Горнак Данила Андреевич,
МГТУ «СТАНКИН»

Научный руководитель:
Стоякова Ксения Леонидовна, к.п.н., доцент
Кафедра информационных технологий и
вычислительных систем МГТУ «СТАНКИН»

ЗНАЧЕНИЕ ВЕРСИОННОСТИ И ИСТОРИЧНОСТИ ДАННЫХ ПРИ ПОСТРОЕНИЕ ЕДИНОГО ХРАНИЛИЩА ДАННЫХ ПО ПРЕДМЕТНЫМ ОБЛАСТЯМ

Аннотация: Статья посвящена важности версионности и историчности данных при построении единого хранилища данных по предметным областям. В условиях роста объемов информации и необходимости оперативного анализа данных, версионность и историчность играют ключевую роль в обеспечении достоверности и точности данных.

Ключевые слова: историчность данных, хранилище данных, методология Кимбалла, таблицы измерений.

Когда речь заходит о построение хранилища данных (DWH – Data Warehouse), выбирается из двух основных подхода построения таких хранилищ данных, а именно, методология Кимбалла и методология Инмона. В данной статье мы не будем сравнивать эти два подхода к построению DWH, рассмотрим абстрактное хранилище, которое строится по методу Кимбалла. Для более полного понимания о том, как сроится версионности и историчности данных, в каких таблицах оно используется, а в каких нет, нам потребуется немного углубится в данную методологию.

Методология Кимбалла была разработана в конце 1980-ых – начале 1990-ых годов, эта методология фокусируется на том, что построенные хранилища данных с ее помощью, хорошо поддерживаются, легко масштабируются и обеспечивают быстрый доступ к данным для пользователей. Основные принципы методологии по Кимбаллу следующие:

1. Фокус на бизнес – потребности. Это подразумевает, что проектирование и построение хранилища данных определяется с потребностей бизнес.
2. Многомерная модель данных, данная методология позволяет быстро строить, поскольку не требует нормализации, в свою очередь, это дает ускорение в проектирование хранилища данных при проектировании
3. Таблицы измерений – данный тип таблиц хранит в себе описательные данные, которые характеризуют факты, например: информация о клиентах, договорах и банковских карт.
4. Фактические таблицы или таблицы фактов. Данный тип таблиц содержат в себе числовые данные, которые можно суммировать или агрегировать, например: транзакции по картам, количество заказов и т.д. Каждая строчка данных в таблице фактов, связана с одной или несколькими строчкой данных из таблицы измерений
5. Постепенная интеграция, вместо того чтобы с нуля создать единое централизованное хранилище данных сразу, методология Кимбалла предлагает строить витрины данных по отдельности, затем постепенно интегрировать их между с собой в общее хранилище.

После определения основных принципов методологии Кимбалла, рассмотрим более подробно пункты 3 и 4.

Рассмотрим пункт 4, как сказано выше, в фактических таблицах или таблицах фактов хранятся числовые данные, которые можно суммировать или агрегировать для наглядности. Отобразим пример подобной таблицы.

За основу возьмем транзакцию, совершенную с помощью банковской карты, тогда у нас будут следующие столбцы:

1. operation_id – это поле содержит в себе id транзакции или первичный ключ. Первичный ключ – это уникальный идентификатор записи в таблице баз данных;
2. card_id – данное поле хранит в себе id карты с помощью, которой совершалась транзакция, также данное поле служит вторичным ключом. Вторичный ключ – это любое индексированное поле или набор полей, по которому происходит поиск записей, с его помощью можно связывать таблицы между собой. Вторичный ключ не гарантирует уникальность записи, в отличие от первичного ключа;
3. operation_dt – хранит в себе информация о дате и времени совершения операции;
4. ccy_amt – данное поле отображает сумму операции;
5. insert_dt – содержит в себе информацию о дате и времени вставки записи в таблицу.

