Gravitating to Rigidity: Patterns of Schema Evolution -and its Absence- in the Lives of Tables

Panos Vassiliadis[a], Apostolos V. Zarras[a], Ioannis Skoulis[b,1]

[a] *University of Ioannina, Ioannina, Hellas*
[b] *Bouvet ASA, Oslo, Norway*

Abstract

Like all software maintenance, schema evolution is a process that can severely impact the lifecycle of a data-intensive software projects, as schema updates can drive depending applications crushing or delivering incorrect data to end users. In this paper, we study the schema evolution of eight databases that are part of larger open source projects, publicly available through open source repositories. In particular, the focus of our research was the understanding of which tables evolve and how. We report on our observations and patterns on how evolution related properties, like the possibility of deletion, or the amount of updates that a table undergoes, are related to observable table properties like the number of attributes or the time of birth of a table.

A study of the update profile of tables, indicates that they are mostly rigid (without any updates to their schema at all) or quiet (with few updates), especially in databases that are more mature and heavily updated. Deletions are significantly outnumbered by table insertions, leading to schema expansion. Delving deeper, we can highlight four patterns of schema evolution. The Γ *pattern* indicating that tables with large schemata tend to have long durations and avoid removal, the *Comet pattern* indicating that the tables with most updates are the ones with medium schema size, the *Inverse* Γ *pattern*, indicating that tables with medium or small durations produce amounts of updates lower than expected, and, the *Empty Triangle* pattern indicating that deletions involve mostly early born, quiet tables with short lives, whereas older tables are unlikely to be removed. Overall, we believe that the observed evidence strongly indicates that databases are rigidity-prone rather than evolution-prone. We call the phenomenon *gravitation to rigidity* and we attribute it to the implied impact to the surrounding code that a modification to the schema of a database has.

Email addresses: `pvassil@cs.uoi.gr` (Panos Vassiliadis), `zarras@cs.uoi.gr` (Apostolos V. Zarras), `giskou@gmail.com` (Ioannis Skoulis)

[1] Work conducted while in the Univ. of Ioannina

1. Introduction

Databases evolve over time, both in terms of their contents, and, most importantly, in terms of their schema. Schema evolution affects all the applications surrounding a database, as changes in the database schema can turn the surrounding code to be syntactically or semantically invalid, resulting in runtime crashes, missing or incorrect data. Therefore, the understanding of the mechanics of schema evolution and the extraction of patterns and commonalities that govern this process is of great importance, as we can prepare in time for future maintenance actions and reduce both effort and costs.

The emergence of free open-source software has greatly facilitated the research in the area of schema evolution. Prior to the availability of schema histories in open-source repositories (svn/ sourceforge / github), researchers where unable to work with real world-data to understand how schemata evolve. Remarkably, until the late 00's, there was just a single case study on the topic [1]. Lately, however, the research community has started to exploit the available information [2, 3, 4, 5, 6], as open source repositories provide us with access to the entire history of data-intensive project files, including the versions of the files with the database schema definition.

In this context, we embarked in the adventure of uncovering the internal mechanics of schema evolution, after having collected and analyzed a large number of database histories of open-source software projects. In [7], we have reported on our findings for the compatibility of database schema evolution with Lehman's laws and show that whereas the essence of Lehman's laws holds, the specific mechanics have important differences when it comes to schema evolution. In this paper, which extends [8], we depart from the traditional study of how schemata in their entirety evolve (e.g., how schema size grows, or how the heartbeat of changes unfolds) [2, 3, 4, 5, 6, 7], and come with a different contribution, that zooms into the details of the evolution of individual tables rather than entire relational database schemata. So, in this work, we explore *how evolution-related properties, like the possibility of deletion, life duration, or the amount of updates that a table undergoes, are related to observable table properties like the number of attributes or the version of birth of a table.* Our guiding research questions and their answers follow.

Which tables eventually survive and which ones are deleted? Our study has identified that there exist *"families"* of tables whose survival or removal is related to their characteristics.

- *Wide survivors.* The relation of schema size with duration revealed another interesting pattern, which we call *the* Γ *pattern*: "thin" tables, with small schema sizes, can have arbitrary durations, whereas "wide" tables, with larger schema sizes, have high chances of survival.

- *Entry level removals.* The vast majority of deleted tables are newly born, with few or no updates, quite often quickly removed and, also quite often, all three of them.

- *Old timers: time is on my side.* It is quite rare to see tables being removed at old age; although each data set comes with a few such cases, typically, the area of high duration is overwhelmingly inhabited by survivors!

The two last observations are combined in a pattern which we call the *Empty Triangle* pattern, because of the shape of its scatterplot: if we study the correlation of duration with birth, we see very few tables born after the original versions of the database that (a) do not survive and (b) have a fairly long duration. As most tables who are removed come with mostly short durations, the resulting shape of the scatterplot forms an empty triangle; thus, the name of the pattern.

Which tables are the ones that attract updates? Overall, it is safe to say that low-change tables dominate the landscape. In fact, when we studied the relation of life duration and amount of updates, we observed the *inverse* Γ *pattern* which states that updates are not proportional to longevity: with the exception of few long-lived, "active" tables, all other types of tables come with an amount of updates that is less than expected. These active tables with the larger amount of updates are long lived, frequently come from the early versions of the database and, unexpectedly, they are not necessarily the ones with the largest schema size, but typically start as medium sized. Concerning the last remark, our study of the correlation of schema size at the birth of a table with its update profile revealed a pattern which we call the *Comet pattern*, due to the vast majority of tables with small amount of change and narrow schema, as well as the existence of medium-size tables with many updates and wide tables with medium change.

Gravitation to rigidity. Several reasons work together to drive us to the conclusion that, despite the existence of clearly observable revisions of perfective maintenance, databases are demonstrating less evolution activity than "expected": tables are mostly quiet, deletions are outnumbered by additions, the tables who do not survive live short lives of low activity and frequently in the early stages of the database, old timers do not get deleted, wide tables with many attributes are scarcely removed, and few tables demonstrate "hot" update activity in "mature" databases. All these observations combined together bring out a tendency to avoid change and evolution, which we call *gravitation to rigidity*. We attribute the above phenomena to the *dependency magnet* nature of tables: the more dependent applications can be to their underlying tables, the less the chance of removal is. Large numbers of attributes, or large number of queries developed over time make both wide and old tables unattractive for removal.

Importance of this work. To the best of our knowledge, this is the first time that the profiling of the behavior of individual tables is performed, both at (a) a large scale, in terms of data sets, and, (b) in depth, in terms of the number and essence of the properties that we have studied. Our contributions can be summarized as follows. First, this effort contributes to increasing our knowledge on how tables evolve with specific patterns of change. To the best of our knowledge, this is the first time that, we get to see patterns on which tables survive, get removed, as well as how they change, based on solid evidence (rather than gut feeling or the general impression of a database expert). Equally important is our second contribution: after our study, we have data on the gravitation to rigidity, as now, we have solid evidence that tables do not change a lot. This brings up the danger of rigidity and places particular importance to all the techniques that attack the problem of smooth co-evolution of data and source code. Finally, we believe that in this paper, we also make a methodological contribution, as we show how it is possible to delve into the particularities of table evolution and extract patterns of change.

Roadmap. The rest of this paper is structured as follows. In Section 2 we present the experimental setup of our study. We present our findings in Sections 3 and 4. In Section 5, we discuss threats to validity. In Section 6 we discuss related work. In Section 7, we conclude with a discussion on the practical implications of our findings and open issues. All the material of this research, including links to software and data, can be retrieved via `http://www.cs.uoi.gr/~pvassil/publications/2015_ER/`.

2. Data Compilation and History Extraction

In this section, we present our experimental protocol, including the employed data sets, their collection and their processing, as well as the measures that we have computed for the life of each table in each of these data sets. We have presented the data sets and the collection protocol in detail in [9]; here we give a concise account of our actions and refer the interested reader to [9] for more details.

2.1. Data sets

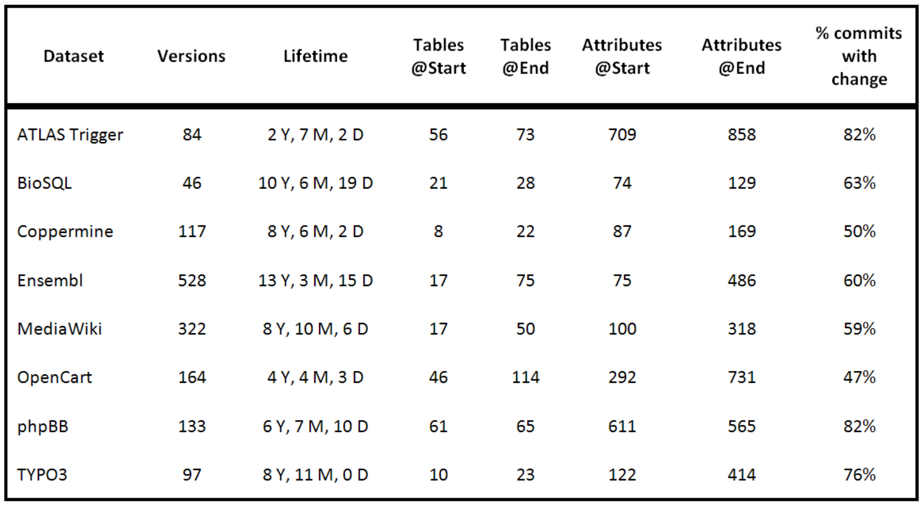
Our datasets include eight datasets that we collected, sanitized, and studied using our open source SQL diff tool, Hecate. Those datasets include Content Management Systems (CMS's), Web Stores, along with Medical and Scientific storages (see Fig. 1).

We have selected *relational schemata* to be studied for their evolution *at the logical level*. These schemata are (a) embedded in *open-source software*, and (b) have a significant amount of committed versions, along with a *fairly long history*.

The software projects hosting the respective databases are described in more detail in [9]. Also, all the datasets can be found in our group's git: `https://github.com/DAINTINESS-Group`.

2.2. Data preprocessing

For each data set, we gathered as many schema versions (DDL files) as we could from their public source code repositories (cvs, svn, git). We have targeted only changes at the database part of the project as they were integrated in the trunk of the project. The files were collected during June 2013. For all of the projects, we focused on their release for MySQL (except ATLAS Trigger,



Dataset	Versions	Lifetime	Tables @Start	Tables @End	Attributes @Start	Attributes @End	% commits with change
ATLAS Trigger	84	2 Y, 7 M, 2 D	56	73	709	858	82%
BioSQL	46	10 Y, 6 M, 19 D	21	28	74	129	63%
Coppermine	117	8 Y, 6 M, 2 D	8	22	87	169	50%
Ensembl	528	13 Y, 3 M, 15 D	17	75	75	486	60%
MediaWiki	322	8 Y, 10 M, 6 D	17	50	100	318	59%
OpenCart	164	4 Y, 4 M, 3 D	46	114	292	731	47%
phpBB	133	6 Y, 7 M, 10 D	61	65	611	565	82%
TYPO3	97	8 Y, 11 M, 0 D	10	23	122	414	76%

Figure 1: The datasets we have used in our study [7]

available only for Oracle). We collected all commits of the database at the time of the trunk or master branch, and ignored all other branches of the project, as well as any commits of other modules of the project that did not affect the database. For each committed version, we have assigned a sequential *version id* and also retained its Unix-time timestamp.

The files were then processed by our tool, Hecate, that allows the accurate detection of changes between subsequent versions of the schema. For each transition between subsequent versions of the schema, Hecate detects: (a) *changes at the table-level*, i.e., which tables were inserted and deleted and (b) *updates at the attribute-level*, and specifically, for each affected table, the attributes inserted, deleted, having a changed data type, or participation in a changed primary key. Hecate is capable of performing this computation for the entire history of collected versions; so, apart from a detailed transcript of the individual changes between subsequent versions, Hecate also reports the aggregate statistics per table for all the history of the database.

Hecate is also freely available as open source software via our group's git: `https://github.com/DAINTINESS-Group`.

