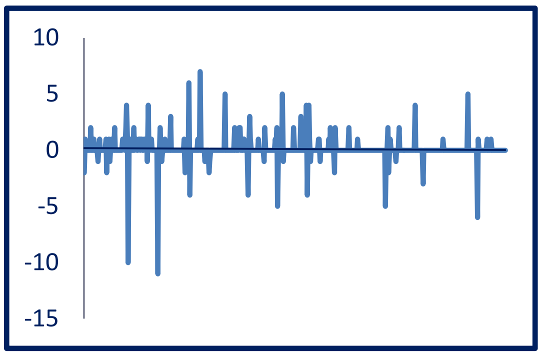

Atlas: #Tables over time
75
70
65
60
55
50
45
Aug-06
Jan-07
Jun-07
Nov-07
Apr-08
Sep-08
Feb-09

Typo3: #Tables over time
25
23
21
19
17
15
13
11
9
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012

BioSQL: #Tables over time
29
27
25
23
21
19
17
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012

PhpBB: #Tables over time
66
65
64
63
62
61
60
59
58
2006
2007
2008
2009
2010
2011
2012

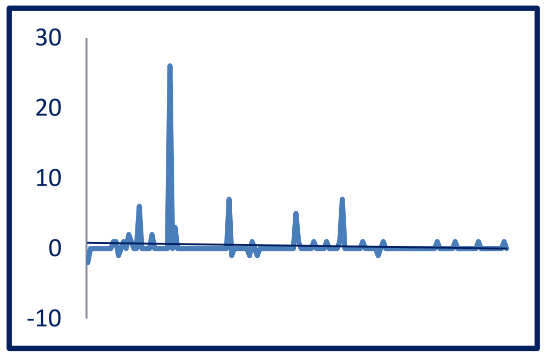
Schema Growth (diff in #tables)

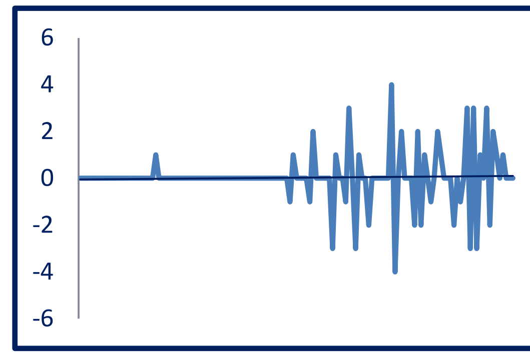


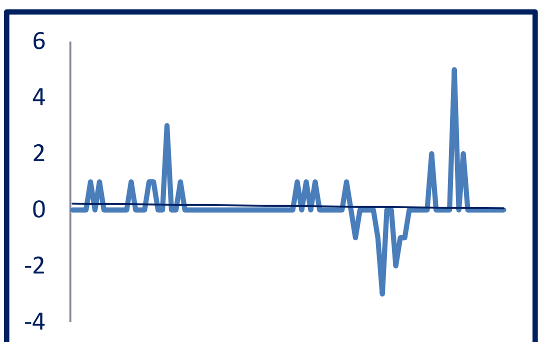


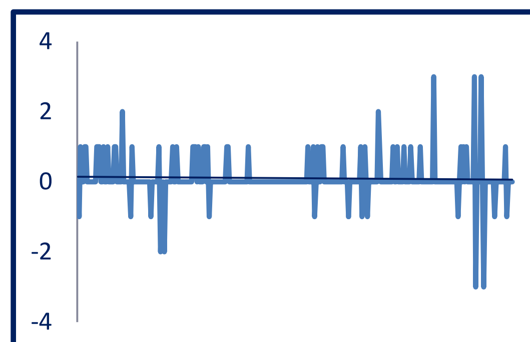










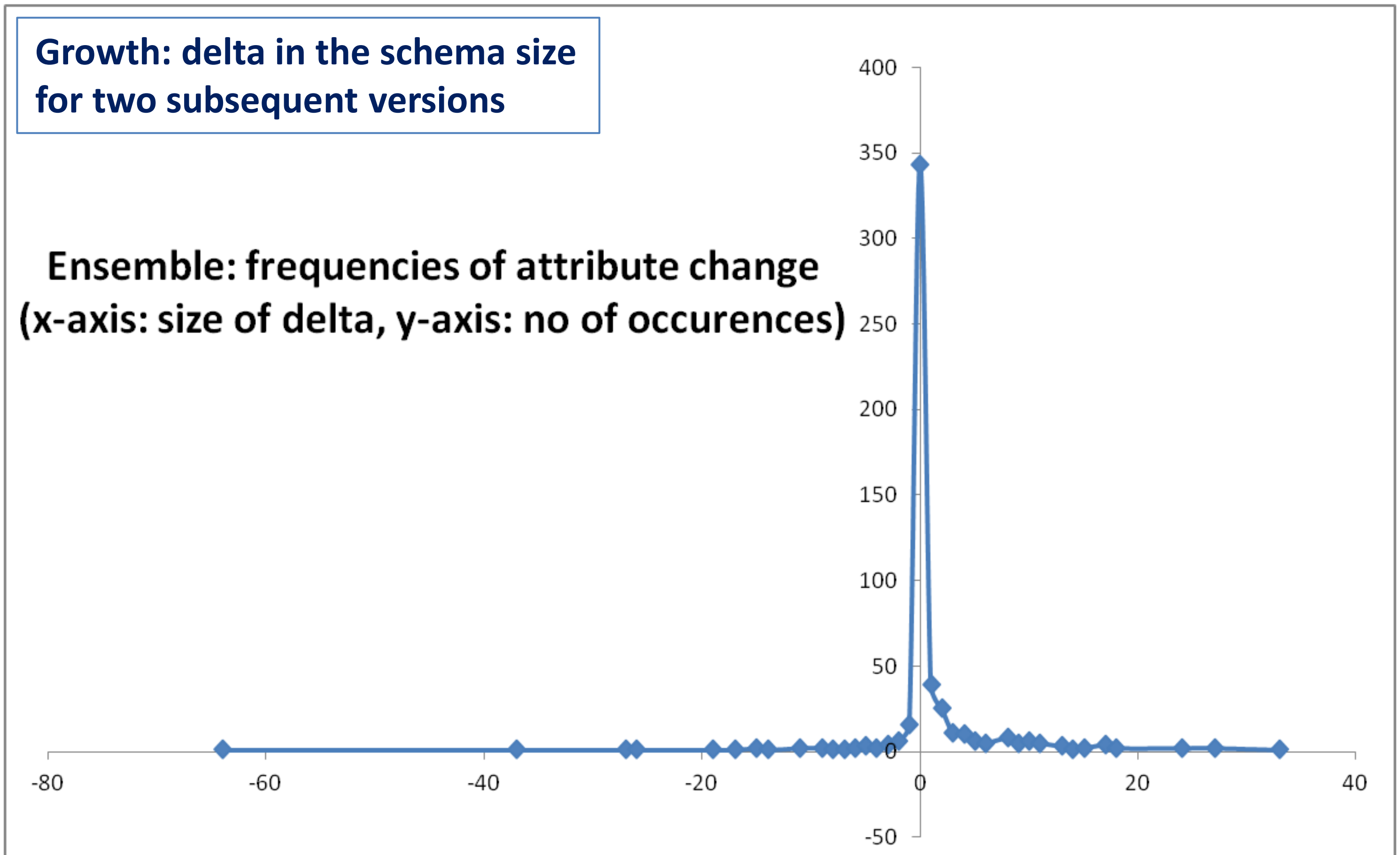


Schema growth is small!

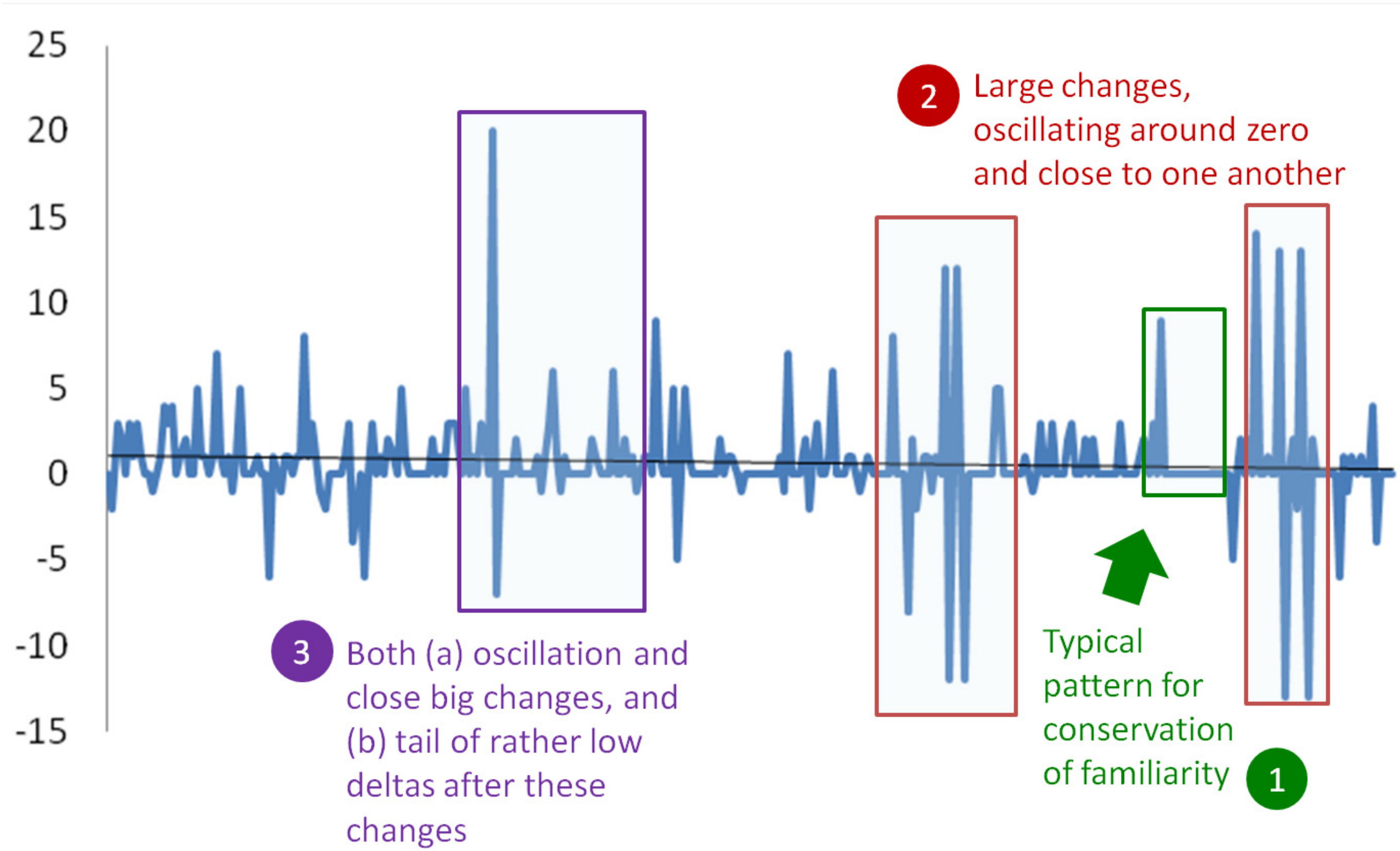
- **Growth is bounded in small values!**
- **Zipfian distribution of growth values around 0**
  - Predominantly: occurrences of zero growth; **almost all deltas are bounded between [-2..2] tables**
  - [0..2] tables slightly more popular => average value of growth slightly higher than 0
- No periods of continuous change; **small spikes instead**

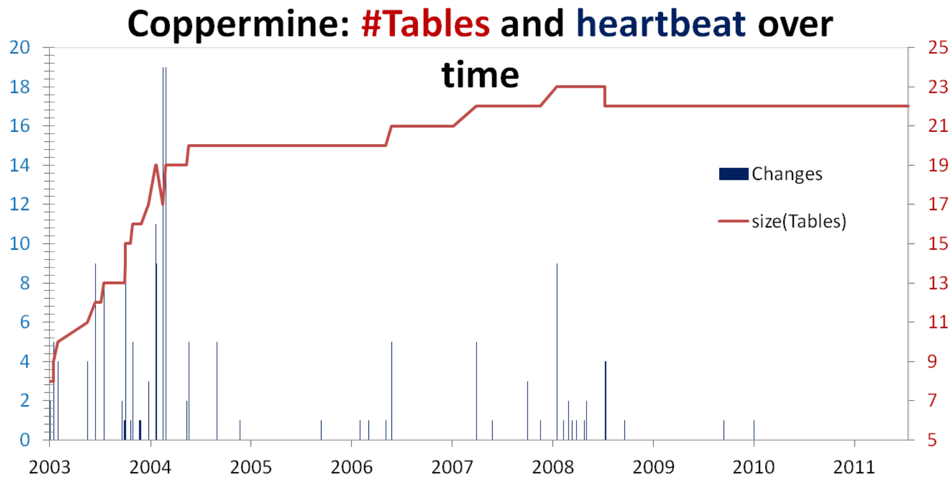
- Due to perfective maintenance, we also have negative values of growth (less than the positive ones).
- Oscillations exist too: positive growth is followed with immediate negative growth or stability

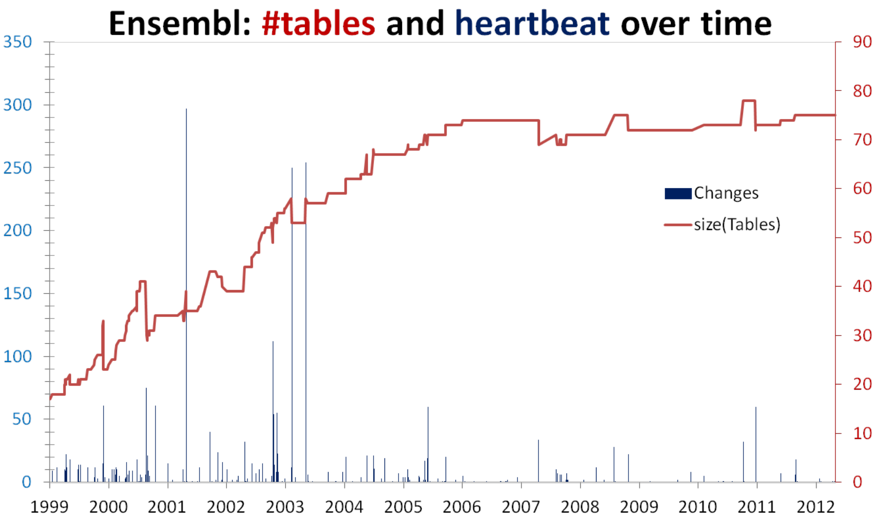
Zipfian model in the distribution of growth frequencies

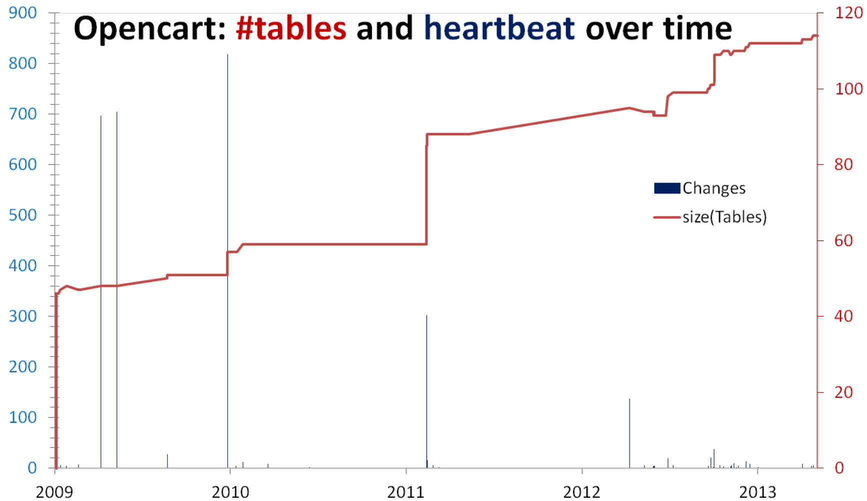


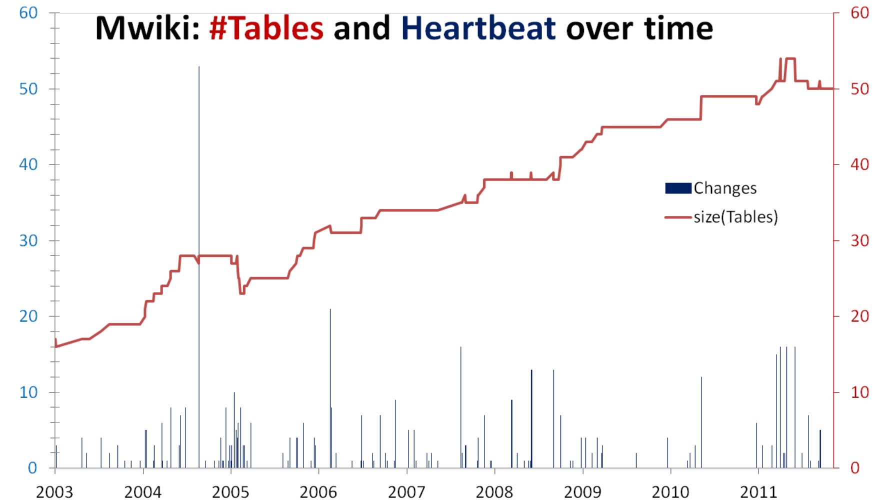
What happens after large changes?

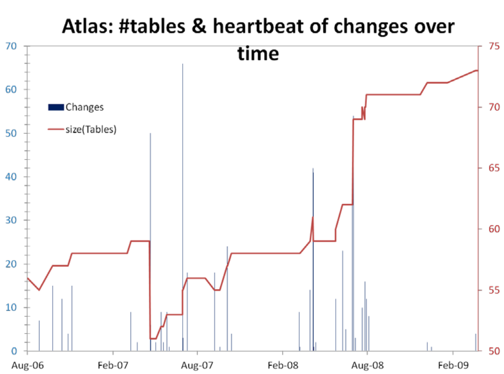












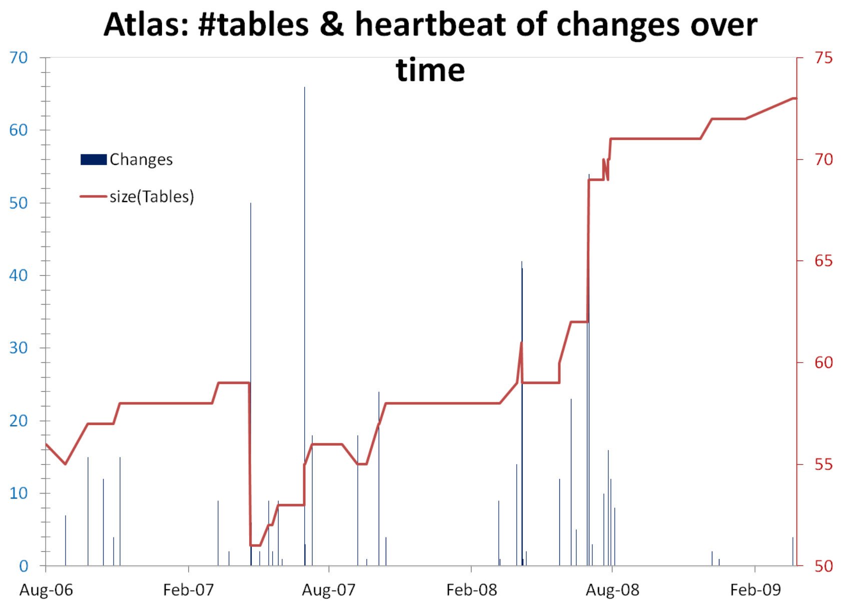
[With exceptions]

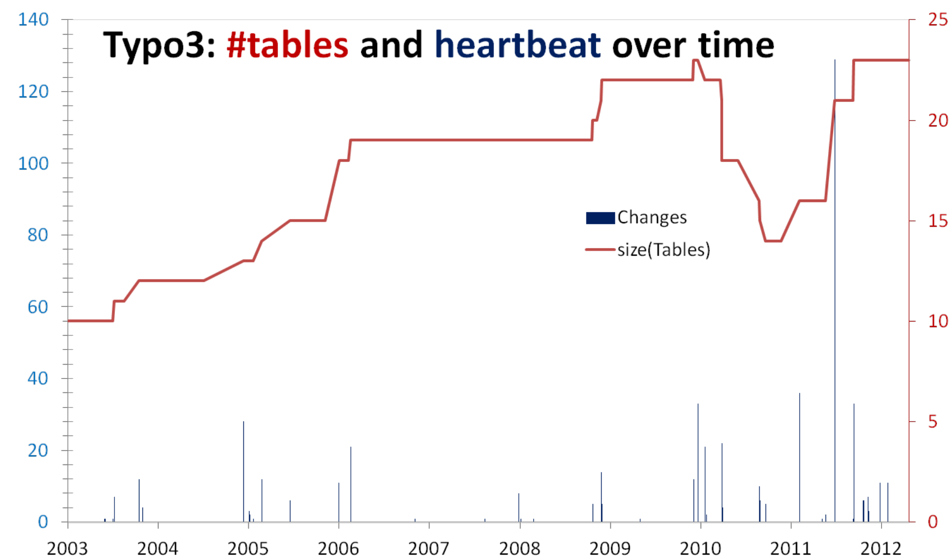
Density: **focused maintenance** effort

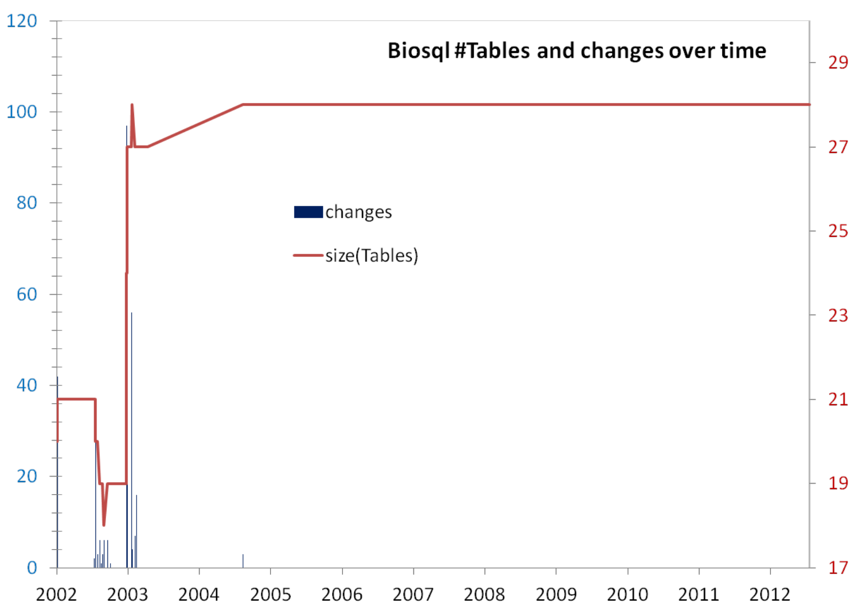
Progressive cooling : early –maintenance density >> later stages

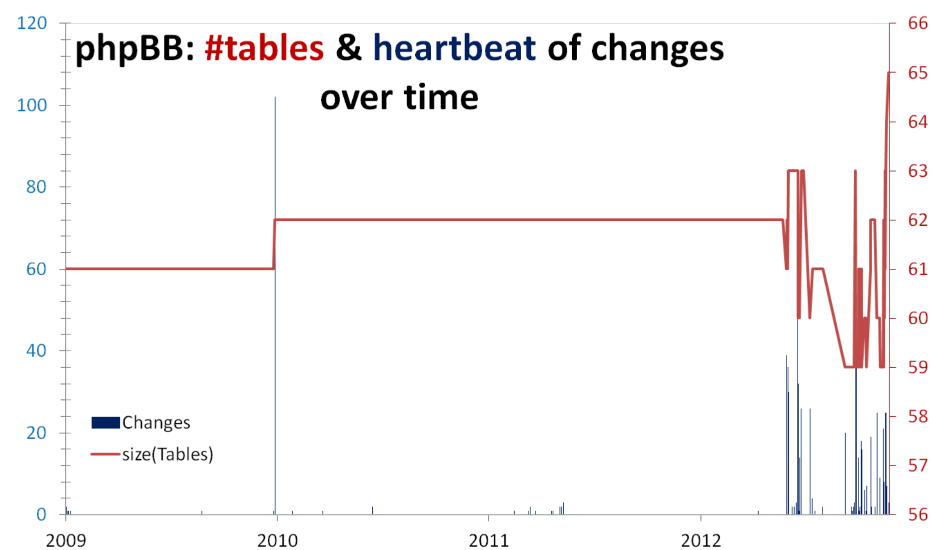
Several **spikes**, many **zero-change** periods/versions

#tables & heartbeat of changes over time









Main results

Schema size (#tables, #attributes) supports the assumption of a feedback mechanism

- Schema size grows over time; not continuously, but with bursts of concentrated effort
- Drops in schema size signify the existence of perfective maintenance
- Regressive formula for size estimation holds, with a quite short memory

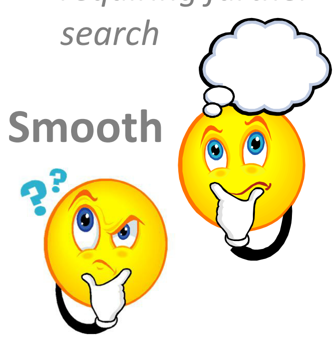
Schema Growth (diff in size between subsequent versions) is small!!

