Выбор модели версионирования данных при проектировании информационных систем

Б. А. Черныш, А. В. Мурыгин

Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнёва, Красноярск, Россия

Аннотация. В статье дается обзор существующих механизмов версионирования данных, приводятся их характерные отличия, достоинства и недостатки, а также примеры использования. Предлагается способ сравнительной оценки этих механизмов на тестовой реляционной базе данных с использованием серии операций над версиями. Данная методика была использована авторами при проектировании и разработке интегрированной информационной системы. Результаты выполнения для разных типов сведены в таблицы и графически представлены в виде диаграммы. На основании исходных требований к системе, характеристик моделей и полученных результатов оценки выполнен анализ эффективности исследуемых механизмов. Результатом анализа является выбор наиболее оптимальной модели с точки зрения скорости выполнения операций с версиями, целостности данных и гибкости работы с атрибутами. Предлагаемая методика не ограничивается использованием реляционной базы данных и может быть адаптированы для других типов хранилищ.

Ключевые слова: база данных, версионность, версионирование, Slowly Changing Dimension, SCD, Hibernate Envers, Aras Innovator, SQL, NoSQL, SciCMS.

DOI 10.14357/20718632230313

Введение

Системы управления базами данных (СУБД) являются фундаментальной частью любого программного обеспечения (ПО), ориентированного на обработку больших объемов структурированной информации. Выбор того или иного типа СУБД (реляционные, документо-ориентированные, «ключ-значение», графовые) должен осуществляться на основании требований к структуре данных и способам их обработки. Выбранная СУБД, в свою очередь, также может накладывать ограничения на архитектуру информационной системы (ИС), исходя из своих возможностей. Эти ограничения могут касаться проектируемой структуры, типов данных, отношений между сущностями выполняемых операций. Однако наряду с рассмотренными ограничениями в дисциплине обработки данных существуют архитектурные задачи, способы решения которых применимы в любой СУБД независимо от природы. Одной из таких задач является поддержка версионности и истории хранения сущностей. Необходимость версионирования может быть обусловлена различными факторами: требованиями аудита, потребностями аналитики, особенностями бизнес-процессов на предприятии (например, в производстве при модернизации изделия выпускаются новые версии конструкторской документации) и т.д.

1. Способы версионирования, их достоинства и недостатки. Примеры реализации

Существующие способы решения задач версионирования данных наиболее полно описаны в работе [1], где вводится понятие Slowly Changing Dimension (SCD) – медленно меняющееся измерение. В рамках SCD существует 8 типов, определяющих как история изменений может быть отражена в модели данных. Их краткое описание приведено в Табл. 1. Несмотря на то, что рассматриваются примеры для реляционных баз данных (БД), описанные методики применимы и для других типов СУБД.

В приведенном описании типы 0 и 1 фактически не являются методами версионирования, и в чистом виде для этой цели применяться не могут. Типы 5 – 7 являются гибридными и основаны на вышестоящих типах. Выбор в пользу того или иного метода может быть выполнен на основе анализа требований к версионированию данных, оценке операций, которые должны выполняться наиболее эффективно. В Табл. 2 перечислены достоинства и недостатки основных типов SCD.

Наибольшее распространение на практике получили типы 2, 4 и 7. Так, например, модуль Envers (в составе широко распространенной библиотеки объектно-реляционного отображения Hibernate) [3] использует отдельную таблицу с переносом в нее старых записей при обновлении основной таблицы (тип 4). В системе управления контентом Strapi [4] имеется плагин Content Versioning [5], реализованный на базе типа 2 (добавление записи в основную таблицу с обновлением атрибутов версии). Система управления жизненным циклом изделия Aras Innovator [6] также хранит все записи в основной таблице с использованием атрибутов версии и дополнительным атрибутом конфигурации, общим для всех версий конкретной сущности (тип 7).

