УДК 004.652 DOI: 10.33310/2524-0978-2019-1-7-80-87

Михайло ЩЕРБИНА
profiler4100v2@gmail.com
ORCID: 0000-0001-7982-1298
Андрій ОХРИМЕНКО
okhrymenkoandrii@gmail.com
ORCID: 0000-0003-2468-1958
Світлана КУЗНІЧЕНКО
skuznichenko@gmail.com
ORCID: 0000-0001-7982-1298
м. Одеса

ПРОГРАМНА СИСТЕМА СТВОРЕННЯ ТА ЗБЕРІГАННЯ EAV СТРУКТУР ДАНИХ

Розроблено програмну систему створення та зберігання EAV структур даних для фреймворка YII2. Наведено призначення системи, її основні функціональні можливості, а також засоби реалізації.

***Ключові слова**: Entity-atribute-value модель, база даних, фреймворк YII2, NOSQL, SQL, візуальний редактор WYSIWYG.*

Постановка проблеми

Розробка програмного продукту (ПП) є складним процесом, який з часом часто потребує часткової, а іноді і повної реструктуризації програмного забезпечення (ПЗ). В якості вирішення проблеми можна розглянути розробку ПЗ, що дозволяє швидко створювати прототип майбутньої інформаційної системи із змінною структурою метаданих. Це дозволить поєднати проектування бази даних (БД) з конструюванням, а також знизити ризики від помилок при проектуванні БД.

Подібним альтернативним підходом до організації БД є адаптована вертикальна модель даних «Сутність-атрибут-значення» (Entity-attribute-value model, EAV). Модель EAV – це модель даних, що дозволяє описати сутності, в яких кількість атрибутів (властивостей, параметрів), що характеризують їх, потенційно величезна, але та кількість, яка реально буде використовуватися в конкретній сутності, відносно мала [1].

Класична модель EAV складається з трьох таблиць (рис. 1) [2]:

Entity – таблиця відповідає за сутність в базі даних (аналог об'єктів в ООП).

Attributes – таблиця відповідає за атрибути в базі даних (аналог атрибутів в ООП). Таблиця атрибутів може містити такі стовпці: ідентифікатор атрибуту, назва атрибуту, опис, тип даних та стовпці, які допомагають перевірити вхідні дані, наприклад, максимальна довжина рядка та регулярний вираз, набір дозволених значень і т.п.

Values – таблиця відповідає за значення атрибутів.

Перевагою моделі EAV є гнучкість рішення і високий рівень універсальності. До недоліків моделі можна віднести:

– складність дотримання цілісності даних, в результаті чого погіршується їх якість;

– складність запитів для проведення операцій над даними;

– низька продуктивність виконання запитів.

На даний час існують лише напрацювання ПЗ для зберігання EAV структур для фреймворку Yii2, що не були доведені до готового ПП, тому актуальним завданням є створення подібного відкритого ПЗ.

