DOI: https://doi.org/10.26565/2304-6201-2024-62-06

УДК (UDC) 004.942

Сузікова Олена Геннадіївна
канд. псих.наук, ст.викл. кафедри прикладної математики
Харківський національний університет імені В. Н. Каразіна,
майдан Свободи, 6, м. Харків, 61022
e-mail: osuzikova@karazin.ua
https://orcid.org/0009-0002-1305-1256

Роль семантичного аналізу в подоланні обмежень реляційної моделі даних

Актуальність. Стаття присвячена визначенню ролі семантичного аналізу при подолання обмежень традиційної реляційної моделі даних. Потреба в семантичному аналізі даних виникає внаслідок запиту на більш виразні та ємні концептуальні моделі даних. Проблема, що досі повністю не вирішена, полягає в тому, що класичні реляційні модель не мають прямої підтримки семантики даних - зв'язків, абстракції даних, успадкування, поліморфізму, інкапсуляції, складних об'єктів, а також динамічних властивостей об'єктів. Під семантикою розуміється використання певних конструкцій та методів для того, щоб виражати особливості прикладного середовища, які залишаються поза традиційною реляційною моделлю. Семантичний аналіз дозволяє визначити більш складні відношення та взаємодії між сутностями, включаючи класифікації, агрегації (складні об'єкти, які містять інші об'єкти), асоціації (зв'язки між сутностями). тощо. Ціллю вживання компонентів семантики є підвищення рівня абстракції при проектуванні, що робить модель більш універсальною. Абстракція в свою чергу - це в створення узагальнених моделей або класів, які представляють сутності в базі даних. Семантичний аналіз допомагає визначити, як саме дані будуть зберігатися та оптимізовані в базі даних. Він включає в себе вибір типів даних, індексування, нормалізацію/денормалізацію та інші аспекти проєктування бази даних. **Мета.** В дослідженні розглянуті питання семантичного аналізу даних при побудуванні реляційних моделей та реляційних баз даних. **Методи дослідження.** В статті поряд з теоретичним аналізом проблеми зібрано та проаналізовано за допомогою семантичного аналізу фактичний матеріал- структури даних з існуючих проєктів та хмарних додатків (кращі практики), зроблені висновки щодо можливостей підтримки семантики засобами існуючих реляційних моделей та реляційних баз даних. **Результати.** Було зроблено опис елементів семантики, проаналізована їхня роль в побудові моделі, виділено ряд семантичних шаблонів, як засобів подолання обмежень реляційної моделі.

Ключові слова: *семантичний аналіз даних, реляційна модель, концептуальні моделі даних, системи баз даних, моделі даних, логічний дизайн бази даних, семантичний аналіз предметної області.*

Як цитувати: Сузікова О. Г. Роль семантичного аналізу в подоланні обмежень реляційної моделі даних. *Вісник Харківського національного університету імені В.Н.Каразіна, сер. «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління»*. 2024. вип. 62. С.55-69. https://doi.org/10.26565/2304-6201-2024-62-06

How to quote: O. G. Suzikova, " The role of semantic analysis in overcoming the limitations of the relational data model " *Bulletin of V.N. Karazin Kharkiv National University, series "Mathematical modelling. Information technology. Automated control systems*, vol. 62, pp.55-69, 2024. [In Ukrainian] https://doi.org/10.26565/2304-6201-2024-62-06

1. Постановка проблеми

Концепція реляційної моделі була запропонована У.Коддом у 1969 році, це була модель зберігання даних у двомірних таблицях, де основною одиницею був кортеж, кортеж описував певну сутність, а поля кортежу описували атрибути цієї сутності [1]. Модель набула великої популярності саме тому, що була зручною, теоретично обґрунтованою та інтуїтивно зрозумілою. Розробники оцінили її ще й тому, що вона надала фахівцям методологію моделювання, не залежну від деталей фізичної реалізації. Але й по цей час багато розробників вважають, що реляційна модель не пропонує достатньо багатої концептуальної моделі для аспектів предметної галузі, що моделюється, які не відображаються на таблицях напряму[2].

З моменту створення реляційного підходу та концепції реляційних баз даних пройшло вже багато часу, але ще на зорі їх використання дослідники зазначали, що реляційна модель досить обмежена і не може відобразити область діяльності людей, що автоматизується, досить добре. У відповідь на цей виклик були зроблені численні спроби збагатити підхід і прилизувати можливості реляційних баз даних до можливостей об'єкт-орієнтованих мов, які оперують із більш складними структурами і, як результат, діяти на більш високому рівні абстракції.

Виникли об'єктно-орієнтована модель та об’єкт реляційна модель, що мали за мету надати більш досконалі засоби для відображення об'єктів реального світу, ніж реляційна модель[3].[4]. Їх спроби розширити традиційний реляційний підхід базувалися на намаганні реалізувати принципи з наступного списку:

-значеннями полів в таблицях (атрибутів відносин) можуть бути не тільки літеральне значення, а й складні об'єкти;

-значення атрибутів відносин не обов'язково є атомарними ;

-при побудові таблиць може використовуватися механізм успадкування;

-класи (таблиці) можуть включати певні операції (останній принцип вже імплементовано в механізмах вбудованих процедур в конкретних СУБД).

Але тут дослідники зіткнулися з наступною проблемою. Якщо системі управління базою даних закладено можливість оперування складними об'єктами, то процеси пошуку та перетворення інформації у базі даних істотно уповільнюються, оскільки самі об'єкти - складні і важкі. Зі зростанням потужності обчислювальної техніки ситуація покращується, але не радикально. Уповільнення процесів – це плата за складність об'єктів системи.

У той самий час цю проблему програмістам доводиться вирішувати щодня практично. У цьому плані виявився незаслужено забутим семантичний аналіз предметної галузі, який якраз у багатьох випадках дозволяє подолати протиріччя. Під семантичним аналізом мається на увазі змістовний аналіз даних та предметної області, результати якого дають можливість побудувати прийнятну та адекватну модель для реалізації в базі даних.

Метою цього дослідження є визначення ролі семантичного аналізу при подолання обмежень традиційної реляційної моделі даних.

Завданнями, що вирішуються, є вивчення сучасного стану проблеми, описання елементів семантики та засобів аналізу, та виділення семантичних шаблонів, як засобів подолання обмежень реляційної моделі.

2. Вивчення сучасного стану проблеми

Зараз словосполучення "семантичний аналіз" застосовується в основному в контексті великих даних та вирішення задачі аналізу природної мови. Але в даному випадку мається на увазі аналіз будь-якої предметної галузі, її сутностей та зв'язків між ними за змістом.

У відповідь на цей виклик з'явилися семантичні моделі даних. Як вказується в [2], певні розробки можна умовно класифікувати як "семантичні" моделі даних, оскільки їхньою відмінністю є те, що вони намагаються надати більше семантичного вмісту, ніж традиційна реляційна модель.

В центрі семантичного аналізу находиться певна предметна галузь, що автоматизується. Це частина реального світ та практичної діяльності людей, суть моделювання – у виділенні певних важливих об’єктів (сутностей) цієї області та їх аспектів з ціллю подальшого представлення їх таблицями бази даних та використання при побудові автоматизованого процесу. [2]

Структура або схема моделі бази даних утримує в собі як самі об’єкти, так і різні типи взаємозв’язки між ними. Об’єкти зазвичай не статичні, набір атрибутів описує певний стан такого об’єкту. Таким чином моделі задають певні структури даних для операцій моделювання, які використовуються для подальшого маніпулювання об’єктами схеми бази даних. Маніпуляції – це бізнес логіка того програмного продукту, що створюється для автоматизації.

Переваги реляційної моделі в тому, що вона, хоча й не здатні виражати всі зв’язки між об’єктами предметної області, але надає зручні засоби для зберігання та відображення даних фізично.

