

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод автоматизації проектування
сховища даних

(тема)

Виконав:

студент II курсу, групи СПзм-18-2
Любченко Д.С.
(прізвище, ініціали)

Спеціальність

123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Смеляков С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ

(підпис)

Коваленко А.А.

(прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет _____ навчально-науковий центр заочної форми навчання

Кафедра _____ електронних обчислювальних машин

Рівень вищої освіти _____ другий (магістерський)

Спеціальність _____ 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми _____ освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма _____ Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

“ ____ ” _____ 20__ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____ Любченку Дмитру Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Метод автоматизації проектування сховища даних

затверджена наказом по університету від “ 30 ” березня 2020 р. № 43 Стз

2. Термін подання студентом роботи до екзаменаційної комісії _____ 18 травня 2020 р.

3. Вхідні дані до роботи _____ 1) алгоритми побудови тензорних моделей;
_____ 2) дані реляційних та NoSQL баз даних;

4. Перелік питань, що потрібно опрацювати в роботі _____

- _____ 1) Аналіз існуючих моделей та методологій проектування сховищ даних;
- _____ 2) Аналіз проблем зв'язаних із проектуванням сховищ даних;
- _____ 3) Розробка методу автоматизації сховищ даних;
- _____ 4) Експериментальні дослідження;
- _____ 5) Висновки;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) слайди презентацій.

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури з теми	30.03.20-07.04.20	
2	Розгляд методів та методологій проектування сховищ даних	08.04.20-13.04.20	
3	Розгляд тензорної алгебри і тензорних методів	14.04.20-20.04.20	
4	Аналіз проблем автоматизації проектування сховищ даних	21.04.20-23.04.20	
5	Розробка методу автоматизації	24.04.20-30.04.20	
6	Оформлення матеріалів атестаційної роботи	1.05.20-13.05.20	
7	Подання атестаційної роботи керівникові та її попередній захист	13.05.20-14.05.20	
8	Подання атестаційної роботи на рецензування	14.05.20-15.05.20	

Дата видачі завдання 30 березня 2020 р.

Студент _____
(підпис)

Керівник роботи _____
(підпис)

проф. Смеляков С.В.
(посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 69 с., 13 рис., 4 табл., 1 дод., 19 джерел.

БАЗИ ДАНИХ, СХОВИЩА ДАНИХ, ПРОЕКТУВАННЯ, АВТОМАТИЗАЦІЯ, ТЕНЗОРНІ МЕТОДИ, ТЕНЗОРНА АЛГЕБРА.

Метою атестаційної роботи є розробка програмного забезпечення, яке дозволяє користувачам будувати та використовувати сховища даних на основі реляційних та NoSQL баз даних.

Об'єктом дослідження є покращення методів автоматизації проектування сховищ даних.

Теоретичною значимістю цього дослідження є розробка оптимального методу автоматизації проектування сховищ даних.

У ході виконання атестаційної роботи досліджені існуючі моделі та методолгії проектування, визначені їх недоліки та переваги та експериментальна розробка додатку автоматизації сховищ даних.

ABSTRACT

Master's thesis: 69 pages, 13 figures, 4 tables, 1 appendices, 19 sources.

DATABASE, DATA WAREHOUSE, DESIGN, AUTOMATION, TENSOR METHODS, TENSOR ALGEBRA.

The purpose of the certification work is to develop software that allows users to build and use data warehouses based on relational and NoSQL databases.

The object of research is to improve the methods of automation of data warehouse design.

The theoretical significance of this study is the development of an optimal method for automating the design of data warehouses.

During the attestation work the existing models and design methodologies are investigated, their shortcomings and advantages are determined and the experimental development of the data warehouse automation application is experimentally developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	7
ВСТУП	8
1 ПРОБЛЕМИ ПРОЕКТУВАННЯ СХОВИЩ ДАНИХ.....	10
1.1 Роль сховищ даних в системах прийняття рішень	10
1.2 Моделі сховищ даних	11
1.3 Методологія проектування сховищ даних	24
1.4 Аналіз проблем сховищ даних.....	27
1.5 Висновки до розділу один.....	32
2 МЕТЕМАТИЧНА МОДЕЛЬ СТРУКТУРИ СХОВИЩА ДАНИХ	33
2.1 Тензорні методи	33
2.2 Тензорна модель реляційного сховища даних.....	37
2.3 Тензорна модель запитів до реляційної бази даних	40
2.4 Тензорна модель реляційного сховища даних.....	44
2.5 Висновки до розділу два.....	47
3 АВТОМАТИЗАЦІЯ СХОВИЩ ДАНИХ.....	48
3.1 Проблеми автоматизації сховищ даних.....	48
3.2 Метод автоматизації проектування сховищ даних.....	51
3.3 Висновки до розділу три	54
4 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА	55
4.1 Результат експерименту	55
4.2 Висновки до розділу чотири	59
ВИСНОВКИ.....	60
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	61
ДОДАТОК А Графічний матеріал атестаційної роботи	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

РБД – реляційна база даних

РОД – різні оперативні джерела

РСД – реляційні сховища даних

СД – сховище даних

СППР – системи підтримки прийняття рішень

ВІ – позначення комп’ютерних методів та інструментів для бизнес-аналізу
(англ., business intelligence)

ETL – процедура вилучення запису з джерел даних і підготовка їх до процесу перетворення (англ. Extract, Transform, Load). При розробці процедури вилучення даних в першу чергу необхідно визначити частоту вивантаження даних з OLTP-систем або окремих джерел

OLAP – інтерактивна система що дозволяє переглядати різні підсумки по багатовимірних даних (англ., online analytical processing)

ВСТУП

Автоматизація сховищ даних використовує технологію для підвищення ефективності та підвищити ефективність процесів зберігання даних. Дані автоматизація складів - це більше, ніж просто автоматизація ETL розвиток. Він автоматизує весь життєвий цикл зберігання даних від планування, аналізу та проектування через розробку та поширюючись на операції, технічне обслуговування та управління змінами.

Прийняття автоматизації сховищ даних змінює наш погляд про створення сховищ даних. Широко прийнята практика у світі – обширний передовий аналіз, дизайн та моделювання залишилися позаду менталітет змінюється від «поправити це в перший раз» до «розвиватися швидко і розвивайтесь часто». Такий підхід акуратно вписується у спритний практики розробки, але автоматизація сховищ даних не працює вимагають, щоб ви просувалися. Ви можете досягти значної швидкості, якості та економія коштів без повного використання та впровадження спритна методологія.

Автоматизація зберігання даних має багато таких же переваг, що і у виробництві:

- підвищення продуктивності та швидкості виробництва;
- зменшення ручного зусилля;
- покращена якість та послідовність;
- краще управління та можливості оптимізації процесів;

Виробнича паралель справедлива при складанні даних склад; ми можемо вважати це фабрикою інформації. Однак, зберігання даних складніше, ніж виготовлення продукції. Виробнича продукція зазвичай постачається споживачеві робота виконана. Склади даних повинні підтримуватися через тривалий час життєвий цикл, коли змінюються вихідні дані, вимоги бізнесу, і

основні технології є постійними міркуваннями. Автоматизація допомагає здійснити правильні зміни правильними способами і як швидко, як вони потрібні.

Шаблони дизайну, стандарти, метадані та повторне використання є основою автоматизації сховищ даних. Складування даних пропонує можливість застосовувати багато моделей дизайну, включаючи:

- архітектурні зразки, такі як ступиці, шини та гібриди архітектури;
- структури структури даних, включаючи нормалізовані, денормовані, і багатовимірність;
- шаблони управління даними, такі як для управління ключами і часові відхилення;
- шаблони інтеграції даних, включаючи ETL, ELT, віртуалізацію та федерація;

Шаблони дизайну можуть поєднуватися з архітектурою, дизайном та стандарти впровадження та передовий досвід для створення багаторазових даних складові складові. Шаблони, стандарти та компоненти захоплюються та описуються як метадані. Потім описується кожне використання компонентів та взаємозв'язки між ними більш детальні та конкретні метадані. Хороші дані. Платформа автоматизації складських приміщень дозволяє легко переступити прогалину багаторазове використання для повторного використання.

1 ПРОБЛЕМИ ПРОЕКТУВАННЯ СХОВИЩ ДАНИХ

1.1 Роль сховищ даних в системах прийняття рішень

Сховище даних (СД) є інформаційною основою оперативно-аналітичних систем підтримки прийняття рішень (СППР) і, отже, від нього залежить ефективність цих систем. У сховищі відбувається інтеграція даних з різних оперативних джерел (РОД). Потім виконується обробка даних, далі ці дані готові для використання аналітичними додатками. Слід мати на увазі, що дані, які обробляються в СД та наступному аналізі, як правило, являються чисельними показниками, тобто характеристики розглянутих процесів (об'єм продуктивності, середній запас товарів на складах і т.п.) Результат аналітичної обробки послідовних цих числових значень являється основою для прийняття управлінських рішень.

СД – предметно-орієнтований, інтегрований, підтримуючий хронологію набору даних, організований для цілей прийняття рішень.

Ральф Кімболл сформував такі основні потреби до СД [1]:

- підтримка високої швидкості отримання даних із сховища;
- спромога отримати та порівняти так звані зрізи даних;
- повнота та цілісність збережених даних;
- підтримка якісного процесу поповнення даних;

СД дозволяють розвантажити оперативні бази даних. Таким чином, надається можливість ефективніше і швидше отримувати необхідну інформацію для перевірки висунутих гіпотез. Перевірка здійснюється на підставі інформації про аналізовану предметну область. Як правило, найбільш зручним способом подання такої інформації для людини є залежність між параметрами об'єктів, що візуалізується у вигляді графіків, діаграм, рівнянь трендів, чисельної оцінки

динаміки і т.п. У процесі аналізу даних і пошуку рішень часто виникає необхідність в побудові залежностей між різними параметрами. Крім того, число таких параметрів може варіюватися в широких межах. Традиційні засоби аналізу, які оперують даними, які представлені у вигляді таблиць реляційної БД, не можуть повною мірою задовольняти таким вимогам.

Технологія комплексного багатовимірного аналізу отримала назву OLAP. OLAP - це ключовий компонент організації сховищ даних. Концепція OLAP в була описана в 1993 році Едгаром Коддом, відомим дослідником БД і автором реляційної моделі даних. За Коддом [2] дає наступне визначення.

OLAP – технологія оперативної аналітичної обробки даних, яка використовує методи і засоби для збору, зберігання і аналізу багатовимірних даних в цілях підтримки процесу прийняття рішень.

1.2 Моделі сховищ даних

Моделі даних, включаючи СД, можна поділити на чотири типи:

Ієрархічна модель даних - логічна модель даних в вигляді дерева, що представляє собою сукупність елементів, розташованих в порядку їх підпорядкування від загального до конкретного. В ієрархічних моделях основна структура представлення даних має форму дерева. На найвищому (першому) рівні ієрархії знаходиться тільки одна вершина, яка називається коренем дерева.

Мережева модель даних - логічна модель даних, що є розширенням ієрархічного підходу, сувора математична теорія, що описує структурний аспект, аспект цілісності і аспект обробки даних в мережевих базах даних.

Реляційна модель даних - логічна модель даних, прикладна теорія побудови баз даних, яка є додатком до завдань обробки даних таких розділів математики, як теорія множин і логіка першого порядку. На реляційній моделі даних будуються реляційні бази даних.

Реляційна модель даних включає наступні компоненти: Структурний аспект (складова) - дані в базі даних являють собою набір відносин.

Різниця між ієрархічною моделлю даних і мережевий полягає в тому, що в ієрархічних структурах запис-нащадок повинна мати в точності одного предка, а в мережевій структурі даних у нащадка може бути будь-яке число предків.

OLAP - технологія обробки даних, яка полягає в підготовці сумарною (агрегированной) інформації на основі великих масивів даних, структурованих по багатовимірному принципу. Реалізації технології OLAP є компонентами програмних рішень класу Business Intelligence.

В основі OLAP лежить поняття гіперкуба, або багатомірного куба даних, в осередках якого зберігаються аналізовані дані. Факт - це числова величина яка розташовується в осередках гіперкуба. Один OLAP-куб може мати одну або декілька показниками.

Вимірювання (dimension) - це безліч об'єктів одного або декількох типів, організованих у вигляді ієрархічної структури і забезпечують інформаційний контекст числового показника. Вимірювання прийнято візуалізувати у вигляді ребра багатовимірного куба.

Факт - це числова величина яка розташовується в осередках гіперкуба. Один OLAP-куб може мати одну або декілька показниками. Вимірювання - це безліч об'єктів одного або декількох типів, організованих у вигляді ієрархічної структури і забезпечують інформаційний контекст числового показника.

Вимірювання прийнято візуалізувати у вигляді ребра багатовимірного куба.

Об'єкти, сукупність яких і утворює вимір, називаються членами вимірювань. Члени вимірювань візуалізують як точки або долі, що відкладаються на осях гіперкуба.

Осередок - атомарна структура куба, відповідна повного набору конкретний значень вимірів.

Ієрархія - групування об'єктів одного виміру в об'єкти більш високого рівня. Наприклад - день-місяць-рік. ієрархії в вимірах необхідні для можливості агрегації і деталізації значень показників відповідно до їх ієрархічній структурі.

Ієрархія цілком ґрунтується на одному вимірі і формується з рівнів.

OLAP-функціональність може бути реалізована різними способами, як найпростішими, такими як аналіз даних в офісних додатках, так і більш складними - розподіленими аналітичними системами, заснованими на серверних продуктах.

Архітектура інформаційної системи - концепція, яка визначає модель, структуру, виконувані функції і взаємозв'язок компонентів інформаційної системи.

Операції над даними. При виконанні операції зріз формується підмножина гіперкуба, в якому значення одного або більше вимірювань фіксоване (наприклад, значення параметрів для фіксованого вимірювання Бригада). Операція обертання змінює порядок подання вимірювань, забезпечуючи уявлення куба в більш зручній для сприйняття формі.

Консолідація - операція переходу від детального представлення даних до агрегованого.

Агрегування - метод узагальнення моделей. операцією, протилежної декомпозиції, є агрегування - об'єднання частин в ціле. Операція декомпозиції застосовується на етапі аналізу системи.

