dComFra

Digital competence framework for Ukrainian teachers and other citizens

598236-EPP-1-2018-1-LT-EPPKA2-CBHE-SP

Модуль 21 – Удосконалені бази даних

(Advanced Database)

Версія 1.0

Delivery date: 15 October 2020

Prepared by: Chernivtsi National University (ChNU) (P-10)

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Навчальний посібник з курсу «Цифровий маркетинг», підготовленого в рамках реалізації проекту ERASMUS+ KA2 dComFra **598236-EPP-1-2018-1-LT-EPPKA2-CBHE-SP**

Deliverable 4.2: Developed dComFra resources
Duration of deliverable 4.2 – from 15-06-2020 to 25-02-2021

Contributor – Yuriy Fedkovych Chernivtsi National University (ChNU)
Evaluator – Taras Shevchenko National University of Kyiv (TSNUK)
Consults – University Politechnica of Bucharest (PUB)

Editors	Heorhii Vorobets, Serhii Balovsiak
Contributors	Serhii Balovsiak, Iryna Lisovenko, Yuliya Tanasyuk

Website: *https://dcomfra.vdu.lt*
Facebook: *dComFra Project, #dComFra*

CONTENTS

ВСТУП

В модулі розглянуто основні концепції, структуру, принципи побудови та використання удосконалених реляційних баз даних.

Мета модуля полягає в наданні здобувачам ключових концепцій, навичок та вмінь для створення, модифікації та використання удосконалених реляційних баз даних.

Здобувачі, що успішно опанують усі теми модуля, зможуть:

• Розуміти принципи роботи систем керування базами даних, ключові концепції побудови баз даних та їх життєвого циклу.

• Проектувати та програмно реалізовувати реляційні бази даних, виконувати ефективний пошук даних у таблицях і контролювати допустимість значень для вхідних даних.

• Формувати SQL запити для створення та модифікації таблиць. Виконувати додавання, модифікацію та видалення записів у таблицях. Використовувати у запитах різні види умов, символи підстановки, обчислення та групування інформації.

• Розширювати функціональність форм за рахунок використання різних типів елементів керування та підформ.

• Показувати у звітах дані таблиць і результати обчислень у заданих форматах. Відображати структуровану інформацію на колонтитулах і створювати підзвіти.

• Виконувати автоматизацію типових операцій з базами даних з використанням макросів. Виконувати імпорт даних різних типів у базу даних. Використовувати хмарні сервіси для роботи з базами даних.

1. КОНЦЕПЦІЯ БАЗ ДАНИХ

1.1 Проектування, створення та використання баз даних

1.1.1 Принципи побудови систем керування базами даних. Типи моделей баз даних: ієрархічна, мережева, реляційна, об'єктно-орієнтована

База даних (БД) – це набір структурованих і логічно взаємопов'язаних даних, які зберігаються в електронних сховищах та призначені для задоволення інформаційних потреб користувачів. Звичайно БД створюються для зберігання і доступу до даних, які містять відомості про певну предметну область [1-2]. У БД дані зберігаються у заданому форматі та впорядковуються відповідно до моделі організації даних. Доступ до таких даних здійснюється з одного або з кількох комп'ютерів. Головне призначення БД – гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладним програмам.

За стандартом ISO/IEC 2382:2015 [3] база даних (англ. database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами. В загальному випадку БД містить схеми, таблиці, збережені процедури та інші об'єкти.

БД звичайно застосовуються у складі інформаційних систем (наприклад, веб-сайтів, інтернет-магазинів, довідкових систем, систем комп'ютерного розпізнавання облич), що забезпечує такі переваги:

1. Спільне збереження даних для всіх прикладних програм.

2. Усувається багаторазове введення та дублювання одних і тих самих даних.

3. Не виникають проблеми зміни прикладних програм у зв'язку із заміною фізичних пристроїв або зміною структури даних.

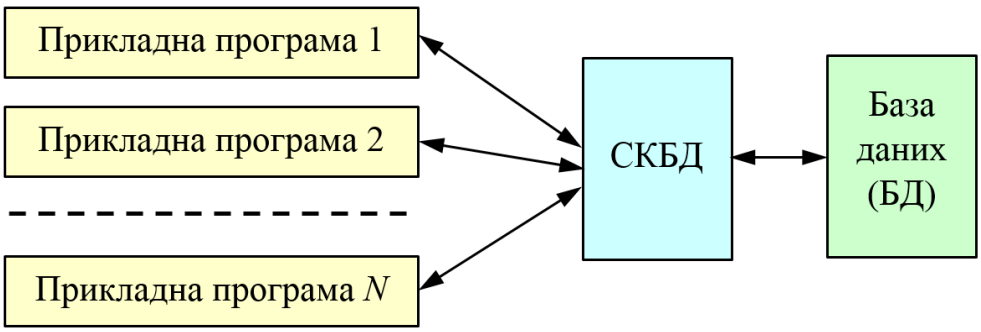
4. Підвищується рівень надійності збереження та захищеності інформації.

5. Зменшується надлишковість даних.

