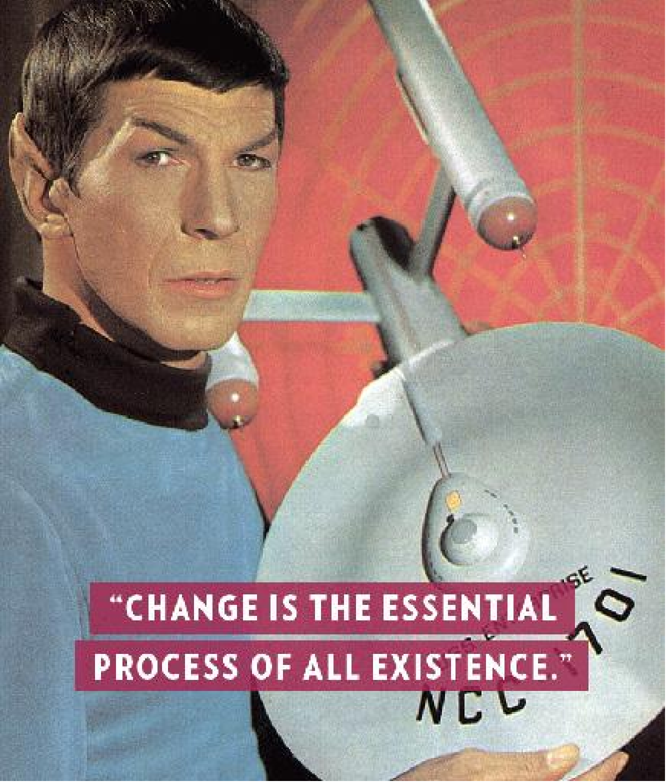
# Schema Evolution and Gravitation to Rigidity: a tale of calmness in the lives of structured data

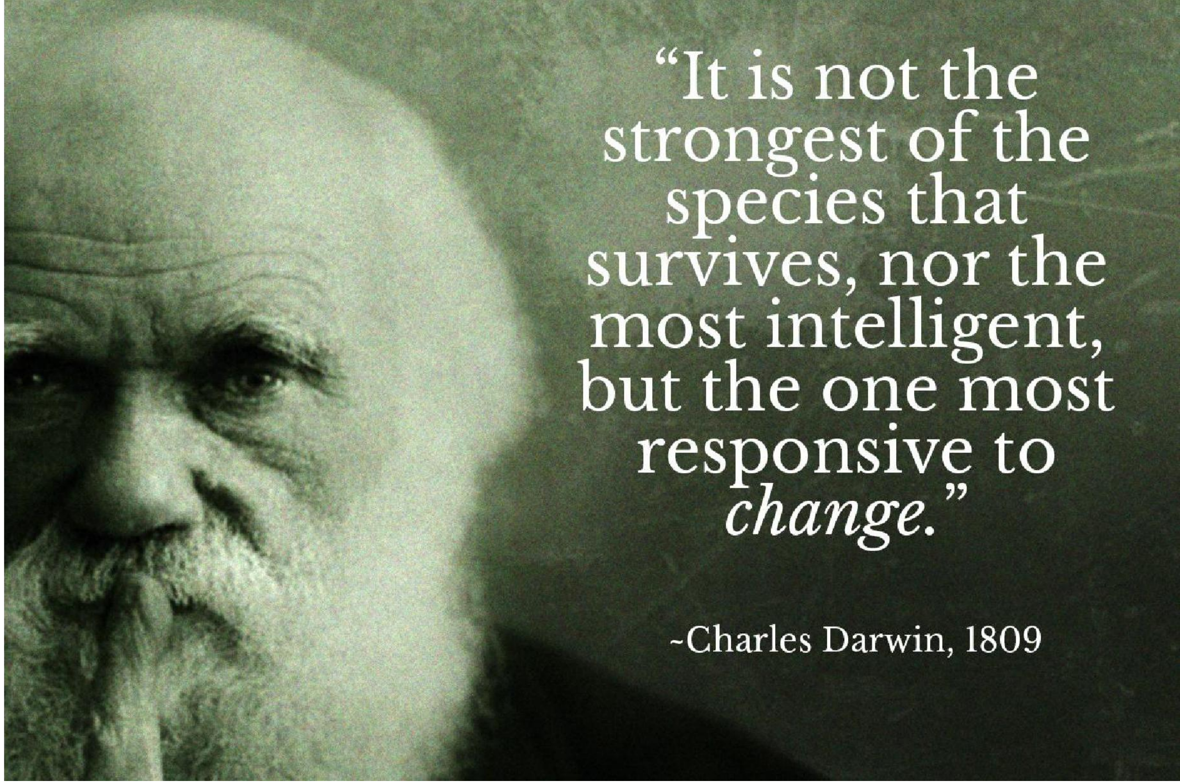
Panos Vassiliadis

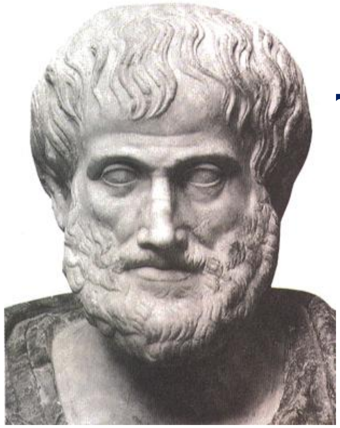
joint work with: Apostolos Zarras, Ioannis Skoulis, Petros Manousis, Fanis Giahos, Michael Kolozoff, Athanasios Pappas, Maria Zerva

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 |

- Preventive maintenance: modification of a software product after delivery to detect and correct latent faults in the software product before they become operational faults.
- **Corrective** maintenance: reactive modification (or repairs) of a software product performed after delivery to **correct discovered problems**.
- **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**.
- **Adaptive** maintenance: modification of a software product performed after delivery **to keep a software product usable in a changed or changing environment**.

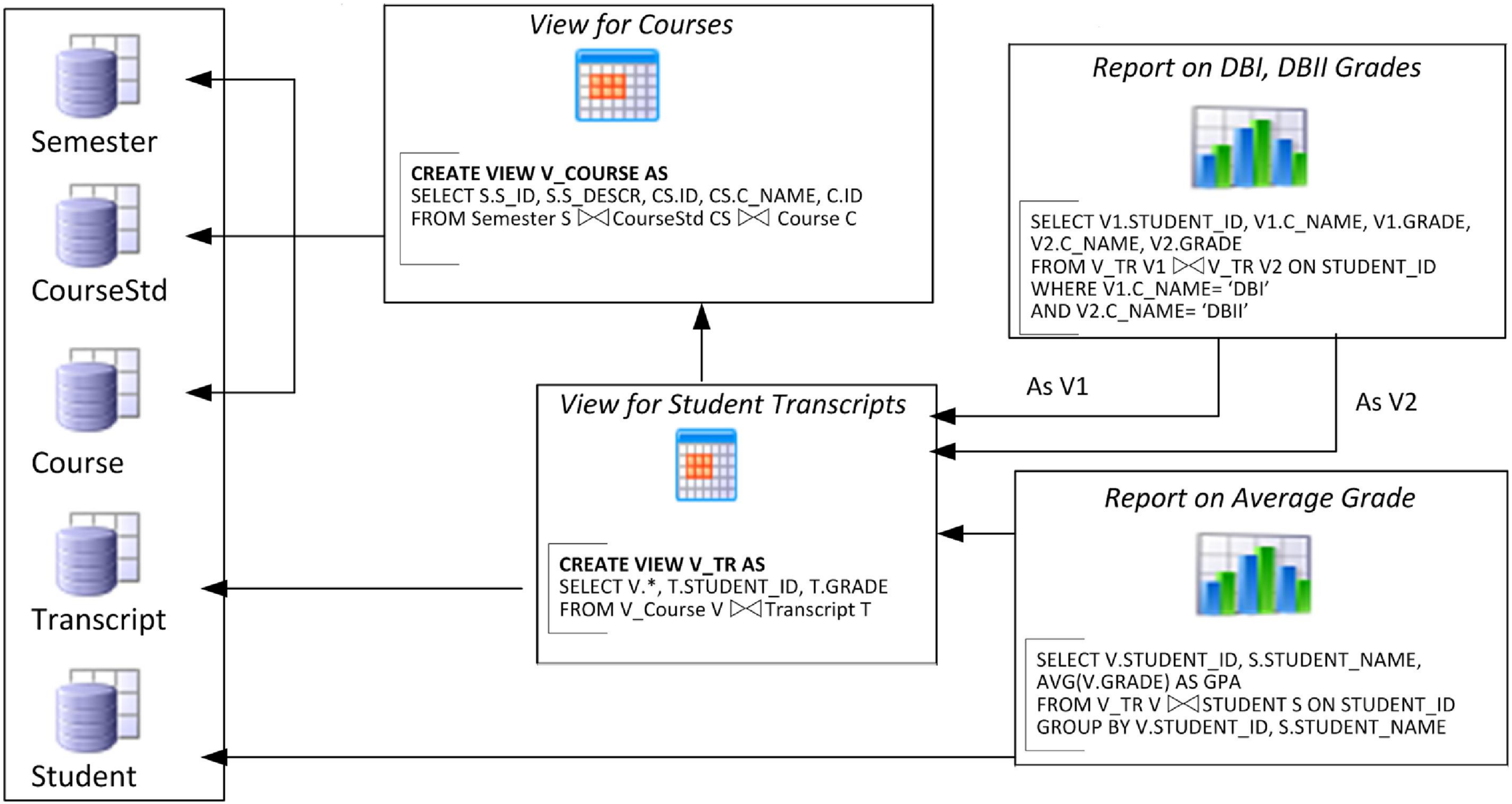
# Database Evolution: why and what

- 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

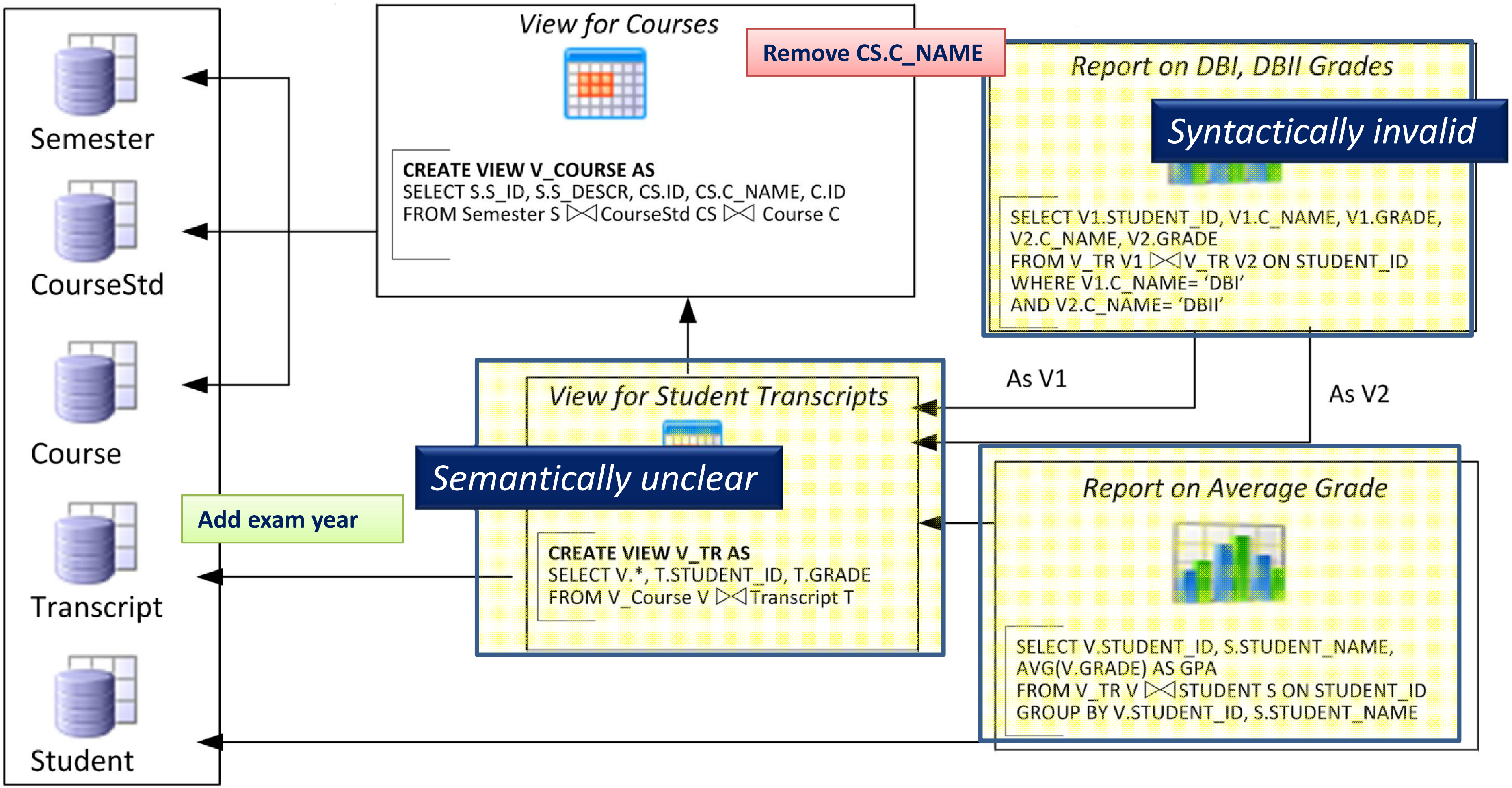
# Why is (schema) evolution so important?

- Software and DB **maintenance** makes up for **at least 50% of all resources spent in a project**.
- **Dependency magnets**
  - **Databases are rarely stand-alone**: typically, an entire ecosystem of applications is structured around them **=>**
  - Typically, **development waits till the “db backbone” is stable** and applications are “safely” build on top of it, as…
  - **… changes in the schema can impact a large (typically, not traced) number of surrounding applications, without explicit identification of the impact & can cause several (parts of) different applications to crash, slow down, or miss data, causing the need for emergency repairing**

# 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 evolution

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

# We would really love to…

**Engineering goals**

- **… "design for evolution"** and **minimize the impact of evolution** to the surrounding applications by introducing appropriate mechanisms in our DBMS's, applying design patterns & avoiding anti-patterns **in both the db and the code** in a way that insulates applications from unwanted schema change impacts
- **… plan in advance** administration and perfective maintenance tasks and resources, instead of responding to emergencies

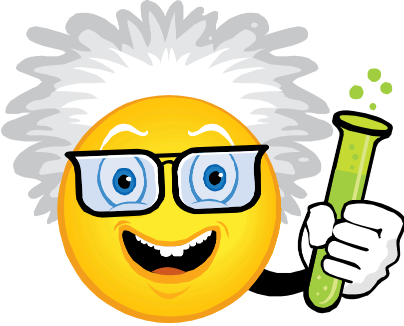
**Scientific goals**

- … (btw) detect & assess if there exist **fundamental flaws in our Paradigms** (like the relational model or the development of data-intensive applications)
- … (with your permission) satisfy the **scientific curiosity** on gaining more knowledge on how things work
- … but first, …

… but, first, we must answer this:

# WHAT ARE THE "LAWS" OF DATABASE SCHEMA EVOLUTION?

# 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?

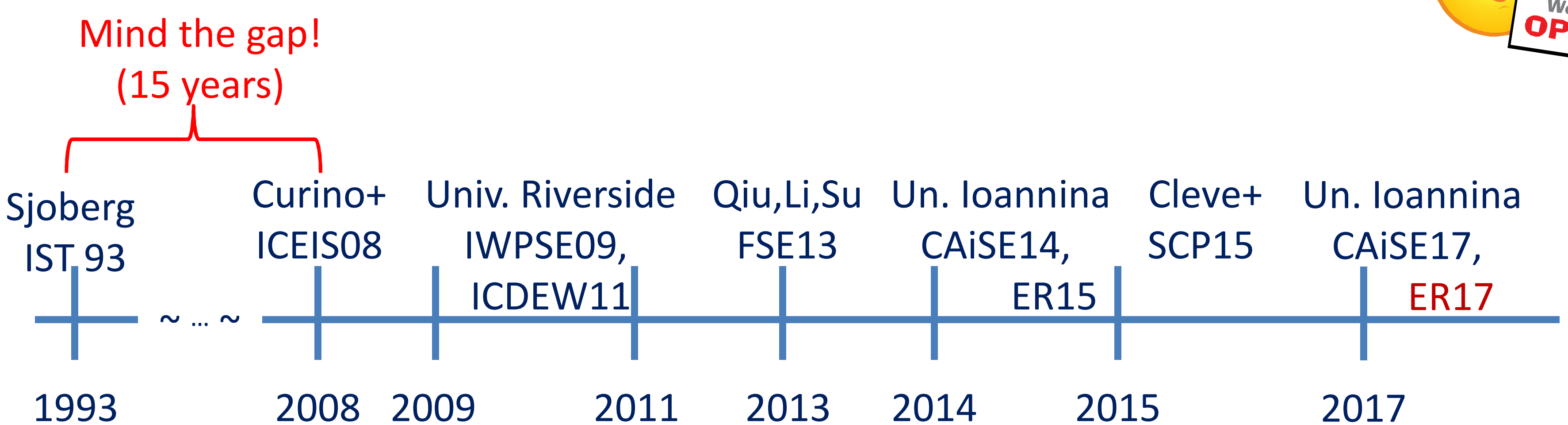
- Can we exploit findings to engineer data-intensive ecosystems that withstand change gracefully?

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

# Do we know the mechanics of schema evolution?

- 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, so…
- … 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 and code available at:** https://github.com/DAINTINESS-Group

# Scope of our studies

- **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 |

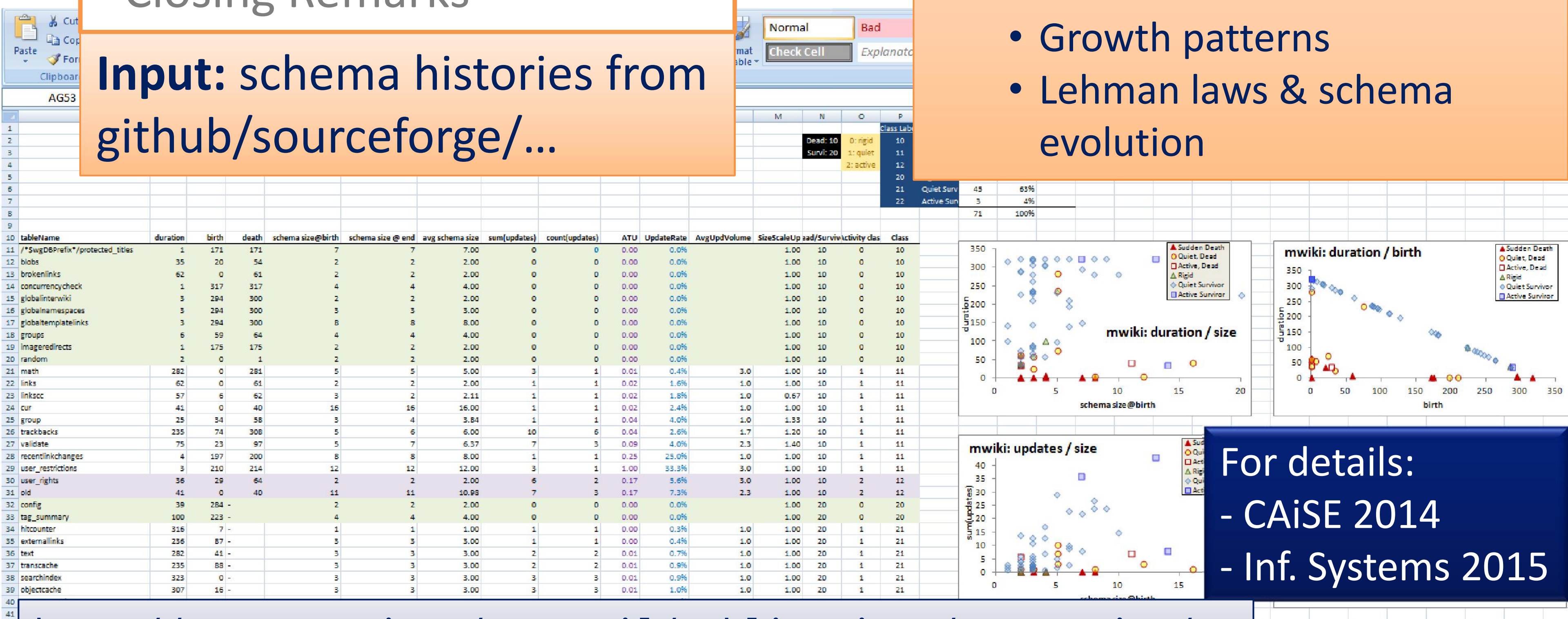
# How does the schema size evolve?

<u>Outline</u>
- **Schema size evolution**
- Foreign Key Evolution
- Table Evolution
- Closing Remarks

**Input:** schema histories from github/sourceforge/…

**Output**: <u>properties & patterns on the evolution of schema size (no. tables)</u>

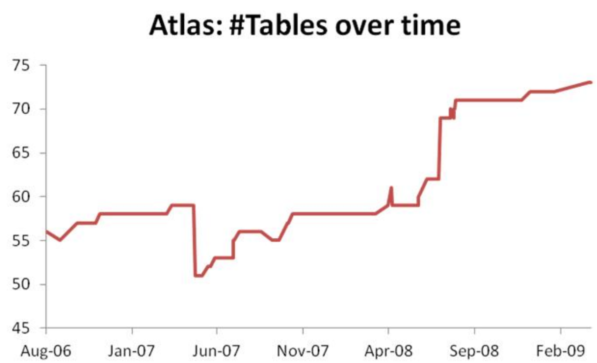
- Not covered here:
  - Growth patterns
  - Lehman laws & schema evolution

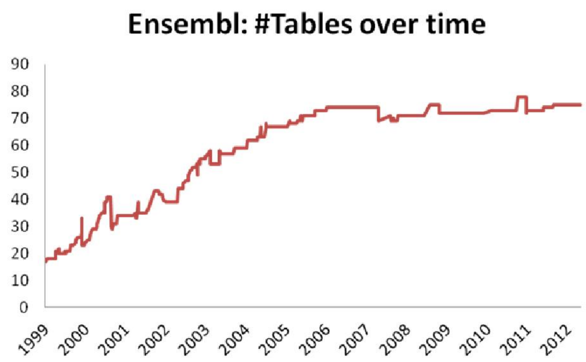


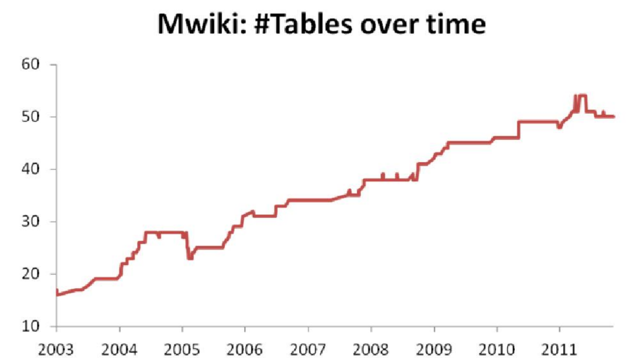

For details:
- CAiSE 2014
- Inf. Systems 2015

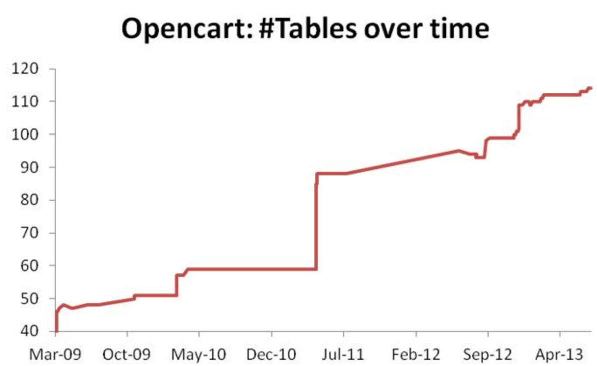
http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

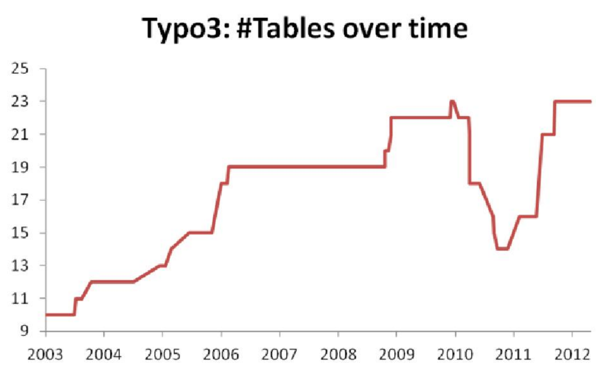
# Schema Size (relations)

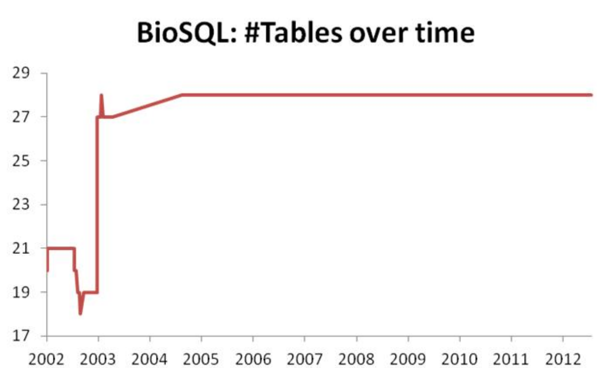


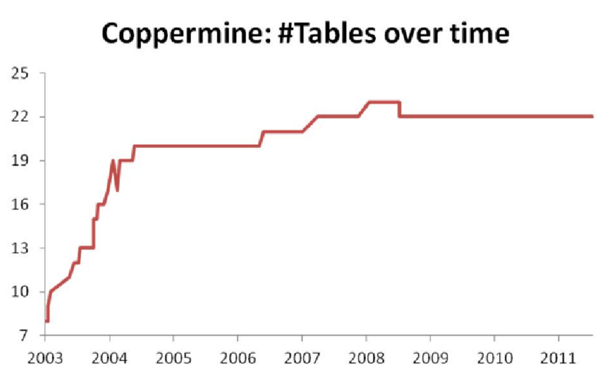



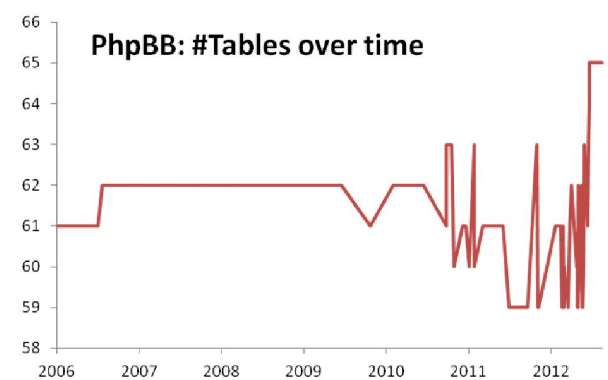



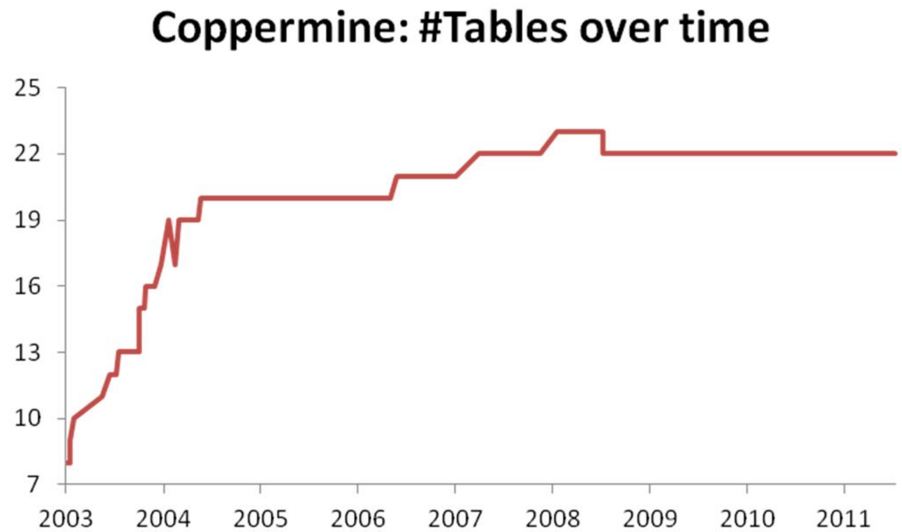



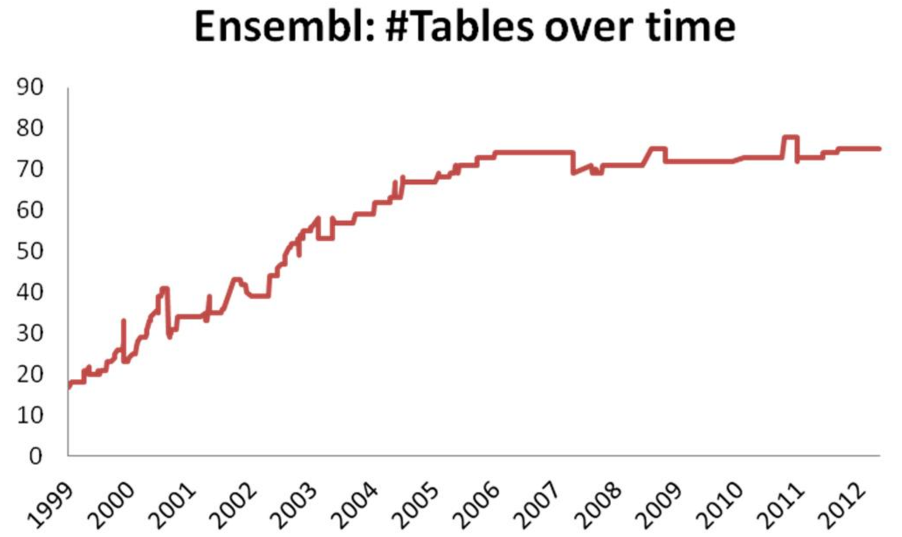



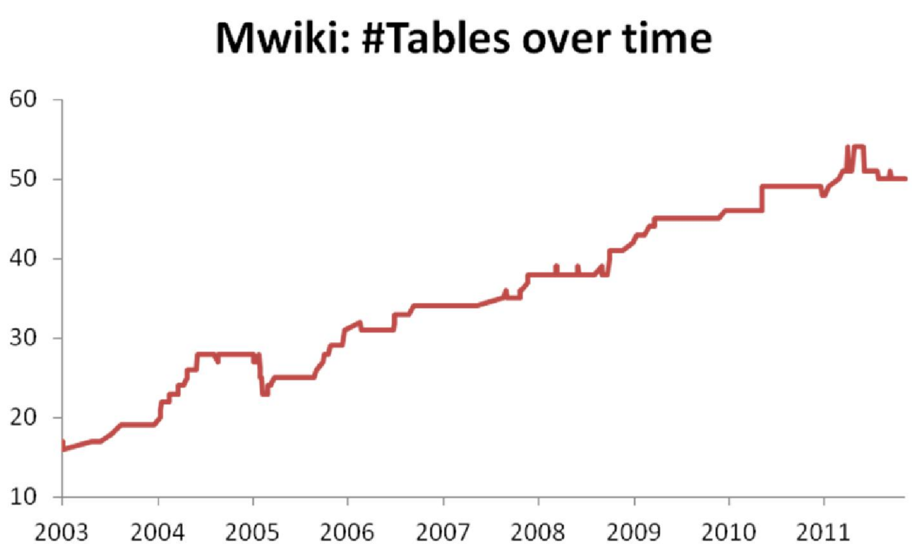



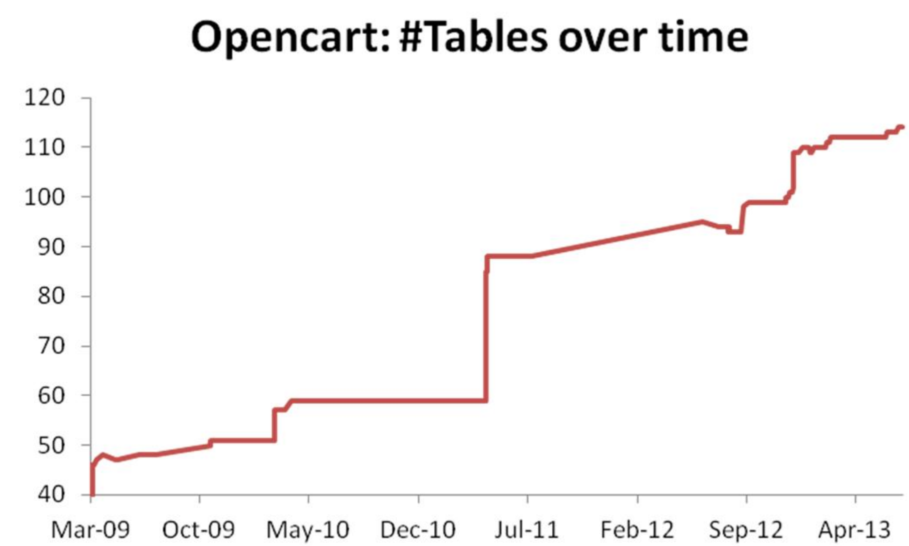
# Highlights of Schema Size Evolution

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

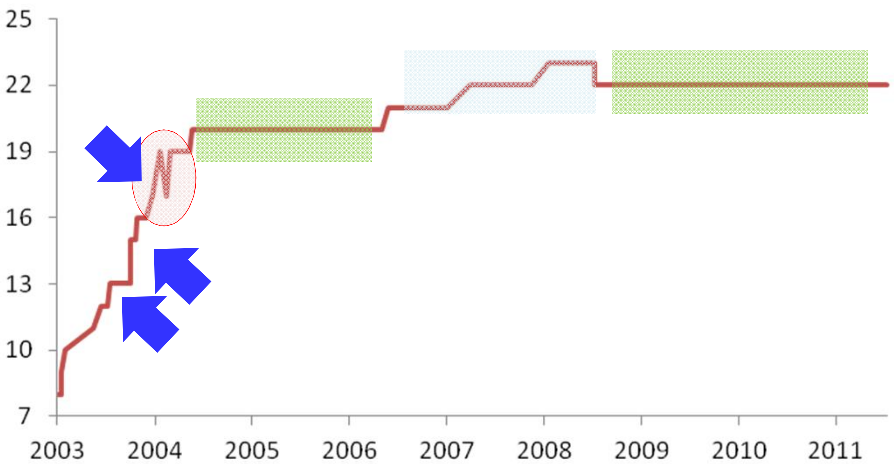

Coppermine: #Tables over time
25
22
19
16
13
10
7
2003
2004
2005
2006
2007
2008
2009
2010
2011


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

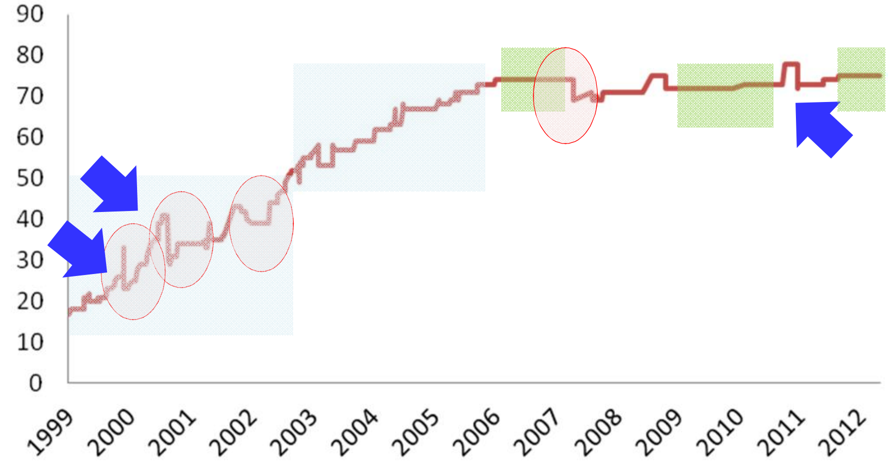

Mwiki: #Tables over time
60
50
40
30
20
10
2003
2004
2005
2006
2007
2008
2009
2010
2011


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



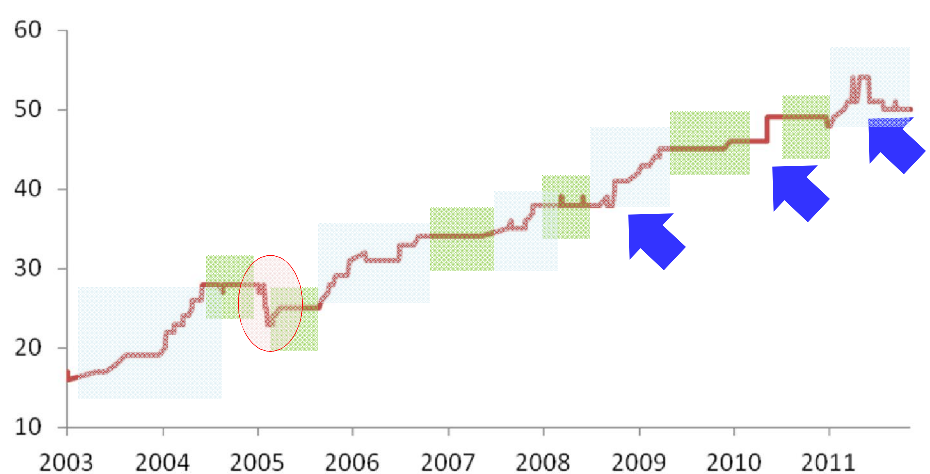

## Ensembl: #Tables over time



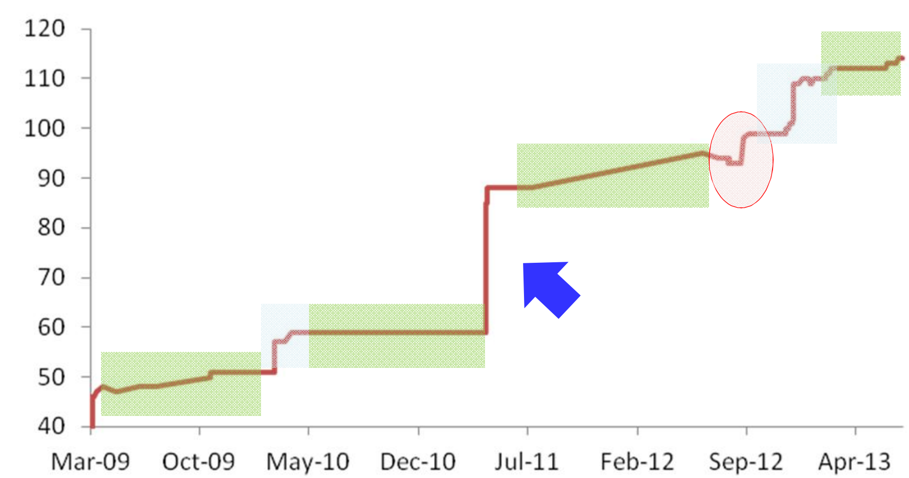

Growth over time
Calmness periods

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

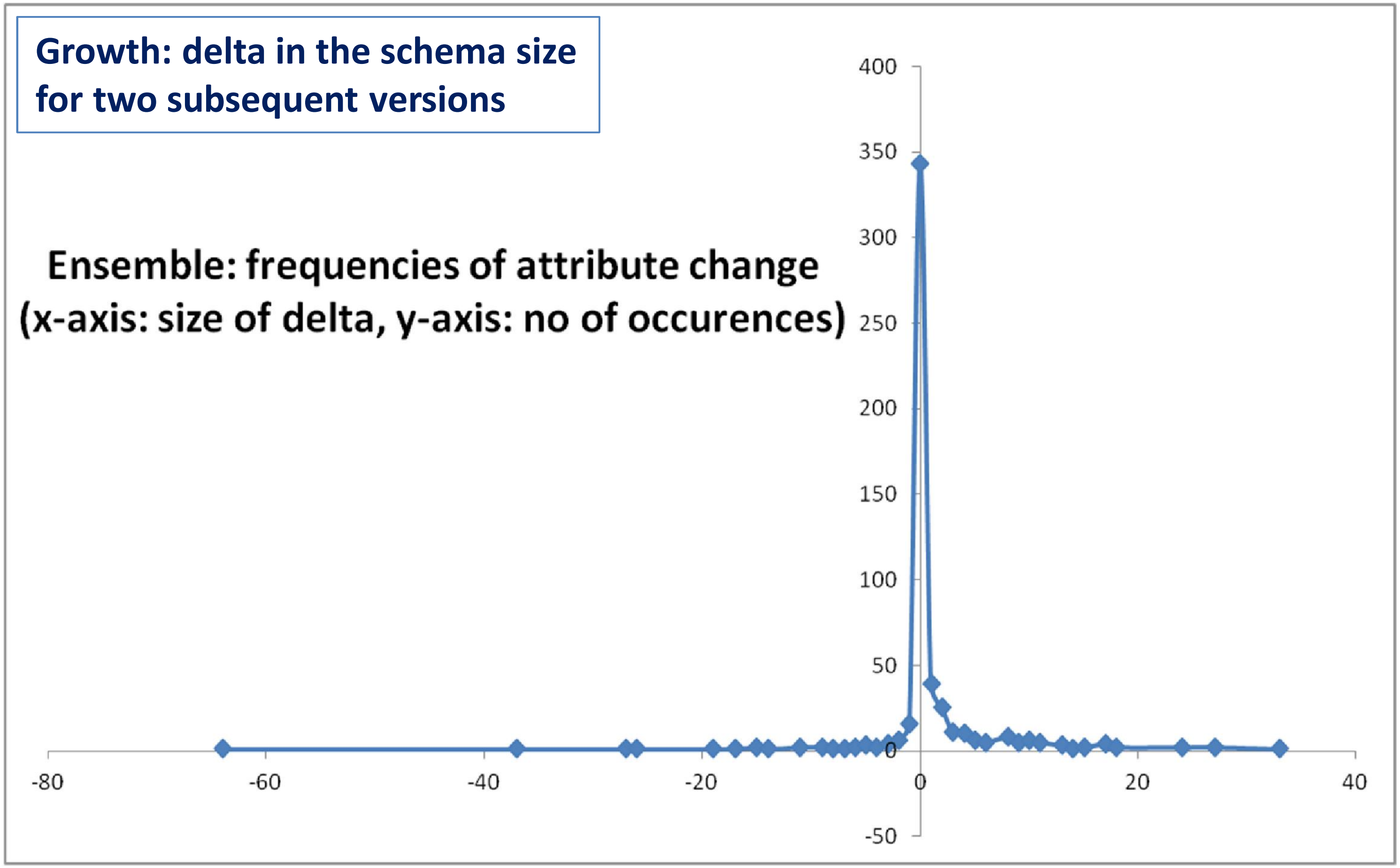
## Mwiki: #Tables over time

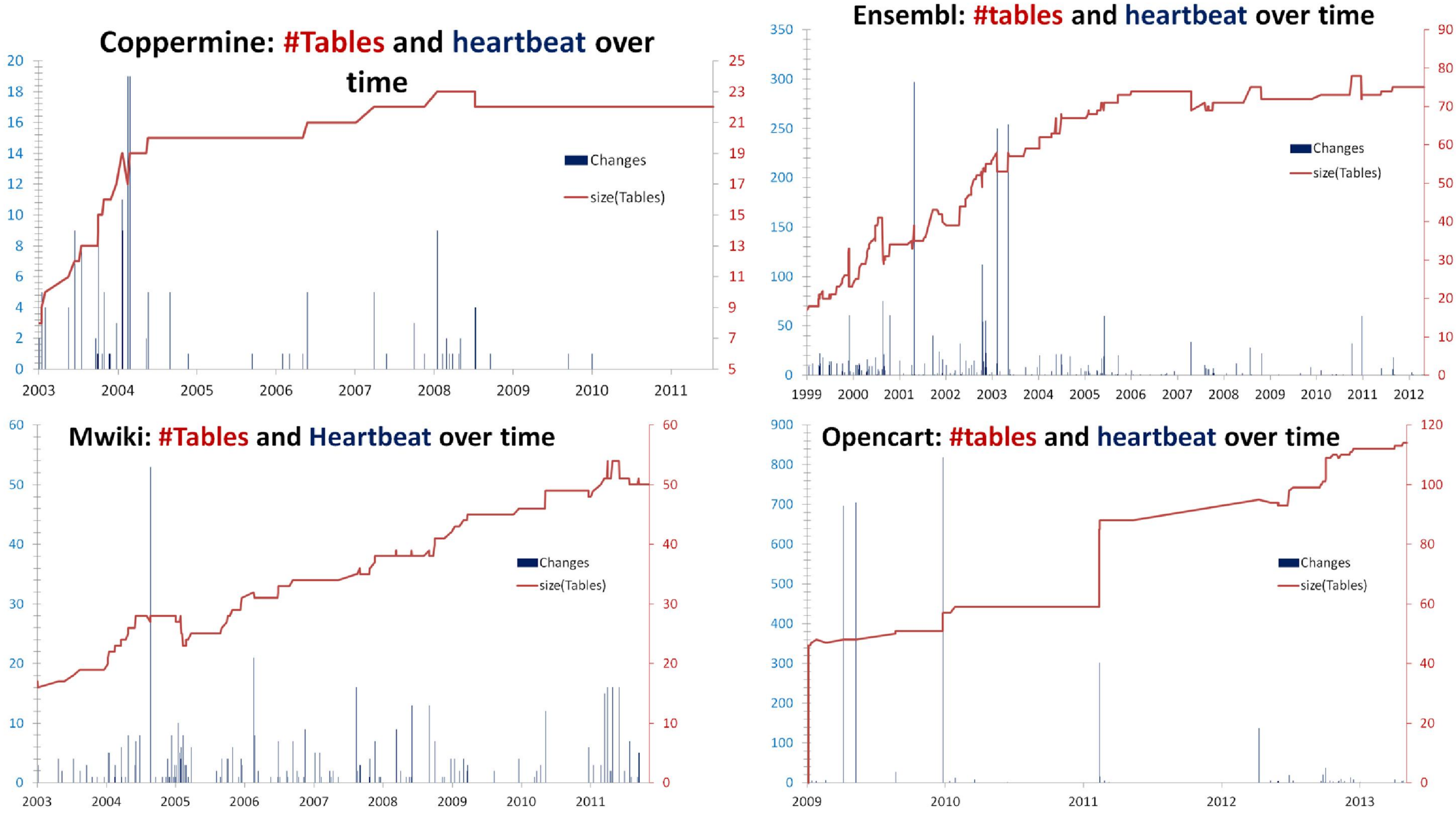



## Opencart: #Tables over time



# Zipfian model in the distribution of growth frequencies






[With exceptions]