Типовими архітектурою для систем складування даних прийнято вважати такі:

- системи з глобальним ХД;
- системи з незалежними кіосками даних;
- системи з інтегрованими кіосками даних;
- системи, розроблені на основі комбінації з перерахованих вище архітектур.

Глобальне сховище даних (Global data warehouse), або сховище даних масштабу організації, - це таке ХД, в якому будуть підтримуватися всі дані організації або велика їх частина. Це найповніше інтегроване ХД з високою ступенем інтенсивності доступу до консолідованих даних і використанням його всіма підрозділами організації або керівництвом організації в рамках основних напрямків діяльності організації.

Централізоване глобальне ХД характерно для організацій, розташованих територіально в одній будівлі. воно підтримується відділом інформаційних систем організації. Розподілене глобальне ХД також може бути використано в рамках організації в цілому. Воно фізично розподіляється по підрозділах організації і також підтримується відділом інформаційних систем.

Гібридні сховища даних. Багатовимірні і реляційні моделі сховищ даних мають свої переваги і недоліки. Наприклад, багатовимірні моделі дозволяють швидше отримати відповідь на запит, але не дають можливості ефективно управляти такими ж великими обсягами даних, як реляційні моделі. логічно було б використовувати таку модель ХД, яка представляла б собою комбінацію реляційної і багатовимірної моделей і дозволяла б поєднувати високу продуктивність, характерну для багатовимірної моделі, і можливість зберігати скільки завгодно великі масиви даних, властиву реляційній моделі.

Багатовимірні сховища зазвичай містять агрегатні дані (Наприклад, суми, середні значення, кількість значень) для різних вибірок. Найчастіше такі агрегатні функції утворюють багатовимірний набір даних, званий кубом, осі якого (звані вимірами) містять параметри, а осередки - залежать від них агрегатні дані (іноді їх називають заходами).

Концепція в загальному сенсі представляє деяку систему поглядів на процес або явище. Складовими частинами концепції є сукупність принципів і методологія. під методологією розуміється сукупність методів вирішення проблеми. принцип - правила, якими слід керуватися в діяльності. Часто

принципи формуються у вигляді обмежень і вимог, в Зокрема, вимог до баз даних.

В основі концептуальної моделі сховища даних лежить багатовимірна модель даних. Основними поняттями багатовимірної моделі даних є: куб (гіперкуб) даних, факти, показники (заходи), вимірювання, ієрархії, агрегати, зріз.

Застосування реляційної моделі при створенні ХД в ряді випадків дозволяє отримати переваги, особливо в частині ефективності роботи з великими масивами даних і використання пам'яті комп'ютера. На основі реляційних сховищ даних (РСД) будуються ROLAP-системи. Дані поділяються на вимірювання і факти. Вимірювання - це категоріальні атрибути, найменування і властивості об'єктів, що беруть участь в деякому бізнес-процесі.

Під архітектурою ХД розуміють сукупність програмноапаратних компонент, сукупність технологічних і організаційних рішень, що вживаються для створення, розробки і функціонування ХД, тобто вибір апаратного та програмного забезпечення, вибір способів взаємодії програмно-апаратних компонент, вибір способу вирішення проектної задачі по розробці і створенню ХД. Схема типу зірки (Star Schema) - схема реляційної бази даних, що служить для підтримки багатовимірного представлення містяться в ній даних.

Особливості ROLAP-схеми типу «зірка»:

- одна таблиця фактів (fact table), яка сильно денормалізована. Є центральною в схемі, може складатися з мільйонів рядків і містить підсумовувані або фактичні дані, з допомогою яких можна відповісти на різні питання.

- кілька денормалізованих таблиць вимірів (dimensional table). Мають меншу кількість рядків, ніж таблиці фактів, і містять описову інформацію. Ці таблиці дозволяють користувачеві швидко переходити від таблиці фактів до додаткової інформації.

- таблиця фактів і таблиці розмірності пов'язані ідентифікують зв'язками,

при цьому первинні ключі таблиці розмірності мігрують в таблицю фактів як зовнішніх ключів. Первинний ключ таблиці факту цілком складається з первинних ключів всіх таблиць розмірності.

- агреговані дані зберігаються спільно з вихідними.

Схема типу сніжинки (Snowflake Schema) - схема реляційної бази даних, що служить для підтримки багатовимірного уявлення містяться в ній даних, є різновидом схеми типу «зірка» (Star Schema).

Особливості ROLAP-схеми типу «сніжинка»:

- одна таблиця фактів (fact table), яка сильно денормалізована. Є центральною в схемі, може складатися з мільйонів рядків і містити підсумовувані або фактичні дані, з допомогою яких можна відповісти на різні питання.

- кілька таблиць вимірів (dimensional table), які нормалізовані на відміну від схеми «зірка». мають менше кількість рядків, ніж таблиці фактів, і містять описову інформацію. Ці таблиці дозволяють користувачеві швидко переходити від таблиці фактів до додаткової інформації. Первинні ключі в них складаються з єдиного атрибута (відповідають єдиному елементу вимірювання).

- таблиця фактів і таблиці розмірності пов'язані ідентифікують зв'язками, при цьому первинні ключі таблиці розмірності мігрують в таблицю фактів як зовнішніх ключів. Первинний ключ таблиці факту цілком складається з первинних ключів всіх таблиць розмірності.

- у схемі «сніжинка» агреговані дані можуть зберігатися окремо від вихідних.

Оскільки багатовимірною моделлю даних призначена для надання ВІ-інформації, всі показники, вимірювання, ієрархії повинні мати назви, доступні для розуміння особам, які приймають рішення в організації. Крім того, модель передбачає можливість супроводжувати назви об'єктів розгорнутими описами. Це дозволяє відповідальному за прийняття рішень, упевнитися, що він працює з необхідною аналітичною інформацією.

Факти однозначно визначаються комбінацією значень вимірів; факт існує тільки тоді, коли осередок для конкретної комбінації значень не порожня. Таблиця фактів є основною таблицею сховища даних. Зазвичай, вона містить відомості про об'єкти, події і т.п. Тобто, сукупність даних які будуть надалі аналізуватися. Також говорять про чотирьох найбільш часто зустрічаються типах фактів – транзакційні факти; факти, пов'язані з «миттєвими знімками»; факти, пов'язані з елементами документа; факти про події або стан об'єкта. Таблиця фактів, зазвичай, містить в собі унікальний ключ, який об'єднує усі первинні ключі таблиць вимірів.

На рисунку 1.1 наведено приклад спрощеного гіперкуба, розробити конструкцію такого для аналізу надходжень до бюджету підприємства. «Надходження до бюджету» – єдиний показник даного гіперкуба. Три виміри гіперкуба: «Час», «Доходи бюджету», «ОКАТО» – дозволяють аналізувати факти надходжень до бюджету в контексті:

- певного часу (конкретний місяць певного року, конкретний квартал певного року);
- виду доходу бюджету (ПДВ, податок на прибуток, ПДФО, податкові доходи);
- адміністративно-територіальної приналежності кореспондентів (перший район, другий район, третій район).

В осередках гіперкуба «зберігаються» фактичні значення показника (факти). Факти можуть бути адитивними, напівадитивними і неадитивними. У наведеному (рисунок 1.1) прикладі представлені адитивні факти, так як для показника «Надходження до бюджету» може бути виконано підсумовування по всіх вимірах куба даних.

У гіперкубі (рисунок 1.1) відображені як деталізовані, так і агреговані значення показника «Надходження до бюджету». Деталізовані значення – це факти надходжень до бюджету за конкретним видом доходу бюджету за

конкретний місяць від кореспондентів певного району. Прикладом детального значення може служити факт надходжень до бюджету з податку на прибуток в лютому 2013 р від кореспондентів першого району – 25 тис. грн. Прикладом агрегованого значення в даному гіперкубі можуть бути всі податкові надходження в бюджет в лютому 2013 р від кореспондентів першого району – 75 тис. грн.

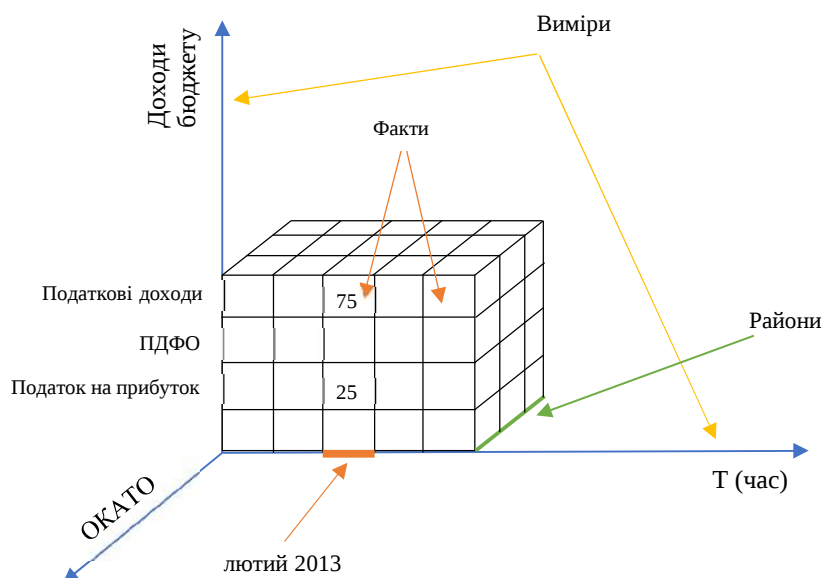


Рисунок 1.1 – Приклад куба даних для аналізу

Класична модель «зірка», основана на реляційній алгебрі і включає в себе одиничну таблицю фактів з множини таблиць вимірів з'єднаних з нею. Таблиця фактів в даній моделі складається з набору атрибутів вимірів, які формують первинний ключ і декілька чисельних атрибутів мір. Таблиці вимірів містять мінімум одне описуюче поле – ключове поле (сурогатний ключ). Нерідко таблиця вимірів може містити поля, які вказують на додаткові атрибути, які знаходяться у

вхідній оперативній базі, або атрибути, які відповідають за групування цих даних. Кожна таблиця вимірів повинна знаходитися по відношенню «один до багатьох» з таблицею фактів.

Модель «сніжинка» відрізняється від «зірки» тим, що має хоча б одну таблицю вимірів, яка посилається на таблицю, яка знаходиться по відношенню до неї «один до багатьох». Дані структури являються історично першою змогою Біллом Інманом привести множину взаємопов'язаних реляційних таблиць до денормалізованої структури.

Логічні моделі сховищ даних за допомогою схеми «Зірка» або її варіанту – схеми «Сніжинка».

Одна схема «Зірка», як і гіперкуб, охоплює, наприклад, окрему сферу бізнесу підприємства. Так, схема «Зірка» (рисунк. 1.2) охоплює сферу продажів. У відповідній концептуальній моделі «Сума» – представляє єдиний показник гіперкуба, а «Клієнт», «Продавець», «Час», «Товар» – його чотири виміри. Схема «Зірка» дає досить високу швидкість виконання запитів через те, що дані денормалізовані.

Неможливо створити універсальну денормалізовану структуру даних, що забезпечує високу продуктивність при виконанні будь-якого аналітичного запиту. Тому схема «Зірка» будується так, щоб забезпечити найвищу продуктивність при виконанні одного найважливішого запиту або для групи схожих запитів.

Так, сховище, розроблене на основі схеми (рисунк 1.2), дозволить оперативної відповісти на питання, пов'язані продажами в компанії.

Всі факти, необхідні для відповіді на запити, поміщаються в таблицю фактів. Так, (рисунк 1.2) в таблиці фактів «Продаж» всі факти щодо суми продажів розміщуються в колонці «Сума».

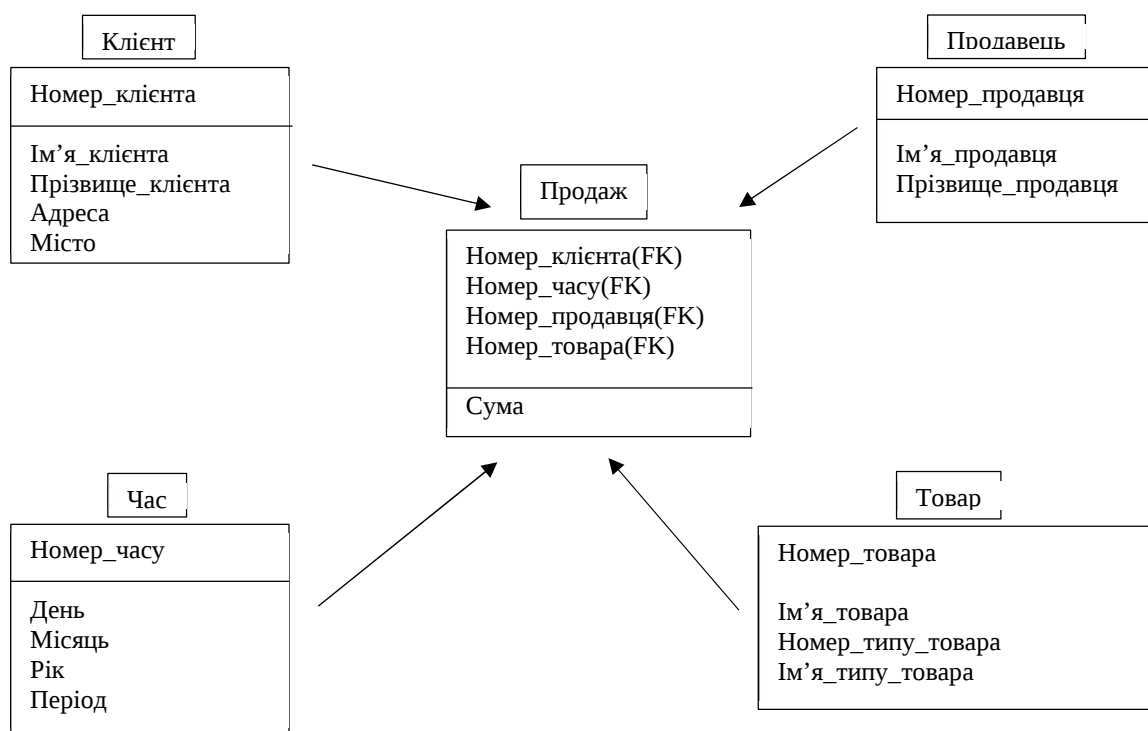


Рисунок 1.2 – Приклад схеми «Зірка»

Таблиця фактів може складатися з мільйонів рядків і містити агреговані або деталізовані дані, які можуть допомогти відповісти на необхідні питання. Вона містить дані, які зберігалися б у багатьох таблицях нормалізованих реляційних баз даних.

