УДК 004.652

С. В. Осієвський, О. Ю. Несміян

СИНТЕЗ СИСТЕМИ КЕРУВАННЯ АЛОФОННИМИ БАЗАМИ ДАНИХ ДЛЯ ОРГАНІЗАЦІЇ ВЗАЄМОДІЇ ОПЕРАТОРА АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ ПОВІТРЯНИМ РУХОМ (АС КПР) З КОМПЛЕКСОМ ЗАСОБІВ АВТОМАТИЗАЦІЇ (КЗА)

Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків

Анотація. У статті розглянуті особливості реалізації сховищ даних і їх обмеження, а також описана послідовність представлення даних в абстрактній предметній області, які дозволяють запропонувати деякий альтернативний підхід до організації сховищ даних, орієнтований на використання контейнерів. Даний підхід поєднує в собі переваги реляційних баз даних з їх можливістю забезпечення консистентності даних в будь-який момент часу, а також особливості безструктурного зберігання даних, характерних в цілому для NoSQL систем. Також запропоновані шляхи вирішення задачі синтезу системи керування алофонними базами даних для організації взаємодії оператора АС КПР з КЗА. В основу організації покладено ряд правил, що дозволяють побудувати схему неструктурованих даних, організувати взаємодію бази даних NoSQL з існуючими реляційними рішеннями в основу якої покладено положення теорії категорій. Розроблені правила забезпечують можливість здійснення фізичного та логічного моделювання баз даних NoSQL. Також розроблено стандарти кардинальності для моделювання баз даних NoSQL та введено класифікацію з визначенням пріоритетів класів і настанов щодо моделювання NoSQL. Перевірку запропонованих рішень було здійснено шляхом моделювання та порівняння отриманих результатів з теоретичними положеннями. Розроблено SO-model та обгрунтувано її властивості. Особливістю SO-model являється те, що кожен об'єкт має свій глобальний унікальний ідентифікатор U_{unqID}, що дозволяє зберігати об'єкти одного типу на різних комп'ютерах за рахунок чого підвищити масштабованість сховища даних. Отримані результати свідчать про значне вдосконалення схеми даних неструктурованих та слабоструктурованих баз даних. Запропоновану модель сховища даних за рахунок її властивості уніфікації структури, яка враховує динаміку даних і їх зв'язків, можна розвивати в різних напрямках, використовуючи NoSQL рішення.

Ключові слова: бази даних NoSQL, моделювання баз даних, схеми великих даних, бази даних зберігання документів.

Abstract. The article describes data warehouses implementation peculiarities and their limitations, as well as describing the sequence of data presentation in an abstract subject area, which allows us to propose some alternative approach to the organization of data warehouses, focused on containers usage. This approach combines the advantages of relational databases with their ability to ensure data consistency at any point in time, as well as features of unstructured data storage, typical of NoSQL systems in general. Also, the article proposes ways to solve the problem of synthesis of allophone database management system for organization of interaction between AC ACS operator and ACM. The method is based on a number of rules that allow to build a scheme of unstructured data, to organize the interaction of the NoSQL database with existing relational solutions, which is based on the provisions of the category theory. The developed rules provide the possibility of physical and logical modeling of NoSQL databases. Cardinality standards for NoSQL database modeling have also been developed and a classification has been introduced to prioritize classes and guidelines for NoSQL modeling. The verification of the proposed solutions was carried out by modeling and comparing the obtained results with theoretical positions. The SO-model was developed and its properties were substantiated. A feature of the SO-model is that each object has its own global unique identifier U_{unqID}, which allows you to store objects of the same type on different computers, thereby increasing the scalability of the data store. Obtained results indicate a significant improvement of the data scheme of unstructured and poorly structured databases. The proposed model of data storage, due to its property of unifying the structure, which takes into account the dynamics of data and their connections, can be developed in various directions using NoSQL solutions.

Keywords: NoSQL databases, database modeling, big data schemes, document storage databases.
DOI: https://doi.org/10.31649/1999-9941-2022-54-2-39-46.

Вступ

В даний час, в науковому середовищі, виникає досить велика кількість диспутів щодо питання зберігання та обробки різнорідних (неструктурованих) даних.