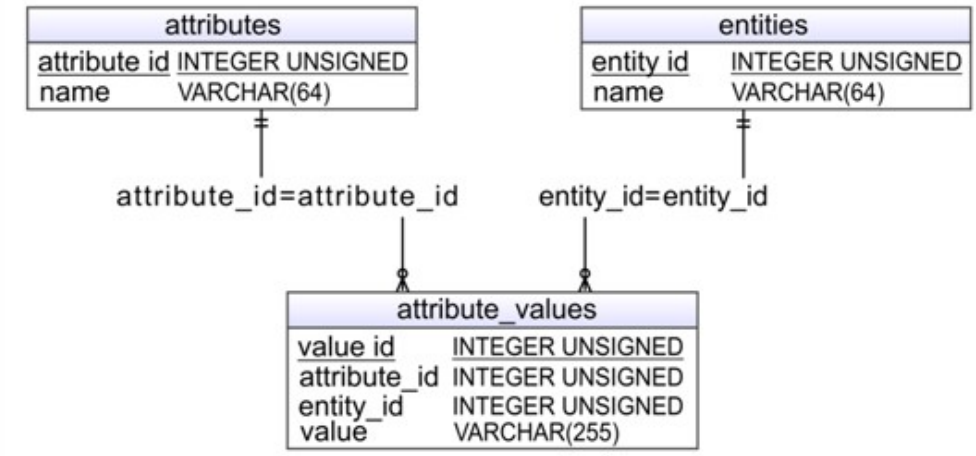

Рис. 1. Класична модель EAV

Аналіз останніх досліджень і публікацій

У наукових публікаціях, присвячених створенню реляційних БД для зберігання розріджених наборів даних пропонуються різні моделі. Окрім адаптованої вертикальної моделі даних «Сутність-атрибут-значення» [2], модель збереження даних DSM (Decomposition Storage model), яка передбачає декластеризацію кожних відносин на стовпці з використанням суррогатних ключів [3, 4], а також підходи, що засновані на використанні wide-таблиць [5] та інтерпретованого формату зберігання атрибутів [6].

Модель EAV є найбільш поширеною моделлю зберігання даних в клінічних системах. Першу появу моделі EAV прийнято пов’язувати з вирішенням проблеми організації та зберігання дуже різноманітних і різнорідних даних про пацієнтів в біомедичних БД [2, 7, 8]. Незважаючи на те, що використання підходу EAV надає гнучкість і можливість зберігання даних в простому, легко підтримуваному форматі, але при цьому відмічається зниження продуктивності в 3-5 разів у порівнянні з традиційною реляційною моделлю зберігання даних [7].

Розвитку моделі EAV дуже сприяють існуючі дослідження в галузі охорони здоров'я і біомедицини [7-9], а також в інших областях, таких, наприклад, як електронна комерція [10] і семантична павутина [11]. Поточні дослідження моделі EAV спрямовані на більш швидке вилучення даних і спеціальну підтримку складних запитів.

Технологічні рішення, які використовують модель EAV – це, перш за все, рішення класу Enterprise Recourse Planning (ERP), Customer Recourse Management (CRM), Enterprise Content Management (ECM) та Content Management System (CMS).

В системах управління контентом (CMS) є конструктори контенту CCK (Content Construction Kit), які призначені для створення нових типів контенту. Вони доцільні у випадку, коли програмісту необхідно змінити фіксовані у CMS форми контенту. Наприклад, змінити набор полів та опцій статті (заголовок, опис, дата публікації, автор, стан та ін.), додавши нову форму коментарів до статей або кілька форм зворотного зв'язку з різними полями для заповнення.

Результати порівняльного аналізу показників ефективності сучасних систем конструювання контенту для Joomla 3 представлені в табл. 1. Передбачається, що система, яка буде реалізована в роботі, перевершить наведені показники.

Постановка завдання

Метою роботи є розробка універсальної програмної системи створення та зберігання даних з гнучкою динамічно змінювальною структурою, призначеної для ро-

боти з фреймворком Yii2. Конфігурація сутностей, що можуть зберігатися системою, може відрізнятися від схеми до схеми, від проекту до проекту, вони можуть легко конфігуруватися за допомогою WYSIWYG редактору.

Табл. 1. Порівняльний аналіз характеристик виконання запитів у ССК

Система конструювання контенту	Cobalt	Seblod	K2	Zoo
Кількість запитів	30	23	26	23
Використана пам'ять, Mb	9,28	12,04	7,6	8,11
Час виконання, s	0,086	0,108	0,056	0,073

Виклад основного матеріалу

Для реалізації ПЗ використана СКБД MySQL та фреймворки AngularJS і Yii2. Середовище розробки JetBrains PHP Storm було вибране за умовою підтримки обох фреймворків, воно дозволяє працювати також з Grunt та Node.js, що використовуються у цьому проекті для автоматизації. Для взаємодії фронт-енду та бек-енду використаний RESTful підхід [12]. Основою є структуризація запитів та відповідей за допомогою JSON формату. Система являє собою аналог ПП для реалізації у OSS та BSS сфері, тому є можливість інтеграції її у інші системи за допомогою RESTfull API.

Програмне забезпечення поділене на 2 основні частини: каталог та конструктор. Каталог дозволяє переглядати об'єкти що створені користувачами, це є фактичні дані. Конструктор надає можливість переглядати поля і характеристики тих чи інших об'єктів, тобто редагувати структуру даних. Між конструктором та каталогом можна провести аналогію як між класом та екземпляром класу в ООП.