2.3. Measures used

Given all these extracted data, we were able to detect, for each table (characterized by its *table name*) the following characteristics:

- *duration* (in number of versions)
- *birth* and *death* (as version id)

- the table's *schema size* in three variants: (i) at its *birth*, (ii) at the *end of its lifetime* (or the end of the data set, if the table survives until the last studied version), and (iii) as an *average schema size* during its lifetime
- the total amount of updates at the attribute level performed to the table – denoted as $sum(updates)$ (to be crystal clear: for each table, for each version of its lifetime, we added the measures obtained for the four aforementioned categories of attribute updates into a single measure)

On top of these fundamental measures, other measures can be defined. In particular, to classify the tables according to their update activity, we have defined the following measure:

$$Average\ Transitional\ amount\ of\ Updates(ATU) = \frac{sum(updates)}{duration}$$

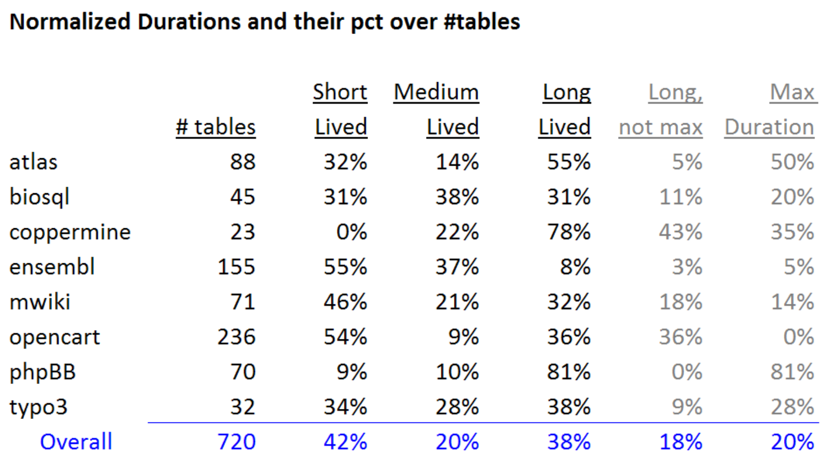
3. Table schema size, duration & updates

In this section, we study the relations between the tables' schema size, duration and updates. The rationale for selecting these properties is quite simple, as we intended to use any information that we could automatically extract, to relate properties that one could know at a given time of the life of a database, with properties that could evolve over time. For example, at the time of birth of a new table, the DBA knows the schema size. Is it possible to make a calculated guess on the possible amount of updates that one could expect the table to sustain as well as its possible duration, given its number of attributes at this point? Of course, one could select several other properties to extract and study. We have chosen the most promising ones and, as the following subsections will show, our choice was rewarding, as there are indeed patterns that one can observe based on these properties. In the rest of this section, we will devote a subsection for the study of each pair of the three measures (schema size, duration and updates). Before proceeding to this study, however, we will present some findings on fundamental statistics for durations and schema sizes (the statistics on the statistical profile of the updates will be covered in the following section).

3.1. Fundamental statistics for table durations and schema size

We have analyzed the lifetime duration of all the tables in all the studied datasets and we have observed that there is an interesting relation of durability with table schema size. We have computed the durations (in number of versions) for each table in each dataset. Then, again for each table, we produced a normalized measure of duration, by dividing the duration of the table by the maximum table duration of its database [2]. This results in all tables having a

[2]with the exception of OpenCart, this number coincides with the number of versions that we have monitored



Normalized Durations and their pct over #tables

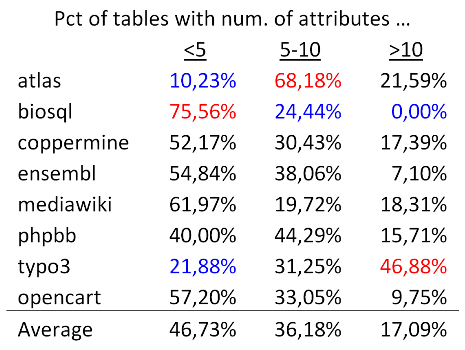
		Short	Medium	Long	Long,	Max
	# tables	Lived	Lived	Lived	not max	Duration
atlas	88	32%	14%	55%	5%	50%
biosql	45	31%	38%	31%	11%	20%
coppermine	23	0%	22%	78%	43%	35%
ensembl	155	55%	37%	8%	3%	5%
mwiki	71	46%	21%	32%	18%	14%
opencart	236	54%	9%	36%	36%	0%
phpBB	70	9%	10%	81%	0%	81%
typo3	32	34%	28%	38%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

Figure 2: Distribution of tables with respect to their normalized duration

normalized duration in the range of (0 . . . 1]. Then, in an attempt to simplify intuition, we decided to classify tables in three categories: (i) short-lived (including both the ones who were removed from the system at some point and the ones whose short duration is due to their late appearance), (ii) tables of medium duration and (iii) long lived tables. To avoid manually specifying the limits for each of these categories, we used a k-means clustering [10] that, based on the normalized duration of the tables, split the data set in three clusters, and specifically at the values 0.33 and 0.77. The details of the breakdown appear in Fig. 2, where we devote one column per category, along with the breakdown of the long-lived category in two subclasses: (a) tables that live long, but not throughout the entire lifetime of the database and (b) tables that live from the very first till the last version that we have monitored (and thus, have a normalized duration equal to 1.0).

A large percentage of tables lives short lives, in all datasets: with the exception of two data sets, in all other cases, more than 30% live short lives. A quite large percentage of tables also lives long lives: in half the data sets the percentage of these tables lies within 30% - 40%, and in three others, long lived tables exceed 50%. Interestingly, the number of long-lived tables that reach maximum lifespan is more frequently than not higher than the number of long-lived who do not. *On average, this percentage reaches 20% of the total number of tables studied!*

Fig. 3 gives the statistical distribution of the average table schema size (i.e., the average number of attributes a table has throughout its lifetime). Typically, the average table schema size has a very small deviation, as tables do not change largely. Thus, the average size is pretty close to the respective max and min size for the studied tables. *On average, half of the tables (approx. 47%) are small tables with less than 5 attributes. The tables with 5 to 10 attributes are approximately one third of the tables' population and the wide tables with more*



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%

Figure 3: Distribution of tables with respect to their average schema size

than 10 attributes are approximately 17% of the tables. The first category has quite a few outliers, both high and low (and a large standard deviation of 20%), whereas the two others do not really oscillate too much (with standard deviations of 14 and 13%, respectively). Interestingly, the datasets with less evolutionary activity (atlas, typo3 and biosql) are the ones concentrating outlier values.

3.2. Relationship of the schema size at birth with the duration of a table

The first pair of measures that we have studied is the combination of schema size at birth with table duration.

To illustrate the correlation of schema size and duration, we start with its visual representation. Specifically, the first column of Fig. 4 gives the relation of a table's schema size (measured as the number of attributes of a table's schema at its birth and depicted in the x-axis of each of the scatterplots) with duration (y-axis in the scatterplots), in three characteristic data sets, Atlas, Coppermine and Mediawiki. We depict the graphical representation of this relationship, again as a scatterplot, in Figure 5, for all the eight datasets that we have studied (observe that we enrich the representation with two combined dimensions, specifically, death-or-survival and update activity).

We observe a phenomenon that we call Γ *pattern*: *tables with small schema sizes can have arbitrary durations, whereas tables with larger schema sizes last long.*

3.2.1. "If you're wide, you survive"

Plainly put, the facts that the first column of Figure 4 and Figure 5 vividly demonstrate are as follows. There is a large majority of tables, in all datasets, whose size is between 0-10 attributes. We have visualized data points with transparency in the figure; therefore, areas of intense color mean that they are overpopulated with data points that fall on the same x,y coordinates. Many of the data sets are limited to really small schema sizes: Biosql tables do not exceed 10 attributes, Ensembl tables do not exceed 20 attributes and Atlas, Mediawiki and Coppermine have one or two tables "wider" than 20 attributes

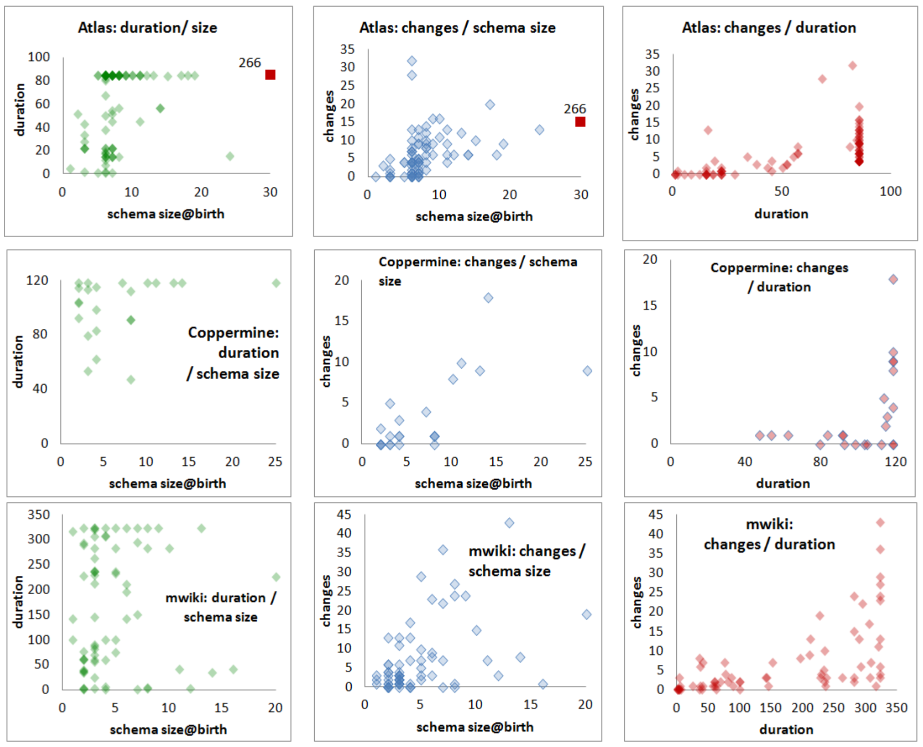


Figure 4: Correlation of schema size (number of attributes), duration (over versions) and change (total number of changes) of all the tables in three datasets

(btw., observe the outlier table of 266 attributes in Atlas, too). Due to their large percentage, narrow tables dominate the area near the vertical axis in the figures. What is interesting however, is that *their small schema size does not determine their duration*: this is depicted as a distribution of data points in parallel with the y-axis for all the small sizes of a table's schema. On the other hand, we observe that *whenever a table exceeds the critical value of 10 attributes in its schema, its chances of surviving are high. As a side observation, we observe that very often, the majority of "wide" tables are created early on and are not deleted afterwards.*

3.2.2. Statistical evidence for the Γ pattern

To test the Γ pattern, we need to check the probability that a table with wide schema survives. In what follows, first, we describe our terminology and statistical profile of the data sets and then we assess the probability that wide tables survive. We also offer a list of deviations from the pattern as well as some conjectures on our observations.