- Growth is small, smaller than in typical software
- The number of changes for each evolution step follows Zipf's law around zero
- Average growth is close (slightly higher) to zero

Grey for results requiring further search

Patterns of change: no consistently constant behavior

- Changes reduce in density as databases age
- Change follows three patterns: **Stillness**, **Abrupt change** (up or down), **Smooth growth upwards**
- Change frequently follows **spike** patterns
- **Complexity** does **not** increase with age



.. **What do we see if we observe the evolution of individual tables?**

http://www.cs.uoi.gr/~pvassil/publications/2015_ER

P. Vassiliadis, A. Zarras, I. Skoulis. How is Life for a Table in an Evolving Relational Schema? Birth, Death & Everything in Between. **ER 2015**

Gravitating to rigidity: Patterns of schema evolution – and its absence – in the lives of tables. Accepted in **Information Systems.**

OBSERVING THE EVOLUTION OF O/S DB SCHEMATA AT THE MICRO LEVEL

Exploratory search of the schema histories for patterns

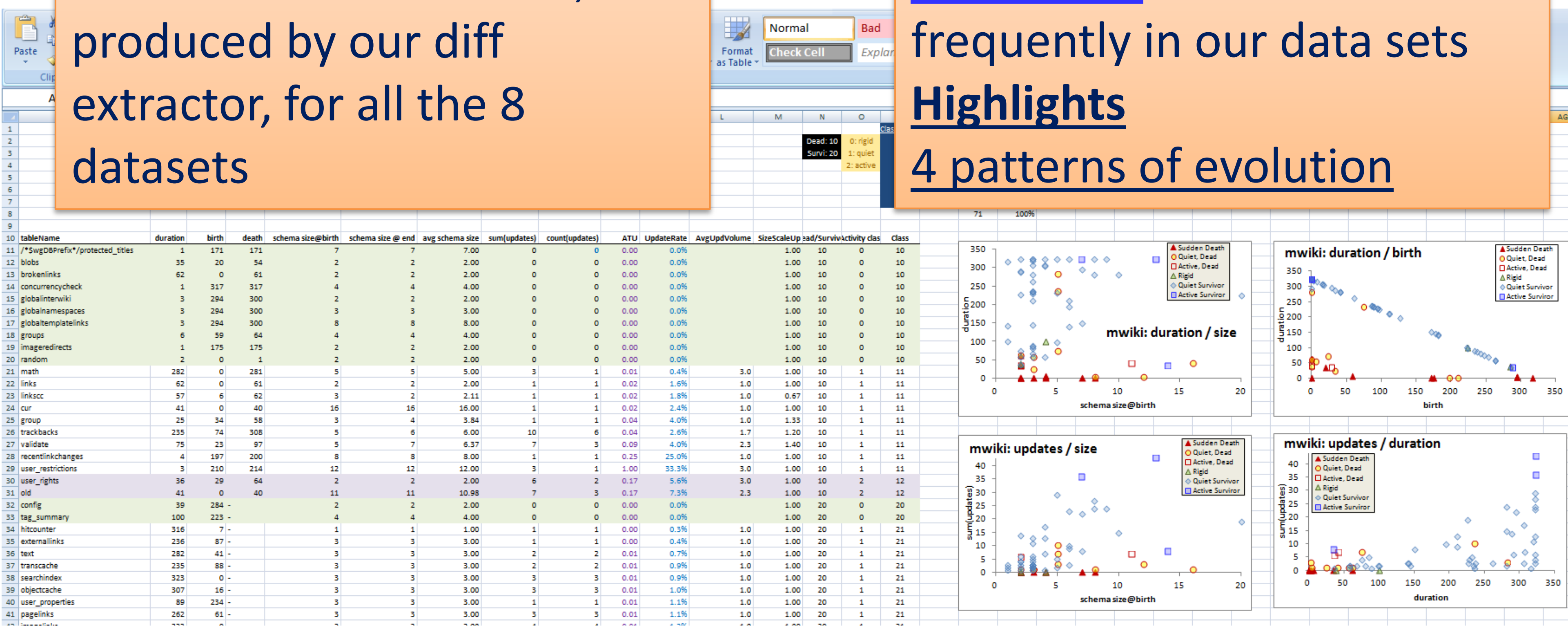
Input: schema histories from github/sourceforge/…

Raw material: details and stats on each table's life, as produced by our diff extractor, for all the 8 datasets

Output: properties & patterns **on table properties (birth, duration, amt of change, …)** that occur frequently in our data sets

Highlights

4 patterns of evolution



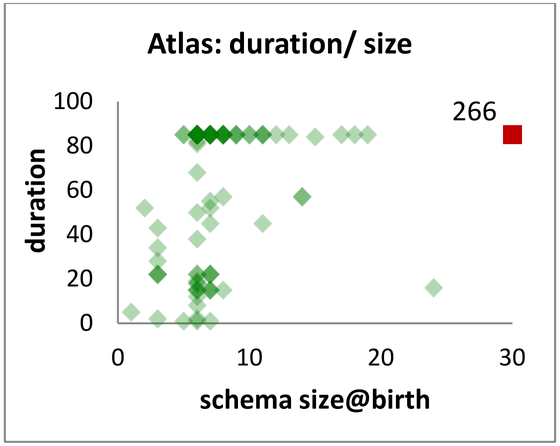
-Statistical properties for schema size, change and duration of tables

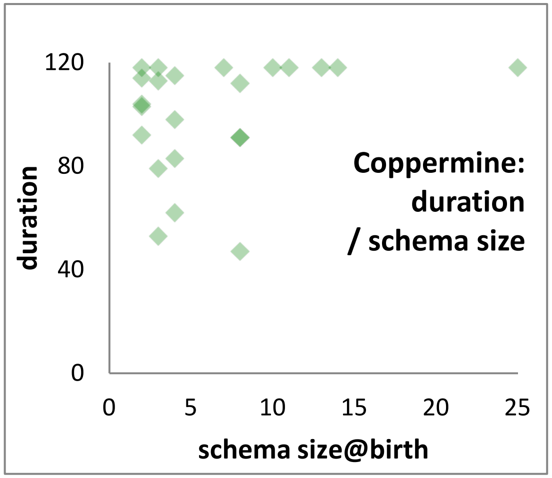
- How are these measures interrelated?

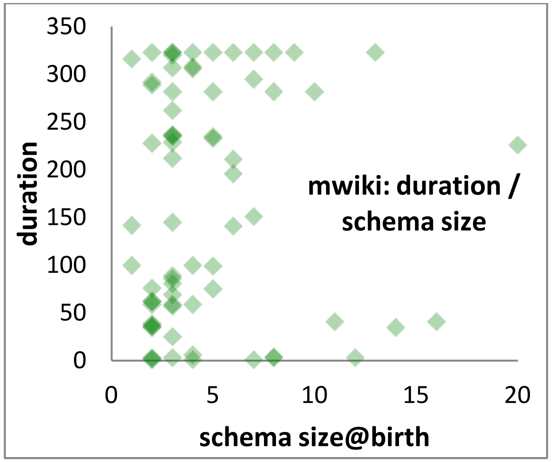
SCHEMA SIZE, CHANGE AND DURATION

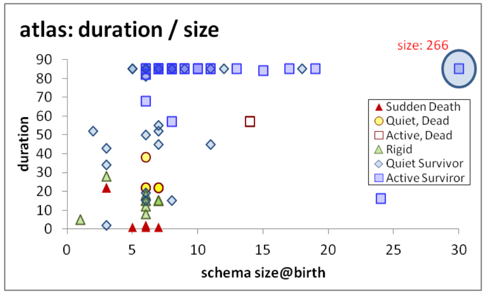
The Gamma Γ Pattern: "if you 're wide, you survive"

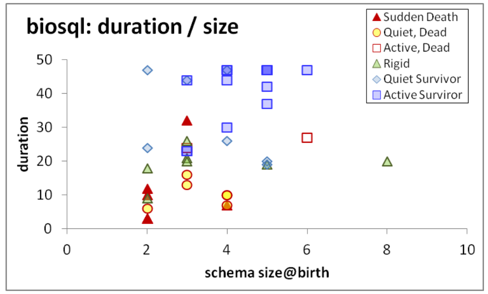
- The Gamma phenomenon:
  - tables with small schema sizes can have arbitrary durations, //small size does not determine duration
  - larger size tables last long
- Observations:
  - whenever a table exceeds the critical value of 10 attributes in its schema, its chances of surviving are high.
  - in most cases, the large tables are created early on and are not deleted afterwards.

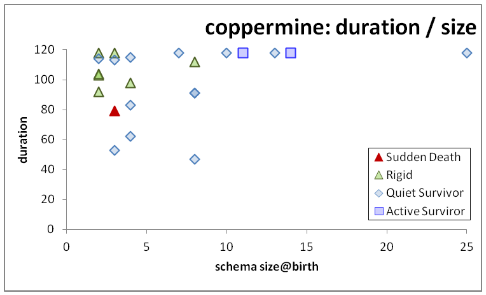


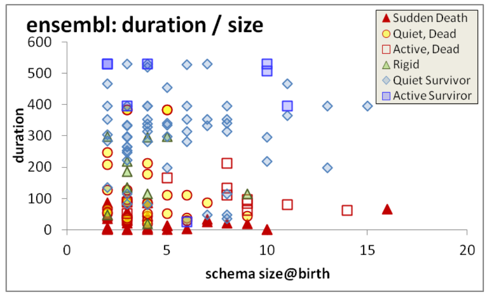


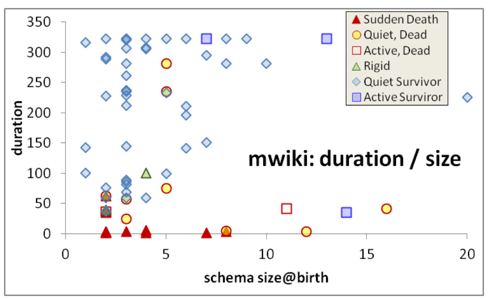


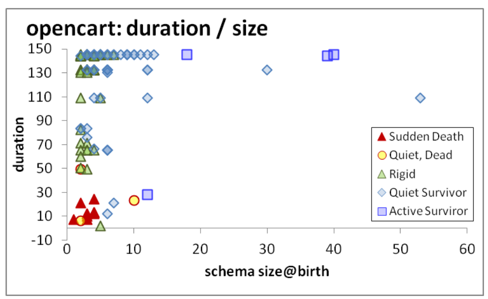


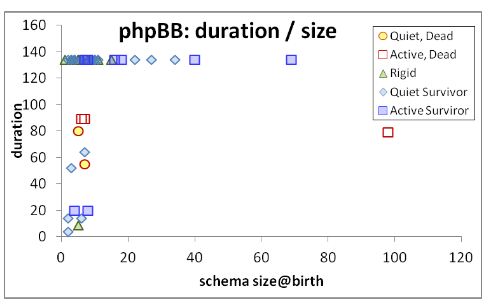


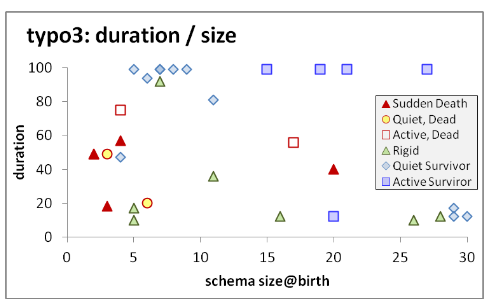












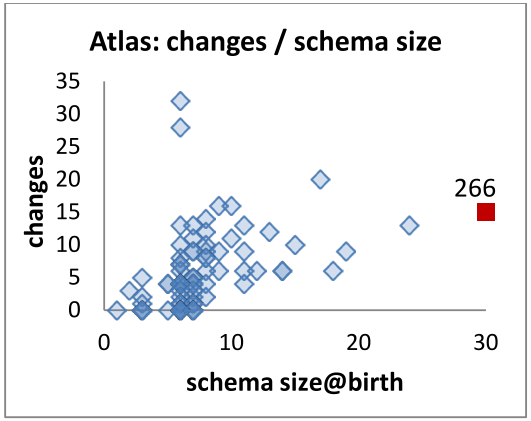
Exceptions

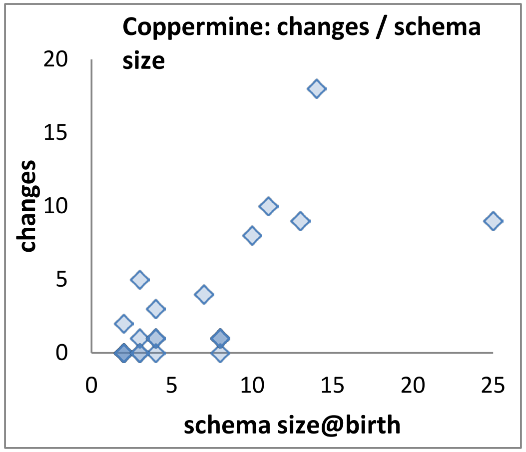
- Biosql: nobody exceeds 10 attributes
- Ensembl, mwiki: very few exceed 10 attributes, 3 of them died
- typo: has many late born survivors

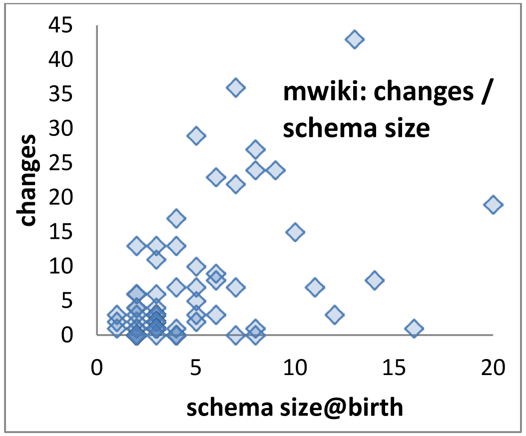
The Comet Pattern

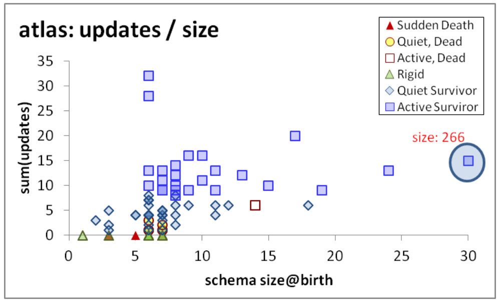
“Comet “ for change over schema size with:

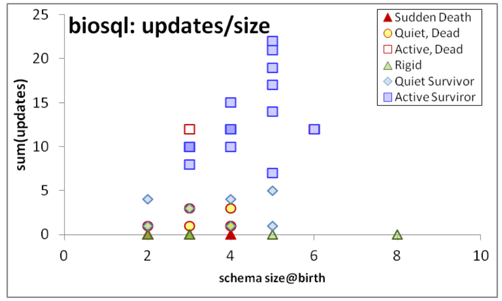
- a large, dense, nucleus cluster close to the beginning of the axes, denoting small size and small amount of change,
- medium schema size tables typically demonstrating medium to large change
  – The tables with the largest amount of change are typically tables whose schema is on average one standard deviation above the mean
- wide tables with large schema sizes demonstrating small to medium (typically around the middle of the y-axis) amount of change.

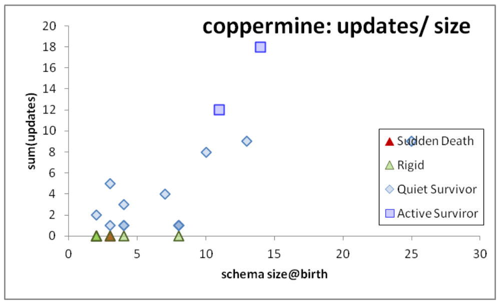


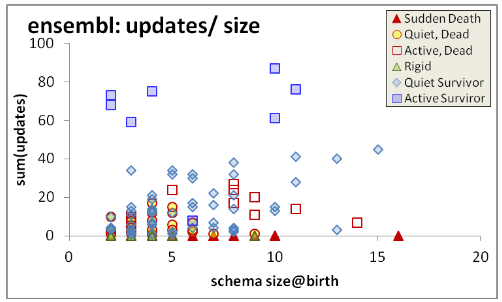


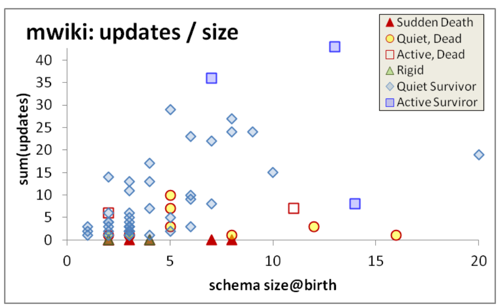


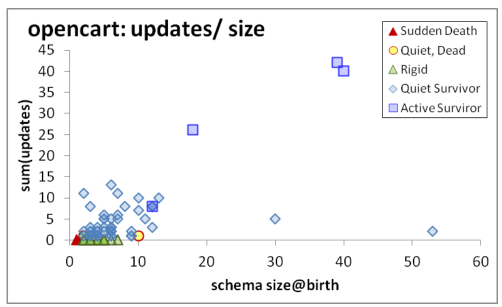


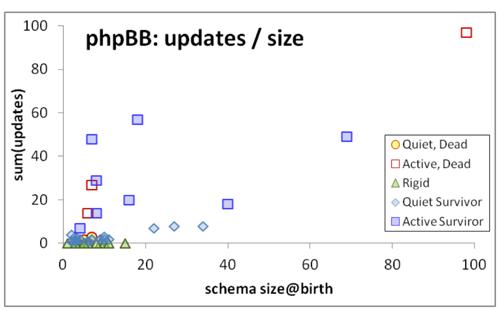


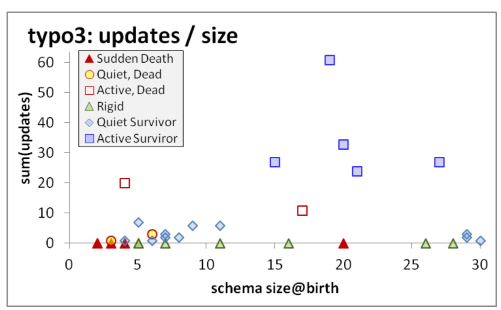




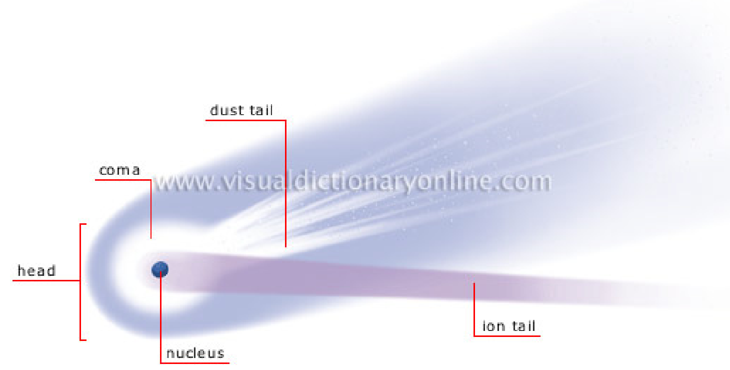


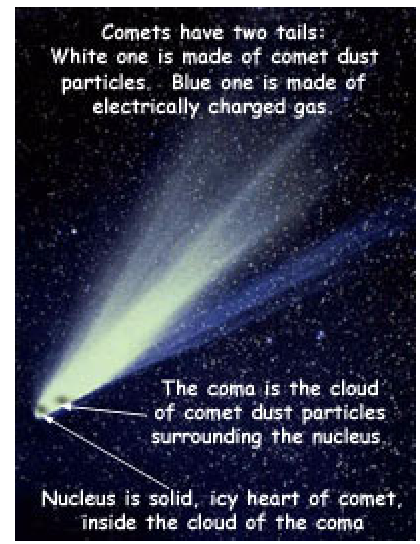






http://visual.merriam-webster.com/astronomy/celestial-bodies/comet.php

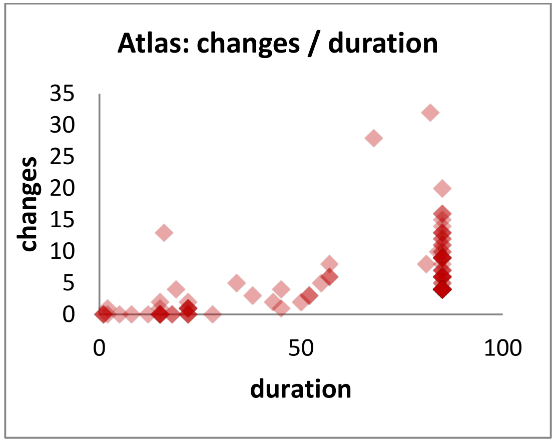


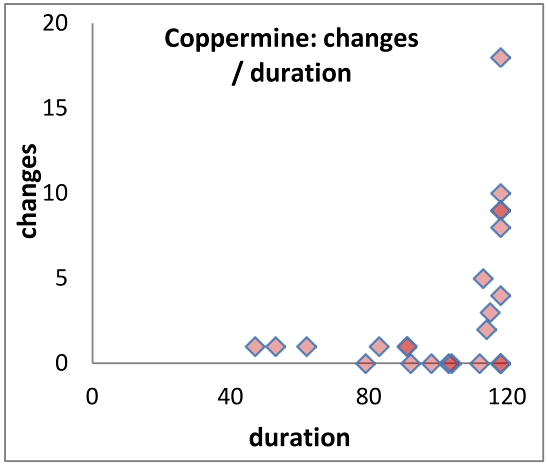


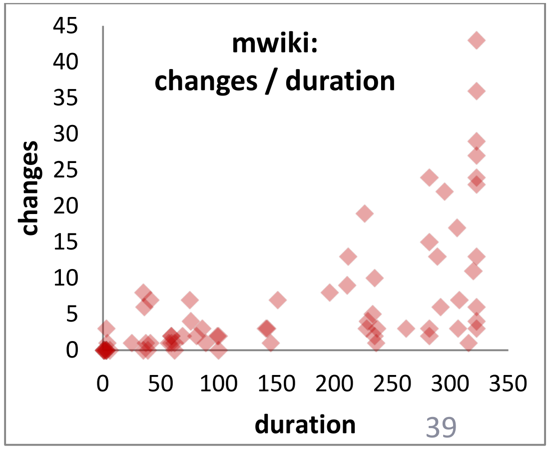
http://spaceplace.nasa.gov/comet-nucleus/en/

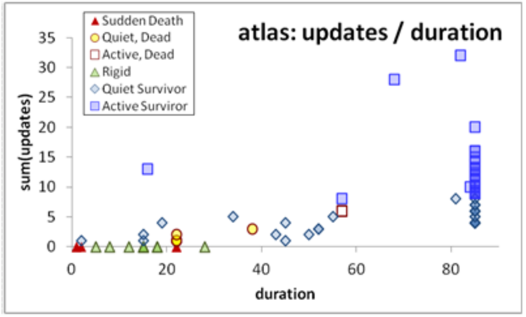
The inverse Gamma pattern

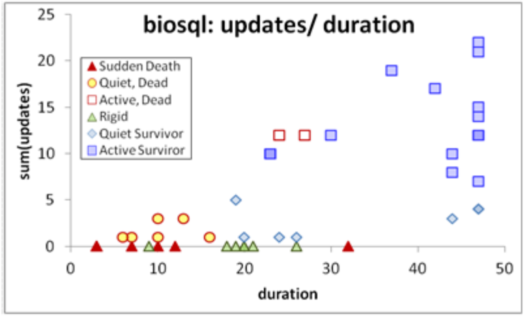
- The correlation of change and duration is as follows:
  - small durations come necessarily with small change,
  - large durations come with all kinds of change activity and
  - medium sized durations come mostly with small change activity (Inverse Gamma).

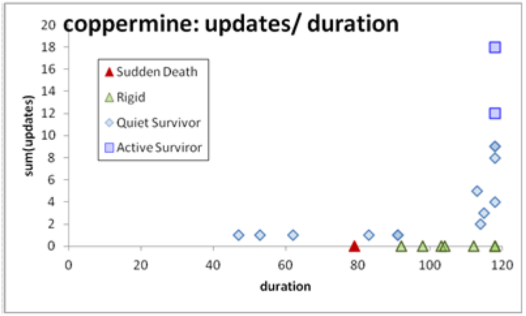


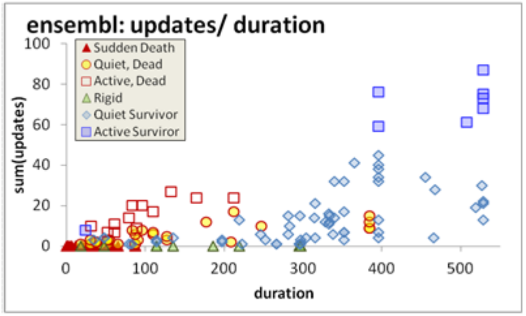


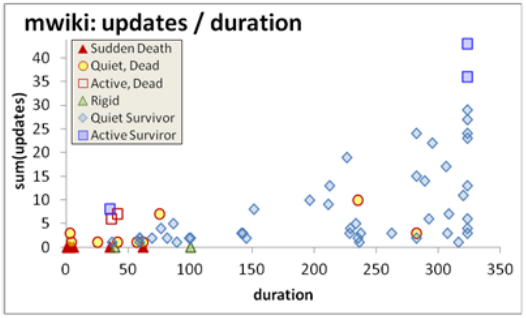


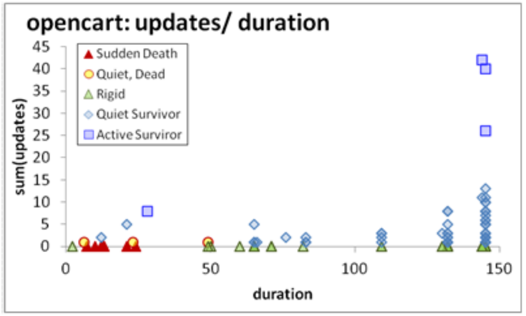

atlas: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 5 10 15 20 25 30 35
0 20 40 60 80
duration

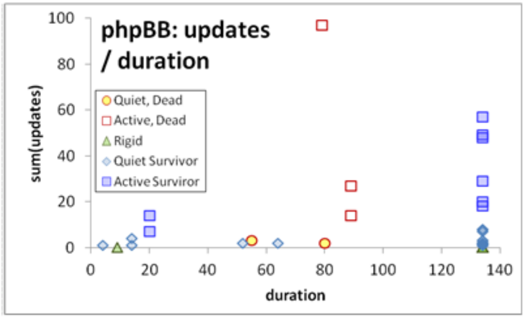

biosql: updates/ duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 5 10 15 20 25
0 10 20 30 40 50
duration

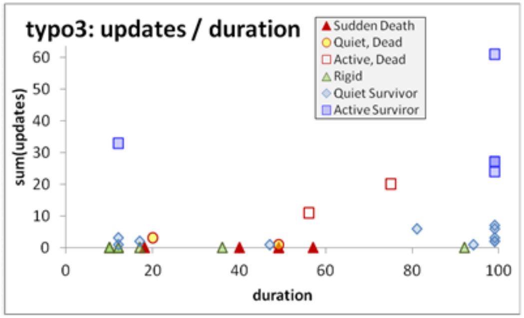

coppermine: updates/ duration
Sudden Death
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 2 4 6 8 10 12 14 16 18 20
0 20 40 60 80 100 120
duration


ensembl: updates/ duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 20 40 60 80 100
0 100 200 300 400 500
duration


mwiki: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 5 10 15 20 25 30 35 40
0 50 100 150 200 250 300 350
duration


opencart: updates/ duration
Sudden Death
Quiet, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 5 10 15 20 25 30 35 40 45
0 50 100 150
duration


phpBB: updates / duration
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 20 40 60 80 100
0 20 40 60 80 100 120 140
duration


typo3: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
0 10 20 30 40 50 60
0 20 40 60 80 100
duration

Who are the top changers?

Who are removed at some point of time?

How do removals take place?

