

A study on the effect of a table’s involvement in foreign keys to its schema evolution

Konstantinos Dimolikas¹, Apostolos V. Zarras¹, and Panos Vassiliadis¹[0000–0003–0085–6776]

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

Abstract. In this paper, we study the evolution of tables in a schema with respect to *the structure of the foreign keys to which tables are related*. We organize a hierarchy of topological complexity for the structure of foreign keys, based on a modeling of schemata as graphs, where tables are classified in increasing order of complexity as: *isolated* (not involved in foreign keys), *source* (with outgoing foreign keys only), *lookup* (with incoming foreign keys only) and *internal* (with both kinds). Our study reveals that this hierarchy reflects also the update behavior of tables: topologically simple tables are more likely to have a life with few or zero schema updates, whereas, topologically complex tables are more likely to undergo high numbers of updates. Early versions of the database attract the large majority of births of complex tables, as opposed to the simple ones, demonstrating a pattern of reducing the introduction of complex, heavily updated constructs in the schema as time progresses.

Keywords: Schema Evolution · Foreign Keys · Software Evolution.

1 Introduction

How is the structure of the foreign keys to which a table is related affecting its behavior during schema evolution? In this paper, we study the evolution of tables in a schema, from the perspective of their *topology* of foreign keys. We study the histories of 6 relational schemata and we extract births and deaths of the tables, as well as the intra-table updates (attribute additions, deletions, data type and primary key updates) they went through from the subsequent versions of their schema definition files. We also extract their foreign key relations, too. We exploit the graph modeling of [9] to model tables as nodes and foreign keys as directed edges, and thus treat a schema version as a graph. A *Diachronic Graph* is the union of all the graphs of the different versions and the main tool we will employ to relate the graph-based characteristics and the activity of tables.

Our first contribution lies in the introduction of a concise taxonomy of graph topological patterns, classifying tables into (a) *isolated* (zero total degree), (b) *source* (zero fan-in), (c) *lookup* (zero fan-out), and, (d) *internal* tables (with both fan-in and fan-out degrees). All the degrees are measured over the aforementioned Diachronic Graph. *Our object of study has been the relationship of the*

topological profile of tables with their evolutionary activity. We have discovered that there is indeed a relationship and the hierarchy of topological complexity actually relates to the behavior of tables. Specifically, our findings indicate that:

1. The topologically complex, *internal tables demonstrate high intra-table schema update activity* and, at the same time, they are almost in their entirety born in the initiating version of the database - in other words, *subsequent versions do not come with births of such topologically complex, internal tables.*
2. At the other end of the complexity spectrum *isolated tables undergo very little or zero change* and, despite the fact that a fair percentage of them is present in the original version of the database, isolated tables *are the most likely to be added in subsequent versions of the history.*
3. In-between the spectrum of isolated and internal tables, *source tables appear to be more similar to isolated, resisting change and being more likely to appear later in the life of a database, and lookup tables being more similar to the internal ones.*

The above have (unexpectedly) revealed that *evolutionary behavior is dependent upon a hierarchy of topological complexity: more topologically complex tables appear to be fewer, active and born only early, with the opposite behavior to topologically simple tables.* Our final contribution is that we discuss our explanation of this observation, which we attribute to the *gravitation to rigidity* phenomenon (i.e., the progressive aversion of developers to modify the schema), along with its implications.

Outline. In Section 2 we survey related work. In Section 3, we delineate the graph modeling used to represent schemata, the data sets and their preprocessing. In Section 4 we relate the topological categories of the individual tables to their activity and in Section 5 we conclude with a discussion of our results.

2 Related Work

Related work in the area of studying schema evolution has been initiated mainly in the turning of the millennium, due to the existence of Free Open Source Software (FOSS) projects that contained databases for their operation. Till then, it was very hard for the research community to have the necessary data to study -let alone publish the findings- in the area of schema evolution. A single case is found in [5]. During the last decade, several works appeared that mainly studied the growth of schemata [2],[3], [13] [4], [1]: we know by now that schemata grow slowly over time, and, in fact with decreasing rate [4], and alterations of change (mostly table insertions and updates) with long periods of calmness [6], [7]. The study of individual tables has been performed in [11], [12], [10] revealing several survival and growth patterns.

