Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

Магістр
(освітній ступінь)

на тему: Методи та засоби трансформації баз даних з ERM-моделі в реляційну модель

Виконав: студент 6 курсу, групи СІм-61
спеціальності
123 «Комп`ютерна інженерія»
(шифр і назва спеціальності (напряму) підготовки)
(підпис) Родзоняк А.В.
(прізвище та ініціали)

Керівник (підпис) Баран І.О.
(прізвище та ініціали)

Нормоконтроль (підпис) Тиш Є.В.
(прізвище та ініціали)

Рецензент (підпис) Скрупський Ю.І.
(прізвище та ініціали)

м. Тернопіль – 2019

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)

Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра комп'ютерних систем та мереж
Освітній рівень магістр
Напрям підготовки
(шифр і назва)
Спеціальність 123 «Комп'ютерна інженерія»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КС
Осухівська Г.М.
«30» 09 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Годзоняк Андрій Васильович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Методи та засоби трансформації баз даних з ERM-моделі в реляційну модель

Керівник проекту (роботи) Баран Ігор Олегович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «21» 09 2019 року № 4/7-85

2. Термін подання студентом проекту (роботи) 16.12.19

3. Вихідні дані до проекту (роботи) Методи та засоби трансформації баз даних

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Вступ 2. Аналіз предметної області 3. Дослідження технологій та механізмів трансформації ERM-схеми в реляційну схему 4. Практична реалізація засобу для трансформації ERM-схеми в реляційну схему та збереження в репозиторії 5. Обґрунтування економічної ефективності 6. Охорона праці та безпека в надзвичайних ситуаціях 7. Екологія

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Методи та засоби трансформації схем баз даних з ERM-моделі в реляційну модель 2. Актуальність теми дослідження 3. Модель "сутність-зв'язок" 4. Ілюстрування ERM-моделі з порівнянням з ER-моделлю 5. Концептуальна діаграма компонентів, що описує принцип роботи зовнішніх інструментів з репозиторієм Oracle Designer 6. Програмна архітектура засобу трансформації 7. Діаграма ієрархії класів, які використовуються для побудови реляційної моделі 8. Діаграма класів, яка використовується для трансляції ERM-схеми.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата: завдання видав	Підпис, дата: завдання прийняв
Охорона праці	Осуховська Г.М.	21.11.19	4.12.19
Обґрунтування економ. ефективн.	Тосів О.Б.	Тосів	Тосів
Безпека в НС	Стручок В.С., ст. викл. каф. ОХ	[illegible]	[illegible]
Екологія	Лесюта О.Н., доц.	[illegible]	[illegible]

7. Дата видачі завдання 30.09.19

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Аналіз предметної області	7.10.19	виконано
2	Дослідження технологій та механізмів трансформації ЕЕМ-схеми в реляційну схему	21.10.19	виконано
3	Практичне дослідження та реалізація засобу для трансформації ЕЕМ. схеми в реляційну схему та збереження в репозиторії	4.11.19	виконано
4	Обґрунтування економічної ефективності	11.11.19	виконано
5	Охорона праці та безпека в надзвичайних ситуаціях	18.11.19	виконано
6	Екологія	25.11.19	виконано
7	Попередній захист	28.11.19	виконано
8	Захист дипломної роботи		

Студент [signature] (підпис) Возняк Л.З. (прізвище та ініціали)

Керівник проекту (роботи) [signature] (підпис) Баран І.О. (прізвище та ініціали)

АНОТАЦІЯ

Методи та засоби трансформації баз даних з ERM-моделі в реляційну модель // Дипломна робота за освітнім рівнем «магістр» // Родзоняк Андрій Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра комп'ютерних систем та мереж, група СІм–61 // Тернопіль, 2019 // с. – , рис. – , табл. – , аркушів А1 – *10* , бібліогр. – .

Ключові слова: СХЕМА ДАНИХ, МОДЕЛЬ ДАНИХ, РЕПОЗИТОРІЙ, ERM-МОДЕЛЬ, РЕЛЯЦІЙНА МОДЕЛЬ, ТРАНСЛЯЦІЯ СХЕМ, ORACLE DESIGNER

Дипломна робота присвячена дослідженню технологій та методів трансформації схем баз даних з ERM-моделі в реляційну модель та створенню на їх основі програмного засобу в CASE-системі Oracle Designer для трансляції та зберігання ERM-схем баз даних.

У першому розділі проаналізовані основні моделі семантичного моделювання даних, зокрема моделі «сутність – зв'язок», розширена модель «сутність – зв'язок» «сутність – зв'язок – відображення», реляційна модель. Наведено їх ключові поняття та особливості використання.

У другому розділі наведено загальні відомості про методику трансформації ERM-схеми в реляційну схему, проаналізовані технології ручної та автоматичної трансформації. Описано CASE-систему Oracle Designer як базову платформу для проведення дослідження та механізм доступу до об'єктів ERM-схем.

У третьому розділі наведено програмну архітектуру засобу для трансформації ERM-схеми в реляційну. Описано модуль трансляції ERM-схеми в реляційну схему. Відображено процес практичної реалізації трансляції схем та встановлення плагіну у Visual Studio.

У четвертому розділі обґрунтовано економічну доцільність дослідження методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель шляхом розрахунку показників економічної ефективності, що дало змогу

протягом приблизно двох років компенсувати витрати на використання запропонованих технічних рішень.

В п’ятому розділі розглянуто важливі питання охорони праці та безпеки в надзвичайних ситуаціях, зокрема дослідження стійкості роботи ОГД в умовах надзвичайних ситуацій мирного та воєнного часу та можливість роботи в умовах радіаційного, хімічного, бактеріологічного забруднення і враховано основні способи захисту населення, які необхідно застосовувати. Розроблено заходи щодо планування дій із запобігання та пом’якшення наслідків в умовах надзвичайних ситуацій.

В шостому розділі «Екологія» розглянуто метод екологічної статистики і проблему екологічної відповідальності та організації «Зеленого офісу».

ABSTRACT

Methods and tools of databases transformation ERM-model into a relational model // Master thesis // Rodzoniak Andrii // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Systems and Nets, group CIm - 61 // Ternopil, 2019 // p.- , fig. – , table. – , Sheets A1 - *10*, Ref. - .

Keywords: DATA SCHEMA, DATA MODEL, REPOSITORY, ERM MODEL, RELATIONAL MODEL, SCHEMES TRANSLATION, ORACLE DESIGNER

Thesis deals with the research of technologies and methods of transformation of database schemas from ERM-model into relational model and creation on their basis of software in CASE-system Oracle Designer for translation and storage of ERM-schemas of databases.

The first section analyzes the basic models of semantic data modeling, including the "entity - communication" model, the extended entity "entity - communication" "entity - communication - reflection", and the relational model. Their key concepts and usage features are given.

The second section provides general information on the method of transformation of the ERM scheme into a relational scheme, and analyzes the technologies of manual and automatic transformation. Oracle Designer CASE is described as a basic research platform and mechanism for accessing ERM schema objects.

The third section describes the software architecture of a tool for transforming an ERM scheme into a relational one. The module of translation of ERM-scheme in relational scheme is described. The process of practical implementation of schema translation and installation of the plugin in Visual Studio is reflected.

The fourth section substantiates the economic feasibility of investigating methods and tools for transforming database schemas from an ERM model into a relational model by calculating cost-effectiveness indicators, which made it possible to offset the costs of using the proposed technical solutions for approximately two years.

The fifth section addresses important issues of occupational safety and health in emergency situations, including the study of the resilience of work in emergency situations of peacetime and wartime and the possibility of working in conditions of radiation, chemical, bacteriological contamination, and considers the main methods of population protection that need to be applied. . Measures have been developed to plan for the prevention and mitigation of emergencies.

The sixth section of Ecology deals with the method of environmental statistics and the problem of environmental responsibility and the organization of the Green Office.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – прикладний програмний інтерфейс (інтерфейс програмування застосунків)

CASE (Computer-Aided Software Engineering) - набір інструментів і методів програмної інженерії для проектування програмного забезпечення

EER ((Extended Entity-Relationship) model – розширена модель «сутність-зв’язок»

ER (Entity-Relationship) model – модель «сутність-зв’язок»

ERM (Entity - Relationship - Mapping) model – модель «сутність - зв'язок - відображення»

IDE (Integrated Development Environment) – інтегроване середовище розробки

OD – Oracle Designer

PL/SQL (Procedural Language / Structured Query Language) – мова програмування, яка використовується для доступу до баз даних Oracle.

RON – Repository Object Navigator, компонент OD

SQL (Structured Query Language) – мова структурованих запитів для взаємодії користувача з базами даних

XML (eXtensible Markup Language) – розширювана мова розмітки

ІС – інформаційна система

БД – база даних

ПЗ – програмне забезпечення

ПрО – предметна область

Репозиторій - спеціалізована база даних CASE-інструменту, в якій зберігаються аналітичні та проектні артефакти, що розробляється

ОС – операційна система

НС – надзвичайна ситуація

СУБД – система управління базами даних

ВСТУП

Актуальність теми. Технологія БД, спрямована на реалізацію потреби людини зберігати, упорядковувати і систематизувати інформацію, перетворилася на серйозну методологію і набула великого програмного інструментарію моделювання та зберігання великих масивів даних довільного типу і структури, а також обробки запитів і операцій оновлення даних. Починаючи з найпростіших сховищ даних, технологія зберігання БД розвинулася до складних інформаційних структур, що містять в собі свою внутрішню логіку. Технологія БД завоювала настільки сильну довіру як потужний інструмент зберігання та обробки даних, що використовується у всіх сферах діяльності людини. Тому актуальність проектування продуктивних і надійних БД з урахуванням зростаючих на них навантажень стоїть дуже гостро. Крім того, на сьогоднішній день потрібно високоякісно проектувати і впроваджувати БД за досить короткий час. Більш того, в умовах сучасного бізнесу і великих підприємств спроектовані БД повинні відрізнятися високою надійністю зберігання інформації, швидкою обробкою інформації, а так само малим часом відгуку.

У зв'язку з цими суперечливими вимогами процес розробки складної БД повинен був проходити з урахуванням пріоритетних вимог замовників в кілька етапів і, найчастіше, в кілька ітерацій на кожному етапі, щоб дотримати всі особливості кожної конкретної предметної області. Тому при проектуванні складних БД розробник повинен був покладатися на своє сприйняття світу, життєвий досвід, знання технологій і частку фантазії.

Альтернативою такому підходу стало використання семантичної методики проектування БД, при якій основна увага приділяється переносу законів ПрО в схему семантичної моделі, багатою виразними здібностями. Для застосування семантичної методики з використанням потужної семантичної моделі постала необхідність в розробці зручного CASE-інструменту для автоматизованої трансляції схем БД в реляційну модель.

Зв’язок із науковими програмами, планами, темами. Магістерська робота виконана відповідно до наукової тематики Тернопільського національного

технічного університету імені Івана Пулюя, кафедри комп'ютерних систем та мереж.

Мета роботи: дослідити існуючі методи та засоби трансформації схем баз даних з ERM-моделі в реляційну модель та розробити модуль автоматизованої трансляції та зберігання схем з ERM-моделі в реляційну модель для CASE-засобу.

Об'єкт дослідження: процес трансляції ERM-схеми в реляційну схему.

Предмет дослідження: технології, алгоритми та методи трансформації схем баз даних з ERM-моделі в реляційну.

В роботі поставлено та розв'язано **наступні задачі**:

– проаналізувати основні моделі даних, отримані з моделі «сутність-зв'язок»;

– дослідити технології ручної та автоматизованої трансляції ERM-схеми в реляційну;

– розробити прототип програмного засобу для трансформації ERM-схеми в реляційну схему;

– реалізувати репозиторій для ERM-схем в CASE-системі Oracle Designer.

Наукова новизна отриманих результатів:

– розширено структуру та функціональність ядра CASE-системи;

– розроблено динамічну бібліотека, яка містить класи для трансформації схем з ERM-моделі в реляційну;

– розроблений виконуваний прототип модуля в середовищі Visual Studio;

– впроваджено спроектований набір призначених для користувача розширень та створений механізм доступу для роботи з репозиторієм.

Методи дослідження: Метод теоретичного дослідження та експериментальний з використання персонального комп'ютера. Методологічну основу дослідження становлять фундаментальні положення комп'ютерної на програмної інженерії, наукові дослідження вітчизняних і зарубіжних компаній та вчених у сфері комп'ютеризованих систем.

Практичне значення одержаних результатів. Розроблений прототип модуля може легко інтегруватися в середовище Visual Studio для отримання реляційних схем, на основі яких можна генерувати SQL-скрипти для побудови баз даних.

Репозиторій дозволяє на практиці перевірити придатність тих чи інших пропозицій щодо вдосконалення самої семантичної моделі.

Публікації. Результати дослідження доповідалися на VII науково-технічній конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (11-12 грудня 2019 р.) у вигляді тез:

1. Родзоняк А.В. Трансформація схем баз даних з ERM-моделі в реляційну. Інформаційні моделі, системи та технології: Праці VII наук.-техн. конф. (Тернопіль, 11-12 грудня 2019 р.) Тернопіль, 2019. С. 88.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається з вступу, 6 розділів, висновків, списку використаної літератури та додатків. Обсяг роботи: пояснювальна записка – арк. формату А4, графічна частина – 10 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Модель «сутність – зв’язок»

Логічну структуру в якій зберігаються дані в БД, називають моделлю даних. У реальному проектуванні структури БД застосовується так зване семантичне моделювання, в основі якою лежить аналіз змісту даних. Як інструмент семантичного моделювання використовуються різні варіанти діаграм «сутність - зв'язок». На використанні різновидів даної моделі базуються більшість сучасних підходів до проектування моделей даних. Модель «сутність-зв'язок» є простою візуальною моделлю даних. Модель "сутність - зв'язок" (ER-модель Entity-Relationship model або Entity-Relationship diagram) – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. Вона визначає значення даних в контексті їх взаємозв'язку з іншими даними

Модель "сутність-зв'язок" – це модель даних, що використовується при проектуванні різноманітних моделей (ІС, БД, архітектур комп'ютерних додатків та інших систем) і є високорівневою концептуальною моделлю. Вона ґрунтується на деякій важливій семантичній інформації про реальний світ і є графічною нотацією, за допомогою якої можна описувати об'єкти логічних моделей даних і відношення між об'єктами. У даному контексті модель "сутність-зв'язок" є метамоделлю даних, тобто засобом специфікації логічних моделей даних, що будуються на основі вихідної концептуальної моделі даних. [1]

Важливим є той факт, що з моделі "сутність-зв'язок" можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найбільш загальною. Будь-який фрагмент наочної області може бути представлений як множина сутностей, між якими існує деяка множина зв'язків.

Модель "сутність-зв'язок" запропонував П. Чен із метою впорядкування задачі проектування моделей, її проект був опублікований у 1976 р. [2]. Російський переклад було опубліковано в 1995 році [3]. Дана модель задовольняє дві важливі умови:

– потужність її засобів дозволяє досить адекватно описувати структуру різноманітних ПрО;

– розрив між можливостями моделі та CASE-засобів (Computer Aided Software Engineering), що її підтримують, не надто великий.

Розглядаючи модель "сутність-зв'язок", варто використовувати формальне означення поняття "модель даних", яке дав в роботі [4] Кодд. Модель даних визначається як сукупність:

– колекції типів (де тип – це множина значень);

– колекції операцій над екземплярами цих типів;

– колекції застосовуваних до типів і операцій обмежень цілісності.

Будь-яка розвинена модель (даних) містить три компоненти:

– опис структури даних (структурна частина);

– опис обмежень цілісності;

– опис операцій над даними (маніпуляційна частина).

Потрібно зауважити, що модель "сутність-зв'язок" не є формальною моделлю або є такою не в першу чергу. Фактично вона складається з набору переважно неформальних концепцій, але в ній присутні й формальні аспекти.

Модель «сутність-зв'язок» заснована на поняттях теорії множин, теорії відношень, логіки, теорії ґраток [5]. Її розглядають у такі етапи:

– визначення структури даних: сутності, типу сутності, зв'язку, типу зв'язку, атрибутів тощо;

– задання обмежень цілісності: обмежень на типи зв'язків та атрибутів, обмежень на значення атрибутів, ключів тощо;

– задання операцій над даними.

Базовими елементами моделі є сутності, атрибути і зв'язки. З об'єктами моделі "сутність-зв'язок" пов'язані поняття:

– тип - набір однорідних предметів, явиш, що виступають як єдине ціле;

– екземпляр - конкретний елемент набору, який (набір) визначає деякий тип;

– множина - конкретний набір екземплярі типу в деякий момент часу.

Сутність (entity) - це об'єкт, який може були ідентифікований якимось способом, що відрізняє його від інших об'єктів.

Тип сутності визначає набір однорідних сутностей деякого типу.

Набір сутностей (entity set) множини сутностей одного типу (що володіють однаковими властивостями). Сутність фактично є множиною атрибутів, які описують властивості всіх членів даного набору сутності.

Тип сутності визначає множину однорідних об'єктів, а «екземпляр сутності» - конкретний об'єкт з множини об'єктів.

Взаємовідношення сутностей визначається зв'язками.

Тип зв'язку - це осмислена асоціація між типами сутностей (зокрема, асоціація може бути задана на одному типі сутності).

Зв'язок - це асоціація між сутностями, що належать відповідним типам сутностей, які беруть участь у даному типі зв'язку.

Множина зв'язку це множина всіх зв'язків типу зв'язку в деякий момент часу. Охоплені зв'язком сутності називаються учасниками даного зв'язку. Кількість учасників зв'язку називається степенем зв'язку (relationship degree), або його арністю. Залежно від степеня зв'язку розрізняють:

– бінарні зв'язки (binary relationships);

– багатосторонні зв'язки (multiway relationships).

Бінарний зв'язок здатний з'єднати будь-яку сутність одного типу сутності з будь - якою сутністю іншого типу сутності, зокрема, з будь-якою сутністю того самого типу сутності.

Багатосторонній зв'язок охоплює більше двох типів сутностей або один тип сутності, в якому сутності беруть участь кілька разів у різних ролях. Засобом, за допомогою якого визначають властивості типу сутності або типу зв'язку, є атрибут.

Атрибут - поіменована характеристика сутності. За допомогою атрибутів моделюються властивості сутностей. Основна роль атрибутів – опис властивостей сутності. Інша роль ідентифікація екземпляра сутності. Тобто кожен екземпляр сутності повинен мати унікальну назву. Як назва виступає один або декілька атрибутів.

Множину значень, яких може набувати атрибут, називають доменом атрибуту. Домен визначає всі потенційні значення, які можуть бути присвоєні атрибуту. Розрізняють такі класи атрибутів:

– прості (simple) і складені (composite);

– однозначні (single-valued) і багатозначні (multivalued);

– базові (stored) і похідні (derived);

– ключі (keys).

Простий атрибут складається з одного компонента з незалежним існуванням: складений з кількох компонентів, кожен з яких характеризується незалежним існуванням. Прості атрибути іноді називають атомарними. Складений атрибут становить деяку композицію простих (або складених) атрибутів. Такі взаємозв'язки вказують на ієрархію атрибутів.

Ключ - атрибут або підмножина атрибутів, що унікальною, визначає сутність у складі типу сутності.

Виділяють такі види ключів:

– потенційний (candidate) - атрибут або набір атрибутів, що унікально ідентифікують сутності типу сутності (зв'язки типу зв'язку);

– первинний (primary) - деякий вибраний потенційний ключ типу сутності (зв'язку) для спрощення роботи із сутностями (зв'язками);

– альтернативний (alternative) потенційний ключ, що не є первинним.

На сьогодні не існує єдиного загальноприйнятого стандарту для моделі "сутність-зв'язок", але є набір загальних конструкцій, що лежать в основі більшості її варіантів (див. табл. 1.1). Така ситуація виникла через те, що різні автори пропонують свої елементи моделі й відповідну термінологію. [6]
Незважаючи на те, що модель "сутність-зв'язок" є високорівневим зображенням, вона не дає однозначного рішення для визначення проектантом вимог до її об'єктів. Наприклад, не завжди можна визначити, до якого елемента моделі можна віднести певний об'єкт (до типу сутності, зв'язку або атрибута). Аналіз ПрО та побудова її моделі є суб'єктивним процесом, тому проектанти можуть створювати різні, але допустимі інтерпретації одного й того самого факту.

Вибір варіанта значною мірою залежить від проектанта. Слід зауважити, що атрибут реалізувати значно простіше, ніж тип сутності або тип зв'язку. Відсутня абсолютна різниця між типами сутностей, зв'язків та атрибутами. Атрибут є таким (власне атрибутом) тільки тоді, коли пов'язаний із деяким типом сутності або зв'язку, в іншому контексті він може виступати самостійним типом сутності.

Таблиця 1.1

Ключові поняття моделі "сутність-зв'язок" та їх назви

Поняття (змістовне пояснення)	1 варіант	2 варіант	3 варіант	4 варіант	5 варіант
Тип сутності (entity type – складається, містить, породжує свої екземпляри)	Тип сутності	Множина сутності	Тип сутності	Клас сутності	Тип сутності
Екземпляр сутності (entity occurrence – належить своєму типу сутності, породжується своїм типом сутності)	Екземпляр сутності	Екземпляр сутності	Сутність	Екземпляр сутності	Сутність
Множина сутності (entity set – конкретний набір екземплярів типу сутності в деякий момент часу)	–	–	–	–	Множина сутності
Атрибут (attribute)	Властивість	Атрибут	Атрибут	Атрибут	Атрибут
Тип зв'язку (relationship type – складається, містить, породжує свої екземпляри)	Тип зв'язку	Зв'язок	Тип зв'язку	Клас зв'язку	Тип зв'язку
Екземпляр зв'язку (relationship occurrence – належить своєму типу зв'язку, породжується своїм типом зв'язку)	Екземпляр зв'язку	Екземпляр зв'язку	Зв'язок	Екземпляр зв'язку	Зв'язок
Множина зв'язку (relationship set – конкретний набір екземплярів типу зв'язку в деякий момент часу)	–	–	–	–	Множина зв'язку

Модель даних має сенс, якщо супроводжується мовою маніпулювання даними, яка підтримує всі її концепції. В основу мови маніпулювання даними моделі даних може бути покладена алгебра або числення даної моделі. Існують також мови, в основі яких лежать як алгебра, так і числення.

1.2. Розширена модель «сутність – зв’язок»

Із 80-х рр. минулого сторіччя почали розповсюджуватися нові типи додатків БД: мультимедійні додатки, інструменти автоматизованої підготовки виробництва та автоматизованого проектування. Виявилося, що для створювання моделі даних для цих БД недостатньо понять моделі "сутність-зв'язок", тому з'явилася розширена модель "сутність-зв'язок", або EER-модель (Extended Entity-Relationship model). Розширена модель містить усі концепції моделі "сутність-зв'язок", а також власні.

Як і у випадку моделі "сутність-зв'язок", не існує єдиного загальноприйнятого стандарту для розширеної моделі, але є набір загальних конструкцій, що лежать в основі більшості її варіантів.

Далі наведемо опис і формалізацію таких загальних конструкцій: типів сутностей суперклас і підклас; типів зв'язку суперклас і підклас, isa; концепцій уточнення й узагальнення, а також категоризації згідно з [6].