Для створення структур даних був розроблений WYSIWYG редактор, який дозволяє у два кліки мишею створювати сутності та поля для них. Для того, щоб сутності можна було використовувати повторно були введені схеми, поля можна додавати як до самої сутності, так і до комбінації сутності-схеми. Це дозволяє у подальшому розробляти різноманітні інтерфейси для кожного типу користувача на одній системі керуванням контентом.

Для того, щоб можна було інтегрувати систему до інших систем були розроблені класи, які дозволяють використовувати сутності у інших модулях фреймворку, наприклад, інших програмістів, які створені у WYSIWYG редакторі з використанням драйверу підключення до БД ActiveRecord.

В якості прикладу розглянемо етапи розробки концептуальної бази даних інформаційно-пошукової системи для пошуку вакансій для людей з обмеженими можливостями. Цей процес включає в себе аналіз об'єктів реального світу, які необхідно змоделювати в базі даних і складається з етапів:

– ідентифікація функціональної діяльності предметної області. В даному випадку мова йде про діяльність організації. В якості функціональної діяльності можна розглядати ведення реєстру вакансій в компаніях, які мають робочі місця для людей з обмеженими можливостями;

– ідентифікація об'єктів, які здійснюють цю функціональну діяльність. Для цього треба ідентифікувати всі сутності і взаємозв'язки між ними. Процес «ведення реєстру вакансій» ідентифікує такі сутності: OBJECT, ATTRIBUTE, ATTRIBUTE_TYPE, OBJECT_TYPE;

– ідентифікація характеристик цих сутностей;

– ідентифікація взаємозв'язків між сутностями.

Перерахуємо сутності:

– сутність OBJECT_TYPE може включати такі характеристики як: ідентифікатор, ідентифікатор батька, закодоване ім'я, ім'я, та опис об'єктного типу;

– сутність ATTRIBUTE_TYPE може включати такі характеристики як: ідентифікатор, ідентифікатор об'єктного типу, закодоване ім'я, ім'я, тип поля, ім'я вкладки, особлива конфігурація поля;

– сутність OBJECT може включати такі характеристики як: ідентифікатор, ідентифікатор батька, ідентифікатор об'єктного типу, закодоване ім'я, ім'я, та опис об'єкту;

– сутність ATTRIBUTE може включати такі характеристики як: ідентифікатор, ідентифікатор об'єкту, ідентифікатор атрибутного типу, значення та значення дати.

Наступний етап проектування БД полягає у встановленні відповідності між сутностями і характеристиками предметної області та відносинами і атрибутами в нотації обраної СКБД. Оскільки кожна сутність реального світу володіє якимись характеристиками, які в сукупності утворюють повну картину її прояви, можна поставити їм у відповідність набір відносин (таблиць) і їх атрибутів (полів). Кожна таблиця в реляційній базі даних задовольняє умові, відповідно до якої в позиції на перетині кожного рядка і стовпчика таблиці завжди знаходиться єдине значення, і ніколи не може бути безлічі таких значень.

Вигляд проектованої бази даних після нормалізації демонструють табл. 2-5.

Табл. 2. Структура таблиці «OBJECT_TYPE»

Ім'я поля	Тип
ID	INT(11)
PARENT_ID	INT(11)
CODE	VARCHAR(255)
NAME	VARCHAR(255)
DESCRIPTION	TEXT

Табл. 3. Структура таблиці «ATTRIBUTE_TYPE»

Ім'я поля	Тип
ID	INT(11)
OBJECT_TYPE_ID	INT(11)
CODE	VARCHAR(32)
NAME	VARCHAR(255)
FIELD_TYPE	VARCHAR(255)
FIELD_TAB	VARCHAR(255)
CONFIG	TEXT

Табл. 4. Структура таблиці «OBJECT»

Ім'я поля	Тип
ID	INT(11)
PARENT_ID	TEXT
OBJECT_TYPE_ID	DECIMAL(10,2)
NAME	TINYINT(1)
DESCRIPTION	TEXT

Табл. 5. Структура таблиці «ATTRIBUTE»