To the best of our knowledge, the first work that studied how foreign keys evolve in the context of schema evolution of relational databases is [8] and its long version, [9]. The study was mainly done from the macroscopic, schema-level point of view and revealed that the evolution of foreign keys depends a lot on

the idiosyncrasy of the database itself. In some cases, foreign keys are treated as an integral part of the system, evolving along with their tables, whereas in some other cases, only a small subset of the tables is involved in foreign keys, while birth and death of foreign keys is mostly out of synch with the respective table events. The extremity of this treatment is demonstrated in two data sets where the foreign keys were completely removed from the schema. Another serious problem observed was that within the 20 data sets collected, the mere existence of foreign keys was evident in only 7 of them. In [9], the way that the total degree of the graph modeling affects survival is discussed too: high degree tables are survivors (with removed tables being mostly low or zero degree) and active. The current paper differentiates itself from the related work as it complements the macroscopic observations of [9] on how the *entire schema* evolves, with observations *at the level of individual tables*, and characterizes the evolution of individual tables *with respect to their graph characteristics, and in particular, classes of their graph topology*.

3 Background and Experimental Setup

In this section, we discuss our underlying graph model for schemata with foreign keys, our data sets and tool, the classification of tables in topological categories, and the handling of the problem of tables with more than one label in their history.

3.1 Modeling as a Graph and the Diachronic Graph

We follow the modeling of [9], which we also quote here for completeness. We treat a relational schema as a set of relations, along with their foreign key constraints. A relation is characterized by a name, a set of attributes and a primary key. A foreign key constraint is a 1:1 mapping between a set of attributes S in a relation, R_S , called the source of the foreign key, and a set of attributes T in a relation R_T , called the target of the foreign key. At the extensional level, the semantics of the foreign key denote a subset relation between the instances of the source and the instances of the target attributes. We model a database schema as a directed graph $G(V; E)$, with relations as nodes and foreign keys as directed edges, originating from their source and targeted to their target. *The Diachronic Graph* of the history of a schema is the union of all the nodes and edges that ever appeared in the history of the schema.

3.2 Data sets and their preprocessing

We base our study to the 6 data sets of [8], [9] (Fig. 1). Our set includes CMS's, resource management toolkits and scientific databases. Two of the datasets, SlashCode and Zabbix, demonstrate the explicit removals of foreign keys from the schema, with the former also introducing foreign keys late in the schema history. We have decided to work only with the periods where foreign keys were

Datasets	Versions	Tables @start	Tables @end	Tables @DG	Tables Growth	FKs @start	FKs @end	FKs @DG	FKs Growth
Atlas	85	56	73	88	30,4%	61	63	88	3,3%
BioSQL	47	21	28	45	33,3%	17	43	79	152,9%
Castor	194	62	74	91	19,4%	6	10	13	66,7%
Egee	17	6	10	12	66,7%	3	4	6	33,3%
Slashcode	399	42	87	126	107,1%	0	0	47	0,0%
Zabbix	160	15	48	58	220,0%	10	2	38	-80,0%

Fig. 1. Statistics for the datasets used in our study [8], [9]

present in the schema (versions 74 to 260 for Slaschcode and 1 to 150 for Zabbix), since no table could possibly have any topological properties outside these periods. All the metrics reported have been obtained via our Parmenidian Truth tool that models, visualizes and quantifies the evolution of schemata with foreign keys. Both due to the identical nature of the data sets and their processing, and the lack of space, we refer the reader to [8], [9] for a discussion of the *threats to validity* for the scope (Free Open Source Software), external, and measurement validity. *Both our tool and our data are publicly available for the research community at our Github repository <https://github.com/DAINTINESS-Group>.*

3.3 The topological categories of tables

In this subsection, we present the topological categories of tables based on their references to and from other tables. Fig. 2 depicts the distribution of tables over the combination of their in- and out-degrees at the Diachronic Graph for the 6 datasets. In the sequel, we introduce the different *topological categories*, or *labels*, which are determined on the basis of the topology of the Diachronic Graph (Fig. 3).