BIRTHDAY & SCHEMA SIZE & MATTERS OF LIFE AND DEATH

Quiet tables rule, esp. for mature db's

Table distribution (pct of tables) wrt their activity class

	#tables	DIED				SURVIVED				*Aggregate per update type*		
		No change	Quiet	Active	**Total**	No change	Quiet	Active	**Total**	No change	Quiet	Active
atlas	88	8%	7%	2%	**17%**	13%	42%	**28%**	**83%**	20%	49%	**31%**
biosql	45	20%	13%	4%	**38%**	16%	16%	**31%**	**62%**	36%	29%	**36%**
phpbb	70	0%	3%	4%	**7%**	50%	31%	**11%**	**93%**	50%	34%	**16%**
typo3	32	16%	6%	6%	**28%**	22%	34%	**16%**	**72%**	38%	41%	**22%**
coppermine	23	4%	0%	0%	**4%**	30%	57%	**9%**	**96%**	35%	**57%**	**9%**
ensembl	155	24%	20%	8%	**52%**	6%	35%	**7%**	**48%**	30%	**55%**	**15%**
mwiki	71	14%	13%	3%	**30%**	3%	63%	**4%**	**70%**	17%	**76%**	**7%**
opencart*	128	9%	2%	0%	**11%**	42%	44%	**3%**	**89%**	51%	46%	**3%**

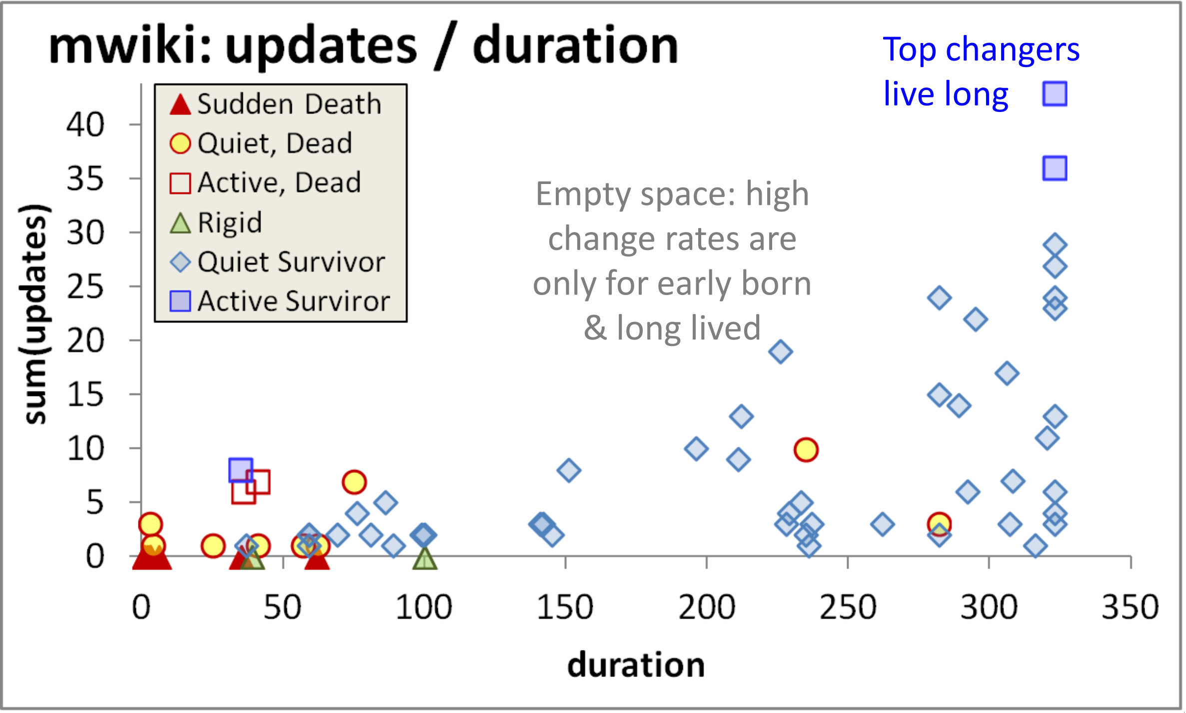
Non-survivors

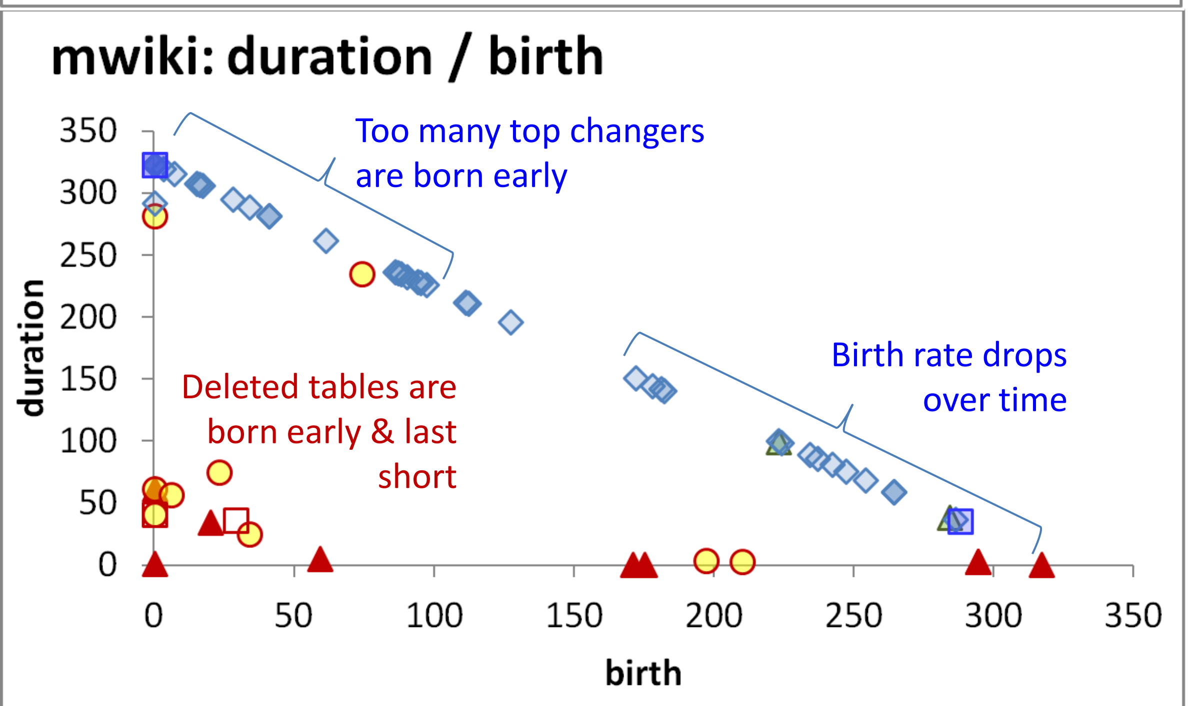
- Sudden deaths mostly
- Quiet come ~ close
- Too few active

Survivors

- Quiet tables rule
- Rigid and active then
- Active mostly in "new" db's

Mature DB's: the pct of active tables drops significantly

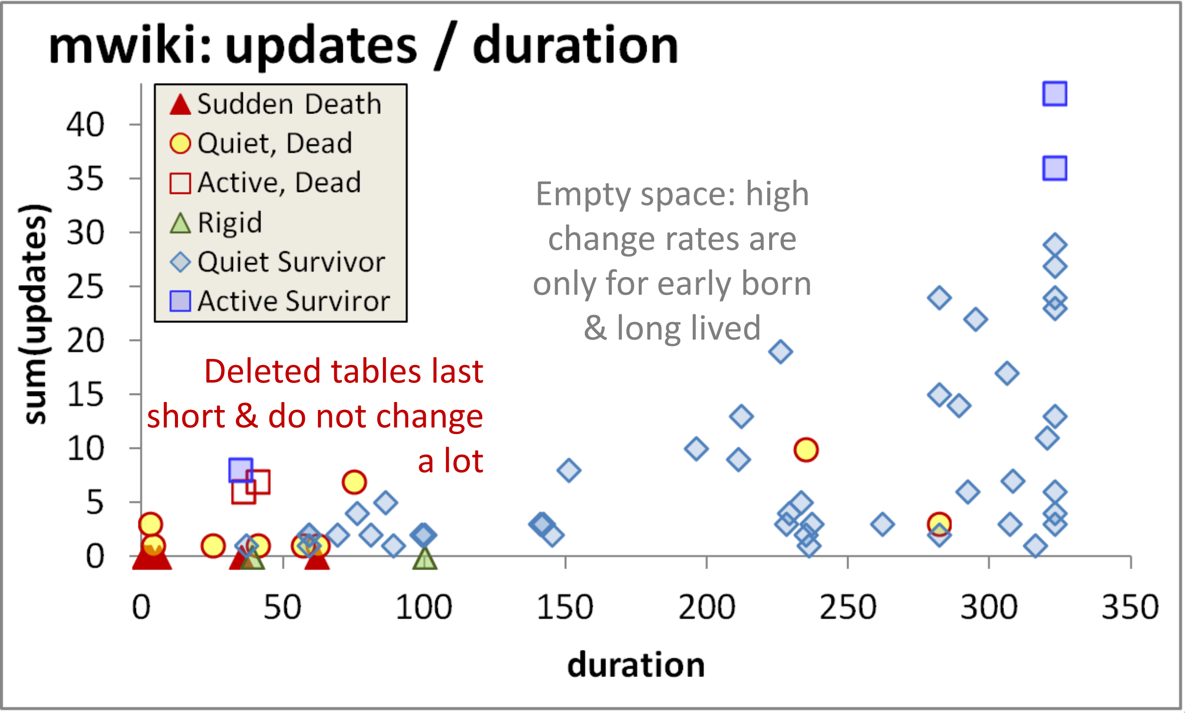


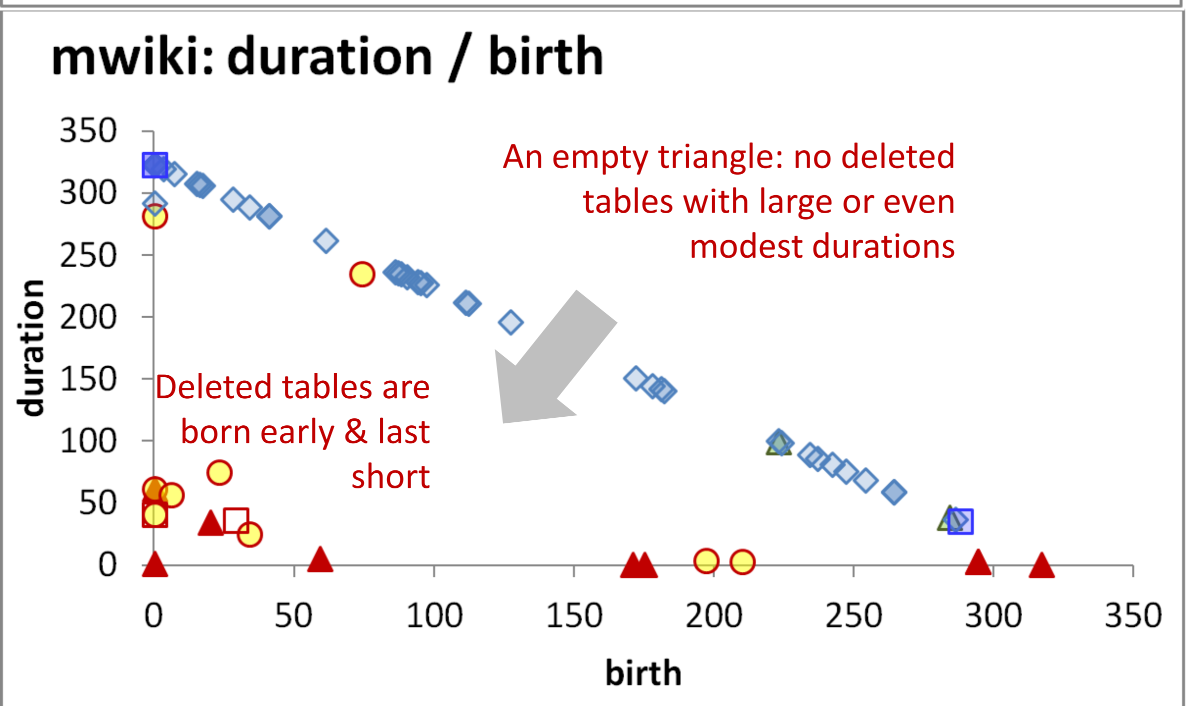


Longevity and update activity correlate !!

The few top-changers (in terms of avg trans. update – ATU)

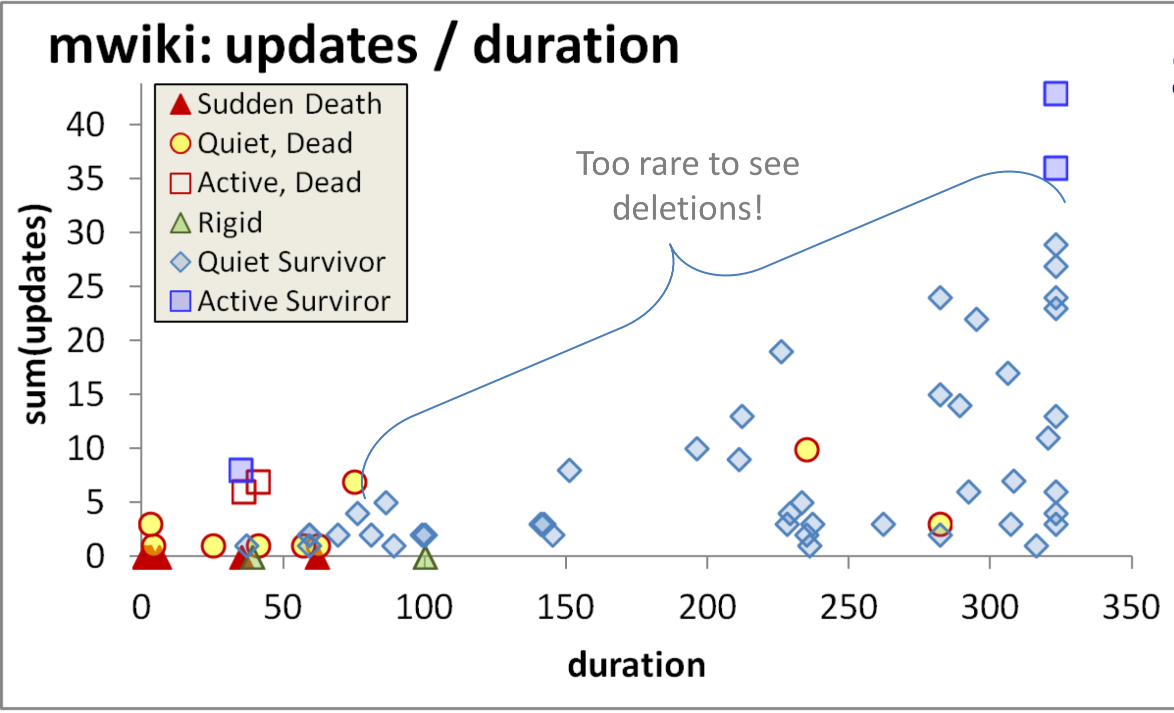
- are long lived,
- typically come from the early versions of the database
- due to the combination of high ATU and duration => they have high total amount of updates, and,
- frequently survive!

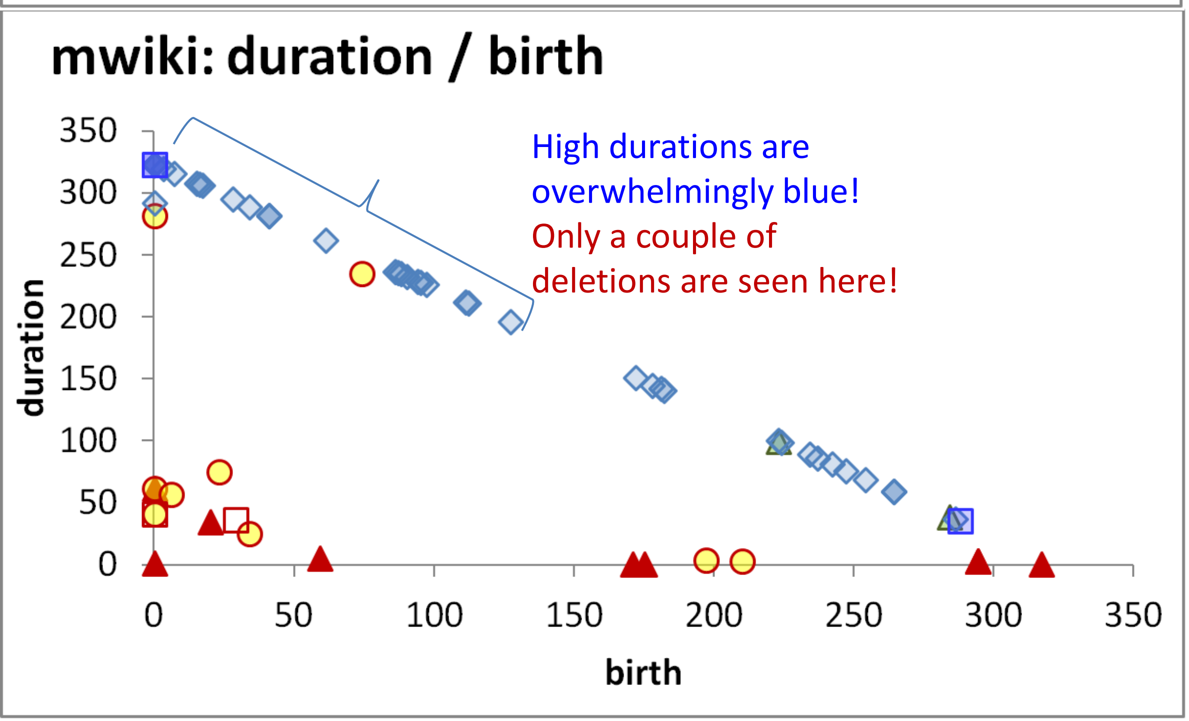




Die young and suddenly

- There is a very large concentration of the deleted tables in a small range of newly born, quickly removed, with few or no updates…
- …. resulting in very low numbers of removed tables with medium or long durations (empty triangle).

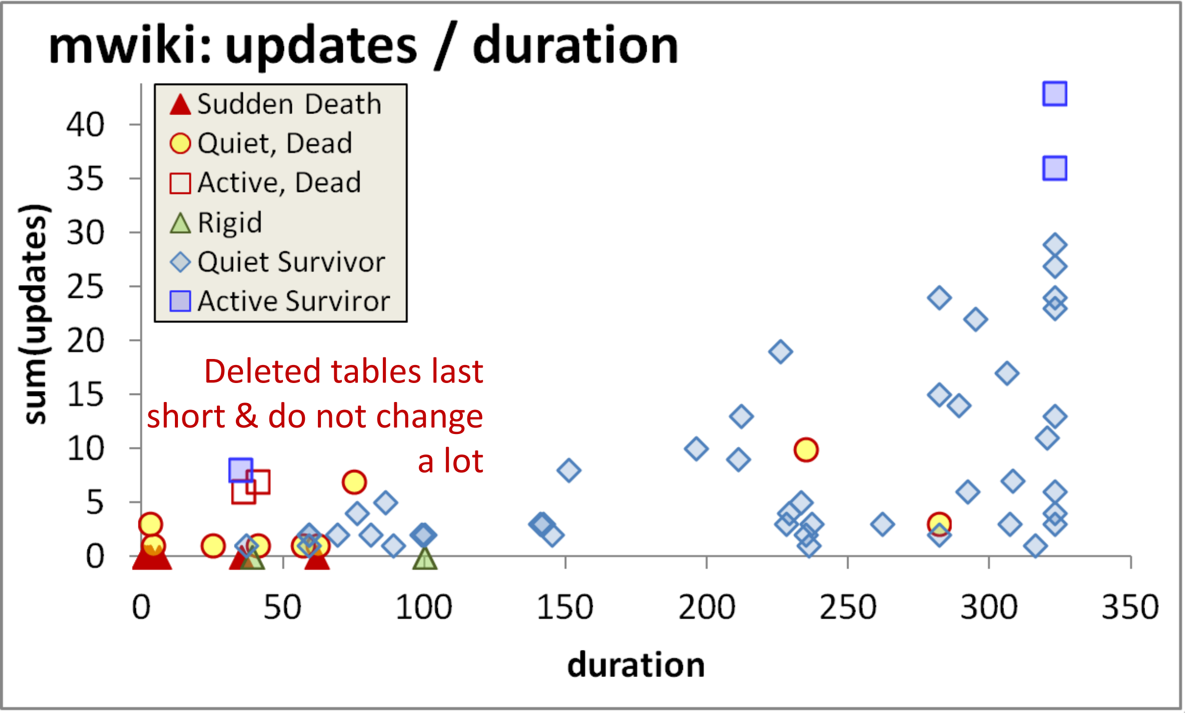


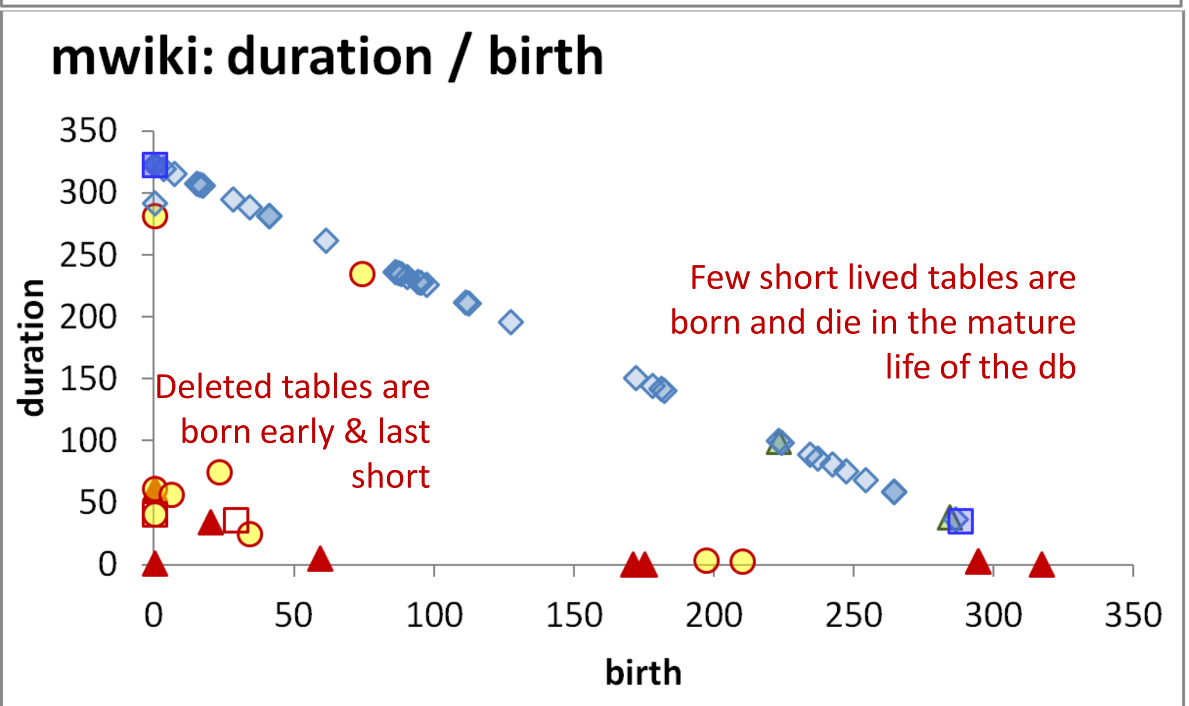


Survive long enough & you ‘re probably safe

It is quite rare to see tables being removed at old age

Typically, the area of high duration is overwhelmingly inhabited by survivors (although each data set comes with a few such cases)!

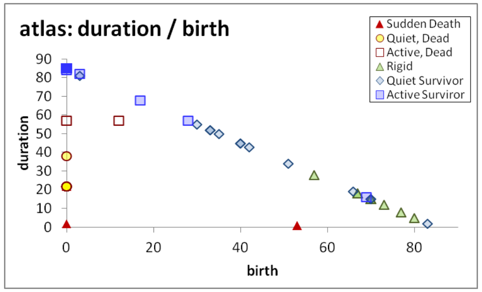


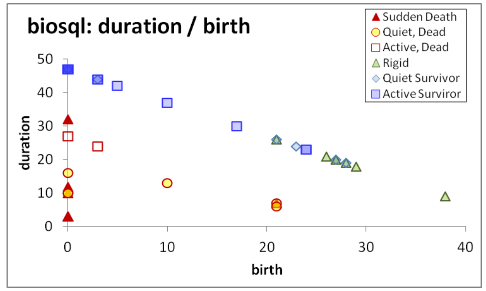


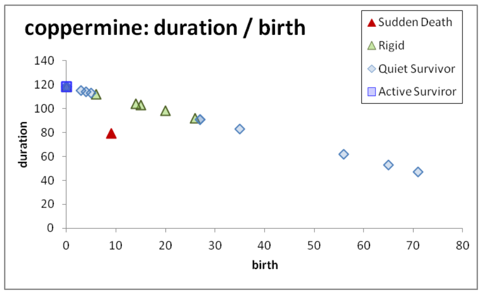
Die young and suddenly

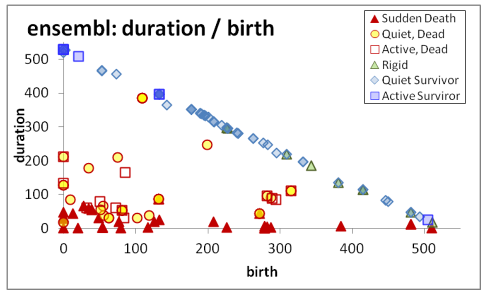
[Early life of the db] There is a very large concentration of the deleted tables in a small range of newly born, quickly removed, with few or no updates, resulting in very low numbers of removed tables with medium or long durations.

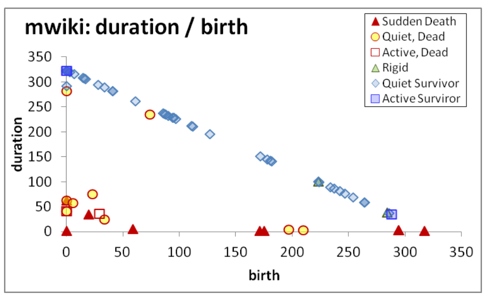
[**Mature db**] After the early stages of the databases, we see the birth of tables who eventually get deleted, but they mostly come with very small durations and sudden deaths.

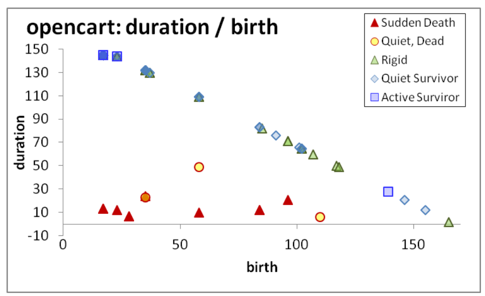

atlas: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

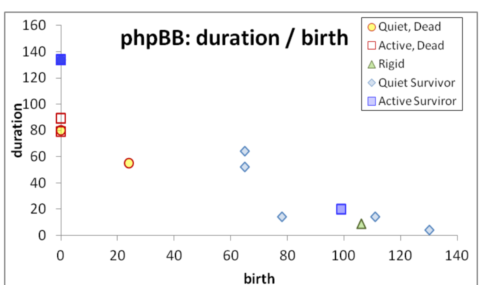

biosql: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

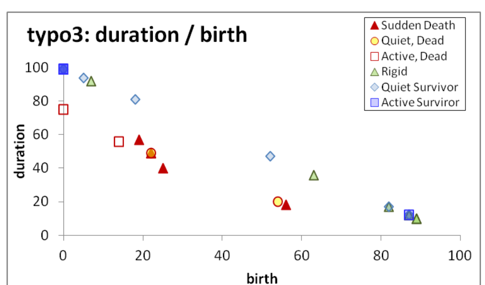

coppermine: duration / birth
Sudden Death
Rigid
Quiet Survivor
Active Surviror
duration
birth

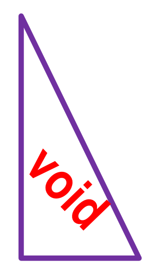

ensembl: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


mwiki: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

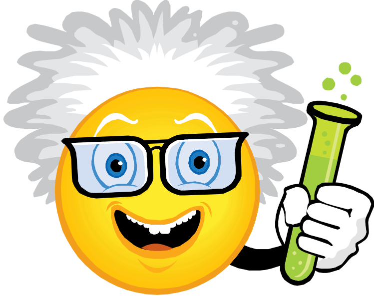

opencart: duration / birth
Sudden Death
Quiet, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

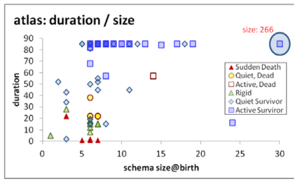

phpBB: duration / birth
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


typo3: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

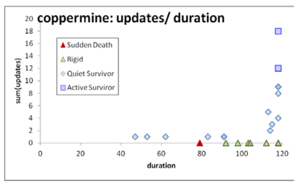

void

Regularities on table change do exist!

