SCHEMA EVOLUTION IN WIKIPEDIA
Toward a Web Information System Benchmark[*]

Carlo A. Curino[1], Hyun J. Moon[2], Letizia Tanca[1] and Carlo Zaniolo[2]
[1]*DEI, Politecnico di Milano, Italy*
[2]*CSD, UCLA, Los Angeles, CA, U.S.A.*

Keywords: Schema evolution, Wikipedia, Case Study, Benchmark.

Abstract: Evolving the database that is at the core of an Information System represents a difficult maintenance problem that has only been studied in the framework of traditional information systems. However, the problem is likely to be even more severe in web information systems, where open-source software is often developed through the contributions and collaboration of many groups and individuals. Therefore, in this paper, we present an in-depth analysis of the evolution history of the Wikipedia database and its schema; Wikipedia is the best-known example of a large family of web information systems built using the open-source software MediaWiki. Our study is based on: (i) a set of Schema Modification Operators that provide a simple conceptual representation for complex schema changes, and (ii) simple software tools to automate the analysis. This framework allowed us to dissect and analyze the 4.5 years of Wikipedia history, which was short in time, but intense in terms of growth and evolution. Beyond confirming the initial hunch about the severity of the problem, our analysis suggests the need for developing better methods and tools to support graceful schema evolution. Therefore, we briefly discuss documentation and automation support systems for database evolution, and suggest that the Wikipedia case study can provide the kernel of a benchmark for testing and improving such systems.

1 INTRODUCTION

Every Information System (IS) is the subject of a constant evolution process to adapt the system to many factors such as changing requirements, new functionalities, compliance to new regulations, integration with other systems, and new security and privacy measures. The data management core of an IS is one of the most critical portions to evolve. Often based on Relational DataBase (DB) technology, the data management core of a system needs to evolve whenever the revision process requires modifications in the logical and physical organization of the data. Given its fundamental role, the evolution of the DB underlying an IS has a very strong impact on the applications accessing the data; thus, support for graceful evolution is of paramount importance. The complexity of DB and software maintenance, clearly, grows with the size and complexity of the system. Furthermore, when moving from intra-company systems – typically managed by rather small and stable teams of developers/administrators – to collaboratively-developed-and-maintained public systems, the need for a well-managed evolution becomes indispensable. Leading-edge web projects, characterized by massive collaborations and fast growth, experience a relentless drive for changes, which in turn generates a critical need for widespread consensus and rich documentation. Schema evolution has been extensively studied in the scenario of traditional information systems. An authoritative and comprehensive survey of the approaches to relational schema evolution and schema versioning is presented in (Roddick, 1995). More recently, (Ram and Shankaranarayanan, 2003) has surveyed schema evolution on the object-oriented, relational, and conceptual data models. Case studies on schema evolution on various application domains appear in (Sjoberg, 1993; Marche, 1993). Schema evolution has also been studied in the context of *model management* – research which aims at developing a systematic approach to schema management and mapping (Bernstein, 2003). Other interesting approaches tackled the problem of schema evolution in XML (Moro et al., 2007), data warehouse (Rizzi and Golfarelli, 2007) and object-oriented databases (Galante et al., 2005; Franconi et al., 2001).

Of particular interest, are Web Information Systems (WIS), often based on open-source solutions.

[*]This work has been partially funded by the project MIUR-FIRB ARTDECO and the NSF project IIS 0705345.

This large and fast-growing class include, among many other examples: Content Management Systems, Wiki-based web portals, E-commerce systems, Blog, and Public Scientific Databases from 'Big Science' Projects. The common denominator among these systems is the collaborative and distributed nature of their development and content management. Among the best known examples we have: *MediaWiki*[1], a website software underlying a huge number of web portals, including Wikipedia [2], *Joomla*[3], a complete Content Management System (CMS) and Web Application Framework, and *TikiWiki*[4], an open source groupware and CMS solution.

Moreover, inasmuch as large collaborative projects are now very common in natural science research, their reliance on databases and web systems as the venue needed to promptly shared results and data has created many large Scientific Databases, including the Human Genome DB[5], HGVS[6], and many others. Although different in many ways, these all share a common evolution problem for which the slow labor-intensive solutions of the past have become inadequate. New conceptual and operational tools are needed to enable graceful evolution by systematically supporting the migration of the DB and the maintenance of its applications. Among the desiderata in such a scenario, we seek systems that preserve and manage the past contents of a database and the history of its schema, while allowing legacy applications to access new contents by means of old schemas (Moon et al., 2008; Curino et al., 2008b).

In the rest of this paper, we shall analyze the case of MediaWiki, a data-intensive, open-source, collaborative, web-portal software, originally developed to run Wikipedia, a multilingual, web-based, free-content encyclopedia: this platform is currently used by over 30,000 wikis, for a grand total of over 100 million pages[7]. While the Wikipedia content evolution has been analyzed previously (Almeida et al., 2007), this report is the first that focuses on the problem of DB schema evolution. MediaWiki has seen, during its 4 years and 7 months of life, 171 different DB schema versions released to the public by means of a CVS/Subversion versioning system[8]. As one can easily imagine, every schema change has a profound impact on the application queries and the code managing the results, which must thus be revised. In the case of MediaWiki, we observed in our analysis that only a small fraction (about 22%) of the queries designed to run on old schema versions are still valid throughout the schema evolution (see discussion in Section 3.4).