Fig. 2 shows the strong presence of *isolated* tables, i.e., tables with no in-citing edges and zero total degree, in 4 of the 6 datasets. Moreover, in 2 of these datasets, namely Castor and SlashCode, zero degree tables constitute an overwhelming majority.

Leaving isolated tables aside, the next most populous category consists of *source* tables with no incoming references and at least one outgoing foreign key. This category of tables includes populations varying from 19% to 62%.

The third category includes tables with only incoming references, so we refer to them with the label *lookup*. In the 6 datasets, there is a small group of tables that lie in this category, not exceeding the value of 36%, and in 5 out of the 6 data sets they are less than 20% of the tables. However, due to their "reusable" nature, they typically achieve degrees much higher than the source tables.

The last category contains tables that have both in- and out-degrees. We refer to this category as *internal* tables. By definition, internal tables come with the most complicated topological structure. Although Fig. 2 shows that this category is not very large, it comes with interesting properties, as we will demonstrate in the sequel.

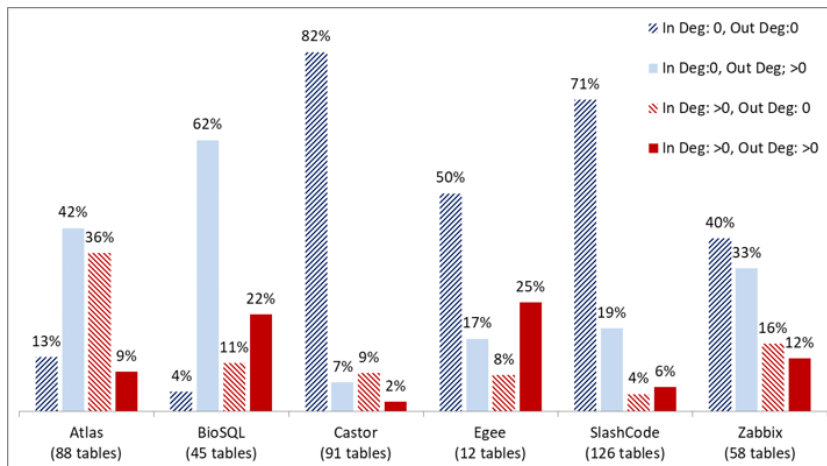


Fig. 2. Breakdown of tables wrt In- and Out-Degrees at the Diachronic Graph

3.4 Table Labeling For Multi-Label Nodes

Having introduced the topological categories, the next issue to resolve was the labeling of the tables. *Given just a single graph as input, the labeling of the tables is straightforward with a single pass over the nodes, as the categories are disjoint and independent of a node's neighborhood.*

The problem arises when the entire history of a schema is concerned. In this case, the input to the problem is a sequence of graphs. Then, it is possible that there are tables that change label throughout their history and as a result we end up with the following categories of tables with respect to their labels:

- *Single label* tables, which have a unique topological label throughout their entire lives.
- *Multi-label* tables, which have more than one label during their existence in the dataset.

Fig. 4 presents the distribution of tables between the ones with a single label and those with more than one label. Apart from Zabbix, in the rest of the datasets, the large majority of tables have a single label in their lives.

A problem that arises is that we would like to relate the labels of the tables to their activity profile and a multi-labeling scheme would not facilitate a statistical study along these lines. So fundamentally, we want to address the problem: *can we assign a single topological label to a table in a way that does not invalidate our statistical analysis and characterizes a table as accurately as possible?* To address this problem, we have manually inspected the tables with change-of-category and decided to assign a single label to each of them, since their number is so small that would not entail any major loss of information. We have distilled the phenomena of label changes for a table in 6 categories.





