Do People Use Naming Conventions in SQL Programming?

Aggelos Papamichail[1], Apostolos V. Zarras[1], and Panos Vassiliadis[1]

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

Abstract. In this paper, we investigate the usage of naming conventions in SQL programming. To this end, we define a reference style, consisting of naming conventions that have been proposed in the literature. Then, we perform an empirical study that involves the database schemas of 21 open source projects. In our study, we evaluate the adherence of the names that are used in the schemas to the reference style. Moreover, we study how the adherence of the names to the reference style evolves, during the lifetime of the schemas. Our study reveals that many conventions are followed in all schemas. The adherence to these conventions is typically stable, during the lifetime of the schemas. However, there are also conventions that are partially followed, or even not followed. Over time, the adherence of the schemas to these conventions may improve, decay or remain stable.

Keywords: Naming conventions, Coding styles · SQL programming.

1 Introduction

Take a look at the code snippet that is given in Listing 1. It is a typical SQL table definition from the database schema of Joomla (Table 2). There are several naming issues that clutter the definition of the table. For instance, the name of the table, `‘#__menu‘`, begins with a sequence of special characters. Moreover, the table name and the column names are quoted. In general, the use of special characters and quotes in names is not considered a good practice, for compatibility and portability reasons [9, 11]. Several column names consist of multiple terms. Concerning readability, this practice is perfectly fine. However, the way of separating the terms is not consistent. For some multi-term names the terms are separated with underscores (e.g., `‘checked_out‘`, `‘checked_out_time‘`), other multi-term names are in camelCase (e.g., `‘browserNav‘`), while there are also multi-term names without any separation between the constituent terms (e.g., `‘menutype‘`, `‘utaccess‘`). Another possible readability problem is the use of acronyms in some column names (e.g., `‘lft‘`, `‘rgt‘`) [9, 11]. From a lexicographical point of view, table names are typically in plural or in some collective form, while column names are in singular form [9, 11]. However, in Listing 1 the name of the table is in singular form.

```
--
-- Table structure for table `#__menu`
--

CREATE TABLE `#__menu` (
  `id` int(11) NOT NULL auto_increment,
  `menutype` varchar(75) default NULL,
  `name` varchar(255) default NULL,
  `alias` varchar(255) NOT NULL default '',
  `link` text,
  `type` varchar(50) NOT NULL default '',
  `published` tinyint(1) NOT NULL default 0,
  `parent` int(11) unsigned NOT NULL default 0,
  `componentid` int(11) unsigned NOT NULL default 0,
  `sublevel` int(11) default 0,
  `ordering` int(11) default 0,
  `checked_out` int(11) unsigned NOT NULL default 0,
  `checked_out_time` datetime NOT NULL default '0000-00-00 00:00:00',
  `pollid` int(11) NOT NULL default 0,
  `browserNav` tinyint(4) default 0,
  `access` tinyint(3) unsigned NOT NULL default 0,
  `utaccess` tinyint(3) unsigned NOT NULL default 0,
  `params` text NOT NULL,
  `lft` int(11) unsigned NOT NULL default 0,
  `rgt` int(11) unsigned NOT NULL default 0,
  `home` INTEGER(1) UNSIGNED NOT NULL DEFAULT 0,
  PRIMARY KEY  (`id`),
  KEY `componentid` (`componentid`,`menutype`,`published`,`access`),
  KEY `menutype` (`menutype`)
) TYPE=MyISAM CHARACTER SET `utf8`;
```

Listing 1. A typical SQL table definition in Joomla.

Using appropriate naming conventions in source code is important for portability, readability and maintainability reasons [14]. In this paper, we investigate the use of naming conventions in SQL programming. Specifically, we perform an empirical study that involves 21 database schemas found in respective free and open source (FOSS) projects. To begin, we introduce *a reference style* that consists of a set of naming conventions, which have been proposed in the literature [9, 11]. Then, we focus on two issues: (1) *we assess the adherence of the names that are used in the schemas to the naming conventions of the reference style*; (2) *we investigate the evolution of the schemas, to see if the adherence of the names to the conventions improves, decays or remains stable*. To assess the adherence of a schema to the reference style *we developed a tool*, called *DBSea*, which is available as an open source project [1].