Таблиця 1.2

Ключові поняття розширеної моделі «сутність – зв’язок» та їх назви

Поняття	1 варіант	2 варіант	3 варіант	4 варіант	5 варіант
Суперклас	Супертип	Базовий клас	Суперклас	Надтип	Суперклас
Підклас	Підтип	Підклас	Підклас	Підтип	Підклас
Зв’язок суперклас /підклас	–	–	Зв’язок суперклас /підклас	–	Зв’язок суперклас/підклас
Зв’язок is-a	Зв’язок is-a	Зв’язок is-a	–	Зв’язок типу «Є»	Зв’язок is-a

Тип сутності суперклас – це тип сутності, що містить одну або кілька допоміжних сукупностей його сутностей, які мають бути зображені в моделі даних. Тип сутності підклас – допоміжна сукупність сутностей деякого типу сутності (суперклас), що має бути зображена в моделі даних. Уточнення того ж самого типу сутності можна виконувати на основі його характерних

особливостей, тобто один тип сутності суперклас може мати кілька множин типів сутностей підклас, що відображають різні способи групування сутностей типу сутності суперклас. Кожна сутність типу сутності підклас є сутністю відповідного типу сутності суперклас, але сутність типу сутності суперклас не обов'язково є сутністю деякого типу сутності підклас. Слід зауважити, що тип сутності підклас має містити щонайменше одну сутність, інакше не має сенсу його створювати.

1.3. Модель «сутність – зв’язок – відображення»

Початково своєю назвою модель «Сутність - Зв'язок - Відображення» або скорочено ERM-модель (від англійського Entity - Relationship - Mapping) [7] зобов'язана двом базовим поняттям моделі - «об'єкт» і «відображення». Раніше вона так і називалася: «Об'єкт - Відображення» [8]. Пізніше для зручності проектування БД з використанням цієї моделі в неї було додано традиційні поняття ER-моделі Чена [2,3,5] як похідні від базових понять і була утворена ERM-модель в своєму нинішньому вигляді.

ERM-модель продовжує розвиток ER-моделі в бік більш детального опису закономірностей ПрО. При цьому зберігаються основні, практично затребувані конструкції, а також вводяться нові, не менш корисні елементи, що істотно підвищують виразні можливості моделі. Кожна з груп структурних понять - базова і похідна, грає свою роль в процесі проектування - перша забезпечує виразну потужність і математичну обгрунтованість моделі, друга полегшує розуміння моделі людиною. Для використання обох груп структурних понять взаємодоповнюючим чином запропоновані правила перетворення ERM-схем з мови похідних понять на мову базових понять і навпаки. Вони дозволяють здійснювати взаємно-однозначні перетворення схеми, необхідні для поповнення схеми, перевірки її на несуперечливість і трансформації в СУБД-орієнтовані моделі, що служать як цільових моделей процесу проектування БД.

Варто перерахувати нововведення ERM-моделі в порівнянні з ER-моделлю [7] :

– явно введено поняття «клас» як узагальнення понять «множина сутностей», «множина зв'язків» і «множина значень» (в попередніх версіях

ERмоделі воно було відсутнє). Саме класи будуть утворювати області визначення і області значень відображень;

– спеціалізації і категоризації підняті на рівень класу, що дозволяє розглядати ієрархії узагальнення не тільки множин сутностей, але також і множин зв'язків, і множин значень;

– введено поняття «відображення». Визначення кожного відображення включає крім іншого вказівку ролей образів і прообразів, а також класів, об'єкти яких грають ці ролі;

– виділено окремі випадки відображень - атрибутивні і реляційні відображення. Перші повністю відповідають аналогам в моделі Чена. Другі визначаються множинами зв'язків, і в якості образів і прообразів в них виступають суті;

– між відображеннями вводяться відносини слідства і еквівалентності;

– визначено алгебра відображень - набір операцій, що задаються на множині відображень. Кожна операція має одне або два відображення на вході і продукує одне відображення на виході;

– для спеціалізацій введено поняття підстави поділу. У цій іпостасі виступають відображення.

Ключовим моментом, що спричинило за собою можливості більш глибокого аналізу ПрО, є подальша декомпозиція поняття «множина зв'язків» (в ERмоделі при цьому розглядаються тільки ролі сутностей) і виділення так званих семантично значущих відображень. Кожна множина зв'язків ступеня n визначає 2n-2 відображень між множинами сутностей.

Насправді відображення в моделюванні даних притаманні не тільки множині зв'язків, вони пронизують всю схему даних і дуже тонко висловлюють семантику ПрО.

Основні базові поняття ERM-моделі забезпечують більшу частину виразної потужності моделі [8]. Однак людина не завжди використовує цей вельми абстрактний рівень мислення. Для простоти роботи зі схемою виділяються приватні види об'єктів, класів і відображень, що утворюють множину похідних понять моделі. Об'єкти, мислимі в висловлюваннях про ПрО як передпозначки, являють собою сутності , а класи таких об'єктів є не що інше, як множини

сутностей. Ідеальні об'єкти, такі, як числа, дати, рядки символів, є значеннями . Вони не володіють властивостями, характеристиками і не вступають у відносини з іншими об'єктами, крім того, що є значеннями характеристик об'єктів.

Слід зазначити, що поряд з основними структурними базовими поняттями - клас і відображення, для більш повного вираження семантики ПрО використовуються додаткові базові поняття - спеціалізація і категоризація, ролі об'єктів-прообразів і об'єктів-образів в примірниках відображень, а також операції і відногення (власності і рівності - для класів, слідства і еквівалентності - для відображень), визначені на множинах класів і відображень.

Відображення, яке ставить у відповідність об'єкту істинне значення, називається відображенням-властивістю. Якщо в якості області значень в відображенні використовується довільна множина значень, таке відображення будемо називати відображенням-характеристикою. Відображення-властивості є окремим випадком відображень-характеристик. Відображення-характеристики є не чим іншим, як атрибутними відображеннями, або просто атрибутами. [8]

Для прикладу, розглянемо варіант ERM-діаграми, зображений на рис. 1.1.

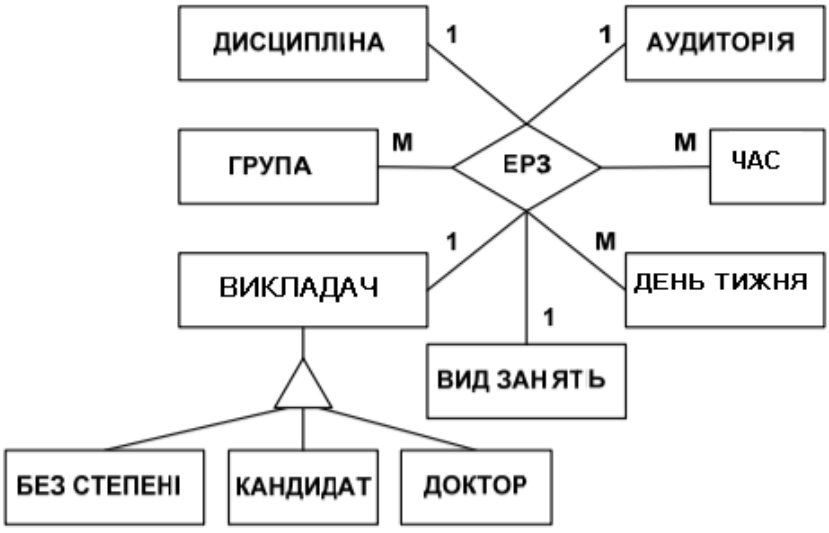


Рис. 1.1. Початкова ERM-схема.

Цей варіант діаграми практично збігається з ER-діаграмою Чена. Додана лише спеціалізація викладачів в залежності від наявності або відсутності у них того чи іншого наукового ступеня.

На схемі представлено основний об'єкт планування академічної діяльності – елемент розкладу занять (ЕРЗ). На думку проектувальника, він являє собою тип зв'язків, в кожній з яких задіяні конкретні дисципліна, студентська група, викладач, вид занять, аудиторія, час і день тижня. Дійсно, саме ці елементи вказані в кожній клітинці розкладу занять.

Таким чином, семантична модель даних «Сутність - Зв'язок - Відображення» поєднує в собі природність її понять для людини і повноту уявлення властивостей даних. Велика детальність опису ПрО в цій моделі неминуче повинна приводити до підвищеної складності її використання.

Добре продумана методика ERM-моделювання та CASE-засобу, що автоматизують побудову базових елементів ERM-схеми, синонімічних явно зазначеним проектувальником, частково спрощують його завдання. Людина фіксує семантику ПрО в зручних і зрозумілих йому формах. Переклад цих описів на низькорівневу, але більш виразну мову базових понять ERM-моделі повинен здійснюється автоматично. В результаті цих перетворень ERM-схеми у проектувальника з'являється можливість продовжити формалізацію своїх уявлень про ПрО на новому, недоступному раніше базовому рівні.

Таким чином, методика ERM-моделювання сприяє точної і повної формалізації уявлень про ПрО.

1.4 Особливості реляційної моделі даних та основні поняття реляційної БД

Реляційна модель даних - логічна модель даних, в основі якої лежить табличне представлення даних. Подання даних в цій моделі не залежить від способу їх фізичної організації. Це забезпечується за рахунок використання математичного поняття - відношення (таблиця).

Ця модель характеризується простотою структури даних, зручним для користувача табличним представленням і можливістю використання формального апарату алгебри відношень і реляційного обчислення для обробки даних.

Визначимо три складові частини реляційної моделі даних:

- структурна;
- маніпуляційна;

- цілісна.

Структурна частина моделі визначає, що єдиною структурою даних є нормалізоване n-арне відношення. Відношення зручно представляти у формі таблиць. де кожен рядок називається кортежом, а кожен стовпець – атрибутом, визначений на деякому домені. Даний неформальний підхід до поняття відношення дає більш звичну для розробників і користувачів форму представлення, де реляційна БД являє собою скінченний набір таблиць.

Маніпуляційна частина моделі визначає два фундаментальних механізми маніпулювання даними реляційну алгебру і реляційне числення. Основною функцією маніпуляційної частини реляційної моделі є забезпечення заходів реляційності будь-якої конкретної мови реляційних БД: мова називається реляційною, якщо вона має не меншу виразність і потужність, ніж реляційна алгебра або реляційне числення.

Цілісна частина моделі визначає вимоги цілісності сутностей і цілісності посилань. Перша вимога полягає в тому, що будь-який кортеж будь-якого відношення відмінний від будь-якою іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення має володіти первинним ключем. Вимога цілісності щодо посилань, або вимога зовнішнього ключа полягає втому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, на яке йде посилання, повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (тобто ні на що не вказувати).

Можна провести аналогію між елементами реляційної моделі даних і елементами моделі «сутність-зв'язок». Реляційні відношення відповідають наборам сутностей, а кортежі - сутностям. Тому, як і в моделі «сутність-зв'язок», стовпці в таблиці, що представляє реляційне відношення, називають атрибутами. Кожен атрибут визначений на домені, тому домен можна розглядати як множина допустимих значень даного атрибуту. Кілька атрибутів одних відношень і навіть атрибути різних відношень можуть бути визначені на одному і тому ж домені.

Іменована множина пар «ім'я атрибута ім'я домену» називається схемою відношення. Потужність цієї множини називають ступенем чи «арністю» відношення. Набір іменованих схем відносин являє собою схему БД.

Атрибут, значення якого однозначно ідентифікує кортежі, називається ключовим (або просто ключем). Якщо кортежі ідентифікуються тільки зчепленням значень декількох атрибутів, то говорять, що відношення має складовий ключ. Ставлення може містити кілька ключів. Завжди один із ключів оголошується первинним, його значення не можуть оновлюватися. Всі інші ключі відносини називаються можливими ключами.

Па відміну віл ієрархічної і мережної моделей даних в реляційної відсутнє поняття групових відношень. Для відображення асоціацій між кортежами різних відносин використовується дублювання їх ключів.

Переваги реляційної моделі:

– простота і доступність для розуміння користувачем. Єдиною виикористовуваною інформаційною конструкцією є «таблиця»;

– суворі правила проектування, які базуються на математичному апараті;

– повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;

– для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній нам'яті.

Недоліки реляційної моделі:

– далеко не завжди ПрО може бути представлена у вигляді таблиць;

– в результаті логічного проектування з'являється множина «таблиць». Це призводить до труднощів розуміння структури даних;

– БД займає відносно багато зовнішньої пам'яті;

– відносно низька швидкість доступу до даних.

Дані в реляційних БД представляються у формі таблиць. Від способу організації даних та структури таблиць залежить ефективність утвореної БД та зручність роботи з нею.

Кожна реляційна таблиця являє собою двовимірний масив і має такі властивості:

– кожний слемент таблиці - це один елемент даних;

– всі комірки в стовпці таблиці однорідні, тобто всі елементи в стовпці мають однаковий тип;

– кожний стовпець має унікальне ім'я;

– однакові рядки в таблиці відсутні;

– порядок наступності рядків і стовпців може бути довільним.

Основними поняттями реляційних БД є тип даних, домен, атрибут, кортеж, первинний ключ, відношення. Співвідношення між цими поняттями показані на рисунку 1.2.

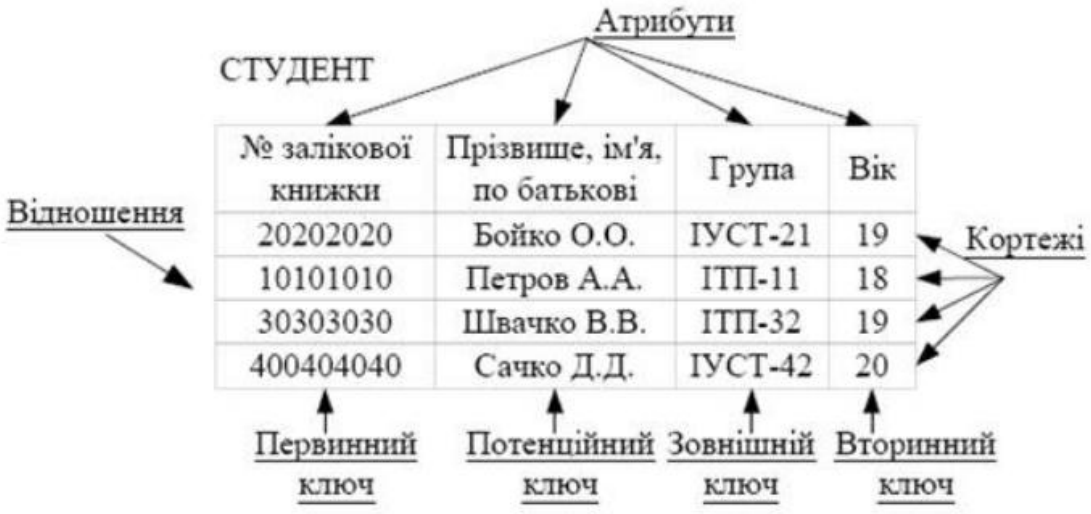


Рис. 1.2. Співвідношення між основними поняттями реляційних БД

Відношення - фундаментальне поняття реляційної моделі даних, яке задається переліком своїх елементів та перерахунком їх значень. Математично відношення може бути визначене як множина кортежів, що є підмножиною декартового добутку фіксованого числа областей D (доменів).

Таким чином, *n* -арним відношенням *R* називають підмножину декартового добутку доменів *D1, D2….Dn.* Домени *D1, D2….Dn* є допустимими потенційними множинами значень даного типу і характеризуються такими властивостями:

– наявність унікальної назви:

– визначеність на іншому домені або на типі даних:

– наявність певного смислового навантаження.

Типи даних в теорії реляційних БД відповідають типам даних у мовах програмування. Серед них можуть бути:

– числові:

– символьні:

– дата і час;

– спеціалізовані числові дані;

– малюнки та медіа-файли;

– інші прості типи даних.

Відношення графічно можна представити у вигляді таблиці, стовпці якої називаються атрибутами і відповідають входженням доменів у відношення.

Кількість атрибутів відношення називають його степенем.

Кількість кортежів - кардинальністю.

Схемою (заголовком) відношення називають множину пар (*A, D),* де *A* – назва атрибуту, *D* - відповідний домен.

Рядки таблиці називаються кортежами і є впорядкованими наборами з *n* значень (*A, D, v)* де *v* - відповідне значенням атрибуту.

Тілом відношення називається невпорядкована множина різних кортежів.

Зв'язок між таблицями встановлює відношення між співпадаючими значеннями в ключових полях. У більшості випадків із ключовим полем однієї таблиці (головної таблиці), що є унікальним ідентифікатором кожного запису, зв'язується зовнішній ключ іншої таблиці (підпорядкованої таблиці).

Міжтабличний зв'язок - це відношення, встановлені між полями (стовпцями) двох таблиць.

В моделі «Сутність-зв'язок» використовуються бінарні зв'язки, яких може бути три види:

– один до одного 1:1(запис у таблиці А має рівно один зв'язаний запис в таблиці В і навпаки);

– один до багатьох 1:М (кожному запису в таблиці А можуть відповідати кілька записів у таблиці В (в тому числі і нуль записів), але запис у таблиці В має рівню один відповідний йому запис в таблиці А);

– багато до багатьох М:М (одному запису в таблиці А можуть відповідати кілька записів у таблиці В (в тому числі і нуль записів), а одному запису в таблиці В кілька записів у таблиці А).

Окремим видом зв'язків у реляційних БД є рекурсивні зв'язки, які також є декількох типів:

– рекурсивний зв'язок «один-до-одного» (відображає структуру даних типу «черга»);

– рекурсивний зв'язок «один-до-багатьох» (відображає структуру даних типу «дерево»);

– рекурсивний зв'язок «багато-до-багатьох» (відображає структуру даних типу мережа»).

На таблиці, які є графічним представленням відношень у теорії реляційних БД, накладаються такі обмеження:

– відсутність кортежів-дублікатів;

– атомарність значень атрибутів: відсутність повторюваних структур даних;

– довільність порядку кортежів відношення;

– унікальність імен атрибутів відношення.

Для унікальної ідентифікації кожного кортежу відношення розглядають поняття реляційного ключа.

Основні операції реляційної алгебри, відповідно до особливостей їх виконання, можна розділити на декілька груп:

– об'єднання, перетин, різниця відношень;

– розширений прямий добуток відношень;

– проекція.

Для більш зручної роботи з БД стали вводитися додаткові елементи, які були вельми корисні в обробці, а також представленні даних. Більш того, з часом стали вводитися елементи, які підтримували обмеження цілісності згідно певній логіці і працювали в автоматичному режимі.

Базове відношення - відношення, кортежі якого фізично зберігаються в БД.

Представлення (view) - динамічний результат однієї або декількох реляційних операцій над базовими відношеннями з метою створення деякого іншого відношення. Представлення є віртуальним відношенням (virtual relation), яке реально в базі даних не існує, але створюється на вимогу окремого користувача в момент звернення до представлення.

Тригер (trigger) - це програма на мові програмування СУБД (для Oracle - це PL/SQL) для забезпечення цілісності даних і реалізації складної бізнес-логіки , яка автоматично виконується СУБД в момент настання певної події. Тригер подібний до процедури СУБД тим, що є іменованим блоком з розділами оголошення,

виконання і обробки умов. Однак, на відміну від процедури, тригер виконується неявно в кожному випадку виникнення критичної події, і не має параметрів.

Приведення тригера в дію іноді називають запуском (firing), або активізацією тригера. Момент запуску тригера визначається за допомогою ключових слів BEFORE (тригер запускається до виконання пов'язаної з ним події, наприклад, до додавання запису) або AFTER (після події). У разі, якщо тригер викликається до події, він може внести зміни в запис, що модифікуються подією (за умови, що подія - не вилучення запису). Деякі СУБД накладають обмеження на оператори, які можуть бути використані в тригері (наприклад, може бути заборонено вносити зміни в базове представлення, на яке встановлюється тригер).

Крім того, тригери можуть бути прив'язані не до базового відношенню, а до представлення (VIEW). В цьому випадку з їх допомогою реалізується механізм «Оновлюваного уявлення». В цьому випадку ключові слова BEFORE і AFTER впливають лише на послідовність виклику тригерів, так як власне подія (видалення, вставка або оновлення) не відбувається.

У деяких СУБД тригери можуть викликатися не для кожної запису, який модифікується, а один раз на зміну таблиці. Такі тригери називаються табличними (див. лістинг 1.1).

Лістинг 1.1 Тригер на рівні таблиці

```
CREATE OR REPLACE TRIGGER DistrictUpdatedTrigger
AFTER UPDATE ON district
BEGIN
INSERT INTO info VALUES ('table "district" has changed');
END;
```

Для відмінності табличних тригерів від малих вводяться додаткові ключові слова при описі малих тригерів. У Oracle це словосполучення FOR EACH ROW.

Тут наведено лише основи синтаксису завдання тригерів – потужного механізму організації складної логіки всередині СУБД. Більш того, збережені процедури в СУБД мають гарну продуктивність, яка не на багато знижує час відгуку СУБД на зовнішню подію. Основна проблема, яка виникає при

використанні цього механізму – залежність синтаксису мови написання тригера від кожної конкретної СУБД. Єдиним пом'якшувальною моментом є те, що більшість реляційних СУБД дотримуються стандарту SQL-89, що дозволяє знижувати залежність від синтаксису кожного конкретного діалекту мови програмування тригерів

1.4. Висновки до розділу

У цьому розділі проаналізовані основні моделі семантичного моделювання даних, зокрема моделі «сутність – зв’язок», розширена модель «сутність – зв’язок» «сутність – зв’язок – відображення», реляційна модель. Наведено їх ключові поняття та особливості використання.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ТА МЕХАНІЗМІВ ТРАНСФОРМАЦІЇ ERM-СХЕМИ В РЕЛЯЦІЙНУ СХЕМУ

Успіх семантичної методики проектування безпосередньо залежить від потужності застосовуваної семантичної моделі даних. Дійсно, здатність семантичної моделі формально описати будь-яку, як завгодно складну ПрО - одна з головних переваг семантичної методики. Але залишається і другий важливий етап - ефективне перенесення семантичної схеми в СУБД-орієнтовану схему. Для цього потрібно реалізувати детально пророблений надійний набір правил цієї трансформації.

Надійність правил означає, що пропоновані перетворення будують бездоганну і ефективну схему. Правила залежать від детальності обліку численних факторів семантичної схеми. Досягти ідеальної логічної схеми дозволяє проектування, при якому жоден з елементів семантичної схеми не залишається неврахованих. Будь-який спрощений розгляд здатен привести в окремих випадках до неефективних схем БД.

2.1. Загальні відомості

У перспективі, поєднання виразної семантичної моделі даних і потужного набору правил її перетворення в логічні моделі СУБД в стані забезпечити ідеальну семантичну методику, з використанням якої можна буде отримувати бездоганні схеми БД. На вирішення цієї проблеми спрямовані наукові дослідження Бабанова А. М. [7-9].

Наступним етапом поліпшення процесу семантичного моделювання є автоматизація даного перетворення і реалізація семантичної методики у вигляді CASE-засобу - програмної системи, що автоматизує працю проектувальника схем БД. В даний момент в цьому напрямку реалізовані лише найслабші семантичні моделі, які малоефективні, внаслідок чого залишається лише реалізація трансформації на папері.

Застосування семантичної методики проектування схем БД з використанням ERM-моделі в якості семантичної моделі передбачає не тільки оригінальну методику семантичного моделювання, але також не менше оригінальну методику перетворення ERM-схеми в реляційну схему. Якщо класична методика проектування реляційної схеми і методика перетворення ER-схеми в реляційну забезпечує результат в ході виконання єдиного алгоритму без уточнення на кожному кроці цього алгоритму в ході виконання ми забезпечуємо ті чи інші якості схеми БД, то пропонована далі методика абсолютно чітко фіксує мету кожного етапу і ті дії, які призводять до цієї мети найкращим чином.

Неадекватна схема часто призводить до незадоволеності кінцевих користувачів, додаткових проблем адміністратора БД і розробників прикладного ПЗ. Природно, виникає питання про критерії, задоволення яких сприяє підвищеному якості цього такого важливого елемента системи.