Our analysis was made possible by the collaborative, public, and open-source nature of the development, documentation and release of MediaWiki and Wikipedia.

Contributions. The main contributions of this paper are the following: (i) we present the first schema evolution analysis of a real-life Web Information System DB, by studying the MediaWiki DB backend. This provides a deep insight on Wikipedia, one of the ten most popular websites to date[9] and reveals the need for DB schema evolution and versioning techniques, and (ii) we provide and plant the seeds of the first public, real-life-based, benchmark for schema evolution, which will offer to researchers and practitioners a rich data-set to evaluate their approaches and solutions. As a part of the benchmark, we also release a simple but effective tool-suite for evolution analysis.

The paper is organized as follows: Section 2 presents the MediaWiki architecture, Section 3 discusses statistics on the MediaWiki schema evolution, Section 4 presents the tool-suite we developed to carry out such analysis. Section 5 shows how this analisys is contributing to the definition of a unified benchmark for schema evolution. Section 6 is devoted to related works, and Section 7 draws our conclusions.

2 MEDIAWIKI

In this section we briefly discuss the MediaWiki software architecture and DB schema (as in the current version of November 2007[10]), to provide the reader with a broad understanding of the internals of the system we are going to analyze.

2.1 Architecture

The MediaWiki software is a browser-based web-application, whose architecture is described in details in (Wikimedia Foundation, 2007). As shown in Figure 1, the users interact with the PHP frontend through a standard web browser, submitting a page

[1] Available at: http://www.mediawiki.org.

[2] Available at: http://wikipedia.org.

[3] Available at http://www.joomla.org.

[4] Available at http://www.tikiwiki.org.

[5] Available at http://www.gdb.org/.

[6] Available at http://www.hgvs.org/index.html

[7] See http://s23.org/wikistats/.

[8] See: http://svn.wikimedia.org/viewvc/mediawiki/trunk-/phase3/maintenance/tables.sql.

[9] Source: http://www.alexa.com.

[10] The current version is the 171st schema version corresponding to the SVN commit revision 25635.

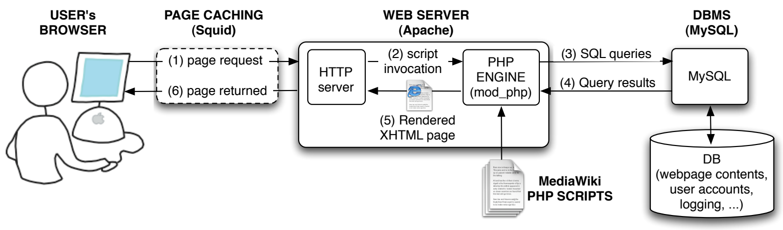

Figure 1: MediaWiki Software Architecture.

request (e.g., a search for pages describing "Paris"). The frontend software consists of a simple presentation and management layer (MediaWiki PHP Scripts) interpreted by the Apache PHP engine. The user requests are carried out by generating appropriate SQL queries (or updates), that are then issued against the data stored in the backend DB (e.g., the database is queried looking for article's text containing the term "Paris"). The backend DB can be stored in any DBMS: MySQL, being open-source and scalable, is the default DBMS for the MediaWiki software. The results returned by the DBMS are rendered in XHTML and delivered to the user's browser to be displayed (e.g., a set of of links to pages mentioning "Paris" is rendered as an XHTML list). Due to the heavy load of the Wikipedia installation of this software, much of effort has been devoted to performance optimization, introducing several levels of caching (Rendered Web Page, DB caches, Media caches), which is particularly effective thanks to the very low rate (0.04%) of updates w.r.t. queries. Obviously, every modification of the DB schema has a strong impact on the queries the frontend can pose. Typically each schema evolution step can require several queries to be modified, and so several PHP scripts (cooperating to interrogate the DB and render a page) to be manually fixed, in order to balance the schema changes.

2.2 Database Schema

The DB, in the current version, presents 34 tables with, all in all, 242 columns. It holds the entire website content, over 700 GBytes in the case of Wikipedia. The tables can be functionally grouped as follows:
Article and content management (6): `page`, `revision`, `text`, `image`, `user_newtalk`, `math`.
History and archival management (4): `archive`, `filearchive`, `oldimage`, `logging`.
Links and website structure (9): `categorylinks`, `externallinks`, `imagelinks`, `interwiki`, `langlinks`, `pagelinks`, `redirect`, `templatelinks`, `trackbacks`.
Users and permissions (5): `user`, `user_groups`, `ipblocks`, `watchlist`, `page_restrictions`.
Performance and caching (7): `objectcache`, `querycache`, `querycache_info`, `job`, `querycachetwo`, `transcache`, `searchindex`.
Statistics and special features support (3): `recentchanges`, `hitcounter`, `site_stats`.

Note the presence of many tables devoted to performance tuning, by means of caching and indexing, and to preservation of deleted or historical copies of the system's main content, e.g., articles and images.

3 SCHEMA EVOLUTION IN MEDIAWIKI

In this section, we analyze the schema evolution of MediaWiki based on its 171 schema versions, as committed to SVN between April 2003 (first schema revision) and November 2007 (date of this analysis).