Табл. 1. Типы SCD

Наименование	**Принцип работы**
Тип 0	Хранение оригинала. Данные после записи в таблицу никогда не изменяются. Этот метод не поддерживает версионности и используется лишь как нулевая точка отсчета в методологии SCD.
Тип 1	Перезапись. Информация в таблице заменяется на новую. В чистом виде этот метод тоже не содержит версионности и используется лишь там, где история фактически не нужна.
Тип 2	Добавление новой строки. Создается запись, содержащая новую информацию, старая информация не изменяется [2]. Для каждой записи добавляются поля атрибутов данной версии, например: номер версии, дата начала и конца периода существования версии.
Тип 3	Добавление нового атрибута. Запись в таблице содержит дополнительные столбцы для отражения изменения. При получении новых данных старые данные копируются в эти столбцы и перезаписываются текущими значениями.
Тип 4	Добавление отдельной таблицы для часто изменяющихся атрибутов. Основная таблица всегда перезаписывается текущими данными. При построении хранилищ данных таблица фактов содержит ссылки на основную и дополнительную таблицы измерений.
Тип 5	Аналогично типу 4, добавляется отдельная таблица для часто изменяющихся атрибутов. При построении хранилищ данных таблица фактов и основная таблица измерений имеют ссылку на дополнительную таблицу измерений.
Тип 6	Комбинация типов 1, 2 и 3. Создается запись, содержащая новую информацию (тип 2) и дополнительные атрибуты для отражения изменения (тип 3). При этом дополнительные столбцы в одной или всех предыдущих версиях перезаписываются новыми данными (тип 1).
Тип 7	Аналогично типу 2, создается запись, содержащая новую информацию. Дополнительно создается ключ, определяющий принадлежность версии определенной сущности. Такая конфигурация позволяет, например, получать все версии для конкретной записи.

Табл. 2. Достоинства и недостатки основных типов SCD

Наименование	Достоинства	Недостатки
Тип 1	• не добавляется избыточность • простая структура	• не хранится истории
Тип 2	• хранится полная история версий • простой доступ к данным	• добавляется избыточность в виде дублирующихся атрибутов • усложняется логика обработки, если для аналитики требуется согласование данных в таблице фактов с конкретными версиями измерения
Тип 3	• небольшой объем данных • простой и быстрый доступ к истории	• ограниченная история
Типы 4 и 5	• быстрая работа с текущими версиями	• разделение сущности на разные таблицы
Тип 6	• хранится полная история версий • простой и быстрый доступ к измененным атрибутам	• усложняется структура и логика обработки за счет смешивания методов обработки
Тип 7	• простой доступ к истории конкретной сущности	• усложняется логика обработки за счет введения дополнительного атрибута

Требования к модели версионирования и оценка эффективности существующих моделей исходя из этих требований

При проектировании авторами интегрированной информационной системы SciCMS [7] поддержка контроля версий являлась одним из базовых требований. Основное предназначение системы – сопровождение жизненного цикла наукоемкой продукции [8]. В решении данной задачи версионирование позволяет иметь разные версии изделия в рамках единого дерева документации. Дополнительным преимуществом такой модели хранения является значительное упрощение подготовки документации для заказчиков, основываясь на конкретной версии изделия. Задачи аналитики и учета также требуют представления информации об изделиях в разрезе их версий. Таким образом, основные требования к контролю версий в системе следующие:

- поддержка ссылочной целостности [9] на уровне СУБД (при добавлении или удалении записей ссылки на старые версии должны сохраняться);
- простая и быстрая выборка записей как текущей, так и произвольной версии;
- возможность получения полного снимка дерева любой версии;
- быстрое создание новых версий;
- возможность удаления произвольной версии либо всех версий определенной сущности;
- возможность копирования либо переноса ссылок (внешних ключей) из связанных записей на разные версии сущности.