Name	Figure	Description
ISOLATED		Tables without FK's
SOURCE		Tables having only outgoing FK's
LOOKUP		Tables having only incoming FK's
INTERNAL		Tables with at least 1 incoming and at least 1 outgoing FK

Fig. 3. Table Categories Based on the Topology of the Diachronic Graph

Datasets	Total #tables	#Tables with...	
		single label	>1 label
Atlas	88	76	12
BioSQL	45	39	6
Castor	91	84	7
Egee	12	9	3
SlashCode	126	97	29
Zabbix	58	30	28

Fig. 4. Distribution of Tables over the Single and Multi-labels Categories

1. Changes that include an ephemeral transition to a different category and the return to the former category.
2. Changes from the *isolated* category to a different category.
3. Changes soon after the table's "birth".
4. Changes leading to labels assigned for a short period in terms of the number of versions.
5. Changes caused by changing self-references to the table.
6. All other changes.

The resolution of multiple labels into a representative, single label has been straightforward. See Fig. 5 on the categories and the rules for their resolution.

4 Research Findings

In this Section, we relate the topological category of a table to its behavioral characteristics in order to assess whether the former is correlated to the latter.

Rule	Description of Changes	Specific Criteria	Rule Decision
R0	No category change	$label_i = label_{i-1}$	The respective category
R1	Ephemeral category changes (DO-UNDO) in successive versions	$label_{i-1} = label_{i+1} \neq label_i$	Remove ephemeral $label_i$ and keep the remaining category $label_{i+1}$
R2	Changing from ISOLATED to another category	$label_{i-1} \neq label_i$ $label_{i-1} = ISO$	Remove <i>ISO</i> and keep the post-change label $label_i$
R3	Changing category in less than 10 versions after the First Known Version (FKV)	$label_{i-1} \neq label_i, i < 10$	Remove the first labels and keep the post-change label $label_i$
R4	Changing to a category for a short period of less than 10 versions	$label_i = \dots = label_{i+k}$ $k < 10, label_{i-1} \neq \{label_i, \dots, label_{i+k-1}\}$	Remove the period's labels & keep the pre-change label, $label_{i-1}$
R5	Changing category due to the presence of self-references	<i>An FK is added from the table to itself</i> \Rightarrow $label_i = INTERNAL$	Remove $label_i$ & keep the pre-change label, $label_{i-1}$
R6	Changes not abiding by any of the previous rules	-	Return the Most Frequent Label in the table's history

Fig. 5. Rules for Tables' Categories Determination

4.1 Birth

In this subsection we investigate if the birth versions of the tables are related to their topological categories. We are particularly interested in the relationship between the probability that a table is born in the originating version of the schema history and the topological category it belongs to, as, based on previous findings, a large percentage of the schema was created at the very early stages of the schema's history. In this context, we can formulate the relevant research question as follows:

Research Question: how is the topological category of a table related to the probability of being born in the originating version of the schema history?

Fig. 6 depicts the potential the tables of each topological category have to exist in the first version of their schema's history. As we want to perform statistical analyses, Egee is omitted due to its very small size. The patterns that we encounter with reference to the probability of tables being "born" in the earliest version of their schema can be summarized as follows:

- The tables of the *internal* category are 100% certain to be "born" in the originating version in three out of the five datasets. In BioSQL and Zabbix, although the overall population of the *internal* tables is not present in the first version, the probability for an internal table to be born at v0 is 67% for both data sets. Overall, there is no doubt that internal tables are almost entirely early born and *it is really highly unlikely to see internal tables being born later in the life of a schema.*
- *Lookup* tables have higher probabilities to be "born" in the first version compared to the respective average probability, and in fact, their majority is present at the first version of the schema for 4 out of 5 data sets.