3.1 Basic Statistics

Schema Size Growth. In Figure 2 and 3, we report the size of MediaWiki DB schema in history, in terms of the number of tables and columns, respectively. The graphs show an evident trend of growth in sizes, where the number of tables has increased from 17 to 34 (100% increase) and the number of columns from 100 to 242 (142%). Sudden drops in the graphs are due to schema versions with syntax errors, i.e., schema versions that could not be properly installed. In both graphs we observe different rates of growth over time, which seem to be related to the time periods preceding or following official releases of the overall software (see Table 1).

Schema growth is due to three main driving forces as follows: (i) performance improvement, e.g., introduction of dedicated cache tables, (ii) addition of new features, e.g., support for logging and content validation, and (iii) the growing need for preservation of DB content history, i.e., introduction of tables and columns to store outdated multimedia content such as

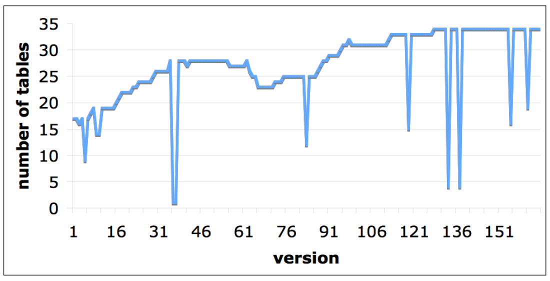

Figure 2: MediaWiki Schema Size: the Number of Tables.

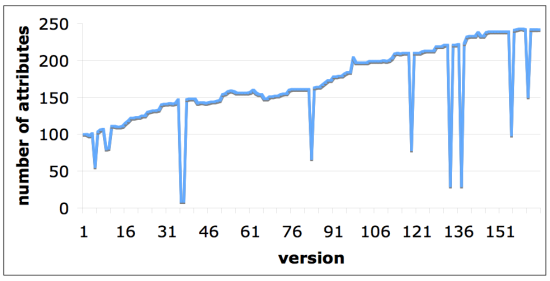

Figure 3: MediaWiki Schema Size: the Total Number of Columns.

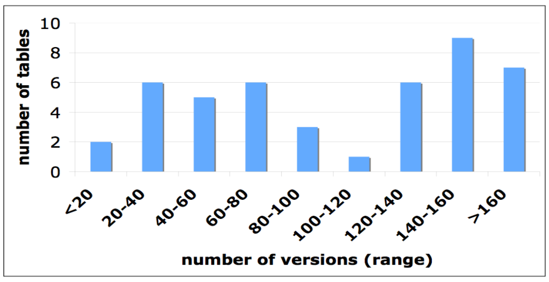

Figure 4: Histogram of Table Lifetime.

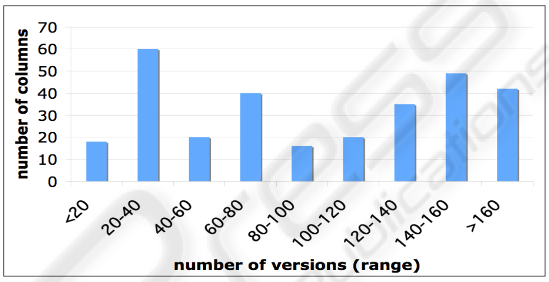

Figure 5: Histogram of Column Lifetime.

the “filearchive” table.

Table/Column Lifetime. Figure 4 shows a histogram representation of the table lifetimes, in terms of number of versions. The lifetimes range from very long ones, e.g., the *user* table that was alive throughout the entire history, to short ones, e.g., *random* table that only survived for two revisions. On average, each table lasted 103.3 versions (60.4% of the total DB history). Figure 5 presents lifetimes of columns in histogram, where columns lasted 97.17 versions on average (56.8% of the total DB history). Interestingly, both figures show that there are two main groups of tables and columns: “short-living” and “long-living”. The former might be due to the fact that the schema has been growing lately so a significant portion of tables and columns has been introduced only recently. The latter can be explained noting that the core tables/columns tend to be rather stable throughout the entire history.

Per-month Revision Count. In Figure 6, we show how many schema versions were committed during each month in history, providing an estimation of the development effort devoted to the DB backend over time.

3.2 Macro-Classification of Changes

We group the 170 evolution steps based on the types of evolution they present as in Table 2. While the “actual schema changes” have an impact on the queries, as they modify the schema layout, the evolution of the DBMS engine, indexes, and data types, (while being relevant to performance) does not require any query correction, because of the physical data-independence provided by the DBMS. Table 2 shows the frequencies[11] of the types of changes among the 170 evolution steps. In particular, the table highlights that:

(i) almost 55% of the evolution steps involve *actual schema changes* (further discussed in Section 3.3); (ii) over 40% of the evolution steps involve index/key adjustments and this is due to the performance-critical role of the DB in a data-intensive, high-load, website such as Wikipedia; (iii) 8.8% of the evolution steps were rollbacks to previous schema versions; (iv) 7.6% of the analyzed evolution steps present *only* documentation changes.