Density: **focused maintenance** effort

**Progressive cooling** : early –maintenance density >> later stages

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

# How do foreign keys evolve?

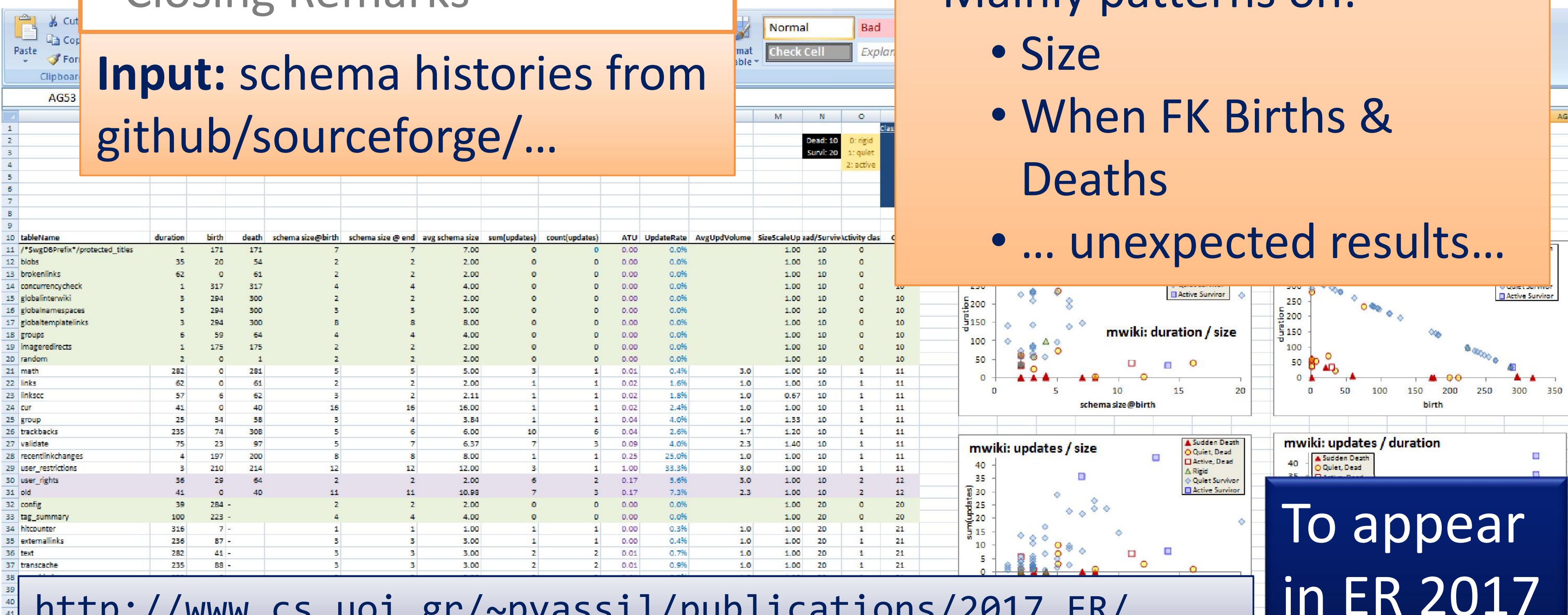
Outline

- Schema size evolution
- **Foreign Key Evolution**
- Table Evolution
- Closing Remarks

**Input:** schema histories from github/sourceforge/…

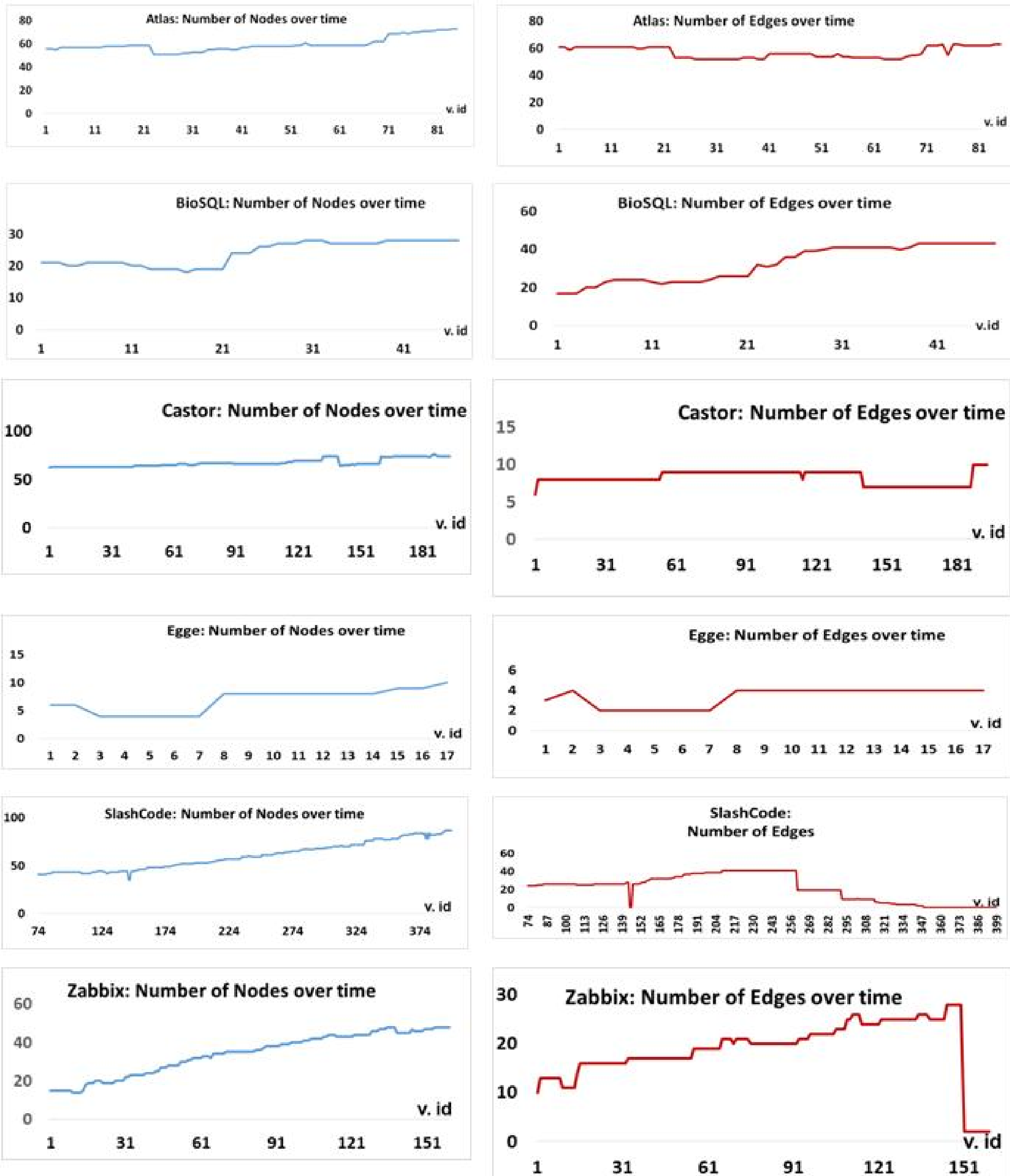
**Output**: properties & **patterns** on the evolution of **foreign keys**

- Mainly patterns on:
  - Size
  - When FK Births & Deaths
  - … unexpected results…




http://www.cs.uoi.gr/~pvassil/publications/2017_ER/

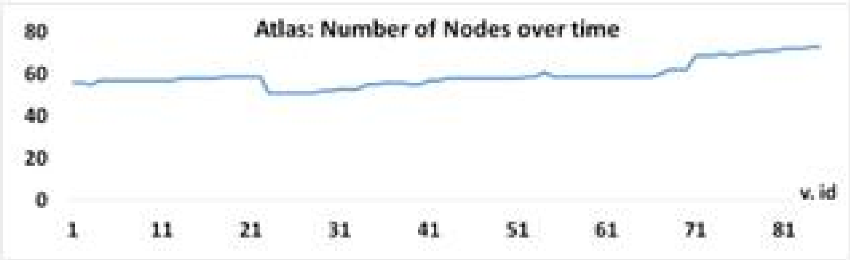
To appear in ER 2017

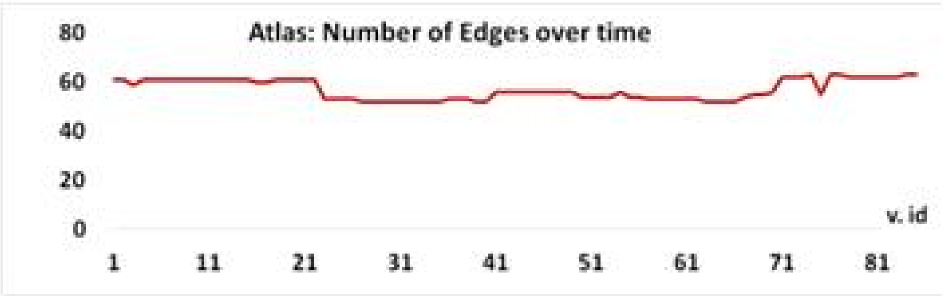


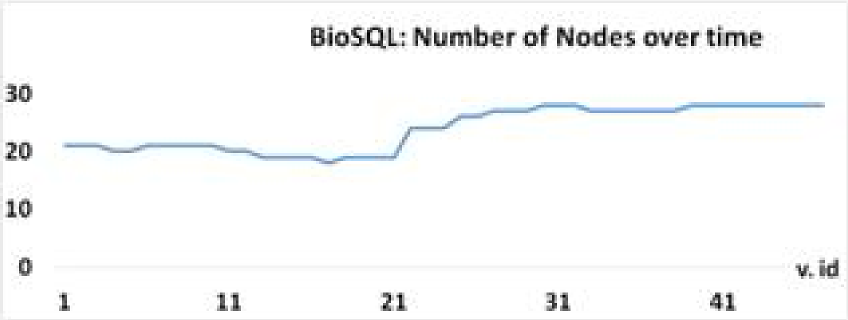

# Evolution of Tables & FK's

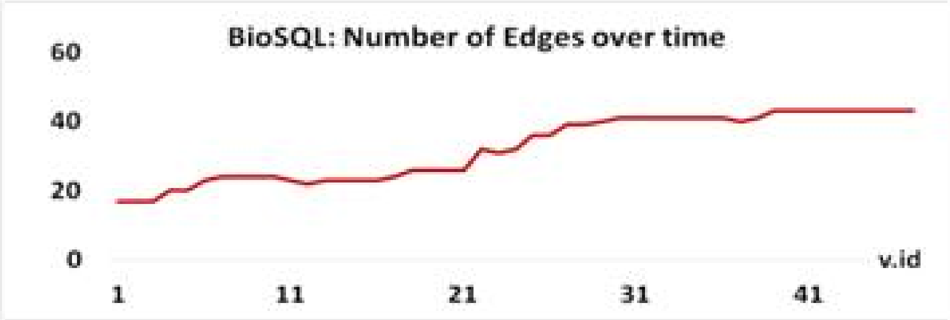
- Tables grow in all cases (known from previous research) with periods of slow growth, calmness, spikes of extension, and occasional cleanups
- Foreign Keys are treated with different mentalities. 3 families:
  - Scientific
  - Comp. Toolkits
  - CMS's

# Evolution of Tables & FK’s: Scientific projects



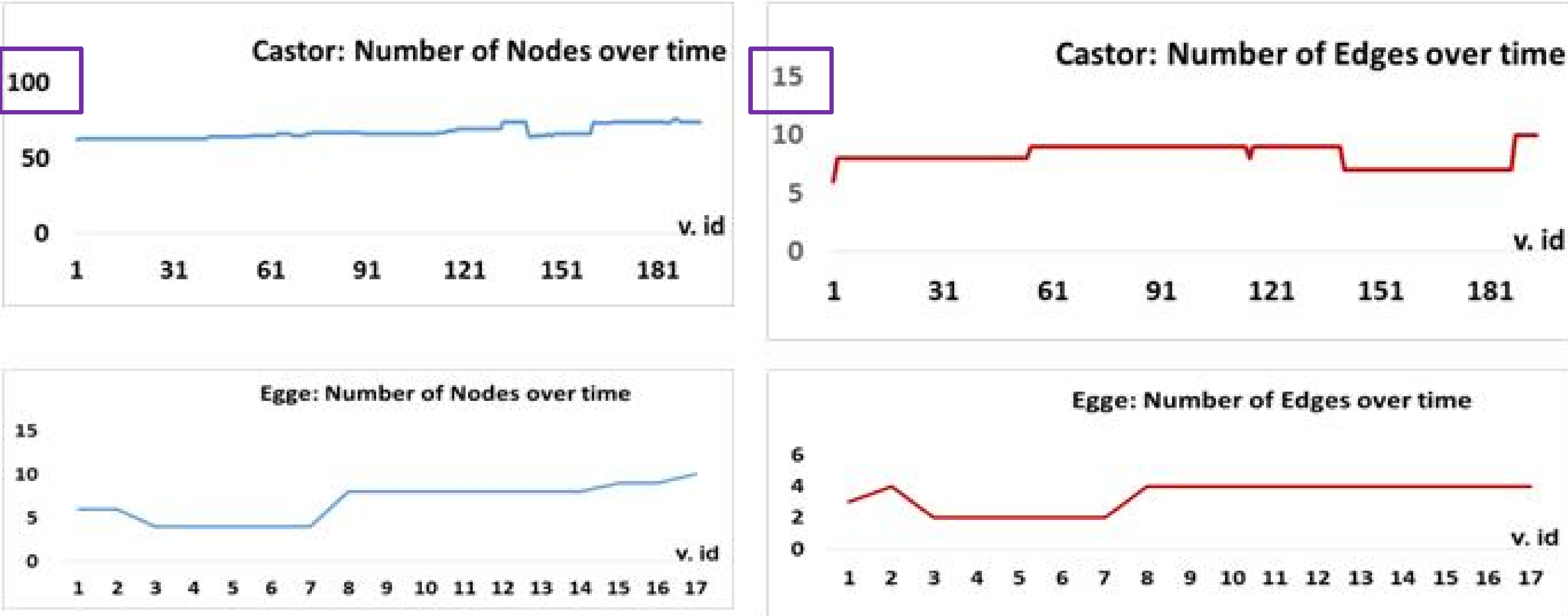



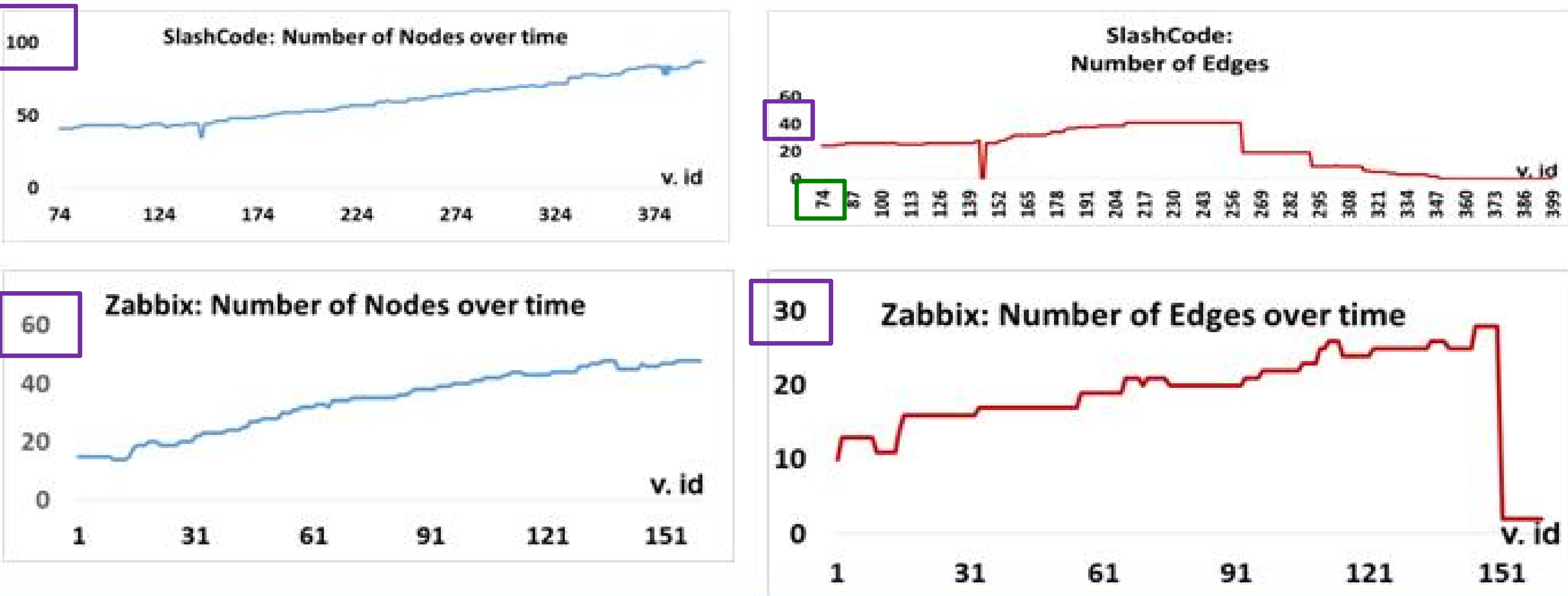

- Tables and FKS grow in synch, in both cases
- Growth comes with expansion periods, shrinkage actions, and periods of calmness in terms of both tables and foreign keys.

# Evolution of Tables & FK's: Computational Resource Toolkits



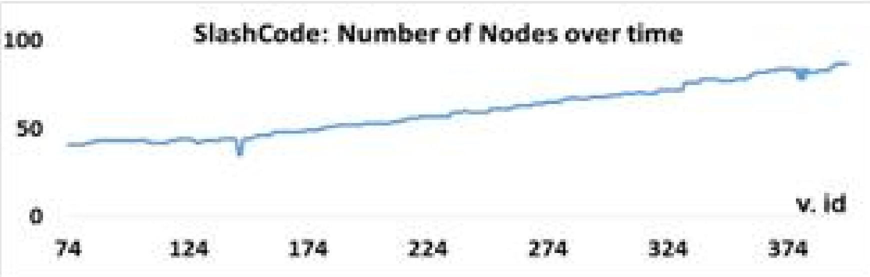

- Tables and FKS grow little and slowly; for Castor, not exactly in sync
- Castor: observe **how scarce FK's are** (too few tables come with FK's, see vertical axis)

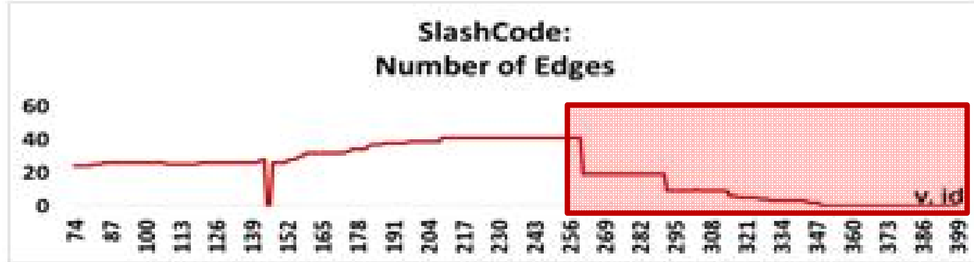
# Evolution of Tables & FK's: Content Management Systems (CMS's)

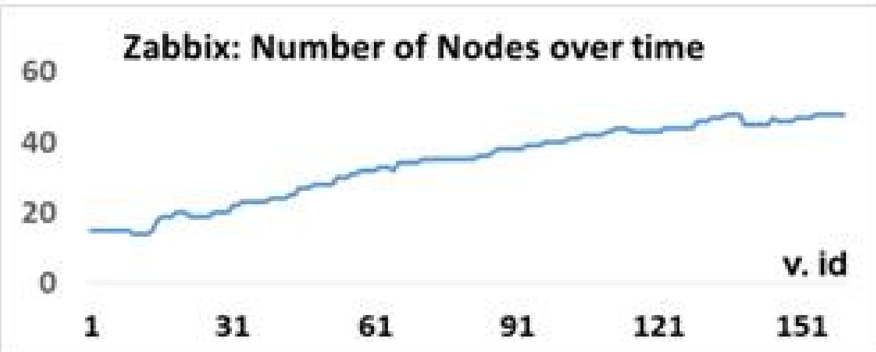


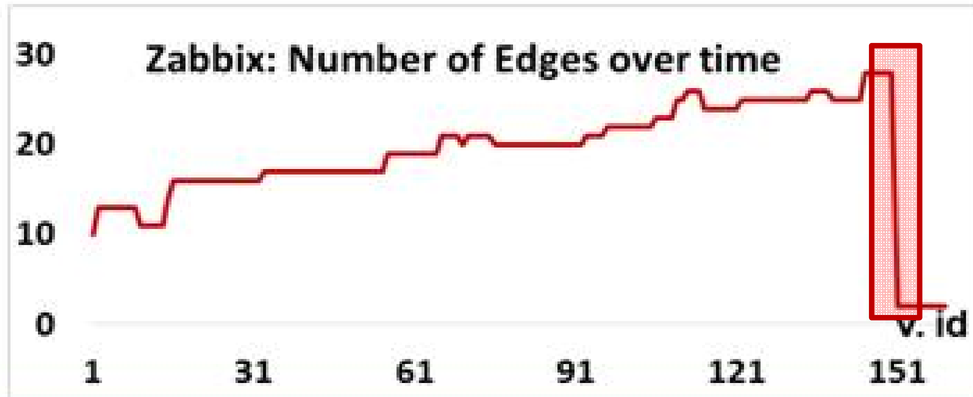

- **FK scarcity**: really big at Slashcode, moderate at Zabbix
- Slashcode started without foreign keys at all; 1st set of FK's in v. 74. Zabbix seems to show a certain degree of syncronized growth
- **Yet, … both CMS's end up with no FK's!!** *-> see next*

# What an unpleasant surprise: developers can resort in full removal of foreign keys!



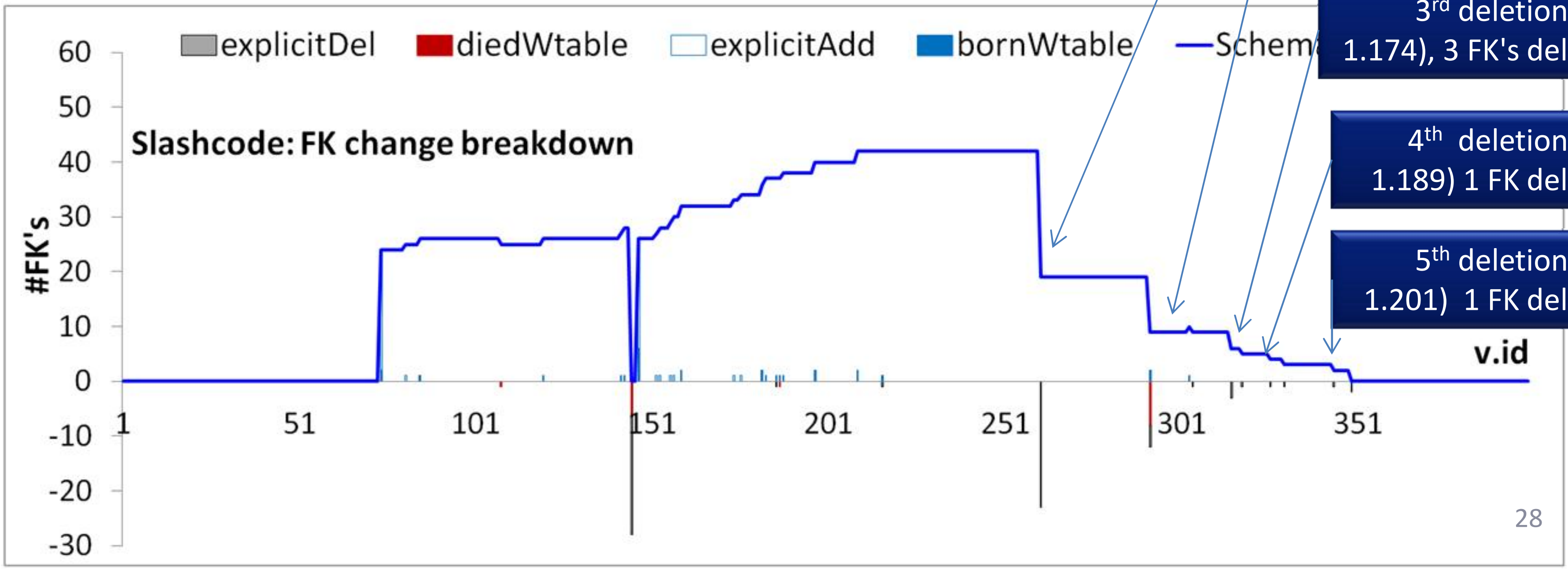




- Slashcode: there is a clear phase of **progressive removal**
- Zabbix: **abrupt removal** of almost the entire set of foreign keys in a single transition. We have <u>no knowledge on why</u> this happened, & it is <u>unexpected</u> based on how FK's had been treated till then…

# Slashcode: the disappearing FK's




1st massive foreign key removal (rev 1.120), 22 FK's deleted.

2nd massive deletion (rev 1.151), 10 FK's deleted

3rd deletion (rev 1.174), 3 FK's deleted

4th deletion (rev 1.189) 1 FK deleted

5th deletion (rev 1.201) 1 FK deleted

"Commented-out foreign keys are ones which currently cannot be used because they refer to a primary key which is NOT NULL AUTO INCREMENT and the child's key either has a default value which would be invalid for an auto increment field, typically NOT NULL DEFAULT '0'.
Or, in some cases, the primary key is e.g. VARCHAR(20) NOT NULL and the child's key will be VARCHAR(20). The possibility of NULLs negates the ability to add a foreign key. <= That's my current theory, but it doesn't explain why discussions.topic SMALLINT UNSIGNED NOT NULL DEFAULT '0' is able to be foreign-keyed to topics.tid SMALLINT UNSIGNED NOT NULL AUTO INCREMENT"

1st massive foreign key removal (rev 1.120), 22 FK's deleted.

"Stories is now InnoDB and these other tables are still MyISAM, so no foreign keys between them."

2nd massive deletion (rev 1.151), 10 FK's deleted

"This doesn't work, makes createStory die. These don't work, should check why..."

3rd deletion (rev 1.174), 3 FK's deleted

"This doesn't work, since in the install pollquestions is populated before users, alphabetically"

4th deletion (rev 1.189) 1 FK deleted

"This doesn't work, since discussion may be 0."

5th deletion (rev 1.201) 1 FK deleted

# Slashcode: what did the comments say?

- **The main problem seems to be the difficulty of developers with the tuning and handling of both foreign and primary keys.**
- Sometimes difficulties are hard -- e.g., different storage engines, typically due to performance reasons
- Some difficulties are complicated due to technicalities like autonumbering
- Sometimes fixes could be found with some effort (e.g., changing the order of table population, or using numeric data types for primary keys, or inserting some “goalkeeper” values at FK target table)

# Scarcity of Foreign keys

- A 2013 collection of schema histories, lists **21 data sets**, -- some have more than one target DBMS variants.

```
$ cd RESEARCH/Github/EvolutionDatasets
$ ls -d * */*
CERN         CMS's/Coppermine  CMS's/XOOPS      Med
CERN/Atlas   CMS's/DekiWiki    CMS's/Zabbix     Med/Ensembl
CERN/CASTOR  CMS's/Joomla 1.5  CMS's/e107       Med/biosql
CERN/DQ2     CMS's/NucleusCMS  CMS's/opencart   README.md
CERN/DRAC    CMS's/SlashCode   CMS's/phpBB
CERN/EGEE    CMS's/TikiWiki    CMS's/phpwiki
CMS's        CMS's/Typo3       CMS's/wikimedia
```

- **How many data sets contain foreign keys?**
- Try this (also backed by manual sampling):

```
grep -rl "FOREIGN" . >> ALL-FKs-by-grep.ascii
awk '{split($0,a,"/"); print a[2],a[3]}' ALL-FKs-by-grep.ascii  |
uniq
```

# Scarcity of Foreign keys

**- How many data sets, out of the 21, contain foreign keys?**

CERN Atlas
CERN CASTOR
CERN EGEE
CMS's SlashC
CMS's Zabbix
Med biosql

CERN DQ2
CERN DIRAC
Med Ensembl

The **6** data sets reported here

+

**DQ2** (only in the mySQL, not in the Oracle version): FK's in 19 versions out of the 55.
Starts with 2 FK's and ends with 1.

**DIRAC** (not in the production folder, only at python+mysql).
9 tables at first version, 15 tables at last version
Starts with 10 FK's, ends with 8

**Ensembl**: not able to link FK DDL files to table evolution, yet

**- 9 out of the 21 data sets do** (including 3 that are really small for harnessing valuable results, spec., Egee, DQ2, DIRAC)

**you're not welcome here...**

**#sorrynotsorry**

http://www.boldomatic.com/view/post/G_xPI

# Foreign Key Evolution comes with different treatments:

- Sometimes, **FK's are treated as an integral part of the system**, and they are born and evicted along with table birth and eviction.
- Other times, **FK's are treated as a disposable add-on**: only a small subset of the tables involved in FK's; birth and eviction of FK's rarely performed in synch with their tables. If technical difficulties arise, it is possible to witness the **complete removal of FK's** from the schema.
- Another sign of concern is that in all the CMS' we collected, **FK's are too scarce**
- More results in the paper: **stats**, **threats to validity**, and, the treatment of the **evolving schema as an evolving graph**

http://www.cs.uoi.gr/~pvassil/publications/2017_ER/

To appear in ER 2017

# How do individual tables evolve?

<u>Outline</u>

- Schema size evolution
- Foreign Key Evolution
- **Table Evolution**
- Closing Remarks

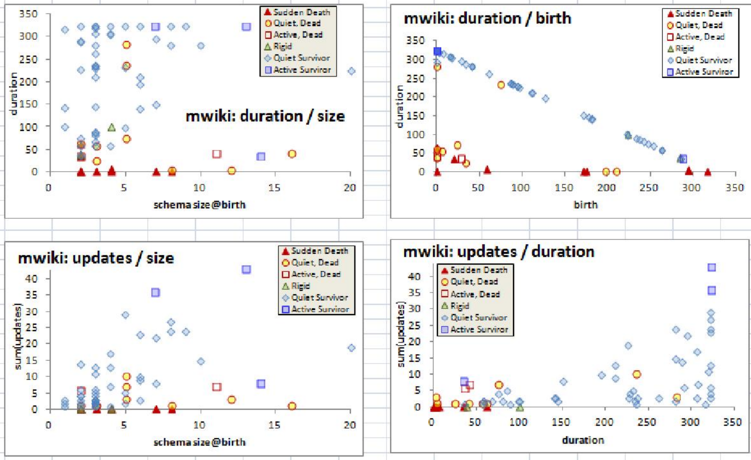
**Input:** schema histories from github/sourceforge/…

**Output**: <u>properties & patterns</u> **<u>on table properties (birth, duration, amt of change, …)</u>**

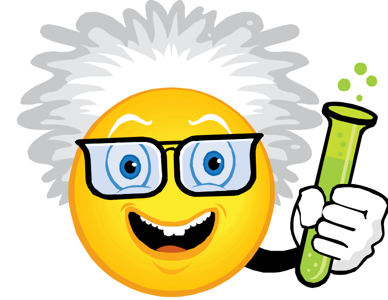
**<u>Highlights</u>**

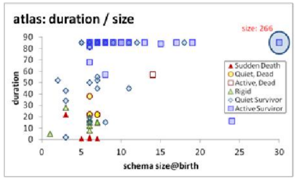
<u>4 patterns of evolution,</u>
<u>here we focus on two of them</u>

| | tableName | duration | birth | death | schema size@birth | schema size @ end | avg schema size | sum(updates) | count(updates) | ATU | UpdateRate | AvgUpdVolume | SizeScaleUp | ead/Surviv | ctivity clas | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11 | /*$wgDBPrefix*/protected_titles | 1 | 171 | 171 | 7 | 7 | 7.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 12 | blobs | 35 | 20 | 54 | 2 | 2 | 2.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 13 | brokenlinks | 62 | 0 | 61 | 2 | 2 | 2.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 14 | concurrencycheck | 1 | 317 | 317 | 4 | 4 | 4.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 15 | globalinterwiki | 3 | 294 | 300 | 2 | 2 | 2.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 16 | globalnamespaces | 3 | 294 | 300 | 3 | 3 | 3.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 17 | globaltemplatelinks | 3 | 294 | 300 | 8 | 8 | 8.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 18 | groups | 6 | 59 | 64 | 4 | 4 | 4.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 19 | imageredirects | 1 | 175 | 175 | 2 | 2 | 2.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 20 | random | 2 | 0 | 1 | 2 | 2 | 2.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 10 | 0 | 10 |
| 21 | math | 282 | 0 | 281 | 5 | 5 | 5.00 | 3 | 1 | 0.01 | 0.4% | 3.0 | 1.00 | 10 | 1 | 11 |
| 22 | links | 62 | 0 | 61 | 2 | 2 | 2.00 | 1 | 1 | 0.02 | 1.6% | 1.0 | 1.00 | 10 | 1 | 11 |
| 23 | linkscc | 57 | 6 | 62 | 3 | 2 | 2.11 | 1 | 1 | 0.02 | 1.8% | 1.0 | 0.67 | 10 | 1 | 11 |
| 24 | cur | 41 | 0 | 40 | 16 | 16 | 16.00 | 1 | 1 | 0.02 | 2.4% | 1.0 | 1.00 | 10 | 1 | 11 |
| 25 | group | 25 | 34 | 58 | 3 | 4 | 3.84 | 1 | 1 | 0.04 | 4.0% | 1.0 | 1.33 | 10 | 1 | 11 |
| 26 | trackbacks | 235 | 74 | 308 | 5 | 6 | 6.00 | 10 | 6 | 0.04 | 2.6% | 1.7 | 1.20 | 10 | 1 | 11 |
| 27 | validate | 75 | 23 | 97 | 5 | 7 | 6.37 | 7 | 3 | 0.09 | 4.0% | 2.3 | 1.40 | 10 | 1 | 11 |
| 28 | recentlinkchanges | 4 | 197 | 200 | 8 | 8 | 8.00 | 1 | 1 | 0.25 | 25.0% | 1.0 | 1.00 | 10 | 1 | 11 |
| 29 | user_restrictions | 3 | 210 | 214 | 12 | 12 | 12.00 | 3 | 1 | 1.00 | 33.3% | 3.0 | 1.00 | 10 | 1 | 11 |
| 30 | user_rights | 36 | 29 | 64 | 2 | 2 | 2.00 | 6 | 2 | 0.17 | 5.6% | 3.0 | 1.00 | 10 | 2 | 12 |
| 31 | old | 41 | 0 | 40 | 11 | 11 | 10.98 | 7 | 3 | 0.17 | 7.3% | 2.3 | 1.00 | 10 | 2 | 12 |
| 32 | config | 39 | 284 | - | 2 | 2 | 2.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 20 | 0 | 20 |
| 33 | tag_summary | 100 | 223 | - | 4 | 4 | 4.00 | 0 | 0 | 0.00 | 0.0% | | 1.00 | 20 | 0 | 20 |
| 34 | hitcounter | 316 | 7 | - | 1 | 1 | 1.00 | 1 | 1 | 0.00 | 0.3% | 1.0 | 1.00 | 20 | 1 | 21 |
| 35 | externallinks | 236 | 87 | - | 3 | 3 | 3.00 | 1 | 1 | 0.00 | 0.4% | 1.0 | 1.00 | 20 | 1 | 21 |
| 36 | text | 282 | 41 | - | 3 | 3 | 3.00 | 2 | 2 | 0.01 | 0.7% | 1.0 | 1.00 | 20 | 1 | 21 |
| 37 | transcache | 235 | 88 | - | 3 | 3 | 3.00 | 2 | 2 | 0.01 | 0.9% | 1.0 | 1.00 | 20 | 1 | 21 |
| 38 | searchindex | 323 | 0 | - | 3 | 3 | 3.00 | 3 | 3 | 0.01 | 0.9% | 1.0 | 1.00 | 20 | 1 | 21 |
| 39 | objectcache | 307 | 16 | - | 3 | 3 | 3.00 | 3 | 3 | 0.01 | 1.0% | 1.0 | 1.00 | 20 | 1 | 21 |
| 40 | user_properties | 89 | 234 | - | 3 | 3 | 3.00 | 1 | 1 | 0.01 | 1.1% | 1.0 | 1.00 | 20 | 1 | 21 |
| 41 | pagelinks | 262 | 61 | - | 3 | 3 | 3.00 | 3 | 3 | 0.01 | 1.1% | 1.0 | 1.00 | 20 | 1 | 21 |

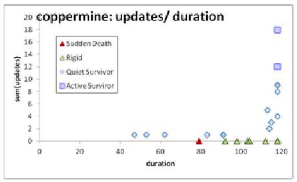


# Regularities on table change do exist!

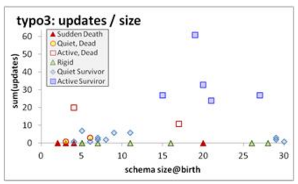




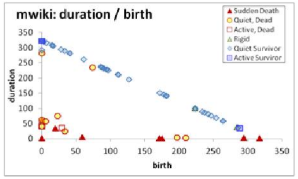

**If you're wide, you survive**

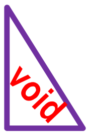



Top-changers typically live long, are early born, survive …




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



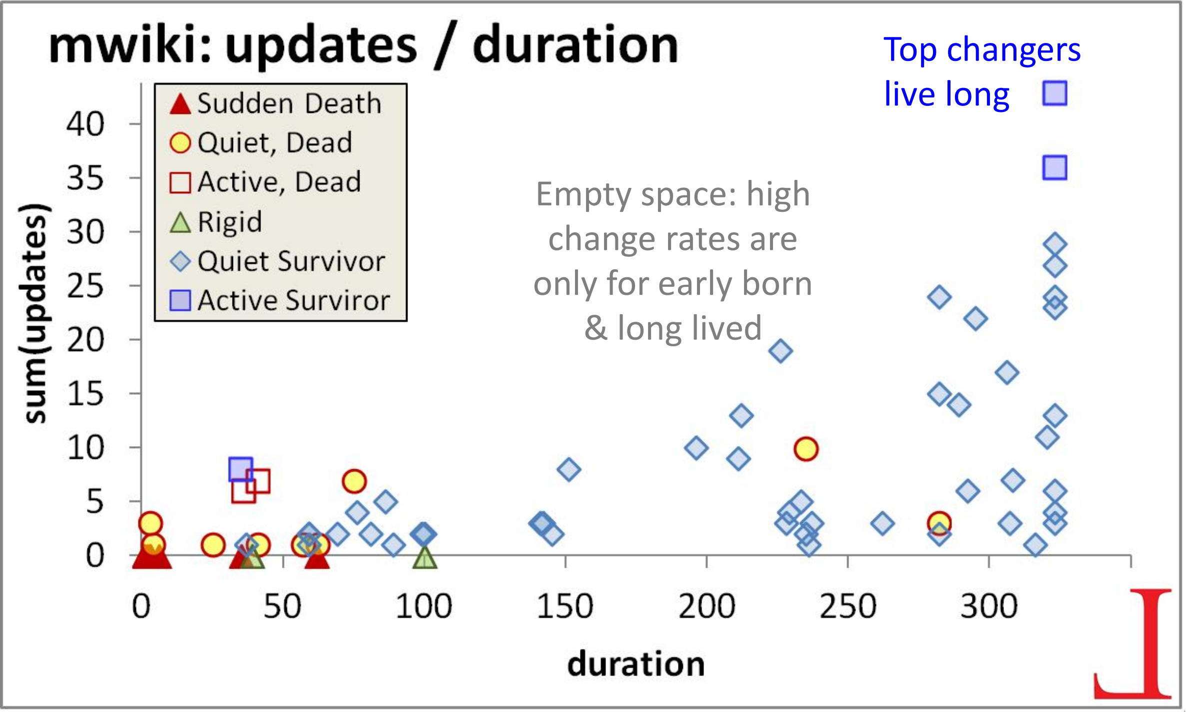



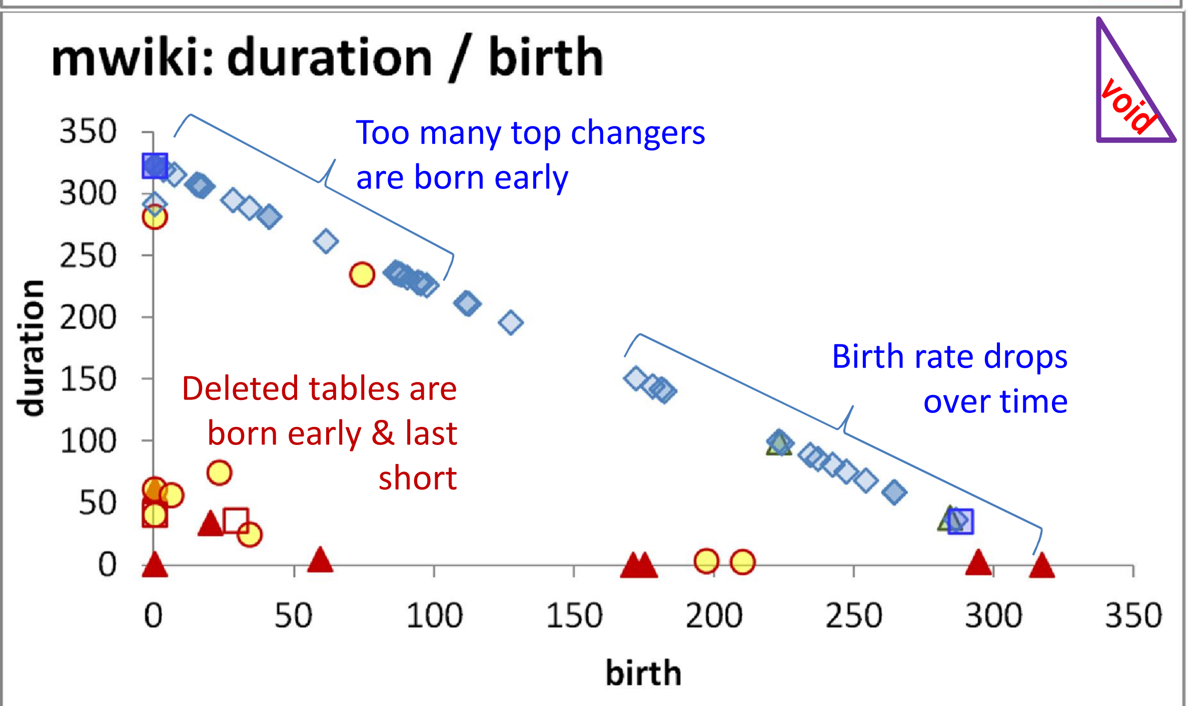

**Progressive cooling**: most change activity lies at the beginning of the db history

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

For details:
- ER 2015
- Inf. Sys. 2017

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

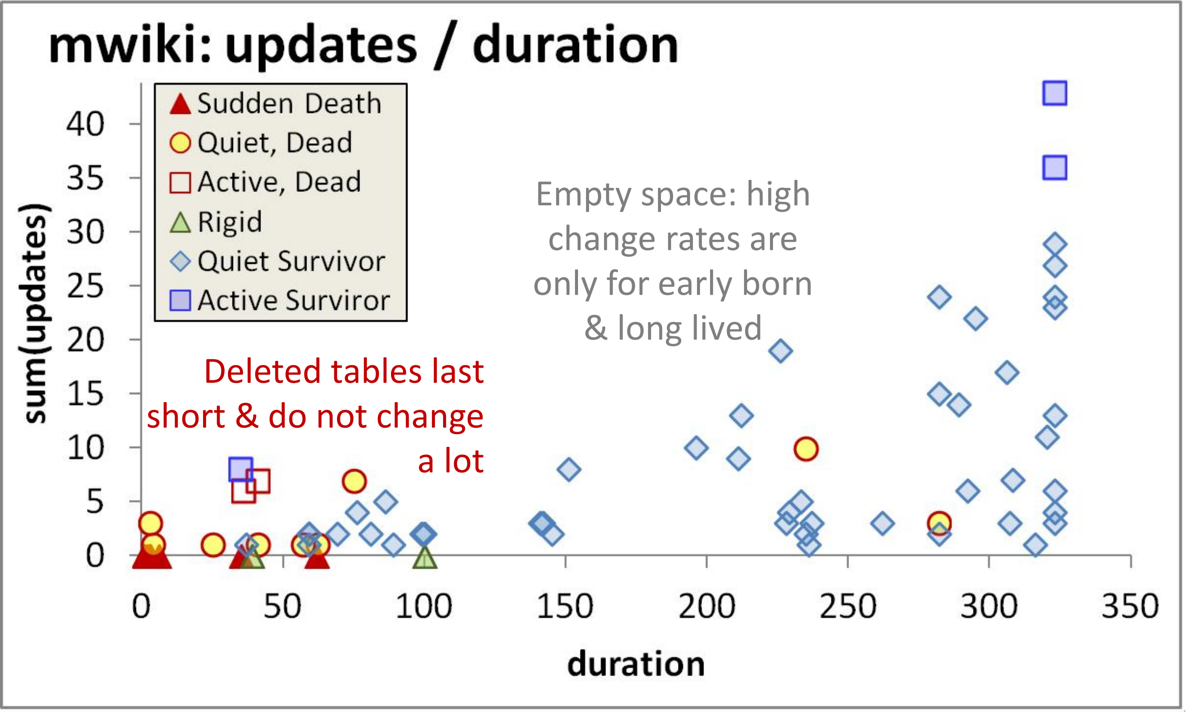


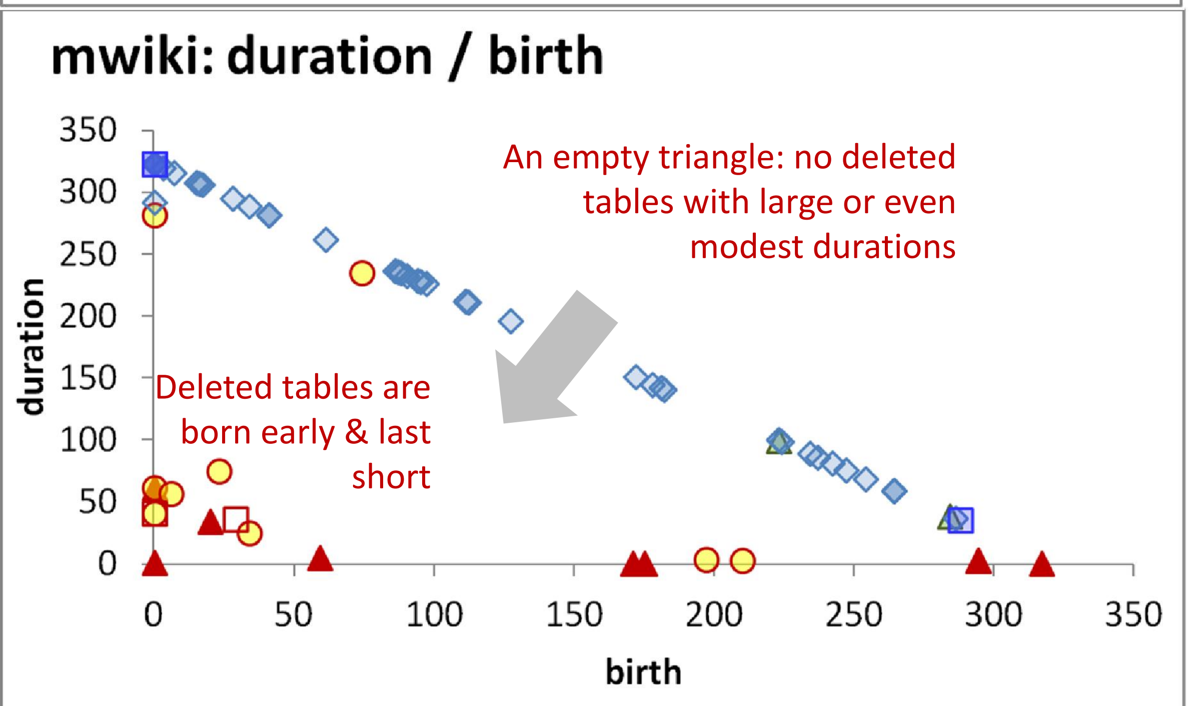




# Longevity and update activity correlate !!

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

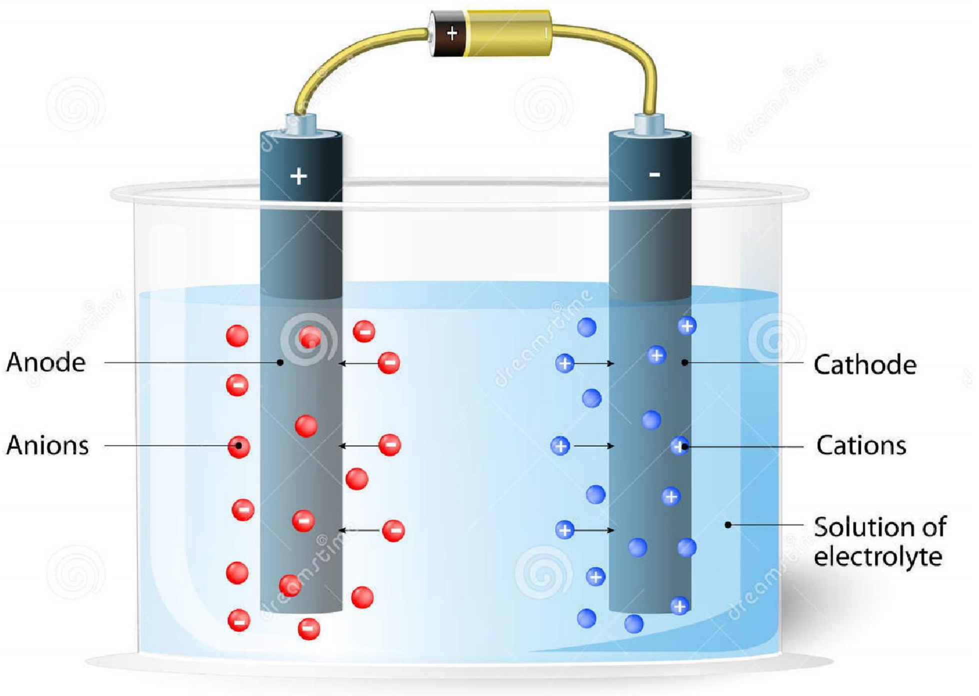
- are long lived,
- typically come from the early versions of the database
- due to the combination of high ATU and duration => they have high total amount of updates, and,
- frequently survive!