Probability To Be Born @V0 Per Topological Category (Percentages Over #Tables Of Each Topological Category)

	Isolated		Source		Lookup		Internal		ALL TABLES	
	#Tables	Born @v0	#Tables	Born @v0	#Tables	Born @v0	#Tables	Born @v0	#Tables	Born @v0
Atlas	11	<i>9%</i>	38	61%	32	78%	7	100%	88	64%
BioSQL	2	100%	29	38%	8	50%	6	67%	45	47%
Castor	75	64%	6	83%	9	89%	1	100%	91	68%
SlashCode	35	<i>43%</i>	22	73%	7	86%	4	100%	68	60%
Zabbix	22	<i>9%</i>	20	30%	11	45%	3	67%	56	27%

Fig. 6. Probability to be "born" in the First Version per Topological Category (*blue italics*: lower than average, **red bold**: higher than average, both by at least 10%)

- Coming to *source* tables, the probability for a *source* table to be introduced in the first version of its dataset's history is, approximately, in accordance with the average probability and, in all datasets, it is lower than the respective potential of the *lookup* tables. In 2 out of 5 data sets, this probability is lower than the average probability of being born at v0. This signifies that it is easier for DBA's to add new source tables to the database during the evolution of the schema than it is to add lookup and internal ones.
- The tables of the *isolated* category have the lowest potential for being "born" in the originating version of their datasets, in four of the five datasets. Equivalently, we can claim that it is easier to add tables of this category over the course of a database's schema evolution than introducing tables of any other category.

The common features among the datasets related to the probability for a table to be "born" in the originating version if it belongs to a certain topological category are supported to some extent by the statistical evidence that assess the independence of the birth version from the topological categories. Specifically, we performed the Chi-square and Fisher statistical tests by utilizing a contingency table consisted of four rows representing the topological categories and two columns corresponding to tables born in the first version and those that are not. The p-values that do not exceed the limit of 5% are 4.74E-02 for Atlas, 1.36E-02 for SlashCode and 3.22E-02 for Zabbix.

To sum up, we observe that internal and lookup tables are more likely to be born in the originating version of their dataset's history, which, expressed in a different way, means that it is quite unlikely that they are "born" after this version. In contrast, source tables follow the trend of the general population and isolated tables are the ones with higher chances to be born in versions succeeding the originating one.

Breakdown of Tables over their Activity Class
(Percentages over Total #Tables)

	Total #Tables	Activity Class			Activity Class (%)		
		RIGID	QUIET	ACTIVE	RIGID	QUIET	ACTIVE
Atlas	88	18	43	27	20%	49%	31%
BioSQL	45	16	13	16	36%	29%	36%
Castor	91	57	31	3	63%	34%	3%
SlashCode	68	15	38	15	22%	56%	22%
Zabbix	56	23	30	3	41%	54%	5%

Fig. 7. Distribution of Tables per Activity Class (for each data set, the **largest** value is in **red bold** and the *smallest* in *blue italics*)

4.2 Activity

The next issue that we are interested in is that of the update profile of the tables with respect to their topological categories. To ease the process of analyzing the update behavior of the tables with respect to their topological categories we utilize the activity classes defined in [11], which are summarized as follows:

- *Rigid* tables experience no updates throughout their entire life in the dataset.
- *Quiet* tables are tables with the total number of updates not exceeding the value of 5 and the Average Transitional Update (ATU) to be less than 0.1.
- *Active* tables are tables which undergo more than 5 updates and have an ATU higher than 0.1.

The *Average Transitional Update* (ATU) of a table is defined as the fraction of the sum of updates the table undergoes throughout its life over its duration [11]. The updates include attribute addition, deletion, change of data type and change of primary key.

Research Question: is there a relationship between the topological category of a table and its update activity?

Fig. 7 presents the distribution of the tables over the aforementioned activity classes. Tables tend to be mostly rigid and quiet. Next, we examine the impact of the topological categories of the update activity of the tables. The upper part of Fig. 8 depicts the probability for a table of a certain topological category to develop a certain update activity during its existence in its dataset. We outline the most interesting information derived from this figure in the following list:

- *Isolated* tables experience no or few updates with a probability that is higher than 82%. Overall: *isolated tables are mostly rigid and very rarely active!*
- The likelihood for a *source* table to undergo no or few changes throughout its life is at least 82% in all datasets, apart from BioSQL. *Source tables follow quite closely the overall pattern of the dataset, and they tend to be mostly quiet or rigid, and rarely active.*