Варто навести основні критерії якості проекту схеми БД є :

– інформаційна повнота : схема БД відповідає цьому критерію, якщо вона повністю забезпечує інформаційні потреби всіх майбутніх користувачів ІС;

– інформаційна коректність : схема БД відповідає цьому критерію, якщо в ній знайшли відображення всі виявлені на етапі аналізу ПрО типові властивості інформаційних об'єктів. Йдеться, перш за все, про обмеження цілісності, перевірка яких при всіх модифікаціях БД буде сприяти тому, що БД завжди буде перебувати в максимально несуперечливому стані, і, отже, відповіді системи на інформаційні запити користувачів будуть більш достовірними;

– інформаційна ненадмірність : схема БД відповідає цьому критерію, якщо в ній відсутні надлишкові структури даних і обмеження цілісності. У тих випадках, коли модель даних (зокрема - реляційна модель) змушує вдаватися до дублювання даних (необхідне дублювання) для представлення зв'язків між елементами БД, цей критерій вимагає, щоб таке дублювання було мінімальним. Відзначимо, що дотримання цього останнім критерієм не повинно йти на шкоду раніше згаданим критеріям повноти і коректності.

Вся методика трансформації ERM-схеми в реляційну схему передбачає послідовне виконання трьох груп правил для забезпечення результату з найкращою якістю:

– перша група правил (правила породження структур) служить основою для побудови, хоч і в більшості випадків надлишкових, але які задовольняють всі потреби користувачів структури даних;

– друга група правил (правила породження обмежень цілісності) забезпечує для структур максимально повне перенесення обмежень цілісності ERM-схеми;

- третя група правил (оптимізаційні правила) спрямована на виключення надлишкових структур і обмежень цілісності.

2.2. Технологія ручної трансформації

Технологія виконання ручного перетворення «ERM-схема - реляційна схема» здійснюється за такими правилами, які були отримані Бабановим А. М. в роботах [7-9].

В дипломній роботі наведені тільки самі правила, без уточнюючих пояснень.

Правила породження структур.

Правило 1. Кожній множині сутностей відповідає своє відношення, що включає наступні атрибути:

– сурогатний первинний ключ;

– однозначні атрибути цієї множини сутностей.

Правило 2 . Для кожної множини зв'язків ступеня n (представленого в графі класів ребром з n кінцями) будуємо відношення реляційної моделі з n атрибутами, кожен з яких є зовнішнім ключем, який є дублікатом первинного ключа відношення, побудованого для відповідної множини сутностей.

Правило 3 . Кожній багатозначній характеристиці кожного класу (множині сутностей або множині зв'язків) відповідає своє бінарне відношення з атрибутами:

– зовнішній ключ, який є дублікатом первинного ключа класу;

– атрибут для значення характеристики.

Правило 4 . Якщо група класів (множин сутностей або множин зв'язків) утворює незалежну ієрархію спеціалізацій, для ідентифікації об'єктів всіх класів цієї групи використовується один сурогатний ключ. Він визначається в першу чергу на рівні кореневого класу в якості первинного ключа йоговідношення.

Правило 5 . Для подання фактів множинного успадкування (клас належить одночасно спеціалізаціям різних батьківських класів) створюється одне додаткове відношення з чотирма атрибутами: \

- ім'я відношення суперкласу (SUPERCLASS);
- значення первинного ключа об'єкта щодо суперкласу (SUPER_ID);
- ім'я відношення підкласу (SUBCLASS);
- значення первинного ключа об'єкта щодо підкласу (SUB_ID).

Правила породження обмежень цілісності.

Правило 6. Обмеження цілісності на області значень відображень-характеристик. У реляційної моделі їх іноді називають обмеженнями цілісності на значення атрибутів.

Правило 7. Обмеження цілісності на відображення-характеристики і зворотні їм відображення. У реляційній моделі їх іноді називають обмеженнями унікальності і визначеності.

Правило 8. Обмеження цілісності на відображення між множинами сутностей. Ці обмеження цілісності втілюються в реляційної моделі у вигляді так званих обмежень посилальної цілісності, пов'язаних з визначенням зовнішніх ключів відносин, утворених з множин зв'язків.

Правило 9. Обмеження цілісності на спеціалізації .

Правило 10. Повністю надлишкові відношення для множин зв'язків, видалення яких не призводить до зміни інших відносин.

Правило 11. Надлишкові відношення для множин зв'язків, видалення яких призводить до зміни інших відносин.

Правило 12. Повністю надлишкові відношення для множин сутностей, видалення яких не призводить до зміни інших відносин.

Правило 13. Надлишкові відношення для множин сутностей, видалення яких призводить до зміни інших відношень.

Необхідно зазначити, що на завершальній фазі запропонованої методики потрібно ще раз ретельно перевірити чи залишилися відношення на задоволення критеріїв повноти і коректності - чи всі форми висловлювань та обмеження цілісності і раніше підтримуються схемою.

2.3. Технологія автоматичної трансформації

Автоматизація трансляції схем накладає обмеження на наведені вище правила перетворення, так як спочатку вони були розроблені для застосування їх людиною. У даній роботі для реалізації правил в алгоритмі трансляції був проведено їх додатковий аналіз. Його результатом став набір деталізованих правил, зміни в яких торкнулися порядок дій змін, вибір необхідних конструкцій, побудова елементів і їх іменуванні. Перетворені правила лягли в основу розробленого алгоритму. Запропонований алгоритм, використовуючи отримані правила в якості основи, послідовно, крок за кроком, в автоматичному режимі проводить трансляцію схем

З метою глибокої деталізації правил перетворення було проведено аналіз можливих варіантів породження структур для таких понять ERM-моделі як спеціалізація і категоризація, так як їх первісна реалізація багато в чому може вплинути на наступні етапи. [10]

Підсумком проведеної роботи став список можливих варіантів представлення спеціалізації (n- кількість підкласів) в реляційній схемі:

– метод 0: 1 відношення+ атрибут. Створюється одне відношення, яке включає однозначні атрибути суперкласу і атрибут з доменом імен підкласів в множині можливих значень, а також включає однозначні атрибути всіх підкласів, тобто кожен об'єкт в рамках спеціалізації описується кортежем тільки одного відношення;

– метод 1: 1 відношення + n атрибутів. Створюється одне відношення, яке включає однозначні атрибути суперкласу і всіх підкласів, для кожного підкласу в суперкласі створюється атрибут-прапорець, тобто кожен об'єкт в рамках спеціалізації описується кортежем тільки одного відношення;

– метод 2: n відношень. Створюється n відношень - по одному для кожного підкласу, однозначні атрибути суперкласу включаються в кожне з цих відносин.

Метод 3: n + 1 відношення. Для суперкласу і кожного з підкласів створюється по одному окремому відношенню. Один об'єкт може описуватися кортежами декількох відношень від 1 до n +1.

Метод 4: n + 2 відношення. Крім n відношень для підкласів і одного для суперкласу створюється сполучне відношення, яке має наступні атрибути: ім'я батьківського відношення; первинний ключ об'єкта в батьківському відношенні; ім'я дочірнього відношення; первинний ключ об'єкта в дочірньому відношенні.

Метод 5: 2^n відношення. Для будь-якого можливого піддерева в ієрархії спеціалізацій, що включає кореневий суперклас, створюється одне відношення, яке містить в якості атрибутів все однозначні атрибути множин сутностей, що належать піддереву.

Окрім наведених вище базових методів, додатково існують і гібридні методи.

Метод 6 (гібрид 2 і 3 методів): n + 1 відношення. Для суперкласу і кожного з підкласів створюється по одному окремому відношенню. Крім цього, однозначні атрибути суперкласу додаються в кожне відношення, побудоване для підкласів.

Метод 7 (гібрид 1 і 3 методів): n + 1 відношення. Для суперкласу і кожного з підкласів створюється по одному окремому відношенню.

Метод 8 (гібрид 3 і 4 методів): n + 1 відношення. Для суперкласу і кожного з підкласів створюється по одному окремому відношенню.

Такий широкий вибір можливих варіантів дозволить надалі вибрати найкращий варіант побудови відношень. На основі наведеного вище можна зробити висновок що, чим більш ефективна і гнучка структура відношень використовується для представлення категоризації або спеціалізації, тим складніше поставити для них відповідні реляційні обмеження цілісності (як декларативними, так і процедурними способами).

2.4. Oracle Designer як базова платформа

В якості базової платформи для створення транслятора та сховища ERM-схем була обрана вже наявна CASE-система Oracle Designer. Центральною частиною цієї системи є репозиторій, що містить специфікації проекту на всіх його етапах і забезпечує узгоджену роботу всіх його учасників. Для доступу до сховища і управління ним використовується спеціальний засіб (навігатор по

об'єктах сховища), що дозволяє переглядати і модифікувати об'єкти, що зберігаються в репозиторії, а також здійснювати адміністративні функції: видалення, управління доступом, експорт / імпорт і т. п.

Набір інструментальних засобів Oracle Designer пропонує інтегроване рішення для розробки прикладних систем корпоративного рівня для Web і клієнт / серверних додатків. Oracle Designer бере участь в кожній фазі життєвого циклу розробки ПЗ - від моделювання бізнес-процесів до впровадження. Застосування єдиного сховища робить можливим використання будь-яких його компонент для швидкої розробки масштабованих, крос-платформних розподілених додатків. [12] Завданням Oracle Designer є збір даних про потреби користувачів і автоматизація побудови гнучких графічних додатків. Oracle Designer використовується не тільки для створення додатків, але і для ведення обліку змін, які неминуче відбуваються при експлуатації системи.

Графічні моделі визначень проекту, поєднані з репозиторієм, який розраховано на багато користувачів, істотно полегшують роботу з Oracle Designer. Інструментальні засоби побудовані на базі загальноприйнятих методик, що охоплюють весь життєвий цикл розробки. Це забезпечує гнучкість і відкритість підходу до розробки ПЗ за рахунок використання тільки тих частин продукту, які потрібні в даній задачі. В рамках процесу розробки забезпечується підтримка методів інформаційного проектування, RAD, JAD, водоспадного методу (waterfall), ітеративного методу, а також індивідуального підходу, обраного компанією.

Підвищена складність проектування і розробки прикладних систем є наслідком величезної кількості можливостей при їх реалізації. Ця складність призводить до зростання ризикованості таких проектів. Отже, потрібно такий метод розробки, який дозволяє врахувати всі варіанти, допускає розгортання в альтернативних середовищах (наприклад, клієнт / сервер і / або Web) і полегшує швидкі зміни проекту там, де це необхідно.

Стандартизація зовнішнього вигляду програм викликає певні труднощі при організації розробки. Користувачам потрібні прості, узгоджені і ефективні інтерфейси. Багато організацій розробили керівництва по стилю програмування (style guides), в яких документуються вимоги до зовнішнього вигляду і

характеристикам інтерфейсу користувача (UI), але досягнення бажаного ефекту все ще залежить від доброї практики (або пам'яті) розробників.

Кращим рішенням є інтеграція керівництва по стилю програмування з інструментальними засобами, що формують код програми. В кінцевому підсумку це дозволить розробникам приділяти більше часу на аналіз бізнес-вимог користувачів. [12]

Для вирішення перерахованих проблем Oracle спроектував і розробив унікальний комплект багатокористувацьких засобів для моделювання, проектування і генерації додатки - Oracle Designer.

Для моделювання бізнес-об'єктів використовується комбінація методів побудови діаграм бізнес-процесів, зв'язків сутностей, ієрархії функцій і потоків даних, і їх визначення, завантажені в репозиторій. Ці методи бізнес-моделювання гарантують точність і узгодженість визначень вимог для численних великомасштабних проектів. Оскільки ці моделі використовуються також для отримання ескізного проекту БД і програмних модулів, можна легко простежити і протестувати цілісність розробки.

Oracle Designer один раз визначає і багато разів використовує правила, що керують належною поведінкою і формою відображення даних. Ці правила можна перевизначити під час проектування для будь-яких конкретних цілей, але в будь-якому випадку характеристики, використовувані за замовчуванням в інших додатках, зберігаються. Oracle Designer дозволяє розробникам вбудовувати керівництва по стилю користувальницького інтерфейсу безпосередньо в набір інструментальних засобів генерації програмного коду, використовуючи для цього потужну комбінацію налаштувань (Preferences) і шаблонів (Templates). Налаштування для елементів встановлюються для всієї прикладної системи і багаторазово генеруються для всіх модулів, гарантуючи їх однаковість.

Інструментарій Oracle Designer забезпечує інтегроване рішення для розробки додатків для середовищ Web і клієнт / сервер масштабу підприємства. Oracle Designer бере участь у всіх фазах життєвого циклу розробки ПЗ - від бізнес-моделювання до впровадження. Підхід, в основі якого лежить концепція сховища, робить можливим використання будь або навіть всіх його компонент для швидкої розробки масштабованих, крос-платформних розподілених додатків.

Відомо, що завдання розробки виконуються більш продуктивно і точно, якщо користуватися інструментальними засобами, що працюють на наочній мові діаграм. До числа таких задач можна віднести визначення, модифікацію і розуміння компонент системи і зв'язків між ними. Діаграми разом зі звітами, утилітами і генераторами забезпечують надійне інтегроване середовище для проектування систем. Набір інструментарію для моделювання Oracle Designer забезпечує багатий набір діаграм для підтримки діяльності з проектування і реалізації проекту.

За рахунок надання інструментальних засобів підтримки як об'єктно-орієнтованих моделей моделювання, так і моделей типу "сутність-зв'язок" Oracle забезпечує гнучкий метод бізнес-моделювання. Обидва будівельники діаграм (diagrammers) підтримують стандартні угоди для відповідних стилів моделювання: Уніфікована Мова Моделювання (Unified Modeling Language - UML) підтримується розробником моделей (modeler) об'єктно-орієнтованого типу, а моделювання ER - розробником моделей "сутність-зв'язок".

На основі визначень, що зберігаються в репозиторії, генеруються додатки, які потім розгортаються в середовищах клієнт / сервер або Web. Використовуючи Oracle Developer, на основі визначення модуля або навіть цілого додатки можна отримати систему, яка розгортається в обох середовищах, без зміни яких би то не було частин визначення. Це надзвичайно продуктивний шлях повторного використання визначення додатків.

Oracle Designer надає можливості реінжинірингу і повторної генерації проекту серверної частини як для БД Oracle, так і для інших БД. Це дозволяє провести міграцію БД з "успадкованих" систем безпосередньо в репозиторій Oracle Designer для генерації БД Oracle, здатної повністю використовувати переваги надійного, що масштабується сервера БД.

Аналогічно, ми можемо зробити реінженіринг проекту додатків, побудованих на мові Visual Basic або Developer Reports і Forms Developer, включаючи логіку програми. За умови, що програма побудована в Oracle Developer, використовуючи можливість реінженірінга проекту, ми можемо помістити визначення проекту в репозиторій, внести в них потрібні зміни і повторно згенерувати додаток. Якщо в згенерований додаток за допомогою

Developer внесені якісь зміни, наприклад, додана додаткова бізнес-логіка в формі тригерів PL/SQL, вони також можуть бути визначені і поміщені в репозиторій, і їх не доведеться переписувати в процесі подальшої генерації. Ця здатність змінювати додаток поза рамками Oracle Designer, визначати зміни і повторно генерувати (зберігаючи зміни) відома під назвою "циклічне проектування" (Round-Trip Engineering) і є основним елементом продуктивного середовища проектування та розробки. Якщо ми приймаємо, що додаток змінюється протягом свого життєвого циклу, то здатність підтримувати таке високопродуктивне циклічне проектування є одним з основних переваг використання Oracle Designer.

У Oracle Designer включені засоби для управління вмістом сховища та доступом до нього користувача. Використовуючи засоби управління репозиторієм, можна визначити прикладні системи, які розподіляють об'єкти між численними проектами. Крім того, ці ж засоби використовуються, щоб поширити визначення на численні системи додатків, тим самим сприяючи багаторазового використання об'єктів в середовищах розробки.

Засоби виділення, завантаження і злиття даних сховища підтримують розподілену розробку, при якій можуть використовуватися численні репозиторії Oracle Designer. Визначення з одного сховища завантажуються в інші і узгоджуються, гарантуючи, що розробники, які не мають можливості працювати в команді, можуть все-таки отримати користь з результатів роботи інших розробників. Так само репозиторій служить опорою поняття "мобільна розробка". В рамках цього поняття аналітики або дизайнери системи можуть дистанційно (тобто, не покидаючи своїх постійних робочих місць) працювати з користувачами або клієнтами, щоб спільно визначити вимоги, або прототипи.

Відштовхуючись від створеної моделі зв'язку сутностей (ER), перетворювач проектів БД (Database Design Transformer) може автоматично виконати ескізний проект БД з усіма таблицями, стовпцями, індексами, і обмеженнями цілісності.

Репозиторій Oracle Designer налаштований таким чином, що він має можливість управляти об'єктами, використовуючи власний програмний інтерфейс. Опис нового об'єкта можна ввести в репозиторій за допомогою діалогового інтерфейсу і без потреби в програмуванні. Доступ до нових об'єктів можна отримувати з наявних інструментальних засобів і легко маніпулювати

ними, використовуючи для цього Матричний Діаграмер (Matrix Diagram) або Навігатор Об'єктів Репозиторія (Repository Object Navigator).

Крім усього іншого, система на підставі концептуальних моделей виробляє технічні специфікації майбутньої системи, при цьому початковий варіант специфікацій може бути отриманий автоматично. Для цього застосовуються діаграми схем БД (розширення ER-діаграм), діаграми взаємодій модулів і схеми модулів, що описують структуру модулів з позицій використовуваних в них даних [11]. Основна перевага даного інструменту полягає в тому, що у нього є можливість зміни структури сховища, причому для цього не потрібно писати будь-який плагін - механізм зміни структури сховища працює в діалоговому режимі. Таким чином, цей CASE-засіб краще підходить для досягнення поставленої мети.

Оскільки можливість для користувача налаштування сховища стала однією з основних причин вибору даного CASE-засобу, необхідно детальніше зупинитися на особливості цього механізму.

Розширення користувача (user extensions) - це додаткові властивості або типи, які додаються до існуючого репозиторію [11]. Фактично, будь-який репозиторій має свої метаописи - сукупність описів типів, їх властивостей, а також зв'язків між об'єктами цих типів. Механізм призначеного для користувача розширення Oracle Designer базується на тому, що надає засоби для внесення змін до метаописів свого сховища.

Всі структури даних, що знаходяться в репозиторії, можна розділити на три основні типи.

1. Елементні типи (element types) - типи можливих елементів, які можуть міститися в репозиторії. Стандартними елементними типами OD є функції, модулі, сутності, таблиці, колонки та ін. Кожен елементний тип складається з набору властивостей - дискретних характеристик даного типу.

2. Асоціативні типи (association types) - типи можливих відношень між об'єктами тих чи інших елементних типів. Також, як і елементні типи, вони складаються з набору певних властивостей.

3. Текстові типи (text types) - типізовані описи структури даних конкретних властивостей елементного або асоціативного типу. Стандартними текстовими

типами OD є такі типи як опис, примітка, текст вибору (Select Text) та ін. Текстовий тип являє собою одну область, що містить кілька рядків з текстом.

Кожен тип сховища можна описати як комбінацію властивостей і способу використання текстового типу. Спосіб використання текстового типу вказує на вид тексту, що зберігається для елемента або асоціації [11].

На рисунку 2.1 представлена діаграма, що описує зв'язку структурних елементів сховища.

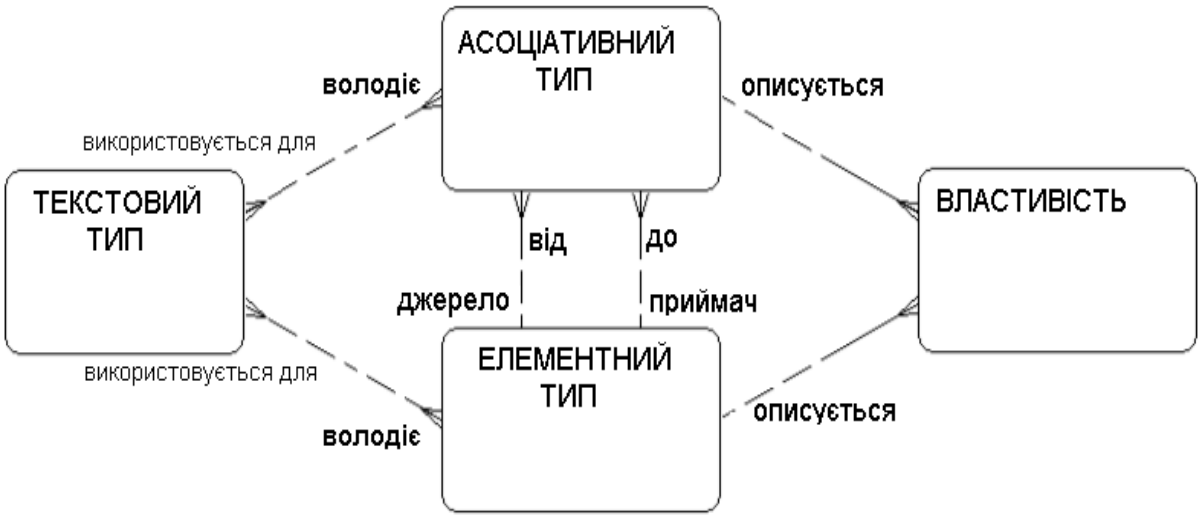


Рис. 2.1. ER-діаграма, яка описує зв'язки структурних елементів сховища OD

Робота з користувацькими розширеннями здійснюється за допомогою інструменту Repository Administration Utility (RAU), що входить в пакет даного CASE-засобу. До його складу входять три утиліти для роботи з розширеннями користувача:

– Maintain User Extensions (утиліта для редагування структури сховища);

– Load User Extensions (утиліта для розгортання структури сховища з файлу дампа (з розширенням .dmp) або спеціального текстового файлу);

– Extract User Extension (утиліта для створення резервної копії структури сховища у вигляді файлу дампа або спеціального текстового файлу).

Робота зі створення елементних, асоціативних і текстових типів відбувається в діалоговому режимі. Для того щоб створити користувальницький тип, необхідно вказати його обов'язкові властивості, а також (крім текстових типів) створити набір його атрибутів.

Для створення елементного типу необхідно заповнити набір його імен. Всього їх п'ять: коротке ім'я, звичайне ім'я, звичайне ім'я в множині, що відображається в системі ім'я, що відображається в системі ім'я у множині.

Для створення асоціативного типу також необхідно заповнити набір з п'яти його імен, а також вказати два елементних типу - учасника даного відношення. Крім самих учасників вказуються і їх максимальні кардинальні числа в даному відношенні («One» або «Many»). Слід зазначити, що відсутність можливості задання точних цілочисельних максимальних кардинальних чисел, а також мінімальних кардинальних чисел є серйозним упущенням розробників даного CASE -засобу, оскільки в разі потреби задання таких обмежень їх доведеться реалізовувати програмно. Для створення текстового типу необхідно заповнити його коротку назву, а також текстовий опис, що дає інформацію про цей тип.

На малюнку 2.2 зображено фрагменти інтерфейсу утиліти Maintain User Extensions.