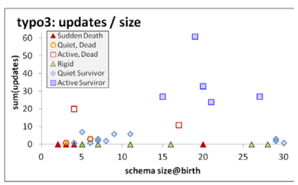




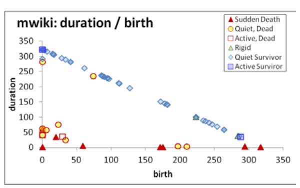
If you're wide, you survive

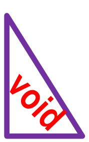


Top-changers typically live long, are early born, survive …



… and they are not necessarily the widest ones in terms of schema size





Progressive cooling: most change activity lies at the beginning of the db history

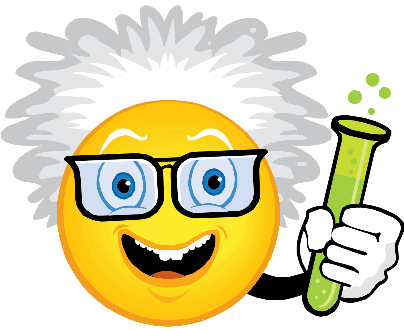
Void triangle: The few dead tables are typically quiet, early born, short lived, and quite often all three of them

Where we stand

Open issues

… and discussions …

OPEN ISSUES



Where we stand

- We have a first glimpse of the mechanics of schema evolution for FoSS ecosystems
- We have a first understanding of schemata growing, changed in focused periods of maintenance and progressively “cooling” down
- We have a first understanding of patterns relating to how tables change, given their size, update behavior, time of birth, …

To probe further (code, data, details, presentations, …)

http://www.cs.uoi.gr/~pvassil/projects/schemaBiographies/

Are there “laws” of schema evolution?

- Collect more test cases
- Tools for the automation of the process
  - Extract changes & verify their correctness (what happened)
  - Link changes to expressed user req’s / bugs / … (why it happened & by whom)
  - Extract sub-histories of focused maintenance (how it happened & when)
  - Co-change of schema and code (what is affected in the code)
  - Visualization
- Consolidate the fundamental laws that govern evolution && forecast it (what will change)

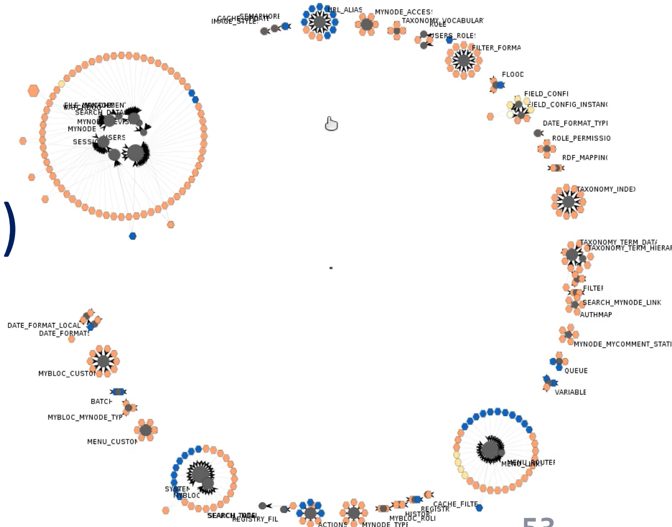
Unexplored research territory (risky but possibly rewarding)

- Weather Forecast: given the history and the state of a database, predict subsequent events
  - Risky: frequently, changes come due to an external, changing world and have “thematic” affinity.
  - Big & small steps in many directions needed (more data sets, studies with high internal validity to find causations, more events to capture, …)
- Engineer for evolution: To absorb change gracefully we can try to (i) alter db design and DDL; (ii) encapsulate the database via a “stable” API; …

Management of ecosystems' evolution

- Can we find these constructs that are most sensitive to evolution?
  - Metrics for sensitivity to evolution?

- Automation of the reaction to changes
  - self-monitoring
  - impact prediction
  - auto-regulation (policy determination)
  - self-repairing



http://www.cs.uoi.gr/~pvassil/projects/hecataeus/

Take Away Message

- Evolution is **viciously omnipresent**; due to its huge impact, **it is leading to non-evolvable (rigid) data & software structures**
- Practically:
  - **Plan for evolution**, well ahead of construction
  - So far, our solutions and tools help only so much
  - Industry not likely to help
- This is why **we can and have to do research**
  - We can do **pure scientific research** to find laws
  - We can do practical work for **tools and methods** that reduce the pain

… and don’t forget to put everything in the git …

Thank you!

Q&A

https://github.com/DAINTINESS-Group/

http://www.cs.uoi.gr/~pvassil/

DB Schema Evolution

Papers, Data sets, Code, Results

projects/schemaBiographies/

Architecture Graphs && Hecataeus

projects/hecataeus/

daintiness

DAta INTensive Information EcoSystemS Group

DAta INTensive Information EcoSystemS Group, Univ. Ioannina, Hellas

Ioannina, Greece

Filters ▾ | Find a repository... | + New repository

ParmenidianTruth — Java ★0 ⑂0

Visualizes the story of a database's schema as a pptx presentation

Updated 14 days ago

EvolutionDatasets — PLSQL ★0 ⑂1

forked from giskou/EvolutionDatasets

Updated 15 days ago

Hecate — Java ★0 ⑂3

forked from giskou/Hecate

Diff visualization between 2 SQL schemas

Updated on 2 Apr

Plutarch_Parallel_Lives — Java ★0 ⑂1

Visualizes the evolution of the tables of a database schema as parallel lives

Updated on 2 Apr

Hecataeus — Java ★0 ⑂2

forked from pmanousis/Hecataeus

Database evolution what-if analysis tool

Updated on 24 Oct 2014

AUXILIARY SLIDES

What are the “laws” of database (schema) evolution?

- How do databases change?
- In particular, how does the schema of a database evolve over time?

- Long term research goals:
  - Are there any “invariant properties” (e.g., patterns of repeating behavior) on the way database (schemata) change?
  - Is there a theory / model to explain them?
  - Can we exploit findings to engineer data-intensive ecosystems that withstand change gracefully?

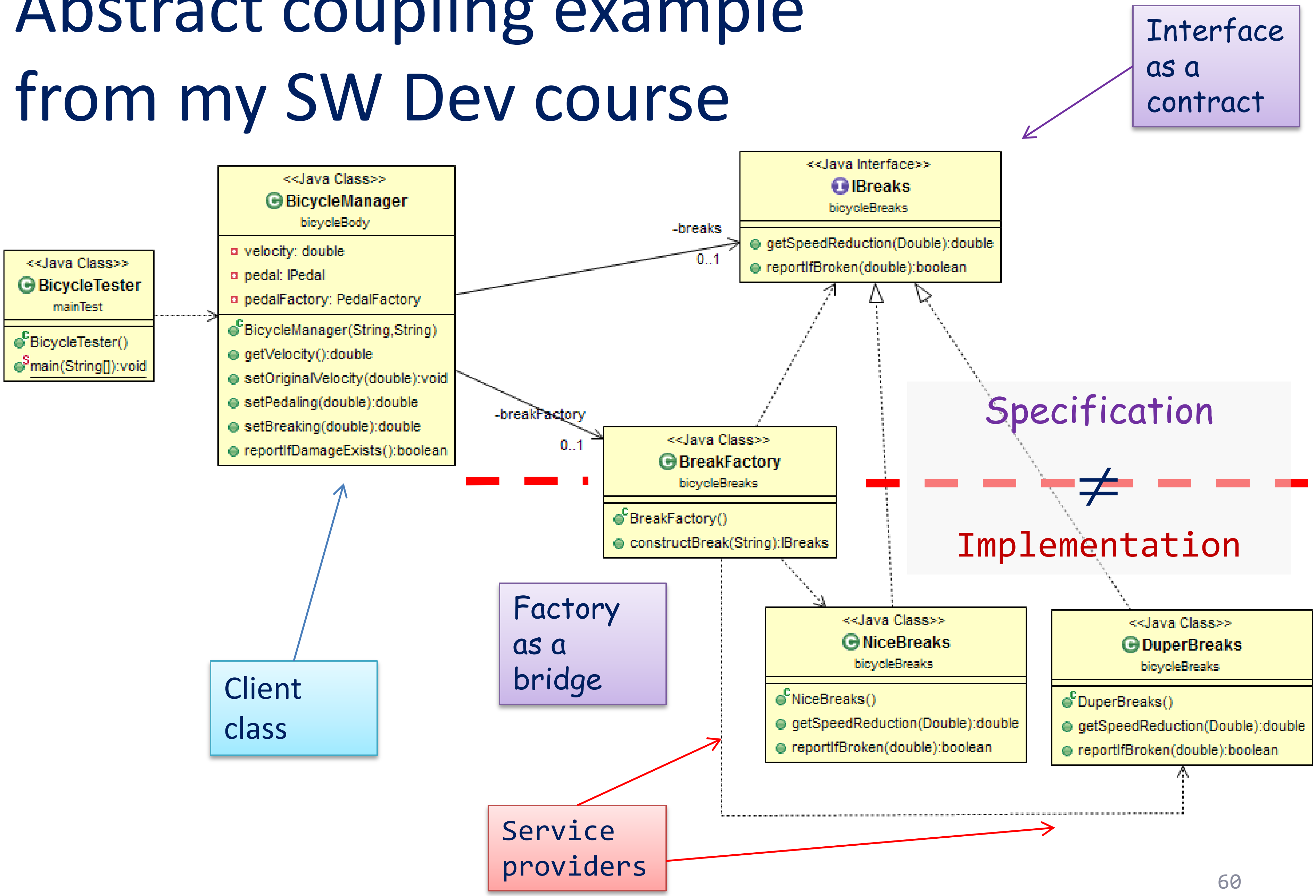
Why care for the "laws"/patterns of schema evolution?

- Scientific curiosity!
- Practical Impact: DB's are dependency magnets. Applications have to conform to the structure of the db…
  - typically, development waits till the "db backbone" is stable and applications are build on top of it
  - slight changes to the structure of a db can cause several (parts of) different applications to crash, causing the need for emergency repairing

… nowadays, to be complemented with API-based db access (Drupal)

```
function _profile_get_fields($category, $register = FALSE) {
  $query = db_select('profile_field');
  if ($register) {
   $query->condition('register', 1);
  }
  else {
   $query->condition('category', db_like($category), 'LIKE');
  }
  if (!user_access('administer users')) {
   $query->condition('visibility', PROFILE_HIDDEN, '<>');
  }
  return $query
    ->fields('profile_field')
    ->orderBy('category', 'ASC')
    ->orderBy('weight', 'ASC')
    ->execute();
}
```

Abstract coupling example from my SW Dev course



Datasets

https://github.com/DAINTINESS-Group/EvolutionDatasets

- Content management Systems
  - MediaWiki, TYPO3, Coppermine, phpBB, OpenCart
- Medical Databases
  - Ensemble, BioSQL
- Scientific
  - ATLAS Trigger

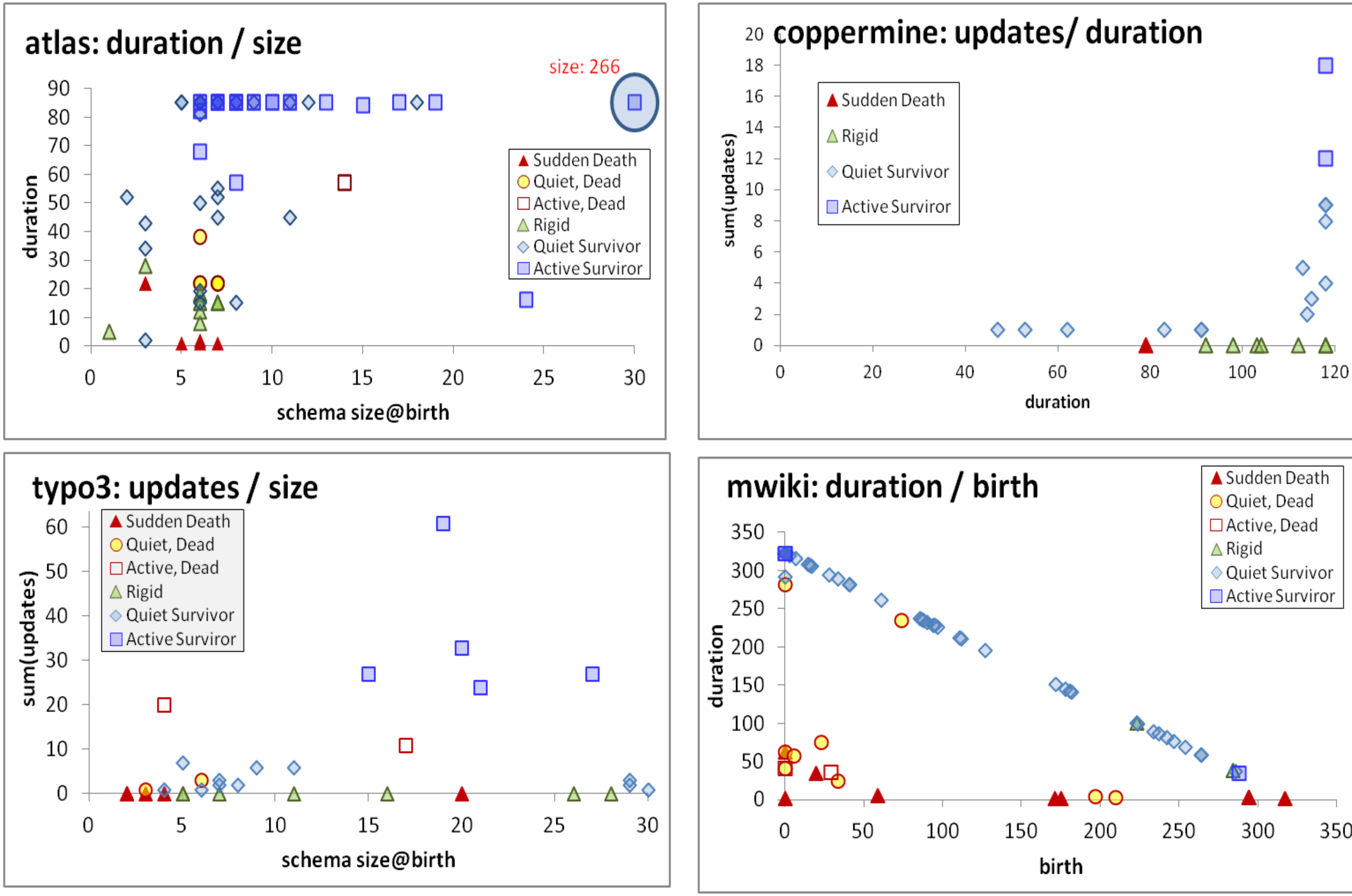
Data sets

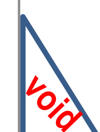
Dataset	Versions	Lifetime	Tables Start	Tables End	Attributes Start	Attributes End	Commits per Day	% commits with change	Repository URL
ATLAS Trigger	84	2 Y, 7 M, 2 D	56	73	709	858	0,089	82%	http://atdaq-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/Trigger/TrigConfiguration/TrigDb/share/sql/combined_schema.sql
BioSQL	46	10 Y, 6 M, 19 D	21	28	74	129	0,012	63%	https://github.com/biosql/biosql/blob/master/sql/biosqldb-mysql.sql
Coppermine	117	8 Y, 6 M, 2 D	8	22	87	169	0,038	50%	http://sourceforge.net/p/coppermine/code/8581/tree/trunk/cpg1.5.x/sql/schema.sql
Ensembl	528	13 Y, 3 M, 15 D	17	75	75	486	0,109	60%	http://cvs.sanger.ac.uk/cgi-bin/viewvc.cgi/ensembl/sql/table.sql?root=ensembl&view=log
MediaWiki	322	8 Y, 10 M, 6 D	17	50	100	318	0,100	59%	https://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/maintenance/tables.sql?view=log
OpenCart	164	4 Y, 4 M, 3 D	46	114	292	731	0,104	47%	https://github.com/opencart/opencart/blob/master/upload/install/opencart.sql
phpBB	133	6 Y, 7 M, 10 D	61	65	611	565	0,055	82%	https://github.com/phpbb/phpbb3/blob/develop/phpBB/install/schemas/mysql_41_schema.sql
TYPO3	97	8 Y, 11 M, 0 D	10	23	122	414	0,030	76%	https://git.typo3.org/Packages/TYPO3.CMS.git/history/TYPO3_6-0:/t3lib/stddb/tables.sql

Hecate: SQL schema diff extractor

- Parses DDL files
- Creates a model for the parsed SQL elements
- Compares two versions of the same schema
- Reports on the diff performed with a variety of metrics
- Exports the transitions that occurred in XML format

https://github.com/DAINTINESS-Group/Hecate





To probe further (code, data, details, presentations, …)

http://www.cs.uoi.gr/~pvassil/publications/2015_ER/

	Duration & Birth	*Schema size*
Activity	Top-changers (high ATU) are born early, live long, have large amt of update Inverse Γ : - Top-changers: mostly at long durations - Long duration: all kinds of change	 Comet: - Many updates: typically at medium schema size @ birth - Large schema at birth: medium amount of updates
Rigidity	Inverse Γ : - small duration → small change - medium duration → small or medium change	Comet: ~70% of tables ∈ 10x10 narrow & quiet box
Survival	Γ : the majority of wide tables are created early on and survive Heaven can wait for old-timers	Γ : if you 're wide, you survive
Death	Dead tables: quiet, early born, short-lived, and quite often all three of them	

SCOPE OF THE STUDY && VALIDITY CONSIDERATIONS

Data sets

Dataset	Versions	Lifetime	Tables Start	Tables End	Attributes Start	Attributes End	Commits per Day	% commits with change	Repository URL
ATLAS Trigger	84	2 Y, 7 M, 2 D	56	73	709	858	0,089	82%	http://atdaq-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/Trigger/TrigConfiguration/TrigDb/share/sql/combined_schema.sql
BioSQL	46	10 Y, 6 M, 19 D	21	28	74	129	0,012	63%	https://github.com/biosql/biosql/blob/master/sql/biosqldb-mysql.sql
Coppermine	117	8 Y, 6 M, 2 D	8	22	87	169	0,038	50%	http://sourceforge.net/p/coppermine/code/8581/tree/trunk/cpg1.5.x/sql/schema.sql
Ensembl	528	13 Y, 3 M, 15 D	17	75	75	486	0,109	60%	http://cvs.sanger.ac.uk/cgi-bin/viewvc.cgi/ensembl/sql/table.sql?root=ensembl&view=log
MediaWiki	322	8 Y, 10 M, 6 D	17	50	100	318	0,100	59%	https://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/maintenance/tables.sql?view=log
OpenCart	164	4 Y, 4 M, 3 D	46	114	292	731	0,104	47%	https://github.com/opencart/opencart/blob/master/upload/install/opencart.sql
phpBB	133	6 Y, 7 M, 10 D	61	65	611	565	0,055	82%	https://github.com/phpbb/phpbb3/blob/develop/phpBB/install/schemas/mysql_41_schema.sql
TYPO3	97	8 Y, 11 M, 0 D	10	23	122	414	0,030	76%	https://git.typo3.org/Packages/TYPO3.CMS.git/history/TYPO3_6-0:/t3lib/stddb/tables.sql

Scope of the study

- **Scope**:
  - databases being part of open-source software (and not proprietary ones)
  - long history
  - we work only with changes at the logical schema level (and ignore physical-level changes like index creation or change of storage engine)
- We encompass datasets with different domains ([A]: physics, [B]: biomedical, [C]: CMS's), amount of growth (shade: high, med, low) & schema size
- We should be very careful to not overgeneralize findings to proprietary databases or physical schemata!

FoSS Dataset	Versions	Lifetime	Tables @ Start	Tables @ End
ATLAS Trigger [A]	84	2 Y, 7 M, 2 D	56	73
BioSQL [B]	46	10 Y, 6 M, 19 D	21	28
Coppermine [C]	117	8 Y, 6 M, 2 D	8	22
Ensembl [B]	528	13 Y, 3 M, 15 D	17	75
MediaWiki [C]	322	8 Y, 10 M, 6 D	17	50
OpenCart [C]	164	4 Y, 4 M, 3 D	46	114
phpBB [C]	133	6 Y, 7 M, 10 D	61	65
TYPO3 [C]	97	8 Y, 11 M, 0 D	10	23

Can we generalize out findings broadly?

External validity

- We perform an **exploratory study to observe frequently occurring phenomena** within the scope of the aforementioned population
- **Are our data sets representative enough?** Is it possible that the observed behaviors are caused by sui-generis characteristics of the studied data sets?
  - Yes: we believe we have a good population definition & we abide by it
  - Yes: we believe we have a large number of databases, from a variety of domains with different profiles, that seem to give fairly consistent answers to our research questions (behavior deviations are mostly related to the maturity of the database and not to its application area).
  - Yes: we believe we have a good data extraction and measurement process without interference / selection / … of the input from our part
  - Maybe: unclear when the number of studied databases is large enough to declare the general application of a pattern as "universal".

Can we generalize out findings broadly?

External validity

- Understanding the represented population
  - Precision: all our data sets belong to the specified population
  - Definition Completeness: no missing property that we knowledgably omit to report
  - FoSS has an inherent way of maintenance and evolution
- Representativeness of selected datasets
  - Data sets come from 3 categories of FoSS (CMS / Biomedical / Physics)
  - They have different size and growth volumes
  - Results are fairly consistent both in our ER'15 and our CAiSE'14 papers
- Treatment of data
  - We have tested our "Delta Extractor", Hecate, to parse the input correctly & adapted it during its development; the parser is not a full-blown SQL parser, but robust to ignore parts unknown to it
  - A handful of cases where adapted in the Coppermine to avoid overcomplicating the parser; not a serious threat to validity ; other than that we have not interfered with the input
  - Fully automated counting for the measures via Hecate

GitHub, Inc. (US) https://github.com/DAINTINESS-Group

daintiness
Data-Intensive Information Ecosystems
Dept. of Comp. Science & Engineering University of Ioannina

DAta INTensive Information EcoSystemS Group

DAta INTensive Information EcoSystemS Group, Univ. Ioannina, Hellas

Ioannina, Greece

Repositories | People 7 | Teams 4 | Settings

Filters | Find a repository...

EvolutionDatasets
forked from giskou/EvolutionDatasets
Updated on 31 Jul

Hecate
forked from giskou/Hecate
Diff visualization between 2 SQL schemas
Updated on 2 Apr

Java ★ 0 ⑂ 4

Most importantly:
we are happy to invite you to reuse /test /assess /disprove /… all our code, data and results!

To probe further (code, data, results, …)

http://www.cs.uoi.gr/~pvassil/publications/2015_ER/

https://github.com/DAINTINESS-Group

Internal validity

- Can we confirm statements A=>B? No!
- Are there any spurious relationships? Maybe!

- Internal validity concerns the accuracy of cause-effect statements: “change in A => change in B”
- **We are very careful to avoid making strong causation statements!**
  - In some places, we just <u>hint</u> that we <u>suspect</u> the causes for a particular phenomenon, in some places in the text, but <u>we have no data, yet, to verify our gut-feeling</u>.
  - And yes, it is quite possible that our correlations hide cofounding variables.

Is there a theory?

- Our study should be regarded as a pattern observer, rather than as a collection of laws, coming with their internal mechanics and architecture.
- It will take too many studies (to enlarge the representativeness even more) and more controlled experiments (in-depth excavation of cause-effect relationships) to produce a solid theory.
- **It would be highly desirable if a clear set of requirements on the population definition, the breadth of study and the experimental protocol could be solidified by the scientific community (like e.g., the TREC benchmarks)**
- … and of course, there might be other suggestions on how to proceed…

RELATED WORK

Timeline of empirical studies

Timeline of empirical studies

Sjoberg @ IST 93: 18 months study of a health system.
139% increase of #tables ; 274% increase of the #attributes

Changes in the code (on avg):

- relation addition: 19 changes ; attribute additions: 2 changes
- relation deletion : 59.5 changes; attribute deletions: 3.25 changes

An **inflating period** during construction where almost all changes were additions, and a **subsequent period** where additions and deletions where balanced.

Timeline of empirical studies

Curino+ @ ICEIS08: Mediawiki for 4.5 years
100% increase in the number of tables
142% in the number of attributes.

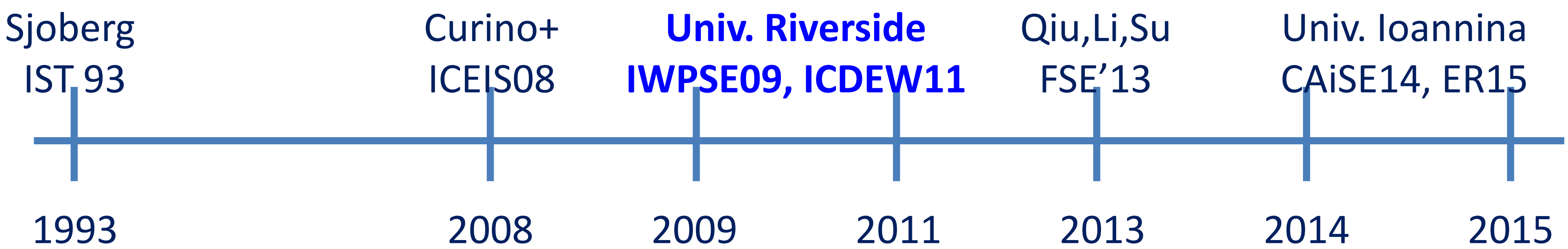
45% of changes do not affect the information capacity of the schema (but are rather index adjustments, documentation, etc)

1993	2008	2009	2011	2013	2014	2015
Sjoberg IST 93	**Curino+ ICEIS08**	Univ. Riverside IWPSE09, ICDEW11		Qiu,Li,Su FSE'13	Univ. Ioannina CAiSE14, ER15	

Timeline of empirical studies

IWPSE09: Mozilla and Monotone (a version control system)
Many ways to be out of synch between code and evolving db schema

ICDEW11: Firefox, Monotone , Biblioteq (catalogue man.) , Vienna (RSS)
Similar pct of changes with previous work
Frequency and timing analysis: **db schemata tend to stabilize over time**, as there is more change at the beginning of their history, but seem to converge to a relatively fixed structure later



Timeline of empirical studies

Qiu,Li,Su@ FSE 2013: 10 (!) database schemata studied.
Change is focused both (a) with respect to time and (b) with respect to the tables who change.

