Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э.Баумана (национальный исследовательский университет)»

На правах рукописи

Гапанюк Юрий Евгеньевич

Проектирование метаграфовой СУБД «MetagraphDB». Версия 0.1

Москва — 2021

Глава 1. Система управления базами данных на основе метаграфовой модели

1.1 Проектирование реляционной схемы данных для хранения метаграфовой информации

В данном разделе, на основе ранее проведенных экспериментов, разрабатывается модель данных в нотации СУБД PostgreSQL для хранения метаграфовой информации.

Поскольку терминология ER-модели и метаграфовой модели отчасти пересекается, то для того, чтобы избежать многозначности, будем использовать следующие термины:

1. Атрибут – атрибут метаграфовой модели, в соответствии с определением.
2. ER-атрибут – атрибут в модели сущность-связь.

При создании схемы данных будем использовать подход на основе преобразования метаграфа к плоскому графу, рассмотренный в [1; 2].

1.1.1 Хранение вершин, метавершин и ребер

Для вершин, метавершин и ребер могут быть предложены три варианта хранения:

1. Для хранения вершин, метавершин и ребер используются три отдельных сущности.
2. Для хранения вершин и метавершин используются первая сущность, а для хранения ребер вторая сущность.
3. Для хранения вершин, метавершин и ребер используется общая сущность.

С точки зрения метаграфовой модели, варианты 1 и 2 позволяют добиться наиболее точного соответствия между формальными описаниями вершины, метавершины и ребра и их представлениями на уровне схемы данных.

Таблица 1 — Типы вершин-предикатов

Значение	Описание
0	Метавершина или вершина
1	Ребро

Также вершина может рассматриваться как частный случай метавершины, которая не содержит вложенных элементов. Поэтому вариант 2 позволяет объединить вершины и метавершины в единую сущность, которая соответствует вершине-предикату. Преимуществом данного варианта является уменьшение количества сущностей, что позволяет упростить программную реализацию.

Но с точки зрения преобразования метаграфа к плоскому графу предпочтительным является вариант 3. Фактически в этом случае сущность используется для хранения вершины-предиката со стереотипом, который может соответствовать вершине, метавершине или ребру. Назовем данную сущность «вершина-предикат», также для физической схемы данных будем использовать название «nodes». Для обеспечения программной реализации введем следующие ER-атрибуты:

1. nodeid – числовой первичный ключ, используемый для создания схемы данных.
2. id – уникальный строковый ключ вершины-предиката.
3. nodetype – тип вершины-предиката.

Возможные типы вершин-предикатов (значения ER-атрибута «nodetype») представлены в таблице 1.

Необходимо отметить, что для «метавершины» и «вершины» не следует вводить различные статусы, так как в случае добавления к вершине «вершин-предикатов» нижнего уровня, данная вершина автоматически превращается в метавершину.

Сущность «вершина-предикат» содержит следующие связи с другими сущностями:

1. Соединения между вершинами, метавершинами и ребрами.
2. Множество атрибутов, принадлежащих метавершине/вершине/ребру.

Таблица 2 — Типы вершин-предикатов

Значение	Описание
0	Включение в метавершину №1 другой метавершины, вершины или ребра
1	Соединение ребра №1 с исходной вершиной/метавершиной №2.
2	Соединение ребра №1 с конечной вершиной/метавершиной №2.

1.1.2 Точки соединения между вершинами, метавершинами и ребрами

Для обеспечения соединения между вершинами, метавершинами и ребрами введем сущность «точка_соединения», также для физической схемы данных будем использовать название «linkpoints». Сущность содержит информацию о соединении «вершины-предиката» №1 и «вершины-предиката» №2. Для обеспечения программной реализации введем следующие ER-атрибуты:

1. linkid – числовой первичный ключ, используемый для создания схемы данных.
2. nodeid1 – числовой вторичный ключ первой вершины-предиката.
3. nodeid2 – числовой вторичный ключ второй вершины-предиката.
4. id1 – уникальный строковый ключ первой вершины-предиката.
5. id2 – уникальный строковый ключ второй вершины-предиката.
6. linktype – тип связи между первой и второй вершинами-предикатами.

Возможные связи между первой и второй вершинами-предикатами представлены в таблице 2.