3.3 Micro-Classification of Changes

Schema Modification Operators. To better understand the Relational DB schema evolution, we introduce a classification of the “actual schema changes”. Different formalisms can be exploited for this purpose. Shneiderman and Thomas proposed in (Shneiderman and Thomas, 1982) a comprehensive set of schema changes, including structural schema changes and also changes regarding the keys and dependencies. More recently, Bernstein et al. have also

[11] Please note that each evolution step might contain more than one type of change.

Table 1: MediaWiki Software Releases and the Number of DB Schema Versions Immediately Preceding Each Release.

software releases	release date	schema version used (ordinal)	# of schema versions
1.1	Dec 8, 2003	7	7
1.2	Mar 24, 2004	14	7
1.3	Aug 11, 2004	28	14
1.4	Mar 20, 2005	48	20
1.5	Oct 5, 2005	79	31
1.6	Apr 5, 2006	93	14
1.7	Jul 7, 2006	102	9
1.8	Oct 10, 2006	110	8
1.9	Jan 10, 2007	127	17
1.10	May 9, 2007	145	18
1.11	Sep 10, 2007	171	26

Table 2: Macro-Classification of Schema Changes (One evolution step may have more than one change type).

Type of Change	# of evolution steps	% of evolution steps
Actual Schema	94	54.9%
Index/Key	69	40.3%
Data Type	22	12.8%
Syntax Fix	20	11.7%
Rollback	15	8.8%
Doc Only	13	7.6%
Engine	6	3.5%

proposed a set of schema evolution primitives using algebra-based constraints as their primitives (Bernstein et al., 2006).

Among several options, we chose the Schema Modification Operators (SMOs) that we proposed in (Moon et al., 2008; Curino et al., 2008b) (briefly described in Table 3). SMOs capture the essence of the existing works, but can also express schema changes not modeled by previous approaches. For example, by using function[12] in the `ADD COLUMN` operator, SMOs can support semantic conversion of columns (e.g., currency exchange), column concatenation/split (e.g., different address formats), and other similar changes that have been heavily exploited in modeling MediaWiki schema changes. The effectiveness of SMOs have been validated in (Moon et al., 2008; Curino et al., 2008b), where the PRISM and PRIMA systems used SMOs to describe schema evolution in transaction-time databases and to support historical query reformulations over multi-schema-version transaction-time databases.

The syntax of SMO is similar to that of SQL DDL, and provides a concise way to describe typical modifications of a database schema and the corresponding data migration. Every SMO takes as input a schema

[12]Both from system libraries and user defined.

Table 3: Schema Modification Operators (SMOs).

SMO	Description
`CREATE TABLE`	introduces a new, empty table to the database
`DROP TABLE`	removes an existing table from the schema
`RENAME TABLE`	changes a table name.
`DISTRIBUTE TABLE`	takes as input a source table and distribute tuples into two newly generated tables, according to the specified conditions.
`MERGE TABLE`	takes two source tables with the same schema and creates a new table storing their union.
`COPY TABLE`	creates a duplicate of an existing table
`ADD COLUMN`	introduces a new column into the specified table. The new column is filled with values generated by a user-defined constant or function.
`DROP COLUMN`	removes an existing column from a table.
`RENAME COLUMN`	changes the name of a column.
`COPY COLUMN`	makes a copy of a column into another table, filling the value according to a join condition between source and target tables.
`MOVE COLUMN`	same COPY COLUMN but the original column is dropped.

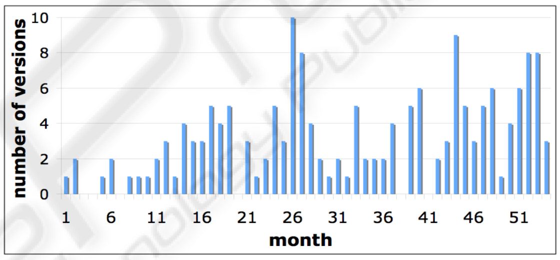

Figure 6: Schema Versions Committed during Each Month.

and produces as output a new version of the same schema. Table 3 presents a list of SMOs, operating on tables (the first six) and on columns (the last five) of a given DB schema, together with a brief explanation. Note that SMOs can be arbitrarily combined to describe complex structural changes, as those occured in the MediaWiki DB schema evolution.

Classification using SMOs. In this context we exploit SMOs as a pure classification instrument to provide a fine-grained analysis of the types of change the schema has been subject to. While there might be several ways to describe a schema evolution step by means of SMOs, we carefully select, analyzing the available documentation, the most natural set of SMOs describing each schema change in the MediaWiki history. Table 4 shows the distribution of the SMOs, presenting, for each type, how many times it has been used in the entire schema evolution history. Is interesting to notice that the more sophisticated SMOs (e.g., `MERGE TABLE`) while being indispensable are not very common. The balance between column/table additions and deletions highlights the "content preserving" attitude of Wikipedia[13].

[13]The main noticeable exception is the set of informa-

Table 4: Micro-Classification of Schema Changes Using SMOs and Frequencies.