Несмотря на широкое практическое использование рассмотренных ранее типов SCD [2;3;5;6], недостаточно изученным остается вопрос их применимости для конкретных условий эксплуатации ИС. В открытых источниках отсутствуют методики по выбору подходящей модели версионирования на основании строго определенных требований к проектируемым системам. Так, в работе [1] авторы методологии SCD указывают, что выбор того или иного типа зависит от структуры и требований к хранилищу данных, но объективных данных об их реальной эффективности не приводится. Источники [2;10] описывают применение типов 1, 2 и 3 при построении хранилищ в СУБД Microsoft SQL Server. При этом не рассмотрены остальные типы и не выполнена оценка скорости выполнения операций. В работе [3] описывается модель версионирования, реализованная на базе типа SCD 4, но явно не упоминается методология SCD и не приводится обоснование применения

выбранного типа. Вопросы производительности моделей версионирования затрагиваются в работе [11], где оценивается эффективность типов SCD 1-4 в реляционной базе данных в сравнении с объектно-реляционной. Однако проблемы выбора типов и их производительности относительно друг друга не рассмотрены.

Для выявления закономерностей в поведении различных типов SCD в процессе разработки SciCMS было проведено сравнительное исследование эффективности операций с версиями. Комплексная оценка результатов, помимо собственно замеров, также учитывает характеристики рассмотренных моделей и специфические требования к контролю версий в ИС. В исследование вошли отвечающие предъявляемым условиям типы 2, 4 и 7. SQL-операции с версиями выполнены на реляционной СУБД Oracle. Структура тестовой схемы данных приведена на Рис. 1.

Таблица DOC_ENTRIES_FULL содержит элементы дерева документов с основными атрибутами (ID, PARENT_ID, LABEL, NAME) и атрибутами версий (CONFIG_ID, REV_ID, GENERATION, IS_CURRENT, START_DATE, END_DATE), необходимыми для реализации модели версионирования по типам 2 и 7. Таблица DOC_ENTRIES_SHORT содержит только собственные атрибуты для хранения последних версий элементов дерева. История версий хранится в таблице DOC_ENTRIES_REV, которая дополнительно к собственным атрибутам записей также имеет атрибуты версий (REV_ID, REV_DATE, REV_TYPE). Атрибут REV_TYPE может принимать три значения: 0 – запись вставлена, 1 – запись обновлена, и 2 – запись удалена. Таблицы DOC_ENTRIES_SHORT и DOC_ENTRIES_REV реализуют модель SCD по типу 4. Все столбцы, участвующие в запросах, проиндексированы. Число записей в таблицах DOC_ENTRIES_FULL и DOC_ENTRIES_REV – 471 445, в таблице DOC_ENTRIES_SHORT – 59 118. Число версий каждой сущности варьируется от 1 до нескольких десятков. Тексты SQL-запросов для различных операций приведены в Табл. 3. Тип 7 отличается от типа 2 наличием дополнительного атрибута CONFIG_ID. Роль этого атрибута в типе 2 может играть любой неизменяемый естественный ключ (в нашем случае – это атрибут LABEL). В остальном модели 2 и 7 не отличаются, поэтому в эксперименте эти типы объединены. Отметим также, что семантика модели 4 не предусматривает физического удаления версий из БД, однако операция удаления всех версий определенной сущности включена в эксперимент для проведения всесторонней сравнительной оценки. Операции создания и удаления версии являются составными и выполняются в рамках одной транзакции, время выполнения рассчитывается как суммарное время для составляющих операцию запросов.