С учетом вышеуказанных столбцов таблица будет иметь следующий вид (см. Таблица 1):

Таблица 1

operation_id	*card_id*	*operation_dt*	*ccy_amt*	*insert_dt*
1010	*1015*	*2025-03-13 08:16:56*	*150.00*	*2025-03-13 14:00:00*
1011	*1025*	*2025-03-13 11:46:32*	*500.00*	*2025-03-13 14:00:00*
1012	*1015*	*2025-03-14 12:32:46*	*1000.00*	*2025-03-14 14:00:00*

После того как мы представили таблицу, можно говорить о версионности данных или историчности. В многомерном моделирование версионность или историчность данных называется Slowly Changing Dimensions (SCD) – редко изменяющиеся измерения, то есть измерения, в которых не ключевые атрибуты имеют тенденцию изменятся. Атрибут – в DWH это характеристика или свойство измерения, используемого для описания фактов, если рассматривать Таблица 1, то атрибуты здесь: operation_id card_id, operation_dt, ccy_amt и insert_dt. Всего существует 3 основных типа SCD, которые определяют, как может быть отображена история изменения в модели. Далее рассмотрим все существующие типы.

В основном версионность данных используется для таблиц измерений, а не таблиц фактов, поэтому для большей наглядности создадим еще одну таблицу, но на этот раз это будет таблица измерений. За основу возьмем таблицу, описывающую банковскую карту, таким образом у нас будут следующие атрибуты:

1. sid – уникальный ключ для каждой записи или первичный ключ;
2. card_id – id карты;
3. client_name – имя и инициалы держателя карты;
4. card_dt – дата окончания действия карты;
5. status – статус карты, принимает в себя два значения active и close, то есть активная/закрытая карта;
6. insert_dt – поле содержит в себе информацию о дате и времени вставки записи в таблицу.

Таблица будет выглядит следующим образом (см. Таблица 2)

Таблица 2

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Active*	*2025-02-01 14:56:32*

Нулевой тип или SCD – 0, суть типа заключается в том, что данные после первого попадания в таблицу больше никогда не изменяются, данный тип похож на копилку, мы постоянно накапливаем данные и далее с ними ничего не происходит. Рассмотрим на примере таблицы: добавляются новые строчки данных, не затрагивая уже существующие. Данный метод редко используется и чаще всего используется как начальная точка отсчета SCD.

Первый тип или SCD – 1, говоря простым языком, это перезапись старой строчки данных на новую. Примеры рассматриваются в Таблица 2, так как эта таблица является таблицей измерений, будем изменять атрибут status_code, изменять статус «активный» на «закрытый». Итак, если мы применяем первый тип SCD, то таблица будет выглядит следующим образом (см. Таблица 3).

Таблица 3

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Closed*	*2025-03-01 13:26:32*

Мы изменили значение атрибута status_code на closed (отмечено красным), также изменился атрибут insert_dt, так как мы перезаписали строчку данных. Как можно заметить данная таблица не хранит предыдущее значение, и нельзя посмотреть, когда и какой атрибут изменился после перезаписи строчки данных. Также данный тип можно использовать и для таблицы фактов. Рассматривая пример в Таблица 1, можно предположить, что изменится сумма транзакции: к ней добавится комиссия за оплату.

Теперь рассмотрим второй тип SCD или SCD – 2. Данный тип SCD уже хранит в себе информацию о том, когда атрибут изменился, а также хранит в себе предыдущее состояние данных. При использование такого типа SCD добавляются два новых столбца – start_dt и end_dt. Start_dt хранит в себе информацию о дате начала записи, то есть если наша запись появилась 2025-03-01, то и start_dt будет таким же. End_dt, в свою очередь, хранит информацию о дате окончания действия записи, изначально значение определено 9999-12-31, такое обозначение используется для актуальной записи, которая не была изменена. Таблица с добавлением новых полей будет выглядеть следующим образом (см. Таблица 4).

Таблица 4

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*	*start_dt*	*end_dt*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Active*	*2025-02-01 14:56:32*	*2025-02-01*	*9999-12-31*

Теперь рассмотрим пример, как и с SCD – 1 также изменим атрибут status_code на closed. Теперь таблица будет выглядеть следующим образом (см. Таблица 5).