Для покращення моделі були запропоновані певні ідеї, першою була ідея про фізичну незалежність даних. Дослідники баз даних відчули, що користувач повинен бути вільним від деталей фізичної структури бази даних. Таким чином, користувач міг моделювати дані у спосіб, подібний до людського сприйняття програми.

Друга ідея передбачала захоплення додаткової семантики в процесі моделювання даних. Існуючі моделі були здатні визначати деякі семантики даних [5],[6],[7],[8].

Суть того, що розуміється під семантикою - це використання певних конструкцій та методів для того, щоб виражати певні особливості прикладного середовища, які залишаються поза традиційною реляційною моделлю.

Семантичний аналіз починається з розуміння бізнес-процесів та потреб користувачів системи. Це вимагає співпраці з представниками бізнесу та експертами предметної області для виявлення основних сутностей, взаємозв'язків і вимог до даних[9].

Після розуміння предметної області проводиться ідентифікація потрібних сутностей (об'єктів), їх зв'язків. що включає в себе визначення того, що є основними сутностями і як вони пов'язані одна з одною [9].

Для кожної сутності визначаються її атрибути, які відображають її характеристики та властивості. Це важливо для визначення як інформації буде зберігатися в базі даних. Це те, що відноситься до традиційної реляційної моделі даних "Entity-Relationship"[6],[9].

За останні кілька років новою парадигмою опису бізнес-процесів, у тому числі і для інформаційних систем, стала методологія, орієнтована не просто на сутності предметної галузі, а на артефакти бізнесу. Отже, бізнес-процеси описуються з погляду артефактів, якими бізнес маніпулює під час інформаційного обміну[10].

Інформація в системі це дані, визначені через артефакти, керовані бізнесом, разом з обмеженнями, що накладають умови на ці дані та які безпосередньо випливають із вимог предметної галузі організації[11].

Артефактно-орієнтоване моделювання ставить артефакти (значущі бізнесу об'єкти даних) та їх життєві цикли у фокус моделювання будь-яких бізнес-процесів. Цей тип моделювання за своєю суттю є декларативним: потік управління бізнес-процесом взагалі не моделюється, а випливає із життєвих циклів артефактів[10].

Зв'язування бізнес-процесів та їх даних, як і раніше, залишається відкритою проблемою, особливо при розгляді концептуальних моделей. Спільний розгляд моделей процесів та даних на концептуальному рівні дає багато переваг. зокрема, покращує розуміння всього інформаційного процесу в цілому, незалежно від інструментів, які використовуються для створення інформаційної системи[12].

Однією з причин, через яку артефактно орієнтована парадигма полегшує моделювання даних, є здатність моделювати відносини між об'єктами різної потужності.

Іншими словами семантичний аналіз дозволяє визначити більш складні відношення та взаємодії між сутностями, включаючи класифікації, агрегації (складні об'єкти, які містять інші об'єкти), асоціації (зв'язки між сутностями). тощо.

Також семантичний аналіз допомагає визначити, як саме дані будуть зберігатися та оптимізовані в базі даних. Це включає в себе вибір типів даних, індексування, нормалізацію/денормалізацію та інші аспекти проектування бази даних.

Ще одна функція семантичного аналізу - він допомагає визначити правила та обмеження, які забезпечують цілісність та безпеку даних в базі даних. Наприклад, обмеження вставки, видалення, зміни. Якщо об’єкти пов’язані через зв’язки, то вставка, видалення або модифікація одного об’єкта вплине на статус існування інших об’єктів, пов’язаних із ним.

Семантичний аналіз є важливою частиною проектування бази даних, оскільки він допомагає створити структуру даних, що точніше відображає реальну предметну область та більше відповідає потребам бізнесу та користувачів. Він також сприяє покращенню ефективності та продуктивності системи у подальшому використанні, полегшує поширення чи портування системи у майбутньому.

3. Елементи семантики для кращого розуміння структури даних

Ще в ранніх статях, що друкувались у 80-90х роках дослідники намітили компоненти семантики для майбутнього вдосконалення моделей даних. Ціллю вживання компонентів семантики є підвищення рівня абстракції при проектуванні, що робить модель більш універсальною. Абстракція в свою чергу - це в створення узагальнених моделей або класів, які представляють сутності в базі даних.

Абстракція дозволяє виділити головні характеристики сутностей та приховати деталі внутрішнього представлення. Абстракція у складних процесах чи системах дозволяє звести нетипові об'єкти або поведінку до типового, за рахунок чого модель суттєво спрощується[13].

Використання абстракцій дозволяє покращити модель даних і полегшити розуміння структури бази даних. Крім того, це дозволяє імплементувати більш зрозумілі та керовані інтерфейси для користувачів бази даних.

Наприклад, в статті Шміда та Свенсона [6] описані зв’язки та асоціації, а також пов’язана з ними семантика. В статті Сміта та Сміта [7] описаний шар абстракції в моделях, що використовувалися для моделювання бази даних. Тобто такі аспекти семантичного аналізу даних як абстракції, асоціації, класифікації, узагальнення та агрегації були відомі та описані теоретично вже досить давно. І саме вини є складовими тих моделей, що підпадають під категорію семантичних моделей даних.

Агрегація — включає об'єднання кількох об'єктів або сутностей в одну більш складну сутність. Це дозволяє моделювати комплексні відносини між даними та забезпечує більше структуровану та легко розуміючу базу даних. Це засіб, за допомогою якого зв’язки між типами низького рівня можна вважати типом вищого рівня. Реляційна модель даних може використовувати цю концепцію, наприклад, шляхом агрегування атрибутів для формування відношення. Таким чином семантичне моделювання даних дозволяє агрегувати типи сутностей (або відносини) для формування сутностей вищого порядку. За рахунок цього агрегація допомагає краще розуміти логічну структуру даних та розширює можливості моделювання складних об'єктів та їх взаємозв'язків.

Узагальнення —вказує на відносини між сутностями або класами, де одна сутність є більш загальною (батьківською), а інша - більш конкретною (дочірньою). Це засіб, за допомогою якого відмінності між подібними об’єктами ігноруються для формування типу вищого порядку, у якому можна підкреслити подібність. Узагальнення створює ієрархію, де батьківська сутність має загальні атрибути, які спільні для всіх дочірніх сутностей. Вона в цілому допомагає моделювати спадкування в базі даних, де конкретні сутності успадковують атрибути та характеристики від батьківської сутності. Це дозволяє використовувати загальні атрибути для всіх дочірніх сутностей.

Класифікація —включає в себе розділення сутностей або об'єктів на категорії чи класи на основі спільних характеристик. Це форма абстракції, у якій набір об’єктів вважається класом об’єкта вищого рівня. Класифікація дозволяє групувати подібні сутності разом та визначати загальні атрибути та властивості для кожної категорії. На практиці це допомагає створити більш організовану та структуровану базу даних, де сутності групуються за їхніми спільними характеристиками. Це полегшує пошук та аналіз даних.

Асоціація — асоціативні зв'язки вказують на взаємозв'язки між різними сутностями або об'єктами в базі даних. Це форма абстракції, у якій зв’язок між об’єктами-членами вважається об’єктом набору вищого рівня (Brodie, 1984)[14].Вона дозволяє представити, як об'єкти взаємодіють та співпрацюють один з одним. На практиці це сприяє створенню більш зрозумілих та реалістичних моделей даних[15].

Залежності – ще один важливий елемент семантики. Функціональні залежності - це концепція, яка визначає взаємозв'язок між атрибутами в базі даних. Вони грають важливу роль у визначенні того, як дані будуть зберігатися та організовані в таблицях.

Визначення функціональних залежностей - це один з видів аналізу семантики даних. Він є основою процедури нормалізації. Важливо правильно визначити функціональні залежності для кожної таблиці та атрибута в базі даних, оскільки це визначає структуру та забезпечує правильність операцій з базою даних.