У порівнянні з електронними таблицями БД характеризуються вищою структурованістю та захищеністю даних, а також збереженням даних не тільки в оперативній пам'яті комп'ютера, але й в енергонезалежній пам'яті (наприклад, на жорстких дисках).

Системи керування базами даних (СКБД) – це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, контроль, керування та використання баз даних (за стандартом ISO/IEC 2382:2015 [3]). СКБД відіграє роль інтерфейсу між прикладними програмами і базами даних, що забезпечує їх незалежність (рис. 1.1); при цьому доступ до БД користувачі отримують за допомогою відповідних прикладних програм.

Рисунок 1.1: Забезпечення незалежності прикладних програм і бази даних



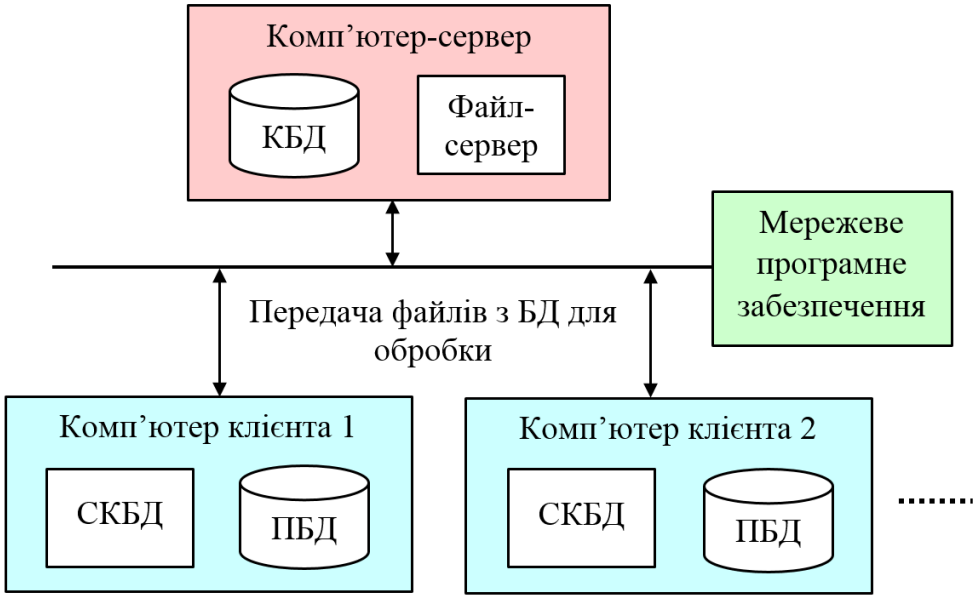
За призначенням розрізняють три основних види СКБД: універсальні, спеціалізовані та розроблені для конкретного замовника. Спеціалізовані СКБД створюються для керування БД конкретного призначення: бухгалтерськими, складськими, банківськими та ін.

За архітектурою розрізняють СКБД «файл-сервер» та «клієнт-сервер».

Архітектура «файл-сервер» передбачає виділення одного з комп'ютерів мережі як головного (сервер) (рис. 1.2) [4]. На такому комп'ютері зберігається спільна централізована БД і виконується керуюча програма. Усі інші комп'ютери мережі (клієнти) виконують функції робочих станцій, за допомогою яких та відповідних програм підтримується доступ користувацької системи до БД. Файли БД відповідно до призначених для користувача запитів передаються на робочі станції, де в основному і проводиться обробка даних.

Рисунок 1.2: Структура інформаційної системи з архітектурою «файл-сервер»;

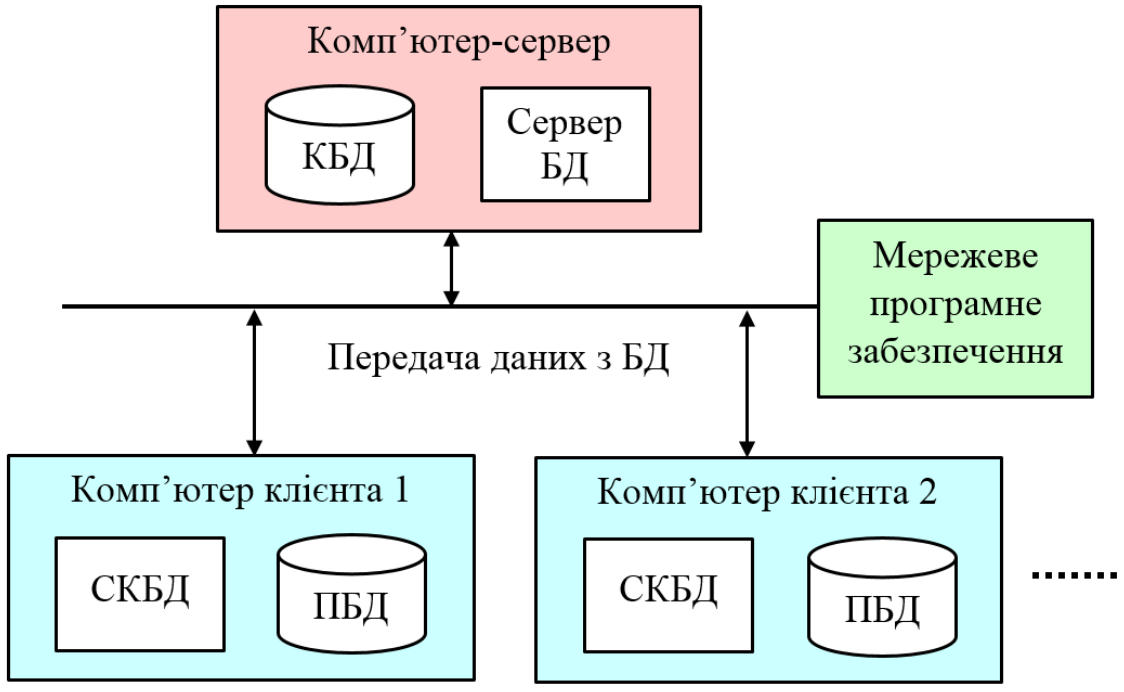
КБД – корпоративна БД, ПДБ – персональна БД