Timing: 7 out of 10 databases reached 60% of their schema size within 20% of their early lifetime.
Change is frequent in the early stages of the databases, with inflationary characteristics; then, the schema evolution process calms down.

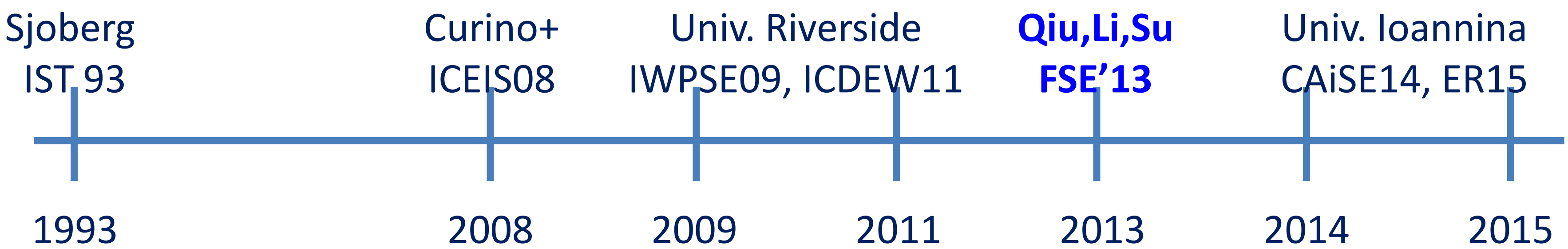
Tables that change: 40% of tables do not undergo any change at all, and 60%-90% of changes pertain to 20% of the tables (in other words, 80% of the tables live quiet lives). The most frequently modified tables attract 80% of the changes.

Timeline of empirical studies

Qiu,Li,Su@ FSE 2013: **Code and db co-evolution, not always in synch**.

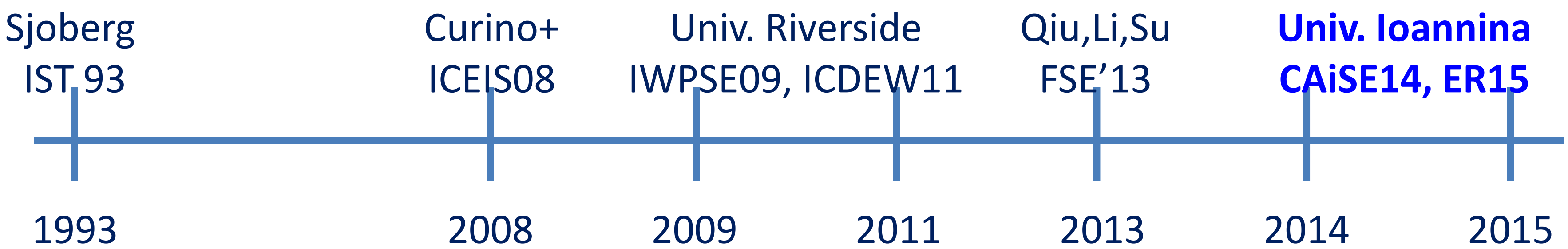
- Code and db changed in the same revision: 50.67% occasions
- Code change was in a previous/subsequent version than the one where the database schema change: 16.22% of occasions
- database changes not followed by code adaptation: 21.62% of occasions
- 11.49% of code changes were unrelated to the database evolution.

Each atomic change at the schema level is estimated to result in 10 -- 100 lines of application code been updated;
A valid db revision results in 100 -- 1000 lines of application code being updated



Timeline of empirical studies

CAiSE14: DB level
ER'15: Table level



STATS

Statistical study of durations

- Short and long lived tables are practically equally proportioned
- Medium size durations are fewer than the rest!
- Long lived tables are surprisingly too many
  - in half the data sets they are the most populated group
  - in all but one data set they exceed 30%

Normalized Durations and their pct over #tables

	# tables	Short Lived	Medium Lived	Long Lived	Long, not max	Max Duration
atlas	88	32%	14%	55%	5%	50%
biosql	45	31%	38%	31%	11%	20%
coppermine	23	0%	22%	78%	43%	35%
ensembl	155	55%	37%	8%	3%	5%
mwiki	71	46%	21%	32%	18%	14%
opencart	236	54%	9%	36%	36%	0%
phpBB	70	9%	10%	81%	0%	81%
typo3	32	34%	28%	38%	9%	28%
Overall	720	42%	20%	38%	18%	20%

Way too many long-lived tables live throughout the entire lifespan (Max Duration) of the database

Tables are mostly thin

- On average, half of the tables (approx. 47%) are thin tables with less than 5 attributes.
- The tables with 5 to 10 attributes are approximately one third of the tables' population
- The large tables with more than 10 attributes are approximately 17% of the tables.

Pct of tables with num. of attributes …

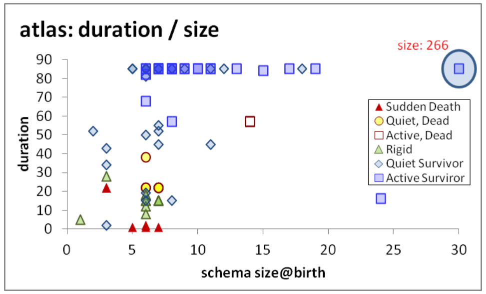
	<u><5</u>	<u>5-10</u>	<u>>10</u>
atlas	10,23%	68,18%	21,59%
biosql	75,56%	24,44%	0,00%
coppermine	52,17%	30,43%	17,39%
ensembl	54,84%	38,06%	7,10%
mediawiki	61,97%	19,72%	18,31%
phpbb	40,00%	44,29%	15,71%
typo3	21,88%	31,25%	46,88%
opencart	57,20%	33,05%	9,75%
Average	46,73%	36,18%	17,09%

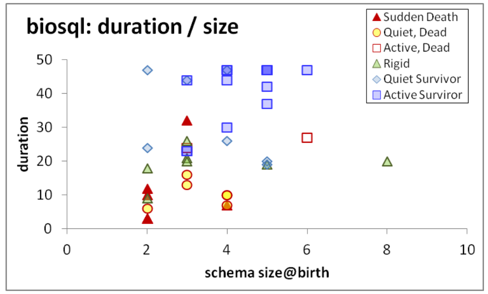
THE FOUR PATTERNS

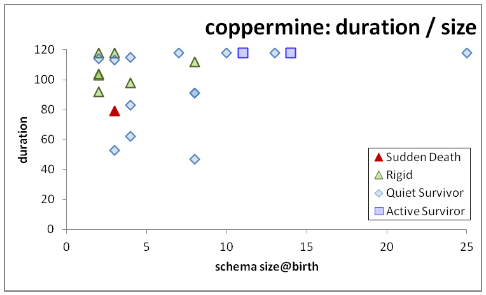
Schema size @ birth / duration

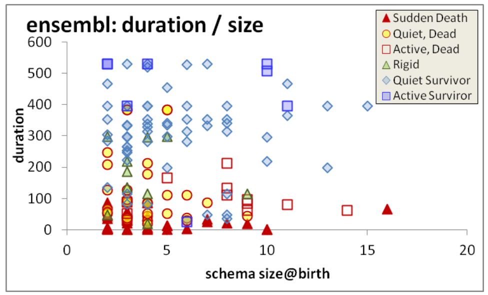
If you ‘re wide, you survive
a.k.a (only the thin die young, ~~all~~ the wide ones seem to live forever)

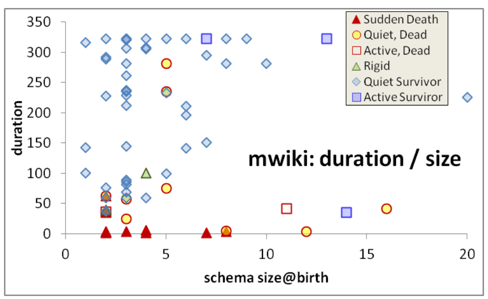
THE GAMMA PATTERN

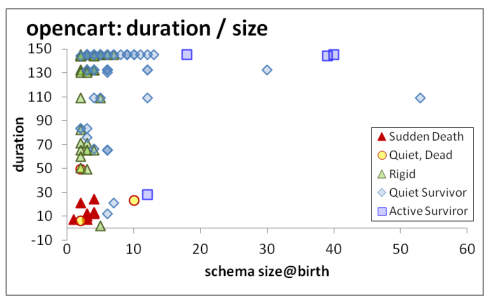


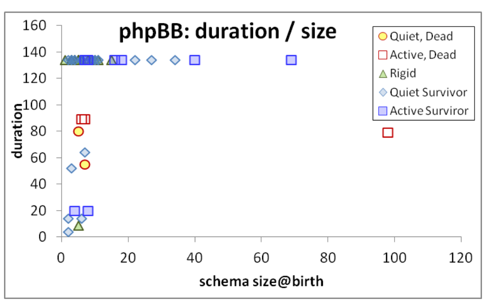


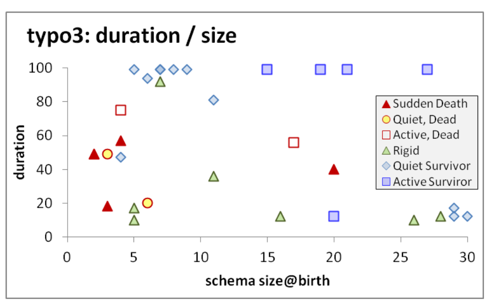












Exceptions

- Biosql: nobody exceeds 10 attributes
- Ensembl, mwiki: very few exceed 10 attributes, 3 of them died
- typo: has many late born survivors

Stats on wide tables and their survival

	# Tables	# Wide tables	As pct over #Tables... ...Wide	...Wide of long duration	As pct over the set of Wide Tables ... ... Survivors	... Early Born & Survivors	... of Long Duration
coppermine	23	4	17%	17%	100%	100%	100%
phpBB	70	11	16%	14%	91%	91%	91%
opencart*	128	12	9%	7%	100%	75%	75%
atlas	88	14	16%	11%	86%	71%	71%
typo3	32	15	47%	13%	87%	33%	27%
mwiki	71	6	8%	1%	50%	33%	17%
ensembl	155	9	6%	0%	67%	56%	0%
biosql	45	0	0%	0%	NA	NA	NA

Definitions:

Wide schema: strictly above 10 attributes.

The top band of durations (the upper part of the Gamma shape): the upper 10% of the values in the y-axis.

Early born table: ts birth version is in the lowest 33% of versions;

Late-comers: born after the 77% of the number of versions.

Whenever a table is wide, its chances of surviving are high

	# Tables	# Wide tables	*As pct over #Tables...* ...Wide	...Wide of long duration	*As pct over the set of Wide Tables ...* ... Survivors	... Early Born & Survivors	... of Long Duration
coppermine	23	4	17%	17%	**100%**	100%	100%
phpBB	70	11	16%	14%	**91%**	91%	91%
opencart*	128	12	9%	7%	**100%**	75%	75%
atlas	88	14	16%	11%	**86%**	71%	71%
typo3	32	15	47%	13%	**87%**	33%	27%
mwiki	71	6	8%	1%	50%	33%	17%
ensembl	155	9	6%	0%	67%	56%	0%
biosql	45	0	0%	0%	NA	NA	NA

Apart from mwiki and ensembl, all the rest of the data sets *confirm the hypothesis with a percentage higher than 85%*. The two exceptions are as high as 50% for their support to the hypothesis.

Wide tables are frequently created early on and are not deleted afterwards

			As pct over #Tables…		*As pct over the set of Wide Tables …*		
	# Tables	# Wide tables	…Wide	…Wide of long duration	… Survivors	… Early Born & Survivors	… of Long Duration
coppermine	23	4	17%	17%	100%	100%	100%
phpBB	70	11	16%	14%	91%	91%	91%
opencart*	128	12	9%	7%	100%	75%	75%
atlas	88	14	16%	11%	86%	71%	71%
typo3	32	15	47%	13%	87%	33%	27%
mwiki	71	6	8%	1%	50%	33%	17%
ensembl	155	9	6%	0%	67%	56%	0%
biosql	45	0	0%	0%	NA	NA	NA

Early born, wide, survivor tables (as a percentage over the set of wide tables).
- in half the data sets the percentage is above 70%
- in two of them the percentage of these tables is one third of the wide tables.

Whenever a table is wide, its duration frequently lies within the top-band of durations (upper part of Gamma)

	# Tables	# Wide tables	*As pct over #Tables...* ...Wide	...Wide of long duration	*As pct over the set of Wide Tables ...* ... Survivors	... Early Born & Survivors	... of Long Duration
coppermine	23	4	17%	17%	100%	100%	100%
phpBB	70	11	16%	14%	91%	91%	91%
opencart*	128	12	9%	7%	100%	75%	75%
atlas	88	14	16%	11%	86%	71%	71%
typo3	32	15	47%	13%	87%	33%	27%
mwiki	71	6	8%	1%	50%	33%	17%
ensembl	155	9	6%	0%	67%	56%	0%
biosql	45	0	0%	0%	NA	NA	NA

What is probability that a wide table belongs to the upper part of the Gamma?

- there is a very strong correlation between the two last columns: the Pearson correlation is 88% overall; 100% for the datasets with high pct of early born wide tables.
-
- *Bipolarity on this pattern: half the cases support the pattern with support higher than 70%, whereas the rest of the cases clearly disprove it, with very low support values.*

Long-lived & wide => early born and survivor

	# Tables	# Wide tables	*As pct over #Tables...* ...Wide	...Wide of long duration	*As pct over the set of Wide Tables ...* ... Survivors	... Early Born & Survivors	... of Long Duration
coppermine	23	4	17%	17%	100%	100%	100%
phpBB	70	11	16%	14%	91%	91%	91%
opencart*	128	12	9%	7%	100%	75%	75%
atlas	88	14	16%	11%	86%	71%	71%
typo3	32	15	47%	13%	87%	33%	27%
mwiki	71	6	8%	1%	50%	33%	17%
ensembl	155	9	6%	0%	67%	56%	0%
biosql	45	0	0%	0%	NA	NA	NA

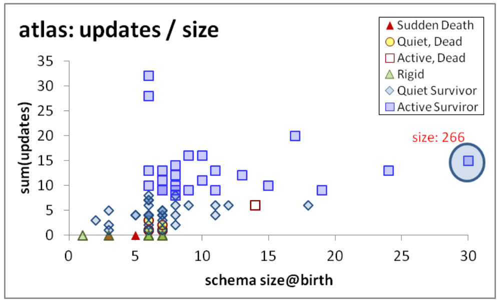
Subset relationship

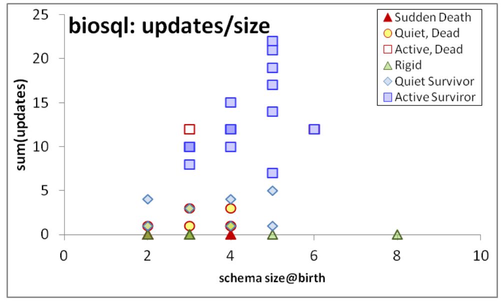
In all data sets, if a wide table has a long duration within the upper part of the Gamma, this deterministically (100% of all data sets) signifies that the table was also early born and survivor.

If a wide table is in the top of the Gamma line, it is deterministically an early born survivor.

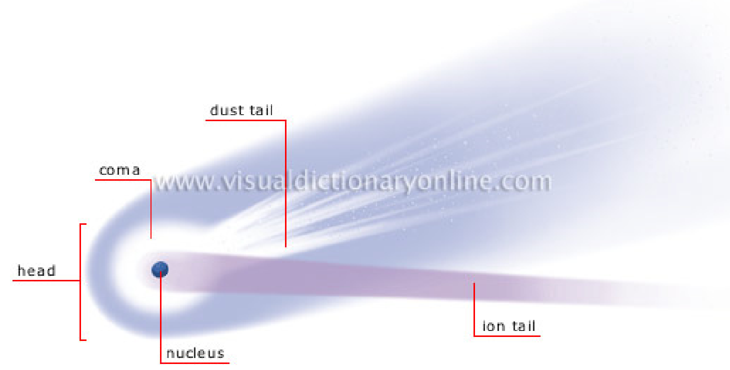
Schema size and updates

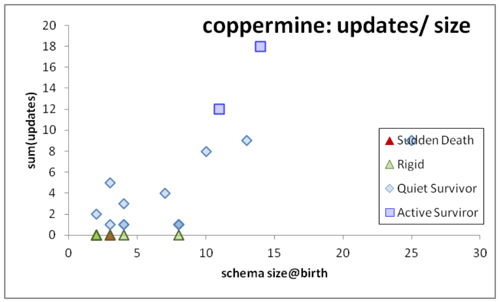
THE COMET PATTERN

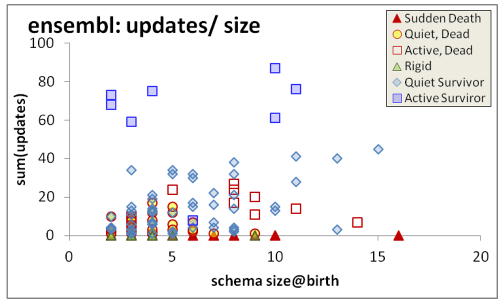




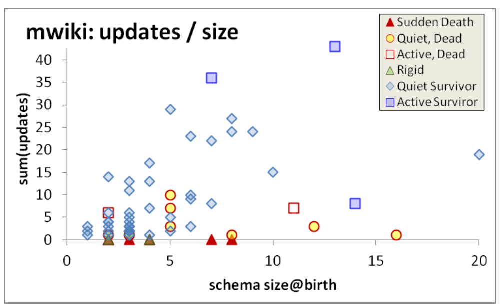
http://visual.merriam-webster.com/astronomy/celestial-bodies/comet.php

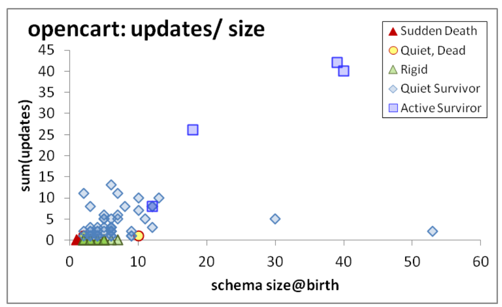


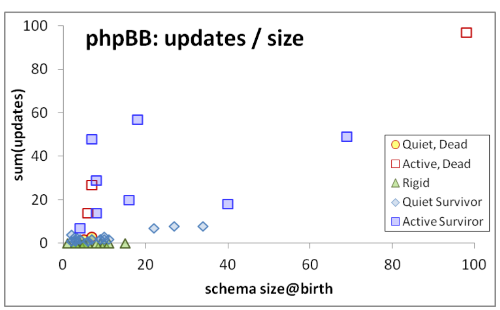


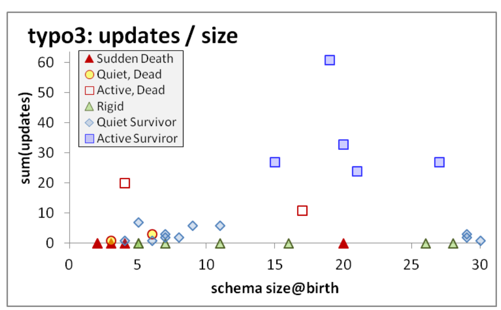


http://spaceplace.nasa.gov/comet-nucleus/en/









Statistics of schema size at birth and sum of updates

		Schema size at birth					Sum of updates				
	#tables	max	mean (μ)	stdev (σ)	median	mode	max	mean (μ)	stdev (σ)	median	mode
atlas--	87 / 88	24	7.53	3.67	7	6	32	5.86	11.81	4	0
biosql	45	8	3.6	1.37	3	2	22	5.38	11.91	1	0
coppermine	23	25	6.52	5.35	4	2	18	3.3	7.98	1	0
ensembl	155	16	4.98	2.98	4	3	87	10.38	27.05	3	0
mwiki	71	20	4.79	3.64	3	3	43	6.92	16.03	3	0
ocart*	128	53	5.73	7.02	4	3	42	2.56	8.56	0	0
phpBB	70	98	9.39	14.63	5	3	97	6.33	22.17	0.5	0
typo3	32	30	12.69	9.26	8.5	4	61	7.53	20.89	1.5	0

/ atlas: excluded table l1_prescale_set from the analysis (266 attributes; second largest value: 24); open cart: after version 22*/*

Typically: ~70% of tables inside the box

		In the box		Out of the box	
	#tables	count	pct	count	pct
atlas--	88	62	70%	26	30%
biosql	45	31	69%	14	31%
coppermine	23	18	78%	5	22%
ensembl	155	100	65%	55	35%
mwiki	71	50	70%	21	30%
ocart*	128	110	86%	18	14%
phpBB	70	51	73%	19	27%
typo3	32	16	50%	16	50%

/ atlas: excluded table l1_prescale_set from the analysis (266 attributes; second largest value: 24); open cart: after version 22*/*

Typically, around 70% of the tables of a database is found within the 10x10 box of *schemaSize@birth* x *sumOfUpdates* (10 excluded in both axes).

Top changers tend to have medium schema sizes

Schema size @ birth. Statistics for ...		... the entire data set				... the top changers		
	#tables	max	mean (μ)	stdev (σ)	μ+σ	avg sc. size for top 5%	sc. size of top 1	avg top 5% / max
atlas--	87	24	7.53	3.67	11.20	9.60	6	0.40
biosql	45	8	3.60	1.37	4.97	5.00	5	0.63
coppermine	23	25	6.52	5.35	11.87	12.50	14	0.50
ensembl	155	16	4.98	2.98	7.97	7.13	10	0.45
mwiki	71	20	4.79	3.64	8.43	8.25	13	0.41
ocart*	128	53	5.73	7.02	12.74	17.43	39	0.33
phpBB	70	98	9.39	14.63	24.02	48.00	98	0.49
typo3	32	30	12.69	9.26	21.95	19.50	19	0.65
Pearson with avg top 5%		***0.96***	***0.58***	***0.97***	***0.87***		***0.97***	

/ atlas: excluded table l1_prescale_set from the analysis (266 attributes; second largest value: 24); open cart: after version 22*/*

For every dataset: we selected the top 5% of tables in terms of this sum of updates and we averaged the schema size at birth of these top 5% tables.

Top changers tend to have medium schema sizes

Schema size @ birth. Statistics for ...		... the entire data set				... the top changers		
	#tables	max	mean (μ)	stdev (σ)	μ+σ	avg sc. size for top 5%	sc. size of top 1	avg top 5% / max
atlas--	87	24	7.53	3.67	11.20	9.60	6	0.40
biosql	45	8	3.60	1.37	4.97	5.00	5	0.63
coppermine	23	25	6.52	5.35	11.87	12.50	14	0.50
ensembl	155	16	4.98	2.98	7.97	7.13	10	0.45
mwiki	71	20	4.79	3.64	8.43	8.25	13	0.41
ocart*	128	53	5.73	7.02	12.74	17.43	39	0.33
phpBB	70	98	9.39	14.63	24.02	48.00	98	0.49
typo3	32	30	12.69	9.26	21.95	19.50	19	0.65
Pearson with avg top 5%		***0.96***	*0.58*	***0.97***	***0.87***		***0.97***	

/ atlas: excluded table l1_prescale_set from the analysis (266 attributes; second largest value: 24); open cart: after version 22*/*

The average schema size for the top 5% of tables in terms of their update behavior is close to one standard deviation up from the average value of the schema size at birth(i.e., very close to mu+$sigma$). //*except phpBB*

Top changers tend to have medium schema sizes

Schema size @ birth. Statistics for ...		... the entire data set				... the top changers		
	#tables	max	mean (μ)	stdev (σ)	μ+σ	avg sc. size for top 5%	sc. size of top 1	avg top 5% / max
atlas--	87	24	7.53	3.67	11.20	9.60	6	0.40
biosql	45	8	3.60	1.37	4.97	5.00	5	0.63
coppermine	23	25	6.52	5.35	11.87	12.50	14	0.50
ensembl	155	16	4.98	2.98	7.97	7.13	10	0.45
mwiki	71	20	4.79	3.64	8.43	8.25	13	0.41
ocart*	128	53	5.73	7.02	12.74	17.43	39	0.33
phpBB	70	98	9.39	14.63	24.02	48.00	98	0.49
typo3	32	30	12.69	9.26	21.95	19.50	19	0.65
Pearson with avg top 5%		***0.96***	***0.58***	***0.97***	***0.87***		***0.97***	

/ atlas: excluded table l1_prescale_set from the analysis (266 attributes; second largest value: 24); open cart: after version 22*/*

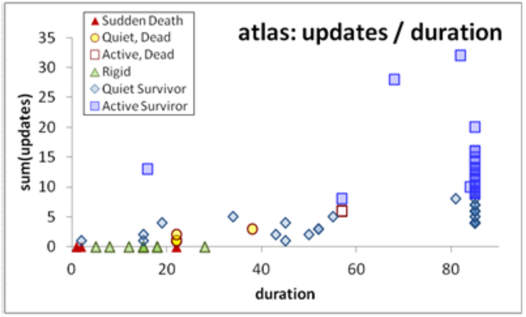
- In 5 out of 8 cases, the average schema size of top-changers within 0.4 and 0.5 of the maximum value (practically the middle of the domain) and never above 0.65 of it.
- Pearson: the maximum value, the standard deviation of the entire data set and the average of the top changers are very strongly correlated.

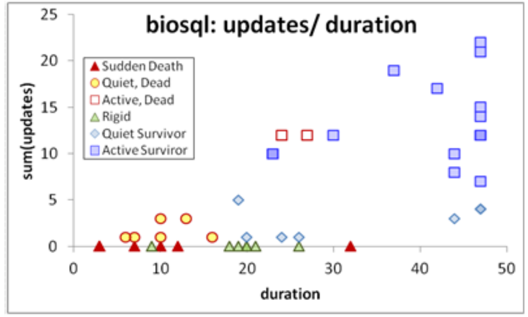
Wide tables have a medium number of updates

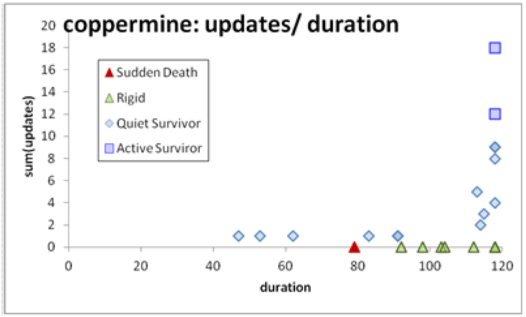
Total amt. of updates. Statistics for ...	... the entire data set						... the top 5% with respect to schema size at birth (top wide)			
	#tables	max	mean (μ)	stdev (σ)	μ+σ	max/2	avg upd. of top 5%	upd. of top 1	avg of top 5% / max	Top up. in wide?
atlas	88	32	5.86	11.81	11.81	16.0	**12.60**	20	0.39	N
biosql	45	22	5.38	11.91	11.91	11.0	**8.00**	0	0.36	N
coppermine	23	18	3.30	7.98	7.98	9.0	**13.50**	9	0.75	Y
ensembl	155	87	10.38	27.05	27.05	43.5	**28.22**	0	0.32	N
mwiki	71	43	6.92	16.03	16.03	21.5	**17.75**	19	0.41	Y
ocart*	128	42	2.56	8.56	8.561	21.0	**14.55**	2	0.35	Y
phpBB	70	97	6.33	22.17	22.17	48.5	**43.00**	97	0.44	Y!
typo3	32	61	7.53	20.89	20.89	30.5	**2.00**	1	0.03	N
Pearson with avg top 5%			*0.27*	*0.59*	*0.50*	*0.74*			*0.79*	

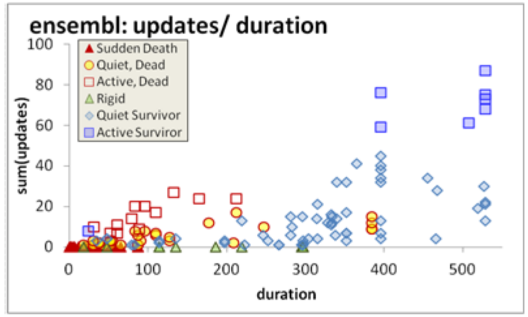
For each data set, we took the top 5% in terms of schema size at birth (**top wide**) and contrasted their update behavior wrt the update behavior of the entire data set.
Typically, the avg. number of updates of the top wide tables is close to the 50% of the domain of values for the sum of updates (i.e., the middle of the y-axis of the comet figure, measuring the sum of updates for each table).
This is mainly due to the (very) large standard deviation (twice the mean), rather than the --typically low -- mean value (due to the large part of the population living quiet lives).