При цьому, якщо вести мову про структуровані дані, то традиційні бази даних, в даний час, забезпечують реалізацію всіх технічних рішень в КЗА, це підтверджується роботами [7-11]. Зокрема вони забезпечують централізований контроль даних, усунення невідповідностей та надмірність контролю [1]. Поява великих та неструктурованих даних з їх нетрадиційними характеристиками та слабоструктурованих даних, що вимагають спільного зберігання та обробки зумовили необхідність створення гнучких, масштабованих та прихованих баз даних – NoSQL [2-4].

Термін NoSQL не введено як повноцінну заміну SQL та традиційних баз даних, він призначений заповнити прогалини, що утворилися внаслідок постійного розширення розміру, складності, різноманітності та мінливості даних.

Актуальність теми

З метою отримання адекватних рішень, щодо забезпечення вимог до обробки та використання даних, проведено аналіз існуючих рішень та провідних наукових досліджень, зокрема: в роботі [5] запропоновано використання нових позначень та стилів кардинальності, в роботі [6] розроблені та запропоновані до використання пропозиції щодо моделювання баз даних та особливостей зберігання документів

NoSQL. Результати цих досліджень дають можливість провести поточне дослідження в аспекті дослідження семантичного відображення сутностей та пріоритетності відношень.

Рішення, які пропонуються, відображені в моделі, що віддзеркалює структури схеми для баз даних зберігання документів NoSQL. Ця модель спрямована на подальше спрощення процесу моделювання NoSQL.

Першим рішенням вищезгаданої задачі було встановлення стандартів кардинальності для моделювання баз даних NoSQL. Другим рішенням являється класифікація та визначення пріоритетів класів та настанов щодо моделювання NoSQL. Ці рішення вимагають значно високого рівня знань, тим самим мотивуючи використання положень, що запропоновані для вирішення задачі синтезу системи керування алофонними базами даних для організації взаємодії оператора АС КПР з КЗА.

Мета

Грунтуючись на завданях, що потребують вирішення визначено за мету розробку науково – теоретичного апарату, що дозволить досягти підтримання темпоральності на рівні зв'язків між сутностями ER-моделі за рахунок відсутності обмеження на підтримання такої цілісності даних в базовому функціоналі SO-model. Досягнння зазначеної мети дозволить знайти рішення задачі синтезу системи керування алофонними базами даних для організації взаємодії оператора АС КПР з КЗА.

Задачі

Виходячи з вищезазначених існуючих рішень та теоретичних положень теорії побудови БД, рішення основної задачі вбачається в послідовній реалізації наступних етапів:

– розробка правил локалізації характеристик об'єктів алофонної БД;

– розробка конструкторів об'єктів алофонної БД;

– розробка SO-моделі реалізації керуючих функції алофонних БД для організації керування алофон ними БД.

Розв'язання задач

В сучасному світі швидкість доступу до інформації стає вкрай важливим фактором. Задля підвищення швидкості доступу до інформації було розроблено низку нових підходів до зберігання даних, у тому числі NoSQL БД, які зарекомендували себе як надійні, швидкі БД які легко масштабуються [12]. Проте, такі рішення мають один суттєвий недолік – відсутність контролю консистентності даних.

Для отримання кращого результату по швидкості і консистетності пропонується використання ER-моделей у NoSQL рішеннях. Доведемо, що будь-яка ER-модель може бути реалізована в SO-модель.

Можливі два варіанти первинного ключа однієї сутності. Вони представлені на рисунку 1. Сформуємо правила роботи з первинними ключами таких сутностей

Правило 1. Первинний ключ сутності (Entity_1) є простим (рисунок 1(а)). Така сутність відповідає множині об'єктів в SO-model. Кожен екземпляр сутності відображається в відповідний об'єкт SO-model. Всі об'єкти цього множини мають однаковий тип – ім'я сутності. Атрибут (Attribute_1) − первинний ключ задає ім'я об'єкта, всі інші атрибути задають властивості об'єкта.