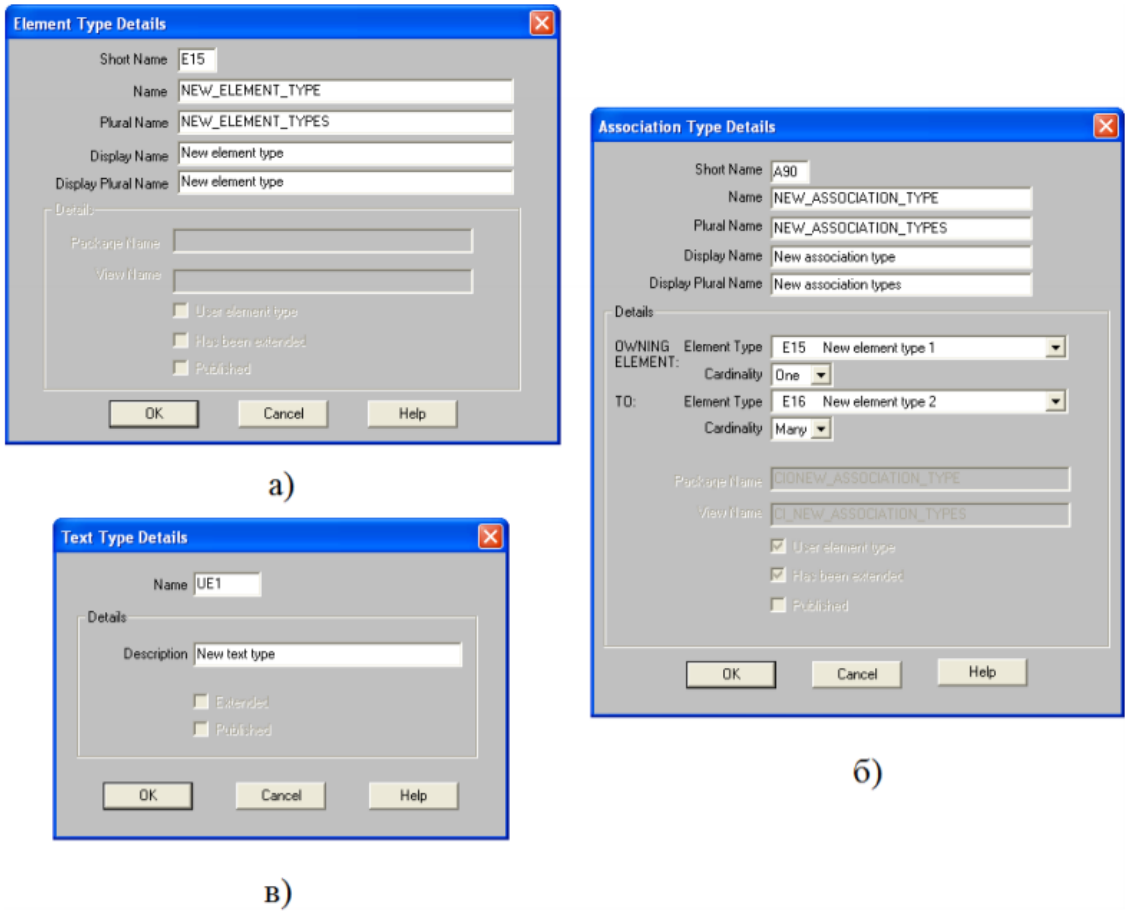
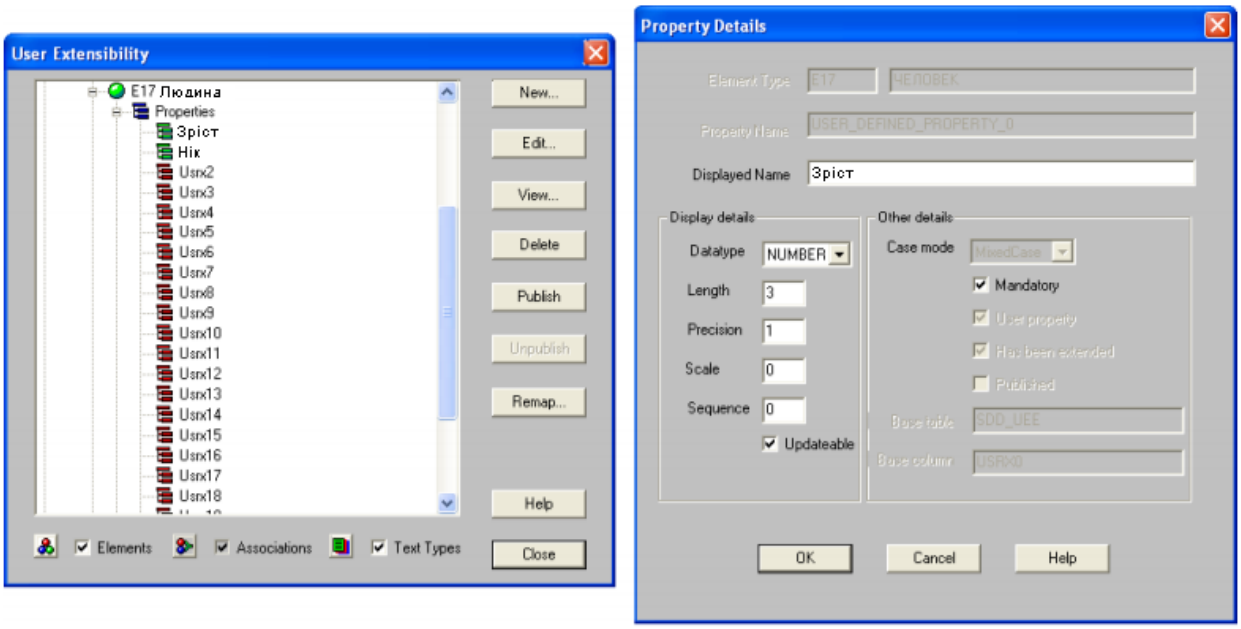


Рис. 2.2 . Фрагменти інтерфейсу утиліти Maintain User Extensions, що ілюструють задання основних властивостей при створенні: а) елементного типу; б) асоціативного типу; в) текстового типу

Для створення текстових типів більше нічого не потрібно. Для завершення створення елементного або асоціативного типів потрібно створити для них групи атрибутів. Взагалі, в інструменті вони позначаються як Properties, але для того щоб не плутати їх з основними властивостями призначених для користувача типів, будемо називати їх атрибутами. Для кожного елементного або асоціативного типу спочатку виділено 20 незаповнених атрибутів. Спочатку вони мають імена «UsxrN», де N - число від 0 до 19. Як атрибути задаються особливості, що описують конкретні екземпляри елементного або асоціативного типу. Для цього необхідну кількість незаповнених атрибутів іменується згідно назвам цих особливостей, а також задаються їх типи даних.

На малюнку 2.3 представлені фрагменти інтерфейсу утиліти Maintain User Extensions, що ілюструють завдання атрибутів на прикладі елементного типу «Людина».

Крім імені та типу, для атрибута можна вказати його обов'язковість, а також можливість зміни значення.



а) б)

Рис. 2.3. Фрагменти інтерфейсу утиліти Maintain User Extensions, що ілюструють завдання атрибутів на прикладі елементного типу «Людина»: а) структура атрибутів елементного типу; б) опис конкретного атрибута

Після того, як призначені для користувача розширення створені, необхідно перебудувати репозиторій за допомогою утиліти Recreate інструменту Repository Administration Utility.

Після цього призначені для користувача розширення відразу доступні в системі. Роботу з ними можна здійснювати за допомогою діалогової утиліти Repository Object Navigator. Дана утиліта не потребує змін для роботи з новими типами призначених для користувача розширень, що є ще одним безперечною перевагою системи Oracle Designer.

Ще одним плюсом CASE-системи Oracle Designer є той факт, що при створенні призначених для користувача розширень автоматично генерується механізм доступу до їх екземплярів. У даній CASE-системі ці механізми носять назву API. API Oracle Designer - це набір представлень БД пакетів PL/SQL в схемі власника сховища, які забезпечують можливість безпечного доступу до даних сховища. Сховище складається з відносно невеликого числа таблиць, в яких містяться реальні дані. Між цими таблицями існують складні (недокументовані) відносини, і Oracle не підтримує прямого доступу до них через стандартні SQL-оператори мови маніпулювання даними. Однак існує множина представлень цих таблиць, в яких відображаються реальні об'єкти сховища. Їх стовпці майже повністю відповідають властивостям того елемента, який характеризується цим поданням [11].

Крім представлень, API містить пакети PL/SQL, що дозволяють безпечно змінювати вміст таблиць із засобу, зовнішнього по відношенню до Oracle Designer. З допомогою цих пакетів можна доповнити утиліти Oracle Designer власними програмами або програмними конструкціями [11]. Цей факт дозволяє говорити про те, що API Oracle Designer доступний для редагування, що в подальшому дозволить впровадити механізм перевірки на несуперечливість ERM-схем. Варто відзначити, що пакети API -це єдиний спосіб запису інформації в репозиторій, і всі інструментальні засоби використовують їх незалежно від того, чи входять ці кошти до складу Oracle Designer чи ні [11].

На рисунку 2.4 зображена концептуальна діаграма компонентів, яка ілюструє принцип роботи зовнішніх інструментів з репозиторієм Oracle Designer.

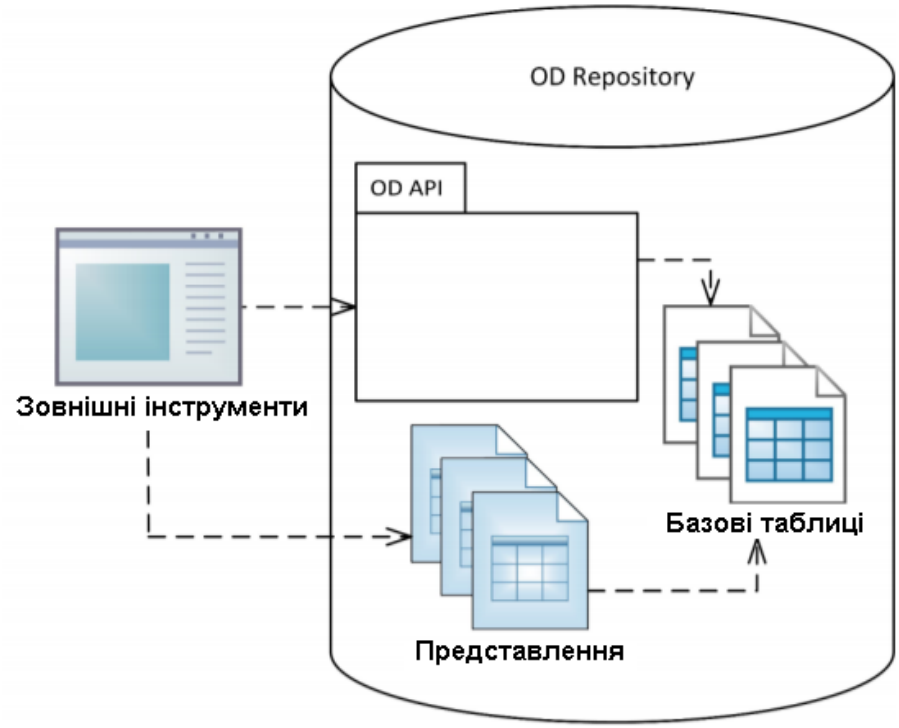


Рис. 2.4. Концептуальна діаграма компонентів, що ілюструє принцип роботи зовнішніх інструментів з репозиторієм Oracle Designer

Для отримання даних, що зберігаються в репозиторії, зовнішні інструменти звертаються до представлення. Для додавання, зміни і видалення даних інструменти звертаються до відповідних функцій API Oracle Designer.

API включає в себе більше 400 пакетів PL / SQL. Вони організовані подібно до представлень, тому для кожного користувача типу сховища існує свій пакет. У кожному пакеті містяться процедури з наступними іменами:

– INS - для операцій введення;

– UPD - для операцій оновлення;

– DEL - для операцій видалення;

– SEL - для операцій вибірки [10].

Кожна функція являє мінімальну транзакцію роботи з репозиторієм.

Функція або відпрацьовує повністю, або проводиться її повний відкат. В результаті в будь-який момент часу репозиторій зберігає несуперечливий стан.

Таким чином, розглянувши особливості механізму призначених для користувача розширень, можна зробити наступні висновки:

– великою перевагою даного CASE-засобу є можливість діалогової зміни структури сховища. Це дозволяє значно скоротити час і трудомісткість розробки;

– відпадає необхідність реалізації механізму доступу до об'єктів, оскільки при створенні призначених для користувача розширень автоматично створюється API для роботи з їх екземплярами. З тієї ж причини відпадає необхідність обробки транзакцій при роботі з системою;

– відпадає необхідність реалізації діалогового інструменту для роботи з репозиторієм, він надається вже в готовому вигляді;

– з огляду на те, що механізм доступу до об'єктів є PL/SQL-пакетами процедур і функцій, він може бути підданий виправленню. Це дозволить локалізувати функціонал перевірки на несуперечливість у вигляді окремого компонента, а виклик перевірки впровадити в самі функції механізму доступу;

– робота зі структурними елементами ERM-схем має досить складну структуру, оскільки зміни одних елементів ERM-моделі можуть породжувати зміни інших її елементів. Отже, логіку залежностей між структурними елементами слід локалізувати і, бажано, на стороні сховища (а не клієнтів). Таким чином, з'являється ще одна задача - реалізація компонента доступу, локалізується логіка залежностей між структурними елементами ERM-моделі і представляє собою надбудову над стандартним API Oracle Designer.

Переваги Oracle Designer є наслідком чіткого розмежування між компонентами архітектури додатку, які описуються і зберігаються в єдиному інтегрованому середовищі, або репозиторії. Такий поділ компонент проекту дозволяє швидко створювати і модифікувати системи з чітким розумінням взаємозалежностей між компонентами цієї системи і аналізувати ймовірні впливи на змінив, які в ній відбуваються.

2.5. Механізм доступу до об’єктів ERM-схем

При створенні призначених для користувача розширень автоматично створюється механізм доступу до об'єктів створюваних типів. Даний механізм використовується в тому числі і для роботи з об'єктами через інструмент RON.

Одна з вимог, полягає в тому, що повинна бути забезпечена можливість роботи з репозиторієм за допомогою ряду інших, зовнішніх додатків. Зокрема, це графічний редактор ERM-схем, генератор реляційних схем тощо. Теоретично,

дані програми можуть працювати і за допомогою API , що генерується, проте є ряд недоліків, що виникають при цьому підході.

Перший недолік полягає в тому, що API, яке генерується, має досить складну структуру. Перед здійсненням безпосередньої операції над об'єктом необхідно здійснити ряд допоміжних дій, таких як вказівка поточного контексту, ініціалізація сесії, перевірка транзакції і обробка помилок. Всі ці дії повинні будуть проводитися виключно зовнішнім додатком [13]. До того ж представлення, через які читаються дані про об'єкти, мають досить складні й заплутані зв'язки з описами прикладних систем, в рамках яких вони створені. Всі ці фактори суттєво ускладнять процес розробки нового або інтеграції вже існуючого стороннього додатка. Тривалість процесу розробки такого додатка також буде збільшена.

Другий недолік полягає в тому, що логіка роботи з деякими структурними сутностями ERM-моделі виходить далеко за рамки простого додавання, зміни або видалення будь-якого об'єкта системи. Наприклад, зв'язування множин сутностей із множиною зв'язку з точки зору графічної нотації ERM-моделі супроводжується з'єднанням відповідних структурних сутностей. Однак в репозиторії для кожної множини сутностей буде створена роль, яка буде зберігати мінімальні і максимальні числа зв'язку. Якщо зовнішній додаток буде використовувати API, що генерується, відповідальність за створення і своєчасне видалення цих ролей ляже саме на нього. Таким чином, логіка роботи зі структурними об'єктами ERM-моделі буде істотно ускладнена і розбита за кількома архітектурними шарами.

Третій недолік полягає в тому, що в умовах, коли логіка залежностей між структурними елементами ERM-моделі розподілена за різними шарами системи, істотно ускладнюється можливість внесення в цю логіку будь-яких коректив. При зміні логіки будь-якого структурного елементу ERM-моделі відповідним змінам піддасться його обробка як на рівні сховища, так і на рівні зовнішнього застосування, що ускладнить процес підтримки системи.

З огляду на вищесказане, пропонується відокремити логіку роботи зі структурними сутностями моделі в окремий пакет процедур і функцій, які здійснюють додавання, зміни, видалення і вибірку об'єктів ERM-моделі.

Механізм доступу локалізує в собі звернення до сховища через API. Кожен виклик тієї чи іншої процедури або функції механізму породжує один або більше

викликів згенерованого API, попередню підготовку до них, а також подальшу обробку і повернення результатів. Для реалізації такого механізму необхідно було дослідити принципи роботи з генеруються API. Виклик процедури або функції API супроводжується послідовністю дій, представленій в таблиці 2.1

Таблиця 2.1

Послідовність дій при зверненні до API Oracle Designer

№ з/р	Дія
1	ініціалізація - оголошується, яка прикладна система (і її версія) використовується
2	встановлення початку транзакції
3	завантаження значень змінної запису - в змінну, що є одним з аргументів функцій API, вводяться значення, що представляють потенційні зміни даних сховища
4	завантаження покажчиків змінної запису - вказується факт заповнення того чи іншого поля змінної запису
5	виконання операції - виклик процедури або функції згенерованого API
6	перевірка успішності операції
7	підтвердження змін і закриття транзакції - в разі помилки відбувається повний відкат транзакції

Виконання послідовності з таблиці 2.1 є обов'язковим. В умовах відсутності ініціалізації прикладної системи або відкриття транзакції виклик функцій API неможливий.

Конструктивно механізм доступу є пакетом процедур і функцій. Для кожного типу структурних сутностей моделі є 4 обов'язкові операції - додавання, зміна, видалення і вибірка даних.

Також для деяких типів структурних об'єктів створені операції, що надають доступ до додаткової інформації, пов'язаної з об'єктом (наприклад, для операції над класами - список класів-операндів і ін.). Також в присутній ряд процедур, які локалізують звернення до згенерованої API.

На рисунку 2.5 зображена концептуальна діаграма компонентів, що ілюструє розташування даного пакета в розробці.

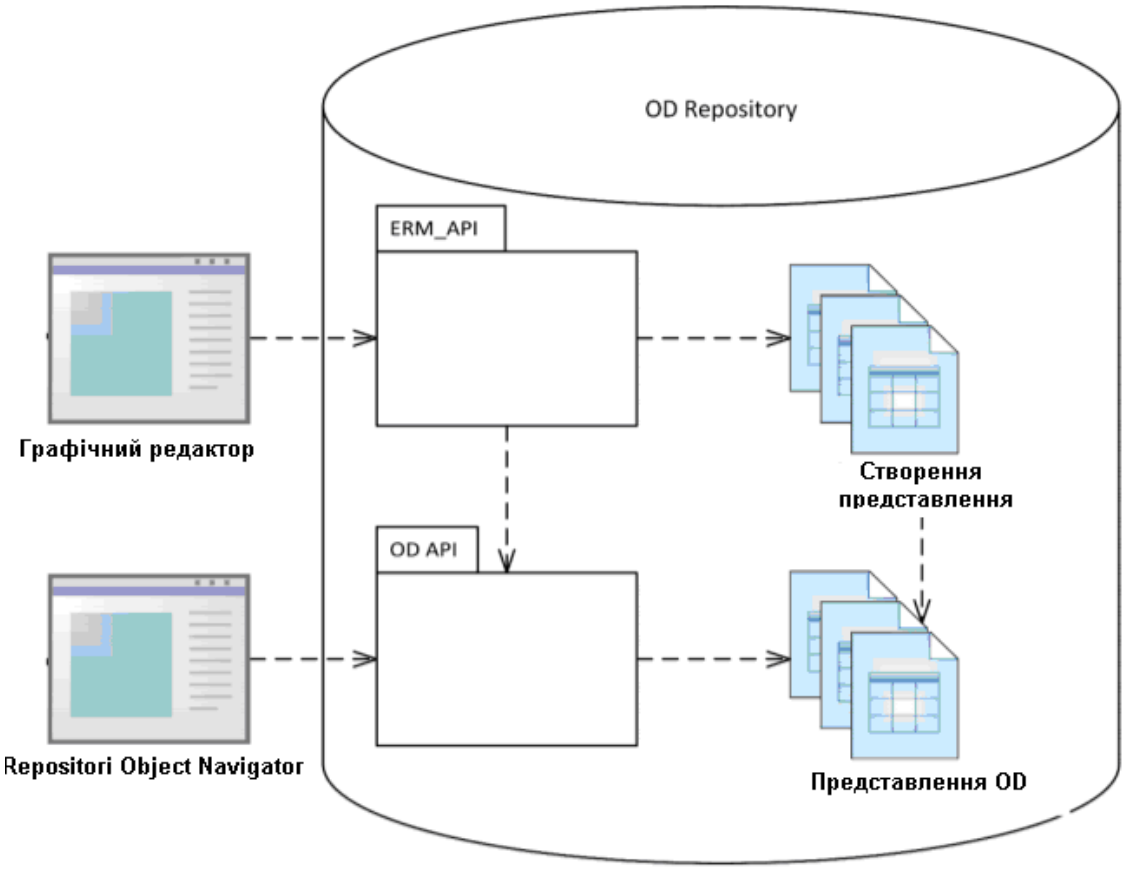


Рис.2.5. Концептуальна діаграма компонентів

2.6. Висновки до розділу

У цьому розділі наведено загальні відомості про методику трансформації ERM-схеми в реляційну схему, проаналізовані технології ручної та автоматичної трансформації. Описано CASE-систему Oracle Designer як базову платформу для проведення дослідження та механізм доступу до об’єктів ERM-схем.

РОЗДІЛ 3
ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ЗАСОБУ ДЛЯ ТРАНСФОРМАЦІЇ ERM-СХЕМИ В РЕЛЯЦІЙНУ СХЕМУ ТА ЗБЕРЕЖЕННЯ В РЕПОЗИТОРІЇ

3.1. Програмна архітектура засобу

3.1.1 Структура проектного модуля.

Програмним середовищем розробки засобу трансформації була обрано IDE Microsoft Visual Studio .NET. Корпорація Microsoft в даний час серйозно займається просуванням і популяризацією .NET, Тому розробка додатків в середовищі .NET є перспективним напрямком, і написаний код буде придатний для повторного використання. Базовою мовою програмування було обрано C #, яка є найбільш зручною мовою для розробки призначених для користувача додатків і при цьому досить простою для роботи не тільки з графікою, але і з зберіганням даних.

Програмна архітектура засобу наведена на рисунку 3.1. Сам засіб представлений пакетом TransformationTools, який пізніше набуде вигляду динамічної бібліотеки. Даний пакет містить класи, необхідні для трансляції ERM-схеми в реляційну схему.

Модуль в своїй роботі використовує три бібліотеки:

– RelationSchemeTools - пакет для класів, що забезпечують реалізацію опису реляційної схеми;

– IRelationSchemeTools - пакет для інтерфейсів, які повинні підтримувати класи реляційної схеми. Надає можливість для подальшого розширення класів реляційної схеми;

– ERM.Meta - пакет для класів, опису ERM-схеми.