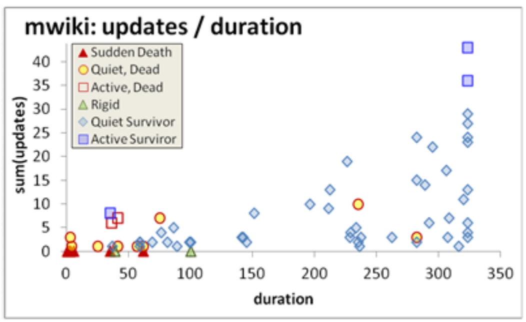
INVERSE GAMMA

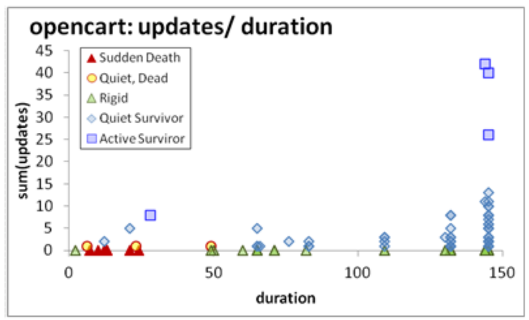

atlas: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

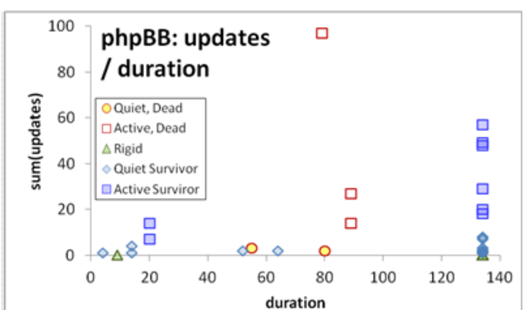

biosql: updates/ duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

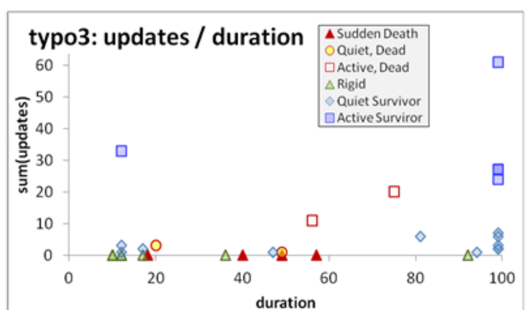

coppermine: updates/ duration
Sudden Death
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration


ensembl: updates/ duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

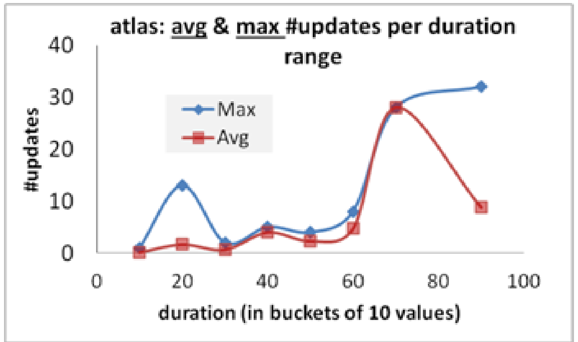

mwiki: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

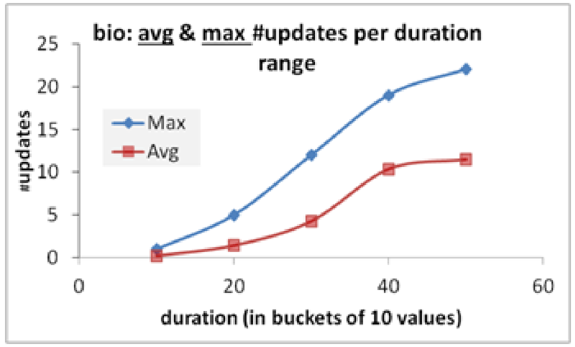

opencart: updates/ duration
Sudden Death
Quiet, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

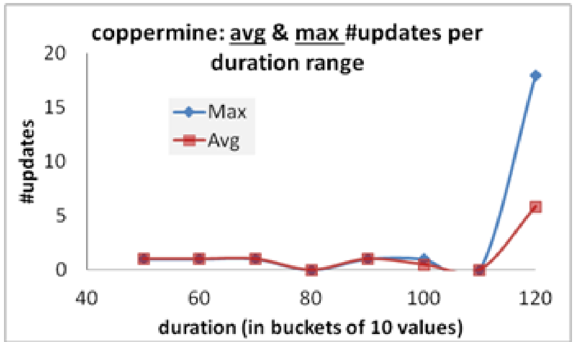

phpBB: updates
/ duration
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

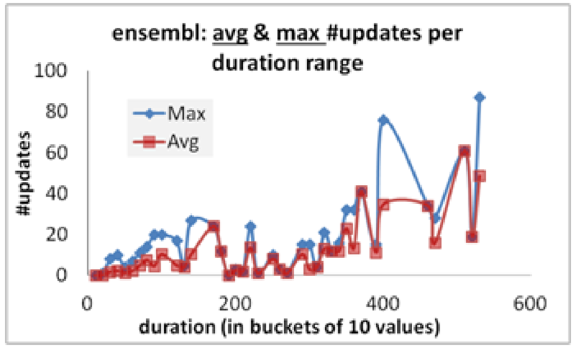

typo3: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

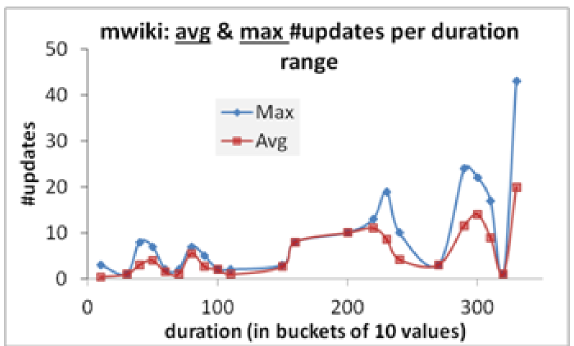
Skyline & Avg for Inverse Gamma

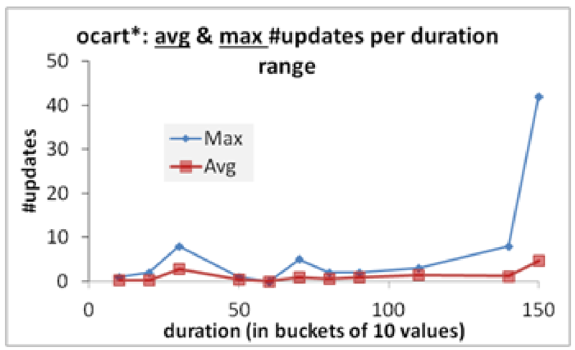


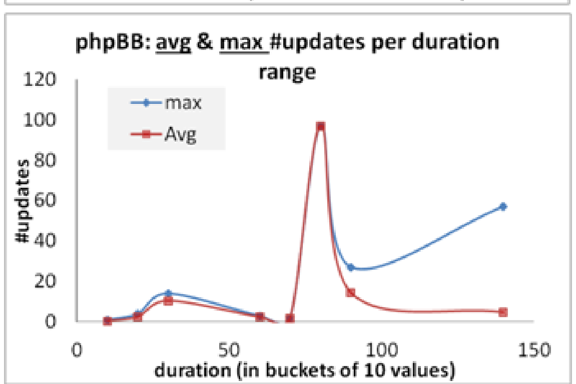


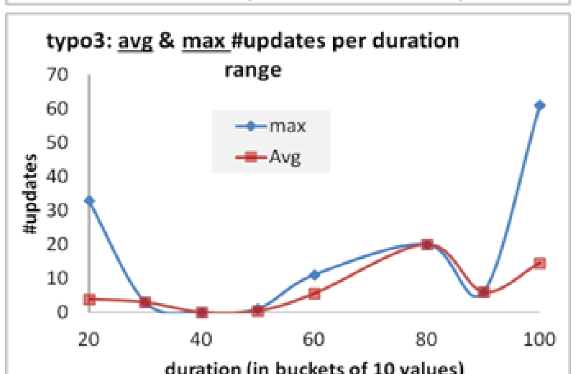




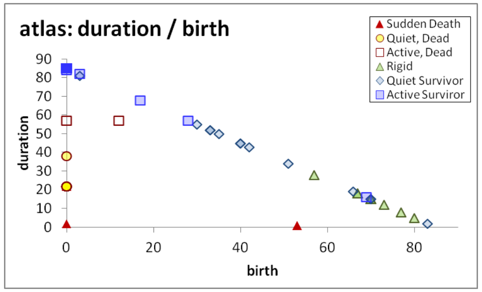


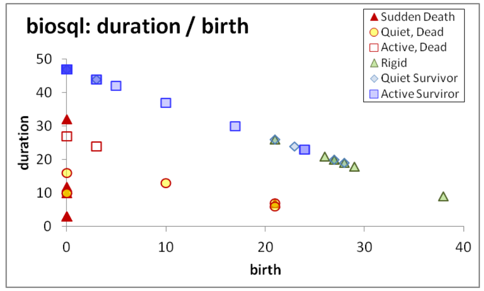


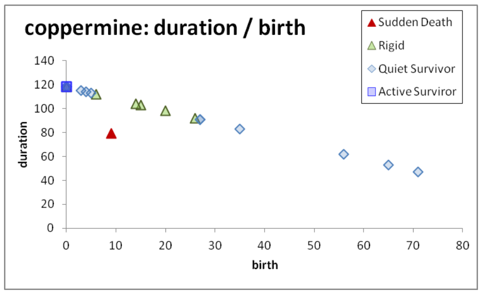


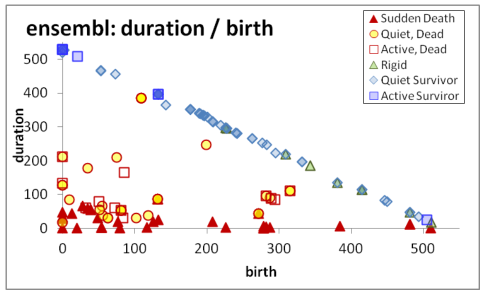


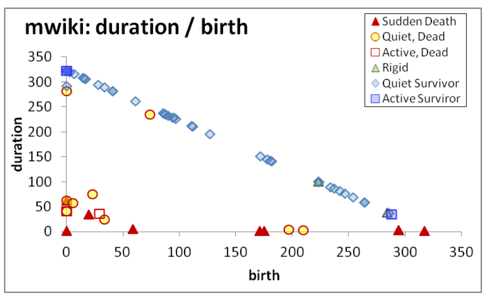
THE EMPTY TRIANGLE PATTERN

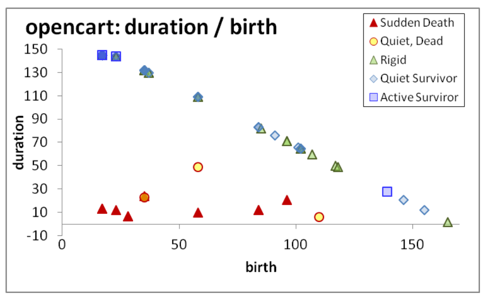

atlas: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

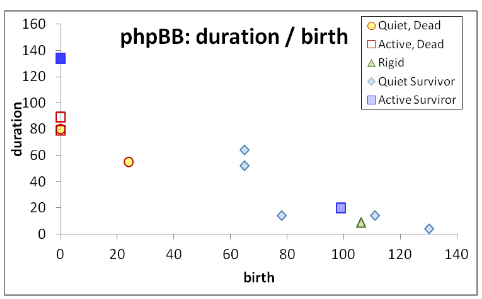

biosql: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

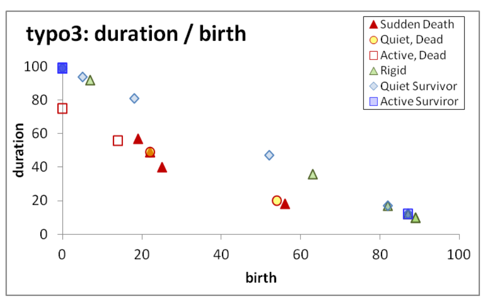

coppermine: duration / birth
Sudden Death
Rigid
Quiet Survivor
Active Surviror
duration
birth

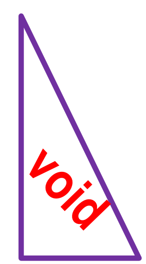

ensembl: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


mwiki: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


opencart: duration / birth
Sudden Death
Quiet, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


phpBB: duration / birth
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


typo3: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


void

Top changers: early born, survivors, often with long durations, and often all the above

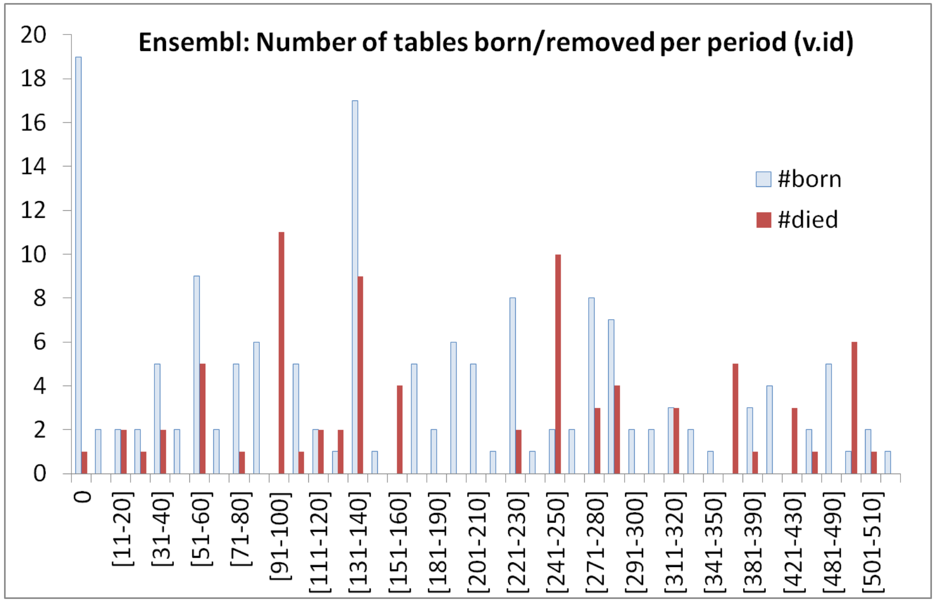
	atlas	biosql	coppermine	ensembl	mwiki	ocart*	phpBB	typo3
Tables	88	45	23	155	71	128	70	32
Active	27	16	2	23	5	4	11	7
active tables(%)	31%	36%	9%	15%	7%	3%	16%	22%
As percentages over active								
Born early	96%	81%	100%	78%	80%	75%	82%	86%
Survivors	93%	88%	100%	48%	60%	100%	73%	71%
Long duration	85%	69%	100%	22%	40%	75%	55%	57%
Born early, survive, live long	85%	69%	100%	22%	40%	75%	55%	57%

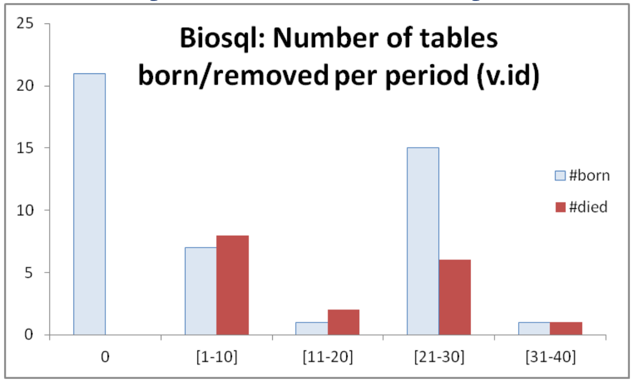
- In all data sets, active tables are born early with percentages that exceed 75%
- With the exceptions of two data sets, they survive with percentage higher than 70%.
- The probability of having a long duration is higher than 50% in 6 out of 8 data sets.
- Interestingly, **the two last lines are exactly the same sets of tables in all data sets!**
    - An active table with long duration has been born early and survived with prob. 100%
    - An active, survivor table that has a long duration has been born early with prob. 100%

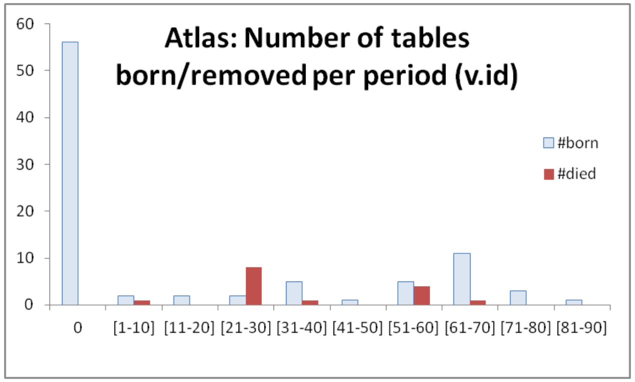
Dead are: quiet, early born, short lived, and quite often all three of them

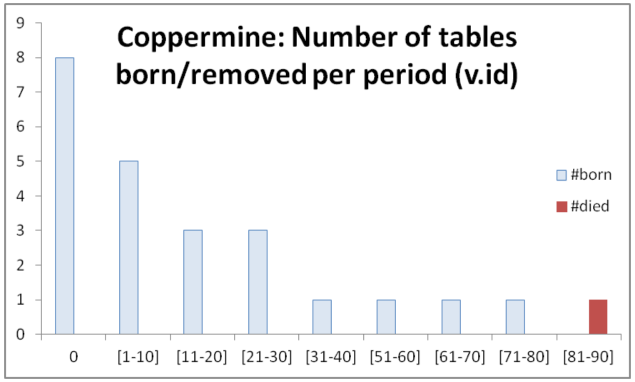
	atlas	biosql	coppermine	ensembl	mwiki	ocart*	phpBB	typo3
tables	88	45	23	155	71	128	70	32
dead	15	17	1	80	21	14	5	9
dead tables(%)	17%	38%	4%	52%	30%	11%	7%	28%
As percentages over # dead								
Few updates	87%	88%	100%	85%	90%	100%	40%	78%
Early born	80%	82%	100%	70%	62%	71%	100%	78%
Short-lived	80%	76%	0%	89%	90%	100%	0%	22%
Few upd's, early born, short duration	60%	59%	0%	51%	43%	71%	0%	0%
Do tables die of old age?								
long durations	48	14	18	13	23	86	57	12
long durations, dead	0	0	0	0	1	0	0	0
Dead among long-lived (%)	0%	0%	0%	0%	4%	0%	0%	0%

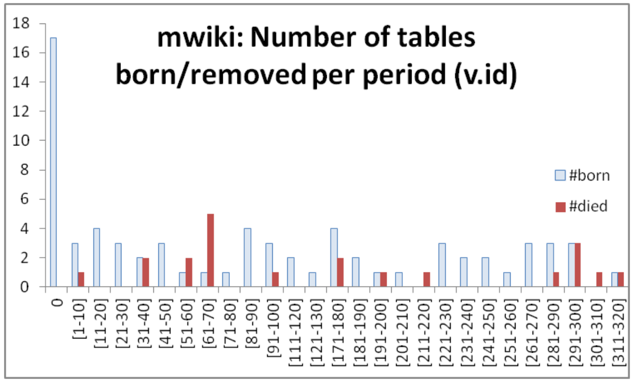
Most births &deaths occur early (usually)