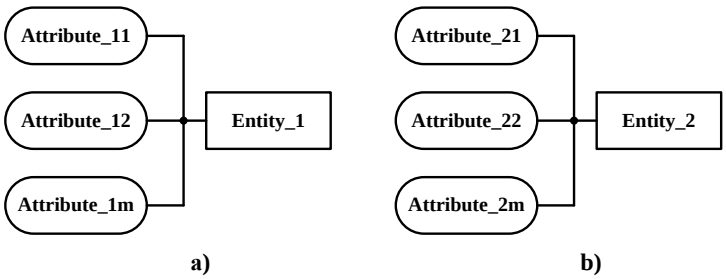

Рисунок 1 – Одна сутність

Позначимо:

$Entity_j.Name$ – ім'я j-й сутності (Entity_j, j = 1,2), відповідає типу об'єкта;

$Attribute_{jk}.Name$ – ім'я *k*-го атрибута *j*-ї сутності;

$Attribute_{jk}.Value$ – значення *k*-го атрибута *j*-ї сутності;

$time_j$ – момент часу початку існування сутності і її властивостей;

m – кількість атрибутів сутності, а також кількість властивостей об'єктів відповідного типу;

n – кількість примірників сутності, а також кількість об'єктів відповідного типу;

$U_{unqID_{obji}}$ – унікальний ідентифікатор *i*-го об'єкта;

$U_{unqID_p ik}$ – унікальний ідентифікатор *k*-ї властивості *i*-го об'єкта.

Тоді множина об'єктів SO-model будується наступним чином:

$$\left\{< U_{unqID_{obji}}, Entity_{\,j}.Name, P_i, < Attribute_{\,jk}.Value, time_j >>_{i=\overline{1,n}} \right\}$$

$$\forall i : P_i = \left\{ \left\langle U_{unqID_p 1k}, Attribute_{1k}.Name, \left\{ \left\langle Attribute_{1k}.Value, null, time_1 \right\rangle \right\} \right\rangle_{k=2..m} \right\}$$

Правило 2. Є одна сутність (Entity_2), первинний ключ якої є складовим − наприклад, складається з атрибутів Attribut_21, Attribute_22 (рисунок 1 (b)). В цьому випадку відображення будується аналогічно попередньому випадку, за винятком імені об'єкта, яке можна побудувати з'єднанням частин складеного ключа, крім цього, кожна частина складеного первинного ключа додається як окрема властивість об'єкта.

$$\left\{ \left\langle U_{unqID_{obji}}, Entity_2.Name, P_i \left\langle Attribute_{21}.Value : Attribute_{22}.Value, time_1 \right\rangle \right\rangle_{i=1,..,n} \right\}$$

$$\forall i : P_i = \left\{ \left\langle U_{unqID_p ik}, Attribute_{2k}.Name, \left\{ \left\langle Attribute_{21k}.Value, null, time_1 \right\rangle \right\} \right\rangle_{k=1..m} \right\}$$

Зауважимо, що якщо розглядати статичну SO-модель (значення $time_i$ не змінюється у імені об'єктів і їх властивостей), то побудоване відображення примірників сутностей ER-моделі в об'єкти SO-моделі є бієктивне, тобто можливе взаємно однозначне відображення об'єктів SO-моделі по сутності і екземпляри сутностей ER-моделі.

Правило 3. Рекурсивне відношення ER-моделі даних може бути реалізовано в SO-model.

Можливі види рекурсивних відносин в ER-моделі показані на рисунку 2.

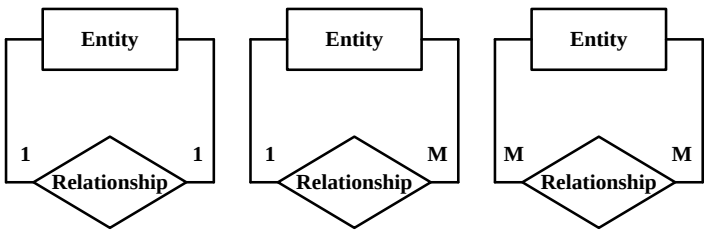

Рисунок 2 – Рекурсивні зв'язки

В базовому функціоналі SO-model відсутній механізм фіксації виду відносин – один до одного, один до багатьох, багато до багатьох. Дане розмежування виду відносин може бути перенесено на програмні додатки по взаємодії зі сховищем. Тому для доказу досить розглянути тільки один вид відносин: багато до багатьох, вважаючи інші види окремими випадками цих відносин з накладеними певними обмеженнями.