SMO type	# of usages	% of usage	% per version
`CREATE TABLE`	24	8.9%	14%
`DROP TABLE`	9	3.3%	5.2%
`RENAME TABLE`	3	1.1%	1.75%
`DISTRIBUTE TABLE`	0	0.0%	0%
`MERGE TABLE`	4	1.5%	2.33%
`COPY TABLE`	6	2.2%	3.5%
`ADD COLUMN`	104	38.7%	60.4%
`DROP COLUMN`	71	26.4%	41.5 %
`RENAME COLUMN`	43	16.0%	25.1 %
`MOVE COLUMN`	1	0.4%	0.58%
`COPY COLUMN`	4	1.5%	2.33%
Total	**269**	**100%**	–

Figure 7 shows the number of SMOs (overall) for each evolution step. The curve shows how the schema evolution has been mainly a continuous process of adjustment, with few exceptions shown as spikes in the figure, coinciding with major evolution steps, such as:
(i) v6696 - v6710 (41_{st}-42_{nd}), 92 SMOs: a change in the storage strategy of the article versions,
(ii) v9116 - v9199 (61^{st}-62^{nd}), 12 SMOs: a change in link management, and
(iii) v20393 - v20468 (138^{th}-139^{th}), 9 SMOs: history management (log features added to several tables).

3.4 The Impact on the Applications

In order to study the effect of schema evolution on the frontend application, we analyze the impact of the schema changes on six representative sets of queries. Each experiment tests the success or failure of a set of queries, originally designed to run on a specific schema version, when issued against other schema versions.

To simulate a case where current applications are run on databases under older schema versions, we test three sets of queries, valid on the last schema version, on all the previous schema versions (Figure 8). Also, to study how legacy applications succeed or fail on newer versions of the database schema, we test three sets of legacy queries on all the subsequent schema versions (Figure 9). The six sets considered in our experiments are as follows:
Real-world Templates, Current (Figure 8). The 500 most common query templates (extracted[14] from over 780 millions of query instances), derived from the Wikipedia on-line profiler[15] and post-processed for cleaning[16].

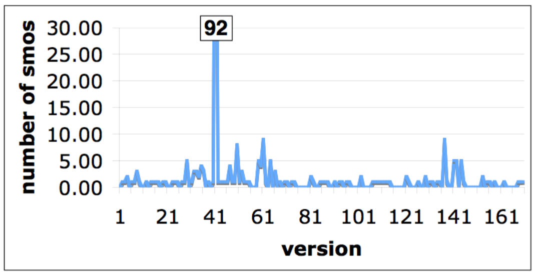

Figure 7: Number of SMOs Used in Each Evolution Step.

Lab-gen Queries, Current (Figure 8). 2496 query instances generated by a local installation of the current version of MediaWiki (release 1.11, schema version 171), interacting with the frontend[17] and logging the queries issued against the underlying MySQL DBMS.

Lab-gen Templates, Current (Figure 8). 148 templates of queries extracted from the above queries.

Lab-gen Queries, Legacy (Figure 9). 4175 query instances generated by a local installation of an old version of MediaWiki (release 1.3[18], schema version 28), interacting with the frontend and logging the queries issued against the underlying MySQL DBMS.

Lab-gen Templates, Legacy (Figure 9). 74 templates extracted from the above lab-gen queries, legacy.

Synthetic Probe Queries, Legacy (Figure 9). 133 synthetic queries accessing single columns (i.e., `select` $tab_j.att_i$ `from` tab_j) of schema version 28, designed to highlight the affected schema portion.

Each set has been tested against all schema versions: the resulting query execution success rates are shown in Figure 8 and Figure 9. The outliers in the graphs (sudden and extremely low values) are due to syntactically incorrect DB schema versions.

The first three sets are shown in Figure 8. It is interesting to notice that:

- proceeding from right to left, the series of descending steps illustrates the increasingly low compatibility of queries and schemata.

tion supporting the *user rights* management, which has been strongly reduced in the DB after version v9335 (65th version), as it was moved to the application layer.

[14]The templates are extracted ignoring constants and retaining only the query structure.

[15]Available on-line at http://noc.wikimedia.org/cgi-bin/report.py.

[16]The cleaning process removes syntactical errors produced by the template extraction of the Wikipedia profiler.

[17]In order to generate as many as possible types of queries, we tried to trigger all features accessible from the web browser.

[18]The oldest version compatible with the environment of our experimental setting.

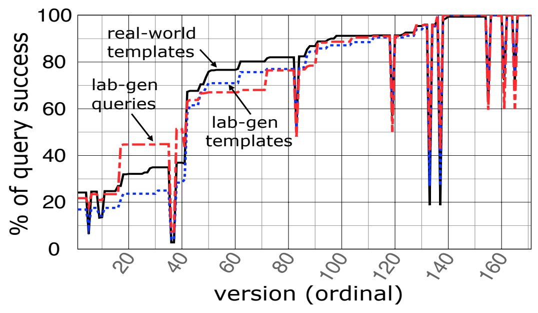

Figure 8: Average query success rate against *preceding* schema versions (the queries are designed for the last version, and run against all the previous versions).