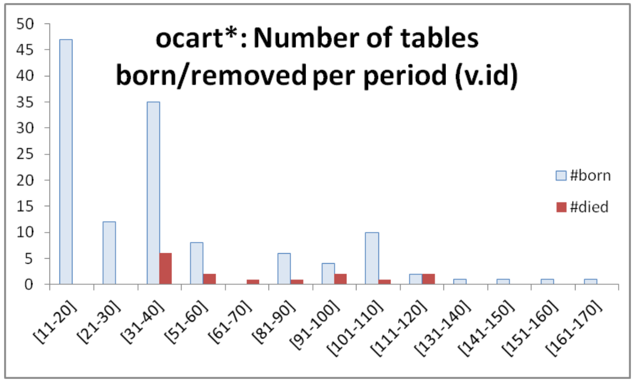


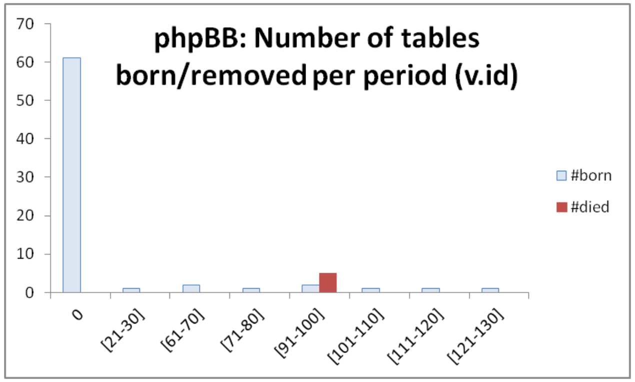


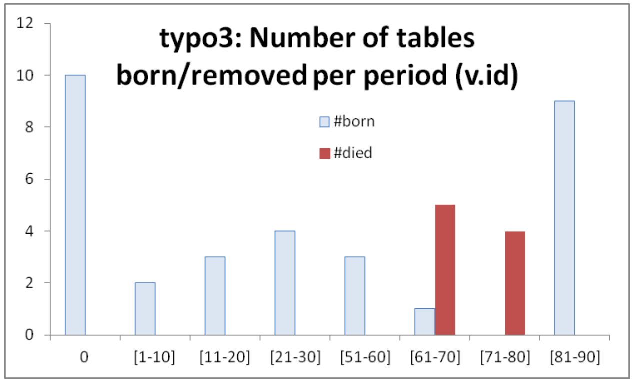


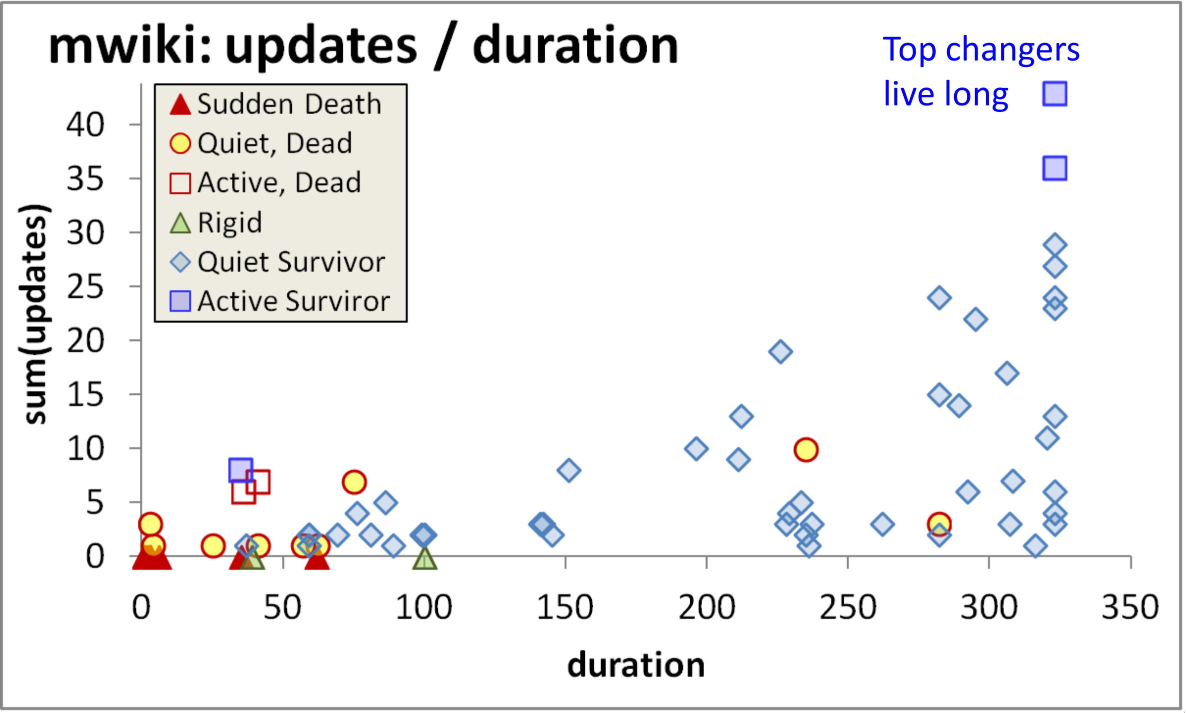


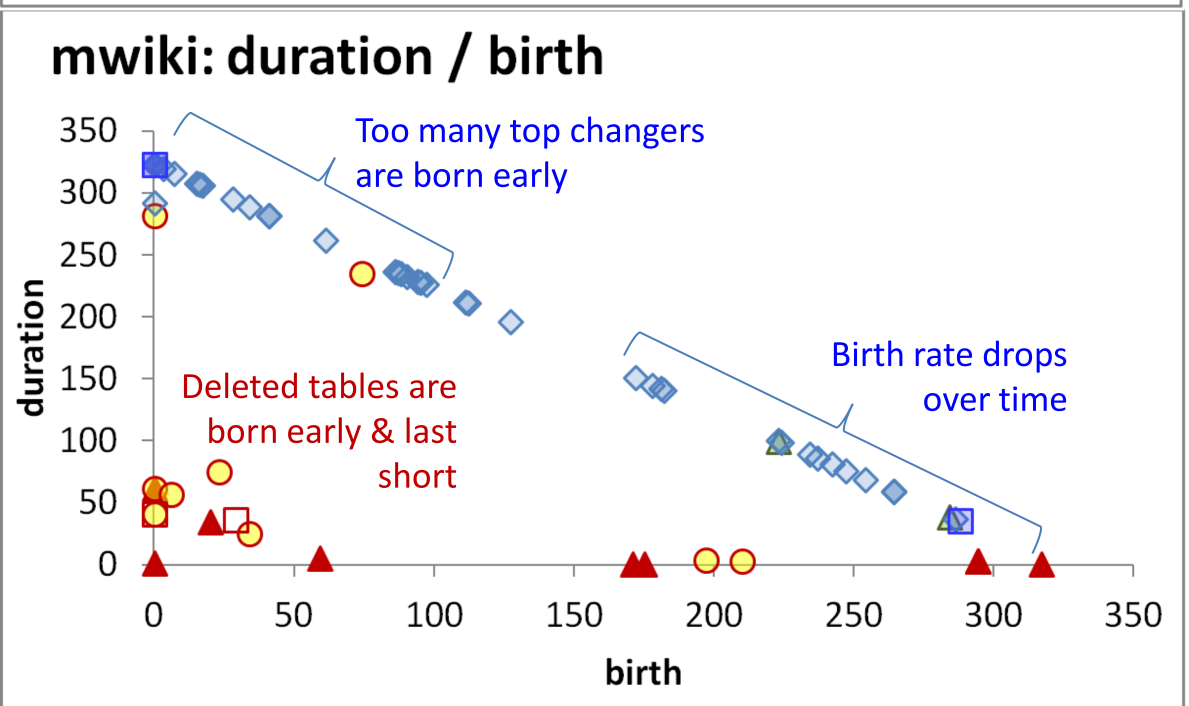






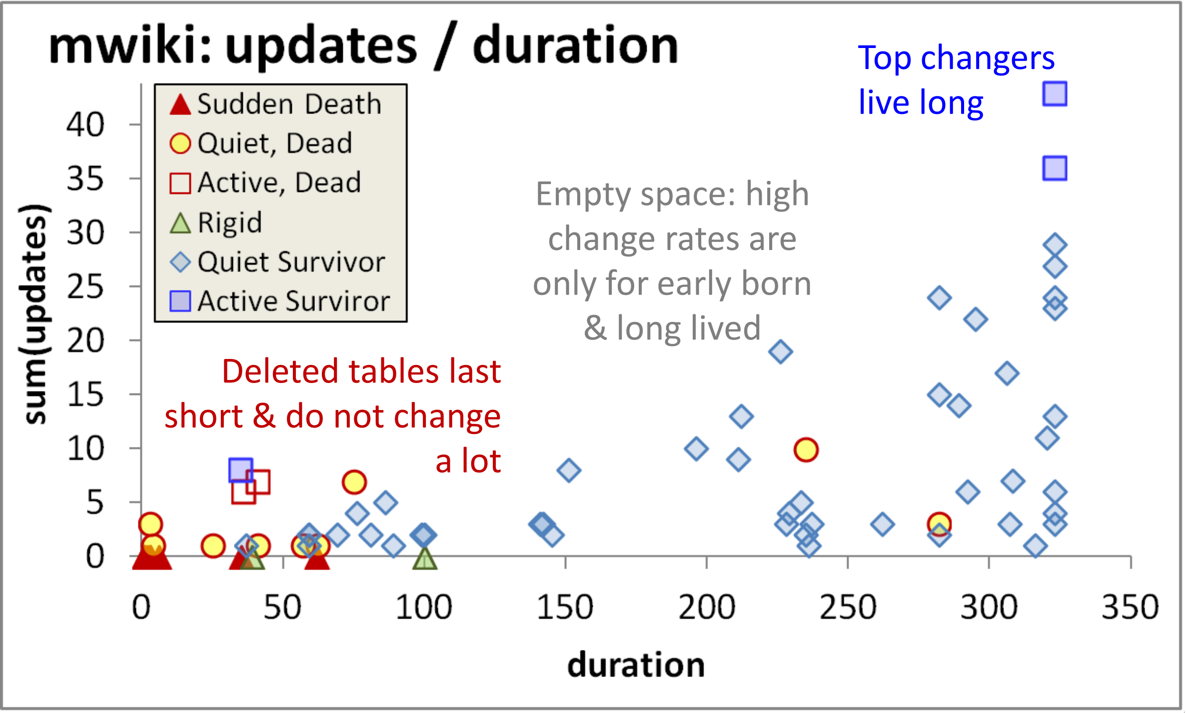


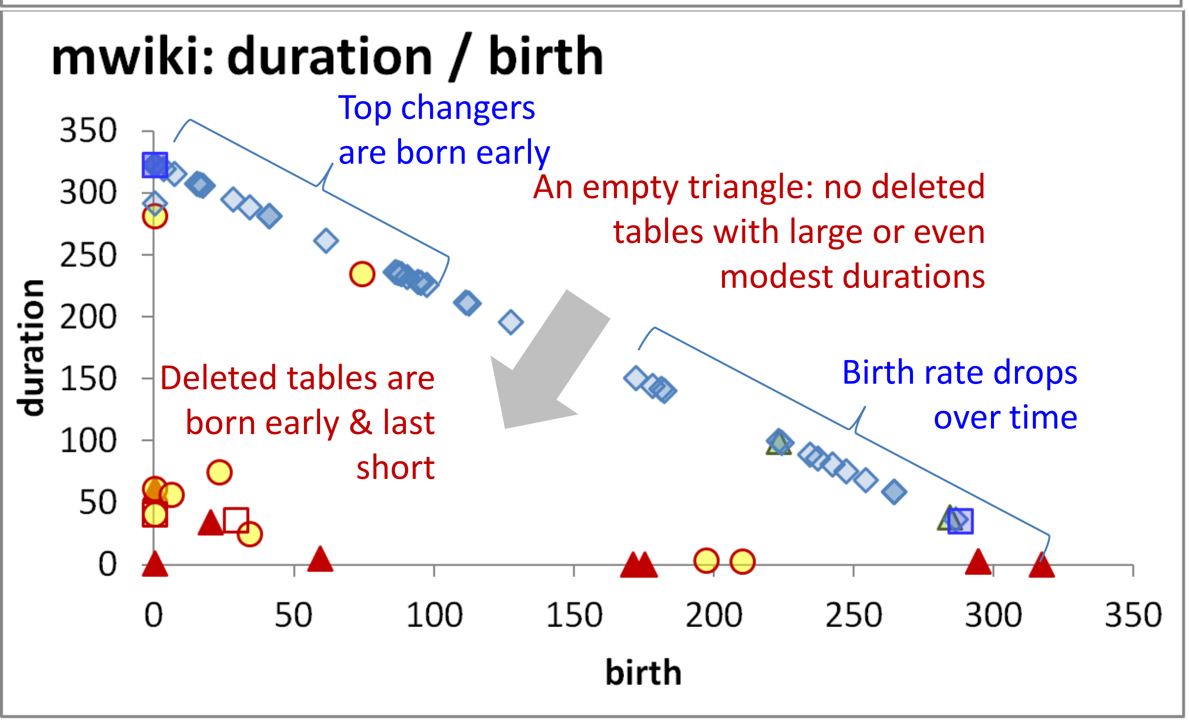




Longevity and update activity correlate !!

- Remember: top changers are defined as such wrt ATU (AvgTrxnUpdate), not wrt sum(changes)
- Still, they dominate the sum(updates) too! (see top of inverse Γ)
- See also upper right blue part of diagonal: too many of them are born early and survive => live long!





All in one

- Early stages of the database life are more "active" in terms of births, deaths and updates, and have higher chances of producing deleted tables.
- After the first major restructuring, the database continues to grow; however, we see much less removals, and maintenance activity becomes more concentrated and focused.

Roadmap

- Evolution of views
- Data warehouse Evolution
- A case study (if time)
- **Impact assessment in ecosystems**
- Empirical studies concerning database evolution
- Open Issues and discussions

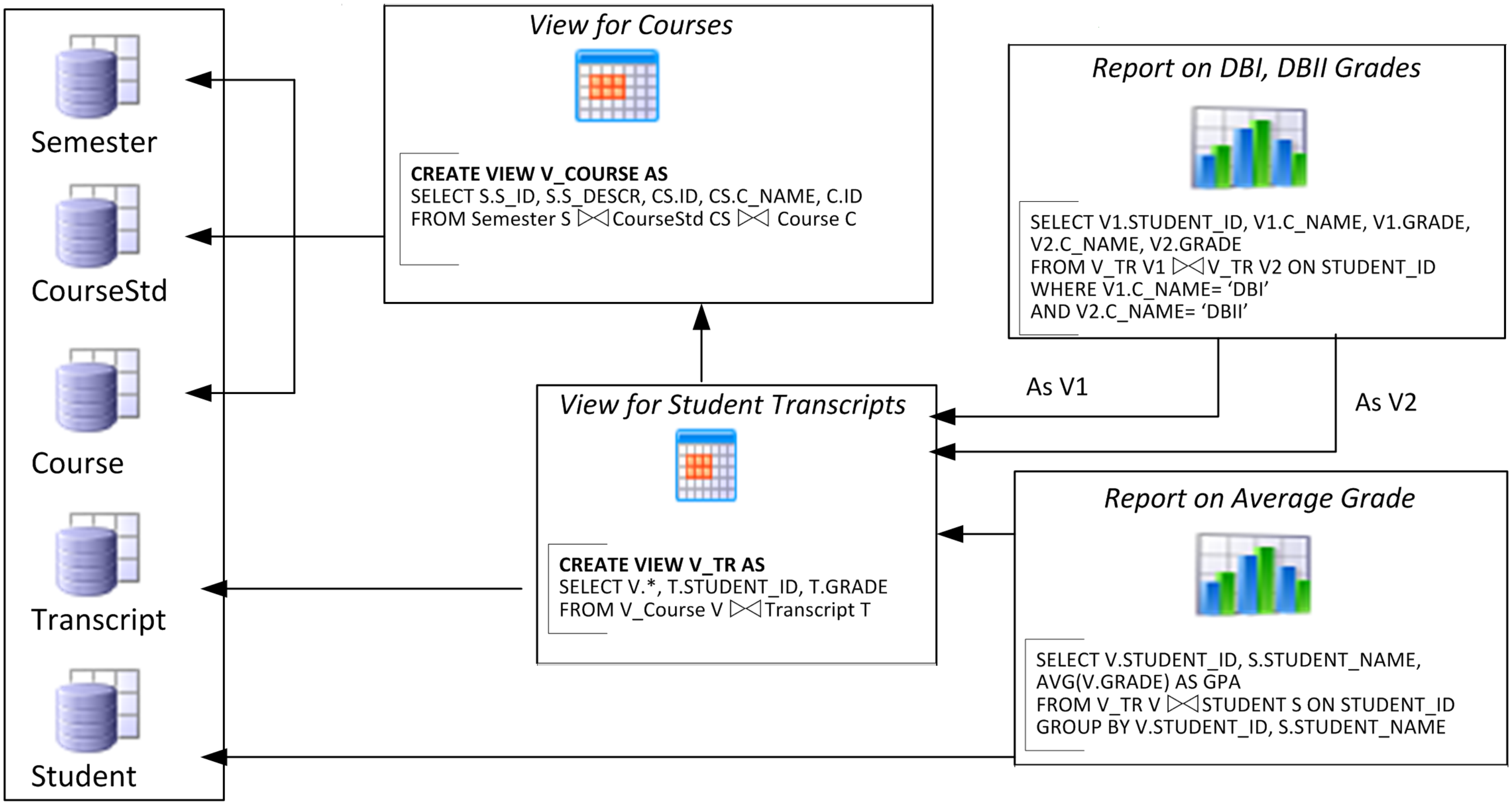
… and data intensive ecosystems…

IMPACT ASSESSMENT

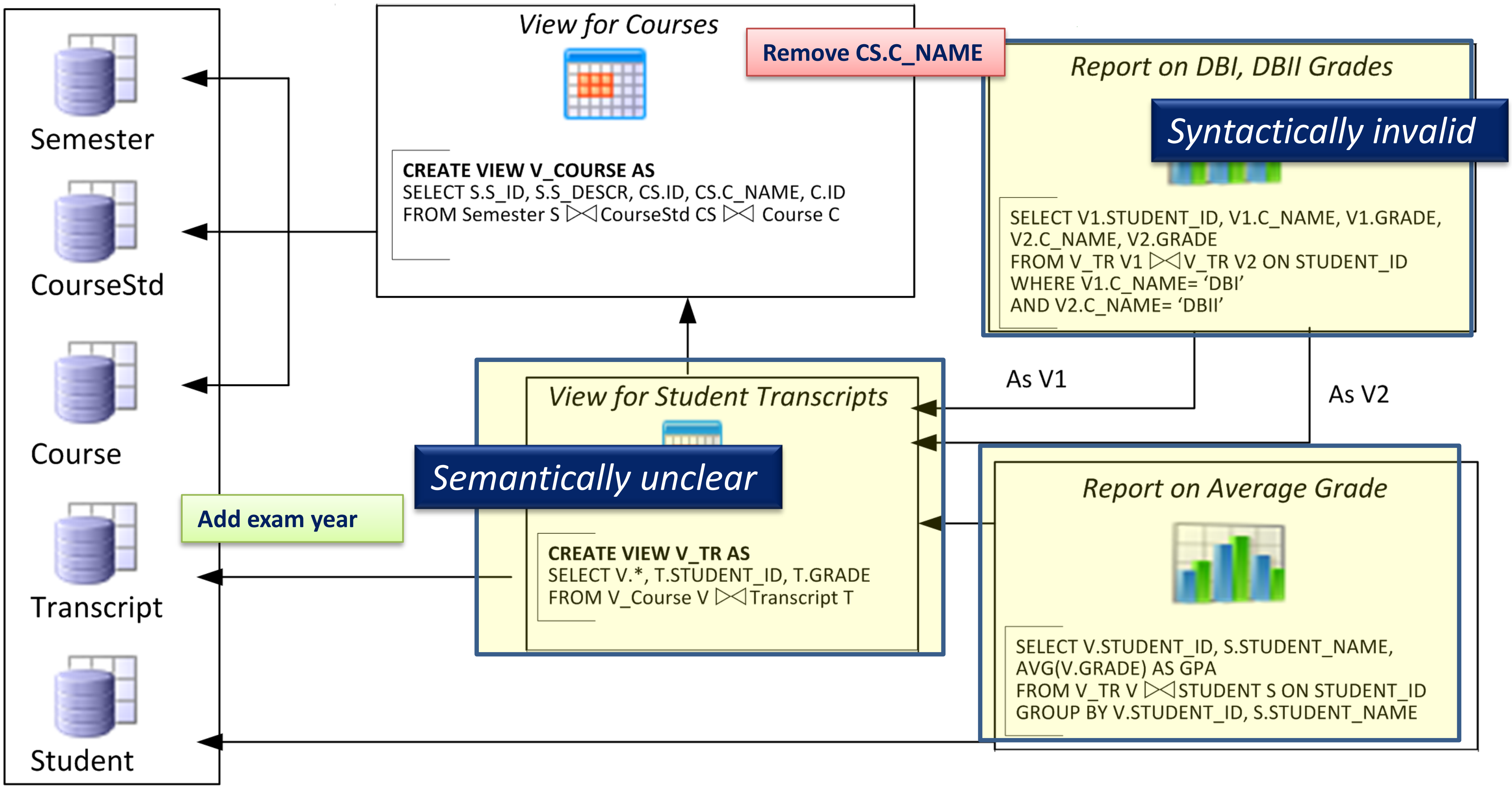
Data intensive ecosystems

- Ecosystems of **applications**, built on top of one or more **databases** and strongly dependent upon them
- Like all software systems, they too change…

Evolving data-intensive ecosystem



Evolving data-intensive ecosystem



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

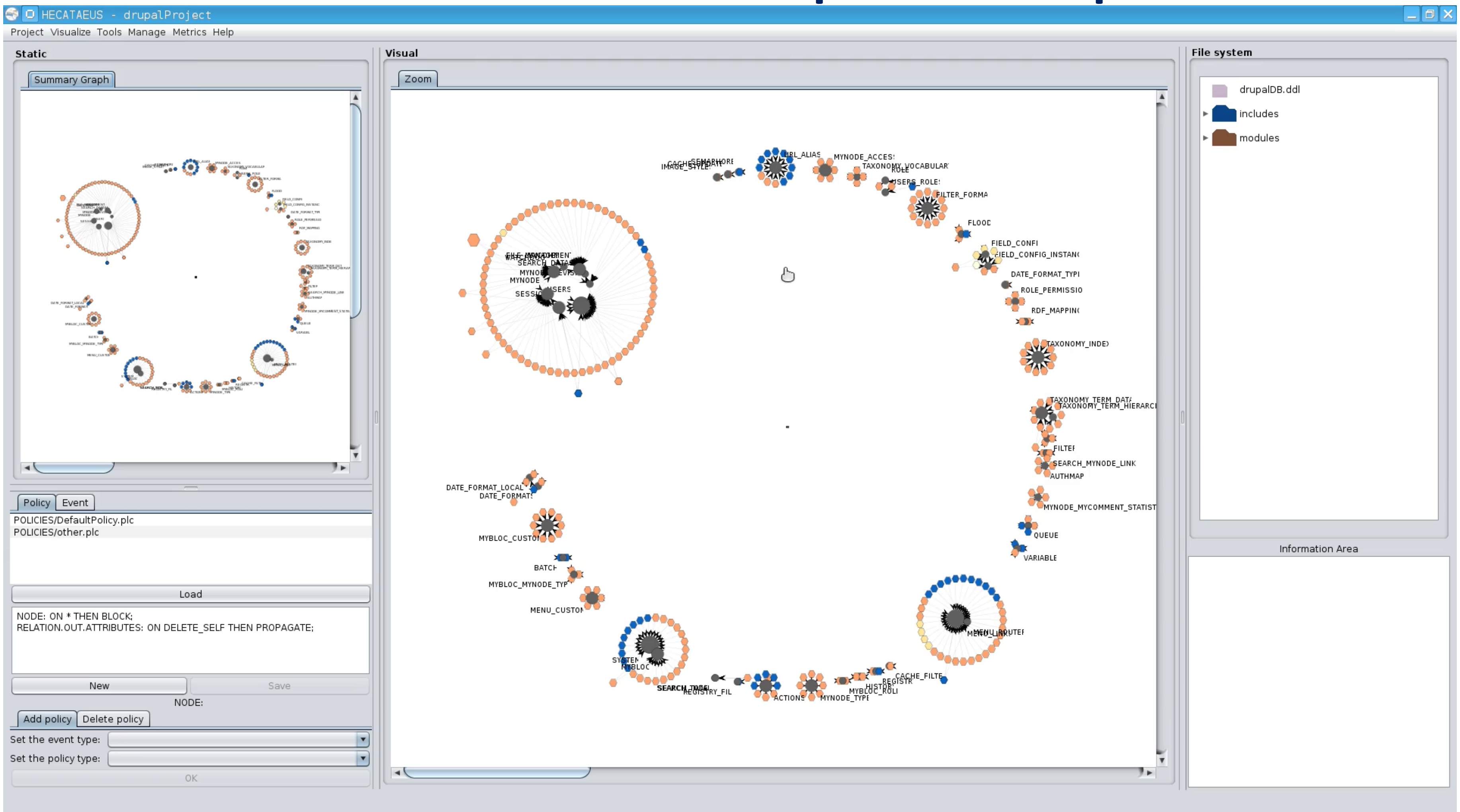
We would like: evolution predictability

i.e., control of what will be affected

before changes happen

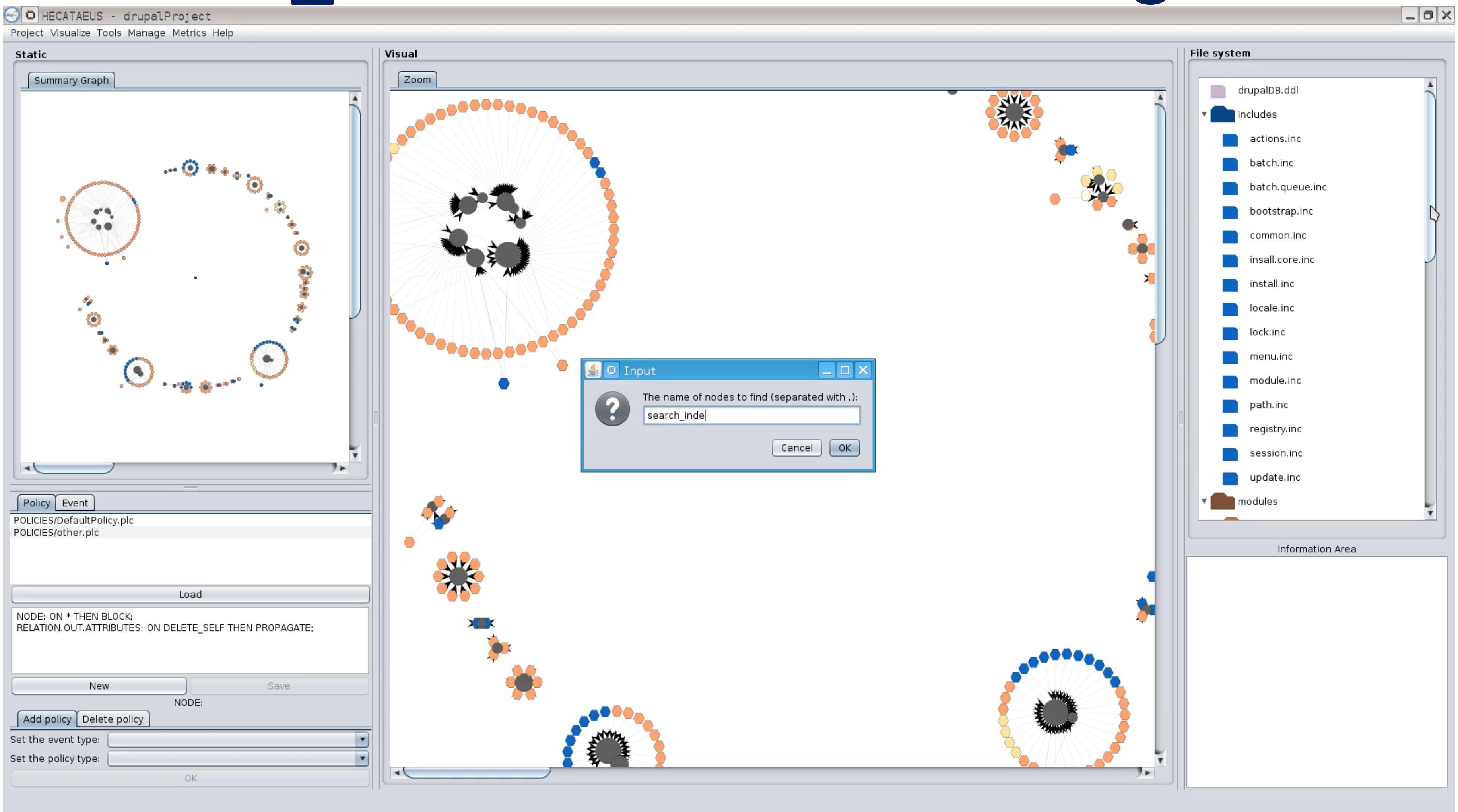
- Learn what changes & how
- Find ways to quarantine effects

The **Hecataeus** tool & method.
Here: a first map of Drupal

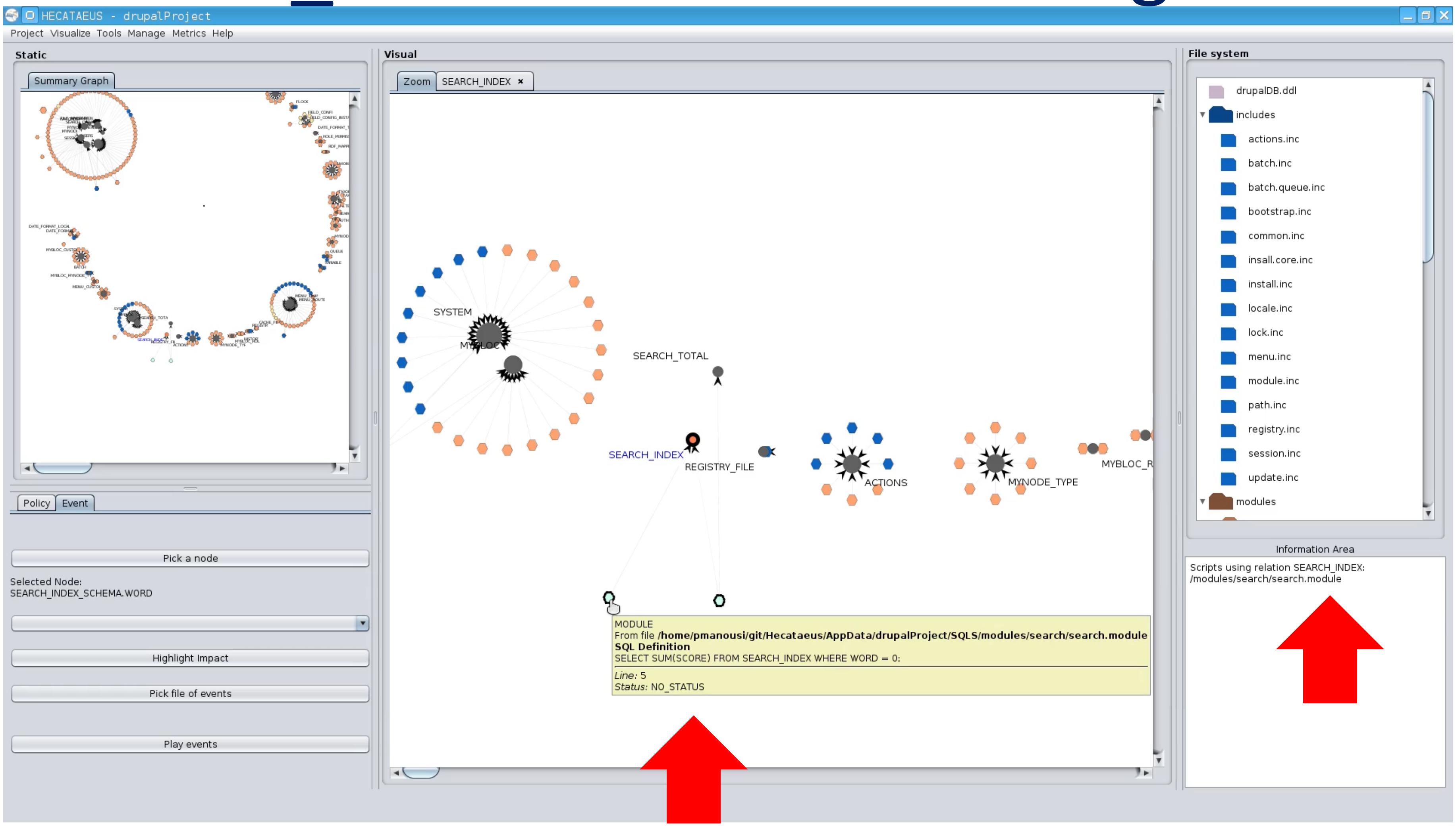


http://www.cs.uoi.gr/~pvassil/projects/hecataeus/

What happens if I modify table search_index? Who are the neighbors?