The rest of this paper is structured as follows. In Section 2, we discuss related work. In Section 3, we detail the reference naming style and the setup of our study. In Section 4, we present our findings. Finally, in Section 5 we conclude with a summary of our contribution and the future perspectives of this work.

2 Related Work

Several interesting empirical studies have been performed regarding the usage of names in source code. According to these studies, the usage of full word identifiers

[1] github.com/apapamichail/DBsea

improves source code readability [13, 4]. Typically, short identifiers take longer to understand [10]. Nevertheless, in some cases single letter identifiers may convey meaningful information [5]. The styles used for separating multi-term identifiers like CamelCase and underscores are also important for software comprehension [6]. The impact of each style varies depending on the development task and the developers' experience. Further research efforts study naming patterns and anti-patterns for classes, attributes, methods, variables and so on [7, 8, 3]. Moreover, there are studies that report naming patterns used in visual programming [16]. Another line of research, concerns techniques for the recommendation of class, variable and method names [1, 2, 12] that can be used to improve the readability of the code.

Differently from the aforementioned efforts, *in this paper we perform an empirical study that concerns the usage of naming conventions in SQL programming.*

3 Setup

In this section, we discuss in detail the naming conventions and the database schemas that we consider in our study.

3.1 Reference SQL naming style

The naming conventions that we consider come from Joe Celko's SQL programming style [9], Simon Holywell's SQL style guide [11], and the ISO-11179 naming standard. We do not claim that this list of conventions is complete, neither that it covers the in-house style of every possible organization. However, we believe it is a good starting point for our study as they come from 3 well-known sources that are not specific to any particular DBMS. We do not consider the assumed naming conventions as ground truth. Instead, we assess the extent to which they are actually used in practice.

Table 1, summarizes the naming conventions that we consider. In the table, each convention is introduced with a brief description and an acronym that we use to facilitate the visualization of the results. We categorize the conventions with respect to their scope, which can be tables and/or columns. Moreover, we categorize the conventions with respect to their purpose, which can be to facilitate portability, readability, and maintainability.

For portability reasons between different commercial and open source DBMSs it is better to start SQL elements names with letters (SWL) and end them with letters or numbers (EWL). In addition, it is better to avoid using special characters (ASC), spaces (AUS) and delimiters (AUD). Moreover, it is recommended to use names of a proper length (UPL), not exceeding respective standard upper bound limits. Celko provides a table with various identifier length limits that have been assumed in different DBMSs. Based on this table, the limit that we assume in our study is 30 characters.

Table 1. Reference style.

Purpose	Scope	Acronym	Mnemonic
Portability	Tables & Columns	SWL	Start With Letter
		EWL	End With Letter or number
		ASC	Avoid Special Characters
		AUS	Avoid Using Spaces
		AUD	Avoid Using Delimiters
		UPL	Use Proper Length
Readability, Maintainability		UTS	Uniform Term Separation
		UMW	Use More Words
		ACC	Avoid Camel Case
		ACU	Avoid Consecutive Underscores
		ARW	Avoid Reserved Words

Purpose	Scope	Acronym	Mnemonic
Readability, Maintainability	Tables only	SWC	Start With Capital
		TIP	Tables In Plural
		ACN	Avoid Concatenating Names
	Columns only	USP	Use Standardized Postfixes
		CIS	Columns In Singular
		DCN	Different Column Names
		ANP	Avoid Names by Place
		AII	Avoid "Id" as Identifier

For readability reasons, it is better to use a uniform term separation style (UTS) for multi-term names. To facilitate understanding, names that consist of multiple terms should contain more words than acronyms (UMW). According to [9], names in CamelCase should be avoided (ACC) because empirical evidence indicates that they disrupt the flow of reading, by making the eye concentrate on case changes. Similarly, the use of consecutive underscores (ACU) and reserved words (ARW) in names is not a good practice.

In the context of an SQL schema, tables are unique concepts that represent collections of related data. Therefore, table names should be treated like proper nouns, starting with a capital letter (SWC). As tables represent collections of related data, it is expected that table names should be in plural (TIP). It would also be good to avoid concatenating table names (ACN) to name relations between them. This is a common practice for naming relations, but the concatenated names do not reveal the purpose of the relation.