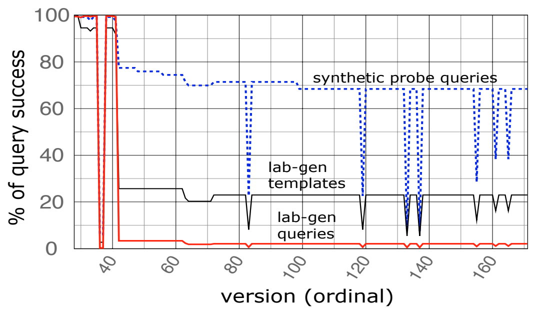

Figure 9: Average query success rate against *following* schema versions (the queries are designed for the 28^{th} version, and run against all the following versions).

- the sudden drop in query success – of about 30% – which appears between commit revisions v6696 and v6710 (41^{st}-42^{nd})[19] highlights one of the most intense evolution steps of the MediaWiki data management core, involving a deep change in the management of article revisions;
- the lab-generated and real-world templates carry very similar information. This seems to indicate that our local query generation method is capable of producing a representative set of queries.

Figure 9 shows a graph of the average execution success rates for the latter three query sets. Some interesting observations are as follows:

- the synthetic probe queries, by failing when columns or tables are modified, highlight the portion of the schema affected by the evolution. The figure shows how the schema evolution invalidates at most 32% of the schema.
- in the last version, a very large portion (77%) of the lab-gen templates fails due to the evolution.
- for lab-gen templates, the big evolution step between commit revisions v6696 and v6710 (41^{st} - 42^{nd}) invalidates over 70% of the queries.
- lab-gen templates failure rate compared to synthetic probe queries failure rate exposes how the schema modifications affected a schema portion heavily used by the applications (32% of the schema being affected invalidates 77% of the query templates).
- the gap between the success rate of legacy query instances (2.9%) and legacy query templates (22%) shows that the failing templates actually correspond to the most common query instances.

Finally it is interesting to notice that the number of features of the MediaWiki software has grown in time; this explains the growth in the number of the query templates extracted from legacy queries (74) and current queries (148).

These experiments provide a clear evidence of the strong impact of schema changes on applications, and sustain the claim for better schema evolution support.

4 ANALYSIS TOOL SUITE

To collect the statistics described in this paper, we developed a set of tools, organized in a tool-suite available on-line (Curino et al., 2008a). This step-by-step process, primarily designed to help researchers to gain better insight in the schema evolution of existing Information Systems, can be effectively exploited by DB administrators and developers, in any data-centric scenario, to analyze the history of the DB schema and create a (summarized) view of its evolution history. The tool suite will support the analysis of the evolution process and help to highlight possible flaws in the design and maintenance of the Information System. Moreover, Researchers and designers of support methods and tools for DB evolution and versioning, can exploit this to test their approaches against real-life scenarios.

We now discuss the features of our tool-suite referring to its application to the MediaWiki DB: first of all, by means of an appropriate tool, the 171 MediaWiki DB schema versions have been downloaded from SVN repository and batch-installed in a MySQL DBMS[20]. We developed a tool, named *statistics_collection*, that can be applied on this data to derive the basic statistics of schema versions, such as schema size and average table/column lifetime. The *statistics_collection* tool queries the MySQL data dictionary (the **`information_schema`** meta-database) to gather the statistical measures presented

[19]See (Curino et al., 2008a) for SVN commit version to ordinal numbers conversion.

[20]MySQL version 5.0.22-Debian.

in Section 3.1.

For fine-grained view of the schema evolution we also provide the *SMO_extractor* tool. This tool, by operating on the differences between subsequent schema versions, semiautomatically extracts a set of candidate SMOs describing the schema evolution, minimizing the user effort[21]. To estimate query success against different schema versions, the users can exploit a tool named *query_success analyzer*. This tool performs a query success rate analysis by batch-running its input queries against all schema versions.

For users' convenience, we also provide a *log_analyzer* which can be used to extract and clean the SQL query instances and templates from the raw **`mysql_log`** format.

Every component of the tool-suite stores the collected information, in a non-aggregated form, in a database, named *evolution_metadb*. This database is later queried to provide statistical measures of the schema evolution, and provides a flexible SQL-based interface, we exploit to compute the graphs and tables of this paper.

5 TOWARD A UNIFIED BENCHMARK

DB schema evolution has been recognized to be a relevant problem among both researchers and practitioners, but despite the number of proposed solutions (Roddick, 1995; Ram and Shankaranarayanan, 2003; Bernstein, 2003; Velegrakis et al., 2003; Yu and Popa, 2005), a unified benchmark is currently missing – although needed as noted in (Bernstein et al., 2006).

The case study we present in this paper represents our initial step towards the definition of a reusable and standardized benchmark. To the best of our knowledge, this is the first attempt to provide a real-world DB schema evolution benchmark to date.

The benchmark we are developing will contain the results of the analysis of several case studies of open-source systems, currently under development, together with the MediaWiki example presented here. In addition we are developing a set of tools to support our benchmarking procedure. Among such tools we have the *query_success analyzer* discussed in the previous section and a *data_generator*, used to batch-populate with synthetic data (of variable size) all available versions of the DB under analysis. The *data_generator* produces randomized data, but is tailored to create DB contents that maximize the query answer predictability by means of data regularity.

While the overall benchmark is still under development, we made available on-line at (Curino et al., 2008a) our MediaWiki data-set to provide researchers and practitioners data to evaluate and test their approaches. This data-set has already been successfully exploited to test the PRISM and PRIMA system in (Moon et al., 2008; Curino et al., 2008b).

