DOI 10.37882/2223–2966.2020.08.30

# МЕТОДИЧЕСКИЕ ОСНОВЫ ВЫБОРА ПЛАТФОРМ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ В ИНТЕЛЛЕКТУАЛЬНОМ СИТУАЦИОННОМ ЦЕНТРЕ

## METHODOLOGICAL FOUNDATIONS FOR SELECTING PLATFORMS FOR PRESENTATION OF INFORMATION IN THE INTELLECTUAL SITUATION CENTER

***V. Simankov***
***M. Drilenko***

*Summary.* The goal of the current work is to find the most optimal data storage system for integrating various information models in an intelligent situational center.

To achieve these goals, it is necessary to consider the existing models of information representation in modern non-relational data storage systems, the most effective data representation models are highlighted, which are key-value (Redis), document-oriented (SSDB), columnar (Cassandra) and graph (Neo4j) storage, which will be used for a comprehensive presentation of data in an intelligent situational center [4]. For the parallel application of these information storages, it is necessary to form your own rules for the automatic transition from the logical representation of data for each of the most relevant database platforms; the transition can be achieved by applying a number of transformations.

*Keywords:* data storage system, information, information models, data presentation models, information storage.

***Симанков Владимир Сергеевич***
*Д.т.н., профессор, Кубанский государственный технологический университет*
***Дриленко Максим Владимирович***
*Аспирант, Кубанский государственный технологический университет*
*max@russia.ms*

*Аннотация.* Цель текущей работы заключается в поиске наиболее оптимальной системы хранения данных для интеграции различных информационных моделей в интеллектуальном ситуационном центре.

Для достижения поставленных целей необходимо рассмотреть существующие модели представления информации в современных нереляционных системах хранения данных, выделены наиболее эффективные модели представления данных, которыми являются ключ-значение (Redis), документно-ориентированное (SSDB), колоночное (Cassandra) и графовое (Neo4j) хранилища, которые и будут использоваться для всеобъемлющего представления данных в интеллектуальном ситуационном центре [4]. Для параллельного применения указанных хранилищ информации необходимо сформировать собственные правила автоматического перехода от логического представления данных для каждой из наиболее актуальных платформ баз данных, переход может быть достигнут путем применения ряда преобразований.

*Ключевые слова:* система хранения данных, информация, информационные модели, модели представления данных, хранилища информации.

Для каждой системы можно рассмотреть два типа преобразования[1]: (1) преобразования, генерирующие элементы, необходимые для реализации информации в системы хранения; это элементы, которые должны быть заранее объявлены перед началом подачи данных, и (2) преобразования, создающие директивы поддержки, полезные для реализации обработок; эти директивы дают методы использования атрибутов и реализации отношений.

## Модель Cassandra

Как и в реляционных системах[1], модель данных в Cassandra должна быть заранее зафиксирована. Это касается имени БД, названия столбцов и колонок. Реализация отношений оговорена в руководящих принципах.

a. Элементы, необходимые для реализации преобразования

- Шаг 1: Каждая база данных БД трансформируется в пространство ключей KS, где KS.N = BD.N.
- Шаг 2: Каждая таблица t ∈ BD преобразуется в семейство столбцов f ∈ F, где f.N = t.N, а каждый атрибут $a^t \in t$. At преобразуется в столбец cl, где cl.N = at.N, cl.Ty = at.Ty, а затем добавляется в список столбцов его преобразованного контейнера f, например, cl ∈ f.CL.
- Идентификатор строки t трансформируется в идентификатор строки f, где $PKey^f.N = Idt.N$, затем добавлены в список столбцов f для $PKey^f \in f.CL$.

b. Руководящие принципы поддержки

§ Шаг 3: Как правило, взаимосвязь между таблицами может быть реализована тремя способами: (1) добавление ссылок в одну из связанных таблиц, (2) вложение данных или (3) создание новой таблицы, содержащей ссылки потому, что Cassandra не поддерживает объединение, а только