Саме змістовний аналіз об'єктів дозволяє повісити рівень абстракції у програмі, що в результаті робить її більш компактною, універсальною, полегшує розширення функціоналу, портування та зміну коду. Оскільки семантика зв’язків втілена в операціях об’єднання, які явно вимагаються в реляційному запиті, запит буде мати більш компактний вираз у семантичній моделі.

Семантичні моделі даних зазвичай є розширеннями, тобто більш повними моделями в тому сенсі, що користувач здатний витягти повний спектр інформації з бази даних так само легко, як і в попередніх моделях. Однак усі ці моделі також забезпечують економію вираження, яку можна вважати сильнішою за повноту в такому сенсі: користувач не тільки може витягти точно таку саму інформацію, але більшу частину цієї інформації можна витягнути з більшою легкість.

Наприклад, з реляційною моделлю користувач повинен знати про атрибути, задіяні в неявному визначенні міжреляційних зв’язків, і виконувати складні операції саме над цими атрибутами, використовуючи проекції та об’єднання, щоб отримати інформацію через ці зв’язки. Якщо семантика відношення врахована в структурі даних, процес суттєво полегшується. У семантичній моделі операції чітко визначені у зв’язках. Таким чином, семантика вбудована в саму модель даних.

Підтримка цілісності. Традиційні моделі змушують користувача відстежувати зв’язки між об’єктами бази даних або підтримувати внутрішньо об’єктну узгодженість за допомогою навігації зв’язків на фізичному рівні. Семантичні моделі забезпечують механізми для визначення обмежень цілісності та водночас дозволяють користувачеві переглядати дані на рівні, відділеному від структури запису низького рівня[16].

4. Покращення реляційної моделі через абстракції

Абстракції представляють собою загальні та спрощені моделі або класи об'єктів, які можуть бути складними та різноманітними в реальному світі. Абстракції відкидають деталі та зосереджуються на ключових характеристиках.

Застосування абстракцій у реляційній моделі:

У реляційній моделі бази даних абстракції можуть бути представлені як окремі таблиці, де кожен рядок таблиці відображає конкретний об'єкт абстракції, а стовпці представляють атрибути абстракції.

Абстракції дозволяють спростити структуру даних, зробити її більш зрозумілою та дозволяють здійснювати операції над об'єктами абстракції на більш високому рівні абстракції.

Приклад використання абстракцій.

Розглянемо приклад створення абстракції для "користувача" в межах корпоративного порталу з великою кількістю ролей. Замість включення всіх можливих атрибутів користувача (ім'я, адреса, телефон, електронна пошта, тощо) в одну таблицю, можна створити абстракцію User. Ця абстракція міститиме базові атрибути, які є спільними для всіх користувачів, та можливо, посилання на інші таблиці для більш докладних даних. Це дозволить підтримувати однакову структуру для всіх користувачів і легше розширювати систему в майбутньому.

Значення використання абстракцій:

- Спрощення моделювання складних концепцій у реляційній моделі.
- Зменшення кількості дублювання даних та забезпечення їхньої єдності.
- Більша зрозумілість бази даних та легкість у підтримці.
- Підтримка вищого рівня абстракції для операцій над даними, що в цілому покращує продукт.

Використання абстракцій допомагає покращити якість та розуміння структури реляційної бази даних, зробити її більш модульною та підготовленою до розширення, що важливо при проектуванні баз даних для складних концепцій і систем.

Використання агрегацій та композицій

Агрегація в реляційній базі, як вже відмічалось, вказує на створення зв'язків між об'єктами, де один об'єкт (батьківський) містить в собі інші об'єкти (дочірні) як частини або підкомпоненти. Цей тип зв'язку дозволяє структурувати дані та представляти складні об'єкти як агрегати із відомими властивостями та залежностями.

Виділяють також композицію. Композиція - це особливий вид агрегації, де дочірні об'єкти пов'язані з батьківським об'єктом таким чином, що вони існують лише в межах цього батьківського об'єкта. Якщо батьківський об'єкт видаляється, то і всі його дочірні об'єкти також видаляються.

Використання агрегацій та композицій допомагає краще моделювати складні відносини між об'єктами та їхніми частинами в реляційній базі даних. Наприклад, якщо ви моделюєте автомобіль, агрегація дозволяє вам представити двигун, колеса, крісла та інші компоненти автомобіля як частини цього автомобіля.

Агрегація та композиція також грають важливу роль в моделюванні ієрархії підкласів в базі даних. Вони дозволяють структурувати дані та створювати зв'язки між батьківськими та дочірніми класами.

Наприклад, якщо у вас є базовий клас "Авто" і підкласи, такі як "Двигун" та "Ходова частина", агрегація допомагає представити зв'язок між "Авто" і "Двигун" де "Авто" є батьківським класом, а "Двигун" є дочірнім класом.

Значення використання агрегацій та композицій:

- Структурування та моделювання складних відносин між об'єктами в реляційній базі даних.
- Підтримка ієрархічних структур та відносин між батьківськими та дочірніми класами в моделях даних.

• Забезпечення можливості представляти складні об'єкти та їхні відносини в базі даних з точністю до деталей.

• Агрегація та композиція є важливими інструментами при моделюванні структури даних в реляційних базах даних, особливо при роботі з об'єктами, які мають складні відносини та ієрархії.

Крім того, до засобів семантичного аналізу можна віднести знання про структуру та складність об’єкта та функціональні залежності в системі атрибутів. Функціональні залежності, наприклад, широко використовуються для вдосконалення бази зданих за процедурою нормалізації, така процедура дозволяє запобігти дублюванню інформації. Зворотна процедура денормалізації навпаки дзволяє дублювати потрібні дані, якщо це відповідає вимогам проекту.

Використання класифікації та узагальнення

Як вже відмічалось, класифікація – це розбиття об’єктів на групи за певними характеристиками. У системі керування персоналом можна використовувати класифікацію для групування працівників за функціональними ролями, наприклад, "менеджери," "розробники," "QA-інженери" і т.

Узагальнення може бути використане для моделювання ієрархії, де є певна батьківська сутністю, а її дочірні сутності зі спільними атрибутами. Класифікація може бути використана для групування продуктів у магазині за їхніми категоріями, підкатегоріями тощо (наприклад, "побутова техніка"> "техніка для кухні"> "дрібна техніка для кухні").

Класифікація та узагальнення допомагають краще організувати та моделювати дані в базі даних, створюючи логічні зв'язки та ієрархії між сутностями. Це полегшує розуміння та управління даними в системі.

Значення використання класифікації та узагальнення:

• Спрощення структури даних шляхом групування сутностей та атрибутів за спільними ознаками.

• Зменшення дублювання даних та сприяння консистентності в базі даних.

• Можливість використовувати поліморфізм для роботи з різними класами через їх загальний суперклас.

• Забезпечення більшої гнучкості та розширюваності бази даних при зміні вимог до системи.

Використання асоціативних зв'язків

Асоціативні зв'язки в базах даних грають важливу роль у відображенні складних відносин між об'єктами. Ці зв'язки дозволяють з'єднувати два або більше об'єктів у базі даних та визначати їхні взаємовідносини. Це особливо корисно в ситуаціях, коли об'єкти мають складну систему відносин (тобто не один зв’язок).

Асоціативні зв'язки, наприклад, можуть містити додаткову інформацію про сам зв'язок. Наприклад, в асоціативному зв'язку між "Користувачами" і "Товарами" може бути включено інформацію про номер та дату ліцензиії або номер гарантійного зобов’язання, та граничну дату дії.

