Survival in schema evolution: putting the lives of survivor and dead tables in counterpoint

Panos Vassiliadis and Apostolos V. Zarras

Department of Computer Science and Engineering, University of Ioannina, Greece
{pvassil, zarras}@cs.uoi.gr

Abstract. How can we plan development over an evolving schema? In this paper, we study the history of the schema of eight open source software projects that include relational databases and extract patterns related to the survival or death of their tables. Our findings are mostly summarized by a pattern, which we call "electrolysis pattern" due to its diagrammatic representation, stating that dead and survivor tables live quite different lives: tables typically die shortly after birth, with short durations and mostly no updates, whereas survivors mostly live quiet lives with few updates – except for a small group of tables with high update ratios that are characterized by high durations and survival. Based on our findings, we recommend that development over newborn tables should be restrained, and wherever possible, encapsulated by views to buffer both infant mortality and high update rate of hyperactive tables. Once a table matures, developers can rely on a typical pattern of gravitation to rigidity, providing less disturbances due to evolution to the surrounding code.

Keywords: Schema evolution, Evolution patterns, Table Survival

1 Introduction

The study of schema evolution in an attempt to dig out patterns and regularities is an important endeavor in order to understand its mechanics and plan software design and development on top of databases. However, this problem has attracted little attention by the research community so far. To a large extent, the possibility of actually studying schema evolution emerged from the availability of schema histories embedded in open source software projects, publicly available via Github. So far, research efforts [6], [2], [4], [11], [5] – see Sec. 2– have demonstrated that schemata grow over time, mostly with insertions and updates, and are frequently out of synch with their surrounding code. However, we are still far from a detailed understanding of how individual tables evolve and what factors affect their evolution. In our latest work [10, 9], we have performed a first study towards charting the relationship of factors like schema size and version of birth to the duration and the amount of change a table undergoes. In this paper, we continue along this line of work by answering a fundamental question on the survival of a table that has not been answered so far: "how are survival, activity behavior and duration of a table interrelated?". To the best of our knowledge, the problem was only initially touched in [10, 9] and the insights of this paper are completely novel in the related literature.

Following the research method of our previous work, we have performed a large study of eight data sets with the schema history of databases included in open source projects (see Sec. 3 for our experimental setup). Our results are detailed in Sec. 4; here, we can give a concise summary of our findings as follows. The antithesis of the durations between dead and survivor tables is striking: table deletions take place shortly after birth, resulting in short durations for the dead tables; this is to be contrasted with the large number of survivors with high (and frequently, maximum) durations. When activity profile, duration and survival are studied together, we observe the *electrolysis pattern*, named after the paradigm of positive and negative ions in electrolysis moving towards opposite directions: Not only dead tables cluster in short or medium durations, and practically never at high durations, but also, with few exceptions, the less active dead tables are, the higher the chance to reach shorter durations. In contrast, survivors are mostly located at medium and high durations and the more active survivors are, the stronger they are attracted towards high durations, with a significant such inclination for the few active survivors, that cluster in very high durations.

Why is the knowledge of patterns in life and death of tables so important? We believe that our study gives solid evidence on a phenomenon that we call *gravitation to rigidity*[1] stating that despite some valiant efforts, relational schemata suffer from the tendency of developers to minimize evolution as much as possible in order to minimize the resulting impact to the surrounding code. Sec. 5 discusses possible explanations on the relationship of the observed phenomena with gravitation to rigidity. Equally importantly, understanding the probability of update or removal of a table can aid the development team in avoiding to invest too much effort and code to high-risk parts of the database schema. To this end, in Sec. 5 we provide recommendations to developers, based on our findings and also suggest roads for future work.

2 Background and Related Work

The first known case study of schema evolution, published in 1993 [6], monitored the database of a health management system for 18 months, to report the overall increase of schema size over time and the percentage breakdown for different types of changes. After this study, it was only 15 years later that research was revived on the problem. The key to this revival was the existence of open source software repositories exposing all the code of a software project in all its history. Software projects based on relational databases, thus, would expose the entire history of their schema. There is a handful of works since the late '00s [2] [4] [11] [5] that have assessed the evolution of databases involved in open source software projects.

[1] *Rigidity* is used in its software engineering meaning, referring to software that is hard to evolve and maintain.

In [2], the authors report findings on the evolution of Mediawiki, the software that supports Wikipedia. The authors of this work, and also in the followup work on "algebrizing" schema modification operations [3] should be accredited for the public release of schema histories that they collected. Several works followed, where the authors have primarily worked on (a) the schema size, which grows over time but with progressively less rate [5], (b) the absence of total synchronization between source code and database schema, as schemata evolve [4] [11], and, (c) the impact of schema change to the surrounding code [5], which requires a significant amount of code modifications. [5] is also presenting preliminary results on the timing of the schema modifications, reporting that the early versions of the database included a large part of the schema growth. A study presented in [1] verifies the observations of other works concerning the trend of increase in schema size and the reluctance in the deletion of tables.