- Activity shifts "rightwise" in Fig. 8, when it comes to *lookup* tables. Again, lookup tables are mostly quiet, but now, the odds are more in favor of being active, compared to the average behavior and compared to the probability of being rigid. In 3 out of 5 data sets, active lookup tables surpass 35%. *Overall, lookup tables are more prone to changes both with respect to categories of less topological complexity and with respect to the average behavior (which is expected, since the categories of low topological complexity are the most numerous ones).*
- *Internal* tables are mostly active! With the exception of Castor, having just one internal table, in all other cases, the majority of active internal tables is absolute. In other words, *the internal tables are expected to be mostly active, with a probability higher than in any other activity category!*

The bottom part of Fig. 8 presents the probability for a table with a certain activity profile to belong to a specific topological category. In a nutshell, we can identify the subsequent commonalities among the datasets:

- *Rigid tables are mostly isolated, or source, in the case where isolated tables do not really exist in the dataset.* The probability for a rigid table to be lookup is much lower compared to average and almost zero to be internal in 4 of the 5 data sets.
- It is fairly straightforward to observe that *the distribution of the quiet tables over the topological categories is in agreement with the aggregate one in all datasets.* In three of the five datasets, *quiet* tables are likely to belong to the *source* category, with the exceptions of Castor and SlashCode, in which *quiet* tables tend to be *isolated*.
- *Active tables are mostly inclined towards higher topological complexity.* In all data sets, (even in BioSQL where the distribution follows the average distribution of the entire data set very closely), the chances for an active table to belong to a topologically complex category are much higher than average. It is as if an attracting force is pulling active tables to the rightmost columns of higher topological complexity.

The statistical evidence provided by Chi-square and Fisher tests is fairly strong. We utilized a contingency table of four rows, each for a topological category, and three columns for the different activity classes. The p-values derived from these tests are below the critical value of 5% in four of the five datasets (except Biosql), ranging from 9.6E-05 (Zabbix) to 3.89E-02 (Castor). *The statistical results confirm that tables with different topological categories are subjects to different amounts of updates. Altogether, we established that the topological category of a table is related to its update activity. Isolated and source tables are inclined towards zero or few updates in their lifetime, lookup tables with few or many changes and internal tables with an inclination to active lives with many updates.*

Why do active tables change? The answer is that *they grow in schema size, i.e., in number of attributes.* We studied *schema resize* to see how attribute additions and deletions affect tables. In terms of their number of attributes, 2% -

PROBABILITY FOR A TABLE OF A TOPOLOGICAL CATEGORY TO DEVELOP A CERTAIN UPDATE ACTIVITY (PERCENTAGES OVER TOTAL #TABLES OF EACH TOPOLOGICAL CATEGORY)

Total #Tables	TOPOLOGICAL CATEGORY															
	ISOLATED				SOURCE				LOOKUP				INTERNAL			
	RIGID	QUIET	ACTIVE	Total #Tables	RIGID	QUIET	ACTIVE	Total #Tables	RIGID	QUIET	ACTIVE	Total #Tables	RIGID	QUIET	ACTIVE	Total #Tables
Atlas	27%	55%	18%	38	29%	58%	13%	32	13%	47%	41%	7	0%	0%	100%	88
BioSQL	100%	0%	0%	29	34%	31%	34%	8	25%	38%	38%	6	33%	17%	50%	45
Castor	67%	32%	1%	6	67%	17%	17%	9	33%	56%	11%	1	0%	100%	0%	91
SlashCode	34%	54%	11%	22	14%	68%	18%	7	0%	43%	57%	4	0%	25%	75%	68
Zabbix	55%	41%	5%	20	35%	65%	0%	11	27%	73%	0%	3	33%	0%	67%	56

PROBABILITY FOR A TABLE OF AN ACTIVITY CLASS TO BELONG TO A CERTAIN TOPOLOGICAL CATEGORY (PERCENTAGES OVER TOTAL #TABLES OF EACH ACTIVITY CLASS)