Terminology. We consider a schema to be wide, when it is strictly above 10 attributes. The top band of durations (the upper part of the Γ shape) is

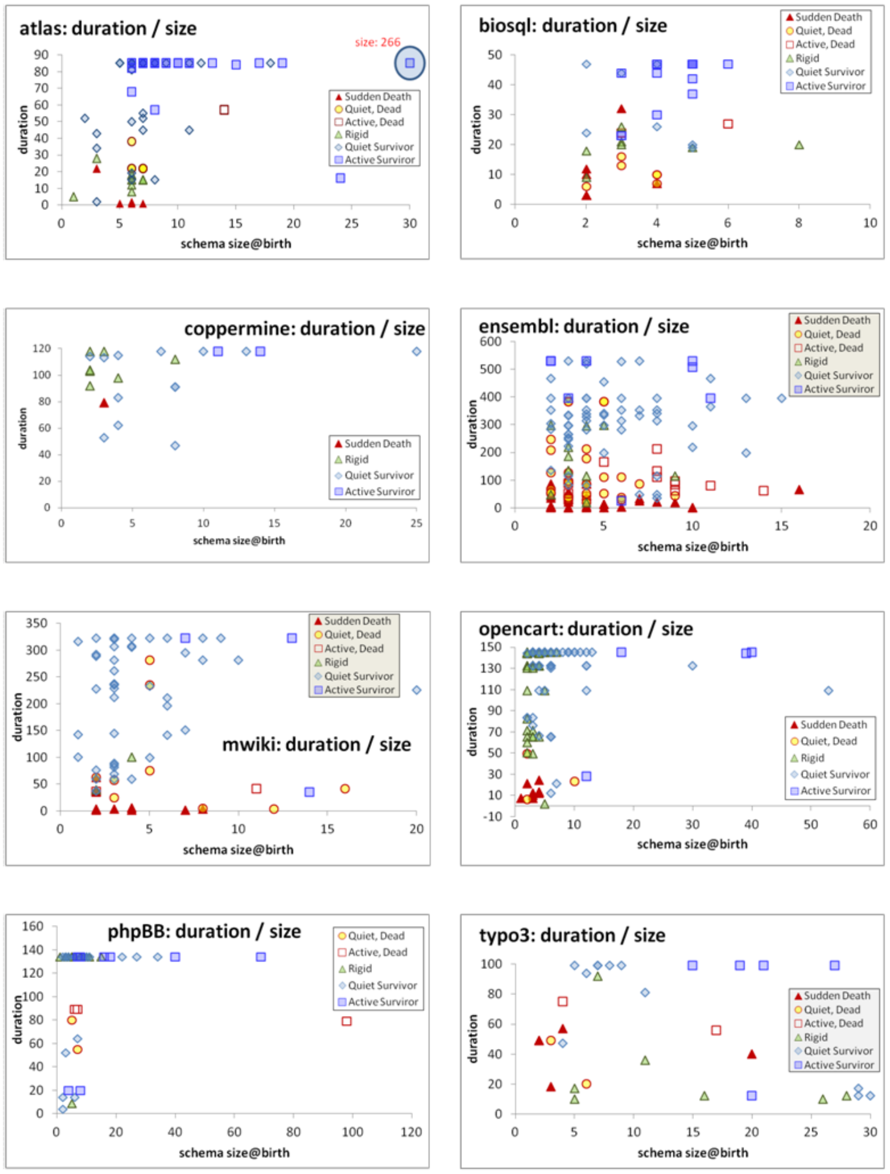


Figure 5: The Γ pattern in all data sets

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

Figure 6: Statistics on wide tables and their survival

determined as the range of values that is above 90% of the maximum duration (i.e., the upper 10% of the values in the y-axis). We consider a table to have been born early if its birth version is in the lowest 33% of versions; respectively, late-comers are born after the 77% of the number of versions.

All data sets include wide tables, with percentages that typically range between 6% and 17%, with two exceptions: (i) Biosql without any wide tables at all (and therefore, excluded from the rest of this discussion), and, (ii) typo3 with a 47% of wide tables in its schema.

Research Question: What is the probability that whenever a table is wide, its chances of surviving are high? The percentage of tables with a wide schema that have survived is presented in the fifth column of Figure 6 (overwhelmingly in green font color). Apart from the two data sets with the largest history, Ensembl and Mediawiki, 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.

Exceptions. There are deviations (not exactly exceptions) to the Γ pattern. Specifically, we can count Biosql out of consideration as no table exceeds 10 attributes. The two datasets that are the longest in duration, Ensembl and Mediawiki also have very few tables exceeding the limit 10 attributes; moreover, in each of these cases, 3 of the wide tables died. The Typo3 data set is more of an exception as it comes with several late born survivors, and thus it reminds quite a lot a C shape. Overall, although we consider the issue of exceptions important, we believe that (a) the discrepancies are not major and (b) the basic pillar of the pattern that is summarized as ”if you ’re wide, you survive” is strongly supported by our experimental evidence.

Conjectures. We conjecture that the explanation for the ”if you ’re wide, you survive” part of the Γ pattern is due to the impact a table deletion has: the wider a table, the higher a chance it acts as a fact table, frequently accessed from the queries in the applications surrounding the database. Thus, its removal incurs high maintenance costs, making application developers and administrators rather reluctant to remove it.

3.2.3. Accompanying phenomena and observations

Apart from the essence of the Γ pattern, our data collection allows us to put more research questions. Specifically, we have assessed the following research questions:

Research Question: What is the probability that wide tables are frequently created early on and are not deleted afterwards? A typically large subset of the wide-and-survivor tables has also been created early. The sixth column of Figure 6 (labeled "Early Born and Survivors") demonstrates the percentage of early born, wide, survivor tables over the set of wide tables. Impressively, we see that in half the data sets the percentage is above 70% and in two of them the percentage of these tables is one third of the wide tables.

Research Question: What is the probability that whenever a table is wide, its duration frequently lies within the top-band of durations (upper part of Γ*)*? Finally, we also study the probability that a wide table belongs to the upper part of the Γ. This is depicted in the last column of Figure 6. We observe that there is a very strong correlation between the two last columns of the figure; in fact, the Pearson correlation is 88% overall, and 100% for the datasets with high percentage of early born wide tables (appearing in blue font color in the figure). *Overall, the data are clearly bipolar on this pattern: half the cases answer the research question positively, with support higher than 70%, whereas the rest of the cases clearly disprove it, with very low support values.*

In all data sets, *if a wide table has a long duration within the upper part of the* Γ*, this deterministically signifies that the table was also early born and survivor.* All three properties coexist in 100% of all data sets. In other words, there is a subset relationship and all tables fulfilling the criteria for the last column of Figure 6 also fulfill the criteria for the penultimate one. Plainly:

If a wide table is in the top of the Γ *line, it is typically (deterministically in our data sets) an early born survivor.*

3.3. Relationship of the schema size at birth with the number of updates of a table

In this subsection, we study the correlation of schema size at birth and amount of updates for the tables of the studied data sets.

As in the previous subsection, we start with the discussion of the scatterplots that visually represent this correlation. Remember that we use one scatterplot per data set, with each point of the scatterplot referring to a table of the respective schema. Data points are visualized with transparency to allows us to see the population's density. If we observe the scatterplots at the middle column of Figure 4, as well as Figure 7, we can see the relation of a table's schema size at its birth (depicted in the x-axis of each of the scatterplots) with the amount of change the table undergoes (y-axis in the scatterplots). There are two main clusters in the plots: (*a) a large, dense cluster close to the beginning of the axes, denoting small size and small amount of change, and, (b) a sparse set of outliers, broken in two subcategories: (b1) medium schema size tables typically demonstrating medium to large amounts of changes and (b2) "wide" tables with large schema sizes demonstrating small to medium amount of change.*

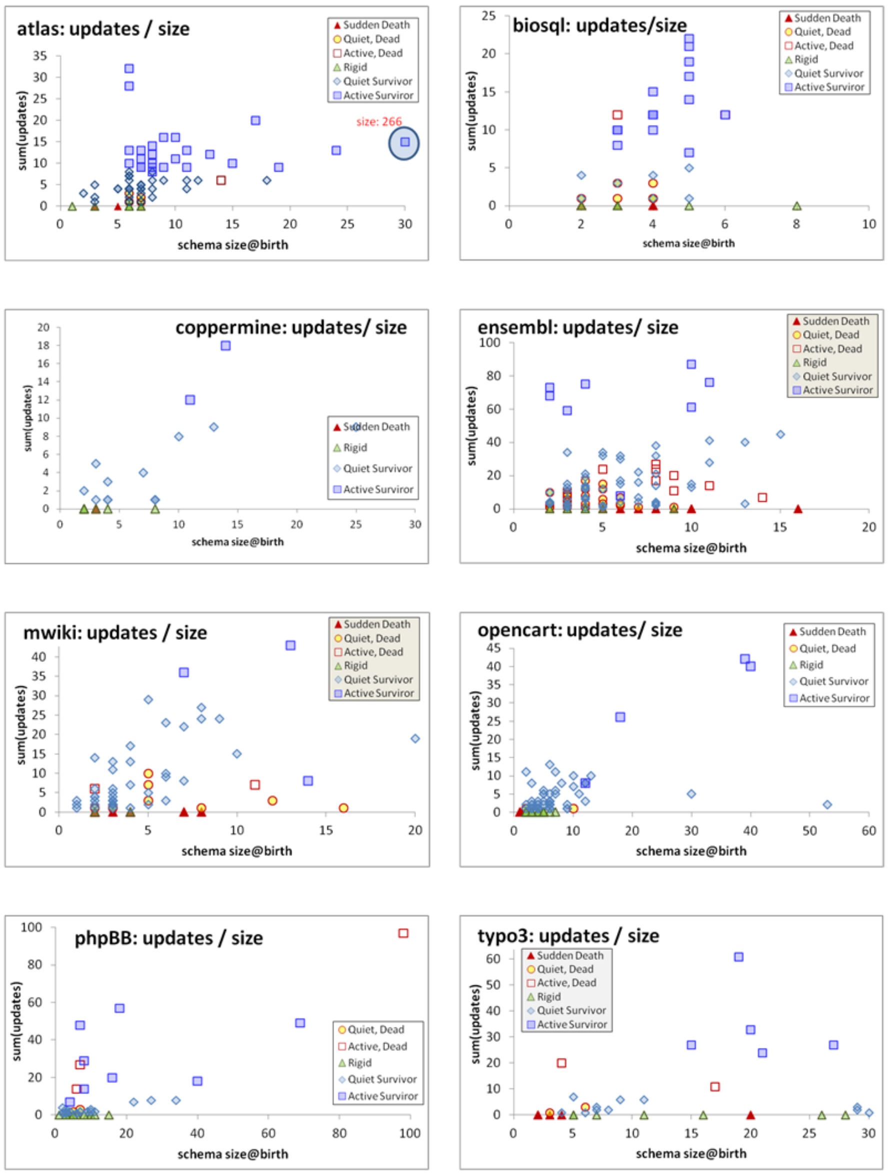
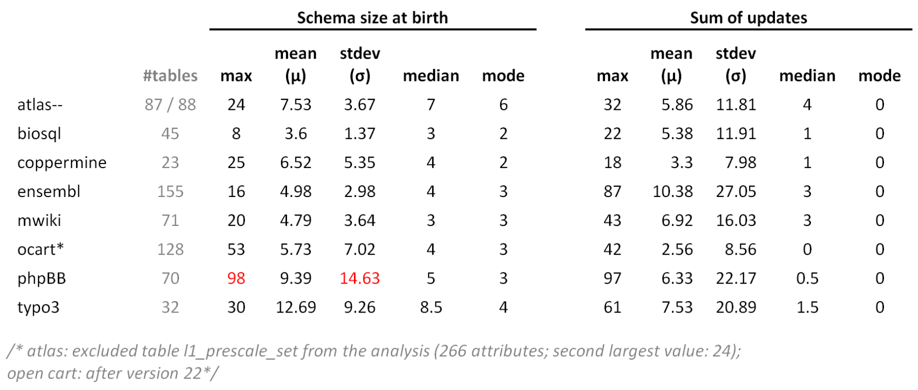


Figure 7: The Comet pattern in all data sets



		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*/*

Figure 8: Statistics of schema size at birth and sum of updates for the studied data sets

We refer to this distribution of a core group near the beginning of the axes along with two tails outside it as the *comet pattern*. We have chosen this term as the real-world comets are composed of (a) a central part, called *nucleus*, (b) a surrounding cloud of particles around the nucleus, called *coma*, and two tails: (c) a white dust tail and (d) a blue gas tail, of electrically charged gas (see `http://spaceplace.nasa.gov/comet-nucleus/en/` for more).

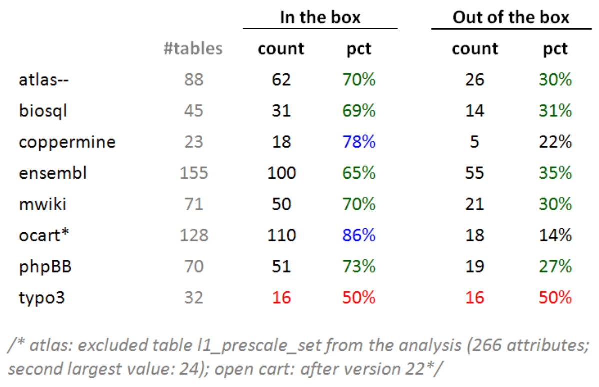
3.3.1. The comet's nucleus and tails

The first "nucleus" cluster is typically contained within a box of size 10×10 (i.e., no more than 10 attributes typically result in no more than 10 changes). This is attributed to the small pace of change that tables undergo (cf. Sec. 4), resulting in small probabilities of attribute updates for narrow tables. At the same time, in most of the datasets, *the tables with the largest amount of change are not necessarily the largest ones in terms of attributes, but tables whose schema is on average one standard deviation above the mean.* Typically, medium sized tables demonstrate all kinds of change behaviour as they cover the entire y-axis, whereas the (few) tables with large schema size demonstrate observable change activity (i.e., not zero or small), which is found little below the middle of the y-axis of the plot in many cases.