Для реалізації рекурсивних відносин багато до багатьох в SO-model в об'єкти, які відповідають екземплярам сутності, крім атрибутів сутності, додається властивість з типом об'єкта. Причому значення цієї властивості, що має посилання, вказує на U_{unqID} об'єкта, який відповідає структурі відносин:

$$\left\{ \left\langle U_{unqID_{obj_i}}, Entity.Name, P_i (Attribute_1.Value, time_1) \right\rangle_{i=1..n} \right\}$$

$$\forall i : P_i = \left\{ \left\langle U_{unqID_p ik}, Attribute_k.Name, \left\{ \left\langle Attribute_k.Value, null, time_1 \right\rangle \right\} \right\rangle_{k=1..m} \right\}$$

$$\left\{\left\langle U_{unqID_p i(m+1)}, Entity.Name, \begin{Bmatrix} null, \\ U_{unqIDobj}, \\ tyme_1 \end{Bmatrix} \mid T_{obj}(\overline{U}_{obj}(U_{unqIDobj})) = Entity.Name \right\rangle\right\}$$

Правило 4. Зв'язки сутностей ER-моделі даних можуть відображатися в зв'язку об'єктів в SO-model.

На рисунках 3 – 5 показані можливі види зв'язків між сутностями в ER-моделі даних та модальність зв'язків: подвійна лінія зв'язку – «повинен», одинарна лінія зв'язку – «можливо». Введення обмежень на створення нових об'єктів певного типу, додавання і зміни властивостей об'єктів дозволяють реалізувати всі види зв'язків ER-моделі даних (причому реалізація цих обмежень може бути винесена за рамки базового функціоналу SO-model).

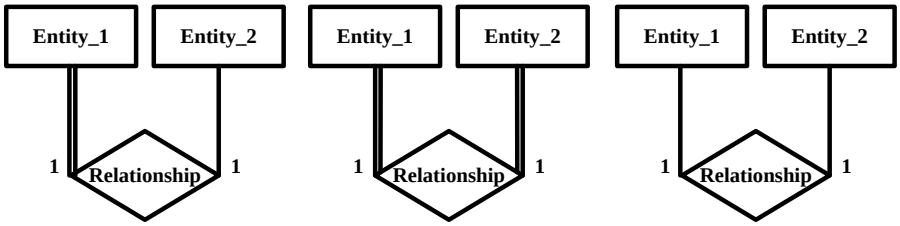

Рисунок 3 – Види зв'язків сутності один до одного

1. Нехай екземпляри сутності «Entity_1» при простому первинному ключі «Attribute_11» відображаються в об'єкти SO-model:

$$\left\{\left\langle U_{unqID_{obj}}, Entity_1.Name, P_i\left\langle Attribute_{11}.Value, time_1\right\rangle\right\rangle_{i=1..n}\right\}$$

$$P_i = \left\{\left\langle U_{unqID_{pk}}, Attribute_{1k}.Name\left\{\left\langle Attribute_{1k}.Value, null, time_1\right\rangle\right\}\right\rangle_{k=1...m}\right\}$$

$F(Object_{ji})$ – зовнішнє правило, що задає наявність зв'язку один до одного для об'єкта, відповідного *i*-му примірнику *j*-ї сутності. В цьому випадку об'єкти, відповідні екземплярам «Entity_2», можна виділити із значень первинного ключа примірників «Entity_1» наступним чином:

$$A \Leftarrow Object.Type(Entity_1.Name)$$

$$B \Leftarrow red(Ind[x], if\ (F(x), x, NULL), X))$$

$$C \Leftarrow Ind[x], Get.Name(x, Attribute_{11}.Name), B)$$

$$D \Leftarrow Ind([x], Object.Create(GenU_{unqID}, Attribute_{11}.Name,$$
$$Property.Value.Values.Time(x, time)time), C)$$

$$Ind([x, x*].\mathrm{mod}(x \leftarrow x^*), B, ind([x_1, x_2, x_3],$$
$$Property.Link.Change(x_1, x_2, x_3, time), B, C, ind([z], z.U_{unqID}, D)))$$

2. Нехай екземпляри сутності «Entity_1» при складеному первинному ключі «Attribute_11», «Attribute_12» відображаються в об'єкти SO-model:

$$\left\{\left\langle U_{unqID_{obji}}, Entity_1 Name, P_i\left\langle Attribute_{11}.Value : Attribute_{12}.Value, time_1\right\rangle\right\rangle_{i=1..n}\right\}$$

$$P_i = \left\{ \left\langle U_{unqID_{pk}}, Attribute_{1k}.Name \left\{ \left\langle Attribute_{1k}.Value, null, time_1 \right\rangle \right\}_{k=1...m} \right\rangle \right\}$$