Total #Tables	ACTIVITY CLASS																	
	RIGID						QUIET						ACTIVE					
	ISOLATED	SOURCE	LOOKUP	INTERNAL	LOOKUP	INTERNAL	ISOLATED	SOURCE	LOOKUP	INTERNAL	LOOKUP	INTERNAL	ISOLATED	SOURCE	LOOKUP	INTERNAL		
Atlas	17%	61%	22%	0%	43	14%	51%	35%	0%	27	7%	19%	48%	13%	88			
BioSQL	13%	63%	13%	13	0%	69%	23%	8%	16	0%	63%	19%	4%	45				
Castor	88%	7%	5%	31	77%	3%	16%	3%	3	33%	33%	0%	82%	91				
SlashCode	80%	20%	0%	38	50%	39%	8%	3%	15	27%	27%	20%	51%	68				
Zabbix	52%	30%	13%	4%	30	43%	27%	0%	3	33%	0%	0%	39%	56				

Fig. 8. Probability for a Table of a Topological Category to Develop Specific Update Activity and vice versa

6% of tables shrink, 25% - 47% of tables increase, and 50% - 69% remain stable. Compared to the average probability of resize, we observe two different patterns that are consistent in all datasets: (a) *isolated* and *source* tables follow the average probability for size reduction, have higher probability for size steadiness and lower for size expansion, and, (b) *lookup* and *internal* tables have a potential for size reduction lower than the average with few exceptions, a probability for size steadiness below the average and a higher likelihood for size expansion.

5 An Unexpected Finding and Lessons Learned

The purpose of this research was to uncover patterns in the evolutionary behavior of tables with respect to their relationship with foreign keys, in order to derive useful knowledge on how developers evolve tables. In the process, we encountered patterns that were rather unexpected, although explainable in retrospect.

A topological hierarchy and its evolutionary behavior. After observing that different topological categories differ in their evolution, we came at an unexpected finding: there is *a hierarchy of topologically increasing complexity* reflected on how tables are evolved by developers of FOSS projects.

Top. Complexity Hierarchy : Isolated \rightarrow Source \rightarrow Lookup \rightarrow Internal (1)

We have discovered that the complexity spectrum that results from this hierarchy relates to the behavior of tables. On the high end of the complexity spectrum, the internal tables demonstrate quite a different life than the isolated tables at the other end of it. *Complex internal tables* demonstrate high activity –which means the undergo attribute additions, deletions and type updates– whereas *isolated* tables undergo very little if zero change. Remember that we are studying data sets with hundreds of commits spanning into several years of monitoring. At the very same time, internal tables are almost totally born at the earliest version of the database history: *in other words, there are no internal, topologically complex, and, probably active, tables born after the initiation of the database.* The phenomenon is quite opposite for isolated tables: despite the fact that a fair percentage of them is present in the original version of the database, isolated tables are the most likely to be added in subsequent versions of the history. *As time passes, it appears as if people are disinclined to add more complex structures to their database!* In-between the spectrum of isolated and internal tables, *source* tables appear to be more similar to the isolated ones, resisting change and being more likely to appear later in the life of a database, and *lookup* tables being more similar to the internal ones. Last but not least, let us mention that *isolated and source tables are the most populous categories whereas lookup and internal tables are progressively smaller in numbers.*

In a nutshell, this study reveals *the existence of a complexity spectrum ranging from (a) a populous, rigid, easily born, topologically simple end, all the way to (b) a less populous, active (due to attribute additions), early born (and not later), topologically complex end.*

Why is this happening? As also noted in the past [12], [8], the main force that seems to govern schema evolution, at least in the Free Open Source Software (FOSS) setting that we study, is *gravitation to rigidity*, due to the difficulty of altering the schema of a database when surrounding code is built upon it. The same seems to be observed here too: (a) *inactive, topologically simple tables are much more populous and easy to create than complex and active ones*; (b) *very few tables change topological category (Fig. 4), with most changes in the ephemeral or short-lasting categories of label-changes*; (c) *different topological categories seem to have different evolutionary behaviors* – specifically, most of the activity of the high-end of the complexity spectrum is due to the addition of attributes to the existing structures, quite differently from the lower end of the spectrum, where administrators are more inclined towards building new tables.