3.3.2. Statistical evidence for the Comet pattern

Before assessing the Comet pattern, it is worth presenting some basic statistics on schema size and amount of updates that we have measured for all our data sets (see Figure 8). We have measured the fundamental statistical properties (mean, max, standard deviation, median and mode) for both the schema size at birth and the sum of updates for all data sets (outliers, as described in the figure, have been removed to avoid distorting the statistical measures with extremities).

To assess the comet pattern we have statistically assessed three measurable quantities. Specifically, we have measured (a) the percentage of the population



	#tables	In the box count	In the box pct	Out of the box count	Out of the box 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*/*

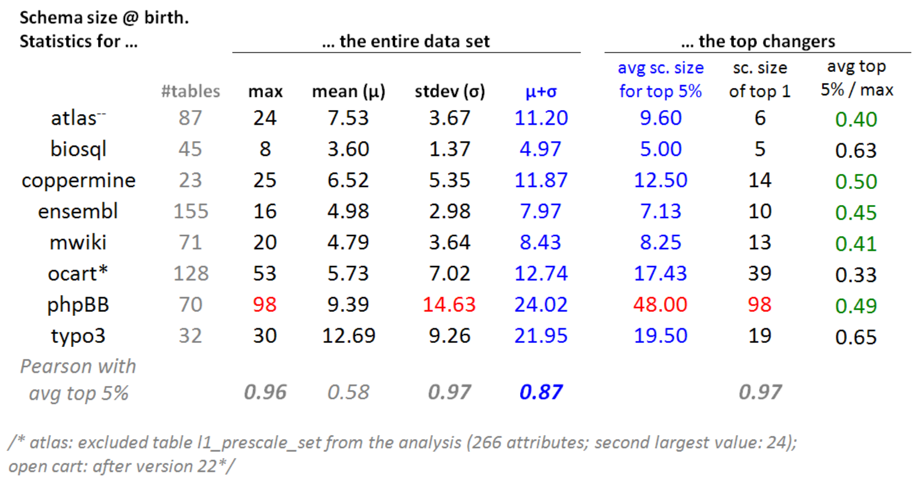
Figure 9: Comet pattern: the percentage of the population that lies within the 10×10 box

that lies within the 10×10 box (at most 10 attributes in schema and 10 updates in their lifetime), (b) the schema size at birth of the tables that eventually demonstrate a large number of updates in their lifetime and (c) the update behavior of tables with large schema sizes.

Research question: What percentage of the population comes with narrow schema size and quiet update behavior? (equivalently: what happens at the nucleus of the comet?) We present the data answering this question in Figure 9. Typically, around 70% of the tables of a database is found within the 10×10 box of schema size at birth × sum of updates (value 10 is not counted as part of the box for neither dimension). In two cases, the percentage rose above 75% and in a single case (Typo3) the population of tables were split in half. Overall, our statistical assessment says that *around 70% of the tables of a database are typically found inside the narrow-and-quiet box.*

Research question: What tends to be the schema size of tables with large amounts of updates? (equivalently: can we verify that the tail of the comet concerning top amounts of updates concerns medium sized schemata?) As already mentioned, we have computed the sum of updates performed to each table of each data set. Here, for every dataset, we selected the top 5% of tables in terms of this sum of updates. We averaged the schema size at birth of these top 5% tables. The resulting numbers are depicted in the Figure 10. With the exception of phpBB, which is quite biased in its statistics, in all the other data sets, 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. It is also noteworthy that in 5 out of 8 cases, the average schema of the top tables in terms of sum of updates is within 0.4 and 0.5 of the maximum value (on average, it reaches 48% of the maximum value – practically, the middle of the domain) and never above 0.65 of it. Overall, based on this evidence, we can conclude that *tables with large numbers of updates tend to typically have medium schema sizes.*

A second noteworthy observation is shown in the bottom line of the table, in



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*/*

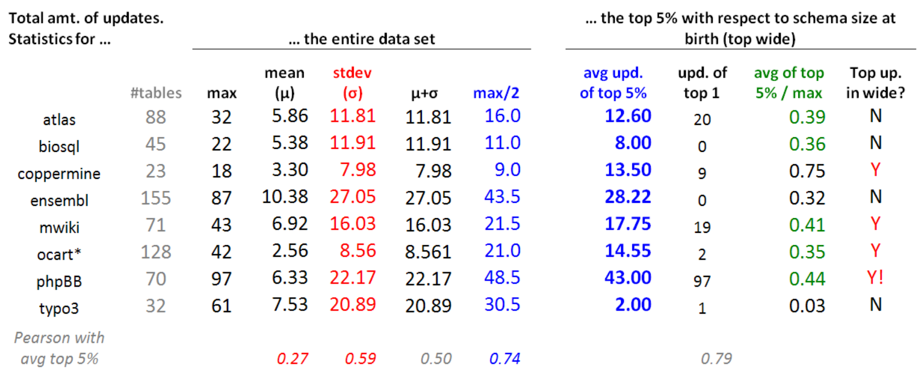
Figure 10: Comet pattern: Tables with large amounts of updates tend to have medium schema sizes

italic grey font, where we have computed the Pearson correlation of each column with the average schema size of the top changers: observe how extremely strong is the correlation of the maximum value, the standard deviation of the entire data set and the average of the top changers.

Research question: What tends to be the update behaviour of tables with wide schema size at birth? (equivalently: can we verify that the tail of the comet concerning wide schemata concerns medium amounts of updates?) We have also studied the update behavior of the tables with the largest schemata. For each data set, we have taken the top 5% in terms of schema size at birth and observed the relationship of their update behavior contrasted to the update behavior of the rest of the data set. The resulting numbers are depicted in Figure 11.

As in the previous examination, we isolated the top 5% of tables with respect to their schema size at birth (which we also call *top-wide* tables) and averaged the amount of their updates. The penultimate column in Figure 11 depicts the detailed numbers. The average value for this metric is close to the 38% of the domain of values for the sum of updates (i.e., slightly below the middle of the y-axis of Figure 7, measuring the sum of updates for each table). The average value over all data sets is 38% – dropping to 37% if we exclude the two extreme outlier cases. In 5 out of 8 cases, the average amount of updates for the top-wide tables is within 35% - 45% of the maximum value for the amount of updates. Moreover, in 4 out of 8 of the data sets, the table with the largest amount of updates was included in the set of top-wide tables. In one case (phpBB), the table was top 1 with respect to both categories.

On the left hand side of Figure 11, observe that the total amount of updates per table comes with a (very) large standard deviation. The mean value of updates is very low in all cases, due to the large percentage of the population



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*		

Figure 11: Comet pattern: Tables with wide schemata tend to have medium amounts of updates

leading quiet lives. The standard deviation, at the same time, is practically twice the mean value, practically meaning quite different statistical profiles for the studied tables.

3.4. Relationship of the duration with the number of updates of a table

Finally, we have studied the correlation of duration and amount of updates for the tables of the studied schemata.

As in the previous cases, we first discuss the visual properties of the respective scatterplots, and then move on to present the statistical evidence supporting the reported pattern. The last column of Figure 4, as well as Figure 12, demonstrate the relation of a table's duration (depicted in the x-axis of each of the scatterplots) with the amount of change the table undergoes (y-axis in the scatterplots). We observe a phenomenon that we call the *Inverse* Γ *pattern*: *tables with small duration undergo small change, tables with medium duration undergo small or medium change, and, long-lived tables demonstrate all kinds of change behavior.*

3.4.1. Inverse Γ*: a first glimpse of gravitation to rigidity*

The vast majority of tables have "calm" lives, without too much change activity; therefore, there is a high chance that a table will be low in the y-axis of the plot. In detail, all the scatterplots present three clusters: The absence of extremities means that a short lifetime cannot really produce anything but small (typically close to zero) change; medium duration has some higher chance of producing change in the range of 5 to 10 changes; and high amounts of change are only found in tables with long lives. Still, instead of a full triangle, the chart demonstrates mainly an inverse Γ: there is a striking scarcity of tables with mid-sized durations and mid-range change that would be part of the diagonal, or fill the interior of the triangle (visual clue: color intensity in our charts is an indicator of population density, due to overlapping semi-transparent points).

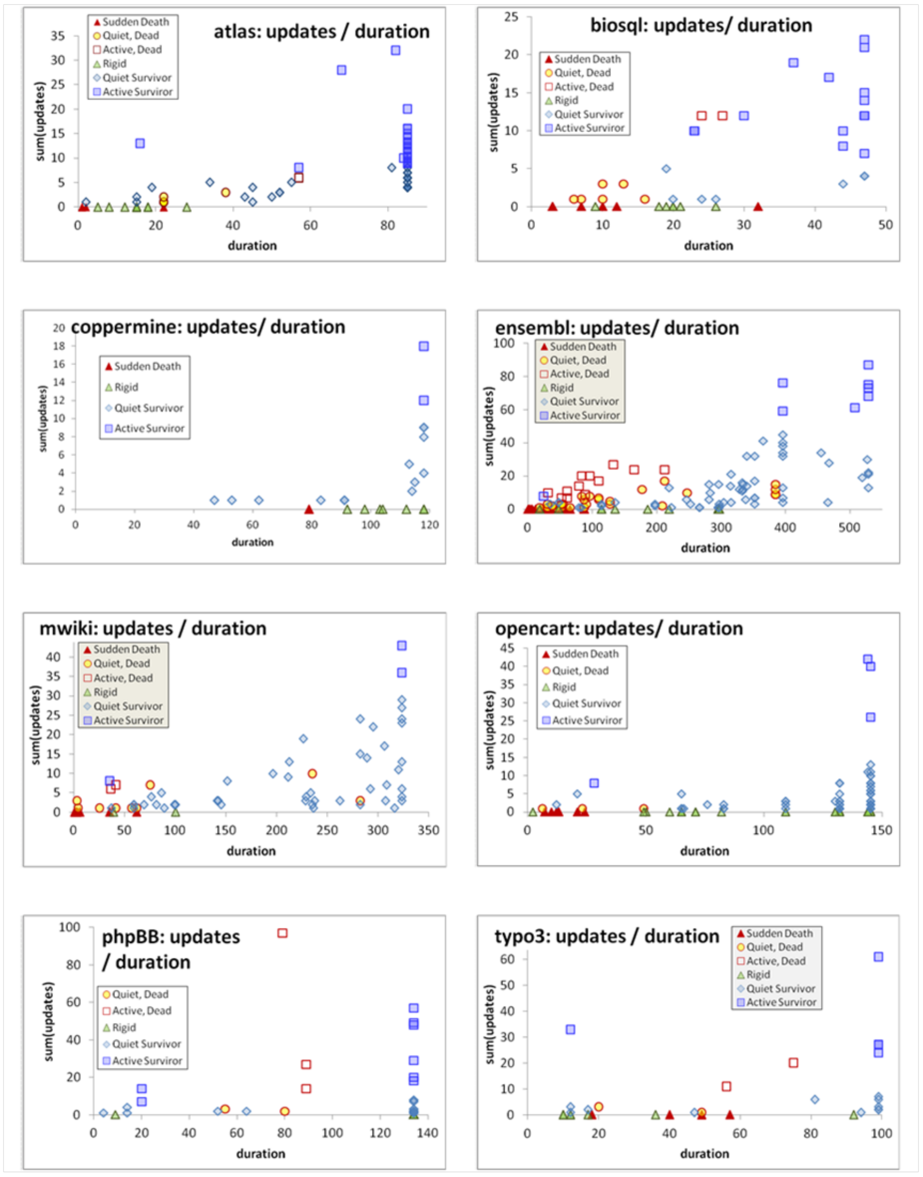


Figure 12: The Inverse Γ pattern in all data sets

The majority of tables having quiet, calm lives, pushes the triangle to become an inverse Γ.