Our recent involvement in the area is based on the study of the history of the schema of eight open source software projects. In [7], also presented in full length in [8], we have worked at the macroscopic level to study how the schema of a database evolves in terms of its size. We have found evidence that *schemata grow over time in order to satisfy new requirements, albeit not in a continuous or linear fashion, but rather, with bursts of concentrated effort interrupting longer periods of calmness and drops, signifying perfective maintenance.*

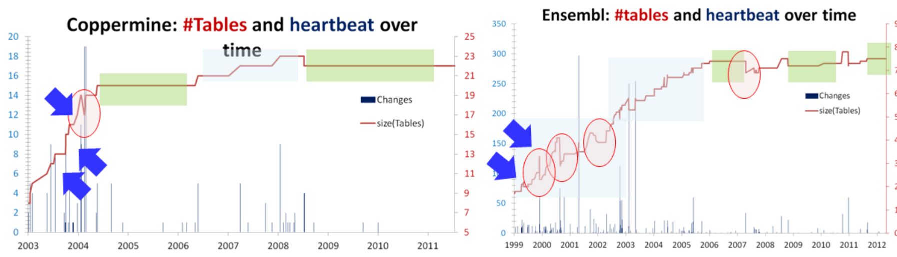


Fig. 1. Summary of [8] with schema growth over time (red continuous line) along with the heartbeat of changes (spikes) for two datasets. Overlayed darker green rectangles highlight the calmness periods, and lighter blue rectangles highlight smooth expansions. Arrows point at periods of abrupt expansion and circles highlight drops in size.

Whereas all related work had focused on the study of *schema* size, in [10], also presented in full length in [9], we have worked on the identification of frequently encountered patterns on *table* properties (e.g., birth, duration, amount of change). We identified four major patterns on the relationship of such properties. The *Γ pattern* on the relationship of the schema size of a table at its birth with its overall duration indicates that tables with large schemata tend to have long durations and avoid removal. The *Comet pattern* on the relationship of the schema size of a table at its birth with its total amount of updates indicates that the tables with most updates are frequently the ones with medium schema size. The *Inverse Γ pattern* on the relationship of the amount of updates and the duration of a table indicates that tables with medium or small durations produce amounts of updates lower than expected, whereas tables with long duration expose all sorts of update behavior. The *Empty Triangle* pattern on the relationship of a table's version of birth with its overall duration indicates a significant absence of tables of medium or long durations that were removed –thus, an empty triangle – signifying mainly short lives for deleted tables and low probability of deletion for old timers.

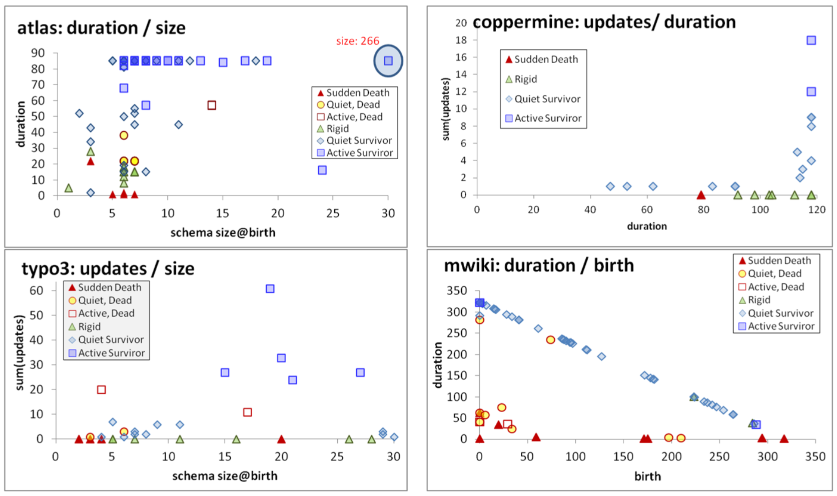


Fig. 2. The 4 patterns of [10], [9]: Gamma (top left), inverse Gamma (top right), comet (bottom left) and empty triangle (bottom right).

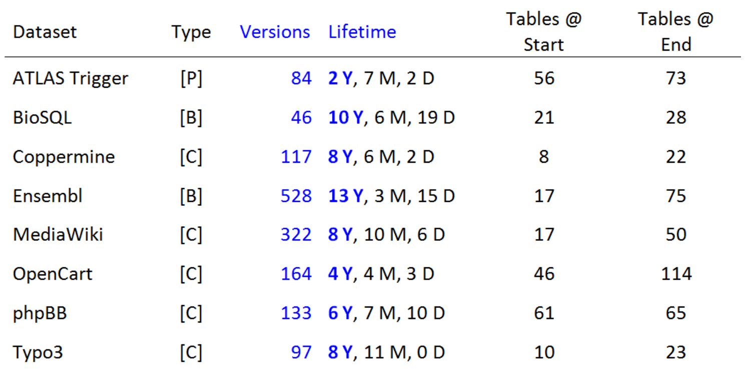
Although insightful, the aforementioned findings *have not exhausted the search on factors affecting survival*, and so, in this paper, we extend our knowledge by exploring *how survival is related to duration and activity profile*. To the best of our knowledge this is the first comprehensive study of this kind in the literature.

3 Experimental Method

In this section, we briefly present our experimental method. Here we can only provide a self-contained, condensed description, so, we will kindly refer the interested reader to [7] for a detailed description and to our *Schema Biographies* website[2] containing links to all our results, data, code and presentations that are made publicly available to the research community.