Concerning column names, when needed, it would be good to use standardized postfixes (USP); a list of such postfixes is provided in [9]. Columns, represent specific properties of the related data. Hence, it is expected that column names should be in singular form (CIS). Column names should be different from table names (DCN). Defining column names by place should also be avoided (NBP), in the sense that column names should not include table names as prefixes or suffixes. Moreover, using "id" to name primary keys is not a good practice (AII), because it does not reveal the purpose of the keys.

3.2 Database schemas

In our study we consider a well-established large collection of database schemas, the only available that comprises multiple schema versions. We have used this collection in previous studies to investigate the evolution of database schemas [15, 18, 17]. The collection consists of five scientific projects from CERN, two medical

Table 2. Schemas statistics.

Case Study		# Revisions	# Tables at		# Columns at		Domain
			First Known Version	Last Known Version	First Known Version	Last Known Version	
ATLAS	A particle physics experiment at CERN	84	73	56	709	857	Scientific
CASTOR	A hierarchical storage system for physics data	192	62	74	632	838	
SRM2	A client system for CASTOR.	58	11	11	54	84	
DQ2	A data management system for ATLAS.	54	10	26	116	184	
EGEE	A project that provides access to high-throughput	16	6	9	34	63	
Ensembl	A project that concerns genome databases.	528	19	75	82	486	Medical
BioSQL	A shared database for storing sequence data.	47	21	28	227	731	
Typo3	A CMS for managing any kind of digital content.	98	10	23	122	421	CMSs
PhpBB	An Internet forum package.	133	61	65	613	565	
PhpWiki	A wiki that supports multiple storage back-ends.	21	10	10	33	49	
SlashCode	The web site for All Things Slash.	398	42	87	259	610	
Zabbix	An enterprise network monitoring project.	27	47	48	312	313	
e107	A project for the creation of dynamic sites.	17	33	34	261	274	
Coppermine	A photo gallery project.	117	8	22	85	169	
DekiWiki	A platform for content and mashups.	16	28	40	204	315	
Nucleus	A simple CMS for web blogs.	4	20	20	110	112	
OpenCart	An e-commerce platform for online merchants.	165	48	114	74	230	
TikiWiki	A system for wikis, forums and blogs.	153	207	215	1528	1628	
XOOPS	A platform for community websites.	7	31	32	297	129	
MediaWiki	The platform of Wikimedia projects.	322	17	50	100	318	
Joomla	A project for publishing web content.	45	35	36	307	321	

projects and eleven CMS projects. Table 2, gives detailed statistics regarding the database schemas of the projects. Specifically, for each schema the table provides the number of versions it went through, the total number of tables and the total number of columns in the first and the last known versions of the schema. All the data sets are available at the web site of the DAINTINESS group [2].

4 Research Questions & Answers

In this section we discuss the findings of our study, organized with respect to the research questions that we investigate. To address our questions we define respective metrics that measure the adherence of the table/column names used in the examined schemas to the conventions of the reference style, and the way that the adherence of the names to the naming conventions evolves, during the lifetime of the schemas. Table 3, gives details about the basic notions that we assume for the definition of the metrics.

4.1 Is the reference style followed by the schemas?

To address our first research question, we define a simple metric, called *Adherence Indicator (AI)*.

[2] github.com/DAINTINESS-Group/EvolutionDatasets

Table 3. Basic notions and notation.