Ім'я поля	Тип
ID	INT(11)
OBJECT_ID	INT(11)
ATTRIBUTE_TYPE_ID	INT(11)
VALUE	TEXT
DATE_VALUE	TIMESTAMP

Таблиця «OBJECT_TYPE» містить інформацію про типи об'єктів. Таблиця «ATTRIBUTE_TYPE» містить інформацію про типи полів, які відносяться до об'єктів. Таблиця «OBJECT» містить всі екземпляри об'єктів. Таблиця «ATTRIBUTE» містить в собі інформацію про всі екземпляри полів що відносяться до екземплярів об'єктів.

Для зберігання сутностей була створена структура у базі даних MySQL, наведена на рис. 2. Такий формат зберігання дозволяє міняти структуру сутності в залежності від користувача чи проекту.

Розглянемо алгоритм роботи з EAV структурами за допомогою ActiveRecord. На початку система опрацьовує POST запит до /eav-service/get-object, де параметрами objectIds та withAttributes є ідентифікатори об'єктів та потрібність повного вивантаження об'єктів відповідно. Після цього виконується вилучення об'єктів з БД та наповнення структур сутностей атрибутами. Якщо атрибуту не має, але є атрибутний тип, то значення виставляється у NULL, якщо об'єктів не знайдено, то відбувається викид виключної ситуації та виводиться повідомлення про помилку. Розгорнутий алгоритм роботи RESTfull API для отримання EAV сутностей представлений на рис. 3.

За допомогою WYSIWYG-EAV редактору структур даних (рис. 4) користувач створює свої структури даних, які потім буде наповнювати. Тут реалізовані наступні дії: виділення, редагування, клонування, переміщення, додавання та видалення сторінок, структур, вкладок, автоматизованих завдань та схем у системі.

За допомогою WYSIWYG-EAV редактору сутностей користувач наповнює свою систему тими чи іншими сутностями, яким відповідає своя EAV-структура. Як і в редакторі структур тут реалізовані наступні дії: виділення, редагування, клонування, переміщення, додавання та видалення сутностей з системи. Якщо потрібно, то розробник може спадкуватися від класу EavPage та створювати свої обробники та інтерфейси у цій системі (рис. 5).

На зображеннях конструктора (рис. 4) та каталогу (рис. 5) наведена реалізована схема IoT для системи розумного будинку. Створені базові структури об'єктів «Device» (пристрій), та «Location» (локація), а також дочірні структури пристроїв, такі як «ESP8266», «ESP32», «Raspberry Pi», «Router», «Switch». На пристроях налаштований MQTT клієнт.