Для аналізу корисні таблиці фактів, що містять детальні дані. Наприклад, краще взяти за основу факти продажу окремого товару окремого клієнта, а не суми продажів для різних країн. Агреговані дані будуть обчислені.

Таблиця фактів є центральною таблицею в схемі «Зірка».

Таблицю фактів оточують менші таблиці, звані таблицями розмірності або вимірювань. Таблиці вимірювань містять незмінні або рідко змінювані дані типу довідника. У розглянутому прикладі (рисунок 1.2) передбачається, що записи в таблицях «Клієнт», «Продавець», «Час», «Товар» не змінюються або змінюються рідко.

Таблиці вимірювань з'єднані з таблицею фактів у вигляді зірки радіальними зв'язками. У цих зв'язках таблиці розмірності є батьківськими, таблиця факту – дочірньою. Таблиця фактів і таблиці розмірності пов'язані ідентифікують зв'язками, при цьому первинні ключі таблиці розмірності мігрують в таблицю фактів як зовнішніх ключів (рисунок 1.2).

Таблиця фактів, як правило, містить унікальний складовою ключ, який об'єднує первинні ключі таблиць вимірів (рисунок 1.2). Найчастіше це цілочисельні значення або значення типу «дата/час».

У схемі «Зірка» вимірювання денормалізовані. Так, на схемі (рисунок 1.2) в таблиці вимірювання «Час» неключових стовпці «День», «Місяць», «Рік», «Період» функціонально залежні; в таблиці вимірювання «Товар» функціональна залежність простежується між неключових стовпцями «Найменування товару» і «Найменування типу товару». Внаслідок цього, з одного боку, в сховище спостерігається дублювання даних, а з іншого – прискорення часу виконання запитів і спрощення сприйняття схеми. Тому в довгострокових рекомендується використовувати саме схему «Зірка».

У короткострокових пілотних проектах для скорочення часу розробки сховища іноді використовують схему «Сніжинка» з її нормалізованими таблицями вимірювань, так як вона ближче по структурі до джерел – нормалізованим баз даних OLTP-систем.

На рис. 1.3. представлений схема «Сніжинка», на основі якої планується виконати пілотний проект по розробці сховища даних для аналізу продажів компанії. В даній схемі таблиці «Клієнт», «Продавець», «Час», «Товар» були запозичені з баз даних OLTP-систем, і залишилися нормалізованими (див. Додаткові таблиці вимірювань «Регіон», «Період», «Тип товару»). Це дозволить скоротити терміни проекту. Однак швидкість виконання аналітичних запитів до такого сховища буде нижче, ніж до сховища, побудованому на основі схеми «Зірка» (рисунок 1.3).

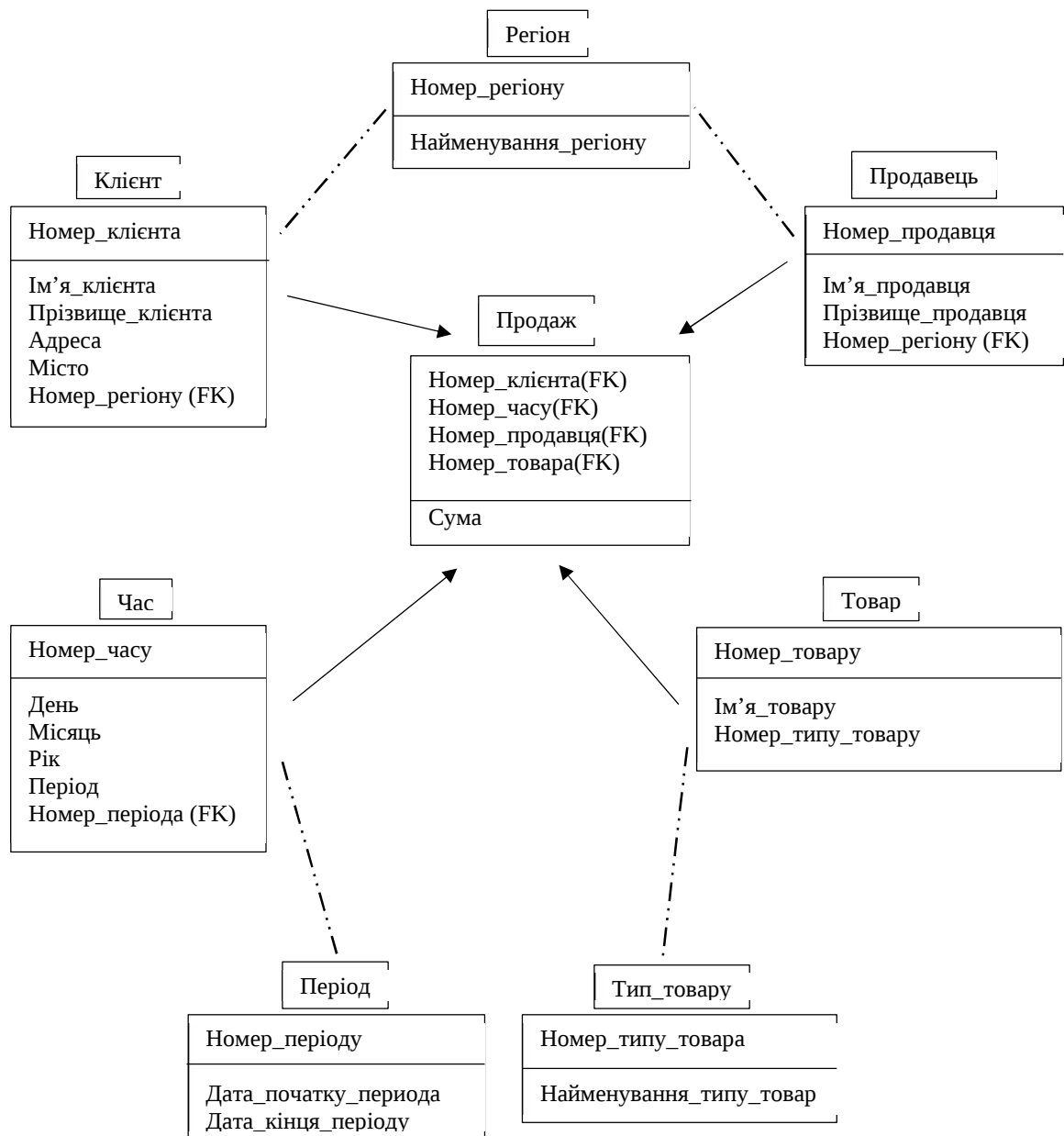


Рисунок 1.3 – Приклад схеми «Сніжинка»

Додаткові таблиці вимірювань в схемі «Сніжинка», зазвичай відповідають верхнім рівням ієрархії вимірювання і знаходяться в співвідношенні «один до багатьох» з головною таблицею вимірювань, що відповідає нижньому рівню ієрархії. Ці додаткові таблиці іноді називають консольними. На схемі,

представленої на рис. 1.3, таблиці «Регіон», «Період», «Тип товару» є консольними.

Консольні таблиці можуть бути пов'язані тільки з таблицями вимірювань, причому консольна таблиця в зв'язку з цим батьківська, а таблиця вимірювань - дочірня. Зв'язок може бути ідентифікуючою або неідентифікуючою. Консольна таблиця не може бути пов'язана з таблицею факту.

Вона використовується для нормалізації даних в таблицях вимірів.

Бувають ситуації, коли в схемі «Зірка» денормалізована таблиця вимірювань виявляється занадто великий (мається на увазі кількість колонок), при цьому до частини колонок запити виконуються частіше, ніж до інших колонок.

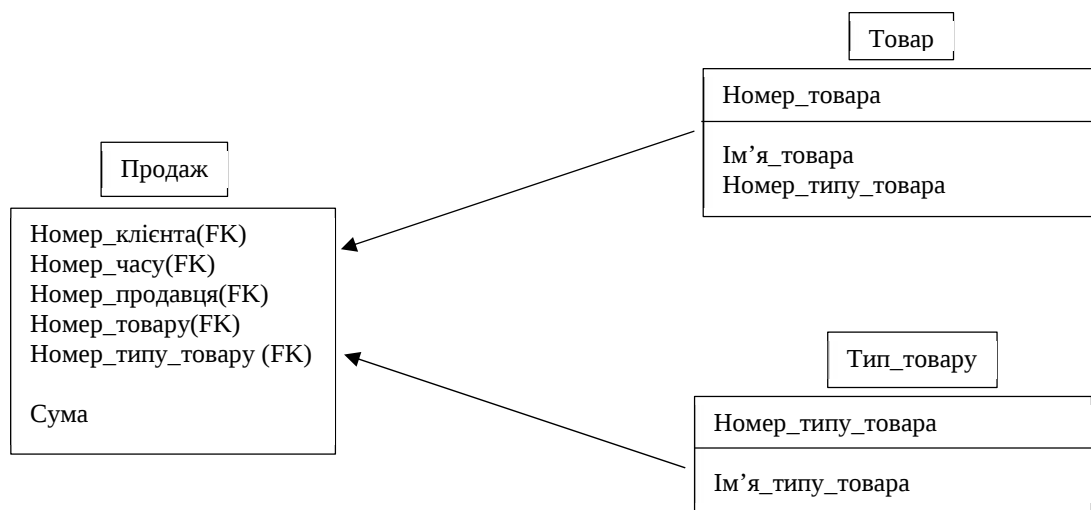


Рисунок 1.4 – Таблиця вимірів розбита на дві зв'язні таблиці

Наприклад, в результаті аналізу статистики по видах запитів до сховища було виявлено, що кількість запитів про продажі з деталізацією по найменуванню товару в десять разів менше, ніж кількість запитів про продажі за типами товарів.

У таких випадках змінюють схему «Зірка»: таблицю вимірювання розбивають на дві окремі таблиці, зв'язавши їх неідентифікуючим зв'язком (рисунки 1.4)

Сховище даних підприємства містить дані з різних сфер діяльності і складається з набору зв'язкових схем «Зірка» або «Сніжинка». Узгодження різних схем проводиться через загальні таблиці вимірювань.

1.3 Методологія проектування сховищ даних

Існуючі методи до розробки СД з точки зору парадигми ґрунтуються на двох підходах [5-7]. Перший, запропонований Р. Кімболом, ставить на чільне місце орієнтацію на аналітичні вимоги до системи (підхід demand-driven) [7], другий орієнтується на джерела даних (підхід Б. Інмона, або Supply-driven) [8].

Підхід Білла Інмона орієнтований на суб'єкти або підрозділи підприємства сховища даних, які призначені для задоволення потреб в даних і звітності цільової групи бізнес-користувачів [1].

У цій моделі є кілька переваг. По-перше, всі корпоративні дані повністю документовані. Цей процес дає організації повне уявлення про свої процеси, продуктах / послугах, клієнтів, постачальників і т.п. Така документація неоціненна для організації, оскільки в більшості випадків до цього моменту кожна система запускала ізолювано. Тепер організація дійсно може визначати різні процеси, продукти або партії, з якими вони взаємодіють на постійній основі.

По-друге, дані ефективно зберігаються в 3-й нормальній формі в одному репозиторії. Ця методологія зберігання полегшує пошук і зберігання даних з транзакційних систем, вже визначених в 3-й нормальній формі.

Нарешті, дані легко доступні для вилучення в вітрини даних для бізнес-користувачів. Тепер, коли дані повністю визначені і ефективно зберігаються, команда по роботі зі сховищем даних. Може створювати дані, що відносяться до бізнес-одиницям. Ці вітрини даних створюються, щоб бізнес-одиниця могла

швидко і ефективно відповідати на їхні запитання. Вона також забезпечить користувачеві деталізовані дані, що підтримують вітрину даних, а також походження даних. Ця інформація часто має вирішальне значення при схваленні і функціональному використанні вітрини даних бізнес-користувачем.

Проте, така структура може створити деякі проблеми організації. По-перше, завдання документування та визначення повного сховища для всієї організації досить трудомістка. Така методологія проектування – довгий, трудомісткий процес, який, не дивлячись на те, що є неоцінним, вимагає, щоб ІТ-команда зі зберігання даних тісно співпрацювала з бізнес-користувачами, щоб забезпечити облік і зберігання достовірних даних в правильну структуру. Тому пройде багато часу між стартом проекту і формуванням файлом даних. Цей «розрив в поставках» став причиною «смерті» багатьох проектів зі зберігання даних, оскільки в залежності від розміру організації і досвіду команди сховища даних є початковим етапом виявлення і документування може ніколи не завершитися до скасування проекту.

Друга проблема – відсутність гнучкості, що надається цією моделлю. Якщо ви додасте нову бізнес-одиницю, новий додаток або сильно відрізняється продукт або послугу, вам доведеться змінити існуючу модель даних, щоб точно документувати і визначати новий стан бізнесу при збереженні предметно-орієнтованого, незмінного, хронологічного і інтегрованого аспектів наявних в сховищі даних. Це також часто ставить під сумнів цінність сховища даних.

Методологія Ральфа Кімбола носить скоріше тактичний характер і є антитезою методології Інмона. Визначення сховища даних по Кімболу – це «копія транзакційних даних, спеціально структурованих для запиту і аналізу». Він вважає, що ви повинні почати з тактичного рівня, зосередившись спочатку на пакеті даних, тим самим забезпечуючи безпосередню цінність для бізнес-користувачів. Сховище даних Кімбола покликана просто використовувати колекцію даних в цілому. Підхід Кімбола зачіпає тільки дані, необхідні для вітрин даних [1].

Вітрини даних Кімбола складаються з вихідних даних, перетворених з 3-й нормальної форми в багатовимірну модель. Ця модель складається з фактів і вимірювань. Факти обчислюють заходи по об'єктах в певний момент часу. Вимірювання – це контейнери для уточнюючих елементів об'єктів, за якими згруповані заходи [1].