Experimental protocol. We have collected the version histories of 8 data sets that support open source software projects. For each dataset we gathered as many schema *versions* (DDL files) as we could from their public source code

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



Dataset	Type	Versions	Lifetime	Tables @ Start	Tables @ End
ATLAS Trigger	[P]	84	**2 Y**, 7 M, 2 D	56	73
BioSQL	[B]	46	**10 Y**, 6 M, 19 D	21	28
Coppermine	[C]	117	**8 Y**, 6 M, 2 D	8	22
Ensembl	[B]	528	**13 Y**, 3 M, 15 D	17	75
MediaWiki	[C]	322	**8 Y**, 10 M, 6 D	17	50
OpenCart	[C]	164	**4 Y**, 4 M, 3 D	46	114
phpBB	[C]	133	**6 Y**, 7 M, 10 D	61	65
Typo3	[C]	97	**8 Y**, 11 M, 0 D	10	23

Fig. 3. Datasets used in our study

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, available only for Oracle). The files were then processed by our tool, Hecate, that detected, in a fully automated way, ((a) changes at the table-level, i.e., which tables were inserted and deleted, and (b) updates at the attribute-level, and specifically, attributes inserted, deleted, having a changed data type, or participation in a changed primary key.

Reported Measures. Hecate pair-wise compared subsequent files and reported the changes for each *transition* between subsequent versions. The details of each particular change along with collective statistics per table, as well as for the entire schema were also reported. An important part of the produced measures involves information on the update profile of each table, including the total number of changes it went through, the change rate etc. We have classified tables in profiles concerning (a) their *survival* (i.e., their presence in the last version of the schema history or not), characterizing them as *survivors* or *dead*, (b) their *activity behavior*, characterizing them as *rigid* (if they go through zero updates), *active* (if their rate of change is higher than 0.1 changes per transition) and *quiet* otherwise, and, (c) by the combination of the above via their Cartesian product, which we call *LifeAndDeath* profile.

Scope. Concerning the scope of the study, we would like to clarify that we work only with changes at the logical schema level (and ignore physical-level changes like index creation or change of storage engine). Also, the reader is advised to avoid generalizing our findings to proprietary databases, outside the realm of open source software.

4 Survival and duration: how dead tables differ from survivors

In this section, we first explore whether there is a difference in the duration between survivor and dead tables. Then, we examine how table duration, survival and activity behavior interrelate.

4.1 Oppositely Skewed Durations

We have studied how the duration of tables is distributed in different duration ranges, thus creating a histogram of durations. We have discriminated between dead and survivor tables, so we have a histogram for each of these two classes. Fig. 4 depicts the respective histograms. We observe a phenomenon, which we call the *oppositely skewed durations* or *opposite skews* pattern.

The oppositely skewed durations pattern. When one constructs the histograms for the durations of dead vs survivor tables one can observe a symmetry in the histograms of the two classes. *The dead tables are strongly biased towards short durations (left-heavy), often with very large percentages of them being removed very shortly after birth. In quite the opposite manner, the survivor tables are mostly gathered at the other end of the spectrum (right-heavy), i.e., at high (frequently: max) durations.*

Exceptions to the pattern. Exceptions to the pattern do exist, albeit they do not significantly alter its validity. Coppermine's single deleted table was removed at 6 years of age. The phpBB database, which is otherwise too rigid, has 5 deleted tables that were removed at significantly larger durations than the typical in other data sets (in fact after 5 or 6 years of lifetime, all 5 being removed in the same version). The typo3 database, also has a set of 9 removed tables, again with quite high durations (7 of which had a lifetime between 4 and 8 years at the time of their removal).

Gravitation to rigidity. We attribute the tendency to short durations for the deleted tables to the cost that deletions have for the maintenance of the software that surrounds the database. The earlier a table is removed, the smaller the cost of maintaining the surrounding code is. Thus, when the table has been involved in several queries found in several places in the code, it is always a painstaking process to locate, maintain and test the application code that uses it. At the same time, the reluctance for removals allows tables who survive the early stages to "remain safe". Thus, they grow in age without being removed. This fact, combined with the fact that the starting versions of the database already include a large percentage of the overall population of tables, results in a right-heavy, left-tailed distribution of survivor tables (for 6 out of 8 data sets, survivor durations reaching the final bucket of the respective histogram exceed 45%).

4.2 The electrolysis pattern

What happens if we relate duration with activity? The research question that is guiding us here is to discover whether there are patterns in the way survival,