Асоціативні зв'язки можуть бути використані для вирішення різних завдань в базі даних:

Зв'язки багато-до-багатьох: асоціативні зв'язки часто використовуються для моделювання відносин багато-до-багатьох між об'єктами[10]. Наприклад, в базі даних для інтернет-магазину може бути асоціативний зв'язок між таблицею "Користувачі" і "Товари", де кожен користувач може мати багато товарів у своєму кошику, і кожен товар може бути у кошику декількох користувачів.

Запити і фільтрація: асоціативні зв'язки дозволяють виконувати складні запити, щоб знайти певні зв'язки або фільтрувати дані на основі цих зв'язків.

Відносини при семантичному моделюванні аналізуються з точки зору їх змісту та призначення. Концептуально конструкт відношення може відображатися в моделі як атрибут, сутність, незалежний елемент або функція. Відношення, втілене атрибутами, — це зв’язок, у якому атрибут одного об’єкта пов’язаний з іншим об’єктом, вказує на нього або походить від нього. Це традиційний підхід.

Альтернативно відносини також можна розглядати як незалежні об’єкти, відмінні від сутностей. Це не означає, що фізична база даних представляє зв’язки та сутності з різними структурами, але принаймні на концептуальному рівні сутності розглядаються окремо від зв’язків.

Зв’язок представляється як сутність, якщо зв’язок двох або більше об’єктів концептуально описує окремий об’єкт моделі.

Значення використання асоціацій:

- Можна відтворити більш складні зв’язки ("багато до багатьох")
- Можна передати додаткову інформацію
- Можна відтворити кілька типів зв’язку між тими самими об’єктами
- Це робить операції з базою даних більш ефективними і швидкими.
- Дозволяють уникнути дублювання даних та зберігати відносини між об'єктами зручним способом.

Використання функціональних залежностей

Функціональні залежності допомагають визначити, які атрибути залежать від інших, і які обмеження можна застосовувати до цих залежностей. Вони допомагають визначити ключові атрибути в таблицях бази даних. Ключові атрибути визначаються як ті, від яких залежать інші атрибути, і які можна використовувати для унікальної ідентифікації записів в таблиці.

Припустимо, у вас є таблиця "Замовлення" (Orders) у базі даних, і ця таблиця містить наступні атрибути:

- order_id (ідентифікатор замовлення)
- customer_id (ідентифікатор клієнта, який зробив замовлення)
- order_date (дата замовлення)
- lineitem_id (це рядок, що утримує товар, що замовляється, його кількість та ціну)
- total_amount (загальна сума замовлення)

У цій таблиці існує функціональна залежність між "order_id" та "customer_id". Це означає, що кожне "order_id" залежить від конкретного "customer_id". Іншими словами, кожен замовник (customer) може мати багато замовлень, і для кожного замовлення існує конкретний клієнт (customer).

Очевидно, що існує функціональна залежність між "order_id" та "order_date", бо кожне замовлення має свою унікальну дату замовлення.

Крім того, "total_amount" залежить від "order_id". Кожне замовлення має свою власну загальну суму.

Також "total_amount" залежить від "lineitem_id". Кожна сума замовлення формується за рахунок виду товару, ціни та кількості.

Ці функціональні залежності визначають взаємозв'язок між різними атрибутами в таблиці "Замовлення". Вона показує, як атрибути залежать один від одного та допомагають відобразити взаємозв'язок між клієнтами, замовленнями та їх даними в базі даних.

Функціональні залежності грають ключову роль в процесі нормалізації даних. Нормалізація полягає в розбитті таблиць на менші та більш атомарні структури для зменшення дублювання та заборони аномалій даних. Функціональні залежності визначають, які атрибути повинні бути в окремих таблицях для забезпечення нормалізації.

Розуміння функціональних залежностей допомагає покращити оптимізацію запитів в базі даних. Якщо ви знаєте, які атрибути залежать від інших, ви можете ефективніше створювати запити та індекси для прискорення доступу до даних.

Також функціональні залежності допомагають уникнути неправильного проектування бази даних та ризику помилок у даних. Вони надають ясний варіант того, як атрибути пов'язані між собою.

Значення використання функціональних залежностей:

- Зменшують кількість помилок
- Допомагають уникнути аномалій даних, таких як дублювання або втрати даних
- Дозволяють обмеження можливостей вставки, оновлення та видалення даних в таблицях
- Грають ключову роль в процесі нормалізації даних
- Допомагають ефективніше створювати запити та індекси для прискорення доступу до даних.

5. Семантичні шаблони як засоби подолання обмежень реляційної моделі з прикладами з проектів.

Патерни проектування баз даних - це загальні, перевірені часом рекомендації та рішення для моделювання та розробки баз даних. Як і в інших сферах програмування, семантичні патерни проектування баз даних служать як шаблони або рекомендації, які допомагають вирішувати типові завдання та проблеми при розробці та керуванні базами даних.

Основна ідея застосування патернів проектування баз даних полягає в тому, щоб використовувати так звані "best practice" вже перевірені та ефективні способи роботи з даними, замість того, щоб винаходити їх заново для кожного проекту. Саме патерни та шаблони пропонують шляхи подолання обмежень традиційної реляційної моделі даних. Поряд з цим патерни проектування допомагають забезпечити більшу стабільність, підтримуваність та розширюваність бази даних, а також знизити ризик помилок та неправильного проектування.

Одним з найбільш відомих дослідників патернів при проєктуванні реляційних баз даних є Мартін Фаулер [17]. Він виділив більш ніж 50 патернів для реляційних баз даних, що дозволяють покращити відображення предметної області.

Серед найбільш відомих його патернів є патерни для наслідування та відтворення асоціативних зв'язків.

Патерн "Наслідування конкретної таблиці"

Реляційні бази даних не підтримують успадкування – факт, який ускладнює об'єктно-реляційне відображення. Цей патерн використовує узагальнення для створення ієрархії таблиць в базі даних. В Concrete Table Inheritance існує таблиця для кожного конкретного класу в ієрархії успадкування, та кожна з них представляє собою конкретний об'єкт.

На основі шаблонів Фаулера можна створити в базі даних певні аналоги об'єктів та класів, що властиві об'єкт орієнтованим мовам програмування. Назвемо їх шаблонами.

Шаблон "Суперклас"("Superclass")

Цей підхід має на меті використання суперкласу для узагальнення спільних атрибутів, які є характерними для декількох класів. Наприклад, якщо у вас є класи "Користувачі" та "Команда" ви можете створити суперклас "Користувач", який містить загальні атрибути, такі як ім'я та електронна пошта, і нащадки цього класу будуть мати додаткові атрибути, наприклад, позиції для членів команди.

Приклад з проєкту Zendesk, де імплементовано суперклас User, до якого входять три підкласи організовані за різними ролями в системі (що означає, що в них різна логіка в системі) - це Admin, Agent, End-User. При чому адміністратори та агенти – це члени команди, а кінцевий юзер – це клієнт.