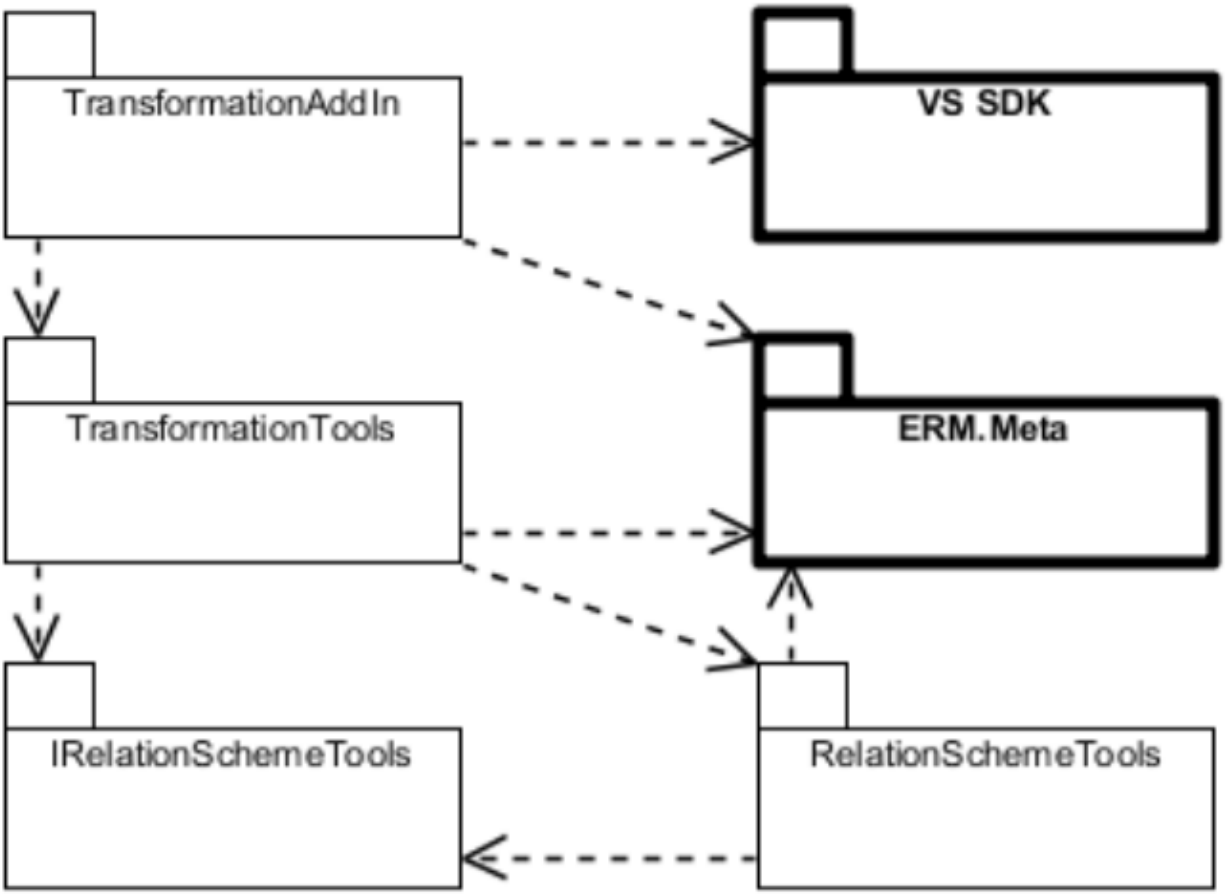


Рис. 3.1. Програмна архітектура засобу

Для інтеграції модуля в IDE MS Visual Studio був розроблений плагін, який представлений пакетом TransformationAddin. Плагін створюється з використанням пакета VS SDK - бібліотеки компонентів розширення середовища Visual Studio.

3.1.2. Об'єктне подання реляційної моделі. Діаграма ієрархії класів, які використовуються для подання реляційної моделі, наведена на рисунку 3.2 (операції класів для наочності опущені)

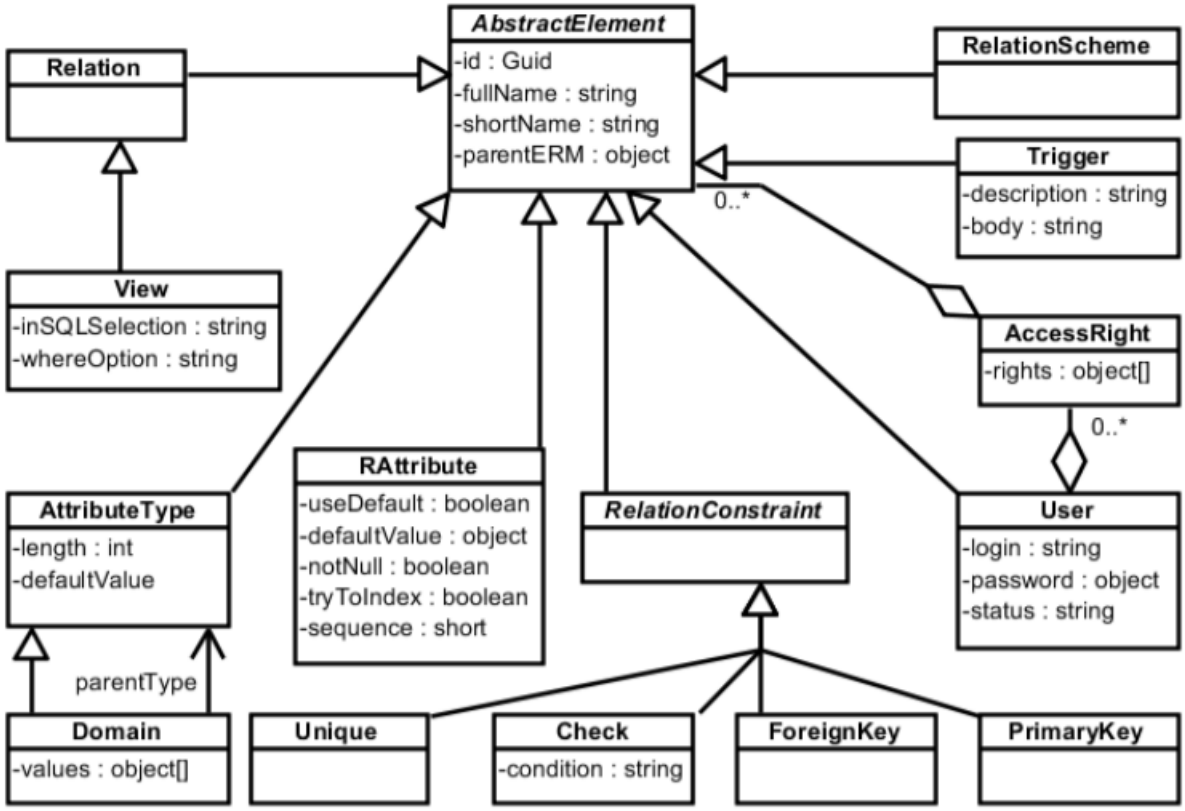


Рис. 3.2. Діаграма класів реляційної схеми, що ілюструє ієрархію класів

Варто навести опис класів, які використовуються:

– User - клас, що описує конкретного користувача в системі. Клас в відповідних полях зберігає пароль і статус перебування користувача в системі, його особисту схему, а так само всі права доступу до об'єктів;

– AccessRight - клас, що описує права доступу на кожен об'єкт в системі.

Класи, що відповідають за подання реляційної схеми в оперативній пам'яті:

– RelationScheme - клас, що описує реляційну схему, який зберігає всі відношення та подання. Містить в собі методи завантаження і збереження її в файл, або динамічний потік, що дозволяє його передавати по мережі;

– Relation - клас, що описує відношення, зберігає в собі атрибути і обмеження цілісності;

– View - клас, що описує представлення, зберігає в собі посилання на відношення або SQL-запит, який відтворює необхідне представлення. Містить операцію, яка обробляє SQL-запит, заповнюючи поля представлення;

– RAttribute - клас, що описує атрибут відношення, який містить в собі поля необов'язковості атрибута, початкового значення;

– Domain - клас, що описує домен, містить в собі поле, що містить посилання або на ще один домен, або на тип атрибута, а також опис допустимих значень, що накладаються на даний домен;

– AttributeType - клас, що описує тип атрибута, містить в собі числове поле, кодує тип. Надалі воно використовується для визначення потрібного типу, реалізованого для конкретної СУБД;

– RelationConstraint - клас, що описує обмеження цілісності в цілому, містить поле посилань на атрибути, на які накладається обмеження;

– PrimaryKey - клас обмеження, що описує первинний ключ відношення;

– ForeignKey - клас обмеження, що описує зовнішній ключ відношення, містить поле на зовнішнє відношення, а так само поле посилань на атрибути зовнішнього відношення;

– Unique - клас обмеження, що описує умову унікального значення атрибута;

– Check - клас обмеження, що описує конструкцію умови, що накладається на атрибут. Містить текстове поле для зберігання інформації про умови обмеження;

– Trigger - клас, який реалізує зберігання опису процедурної реалізації обмеження цілісності.

Так як для роботи системи може знадобитися зміна створених класів, то для взаємодії елементів схеми один з одним були використані інтерфейси, що дозволяє гнучко змінювати класи під конкретну задачу.

Взаємозв'язок класів в системі можна побачити на діаграмі класів, зображеній на рисунку 3.3 (поля і операції класів для наочності опущені).

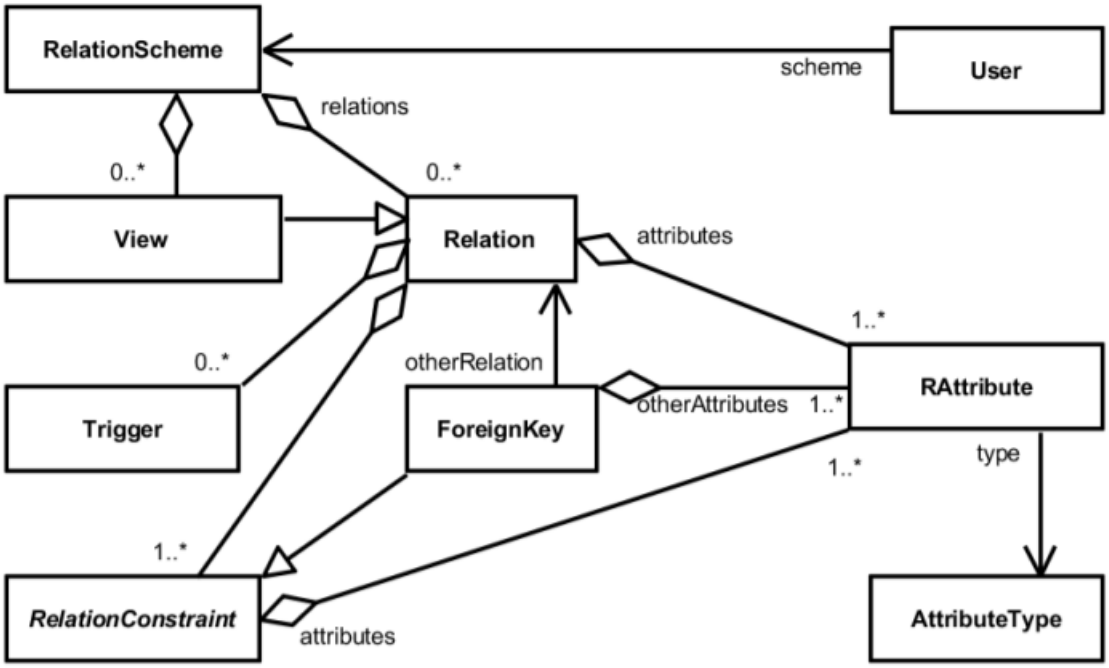


Рис. 3.3. Діаграма класів реляційної схеми, що ілюструє залежність класів

3.2 Модуль трансляції ERM-схеми в реляційну схему

Діаграма класів, яка використовується для трансляції ERM-схеми в реляційну схему, наведена на рис. 3.4.

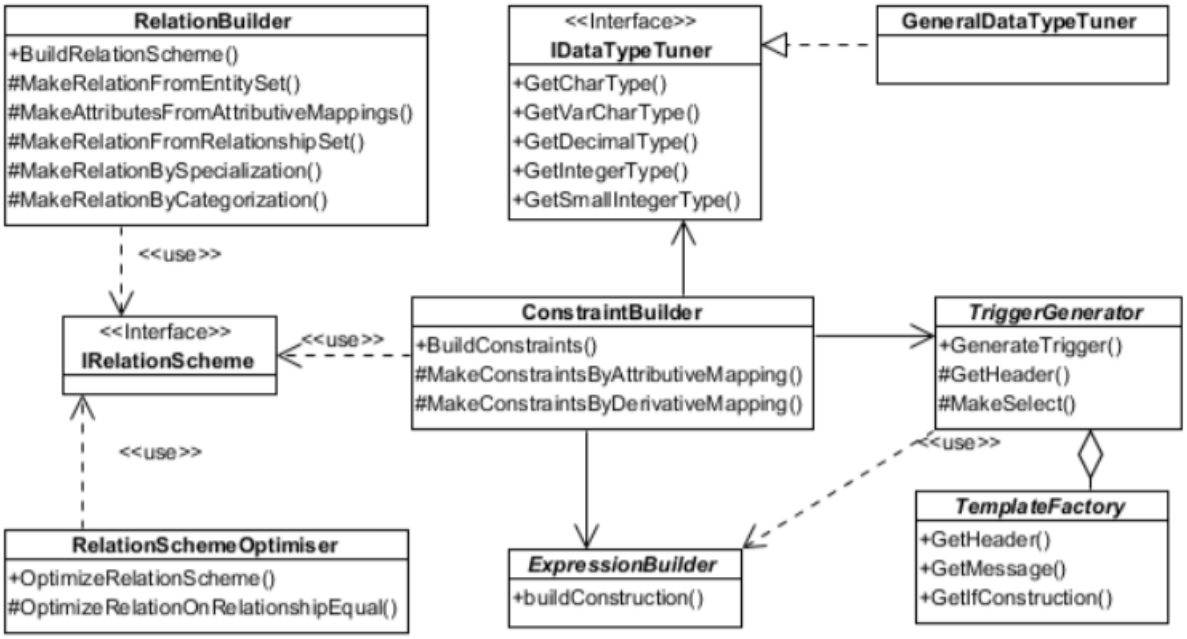


Рис. 3.4. Діаграма класів, які використовуються для трансляції схем

Опис класів, які використовуються:

– RelationBuilder - клас, який реалізує перший етап перетворення, використовуючи правила породження відношень. Містить основну операцію BuildRelationScheme () , що відповідає за перетворення схем, на вхід якої надходить ERM-схема і нова реляційна схема користувача. Операція послідовно викликає внутрішні операції, які реалізують правила 1 – 5 методики трансформації (див. п. 2.2). В операції MakeRelationFromEntitySet () переглядаються всі множини сутностей для конструювання відповідних відношень, для кожного нового відношення викликається операція MakeAttributesFromAttributiveMappings () , яка будує все атрибути (або відношення у випадку багатозначної характеристики) для даного відношення. Далі викликається операція MakeRelationFromRelationshipSet () , в якій проводиться перегляд множини зв'язків з подальшим конструюванням відповідних відношень, після чого також викликається операція MakeAttributesFromAttributiveMappings () . Далі послідовно викликаються операції MakeRelationBySpecialization () і MakeRelationByCategorization () , що відповідають за коригування відносин, отриманих на основі класів що беруть участь в спеціалізації або категоризації;

– IDataTypeTuner – інтерфейс класу, який потрібно реалізувати, відповідає за налаштування типів даних під конкретну СУБД, так як типи даних можуть помітно відрізнятися або ж зовсім не підтримуватися;

– GeneralDataTypeTuner - клас, який реалізує інтерфейс IDataTypeTuner, відповідає за загальне налаштування типів даних, прийнятих в стандарті SQL89, конструюючи відповідні об'єкти класу AttributeType;

– TriggerGenerator - абстрактний клас, який необхідний для розбору конструкцій відображень і генерації відповідного тригера. Містить основну операцію GenerateTrigger () , на вхід якого надходить конструкція відображень ERM-схеми, всередині операції можуть бути викликані операції GetHeader () або MakeSelect ();

– TemplateFactory - абстрактний клас, реалізує шаблон «Абстрактна фабрика», який необхідний для реалізації конкретної фабрики, що містить шаблони для генерації тригерів під конкретну СУБД;

– ExpressionBuilder - абстрактний клас, який реалізує в операції buildConstruction () побудову рядка, що містить вираз з посиланнями на потрібні елементи;

– ConstraintBuilder - клас, який реалізує другий етап перетворення, використовуючи правила породження обмеження цілісності. Містить основну операцію BuildConstraints () , на вхід якої надходить ERM-схема і реляційна схема користувача, отримана після першого етапу. Операція послідовно викликає внутрішні операції, які реалізують правила 6 - 9 методики трансформації. Так само клас містить поле tuner , що відповідає за реалізацію правила 6, в якому переносяться всі обмеження на тип атрибута з ERM-схеми, за допомогою виклику операції MakeConstraintsByAttributiveMapping () ;

– RelationSchemeOptimiser - клас, який реалізує третій етап перетворення, використовуючи оптимізаційні правила. Містить основну операцію OptimizeRelationScheme () , на вхід якої надходить реляційна схема користувача, отримана після другого етапу. Дана операція викликає внутрішню операцію OptimizeRelationOnRelationshipEqual () , яка частково реалізує правило 10 і 11. Потрібен подальший розвиток цього класу.

Послідовність роботи модуля представлена на діаграмі послідовності (рисунок 3.5).

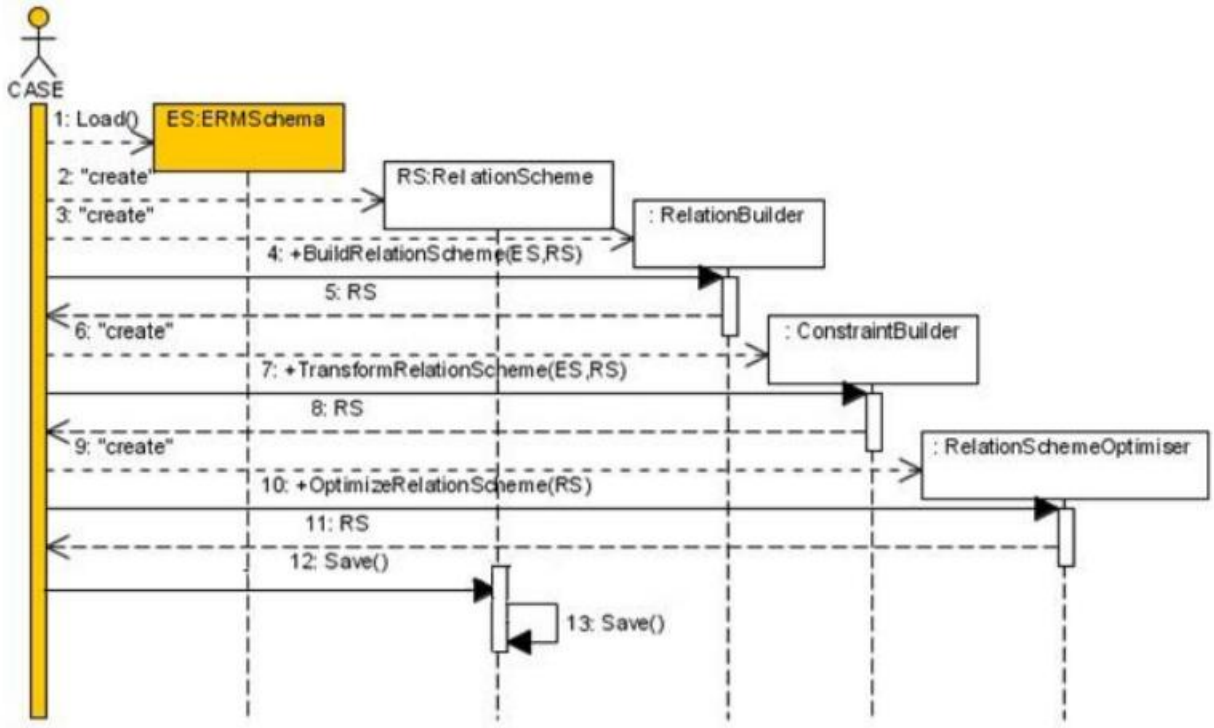


Рис. 3.5. Діаграма послідовності роботи модуля трансляції

На даній діаграмі видно, що спочатку потрібно завантажити ERM-схему, після чого створити порожню реляційну схему, яка і буде використовуватися як основа для побудови.

Наступним кроком створюється RelationBuilder і викликається метод побудови відношення, де йому в якості параметрів передається завантажена ERM-схема і порожня реляційна схема, після чого повертає реляційну схему з побудованими відношеннями.

Другим етапом створюється ConstraintBuilder, що налаштовується під потрібну СУБД, і викликається метод для побудови обмежень цілісності, якому в якості параметрів також передається ERM-схема і реляційна схема. Метод повертає повну реляційну схему, можливо з надмірністю.

Третій етап трансляції завершується створенням RelationSchemeOptimiser і викликом методу, якому передається отримана повна реляційна схема. Метод виконує усунення надмірності в реляційній схемі.

Останній етап – збереження отриманої реляційної схеми.

У підсумку, після роботи модуля, виходить повністю побудована реляційна схема, на основі якої можлива генерація SQL-скриптів.

3.3 Практична реалізація трансляції схем

Для інтеграції в середовище розробки Visual Studio і візуального представлення результатів роботи спроектованого модуля, був розроблений плагін, який надає інтерфейси взаємодії з CASE-системою і середовищем розробки. [14]

Інтерфейс плагіна представлений у вигляді кнопки в меню інструментів і вікном візуалізації, показаним на рисунку 3.6.

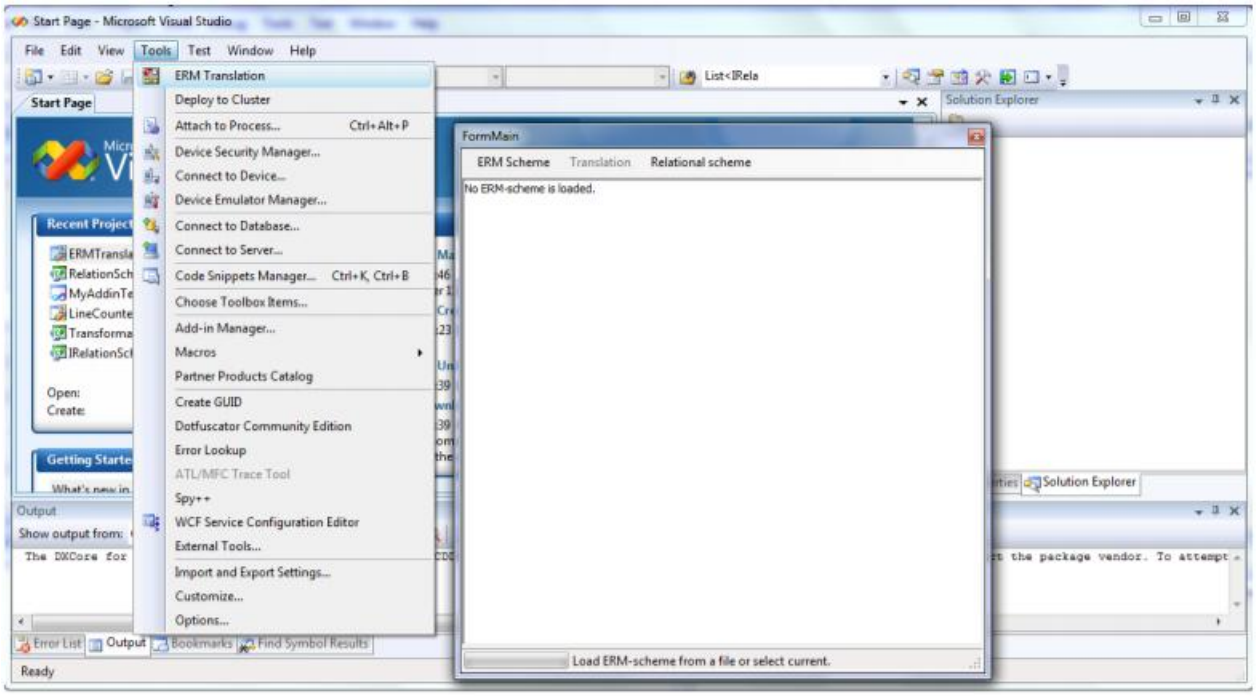


Рис. 3.6. Інтерфейс плагіна-інструменту в середовищі Visual Studio

Інтерфейс вікна плагіна простий, але надає основні можливості, які використовуються в роботі з модулем. З усіх функцій інтерфейсу можна виділити кілька основних:

– вибір необхідної ERM-схеми для завантаження;

– запуск процесу трансляції після натискання кнопки;

– візуалізація побудованої реляційної схеми у вигляді тексту;

– збереження отриманої реляційної схеми в файл.