Недоліком такої архітектури є висока інтенсивність передачі оброблюваних даних (трафік) за рахунок передачі надлишкових даних: клієнтам передаються всі файли БД, навіть якщо їм потрібен 1 запис з таблиці (яка може містити, наприклад, 100 000 записів). Тому файл-серверні СКБД (наприклад, DBase, FoxPro, Paradox, MS Access) застосовуються для вирішення відносно простих прикладних задач.

Архітектура «клієнт-сервер» застосовується в сучасних мережевих інформаційних системах для роботи із загальною базою даних [4]. Як сервер баз даних використовується комп'ютер (або комп'ютери), який містить БД, СКБД та пов'язане з ними програмне забезпечення (рис. 1.3). Сервер призначений для надання користувачам (клієнтам) інформаційної системи доступу до БД. Клієнти, які працюють із даними (на різних комп'ютерах мережі), надсилають відповідні запити серверу. Сервер їх отримує, опрацьовує, та надсилає відповідь клієнту. Сучасні СКБД (MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird та інші) працюють відповідно до цієї архітектури. Сервер баз даних, як правило, є достатньо потужною багатопроцесорною системою, яка використовує масиви дисків RAID для підвищення надійності зберігання даних.

Рисунок 1.3: Структура інформаційної системи з архітектурою «клієнт-сервер»; КБД – корпоративна БД, ПДБ – персональна БД



Перевагою архітектури «клієнт-сервер» є поєднання централізованого зберігання, обслуговування і колективного доступу до спільної корпоративної інформації з індивідуальною роботою користувачів над персональною інформацією. Висока швидкодія клієнт-серверної архітектури забезпечується тим, що клієнту або серверу надсилається не вся БД, а тільки її окремі записи.

У загальному випадку БД є сукупністю інформації, організованої у вигляді множин, де кожна множина містить записи уніфікованого виду. Самі записи складаються з полів. Звичайно множини називають таблицями, записи – рядками таблиць, а поля – стовпцями.

Проектування БД починається з етапу концептуального проектування, під час якого здійснюється формальний опис предметної області, виявляються сутності предметної області, їх характеристики та взаємозв'язки. У результаті концептуального проектування будується спрощена та узагальнена структура БД, яка конкретизується у процесі подальшої розробки БД.

Фізично БД зберігаються на жорсткому диску у вигляді одного або кількох файлів (залежно від використаної СКБД).

За характером організації зберігання даних розрізняють локальні та розподілені БД.

Розглянемо детальніше основні типи моделей баз даних (з різною структурою): ієрархічну, мережеву, реляційну, об'єктно-орієнтовану.

Ієрархічні бази даних підтримують деревоподібну організацію інформації [4, 5], тобто зв'язки між даними можна описати за допомогою впорядкованого графу без циклів (дерева) (рис. 1.4). Кожна вершина графу є записом, яка містить інформацію про певний екземпляр об'єкта (рис. 1.5). Зв'язки між записами описуються у вигляді відносин «предок/ нащадок», при цьому у кожного запису є тільки один батьківський запис. Це допомагає підтримувати цілісність за посиланнями: коли запис видаляється з дерева, всі його нащадки також видаляються.

Наприклад, в ієрархічній БД, яка описує структуру університету (рис. 1.5), корінь дерева (кореневий запис, рівень дерева 1) містить загальну інформацію про університет. Нащадками кореневого запису є записи з інформацію про факультети (рівень дерева 2), а їх нащадками є записи з інформацією про кафедри (рівень дерева 3). Поля записів зберігають дані певних типів (наприклад, числові та символьні).