У цьому випадку порядок дій може бути аналогічний попередньому пункту, відмінність полягає в тому, що об'єкти виділяються з кожного атрибута складеного первинного ключа «Entity_1»:

$$A \Leftarrow Object.Type(Entity_1.Name)$$

$$B \Leftarrow red(Ind[x], if\ (F(x), x, NULL), X))$$

$$C \Leftarrow Ind[x], Get.Name(x, Attribute_{11}.Name), B)$$

$$D \Leftarrow Ind[x], Get.Name(x, Attribute_{12}.Name), B)$$

$$E \Leftarrow Ind([x], Object.Create(GenU_{unqID}, Attribute_{11}.Name,$$
$$Property.Value.Values.Time(x, time)time), C)$$

$$F \Leftarrow Ind([x], Object.Create(GenU_{unqID}, Attribute_{12}.Name,$$
$$Property.Value.Values.Time(x, time)time), D)$$

$$Ind([x, x*], \mathrm{mod}(x \leftarrow x^*), B, ind([x_1, x_2, x_3],$$
$$Property.Link.Change(x_1, x_2, x_3, time), B, C, ind([z], z.U_{unqID}, E)))$$

$$Ind([x, x*], \mathrm{mod}(x \leftarrow x^*), B, ind([x_1, x_2, x_3],$$
$$Property.Link.Change(x_1, x_2, x_3, time), B, C, ind([z], z.U_{unqID}, F)))$$

Модальність зв'язків: «Entity_1» – «повинен»; «Entity_2» – «можливо». Надалі дозволяється додавати нові об'єкти в множину об'єктів типу «Entity_2», а також забороняється додавати об'єкти «Entity_1» без виділення нового об'єкта зі значення первинного ключа і додавання його в множину «Entity_2».

Модальність зв'язків: «Entity_1» – «повинен»; «Entity_2» – «повинен».

Порядок дій збігається з попереднім випадком. Однак після виділення об'єктів забороняється додавати нові об'єкти в множину об'єктів типу «Entity_2», а також забороняється додавати об'єкти «Entity_1» без виділення нового об'єкта зі значення первинного ключа і додавання його в множину «Entity_2».

Реалізація зв'язків один до багатьох.

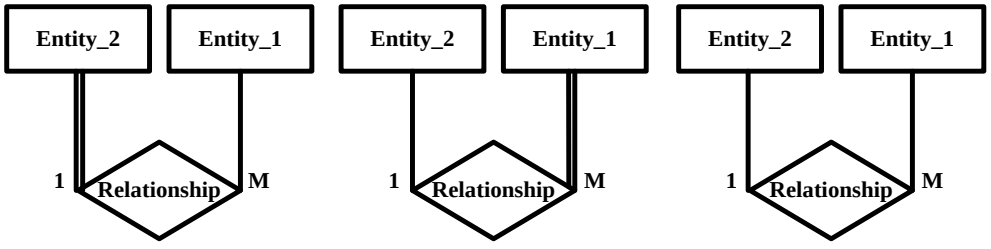

Рисунок 4 – Види зв'язків сутності один до багатьох

Для доказу аналогічно попередньому випадку приймемо угоду про завдання зовнішньої функції $F(Object_{ji})$, що визначає наявність зв'язку для об'єкта. На відміну від зв'язку один до одного в даному випадку різні об'єкти сутності «Entity_1» можуть бути пов'язані з однаковим об'єктом сутності «Entity_2». Тому об'єкти сутності «Entity_2» повинні створюватися в одному екземплярі.

1. У разі простого первинного ключа такий зв'язок може бути реалізований наступним чином (вводиться допоміжна функція $Name_{UunqID}$, яка для заданої множини об'єктів X і заданого імені $name$ на момент часу $time$ формує підмножину унікальних ідентифікаторів об'єктів, які в даний момент часу мають ім'я $name$):

$$A \Leftarrow Object.Type(Entity_1.Name)$$

$$B \Leftarrow red(Ind[x], if\ (F(x), x, NULL), X))$$

$$Name_{UungID}(X, name, time) = red\left(ind\left([x], if\left(\begin{matrix} mem(name, xTimeToName(time)), \\ x.U_{ungID}, NULL \end{matrix} \right), X \right) \right)$$