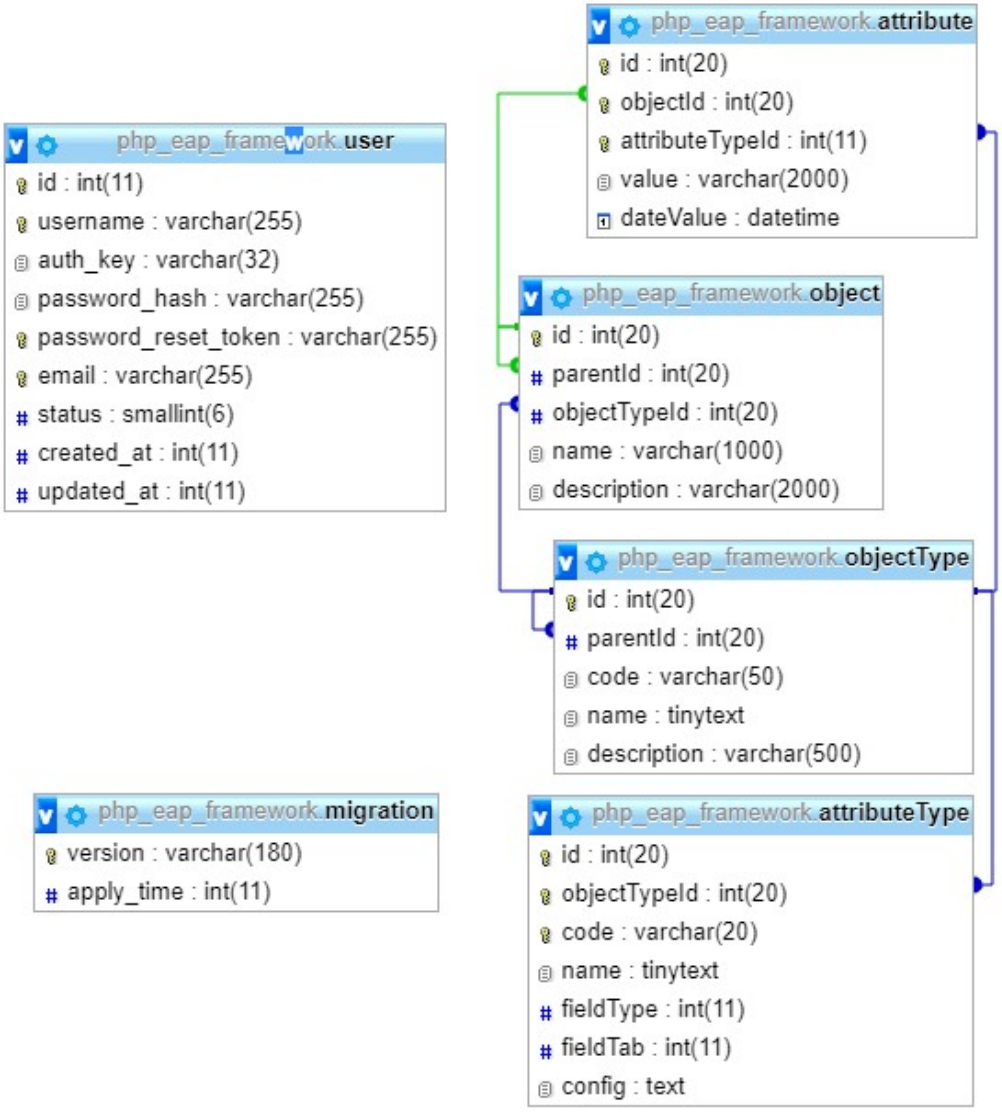

Рис. 2. Структура зв'язків між таблицями, що використовуються для зберігання EAV сутностей

ISSN 2520-2820 (online)
ISSN 2524-0978 (print)