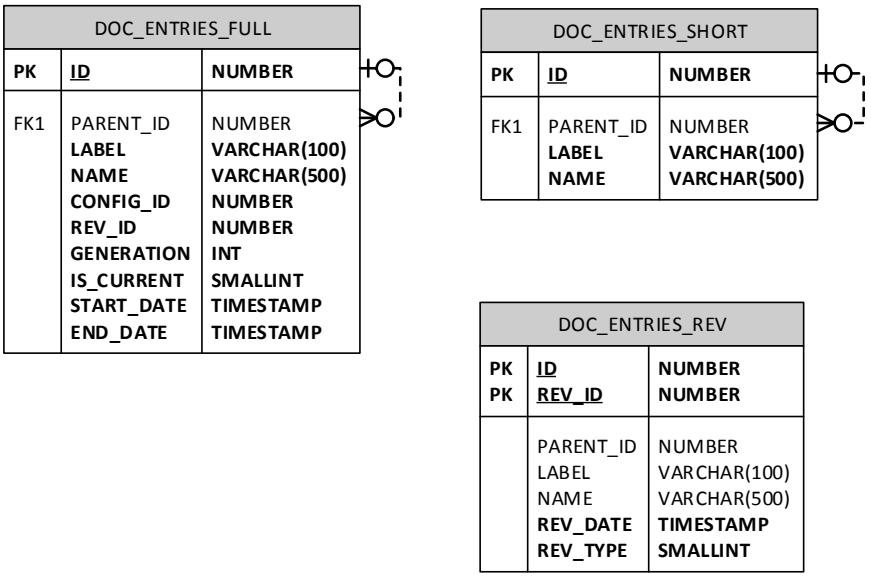

Рис. 1. Тестовая схема данных

Табл. 3. SQL-запросы операций с версиями

Операция	**Тип 2 (7)**	**Тип 4**
Получение версии	`SELECT * FROM doc_entries_full WHERE config_id = 1 AND rev_id = 2;`	`SELECT * FROM doc_entries_rev WHERE label = 'Label' AND rev_id = 2;`
Получение версии на определенную дату (01.01.2022)	`SELECT def.* FROM doc_entries_full def WHERE def.config_id = 1 AND def.start_date = (SELECT MAX(def2.start_date) FROM doc_entries_full def2 WHERE def2.config_id = 1 AND def2.start_date <= TO_TIMESTAMP('01.01.2022', 'DD.MM.YYYY') AND def2.end_date > TO_TIMESTAMP('01.01.2022', 'DD.MM.YYYY') );`	`SELECT der.* FROM doc_entries_rev der WHERE der.id = 1 AND der.rev_date = (SELECT MAX(der2.rev_date) FROM doc_entries_rev der2 WHERE der2.id = 1 AND der2.rev_date < TO_TIMESTAMP('01.01.2022', 'DD.MM.YYYY');`
Получение последней версии	`SELECT * FROM doc_entries_full WHERE config_id = 1 AND is_current = 1;`	`SELECT * FROM doc_entries_short WHERE id = 1;`
Получение всех последних версий	`SELECT * FROM doc_entries_full WHERE is_current = 1;`	`SELECT * FROM doc_entries_short;`

Создание новой версии	`UPDATE doc_entries_full SET is_current = 0, end_date = CURRENT_TIMESTAMP WHERE config_id = 1 AND is_current = 1;`
	
	`INSERT INTO doc_entries_full (id, parent_id, label, name, config_id, start_date, end_date, is_current, rev_id, generation) VALUES (doc_entry_seq.nextval, NULL, 'Label', 'New Name', 762166, CURRENT_TIMESTAMP, TO_TIMESTAMP('31.12.9999', 'DD.MM.YYYY'), 1, rev_seq.nextval, gen_seq.nextval);`
		`INSERT INTO doc_entries_rev (id, rev_id, parent_id, label, name, rev_date, rev_type) SELECT id, rev_seq.nextval, parent_id, label, name, CURRENT_TIMESTAMP, 1 FROM doc_entries_short WHERE id = 1;`
		
		`UPDATE doc_entries_short SET parent_id = NULL, label = 'Label', name = 'New Name' WHERE id = 1;`

Удаление версии	`DELETE FROM doc_entries_full WHERE config_id = 1 AND is_current = 1;`
	
	`UPDATE doc_entries_full SET is_current = 1, end_date = TO_TIMESTAMP('31.12.9999', 'DD.MM.YYYY') WHERE config_id = 1 AND generation = (SELECT MAX(def.generation) FROM doc_entries_full def WHERE def.config_id = 1);`
		`INSERT INTO doc_entries_rev (id, rev_id, parent_id, label, name, rev_date, rev_type) SELECT id, rev_seq.nextval, parent_id, label, name, CURRENT_TIMESTAMP, 2 FROM doc_entries_short WHERE id = 1;`
		
		`DELETE FROM doc_entries_short WHERE id = 1;`

Удаление всех версий (не предусмотрено для типа 4)	`DELETE FROM doc_entries_full WHERE config_id = 1;`	`DELETE FROM doc_entries_rev WHERE id = 1;`
		
		`DELETE FROM doc_entries_short WHERE id = 1;`

Выполнено по 10 запросов для каждой операции в рамках соответствующей модели SCD. Результаты профилирования приведены в Табл. 4 и 5. Для оценки стабильности результатов в таблицах приводится стандартное отклонение времени на основании несмещенной оценки дисперсии. Рассчитано по формуле:

$$S_0 = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_i - \bar{t})^2},$$

где t_i - значение времени i-го запроса (в миллисекундах), $\bar{t}$ – среднее значение по всем запросам, n – общее число запросов.