При роботі з плагіном першим етапом відбувається вибір необхідної ERM-схеми в вигляді файлу з розширенням .schm через діалог, що надається ОС, або ж викликом контекстного меню на діаграмі, після чого відбувається завантаження і ініціалізація ERM-схеми з файлу. Крім цього створюється нова порожня реляційна схема.

Роботу додатка після завантаження ERM-схеми можна простежити на діаграмі станів, представленій на рисунку 3.7.

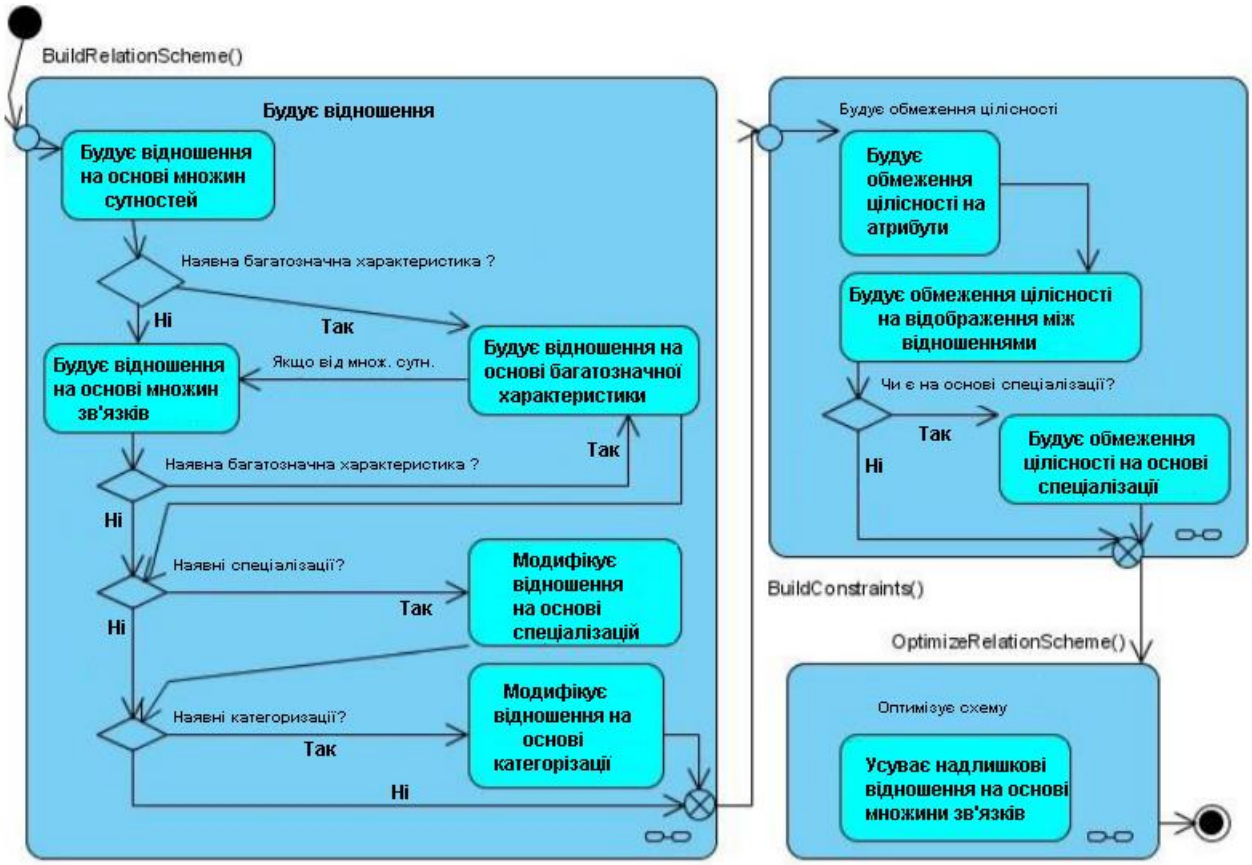


Рис. 3.7. Діаграма станів додатку

На діаграмі видно, як додаток будує на основі ERM-схеми реляційну, проходячи три основних етапи трансляції. Після завершення процесу трансляції отримана реляційна схема виводиться вікно у вигляді тексту, в якому детально описується побудована схема з усіма відношеннями і обмеженнями цілісності. Після цього побудовану реляційну схему можна зберегти в файл.

Результати роботи модуля можна розглянути на прикладі. Спочатку потрібно побудувати ERM-схему в CASE-засобі або ж завантажити наявну з файлу. Після завершення формування схеми в пам'яті, за запитом користувача може бути запущений процес трансляції, де вона передається модулю для подальшої трансляції в реляційну схему. На виході модуля утворюється реляційна схема з усіма відношеннями і обмеженнями цілісності. Результат трансляції можна побачити на рисунку 3.8.

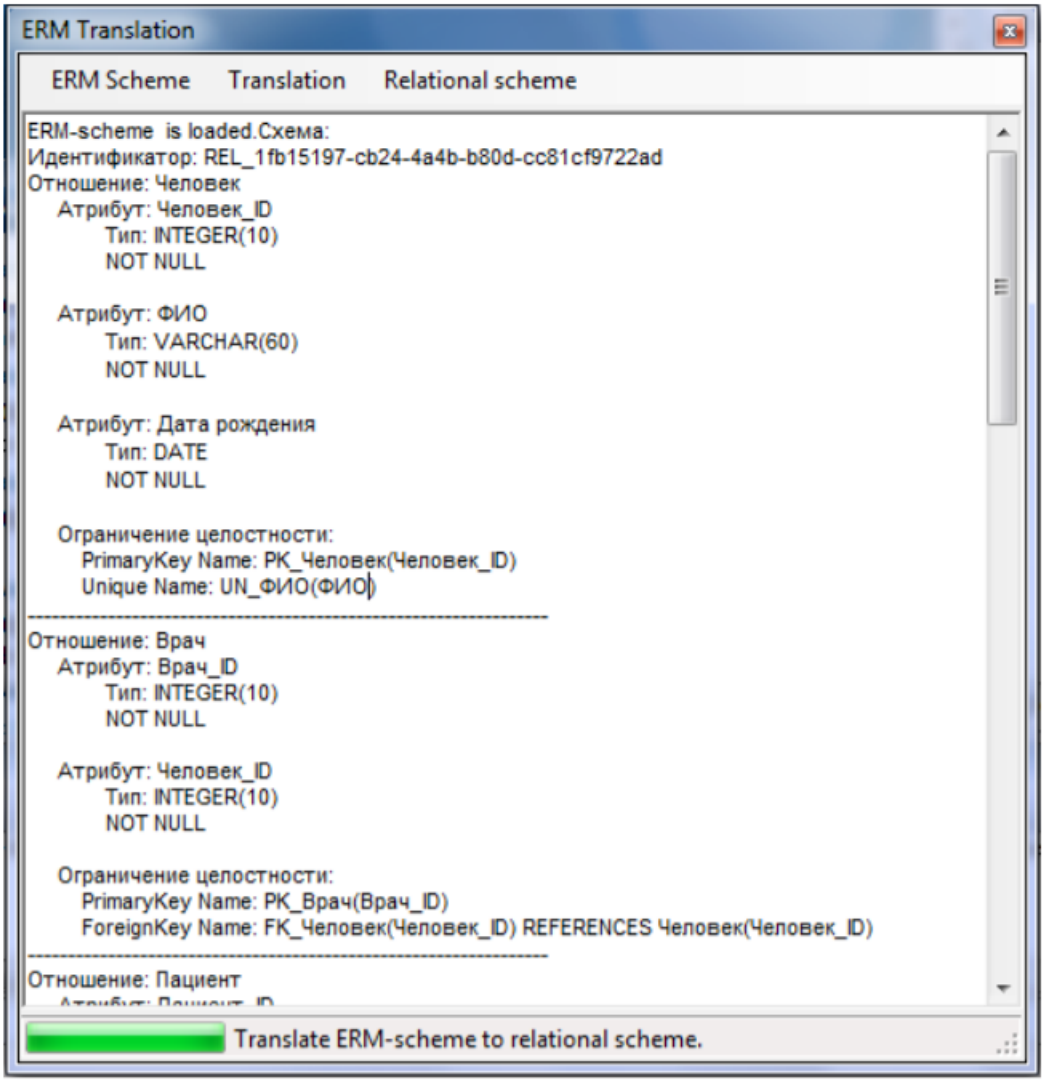


Рис. 3.8. Результат роботи алгоритму трансляції

За відображеними результатами роботи , можна переконатися, що були побудовані відношення, атрибути, а також обмеження цілісності, що накладаються на них.

3.4 Встановлення плагіну у Visual Studio

Перед початком процесу необхідно переконатися, що на комп’ютері попередньо встановлена Visual Studio не нижче версії 2008 і пакети .NET Framework не нижче версії 3.5.

Спершу Відкрийте подвійним кліком файл ERMTranslationAddinSetup, дочекайтеся появи вікна установки (рис. 3.9).

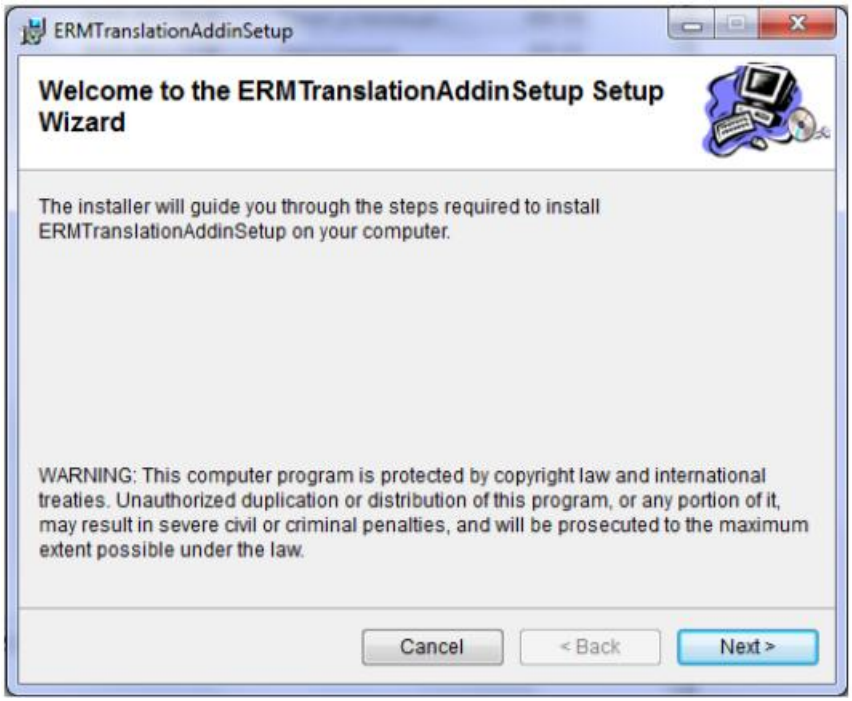


Рис. 3.9. Вікно встановлення ERMTranslationAddinSetup

Потім необхідно натиснути кнопку Next. Далі з’явиться вікно вибору шляху установки (рис. 3.10).

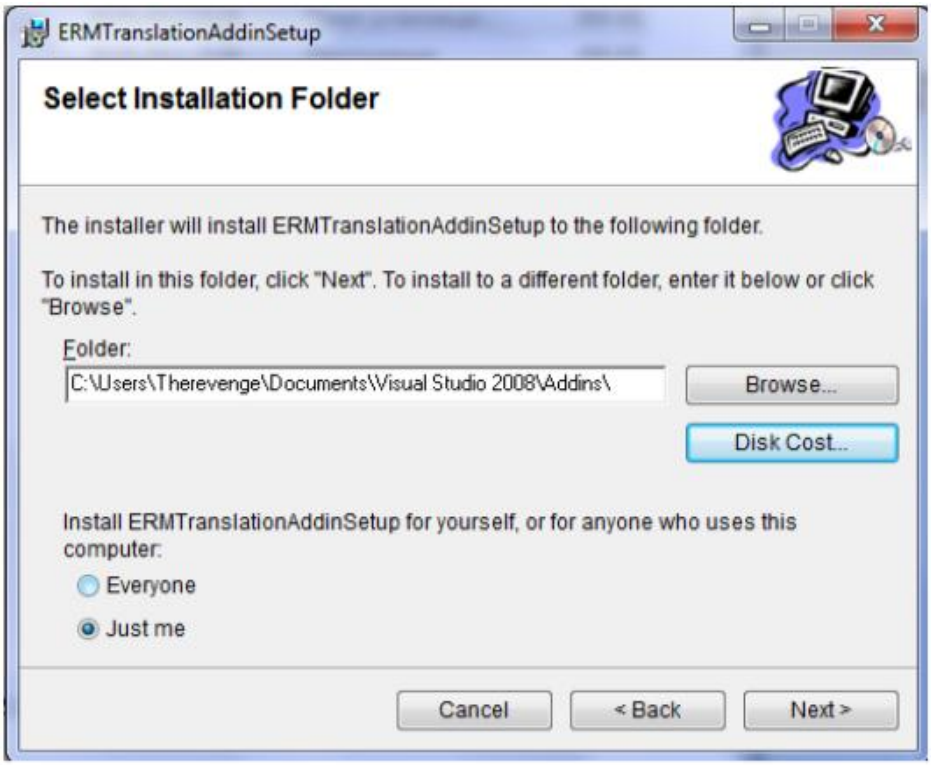


Рис. 3.9. Вікно вибору шляху установки

Потім повно натиснути кнопку «Browse ..», після чого необхідно вибрати у вікні відповідний шлях до папки Addins вашої встановленої Visual Studio. Папка Visual Studio XXXX може лежати не в папці Documents поточного користувача, а в папці Commons. Переконайтеся в правильності шляху до папки Addins, інакше плагін не зможе завантажитися в середовище розробки!

Натисніть кнопку Next і ще раз Next для того, щоб почати установку плагіна. Після завершення встановлення буде виведено вікно (рис. 3.10), в якому потрібно буде натиснути кнопку Close.

Необхідно зауважити, що під час установки на ОС Windows Vista або 7 буде запит на права адміністратора, потрібно буде дозволити їх застосувати.

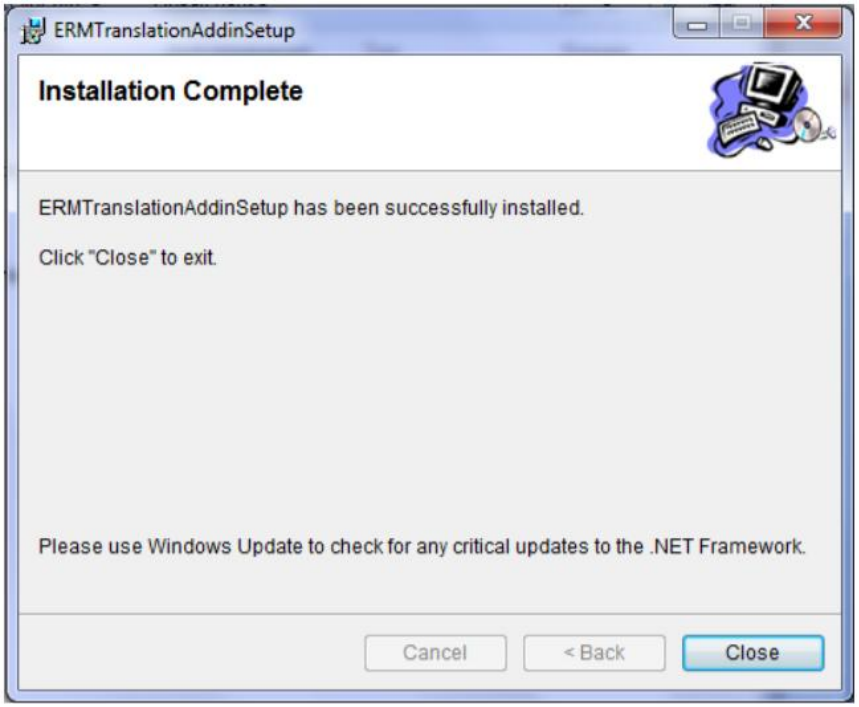


Рис. 3.10. Вікно успішного завершення процесу встановлення ERMTranslationAddinSetup

Робота з плагіном в IDE Visual Studio. При завантаженні середовища розробки Visual Studio, відкрити вікно трансляції можна, клікнувши по кнопці основного меню Tools> ERM Translation (рис. 3.11).

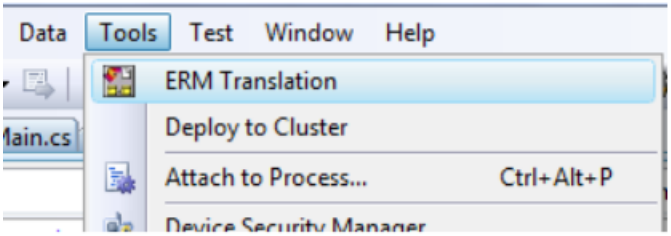


Рис. 3.11. Tools> ERM Translation

При цьому відкриється вікно роботи з модулем трансляції (рис. 3.12):

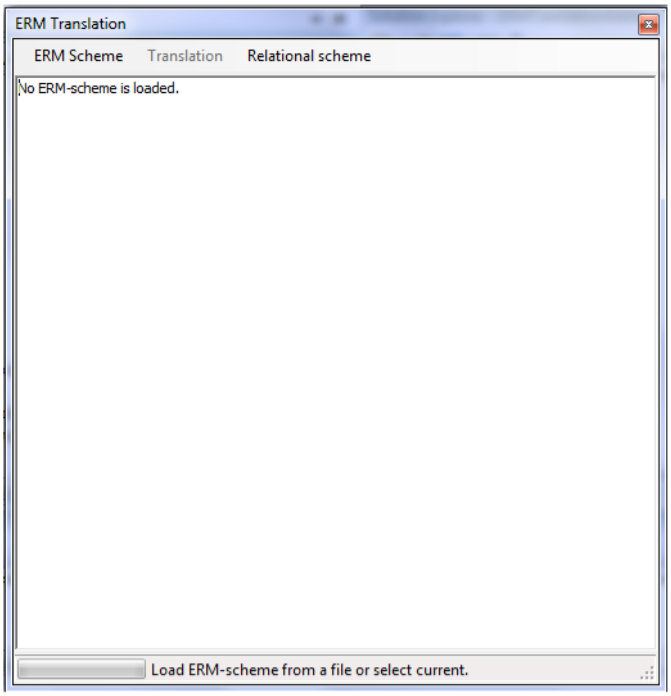


Рис. 3.12. Вікно з модулем трансляції

Користувач також може викликати це вікно, перебуваючи у вікні редактора схем. Необхідно натиснути правою клавішею миші по порожньому простору схеми, після чого з'явиться контекстне меню, де треба вибрати пункт Translate schemes> Open ERM Translate Window (рис. 3.13):

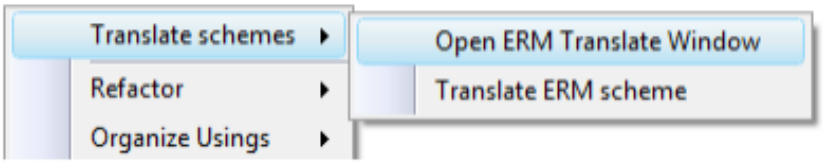


Рис. 3.13. Вибір вікна з модулем трансляції через редактор схем

Завантаження ERM-схеми. У звичайному режимі схема при редагуванні відразу завантажена в пам'ять, варто вибрати тільки потрібну схему. Крім того, можна зробити і завантаження схеми з файлу. Завантажити схему з файлу можна таким способом. В основному меню вікна ERM Translation потрібно вибрати пункт меню ERM Scheme> Open file. Вам відкриється вікно вибору шляху

файлу. Знайдіть потрібний вам файл схеми і натисніть кнопку Open (див. рис. 3.14).

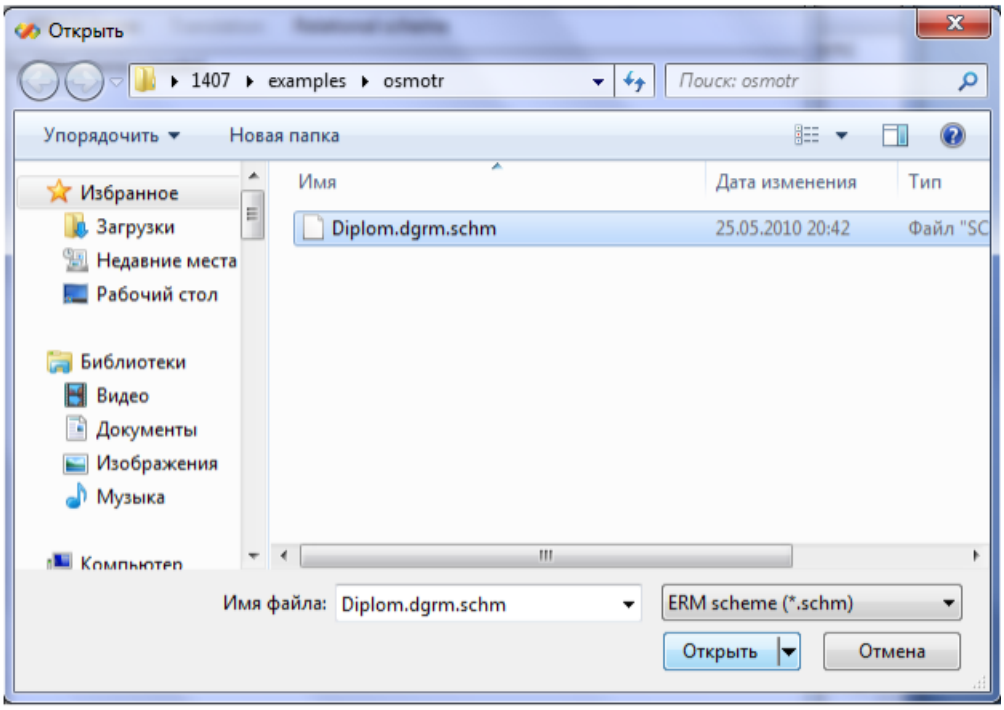


Рис. 3.14. Вікно відкриття файлу з схемою

Після успішного завантаження схеми в основне поле вікна трансляції буде виведено повідомлення «ERM-scheme is loaded»

Трансляція ERM-схеми. Після того, як ERM-схема була завантажена, можна почати трансляцію ERM-схеми в реляційну. Для цього потрібно вибрати пункт основного меню Translation> Run (рис. 3.15):

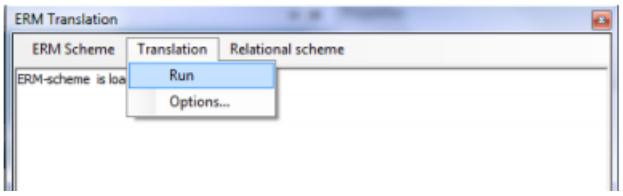


Рис. 3.15. Початок трансляції схеми в реляційну

Після чого почнеться трансляція схеми, а результат трансляції буде виведений в основне поле вікна трансляції (рис. 3.16):

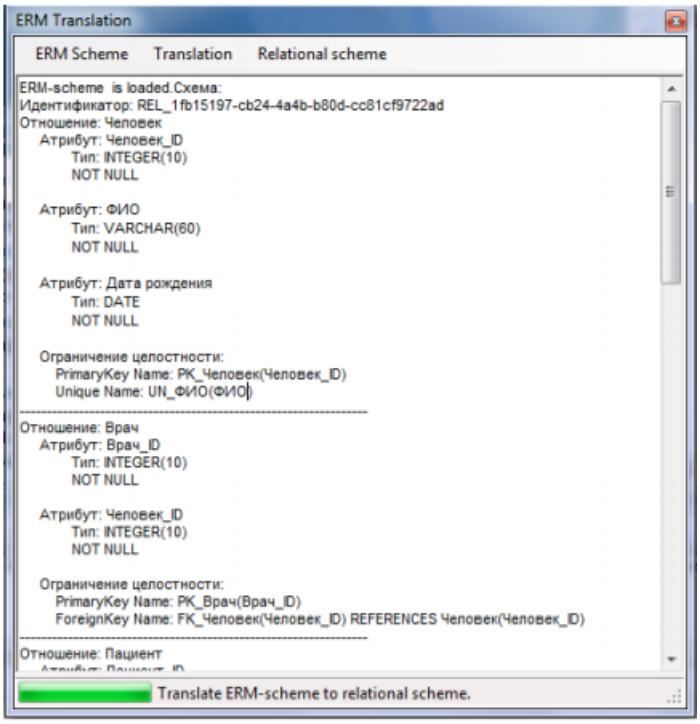


Рис. 3.16. Основне поле вікна трансляції

Збереження реляційної схеми. Після завершення трансляції реляційну схему можна зберегти в файл. Для цього треба вибрати пункт основного меню Relational scheme> Save in file ...: (рис. 3.17). Після чого відбудеться збереження схеми в файл з розширенням .rsh. Новий файл схеми буде розташований в тій же папці, в якій лежить ERM-схема.