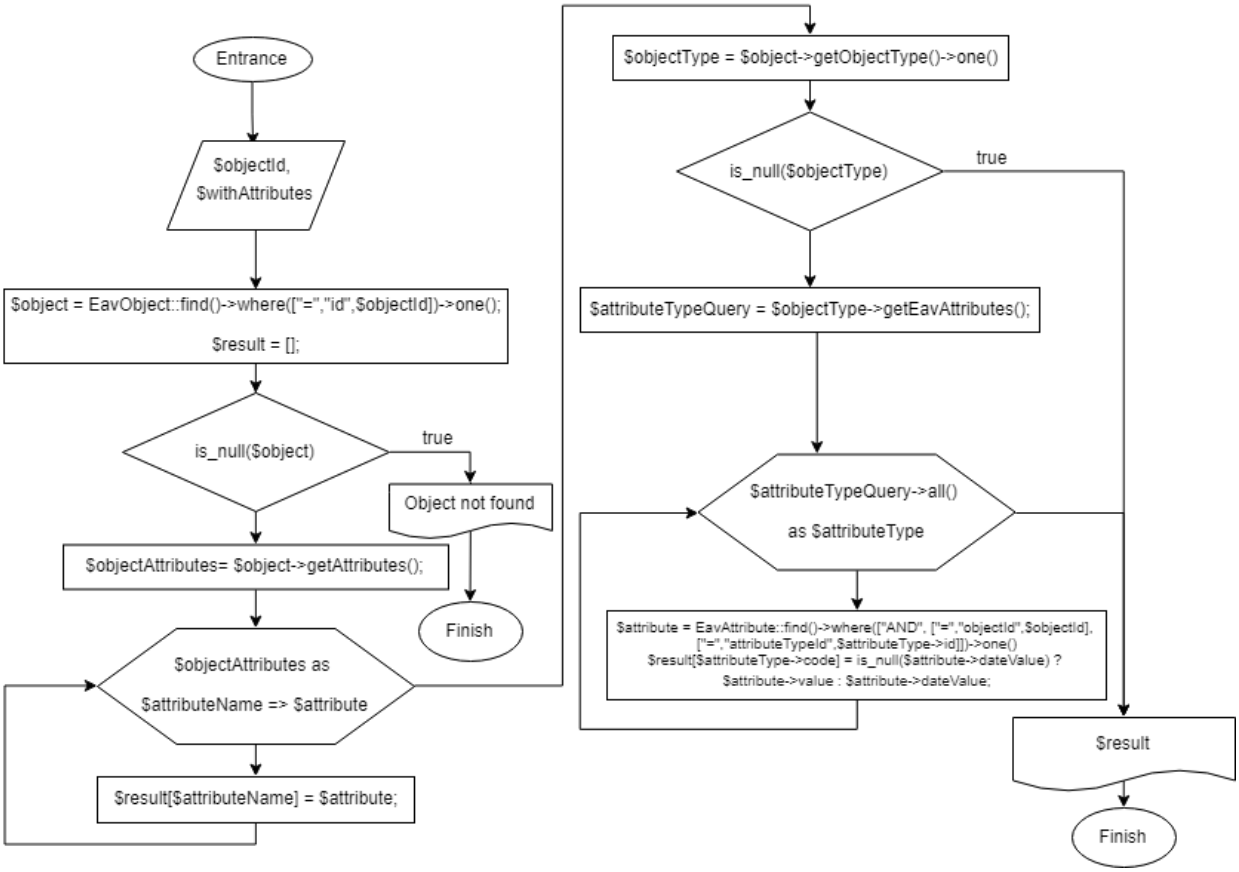

Рис. 3. Розгорнутий алгоритм роботи RESTfull API
для отримання EAV сутностей

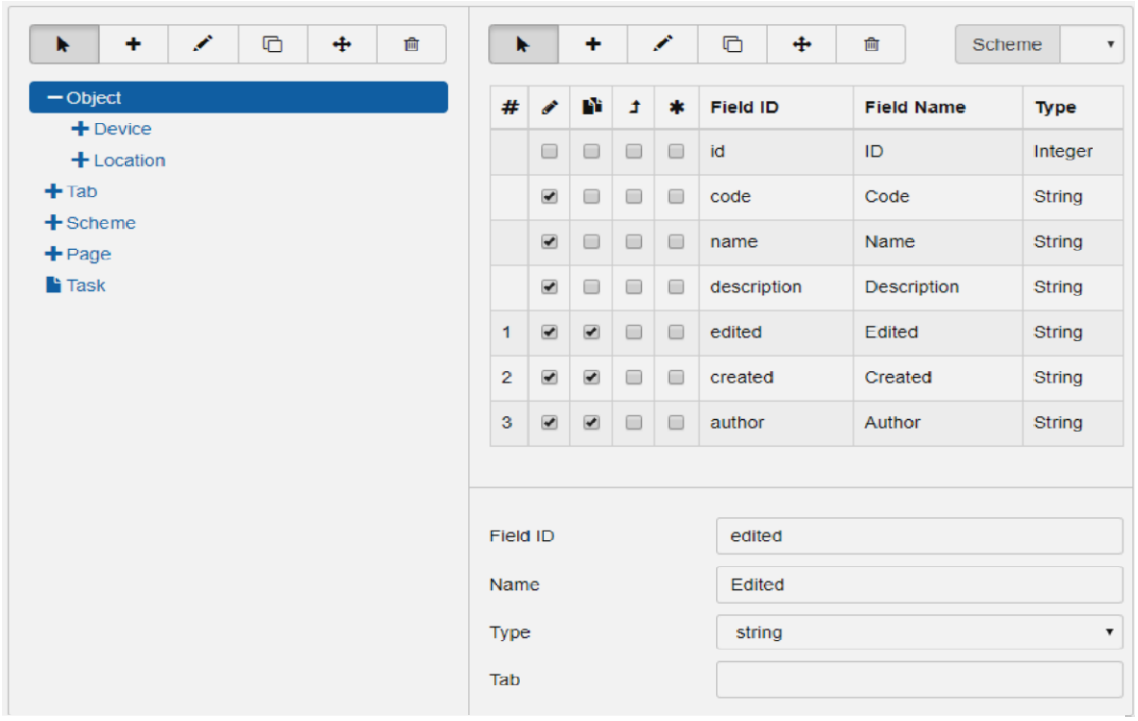

Рис. 4. Конструктор – редактор EAV структур

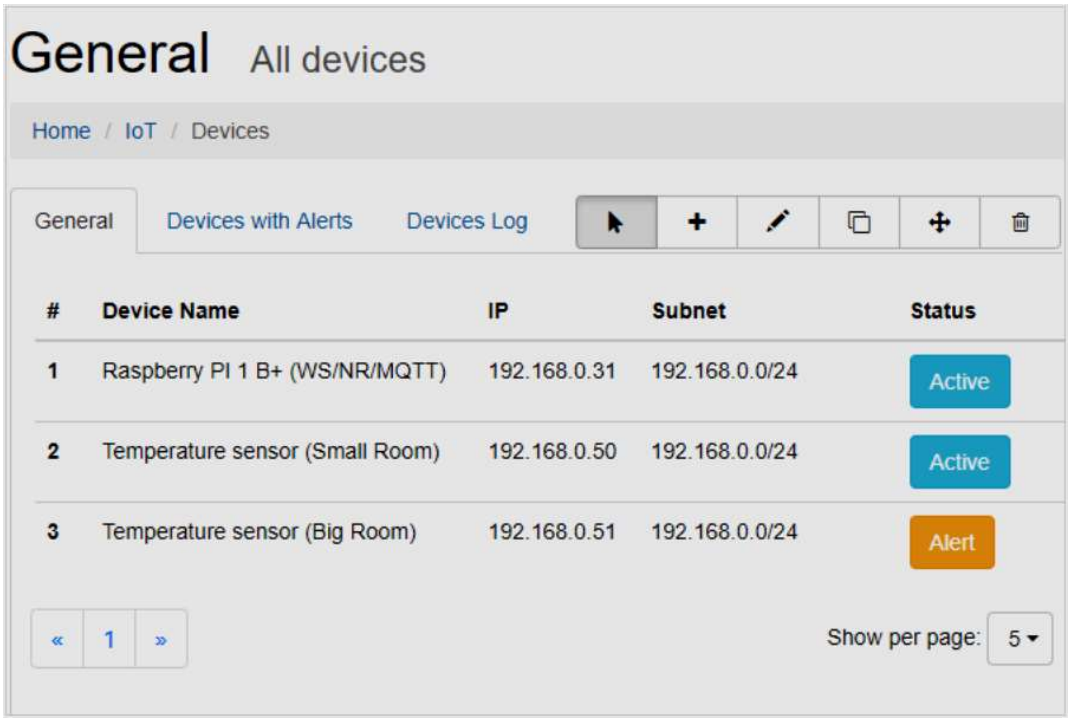

Рис. 5. Каталог – модуль відображення EAV сутностей

У каталозі ПЗ створений екземпляр об'єкту Task (задача), у якому прописано тип запуску, час запуску та шлях до скрипта, що буде виконуватися у вказаний час. Цей скрипт кожні 5 хвилин посилає запит в топік «/dev/ping», який змушує всі пристрої написати в цей же топік свої MAC адреси. Скрипт обновлює дані про дату останньої активності, що записані у кожному з пристроїв, по їх MAC адресі. Таким чином можемо завчасно визначити який з пристроїв вийшов з ладу. Система також має REST API, через який можна ці дії виконувати використовуючи веб-запити. Наприклад, для отримання структури проекту достатньо зробити GET запит за адресою

```
https://example.com/eav-editor/getstructure?
id=1234& scheme=5678,
```

де 1234 – це ID батьківської структури, а 5678 – це ID схеми, яку потрібно вивантажити.