- $\Omega = \{S^1, S^2, \ldots, S^K\}$ is the overall set of schemas that we consider in our study.
- $R_{NC} = \{nc_1, nc_2, \ldots, nc_N\}$ is the set of naming conventions that constitute the reference naming style.
- $H^{S^j} = \{S^j_f, S^j_{f+1}, \ldots, S^j_\ell\}$ denotes a history of subsequent versions of a database schema $S^j \in \Omega$.
- $\Omega^\ell = \{S^1_\ell, S^2_\ell, \ldots, S^K_\ell\}$ is the set of the last known versions of the examined schemas Ω.
- $S^j_i.N_T$ is the set of table names, used in a schema version $S^j_i \in H^{S^j}$.
- $S^j_i.N_C$ is the set of column names, used in a schema version $S^j_i \in H^{S^j}$.
- $\Omega^\ell.N_T = \{S^1_\ell.N_T, S^2_\ell.N_T, \ldots, S^K_\ell.N_T\}$ denotes the sets of table names, used in the last known versions Ω^ℓ of the examined schemas.
- $\Omega^\ell.N_C = \{S^1_\ell.N_C, S^2_\ell.N_C, \ldots, S^K_\ell.N_C\}$ denotes the sets of column names, used in the last known versions Ω^ℓ of the examined schemas.
- $H^{S^j}_T = \{S^j_f.N_T, S^j_{f+1}.N_T, \ldots, S^j_\ell.N_T\}$ denotes the history of tables names, used throughout the history H^{S^j} of $S^j \in \Omega$.
- $H^{S^j}_C = \{S^j_f.N_C, S^j_{f+1}.N_C, \ldots, S^j_\ell.N_C\}$ denotes the history of column names, used throughout the history H^{S^j} of $S^j \in \Omega$.
- $\Omega^{H_T} = \{H^{S^1}_T, H^{S^2}_T, \ldots, H^{S^K}_T\}$, denotes the table names histories of the examined schemas Ω.
- $\Omega^{H_C} = \{H^{S^1}_C, H^{S^2}_C, \ldots, H^{S^K}_C\}$, stands for the column names histories of the examined schemas Ω.

The subscript **T|C** in the formulas (Definitions 1 to 5) indicates that a formula is applicable to tables (**T**) or columns (**C**). Equivalently, in the text we use the term **table/column** to express this property.

Definition 1. ***[Adherence Indicator]*** *The Adherence Indicator is a function $AI(S^j_i.N_{T|C}, nc)$ that takes as input a set of table/column names $S^j_i.N_{T|C}$, used in a schema version $S^j_i \in H^{S^j}$ of a schema $S^j \in \Omega$, and a naming convention $nc \in R_{NC}$. The value of the function gives the percentage of the table/column names that adhere to nc. More formally, $AI(S^j_i.N_{T|C}, nc) = \frac{|S^j_i.N_A|}{|S^j_i.N_{T|C}|} * 100\%$, where $S^j_i.N_A$ is the subset of $S^j_i.N_{T|C}$ that adhere to nc.*

We focus our analysis on the last known versions Ω^ℓ of the examined schemas. Later, (Section 4.2) we show that these versions are representative of the schemas' histories Ω^H.

To begin our analysis, we consider the reference style as a whole. Specifically, our goal is to determine the *adherence of table/column names to the overall style.* To achieve this goal, we calculate the values of $AI(S^j_\ell.N_{T|C}, nc)$ for the naming conventions of the reference style R_{NC} and the sets of table/column names $\Omega^\ell.N_{T|C}$. For each set of table/column names $S^j_\ell.N_{T|C}$, we partition the naming conventions of R_{NC} in three subsets, $P_{S^j_\ell.N_{T|C}} = \{CF, PF, NF\}$, containing the naming conventions that are completely followed ($AI(S^j_\ell.N_{T|C}, nc) = 100\%$),