Таблица 1. Типы справочных столбцов

| Отношение | Решение | Тип контрольной колонки |
|---|---|---|
| **r = (N,{($t_1$,*),($t_2$,1)}))** | Решение 1 | Монозначный |
| | Решение 2 | Многозначный |
| | Решение 3 | Монозначный |
| **r = (N,{($t_1$,1),($t_2$,*)}))** | Решение 1 | Многозначный |
| | Решение 2 | Монозначный |
| | Решение 3 | Монозначный |
| **r = (N,{($t_1$,1),($t_2$,1)}))** | Решение 1 | Монозначный |
| | Решение 2 | Монозначный |
| | Решение 3 | Монозначный |
| **r = (N,{($t_1$,*),($t_2$,*)}))** | Решение 1 | Многозначный |
| | Решение 2 | Многозначный |
| | Решение 3 | Монозначный |

добавление ссылок в одну из связанных таблиц и создание новой таблицы может использоваться для выражения отношений между группами колонок. Оба эти решения касаются использования эталонных столбцов. Опорный столбец — это столбец, который представляет собой ключевое значение соответствующего семейства столбцов. Другими словами, значения ссылочного столбца должны существовать в столбце PKey соответствующей группы столбцов; обратите внимание, что это ограничение (которое аналогично ограничению ссылочной целостности реляционных систем) не управляется автоматически системой Cassandra; его проверка, таким образом, остается ответственностью пользователя. Таким образом, определяется правило, которое преобразует отношения логического уровня следующим образом: для каждого отношения r, связывающего две таблицы $t_1$ и $t_2$, можно рассматривать решения по преобразованию:

**Решение 1:** r преобразуется в столбец $c_1$ со ссылкой на $f_2$ (группу столбцов, соответствующее $t_2$), где cl.N = (f2.N) Ref и cl.Ty = PKey$f_2$.Ty, а затем добавляется в список столбцов $f_1$ (группу столбцов, соответствующую $t_1$) так, что cl $\in$ $f_1$.CL. При инициализации $f_1$, ссылочный столбец cl принимает одно или несколько значений (в соответствии с параметром кардинальности r) Первичного ключа $f_2$.

**Решение 2:** r преобразуется в столбец cl со ссылкой на $f_1$ (семейство столбцов, соответствующее $t_1$), где cl.N = ($f_1$.N) Ref и cl.Ty = PKey$f_1$.Ty, а затем добавляется в список столбцов $f_2$ (семейство столбцов, соответствующее $t_2$) так, что cl $\in$ $f_2$.CL. При инициализации $f_2$, ссылочный столбец cl принимает одно или несколько значений (в соответствии с параметром кардинальности r) Первичного ключа $f_1$.

**Решение 3:** r трансформируется в новое семейство столбцов f, состоящее из двух столбцов $cl_1$ и $cl_2$ со ссылкой соответственно на $f_1$ и $f_2$ (семейства столбцов, соответствующие соответствующим таблицам $t_1$ и $t_2$), где f.N = r.N, f.CL = {$cl_1$, $cl_2$}, $cl_1$.N = ($f_1$.N) Ref, $cl_1$.Ty = PKey$^1$.Ty, $cl_2$.N = ($f_2$.N) Ref и $cl_2$.Ty = PKey.$f_2$.Ty. При конкретизации f, ссылочные столбцы $cl_1$ и $cl_2$ будут принимать значение PKey из $f_1$ и $f_2$ соответственно.

В зависимости от кардинальности r, эталонные колонки, используемые в каждом из этих решений, могут быть однозначными или многозначными. Ниже приведен тип контрольной колонки в соответствии с особенностями отношения и выбранным решением преобразования (Таблица 1).

В отличие от столбцового хранения, применяемого в некоторых реляционных СУБД, хранящих данные по столбцам в сжатом виде, модель «семейство колонок» хранит данные построчно, и обеспечивает высокую производительность, прежде всего, в оперативных сценариях, тогда как для запросов, требующих обхода большого объёма данных с агрегацией результатов, как правило, неэффективна[1][2].

Таким образом, используя колоночное представление информации, с применением ПО Apache Cassandra позволит реализовать системы управления содержимым и регистрацию событий, дополнительное применение временных меток позволит использовать этот вид систем для организации счётчиков, а также регистрации и обработки различных данных, связанных со временем [1].