Рисунок 1.4: Приклад дерева ієрархічної бази даних

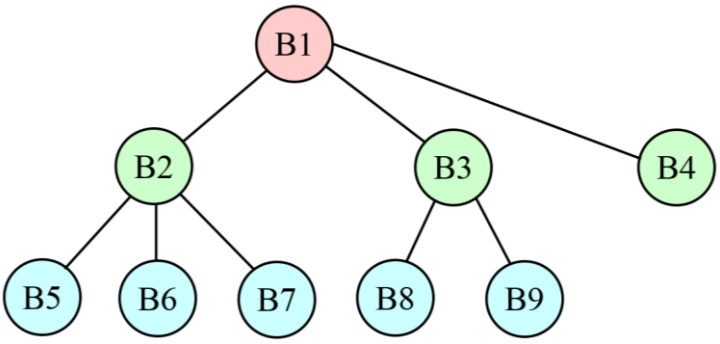
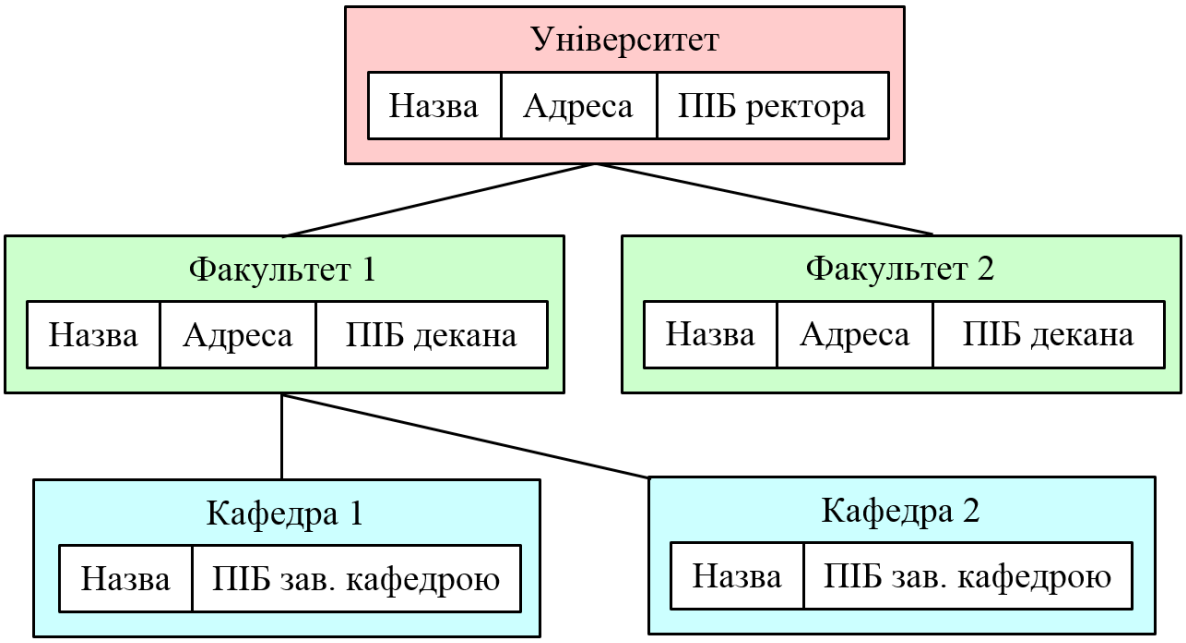


Рисунок 1.5: Приклад ієрархічної бази даних



Перегляд всіх елементів ієрархічної БД звичайно виконується зверху вниз та зліва направо. Реальні ієрархічні БД мають значно складнішу структуру, проте в них завжди підтримується принцип ієрархічності.

Ієрархічні бази даних мають централізовану структуру, тобто безпеку даних легко контролювати. Проте, потрібні певні знання про фізичний порядок зберігання записів, оскільки відношення предок/ нащадок реалізуються у вигляді фізичних покажчиків з одного запису на інший. Це означає, що пошук запису здійснюється методом прямого обходу дерева. Звідси випливає необхідність правильно впорядковувати записи, щоб час їх пошуку був мінімальним. Проте, не всі відношення, що існують в реальному світі, можна виразити в ієрархічній базі даних. Відношення «один до багатьох» є природними, але практично неможливо описати відношення «багато до багатьох» або ситуації, коли запис має кілька предків.

До переваг ієрархічної моделі БД належить ефективне використання пам'яті комп'ютера і висока швидкість виконання основних операцій над даними. Ієрархічна модель БД є ефективною, якщо її дані утворюють певну ієрархію. Таке ієрархічне впорядкування даних спостерігається, зокрема, для структури установи (рис. 1.5), структури пристрою (наприклад, комп'ютера).

Недоліками ієрархічної моделі є її громіздкість при обробці інформації з складними логічними зв'язками, коли дані не утворюють чіткої ієрархії.

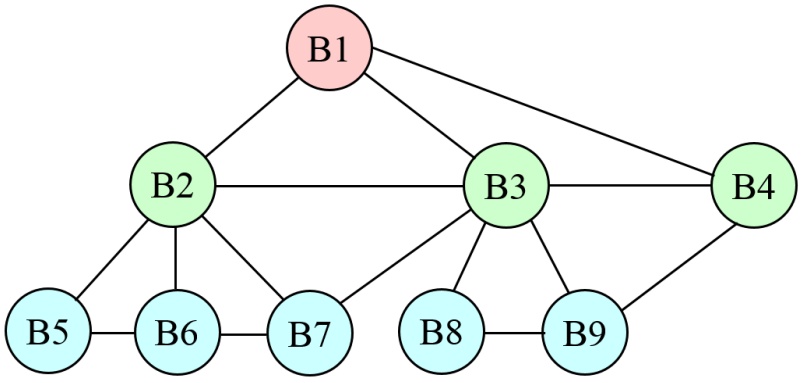
Ієрархічні БД є спеціалізованими і мають вузьку сферу використання, тому на ієрархічній моделі заснована відносно незначна кількість СКБД, наприклад, IMS, PC/Focus, Team/Up, Data Edge.