1.1.3 Хранение атрибутов

Для хранения атрибутов также могут быть предложены два варианта:

1. Поскольку атрибут может быть представлен как частный случай метавершины, содержащий имя и значение, то для хранения атрибутов могут быть использованы сущности метавершин/вершин и ребер.
2. Для хранения атрибутов может быть использована отдельная сущность.

Преимуществом варианта 1 является сокращение количества сущностей и унификация способов хранения различных элементов метаграфовой модели.

При этом, у варианта 1 имеется существенный недостаток, связанный с тем, что каждый атрибут необходимо преобразовать в метавершину атрибута; две вершины, соответствующие наименованию и значению атрибута; а также ребро для связи вершин. Поскольку данные преобразования необходимо выполнять как при записи, так и при чтении атрибута, то это может привести к снижению производительности при обработке большого количества атрибутов.

Поэтому, для реализации будем использовать вариант 2. Назовем данную сущность «атрибут», также для физической схемы данных будем использовать название «attributes».

Сущность «атрибут», в соответствии с определением, содержит следующие ER-атрибуты (в список также включены ER-атрибуты, добавленные для обеспечения программной реализации):

1. attrid – числовой первичный ключ, используемый для создания схемы данных.
2. nodeid – числовой вторичный ключ, используемый для связи с сущностью «вершина-предикат».
3. id – уникальный строковый ключ вершины-предиката, к которой принадлежит атрибут.
4. system – логическое значение. Если значение истинно, то атрибут считается «системным», то есть предназначен для хранения параметров метаграфовой модели. Если значение ложно, то атрибут предназначен для хранения данных моделируемой предметной области.
5. name – наименование атрибута.
6. type – тип данных атрибута.

В список ER-атрибутов также должно входить и значение атрибута. Но в предлагаемой модели значение атрибута является частью механизма темпоральности, который рассмотрен в разделе **??**.

ER-атрибут «type» предназначен для хранения возможных типов данных атрибута. Используемые типы данных атрибутов представлены в таблице 3.

Таблица 3 — Типы данных атрибутов

Тип данных	Описание
string	строка
int	целое число
float	вещественное число
bool	логическое значение
datetime	дата-время
ref	ссылка на «вершину-предикат»

Таблица 4 — Системные атрибуты

Атрибут	Тип данных	Описание
name	string	Наименование метавершины/вершины/ребра
text	string	Текстовое описание метавершины/вершины/ребра
directed	bool	Признак направленности ребра
node_start	bool	Исходная метавершина/вершина ребра
node_end	bool	Конечная метавершина/вершина ребра

1.1.4 Системные атрибуты

Системные атрибуты предназначены для хранения параметров метаграфовой модели и представлены в таблице 4.

1.1.5 Изменяемость элементов метаграфовой модели

Под **изменяемостью** элементов метаграфовой модели будем понимать способность элементов модели изменяться с течением времени.

Для обозначения изменяемости элементов метаграфовой модели будем использовать традиционные термины, применяемые в функциональном программировании для обозначения изменяемости переменных: изменяемые (mutable) и неизменяемые (immutable). Рассмотрим особенность изменяемости элементов модели:

1. Метавершины/вершины/ребра не могут быть изменяемыми, так как не содержат изменяемых элементов, они могут или присутствовать или отсутствовать в базе данных (immutable).
2. Точки соединения между вершинами, метавершинами и ребрами также не могут быть изменяемыми, они могут или присутствовать или отсутствовать в базе данных (immutable).
3. Атрибуты могут быть изменяемыми. Мы предполагаем, что имя и тип данных атрибута неизменны. Но значение атрибута может меняться во времени (mutable).

Реализация темпоральности для метаграфовой модели

Под **темпоральностью** метаграфовой СУБД будем понимать способность отслеживания изменений в метаграфовых данных с течением времени.

Очевидно, что понятия изменяемости и темпоральности тесно связаны, и для различных вариантов изменяемости могут быть предложены различные способы реализации темпоральности.

В данном разделе рассмотрим способы реализации темпоральности для элементов метаграфовой модели на основе особенностей изменяемости этих элементов.