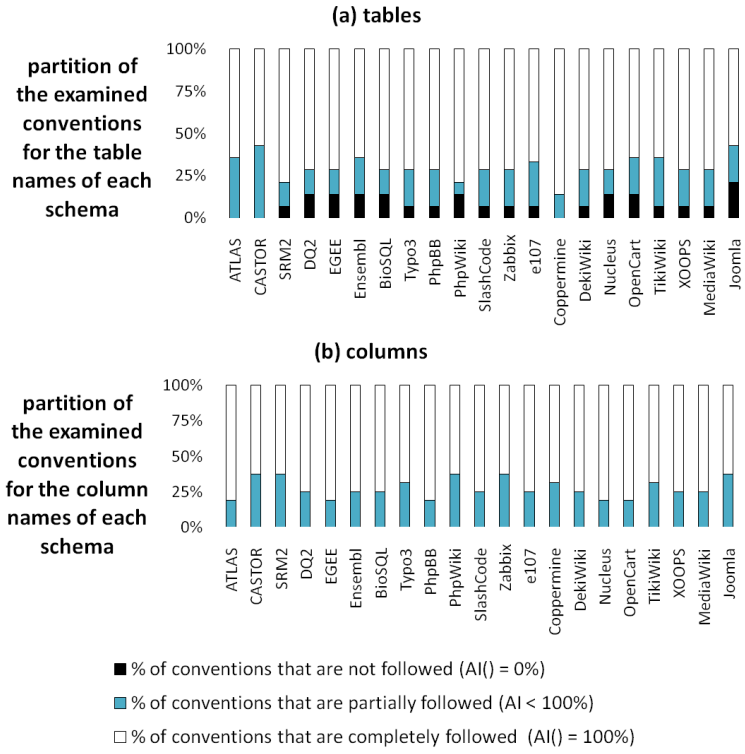

Fig. 1. Adherence of the names used in the schemas to the reference naming style.

partially followed ($AI(S^j_\ell.N_{T|C}, nc) < 100\%$), not followed ($AI(S^j_\ell.N_{T|C}, nc) = 0\%$) by $S^j_\ell.N_{T|C}$, respectively.

Figure 1, shows the results that we obtain. Specifically, for the sets of table/-column names $\Omega^\ell.N_{T|C}$ the figure provides respective stacked bars, describing the partitions of R^{CI}_{NC}. A stacked bar is divided in three parts, each giving the percentage[3] of conventions that belong to a partition subset.

In the results, we observe that *the names used in the schemas do not follow the reference style faithfully.* On the positive side, many conventions are completely followed. Regarding tables, the percentage of naming conventions that are completely followed is higher than 62%, in all schemas. As for columns, the respective percentage of naming conventions is higher than 57%, in all schemas. On the negative side, *several conventions are partially followed* and *few others are not followed at all.* Concerning tables, the percentage of naming conventions

[3] Due to the lack of space we use percentages to give an overview of the results. The complete raw results that we obtained in our study can be found in www.cs.uoi.gr/~zarras/SQLNamingConventions/SQLStatisticsSLA.rar

that are partially followed ranges from 7.14% to 42.8%, while the percentage of conventions that are not followed varies from 0% to 21.43%. Regarding columns, the percentage of naming conventions that are partially followed ranges from 18.75% to 37.50%.

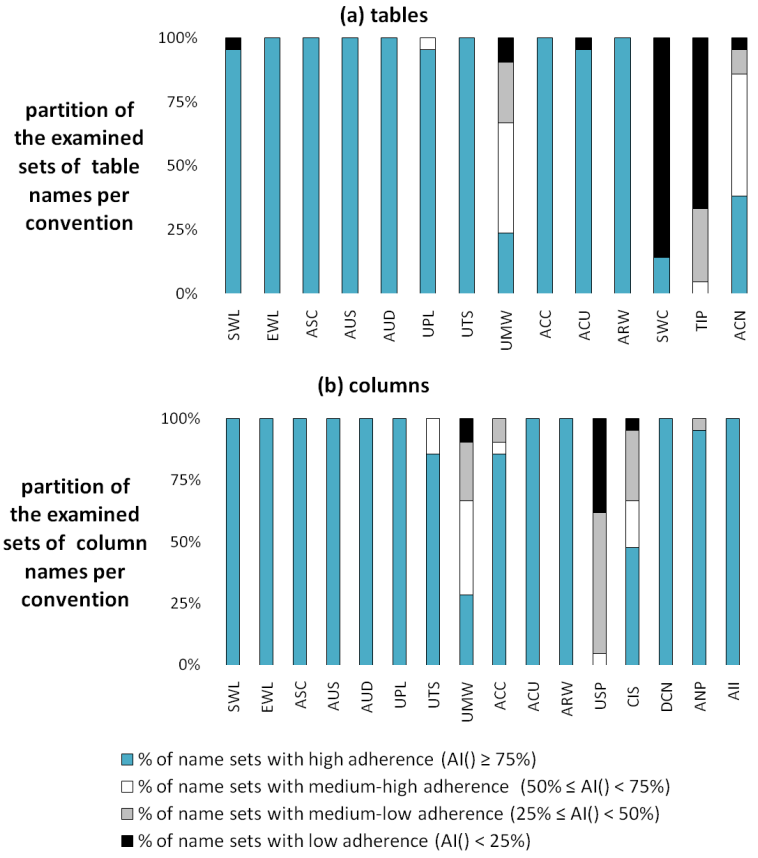

Fig. 2. Adherence of the names used in the schemas to each naming convention.