$$C \Leftarrow Ind([x], Get.Name(x, Attribute_{11}.Name), B)$$

$$F \Leftarrow Object.Type(Attribute_{11}.Name, Property.Value.Values.Time(x, time)time), map[x^*],$$
$$\mathrm{mod}(Object.U_{unqID}(Property.Object(x)) \leftarrow Property.Link.Change(Property.Object(x)),$$
$$x, x^*, time)), Name.U_{unqID}(D, Property.Value.Values.Time(x, time))), \mathrm{mod}(G \Leftarrow$$
$$\Leftarrow Object.Create(Gen.U_{unqID}, Attribute_{11}.Name,$$
$$Property.Value.Values.Time(x/time, time), Object.U_{unqID}(Property.Object(x)) \leftarrow$$
$$\leftarrow Property.Link.Change(Object(x)), x, G.U_{unqID}, time))), C)$$

2. Для реалізації відносини один до багатьох зі складовим первинним ключем «Attribute_11.Name» і «Attribute_12.Name» порядок дій повністю збігається з попереднім пунктом, але зазначена там процедура виконується окремо для кожного атрибута «Attribute_11.Name» і «Attribute_12.Name».

Для реалізації модальності зв'язків необхідно ввести такі обмеження.

Модальність зв'язків: «Entity_1» – «можливо»; «Entity_2» – «повинен».

Дозволяється додавати нові об'єкти в множину об'єктів типу «Entity_1», а також забороняється безпосередньо додавати об'єкти типу «Entity_2». Об'єкти типу «Entity_2» можна додавати лише шляхом їх виділення з об'єктів типу «Entity_1».

Модальність зв'язків: «Entity_1» – «повинен»; «Entity_2» – «можливо».

Після виділення об'єктів типу «Entity_2» з властивості об'єктів «Entity_1» дозволяється додавання нових об'єктів типу «Entity_2», забороняється додавання нових об'єктів типу «Entity_1» без виділення нових об'єктів типу «Entity_2» з відповідних властивостей.

Модальність зв'язків: «Entity_1» – «можливо»; «Entity_2» – «можливо».

Після виділення об'єктів типу «Entity_2» з властивості об'єктів «Entity_1» дозволяється додавання як нових об'єктів типу «Entity_1», так і типу «Entity_2».

Реалізація зв'язків багато до багатьох.

Даний вид зв'язку може бути зведений до двостороннього використання відносин один до багатьох.

Модальність зв'язків: «Entity_1» – «можливо»; «Entity_2» – «можливо».

Дозволяється додавання нових об'єктів як типу «Entity_1», так і типу «Entity_2».

Модальність зв'язків: «Entity_1» – «необхідно»; «Entity_2» – «можливо».

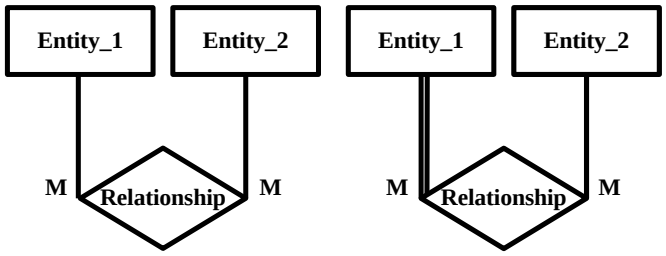

Рисунок 5 – Види зв'язків сутності багато до багатьох

Дозволяється додавання нових об'єктів типу «Entity_2», однак при додаванні нових об'єктів типу «Entity_1» потрібно додавати довідкові значення відповідних властивостей в об'єкти типу «Entity_2» на доданий об'єкт типу «Entity_1».

Правило 5. Будь-яка ER-модель даних відображається в SO-model.

Використовуємо принцип математичної індукції: припустимо, що *n* сутностей з можливими зв'язками між ними вже реалізовані в SO-model. Доведемо, що, виходячи з цього припущення, додавання ще однієї сутності з можливими зв'язками її з іншими сутностями з *n* вже існуючих також піддається реалізації в SO-model. Можливі наступні варіанти:

1. Додавання нової сутності без зв'язків зводиться до випадків, розглянутих в правилі 1;
2. Додавання рекурсивного зв'язку для деякої сутності зводиться до випадків, розглянутих у правилі 2;
3. Додавання нової сутності X і її зв'язку з уже існуючою сутністю B щодо один до одного, один до багатьох, багато до багатьох зводяться до випадків, розглянутих у правилі 3.