Но прежде всего необходимо отметить, что реализация темпоральности в обязательном порядке предполагает наличие шкалы времени. Также возникает вопрос о том, какие события должны наноситься на временную шкалу. В качестве таких событий в предлагаемой реализации используются события сохранения элементов метаграфовой модели в базу данных. В качестве временной метки используется время сервера на котором производится обработка метаграфовой информации.

При добавлении данных используются следующие варианты работы с временной меткой:

1. Если временная метка на задана явно, то в качестве временной метки используется текущее значение даты и времени сервера (с учетом временного пояса).

Таблица 5 — Состояния неизменяемых элементов

Значение	Описание
0	Элемент добавлен.
1	Элемент удален.
2	Элемент восстановлен.

2. При добавлении данных создается символическое значение временной метки. Символическое значение ассоциируется с текущим на момент создания метки значением даты и времени сервера (с учетом временного пояса).

Для хранения символических значений временных меток используется сущность «временная_метка» (также для физической схемы данных будем использовать название «time_table»), которая содержит следующие ER-атрибуты:

1. tid - числовой первичный ключ, используемый для создания схемы данных.
2. time_label – символическое значение временной метки.
3. time – значение временной метки – текущее значение даты и времени сервера (с учетом временного пояса).

Реализация темпоральности зависит от изменяемости элемента. К неизменяемым элементам (immutable) относятся метавершины/вершины/ребра и точки соединения между вершинами, метавершинами и ребрами.

Неизменяемость элемента говорит о том, что он может или присутствовать, или отсутствовать в модели. Для неизменяемых элементов возможны состояния, представленные в таблице 5.

В соответствии с таблицей 5 первоначально элемент добавляется в модель, затем может быть удален из модели. При удалении элемент не удаляется физически из базы данных, а помечается удаленным состоянием и в дальнейшем может быть восстановлен.

Для реализации темпоральности неизменяемых элементов используется сущность «темпоральность_неизменяемых» (также для физической схемы данных будем использовать название «temporal_immutables»), которая содержит следующие ER-атрибуты:

1. tiid - числовой первичный ключ, используемый для создания схемы данных.
2. nodeid – числовой вторичный ключ вершины-предиката.
3. linkid – числовой вторичный ключ, используемый для связи с точками соединения.
4. immstatus – состояние неизменяемого элемента в соответствии с таблицей 5.
5. time – значение временной метки, указанное при выполнении операции добавления, удаления или восстановления. – текущее значение даты и времени сервера (с учетом временного пояса).

К изменяемым элементам (mutable) относятся атрибуты.

Для реализации темпоральности изменяемых элементов используется сущность «темпоральность_изменяемых» (также для физической схемы данных будем использовать название «temporal_mutables»), которая содержит следующие ER-атрибуты:

1. tmid - числовой первичный ключ, используемый для создания схемы данных.
2. attrid – числовой вторичный ключ, используемый для связи с атрибутами.
3. value – значение атрибута (в текстовом формате).
4. time – значение временной метки, указанное при выполнении операции добавления, удаления или восстановления. – текущее значение даты и времени сервера (с учетом временного пояса).

1.1.6 Идентификация элементов метаграфовой модели с помощью ключей

При описании вершин-предикатов, соответствующих метавершине/вершине/ребру используется ER-атрибут id – уникальный строковый ключ вершины-предиката.

Данный строковый ключ генерируется на основе стандарта RFC 4122 [3]. Этот стандарт предполагает, что ключи могут генерироваться независимо в различных узлах распределенной системы, при этом вероятность генерации одинаковых ключей пренебрежимо мала.

1.1.7 Итоговая схема данных для описания метаграфовой модели

Схема данных для описания метаграфовой модели представлен на рис. 1.1.