Мережева модель даних [4, 5] розширює ієрархічну модель, дозволяючи групувати зв'язки між записами в множини. Як і в ієрархічній моделі, зв'язки ведуться від батьківського запису до дочірнього, але при цьому підтримується множинне успадкування (кожен об'єкт може мати більше одного предка). Тобто мережева модель дозволяє відображати різноманітні зв'язки елементів у вигляді довільного графу (рис. 1.6).

Дотримуючись специфікації CODASYL (англ. COnference on DAta SYstems Languages – Конференція по мовам систем обробки даних) [4], мережева модель підтримує DDL (Data Definition Language – мову визначення даних) і DML (Data Manipulation Language – мову обробки даних). Це спеціальні мови, призначені для визначення структури бази даних і складання запитів. У мережній моделі допускаються відношення «багато до багатьох», а записи не залежать один від одного. При видаленні запису видаляються всі його зв'язки, але не самі пов'язані записи. Для підтримки цілісності БД зв'язки встановлюються між існуючими записами.

Перевагою мережевої моделі даних (порівняно з ієрархічною) є можливість утворення довільних зв'язків між записами. Недоліком мережевої моделі є висока складність та жорсткість схеми БД, побудованої на її основі. Тому СКБД, побудовані на мережевій моделі (наприклад, IDMS, db_Vistal), не отримали широкого поширення.

Рисунок 1.6: Приклад графу мережевої бази даних



Реляційні бази даних на даний час є досить поширеними. Реляційна модель даних запропонована співробітником фірми IBM Е. Коддом і заснована на понятті відношення (relatiuon) [1]. Відношення є множиною елементів, які називаються кортежами. При програмуванні відношення реалізується за допомогою двовимірної таблиці (рис. 1.7). Таблиця має рядки (записи) і стовпці (поля, колонки). Кожен рядок таблиці має однакову структуру і складається з полів. Рядкам таблиці відповідають кортежі, а стовпцям (полям) – атрибути відношення (табл. 1.1, табл. 1.2).

Рисунок 1.7: Приклад реляційної бази даних «Монітори»; таблиця «Фірма-виробник» – батьківська, таблиця «Монітор» – дочірня

Монітор

Ключ	Назва	Діагональ, дюйми	Зовнішній ключ
1	ED242QRA	23.6	1
2	C24F396F	23.5	2
3	C27JG54	27	2
4	LC32JG50Q	32	2
5	248E9QHSB	23.6	3
6	272E1CA	27	3

Фірма-виробник

Ключ	Назва
1	Acer
2	Samsung
3	Philips

За допомогою однієї таблиці зручно описувати прості зв'язки між даними, а саме ділення одного об'єкту (сутності, системи), інформація про який зберігається в таблиці, на множину підоб'єктів, кожному з який відповідає рядок (запис) таблиці. При цьому кожен з підоб'єктів має однакову структуру та властивості, які описуються відповідними значеннями полів таблиці.

Наприклад, в таблиці, що містить опис моделей моніторів, у всіх записів буде однаковий набір полів: первинний ключ (скорочено «ключ»), назва моделі, розмір по діагоналі, фірма-виробник і т.д. (рис. 1.7). Такі таблиці легко зображувати в графічному вигляді. Оскільки в рамках однієї таблиці не вдається описати складніші логічні структури даних, тому застосовується зв'язування таблиць за допомогою первинних та зовнішніх ключів (рис. 1.7).

Таблиця із зовнішнім ключем називається дочірньою (наприклад, таблиця «Монітор» на рис. 1.7), а таблиця, яка містить тільки первинний ключ і зв'язана з дочірньою, називається батьківською (наприклад, таблиця «Фірма-виробник» на рис. 1.7).

Таблиця 1.1: Елементи реляційної моделі бази даних [1]

No	Елементи реляційної моделі	Терміни, що застосовується при реалізації БД
1.	Атрибут	Стовпець, поле
2.	Відношення (множина записів, сутність*)	Таблиця
3.	Домен	Множина допустимих значень стовпця
4.	Кортеж	Рядок, запис
5.	Первинний ключ	Унікальний ідентифікатор
6.	Схема відношення	Назви стовпців
7.	Тип даних	Тип значень стовпця

*поняття «сутність» тотожне поняттю «таблиця» при концептуальному проектуванні, проте у загальному випадку «сутність» є описом властивостей об'єкта.