Таким чином, виходячи з припущення можливості ER-моделі даних бути реалізованою в SO-model для n сутностей, слідує її реалізуємість для n+1 сутностей, а оскільки реалізація ER-моделі даних в SO-model для однієї або двох пов'язаних сутностей була показана в правилах 1, 2, 3, то тим самим доведена можливість реалізувати ER-моделі даних в SO-model для будь-якої кількості сутностей і зв'язків між ними.

Акцентуємо увагу на необхідних порівняннях між ER-моделлю і SO-model:

1) сутність ER-моделі (в реляційному підході − це таблиця) відповідає типу об'єкта в SO-model;

2) атрибут сутності ER-моделі (в реляційному підході − стовпець таблиці) відповідає властивості об'єкта SO-model;

3) екземпляр сутності Entity в ER-моделі (в реляційному підході − рядок таблиці) відповідає об'єкту SO-model типу Entity;

4) зв'язок між сутностями Entity_1, Entity_2 по атрибуту первинний ключ Attr1 ER-моделі (в реляційному підході − зв'язок між таблицями в співвідношенні 1:1, 1:Б, Б:Б) відповідає зв'язку об'єктів типу Entity_1 з об'єктами типу Entity_2 за властивістю з ім'ям Attr1 у об'єктів типу Entity2.

Правило 6. В рамках зіставлення ER-моделі і SO-model в запропонованій SO-model підтримується темпоральність сутностей, атрибутів сутності ER-моделі даних, темпоральність зв'язків між сутностями ER-моделі даних.

Підтримка темпоральності на рівні сутностей ER-моделі даних пов'язана з можливістю додавання нових таблиць в базу даних без зв'язків їх з уже існуючими таблицями. Така можливість може бути реалізована в рамках реляційного підходу. Однак виникає неузгодженість між SQL- запитами до бази даних і програмними додатками, що обслуговують її (Impedance Mismatch). Програмні додатки доводиться переробляти. В SO-model темпоральність на рівні сутностей ER-моделі підтримується можливістю додавання об'єктів нового типу.

Підтримка темпоральності на рівні атрибутів сутності ER моделі даних пов'язана з можливістю створення таблиць зі змінним у часі складом стовпців або підтримки для кожного рядка таблиці свого переліку стовпців. Така можливість очевидно відсутня в рамках реляційного підходу. В SO-model така можливість підтримується: для виключення необхідного атрибуту на заданий момент часу достатньо записати в значення заданої властивості, що містить та не містить посилань, «порожньо» (null); для додавання необхідного атрибуту на заданий момент часу $time_p$ необхідно виконати функцію додавання нової властивості у вибраного об'єкту $X_p(Object_j, i_p, n_p)$, а також додати значення цієї властивості, що не містить посилань, за допомогою $PPVt(Obj_j, n_p, v_p, d_p)$.

Оскільки в SO-model рядки таблиці зберігаються окремо у вигляді об'єктів, існує можливість підтримки для кожного об'єкта свого складу атрибутів-властивостей.

Підтримка темпоральності на рівні зв'язків між сутностями ER-моделі даних означає можливість змінювати тип і модальність зв'язків між таблицями бази даних у часі, а також додавати або виключати ці зв'язки. Для реляційного підходу зміна типу або модальності зв'язку призводить до порушення цілісності даних. Якщо такі зміни виконуються, то вони призводять до необхідності суттєвої переробки програмних додатків по роботі з базою даних. Крім того, такі зміни є незворотними, тобто інформація, що раніше цей зв'язок мав інший тип або модальність, втрачається. Додавання або виключення зв'язків, як правило, також призводять до порушення цілісності даних і втрати інформації про структуру бази даних до моменту зміни.

Висновки

В рамках запропонованої SO-model підтримка темпоральності на рівні зв'язків між сутностями ER-моделі даних здійснюється за рахунок відсутності обмеження на підтримання такої цілісності даних в базовому функціоналі SO-model (як було показано, будь-який зв'язок з часом може стати зв'язком багато до багатьох з модальністю «можливо» з обох кінців зв'язку), причому за рахунок фіксації моментів часу в

моделі даних існує можливість збереження історії про вид та модальності зв'язку фактичних даних в будь-який момент часу.