Всі дані про структуру об'єктів, їх поля та атрибути отримані за допомогою сервісу https://skyvia.com/

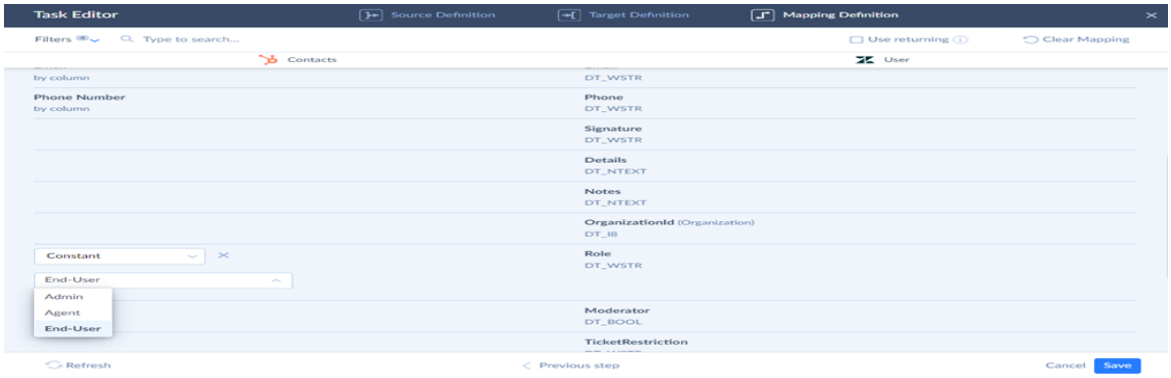

Рис.1.Імпорт в Zendesk, з вибором підкласу юзерів за роллю з суперкласу User.
Джерело: https://skyvia.com/

Множинне успадкування — це механізм, за допомогою якого об'єктам в ієрархії узагальнення/спеціалізації дозволяється успадковувати властивості від кількох об'єктів вищого рівня (наприклад, вони не можуть бути створені без ідентифікаторів об'єктів-батьків). Це зручно для деяких об'єктів, коли властивості, що успадковуються, не перетинаються. Проблема виникає, коли один спеціалізований об'єкт успадковує ту саму властивість від двох об'єктів вищого рівня.

В цьому випадку семантична модель може дозволяти або забороняти використання множинного успадкування, або пропонувати певні вбудовані механізми для обробки конфліктів, які можуть виникнути.

Приклад з платіжної системи Square. Сутність Invoice не може бути створена без попереднього створення сутностей Order та PaymentRequest. При чому інвойс (Invoice) від замовлення (Order) бере властивості, що пов'язані з товарами, кількістю, цінами, тобто з товарним наповненням

інвойсу. А від PaymentRequest успадковує параметри, що пов'язані з засобами та порядком оплати [18].

Патерн "Наслідування таблиці", зустрічається також з назвою "Table Inheritance", "Single Table Inheritance"

Цей патерн також використовує узагальнення та класифікацію для створення ієрархії таблиць в базі даних. На кожен клас або об'єкт створюється окрема таблиця, а головна таблиця містить загальні атрибути. Це дозволяє моделювати класи зі спільними характеристиками та використовувати агрегацію для посилання на окремі таблиці, які містять специфічні атрибути для кожного класу.

На основі цього патерну, наприклад, можна створити шаблон абстрактного класу.

Шаблон "Абстрактний клас" ("Abstract Class")

Абстрактний клас в об'єктних мовах програмування – це клас, що не може мати конкретних екземплярів, але екземпляри можуть бути в його класів-нащадків. Приклад - клас "Тварини", просто тварини в реальному світі не існують, але існують конкретні тварини, що відносяться до його класів-нащадків - Кішок або Собак.

Абстрактний клас в реляційній базі даних можна імплементувати через певні поля, що пов'язують батьківський клас та класи-нащадки. При чому, на відміну від просто наслідування таблиця, що відповідає абстрактному класу може не мати атрибутів крім ключового атрибуту (назви та/або ідентифікатора).

Приклад з реального проекту – платіжна система Square. В системі імплементовано абстрактний клас CatalogItem – це товари, продукти з каталогу, абстрактний продукт не має артікулу - SKU, ціни та валюти продажу, а ось його дочірня структура CatalogItemVaration має артікул -SKU, ціну та валюту, та не може бути створена без Id батьківської структури.

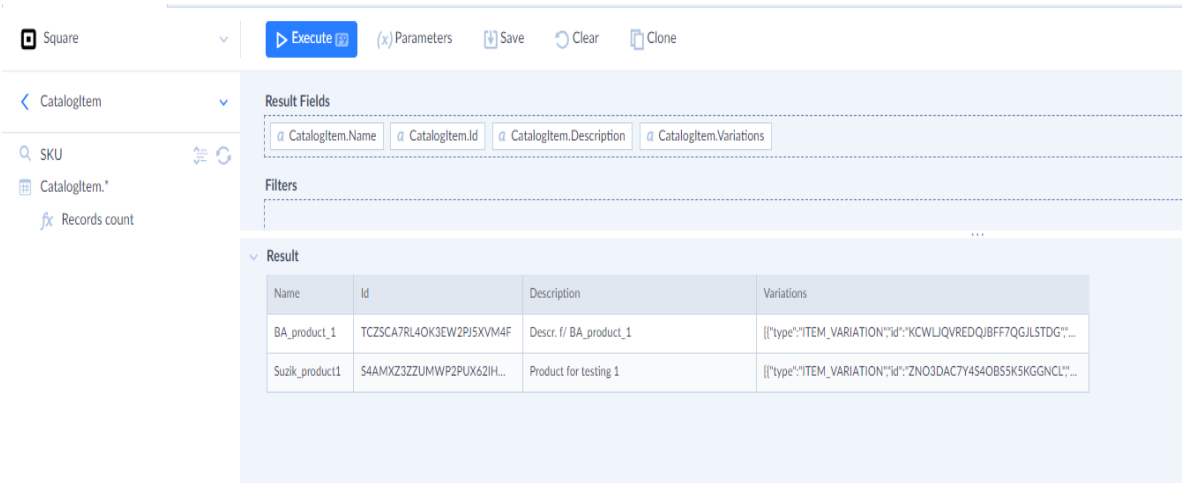

Рис 2. Абстрактний клас CatalogItem з платіжної системи Square (не має артікулу)
Джерело: https://skyvia.com/

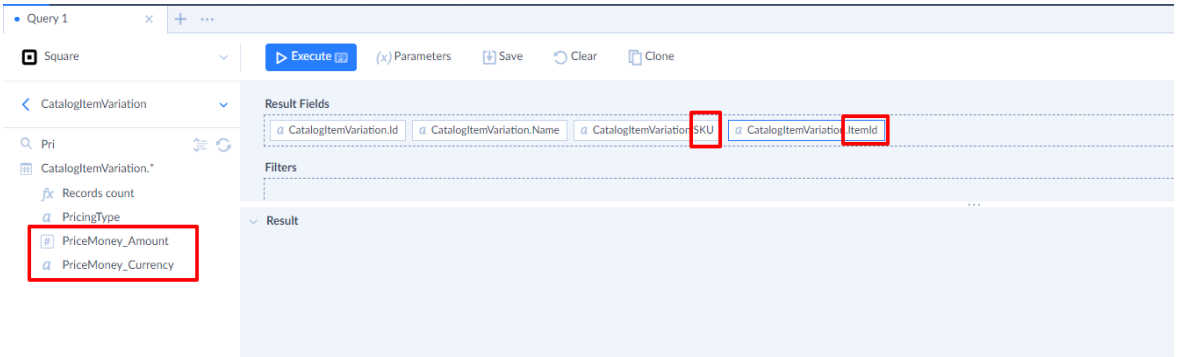

Рис 3. Клас CatalogItemVariation з платіжної системи Square (має артікул, ціну та валюту продажу), що не може бути створений без ідентифікатора батьківського об'єкту CatalogItem
Джерело: https://skyvia.com/

Патерн "Наслідування класу таблиці" ("Class Table Inheritance")

Цей патерн використовує агрегацію для створення ієрархії таблиць, при чому кожна таблиця представляє конкретний клас або об'єкт. В цьому випадку, агрегація дозволяє зв'язувати таблиці з загальними атрибутами, що спільні для всіх класів (наприклад, через ID об'єкта).

На основі цього патерну можна створити такі структури, як Агрегат та Компонент.

Шаблон "Агрегат" ("Agregate")

Це рішення має на увазі використання агрегації для створення складних але цільних структур даних, які включають у себе різні класи. Наприклад, агрегація може бути використана для створення складних об'єктів, що містять в собі інші об'єкти, де кожен об'єкт може належати до різних класів. Зв’язки відтворюється за допомогою зовнішніх ключів.