Основна перевага підходу Кімбола і використання багатовимірного моделювання – це швидкість, з якою бізнес-користувач може отримати вигоду з масиву даних і гнучкість, яку пропонує це моделювання. Вітрини даних зазвичай можуть бути визначені, спроектовані і поставлені менш ніж за 120 днів і в більшості випадків менш ніж за 90 днів після доступності даних. Потім ця модель також дозволяє легко розгорнути факти або вимірювання, щоб додати нові заходи або додаткову інформацію, що описує додається об'єкт. Нарешті, цей метод моделювання вимагає попередньої обробки і зберігання даних таким чином, що агрегування даних фактів може бути легко розрізано і розбите по розмірним стовпцями бізнес-користувачами, практично без допомоги ІТ. Таке поєднання швидкості, гнучкості, масштабованості і прозорості необхідно в швидко мінливого бізнес-середовищі.

При виборі методології слід мати на увазі, що неналежний облік аналітичних потреб користувачів при побудові сховища призводить в кінцевому підсумку до непродуктивності реалізованого рішення, оскільки менеджери не отримують їх цікавить. Однак, по-перше, джерела даних апріорі містять прихований потенціал для аналізу, який може бути виявлений тільки при їх включенні в процедуру розробки багатовимірних схем, по-друге, ігнорування джерел може спричинити за собою створення «порожнього» сховища, оскільки у вихідних даних може просто не виявитися інформації для його заповнення.

На даний момент більшістю експертів найбільш ефективним методом побудови сховища даних в даний час визнається гібридний, що поєднує два попередніх підходу і забезпечує, з одного боку, реалізацію продуктивного, що

відповідає інтересам користувачів сховища, з іншого – охоплення інформаційного потенціалу вихідних даних і гарантію його заповнення. Тут, в свою чергу, можна виділити інкрементальні методи, тобто незалежне вивчення потреб і фактів і подальше об'єднання результатів, і що чергуються, коли при паралельному взаємопов'язаному здійсненні аналізу результати постійно звіряються і коригуються. Для побудови сховища, яке не вимагає просунутих методів реалізації і яке, відповідає потребам бізнесу, досить використання інкрементального гібридного методу, в основу якого ліг спирається на базові кроки підходу Кімбол інструмент, запропонований Л. Кабібо і Р. Торлонія [18].

Метод передбачає поглиблений аналіз джерел даних, не пропонуючи формальних правил виведення багатовимірних параметрів, процес реалізується вручну на основі виявлених потреб користувачів і є в значній мірі інтуїтивним, тому метод можна віднести до гібридних. Після визначення багатовимірних параметрів кожен факт представляється як сутність, далі на базі джерел даних виводяться вимірювання. Крім обліку джерел даних, вибір методу пояснюється малою кількістю вітрин даних, відсутністю необхідності оперування даними досить значного обсягу або складності і потреби в поглибленому інтелектуальному аналізі, відносною простотою реалізації, що не вимагає спеціальної компетенції користувачів.

1.4 Аналіз проблем сховищ даних

Саме сховище має певну структуру і найчастіше побудоване за певної моделі даних. Модель даних являє собою опис всіх сутностей, об'єктів бази даних корпоративного сховища даних і включає в себе: концептуальну модель даних, логічну модель даних і фізичну модель бази даних. На рівні концептуальної моделі визначаються суті і взаємозв'язку між ними. На рівні логічної моделі сутності діляться на бізнес-області, їм дається докладний і повний опис,

прописуються взаємозв'язку. При розробці фізичної моделі бази даних визначається вся структура бази даних - від таблиць і полів в них, до партій і індексів.

Надалі, сформований сховище даних використовується як джерело для звітності: формуються області аналізу і вітрини даних.

На перший погляд може здатися, що структура сховищ даних досить проста. Насправді ряд серйозних проблем виникає вже на етапі проектування і реалізації. Самі ж труднорешаемі проявляються вже на етапі експлуатації і супроводу.

Прийнято виділяти три основні категорії проблем сховищ даних.

Проблеми якості даних – до даних (брудних) належать відсутні, неточні або даремні дані з точки зору практичного застосування (наприклад, представлені в невірному форматі, який не відповідає стандарту). Брудні дані можуть з'явитися з різних причин, таким як помилка при введенні даних, використання інших форматів представлення або одиниць вимірювання, невідповідність стандартам, відсутність своєчасного оновлення, невдале оновлення всіх копій даних, невдале видалення записів-дублікатів і т.п.

Проблеми вибору джерел даних – у сховищах даних проектують схему бази даних цільового сховища даних з використанням засобів моделювання баз даних. Схема бази даних складається з таблиць, стовпців (Полів) таблиць, типів даних і обмежень стовпців, а також зв'язків між таблицями. Проектувальники також визначають відображення (перетворення) схем джерел інформації на схему цільового сховища даних.

Але як проектувальники можуть переконатися в тому, що сховище даних містить всі дані, потрібні додаткам, які будуть над ним виконуватися, і не містить ніяких даних, які додаткам не потрібні? На той момент це ґрунтувалося на основі припущень досвідчених проектувальників.

Їм доводилося виявляти потреби в даних (таблиці і стовпчики), опитуючи

розробників додатків, бізнес-аналітиків (людей, які розуміють потреби додатків і бізнесу) і адміністраторів баз даних. Після початкового створення сховища часто виявлялося, що в ньому відсутні дані, необхідні для отримання відповідей на деякі запити, і присутні дані, які ніколи не потрібні додаткам. Хоча вартість зберігання може бути відносно невеликий, все поля, потрібні й непотрібні, зберігаються в одних і тих же записах і зчитуються і записуються спільно, що уповільнює швидкість вибірки, збільшує час обробки, а також призводить до неефективного використання середовища зберігання.

Проблеми продуктивності і масштабованості – у системах РБД для вибірки невеликої кількості потрібних записів без повного сканування таблиці або бази даних використовуються різні методи доступу, такі як індекси на основі хешування або В + -дерев. Такі методи доступу вельми ефективні при вибірці по одному ключовому полю (або невеликого числа полів), коли результати являють собою малу частину таблиці.

Однак методи доступу в загальному випадку не допомагають при відповіді на запити, результатами яких є значна частина таблиці. Крім того, серйозні проблеми доставляють запити агрегації і переміщення файлів в БД.

В даний час розроблені, успішно розвиваються і застосовуються цілі платформи для оцінки, контролю та управління якістю даних в масштабах підприємства.

Ця платформа часто тісно пов'язані з платформами інтеграції даних, що дозволяє створювати процедури перевірки якості даних безпосередньо всередині інтеграційного процесу.

Широкі можливості програмного забезпечення включають в себе:

- засоби аналізу, профілювання, розбору і очищення даних, що дозволяють архітекторам і аналітикам здійснювати перевірку і стандартизацію даних, покращувати і коригувати будь-які типи даних, включаючи інформацію про замовників, продуктах, фінансових та інших даних;

- інструменти для порівняння даних, які дозволяють визначати зв'язки між записами і наявність дублікатів для проведення подальшої уніфікації даних;
- набори імовірнісних методів для проведення порівняння даних, що спирається на фонетичні та синтаксичні особливості написання;
- функції моніторингу та динамічного формування звітності про якість даних. Це дозволяє організаціям забезпечувати управління якістю даних, що надходять за допомогою єдиного рішення;

У підсумку, проблема якості даних залишилася в тому сенсі, що різноформатних, помилковість, неточність даних неможливо викоринити повністю. Але в той же час вона відійшла на другий план, оскільки з'явився ряд потужних програмних рішень, за допомогою яких можна реалізувати єдиний процес з контролю якості даних.

З формулюванням проблеми вибору джерел даних взагалі можна не погодитися. Джерелами даних для сховища в загальному сенсі є операційні системи. Коли організація вирішує впровадити сховище даних, питання систем-джерел може обмежитися хіба що питанням «які системи є еталонними, а які другорядними?». Більш ймовірно, що під цією проблемою автор має на увазі проблему проектування структури сховища даних. Тобто які дані необхідно зберігати в сховище, щоб подальший аналіз міг вирішувати покладені на нього завдання і відповідати на актуальні бізнес-питання.

Дана проблема не вирішена повністю, але існують спеціальні галузеві моделі даних, які дозволяють максимально прискорити процес проектування структури сховища. Дані моделі покликані скоротити прірву між технічними архітекторами і бізнес-аналітиками. Спочатку визначається коло питань і бізнес-понять, які будуть мати відображення в сховище, потім проводиться деталізація до потрібного ступеня і перехід до понять фізичної структури БД.

У корпоративному сегменті дані моделі добре зарекомендували себе. У менш великих секторах бізнесу ця проблема стоїть менш гостро, тому що значно

менше масштаби сховищ даних.

Тому в підсумку можна сказати, що дана проблема також не є лідируючою.

В ідеалі компанії повинні прагнути, щоб зберігати детальні дані на найнижчому з можливих рівнів в функціонально-нейтральній моделі даних. Це дозволить бізнеспользователям формулювати запити найширшого спектра. Базова посилка полягає в тому, що завжди можна агрегувати детальні дані, проте виконати декомпозицію зведених даних неможливо. Це зовсім не означає, що ніколи не слід використовувати таблиці зведених даних.

Це означає лише, що не можна замінювати детальні дані зведеними. Основна причина створення таблиць зведених даних і денормалізації даних - бажання збільшити продуктивність.

Однозначних шляхів вирішення проблеми продуктивності на даний момент не існує. Є лише ряд методів, які в конкретних ситуаціях можуть дати приріст роізводітельності. Деякі з них:

Уявлення. Адміністратор бази даних може використовувати уявлення таким чином, щоб запропонувати кожному бізнес-підрозділу свою власну логічну функціональність. Уявлення пропонують дані користувачам саме в тому вигляді, на який вони розраховують, навіть якщо вони по-різному визначають один і той же елемент. При використанні уявлень дані повинні зберігатися лише одного разу, а потім перетворюватися в визначення логічного уявлення. Це знижує рівень надмірності даних, гарантує узгодженість даних і спрощує вимоги до управління даними.

Додаткові індекси. У більшості випадків індекси додаються з тим, щоб збільшити продуктивність до необхідного рівня. Основна перевага індексів полягає в тому, що система підтримує їх в паралель з таблицями бази даних. Хоча збереження і автоматичне оновлення індексів пов'язане з певними накладними витратами, набори відповідей завжди будуть узгоджені з детальними даними. Порівняння з детальними даними часто практикується при роботі з невеликими

довідковими таблицями, але необхідно ретельно продумати, чи варто застосовувати цю методику в разі більших і частіше оновлюваних таблиць.

Загальна розширення системи. Якщо мета полягає в тому, щоб отримати сховище даних реального часу, додавання низхідних процесів, таких як таблиці зведених даних і подальше вилучення даних або маніпулювання ними, створять бар'єр на шляху досягнення цієї мети. Якщо вдасться виправдати витрати, одна з можливостей - розширити системну конфігурацію. Цей крок, однак, істотно залежить від можливостей бази даних. Багато баз підтримують істотно послідовну обробку ряду операцій, таких як з'єднання, сортування або агрегування.

Якщо база даних має точки послідовної обробки, розширення, пов'язане з розпаралелюванням, може не привести до зростання продуктивності, оскільки в процесі обробки запитів дане рішення цей крок не прискорить.

При реалізації сховищ даних їх творці, як правило, ставлять собі за основну мету збільшення продуктивності. І в прагненні домогтися короткострокових переваг найчастіше ігнорують необхідність зберегти в подальшому гнучкість системи і можливість виконувати більш різноманітний аналіз. Щоб задовольнити першочергові вимоги, багато компаній швидко переходять на таблиці сукупних даних і функціональні моделі. Але якщо витратити досить невеликий час на те, щоб проаналізувати реальні цілі і розробити для них коректну основу, це гарантує довготривалість і перспективність сховища даних.

1.5 Висновки до розділу один

В першому розділі розглянуто базові моделі та методології проектування сховищ даних, їх недоліки та переваги. А також проаналізовано проблеми які виникають при проектуванні сховищ даних.

2 МЕТЕМАТИЧНА МОДЕЛЬ СТРУКТУРИ СХОВИЩА ДАНИХ

2.1 Тензорні методи

Тензор – це об’єкт, який має 2^r компонент в кожній двовірній системі координат та n^r в n вимірах, де r – ціле число (валентність) в залежності від типу тензора. Якщо ці компоненти позначити символом $T_c^a \dots$ (точки показують на подальші індекси), то тоді координати перетворюються по закону (лінійного перетворення):

$$x^a = C_b^a x^b, \quad (2.1)$$

компоненти тензора перетворюються таким чином:

$$T_r^p \dots = T_c^a \dots C_a^p C_b^q \dots C_r^c C_s^d \quad (2.2)$$

З цієї формули виходить, що легко перейти з одного виміру до іншого.

Побудову тензора краще розглянути на конкретному прикладі. Припустимо, що необхідно описати наступну ситуацію: співробітники, які працюють в деяких районах, сплачують податки, тобто сутності «Співробітник», «Податок» та «Район» знаходяться в деякому формальному відношенні «Документ», які в свою чергу також є сутністю. Формалізуємо ситуацію «перебувати в відношенні», використовуючи відношення H . Тоді той факт, що сутності $x_1, x_2, \dots, x_n$ перебувають у відношенні R , запишеться наступним чином:

$$(x_1, x_2, \dots, x_n, R) \in H. \quad (2.3)$$

Для простоти припустимо, що є два співробітника, два райони, два податки і три документи. Кожній сутності поставимо у відповідність одиничний вектор, взаємно ортогональний з усіма іншими векторами, відповідними іншим сутностям. Ці вектори, в силу їх взаємної ортогональності можемо тепер розглядати як базис прямокутної системи координат. В даному прикладі базис буде виглядати наступним чином (таблиця 2.1).

Таблиця 2.1 – Базис сутностей

Сп1	1	0	0	0	0	0	0	0	0	e_1
Сп2	0	1	0	0	0	0	0	0	0	e_2
П1	0	0	1	0	0	0	0	0	0	e_3
П2	0	0	0	1	0	0	0	0	0	e_4
Р1	0	0	0	0	1	0	0	0	0	e_5
Р2	0	0	0	0	0	1	0	0	0	e_6
Д1	0	0	0	0	0	0	1	0	0	e_7
Д2	0	0	0	0	0	0	0	1	0	e_8
Д3	0	0	0	0	0	0	0	0	1	e_9

Тобто ситуація зараз описується відношенням H (таблиця 2.2).