## Модель SSDB

В SSDB перед вводом данных декларируется только часть модели. Речь идет об указании названия базы данных и названий коллекций. Названия полей, а также способ реализации отношений указаны в руководстве по поддержке.

a. Элементы, необходимые для реализации процесса преобразования

- Шаг 1: Каждая база данных БД трансформируется в базу данных SSDB $BD^{MD}$, где $BD^{MD}$.N = DB.N.
- Шаг 2: Каждая таблица t ∈ BD преобразуется в коллекцию cll ∈ $BD^{MD}$, где cll.N = t.N.

b. Руководящие принципы поддержки

- Шаг 3: Каждый атрибут таблицы $a^t \in t.\ At$ преобразуется в атомное поле $fl^a$, где $fl^a.N = at.N$, $fl^a.Ty = at.Ty$, а затем добавляется к списку полей в его преобразованном контейнере cll типа $fl^a \in cll.FL^a$, где cll — коллекция, соответствующая t-таблице.
- Шаг 4: Идентификатор строки таблицы t трансформируется в идентификатор документов в коллекции cll, соответствующий t, где $Id^{cll}$. N = $Id^t$.N, затем добавляется в список полей cll ($Id^{cll} \in$ cll.FL).
- Шаг 5: Система SSDB позволяет выразить отношения между связанными объектами с помощью полей ссылок или с помощью вложенных документов. Справочное поле — это поле, представляющее собой идентификатор (id) документа, вставленного в другой документ с целью обеспечения взаимосвязи между двумя документами. Другими словами, значения опорного поля должны существовать в поле id ссылающегося документа. Как и Cassandra, это ограничение не управляется автоматически системой SSDB; его проверка остается ответственностью пользователя.

В системе SSDB могут существовать отношения между документами в одном и том же массиве (коллекции) или между документами разных массивов (коллекций). В данной работе все отношения, которые должны быть реализованы, существуют между различными массивами данных (коллекциями). Действительно, коллекция в базе данных SSDB может содержать документы, представляющие объекты различных типов. Например, мы можем хранить в одной коллекции документы, представляющие пациентов, врачей, консультации. В практическом рассмотрении сценарии коллекция (физический уровень) соответствует классу (концептуальный уровень), поэтому в ней содержатся документы, имеющие одинаковую семантику. Таким образом, есть отношения только между отдельными коллекциями.

Определим правило 3, которое преобразует отношения логического уровня следующим образом: для каждого отношения r, связывающего две таблицы $t_1$ и $t_2$, можно рассматривать пять решений преобразования:

**Решение 1:** r трансформируется в поле fl, ссылающееся на документ в $cll_2$ (коллекция, соответствующая $t_2$), где fl.N = ($cll_2$.N) Ref и fl.Ty = $Id^{cll2}$.Ty, а затем добавляется к списку полей в $cll_1$ (коллекция соответствующие $t_1$), например, fl ∈ $cll_1$.FL. При инициализации $cll_1$ в контрольном поле fl берутся одно или несколько значений (в соответствии с кардинальными значениями r; см. таблицу 3.2) идентификатора ida существующего документа в $cll_2$.

**Решение 2:** r трансформируется в поле fl, ссылающееся на документ в $cll_1$ (коллекция, соответствующая t1), где fl.N = ($cll_1$.N) Ref и fl.Ty = $Id^{cll1}$.Ty, а затем добавляется в список полей в $cll_2$ (коллекция, соответствующая $t_2$), например, fl ∈ $cll_2$.FL. При инстанциировании $cll_2$, справочное поле fl принимает одно или несколько значений (в соответствии с кардинальностью r; см. таблицу 3.2) идентификатора id документа, существующего в $cll_1$.

**Решение 3:** r приводит к вложению документа d в $cll_2$ (коллекция, соответствующая t2) в $cll_1$ (коллекция, соответствующая t1), где d ∈ $cll_1$. $FL^{cx}$.

**Решение 4:** r приводит к вложению документа d в cll1 (коллекция, соответствующая t1) в cll2 (коллекция, соответствующая t2), где d ∈ cll2. $FL^{cx}$.

**Решение 5:** r трансформируется в новую коллекцию cll, cll.N = r.N, cll.Fl = {$fl_1$, $fl_2$}, $fl_1$.N = ($cll_1$.N) Ref, $fl_1$.Ty = $Id^{cll2}$.Ty, $fl_2$.N = ($cll_2$.N) Ref и $fl_2$.Ty = $Id^{cll2}$.Ty. $cll_1$ и $cll_2$ — коллекции, соответствующие соответствующим таблицам $t_1$ и $t_2$.