3.4.2. Statistical evidence for the Inverse Γ pattern

The Inverse Γ pattern comes with a simple observation: small and medium durations come with small amounts of change whereas large duration exhibit all kinds of change. Although the scatterplots show this pattern vividly, we have also performed a small transformation of data to simplify the representation of the pattern. To this end, we have divided the duration axis into ranges ("buckets"), in sizes of 10, and counted the average value of each bucket (as a representative) as well as the maximum value, as an indicator of the skyline of values. We graphically depict these two measures for all data sets in Figure 13. Apparently, the best possible indicator for explaining the Inverse Γ is the skyline of values, expressed by the maximum value per bucket (i.e., the value of the table with the maximum number of updates within the bucket). This skyline gives the outline of the upper part of the scatterplot. The skyline clearly shows that the maximum amount of updates is not linearly related with duration, but rather that almost every bucket does not exceed a low height of updates, with the exception, of course, of high durations. When we observe the relationship of max and average amount of updates we can see that they go hand-in-hand in almost all occasions, in all the buckets, again with a single exception to this rule, which is -almost always- the bucket with highest durations. In this case, the maximum value significantly deviates from the average. Overall, we can argue that -with a few exceptions- *we see large amounts of updates only in large durations, whereas, typically, the average amount of updates is small.*

Exceptions. There are exceptions to the Inverse Γ pattern. Biosql shows an empty triangle pattern; however it comes with a rather small number of 45 revisions, along with large numbers of (green) rigid survivors found at the bottom of the triangle. Ensembl, a data set with a high amount of deletions in its profile, comes with two peaks instead of one, at high durations (although this does not really disprove the essence of the patter, but only the geometrical expression that we have used to express it). Ensembl also has a very large number of deleted tables of medium duration that demonstrate medium amounts of activity. Mediawiki has a few survivor tables, born at later stages of the database (and thus, of medium duration), that escape the "gravitation towards rigidity" pattern and demonstrate medium amounts of updates. The rest of the data sets seem to abide by the Inverse Γ pattern with almost single outlier points. A noteworthy exception is phpBB that demonstrates its maximum activity in the middle of the x-axis: this is a single outlier table that eventually died and exceeded all others by an order of magnitude for the large majority and a scale factor of 1.5 with respect to the second highest. Atlas and Typo3 have also a single outlier late-comer survivor with high activity profile.

Conjectures about the gravitation to rigidity. We believe that the Inverse Γ pattern is a quite vivid demonstration of the *gravitation towards rigidity* tendency in the lives of tables. Because of the shape of our diagrammatic representation, where we put the amount of change in the y-axis, we observe a tendency

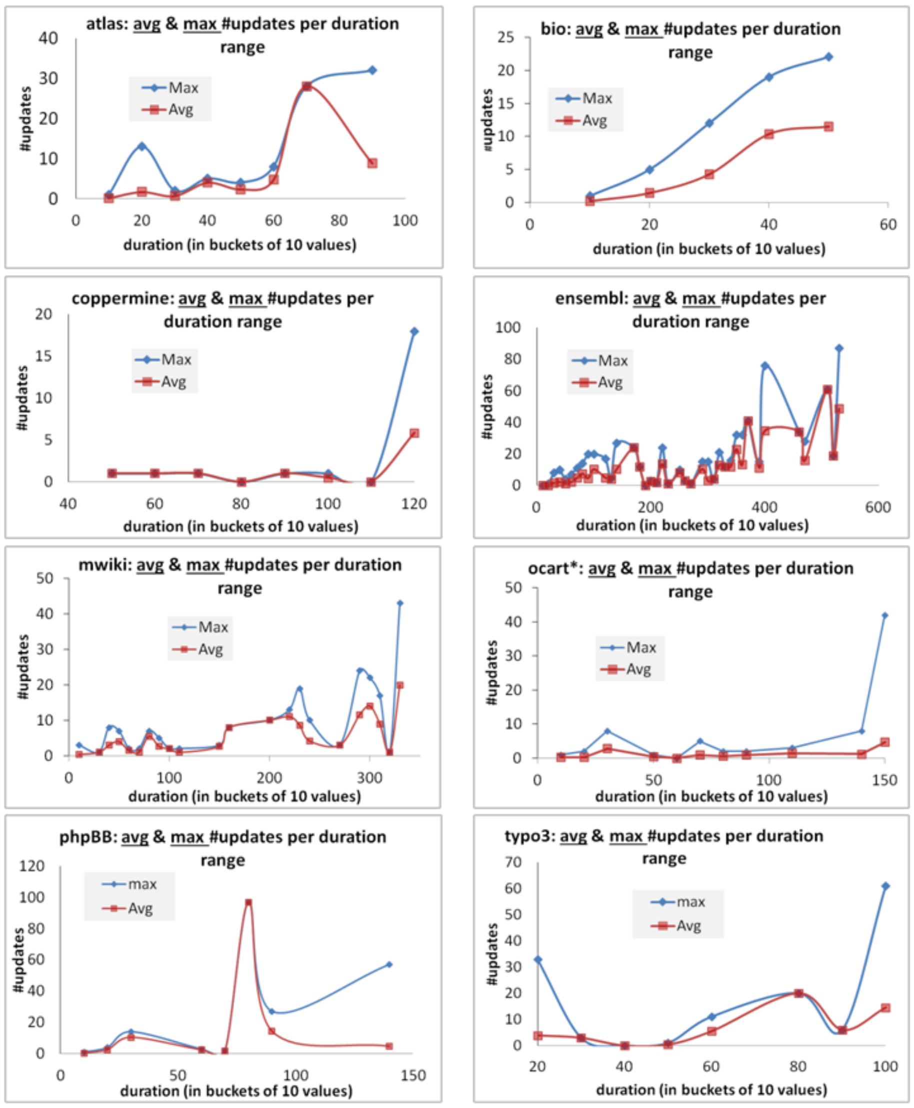


Figure 13: Skyline and average lines for the Inverse Γ pattern

of the tables to be "attracted"/"gravitated" to the lowest possible values, close to the horizontal axis. Although this is a just metaphor, it appears as if there is a gravitation force pulling them towards zero change, with very few of them escaping the force. However, there are several statements that can be made here:

- We do not observe a triangle shape in our scatterplots. An empty triangle would mean that duration is linearly related with the amount of updates (we see an empty triangle in Biosql, in Figure 12). A filled triangle would signify that tables live lives of reasonable update behavior: a filled pattern means there are variations in the update behavior of the tables, related to other factors rather than duration, along with the simple statistical rule that the more you live, the higher the probability to change (which would result to the diagonal of the triangle acting as a simple statistical limit of how high change can go). We see a pale attempt for a filled triangle in Mediawiki. We do not observe neither of these patterns at large, but on the contrary, we mostly observe a gravitation to rigidity for the medium and low durations.
- There is an interesting counter argument to be made here. In our scatterplots, we observe that long durations demonstrate both (a) a high number of tables and (b) a spread of the points throughout all the height of the vertical axis. In other words: is it just randomness? Is it possible that the amount of change is due to other factors and this is why, the more tables a point in the x-axis has, the more spread it demonstrates? Clearly, one could consider suggesting such an argument for the high durations. But at the same time, why then, it does not hold for the other durations too? Even if the amount of tables with medium durations is small, why then they do not typically create an empty triangle, or a vertical line of points in the scatterplot, like the one that takes place at the highest durations? Why is it that even the high durations have a high concentration of values at the low values of the y-axis, thus resulting in a significant deviation of the average from the maximum value in Figure 13?
- We believe that, overall, there is indeed a tendency to avoid change in the schema of a table – which we call "gravitation to rigidity". We conjecture that the gravitation to rigidity is due to the reluctance of developers to interfere with the structure of the tables. Again, we attribute this tendency to the dependency magnet nature of the database: changes in the database schema affect the surrounding code, multiplying thus the maintenance impact and cost.

4. Matters of life and death

Having examined how the schema size, duration and total amount of update of a table are related, in this section, we investigate how the duration and update activity of tables are related to their birth and death. We will devote a subsection to our findings concerning top-changers and another subsection to tables that are removed. As we will see, in both cases, update behavior and survival seem to be strongly related to properties like birth and duration. Before this, however, we will start our account of findings with the study of the distribution of tables to classes formed by the combination of the possible profiles of (a) update activity and (b) survival or death of a table.

4.1. Statistical profile on the correlation of update activity with survival

In this subsection we study how tables are statistically distributed in classes formed by combining their update behavior and their survival or removal. The question is important as it allows us to put all subsequent findings in context.

To classify tables to profiles of update activity we decided to move one step further than using the sum of updates that a table has undergone. This is due to the fact that it is theoretically possible that a table with a short life might not have the possibility of obtaining a large sum of updates, although his change activity is intense within this short life span. So, we define the *Average Transitional amount of Update (ATU)* as the fraction of the sum of updates that a table undergoes throughout its life over its duration. Equivalently, one can think of this measure as *the average number of schema updates of a table per transition.* Thus, the average transitional update of a table practically measures how active or rigid its life has been by normalizing the total volume of updates over its duration.

To form the space of *update profile* $\times$ *survival profile*, we need to define the classes for each dimension.

To address the *survival vs death* separation task, we will employ the following terminology:

- *survivor* is a table that was present in the last version of the database that we examine, and,
- *non-survivor* refers to a table that was eventually eliminated from the database.

We also discriminate the tables with respect to their update profile as follows:

- *Active tables* or *top-changers* are the ones with (a) a high average transitional update, exceeding the empirically set threshold of 0.1, and, (b) a non-trivial volume of updates, exceeding the also empirically set limit of 5 updates.
- *Rigid tables* are the ones with no change at all; we will also refer to the ones that did not survive as *sudden deaths*, as they were removed without any previous change to their schema.
- *Quiet tables* are the rest of the tables, with very few updates or average transitional update less than 0.1. For the tests we made, discriminating further within this category did not provide any further insights.

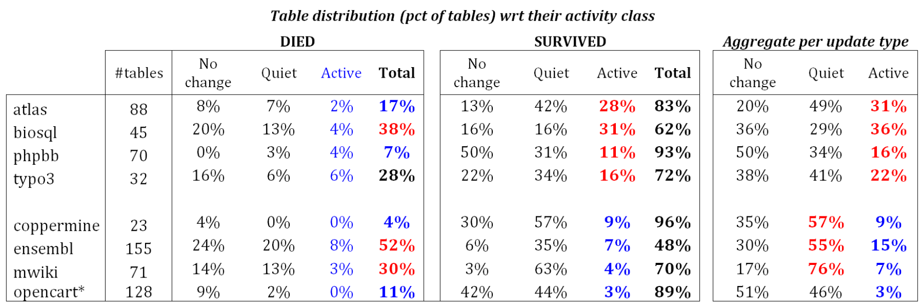


Table distribution (pct of tables) wrt their activity class

		DIED				SURVIVED				Aggregate per update type		
	#tables	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%**

Figure 14: Table classification concerning the average amount of updates per version for all data sets (∗Opencart reports only on the tables born after transition 17)

Then, the space produced by *update profile* × *survival profile* has six possible classes. We have classified all tables of all data sets to these classes and report our findings in Fig. 14. In Fig. 14, we group measurements separately for (a) non-survivors, (b) survivors, and (c) overall. Within each of these groups, apart from the overall statistics, we also provide the breakdown for tables with (a) no changes, (b) quiet change, and (c) active change. Each cell reports on the percentage of tables of a data set that fall within the respective subcategory. We have slightly manipulated a single dataset, for the sake of clarity. OpenCart has an extreme case of tables renamings, involving (a) the entire schema at transitions 17 and 18 and (b) massive renamings in transitions 23 and 35. As these massive renamings involved 108 of the 236 tables of the monitored history, we decided to exclude the tables that were dead before transition 23 from our study, so that their change does not overwhelm the statistics. For example, after the aforementioned short period, only 14 tables were removed in the subsequent 130 revisions.