We conjecture that an explanation for this difference in behavior is the *avoid-to-break-the-code* principle: adding new information via new tables, which can later be removed if not useful, does not result in the necessity to update the surrounding code that queries and updates the existing tables. This leads to *maintenance-by-addition* and simplifies the life of developers, at the expense, of course, of increasing the size of the schema and fragmenting the information into many tables. So, developers augment the database with simple topologies, and if complex topologies need expansion, this is done via attribute injection. A second reason that we conjecture affects the evolutionary profile of tables, is the deployment of projects. Remember we are studying FOSS projects, built to be selected by other organizations. Once a FOSS project has been adopted and deployed by an organization, future upgrades might result in the change of the schema too. Upgrading the schema in the presence of existing data is a painful experience, and simple structures and maintenance-by-addition reduce this pain.

Guidelines based on our findings. Apart from advancing our knowledge with solid evidence, our empirical study on how schema evolution relates to foreign keys in FOSS projects addresses several audiences, as it provides both (a) maintenance clues to curators and evaluators of FOSS projects, and, (b) insights on the adaptability of the relational model to the research community.

Curators. Project curators can expect that the tendency of the schema in the future will be to expand in terms of (a) topologically simple structures and (b) injection of attributes to early-born, complex topological structures. Enforcing maintenance-by-addition will allow lower impact to the surrounding code. In the FOSS universe, where development is not as strictly controlled as in closed projects, it is necessary to reserve cycles and time for schema cleanup and application refactoring to avoid the unregulated management of the schema.

FOSS Evaluators. When selecting a software projects for adoption, an evaluator may use our toolset to analyze the schema history of the schema, in order to see how actively maintained it is, and via what kind of changes. An evaluator will need to also assess the threats posed by the absence of (a) foreign keys and (b) maintenance actions from the side of the curators.

Researchers. We believe that the main recipients for this line of work are the members of the research community, much more than the other categories.

The reason is that the nature of the situation boils down to the fundamentals of the relational model and how relational databases can be coupled to surrounding applications. We, as the research community, need to come up with more flexible ways of building applications on top of databases and/or tools that accurately highlight the points of maintenance in the surrounding code, in the event of 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* (2008)
3. 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)
4. 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)
5. Sjøberg, D.: Quantifying schema evolution. *Information and Software Technology* **35**(1), 35–44 (1993)
6. Skoulis, I., Vassiliadis, P., Zarras, A.: Open-source databases: Within, outside, or beyond Lehman’s laws of software evolution? In: *26th International Conference on Advanced Information Systems Engineering (CAiSE 2014)*, Thessaloniki, Greece, June 16-20, 2014. (2014)
7. Skoulis, I., Vassiliadis, P., Zarras, A.V.: Growing up with stability: How open-source relational databases evolve. *Information Systems* **53**, 363–385 (2015)
8. Vassiliadis, P., Kolozoff, M., Zerva, M., Zarras, A.V.: Schema evolution and foreign keys: Birth, eviction, change and absence. In: *Proceedings of 36th International Conference on Conceptual Modeling (ER 2017)*, Valencia, Spain, November 6-9, 2017. pp. 106–119 (2017)
9. Vassiliadis, P., Kolozoff, M., Zerva, M., Zarras, A.V.: Schema evolution and foreign keys: a study on usage, heartbeat of change and relationship of foreign keys to table activity. *Computing* **101**(10), 1431–1456 (2019)
10. Vassiliadis, P., Zarras, A.V.: Schema evolution survival guide for tables: Avoid rigid childhood and you’re en route to a quiet life. *Journal of Data Semantics* **6**(4), 221–241 (2017)
11. 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 (2015)
12. Vassiliadis, P., Zarras, A.V., Skoulis, I.: Gravitating to rigidity: Patterns of schema evolution - and its absence - in the lives of tables. *Information Systems* **63**, 24–46 (2017)
13. 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)