# 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).



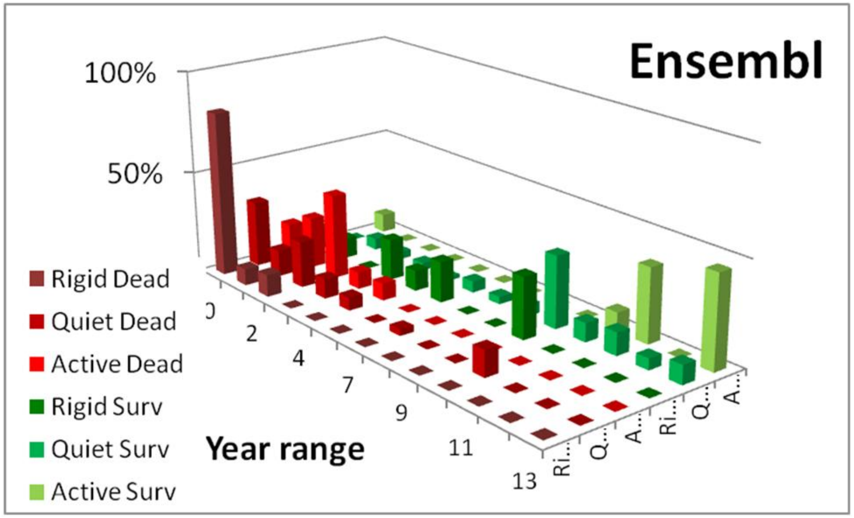


# ELECTROLYSIS PATTERN FOR TABLE ACTIVITIES

For details:
- CAiSE 2017

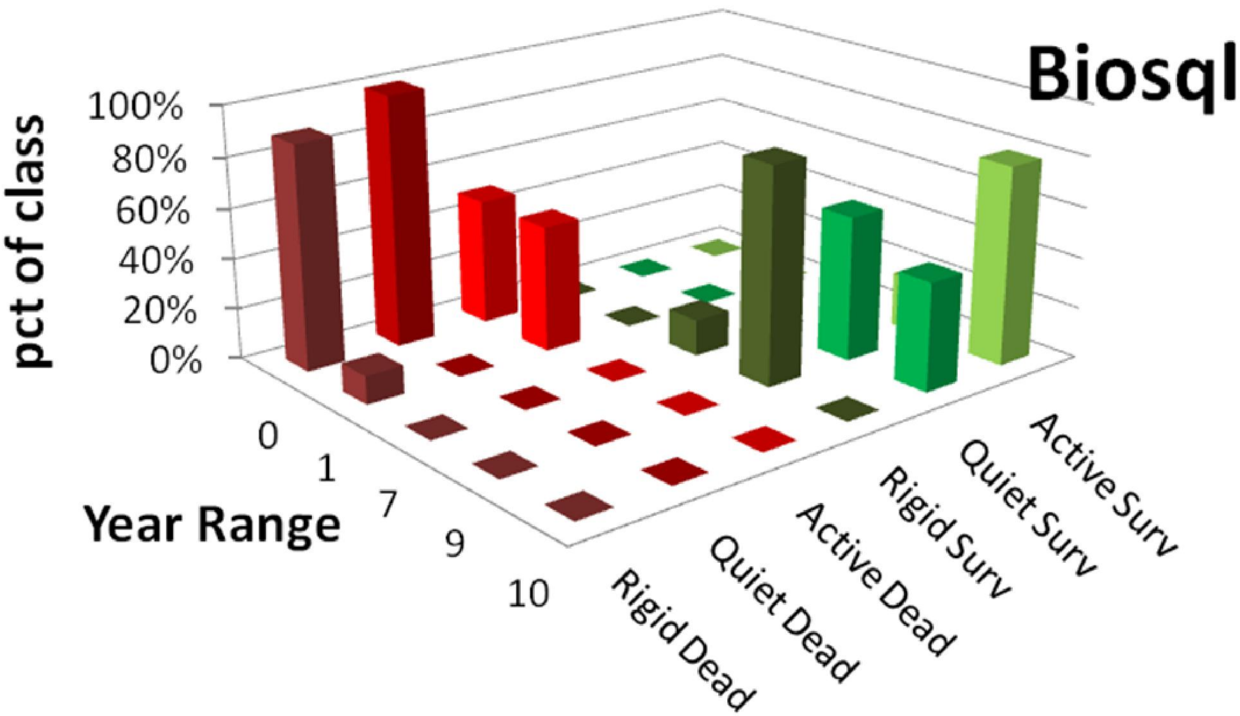
http://www.cs.uoi.gr/~pvassil/publications/2017_CAiSE_Electrolysis

# The electrolysis pattern

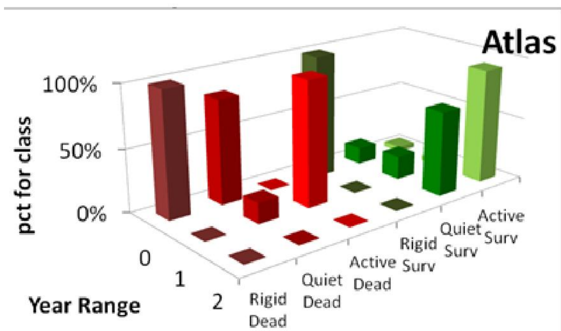


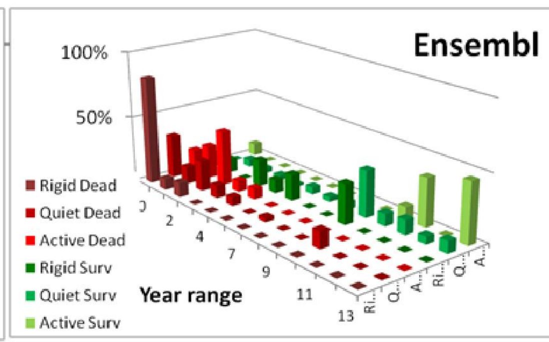

- **Dead tables demonstrate much shorter lifetimes** than survivor ones,
- can be located at short or medium durations, and **practically never at high durations**.
- With few exceptions, the **less active dead tables are, the higher the chance to reach shorter durations**.

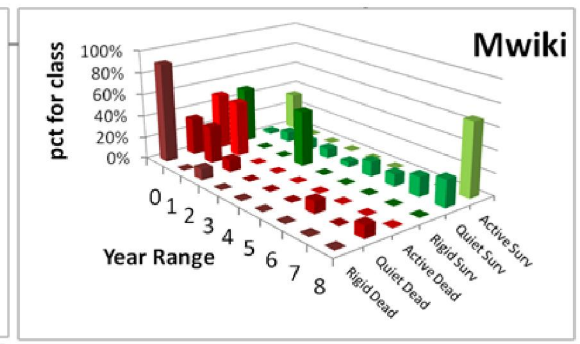
- **Survivors** expose the inverse behavior, i.e., **mostly located at medium or high durations**.
- The **more active survivors are, the stronger they are attracted towards high durations**, with a significant such inclination for the few active ones that cluster in very high durations.

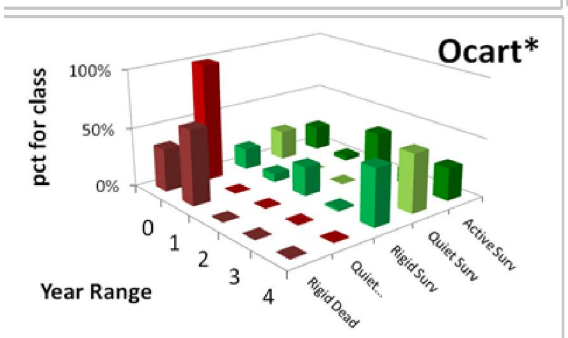


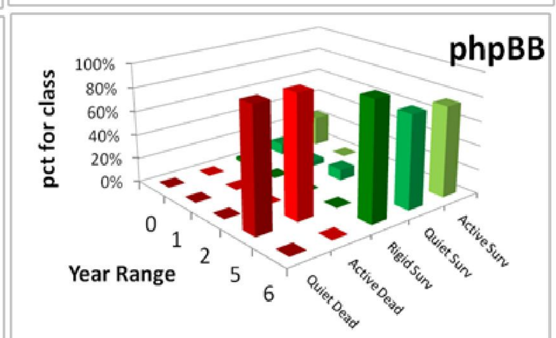

Attn: all pct's are *per class*

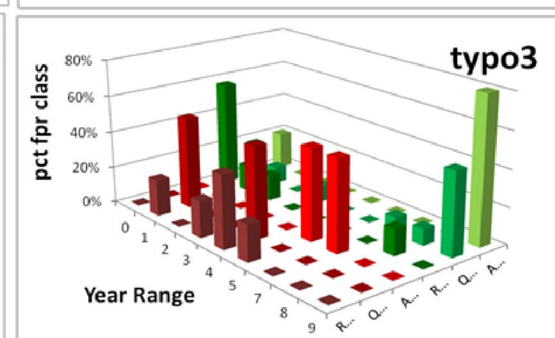


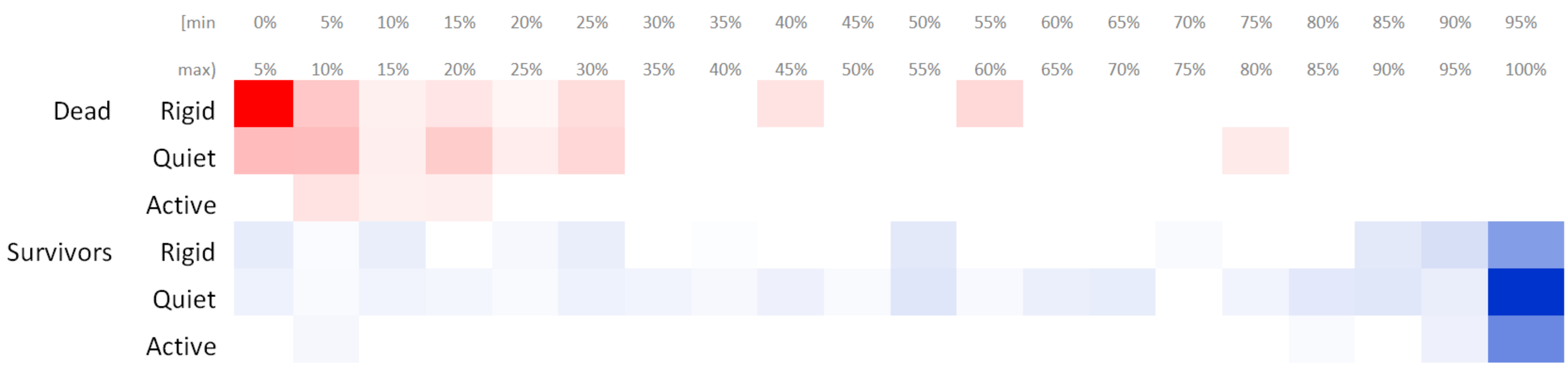



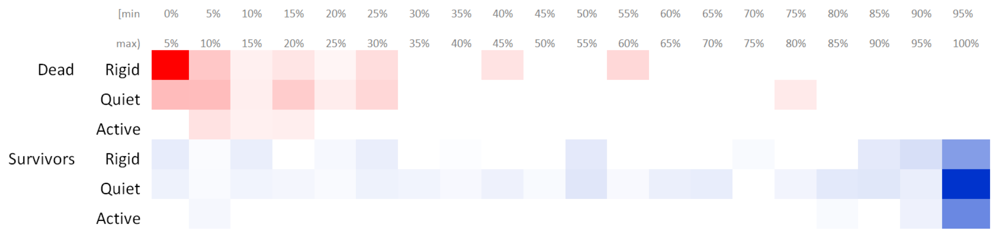
# Electrolysis as a heatmap showing the extreme bias between dead and survivor tables




- For each *LifeAndDeath* value, and for each duration range of 5% of the database lifetime, we computed the percentage of tables (over the total of the data set) whose duration falls within this range.
- We removed cells that corresponded to only one data set

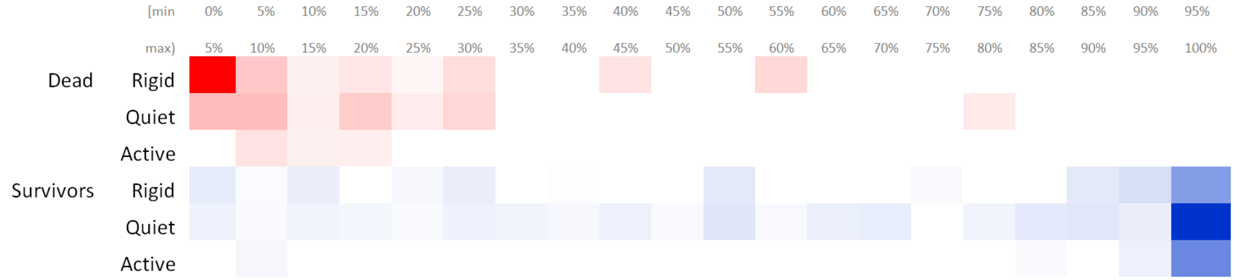
The resulting heatmap shows the polarization in colors: brighter color signifies higher percentage of the population

# Gravitation to Rigidity



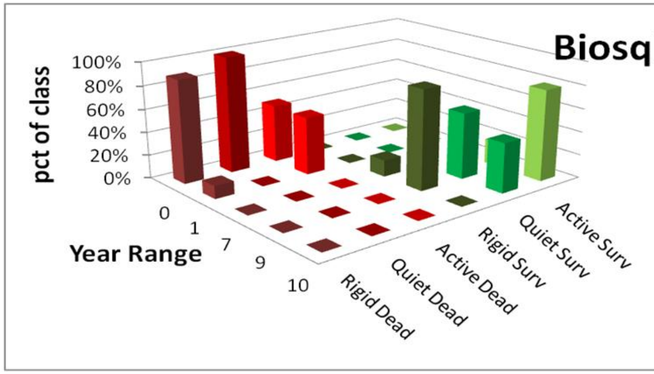

- Although the majority of survivor tables are in the quiet class, we can quite emphatically say that **it is the absence of evolution that dominates**!
  - Survivors vastly outnumber removed tables.
  - Similarly, rigid tables outnumber the active ones, both in the survival and, in particular, in the dead class.
  - Schema size is rarely resized, and only in survivors (not in the paper).
  - Active tables are few and do not seem to be born in other but early phases of the database lifetime.
- Evidently, not only survival is also stronger than removal, but **rigidity is also stronger a force than variability** and the combination of the two forces further lowers the amount of change in the life of a database schema.

# Summarizing…




- Yes, we can indeed find **patterns in the lives of tables,** during schema evolution!
- **Survivors, mostly long-lived (esp. active ones) and quietly active are radically different than dead tables, being mostly short-lived and rigid!**
- **Gravitation to rigidity rules:** we see more absence than presence of schema evolution!

Also studied [not part of the paper]: year of birth, schema size, schema resizing




http://www.cs.uoi.gr/~pvassil/publications/2017_CAiSE_Electrolysis

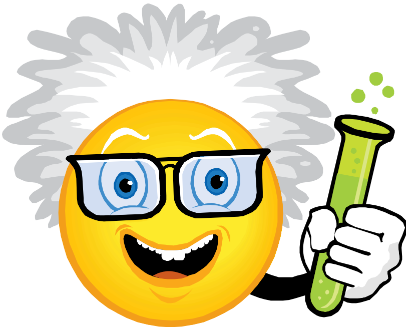
Outline
- Schema size evolution
- Foreign Key Evolution
- Table Evolution
- **Closing Remarks**

Where we stand

Open issues

… and discussions …

# CLOSING REMARKS



# Where we stand

We have a first understanding of …

- gravitation to rigidity, i.e., the mechanics of schema non-evolution for FoSS ecosystems
- schemata growing, changed in focused periods of maintenance and progressively "cooling" down
- patterns relating to how tables change, given their size, update behavior, time of birth, …
- foreign key families of treatment, absence & removals

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

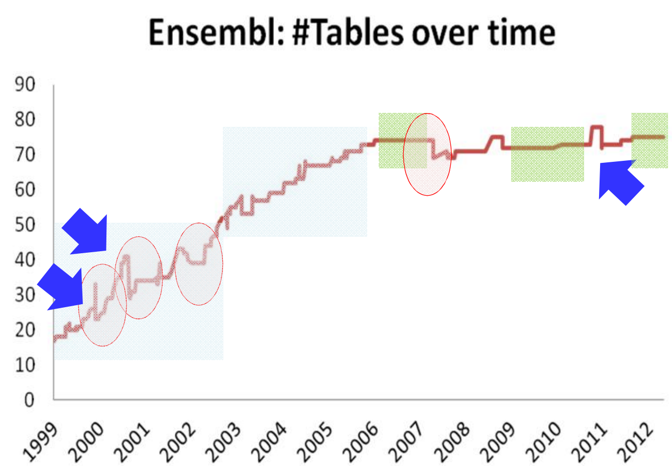
http://www.cs.uoi.gr/~pvassil/projects/schemaBiographies/

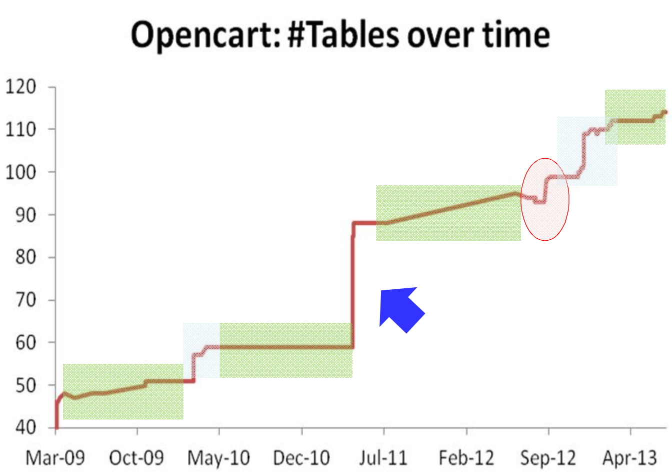
**Gravitation to rigidity:**

- Long calmness, low+focused growth
- Empty triangle, inverse Gamma, electrolysis

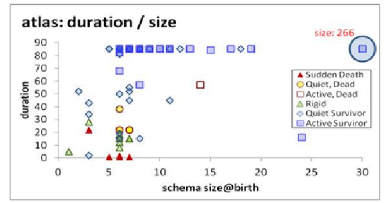
**More absence than presence of evo!**

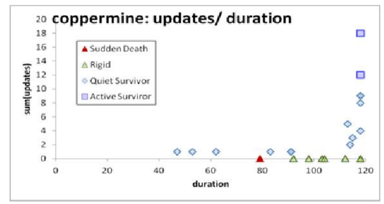
## Schema size

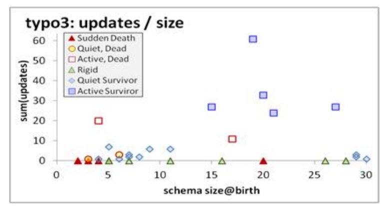


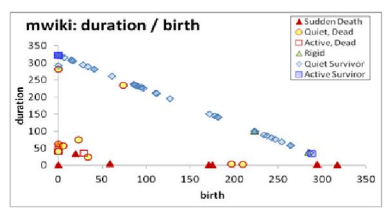



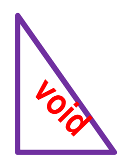

## Individual Tables

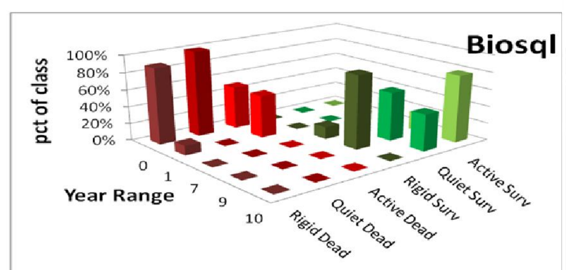


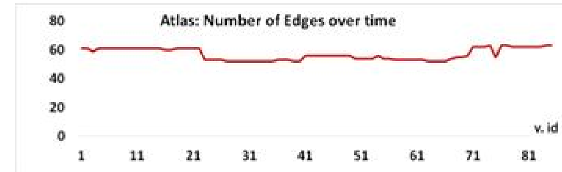



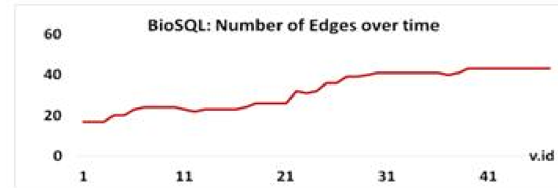



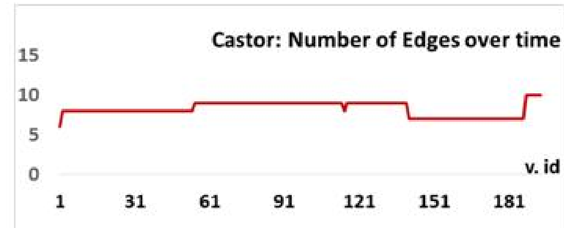



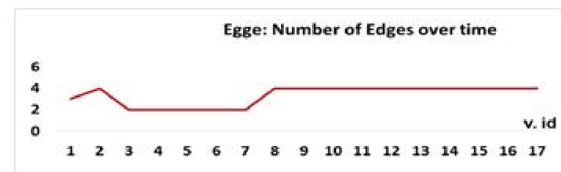



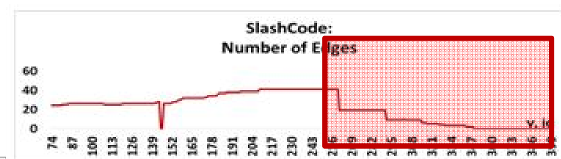

## Foreign Keys

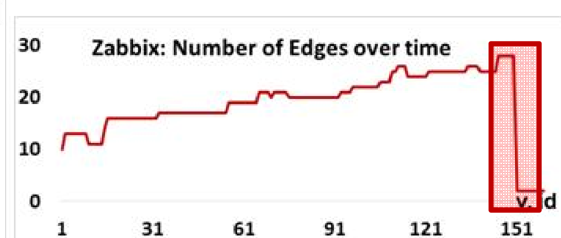


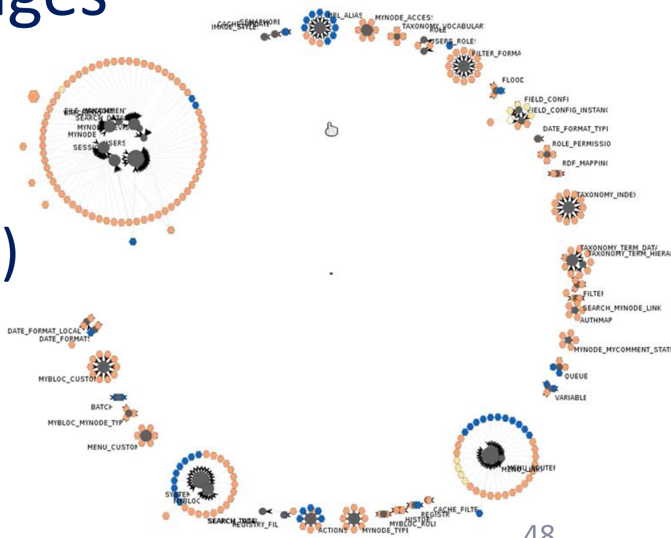
# Where to go from here…

- More studies, by more groups, on more data, to verify / disprove patterns & find new ones
- More tools and techniques to fully automate processing
- Weather Forecast: given the history and the state of a database, predict subsequent events
- 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; …

# How does schema evolution relate to the surrounding software?

- Which parts of the surrounding data-intensive software app's are most sensitive to evolution?
  - Metrics for sensitivity to evolution?
  - Visualization of the architecture & evolution impact

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



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

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

https://github.com/DAINTINESS-Group

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

**Everything HAS TO BE online!**

We are happy to invite you to reuse / test / disprove /… all our code, data and results!

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

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

# Thank you!

- Yes, **we have the data and the tools** to find **patterns of schema evolution** both for the entire schema and for individual parts of it!
- **Gravitation to rigidity rules:** we see more absence than presence of schema evolution!
- Many opportunities to exploit data, code and results for research on more studies, design and visualization of systems

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

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

# AUXILIARY SLIDES

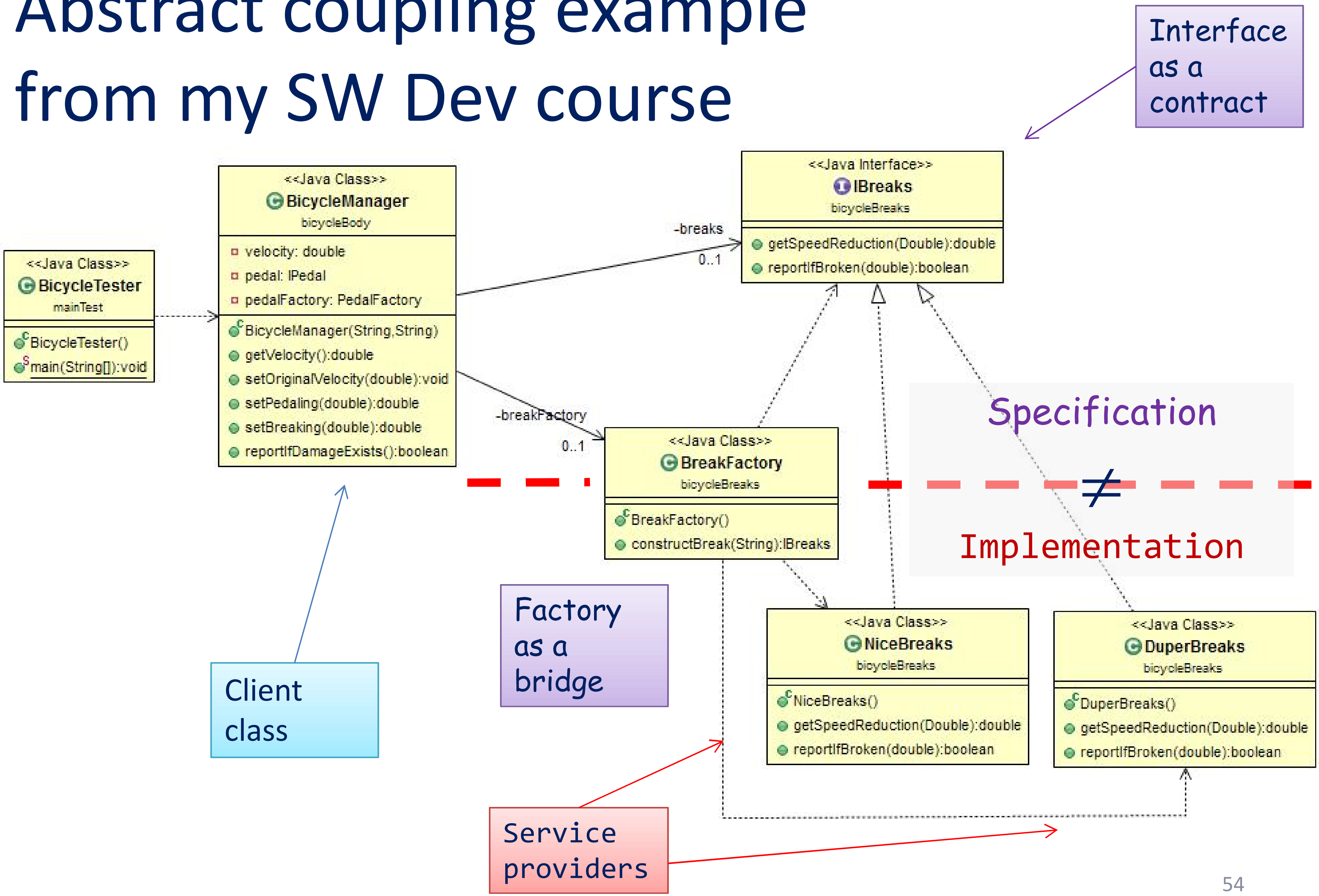
# Embedded queries in the past [Maule+08] …

```
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)

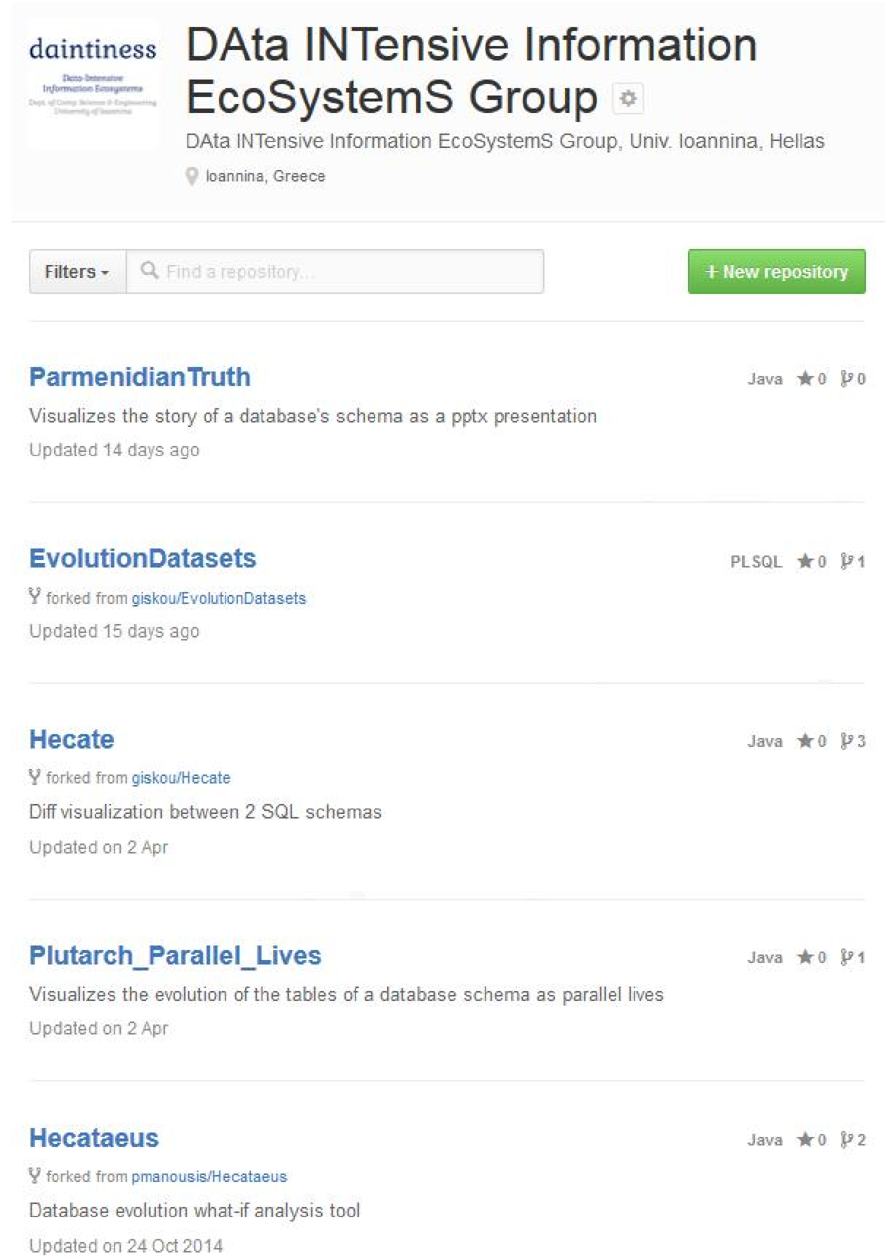
```
function _profile_get_fields($category, $register = FALSE) {
  $query = db_select('profile_field');
  if ($register) {
   $query->condition('register', 1);
  }
  else {
   $query->condition('category', db_like($category), 'LIKE');
  }
  if (!user_access('administer users')) {
   $query->condition('visibility',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

# Abstract coupling example from my SW Dev course



# Put it all online!!

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




My web page

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

has links to …

DB Schema Evolution

Papers, **Data sets, Code, Results**

projects/schemaBiographies/

… and to …

Tools for handling Evolution (Hecataeus)

projects/hecataeus/

# SCOPE OF THE STUDY && VALIDITY CONSIDERATIONS

# 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 |

# 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!

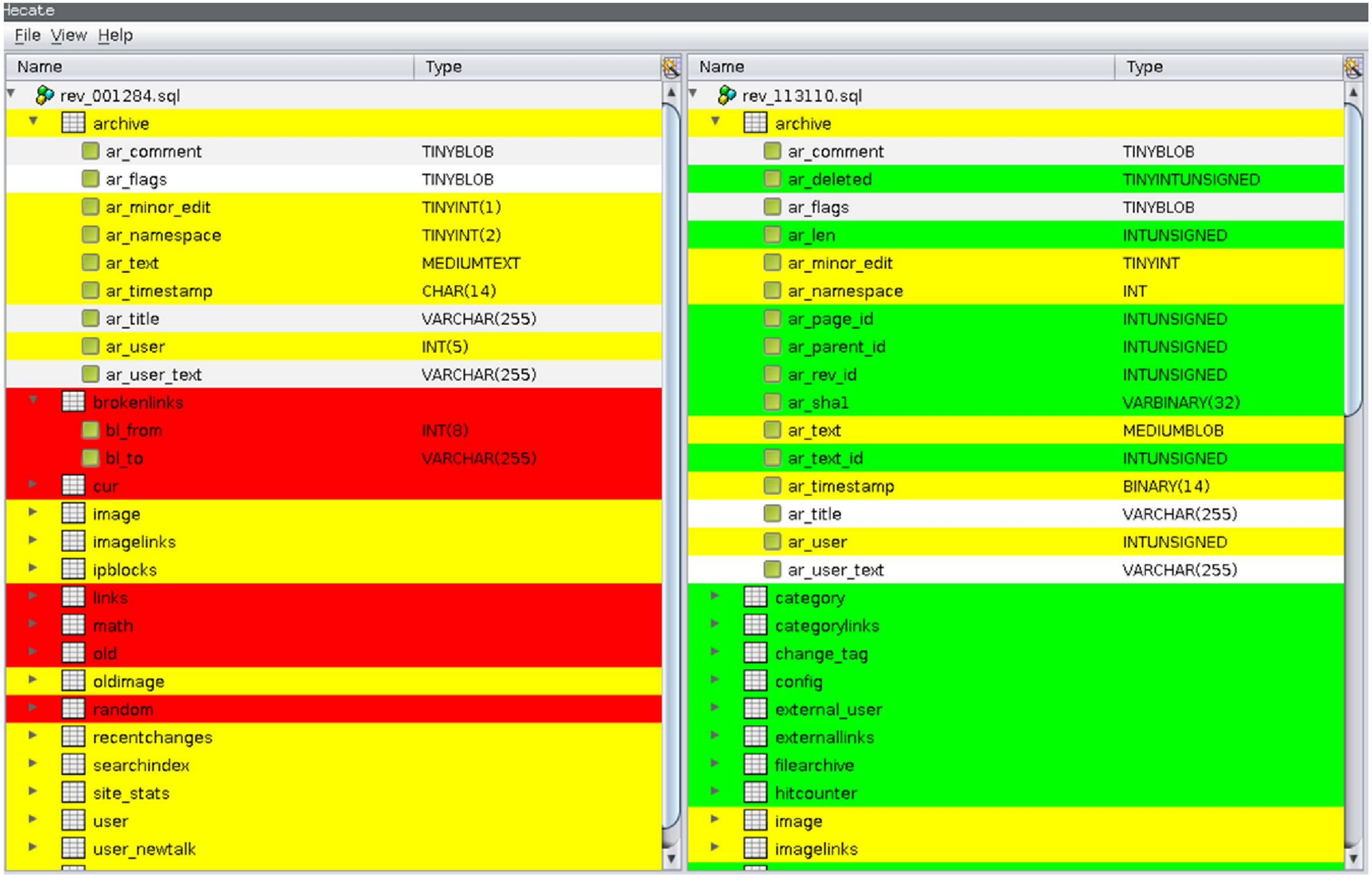
| FoSS Dataset | Versions | Lifetime | Tables @ Start | Tables @ End |
|---|---|---|---|---|
| ATLAS Trigger [A] | 84 | 2 Y, 7 M, 2 D | 56 | 73 |
| BioSQL [B] | 46 | 10 Y, 6 M, 19 D | 21 | 28 |
| Coppermine [C] | 117 | 8 Y, 6 M, 2 D | 8 | 22 |
| Ensembl [B] | 528 | 13 Y, 3 M, 15 D | 17 | 75 |
| MediaWiki [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

- 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

# Hecate: SQL schema diff extractor




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

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

# 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 hint that we suspect the causes for a particular phenomenon, in some places in the text, but we have no data, yet, to verify our gut-feeling.
  – 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):

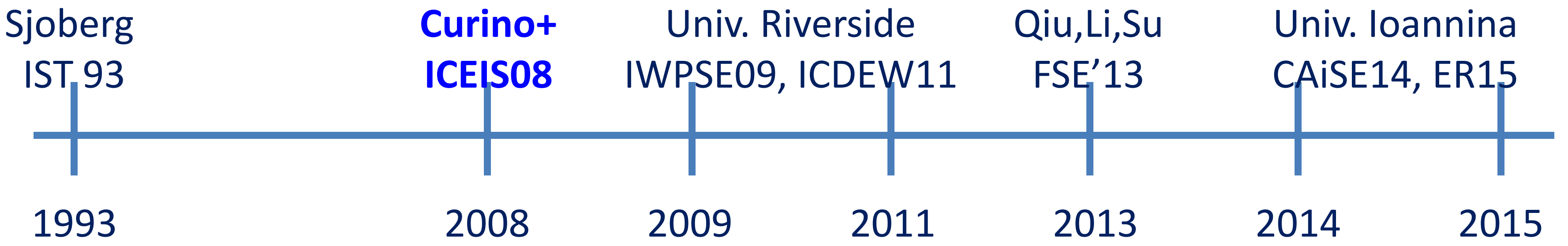
- relation addition: 19 changes ; attribute additions: 2 changes
- relation deletion : 59.5 changes; attribute deletions:  3.25 changes

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.