Таблиця 1.2: Українсько-англійський словник термінів за тематикою реляційних баз даних

No	Український термін	Англійський термін	Приклад
1.	Атрибут	Attribute	
2.	База даних	Database	
3.	Відношення	Relation	
4.	Ідентифікуючий зв'язок	Identifying Relationship	Між батьківською та дочірньою таблицями
5.	З'єднання (таблиць)	Join	Внутрішнє об'єднання (inner join)
6.	Зв'язок (між таблицями)	Relationship	«один-до-багатьох» («one-to-many»)
7.	Зовнішній ключ	Foreign Key (FK)	
8.	Кортеж	Cortege	
9.	Мова структурованих запитів	Structured Query Language – SQL	
10.	Неідентифікуючий зв'язок	Non-Identifying Relationship	
11.	Первинний ключ	Primary Key (PK)	
12.	Система керування базами даних (СКБД)	Database management system	
13.	Сутність	Entity	
14.	Таблиця	Table	

Рядки таблиць можуть бути пов'язані один з одним одним з 4 способів:

1. Зв'язок «один до одного». У цьому випадку рядок першої таблиці відповідає однієї єдиному рядку другої таблиці. На діаграмах такий зв'язок позначається записом 1: 1.

2. Зв'язок «один до багатьох» означає ситуацію, коли рядок однієї таблиці відповідає кільком рядкам іншої таблиці (рис. 1.7). Це найбільш поширений тип зв'язків, який позначається на діаграмах записом 1: N.

3. Зв'язок «багато до одного» означає ситуацію, коли кільком рядкам однієї таблиці відповідає один рядок іншої таблиці (рис. 1.7). Такий зв'язок позначається записом N : 1 і є аналогічним до зв'язку «один до багатьох», але відносно іншої таблиці.

4. Зв'язок «багато до багатьох» означає, що рядки першої таблиці можуть бути пов'язані з довільним числом рядків у другій таблиці. Такий зв'язок позначається як N: M.

Зв'язки між таблицями можуть бути ідентифікуючими (identifying) та неідентифікуючими (non-identifying). При ідентифікуючому зв'язку дочірня таблиця є залежною від батьківської, а при неідентифікуючому – незалежною. Батьківська таблиця є незалежною від дочірньої у будь-якому випадку. Це означає, що для кожного запису батьківської таблиці може бути

відповідний (зв'язаний) запис (або кілька записів) дочірньої таблиці (як показано на рис. 1.7), проте, деяким записам батьківської таблиці може не відповідати жоден запис дочірньої. Тобто, у таблицю «Фірма-виробник» на рис. 1.7 можна додати назву нової фірми, яка не буде пов'язана з жодним записом таблиці «Монітор». При ідентифікуючому зв'язку кожному запису дочірньої таблиці повинен відповідати певний запис батьківської (як показано на рис. 1.7). При неідентифікуючому зв'язку деякі записи дочірньої таблиці можуть бути не пов'язані з записами батьківської – це означає, що у таблиці «Монітор» можна додати модель монітору, для якої не буде вказано фірму-виробник (рис. 1.7).

Ідентифікуючі або неідентифікуючі зв'язки між таблиця вибираються залежно від особливостей предметної області та призначення БД; наприклад, якщо для кожної моделі монітора обов'язково вказувати фірму-виробника – використовуються ідентифікуючі зв'язки, а якщо ж для деяких моделей вказувати фірму-виробника не обов'язково – використовуються неідентифікуючі зв'язки.

У порівнянні з розглянутими вище моделями БД реляційна модель вимагає від СКБД вищого рівня складності. У таких СКБД робиться спроба позбавити програміста від виконання рутинних операцій по управлінню даними, характерних для ієрархічної та мережевої моделей. У реляційної моделі БД є централізованим сховищем таблиць, що забезпечує безпечний одночасний доступ до інформації багатьох користувачів. У рядках таблиць частина полів містить дані, що відносяться безпосередньо до запису, а частина – посилання на записи інших таблиць. Таким чином, зв'язки між записами є невід'ємною властивістю реляційної моделі.

У реляційної моделі досягається інформаційна та структурна незалежність. Записи не пов'язані між собою настільки, щоб зміна одного з них спричинила зміну інших, а зміна структури БД не обов'язково призводить до перекомпіляції працюючих з нею додатків. У реляційних СКБД застосовується мова структурованих запитів SQL, що відносно просто дозволяє формувати різноманітні запити. При застосуванні різних реляційних СКБД певна проблема полягає в тому, що окремі СКБД використовують не тільки стандартні засоби стандарту SQL, але набір команд, унікальних для даної СКБД. Такі унікальні команди враховують специфіку СКБД і збільшують продуктивність обробки даних, але ускладнюють задачу програмістам, які намагаються перейти від однієї СКБД до іншої.

Прикладами реляційних СКБД є MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird, DBase, FoxPro, Paradox, MS Access та інші. При цьому останні версії багатьох СКБД мають певні властивості об'єктно-орієнтованих систем, тому такі СКБД часто називаються об'єктно-реляційними (наприклад, СКБД PostgreSQL, Oracle).

Переваги реляційної моделі даних полягають в простоті, зрозумілості та зручності фізичної реалізації за допомогою комп'ютера, що зумовило широке розповсюдження СКБД з такою моделлю даних.