Таблиця 2.2 – Відношення H

Документи	Райони	Співробітники	Податки
Д1	Р1	Сп1	П1
Д2	Р1	Сп2	П1
Д3	Р2	Сп2	П2

Тоді кортежу в цьому відношенні поставимо відповідний вектор, який проєкції не в нуль на відповідні вектори базиса:

$$\begin{aligned} a_1 &= 1e_1 + 1e_3 + 1e_5 + 1e_7 = (101010100), \\ a_2 &= 1e_2 + 1e_3 + 1e_5 + 1e_8 = (011010010), \\ a_3 &= 1e_2 + 1e_4 + 1e_6 + 1e_9 = (010101001). \end{aligned} \quad (2.4)$$

Дані вектори являються одновалентними тензорами. Представляти змодельовану ситуацію будуть ті його компоненти, які знаходяться на:

- головній діAGONALІ (таблиця 2.1);
- пересічення з вимірами даних Π^4 , тобто:

$$\Pi^4 = A_1 \times A_2 \times A_3 \times A_4, \quad (2.5)$$

де A_1 – вимір, який відноситься до базисних векторів Д1, Д2, Д3,

A_2 – вимір, який відноситься до базисних векторів Сп1, Сп2,

A_3 – вимір, який відноситься до базисних векторів П1, П2,

A_4 – вимір, який відноситься до базисних векторів Р1, Р2.

Якщо представити тензор перерахуванням його ненульових компонентів, тоді 4-валентний тензор, який задає відношення Н запишеться таким чином:
 1.Д1.Р1.П1.Сп1; 1.Д2.Р2.П1.Сп1; 1.Д3.Р2.П2.Сп2; 1.Д1.Д1.Д1.Д1; 1.Д2.Д2.Д2.Д2;
; 1.Р1.Р1.Р1.Р1; 1.Р2.Р2.Р2.Р2.

При роботі БД постійно кружляють два потоки:

- потік запитів відносно тих чи інших даних або відношень;
- потік змін в структурі БД та її вміст.

Роздивимося конкретні приклади запитів до тензора.

Приклад 1. Знайти район, де працює співробітник Сп1.

Для відповіді на цей запит необхідно побудувати простір даних цього запиту Π^4 і спроекувати в нього тензор, а вже з цього тензора знайти відповідь.

1) Побудова простору даних. Для даного запиту простір даних буде виглядати наступним чином: $\Pi^4 = A_2 \times A_4$.

2) Проекція тензора в даний простір. Ця операція виконується за допомогою множення тензора відносини Н на тензор П і згортки цього результату: 1.Д1.П1; 1.Д1.П2; 1.Д2.П1; 1.Д2.П2; 1.Д3.П1; 1.Д3.П2.

Результатом даного кроку буде тензор інформації ТІ.

ТІ: 1.Сп1.Р1; 1.Сп2.Р1; 1.Сп2.Р2.

3) Відповідь знаходиться множенням тензора інформації на тензор запиту, який виглядає наступним чином: ТЗ: 1.Сп1.

В результаті отримаємо: ТВ: 1.Р1.

Відповідь: Район Р1.

Також обробку даного запиту можна описати наступною системою тензорних рівнянь:

$$\begin{cases} e^{\text{Сп}} * T_{\text{ДПСп}}^{\text{Д}} = e^{\text{Д}} \\ e^{\text{Д}} * T_{\text{Д}}^{\text{ПСпР}} = e^{\text{Р}} \end{cases} \quad (2.6)$$

Приклад 2.

Знайти район, де працює співробітник Сп2 з оплатою типу податку П2.

1) $\Pi^4 = A_2 \times A_3 \times A_4$.

2) П: 1.Д1; 1.Д2; 1.Д3.

ТІ: 1.Сп1.П1.Р1; 1.Сп2.П1.Р1; 1.Сп2.П2.Р2; 1.Д1.Д1.Д1.; 1.Д2.Д2.Д2.; 1.Д3.Д3.Д3.

3) ТЗ: 1.П2.Сп2; ТВ: 1.Р2.

Відповідь: Район Р2.

Даний приклад також можна описати системою тензорних рівнянь:

$$\begin{cases} e^{\text{СпП}} * T_{\text{СпПД}}^D = e^D \\ e^D * T_D^P = e^P \end{cases} \quad (2.7)$$

2.2 Тензорна модель структури реляційної бази

Основою пропонованого підходу є той факт, що кожному відношенню БД ставиться у відповідність побудований певним чином тензор, а для маніпулювання тензорами використовуються операції тензорною алгебри.

Нехай в деякій БД існує відношення:

$$R(A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_k; C_1, C_2, \dots, C_m), \quad (2.8)$$

де $A_1, A_2, \dots, A_n$ – n ключових атрибутів, атрибути $B_1, B_2, \dots, B_k$ визначають k зовнішніх ключів, $C_1, C_2, \dots, C_m$ – m інших атрибутів сутностей, які визначаються первинними ключами. Нехай атрибут сутності A_1 має l значень $a_1, a_2, \dots, a_{1l}$, атрибут A_2 – l значень $a_2, a_2, \dots, a_{2l}$, A_n – l значень $a_{n1}, a_{n2}, \dots, a_n$. Аналогічно і для інших атрибутів. Тоді кортежі відношення R можна представити у вигляді таблиці 2.3.

Таблиця 2.3 – Кортежі відношення R

A_1	A_2	...	A_n	B_1	B_2	...	B_k	C_1	C_2	...	C_m
a_1	a_2	...	a_{n1}	b_1	b_2	...	b_{k1}	c_1	c_2	...	c_{m1}
a_1	a_2	...	a_{n2}	b_1	b_2	...	b_{k2}	c_1	c_2	...	c_{m2}
...	...	...	...	...	...	...	...	...	...	...	...
a_{1l}	a_{2l}	...	a_{nl}	b_{1l}	b_{2l}	...	b_{kl}	c_{1l}	c_{2l}	...	c_m

Це відношення можна задати тензором:

$$R_{a_1, a_2, \dots, a_n}^{b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m}. \quad (2.9)$$

Для відношення $R(A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_k; C_1, C_2, \dots, C_m;)$ формування тензорного представлення буде описуватися наступним формальним правилом.

Правило 1:

- стосовно $R(A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_k; C_1, C_2, \dots, C_m;)$ присвоюється довільна заголовна літера, наприклад R;
- п ключовим атрибутам $A_1, A_2, \dots, A_n$, присвоюється п різнойменних довільних індексів, наприклад $a_1, a_2, \dots, a_n$;
- атрибутам $B_1, B_2, \dots, B_k$ та $C_1, C_2, \dots, C_m$ присвоюється $(k + m)$ різнойменних довільних індексів, відмінних від індексів (пункт 2), наприклад $b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m$;
- $a_1, a_2, \dots, a_n$ будуть відповідати коваріантним (нижнім) індексам тензора, що визначає вхідний потік даних, $b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m$ будуть відповідати контраваріантним (верхнім) індексам тензора, визначальним вихідний потік даних (рисунок 2.1);
- тензор матиме вигляд $R_{a_1, a_2, \dots, a_n}^{b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m}$.

Зауваження: Вхід і вихід тензора розглядається з точки зору потоку запитів до суті, таким чином, тензор описує відповідність між ключовими і не ключовими атрибутами, однозначно визначаючи які дані будуть результатом запиту при зверненні до суті через ключові атрибути.

Оскільки досить важко описувати побудова СД на абстрактних відносинах, то подальші математичні моделі будуть розглянуті на конкретному прикладі (рисунок 2.2).

Користуючись правилом 1, зробимо тензорну модель для БД.

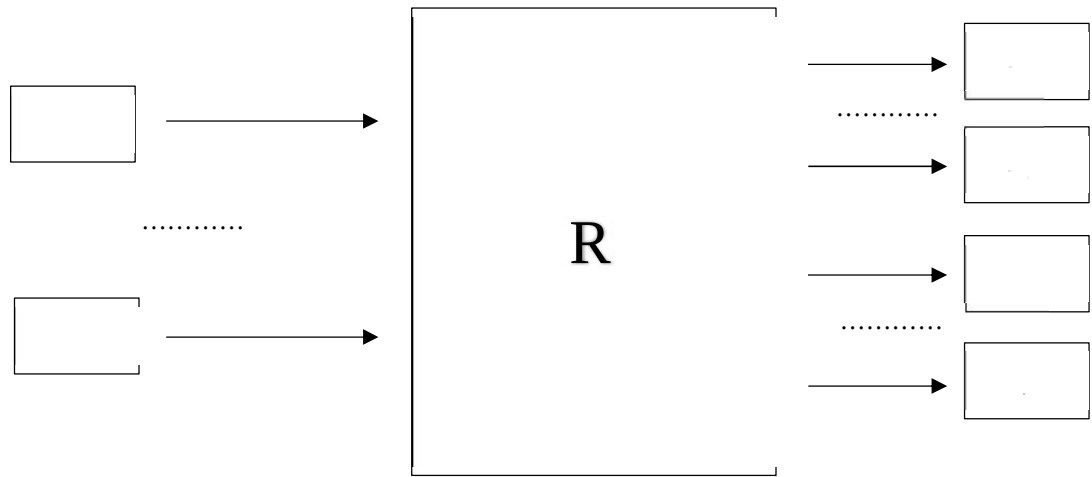


Рисунок 2.1 – Представлення тензору

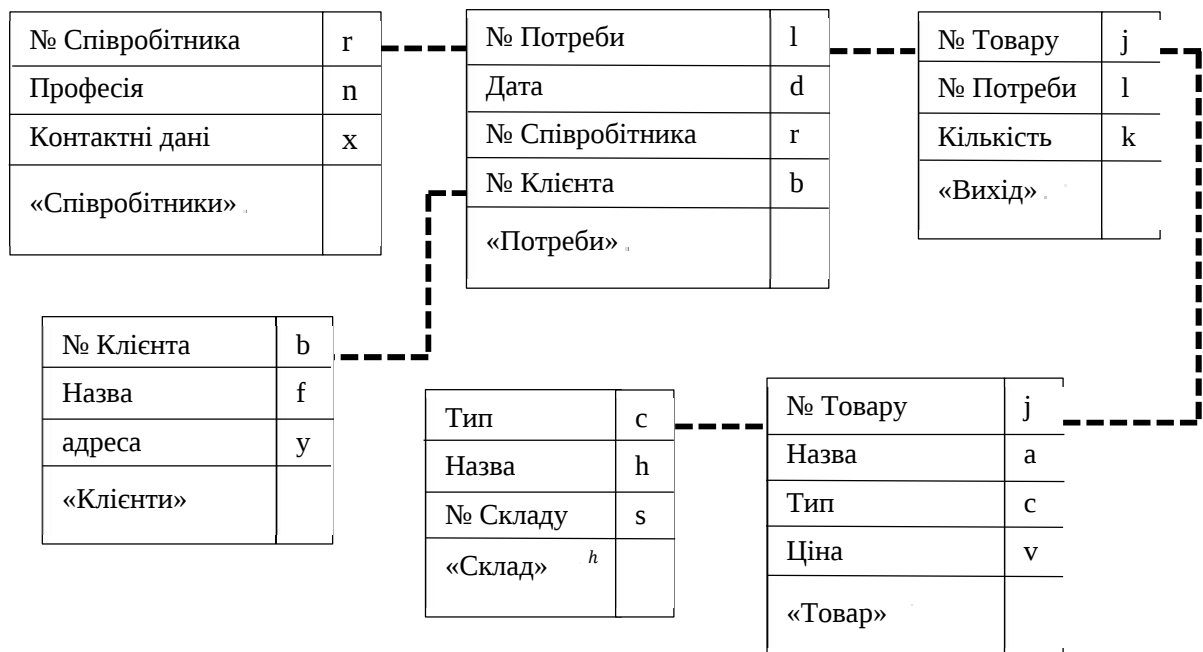


Рисунок 2.2 – Структура БД

Сутності «Співробітник» буде відповідати тензор F_r^n , сутності «Потреби» – тензор N_l^d , «Вихід» – P_j^k , «Товар» – T_j^a , «Склад» – W_c^{sh} , «Клієнти» – Q_b^f .

Зв'язок між сутностями БД, який в реляційній моделі проходить через первинні ключі, в даному випадку буде визначатися наявністю у тензорів однойменних індексів. Таким чином тензорна модель має вираз:

$$F_r^n, N_l^d, P_j^k, T_j^a, W_c^{sh}, Q_b^f. \quad (2.10)$$

2.3 Тензорна модель запитів до реляційної бази даних

Основними складовими структури реляційних сховищ даних є таблиця фактів (fact table) і таблиці вимірювань (dimension tables) [5].

Таблиця фактів є основною таблицею сховища даних. Як правило, вона містить відомості про об'єкти або про події, сукупність яких буде надалі аналізуватися. Зазвичай така таблиця містить унікальний складовою ключ, який об'єднує первинні ключі таблиць вимірювань. Крім цього таблиця фактів містить одне або кілька числових полів, на підставі яких в процесі виконання аналітичних запитів обчислюються агрегатні дані. Таблиці вимірювань містять незмінні або рідко змінювані дані. Таблиці вимірювань містять як мінімум одне описову поле і, як правило, цілочисельне ключове поле. Кожна таблиця вимірювань повинна знаходитися у відношенні «один до багатьох» з таблицею фактів [5].

Так як проявом тензора є N - матриця, яка при $N = 0$ називається константою, при $N = 1$ – вектором і т.п., то обробку запитів опишемо цими матрицями, що представляють тензор.

Наприклад, опишемо запит до сутності «Вихід» (таблиця 2.4).

Таблиця 2.4 – Записи сутності «Вихід»

№ Товару	№ Потреби	Кількість
j_1	l_1	k_1
j_2	l_2	k_2
...	...	...
j_n	l_n	k_n

В матричному представленні тензор P_j^k буде мати вигляд:

$$|k_j|_{n \times n} = \begin{bmatrix} k_1 & k_1 & \dots & k_{1n} \\ k_2 & k_2 & \dots & k_{2n} \\ \dots & \dots & \dots & \dots \\ k_{n1} & k_{n2} & \dots & k_n \end{bmatrix}. \quad (2.11)$$

Правило 2.