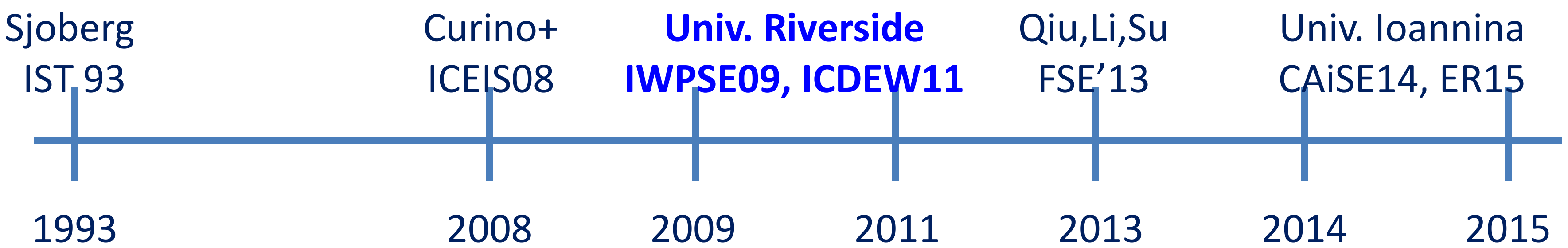
45% of changes do not affect the information capacity of the schema (but are rather index adjustments, documentation, etc)



# 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

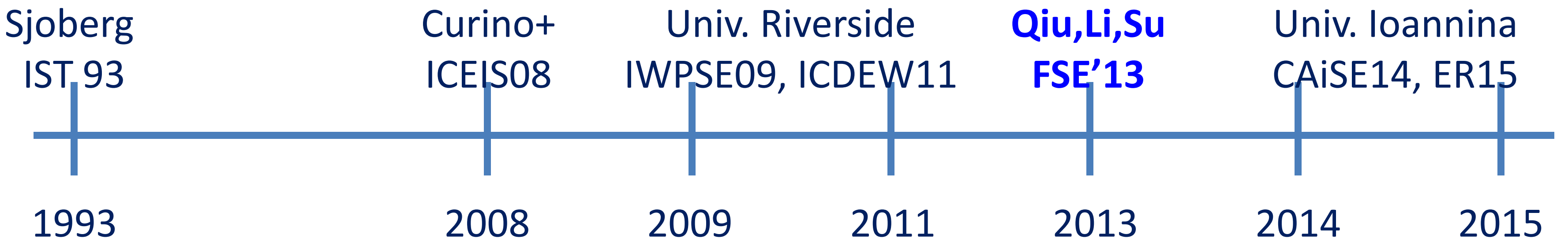


# Timeline of empirical studies

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**

**.. 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 AND LEHMAN LAWS

# Exploratory search of the schema histories for patterns

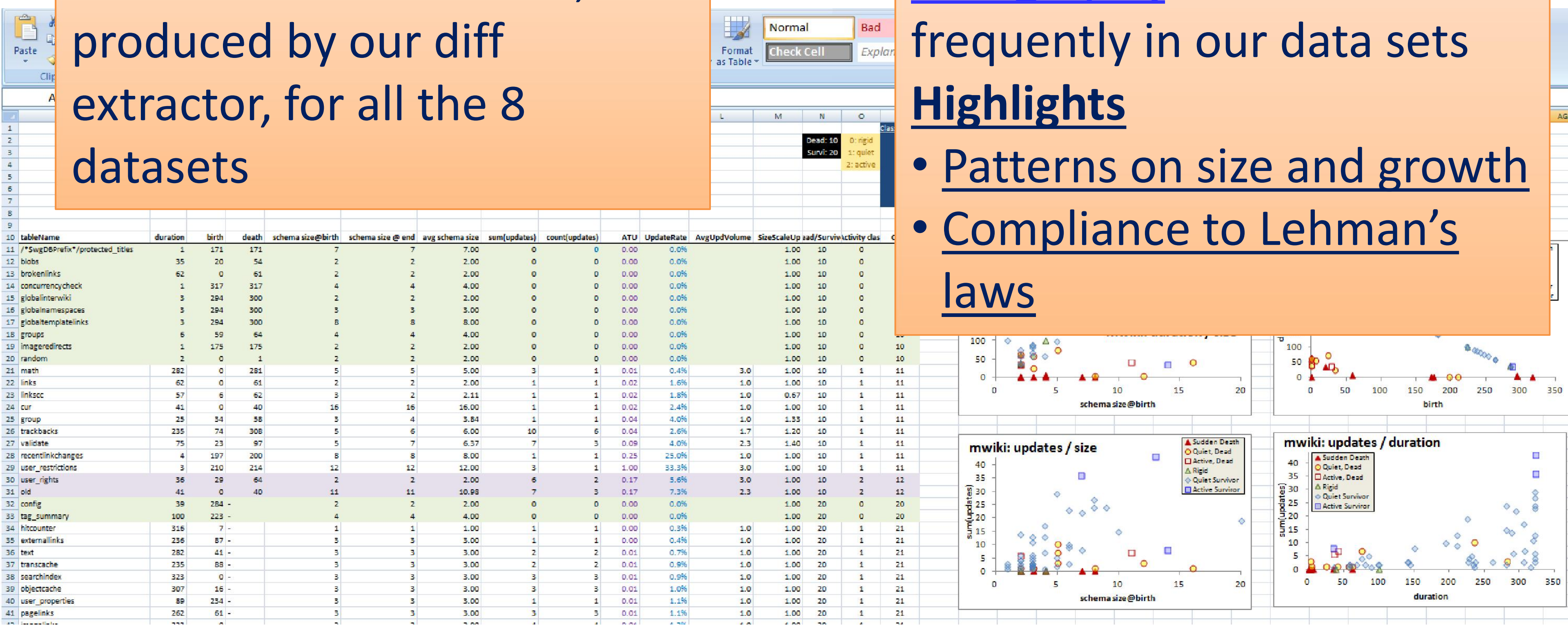
**Input:** schema histories from github/sourceforge/…

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

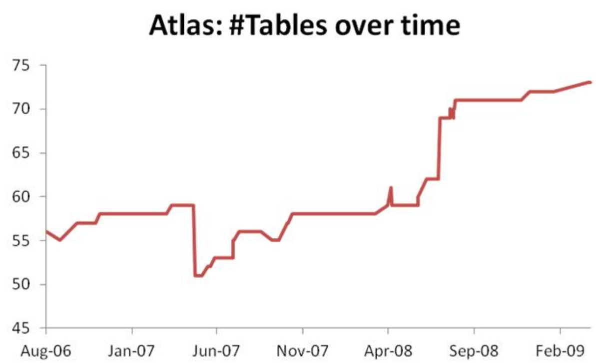
**Output**: properties & patterns on **schema properties (size, growth, changes, …)** that occur frequently in our data sets

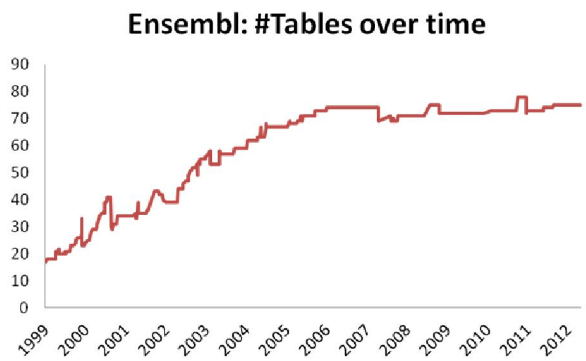
**Highlights**

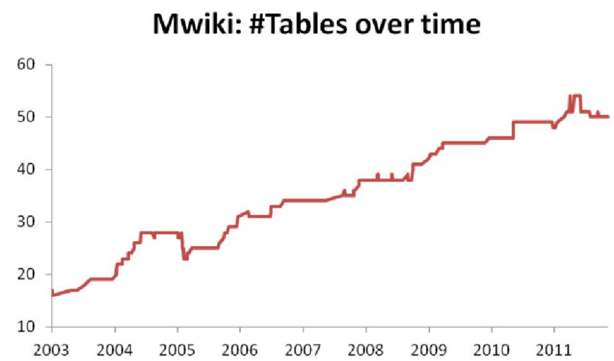
- Patterns on size and growth
- Compliance to Lehman's laws

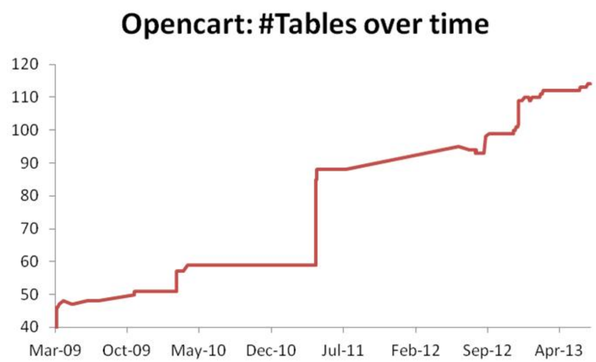


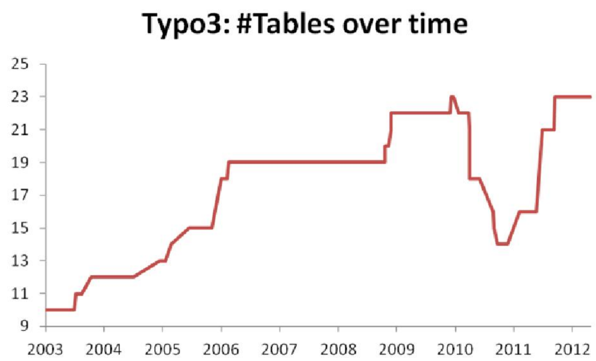
# Schema Size (relations)

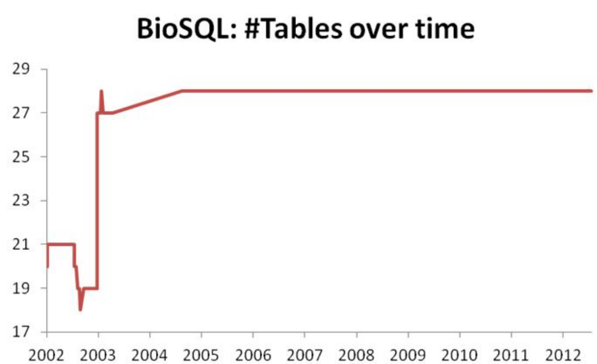


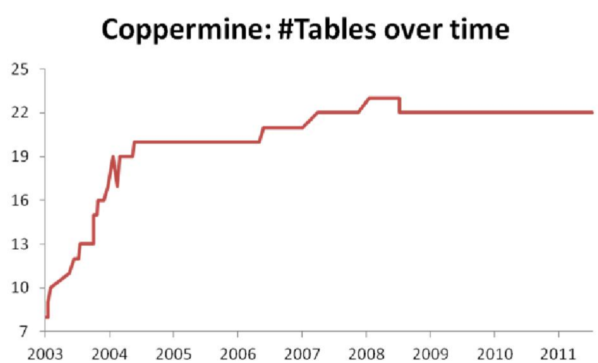



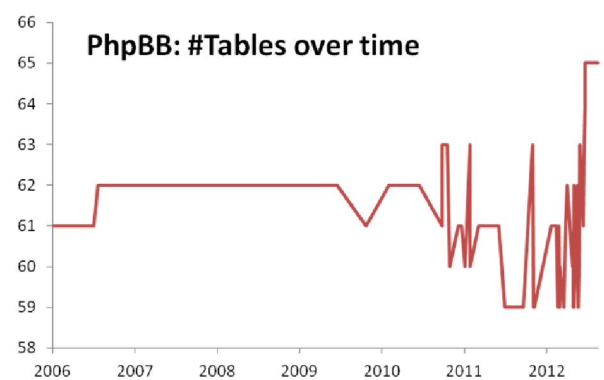



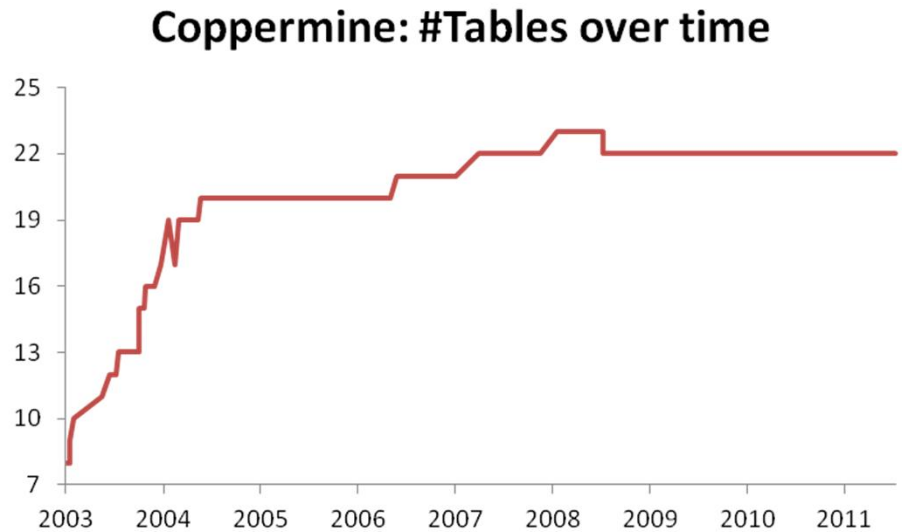



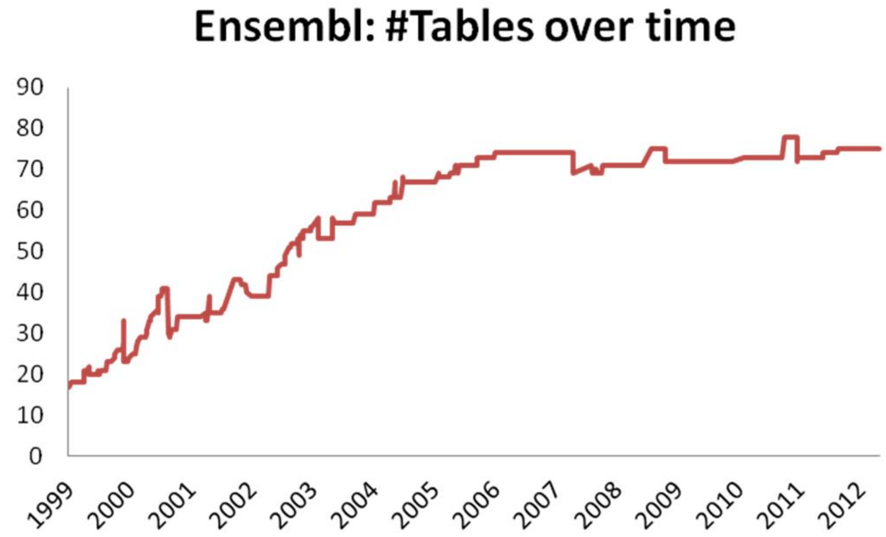



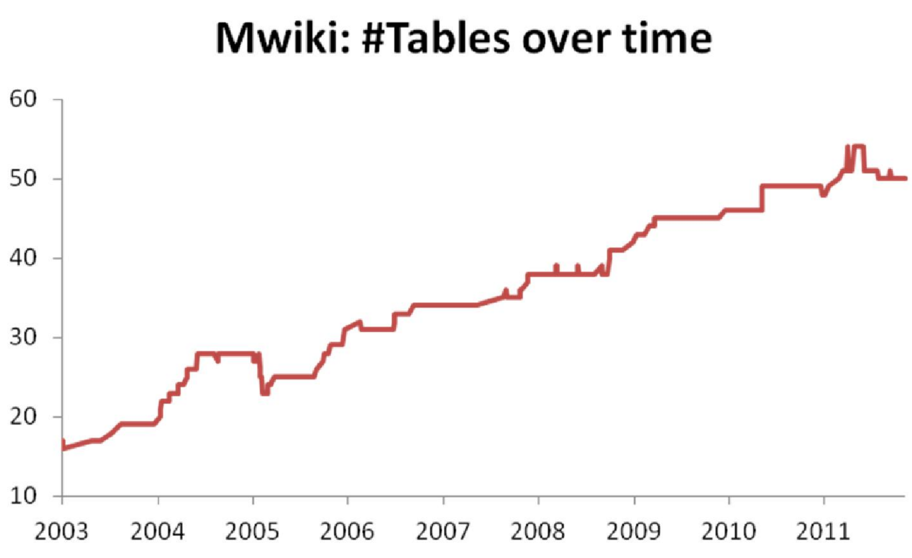

http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

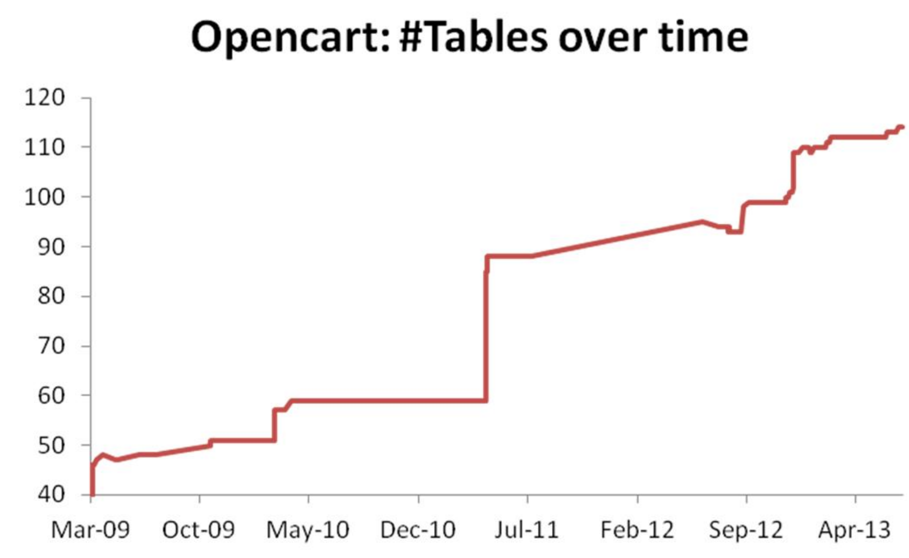
# Schema Size

- **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…

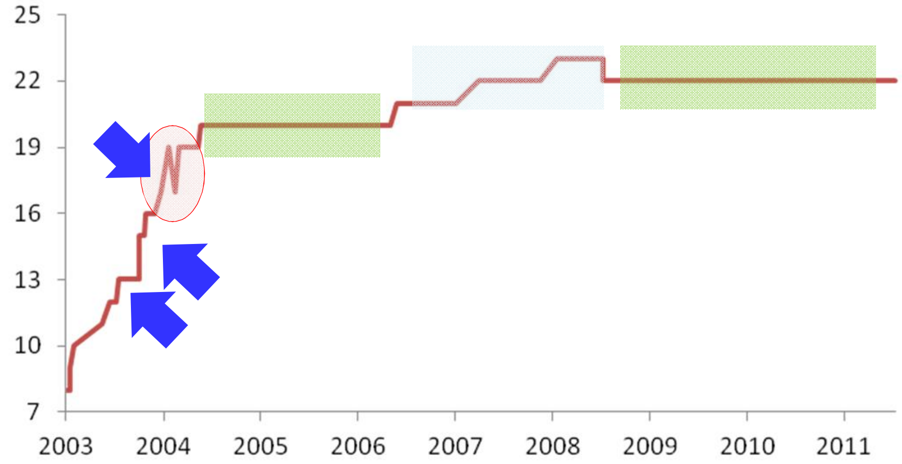

Coppermine: #Tables over time
25
22
19
16
13
10
7
2003
2004
2005
2006
2007
2008
2009
2010
2011


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

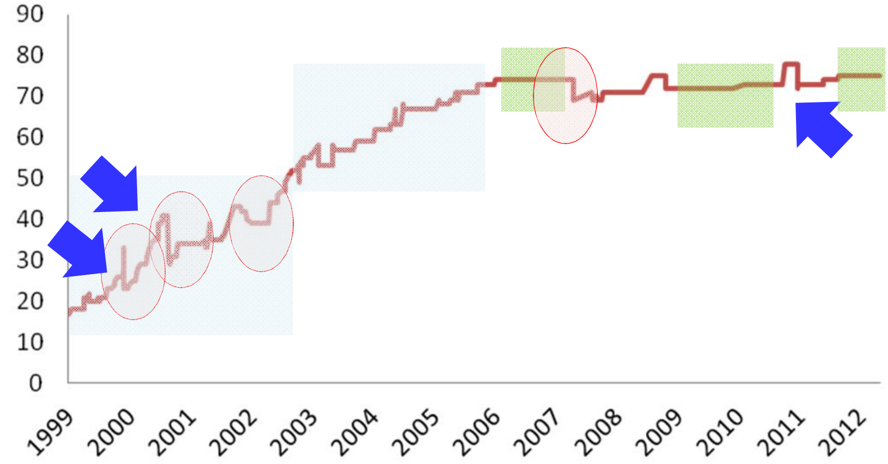

Mwiki: #Tables over time
60
50
40
30
20
10
2003
2004
2005
2006
2007
2008
2009
2010
2011


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



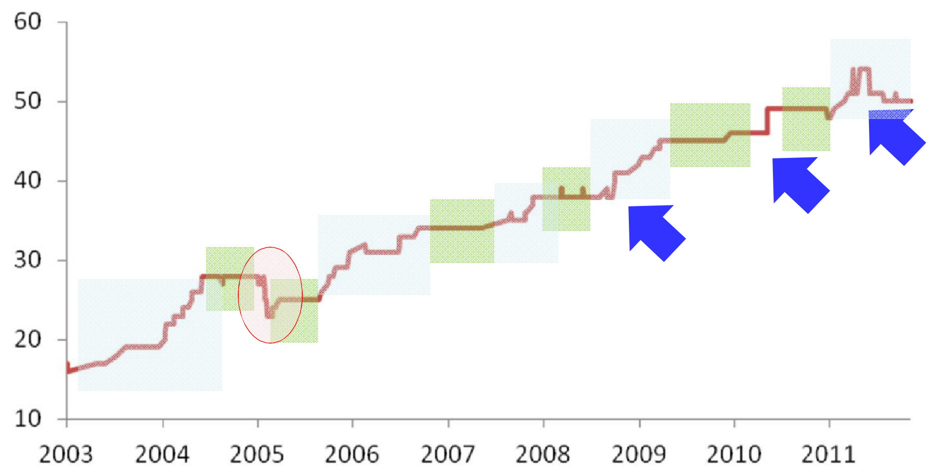

## Ensembl: #Tables over time



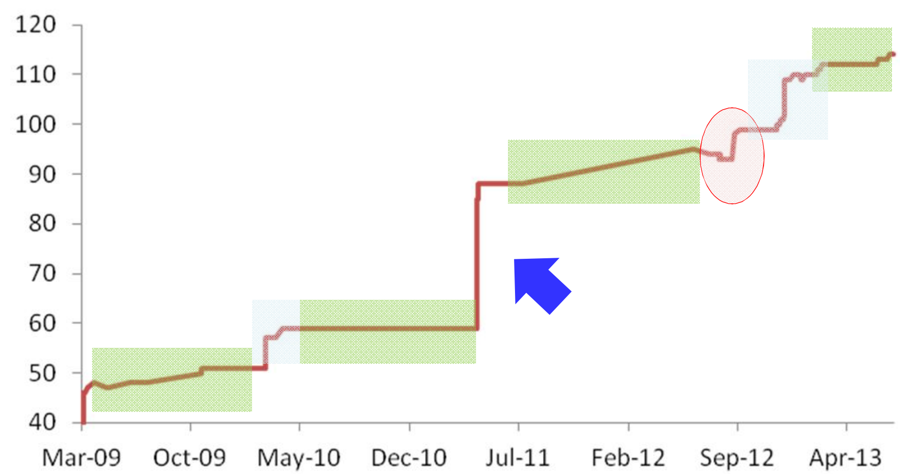

Growth over time
**Calmness periods**

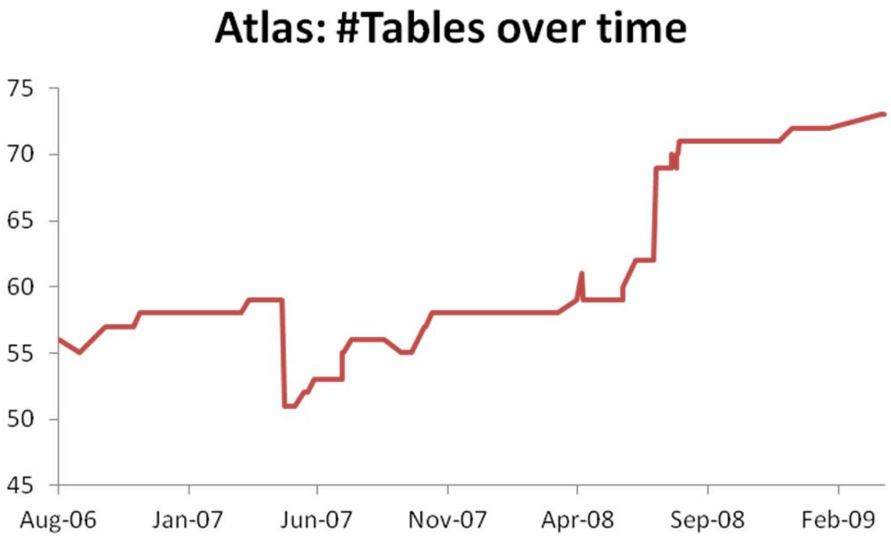
Increase both slow (mostly) and **abrupt**
Occasional abrupt drops (maintenance)

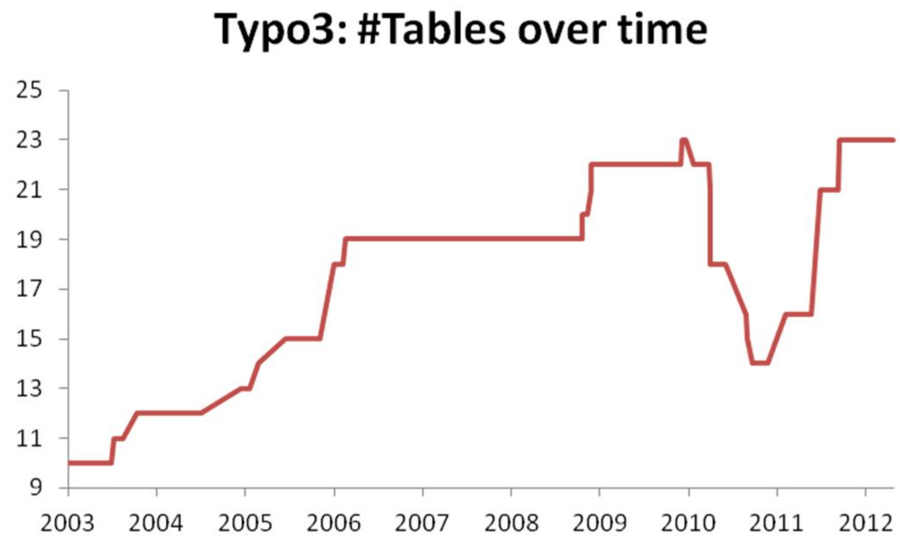
## Mwiki: #Tables over time

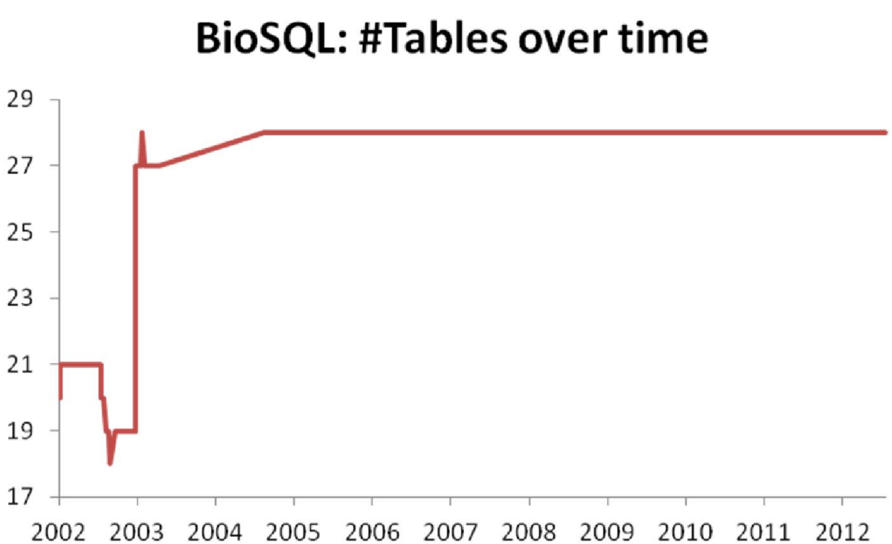


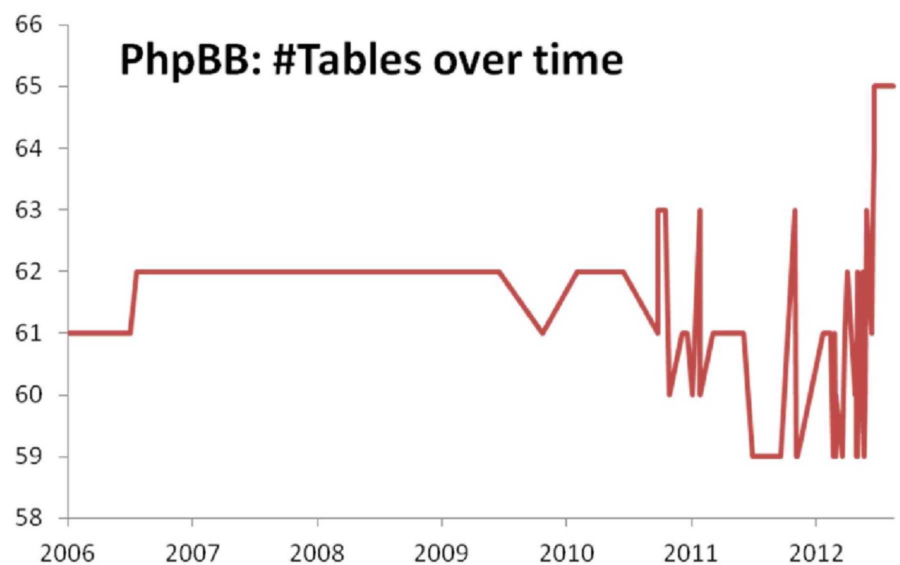

## Opencart: #Tables over time



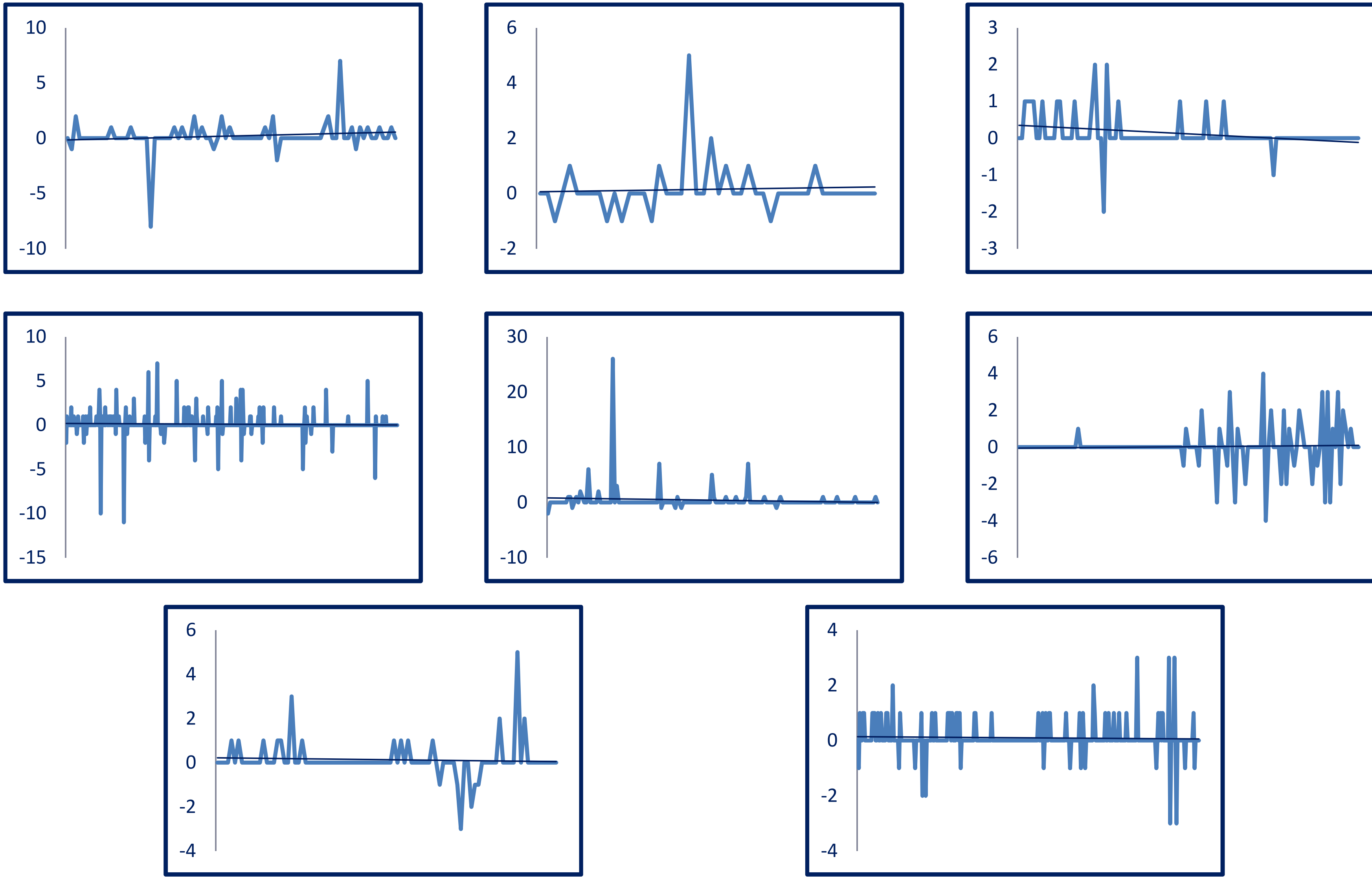

Atlas: #Tables over time
75
70
65
60
55
50
45
Aug-06
Jan-07
Jun-07
Nov-07
Apr-08
Sep-08
Feb-09


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)

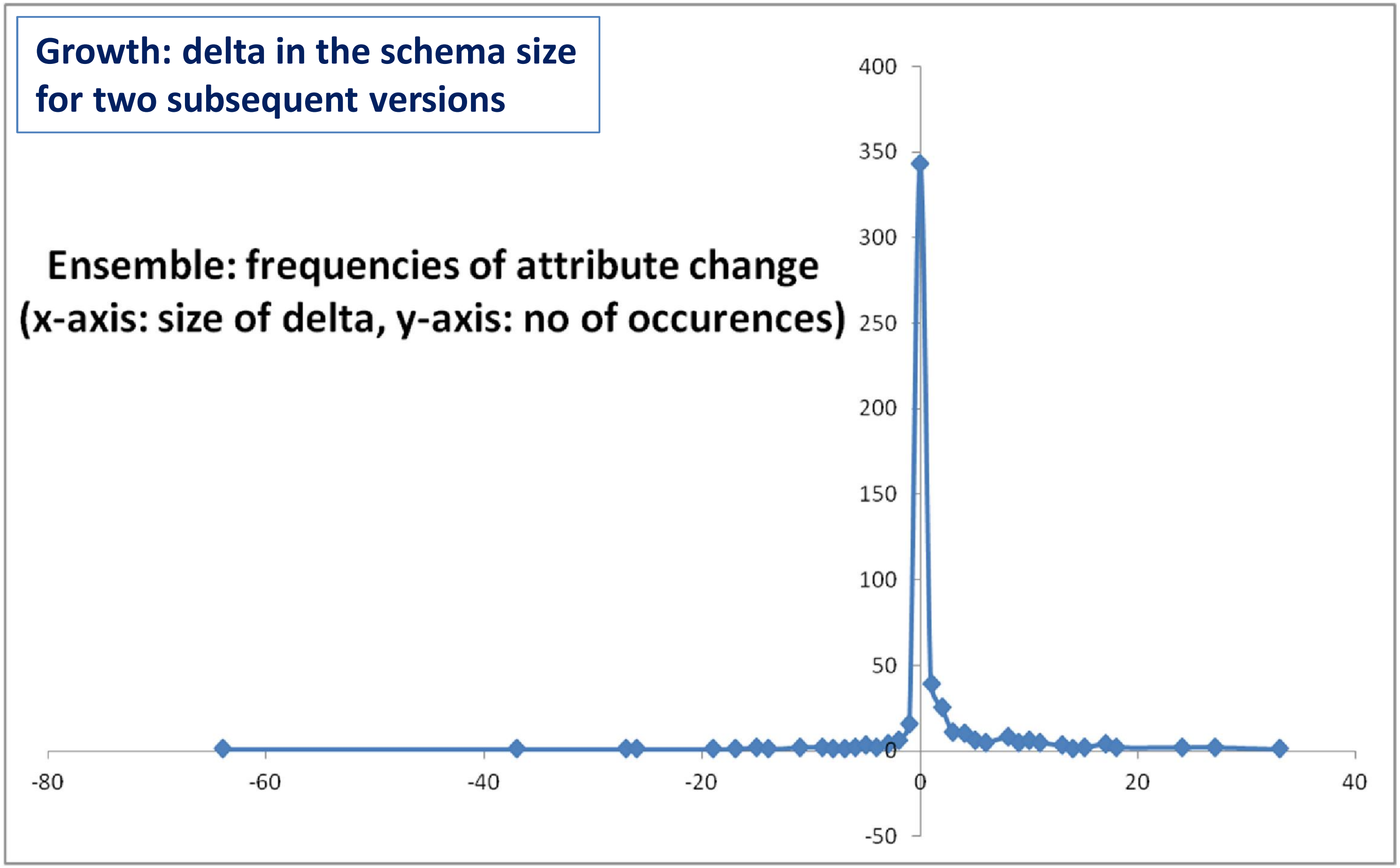


# Schema growth is small!

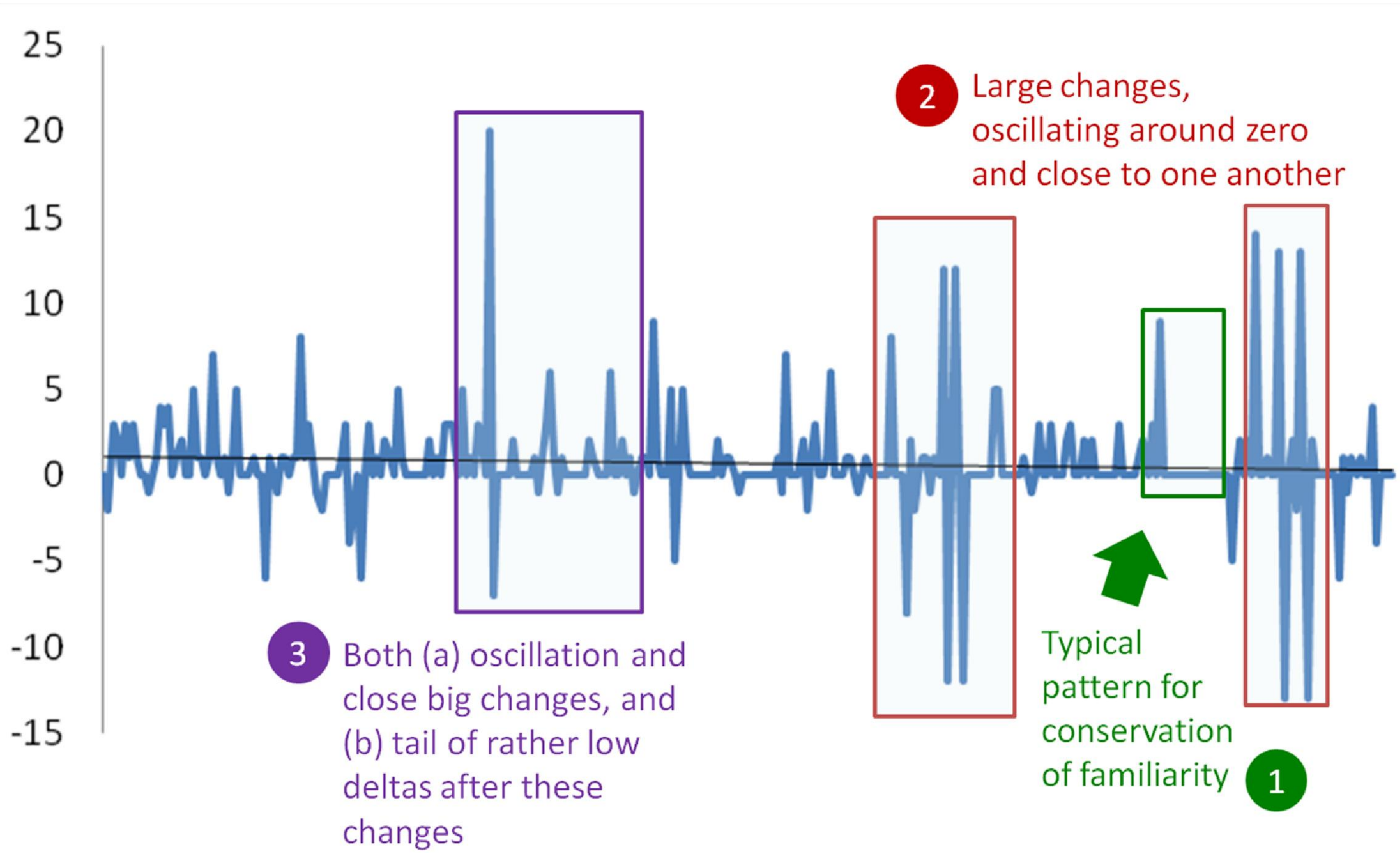
- **Growth is bounded in small values!**
- **Zipfian distribution of growth values around 0**
  - Predominantly: occurrences of zero growth; **almost all deltas are bounded between [-2..2] tables**
  - [0..2] tables slightly more popular => average value of growth slightly higher than 0
- No periods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

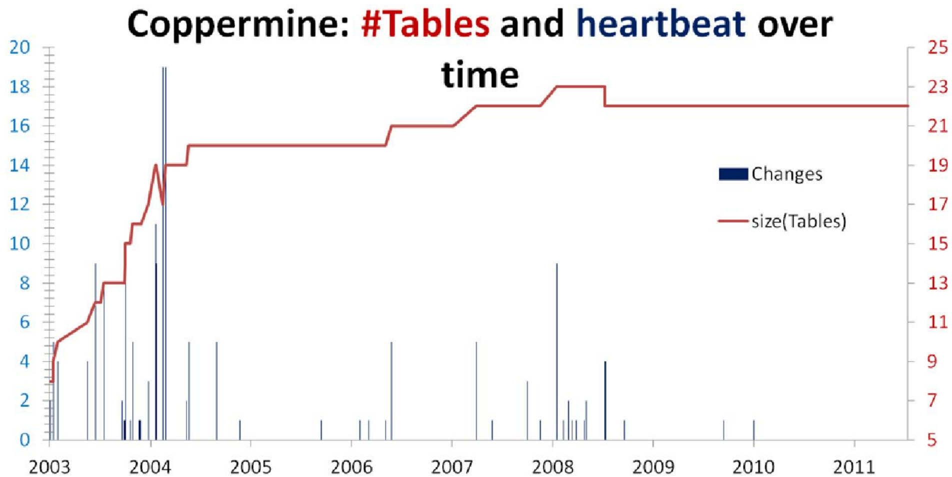
# Zipfian model in the distribution of growth frequencies

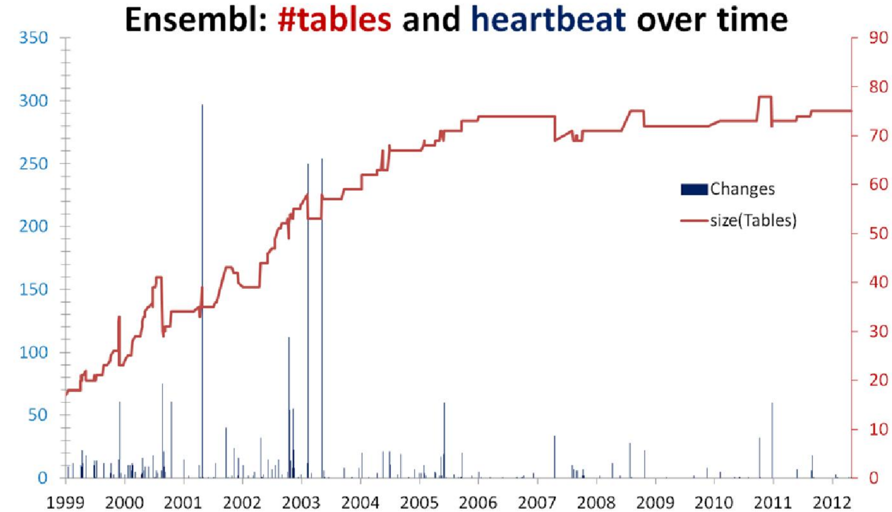


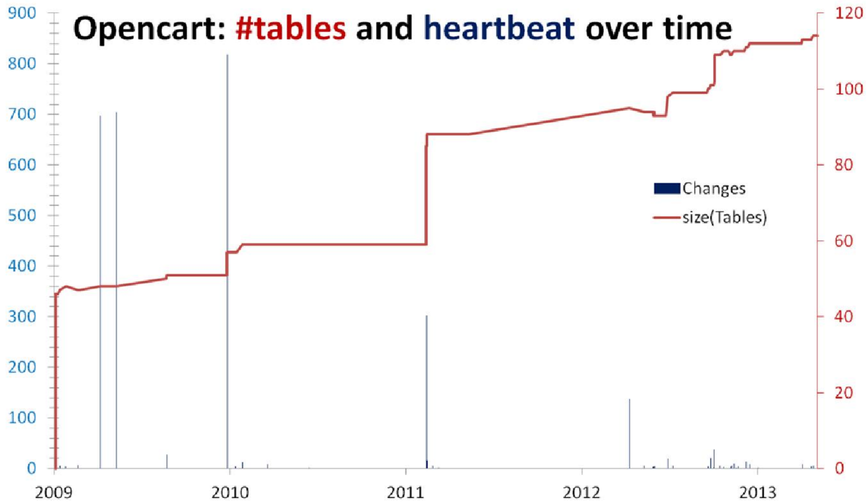
# What happens after large changes?