We believe that, w.r.t. the goal of developing a unified benchmark for DB schema evolution, MediaWiki is an ideal starting point because: (i) it is a real-life application used by 30,000 wikis, including Wikipedia, (ii) its code and data are well-documented and released under GPL License and based on an open-source plaftorm, (iii) several differently-sized DB contents (the DB dump of different public wikis), ranging from tens of KBytes to hundreds of GBytes (Almeida et al., 2007), are available to the public[22], and (iv) there is an on-line profiling system providing real-life queries from the Wikipedia site, along with their frequencies and typical workload details[23].

Benchmark Users. The benchmark under development is mainly intended to: (i) educate database administrators on typical schema evolution scenarios, in order to avoid common design errors and improve the quality of initial schema designs, (ii) support the community of researchers working on the schema versioning / schema evolution problems, (iii) provide researchers and practitioners, designing solutions for data migration, with a rich test-case for tools and methodologies, (iv) provide a rich set of examples of evolution to enable evolution pattern mining.

6 RELATED WORKS

In this section, we compare our analysis with the existing case studies on schema evolution for traditional information systems (Sjoberg, 1993; Marche, 1993). (Sjoberg, 1993) discusses database schema evolution in a health management system (HMS). This careful analysis of nine schema versions shows an increase in the number of tables from 23 to 55 and in the number of columns from 178 to 666 during 18 months (consisting in 5 months of development and 13 months of operational phase). Sjoberg discusses how application queries are affected when the schema evolves, as we do in Section 3.4.

[21]Complex evolution patterns as the one appeared from the 41st and 42nd schema versions in MediaWiki, require the user to refine the set of SMOs according to his/her understanding of the schema evolution.

[22]See http://download.wikimedia.org/.

[23]Available at: http://noc.wikimedia.org/cgi-bin/report.py.

Table 5: Comparison of Schema Growth in MediaWiki and Those in Other Case Studies.

Case	Interval (months)	Number of Tables				Number of Columns			
		First	Last	Increase	Inc/year	First	Last	Increase	Inc/year
Sjoberg-all	18	23	55	139%	92.6%	178	666	274%	182.7 %
Sjoberg-oper	13	47	55	17%	15.7%	528	666	26%	24.0 %
Marche	31.6	9.6	10.6	10%	3.8 %	118.9	139.0	17%	6.5 %
MediaWiki-all	55	17	34	100%	21.8 %	100	242	142%	31.0 %
MediaWiki-oper	48	18	34	89%	22.3 %	106	242	128%	32.0 %

In (Marche, 1993), a collective case study is presented for seven database applications from different application domains. For each application, Marche compares only two schema versions, taken at interval ranging from 6 to 80 months. The author does not specify whether such versions correspond to the development or operational phase of the systems under analysis. This analysis reports an increase in the average number of relations and columns from 9.6 to 10.6, and from 118.9 to 139.0, respectively.

In addition to a major change of environment, from Traditional to Web Information Systems, our work improves the previous case studies as follows:

- **Number of Schema Versions:** We analyze 171 published[24] versions of the schema whereas the previous works use respectively *nine* and *two* versions. This was possible due to the open-source nature of the MediaWiki project, uncommon in case of traditional, proprietary applications.
- **Detailed Schema Evolution Analysis:** We classify schema changes at a finer level of granularity by means of SMOs. We benefited from the rich documentation of SVN schema revisions and of the SQL schema files to obtain insight in each evolution step and derive the corresponding SMOs.
- **Legacy Application Failure Analysis:** (Sjoberg, 1993) studied the effect of schema evolution on applications, *predicting* query failure based on query workload and schema changes between two successive schema versions. In our setting we were able to report the *actual* success rate of the *execution* of queries from an old release of MediaWiki (v1.3) on 144 subsequent schema versions, together with the success rate of 500 templates extracted from millions of queries run on the actual Wikipedia installation of MediaWiki, tested against the 170 previous schema versions.
- **Licensing and Data-set Release:** Thanks to the licensing of MediaWiki and Wikipedia, we are able to release (Curino et al., 2008a) the entire data-set used for our analysis to the public, enabling other researchers to exploit such data to extract their own statistics or to test their approaches.

[24]More schema versions are available in the unstable branches of the versioning system. We focused on the main development branch.

Web IS vs Traditional IS. Table 5 provides results of the MediaWiki schema growth compared to the cases reported in the cited literature.

While (Sjoberg, 1993) reports the growth during the entire studied period (5 months of development and 13 months of operation) and that during the operational phase only, tagged in Table 5 respectively as *Sjoberg-all* and *Sjoberg-oper*, we focus our comparison on the operational phase, which has a bigger impact on users and maintenance costs. For this reason we show as *MediaWiki-oper* the growth of the MediaWiki schema, by removing from the overall history the first six versions – preceding the first official release. (Marche, 1993) does not clearly specify which phase of the software life-cycle each schema version was taken from, so we simply report the available data. Adjusted statistics appear in Table 5. Comparing the time-normalized (Increase/year) schema growth, *MediaWiki-oper* is faster than every previous result in Traditional Information Systems. The operational growth is about 38% more intense than the one of *Sjoberg-oper*, and about 539% than the average of the seven cases of *Marche*.