Next, we focus our analysis on the individual naming conventions. Our objective is to assess the *adherence of table/column names to each naming convention*. To address this issue, for each naming convention $nc \in R_{NC}$ we partition the examined sets of table/column names $\Omega^{\ell}.N_{T|C}$ in four subsets, $P_{nc} = \{A_H, A_{MH}, A_{ML}, A_L\}$, containing the sets of tables/column names that have high ($AI(S^j_{\ell}.N_{T|C}, nc) \geq 75\%$), medium-high ($50\% \leq AI(S^j_{\ell}.N_{T|C}, nc) < 75\%$), medium-low ($25\% \leq AI(S^j_{\ell}.N_{T|C}, nc) < 50\%$), low ($AI(S^j_{\ell}.N_{T|C}, nc) < 25\%$) adherence to nc.

Figure 2, shows the results that we obtain. In particular, for the naming conventions of the reference style, the figure provides corresponding stacked bars, describing the partitions of the examined sets of table/column names $\Omega^{\ell}.N_{T|C}$. A stacked bar is divided in four parts, each giving the percentage of the sets of table/column names that belong to a partition subset.

In general, we observe *high adherence of the sets of table/column names to most naming conventions*. Regarding tables, the percentage of name sets that belong to A_H is higher than 95.24%, in 10 out of 14 conventions. As for columns, the percentage of name sets that belong to A_H is higher than 85.71%, in 13 out of 16 conventions. Concerning tables, the exceptions are UMW, SWC, TIP and ACN. Regarding columns, the exceptions are UMW, USP and CIS. All of these conventions concern the readability of the schemas. In particular, the table/column names that are used in the schemas may comprise more acronyms than words (UMW). Moreover, the schemas may contain concatenated table names (ACN), table names are not in plural form (TIP) and/or table names that do not begin with capital letters (SWL). Similarly, the schemas may contain column names that are not in singular form (CIS) and/or column names with non standardized postfixes (USP).

4.2 Does the adherence of the schemas to the reference style evolve?

Having some clear evidence of adherence to the reference style, we move to the next issue that we consider in our study. We investigate the adherence of the table/column names to the reference style, with respect to the history of the examined schemas.

Specifically, we check if the adherence of the table/column names improves, decays or stays the same, between the first and the last known versions of the examined schemas. For this purpose, we employ the *Adherence Progress Indicator (API)* metric, defined below.

Definition 2. *[**Adherence Progress Indicator**] We define the Adherence Progress Indicator as a function $API(H^{S^j}_{T|C}, nc)$ that takes as input the history of table/column names $H^{S^j}_{T|C}$, used in $S^j \in \Omega$, and a naming convention $nc \in R_{NC}$. The value of the function is the difference between the value of the Adherence Indicator function for the names $S^j_{\ell}.N_{T|C}$, used in the last known version $S^j_{\ell} \in H^{S^j}$ of S^j, and the names $S^j_f.N_{T|C}$, used in the first known version $S^j_f \in H^{S^j}$ of S. Formally, $API(H^{S^j}_{T|C}, nc) = AI(S^j_{\ell}.N_{T|C}, nc) - AI(S^j_f.N_{T|C}, nc)$.*

We calculate the values of $API(H^{S^j}_{T|C}, nc)$ for the naming conventions R_{NC} and the table/column names histories $\Omega^{H_{T|C}}$ that we consider in our study. For each naming convention $nc \in R_{NC}$ we partition the histories in three subsets $P^{H_{T|C}}_{nc} = \{A_I, A_S, A_D\}$, containing histories of table/column names with improved ($API(H^{S^j}_{T|C}, nc) > 0\%$), stable ($API(H^{S^j}_{T|C}, nc) = 0\%$) and decayed ($API(H^{S^j}_{T|C}, nc) < 0\%$) adherence to nc, respectively.