Среднее время выполнения операций в разрезе типов SCD графически представлено на диаграмме (Рис. 2).

На основании полученных результатов можно отметить следующие закономерности:

- операции получения конкретной версии сущности одинаково эффективно выполняются в каждом из рассмотренных типов;
- операции получения всех последних версий значительно быстрее (примерно в 2 раза) выполняются для типа 4;

Табл. 4. Время выполнения операций с версиями для типа 2 (7) (мс)

№ п/п	Получение версии	Получение версии на определенную дату	Получение последней версии	Получение всех последних версий	Создание новой версии	Удаление версии	Удаление всех версий
1	67	70	57	980	23	72	369
2	49	51	31	944	11	70	254
3	52	63	30	893	10	68	285
4	45	58	39	876	9	71	207
5	43	54	30	920	10	72	403
6	44	53	28	903	11	66	238
7	43	52	30	923	10	75	327
8	45	53	36	833	11	67	344
9	59	53	29	864	11	71	361
10	47	59	29	891	10	62	222
$\bar{t}$	**49,4**	**56,6**	**33,9**	**902,7**	**11,6**	**69,4**	**301**
S_0	**7,9**	**6,0**	**8,8**	**41,8**	**4,1**	**3,7**	**68,9**

Табл. 5. Время выполнения операций с версиями для типа 4 (мс)

№ п/п	Получение версии	Получение версии на определенную дату	Получение последней версии	Получение всех последних версий	Создание новой версии	Удаление версии	Удаление всех версий
1	55	52	33	431	10	3	148
2	47	64	15	482	2	2	140
3	53	49	18	412	7	3	138
4	38	53	16	406	2	2	136
5	35	43	23	416	2	2	131
6	38	48	18	470	3	3	133
7	35	43	21	442	3	3	138
8	36	52	20	412	3	2	132
9	41	45	30	426	3	2	134
10	33	51	35	419	5	3	136
$\bar{t}$	**41,1**	**50**	**22,9**	**431,6**	**4**	**2,5**	**136,6**
S_0	**7,9**	**6,2**	**7,2**	**25,8**	**2,6**	**0,5**	**4,9**