Потік запитів до сутності БД показується символом $e^\wedge$, під « $\wedge$ » мається на увазі індекси, показуючи вхідні дані запиту. Тобто можна описати $e^\wedge$ матрицею, в якій одиничні елементи розташовані на позиціях рівних значенню « $\wedge$ ». Описується це виразом:

$$e^j * P_j^k = e^k, \quad (2.12)$$

яке має наступне матричне представлення:

$$|e_j|_{n \times n} \times |k_j|_{n \times n} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & 1_j & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix} \times \begin{bmatrix} k_1 & k_1 & \dots & k_{1n} \\ k_2 & k_2 & \dots & k_{2n} \\ \dots & \dots & k_j & \dots \\ k_{n1} & k_{n2} & \dots & k_n \end{bmatrix} = k_j. \quad (2.13)$$

Аналогічно можна описати прості запити до всіх сутностей БД. Таким

чином, отримаємо тензорну модель обробки простих запитів у вигляді системи:

$$\begin{cases} e^r * F_r^n = e^n \\ e^l * N_l^d = e^d \\ e^j * P_j^k = e^k \\ e^j * T_j^a = e^a \\ e^c * W_c^{sh} = e^{sh} \\ e^b * Q_b^f = e^f \end{cases} \quad (2.14)$$

Для заповнення сховища даних, нам потрібно виконати декілька складних запитів, в яких виконується запит до декількох сутностей. Тому для моделювання складних запитів потрібно описати механізм тензорного представлення зв'язків між сутностями. Нехай в деякій БД існує відношення $R(A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_k; C_1, C_2, \dots, C_m)$, де $A_1, A_2, \dots, A_n$ – n ключових атрибутів, атрибути $B_1, B_2, \dots, B_k$ визначають k зовнішніх ключів, $C_1, C_2, \dots, C_m$ – m інших атрибутів сутностей та відношення:

$$U(B_1, B_2, \dots, B_k; D_1, D_2, \dots, D_q), \quad (2.15)$$

де – $B_1, B_2, \dots, B_k$ – k ключових атрибутів, $D_1, D_2, \dots, D_q$ – q інших атрибутів сутностей. Згідно першому правилу відношення $R(A_1, A_2, \dots, A_n; B_1, B_2, \dots, B_k; C_1, C_2, \dots, C_m)$ та $U(B_1, B_2, \dots, B_k; D_1, D_2, \dots, D_q)$ будуть описуватися $R_{a_1, a_2, \dots, a_n}^{b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m}$ та:

$$U_{b_1, b_2, \dots, b_k}^{d_1, d_2, \dots, d_q}, \quad (2.16)$$

з цього відношення виникає наступне правило.

Правило 3.

Якщо два тензори $R_{a_1, a_2, \dots, a_n}^{b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m}$ та $U_{b_1, b_2, \dots, b_k}^{d_1, d_2, \dots, d_q}$ мають однакові індекси, тобто $b_1, b_2, \dots, b_k$ і у одного з них цей індекс коваріантний, а в іншого контрваріантний, то відношення $U(B_1, B_2, \dots, B_k; D_1, D_2, \dots, D_q)$ знаходиться в зв'язку один до многих и з цього випливає, що ці індекси $b_1, b_2, \dots, b_k$ у тензорі $R_{a_1, a_2, \dots, a_n}^{b_1, b_2, \dots, b_k, c_1, c_2, \dots, c_m}$ являються зовнішніми ключами.

Згідно цього правила тензори T_j^a та W_c^{sh} , введені у розділі 2.2, що описують сутність тензором W_c^{sh} (сутність «Склад») знаходяться у зв'язку один до многих до сутності, яка описується тензором T_j^a (сутність «Товар»).

Приведемо матричний вираз, який взаємодіє з цими сутностями у складному запиті, наприклад «Знайти № Складу, на якому знаходиться товар № 2», тобто на вході мається значення атрибуту «№ Товару», на виході повинні отримати значення атрибуту «№ Складу». Потік запиту отримаємо такого виду:

$$e^j \rightarrow |e_j| = |0, 1, 0, \dots, 0|. \quad (2.17)$$

Тобто нам необхідно отримати значення зовнішнього ключа. В даному математичному представленні цей атрибут, помічений однойменним індексом з ключовим атрибутом сутності «Склад». В нашому прикладі цей індекс являється індексом – с, з цього виходить, що першим етапом необхідно отримати його значення. Для цього опишемо потік запиту до сутності «Товар»:

$$|e_j|_{1 \times n} \times |c_j|_{n \times 1} = |0, 1, 0, \dots, 0| \times \begin{vmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{vmatrix} = c_2, \quad (2.18)$$

де c_2 вказує на індекс потоку запиту, який буде звернений до сутності «Склад».

Математичний вираз, який описує дану обробку цієї частини запиту має вигляд:

$$|e_{c_2}|_{1 \times n} \times |s_c|_{n \times 1} = |0, 0, \dots, 1_{c_2}, 0, \dots, 0| \times \begin{pmatrix} s_1 \\ s_2 \\ \dots \\ s_n \end{pmatrix} = s_{c_2}. \quad (2.19)$$

Приведений матричний вираз зробимо системою тензорних рівнянь, які будуть являти собою модель складного запиту:

$$\begin{cases} e^j \otimes T_j^c = e^c \\ e^c \otimes W_c^s = e^s \end{cases} \rightarrow e^j \otimes T_j^c \otimes W_c^s = e^s. \quad (2.20)$$

2.4 Тензорна модель реляційного сховища даних

На основі двох різновидностей схем СД – «зірка» (таблиці вимірів зв'язані тільки з таблицею фактів) й «сніжинка» – хоча б одна таблиця вимірів посилається на таблицю, яка знаходиться по відношенню зв'язку «один до многих». На основі БД представленої у розділі 2.2 розроблений приклад схеми «зірка» (рисунок 2.3).

Сховище «Вихід товарів» описується наступною системою тензорних виразів:

$$\begin{cases} e^j \otimes P_j^k = e^l \otimes k \\ e^l \otimes N_l^d = e^d \end{cases}. \quad (2.21)$$

Тоді математична модель сховища «Вихід товарів» буде мати вигляд:

$$\begin{cases} e^j \sqcap P_j^k = e^l \sqcap k \\ e^l \sqcap N_l^d = e^d \\ e^r \sqcap K_r^n = e^n \\ e^j \sqcap T_j^a = e^a \\ e^b \sqcap Q_b^f = e^f \end{cases} . \quad (2.22)$$

Тільки при використанні схем типу «зірка» продуктивність досить нормально налаштованих реляційних систем може бути наближеною до продуктивності систем в основі котрих є багатомірні СД.

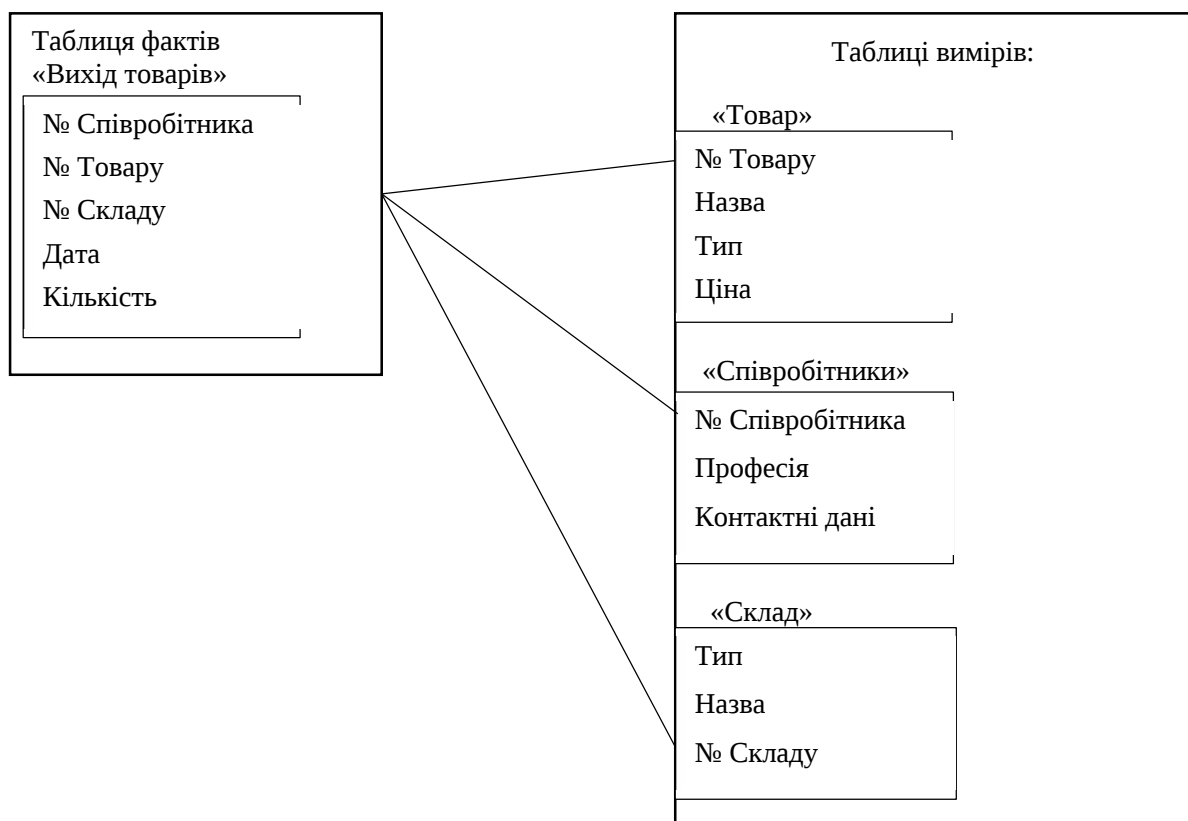


Рисунок 2.3 – Схема «Зірка»

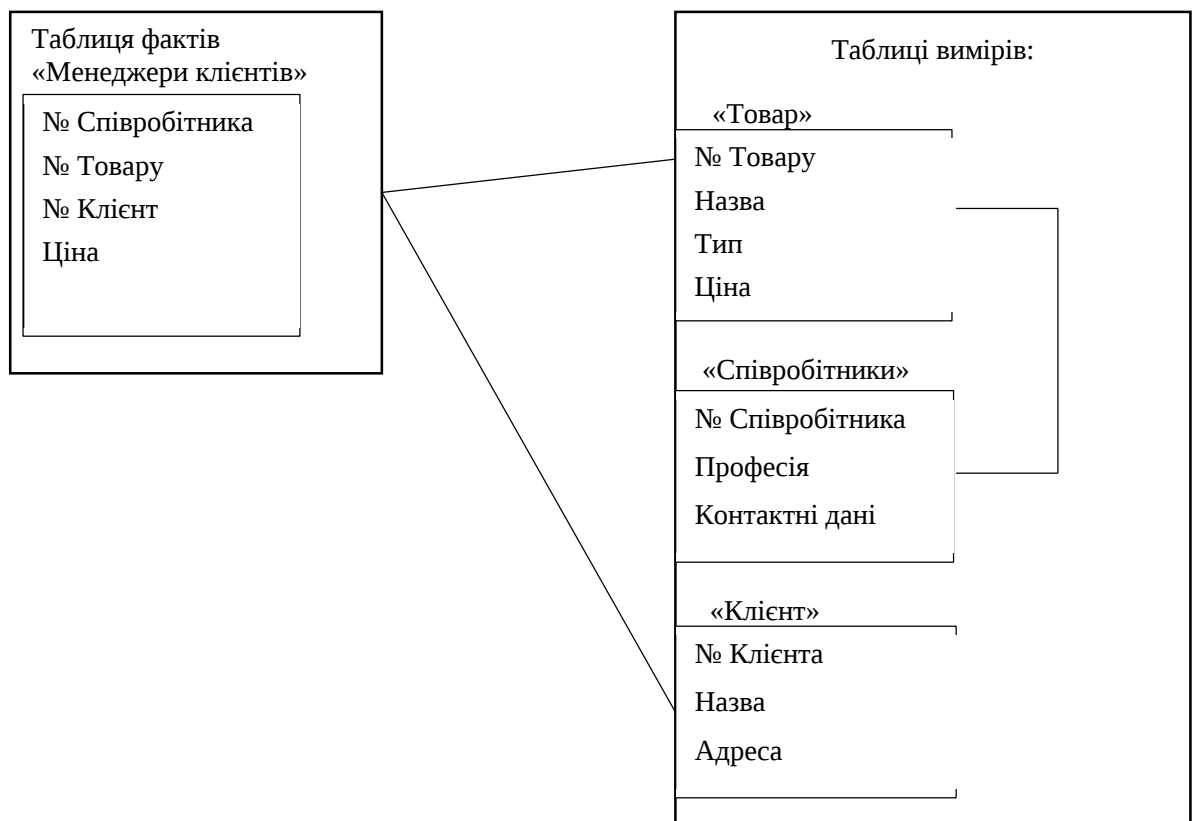


Рисунок 2.4 – Схема «Сніжинка»

Сховище «Менеджери клієнтів», відповідно схемі «сніжинка» будується на основі наступної системи тензорних рівнянь:

Таблиця фактів:

$$\begin{cases} e^j \otimes E_{j,i}^i = E_g^i = e^g \otimes i \\ e^g \otimes M_g^t = e^t \end{cases} \quad (2.23)$$

Таблиця вимірів:

$$e^p \otimes L_p^m = e^m \quad (2.24)$$

Взаємозв'язок сутностей «Співробітник» та «Товар» по відношенню до таблиці фактів утворює ієрархічну структуру, тому математично це описується системою:

$$\begin{cases} e^j \sqcap T_j^a = e^a \\ e^c \sqcap W_c^{sh} = e^{sh} \end{cases} \quad (2.25)$$

Об'єднав вирази у єдину систему отримаємо тензорну модель сховища даних «Менеджери клієнтів»:

$$\begin{cases} e^j \sqcap E_j^i = E_g^i = e^g \sqcap i \\ e^g \sqcap M_g^t = e^t \\ e^p \sqcap L_p^m = e^m \\ e^j \sqcap T_j^a = e^a \\ e^c \sqcap W_c^{sh} = e^{sh} \end{cases} \quad (2.26)$$

2.5 Висновки до розділу два

В цьому розділі було розглянуто побудову тензорів n-рангів, описано на прикладах роботу тензорних методів, також описано прості та складні запити до СД на основі тензорної алгебри.