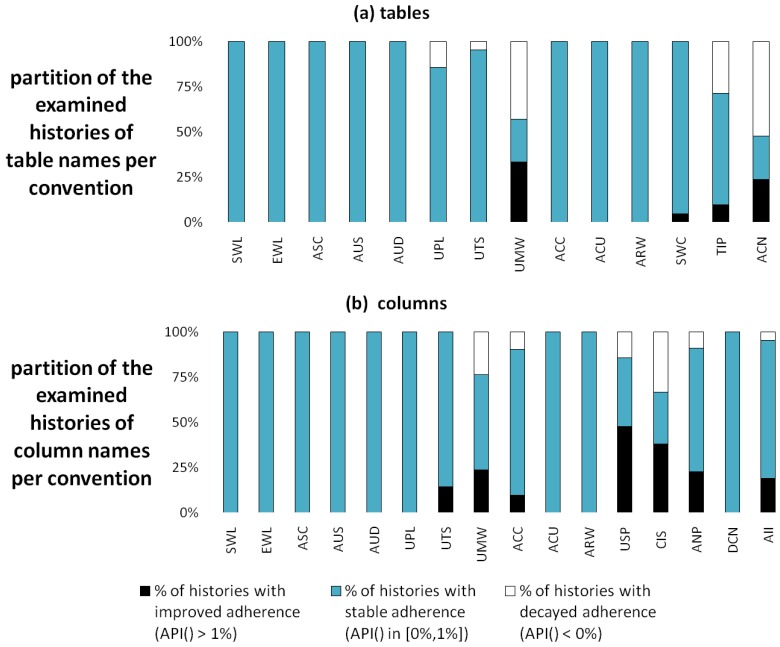

Fig. 3. Progress of adherence to each naming convention.

Figure 3 gives the results that we obtain. For the naming conventions of the reference style, the figure provides corresponding stacked bars, describing the partitions of the examined table/column names histories $\Omega^{H_{T|C}}$. A stacked bar is divided in three parts, each giving the percentage of histories that belong to a partition subset.

In the results we see that *the adherence of the examined table/column names to most of the naming conventions is stable*. Regarding tables, the percentage of histories that belong to A_S is higher than 85.71%, in 11 out of 14 conventions. As for columns, the percentage of histories that belong to A_S is higher than 71.43%, in 14 out of 16 conventions.

The adherence of table/column names to the rest of the naming conventions may *improve*, *decay*, or *remain stable*. Specifically, in the largest percentage of histories, the adherence of table names to UMW and ACN decays (Figure 3(a)). Nevertheless, we also observe considerable percentages of histories with improved and stable adherence. In the case of TIP, the adherence of the table names is stable in a large percentage of histories. However, there is also a notable percentage of histories with decayed adherence, and a small percentage of histories with improved adherence. In the largest percentage of histories, the adherence of column names to USP and CIS improves, while there are also notable percentages of histories with stable and decayed adherence. (Figure 3(b)).

4.3 Threats to Validity

A possible threat to the *construct validity* of our study is deficiencies of the tool that we used for the assessment of the examined schemas. To cope with this threat, we developed DBSea based on well-known open source libraries and tools (WordNet[4], Apache Commons Math[5], ANTLR[6]). For the validation of the tool, we developed an extensive set of unit tests that covers the naming conventions of the reference style. Moreover, we manually checked the correctness of DBSea by inspecting random samples of the collected data. *Internal validity*, is not an issue in our study, as we do not attempt to establish any particular cause-effect relationships.

Regarding *external validity*, our study has been conducted in a well-defined context, database schemas used in FOSS. We studied a reasonable number of schemas with variance in the respective fields of use. The schemas also vary in size and number of versions. Thus, we believe that the examined schemas are representative for the case of open source projects. Nevertheless, studying more schemas from open source and industrial projects, may reveal further interesting observations.

5 Conclusion

In this paper, we defined a reference style consisting of naming conventions that have been proposed in the literature. Then, we assessed whether these conventions are used in practice in a study that involved 21 schemas used in respective FOSS projects. We observed that many conventions are followed in all schemas, but there are also conventions that are partially followed, or not followed at all. During the lifetime of the schemas, the adherence to the conventions that are generally followed is stable, while the adherence to the rest of the conventions may improve, decay, or remain stable.