Clearly, quiet tables dominate the landscape. If we observe the non-survivors, the percentage of tables in each of the categories is typically dropping as the probability of change rises: *sudden deaths are the most frequent, quiet non-survivors come second, and third, a very small percentage of tables in the vicinity of 0-6% dies after some intense activity.* In the case of the survivors, the percentage of rigid tables is often substantial, but, with the exception of phpbb, the landscape is dominated by quiet tables, typically surpassing 30% of tables and reaching even to 60% of the tables for a couple of data sets. Things become quite interesting if one focuses on the more mature data sets (coppermine, ensemble, mediawiki, opencart) whose profile shows some commonalities: in contrast to the rest of the datasets, active survivors are a small minority (approx. 5%) of the population, and similarly, active tables over both survivors and non-survivors attain a percentage that is less than 15%. So, *as the database matures, the percentage of active tables drops* (as already mentioned, evolution activity seems to calm down).

It is also noteworthy that *despite similarities, and despite the obvious fact that at least two thirds of the tables (in the mature databases: 9 out of 10)* *lead quiet lives, the internal breakdown of updates obliges us to admit that each database (and developer community) comes with its own update profile.*

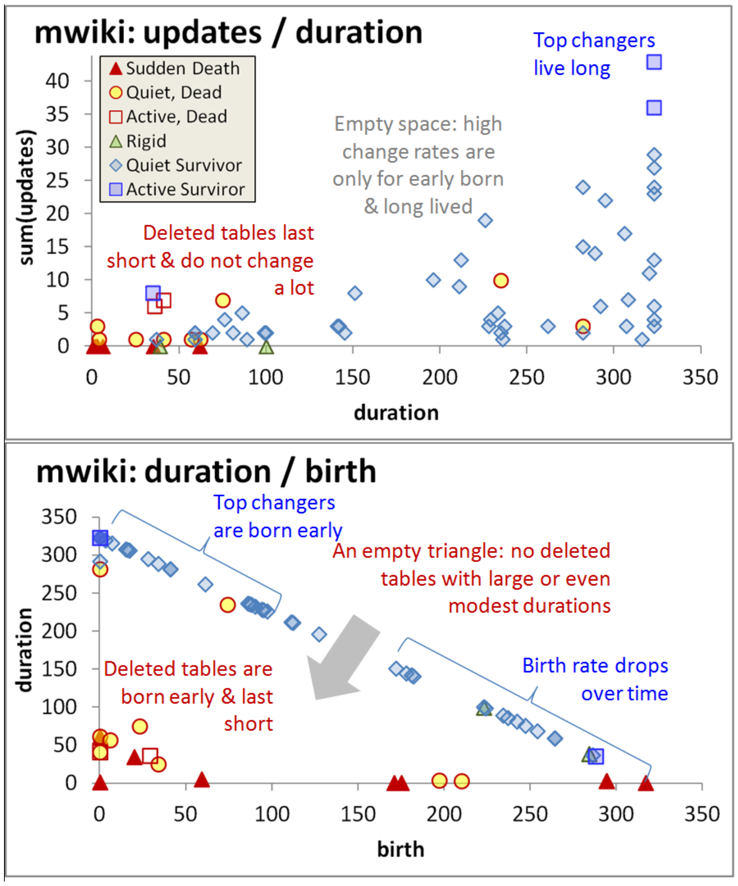


Figure 15: Update profile with respect to survival-vs-death injected in the study of total change over duration (top) and duration over birthday (bottom)

4.2. Table birth, duration & updates

In this subsection, we report on our findings concerning the study of the update profile of tables, and specifically, on patterns related to the characteristics of top changers, and in particular, their birth, duration and survival. As usually, we start with an exemplary demonstration of how these characteristics interrelate and then proceed to establish the observed patterns via statistical profiling.

In Figure 15 we present the most interesting insights from the Mediawiki data set. We have chosen Mediawiki as a fairly representative dataset, as it comes with both a non-negligible amount of removed tables, a large duration, and, a fairly large number of tables (72). Our findings are discussed below.

Figure 15 demonstrates a *concentration of survivor top-changer tables at high durations.* In fact, the statistical processing of our data reveals that *very often, top-changers are born early, live long, have high average transitional update and, consequently a large amount of total update.*

The two arguments that we can make on this phenomenon are detailed as follows.

- *Surviving top-changers live long.* Observe the top chart of Figure 15: the vertical line of the *inverse* Γ in long durations, as well as its nearby region are being populated by survivors; the vertical line demonstrates that several of the top-changers are very long lived. Remember that the top-changers are the ones with the highest average transitional update, i.e., their update is normalized over their duration. This is quite different from the total amount of update that is depicted in the y-axis of the chart: in other words, it is theoretically possible (and in fact, there is indeed such a case in the Mediawiki example, found close to the beginning of the axes) that a short lived table with relatively low total updates is a top-changer. In practice, however, longevity and high average transitional update are closely related.
- Concerning the top-changers, there is an interesting correlation of high update activity, overall change, duration and birthdate: *most (although not all) top-changers are born early, live long, have high average transitional update* and, consequently a large amount of total update. Of course, there exist tables born early who survive and have lower update activity and top-changers that violate this rule; however, in all data sets, it is quite uncommon to observe top-changers outside the "box" or early birth and long duration.

4.2.1. Statistical evidence for the properties of top-changers

Terminology. We consider a table to have been born early if its birth version is in the lowest 33% of versions; respectively, late-deaths take place after the 77% of the number of versions. Durations follow the same pattern: short durations are in the lowest 33% of durations and long durations in the upper 23% (strictly higher than the 77% of the maximum duration). The term "few updates" refers to quiet and rigid tables.

Research Question: What is the probability that a top changer table is born early - lives long - survives? The data of Figure 16 demonstrate the relationship of duration, survival and birth data vividly.

- In all data sets, active tables are born early with percentages that exceed 75%
- With the exceptions of two data sets, active tables survive with percentage higher than 70%. In Mediawiki the percentage is 60% and in Ensemble, where too many tables die, the percentage is 48%

	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%

Figure 16: Top changers: early born, survivors, often with long durations, and often all the above

- The probability of an active table having a long duration is higher than 50% in 6 out of 8 data sets.
- Interestingly, the two last lines of Figure 16 are identical. Probing into the data, the study revealed that (a) the active tables with long duration and (b) the active tables that are born early, survive and have a long duration *are exactly the same sets in all data sets*! In other words:
  - An active table with long duration has been born early and survived with probability 100%
  - An active, survivor table that has a long duration has been born early with probability 100%

4.3. Table death, duration & updates

We have been interested a lot for the case of removed tables. Which are the common characteristics of tables that die? Interestingly, one can say that the *tables that are removed from the schema are quiet, early born, short lived, and quite often all three of them.*

In Figure 17 we see the correlation of birth, duration, survival and update activity in a single diagram, for all eight data sets.

- There is a large concentration of the *deleted tables in a cluster of tables that are newly born, quickly removed, with few or no updates*. Few deleted tables demonstrate either late birth, or long durations, or high average transitional update, or large number of updates (Mediawiki shows just a couple of them in each of the first two categories). In the triangle formed in the bottom figure by the two axes and the line of survivors, the diagonal of survivors is expected: if you are a survivor, the earlier you have been born, the more you live. *The really surprising revelation of the triangle, however, is that (a) the diagonal is almost exclusively made from survivors, and, (b) the triangle is mainly hollow*! Theoretically, it is absolutely legitimate for non-survivors to have long durations and to be born in the middle of the database lifetime: if there was not a pattern of attraction of non-survivors to the beginning of the axes we could legitimately see non-survivors in the

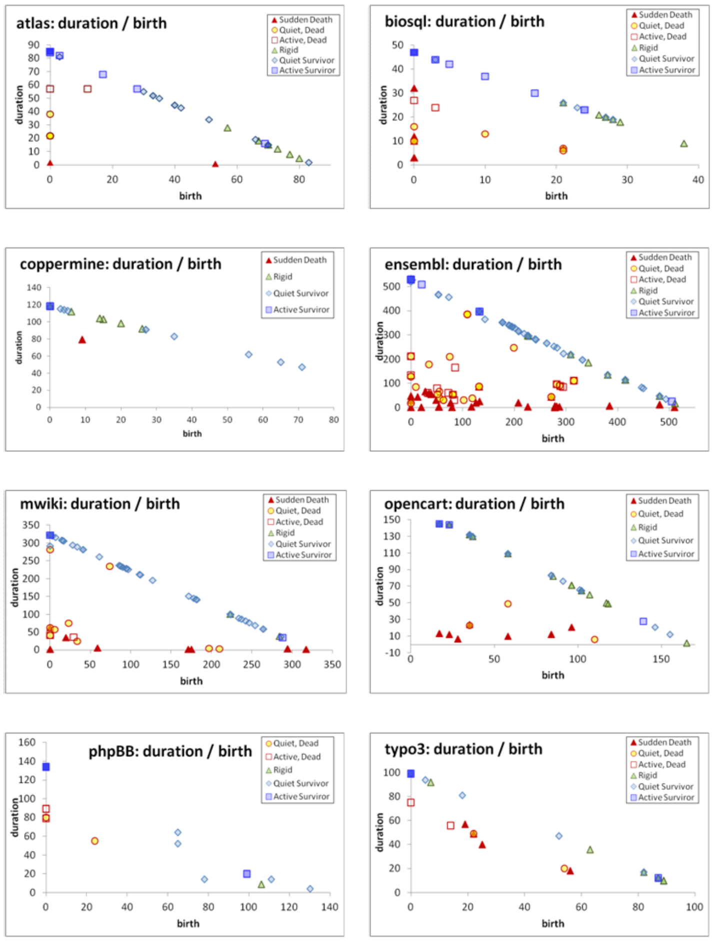


Figure 17: The empty triangle pattern for all data sets

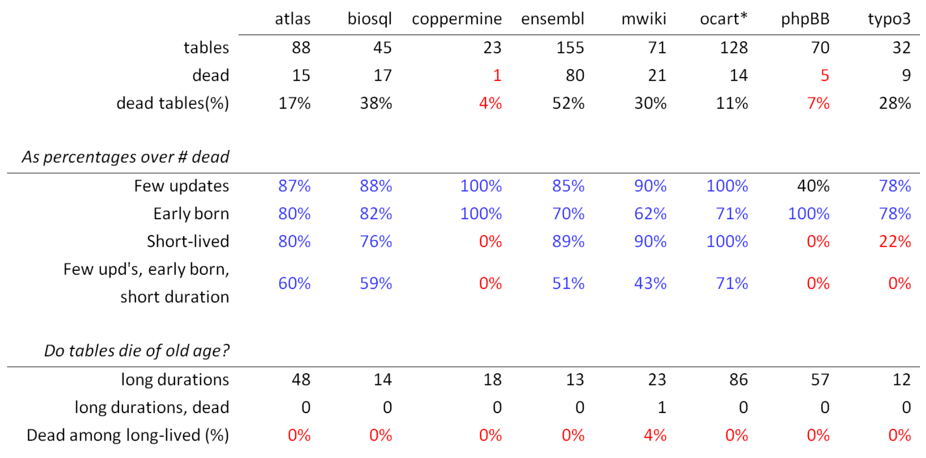
area of the diagonal and, of course, a uniform spread in the interior of the triangle. In fact, in the case of Mediawiki, we do see a couple (but just a couple) of such cases for both the aforementioned possibilities. Overall, we find that the hollowness of the triangle is an unexpected revelation and, so, we refer to the pattern as the *Empty Triangle pattern*.

- *It is quite rare to see tables being removed at old age; although each data set comes with a few such cases, typically, the area of high duration is overwhelmingly inhabited by survivors!* See the upper part of Fig. 15 for the Mediawiki data set, showing just a couple of deleted tables with high durations.

4.3.1. Statistical evidence for the relationship of age and probability of removal

Terminology. We consider a table to have been born early if its birth version is in the lowest 33% of versions; respectively, late-deaths take place after the 77% of the number of versions. Durations follow the same pattern: short durations are in the lowest 33% of durations and long durations in the upper 23% (strictly higher the 77% of the maximum duration). The term ”few updates” refers to quiet and rigid tables.