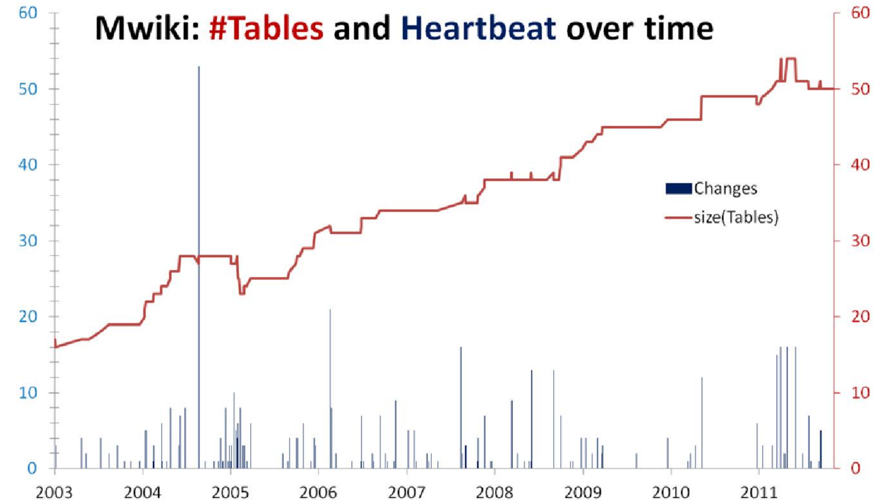


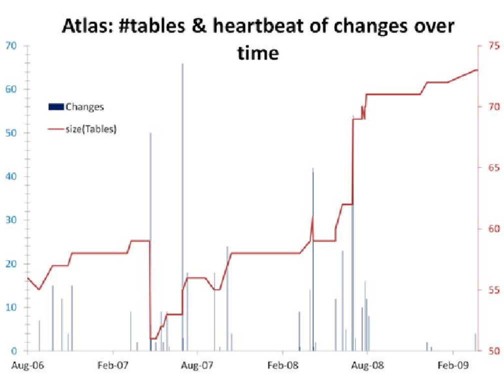

http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/



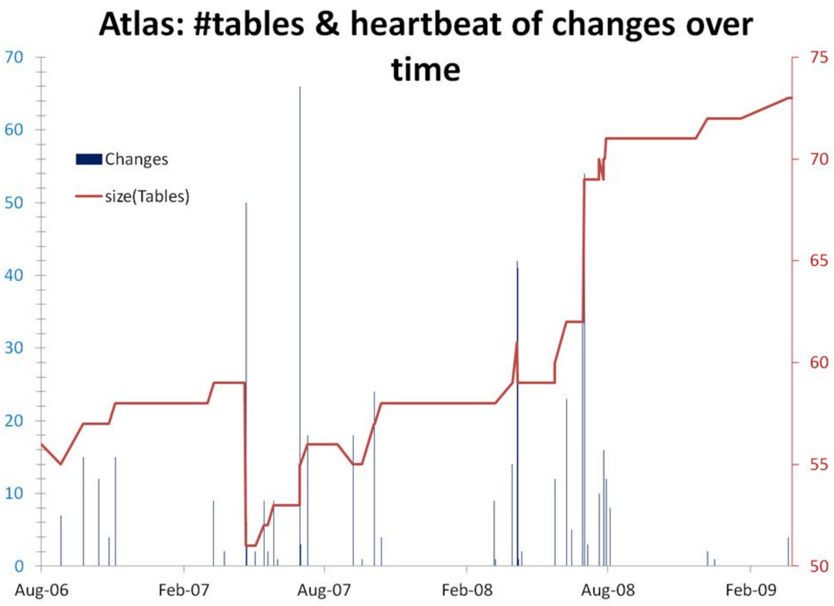

[With exceptions]

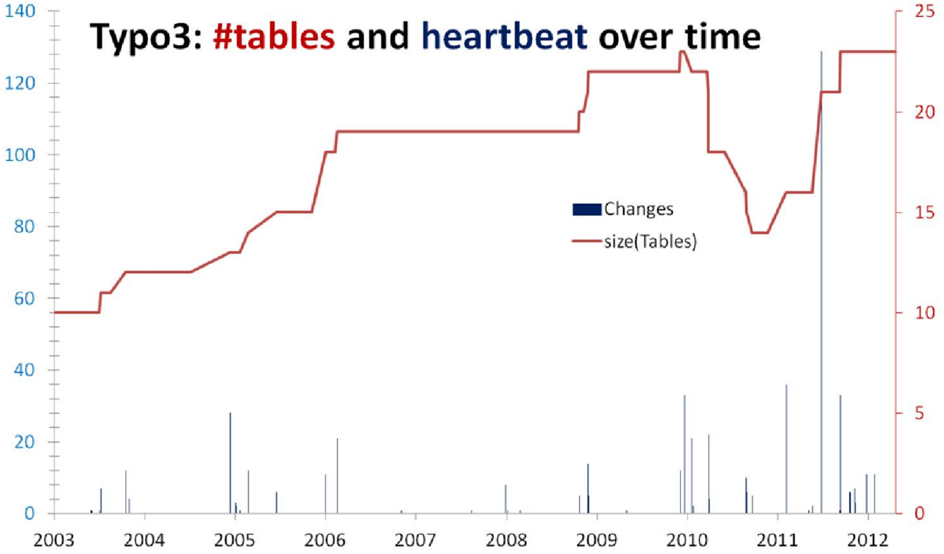
Density: **focused maintenance** effort

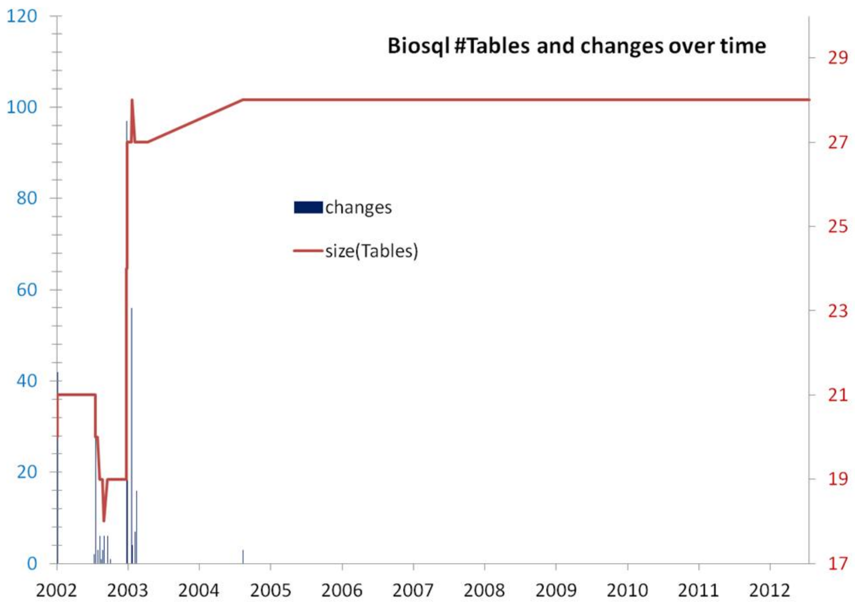
**Progressive cooling** : early –maintenance density >> later stages

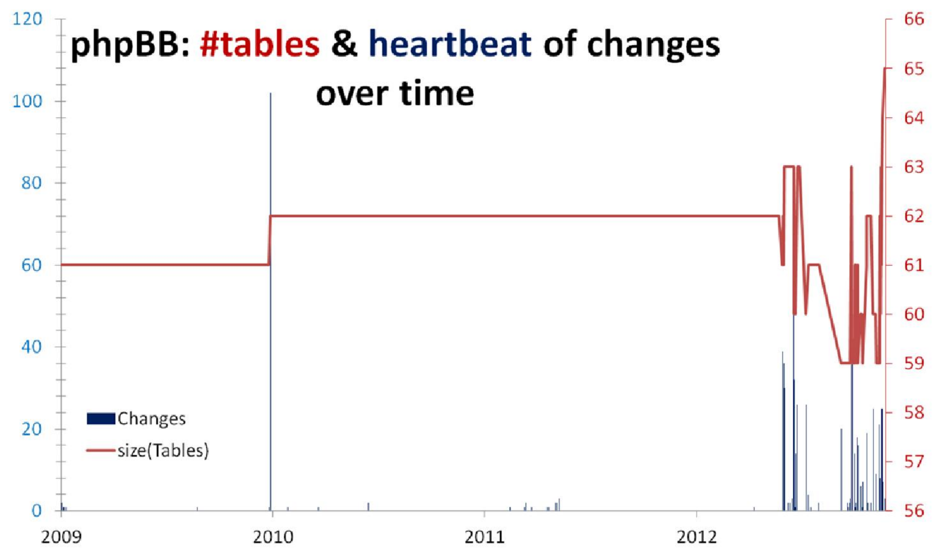
Several **spikes**, many **zero-change** periods/versions

# #tables & heartbeat of changes over time



# How do schemata evolve?

**Schema size (#tables – also: #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
- Large periods of stability

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

- Growth is small, smaller than in typical software
- Average growth is close (slightly higher) to zero

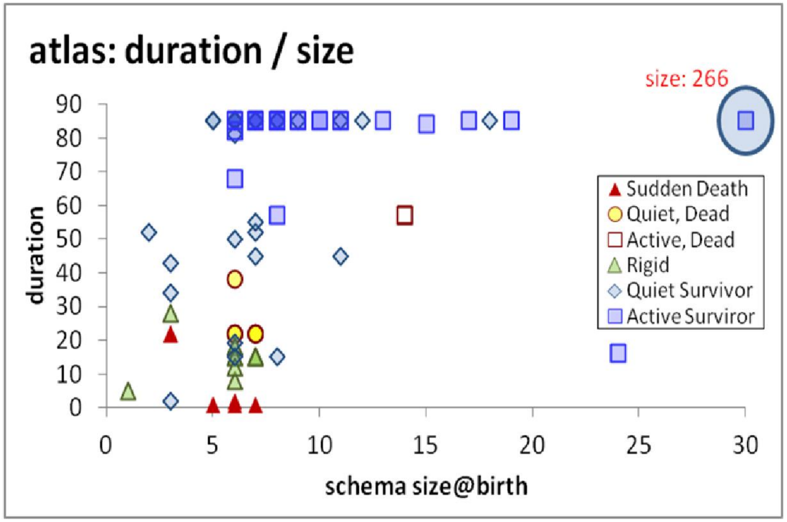
**Gravitation to rigidity:**

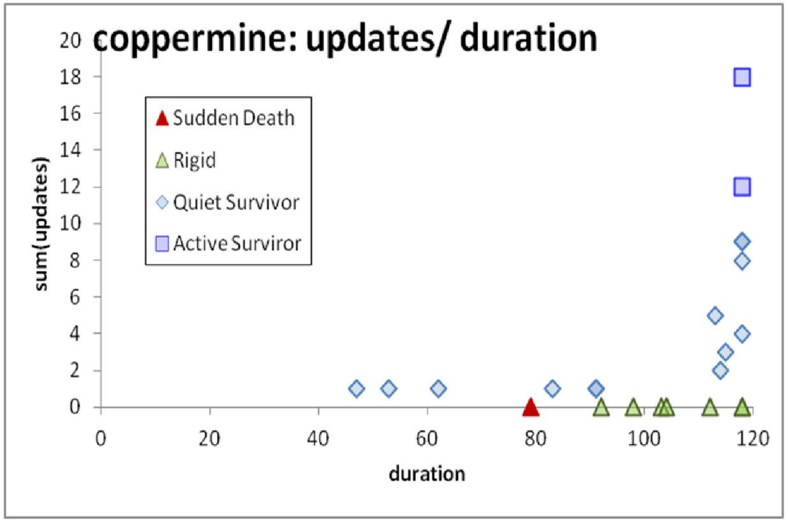
- Large periods of stability
- Change frequency drops with time

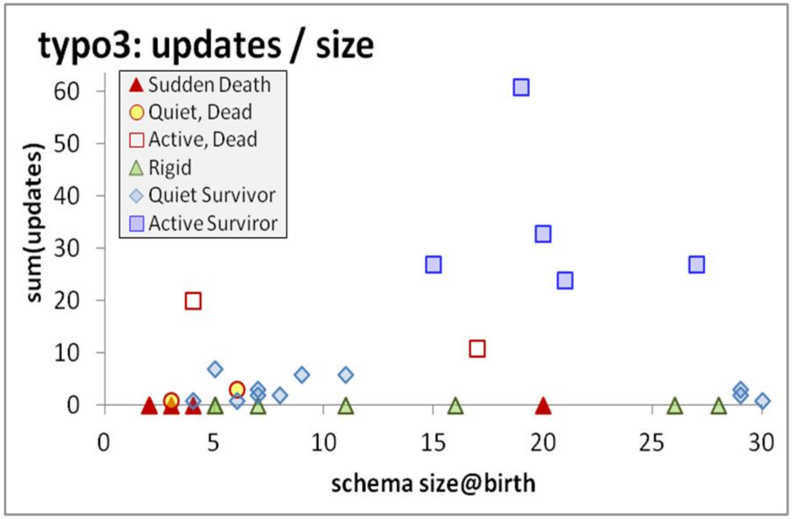
For details:
- CAiSE 2014
- Inf. Systems 2015

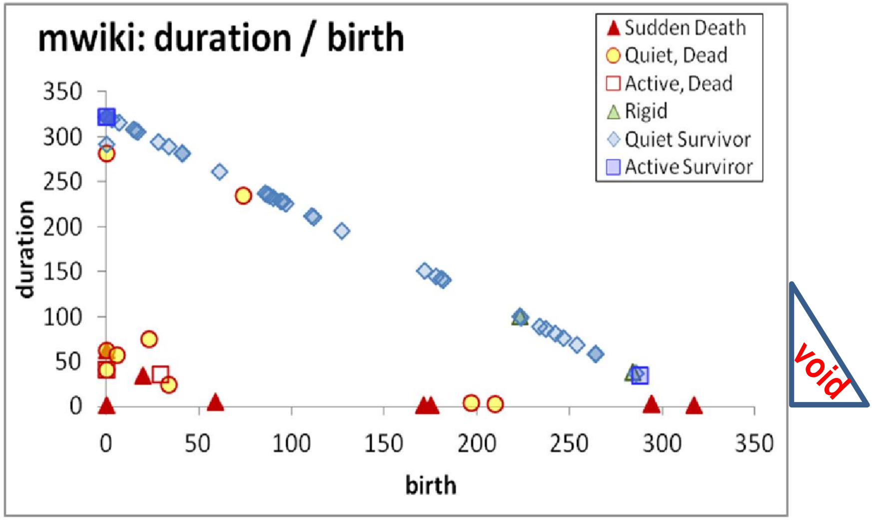
http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

# THE FOUR PATTERNS




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<br><br>Inverse Γ :<br>- Top-changers: mostly at long durations<br>- Long duration: all kinds of change | Comet:<br>- Many updates: typically at medium schema size @ birth<br>- Large schema at birth: medium amount of updates |
| *Rigidity* | Inverse Γ :<br>- small duration → small change<br>- 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<br><br>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 | |

# 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 …

| | <5 | 5-10 | >10 |
|---|---|---|---|
| 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% |

Schema size @ birth / duration

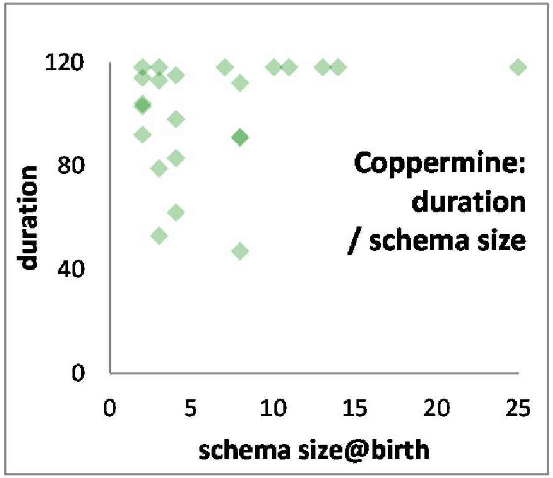
If you 're wide, you survive
a.k.a (only the thin die young, ~~all~~ the wide ones seem to live forever)

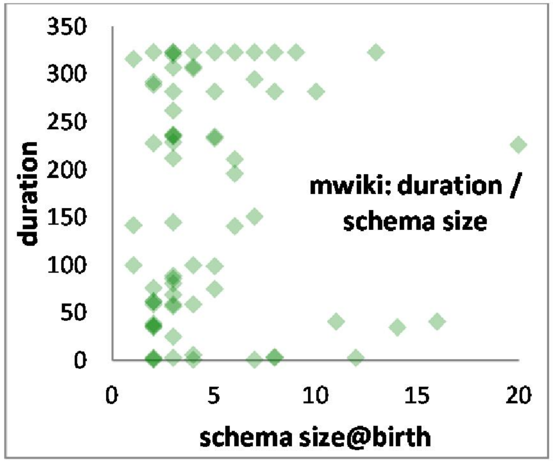
# THE GAMMA PATTERN

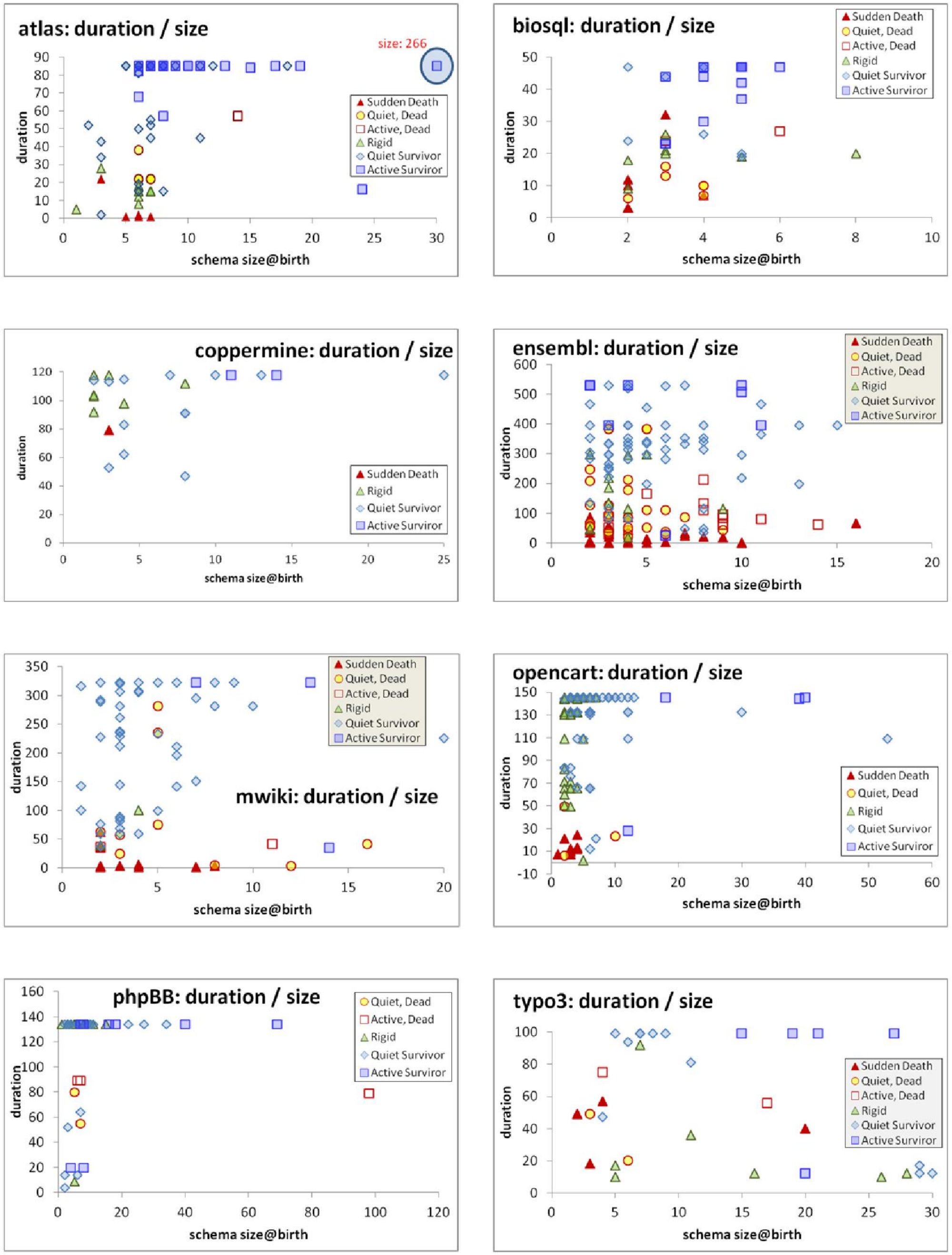
# The Gamma Γ Pattern: "if you 're wide, you survive"

- 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

# 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

| | # 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 |

**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.**

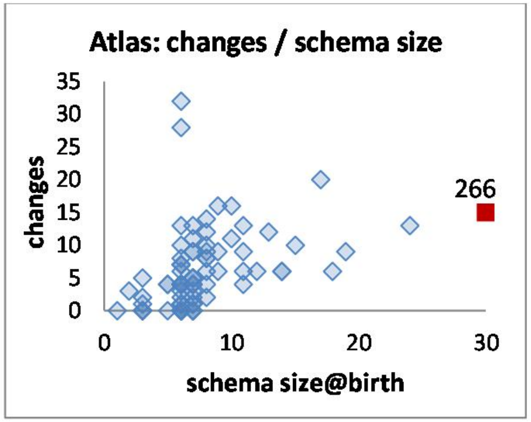
Schema size and updates

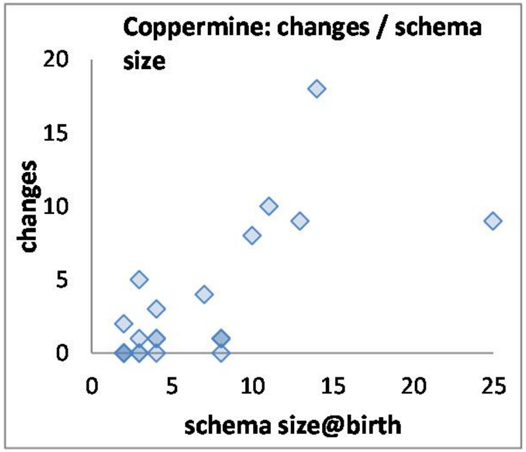
# THE COMET PATTERN

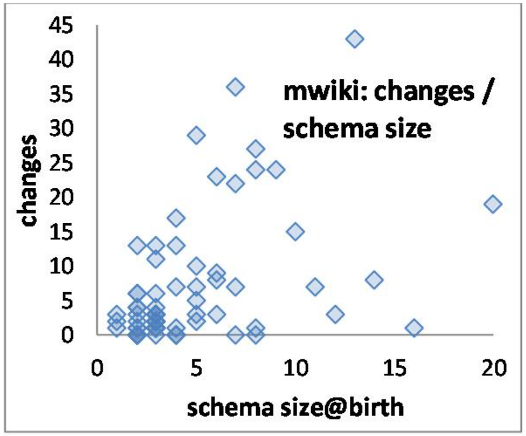
# The Comet Pattern

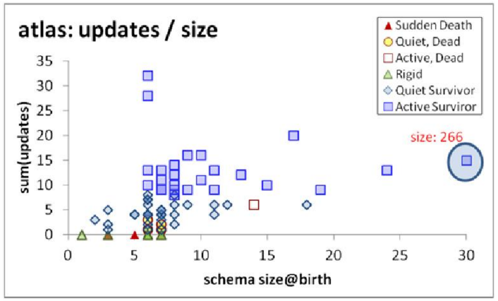
"Comet " for change over schema size with:

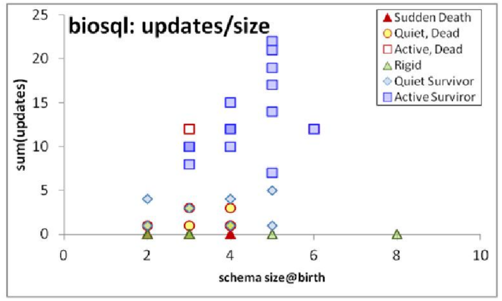
- a large, dense, nucleus cluster close to the beginning of the axes, denoting small size and small amount of change,
- medium schema size tables typically demonstrating medium to large change
  - The tables with the largest amount of change are typically tables whose schema is on average one standard 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.

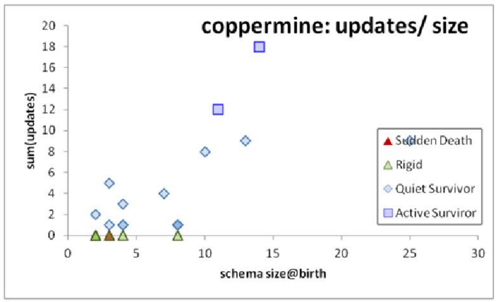


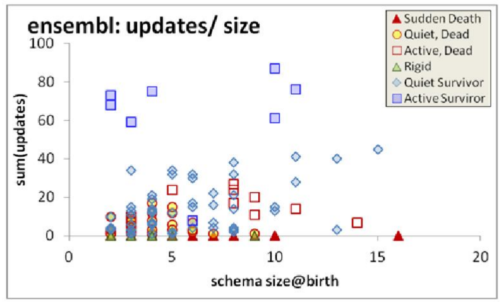



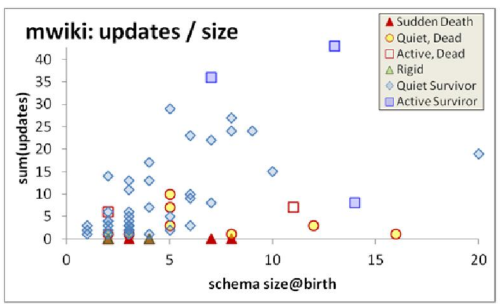



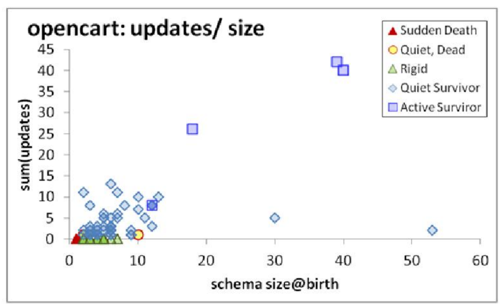


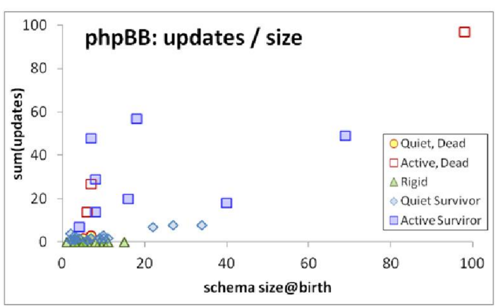



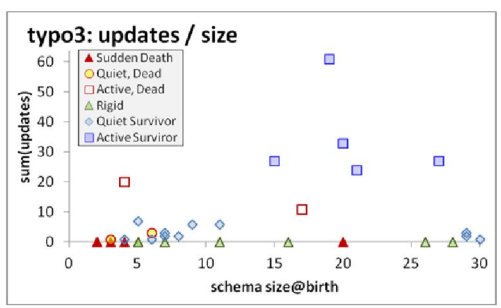



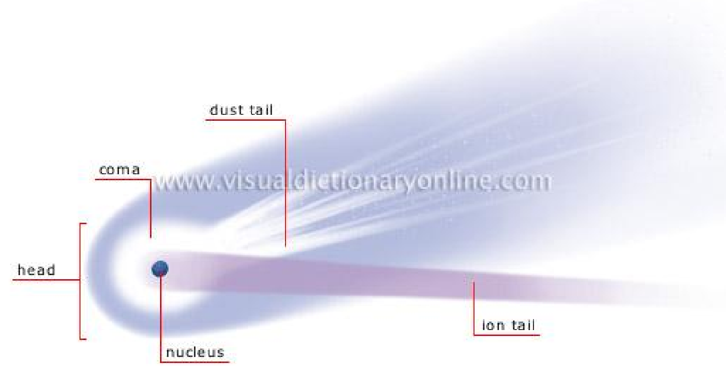




http://visual.merriam-webster.com/astronomy/celestial-bodies/comet.php





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

# 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.

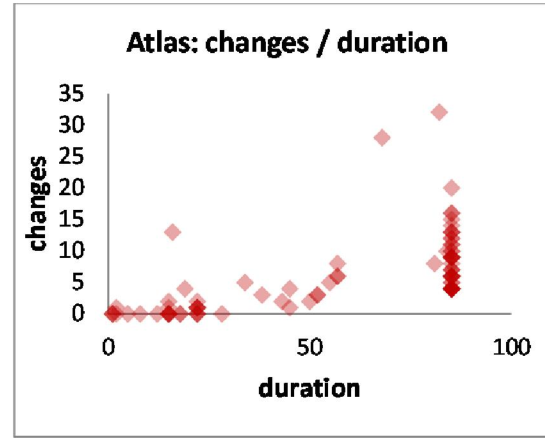
Typically, the avg. number of updates of the top wide tables is close to the 38% of the domain of values for the sum of updates (i.e., the middle of the y-axis of the comet figure, measuring the sum of updates for each table).

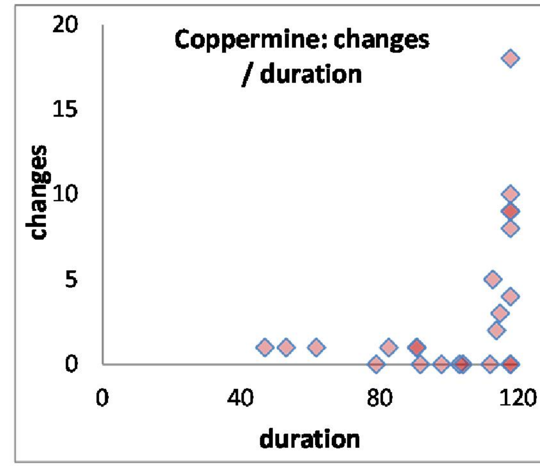
This is mainly due to the (very) large standard deviation (twice the mean), rather than the -- typically low -- mean value (due to the large part of the population living quiet lives).

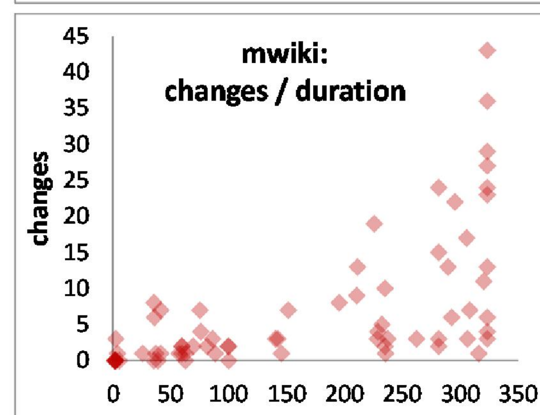
# INVERSE GAMMA

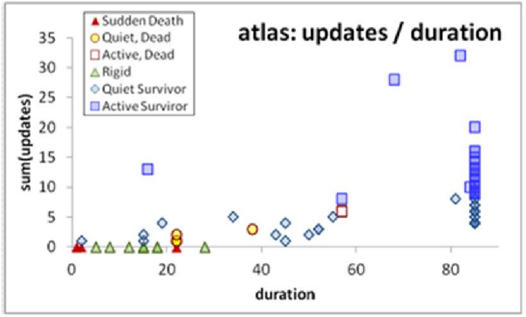
# The inverse Gamma pattern

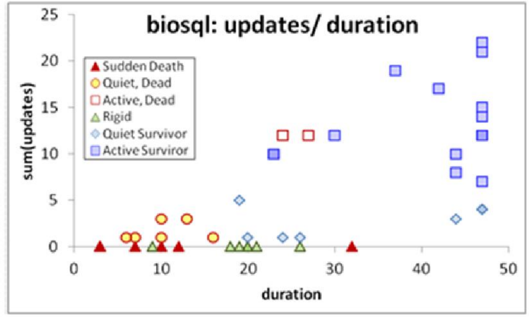
- The correlation of change and duration is as follows:
  - small durations come necessarily with small change,
  - large durations come with all kinds of change activity and
  - medium sized durations come mostly with small change activity (Inverse Gamma).

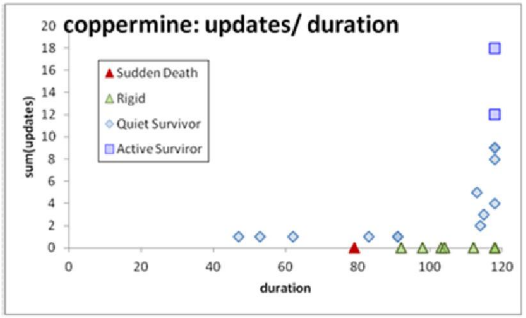


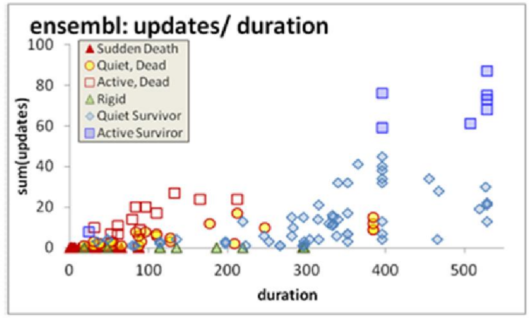



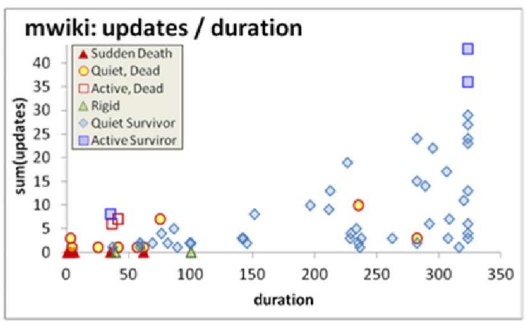



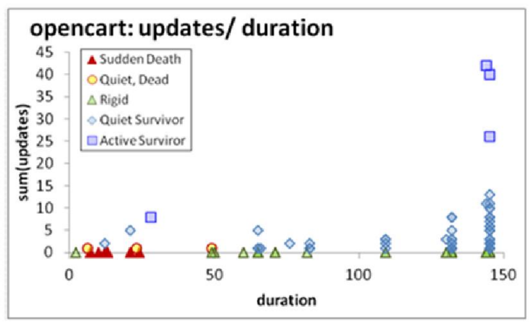

atlas: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

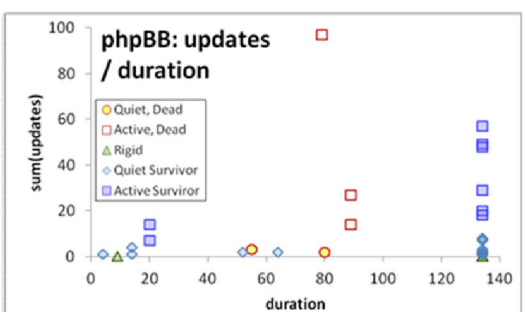

biosql: updates/ duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

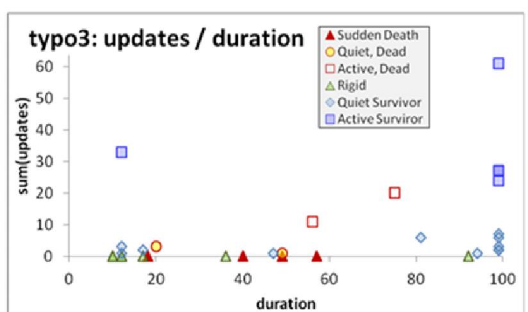

coppermine: updates/ duration
Sudden Death
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration


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

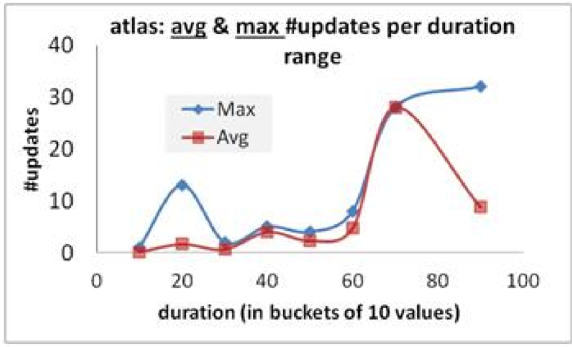

mwiki: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

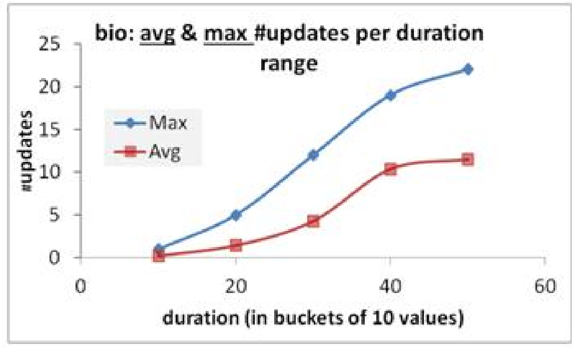

opencart: updates/ duration
Sudden Death
Quiet, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

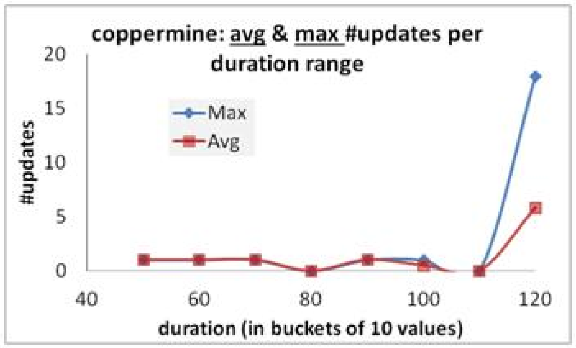

phpBB: updates
/ duration
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

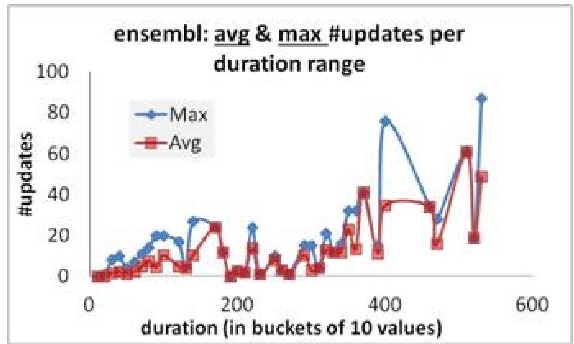

typo3: updates / duration
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
sum(updates)
duration

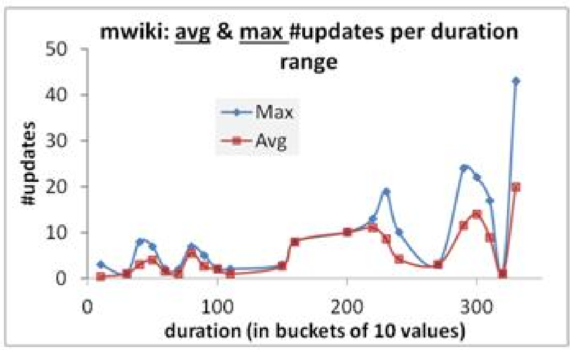
# Skyline & Avg for Inverse Gamma

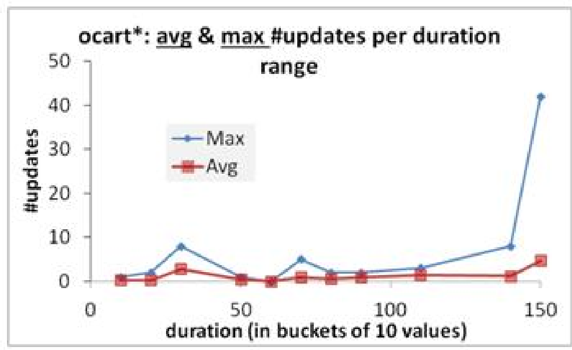


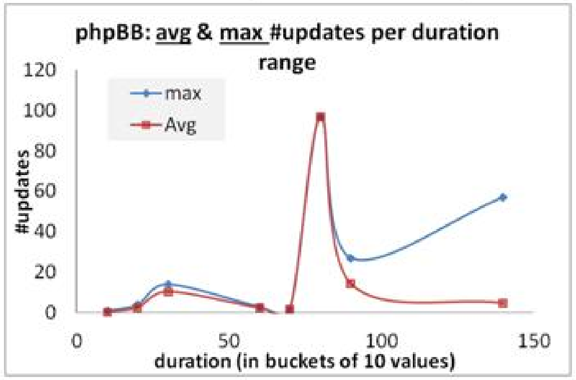



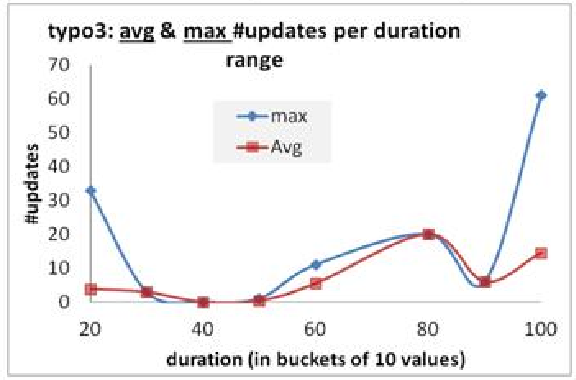



# THE EMPTY TRIANGLE PATTERN

# 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%** |

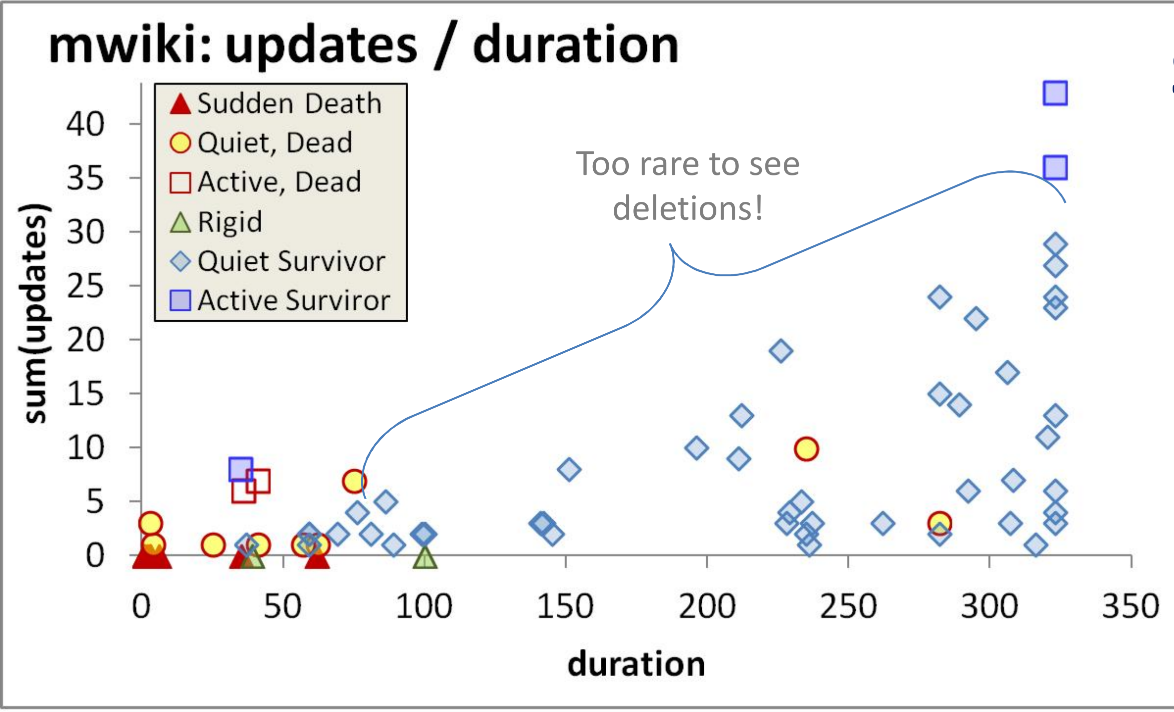
Non-survivors

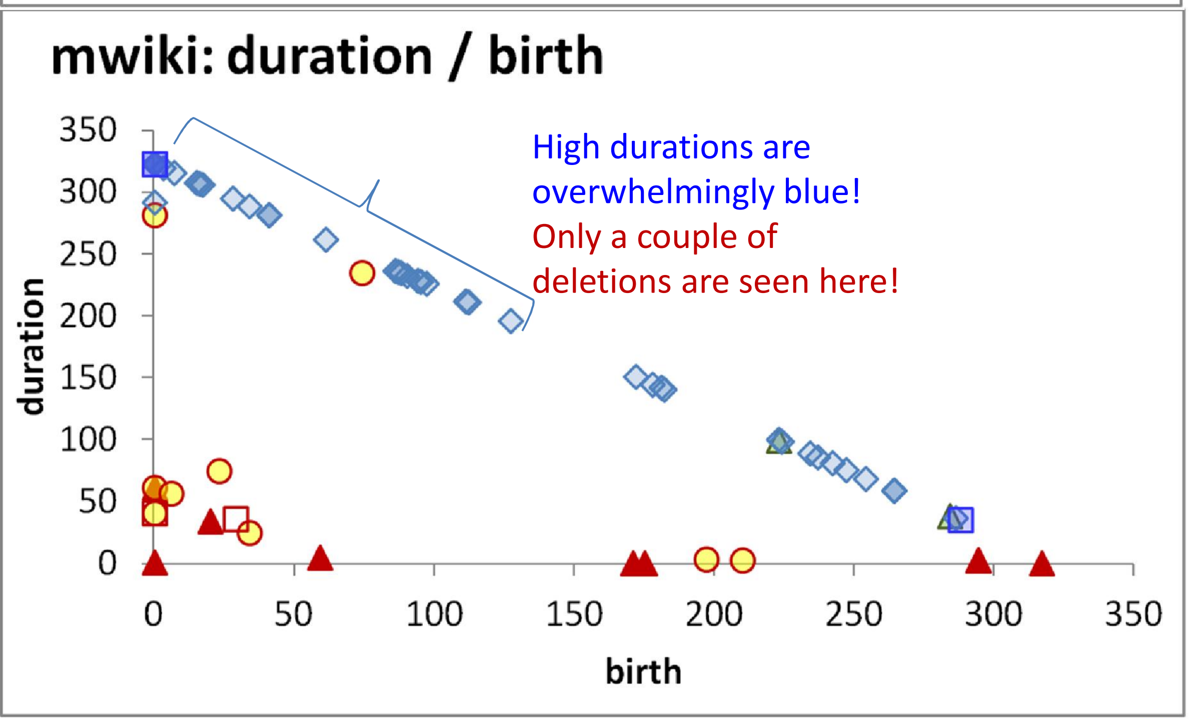
- Sudden deaths mostly
- Quiet come ~ close
- Too few active

Survivors

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

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



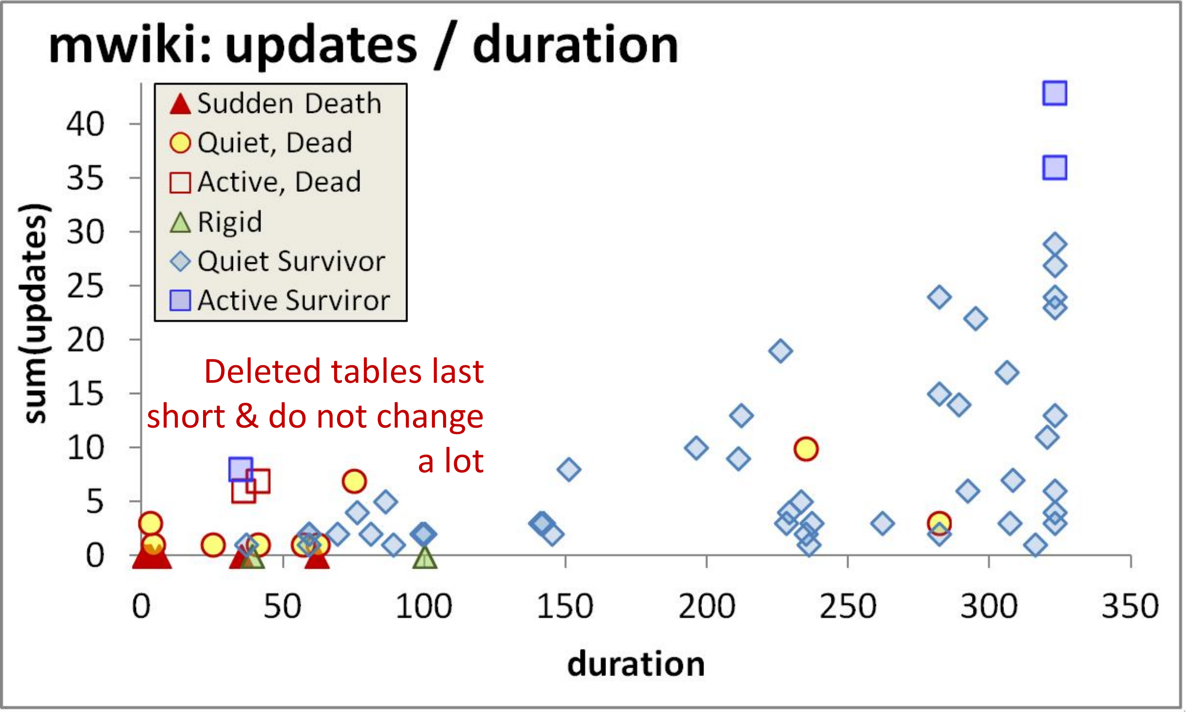



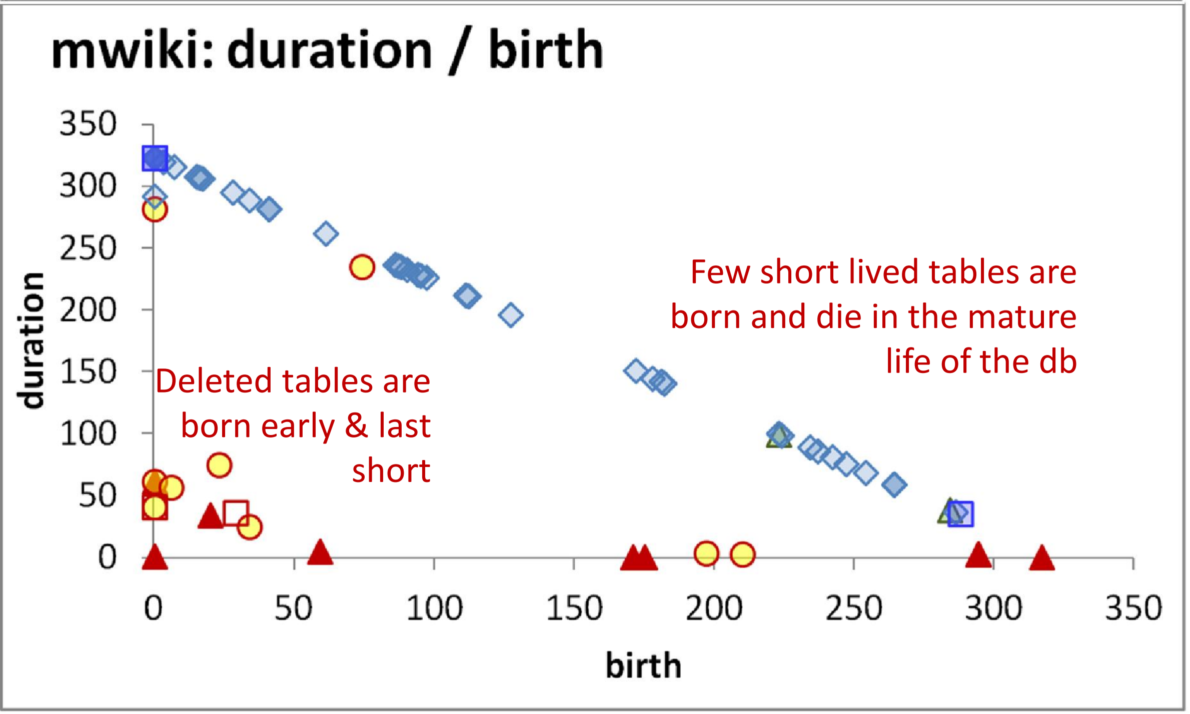

# Survive long enough & you 're probably safe

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 )!