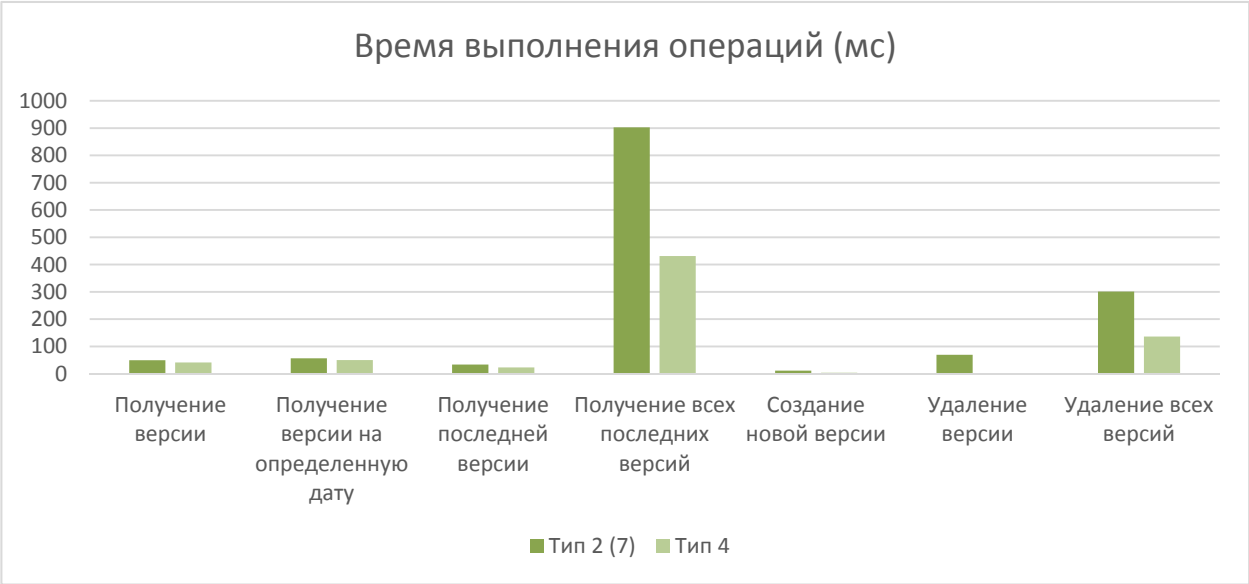

Рис. 2. Время выполнения операций с версиями (мс)

- создание новой версии быстро выполняется в обеих моделях с небольшим преимуществом у типа 4;
- удаление конкретной версии для типа 4 на порядок эффективнее типов 2 и 7;
- тип 4 не предусматривает операции удаления всех версий конкретной сущности, однако данная операция выполняется примерно в 2 раза быстрее и на порядок стабильнее, чем для типа 2 (7);
- время выполнения запросов (за исключением операции удаления всех версий конкретной сущности) имеет сходную (и весьма высокую) стабильность для каждого из рассмотренных типов.

Высокая эффективность выборки всех последних версий для типа 4 обусловлена значительно меньшим (на порядок) размером таблицы актуальных версий (DOC_ENTRIES_SHORT) по сравнению с таблицей DOC_ENTRIES_FULL. Небольшой выигрыш типа 4 при создании версии достигается за счет того, что происходит обновление лишь одной записи в таблице DOC_ENTRIES_SHORT. При этом дополнительная операция вставки в таблицу истории занимает всего 1-2 мс. Аналогично, удаление одной единственной записи обеспечивает типу 4 существенное преимущество во время операции удалении версии.

2. Выбор модели в соответствии с предъявляемыми требованиями и результатами оценки

Принимая во внимание изначальные требования к контролю версий в системе SciCMS, и учитывая результаты оценки эффективности моделей SCD, а также их достоинства и недостатки, можно сделать следующие выводы:

- операции по получению конкретной версии и созданию новой версии в каждом из рассмотренных типов имеют одинаково высокую эффективность и простую семантику;
- наибольшую эффективность при обработке последнего снимка (всех текущих версий) показывает тип 4 (за счет раздельного хранения текущих версий и истории);
- структура и семантика модели 4 не позволяет выполнять весь спектр требуемых операций, в частности, может происходить потеря ссылок (внешних ключей) из связанных записей на разные версии сущности при копировании в таблицу истории.

Последний фактор имеет важное значение, т.к. получение снимка дерева документации любой произвольной версии является одной из ключевых функций в SciCMS. В случае использования типа 4 ссылки могут находится как в

таблице актуальных версий, так и в таблице истории, поэтому способ их эффективной обработки практически нереализуем. Хотя реализация целостности ссылок для типов 2 или 7 также является нетривиальной задачей, она решается за счет рекурсивного создания ссылок начиная от новой корневой версии. При этом, рекурсивное создание поддерева является опциональным, этот параметр может выбираться пользователем в момент создания новой версии. Аналогично для типов 2 и 7 происходит удаление определенной версии с каскадным удалением всех связанных сущностей.

Таким образом, при проектировании системы SciCMS, был выбран механизм версионирования, основанный на типе 7, предполагающий хранение в отдельном атрибуте CONFIG_ID идентификатора сущности, общего для всех ее версий. В отличие от типа 2, такой подход позволяет отказаться от семантики естественного уникального ключа и делает более гибкими операции с любыми неверсионными атрибутами.