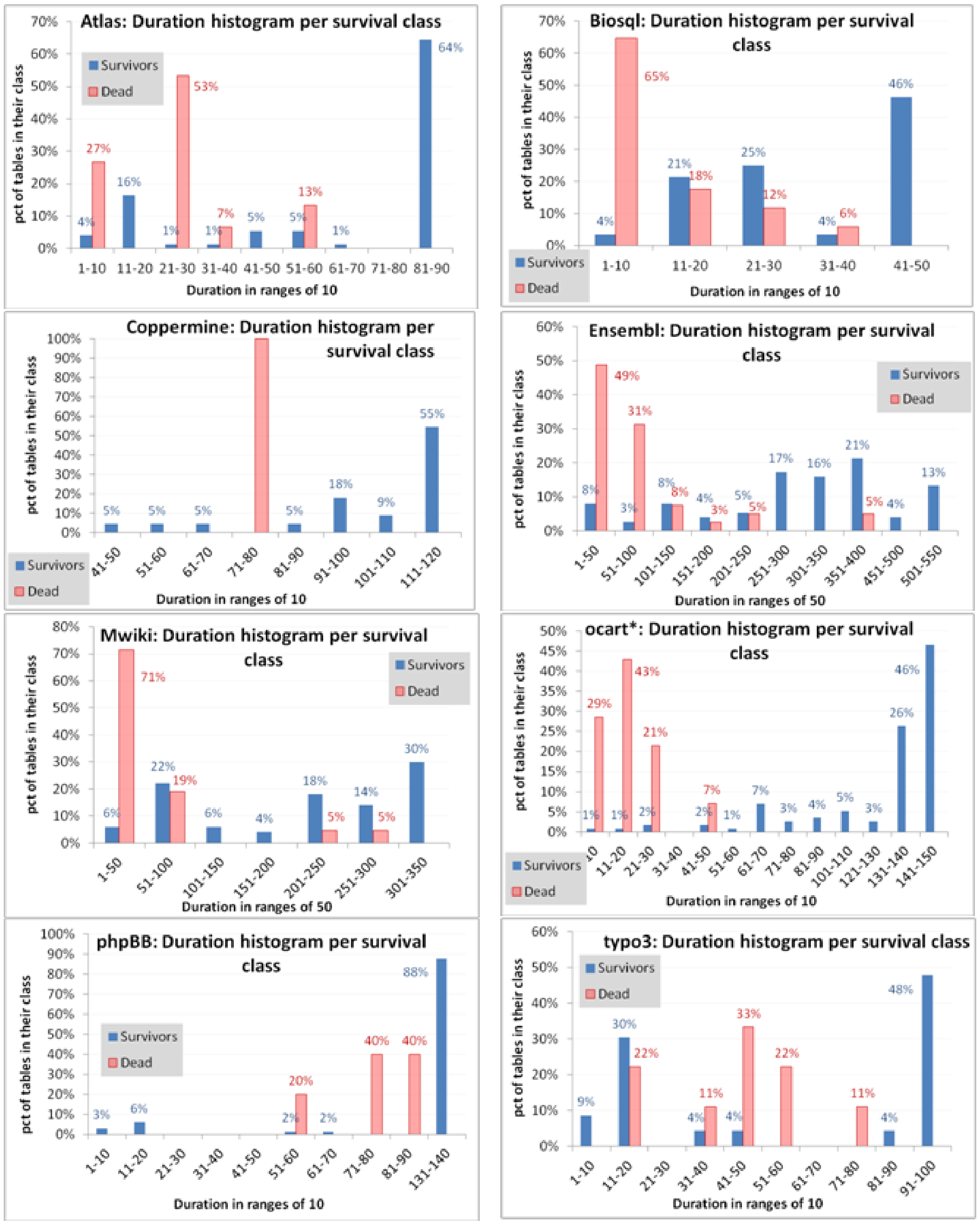


Fig. 4. Histograms of the durations of (a) dead, vs., (b) survivor tables.

duration and activity behavior relate. Our analysis is a refinement of the oppositely skewed durations pattern with activity profile information. Whereas the opposite skews pattern simply reports percentages for duration ranges, here, we refine them by *LifeAndDeath* Class too. So, in the rest of this subsection, we will group the tables according to the *LifeAndDeath* class, which expresses the profile of a table with respect to the combination of *survival* x *activity*, practically composing the two domains $\{dead, survivor\} \times \{rigid, quiet, active\}$ into their Cartesian product. Then, for each of the resulting six classes, we study the durations of the tables that belong to it.

The essence of the pattern. We formulate our observations as a new pattern, which we call the electrolysis pattern. Remember than in an electrolysis experimental device, two electrodes are inserted in water: a negative electrode, or cathode and a positive electrode, or anode. Then, negatively charged ions move towards the positive anode and positively charged ions move towards the negative cathode.

A somewhat similar phenomenon occurs for dead and survivor tables concerning the combination of duration and survival, which we call the *electrolysis pattern*. In Fig. 5, we graphically depict the phenomenon via scatter-plots that demonstrate the *LifeAndDeath* $\times$ *Duration* space for all the studied data sets.