This difference can be attributed to the following reasons: (i) the collaborative, open-source nature of the development and usage of MediaWiki, determines the presence of several independent contributions, influencing the speed of growth, (ii) the success of Wikipedia triggered the need for intense tuning for performance and accessibility, leading to a quicker evolution than traditional IS, and (iii) the interest for maintaining historical information grew during the development, affecting the schema size.

7 CONCLUSIONS

The explosion of Web Information Systems (WIS) is creating a throve of interesting research problems and technical challenges. In particular, the DBMS systems that are at the core of many WIS are now faced with new challenges and requirements—which we have analyzed in this in-depth study of MediaWiki, the sofware behind Wikipedia, a WIS of great renown and importance. Our study shows that MediaWiki has undergone a very intensive schema evolution, as a result of the cooperative, multi-party, open-source development and administration that is common in leading-edge WIS projects. Thus, the WIS environment, (i) contrasts with the smaller, less-open and slow-turnover setting of typical in traditional information systems, (ii) creates a more urgent needs for better automation and documentation tools for supporting graceful schema evolution in WIS. In this paper we analyze and quantify the schema evolution problem of WIS and introduce concepts and tools that represent an important first step toward realizing (ii).

At the conceptual level, we have introduced the Schema Modification Operators (SMOs), that proved effective both in an operational mode to support schema evolution (Moon et al., 2008; Curino et al., 2008b), and in an "a posteriori" mode to support in-depth analysis. Moreover, we also developed a simple set of software tools to facilitate the analysis of schema evolution, and the derivation of the SMOs describing such an evolution. This tool-suite proved effective in the analysis of MediaWiki and is available online at (Curino et al., 2008a). The structured representation of the evolution history of MediaWiki that we derived in this project is also available for downloading (Curino et al., 2008a). Such data-set is currently being extended by analyzing other leading WIS projects in order to create a rich schema evolution benchmark. Once completed, this benchmark will (i) provide the community with a rich set of schema evolution examples that can be studied to avoid common up-front design errors and improve schema management best practices, and (ii) represent a critical validation tool for techniques and systems designed to automate the schema evolution process (including those that are currently under development in our lab). Indeed, the desirability of such a benchmark was stressed in the past by other researchers working in related areas (Bernstein et al., 2006).

ACKNOWLEDGEMENTS

The authors would like to thank Alin Deutsch for the numerous in-depth discussions on schema mapping and query rewriting.

REFERENCES

Almeida, R. B., Mozafari, B., and Cho, J. (2007). On the evolution of wikipedia. In *Int. Conf. on Weblogs and Social Media*.

Bernstein, P. A. (2003). Applying model management to classical meta data problems. In *CIDR*.

Bernstein, P. A., Green, T. J., Melnik, S., and Nash, A. (2006). Implementing mapping composition. In *VLDB*.

Curino, C., Moon, H., Tanca, L., and Zaniolo, C. (2008a). Pantha rei data set [online]: http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Main_Page.

Curino, C. A., Moon, H. J., and Zaniolo, C. (2008b). Graceful database schema evolution: the prism workbench. In *UCLA Tech. Rep., 2008. Submitted for publication*.

Franconi, E., Grandi, F., and Mandreoli, F. (2001). Schema evolution and versioning: A logical and computational characterisation. *Database Schema Evolution and Meta-Modeling*, pages 85–99.

Galante, R. d. M., dos Santos, C. S., Edelweiss, N., and Moreira, A. F. (2005). Temporal and versioning model for schema evolution in object-oriented databases. *Data & Knowledge Engineering*, 53(2):99–128.

Marche, S. (1993). Measuring the stability of data models.*European Journal of Information Systems*, 2(1):37–47.

Moon, H. J., Curino, C. A., Deutsch, A., Hou, C.-Y., and Zaniolo, C. (2008). Managing and querying transaction-time databases under schema evolution. In *UCLA Tech. Rep., 2008. Submitted for publication*.

Moro, M. M., Malaika, S., and Lim, L. (2007). Preserving XML Queries during Schema Evolution. In *WWW*, pages 1341–1342.

Ram, S. and Shankaranarayanan, G. (2003). Research issues in database schema evolution: the road not taken. In *Boston University School of Management, Department of Information Systems, Working Paper No: 2003-15*.

Rizzi, S. and Golfarelli, M. (2007). X-time: Schema versioning and cross-version querying in data warehouses. In *ICDE*, pages 1471–1472.

Roddick, J. (1995). A Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology*, 37(7):383–393.

Shneiderman, B. and Thomas, G. (1982). An architecture for automatic relational database system conversion. *ACM Transactions on Database Systems*, 7(2):235–257.

Sjoberg, D. I. (1993). Quantifying schema evolution. *Information and Software Technology*, 35(1):35–44.

Velegrakis, Y., Miller, R. J., and Popa, L. (2003). Mapping adaptation under evolving schemas. In *VLDB*.

Wikimedia Foundation (2007). The mediawiki workbook 2007 dammit.lt/uc/workbook2007.pdf. [Online].

Yu, C. and Popa, L. (2005). Semantic adaptation of schema mappings when schemas evolve. In *VLDB*.