Research Question: What is the probability that a table that is removed is short-lived? Early-born? What is its update profile? We have tried to answer the above questions, and of course, their combinations. Figure 18 summarizes our findings. In all the data sets but one (phpBB) the tables that are removed are tables with few updates at an overwhelming percentage (over 85% in 6 out of 8 cases). In all data sets, these tables have been born early too (in all cases except for mwiki, the percentage of early born tables as a fraction over the population of removed tables reaches at least 70%). Short lived tables demonstrate a slightly different profile, as we have a bipolar situation: one the one hand, the three datasets with very few dead tables, specifically coppermine, phpBB and typo3, have zero or very few removed tables with short durations, and, on the other hand, in all the other data sets, the percentage of tables who die after a short life exceeds 75%. The situation’s bipolarity is clear when we take the combination of properties: in the ”rigid” data sets, with the few table removals, there are no tables that are early born, short lived and with quiet lives, whereas in the rest of the data sets the percentage of such tables ranges within 43% and 71% (in half the data sets more than 50%). These numbers are simply significant, as here we combine three properties in our test and in half the data sets the absolute majority of the population of dead tables satisfies the test’s predicate. In other words, it is fair to say that *in databases where it is not uncommon to see tables being removed, it is very probable that removed tables are simultaneously quiet, early born, and short lived.*



	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%

Figure 18: Dead tables are: quiet, early born, short lived, and quite often all three of them

Research Question: What is the probability that a table of long duration is removed? Observe the last part of Figure 18. *In all but one of the data sets, there is no occurrence of an old timer being removed!* The pattern is not trivial or easy to ignore: the numbers of tables with long duration, in absolute terms, are really high, ranging from 12 to 86 tables. This kind of rigidity seems to be one of the strongest patterns that we have observed overall in our deliberations.

Exceptions. Exceptions to the pattern of the empty triangle do exist. In the case of Typo3, the diagonal is closely followed by a set of medium-change tables. This is mainly attributed to the fact that there is a short period of 5 transitions, around transition 70 during which almost all table deletions take place. Opencart has the characteristic of massive renamings that removed all the old tables until transition 34. After that, there are a few targeted restructurings of sudden deaths. Ensembl is the only data set with a heavy deletion oriented character (52% of tables removed). In the case of Ensembl, although the area close to the diagonal has just a couple of deleted tables, there are two differences: (a) the attraction to the beginning of the axes last quite longer than every other data set (a box of 135x135) and (b) there are several tables born around the middle of life of the data set that eventually die, having been fairly active and quite long-lived (approximately 200 transitions).

Conjectures. We believe (although cannot prove) that both the empty triangle pattern and the survival of old-timers have to do with the cost of removing a table after some time: applications have been built around it and the maintenance cost to the surrounding queries shrinks the possibility of removal.

5. Scope of the study and threats to validity

5.1. Scope of the study

Our study has a well-specified *scope*. We have chosen datasets as representatives of a population with the following properties:

- the databases of study are part of open-source software projects (and not proprietary ones)
- as Figure 1 clearly demonstrates, the chosen schemata come with substantially long history in terms of time (on average the length of the chosen projects is close to 8 years).

Having framed our scope, we are also very careful to stress what our study does not cover and should not be part of the possible generalizations that one can make:

- We should be very careful to not overgeneralize any findings to proprietary databases or physical schemata. We stress that our study has been performed on databases being part of open-source software. This class of software comes with a specific open modus operandi in terms of committing changes and distributing maintenance work. Thus, we should be very careful to not overgeneralize findings to proprietary databases.

- We would also like to stress that we work only with changes at the logical schema level and ignore physical-level changes like index creation or change of storage engine. We do not take properties of the intentional level (e.g., number of rows, or disk size) into consideration either.

5.2. Measurement and data treatment reliability

Can our measurements be trusted? Is it possible that we erroneously assess the amount of change for the studied data sets?

Overall, we believe we have a good data extraction and measurement process without interference or manipulation of the input from our part.

- 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.
- Hecate provides a fully automated counting for the studied measures. Of course, all subsequent analysis was performed by humans; yet, we have checked and rechecked for errors in spreadsheet formulae (which were, of course, corrected)
- We have not interfered with the input. The only exception was a handful of cases that where adapted in the Coppermine data set to avoid over-complicating the parser; other than that we have not interfered with the input at all. We believe that there is no serious threat to validity from the point of view of the manipulation of data.

5.3. External validity

External validity refers to the possibility of generalizing the findings of a study to a broader context. Can we generalize out findings broadly?

Overall, we believe we have given a specific scope to our data and we are quite confident with our results, as we have covered a fairly large number of databases, from a variety of domains that seem to give consistent answers to our research questions – of course, with all the exceptions that we have mentioned in our previous discourse.

To support the external validity of our study, we will address several concerns that can be raised.

Represented population. Is the population represented by our data sets well defined?

Concerning the *scope* of our study and its *clarity*, we have been very clear from the previous subsection, on defining the scope and framing the possible generalizations. We are very careful to warn the reader against generalizing our findings in the wrong context (closed development environments, physical parts of the schema or other). In terms of *precision*, all our data sets belong to the specified population.

Representativeness of our data sets. Are our data sets representative enough? Is it possible that the observed behaviors are caused by sui-generis characteristics of the studied data sets?

We have a fairly large number of eight studied data sets – at least at the extent of what a single study can span. To address *variety* and avoid a monothematic study, we have chosen to encompass in our study datasets from different domains (specifically, physics, biomedicine, and CMS's). The schema size of the involved data sets varies too: we have schemata ranging from 22 to 114 tables. Also, the amount of growth, although always positive varies too. Moreover, although we have chosen data sets with a large history in terms of time, the history in terms of committed changes to the database schema varies too: the number of number of versions spans from rather few (40) to quite many (500+). Overall, although one could possibly construct a valid argument that application areas or initial properties can affect the way databases evolve (e.g., CMS's can behave differently than scientific databases), in this paper we have focused on patterns that we found common in all data sets. So, we are confident for our results from this perspective.

To avoid possible counter-arguments, we also stress that we are very cautious to avoid declaring any *laws of universal application*. We do support our findings; at the same, we believe that from an epistemological point of view it is only appropriate to wait from several more studies, preferably conducted by several research groups, other than ours, to verify that these patterns hold by and large as laws.

We are strong advocates of the openness and reusability of data, software and results. All our data sets and software are openly available at our group's site at Github (`https://github.com/DAINTINESS-Group`) and all our results are also available at the paper's companion web site (`http://www.cs.uoi.gr/~pvassil/publications/2015_ER/`) to be reused and put to the test by the scientific community.

5.4. Internal validity

Internal validity refers to the case where a dependent variable's behavior is related to the behavior of an independent variable via a cause-effect relationship. In our case, we wish to be clear that (a) we perform an *exploratory study to observe frequently occurring phenomena* within the scope of the aforementioned population and (b) we *are very careful to avoid making strong causation statements!*

In several places in our text, we conjecture on the causes for a particular phenomenon. However, we have not performed any kind of controlled experiment or study to be able to support causation arguments by data. This is simply not the case we are making in this paper. Moreover, we would also like to add that it is indeed quite possible that our correlations hide hidden, confounding variables. Overall, we would like to be crystal clear that matters of causation are outside the scope of our study and we make no such arguments.

6. Related work

The first empirical study on the evolution of database schemas has been performed in the 90's [1]. The author monitored the database of a health management system for a period of 18 months. The results of the study showed that all the tables of the schema were affected and the schema had a 139% increase in size for tables and 274% for attributes. The author further observed a significant impact of the changes to the related queries. Interestingly, the author also reports the observation of a particular inflating period during the construction of the system where almost all changes were additions, and a subsequent period where additions and deletions where balanced.

With the proliferation of open source software, certain other efforts were able to investigate the evolution of databases, along with the applications that depend on them. The second empirical study [2] was performed much later than the previous one. In this study, the authors analyzed the database back-end of MediaWiki, the software that powers Wikipedia, for a period 4.5 years. The results showed a 100% increase in schema size. However, 45% of changes did not affect the information capacity of the schema (but were rather index adjustments, documentation, etc.). The authors further provided a statistical study of lifetimes, change breakdown and version commits. This line of research was based on PRISM (recently re-engineered to PRISM++ [11]), a change management tool.

In [5], the authors worked with 4 case studies and performed a change frequency and timing analysis, which showed that the database schemas tend to stabilize over time, with more change at the beginning of their lifetime and a convergence to a more stable schema later. In [3], two case studies revealed that schemas and source code do not always evolve in sync.

In [6], the authors investigated ten case studies. The authors report that change is focused both (a) with respect to time and (b) with respect to the change profiles of tables . In terms of timing, 7 out of the 10 databases reached 60% of their schema size within 20% of their early lifetime. Also, change is frequent in the early stages of the databases, with inflationary characteristics; then, the schema evolution process calms down. In terms of change profiles, it is interesting that 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. Similarly to other studies, the authors assessed the degree to which code and database schema are in sync. The authors report that there is a large percentage of cases where the two artifacts are out of sync. Finally, it is noteworthy that according to the authors, database schemata evolve frequently during the application lifecycle, with schema changes causing a significant amount of code level modifications: each atomic change at the schema level is estimated to result in $10 - 100$ lines of application code been updated; and a valid database revision results in $100 - 1000$ lines of application code being updated.

Recently, we studied the applicability of Lehman's laws of software evolution in the case of database schemas [7, 9]. In this line of work we report on evidence that schemas grow so as to satisfy new requirements. However, this growth does not evolve linearly or monotonically, but with periods of calmness and short periods of focused maintenance activity. After a sufficient amount of changes, the schema size reaches a more mature level of stability.

Whereas previous work has mostly focused on the macroscopic study of the entire database schema, in this line of work, we zoom into the lives of tables. We study the relationship of table duration, survival and update activity with table characteristics like schema size, time of birth etc. To the best of our knowledge, this is the first time that such findings appear in the related literature. The current paper extends [8] with detailed discussions of the observed patterns and their statistical profile, along with side observations that were omitted from the conference version due to lack of space.

7. Discussion of findings and open issues

To conclude our deliberations, we take a quick tour of our findings, also discussed from the viewpoint of their practical consequences. We complement this discussion with open issues for further research.

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

Figure 19: Summary of findings

7.1. *Summary of findings*

7.1.1. *The Γ Pattern*

The Γ *pattern* states: *tables with small schema sizes can have arbitrary durations, whereas tables with larger schema sizes last long.* In a nutshell, our findings around the Γ *pattern* can be summarized as follows:

- In 6 out of 8 cases, whenever a table exceeds the critical value of 10 attributes in its schema, there is at least an 85% probability that it survives.

- In 5 out of 8 cases, the probability that wide tables are frequently created early on and are not deleted afterwards exceeds 50% (more than 70% in 4 cases).
- Whenever a table is wide, there is a bipolar pattern on its duration: in half the data sets, the duration of wide tables lies within the top-band of durations (upper part of Γ) with support higher than 70%, whereas the rest of the cases clearly disprove it, with very low support values.
- If a wide table is in the top of the Γ line, it is typically (deterministically in our data sets) an early born survivor.

7.1.2. The Comet Pattern

The *Comet pattern* states: *(a) a large, dense cluster close to the beginning of the axes, denoting small size and small amount of change, and, (b) a sparse set of outliers, broken in two subcategories: (b1) medium schema size tables typically demonstrating medium to large amounts of changes and (b2) "wide" tables with large schema sizes demonstrating small to medium amount of change.* In summary, the research findings concerning the *Comet pattern* are as follows:

- Around 70% of the tables are typically found inside a 10×10 narrow-and-quiet "nucleus" box in the *schema size* × *update activity* space.
- Tables with large numbers of updates tend to typically have medium schema sizes at birth ("upper comet's tail"): In all but one data set, the average schema size for the top 5% of tables in terms of their update behavior is, on average, close to half the maximum value of schema size at birth (also close to one standard deviation up from the average value of the schema size at birth).
- Tables with large schema sizes at birth tend to typically have medium numbers of updates ("lower comet's tail"): The amount of updates for the top 5% of tables with respect to their schema size at birth is typically close to the 38% of the domain of values for the sum of updates.