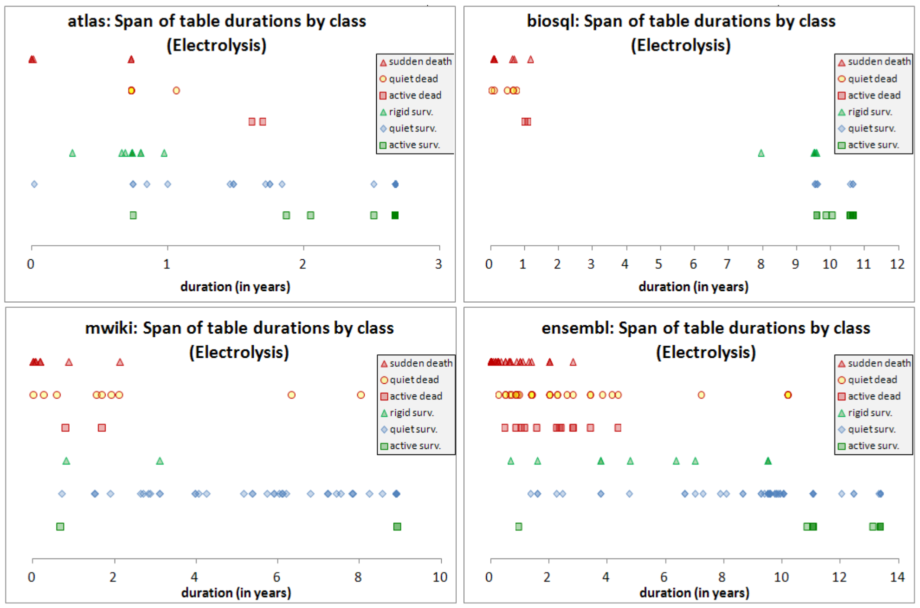


Fig. 5. The Electrolysis pattern. Each point refers to a table with (a) its duration at the x-axis and (b) its LifeAndDeath class (including both survival and activity) at the y-axis (also its symbol). Points are semi-transparent: intense color signifies large concentration of overlapping points.

Electrolysis pattern: Dead tables demonstrate much shorter lifetimes than survivor ones and can be located at short or medium durations, and practically never at high durations. With few exceptions, the less active dead tables are the higher the chance to reach shorter durations. Survivors expose the inverse behavior i.e., mostly located at medium or high durations. The more active survivors are, the stronger they are attracted towards high durations, with a significant such inclination for the few active ones that cluster in very high durations.

Fig. 5 vividly reveals the pattern's highlights. Observe:

- The total absence of dead tables from high durations.
- The clustering of rigid dead at low durations, the spread of quiet dead tables to low or medium durations, and the occasional presence of the few active dead, that are found also at low or medium durations, but in a clustered way.
- The extreme clustering of active survivors to high durations.
- The wider spread of the (quite numerous) quiet survivors to a large span of durations with long trails of points.
- The clustering of rigid survivors, albeit not just to one, but to all kinds of durations (frequently, not as high as quiet and active survivors).

One could possibly argue that the observed clusterings and time spans are simply a matter of numbers: the more populated a class is, the broader its span is. To forestall any such criticism, this is simply not the case. We give the respective numbers in the sequel of this subsection; here, we proactively mention a few examples to address this concern. Rigid dead tables are the most populated group in the dead class, yet they have the shortest span of all. The rigid survivors, who are the second most populated class of the entire population, exhibit all kinds of behaviors; yet, in most of the cases, they are disproportionally clustered and not spread throughout the different categories. Active survivors are also disproportionately clustered at high durations. Overall, with the exception of the quiet survivors that indeed span a large variance of durations, in the rest of the categories, the time span is disproportionate to the size of the population (number of points in the scatter plot) of the respective class.

In-depth study of durations. To understand how tables are distributed in different durations, we have expressed table durations as percentages over the lifetime of their schema. Then, for each *LifeAndDeath* value and for each duration range of 5% of the database lifetime, we computed the percentage of tables whose duration falls within this range. Then, we proceeded in averaging the respective percentages of the eight data sets[3].

Both for lack of space and understanability reasons, we have to condense the detailed data in an easily understandable format. To summarize the detailed

[3] An acute reader might express the concern whether it would be better to gather all the tables in one single set and average over them. We disagree: each data set comes with its own requirements, development style, and idiosyncrasy and putting all tables in a single data set, not only scandalously favors large data sets, but integrates different things. We average the behavior of schemata, not tables here.

data, we regrouped the data in just 3 duration ranges, presented in Fig. 6: (a) durations lower that the 20% of the database lifetime (that attracts a large number of dead tables, esp., the rigid ones), to which we will refer as *low durations*, in the sequel, (b) durations higher that the 80% of the database lifetime (where too many survivors, esp., active ones are found), to which we will refer as *high durations*, and finally, (c) the rest of the durations in between, forming an intermediate category of *medium* durations.

	Rigid Dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv	
[0-20%)	8%	5%	2%	4%	3%	1%	*23%*
[20%-80%)	3%	3%	1%	5%	13%	0%	*26%*
[80%-100%]	0%	0%	1%	14%	24%	12%	*51%*
	12%	*8%*	*3%*	*23%*	*40%*	*14%*	*100%*

Fig. 6. Indicative, average values over all datasets: for each LifeAndDeath class, percentage of tables per duration range over the total of the entire data set.