Прикладом Агрегату може бути сутність Country(країна) в екомерс системі Shopify, Країна утримує в собі незалежні Shipping Zone – зони постачання, від приналежності до певної зони залпежить вартість постачання. Як зони, так і країна можуть бути використані незалежно, і це реальні об’єкти предметної області.

Шаблон "Компонент"("Component")

Цей підхід використовує композицію для створення складних об'єктів, які складаються з менших компонентів. Кожен компонент може мати свої атрибути та пов'язані дані але обов’язково має посилання на складний об’єкт (наприклад, через ID об'єкта). В базі даних, композиція дозволяє структурувати дані об'єктів і їх компонентів у реляційній моделі. Особливості компонента - він існує не сам по собі, а лише як частка іншого об’єкту.

Особливості агрегату та компоненту такі, агрегат - це цільна структура, яка містить в собі колекцію об'єктів та має незалежний життєвий цикл, тоді як компонент - це частина цієї структури, яка існує та функціонує лише в контексті агрегата.

Прикладом компоненту, що сам по собі не існує може бути сутність Line, що використовується у багатьох платіжних та бухгалтерських системах (Stripe, Square, Xero, Quickbooks). Ця сутність представляє собою окремий рядок інвойсу. Сама по собі вона не існує, а є тільки часткою документу.

Патерн "Зіставлення Таблиці Асоціацій" (Association Table Mapping)

Ассоціативні зв'язки в реляційних базах даних відображають відносини між сутностями, коли ці відносини містять додаткову інформацію.

Цей патерн використовує окрему таблицю, яка містить зв'язки між двома або більше сутностями, а також можливі атрибути цього зв'язку. Наприклад, якщо ви маєте сутності "Користувач" і "Продукт," і ви хочете відстежувати взаємозв'язок між користувачами та продуктами, ви можете створити таблицю "UserProduct" з власним ідентифікатором зв’язку та полями, які вказують на користувача, продукт і, можливо інші дані, наприклад, серійний номер продукту, номер ліцензії або гарантійний термін.

Шаблон "Зв’язок як сутність"

В цьому шаблоні створюється спеціальна сутність (або сутності. Якщо є кілька типів зв’язків) для того, щоб здійснювати поєднання двох інших сутностей.

Приклад імплементації з проєкту Hubspot CRM. Раніше (в попередніх версіях API) зв’язки між сутностями були в системі імплементовані як поля, що утримують посилання на одну чи кілька пов’язаних сутностей. У випадку кількох таких зв’язків поле утримувало рядок з вмістом масиву ідентифікаторів. Зараз в поновленій версії API з’явився новий тип об’єктів – Association (Ассоціація), за допомогою цієї сутності можна поєднати два будь які об’єкти в системі, при чому між тими ж двома об’єктами може бути встановлено багато типів різних зв’язків.

Приклад такого зв’язку з системи Hubspot. Сутність CompanyEngagements зв’язує компанію та активності, щоб залучити її до числа клієнтів. Вона має власний ідентифікатор та посилання на сутності, які вона поєднує.

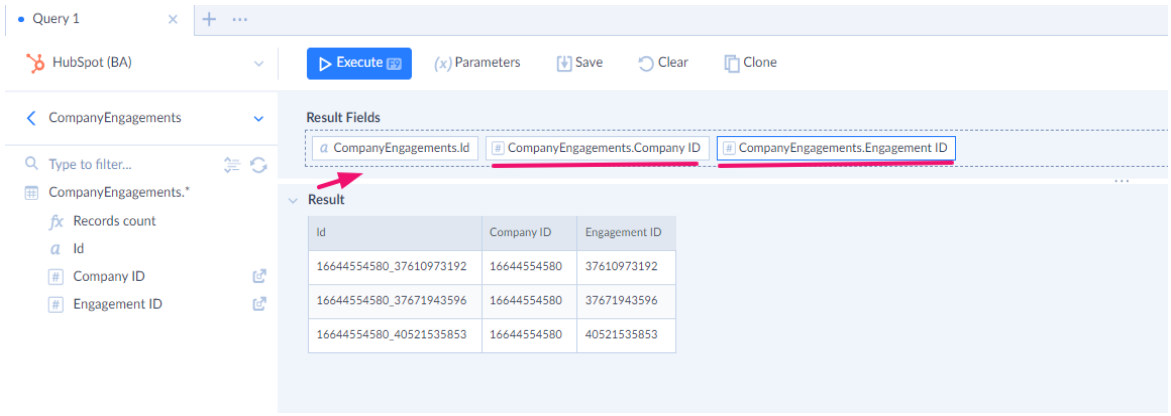

Рис 4. Таблиця CompanyEngagements Hubspot CRM
Джерело: https://skyvia.com/

Тобто, реалізація асоціативних зв'язків може варіюватися в залежності від специфікацій бази даних та потреб проекту.

Шаблон "Інкапсуляція" (Incapsulation)

Інкапсуляція, що є базовим принципом ООР, прямо суперечить принципам відкритості та вивільненості комбінування значень у кортежах SQL, що створює суттєві бар'єри для ефективного використання реляційної бази, як довготривалого сховища складених об'єктів класу (можливо, що це випливає із певних теоретичних неузгодженостей) .

Але рішення можливе, воно має на увазі створення додаткового об'єкту для заміни оригінального для того, щоб обмежити права доступу. Сервіс, який звертається за даними, отримує тільки ті дані, що відкриті для нього, а не до всіх. Фактично – це створення тимчасових або постійних таблиць, що утримують обмежену інформацію, що потрібна для користувача, при чому вся зайва інформація прихована та не є доступною.

Приклад з реального проєкту. В емейл-маркетинг системі Mailchimp нема прямого доступу до таблиці юзерів або контактів системи. Можливе звернення тільки до сутності ListMembers, це підписники з певного листу розсилок.

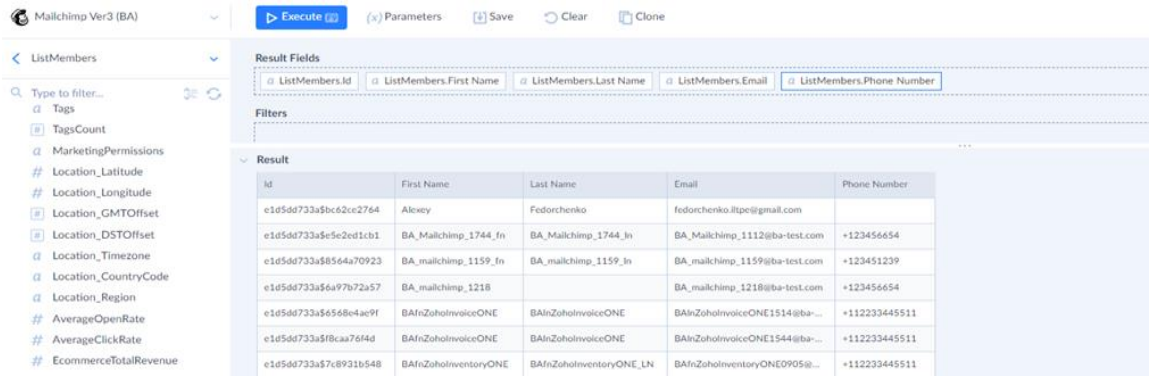

Рис 5. Вибірка з таблиці ListMembers з системи Mailchimp API V3.
Джерело: https://skyvia.com/

Для таблиць з функціональними залежностями широко відомими є підходи з нормалізації чи денормалізації таблиць, що зумовлено завданнями, що вирішуються.