7.1.3. The Inverse Γ Pattern: Gravitation to Rigidity

The *Inverse Γ pattern* states: *tables with small duration undergo small change, tables with medium duration undergo small or medium change, and, long-lived tables demonstrate all kinds of change behavior.*

7.1.4. Quiet tables dominate

We have studied how tables are distributed in the the space produced by *update profile* × *survival profile.*

- If we observe the non-survivors, as the probability of change rises, there are less tables in each category: *sudden deaths are the most frequent, quiet non-survivors come second, and third, a very small percentage of tables in the vicinity of 0-6% dies after some intense activity.*

- In the case of the survivors, the percentage of rigid tables is often substantial, but, with the exception of phpbb, the landscape is dominated by quiet tables, typically surpassing 30% of tables and reaching even to 60% of the tables for a couple of data sets. Active survivors are also a substantial percentage, but with a bipolar pattern: for the more mature data sets, active survivors are a small minority (approx. 5%) of the population.

- Overall, independently of survival: Clearly, *quiet tables dominate the landscape.* Also, *as the database matures, the percentage of active tables drops and ranges lower than 15%.*

7.1.5. Top-changers' profile

The statistical profile of tables with high average amount of updates per transition, also known as top-changers, reveals that very often, *top-changers are born early, live long, and due to their high average transitional update come with a large amount of updates.* Specifically, the probabilities that a top changer table is born early - lives long - survives are as follows:

- In all data sets, active tables are born early with percentages that exceed 75%

- With the exceptions of two data sets, active tables survive with percentage higher than 70%. In Mediawiki the percentage is 60% and in Ensemble, where too many tables die, the percentage is 48%

- The probability of an active table having a long duration is higher than 50% in 6 out of 8 data sets.

Within the population of active tables, survival and being born early (consequently: longevity, too) go hand-in-hand:

- An active table with long duration has been born early and survived with probability 100%

- An active, survivor table that has a long duration has been born early with probability 100%

7.1.6. The Empty Triangle Pattern

The *Empty Triangle pattern* says that *few deleted tables demonstrate either late birth, or long durations, or high average transitional update, or large number of updates*, resulting in a practically hollow triangle in the *birth* $\times$ *duration* space. At the same time, there is often a large concentration of deleted tables in a cluster of tables that are newly born, quickly removed, with few or no updates, whereas "old-timer" tables of long durations are scarcely removed.

Dead tables come with:

- few updates at an overwhelming percentage (over 85% in 6 out of 8 cases);

- early birth (in all cases except for one, the percentage of early born tables as a fraction over the removed ones reaches at least 70%);
- short life in half the cases: datasets with very few dead tables, have zero or very few removed tables with short durations, whereas, in all the other data sets, the percentage of tables who die after a short life exceeds 75%;
- a bipolarity in the combination of the three properties altogether: in the "rigid" data sets, with the few table removals, there are no tables that are early born, short lived and with quiet lives, whereas in the rest of the data sets the percentage of such tables ranges within 43% and 71% (in half the data sets more than 50%).

In other words, it is fair to say that *in databases where it is not uncommon to see tables being removed, it is very probable that removed tables are simultaneously quiet, early born, and short lived.* At the same time, it is quite rare to see tables being removed at old age; although each data set comes with a few such cases, typically, the area of high duration is overwhelmingly inhabited by survivors!

7.2. Practical considerations

Assume you are the chief architect of a large information system. Evolution is inevitable, maintenance takes up to 70% of resources that are always limited, time is pressing and you have to keep the system up to date, correct, without failures and satisfactory for the users. In this subsection, we relate the key findings of our study with *guidelines on designing and building both tables and applications that access them, so as to sustain the evolution of the database part gracefully.*

Encapsulate change via views. Top changers are typically born early and last long. Deleted tables, although quiet, are also born early and last short; consequently, these deletions typically take place early in the lifetime of the project. Although the majority of tables live quiet lives, the ones that depart from the typical path of a quiet survivor can cause a lot of maintenance effort.

To address change, we suggest the usage of views as an *evolution-buffering, "API-like" mechanism* between applications and databases.

> *Mask your applications from tables prone to change (e.g., newly born tables with high probability of change or deletion) as much as possible, especially at the early versions of the database where most of the deletions take place. To this end, use views as a buffering mechanism between the database and the applications that masks change in several evolution scenarios like renamings, table splitting or merging, etc.*

As a side note, we would also like to put a word of caution to the reader. As the Γ pattern suggests that wide tables survive, a concerned designer might be tempted to denormalize the database schema in order to come up with survivor wide tables. We strongly denounce table denormalization as a solution that reduces the number of potentially evolution-prone, thin tables by introducing few, artificially wide, reference tables. Normalization and foreign keys is a fundamental mechanism for ensuring data integrity; its avoidance hides the danger of data inconsistencies. Rather, we suggest that virtual views are defined to mask joins and aggregations from the developers.

Graceful, quiet expansion: databases live quiet lives. Although each database comes with its own idiosyncracy, still, we can coarsely say that databases expand over time. Typically, table deletions take place in earlier stages of the database life, whereas table additions take place throughout all the life of the database. Deletions will occur in later stages too, in a form of perfective maintenance of the schema, but still, they take place with less intensity. If we zoom from the database to the table level, we can see that most tables lead quiet lives with few updates (remember: additions and deletions of attributes, data type and key changes). Medium-rated change is scarcely present too: in fact, the *inverse* Γ pattern states that updates are not proportional to longevity, but rather, only few top-changer tables deviate from a quiet, low-change profile.

Overall, we have named this tendency towards quiet lives **gravitation to rigidity** and we have attributed it to the **dependency magnet** nature of databases: applications are built on top of the database, and even minor changes impact the application code, both syntactically and semantically, requiring significant maintenance effort.

Should we stay assured that things are fine with this situation? We believe that rigidity is a major concern for a software module, as it leads to large effort and cost for any necessary maintenance, and often, to strange design decisions / hacks in order to reduce it. It would be really desirable if code and schema had the elasticity to adapt easily to new requirements, debugging and perfective maintenance. Until we employ our DBMS's with built-in mechanisms that facilitate this flexibility, however, our only remedy to the problem is tracking of the dependencies between the application code and the database (see [12] for our take on the problem). This allows the easy highlighting of impacted parts of the code and the possibility of automatic rewritting of the application.

> *Plan for a quiet expansion of the schema! The database is augmented with new tables all the time; at the same time, although most tables lead quiet lives and do not grow much, some top-changers are "change attractors". To address the database expansion, try to systematically use metadata rich facilities involving the interdependencies of applications and tables to be able to locate where the code should be maintained for added, deleted or modified attributes and tables.*

7.3. Open Issues (quoting the wisdom of Leonardo da Vinci's notebooks)

Possibilities for follow-up work are immense. We will try to summarize the major areas where we think that the research community can make contributions. At the same time, we will also make a critical discussion on the risks that these research paths pose, too.

Can the patterns be turned to laws? Clearly, a first important desideratum would be to uncover the hidden mechanism that drives the evolution of the schema and to provide *causation* statements (i.e., laws of the form " a change in quantity X causes a related change in quantity Y"). Time and again in this paper, we have attributed the patterns of change, the gravitation to rigidity, and the profile of the parts of the database that do evolve more frequently, to the dependency between application code and supporting database. To our point of view, the key to understanding the hidden mechanisms behind schema evolution is to go deep in detail in the study of specific cases on how external requirements, application code and database schema jointly co-evolve. However, this task comes with an increased level of difficulty. On the one hand, the open-source nature of the available schema histories means that developer communities can be spread geographically and without a tight rhythm of development. On the other hand, in closed environments, there is always the difficulty to obtain access to the schema and its history, and be allowed to publish the results for the rest of the community. Overall, the difficulty of the task calls for a larger research program rather than a single study: to be certain of laws rather than observed patterns, we would need a set of in-depth studies, preferably from independent groups, before we can claim knowledge of the specific mechanics behind the evolution of software in general, and schemata in particular. This work, being an observatory effort on actual cases, has been a first step in a long path based on the premise that *wisdom is the daughter of experience*, but, as *truth was the only daughter of time*, long and hard work by many will be required before the laws of schema evolution are unveiled to us.

Can we predict change? Clearly, there is always the ambitious scientific goal of having some "weather forecast" for the forthcoming changes of a database. However, as all the databases that we have studied have their own idiosyncracy (e.g., see Fig. 14) we can safely argue that predicting change is quite more complicated than the a posteriori study of the evolution of the schema. We should warn younger readers on the level of risk this research encompasses: although patterns do exist, we observe them *a posteriori* and at the macroscopic level; it is still unclear if we can *a priori*, safely forecast at the detailed (table and attribute) level, which part of a schema will change and how. To start with, it is still unclear if there is an underlying mechanism dictating how and when changes take place. Moreover, despite the heroic battle of perfective maintenance against rigidity, most changes typically come from external requirements of a changing world, which are, of course, really hard to automatically predict (if explicitly expressed, they should better be taken as input to the forecasting problem). To quote the master: *Necessity is the theme and the inventress, the eternal curb and law of nature.*

Truth is so excellent, that if it praises but small things they become noble. At a more technical level, one can also deal with the possibility of capturing more types of changes (like renamings, normalization actions, etc) that have not been part of our fully automated mechanism for transition extraction. The little experimental evidence that we have so far ([2]) suggests that this kind of operations are a small fraction of the evolution actions that take place in the life of a database. However, we cannot be sure without a thorough investigation. In any case, identifying composite transitions (like e.g., renaming) requires extending our fully automatic change detection mechanism with semi-automatic mechanisms where some expert user verifies the candidate changes that a tool reports.

Every instrument requires to be made by experience. Finally, as already mentioned, one could also work on the arguably feasible engineering goal of having flexible structures for gluing applications to evolving database schemata, thus minimizing application dependency and evolution impact.

References

[1] D. Sjøberg, Quantifying schema evolution, Information and Software Technology 35 (1) (1993) 35–44.

[2] C. Curino, H. J. Moon, L. Tanca, C. Zaniolo, Schema evolution in wikipedia: toward a web information system benchmark, in: Proceedings of ICEIS 2008, Citeseer, 2008.

[3] D.-Y. Lin, I. Neamtiu, Collateral evolution of applications and databases, in: Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops, IWPSE-Evol '09, 2009, pp. 31–40.

[4] M. Hartung, J. F. Terwilliger, E. Rahm, Schema Matching and Mapping, Springer, 2011, Ch. Recent Advances in Schema and Ontology Evolution, pp. 149–190.

[5] S. Wu, I. Neamtiu, Schema evolution analysis for embedded databases, in: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering Workshops, ICDEW '11, 2011, pp. 151–156.

[6] D. Qiu, B. Li, Z. Su, An empirical analysis of the co-evolution of schema and code in database applications, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013, 2013, pp. 125–135.

[7] I. Skoulis, P. Vassiliadis, A. Zarras, Open-source databases: Within, outside, or beyond Lehman's laws of software evolution?, in: Proceedings of 26th International Conference on Advanced Information Systems Engineering - CAiSE 2014, 2014, available at `http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/`.

[8] P. Vassiliadis, A. V. Zarras, I. Skoulis, How is Life for a Table in an Evolving Relational Schema? Birth, Death and Everything in Between, in: Proceedings of 34th International Conference on Conceptual Modeling (ER 2015), Stockholm, Sweden, October 19-22, 2015, 2015, pp. 453–466.

[9] I. Skoulis, P. Vassiliadis, A. V. Zarras, Growing up with stability: How open-source relational databases evolve, Information Systems 53 (2015) 363–385.

[10] M. H. Dunham, Data Mining: Introductory and Advanced Topics, Prentice-Hall, 2002.

[11] C. Curino, H. J. Moon, A. Deutsch, C. Zaniolo, Automating the database schema evolution process, VLDB J. 22 (1) (2013) 73–98.

[12] P. Manousis, P. Vassiliadis, G. Papastefanatos, Automating the adaptation of evolving data-intensive ecosystems, in: Proceedings of 32th International Conference on Conceptual Modeling (ER 2013), Hong-Kong, China, November 11-13, 2013., pp. 182–196.