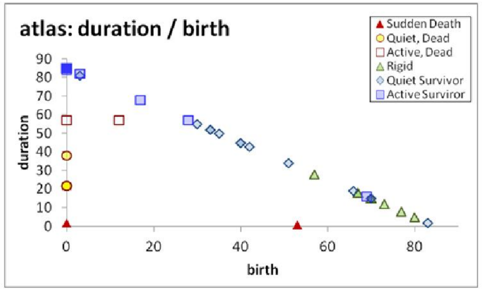
# Die young and suddenly

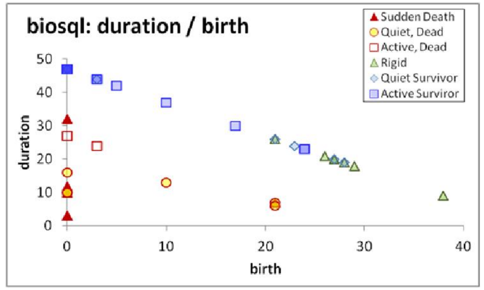


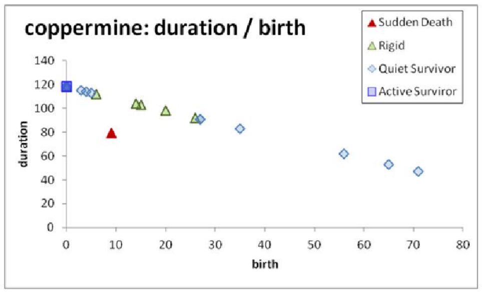



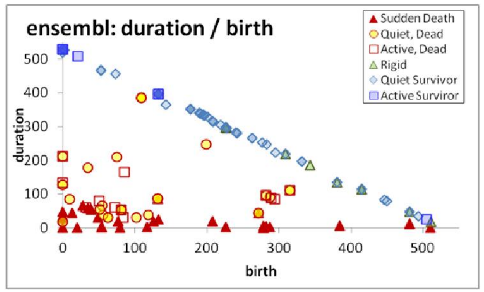

**[Early life of the db]** There is a very large concentration of the deleted tables in a small range of newly born, quickly removed, with few or no updates, resulting in very low numbers of removed tables with medium or long durations.

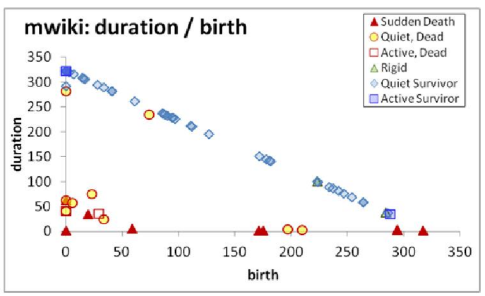
[**Mature db**] After the early stages of the databases, we see the birth of tables who eventually get deleted, but they mostly come with very small durations and sudden deaths.

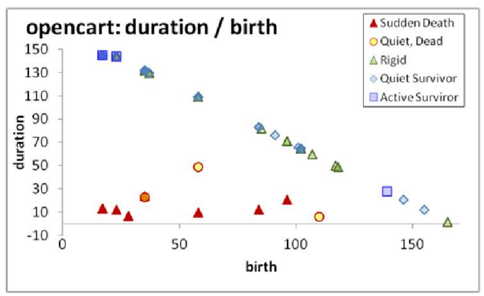

atlas: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

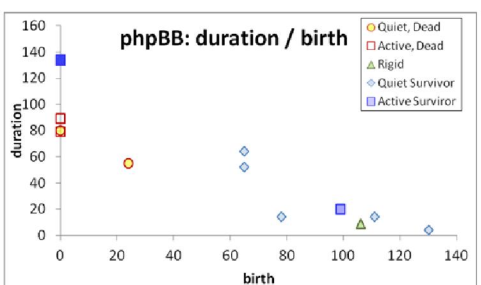

biosql: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth

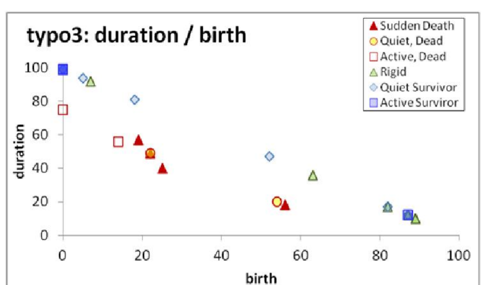

coppermine: duration / birth
Sudden Death
Rigid
Quiet Survivor
Active Surviror
duration
birth

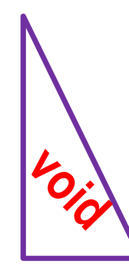

ensembl: duration / birth
Sudden Death
Quiet, Dead
Active, Dead
Rigid
Quiet Survivor
Active Surviror
duration
birth


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

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

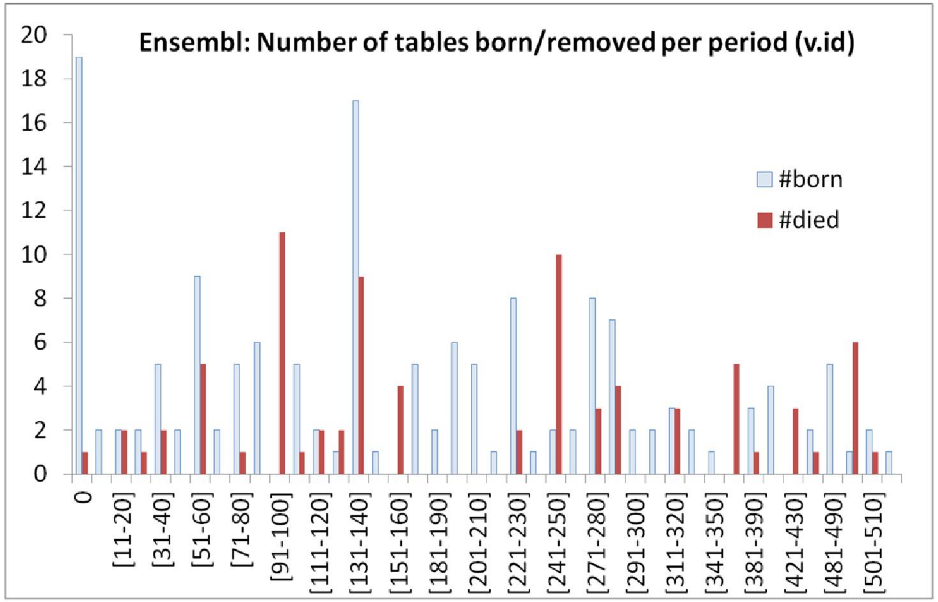
| | atlas | biosql | coppermine | ensembl | mwiki | ocart* | phpBB | typo3 |
|---|---|---|---|---|---|---|---|---|
| Tables | 88 | 45 | 23 | 155 | 71 | 128 | 70 | 32 |
| Active | 27 | 16 | 2 | 23 | 5 | 4 | 11 | 7 |
| active tables(%) | 31% | 36% | 9% | 15% | 7% | 3% | 16% | 22% |
| *As percentages over 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% |

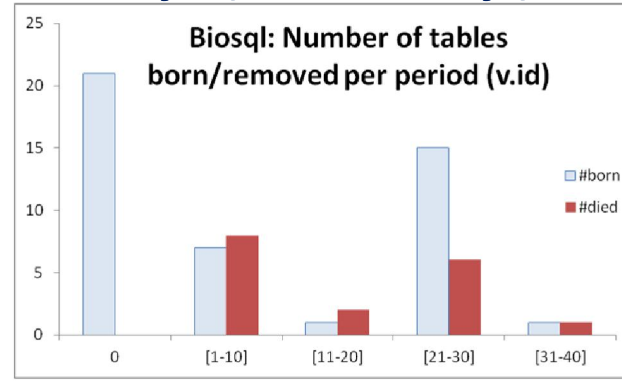
- In all data sets, active tables are born early with percentages that exceed 75%
- With the exceptions of two data sets, they survive with percentage higher than 70%.
- The probability of having a long duration is higher than 50% in 6 out of 8 data sets.
- Interestingly, **the two last lines are exactly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

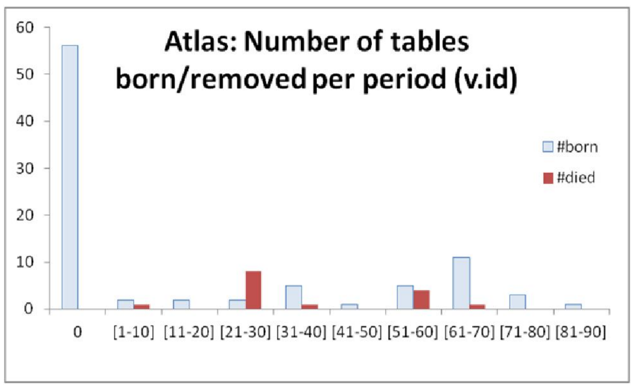
# Dead are: quiet, early born, short lived, and quite often all three of them

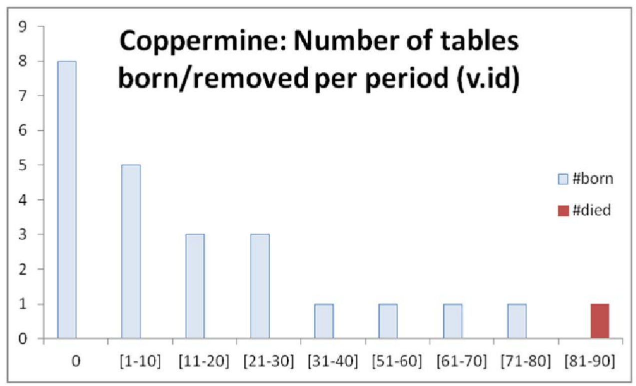
| | atlas | biosql | coppermine | ensembl | mwiki | ocart* | phpBB | typo3 |
|---|---|---|---|---|---|---|---|---|
| tables | 88 | 45 | 23 | 155 | 71 | 128 | 70 | 32 |
| dead | 15 | 17 | 1 | 80 | 21 | 14 | 5 | 9 |
| dead tables(%) | 17% | 38% | 4% | 52% | 30% | 11% | 7% | 28% |
| *As percentages over # 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% |

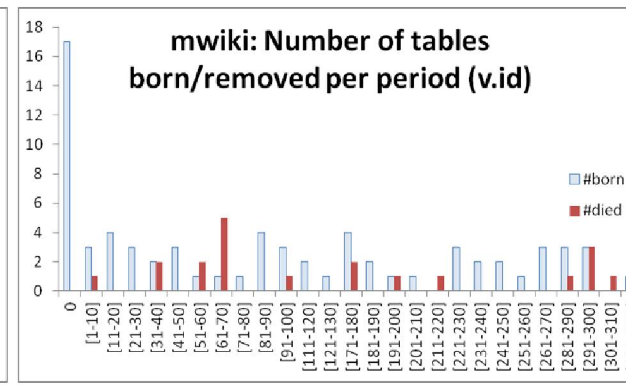
# Most births &deaths occur early (usually)

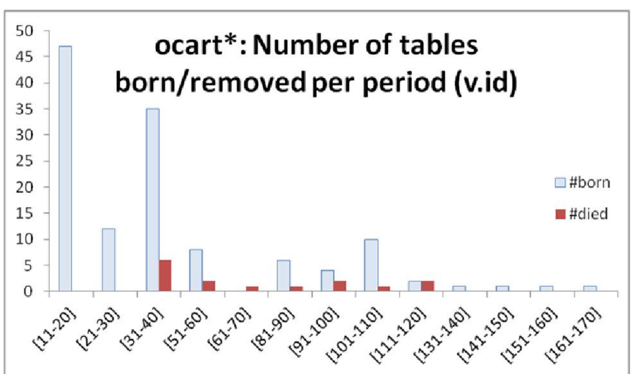


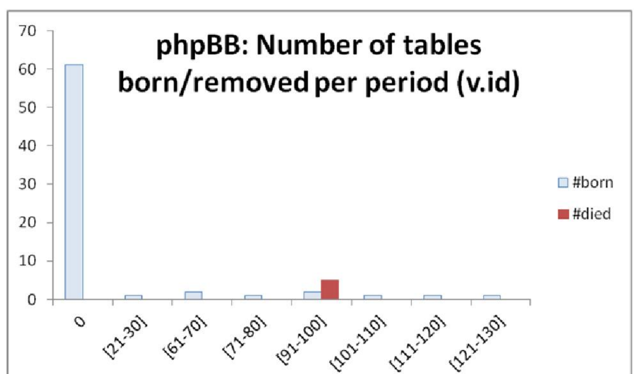



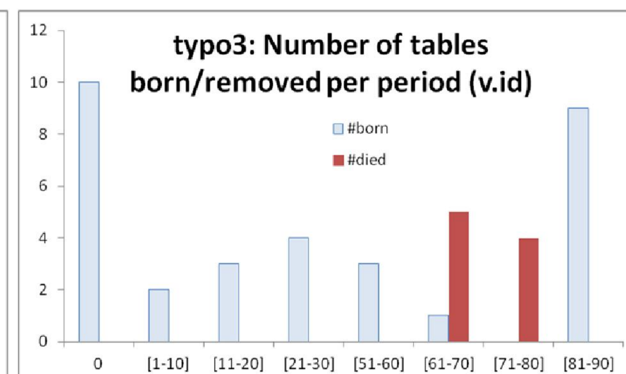



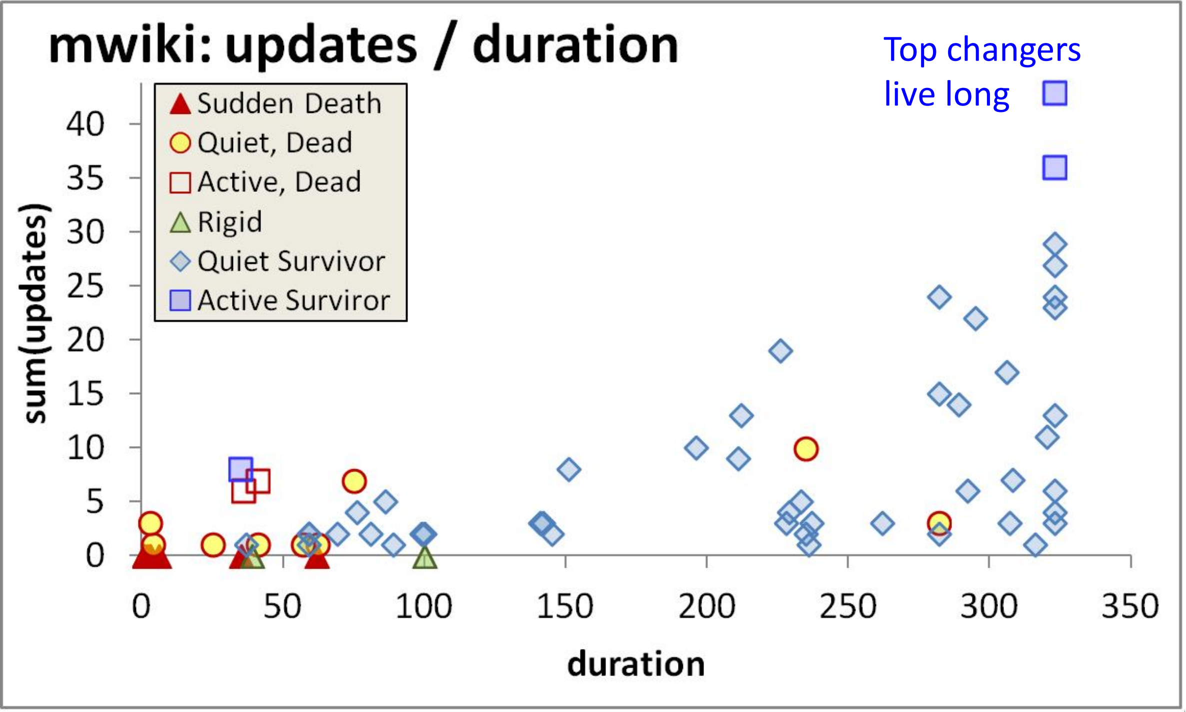



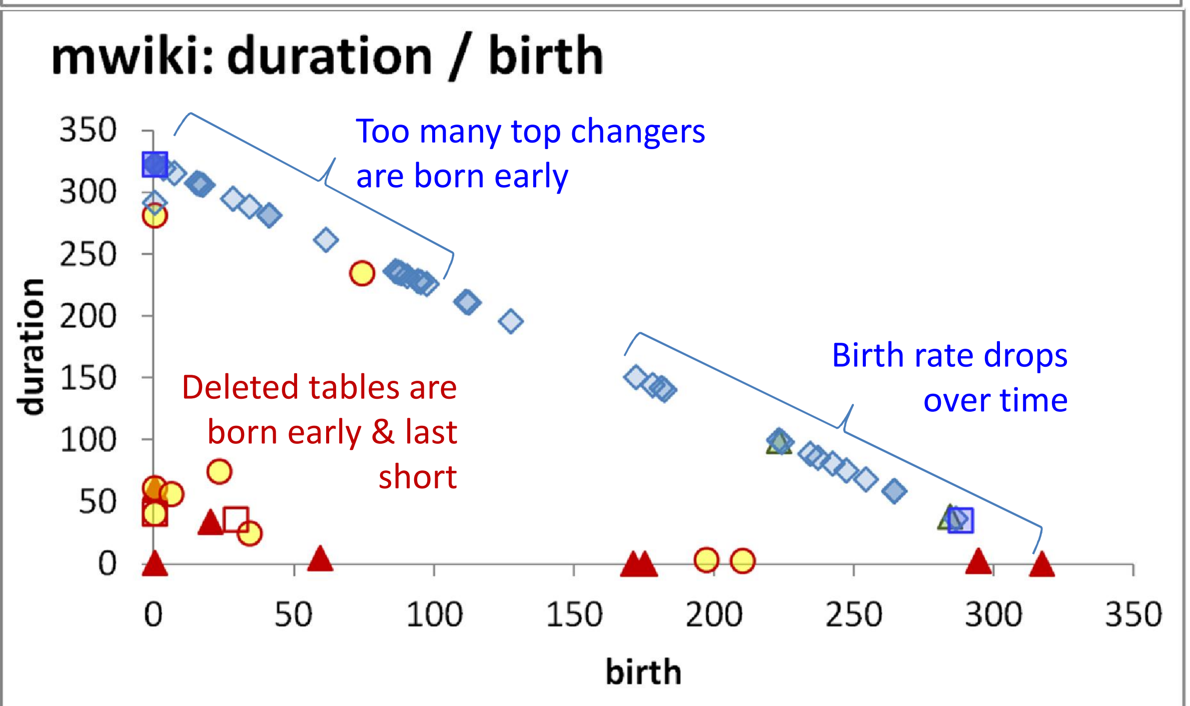



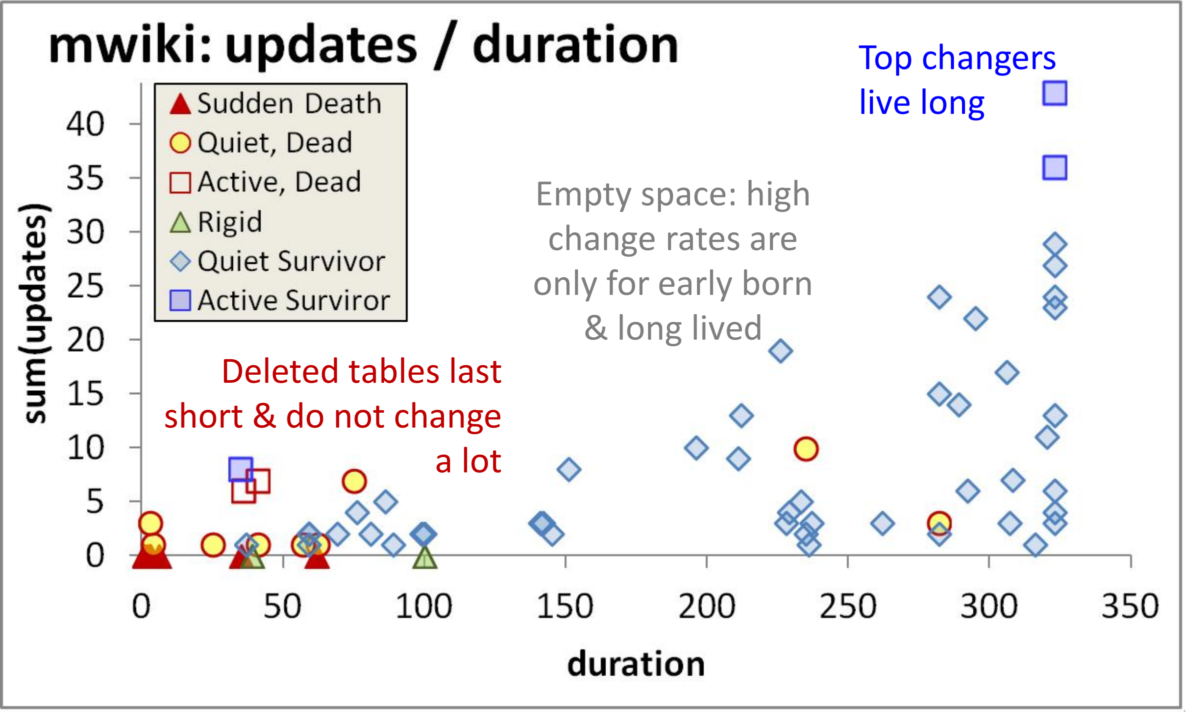



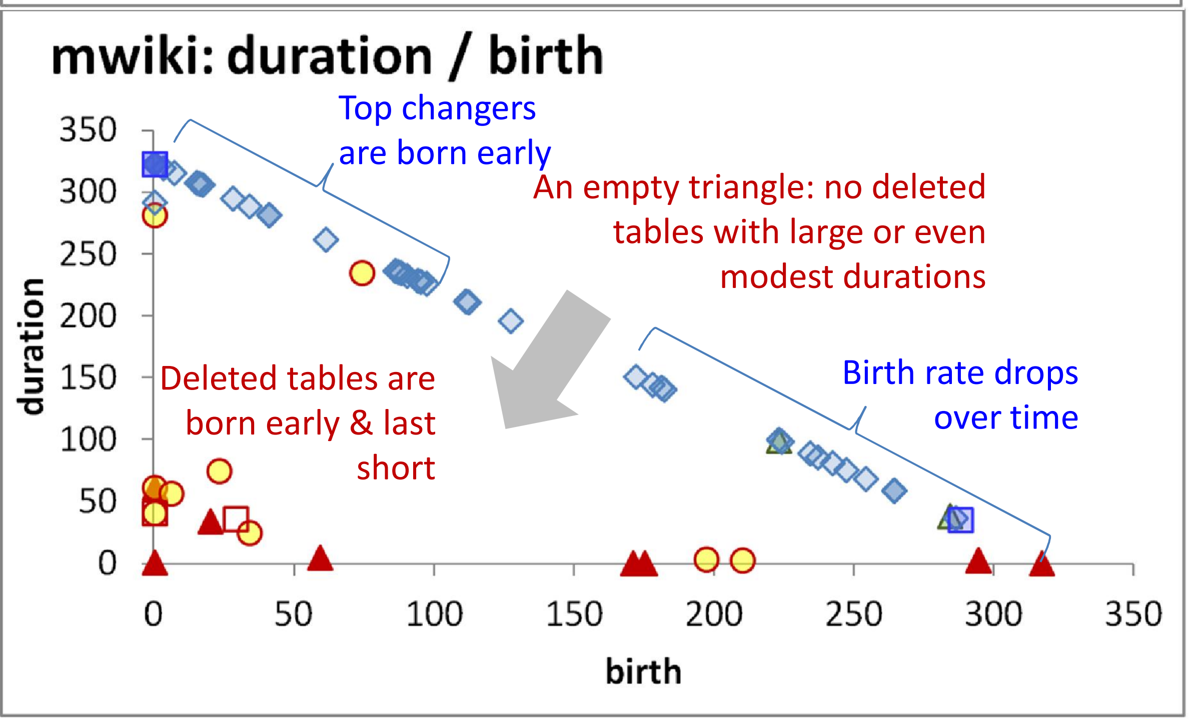



# Longevity and update activity correlate !!

- 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 $\Gamma$)
- See also upper right blue part of diagonal: too many of them are born early and survive => live long!




# All in one

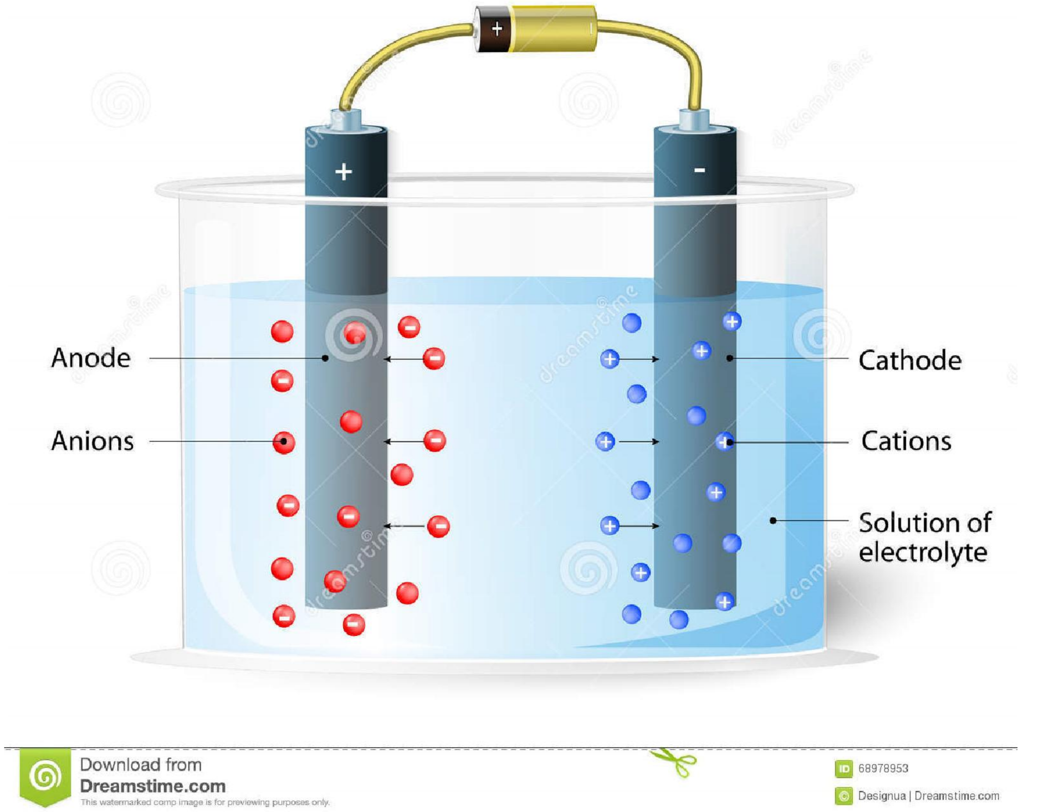
- Early stages of the database life are more "active" in terms of births, deaths and updates, and have higher chances of producing deleted tables.
- After the first major restructuring, the database continues to grow; however, we see much less removals, and maintenance activity becomes more concentrated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

**… How do survivor tables differ from the dead ones (esp., wrt activity & duration)?**

http://www.cs.uoi.gr/~pvassil/publications/2017_CAiSE_Electrolysis

Panos Vassiliadis, Apostolos Zarras. 29th International Conference on Advanced Information Systems Engineering , (**CAiSE 2017**), 12-16 June 2017, Essen, Germany.

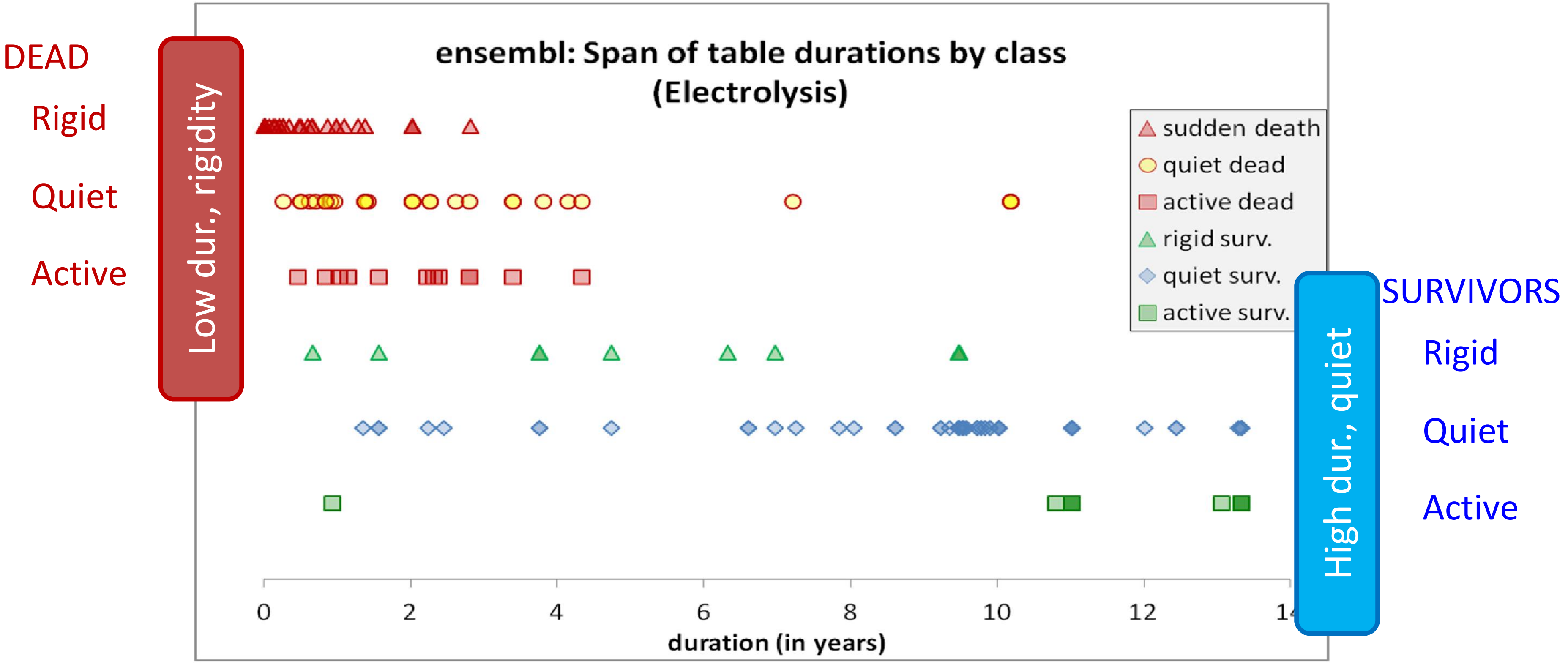
# SURVIVAL IN SCHEMA EVOLUTION: PUTTING THE LIVES OF SURVIVOR AND DEAD TABLES IN COUNTERPOINT



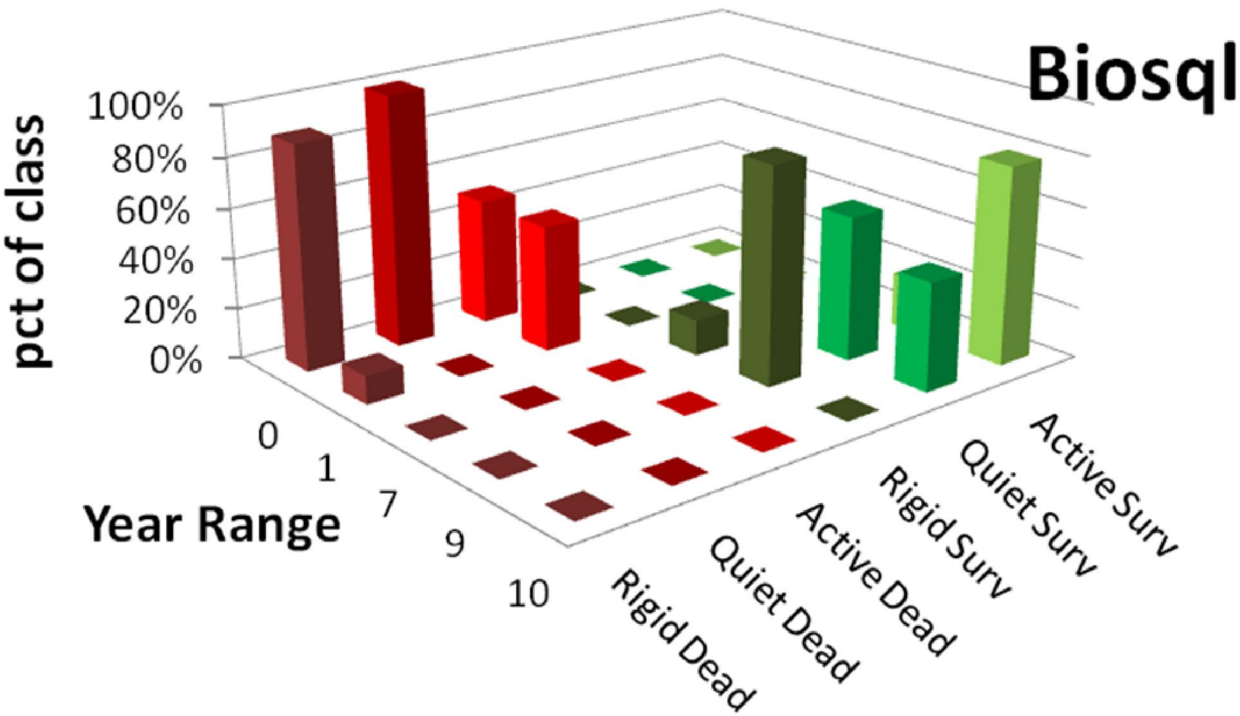


# ELECTROLYSIS PATTERN FOR TABLE ACTIVITIES

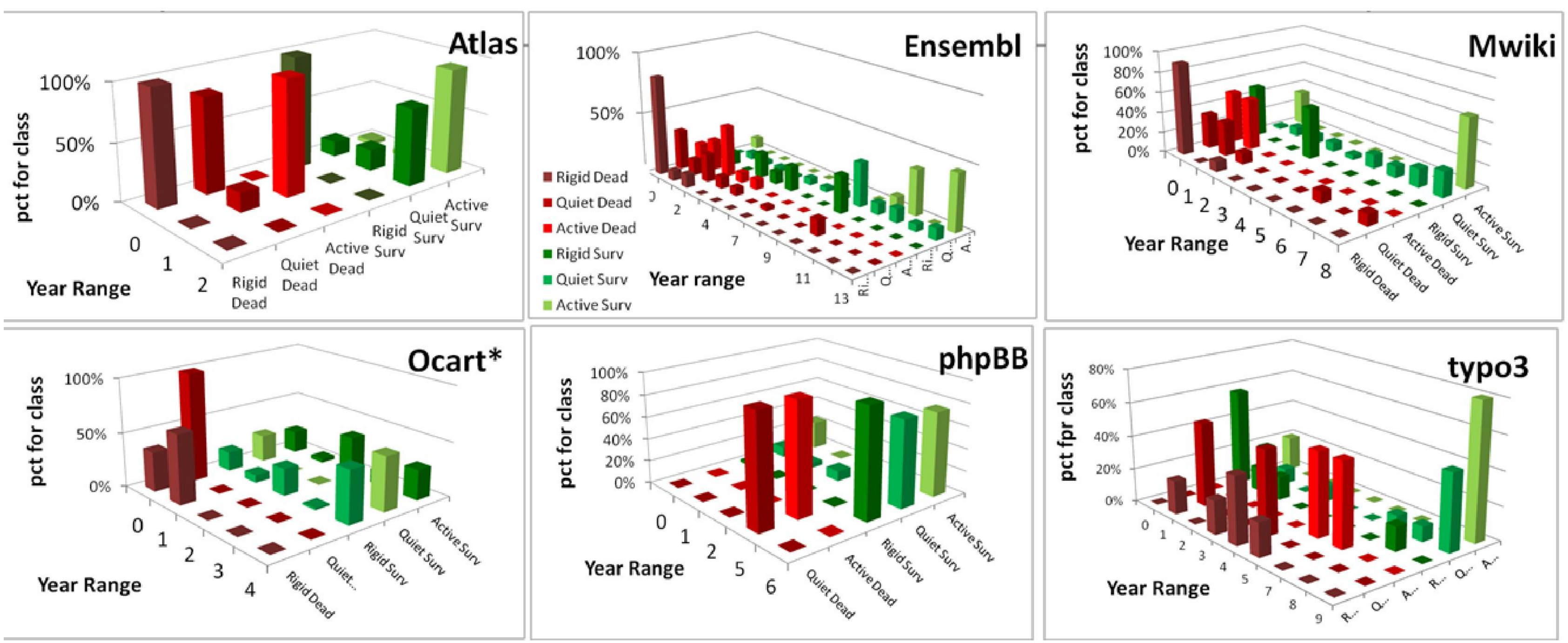
# Duration is related to the Life & Death Class of the tables!



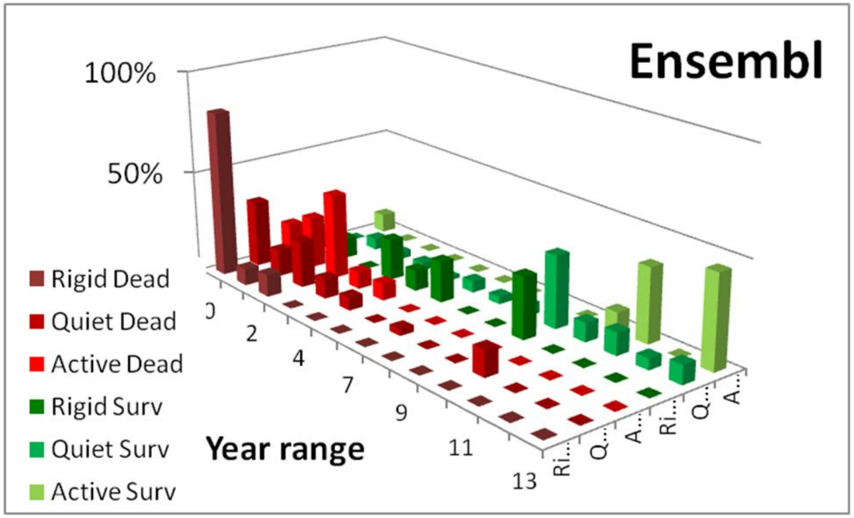

(a) **Survival**: DEAD vs SURVIVORS
(b) **Activity**: Rigid (no change) vs Active (change rate > 10%) vs Quiet (all in between)
(c) **Life And Death (LAD)** class: Survival x Activity




Attn: all pct's are *per class*



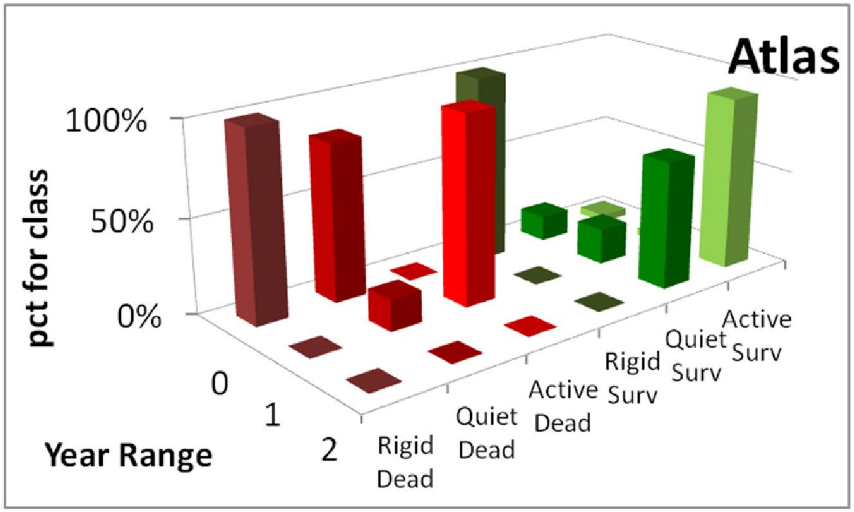
# The electrolysis pattern

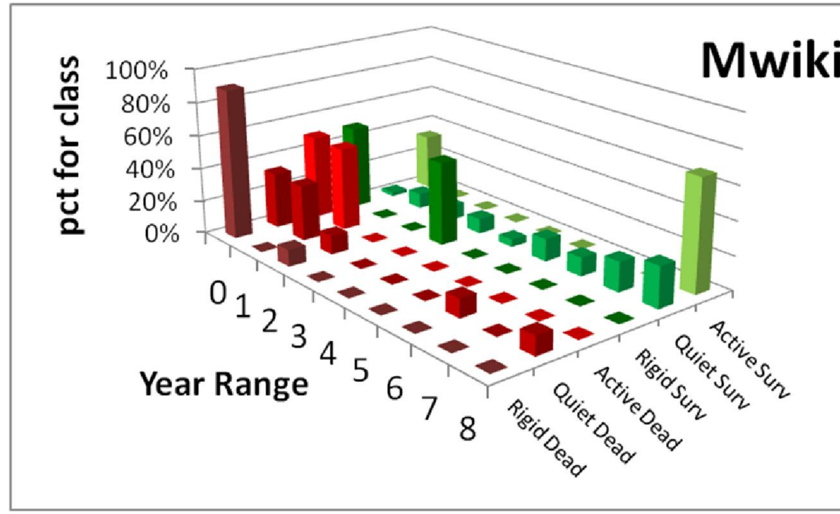



- **Dead tables demonstrate much shorter lifetimes** than survivor ones,
- can be located at short or medium durations, and **practically never at high durations**.
- With few exceptions, the **less active dead tables are, the higher the chance to reach shorter durations**.

- **Survivors** expose the inverse behavior, i.e., **mostly located at medium or high durations**.
- The **more active survivors are, the stronger they are attracted towards high durations**, with a significant such inclination for the few active ones that cluster in very high durations.