В отсутствие специфических требований по поддержке ссылочной целостности версий наиболее предпочтительным типом SCD является тип 4. Он показывает высокие результаты выполнения основных операций с версиями, прост в реализации и может применяться во всех современных СУБД. При этом рассмотренные способы не ограничиваются только реляционными БД и при необходимости могут быть применены для других типов хранилищ (документо-ориентированных, графовых, «ключ-значение»).

Заключение

Одним из распространенных требований при разработке ИС, ориентированных на работу с данными, является поддержка контроля версий и истории изменений. Постановка задачи версионирования зависит от назначения системы и может быть обусловлена такими факторами, как потребности аудита и/или аналитики, особенности бизнес-процессов на предприятии и другими причинами. При выборе из существующих методов версионирования [1] следует четко определить требования к системе, включая необходимый перечень операций с версиями. Предлагаемый в статье способ сравнительной оценки позволяет наглядно представить эффективность выполнения конкретных операций с версиями на тестовых данных. Помимо полученных показателей, необходимо также учитывать имеющиеся ограничения исследуемых моделей (например, не все типы поддерживают ссылолчную целостность при создании новых версий). Таким образом, решение о выборе наиболее эффективного механизма версионирования должно быть основано на трех составляющих: исходные требования, характеристики моделей и полученные результаты оценки. Несмотря на то, что в сравнительном тестировании используется реляционная СУБД и стандартные SQL-запросы, операции версионирования могут быть адаптированы для любого NoSQL [12] хранилища, обладающего набором CRUD-операций. В результате проведенного исследования при проектировании системы SciCMS была выбрана модель версионирования (тип 7), отвечающая предъявляемым требованиям к эффективности выполнения операций, целостности данных и гибкости работы с неверсионными атрибутами.

Литература

1. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Eition. Indianapolis: John Wiley & Sons, Inc., 2013.
2. Бергер А.Б., Microsoft SQL Server 2005 Analysis Services. OLAP и многомерный анализ данных. Санкт-Петербург: БХВ-Петербург, 2007.
3. Bauer C., King G., Gregory G. Java Persistence with Hibernate. Second Edition, Manning Publications Co., 2016.
4. Малахов Ю.А., Михальченко С.С. Анализ современных решений в области разработки веб-приложений // Сборник научных статей Всероссийской конференции «Автоматизация и моделирование в проектировании и управлении». Брянск, 2022.
5. Content Versioning. URL: https://market.strapi.io/plugins/@notum-cz-strapi-plugin-content-versioning (дата обращения: 12.03.2023).
6. Yungpeng L., Utpal R., Seung-Jun S., Y. Tina L. A "Smart Component" Data Model in PLM // 2015 IEEE International Conference on Big Data. Santa Clara, 2015.
7. Черныш Б.А., Мурыгин А.В. Динамическая схема GraphQL в реализации интегрированной информационной системы // Программные продукты и системы, № 4 (35), С. 561-566, 2022.
8. Bazrov B.M., Kheifetz M.L., Hurevich V.L., Popok N.N. Assessment of production manufacturability of the design in the product life cycle // Proceedings of the National

academy of sciences of Belarus. Phisico-technical series, т. 65, № 4, С. 422-432, 2020.
9. Дамирбек К.Г., Жакыпова Э.А.. Обеспечение целостности информации в автоматизированных информационных системах // Современные проблемы механики, С. 66-41, 2018.
10. Rainardi V. Building a Data Warehouse With Examples in SQL Server. New York: Apress, 2008.
11. Loyola R.C., Sepulveda A.U., Hernandez M.W. Optimization slowly changing dimensions of a data warehouse using object-relational // International Conference of the Chilean Computer Science Society (SCCC), 2015.
12. Adriana J., Holanda M. NoSQL: SQL to NoSQL Databases // Advances in Intelligent Systems and Computing, т. 746, С. 938-948, 2018.