Розроблено структуру СД та математично змодельоване із-за допомогою тензорів на основі бази даних реляційне СД.

3 АВТОМАТИЗАЦІЯ СХОВИЩ ДАНИХ

3.1 Проблеми автоматизації сховищ даних

Коли справа доходить до сховищ даних, зібрані дані можуть зберігатися в різних формах. Щоб ефективно аналізувати і використовувати дані, підприємствам з обробки даних часто необхідно інтегрувати інформацію, щоб зробити її сумісною. Це вимагає багато ручної роботи, яка, в свою чергу, може зайняти цінний час. Щоб боротися з цим, засоби автоматизації даних можуть використовуватися, щоб спростити процес. Тут ми розглянемо, чому вони так цінні для організацій.

Оскільки сховища даних зберігають дані в різних формах, це може викликати проблеми для підприємств і контролерів даних, які намагаються ефективно проаналізувати наявні у них дані. Інтеграція - це процес перекладу всіх даних в одну уніфіковану форму. Процес включає в себе виправлення розбіжностей в найменуванні і одиницях виміру. Це дозволяє організаціям більш ефективно аналізувати дані, отримуючи результати, які вони можуть ефективно використовувати.

Основними принципами проектування ХД в даний час є простота (з точки зору кінцевих користувачів) і продуктивність (виправдує очікування кінцевих користувачів). Додатковими критеріями, облік яких бажаний при проектуванні, є накладні витрати (на саме проектування, на повсякденне адміністрування і т.д.), вартість (персоналу, програмних і апаратних засобів), ризики неотримання необхідних результатів, ризики надмірної централізації.

Ці принципи і аспекти покладені в основу сучасних підходів до моделювання сховищ даних. Зазначені підходи можна розділити на дві основні групи:

- підходи, засновані на традиційному ER-моделюванні;
- підходи, засновані на багатовимірному моделюванні даних.

Використання підходів першої групи дозволяє найбільш повно описати предметну область підприємства і формалізувати процес зберігання історичної інформації. Підходи другої групи дозволяють досить швидко побудувати модель для аналізу різних показників, однак мають ряд недоліків, таких як прийняття компромісних рішень в ході вибору способу зберігання історичних даних і т.п.

Складні проблеми розробки сховищ даних, такі як тривалі цикли розробки, погане управління метаданими в існуючому сховищі даних, а також дорогі ресурси розробки зробили традиційні архітектури сховищ даних непридатними для швидко мінливих ринкових умов.

До кінця тисячоліття підприємства виявили, що багато хто з їх систем були погано інтегровані з базами даних і прикладними системами і були нездатні інтегрувати обсяги фрагментованих даних. Це проклало шлях до гнучкої платформи, яка може автоматизувати процеси ETL і легко інтегруватися з корпоративними додатками.

Сьогодні інструменти автоматизації сховища даних еволюціонували з урахуванням нових технологій і вимог бізнесу. До них відносяться вилучення даних в реальному часі, аналіз хмарних даних і сервісів веб-додатків, таких як REST API і SOAP, а також інтеграція з інструментами візуалізації даних.

Автоматизовані інструменти зберігання даних дозволяють підприємствам забезпечити перевагу на ринку за допомогою таких переваг:

1) Покращена якість та точність даних, тобто компанії можуть уникнути невідповідностей, виявлених у ручному ETL, та отримати більш високу ефективність запитів. Інтерфейс програми та точки клацання полегшує вилучення різних даних із баз даних, Excel, файлів з обмеженим доступом та інших джерел. Він також дозволяє користувачам моделювати повільно мінливі розміри та мігрувати дані складських приміщень на інші системи призначення,

такі як хмарний ВІ та засоби візуалізації даних. Отже, підприємства мають не лише доступ до достовірних даних, але й мають більший контроль над розширеною та точнішою звітністю та аналізом.

2) Підвищена гнучкість та швидше використання вартості, тобто швидше розгортання сховищ даних та доступ до огляду даних дають компаніям покращену спритність бізнесу. Це дозволяє їм швидко реагувати на постійно мінливі ринкові умови, такі як несподівані зміни попиту та втрати наявного доходу. Наприклад, роздрібний продавець, що використовує програмне забезпечення для автоматизованого зберігання даних, може скоротити час, необхідний для використання ВІ-звітів та встановлення причин низьких продажів у різних торгових точках та відповідно лічильників. Коротше кажучи, рішення можна приймати швидше і краще відображати зміни на ринку за допомогою кращого аналізу впливу.

3) Більш висока пропускна здатність проектів для зберігання даних та рентабельність інвестицій, тобто відсутність ручного введення дозволяє користувачам набагато швидше будувати та розгортати сховища даних, звільняючи ресурси розробника та знижуючи витрати в процесі. Це дає можливість бізнес-командам більше часу розкривати зрозумілу інформацію, виконувати стратегічні рішення та забезпечувати більшу цінність проекту.

У той же час використання існуючих підходів до розробки ХД породжує цілий ряд проблем. Серед цих проблем як основні можна вказати наступні:

- в ХД виявляються недоступними дані, необхідні для прийняття рішень;
- недолік партнерських відносин між кінцевими користувачами і фахівцями в області інформаційних технологій, що виникає в процесі проектування ХД;
- відсутність явної пізнавальної та концептуальної моделі кінцевих користувачів, що визначають їх точки зору на розроблювальний ХД;
- запізнювання експорту в ХД даних, необхідних для прийняття рішень;

- недостатня подробиця даних, що зберігаються в ХД;
- незручні формати даних, визначені в процесі розробки ХД;
- занадто повільна доставка даних з ХД кінцевим користувачам;
- низька якість даних, що зберігаються в ХД;
- надмірна витрата сил і часу на створення моделі даних розробляється ХД.

Одним з основних симптомів виникнення помилок в процесі створення і експлуатації активних ХД є поява тисяч сутностей, які ніколи не наповнюються реальними даними. Цей симптом є наслідком поширеної думки, ніби зусилля, витрачені на розробку моделі даних, тільки затримують роботу по створенню активного ХД. Згідно з цим думку краще розробляти модель активного ХД на основі моделей реальних джерел даних, заздалегідь розраховуючи на те, що помилки і невідповідність даних будуть виявлятися безпосередньо в ході виконання процедур фізичної реалізації та експлуатації активного ХД.

Таким чином, рішення задачі автоматизації моделювання активних ХД є актуальним і дозволить домогтися як скорочення часу розробки активних ХД, так і підвищення якості створюваного ХД за рахунок відмови від створення надлишкових сутностей на ранніх етапах розробки активного ХД.

3.2 Метод автоматизації проектування сховища даних

На основі вибраного інструментарію мови С# було прийнято зробити метод автоматизації, який можна було би використовувати для проектування СД із сутностей БД розглянутих у розділі 2.2. Але які знаходяться в різних джерелах, наприклад в реляційній та NoSQL базах даних.

Результатом роботи повинні бути отримані необхідні структури СД, тобто вхідні данні представляють собою структури баз даних та їх атрибути, які відповідають кількісним показникам процесів, відображених у БД.

Було прийнято рішення використати технологію WPF та зробити акцент на системну бібліотеку System.Reflection. Завдяки методам рефлексії було розроблено інструмент, який будує об'єкти для сутностей тензорів на основі індексів-ключів.

Інструментом, завдяки якому перетворюються вхідні дані до єдиної форми є розроблений об'єкт-інструмент, який будує динамічний об'єкт (приклад 3.1) та заповнює його властивостями конкретних типів, або масивів типів (приклад 3.2).

```
private static TypeBuilder GetTypeBuilder()
{
    var typeSignature = "MyDynamicType";
    var an = new AssemblyName(typeSignature);
    AssemblyBuilder assemblyBuilder = DefinedDynamicAssembly(an,
        AssemblyBuilderAccess.Run);
    ModuleBuilder moduleBuilder = assemblyBuilder.DefineDynamicModule("MainModule");
    TypeBuilder tb = moduleBuilder.DefineType(typeSignature,
        TypeAttributes.Public |
        TypeAttributes.Class |
        TypeAttributes.AutoClass |
        TypeAttributes.AnsiClass |
        TypeAttributes.BeforeFieldInit |
        TypeAttributes.AutoLayout,
        null);
    return tb;
}
```

Приклад 3.1 – Динамічна побудова об'єкта (файл TypeBuilderHelper.cs)

Коли додаток отримує дані з джерел, в нашому прикладі застосовували MSSql Server та MongoDB. То в супереч різномірності типів даних и структур зв'язків нам потрібно привести всі данні до одного типу, який подалі буде використовуватися для генерації тензорів сховища даних.

Отримавши набір даних з додатку спочатку треба виділити атрибути з яких буде створюватися СД і кожен раз при переключенні підключення до різних БД – зберігати інформацію сутностей у таблицю фактів, за умови, що в ній є дані.

Після визначення атрибутів в таблиці фактів передаємо роботу інструменту

генерації об'єктів-сутностей та після чого ми передаємо данні наступному інструменту, який генерує тензори на основі індексів сутностей.

Тензорна модель, яка приймає дані виражається масивом тензорів $[F_r^n, N_t^d, P_j^k, T_j^a, W_c^{sh}, Q_b^f]$.

```
private static void CreateProperty(TypeBuilder tb, string propertyName,
Type propertyType)
{
    FieldBuilder fieldBuilder = tb.DefineField("_" + propertyName,
propertyType, FieldAttributes.Private);
    PropertyBuilder propertyBuilder = tb.DefineProperty(propertyName,
PropertyAttributes.HasDefault, propertyType, null);
    MethodBuilder getPropMthdBldr = tb.DefineMethod("get_" + propertyName,
MethodAttributes.Public | MethodAttributes.SpecialName |
MethodAttributes.HideBySig, propertyType, Type.EmptyTypes);
    ILGenerator getIl = getPropMthdBldr.GetILGenerator();
    getIl.Emit(OpCodes.Ldarg_0);
    getIl.Emit(OpCodes.Ldfld, fieldBuilder);
    getIl.Emit(OpCodes.Ret);
    MethodBuilder setPropMthdBldr = tb.DefineMethod("set_" + propertyName,
MethodAttributes.Public |
MethodAttributes.SpecialName |
MethodAttributes.HideBySig,
null, new[] { propertyType });
    ILGenerator setIl = setPropMthdBldr.GetILGenerator();
    Label modifyProperty = setIl.DefineLabel();
    Label exitSet = setIl.DefineLabel();

    setIl.MarkLabel(modifyProperty);
    setIl.Emit(OpCodes.Ldarg_0);
    setIl.Emit(OpCodes.Ldarg_1);
    setIl.Emit(OpCodes.Stfld, fieldBuilder);

    setIl.Emit(OpCodes.Nop);
    setIl.MarkLabel(exitSet);
    setIl.Emit(OpCodes.Ret);

    propertyBuilder.SetGetMethod(getPropMthdBldr);
    propertyBuilder.SetSetMethod(setPropMthdBldr);
}
```

Приклад 3.2 – Побудова властивостей для динамічного об'єкта (файл TypeBuilderHelper.cs)

Находимо тензор з контрваріантним індексом, наприклад v та k , знайдений тензор записуємо в масив $O_{u_1}^1$, тобто $O_{u_1}^1 = [T_j^a, P_j^k]$.

В масиві $O_{u_1}^1$ порівнюємо контрваріантні індекси тензорів, якщо два і більше тензорів містять хоча б один однойменний контрваріантний індекс, то вони будуть входити в одну таблицю фактів в якості мір.

В $O_{u_1}^1$ всі тензори мають однойменний коваріантний індекс j , з цього виходить що буде всього одна таблиця фактів з мірами k, v . Таблицю фактів опишемо тензором A_j^k .

Послідовно порівнюємо $O_{u_1}^1$ з $O_{u_2}^2$ з додаванням $+1$. Находимо тензори з однаковими індексами та записуємо в масив $O_{u_3}^3$, якщо не залишилось членів у масиві, то записуємо таблицю фактів, яка буде мати вигляд $A_{l_1}^k$.

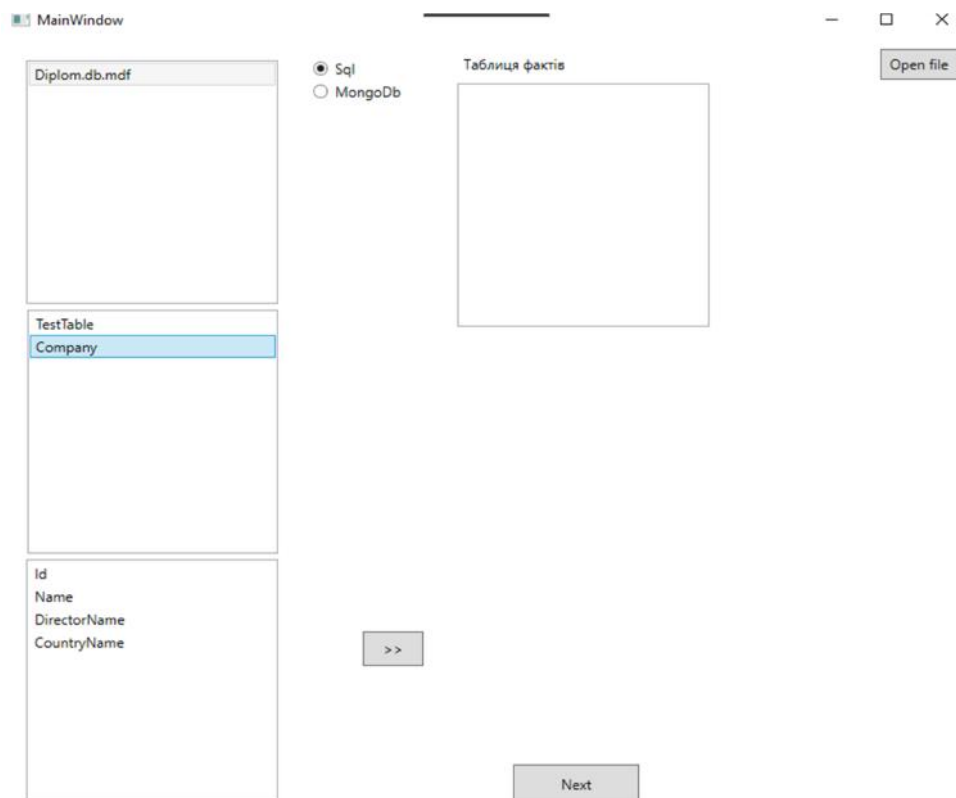
3.3 Висновки до розділу три

В даному розділі проаналізували проблеми, виокремили переваги і недоліки в автоматизації проектування сховищ даних. Розробили інструменти для можливості об'єднання даних різних типів, для легшого маніпулювання реляційними та нереляційними базами даних, для побудови необхідних сутностей у подальшому використанні структурами СД.

4. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

4.1 Результат експерименту

Практичне значення цього метода полягає в скороченні витрат трудових ресурсів при проектуванні СД із різнорідних даних. Додаток призначен дати користувачу змогу об'єднати данні з різних БД та побудувати СД при заданих атрибутах значень, які потрібно проаналізувати.



Рисунки 4.1 – Форми, що відображає вибір джерел даних MSSql Server

The screenshot shows a window titled 'MainWindow' with three main sections on the left:

- A list of MongoDB instances: 'admin.mongodb', 'config.mongodb', 'local.mongodb', and 'test.mongodb' (highlighted).
- A list of databases: 'countries' and 'people' (highlighted).
- A list of fields: '_id', 'Name', 'Age', 'Country', and 'Languages'.

In the center, there are radio buttons for 'Sql' and 'MongoDb' (selected). Below them is a '>>' button.

On the right, there is a section titled 'Таблиця фактів' (Fact Table) with a large empty box. Above it is an 'Open file' button. Below the '>>' button is a 'Next' button.

Рисунки 4.2 – Форми, що відображає вибір джерел даних MongoDB, а також вибір конкретних БД із сутностями при проектуванні СД.

The screenshot shows a window titled 'MainWindow' with three main sections on the left:

- A list of data sources: 'Diplom.db.mdf' (highlighted).
- A list of tables: 'TestTable' and 'Company' (highlighted).
- A list of fields: 'Id', 'Name', 'DirectorName', and 'CountryName' (highlighted).

In the center, there are radio buttons for 'Sql' (selected) and 'MongoDb'.

On the right, there is a section titled 'Таблиця фактів' (Fact Table) with a box containing 'DirectorName' and 'CompanyName'. Above it is an 'Open file' button. Below the '>>' button is a 'Next' button.

Рисунки 4.3 – Форми, в яких визначаються сутності сутність «Company»

The screenshot shows the 'MainWindow' application with the following components:

- Database Selection:** A list of databases: admin.mongodb, config.mongodb, local.mongodb, and test.mongodb. 'test.mongodb' is selected.
- Table Selection:** A list of tables: countries and people. 'people' is selected.
- Attributes:** A list of attributes: _id, Name, Age, Country, and Languages.
- Fact Table Definition:** A section titled 'Таблиця фактів' (Fact Table) with fields: DirectorName, CompanyName, Country, and Name.
- Navigation:** Radio buttons for 'Sql' and 'MongoDb' (selected), a '>>' button, and a 'Next' button.
- Buttons:** An 'Open file' button in the top right corner.

Рисунки 4.4 – Форми, в яких визначаються сутності «People», та їх атрибути на основі яких формується таблиця фактів.

The screenshot shows the 'MainWindow' application with the following components:

- Fact Table Definition:** A section titled 'Факти' (Facts) with fields: DirectorName, CompanyName, Country, and Name.
- Table Selection:** A section titled 'Виберіть виміри' (Select Measures) with a list of tables: Company and people. 'people' is selected.
- Attributes:** A section titled 'Атрибути вимірів' (Measures Attributes) with a list of attributes: _id, Name, Age, Country, and Languages.
- Navigation:** A '>>' button.
- Buttons:** A 'Delete' button and a 'Next' button.

Рисунок 4.5 – Форма, після визначення таблиці фактів СД.

На даний момент програмний продукт використовувався тільки на двох типах серверу: MSSql Server та MongoDB.

Далі, після завершення першого етапу побудови таблиці фактів попадаємо на наступну форму (рисунок 4.5).

По стандарту реалізовано побудову за схемою «Зірка», програма видає перелік вимірів, за якими спроектувати СД.

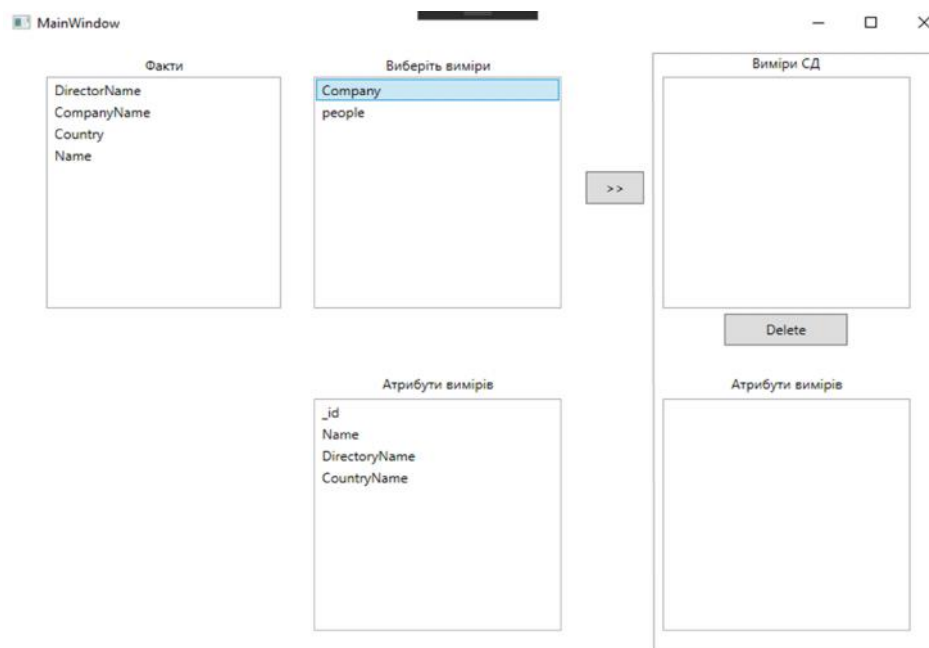


Рисунок 4.6 – Форми, які показують ступені визначення вимірів для проекрованої СД.

Даний програмний додаток реалізує метод автоматизації проектування структури СД. Вибрана схема зірка дає більше переваг у продуктивності замість схеми «сніжинка» тому що дані в ній денормалізовані. Таким чином представлений додаток рішає проблему об'єднання даних з різних БД при проектуванні СД.

The screenshot shows a 'MainWindow' with the following components:

- Факти (Facts):** A list containing DirectorName, CompanyName, Country, and Name.
- Виберіть виміри (Select dimensions):** A list containing Company and people.
- Виміри СД (SD Dimensions):** A list containing Company, which is currently selected.
- Атрибути вимірів (Dimension attributes):** A list containing _id, Name, DirectoryName, and CountryName.
- Buttons:** A '>>' button between the 'Виберіть виміри' and 'Виміри СД' panels, a 'Delete' button below the 'Виміри СД' panel, and a 'Complete' button at the bottom center.

Рисунок 4.7 – Форми, які показують ступені визначення вимірів для проекрованої СД.

4.2 Висновки розділу номер чотири

В даному розділі проілюстровано процес побудови СД на основі методів і алгоритмів автоматизації по схемі «зірка». Показано роботу з реляційною та нереляційною БД.

ВИСНОВКИ

Підсумок атестаційної роботи являється поліпшенням методів автоматизації на основі тензорних моделей, які дозволяють істотно зменшити трудність проектування СД з різних джерел, а саме реляційних та NoSQL баз даних, шлях об'єднання даних.

В ході виконання робіт:

- виконано аналіз робіт в області інформаційних технологій та розглянуті методи та моделі проектування СД;
- визначено переваги та недоліки методології проектування та автоматизації проектування СД на основі реляційних баз даних;
- описано роботу тензорного апарату, складність використання запитів у реляційних та багатовимірних СД;
- описано побудову тензорів, математично змодельоване їх математичне представлення;
- розроблений додаток, який експериментально підтверджує здатність до об'єднання даних із різних БД та використання інформації завдяки рефлексії при побудові тензорів;
- реалізована система, яка автоматизує проектування СД і дозволяє отримувати продуктивні структури для вирішення аналітичних питань основою яких є реляційні та нереляційні БД.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kimball R. The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouse. – New York: John Wiley & Sons, 2000.
2. Codd E.F. Providing OLAP to user-analysyst: An IT mandate. Technical report, 1993.
3. Туманов В.Е. Проектирование хранилищ данных для приложений систем деловой осведомленности (Business Intelligence Systems) 2-е изд., испр. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 958 с.
4. Макарычев, П. П. Построение моделей классов и объектов с применением тензорной методологии / П. П. Макарычев, Н. А. Попова // Университетское образование : сб. ст. XVII Международная науч.-метод. конф. – Пенза : Изд-во ПГУ, 2013. – С. 457–458.
5. Кудинов А.В. Хранилища данных. Цикл лабораторных работ. Часть 1: методологические указания к циклу лабораторных работ по дисциплине «Хранилища данных» для магистрантов, обучающихся по магистерской программе «Компьютерный анализ и интерпретация данных» направление 230100 «Информатика и вычислительная техника» / А.В. Кудинов. – Томск: Изд-во ТПУ, 2008. – 37 с.
6. List B., Bruckner R.M., Machaczek K., Schiefer J. A Comparison of Data Warehouse Development Methodologies Case Study of the Process Warehouse / A. Hameurlain, R. Cicchetti, R. Traunmüller (Eds) // Proceedings of the 13th International Conference on Database and Expert Systems Applications. DEXA 2002. Lecture Notes in Computer Science. Vol. 2453. Springer, Berlin, Heidelberg, 2002. Pp. 203-215. DOI: 10.1007/3-540-46146-9_21.
7. Romero O ., Abelló A. A Survey of Multidimensional Modeling Methodologies // International Journal of Data Warehousing and Mining. 2009. Vol. 5,

issue 2. Pp. 1-23. DOI: 10.4018/ jdwmm.2009040101.

8. Breslin M. Data Warehousing Battle of the Giants: Comparing the Basics of the Kimball and Inmon Models // Business Intelligence Journal. 2004. Vol. 9, issue 1. Pp. 6-20. URL: <http://www.bi-bestpractices.com/view-articles/4768> (дата звернення: 12.04.2018).

9. Romero O., Abelló A. Multidimensional Design / A.M. Tjoa, J. Trujillo (Eds) // Data Warehousing and Knowledge Discovery. DaWaK 2006. Lecture Notes in Computer Science. Vol. 4081. Springer, Berlin, Heidelberg, 2006. Pp. 85-94. DOI: 10.1007/11823728_9.

10. Распределенные базы и хранилища данных / Аносова Н. П., Бородин О. О., Гаврилов Е. С., Марасанов А. М. – ИНТУИТ, 2009.

11. Chizhukhin G. N., Bochkareva Yu. G. Tenzornaya metodologiya v diskretnoy sistemotekhnike [Tensor methodology in discrete systems engineering]. Penza: Izd-vo PGU, 2006, 184 p

12. Phipps C., Davis K.C. Automating Data Warehouse Conceptual Schema Design and Evaluation // Proceedings of the 4th International Workshop on Design and Management of Data Warehouses. Toronto, Canada, May 27, 2002. CEUR Workshop Proceedings. Vol. 58. Pp. 23-32. URL: <http://ceur-ws.org/Vol58/hipps-davis.pdf> (дата звернення: 12.04.2018).

13. Hüsemann B., Lechtenbörger J., Vossen G. Conceptual Data Warehouse Modeling / M. Jeusfeld, H. Shu, M. Staudt, G. Vossen (Eds.) // Proceedings of the 2nd International and Workshop on Design Management of Data Warehouses. Stockholm, Sweden: June 5-6, 2000. CEUR Workshop Proceedings. Vol. 28. Pp. 6.1-6.11. URL: <http://ceur-ws.org/Vol-28/paper6.pdf> (дата звернення: 12.04.2018).

14. Сарка Д. Microsoft SQL Server 2012. Реализация хранилищ данных: учебный курс Microsoft-М.: Изд-во «Русская редакция», 2014. – 816 с.

15. Giorgini P., Mylopoulos J., Nicchiarelli E., Sebastiani R. Formal Reasoning Techniques for Goal Models. S. Spaccapietra, S. March, K. Aberer (Eds.) Journal on

Data Semantics I. Lecture Notes in Computer Science. Vol. 2800. Springer, Berlin, Heidelberg, pp. 1-20, 2003. DOI: 10.1007/978-3-540-39733-5_1.

16. Romero O., Abelló A. Automating Multidimensional Design from Ontologies. Proceedings of the ACM tenth international workshop on Data warehousing and OLAP (DOLAP '07). Lisbon, Portugal: ACM Press, pp. 1-8, 2007. DOI: 10.1145/1317331.1317333.

17. Moody D.L., Kortink M.A. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses. Stockholm, Sweden: June 5-6, 2000. CEUR Workshop Proceedings. Vol. 28, pp. 5.1-5.12. Available at: <http://ceur-ws.org/Vol-28/paper5.pdf> (accessed 12.04.2018).

18. Cuzzocrea A. Data Warehousing and OLAP over Big Data: a survey of the state-of-the-art, open problems and future challenges. International Journal of Business Process Integration and Management. 2015; 17(4):372-377. DOI: 10.1504/IJB-PIM.2015.073665.

19. Cabibbo L., Torlone R. A Logical Approach to Multidimensional Databases. H.-J. Schek, F. Saltor, I. Ramos, G. Alonso (Eds.) Advances in Database Technology – EDBT '98. Proceedings of the 6th International Conference on Extending Database Technology. Valencia, Spain, March 23-27, 1998. Lecture Notes of Computer Science, Vol. 1377. Springer-Verlag Berlin Heidelberg, pp. 183-197, 1998. DOI: 10.1007/BFb0100972