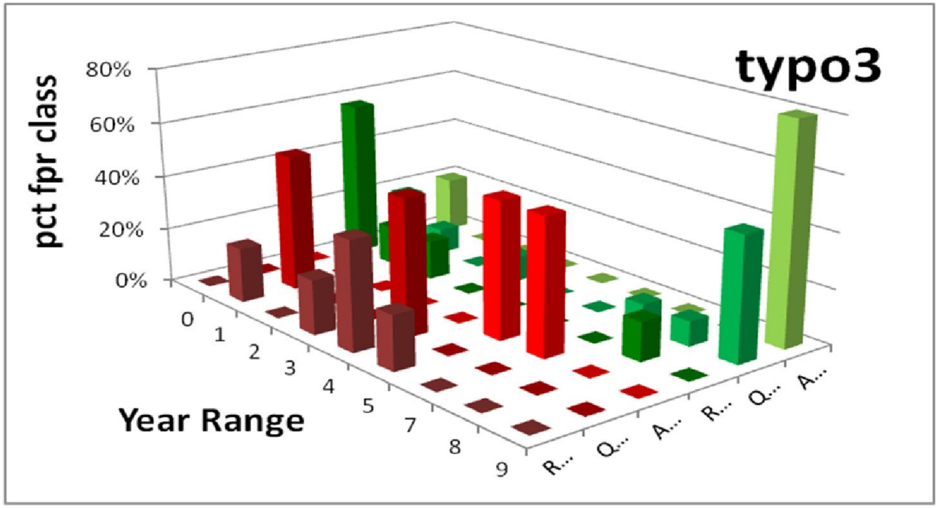
# The electrolysis pattern: survivors

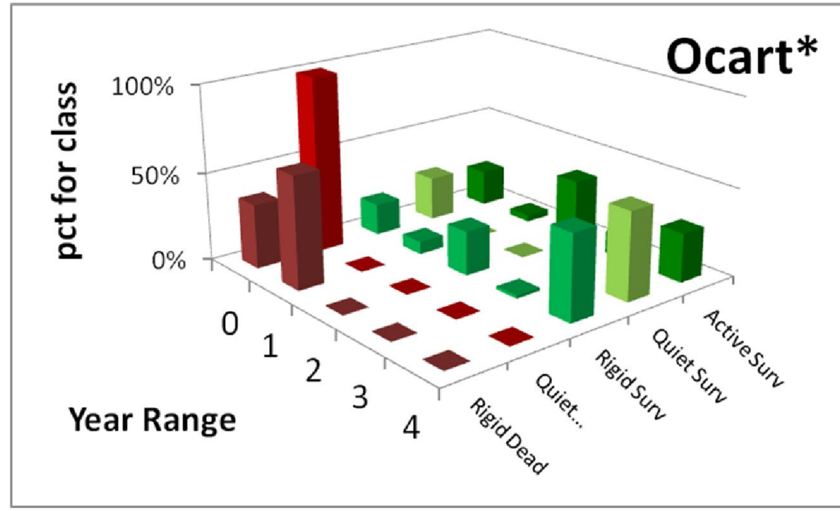



- The extreme clustering of active survivors to high durations
- The wider spread of (quite numerous) quiet survivors to a large span of durations with long trails of points
- The clustering of rigid survivors, albeit not just to one, but to all kinds of durations (frequently, not as high as quiet and active survivors)

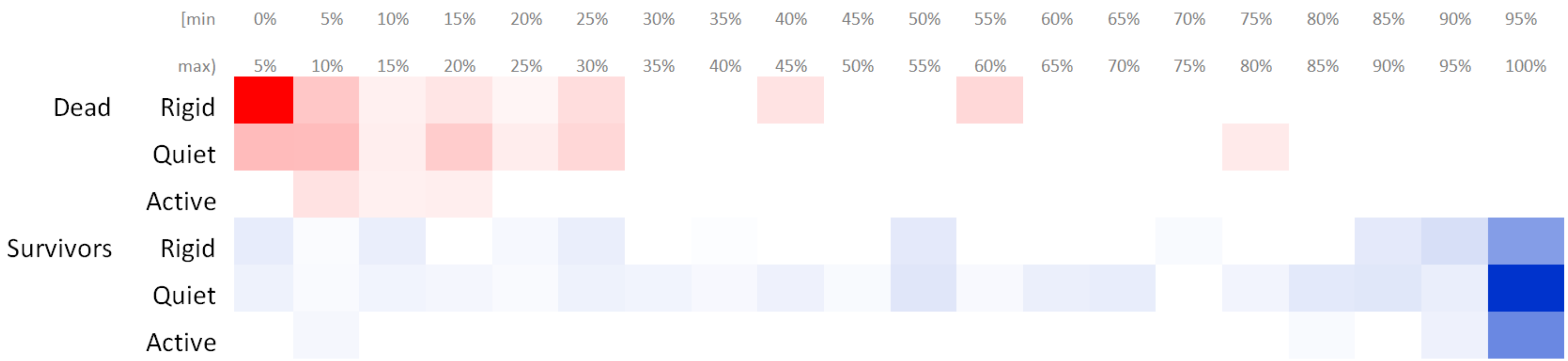
# The electrolysis pattern: dead




- The total absence of dead tables from high durations
- The clustering of rigid dead at low durations,
- the spread of quiet dead tables to low or medium durations, and
- the occasional presence of the few active dead, that are found also at low or medium durations, but in a clustered way

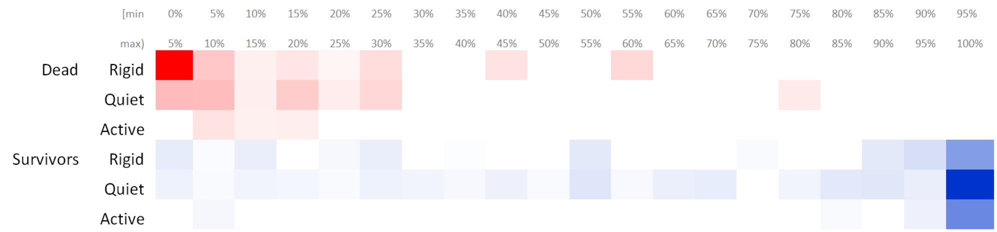
# Electrolysis as a heatmap showing the extreme bias between dead and survivor tables




- For each *LifeAndDeath* value, and for each duration range of 5% of the database lifetime, we computed the percentage of tables (over the total of the data set) whose duration falls within this range.
- We removed cells that corresponded to only one data set

The resulting heatmap shows the polarization in colors: brighter color signifies higher percentage of the population

# Gravitation to Rigidity

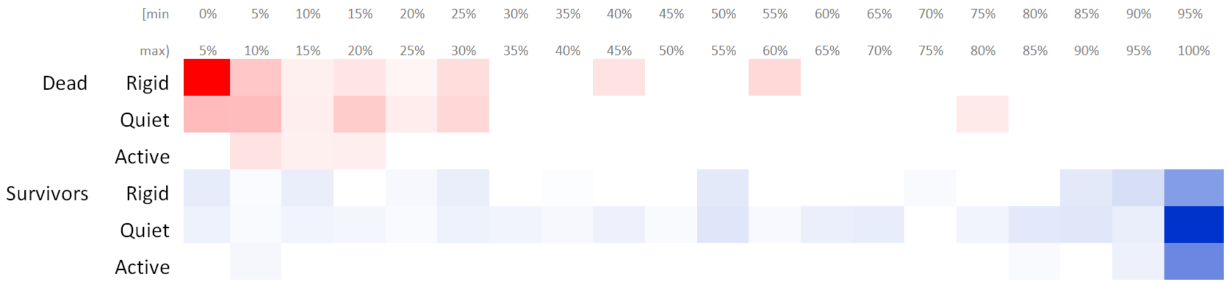


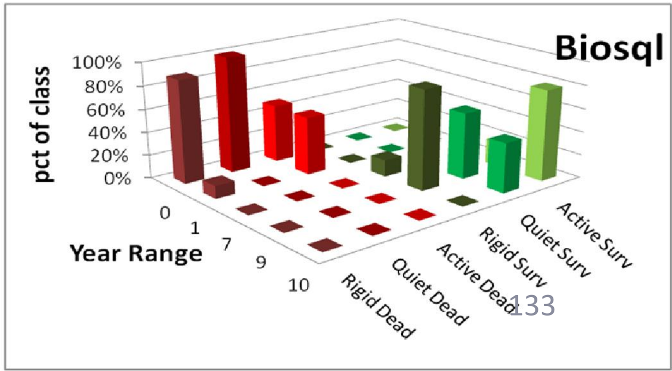

- Although the majority of survivor tables are in the quiet class, we can quite emphatically say that **it is the absence of evolution that dominates**!
  - Survivors vastly outnumber removed tables.
  - Similarly, rigid tables outnumber the active ones, both in the survival and, in particular, in the dead class.
  - Schema size is rarely resized, and only in survivors (not in the paper).
  - Active tables are few and do not seem to be born in other but early phases of the database lifetime.
- Evidently, not only survival is also stronger than removal, but **rigidity is also stronger a force than variability** and the combination of the two forces further lowers the amount of change in the life of a database schema.

# Electrolysis

- Yes, we can indeed find **patterns in the lives of tables,** during schema evolution!
- **Survivors, mostly long-lived (esp. active ones) and quietly active are radically different than dead tables, being mostly short-lived and rigid!**
- **Gravitation to rigidity rules:** we see more absence than presence of schema evolution!

Also studied [not part of the paper]: year of birth, schema size, schema resizing



… **How do foreign keys evolve?**

http://www.cs.uoi.gr/~pvassil/publications/2017_ER

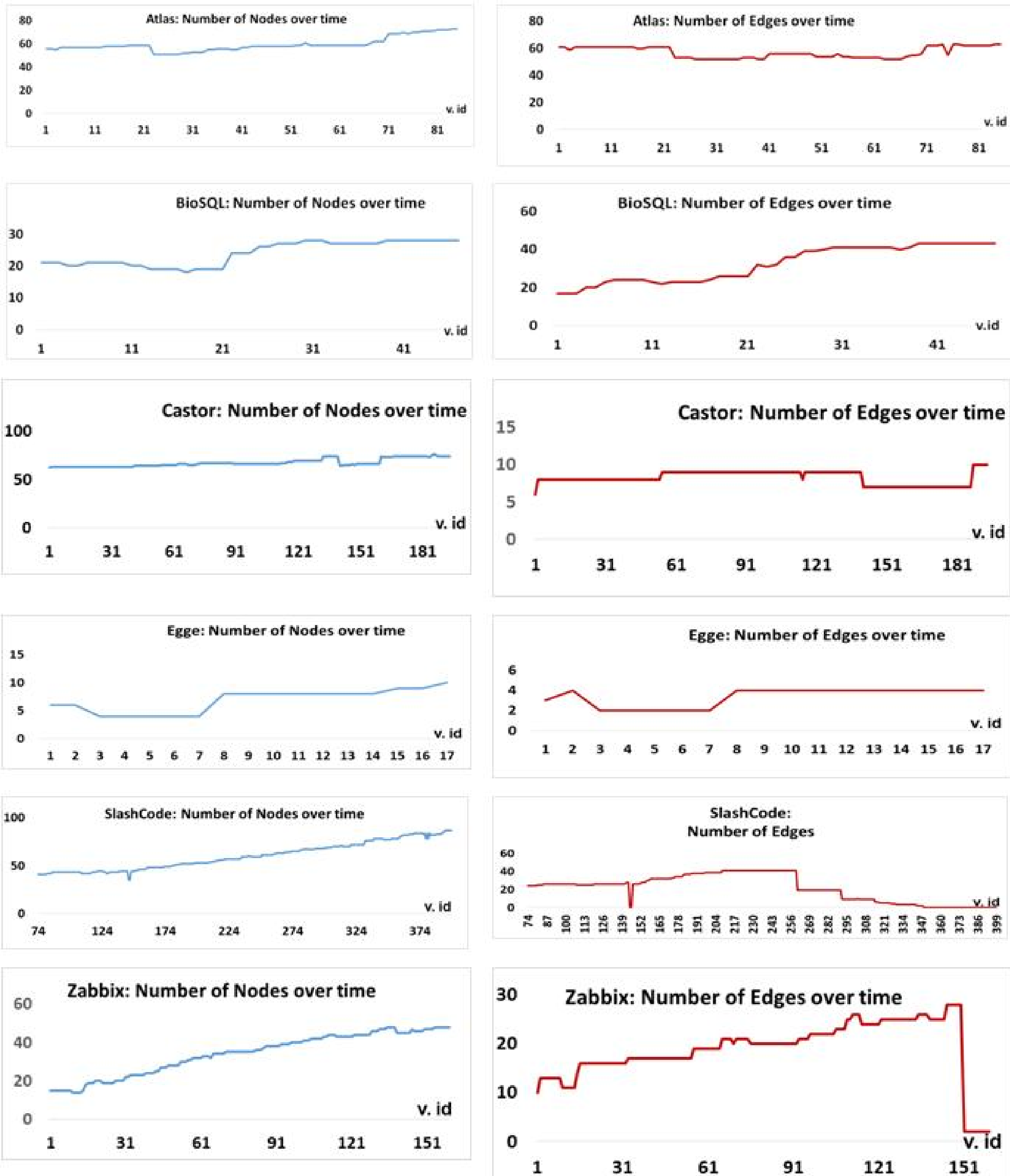
Panos Vassiliadis, Michail-Romanos Kolozoff*, Maria Zerva, Apostolos V. Zarras. 36th International Conference on Conceptual Modeling **(ER 2017)**, Nov. 6th-9th, 2017, Valencia Spain

# SCHEMA EVOLUTION AND FOREIGN KEYS: BIRTH, EVICTION, CHANGE AND ABSENCE

# Research Questions

- **How do foreign keys evolve over time?**
  - Do tables and foreign keys evolve in sync?
  - When & How do foreign keys germinate & die?

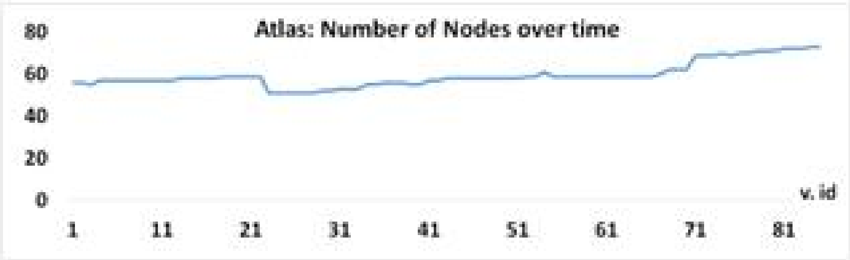
- … as we will see, these questions led to unexpected results and more insights on how developers deal with foreign keys…

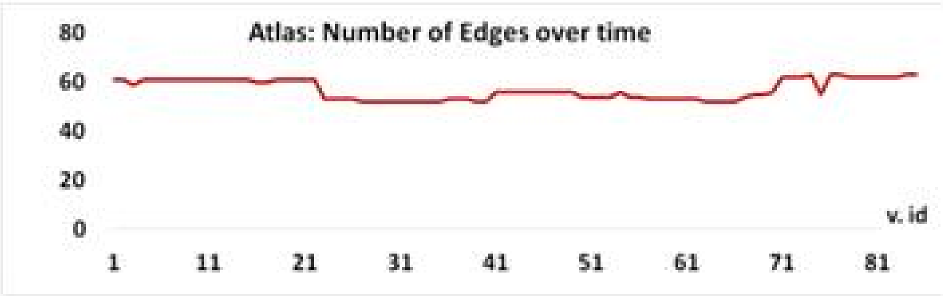


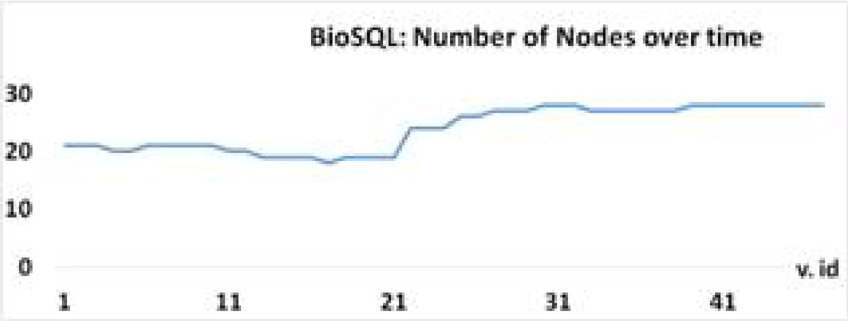

# Evolution of Tables & FK's

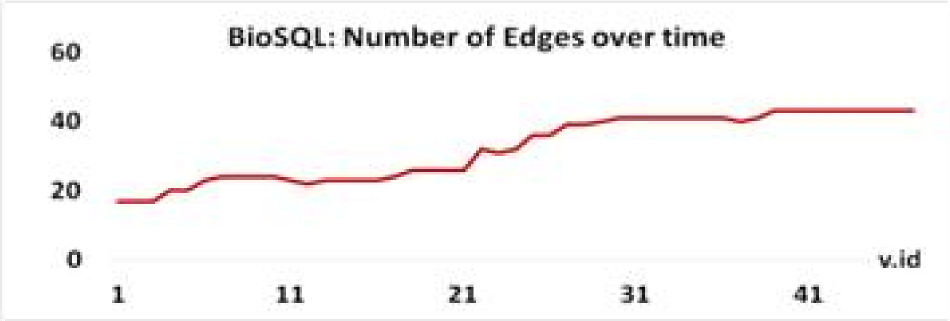
- Tables grow in all cases (known from previous research) with periods of slow growth, calmness, spikes of extension, and occasional cleanups
- Foreign Keys are treated with different mentalities. 3 families:
  - Scientific
  - Comp. Toolkits
  - CMS's

# Evolution of Tables & FK's: Scientific projects



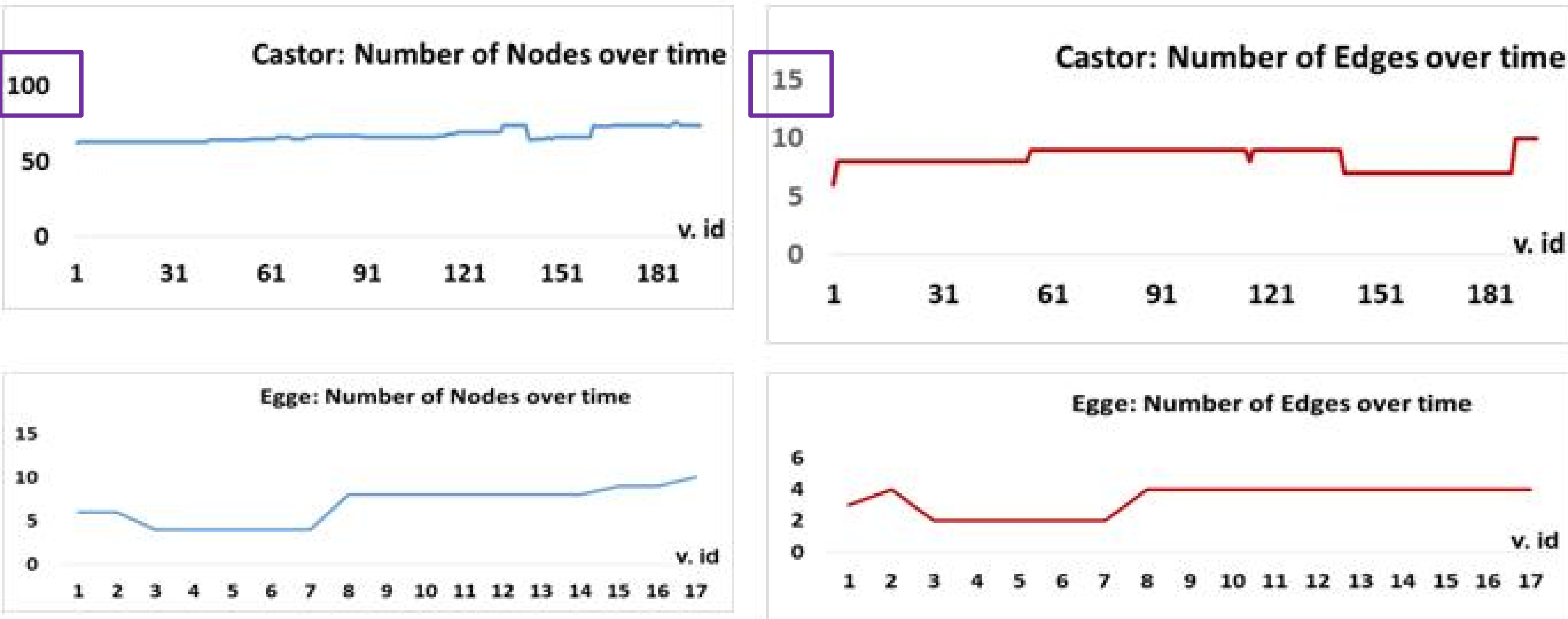



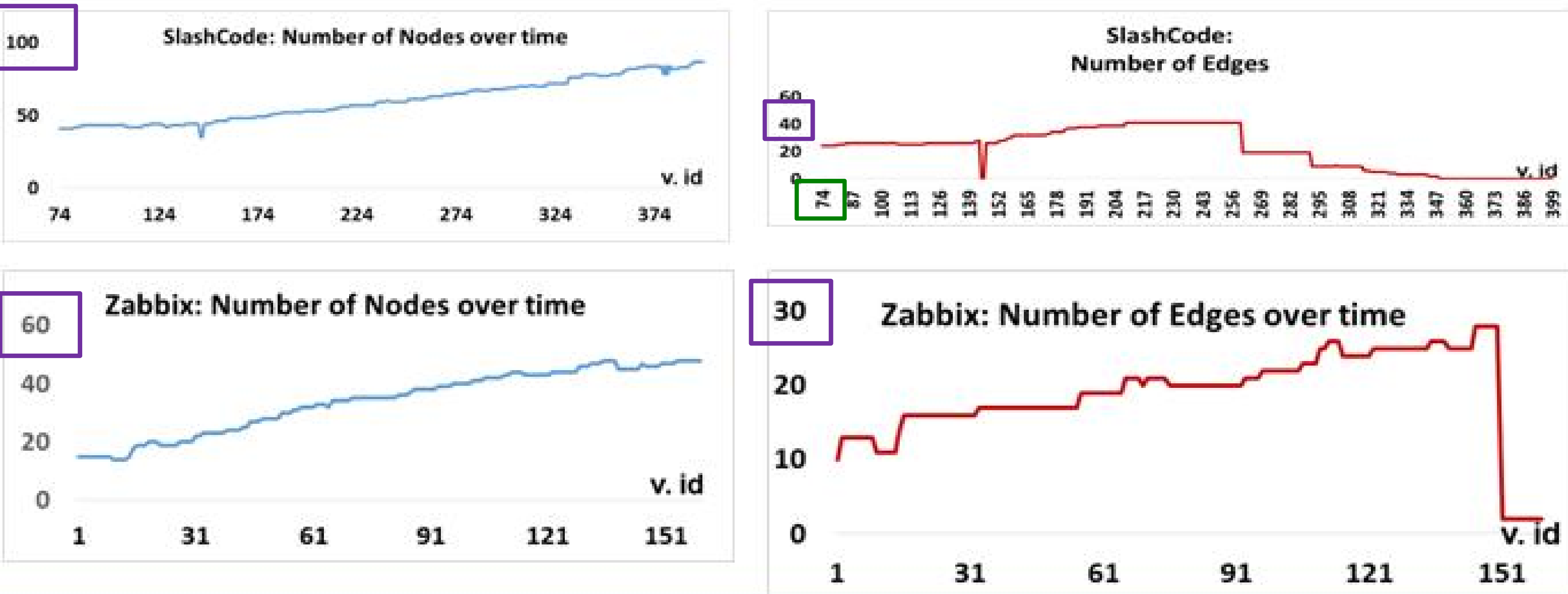

- Tables and FKS grow in synch, in both cases
- Growth comes with expansion periods, shrinkage actions, and periods of calmness in terms of both tables and foreign keys.

# Evolution of Tables & FK's: Computational Resource Toolkits



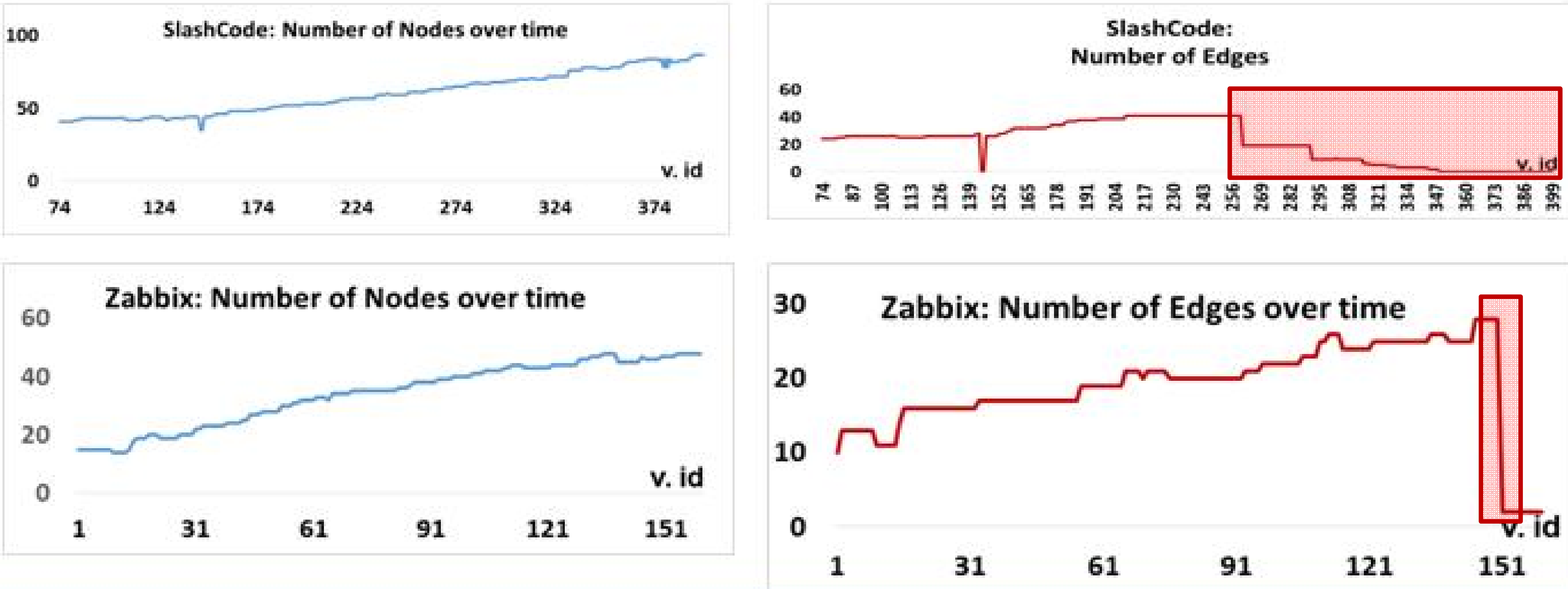

- Tables and FKS grow little and slowly; for Castor, not exactly in sync
- Castor: observe **how scarce FK's are** (too few tables come with FK's, see vertical axis)

# Evolution of Tables & FK's: Content Management Systems (CMS's)




- **FK scarcity**: really big at Slashcode, moderate at Zabbix
- Slashcode started <u>without</u> foreign keys at all; 1st set of FK's in v. 74. Zabbix seems to show a certain degree of syncronized growth
- **Yet, … both CMS's end up with no FK's!!** *-> see next*

# What an unpleasant surprise: developers can resort in full removal of foreign keys!




- Slashcode: there is a clear phase of **progressive removal**
- Zabbix: **abrupt removal** of almost the entire set of foreign keys in a single transition. We have <u>no knowledge on why</u> this happened, & it is <u>unexpected</u> based on how FK's had been treated till then…

# How do FK's germinate and die?

- We classified FK's births and deaths in 4 categories
- Births
  - **Born with table**: when either the source or the target table is born along with the foreign key,
  - **Explicit addition**: when a foreign key is added to two existing tables.
- Deletions
  - **Died with table**: when either the source or the target table is removed along with the foreign key,
  - **Explicit deletion**: when neither of the source or target tables gets deleted and only the foreign key is removed.

# Stats on FK Change

| | | | Atlas | Biosql | Egee | Castor | Slashcode | Zabbix |
|---|---|---|---|---|---|---|---|---|
| | **Diachronic Graph** | **TablesDG** | 88 | 45 | 12 | 91 | 126 | 58 |
| | | **FK'sDG** | 88 | 79 | 6 | 13 | 47 | 38 |
| | **Start/End** | **FKs@start** | 61 | 17 | 3 | 6 | 0 | 10 |
| | | **FKs@end** | 65 | 52 | 5 | 10 | 0 | 2 |
| **#FKs_added …** | **… in absolute numbers** | **Total** | 41 | 81 | 4 | 8 | 77 | 28 |
| | | **Born w/ table** | 37 | 71 | 3 | 2 | 21 | 24 |
| | | **Explicit addition** | 4 | 10 | 1 | 6 | 56 | 4 |
| | **… as pct** | **(%)Born w/ table** | 90% | 88% | 75% | 25% | 27% | 86% |
| | | **(%)Explicit addition** | 10% | 12% | 25% | 75% | 73% | 14% |
| **#FKs_removed …** | **… in absolute numbers** | **Total** | 37 | 46 | 2 | 4 | 77 | 36 |
| | | **Died w/ table** | 25 | 42 | 2 | 2 | 16 | 8 |
| | | **Explicit deletion** | 12 | 4 | 0 | 2 | 61 | 28 |
| | **… as pct** | **(%)Died w/ table** | 68% | 91% | 100% | 50% | 21% | 22% |
| | | **(%)Explicit deletion** | 32% | 9% | 0% | 50% | 79% | 78% |

# Stats on FK Change

| | | | Atlas | Biosql | Egee | Castor | Slashcode | Zabbix |
|---|---|---|---|---|---|---|---|---|
| | Diachronic Graph | TablesDG | 88 | 45 | 12 | | | |
| | | FK'sDG | 88 | 79 | 6 | | | |
| | Start/End | FKs@start | 61 | 17 | 3 | | | |
| | | FKs@end | 65 | 52 | 5 | | | |
| #FKs_added … | … in absolute numbers | Total | 41 | 81 | 4 | | | |
| | | Born w/ table | 37 | 71 | 3 | | | |
| | | Explicit addition | 4 | 10 | 1 | | | |
| | … as pct | (%)Born w/ table | 90% | 88% | 75% | | | |
| | | (%)Explicit addition | 10% | 12% | 25% | | | |
| #FKs_removed … | … in absolute numbers | Total | 37 | 46 | 2 | | | |
| | | Died w/ table | 25 | 42 | 2 | | | |
| | | Explicit deletion | 12 | 4 | 0 | | | |
| | … as pct | (%)Died w/ table | 68% | 91% | 100% | | | |
| | | (%)Explicit deletion | 32% | 9% | 0% | | | |

Atlas, Biosql and Egee (less) deal with **FK's as regular part of the schema**

FK's are, to a large extent …

- Born with tables
- Removed with tables

# Stats on FK Change

| | | | Atlas | Biosql | Egee | Castor | Slashcode | Zabbix |
|---|---|---|---|---|---|---|---|---|
| | Diachronic Graph | TablesDG | | | | 91 | 126 | 58 |
| | | FK'sDG | | | | 13 | 47 | 38 |
| | Start/End | FKs@start | | | | 6 | 0 | 10 |
| | | FKs@end | | | | 10 | 0 | 2 |
| #FKs_added … | … in absolute numbers | Total | | | | 8 | 77 | 28 |
| | | Born w/ table | | | | 2 | 21 | 24 |
| | | Explicit addition | | | | 6 | 56 | 4 |
| | … as pct | (%)Born w/ table | | | | 25% | 27% | 86% |
| | | (%)Explicit addition | | | | 75% | 73% | 14% |
| #FKs_removed … | … in absolute numbers | Total | | | | 4 | 77 | 36 |
| | | Died w/ table | | | | 2 | 16 | 8 |
| | | Explicit deletion | | | | 2 | 61 | 28 |
| | … as pct | (%)Died w/ table | | | | 50% | 21% | 22% |
| | | (%)Explicit deletion | | | | 50% | 79% | 78% |

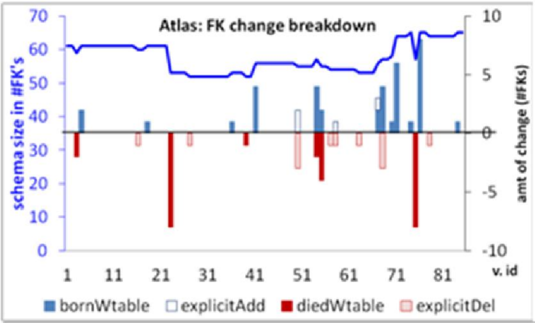
Castor & Slashcode (both with a really small minority of FK's) deal with **FK's as an ad-hoc add on**: FK's are mostly explicitly added/ removed

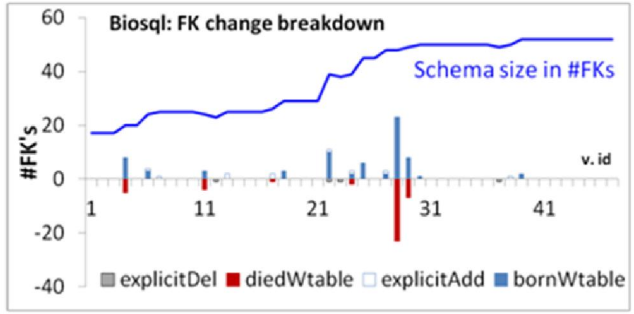
Zabbix has a mixed style: explicit del. and add. w. tables (& a sudden style change)

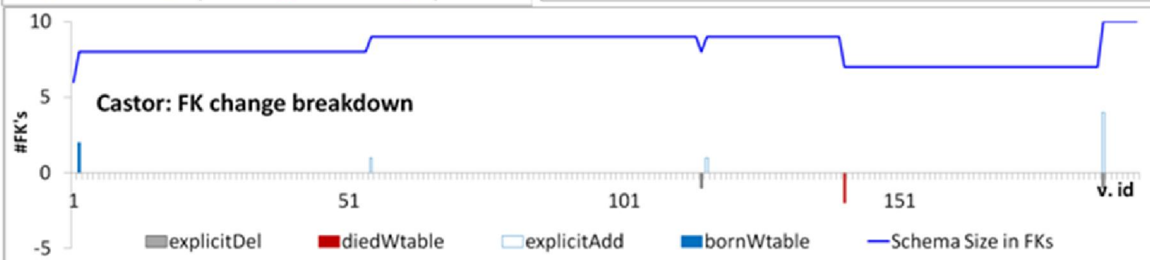
# Families of developer profiles wrt the treatment of Foreign Keys

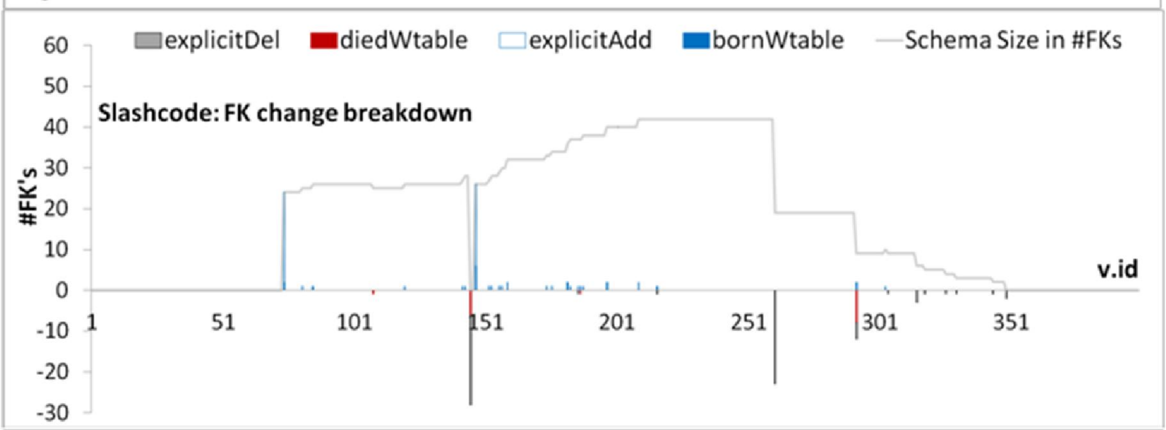
- **Integral part of schema**: fairly large pct of tables involved in FKs, grow in sync with tables, germinate and die with them
- **Disposable Add-on**: small pct of tables involved in FK's, explicit additions and deletions, easy to remove them (in some cases, entirely!)
- **Mixed**: can be with a change of style

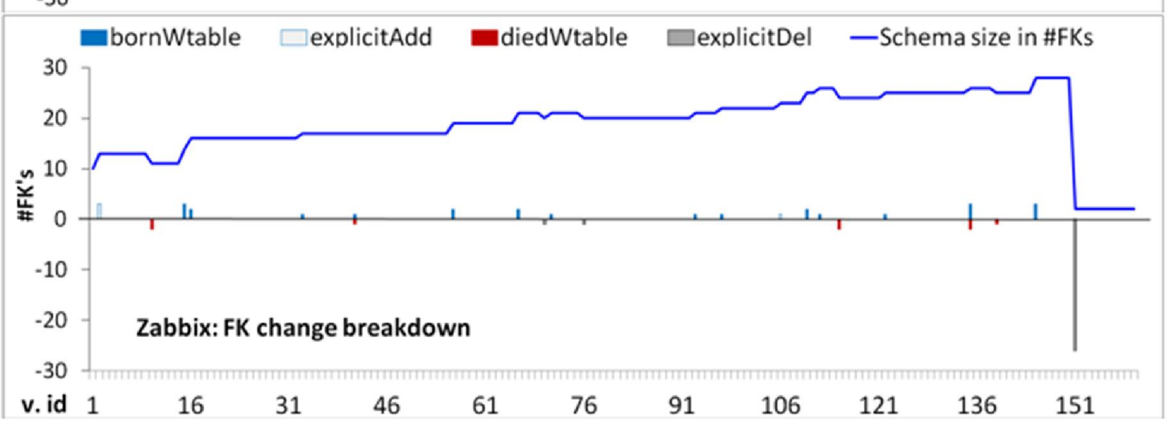
# Heartbeat of change




**Birth & deaths are proportionally spread in time** -- except Atlas.

The **volume of change is typically low**: most changes ~ 1 FK.
Exceptions:
(a) explicit mass add & del,
(b) do-undo actions (Atlas, Slashcode and Castor), and,
(c) restructuring due to table renamings (4 in Biosql, 2 in Zabbix).

# Percentage of transitions with FK change

| | Total # transitions | Total # transitions with FK change | Pct. of transitions with FK change |
|---|---|---|---|
| Atlas | 85 | 25 | 29% |
| BioSQL | 46 | 19 | 41% |
| Egee | 16 | 3 | 19% |
| Castor | 191 | 6 | 3% |
| Slashcode | 398 | 34 | 9% |
| Zabbix | 159 | 22 | 14% |

Common theme in all the data sets: the **consistent scarcity of FK changes**

- Scientific data sets: short active period + treatment of FK's as an integral part of the schema (births and deaths of tables and FK's in sync) => high pct of transitions with FK change
- The rest: FK b&d are rare and explicit (w/o mass removals, would be even less)

# Characteristics of the heartbeat of schemata wrt Foreign Keys

- **Scarcity of FK change**: expectedly very few transitions come with FK change, except for idiosyncratic cases
- **Low volume**: typically 1 FK change at a time, except for mass add/del
- **Birth & deaths are proportionally spread in time**
- Occasional **do-undo** and restructuring due to **table renames**

# Slashcode: the disappearing FK's

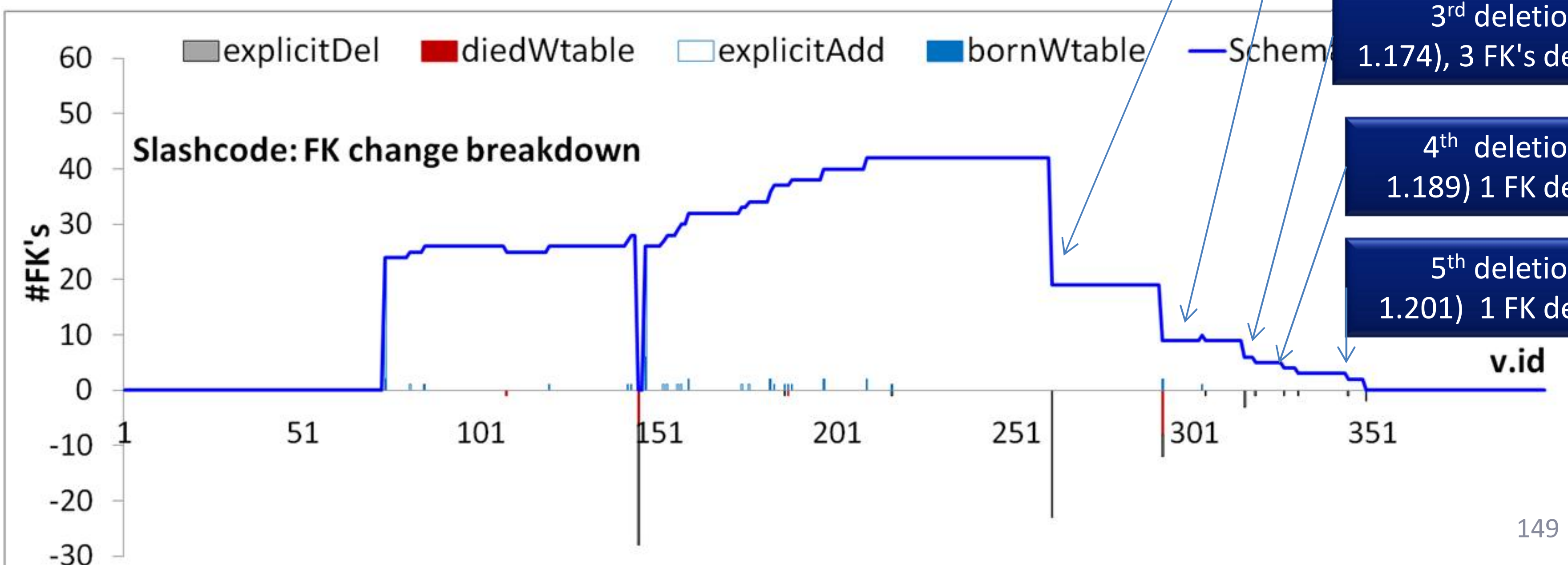
1st massive foreign key removal (rev 1.120), 22 FK's deleted.

2nd massive deletion (rev 1.151), 10 FK's deleted

3rd deletion (rev 1.174), 3 FK's deleted

4th deletion (rev 1.189) 1 FK deleted

5th deletion (rev 1.201) 1 FK deleted



"Commented-out foreign keys are ones which currently cannot be used because they refer to a primary key which is NOT NULL AUTO INCREMENT and the child's key either has a default value which would be invalid for an auto increment field, typically NOT NULL DEFAULT '0'.
Or, in some cases, the primary key is e.g. VARCHAR(20) NOT NULL and the child's key will be VARCHAR(20). The possibility of NULLs negates the ability to add a foreign key. <= That's my current theory, but it doesn't explain why discussions.topic SMALLINT UNSIGNED NOT NULL DEFAULT '0' is able to be foreign-keyed to topics.tid SMALLINT UNSIGNED NOT NULL AUTO INCREMENT"

1st massive foreign key removal (rev 1.120), 22 FK's deleted.

"Stories is now InnoDB and these other tables are still MyISAM, so no foreign keys between them."

2nd massive deletion (rev 1.151), 10 FK's deleted

"This doesn't work, makes createStory die. These don't work, should check why..."

3rd deletion (rev 1.174), 3 FK's deleted

"This doesn't work, since in the install pollquestions is populated before users, alphabetically"

4th deletion (rev 1.189) 1 FK deleted

"This doesn't work, since discussion may be 0."

5th deletion (rev 1.201) 1 FK deleted

# Slashcode: what did the comments say?

- **The main problem seems to be the difficulty of developers with the tuning and handling of both foreign and primary keys.**
- Sometimes difficulties are hard -- e.g., different storage engines, typically due to performance reasons
- Some difficulties are complicated due to technicalities like autonumbering
- Sometimes fixes could be found with some effort (e.g., changing the order of table population, or using numeric data types for primary keys, or inserting some “goalkeeper” values at FK target table)

# Scarcity of Foreign keys

- A 2013 collection of schema histories, lists **21 data sets**, -- some have more than one target DBMS variants.

```
$ cd RESEARCH/Github/EvolutionDatasets
$ ls -d * */*
CERN          CMS's/Coppermine  CMS's/XOOPS      Med
CERN/Atlas    CMS's/DekiWiki    CMS's/Zabbix     Med/Ensembl
CERN/CASTOR   CMS's/Joomla 1.5  CMS's/e107       Med/biosql
CERN/DQ2      CMS's/NucleusCMS  CMS's/opencart   README.md
CERN/DRAC     CMS's/SlashCode   CMS's/phpBB
CERN/EGEE     CMS's/TikiWiki    CMS's/phpwiki
CMS's         CMS's/Typo3       CMS's/wikimedia
```

- **How many data sets contain foreign keys?**
- Try this (also backed by manual sampling):

```
grep -rl "FOREIGN" . >> ALL-FKs-by-grep.ascii
awk '{split($0,a,"/"); print a[2],a[3]}' ALL-FKs-by-grep.ascii  |
uniq
```

# Scarcity of Foreign keys

**- How many data sets, out of the 21, contain foreign keys?**

```
CERN Atlas
CERN CASTOR
CERN EGEE
CMS's SlashC
CMS's Zabbix
Med biosql

CERN DQ2
CERN DIRAC
Med Ensembl
```

The **6** data sets reported here

+

**DQ2** (only in the mySQL, not in the Oracle version): FK’s in 19 versions out of the 55.
Starts with 2 FK's and ends with 1.

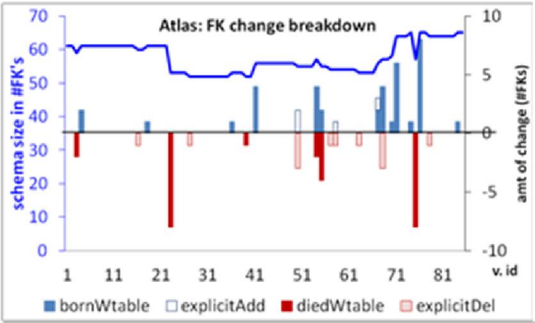
**DIRAC** (not in the production folder, only at python+mysql).
9 tables at first version, 15 tables at last version
Starts with 10 FK's, ends with 8

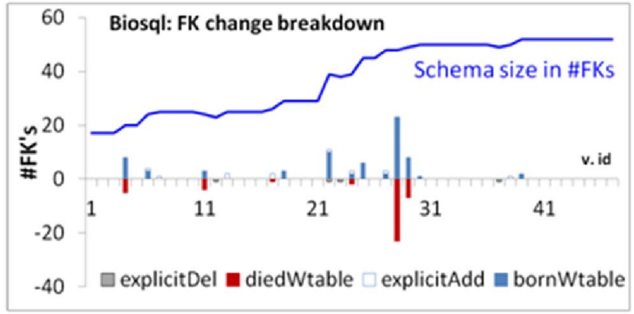
**Ensembl**: not able to link FK DDL files to table evolution, yet

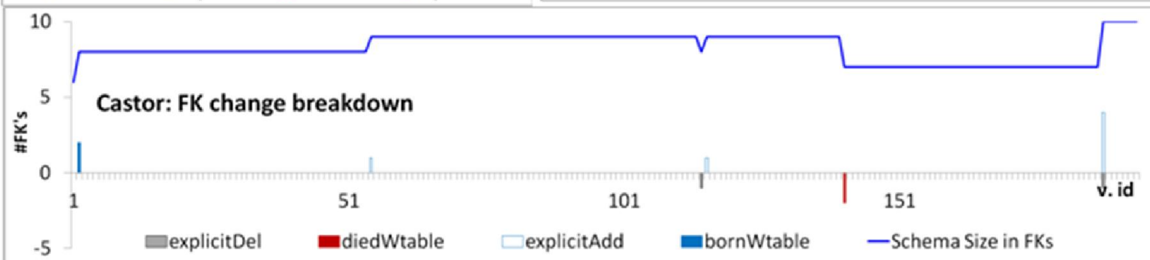
**- 9 out of the 21 data sets do** (including 3 that are really small for harnessing valuable results, spec., Egee, DQ2, DIRAC)

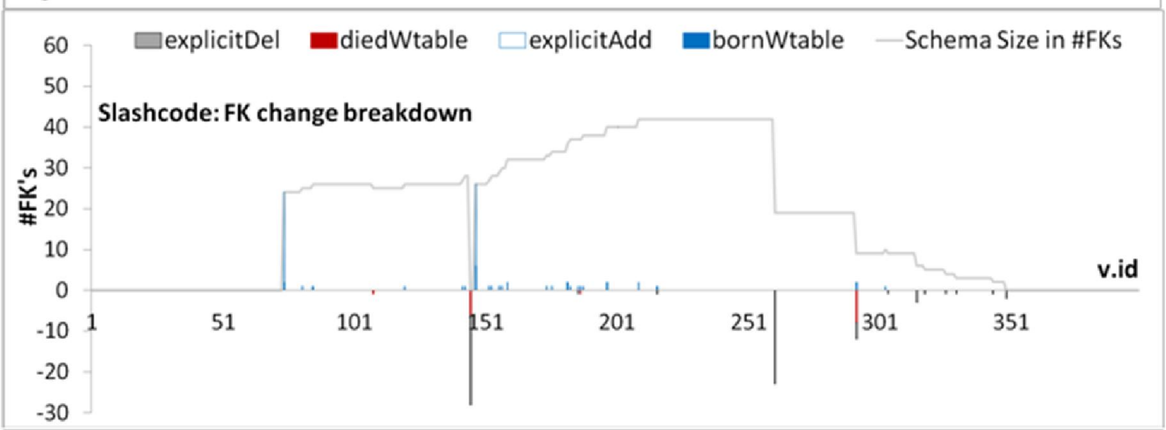

http://www.boldomatic.com/view/post/G_xPI

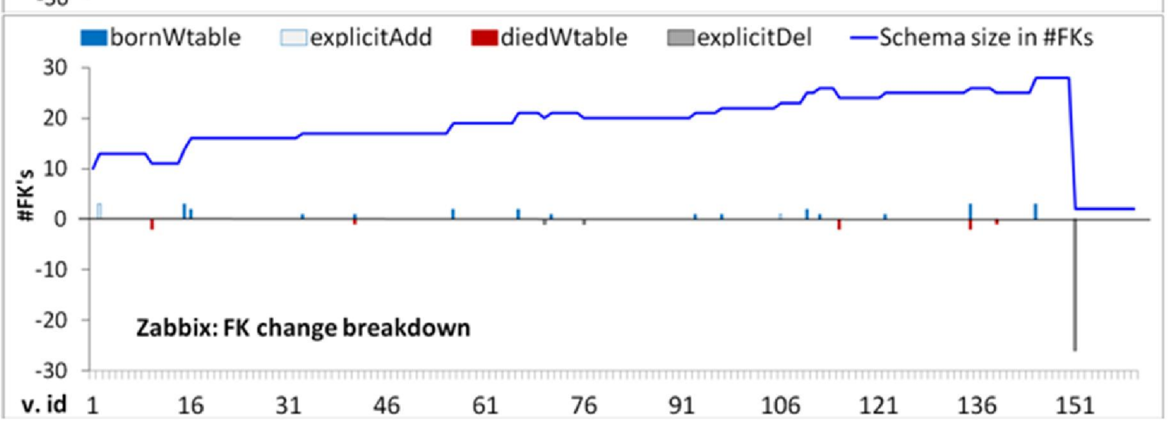
# Heartbeat of change




**Birth & deaths are proportionally spread in time** -- except Atlas.

The **volume of change is typically low**: most changes ~ 1 FK.