Таблица 5

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*	*start_dt*	*end_dt*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Active*	*2025-02-01 14:56:32*	*2025-02-01*	*2025-02-28*
2	*1015*	*Артем Г. А.*	*2035-03-01*	*Closed*	*2025-03-01 13:26:32*	*2025-03-01*	*9999-12-31*

Как видно, теперь в таблице появилась вторая запись с измененными атрибутами status и insert_dt, а атрибут end_dt строки с id = 1 изменился с 9999-12-31 на 2025-02-28. End_dt для предыдущей записи формируется как: start_dt новой записи минус один день. При использование такого метода возможно отследить полную историчность данных и увидеть все происходящие с ними изменения. Минус такого подхода заключается в бесконечности растущей таблицы, что может повлечь за собой замедление обработки таких таблиц. Плюсом такого подхода является полная историчность и отслеживание всех изменений.

Следующим рассмотрим третий тип SCD-3. Смысл данного метода заключается в том, что мы храним предыдущее состояние атрибута в дополнительном столбце. Приведем пример (см. Таблица 6).

Таблица 6

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*	*Previous_status*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Active*	*2025-02-01 14:56:32*	*NULL*

Теперь изменим значение атрибута status на closed, таким образом таблица будет иметь следующий вид (см. Таблица 7).

Таблица 7

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*	*Previous_status*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Closed*	*2025-03-01 12:36:32*	*Active*

По данному методу можно отследить текущий и предыдущий статусы карты, если у карта опять станет активной, то таблица будет выглядеть следующим образом (см. Таблица 8).

Таблица 8

sid	*card_id*	*client_name*	*card_dt*	*status*	*insert_dt*	*Previous_status*
1	*1015*	*Артем Г. А.*	*2035-03-01*	*Active*	*2025-04-01 18:56:32*	*Closed*

Теперь поговорим о плюсах и минусах данного подхода: основным плюсом, по сравнению с SCD-2, является следующее: таблица не растет вглубь, то есть не расширяется, что не будет влиять на скорость чтения таблицы; простой и быстрый доступ к истории изменения данных. Главным минусом при выборе такого метода будет наличие ограниченной историчности, то есть мы должны заранее знать какие атрибуты могут быть изменены. Предположим, что существует таблица из 50 столбцов, где из всех столбцов могут изменяться только 25, в конечном итоге эта таблица будет хранить в себе дополнительных 25 столбцов для отслеживания историчности данных вместо изначальных 50. Также существует вероятность того, что не были учтены все столбцы, которые могут изменяться и придется заново перестраивать таблицу.

Существуют и другие типы SCD, но все они являются гибридами первых трех типов и предназначены для хранения данных специализированных таблиц, где необходимо использовать сразу несколько способов отслеживания историчности данных.

Подводя итоги по рассмотренным методам хранения историчности данных, можно сделать вывод о том, что каждый тип должен подбираться в зависимости от условий архитектуры DWH, ресурсов, выделенных на построение хранилища. Для самого простого отслеживания историчности рекомендуется использовать SCD – 2, он позволяет отслеживать всю историчность и не расширяет таблицу, SCD – 3 подходит для таблиц, где происходят заранее прогнозируемые изменения, SCD – 1 самый простой тип, при котором нет необходимости отслеживать историчность данных.

Список литературы:

1. Концепции хранилищ данных: подход Кимбалла против Инмона. – Текст: электронный // Astera: [сайт]. – URL: https://www.astera.com/ru/type/blog/data-warehouse-concepts/ (дата обращения: 14.03.2025).

2. Версионность и история данных. – Текст: электронный // Хабр: [сайт]. – URL: https://habr.com/ru/articles/101544/ (дата обращения: 14.03.2025)

3. Коллектив авторов, DAMA-DMBOK свод знаний по управлению данными. – 2-е изд. – Москва: Олимп–Бизнес, 2021. – 799 с. – Текст: непосредственный.