Atlas	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0% -20%)	57%	0%	0%	9%	3%	0%
[20%-80%)	43%	100%	100%	91%	30%	12%
[80%-100%]	0%	0%	0%	0%	68%	88%
	100%	*100%*	*100%*	*100%*	*100%*	*100%*

Biosql	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)	100%	100%	100%	0%	0%	0%
[20%-80%)	0%	0%	0%	14%	0%	0%
[80%-100%]	0%	0%	0%	86%	100%	100%
	100%	*100%*	*100%*	*100%*	*100%*	*100%*

Copperm.	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)	0%			0%	0%	0%
[20%-80%)	100%			0%	23%	0%
[80%-100%]	0%			100%	77%	100%
	100%			*100%*	*100%*	*100%*

Ensembl	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)	97%	65%	67%	20%	9%	9%
[20%-80%)	3%	35%	33%	80%	65%	0%
[80%-100%]	0%	0%	0%	0%	26%	91%
	100%	*100%*	*100%*	*100%*	*100%*	*100%*

Mwiki	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)	90%	56%	100%	50%	9%	33%
[20%-80%)	10%	33%	0%	50%	49%	0%
[80%-100%]	0%	11%	0%	0%	42%	67%
	100%	*100%*	*100%*	*100%*	*100%*	*100%*

Ocart*	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)	36%	100%		19%	13%	25%
[20%-80%)	64%	0%		41%	38%	0%
[80%-100%]	0%	0%		41%	50%	75%
	100%	*100%*		*100%*	*100%*	*100%*

phpBB	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)		0%	0%	3%	9%	25%
[20%-80%)		50%	0%	0%	14%	0%
[80%-100%]		50%	100%	97%	77%	75%
		100%	*100%*	*100%*	*100%*	*100%*

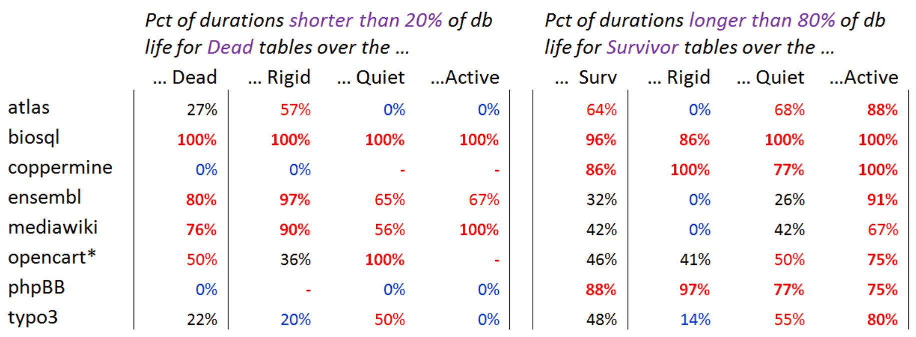
typo3	Rigid dead	Quiet Dead	Active Dead	Rigid Surv	Quiet Surv	Active Surv
[0-20%)	20%	50%	0%	71%	27%	20%
[20%-80%)	80%	50%	50%	14%	18%	0%
[80%-100%]	0%	0%	50%	14%	55%	80%
	100%	*100%*	*100%*	*100%*	*100%*	*100%*

Fig. 7. For each data set, for each LifeAndDeath class, percentage of tables per duration range over the total of the LifeAndDeath class (for each data set, for each column, percentages add up to 100%).

Breakdown per *LifeAndDeath* class. Another research question concerns the breakdown of the distribution of tables within each *LifeAndDeath* class. In other words, we ask: *do certain LifeAndDeath classes have high concentrations in particular duration ranges*?

If one wants to measure the percentage of tables for each value of the *LifeAndDeath Class* over each duration range, we need to calculate each cell as the percentage of the specific class (e.g., each cell in the rigid dead column should measure the fraction of rigid dead tables that belong to its particular duration range).The detailed data per data set, where the value of each cell is presented as percentage over its *LifeAndDeath* class are depicted in Fig. 7.

Finally, in Fig. 8 we zoom only in (a) dead tables at the lowest 20% and (b) survivors at the highest 20% of durations. We count the number of tables, per *LifeAndDeath* class, for the respective critical duration range, and we compute the fraction of this value over the total number of tables pertaining to this *Life-AndDeath* class (columns *Rigid*, *Quiet*, *Active*). For the *Dead* and *Surv* columns, we divide the total number of dead/survivor tables belonging to the respective critical duration over the total number of dead/survivor tables overall.



	Pct of durations shorter than 20% of db life for Dead tables over the ...				*Pct of durations longer than 80% of db life for Survivor tables over the ...*			
	... Dead	... Rigid	... Quiet	...Active	... Surv	... Rigid	... Quiet	...Active
atlas	27%	57%	0%	0%	64%	0%	68%	**88%**
biosql	**100%**	**100%**	**100%**	**100%**	**96%**	**86%**	**100%**	**100%**
coppermine	0%	0%	-	-	**86%**	**100%**	**77%**	**100%**
ensembl	**80%**	**97%**	65%	67%	32%	0%	26%	**91%**
mediawiki	**76%**	**90%**	56%	**100%**	42%	0%	42%	67%
opencart*	50%	36%	**100%**	-	46%	41%	50%	75%
phpBB	0%	-	0%	0%	**88%**	**97%**	**77%**	75%
typo3	22%	20%	50%	0%	48%	14%	55%	80%

Fig. 8. Percentages of dead tables with too short durations and survivor tables with too long durations (red: above 50%, bold: above 75%, blue: below 20%, dash: no such tables).

We observe that in more than half of the cells of the table in Fig. 8, the percentage reaches or exceeds 50%. This clearly demarcates the high concentrations of dead tables in low durations and of survivor tables in high durations.