Exceptions:

(a) explicit mass add & del,

(b) do-undo actions (Atlas, Slashcode and Castor), and,

(c) restructuring due to table renamings (4 in Biosql, 2 in Zabbix).

# Foreign Key Evolution comes with different treatments:

- Sometimes, **FK’s are treated as an integral part of the system**, and they are born and evicted along with table birth and eviction.
- Other times, **FK’s are treated as a disposable add-on**: only a small subset of the tables involved in FK’s; birth and eviction of FK’s rarely performed in synch with their tables. If technical difficulties arise, it is possible to witness the **complete removal of FK’s** from the schema.
- Another sign of concern is that in all the CMS’ we collected, **FK’s are too scarce**
- More results in the paper: **stats**, **threats to validity**, and, the treatment of the **evolving schema as an evolving graph**

**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

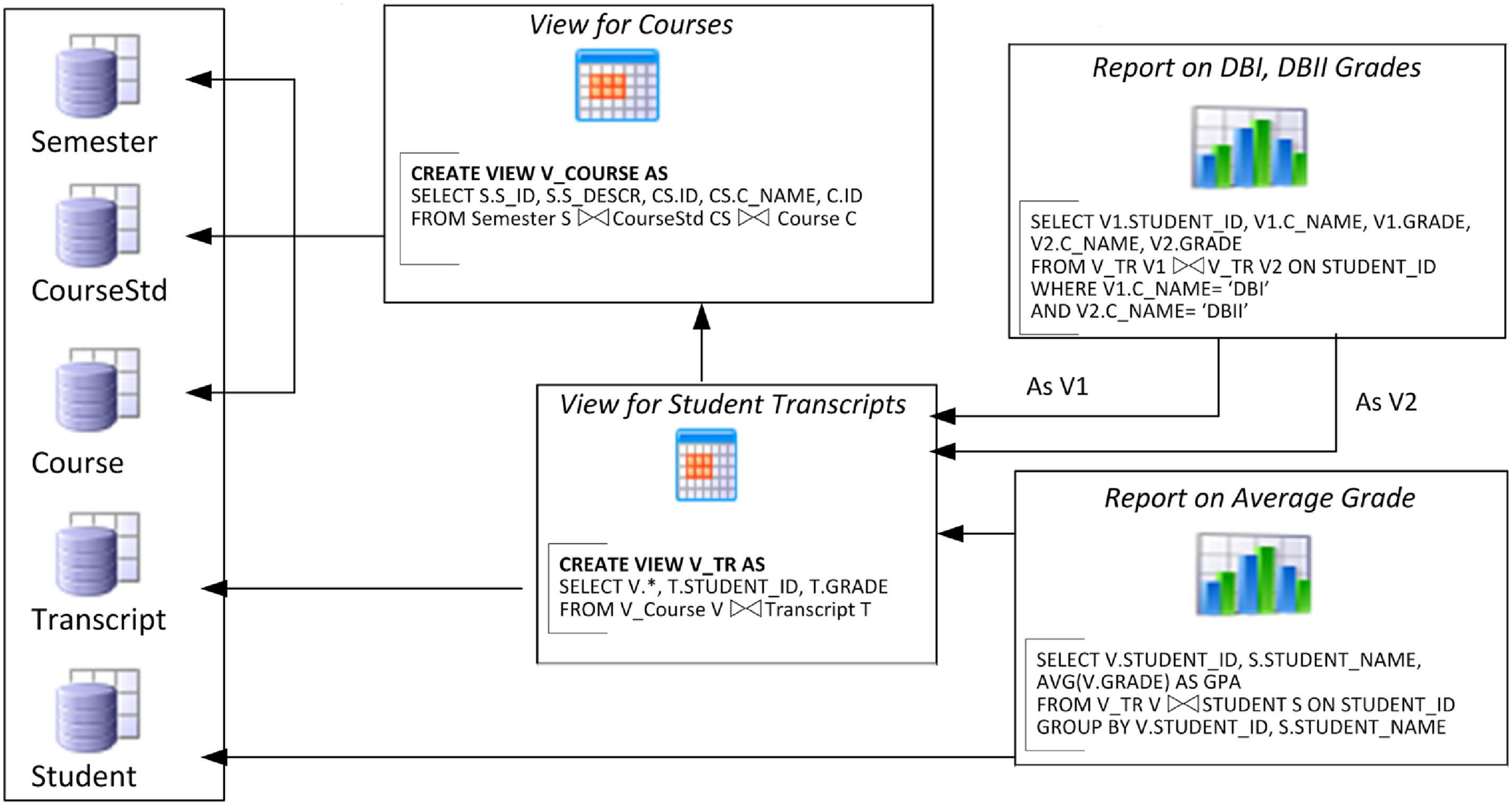
… and data intensive ecosystems…

# IMPACT ASSESSMENT

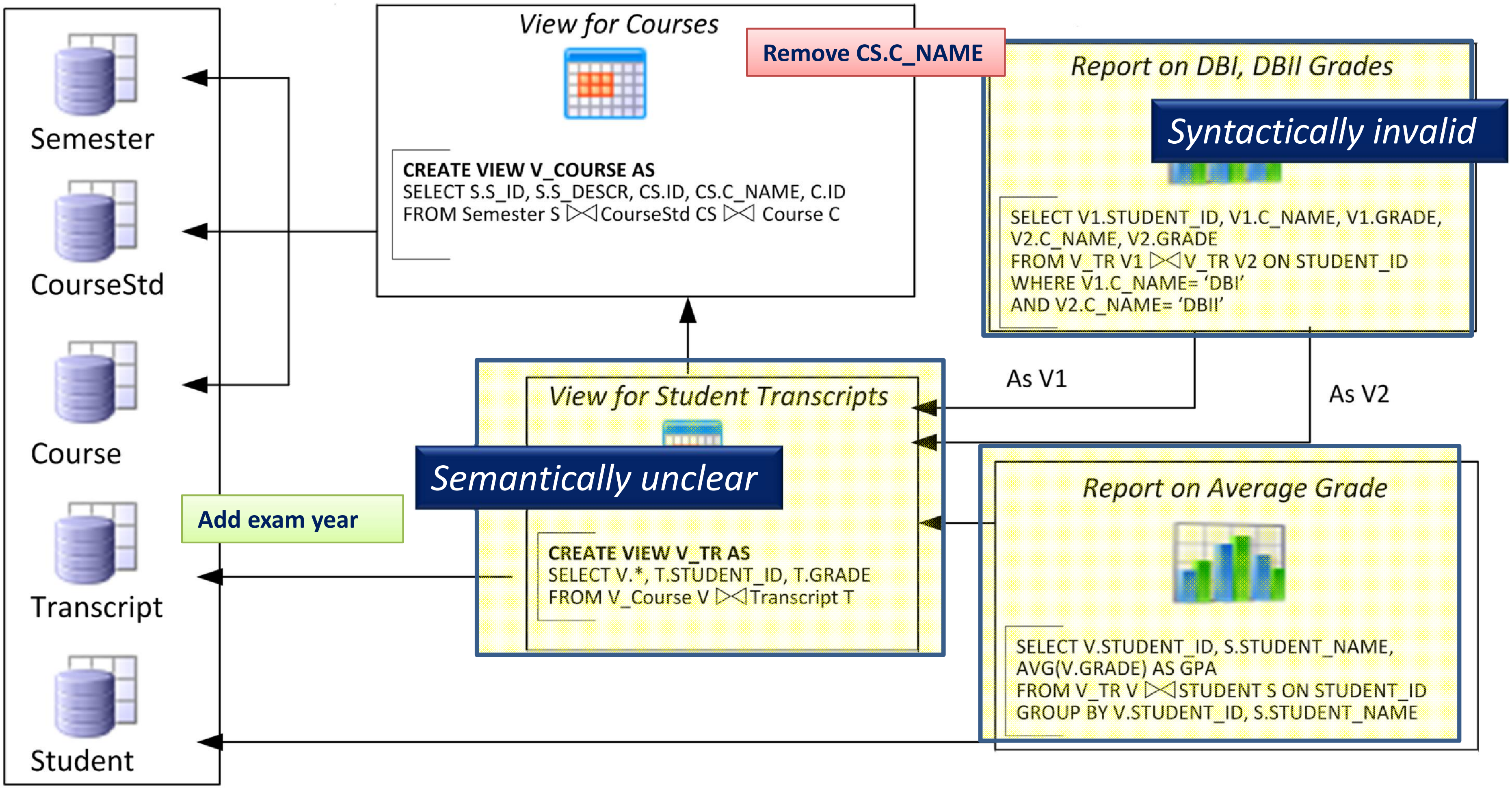
# Data intensive ecosystems

- 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

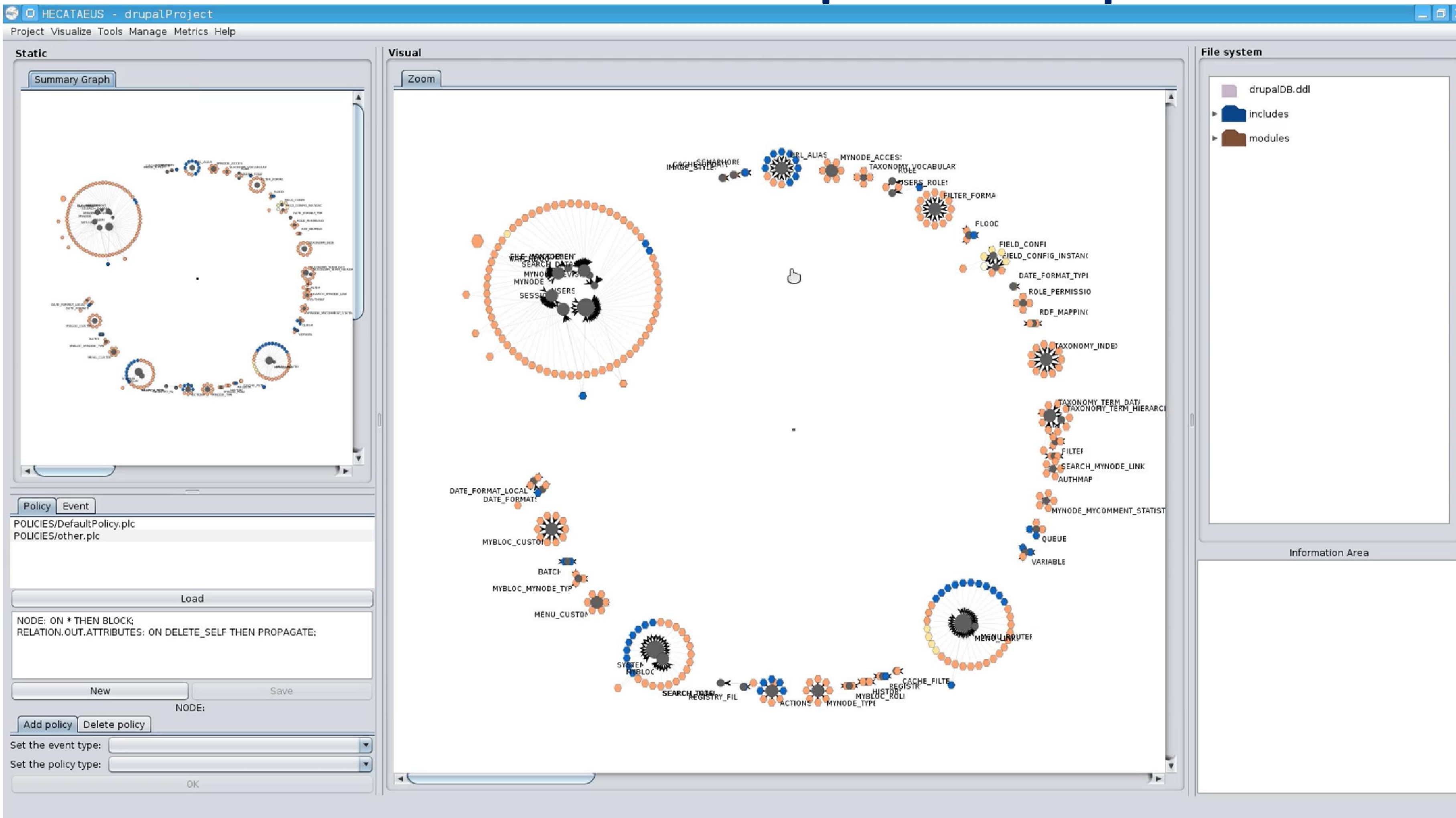
We would like: evolution predictability

i.e., control of what will be affected

before changes happen

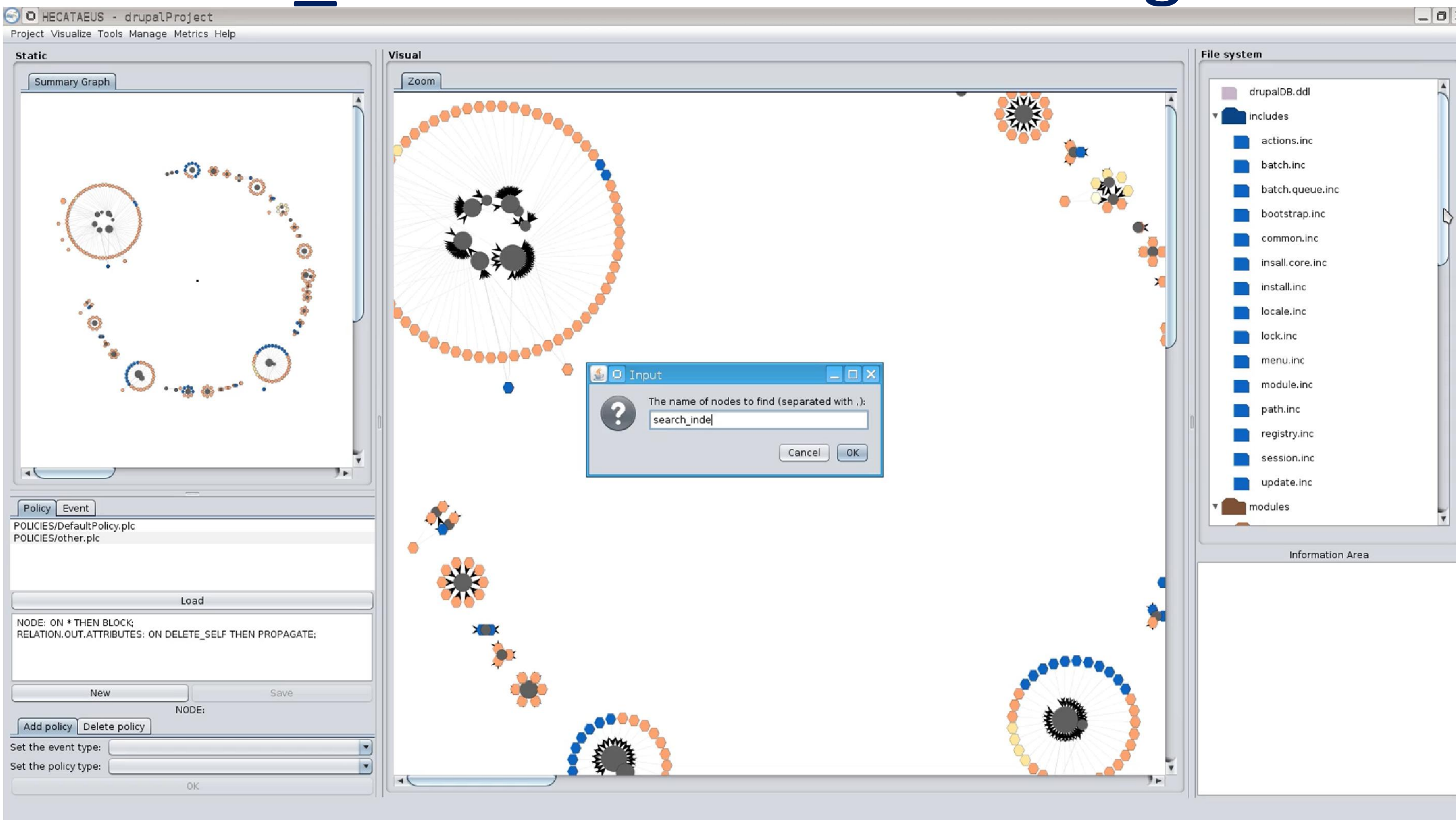
- Learn what changes & how
- Find ways to quarantine effects

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

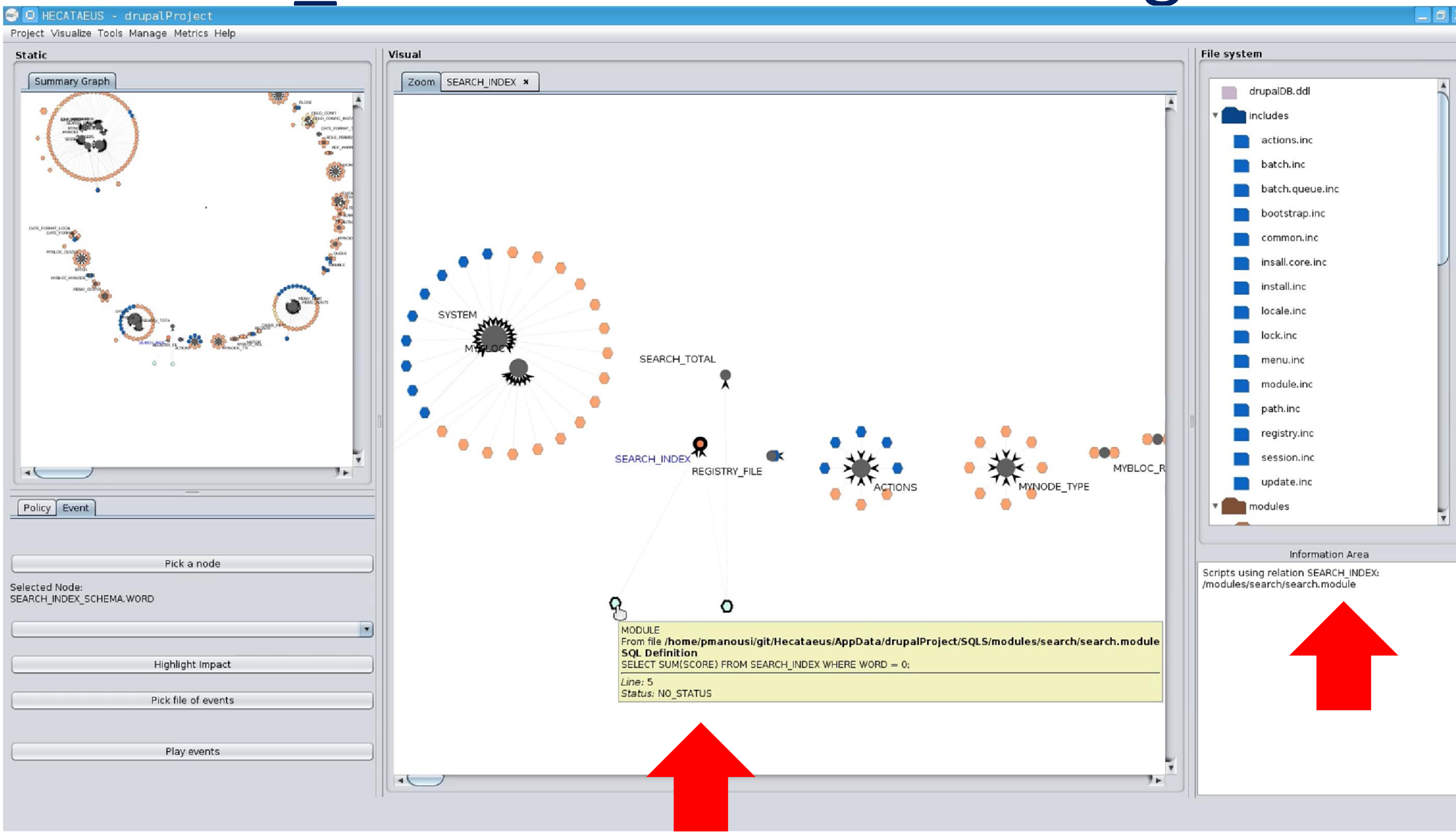



**http://www.cs.uoi.gr/~pvassil/projects/hecataeus/**

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

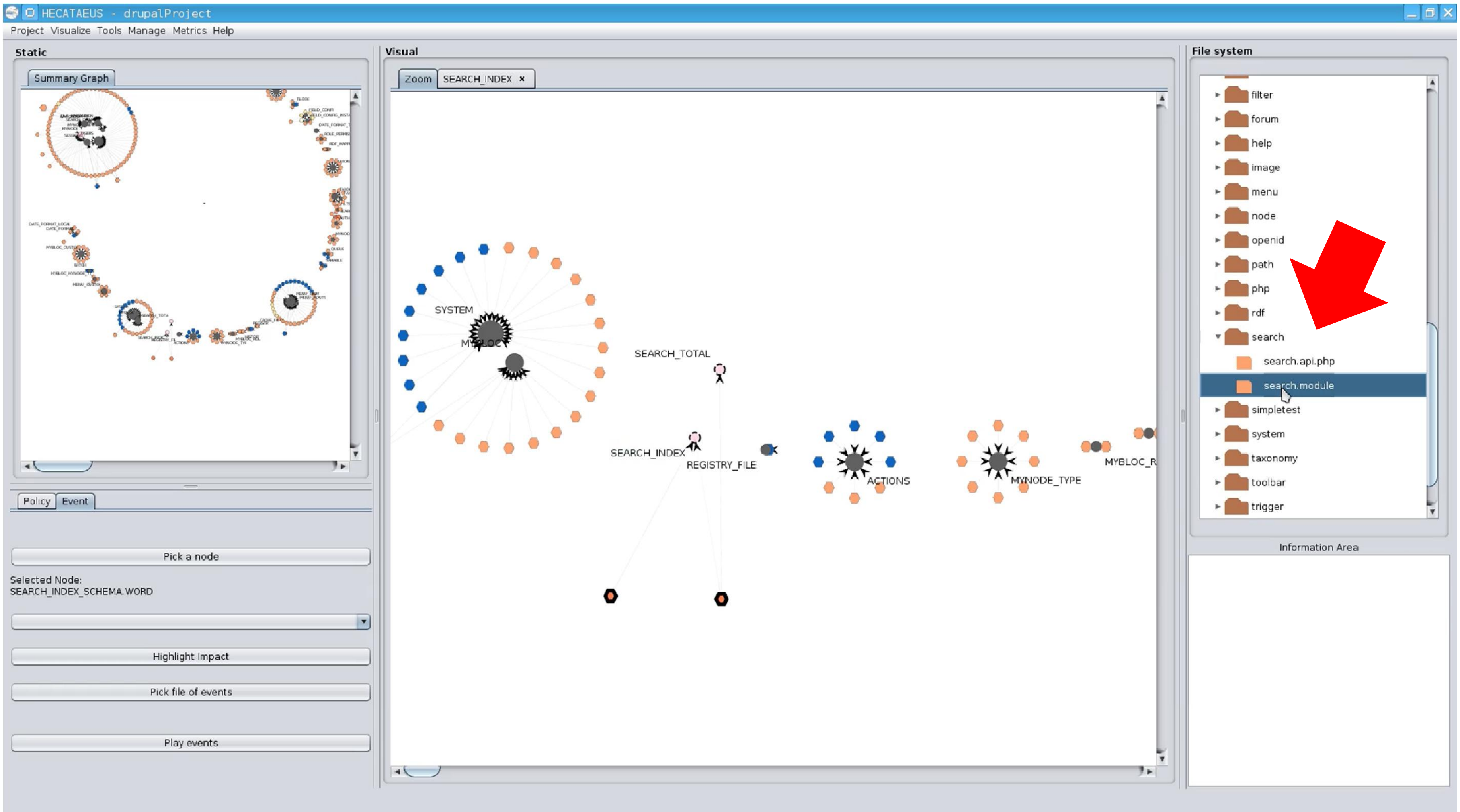


# What happens if I modify table search_index? Who are the neighbors?

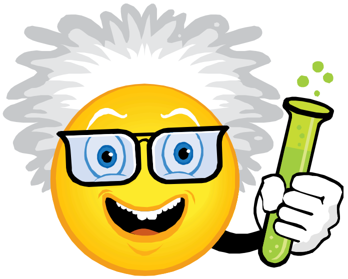



Tooltips with info on the script & query

# In the file structure too…



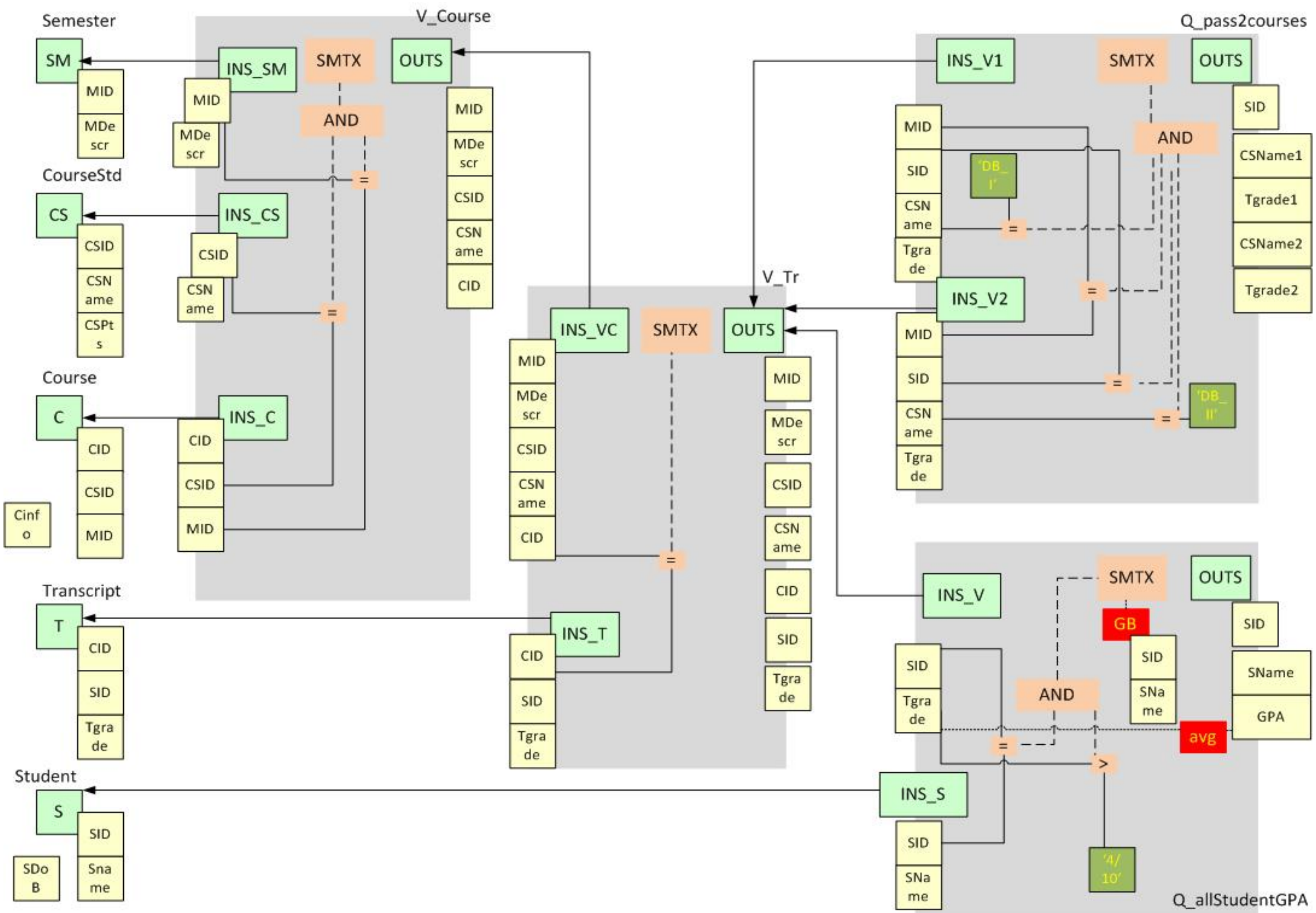
# How to handle evolution?



- 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

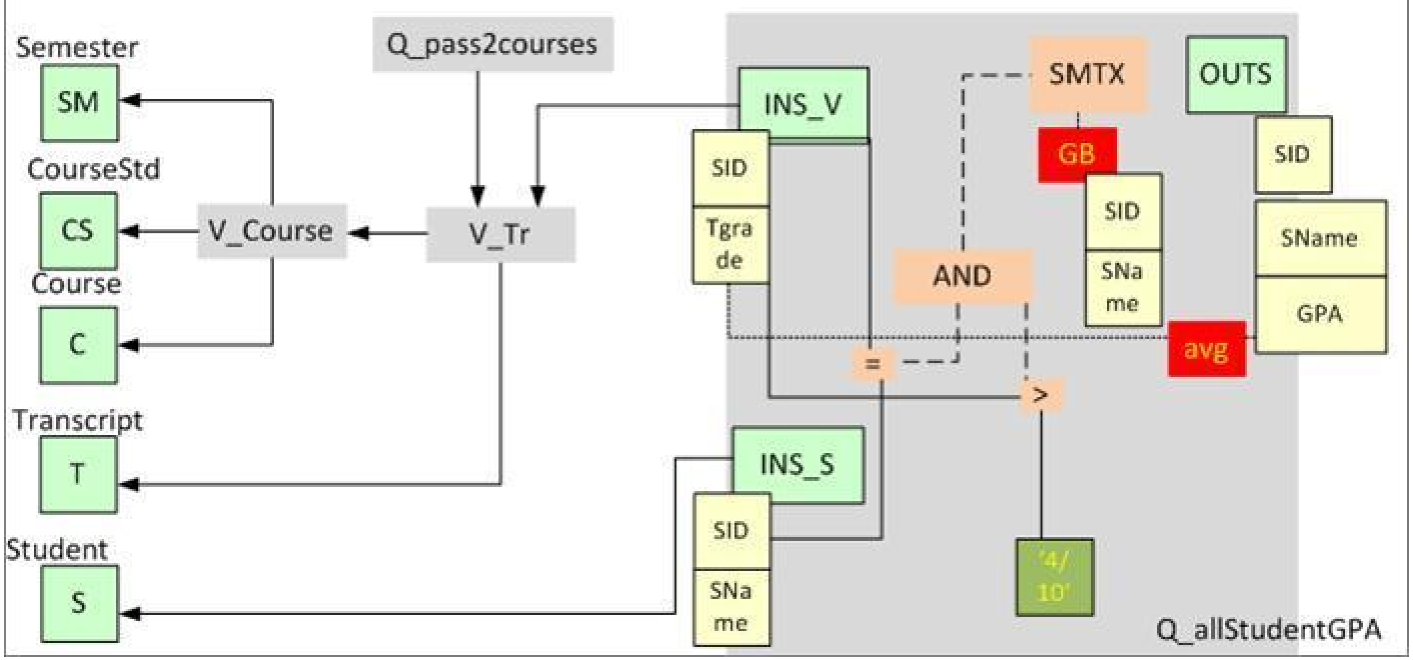


# Architecture Graph

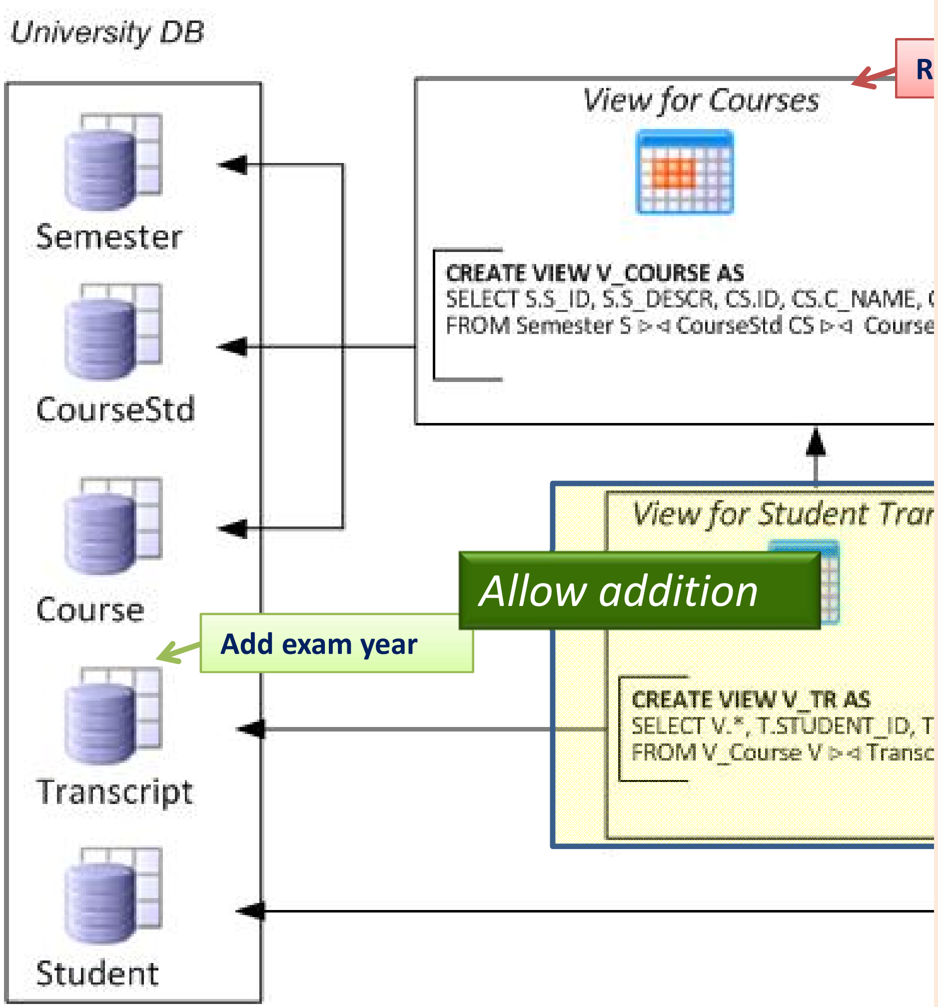
Modules and Module Encapsulation

**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
```



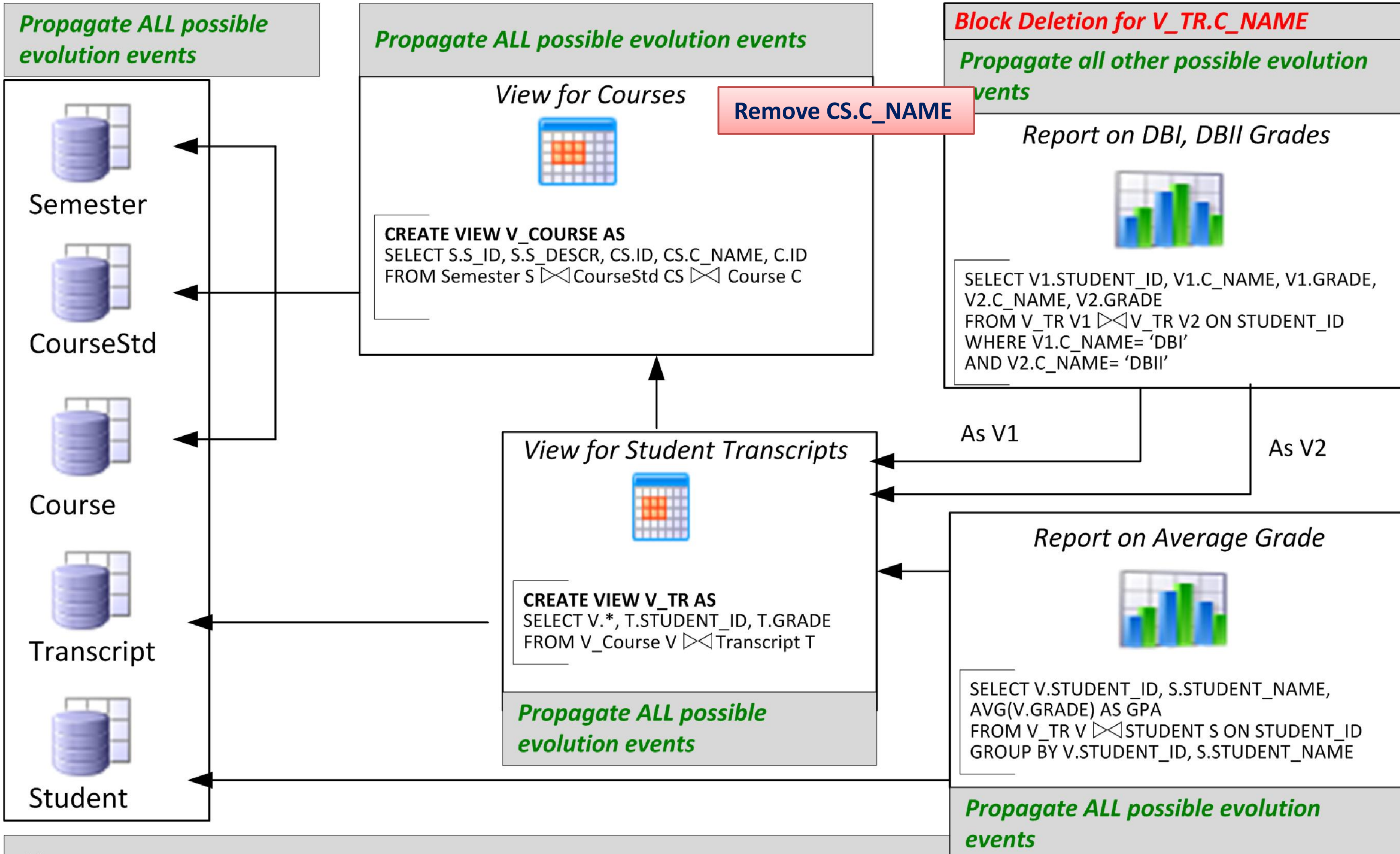
# Policies to predetermine reactions




```
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?



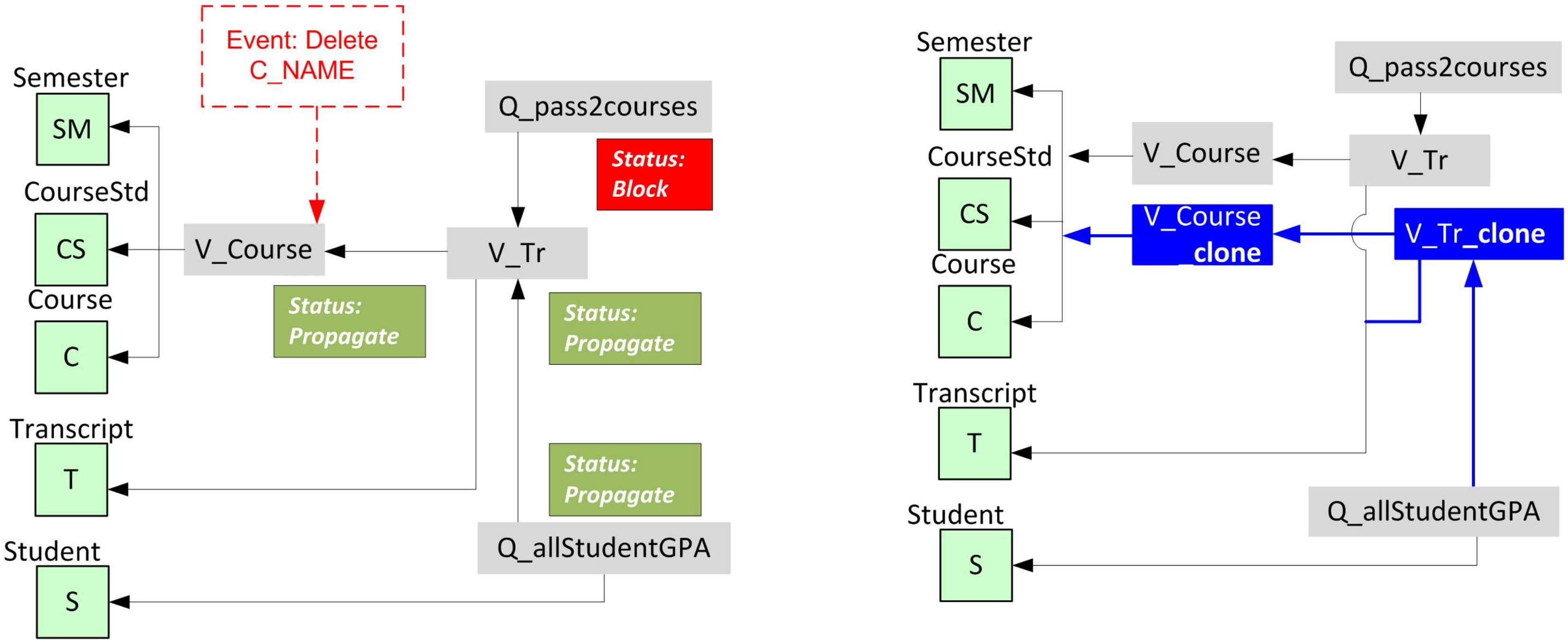

*//2 policy rules suffice to annotate the entire ecosystem:*

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

# Internals of impact assess. & rewriting

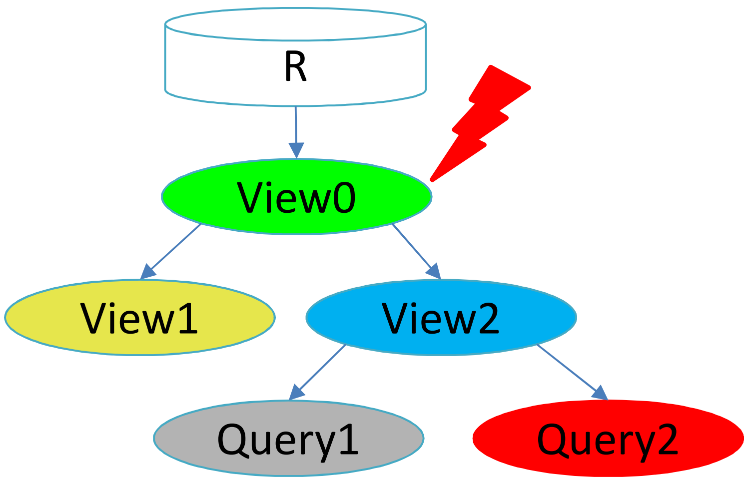
1. **Impact assessment**. Given a potential event, a status determination algorithm makes sure that the nodes of the ecosystem are assigned a status concerning (a) whether they are affected by the event or not and (b) what their reaction to the event is (block or propagate).
2. **Conflict resolution 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

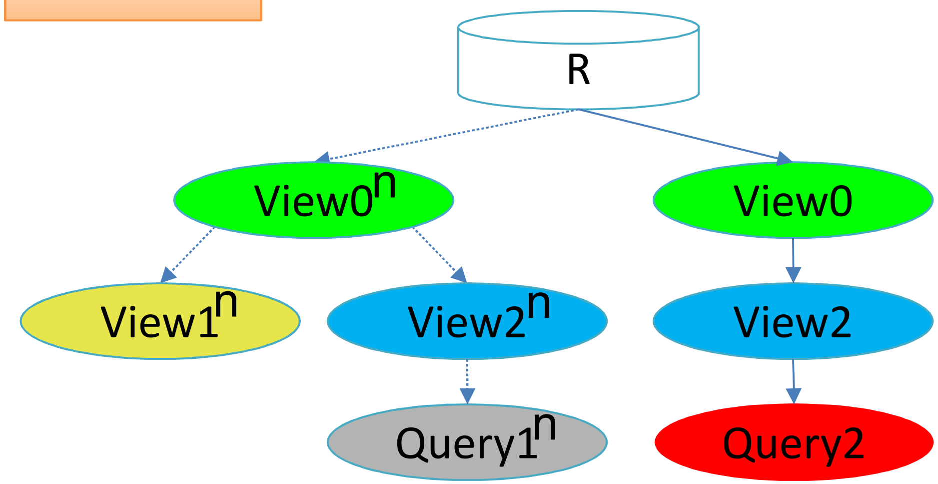


# Conflicts: what they are and how to handle them (more than flooding)

BEFORE




AFTER




- 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