Висновки і перспективи досліджень

1. Розроблена універсальна система створення та зберігання EAV структур даних для роботи з фреймворком Yii2.

2. Представлена система дозволяє швидко розробляти прототип інформаційної системи із змінною структурою метаданих, при цьому зберегати можливості і обмеження СКБД SQL та отримувати додатково можливості СКБД NoSQL.

3. Система використовує для отримання сутності з БД 7 запитів і 1.5Mb пам'яті та затрачує лише 76 мс, що перевершує подібні характеристики більшості існуючих аналогів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Entity-attribute-value model (EAV). Вікіпедія: вільна енциклопедія. URL: https://en.wikipedia.org/wiki/Entity-attribute-value_model (дата звернення: 15.02.2019).
2. Nadkarni P. The EAV/CR Data Model. URL: http://ycmi.med.yale.edu/nadkarni/eav_CR_contents.htm (дата звернення: 15.02.2019).
3. Copeland G., Copeland G.P., Khoshafian S.N. A decomposition storage model. *ACM SIGMOD*. Rec. 1985, 14, P. 268-279.

4. Khoshafian S., Copeland G., Jagodis T., Boral H., Valduriez P. A query processing strategy for the decomposed storage model. *Proceedings of the Third International Conference on Data Engineering*. 1987. P. 636-643.
5. Chu E., Beckmann J., Naughton J. The case for a wide-table approach to manage sparse relational data sets. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. Beijing, China, 11–14 June 2007. P. 821-832.
6. Beckmann J.L., Halverson A., Krishnamurthy R., Naughton J.F. Extending RDBMSs to support sparse datasets using an interpreted attribute storage format. In Proceedings of the 22nd International Conference on Data Engineering (ICDE'06). Atlanta, GA, USA, 3–7 April 2006. P. 58.
7. Chen RS, Nadkarni P, Marenco L, Levin F, Erdos J, Miller PL. Exploring performance issues for a clinical database organized using an entity-attribute-value representation. *J Am Med Inform Assoc*. 2000. №7(5). P. 475-487.
8. Luo G., Frey L.J. Efficient execution methods of pivoting for bulk extraction of Entity-Attribute-Value-modeled data. *IEEE J. Biomed. Health Inform*. 2016. Vol. 20. P. 644-654.
9. Duftschmid G., Wrba T., Rinner C. Extraction of standardized archetyped data from Electronic Health Record Systems based on the Entity-Attribute-Value Model. *Int. J. Med. Inform.* 2010. №79. P. 585-597.
10. Agrawal R., Somani A., Xu Y. Storage and Querying of E-Commerce Data. *VLDB*. Roma, Italy, 2001. Vol. 1. P. 149-158.
11. Abadi D.J., Marcus A., Madden S.R., Hollenbach K. SW-Store: A vertically partitioned DBMS for Semantic Web data management. *VLDB J*. 2009. №18. P. 385-406.
12. REST. Вікіпедія: вільна енциклопедія. URL: https://uk.wikipedia.org/ wiki/REST (дата звернення: 15.02.2019).

Mykhailo SHCHERBYNA, Andrii OKHRYMENKO, Svitlana KUZNICHENKO
Odesa

A SOFTWARE SYSTEM FOR CREATION AND STORAGE EAV DATA STRUCTURES

The software system for creating and storing EAV data structures for the YII2 framework is developed. The purpose of the system, its main functionality, as well as means of realization are given.

Keywords: *Entity-attribute-value model, database, framework YII2, NOSQL, SQL, visual editor WYSIWYG.*

Михаил ЩЕРБИНА, Андрей ОХРИМЕНКО, Светлана КУЗНИЧЕНКО
Одесса

ПРОГРАММНАЯ СИСТЕМА СОЗДАНИЯ И ХРАНЕНИЯ EAV СТРУКТУР ДАННЫХ

Разработана программная система создания и хранения EAV структур данных для фреймворка YII2. Приведены назначение системы, ее основные функциональные возможности, а также средства реализации.

Ключевые слова: *Entity-atribute-value модель, база данных, фреймворк YII2, NOSQL, SQL, визуальный редактор WYSIWYG.*

Стаття надійшла до редколегії 27.02.2019