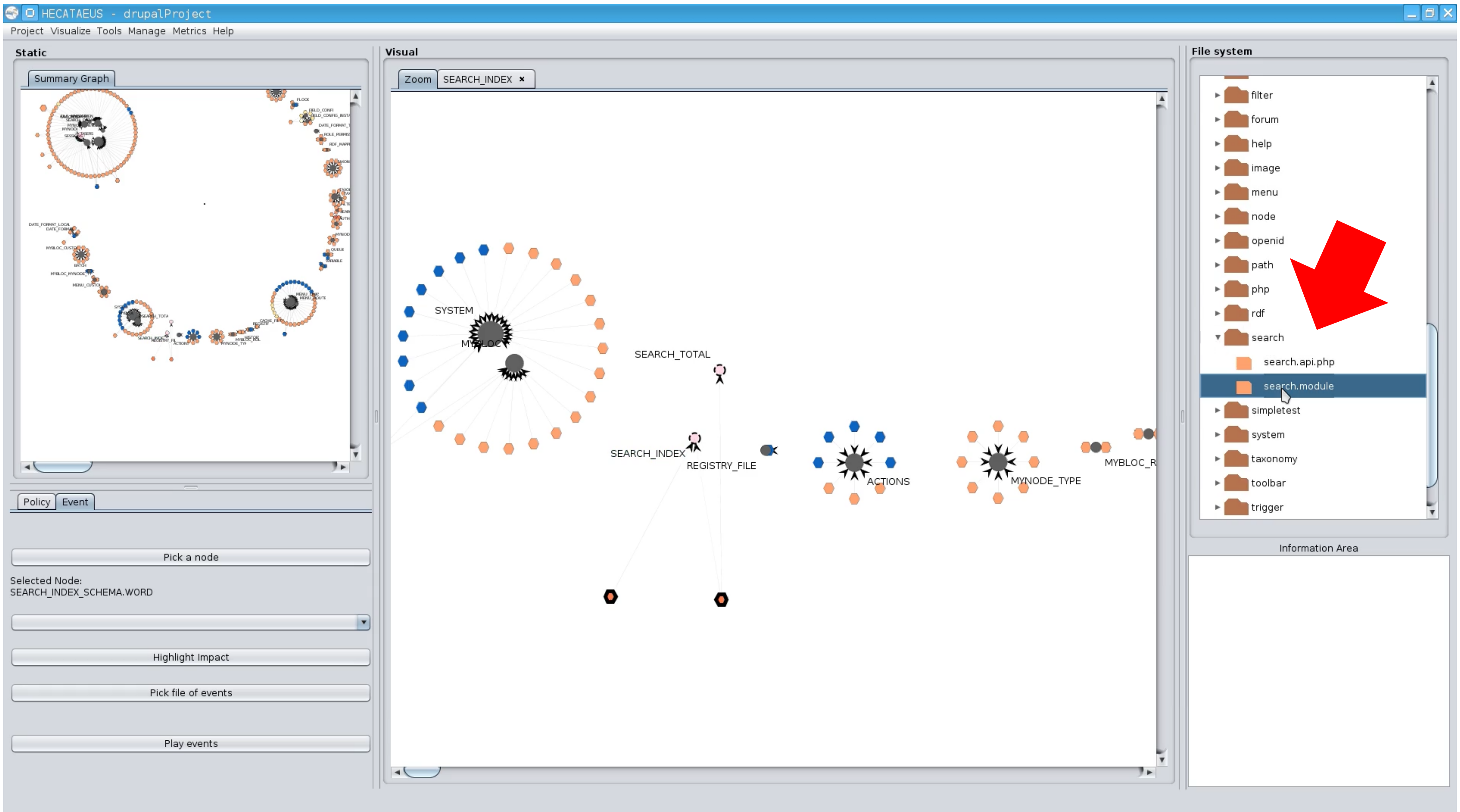


What happens if I modify table search_index? Who are the neighbors?

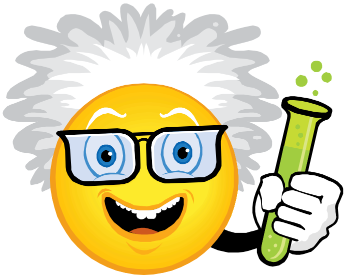


Tooltips with info on the script & query

In the file structure too…



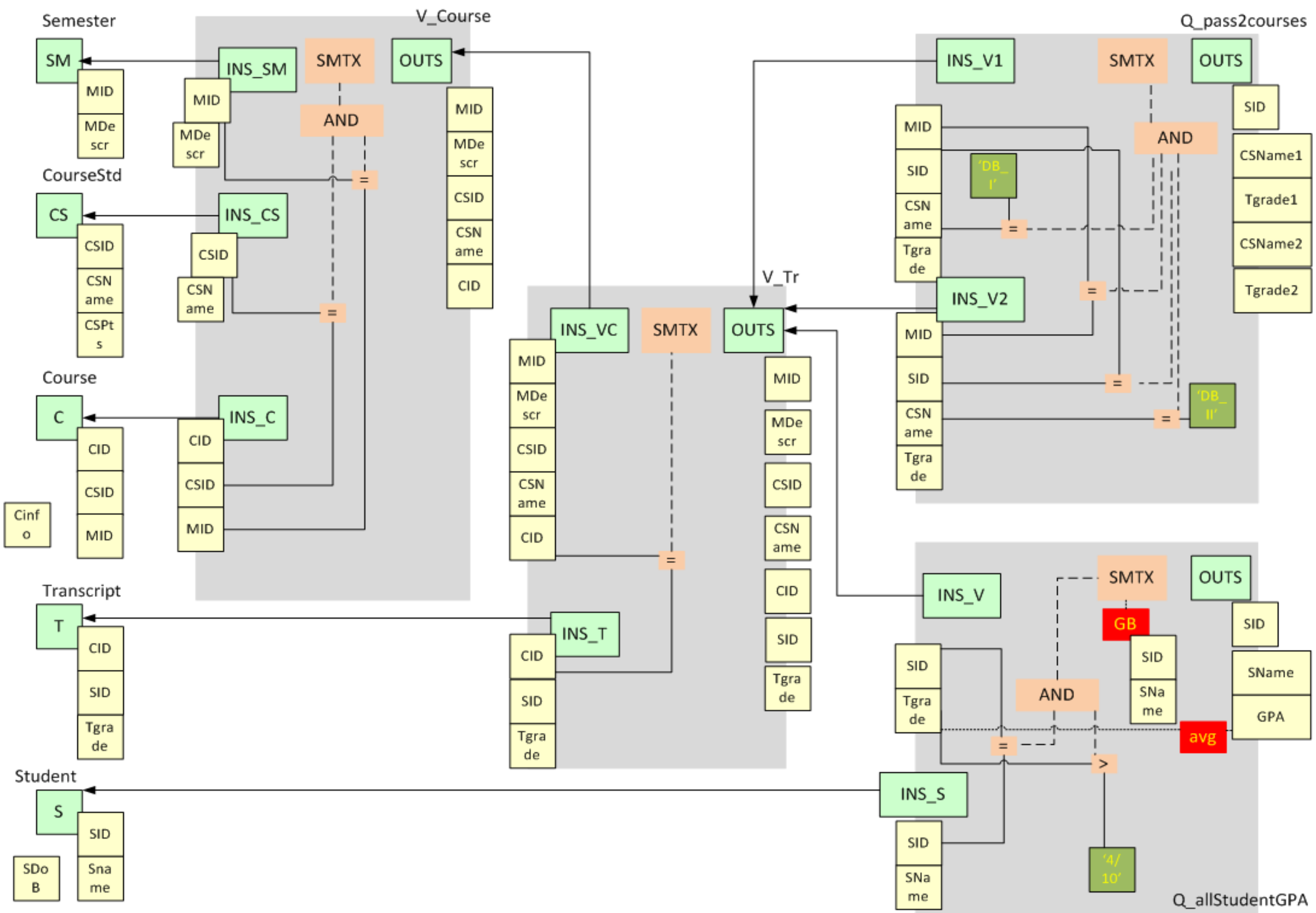
How to handle evolution?



- 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

Manousis+ @ ER 2013 for the details of impact analysis (summary coming)
ER 2014 for the visualization (not here)

University E/S Architecture Graph

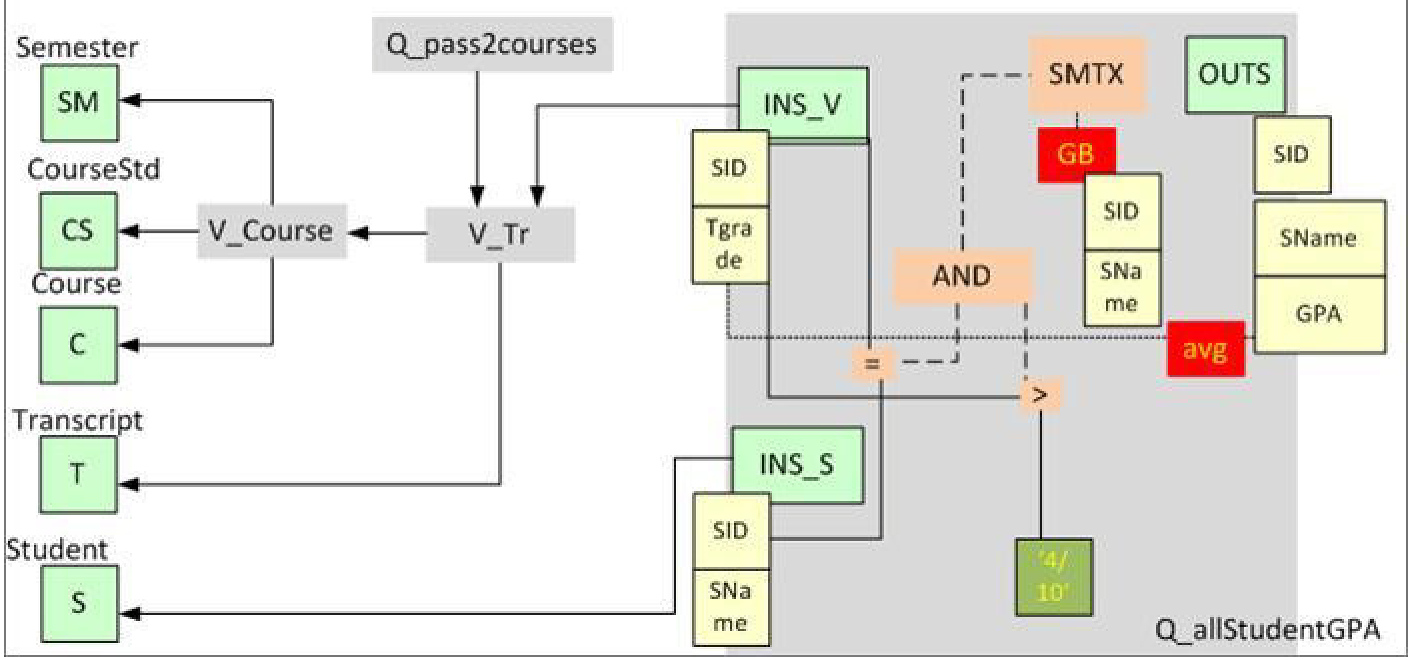


Architecture Graph

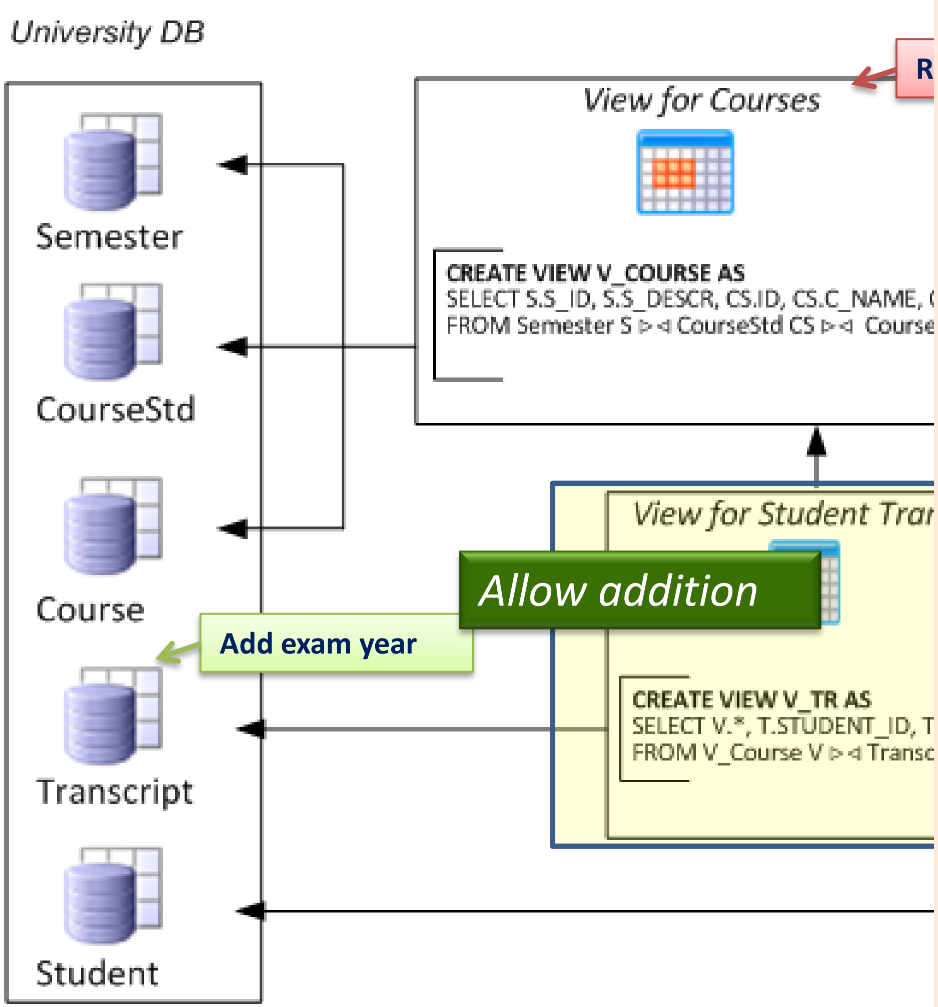
Modules and Module Encapsulation

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



Policies to predetermine reactions



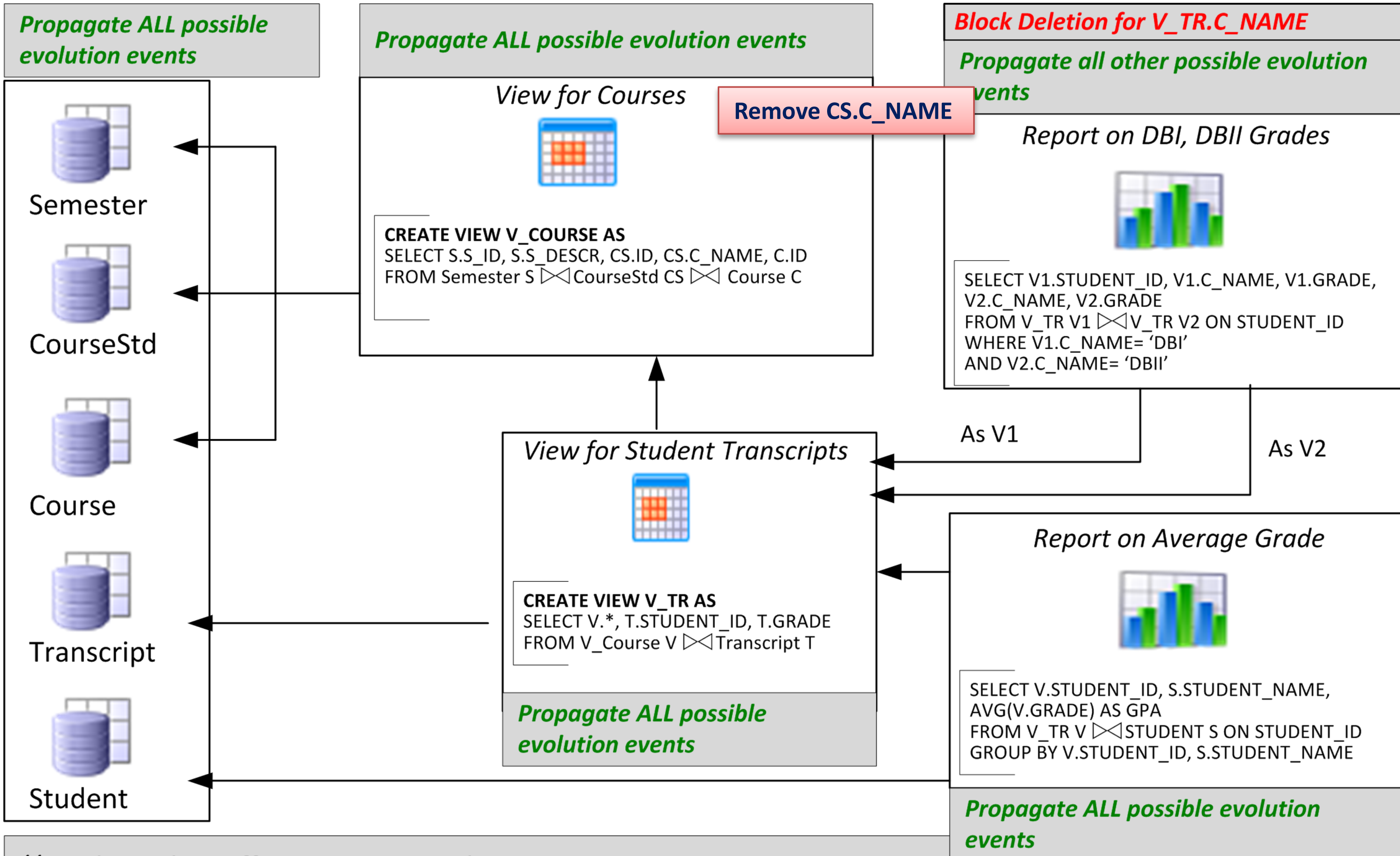

```
RELATION.OUT.SELF: on ADD_ATTRIBUTE then PROPAGATE;
RELATION.OUT.SELF: on DELETE_SELF then PROPAGATE;
RELATION.OUT.SELF: on RENAME_SELF then PROPAGATE;
RELATION.OUT.ATTRIBUTES: on DELETE_SELF then PROPAGATE;
RELATION.OUT.ATTRIBUTES: on RENAME_SELF then PROPAGATE;

VIEW.OUT.SELF: on ADD_ATTRIBUTE then PROPAGATE;
VIEW.OUT.SELF: on ADD_ATTRIBUTE_PROVIDER then PROPAGATE;
VIEW.OUT.SELF: on DELETE_SELF then PROPAGATE;
VIEW.OUT.SELF: on RENAME_SELF then PROPAGATE;
VIEW.OUT.ATTRIBUTES: on DELETE_SELF then PROPAGATE;
VIEW.OUT.ATTRIBUTES: on RENAME_SELF then PROPAGATE;
VIEW.OUT.ATTRIBUTES: on DELETE_PROVIDER then PROPAGATE;
VIEW.OUT.ATTRIBUTES: on RENAME_PROVIDER then PROPAGATE;
VIEW.IN.SELF: on DELETE_PROVIDER then PROPAGATE;
VIEW.IN.SELF: on RENAME_PROVIDER then PROPAGATE;
VIEW.IN.SELF: on ADD_ATTRIBUTE_PROVIDER then PROPAGATE;
VIEW.IN.ATTRIBUTES: on DELETE_PROVIDER then PROPAGATE;
VIEW.IN.ATTRIBUTES: on RENAME_PROVIDER then PROPAGATE;
VIEW.SMTX.SELF: on ALTER_SEMANTICS then PROPAGATE;

QUERY.OUT.SELF: on ADD_ATTRIBUTE then PROPAGATE;
QUERY.OUT.SELF: on ADD_ATTRIBUTE_PROVIDER then PROPAGATE;
QUERY.OUT.SELF: on DELETE_SELF then PROPAGATE;
QUERY.OUT.SELF: on RENAME_SELF then PROPAGATE;
QUERY.OUT.ATTRIBUTES: on DELETE_SELF then PROPAGATE;
QUERY.OUT.ATTRIBUTES: on RENAME_SELF then PROPAGATE;
QUERY.OUT.ATTRIBUTES: on DELETE_PROVIDER then PROPAGATE;
QUERY.OUT.ATTRIBUTES: on RENAME_PROVIDER then PROPAGATE;
QUERY.IN.SELF: on DELETE_PROVIDER then PROPAGATE;
QUERY.IN.SELF: on RENAME_PROVIDER then PROPAGATE;
QUERY.IN.SELF: on ADD_ATTRIBUTE_PROVIDER then PROPAGATE;
QUERY.IN.ATTRIBUTES: on DELETE_PROVIDER then PROPAGATE;
QUERY.IN.ATTRIBUTES: on RENAME_PROVIDER then PROPAGATE;
QUERY.SMTX.SELF: on ALTER_SEMANTICS then PROPAGATE;
```

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

How to handle evolution?



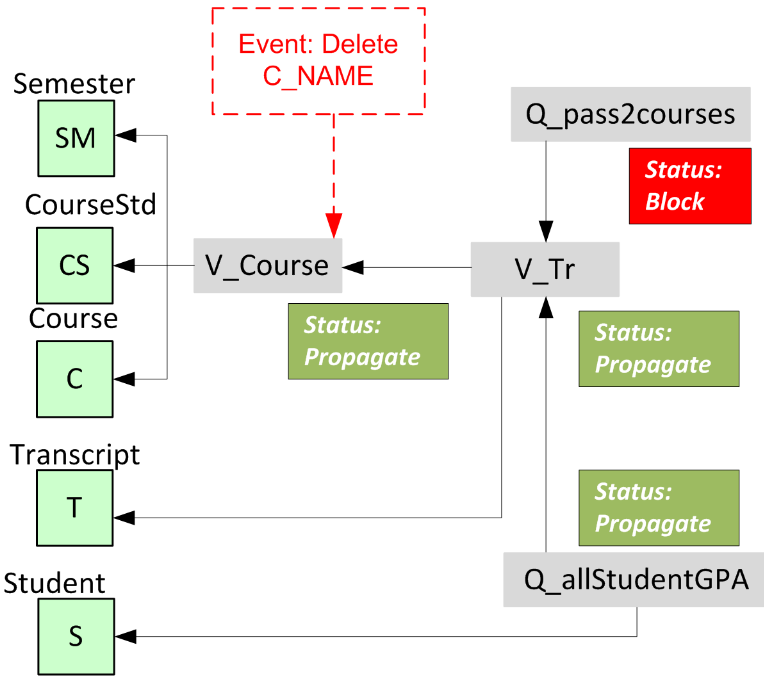
//2 policy rules suffice to annotate the entire ecosystem:

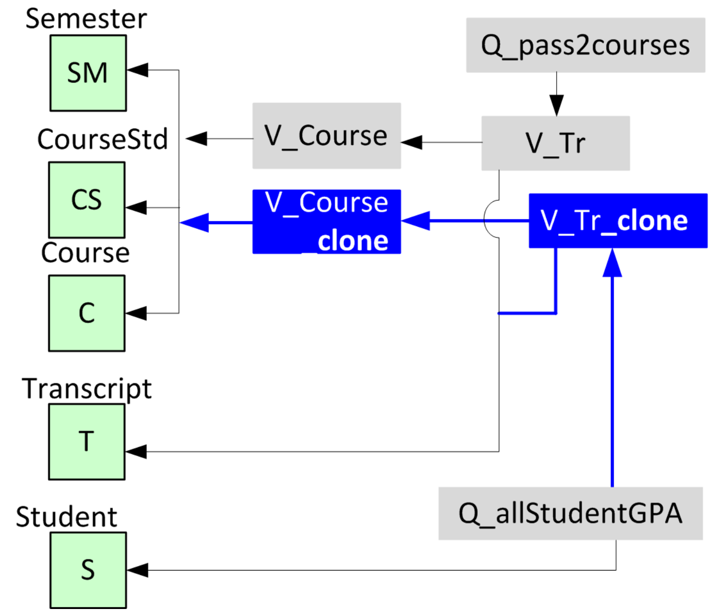
```
NODE: ON * THEN PROPAGATE;
Q_pass2courses_IN_V1.C_SNAME ON DELETE_SELF THEN BLOCK;
```

Internals of impact assess. & rewriting

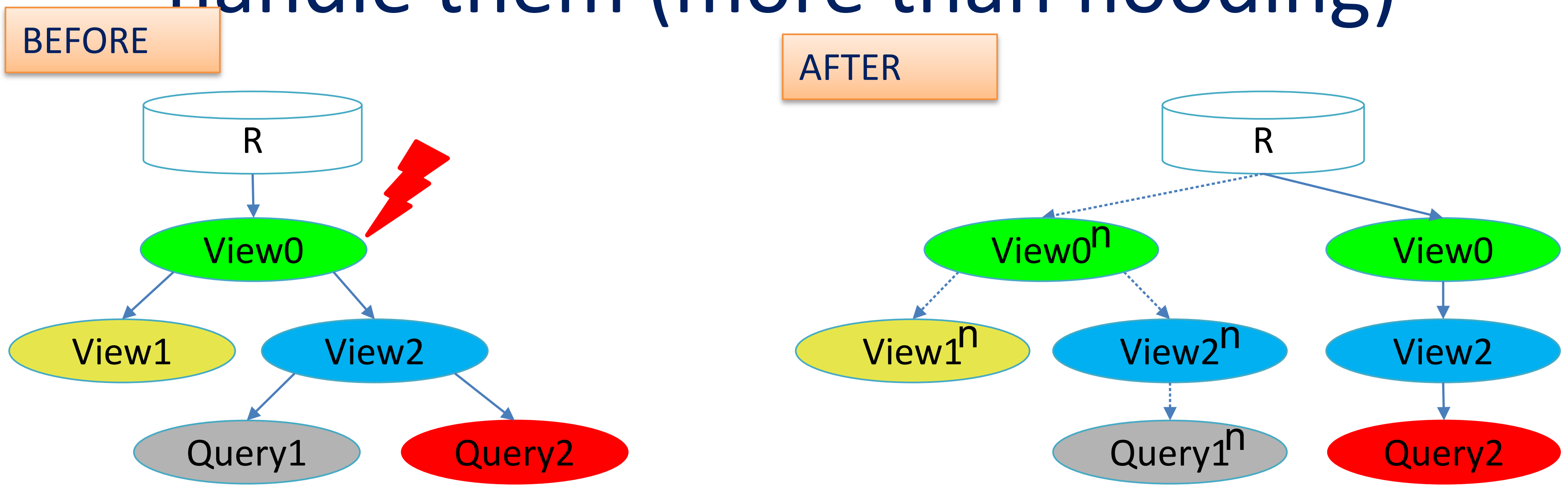
1. **Impact assessment**. Given a potential event, a status determination algorithm makes sure that the nodes of the ecosystem are assigned a status concerning (a) whether they are affected by the event or not and (b) what their reaction to the event is (block or propagate).
2. **Conflict resolution and calculation of variants**. Algorithm that checks the affected parts of the graph in order to highlight affected nodes with whether they will adapt to a new version or retain both their old and new variants.
3. **Module Rewriting**. Our algorithm visits affected modules sequentially and performs the appropriate restructuring of nodes and edges.

Impact assessment & rewriting





Conflicts: what they are and how to handle them (more than flooding)



- View0 initiates a change
- View1 and View 2 accept the change

- Query2 rejects the change
- Query1 accepts the change

- 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

Played an impact analysis scenario: delete attr. ‘word’ from search_index