Нормалізація та денормалізація

Відповідно до потреб певного ресурсу, що проєктується, ви можете вибрати паттерни нормалізації для зменшення дублювання даних або денормалізації для покращення швидкодії запитів. Нормалізаціія видвляє функціональну залежність між полями таблиці. Денормалізація створює певну функціональну залежність в системі даних.

Нормалізація включає в себе розбиття таблиць на менші та більш атомарні структури, за допомогою визначення первинного ключа (Primary Key) для кожної таблиці та встановлення зовнішніх ключів (Foreign Keys) для вираження залежностей між таблицями. Цей процес вимагає

аналізу структури даних та їх взаємозв'язків для визначення оптимальної організації даних в базі даних[18].

Нормалізація може бути поділена на кілька рівнів (результатом буде певна нормальна форма, від першої нормальної форми (1NF) до п'ятої нормальної форми (5NF)). Кожен рівень нормалізації має на увазі певні правила та вимоги щодо організації даних в таблицях.

Денормалізація – в противагу, це процес збільшення кількості дубльованих та зайвих даних у базі даних для підвищення ефективності запитів та покращення продуктивності системи. Вона протистоїть нормалізації, яка мінімізує дублювання даних та забезпечує консистентність даних.

Денормалізація застосовується у випадках, коли швидкість операцій з базою даних стає важливішою за ефективність витрат ресурсів на збереження даних.

Ці шляхи допомагають вдосконалити реляційні структури даних, роблять їх більш гнучкими та ефективними для моделювання різних відносин та ієрархій в базі даних. Розуміння цих паттернів допомагає проектувати більш ефективні та модульні бази даних.

Нормалізація – це доволі розповсюджена процедура при проєктуванні баз даних. Денормалізація більш рідкісна операція. Наведу приклад навмисно зробленої денормалізації:

Hubspot CRM дублює інформацію про асоціативні зв’язки між сутностями, наприклад, при створенні асоціації між компанією та квитком техпідтримки, фактично одна і та ж інформація складається в дві симетричні таблиці CompanyTickets та TicketCompanies для того, щоб прискорити пошук зв’язків.

В реальних проектах можна знайти менш фундаментальні, але не менш корисні рішення та лайфхаки для вирішення практичних завдань. Деякі з них до речі дозволяють обійти обмеження традиційних реляційних баз даних на атомарність полів та використовувати складні об’єкти в якості атрибутів.

Шаблон "Масив" ("Array").

Масив передається в поле як рядок та при отриманні розбирається за допомогою процедури парсінгу.

Приклад з системи Hubspot – сутність Deals має поля-рядки, що містять масиви ідентифікаторів.

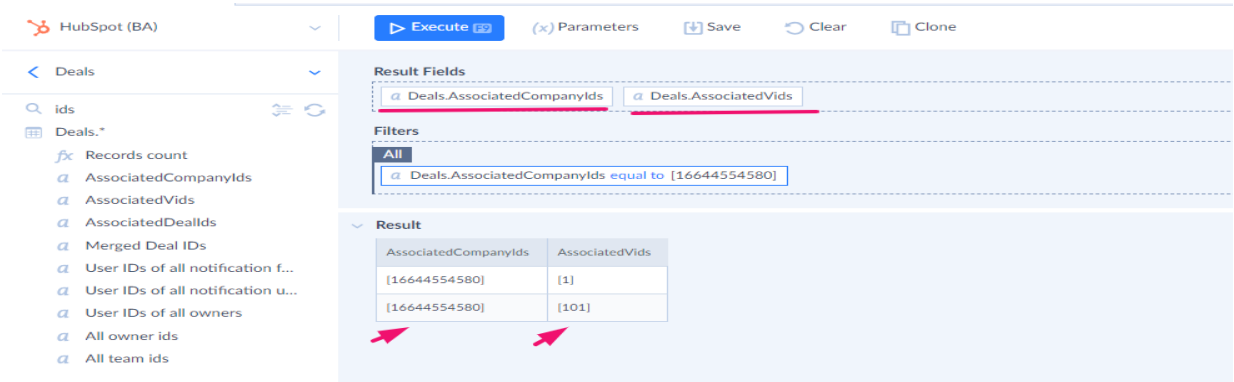

Рис 6. Масиви, що передаються в поля як рядки. Hubspot CRM.
Джерело: https://skyvia.com/

Шаблон "Складний атрибут"(мається на увазі робота з атрибутом – об’єктом з структурою JSON, XML, HTML).

Одною з проблем реляційної моделі є те, що вона не дозволяє використовувати атрибути-сутності, тобто не атомарні складні об’єкти. В деяких випадках, особливо для серійних складних структур даних, використовують структури JSON, XML для зберігання характеристик серійних об’єктів, або конструкції HTML для зберігання певних шаблонів. В цьому випадку атрибути-сутності представлені у вигляді такої структури можна помістити в рядок в одному зі стовпців таблиці.

Тобто, складний атрибут що по суті є конструкцією JSON, XML, HTML передається в поле у вигляді звичайного рядка, але впорядкованого по шаблону. Для відновлення об’єкту рядок розбирається по цьому шаблону.

Приклад з реального проекту: на малюнку ви можете бачити структуру складного атрибуту Line типу рядок об’єкту Invoice з хмарного застосунку Quickbooks Online, цей атрибут відповідає за рядки інвойсу, що утримують продукт, його деталі, одиниці виміру, вартість та підсумкову суму рядка.


```
[
  {
    "Id": "1",
    "Amount": 3500.0,
    "LineNum": 1.0,
    "Description": "Custom Design",
    "DetailType": "SalesItemLineDetail",
    "SalesItemLineDetail_ItemRefId": "4",
    "SalesItemLineDetail_ItemRefName": "Design",
    "SalesItemLineDetail_UnitPrice": 350.0,
    "SalesItemLineDetail_Qty": 10.0,
    "SalesItemLineDetail_ItemAccountRefId": "82",
    "SalesItemLineDetail_ItemAccountRefName": "Design income",
    "SalesItemLineDetail_TaxCodeRefId": "NON"
  },
  {
    "Amount": 3500.0,
    "DetailType": "SubTotalLineDetail"
  }
]
```


Рис 7. Вид атрибуту Line (типу рядок) об’єкту Invoice з хмарного застосунку Quickbooks Online.
Джерело: https://skyvia.com/

Шаблон "Важкий об’єкт"

В таблиці розміщується не сам об’єкт, а тільки його адреса. Для того, щоб фільтрувати або шукати певні об’єкти додаються метадані об’єктів за якими ведеться фільтрація, сортування або пошук.

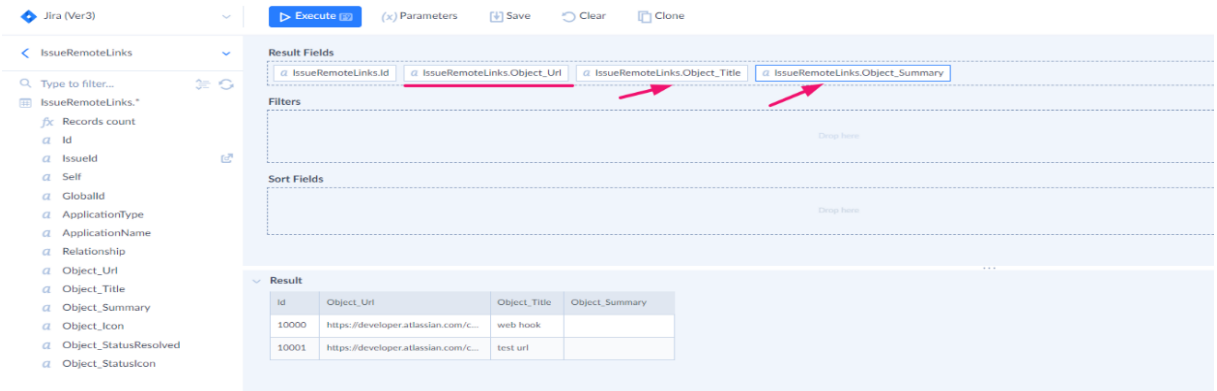

Рис 8. Адреса та метадані важкого об’єкта в Jira..
Джерело: https://skyvia.com/

Приклад з проєкту Jira. В версіях API V2, V3 використано сутність IssueRemoteLinks, що утримує інформацію про різні об’єкти, що відносяться до певної проблеми програмного забезпечення (Issue – це вид завдання, що описує технічну проблему, що треба вирішити). Об'єктами, що відносяться до такого завдання можуть бути файли, зображення, аудіо або відеоматеріали для опису проблеми. Сутність надає адресу такого об'єкту та метадані для пошуку, наприклад, назву та опис.