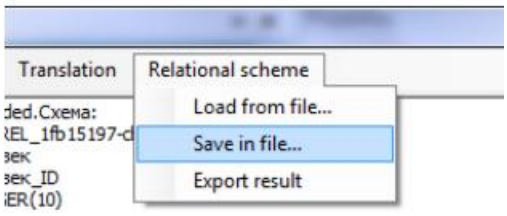


Рис. 3.17. Збереження схеми

Завантаження реляційної схеми. Крім того, можна завантажити існуючий варіант реляційної схеми. Для цього потрібно вибрати пункт основного меню Relational scheme> Load from file ... (рис. 3.18).

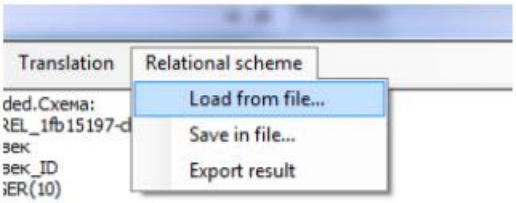


Рис. 3.17. Завантаження раніше збереженої схеми

Відкриється вікно вибору шляху файлу. Треба знайти потрібний файл схеми і натиснути кнопку Open. Після завантаження опис схеми буде виведено в основне поле вікна трансляції (див. рис. 3.16)

3.5. Висновки до розділу

У цьому розділі наведено програмну архітектуру засобу для трансформації ERM-схеми в реляційну. Описано модуль трансляції ERM-схеми в реляційну схему. Відображено процес практичної реалізації трансляції схем та встановлення плагіну у Visual Studio.

РОЗДІЛ 4
ОБГРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Метою цього розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності від дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель, а також прийняття рішення щодо подальшого розвитку і впровадження або ж недоцільність впровадження відповідної розробки.

Для здійснення оцінки потрібно зробити розрахунки трудомісткості кожної операції, що мала місце при проведенні наукових досліджень.

4.1. Розрахунок норм часу на виконання науково-дослідної роботи

Реалізація проекту дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель складається з низки послідовних та взаємопов’язаних етапів. Кожен із етапів реалізації проекту характеризується метою та змістом, оцінкою часу виконання, кількістю та спеціалізацією виконавців, а також приблизною оцінкою вартості.

Реалізація проекту складається із підготовчого етапу, етапу технічної пропозиції, створення технічного завдання, проектування системи, практичної реалізації, тестування, верифікації та заключного етапу.

Норми часу на виконання науково-дослідницької роботи розраховуватимуться на основі середнього часу виконання стадії в годинах, що наведені в таблиці 4.1 разом із інформацією про виконавців і сумарною кількості затраченого часу.

Таблиця 4.1

Операції технологічного процесу та їх час виконання

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Підготовча стадія	Проектний менеджер Інженер-програміст	10
2	Технічна пропозиція	Проектний менеджер Інженер-програміст	10
3	Створення технічного завдання	Проектний менеджер Інженер-програміст	20
4	Проектування системи	Інженер-програміст	20
5	Практична реалізація	Інженер-програміст	135
6	Тестування системи	Тестувальник	20
7	Верифікація системи	Тестувальник Інженер-програміст Проектний менеджер	20
8	Створення документації	Інженер-програміст	20
9	Заключна стадія	Проектний менеджер	10
Разом			265

В підсумку на реалізацію проекту дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель необхідно 265 людино-годин, залучення трьох спеціалістів та виконання дев’яти різноманітних стадій реалізації проекту.

4.2. Визначення витрат на оплату праці та відрахувань на соціальні заходи

Визначення витрат на оплату праці та відрахувань на соціальні заходи прямо залежить від кількості витраченого працівниками часу на роботу, ставки в годину чи місяць, кількість відрахувань на соціальні заходи встановлених в законному порядку на час розрахунку.

В результаті розрахунку потрібно визначити основну та додаткову заробітну плату, витрати на соціальні заходи та на основі цих даних визначити сумарні витрати на оплату праці. Основна заробітна плата нараховується за виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами. Додаткова заробітна плата – це складова заробітної плати працівників,

до якої включають витрати на оплату праці, не пов’язані з виплатами за фактично відпрацьований час.

При розрахунку заробітної плати кількість робочих днів у місяці слід в середньому приймати – 24,5 дні/міс., або ж 196 год./міс. (тривалість робочого дня – 8 год.).

Наймані працівники для дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель працюють згідно контракту, який в якому вказано їхню погодинну ставку. Тобто розрахунок заробітної плати працівників відбуватиметься на базі тарифної ставки та кількості відпрацьованих годин.

У штаті найманих працівників для дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель залучено проектного менеджера, інженера-програміста і тестувальника.

Тарифні ставки учасників процесу дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель:

– Проектний менеджер – 150 грн./год.

– Інженер-програміст – 130 грн./год.

– Тестувальник – 100 грн./год.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с \bullet К_г , \qquad (4.1)$$

де $Т_с$ – тарифна ставка, грн.; $К_г$ – кількість відпрацьованих годин.

Оскільки всі види робіт в виконує три спеціаліста, то основна заробітна плата буде розраховуватись за даною формулою 4.1;

$$З_{осн.} = 150 \cdot 35 + 130 \cdot 200 + 100 \cdot 30 = 34250 \text{ грн.}$$

Додаткова заробітна плата становить 10–15 % від суми основної заробітної плати й визначається за формулою (4.2).

Коефіцієнт додаткових виплат працівникам становить 0,1.

$$Здод. = Зосн. \cdot Кдопл., \quad (4.2)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам

$$З_{дод} = 34250 \cdot 0,1 = 3425 \text{ грн.}$$

Звідси загальні витрати на оплату праці (фонд заробітної плати) визначаються за формулою (4.3):

$$Во.п. = Зосн. + Здод. \quad (4.3)$$

$$В_{о.п.} = 34250 + 3425 = 37675 \text{ грн.}$$

З цієї суми утримуються обов'язкові відрахування на соціальні заходи:

– Єдиний соціальний внесок, що становить 22%;
– Військовий збір, що становить 1,5%;
– податок на доходи фізичних осіб: 18%;

Сума відрахувань становить 41,5% від фонду оплати праці та визначається за формулою:

$$В_{с.з.} = Ф_{оп} \bullet 0,415 \quad (4.4)$$

де $Ф_{оп}$ – фонд оплати праці, грн.

$$В_{с.з.} = 37675 \cdot 0,415 = 15635,125$$

Усі витрати обчислюються детально наведені в таблиці 4.2 та обчислюються за формулою:

$$Взп = ФЗП + ФОП \quad (4.5)$$

$$В_{зп} = 34250 + 15635,125 = 49885,125 \text{ грн.}$$

Таблиця 4.2

Розрахунки витрат на оплату праці

з/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	Нарахув. на ФОП, грн.	Всього витрати на плату праці, грн. (6=3+4+5)
		Тарифна ставка, грн.	Кількість відпрацьованих год.	Фактично нарах. з/пл., грн.			
1.	Проектний менеджер	150	35	5250	525	-	-
2.	Інженер-програміст	130	200	26000	2600	-	-
3.	Тестувальник	100	30	3000	300	-	-
Разом		380	265	34250	3425	15635,125	49885,125

Опираючись на розрахунки витрат на оплату та таблицю результатів 4.2 видно, що всього витрати на плату праці становлять 49885,125 грн.

4.3. Розрахунок матеріальних витрат

Матеріальні витрати є невід'ємною частиною розробки та визначаються як добуток кількості витрачених матеріалів та їх ціни за формулою:

$$M_{ві} = q_i \bullet p_i , \qquad (4.6)$$

де: q_i – кількість витраченого матеріалу і–го виду; p_i – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити за формулою:

$$З_{м.в.} = \sum M_{ві} . \qquad (4.7)$$

Результати проведених розрахунків наведено у таблиці 4.3.

Таблиця 4.3

Результати розрахунків матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна одиниці, грн.	Загальна сума витрат, грн.
1	CD диски	шт.	2	7,45	14,90
2	Папір для друку	листів	500	0,15	75,00
3	Чорнила для принтера	шт.	1	80,00	80,00
Всього					169,90

Згідно проведених розрахунків, матеріальні витрати становлять 169,90 грн.

4.4. Розрахунок витрат на електроенергію

Однією із статей витрат є витрати на електроенергію під час проходження усіх етапів реалізації кінцевого продукту.

Затрати на електроенергію одиниці обладнання визначаються за формулою:

$$З_в = W \cdot T \cdot S, \qquad (4.8)$$

де W – необхідна потужність, кВт; T – кількість годин на реалізацію розробки; S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів. Отже, 1 кВт з ПДВ коштує 2,42 грн.

Потужність комп'ютерів для реалізації кінцевого продукту – 400 Вт, кількість годин роботи обладнання згідно таблиці 4.1 – 265 годин.

Визначимо витрати на електроенергію згідно формули :

$$З_в = 0,4 \cdot 265 \cdot 2,42 = 256,52 \text{ грн.}$$

Згідно формули затрати на електроенергію становлять 256,52 грн.

4.5. Розрахунок суми амортизаційних відрахувань

Для будь якої діяльності характерною є властивість зношування на зниження якості властивостей інструментарію та фондів за допомогою яких ведеться діяльність. Для вирішення проблеми із відновленням даних фондів використовується амортизація, що являє собою процес трансформації вартості основних фондів на вартість продукції, яка щойно була створена, задля повного відновлення основних фондів.

Для визначення амортизаційних відрахувань використовується формула:

(4.9)

де A – амортизаційні відрахування за звітний період, грн.; $Б_B$ – балансова вартість групи основних фондів на початок звітного періоду, грн.; H_A – норма амортизації.

Комп’ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Річний робочий фонд становитиме 2352 годин, так як робочий день становить 8 годин, а кількість робочих днів в місяці становить 24,5 годин.

Для даної розробки засобом розробки є комп’ютер. Його сума становить 18500 грн. Отже, амортизаційні відрахування будуть рівні:

$$A = 18500 \cdot 5\% / 100\% = 925 \text{ грн.}$$

Згідно проведених обчислень амортизаційні відрахування становлять 925 грн.

4.6. Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$H_в = B_{о.п.} \cdot 0{,}2\ldots0{,}6\ , \qquad (4.10)$$

де $H_в$ – накладні витрати.

Отже, накладні витрати становлять згідно формули (4.10):

$$H_в = 37675 \cdot 0{,}2 = 7535 \text{ грн.}$$

Накладні витрати згідно розрахунку формули, становить 7535 грн.

4.7. Складання кошторису витрат та визначення собівартості науково-дослідницької роботи

Результати проведених вище розрахунків наведено у таблиці 4.4.

Таблиця 4.4

Кошторис витрат на НДР

Зміст витрат	Сума, грн.	В % до загальної суми
Витрати на оплату праці	43103,6	63,74
Відрахування на соціальні заходи	15635,125	23,12
Матеріальні витрати	169,9	0,25
Витрати на електроенергію	256,52	0,38
Амортизаційні відрахування	925	1,37
Накладні витрати	7535	11,14
Собівартість	67625,145	100,00

Собівартість ($C_в$) програмного продукту розрахуємо за формулою:

$$C_в = B_{з.п.} + B_{с.з.} + З_{м.в.} + З_в + А + Н_в . \quad (4.11)$$

Отже, собівартість програмного продукту дорівнює:

Св = 49885,125+ 15635,125+ 169,90 + 256,52+ 925+ 7535= 74406,67 грн.

Загальний кошторис витрат та визначення собівартості науково-дослідницької роботи становить 74406,67 грн.

4.8. Розрахунок ціни програмного продукту

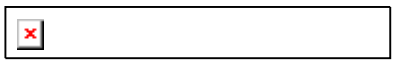
Ціну науково-дослідної роботи можна визначити можна визначити за формулою:

 , (4.12)

де $Р_{рен.}$ – рівень рентабельності (30 %); $К$ – кількість замовлень, од. (встановлюється лише при розробці програмного продукту та мікропроцесорних систем); $В_{н.і.}$ – вартість носія інформації, грн. (встановлюється лише при розробці програмного продукту); *ПДВ* – ставка податку на додану вартість (20%).

Оскільки розробка є прикладною, і використовуватиметься тільки для одного підприємства, то для розрахунку ціни не потрібно вказувати коефіцієнти К та $В_{н.і.}$, оскільки їх в даному випадку не потрібно.

Тоді, формула для обчислення ціни розробки буде мати вигляд:

 (4.13)

Звідси ціна на роботу складе:

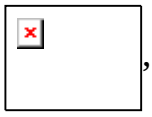
$$Ц = 74406{,}67 \cdot (1 + 0{,}3) \cdot (1 + 0{,}2) = 116074{,}4 \text{ грн.}$$

Загальний розрахунок ціни програмного продукту становить 116074,4 грн.

4.9. Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність ($E_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

 , (4.14)

де $П$ – прибуток; $С_В$ – собівартість.

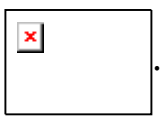
Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = Ц - С_в . \quad (4.15)$$

Розраховуємо плановий прибуток:

$$П_{пл} = 116074{,}4 - 74406{,}67 = 41667{,}73 \text{ грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

 . (4.16)

Тоді,

$$E_р = 41667{,}73 / 74406{,}67 = 0{,}56.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (T_p):

 , (4.17)

Термін окупності дорівнює:

$$T_p = 1 / 0{,}5 = 1{,}78 \text{ р.}$$

Згідно формул плановий прибуток від розробки становить 41667,73 грн., економічна ефективність дорівнює 0,56, а термін окупності становить 1,78 роки що вважається доцільним та економічно вигідним.

4.10. Висновки до розділу

В організаційно-економічній частині дипломної роботи освітнього рівня «магістр» було розраховано основні техніко-економічні показники дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель (див. таблиця 4.5).

Орієнтоване значення економічної ефективності становить 0,56, що є достатньо високим значенням.

Період окупності повинен варіюватися від 1 до 3 років, тоді розвиток вважається доцільним та економічно вигідним. Термін окупності даної роботи становить 1,78 років.

Таблиця 4.5

Техніко-економічні показники науково-дослідної роботи

№ п/п	Показник	Значення
1	Собівартість, грн.	74406,67
2	Плановий прибуток, грн.	41667,73
3	Ціна, грн.	116074,4
4	Економічна ефективність	0,56
5	Термін окупності, рік	1,78

На основі проведених обрахунків можна зробити висновок, що дослідження і використання методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель є доцільним у зв’язку з невеликим терміном окупності та великим обсягом планового прибутку.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Охорона праці

Метою дипломної роботи магістра є дослідження методів та засобів трансформації схем баз даних з ERM-моделі в реляційну модель. Оскільки, проведення робіт з розробки та використання системи передбачає використання комп'ютерної техніки, зокрема ПК та периферійних пристроїв, то обов'язковим є дотримання вимог з охорони праці і техніки безпеки.

Для ефективної і безпечної роботи колективу працівників з розробки ПЗ комп'ютерних систем, в тому числі і фахівців з дослідження методів та засобів трансформації схем баз даних, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці [18].

Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Згідно Вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до цих вимог;

– переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 22.08.2005 N 161, зареєстрованого в Міністерстві юстиції України 05.09.2005 за N 990/11270 (НАПБ Б.06.004-2005);

– Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом Держбуду України від 28.10.98 N 247 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками.

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014. Приміщення, в яких розміщуються робочі місця операторів сервера загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, ДБН В.2.5-56:2010, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп’ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97.

У приміщенні, де одночасно експлуатуються понад п’ять комп’ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним.

Не допускається підключати комп'ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення комп'ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі.

Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

При експлуатації інструментального засобу управління процесом міграції віртуальних машин в обчислювальній хмарі, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18. Організація робочого місця фахівця із дослідження методів та інструментальних засобів управління процесом міграції віртуальних машин в обчислювальній хмар повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Відстань від екрана до ока фахівців, які працюють за комп'ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп'ютера, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля згідно з вимогами ДСанПіН 3.3.2.007-98.

Таким чином, у результаті аналізу вимог щодо охорони праці користувачів комп’ютерів, визначено особливості організації робочих місць, вимог з електробезпеки, природного та штучного освітлення для ефективної і безпечної роботи фахівців з дослідження методів та інструментальних засобів управління процесом міграції віртуальних машин в обчислювальній хмарі.

5.2. Оцінка стійкості роботи об’єкта господарської діяльності (ОГД) в умовах надзвичайних ситуацій (НС) мирного та воєнного часу

НС (extraordinary situation) - це порушення нормальних умов життя і діяльності людей на об’єкті або території, спричинене аварією, катастрофою, стихійним лихом або іншими чинниками, що призвело (може призвести) до загибелі людей та/або значних матеріальних втрат. Загальними ознаками НС є: матеріальні збитки, істотне погіршення стану довкілля, наявність або загроза загибелі людей, чи суттєві погіршення умов їх життєдіяльності.

Національний класифікатор ДК 019:2010 «Класифікатор надзвичайних ситуацій» так класифікує НС воєнного характеру, як пов'язані з наслідками застосування звичайної зброї або зброї масового ураження, під час якої виникають вторинні фактори ураження населення. НС мирного характеру поділяються на: природного характеру; техногенного характеру; соціально-політичного характеру. [19]

Дослідження стійкості об'єкта проводиться в мирний час силами інженерно-технічного складу без відриву від виробництва. Керівником дослідження є начальник ЦЗ об'єкта, тобто керівник виробництва. Тривалість досліджень – 2-3 місяці. Весь процес планування і проведення досліджень поділяють на три етапи. Перший етап - підготовчий: розробка керівних документів (наказ начальника ЦЗ об'єкта, календарний план, план проведення досліджень, визначення складу учасників дослідження та їх підготовка). Другий етап - оцінка стійкості роботи об'єкта за надзвичайних ситуацій. Третій етап - розробка заходів, що підвищують стійкість роботи об'єкта. Дослідження стійкості спрямовані на виявлення найменш стійких елементів з тим, щоб на

основі проведених досліджень спланувати і провести заходи, які підвищують стійкість усього об'єкта в цілому.

Методика оцінки стійкості роботи ОГД до дії вражаючих факторів ядерного вибуху зводиться до розрахунків чи визначення якимось іншим шляхом того граничного значення даного вражаючого фактора, за якого об'єкт чи його частина ще можуть стійко працювати, забезпечуючи виконання запланованих завдань.

Окремо оцінюється стійкість роботи об'єкта до дії:

– ударної хвилі ядерного вибуху (критерієм стійкості об'єкта є максимальне значення надлишкового тиску, під час дії якого будівлі, споруди та обладнання об'єкта ще зберігаються або отримують слабкі чи середні руйнування);

– світлового опромінювання (критерієм стійкості об'єкта є значення того мінімального імпульсу світлового опромінення, при якому може виникнути займання матеріалів чи споруд, в результаті чого на об'єкті виникнуть пожежі);

– вторинних вражаючих факторів (вторинними вражаючими факторами є пожежі, вибухи, затоплення, забруднення атмосфери та місцевості і т. ін.. Джерела вторинних вражаючих факторів на об'єкті й в небезпечному віддаленні від нього повинні виявлятися заздалегідь з метою завчасного прийняття заходів, що направлені на виключення чи зменшення вражаючої дії).

Також необхідно визначити можливості роботи об'єкта в умовах радіоактивного, хімічного й бактеріологічного (біологічного) забруднення. Тут критерієм стійкості роботи об'єкта є допустима доза опромінення, яку можуть одержати робітники й службовці зміни, що працює, за час роботи при дотриманні встановленого режиму захисту. У розрахунках виходять з того, що при радіоактивному забрудненні робітники й службовці знаходяться на робочих місцях у продовж усієї робочої зміни (у воєнний час 10-12 годин), а потім перебувають в захисних спорудах. Також потрібно провести аналіз надійності систем керування, постачання і виробничих зв'язків.

Стійка робота об'єкта у НС може бути досягнута шляхом проведення комплексу організаційних, інженерно-технічних та інших заходів у відповідності до вимог [20]. Вони повинні бути тісно пов'язані з підготовкою і проведенням

рятувальних невідкладних аварійних робіт. Без людських ресурсів і успішної ліквідації наслідків НС проводити заходи забезпечення стійкої роботи об'єкта народного господарства буде практично неможливо.

5.3. Забезпечення безпеки життєдіяльності користувачів ПЕОМ в умовах НС

Забезпечення безпеки (захист) населення у НС є основна задача ЦЗ населення. Основні способи захисту населення у НС відповідно до розділу IV Кодексу цивільного захисту України від 02.10.12 no 5403-vi:

– укриття в захисних спорудах;

– евакуація з небезпечних районів;

– застосування засобів індивідуального захисту.

При цьому захист населення може бути ускладнений такими факторами: ступінь підготовки населення до дій в НС; постійне спостереження за станом навколишнього середовища; своєчасне повідомлення населення про загрозу НС; захист систем водопостачання, продуктів, сировини від небезпечного забруднення; розробка та своєчасне введення в дію режимів захисту людей; організацією планування та проведення профілактичних, санітарно-гігієнічних, протипожежних заходів, обеззараження техніки, території, споруд та ін.

Роль того чи іншого заходу та його питома вага в загальному комплексі заходів захисту залежать від умов в яких він застосовується. Найбільший ефект може бути досягнутий лише при комплексному застосуванні всіх заходів. В Україні створена і функціонує Єдина державна система цивільного захисту населення від небезпечних наслідків, аварій та катастроф техногенного, екологічного, природного та воєнного характеру. Забезпечення безпечної життєдіяльності у НС базується на комплексі організаційних, інженерно-технічних заходів і засобів, спрямованих на збереження життя і здоров'я людини у всіх сферах її діяльності. Для цього необхідно: [18]

– прогнозувати та оцінити можливі наслідки;

– заздалегідь спланувати заходи із запобігання та зменшення вірогідності виникнення НС і скорочення масштабів прояву результатів НС;

– організація робіт в умовах НС та ліквідація її наслідків.