Черныш Борис Александрович. Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнёва, Красноярск, Россия. Аспирант. Область научных интересов: системы управления, информационные технологии. E-mail: borisblack@mail.ru

Мурыгин Александр Владимирович. Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнёва, Красноярск, Россия. Зав. кафедрой информационно-управляющих систем, доктор технических наук, профессор. Область научных интересов: системы управления, информационные технологии. E-mail: avm514@mail.ru

Choosing a Data Versioning Model for Designing Information Systems

B. A. Chernysh, A. V. Murygin

Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia

Abstract. One of the common tasks faced by developers of information systems focused on working with data is the support of versioning and history of storing records. Versioning approaches that exist today are applicable to various application scenarios: online data processing, analytical reporting, distributed storage maintenance, etc. and do not depend directly on the used database. To select the optimal versioning model, it is necessary to determine the key requirements for the system in terms of versioning operations and evaluate the effectiveness of their implementation within each of the available mechanisms. This article provides an overview of these mechanisms, their characteristic differences, advantages and disadvantages, as well as examples of use. A method is proposed for comparative evaluation of these mechanisms on a test relational database using a series of operations on versions. This technique was used by the authors in the design and development of the integrated information system. Based on the initial requirements for the system, the characteristics of the models, and the obtained evaluation results, an analysis of the effectiveness of the studied mechanisms was carried out. The result of the analysis is the choice of the most optimal model in terms of the speed of versioning operations, data integrity, and flexibility in working with attributes.

Keywords: database, version control, versioning, Slowly Changing Dimension, SCD, Hibernate Envers, Aras Innovator, SQL, NoSQL, SciCMS.

DOI 10.14357/20718632230313

References

1. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Eition. Indianapolis: John Wiley & Sons, Inc., 2013.
2. Berger A.B., Microsoft SQL Server 2005 Analysis Services. OLAP i mnogomerniy analiz dannykh [OLAP and multidimensional data analysis]. St/ Petersburg: BHV-Peterburg, 2007.
3. Bauer C., King G., Gregory G. Java Persistence with Hibernate. Second Edition, Manning Publications Co., 2016.
4. Malakhov Y.A., Mikhalchenko S.S. Analiz sovremennykh resheniy v oblasti web-prilizheniy [Analysis of modern solutions in the field of web application development] // Sbornik nauchnykh statey Vserossiyskoy conferentsii «Avtomatizaciya i modelirovanie v proektirovanii i upravlenii» [Collection of scientific articles of the All-Russian conference "Automation and modeling in design and management"]. Bryansk, 2022.
5. Content Versioning. Available at: https://market.strapi.io/plugins/@notum-cz-strapi-plugin-content-versioning (accessed: March 12,.2023).

6. Yungpeng L., Utpal R., Seung-Jun S., Y. Tina L. A "Smart Component" Data Model in PLM // 2015 IEEE International Conference on Big Data. Santa Clara, 2015.
7. Chernysh B.A., Murygin A.V. Dinamicheskaya schema GraphQL v realizatsii integrirovannoy informatsionnoy sistemy [GraphQL Dynamic Schema in Integrated Information System Implementation] // Programmnye produkty i sistemy [Software and systems], № 4 (35), pp. 561-566, 2022.
8. Bazrov B.M., Kheifetz M.L., Hurevich V.L., Popok N.N. Assessment of production manufacturability of the design in the product life cycle // Proceedings of the National academy of sciences of Belarus. Phisico-technical series, vol. 65, № 4, pp. 422-432, 2020.
9. Damirbek K.G., Zhakypova E.A.. Obespecheniye tselostnosti informatsii информации v avtomatizirovannyh informatsionnyh sistemah [Ensuring the integrity of information in automated information systems] // Sovremennye problemy mekhaniki [Modern problems of mechanics], pp. 66-41, 2018.
10. Rainardi V. Building a Data Warehouse With Examples in SQL Server. New York: Apress, 2008.
11. Loyola R.C., Sepulveda A.U., Hernandez M.W. Optimization slowly changing dimensions of a data warehouse using object-relational // International Conference of the Chilean Computer Science Society (SCCC), 2015.
12. Adriana J., Holanda M. NoSQL: SQL to NoSQL Databases // Advances in Intelligent Systems and Computing, vol. 746, pp. 938-948, 2018.

Chernysh Boris A. Postgraduate student, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia. E-mail: borisblack@mail.ru

Murygin Alexander V. PhD, Professor, Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia. E-mail: avm514@mail.ru