В зависимости от кардинальности r эталонные поля, используемые в решениях 1, 2 и 5, могут быть однозначными или многозначными. В таблице 3.2 приведен тип опорного поля в соответствии с кардинальностью соотношения и выбранным решением преобразования.

Таким образом, документно-ориентированная система хранения служит для интеграции иерархических структур данных и находит своё применение в системах управления содержимым и документальном поиск, что немаловажно для использования в интеллектуальном ситуационном центре. Документы могут быть организованы (сгруппированы) в коллекции. Их можно считать отдельным аналогом таблиц реляционных СУБД, но коллекции могут содержать другие коллекции. Хотя документы коллекции могут быть произвольными, для более эффективного индексирования лучше объединять в коллекцию документы с похожей структурой[1][2].

## Модель Neo4j

Модель данных в системе Neo4j задается при вводе данных. Имя узла, а также имена его свойств задаются пользователем при вставке каждого узла. Таким образом, ни один элемент не должен быть зафиксирован пе-

Таблица 2. Типы контрольных полей

| Отношение | Решение | Тип контрольного поля |
|---|---|---|
| **r = (N,{($t_1$,*),($t_2$,1)}))** | Решение 1 | Монозначный |
| | Решение 2 | Многозначный |
| | Решение 5 | Монозначный |
| **r = (N,{($t_1$,1),($t_2$,*)}))** | Решение 1 | Многозначный |
| | Решение 2 | Монозначный |
| | Решение 3 | Монозначный |
| **r = (N,{($t_1$,1),($t_2$,1)}))** | Решение 1 | Монозначный |
| | Решение 2 | Монозначный |
| | Решение 5 | Монозначный |
| **r = (N,{($t_1$,*),($t_2$,*)}))** | Решение 1 | Многозначный |
| | Решение 2 | Многозначный |
| | Решение 5 | Монозначный |

ред подачей питания. Для Neo4J генерируются только директивы поддержки.

a. Руководящие принципы поддержки

- Шаг 1: Каждая таблица t ∈ BD преобразуется в узел v ∈ V, где v.L = t.N, а каждый атрибут $a^t$ ∈ t.At преобразован в состояние $pr^v$, где $pr^v$.N = at.N, $pr^v$.Ty = $a^t$.Ty, затем добавил в список свойств своего трансформированного контейнера v: $pr^v$ ∈ v.PR$^v$, при этом идентификатор строки t трансформируется в идентификатор узла v, где $Id^v$.N = $Id^t$.N, затем добавляется в список свойств $Id^v$ ∈ v.$PR^v$. Кроме того, мы связываем ограничение «Is Unique» с $pr^v$.
- Шаг 2: Каждое отношение r, связывающее две таблицы $t_1$ и $t_2$, преобразуется в дугу e, связывающую два узла $v_1$ и $v_2$, соответствующие связанным таблицам, где e.L = r.N, e.($extrem_1$, $extrem_2$) = ($v_1$, $v_2$).

Таким образом, графовая модель представления информации может применятся для задач, в которых данные имеют большое количество связей, например, социальные сети и выявление всевозможных пересечений объектов между собой, что увеличивает операционный функционал интеллектуального ситуационного центра.

## Модель Redis

В режиме Redis ни одна модель не должна быть настроена заранее перед включением питания. Другими словами, ввод данных осуществляется без предварительной проверки физической модели. Последнее указывается при вводе данных; пользователь вставляет каждую пару (ключ: значение), указывая свой ключ, а также ключи входящих в него пар, если это сложная пара.

Таким образом, процесс трансформации генерирует не физическую модель данных, специфичную для системы Redis, а набор руководящих принципов, которые дают пользователю условия использования ключей и реализации отношений.

a. Руководящие принципы поддержки

- Шаг 1: В Redis, понятие «таблица» не является явным. Таким образом, мы вводим правило, что все пары 1-го уровня в хранилище, имеющие ключ формы XXXX_i, с i любым числовым индексом, будут считаться принадлежащими к таблице XXXX. В процесс трансформации и по этому принципу каждая пара ср$^{cx}$ в хранилище Redis будет соответствовать строке таблицы t, которая существует на логическом уровне, где ср$^{cx}$.Key = [t.N]_i; i — это индекс, относящийся к строке t, при этом каждый атрибут $a^t$ ∈ t. At трансформируется в пару ср$^a$, где ср$^a$.Key = at.N, затем добавляется в список пар его преобразованного контейнера ср$^{cx}$ например ср$^a$ ∈ ср$^{cx}$.Value;
- Шаг 2: Отношения создаются в Redis с использованием ссылочных пар. Опорная пара — это пара, значение которой может быть атомным (соответствует ключу опорной пары) или комплексным (состоит из других атомных опорных пар). Таким образом, для каждого отношения r, существующего на логическом уровне между двумя $t_1$ и $t_2$, можно рассмотреть три решения:

**Решение 1:** r трансформируется в пару $cp^a{}_{ref}\,cp_2{}^{cx}$ (пара соответствующая строке $t_2$), где $cp^a f$.Key = [$t_2$.N] Ref и $cp^a{}_{ref}$.Value = $cp_2{}^{cx}$.Key, а затем добавлена в список пар на $cp^{cx}$ (соответствующая пара до строки $t_1$) $cp^a \in cp^{cx}$.Value.

**Решение 2:** r трансформируется в пару $cp^a{}_{ref}$ и ссылается на $cp_1{}^{cx}$ (пара соответствующая строке t1), где $cp^a{}_{ref}$.key = [t1.N] Ref и $cp^a{}_{ref}$.Value = $cp^{cx}$.Key, а затем добавлена в список пар на $cp^{cx}$ (соответствующая пара в строке $t_2$) $cp^a{}_{ref} \in cp^{cx}$.Value.

**Решение 3:** r трансформируется в пару $cp^{cx}_{ref}$, состоящую из двух справочных пар $cp^{a}_{ref1}$ и $cp^{a}_{ref2}$, соответственно ссылка $cp_1^{cx}$ и $cp_2^{cx}$, где $cp^{cx}_{ref}$.Key = r.N, $cp^{cx}_{ref}$.Value = {$cp^{a}_{ref1}$, $cp^{a}_{ref2}$}, $cp^{a}_{ref1}$.Key = (t1.N) Ref, $cp^{a}_{ref1}$.Value = $cp^{cx}_{1}$. Key, $cp^{a}_{ref2}$.Key = ($t_2$.N) Ref, $cp^{a}_{ref2}$.Value = $cp^{cx}_{2}$.Key.

Таким образом, рассмотренная нереляционная модель «ключ-значение» реализованная на основе ПО Redis является вариантом, использующим ключ для доступа к значению. Такие системы используются для хранения изображений, создания специализированных файловых систем, в качестве кэшей для объектов, а также в системах, спроектированных с прицелом на масштабируемость [3].

Кроме того, без применения такой модели данных невозможно будет интегрировать стандартные реляционные источники данных, ввиду чего требуется использование в том числе данной модели.

## Выводы

Исходя из рассмотренных переходов к каждой указанной информационной модели можно сделать вывод, что все рассмотренные модели необходимо использовать при интеграции данных в интеллектуальном ситуационном центре для обработки неструктурированной и слабоструктурированной информации на основе патернов [4].

Следовательно, необходимо интегрировать все информационные модели: ключ-значение, документно-ориентированное, колоночное и графовое хранилища для всеобъемлющего представления данных в интеллектуальном ситуационном центре.

Такая интеграция потребует преобразования концептуальной модели данных в физическую модель данных, для чего в работе дополнительно представлены различные этапы реализации переходов к существующим четырем типам организации информации. Разработанные подходы можно использовать для нескольких решений реализации отношений между таблицами на логическом уровне, для реализации ссылок или вложенности. Выбор решения, который делается в соответствии с критериями, специфичными для каждой рассмотренной модели данных влияет на эффективность. В данной работе предложен собственный подход к реализации интеграции данных.

ЛИТЕРАТУРА

1. Len Silverston — The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises.— Принстон, США: Wiley Publishing, 2019;
2. David C. Hay — Enterprise Model Patterns: Describing the World (UML Version).— Энн-Арбор, США: Technics Publications, LLC, 2019;
3. Michael Blaha — Patterns of Data Modeling (Emerging Directions in Database Systems and Applications).— Вашингтон, США: CRC Press, 2019;
4. Martin Fowler — Analysis Patterns: Reusable Object Models.— Энн-Арбор, США: CRC Press, 2019.



Журнал «Современная наука: актуальные проблемы теории и практики»