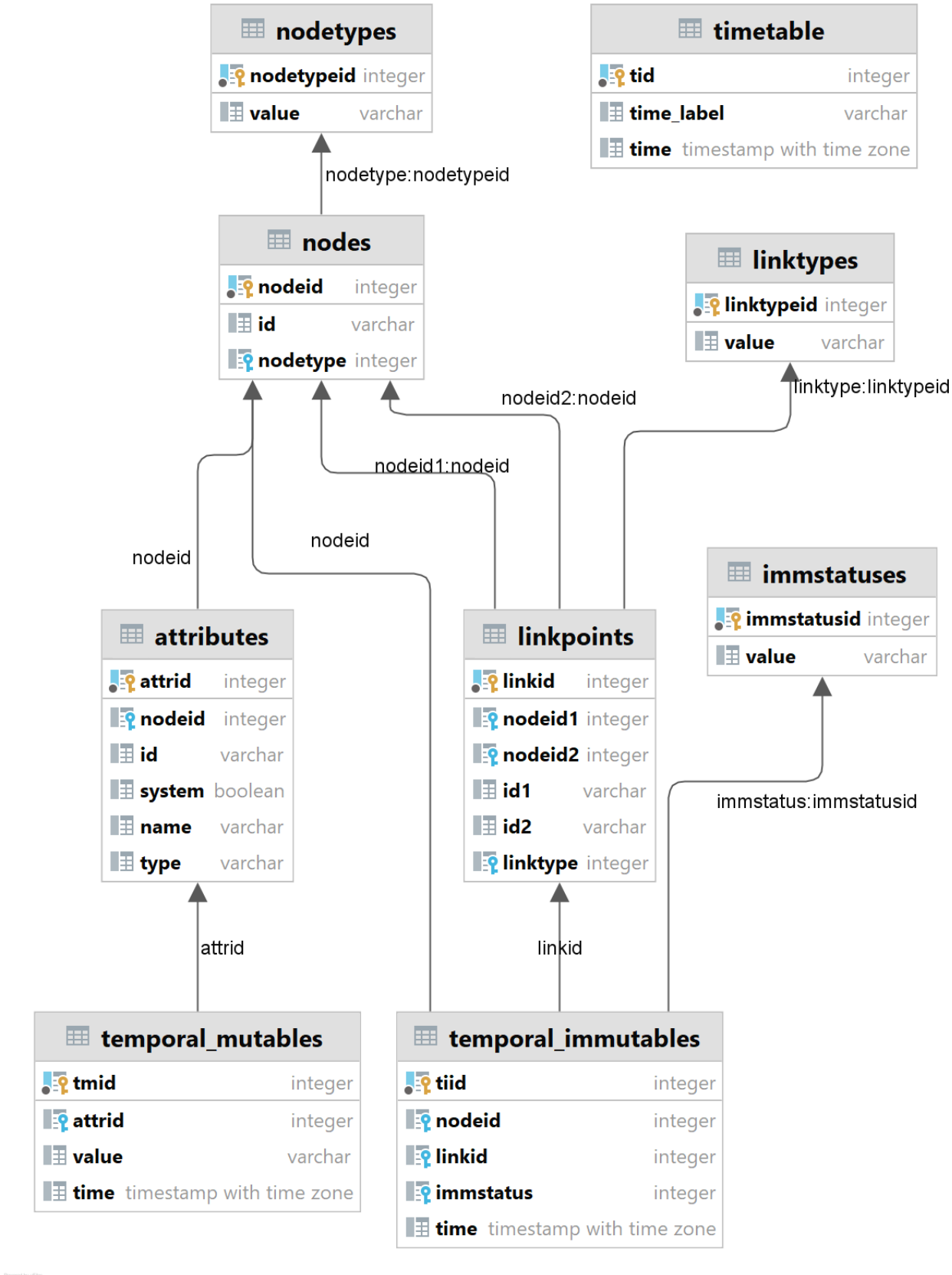

Рисунок 1.1 — Схема данных для описания метаграфовой модели.

Список литературы

1. Storing Metagraph Model in Relational, Document-Oriented, and Graph Databases [Text] / V. Chernenkiy [et al.] // Selected Papers of the XX International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2018). — 2018. — P. 82—89. — URL: http://ceur-ws.org/Vol-2277/paper17.pdf.
2. *Дунин, И. В.* Особенности преобразования метаграфа в модель плоского графа [Текст] / И. В. Дунин, Ю. Е. Гапанюк, Г. И. Ревунков // Динамика сложных систем – XXI век. — 2018. — Т. 12, № 3. — С. 47—51.
3. A Universally Unique IDentifier (UUID) URN Namespace [Text]. — 2005. — URL: https://datatracker.ietf.org/doc/html/rfc4122 (visited on 09/27/2021).