Як було відзначено, реалізація ER-моделі даних в рамках реляційного підходу призводить до проблем масштабування при кластерному, хмарному зберіганні даних. В SO-model кожен об'єкт має свій глобальний унікальний ідентифікатор U_{unqID}. Це дозволяє зберігати об'єкти одного типу на різних комп'ютерах і підвищує масштабованість сховища даних. Також слід зазначити навмисну надмірність: значення кожної властивості може зберігатися з посиланням та без посилання. Відповідно до прийнятого протоколу можна реалізувати різні схеми узгодження цих значень, наприклад: завжди намагатися отримувати значення, що містить посилання, а при недоступності вузла видавати значення, що не містить посилання, при кожному успішному отриманні значення, що містить посилання, синхронізувати значення, що не містить посилання; завжди видавати значення, що не містить посилання, і до визначеного регламенту проводити синхронізацію цього значення зі значенням, що містить посилання. Все це дозволить підвищити доступність системи.

Таким чином, запропоновану модель сховища даних за рахунок її властивості уніфікації структури, яка враховує динаміку даних і їх зв'язків, можна розвивати в різних напрямках, використовуючи NoSQL рішення.

References

[1] H. Garcia-Molina, J.D. Uman, J. Widom, M. Ozsu, P. Valduriez, T. Connolly, C. Begg, R. Elmasri, S. B. Navathe, M. Lin, M. Tsuchiya, S. Member, M. P. Mariani, M. Sharma, G. Singh, R. Virk, "Database systems: a practical approach to design, implementation, and management," *Int. J. Comput, Appl. Technol*. 49(4), 2010, pp. 90-107.

[2] M. L. Chouder, S. Rizzi, R. Chalal, "Enabling self-service BI on document stores," *Workshop Proceedings EDBT/ICDT. Joint Conference Venice*, Italy, 2017.

[3] P. Atzeni, F. Bugiotti, L. Rossi, "Uniform access to NoSQL systems," *Inf. Syst*. 43, 2014, pp. 117-133.

[4] J. G. Enriquez, F. J. Dominguez-Mayo, M. J. Escalona, M. Ross, G. Staples, "Entity reconciliation in big data sources: a systematic mapping study," *Expert Syst, Appl*. 80, 2017, pp. 14-27.

[5] A. A. Imam, S. Basri, R. Ahmad, N. Abdulaziz, M. T. Gonzalez-Aparicio, "New cardinality notations and styles for modeling NoSQL document-stores databases," IEEE Region 10 Conference (TENCON), Penang, Malaysia, 2017.

[6] A. Imam, S. Basri, R. Ahmad, N. Aziz, M.T. Gonzlez-Aparicio, J. Watada, S. Member, "Data modeling guidelines for NoSQL document-store databases," *Computer Science Education. SI on Advancing Theory About the Novice Programmer*. Taylor & Francis, 2018.

[7] Pramod J. Sadalage, Martin Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison-Wesley, 2012, 192 p.

[8] C. J. Date, *Database Design and Relational Theory: Normal Forms and All That Jazz*, O'Reilly, 2012, 260 p. ISBN: 978-1-449-32801-6.

[9] Jan L. Harrington, *Relational Database Design and Implementation*, Morgan Kaufmann, 2016, 935 p.

[10] Louis Davidson, Jessica Moss, *Pro SQL Server Relational Database Design and Implementation*, Apress, 2016, 791 p. ISBN-13 (pbk): 978-1-4842-1972-0.

[11] Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems, 7th Edition*, Pearson, 2015, 1242 p.

[12] Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, *Database Systems: The Complete Book, 2nd Edition*, Prentice Hall, 2009, 1248 p. ISBN-10: 0-13-187325-3; ISBN-13: 978-0-13-187325-4.

Стаття надійшла: 20.05.2022.

Відомості про авторів

Осієвський Сергій Валерійович – кандидат технічних наук, доцент, начальник кафедри Харківського національного університету Повітряних Сил імені Івана Кожедуба.

Несміян Олексій Юрійович – кандидат технічних наук, викладач кафедри Харківського національного університету Повітряних Сил імені Івана Кожедуба.

S. Osiievskyi, O. Nesmiian

ALLOPHONE DATABASE CONTROL SYSTEMS SYNTHESIS FOR INTERACTION ORGANIZATION BETWEEN THE AIRCRAFT AUTOMATED CONTROL SYSTEMS (AC ACS) OPERATOR AND AUTOMATION COMPLEX MEANS (ACM)

Ivan Kozhedub Kharkiv National Air Force University, Kharkiv