Основними недоліками реляційної моделі є відсутність стандартних засобів ідентифікації окремих записів, а також складність опису ієрархічних та мережевих записів.

Об'єктно-орієнтовані бази даних (ООБД) дозволяють програмістам інтерпретувати всі свої інформаційні сутності як об'єкти. Це є важливим для практики, оскільки більшість сучасних мов програмування є об'єктно-орієнтованими. В ООБД моделі даних та методи, що їх обробляють, об'єднуються в структури, які називаються об'єктами. Типи об'єктів називаються класами.

Для стандартизації ООБД створено некомерційний консорціум виробників ООБД (ODMG – Object Data Management Group) [6-8]. Стандарти ODMG описують:

1. Об'єктно-орієнтовану модель (Object Oriented Model, OOM)

2. Мову опису об'єктів (Object Definition Language, ODL)

3. Об'єктну мову запитів (Object Query Language, OQL)

4. Методи зв'язування ООБД з об'єктно-орієнтованими мовами програмування (C++, Java та ін.)

Структуру ООБД графічно можна зобразити у вигляді дерева, вузлами якого є об'єкти. Властивості об'єктів описуються або стандартним типом даних (наприклад, рядковим – string), або типом, який визначається користувачем (визначається як клас – class).

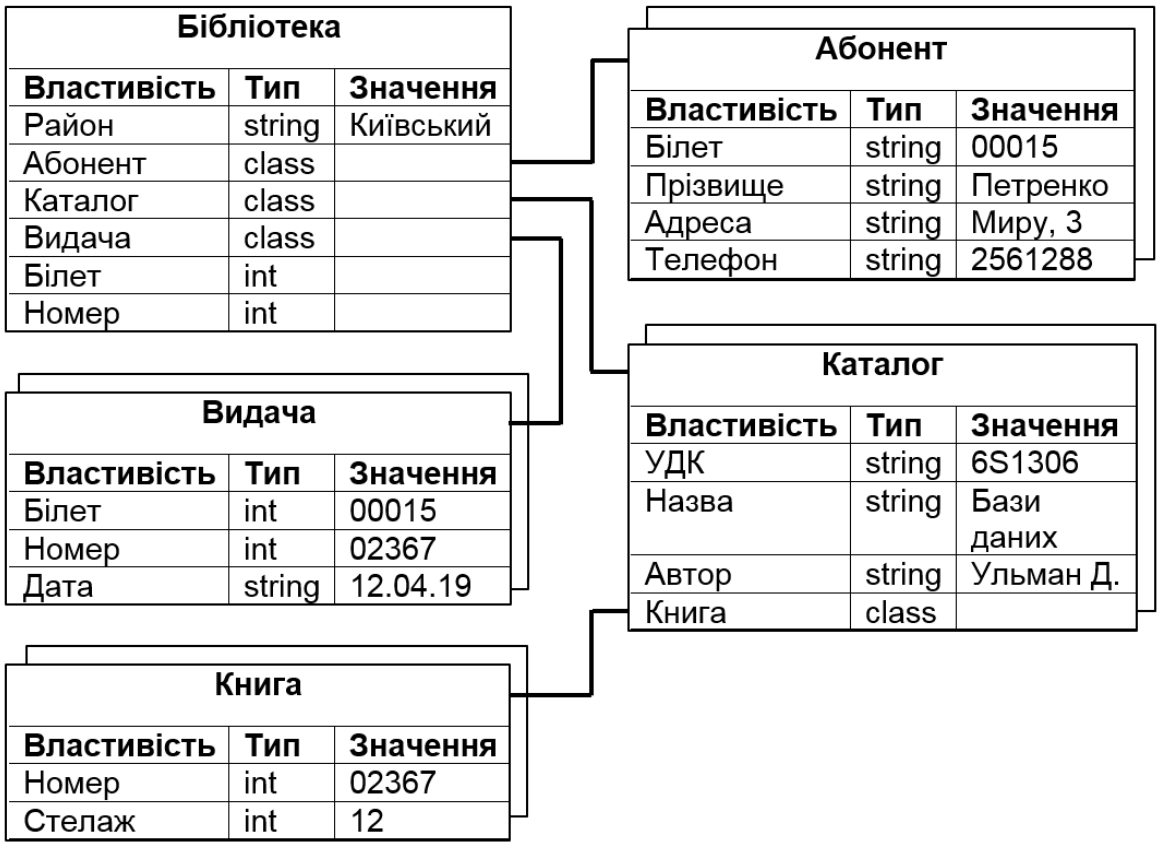
Розглянемо приклад логічної структури ООБД (рис. 1.8) [4]. Об'єкт типу «Бібліотека» є батьківським для об'єктів-екземплярів класів «Абонент», «Каталог», «Видача». Різні об'єкти типу «Книга» можуть мати одного або кількох батьків. Логічна структура ООБД зовні схожа на структуру ієрархічної БД. Основна відмінність між ними полягає в методах маніпулювання даними.