2 Висновки

При проектуванні баз даних однією з ключових складових є розуміння та відображення предметної області. Це завдання вимагає семантичного аналізу, який дозволяє створити таке відображення якомога більш точним, коректним і ефективним.

В той час як найбільш традиційні моделі даних забезпечують лише один засіб представлення даних. Семантичні моделі даних за допомогою абстракцій дозволяють користувачеві моделювати та переглядати дані на багатьох рівнях. Це надає розширені можливості для моделювання ситуацій "реального світу", оскільки перегляд даних на багатьох рівнях узгоджується з тим, як люди бачать світ.

Семантичний аналіз включає в себе наступні ключові аспекти:

- Використання абстракцій - класифікацій, узагальнень, агрегацій та асоціативних зв'язків - у проектуванні бази даних допомагає покращити структуру даних та зробити її більш зрозумілою та ефективною. Це полегшує роботу з базою даних, сприяє кращому розумінню предметної області та сприяє вдосконаленню якості та продуктивності системи у подальшому використанні.
- Використання знань про структуру об’єкту дозволяє вирішити проблему з атомарністю даних реляційних таблиць та наблизити їх можливості до можливостей об’єктних мов програмування.
- Використання найкращих практик, патернів проєктування, перевірених рішень, що дозволяють обійти певні обмеження реляційної моделі. При використанні таких стандартних підходів до проектування можна створити більш організовану та легко розширювану базу даних, що відповідає потребам вашого проекту та краще описує предметну область.

Вибір конкретного методу реалізації, патерну залежить від потреб користувачів та структури даних у проекті. Кожен з цих підходів має свої переваги та недоліки, і важливо враховувати їх при проектуванні конкретної бази даних.

REFERENCES

1. Codd, E. F. Extending the database relational model to capture more meaning. ACM Trans. Database Syst. 4, 4 (Dec.), 1979, pp.397-434.
2. Joan Peckham, Fred J. Maryanski. Semantic Data Models. ACM Comput. Surv. 20(3).1988, pp.153-189.
3. M. A. Garvey, M. S. Jackson. Introduction to Object-Oriented Database.Inf. and Software Technol.- 31, N 10.1989.pp.521-528
4. Woelk, D, Kim, W., and Luther, W.An object-oriented approach to multimedia databases. In Proceedings of the ACM SIGMOD Conference (Washington, D.C.). ACM, New York,1986, pp. 311-325.
5. Chen, P. 1976. The entity—relationship model: Toward a unified view of data. ACM Trans. Database syst. 1, 1 (Mar.), 1976, pp.9-36.
6. Chen, P., Ed. Entity—Relationship Approach: The Use of the ER Concept in Knowledge Representation. North-Holland, Amsterdam. 1985.
7. Schmid, H. A., Swenson, J. R. 1975. On the semantics of the relational data model. In Proceedings of the ACM SIGMOD Conference (San Jose, Calif.). ACM, New York, pp. 211—223.
8. Smith, J. M., Smith, D. C. P. Database abstractions: Aggregation and generalization. ACM Trans. Database Syst. 2, 2 (Mar.), 1977, pp 105-133.
9. Hey, D.C. Data Model Patterns: Conventions of Thought. David C. Hey. Dorset House Publishing, 1996.288 p.
10. Diana Borrego, Rafael M. Gasca, María Teresa Gómez-López, Automating correctness verification of artifact-centric business process models,т Information and Software Technology,Volume 62, 2015, pp187-197.
11. Xavier Oriol, Ernest Teniente, Simplification of UML/OCL schemas for efficient reasoning, Journal of Systems and Software, Volume 128, 2017, pp 130-149.
12. C. Combi, B. Oliboni and F. Zerbato, "Integrated Exploration of Data-Intensive Business Processes," in IEEE Transactions on Services Computing, vol. 16, no. 1, pp. 383-397, 1 Jan.Feb. 2023

13. David Chapela-Campa, Manuel Mucientes, Manuel Lama, Understanding complex process models by abstracting infrequent behavior, Future Generation Computer Systems, Volume 113, 2020, pp 428-440

14. Brodie, M. L. On the development of data models. In On Conceptual Modeling, Perspectives from Artificial Intelligence, Databases, and Programming languages, M. L. Brodie, J. Mylopouious, and J. W. Schmidt, Eds. Springer-Verlag, New York, 1984, pp. 19-48.

15. Silverstone, L. The data Model Resource Book, Vol. 3: Universal Patterns for Data Modeling . Len Silverston.Wiley Computer Publishing, 2009. 648 p.

16. Ambler, S. Refactoring Databases: Evolutionary Database Design. Scott W. Ambler, Premodkumar J. Sadalage .Addison-Wesley, 2006. 384 p.

17. Fowler, M. Patterns of Enterprise Application Architecture. Martin Fowler Addison-Weatley, 2003. 736 p.

18. Simsion, G.C., Witt, G.C. Data Modeling Essentials, Third Edition .Graeme C. Simsion, Graham C. Witt.Morgan Kaufmann Publishers, 2005.560 p.

Suzikova Olena G.

Candidate of Psychological Sciences. Department of Applied Mathematics
V. N. Karazin Kharkiv National University, 4 Svobody Sq., Kharkiv, 61022, Ukraine.
e-mail: osuzikova@karazin.ua
https://orcid.org/0009-0002-1305-1256

The role of semantic analysis in overcoming the limitations of the relational data model

Relevance. The article is devoted to defining the role of semantic analysis in overcoming the limitations of the traditional relational data model. The need for semantic data analysis arises from the demand for more expressive and capacious conceptual data models. The problem that has not yet been fully resolved is that classical relational models do not directly support data semantics - relationships, data abstraction, inheritance, polymorphism, encapsulation, complex objects, and dynamic properties of objects. Semantics refers to the use of certain constructs and methods to express features of the application environment that remain outside the traditional relational model. Semantic analysis allows to define more complex relationships and interactions between entities, including classifications, aggregations (complex objects that contain other objects), associations (links between entities), etc. The purpose of using semantics components is to increase the level of abstraction in design, which makes the model more versatile. Abstraction, in turn, is the creation of generalized models or classes that represent entities in the database. Semantic analysis helps determine how data will be stored and optimized in the database. It includes the selection of data types, indexing, normalization/denormalization, and other aspects of database design. **Objective.** The study considers the issues of semantic data analysis in the construction of relational models and relational databases. **Research methods.** Along with the theoretical analysis of the problem, the article collects and analyzes the actual material - data structures from existing projects and cloud applications (best practices) - using semantic analysis, and draws conclusions about the capabilities of existing relational models and relational databases to support semantics**. Results.** We described the elements of semantics, analyzed their role in building the model, and identified a number of semantic patterns as a means of overcoming the limitations of the relational model.

Keywords: *semantic data analysis, relational model, conceptual data models, database systems, data models, logical database design, semantic analysis of the subject area.*

Надійшла у першій редакції 10.05.2024, в останній - 15.06.2024.
The first version has been received on 10.05.2024, the final version - on 15.06.2024.