Information Systems ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

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, Greece
[b] Bouvet ASA, Oslo, Norway

A R T I C L E I N F O

Article history:
Received 26 March 2016
Accepted 30 June 2016
Recommended by: F. Caino Jr.

A B S T R A C T

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.

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.

* Corresponding author.
E-mail addresses: pvassil@cs.uoi.gr (P. Vassiliadis), zarras@cs.uoi.gr (A.V. Zarras), giskou@gmail.com (I. Skoulis).

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

http://dx.doi.org/10.1016/j.is.2016.06.010
0306-4379/© 2016 Elsevier Ltd. All rights reserved.

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 were unable to work with real world-data to understand how schemata evolve. Remarkably, until the late 00s, there was just a single case study on the topic [1]. Lately, however, the research community has started to exploit the available information [2–6], 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) [2–7], 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

Please cite this article as: P. Vassiliadis, et al., Gravitating to rigidity: Patterns of schema evolution – and its absence – in the lives of tables, Information Systems (2016), http://dx.doi.org/10.1016/j.is.2016.06.010