Необхідно заздалегідь планувати роботи необхідні для запобігання або зменшення можливості їх виникнення та скорочення масштабів наслідків для забезпечення стійкої роботи об'єктів народного господарства в умовах НС. Для здійснення цих заходів важливим є набуття населенням умінь, навичок поводження в умовах НС, що надалі сприятиме зменшенню негативних результатів, ліквідації наслідків надзвичайної ситуацій. Отже, всі можливі дії у разі виникнення НС повинні бути заздалегідь чітко сплановані.

Кінцевий результат планування дій є документ - план, який повинен містити такі елементи: конкретні показники видів робіт, заходів, які треба провести в умовах НС та терміни виконання цих робіт. Важливим є перелік ресурсів, необхідних для виконання плану та зазначення конкретних обов'язків осіб, відповідальних за виконання кожного пункту плану; способи контролю за ходом його виконання. Текстова частина плану може складатися з двох розділів. В першому наведені висновки з оцінки обстановки, яка може скластися в результаті НС. Другий розділ висвітлює заходи щодо забезпечення безпеки населення при загрозі виникнення НС. В ньому треба вказати послідовність дій: порядок оповіщення; організація розвідки і спостереження; підготовка сил і засобів для проведення рятувальних та інших невідкладних робіт; заходи щодо попередження і пом'якшення наслідків НС; прискорене проведення робіт, необхідних для захисту людей і матеріальних цінностей; хід забезпечення медичного, дозиметричного і хімічного контролю; порядок проведення заходів щодо безаварійного припинення виробництва; організація захисту людей і видача населенню ЗІЗ та проведення евакуаційних заходів; керівництво управлінням, порядком і черговістю ведення рятувальних та інших невідкладних робіт у реальних умовах НС; надання повідомлень у вищі органи ЦЗ, в комісію з надзвичайних ситуацій.

Важливим є прогноз та оцінка можливих наслідків НС, які виникають в ході її розвитку і характеру її прояву. Для цього застосовують методи орієнтовного виявлення та оцінки обстановки, яка виникає в результаті стихійних лих, аварій і катастроф, воєнних конфліктів. Складність полягає в тому, що оцінка стану території, характеру і масштабу НС в умовах неповної і

ненадійної інформації, дає можливість орієнтовно визначити характер і обсяг необхідних робіт з ліквідації її наслідків. На основі цього складають довгостроковий прогноз.

Прогнозування обстановки, пов'язаної з виникненням НС, здійснюють і математичними методами із застосуванням комп’ютерів. Вихідними даними для прогнозування обстановки є місця потенційно небезпечних об'єктів - запаси речовин, джерела енергії, чисельність і щільність населення.

Крім цього, з точки зору забезпечення безпеки життєдіяльності робітників та службовців, а також населення, що мешкає поблизу об'єкта, важливе місце займають заходи з недопущення виникнення вторинних вражаючих чинників - пожеж, вибухів, які можуть виникати як під впливом внутрішніх, так і зовнішніх причин.

5.4. Висновки до розділу

В цьому розділі розглянуто важливі питання охорони праці та безпеки в надзвичайних ситуаціях, зокрема дослідження стійкості роботи ОГД в умовах НС мирного та воєнного часу та можливість роботи в умовах радіаційного, хімічного, бактеріологічного забруднення і враховано основні способи захисту населення, які необхідно застосовувати. Розроблено заходи щодо планування дій із запобігання та пом’якшення наслідків в умовах НС.

РОЗДІЛ 6
ЕКОЛОГІЯ

6.1. Метод екологічної статистики

Екологічна статистика — галузь статистики природних ресурсів і навколишнього середовища. Включає дані про стан забруднення природних об'єктів — атмосферного повітря, природних водних об'єктів, ґрунтів, одержувані на підставі моніторингу. В загальному, поняття метод статистики - це сукупність прийомів, способів обробки цифрової інформації, правил і методів дослідження.

Якість природних об'єктів оцінюється показниками: кількість вимірів, середня концентрація, максимальна концентрація, повторюваність концентрації шкідливих домішок вище гранично припустимої концентрації. Дані екологічної статистики використовуються в соціально-економічному аналізі для оцінки результатів заходів щодо зниження шкідливих викидів в атмосферу, забруднених стоків у природні водні об'єкти, визначення взаємозв'язку якості навколишнього середовища і станів здоров'я населення, а також визначення економічного збитку від забруднення навколишнього середовища в зв'язку зі зниженням врожайності сільськогосподарських культур, погіршенням продуктивності у тваринництві, підвищеним зносом будинків, споруджень і т. д. [21]

Середовище і його структурні елементи характеризуються множиною специфічних ознак, кожна з яких має свої параметри. Параметр - це кількісна характеристика ознаки. Таких характеристик для кожної ознаки є чимало. Таким чином, параметрів середовища дуже багато і кожен з них потребує використання системи методів вимірювання. Основними ознаками середовища є: екологічний стан середовища; варіація властивостей і стосунків в середовищі; екологічні зв’язки (стосунки) в середовищі; динаміка і тенденція змін стану середовища;

Екологічний стан середовища - це природна ситуація, яка виникла внаслідок дії фізичних, хімічних і біологічних чинників. Його можна встановлювати вимірюванням і оцінкою двох основних параметрів: продуктивності і забруднення природного середовища.

Екологічні зв'язки виступають на всіх рівнях екологічних систем як безпосередньо між організмами, так і між організмами й оточуючим середовищем. Прикладом є залежність організму від певних чинників, які присутні в оточуючому середовищі - температури повітря і ґрунту, від яких залежить розвиток рослини. Врожай певних рослин може впливати на розмноження популяції синиці чи інших пташиних.

Структурними частинами екологічної статистики є: [21]

– статистика стану і забруднення атмосферного повітря; статистика стану, використання й охорони водних ресурсів; статистика землекористування і земельних угідь; статистика охорони і захисту лісу; статистика знешкодження відходів;

– статистика стану і забруднення атмосферного повітря - підрозділ статистики природних ресурсів і навколишнього середовища, основним завданням якого є збір і узагальнення інформації про виконання заходів щодо охорони атмосферного повітря, про шкідливі викиди в атмосферу. В аналітичній роботі використовуються також дані про якісний стан атмосфери. Виробничі об’єднання (комбінати), підприємства й організації, що мають шкідливі викиди в атмосферу, представляють у статистичні органи звіт про охорону атмосферного повітря, що характеризує виконання ними заходів щодо охорони атмосферного повітря від забруднень, а також викиди шкідливих речовин в атмосферу (без очищення і після очищення), їхнє уловлювання (знешкодження) і утилізацію, оснащення джерел викидів газоочисними і пиловловлюючими спорудженнями. Для підвищення вірогідності статистичної інформації на підприємствах уведені форми первинної звітної документації, що заповнюються регулярно протягом року. Кількісну оцінку шкідливих викидів автотранспорту здійснюють природоохоронні органи на основі даних про пробіг транспортних засобів і нормативів питомих викидів;

– статистика стану, використання й охорони водних ресурсів — підрозділ статистики природних ресурсів і навколишнього середовища, що вивчає запаси водних ресурсів, їхній склад і якість, забезпеченість народного господарства водними ресурсами, водозабір і водоспоживання, утрати води, економію свіжої води за рахунок повторного й оборотного використання води, водовідведення,

скидання стічних вод у природні водойми й ін. водоприймачі (по видах вод, що скидаються);

– статистика землекористування і земельних угідь — підрозділ статистики сільського господарства. Вивчає склад і структуру землекористувачів і земельних угідь, розмір, стан і динаміку земельного фонду, його трансформацію, ступінь використання, якість ґрунтів, ступінь деградації ґрунтів та ін.;

– статистика охорони і захисту лісу розділ статистики лісового господарства, що характеризує охорону лісу від пожеж, порушення встановленого порядку лісокористування й ін. дії, що заподіюють шкоду лісові, а також захист лісу від шкідників і хвороб. Показники охорони і захисту лісу знаходять висвітлення в планах і статистичній звітності;

– статистика знешкодження відходів — підрозділ статистики природних ресурсів і навколишнього середовища, що характеризує утворення, використання, видалення відходів і охорону навколишнього середовища від забруднення ними. У натуральному вираженні враховуються (відповідно до затвердженої номенклатури) маса відходів, що утворяться, (т), їхня утилізація у власному підприємстві і передача для використання ін. підприємствам, вивіз відходів на смітники і сміттєпереробні заводи. У статистиці визначаються розміри земельних площ (га) для складування і знешкодження відходів; витрати на заходи щодо охорони навколишнього середовища від забруднення відходами, включаючи капітальні вкладення на будівництво сміттєпереробних заводів, що забезпечують утилізацію відходів, а також поточні витрати по вивозі і похованню відходів.

6.2. Екологічна відповідальність та організація «Зеленого офісу»

Екологічна відповідальність – це компенсаційна матеріально-фінансова відповідальність за завдану екологічну шкоду; обов'язок суб'єкта економічної діяльності відшкодувати завдану екологічну шкоду; важлива при приватизації та придбанні підприємств у приватну власність, попередній власник якої міг своїми діями викликати екологічні порушення; в разі їх неврахування новому власнику можливо доведеться відповідати за екологічний збиток, нанесений попереднім

власником. Екологічна відповідальність є різновидом цивільної відповідальності (civil liability) і застосовується як інструмент для стримування та запобігання екологічного збитку. Основне завдання полягає у стягненні вартості ліквідації екологічного збитку з боку відповідального за його нанесення. Застосування принципу громадянської відповідальності передбачає досягнення 4 результатів: гарантує компенсацію постраждалій стороні економічного збитку; зберігає довкілля шляхом відшкодування збитку відповідальною стороною; стимулює підприємства застосувати превентивні і передбачливі заходи, такі як аналіз ризику, екоаудит, системи природоохоронного управління, з тим, щоб уникнути, в іншому випадку, надмірних витрат; відповідає принципу «забруднювач платить».

Застосовуються два види компенсаційної відповідальності. Перший — за невияв необхідної обережності (розумної поведінки) — fault liability — (для охорони прав особистості, права приватної власності або права на здоров'я — матеріально-фінансову відповідальність несе сторона, яка не змогла прийняти «розумні» в даних обставинах запобіжні заходи в відношенні навколишнього середовища, сторона — позивач має довести, що відповідач вчинив недозволену дію, що призвело до ековтрат). Другий — безумовна відповідальність (strict liability), при якій тягар доказів переходить до відповідача, який повинен довести свою невинність, і відповідно до якої будь-яка сторона, яка займається діяльністю, що за визначенням несе ризик нанесення збитку (наприклад, якщо вона входить до списків небезпечних видів діяльності), несе матеріально-фінансову відповідальність за всі каліцтва або збиток, викликані даним підприємством, навіть якщо вони не є наслідком очевидної непередбачливість.

Відповідальність за шкоду в результаті непередбачливої поведінки набула більшого поширення в законодавстві європейських країн, у той час як безумовна відповідальність стала вводитися в середині 1990-х рр. з метою обмеження небезпечних видів діяльності, але не втрачаючи принесених ними соціальних вигод. У Європі діють Пропозиція Європейської комісії з цивільної відповідальності за екологічний збиток (the European Commission's Proposal on Civil Liability) та Міжнародний договір Європейського Союзу про збиток в результаті небезпечної для навколишнього середовища діяльності (the Council of

Europe International Treaty for damage resulting from activities dangerous to the environment) — Луганська конвенція від 21 червня 1993 р. Природно, що застосування тільки громадянської компенсаційної відповідальності недостатньо для забезпечення повної безпеки навколишнього середовища від антропогенного впливу, особливо в умовах більш серйозного або повторного забруднення, коли накладення штрафу є лише частиною покарання. При такому підході не повністю використовується принцип платності за забруднення.

«Зелений офіс» — це певна концепція, що стосується організаційних дій і поведінкових звичок та складається з мотиваційних, технічних і освітніх заходів, покликаних допомагати співробітникам компанії дбайливо ставитися до ресурсів офісу, підвищувати корпоративну культуру та її репутацію. Компанії, які реалізують свою діяльність відповідно до рекомендацій зеленого офісу, раціональніше використовують ресурси та енергію, зменшують кількість відходів при роботі в офісних приміщеннях та покращують свою репутацію в очах споживачів, клієнтів та партнерів Основною метою впровадження концепції є зниження навантаження на навколишнє середовище. [22]

Концепція зеленого офісу передбачає дуже широкий спектр заходів, які можна застосувати, зокрема: придбання екологічно чистих товарів; використання ресурсів (електропостачання, водопостачання, теплопостачання). “Зелений друк”:— намагатися розміщувати на одному аркуші паперу якомога більше тексту. “Зелена реклама”— створювати рекламу через інтернет замість використання друкованих оголошень плакатів та банерів; — надсилати листи електронною поштою замість відправки паперових;— проводити презентації замість роздачі друкованих брошур; на етапі утилізації варто дотримуватися правил 3 R: Reduce (скорочуй) — усі дії мають бути спрямовані на скорочення витрат будь-яких ресурсів; Reuse (повторно використовуй) — використовуйте ті матеріали, які можна задіяти ще декілька разів; Recycle (переробляй) — правильно утилізовуйте відходи та сортуйте сміття.

Сьогодні важко знайти ІТ-компанію, у якій немає боксу для збору батарейок, а ще часто ставлять контейнери для макулатури, пластикових кришечок та використаних ламп. Такі ініціативи найпростіші у реалізації та подальшій підтримці, бо, наприклад, бокси з батарейками, з досвіду компаній,

потребують вивозу 1-2 рази на рік. При цьому співробітники компаній залюбки приносять використані батарейки з дому. Теж саме і з пластиковими кришечками. Бокси для їх збирання ставлять мейкерські лабораторії, які потім роблять з них цікаві та корисні штуки — лампи, підставки тощо. Сортування сміття — одна з найпопулярніших екоініціатив, яку зараз активно впроваджують ІТ-компанії. Як правило, компанії ставлять 3-4 контейнери для сміття (папір, пластик, тетрапак, метал). У великих компаніях бокси для паперу встановлюють ще біля кожного принтера. Наступна ініціатива від ТІ компаній - поступова відмова від пластику (перехід на паперові чи власні стаканчики). [22]

Складним моментом у запровадженні зеленого офісу є те, більшість працівників рідко вимикають робочі станції, бо потім в разі чого, їх неможливо буде ввімкнути з дому, чи перед приходом на роботу. Потрібно не витрачати зайву електроенергію.

Майже всі ІТ-компанії, стикалися і стикаються зі складнощами у впровадженні екоініціатив. І в першу чергу це пов’язано з тим, що українці мають інші звички. Тому якщо зі збором батарейок все ще не так погано, то ініціатива із сортуванням сміття виявилася своєрідним викликом для українського ІТ. Для цього компанії створюють інформаційні акції, розробляють мобільні застосунки, проводять волонтерські події з популяризації дбайливого ставлення до природи та навколишнього середовища. Ще ряд важливих моменів: популяризація екологічного транспорту (вело- та електро-) серед своїх працівників та встановлення сонячних панелей для забезпечення офісу енергією.

6.3. Висновки до розділу

В цьому розділі розглянуто метод екологічної статистики і проблему екологічної відповідальності та організації «Зеленого офісу».

ВИСНОВКИ

В рамках даної роботи були досягнуті наступні результати:

– розроблено ядро для роботи з реляційною схемою і користувачами БД;

– розроблений алгоритм для автоматичної трансляції схем на основі деталізованих правил перетворення;

– створено прототип програмного модуля, представлений у вигляді плагіна-інструменту в середовищі Visual Studio, який дозволяє транслювати ERM-схему в реляційну схему. На даний момент модуль може легко інтегруватися в середовище Visual Studio для отримання реляційних схем, на основі яких можна генерувати SQL-скрипти для побудови баз даних;

– реалізований репозиторій для ERM-схем в CASE-системі Oracle Designer. Для цього був спроектований і впроваджений набір призначених для користувача розширень, створений механізм доступу для роботи з репозиторієм зовнішніх додатків. Отриманий репозиторій надає хорошу основу для створення і розвитку інструментарію, що підтримує модель ERM. Він може бути цікавим не тільки з чисто практичної, а й дослідницької точки зору, надаючи можливість на практиці перевірити придатність тих чи інших пропозицій щодо вдосконалення самої семантичної моделі.

Подальшим розвитком проекту можна виділити завдання, пов'язані з доповненням інструменту механізмом, що здійснює генерацію реляційних схем, а також інтеграцією інструменту з графічним редактором ERM-схем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных: полный курс.: пер. с англ. М.: "Вильямс", 2004. 1088 с.

2. Chen P. P. The entity-relationship model – towards a unified view of data. ACM Transactions on Database Systems. March 1976.Vol. 1, No. 1. P. 9–36;

3. Чен П. Модель «сущность – связь» - шаг к единому представлению о даннях. СУБД. 1995. № 3. С. 137-158

4. Codd, E. F. Data Models in Database Management. *Workshop in Data Abstraction, Databases, and Conceptual Modelling*: international conference, June 23–26, 1980, Pingree Park, Colorado: proceedings. 1980. P. 18–36.

5. Chen, P. P. Entity-relationship modeling: historical events, future trends, and lessons learned. *Entity-Relationship Approach to Software Engineering*: international conference, November 27–30, 2001, Yokohama, Japan: proceedings. 2001. P. 71–77.

6. Буй Д.Б., Сільвейструк Л.М. Формалізація моделі "сутність-зв'язок". Монографія. Київ : Київський університет, 2011, 175 с.

7. Бабанов А.М. Семантическая модель «Сущность – Связь – Отображение». Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2007. № 1. С. 77–91.

8. Бабанов А.М. Базовые и производные структурные понятия ERM-модели данных и изоморфное отношение между ними. Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2012. № 4. С. 117–126.

9. Бабанов А.М. Теория семантически значимых отображений и ее применение для проектирования реляционных баз данных: автореф. дис. канд. техн. наук. Том. гос. ун-т.Томск, 2004. 12 с.

10. Бабанов А. М. ERM- модель данных и новые возможности в проектировании баз данных. *Информационные технологии и математическое моделирование (ИТММ-2016)*: Материалы XV международной конференции им. А. Ф. Терпугова (12-16 сентября 2016 г.). Томск: Изд-во Том. ун-та, 2016.Ч.2. с. 80-85.

11. Колетски П., Дорси П. Oracle Designer. Настольная книга пользователя. М.: Лори, 1999. 592 с.

12. Oracle Designer. URL : http://www.interface.ru/oracle/des2000x.htm (дата звернення: 23.11.2019)

13. Donald K. Burleson. Oracle Internals: Tips, Tricks, and Techniques for DBAs. Auerbach Publications, 2001. 896 p.

14. Родзоняк А.В. Трансформація схем баз даних з ERM-моделі в реляційну. *Інформаційні моделі, системи та технології*: Праці VII наук.-техн. конф. (Тернопіль, 11-12 грудня 2019 р.) Тернопіль, 2019. С. 88.

15. CASE-засоби. Загальна характеристика і класифікація, Комерція, Різне, статті. URL : http://easy-code.com.ua/2012/08/case-zasobi-zagalna-xarakteristika-i-klasifikaciya-komerciya-rizne-statti/ (дата звернення: 23.11.2019)/

16. Шилдт Г. C#. Учебный курс. СПб.: Питер; К.: BHV, 2002. 512 с.

17. World Wide Web Consortium. XML Technology. URL: http://www.w3.org/standards/xml/ (дата звернення: 25.11.2019)/

18. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. 267 с.

19. Сакевич В.Ф., Поліщук О.В. Цивільна оборона. Теоретичні основи. Навчальний посібник. — Вінниця : ВНТУ, — 2009. — 136 с.

20. ДБН В.1.2. - 4 - 2006 «Інженерно-технічні заходи цивільного захисту (цивільної оборони)»

21. Тарасова В.В. Екологічна статистика.Київ: Центр учбової літератури, 2008. 392 с.

22. Від збору паперу до сонячної електростанції на терасі. Екоініціативи українських IT-компаній. URL: https://dou.ua/lenta/articles/eco-it-companies/ (дата звернення: 20.11.2019).

Додаток А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ »

11–12 грудня 2019 року

ТЕРНОПІЛЬ
2019

УДК 004.65
А. Родзоняк
Тернопільський національний технічний університет імені Івана Пулюя

ТРАНСФОРМАЦІЯ СХЕМ БАЗ ДАНИХ З ERM-МОДЕЛІ В РЕЛЯЦІЙНУ

UDC 004.65
A. Rodzoniak
(Ternopil Ivan Puluj National Technical University, Ukraine)

TRANSFORMATION OF ERM-MODEL DATABASE IN RELATIONAL MODEL

Зручним підходом є проектування БД, що відповідає всім особливостям предметної області (ПрО), ще на перших етапах розробки с використанням спеціальних інструментів. При цьому всі подальші етапи побудови схеми БД повинні проходити в автоматичному режимі або з мінімальною участю розробника. При цьому підході найбільш перспективною вбачається семантична методика проектування БД [1].

З моделі "сутність-зв'язок" (ER - Entity - Relationship) можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найбільш загальною. Модель «сутність - зв'язок - відображення» або ERM-модель (англ. Entity - Relationship - Mapping) продовжує розвиток ER-моделі в бік більш детального опису закономірностей ПрО.

Поєднання виразної семантичної моделі даних і потужного набору правил її перетворення в логічні моделі СУБД в стані забезпечити ідеальну семантичну методику, з використанням якої можна буде отримувати бездоганні схеми БД. Наступним етапом поліпшення процесу семантичного моделювання є автоматизація даного перетворення і реалізація семантичної методики у вигляді CASE-засобу - програмної системи, що автоматизує роботу проектувальника схем БД.

Методика трансформації ERM-схеми в реляційну схему передбачає послідовне виконання трьох груп правил для забезпечення результату з найкращою якістю: правила породження структур є основою для побудови структур даних, які задовольняють всі потреби користувачів; правила породження обмежень цілісності забезпечують для структур максимально повне перенесення обмежень цілісності ERM-схеми; оптимізаційні правила спрямовані на виключення надлишкових структур і обмежень цілісності [1].

Для розробки прототипу програмного засобу для трансформації ERM-схеми в реляційну схему пропонується використати інтегроване середовище розробки Visual Studio. Корпорація Microsoft в даний час серйозно займається просуванням і популяризацією .NET, Тому розробка додатків в середовищі .NET є перспективним напрямком, і написаний код буде придатний для повторного використання. Базовою мовою програмування варто обрати C #, яка є найбільш зручною мовою для розробки призначених для користувача додатків і при цьому досить простою для роботи не тільки з графікою, але і з зберіганням даних. Для інтеграції в середовище розробки Visual Studio і візуального представлення результатів роботи спроектованого програмного засобу, потрібно розробити плагін, який надає інтерфейси взаємодії з CASE-системою (Oracle Designer) і середовищем розробки. Основні функції плагіна: вибір необхідної ERM-схеми для завантаження; запуск процесу трансляції; візуалізація побудованої реляційної схеми у вигляді тексту; збереження отриманої реляційної схеми в файл. На основі реляційних схем можна генерувати SQL-скрипти для побудови БД.

Література

1. Бабанов А. М. ERM- модель данных и новые возможности в проектировании баз данных // Информационные технологии и математическое моделирование (ИТММ-2016): Материалы XV Международной конференции им. А. Ф. Терпугова (12-16 сентября 2016 г.). Томск: Изд-во Том. ун-та, 2016. Ч. 2. С. 80–85