Observations and Findings. Now, we are ready to quantitatively support the wording of the electrolysis pattern. We organize our discussion by *LifeAnd-Death* class. Our quantitative findings for the electrolysis pattern are delineated in the rest of this subsection.

Dead tables. We already knew from [9] that almost half the dead tables are rigid. Here, we have a clear testimony, however, that *not only are dead tables inclined to rigidity, but they are also strongly attracted to small durations*. The less active tables are the more they are attracted to short durations. *The attraction of dead tables, especially rigid ones, to (primarily) low or (secondarily) medium durations is significant and only few tables in the class of dead tables escape this rule*. Interestingly, in all our datasets, the only dead tables that escape the barrier of low and medium durations are a single table in mediawiki, another one in typo3 and the 4 of the 5 tables that are simultaneously deleted in phpBB.

– *Rigid dead tables, which is the most populated category of dead tables, strongly cluster in the area of low durations (lower than the 20% of the database lifetime) with percentages of 90% – 100% in 3 of the 6 data sets* (Fig. 8). Atlas

follows with a large percentage of 57% in this range. Two exceptions exist: opencart and typo3, having most of their dead tables in the medium range. There are also two exceptions of minor importance: coppermine with a single deleted table and phpBB with a focused deletion of 5 tables at a single time point.
- *Quiet dead tables, which is a category including few tables, are mostly oriented towards low durations*. Specifically, there are 5 data sets with a high concentration of tables in the area of low durations (Fig. 8); for the rest of the data sets, the majority of quiet dead tables lie elsewhere: atlas has 100% in the medium range and phpBB is split in half between medium and large durations.
- Finally, for the very few active dead, which is a category where only six of the eight data sets have even a single table, there are two of them with 100% concentration and another one in 67% of its population in the low durations (Fig. 8). For the rest, atlas has 100% of its active dead in the medium range, phpBB 100% of the active dead in the long range (remember that phpBB has an exceptional behavior) and typo3 is split in half between low and medium durations (Fig. 7).

Survivors. *Survivors have the opposite tendency of clustering compared to the dead ones*. So, there are quite a few cases where survivor tables reach very high concentrations in high durations, and, interestingly, the more active the tables are, the higher their clustering in high durations.

- *Rigid survivors demonstrate a large variety of behaviors*. Rigid survivors are the second most populated category of tables after quiet survivors and demonstrate too many profiles of clustering (Fig. 7): one data set comes with a low-heavy profile, another 3 with a high-heavy profile, another two with a medium-heavy profile, and there is one data set split in half between early and medium durations and another one with an orientation of medium-to-high durations.
- *Quiet survivors, being the (sometimes vast) majority of survivor tables, are mostly gravitated towards large durations, and secondarily to medium ones*. In 6 out of 8 data sets, the percentage of quiet survivors that exceed 80% of db lifetime surpasses 50% (Fig. 7). In the two exceptions, medium durations is the largest subgroup of quiet survivors. Still, quiet survivors also demonstrate short durations too (Fig. 7), so overall, their span of possible durations is large. Notably, in all data sets, there are quiet survivors reaching maximum duration.
- It is extremely surprising that the vast majority of active survivors exceed 80% of the database lifetime in all datasets (Fig. 8). With the exception of three data sets in the range of 67%-75%, *the percentage of active survivors that exceed 80% of the db lifetime exceeds 80% and even attains totality in 2 cases*. Active survivor tables are not too many; however, it is their clustering to high durations (implying early birth) that is amazing. If one looks into the detailed data and in synch with the empty triangle pattern of [9], *the top*

changers are very often of maximum duration, i.e., early born and survivors (Fig. 5).

Absence of evolution. Although the majority of survivor tables are in the quiet class, we can quite emphatically say that *it is the absence of evolution that dominates*. Survivors vastly outnumber removed tables. Similarly, rigid tables outnumber the active ones, both in the survival and, in particular, in the dead class. Active tables are few and are mainly born in the early phases of the database lifetime.

5 Discussion, Take up and Future Work

Why do we see what we see. We believe that this study strengthens our theory that schema evolution antagonizes a powerful gravitation to rigidity. The "dependency magnet" nature of databases, where all the application code relies on them but not vice versa, leads to this phenomenon, as avoiding the adaptation and maintenance of application code is a strong driver towards avoiding the frequent evolution of the database. Some explanations around the individual phenomena that we have observed can be attributed to the gravitation to rigidity:

– Dead tables die shortly after their birth and quite often, rigid: this setting provides as little as possible exposure to application development for tables to be removed.
– As dead tables do not attain high durations, it appears that after a certain period, practically within 10%-20% of the databases' lifetime, tables begin to be "safe". The significant amount of tables that stand the chance to attain maximum durations can be explained if we combine this observation with the fact that large percentages of tables are created at the first version of the database.
– Rigid tables find it hard to attain high durations (unless found in an environment of low change activity). This difficulty can be explained by two reasons. First, shortly after they are born, rigid tables are in the high-risk group of being removed. Second, rigid tables are also a class of tables with the highest migration probability. Even if their duration surpasses the critical 10% of databases lifetime where the mass of the deleted tables lies, they are candidates for being updated and migrating to the quiet class.
– Tables with high durations (i.e., early born) that survive spend their lives mostly quietly (i.e., with the few occasional maintenance changes) – again minimizing the impact to the surrounding code.
– The high concentration of the few active tables to very high durations and survival (which is of course related to early births) is also related to the gravitation to rigidity: the early phases of the database lifetime typically include more table births and, at the same time, gravitation to rigidity says that after the development of a substantial amount of code, too high rate of updates becomes harder; this results in very low numbers of active tables being born later. So, the pattern should not be read so much as "active

tables are born early", but rather as "we do not see so many active tables being born in late phases of the database life".