Our study is a starting point towards the investigation of further issues concerning the usage of naming conventions in SQL programming. For instance, it would be interesting to examine why some projects follow certain conventions more often than others. Another possible issue is to find reasons that make developers deviate from naming conventions. Using good naming practices and conventions is a basic prerequisite for the development of clean SQL code. Nevertheless, it is not the only one; the structure of the code is also important. Looking for best practices, patterns, and quality metrics in this context is an interesting issue for future research. Another interesting research direction concerns tools and techniques for the refactoring of SQL code.

Acknowledgements We would like to thank the anonymous reviewers for their useful suggestions and comments.

[4] wordnet.princeton.edu
[5] commons.apache.org/proper/commons-math/
[6] www.antlr.org/

References

1. Allamanis, M., Barr, E.T., Bird, C., Sutton, C.A.: Learning Natural Coding Conventions. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE). pp. 281–293 (2014)
2. Allamanis, M., Barr, E.T., Bird, C., Sutton, C.A.: Suggesting Accurate Method and Class Names. In: Proceedings of the Joint 23rd ACM SIGSOFT Symposium on the Foundations of Software Engineering and 15th European Software Engineering Conference (FSE/ESEC). pp. 38–49 (2015)
3. Arnaoudova, V., Penta, M.D., Antoniol, G.: Linguistic Antipatterns: What They Are and How Developers Perceive Them. Empirical Software Engineering **21**(1), 104–158 (2016)
4. Avidan, E., Feitelson, D.G.: Effects of Variable Names on Comprehension an Empirical Study. In: Proceedings of the 25th International Conference on Program Comprehension (ICPC). pp. 55–65 (2017)
5. Beniamini, G., Gingichashvili, S., Klein-Orbach, A., Feitelson, D.G.: Meaningful Identifier Names: The Case of Single-Letter Variables. In: Proceedings of the 25th International Conference on Program Comprehension (ICPC). pp. 45–54 (2017)
6. Binkley, D., Davis, M., Lawrie, D., Maletic, J.I., Morrell, C., Sharif, B.: The Impact of Identifier Style on Effort and Comprehension. Empirical Software Engineering **18**(2), 219–276 (2013)
7. Butler, S.: Mining Java Class Identifier Naming Conventions. In: Proceedings of the 34th IEEE-ACM-SIGSOFT International Conference on Software Engineering (ICSE). pp. 1641–1643 (2012)
8. Butler, S., Wermelinger, M., Yu, Y.: A Survey of the Forms of Java Reference Names. In: Proceedings of the 23rd IEEE International Conference on Program Comprehension, (ICPC). pp. 196–206 (2015)
9. Celko, J.: SQL Programming Style. Morgan-Kaufmann (2005)
10. Hofmeister, J.C., Siegmund, J., Holt, D.V.: Shorter Identifier Names Take Longer to Comprehend. Empirical Software Engineering **24**(1), 417–443 (2019)
11. Holywell, S.: SQL Style Guide. www.sqlstyle.guide
12. Kashiwabara, Y., Onizuka, Y., Ishio, T., Hayase, Y., Yamamoto, T., Inoue, K.: Recommending Verbs for Rename Method Using Association Rule Mining. In: Proceedings of the 21st IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER). pp. 323–327 (2014)
13. Lawrie, D., Morrell, C., Feild, H., Binkley, D.: What's in a Name? A Study of Identifiers. In: Proceeedings of the 14th IEEE International Conference on Program Comprehension (ICPC). pp. 3–12 (2006)
14. Martin, R.C.: Clean Code - A Handbook of Agile Software Craftsmanship. Prentice Hall (2009)
15. Skoulis, I., Vassiliadis, P., Zarras, A.V.: Growing Up with Stability: How Open-Source Relational Databases Evolve. Information Systems **53**, 363–385 (2015)
16. Swidan, A., Serebrenik, A., Hermans, F.: How do Scratch Programmers Name Variables and Procedures? In: 17th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM). pp. 51–60 (2017)
17. 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)
18. 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)