Prediction of a table's life and recommendations for developers. How is a table going to live its life? Tables typically die shortly after their birth and quite often, rigid, i.e., without having experienced any update before. So, young rigid tables are the high risk group for being removed.

Typically, if a table surpasses infant mortality, it will likely survive to live a rigid or, more commonly, a quiet live. There is a small group of active tables, going through significant updates. Look for them in the early born survivors, as later phases of the database life do not seem to generate tables that are too active.

Overall, after a table is born, the development of code that depends on it should be kept as restrained as possible – preferably encapsulated via views that will hide the changes from the application code. After the period of infant mortality, it is fairly safe to say that unless the table shows signs of significant update activity, gravitation to rigidity enters the stage and the table's evolution will be low.

Threats to validity. It is always necessary to approach one's study with a critical eye for its validity. With respect to the *measurement validity* of our work, we have tested (i) our automatic extraction tool, Hecate, for the accuracy of its automatic extraction of delta's and measures, and (ii) our human-made calculations. With respect to the *scope* of the study, as already mentioned, we frame our investigation to schemata that belong to open-source projects. This has to do with the decentralized nature of the development process in an open source environment. Databases in closed organizational environments have different administration protocols, their surrounding applications are possibly developed under strict software house regulations and also suffer from the inertia that their evolution might incur, due to the need to migrate large data volumes. So, we warn the reader not to overgeneralize our results to this area. Another warning to the reader is that we have worked only with changes at the logical and not the physical layer. Having said that, we should mention, however, that the *external validity* of our study is supported by several strong statements: we have chosen data sets with (a) fairly long histories of versions, (b) a variety of domains (CMS's and scientific systems), (c) a variety in the number of their commits (from 46 to 528), and, (d) a variety of schema sizes (from 23 to 114 at the end of the study); kindly refer to Fig. 3 for all these properties. We have also been steadily attentive to work only with phenomena that are common to all the data sets. We warn the reader not to interpret our findings as laws (that would need confirmation of our results by other research groups), but rather as patterns. Favorably, some very recent anecdotal evidence, in fact coming from the industrial world, is corroborating in favor of our gravitation to rigidity theory (see the blog entry by Stonebraker et al., at `http://cacm.acm.org/blogs/blog-cacm/208958-database-decay-and-what-to-do-about-it/fulltext`). Based on the above, *we are confident for the validity of our findings within the aforementioned scope*.

Future work. Related literature suggests that database evolution cools down after the first versions. This has been studied for the slowdown of the birth rate, however a precise, deep investigation of the timing of the heartbeat of the database schema with all its births, deaths and updates is still pending. To our view, this practically marks the limits of analyses based on descriptive statistics. The next challenge for the research community lies in going all the way down to the posted comments and the expressed user requirements at the public repositories and try to figure out why change is happening the way it does. Automating this effort is a very ambitious goal in this context. Finally, the validation of existing research results with more studies from other groups, different software tools, hopefully extending the set of studied data sets, is imperative to allow us progressively to move towards 'laws' rather than 'patterns' of change in the field of understanding schema evolution.

References

1. Cleve, A., Gobert, M., Meurice, L., Maes, J., Weber, J.H.: Understanding database schema evolution: A case study. Sci. Comput. Program. 97, 113–121 (2015)
2. Curino, C., Moon, H.J., Tanca, L., Zaniolo, C.: Schema evolution in wikipedia: toward a web information system benchmark. In: Proceedings of ICEIS 2008. Citeseer (2008)
3. Curino, C., Moon, H.J., Deutsch, A., Zaniolo, C.: Automating the database schema evolution process. VLDB J. 22(1), 73–98 (2013)
4. Lin, D.Y., Neamtiu, I.: 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. pp. 31–40. IWPSE-Evol '09 (2009)
5. Qiu, D., Li, B., Su, Z.: 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. pp. 125–135. ESEC/FSE 2013 (2013)
6. Sjøberg, D.: Quantifying schema evolution. Information and Software Technology 35(1), 35–44 (1993)
7. Skoulis, I., Vassiliadis, P., Zarras, A.: 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)
8. Skoulis, I., Vassiliadis, P., Zarras, A.V.: Growing up with stability: How open-source relational databases evolve. Information Systems 53, 363–385 (2015)
9. Vassiliadis, P., Zarras, A., Skoulis, I.: Gravitating to Rigidity: Patterns of Schema Evolution -and its Absence- in the Lives of Tables. Information Systems 63, 24 – 46 (2017)
10. Vassiliadis, P., Zarras, A.V., Skoulis, I.: 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. pp. 453–466
11. Wu, S., Neamtiu, I.: Schema evolution analysis for embedded databases. In: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering Workshops. pp. 151–156. ICDEW '11 (2011)