Для виконання операцій над даними в розглянутій моделі ООБД застосовуються логічні операції, підсилені об'єктно-орієнтованими механізмами інкапсуляції, спадкування та поліморфізму:

1. Інкапсуляція обмежує область видимості імені властивості межами того об'єкта, в якому вона визначена. Наприклад, якщо в об'єкт «Каталог» (рис. 1.8) додати властивість, яка задає номер телефону автора книги і має назву «Телефон», то в об'єктів «Абонент» і «Каталог» будуть властивості з однаковими назвами. Тому значення властивості буде визначатися тим об'єктом, в якому воно інкапсульовано.

2. Спадкування розповсюджує область видимості властивості на всіх потомків об'єкта. Наприклад, всім об'єктам типу «Книга», які є нащадками об'єкта типу «Каталог», можна додати властивості об'єкту-предка («УДК», «Назва» та ін.) (рис. 1.8).

Рисунок 1.8: Логічна структура ООБД «Бібліотека»



Бібліотека

Властивість	Тип	Значення
Район	string	Київський
Абонент	class	
Каталог	class	
Видача	class	
Білет	int	
Номер	int	

Абонент

Властивість	Тип	Значення
Білет	string	00015
Прізвище	string	Петренко
Адреса	string	Миру, 3
Телефон	string	2561288

Видача

Властивість	Тип	Значення
Білет	int	00015
Номер	int	02367
Дата	string	12.04.19

Каталог

Властивість	Тип	Значення
УДК	string	6S1306
Назва	string	Бази даних
Автор	string	Ульман Д.
Книга	class	

Книга

Властивість	Тип	Значення
Номер	int	02367
Стелаж	int	12

3. Поліморфізм в об’єктно-орієнтованих мовах програмування означає здатність одного і того ж програмного коду працювати з різними даними. Тобто, це означає допустимість в об’єктах різних типів мати методи з однаковими іменами. Під час виконання об’єктної програми одні і ті ж методи оперують з різними об’єктами в залежності від типу аргументу. По відношенню до описаної ООБД (рис. 1.8) поліморфізм означає, що об’єкти класу «Книга», які мають різних батьків з класу «Каталог», можуть мати різний набір властивостей. Відповідно, програма роботи з об’єктами класу «Книга» можуть мати поліморфний код.

Створення та модифікація БД супроводжується автоматичним формуванням і наступною корекцією індексів (індексних таблиць), які містять інформацію для швидкого пошуку даних.

Основною перевагою об’єктно-орієнтованої моделі даних (порівняно з реляційною) є можливість враховувати інформацію про складні взаємозв’язки об’єктів. Об’єктно-орієнтована модель даних дозволяє ідентифікувати окремі записи БД і визначати методи їх обробки.

Недоліками об’єктно-орієнтованої моделі є висока складність опису сутностей та відносно низька швидкість виконання запитів.

Постреляційна модель даних є розширеною реляційною моделлю, у якій відсутні обмеження на неподільність даних, що містяться у записах таблиць [4]. Завдяки відсутності обмеження на неподільність даних підвищується ефективність реалізації деяких програм. Постреляційна

модель даних допускає багатозначні поля, значень яких складаються з підзначень. Набір значень багатозрачних полів вважається самостійною таблицею, вбудованою в основну таблицю.

Бази даних NoSQL («Not Only SQL» – «не тільки SQL») використовують багато сучасних СКБД (наприклад, Oracle NoSQL Database, Cosmos DB, MongoDB, MySQL Cluster CGE) [7, 9, 10]. Особливістю баз даних NoSQL є висока швидкість обробки запитів, відсутність потреби у фіксованих схемах таблиць (спрощений механізм додавання полів), денормалізація збережених даних (не забезпечується атомарність /неподільність/ даних), добра масштабованість по горизонталі.

Бази даних NoSQL є ефективними при обробці великих обсягів даних у розподілених системах (наприклад, даних соціальних мереж), оскільки для розподілених систем неможливо одночасно забезпечувати узгодженість, доступність та гарантії допуску до розділів. З цієї причини багато баз даних NoSQL використовують так звану «можливу узгодженість» (яка не гарантує узгодженість в усіх випадках) для забезпечення гарантії доступності, а також розділення розділів із зниженим рівнем узгодженості даних. Тобто в базах даних NoSQL проблеми масштабованості, продуктивності та доступності вирішуються за рахунок денормалізації даних та зменшення їх рівня узгодженості.

1.1.2 Етапи життєвого циклу бази даних: збір і аналіз вимог, концептуальне, логічне та фізичне проектування, створення бази даних, введення даних, підтримка даних, пошук інформації

Життєвим циклом бази даних є неперервний процес, який починається з моменту прийняття рішення про створення бази даних і завершується при закінченні її експлуатації [4, 11, 12]. Основним нормативним документом, який регламентує життєвий цикл програмного забезпечення, є міжнародний стандарт ISO/IEC 12207 (International Organization of Standartization – Міжнародна організація по стандартизації, International Electrotechnical Commission – міжнародна комісія по електротехніці). Життєвий цикл БД складається з таких основних етапів (рис. 1.9).

Після прийняття рішення про створення БД виконується планування основних її етапів життєвого циклу, завдяки чому підвищується прогнозованість та ефективність як проектування, так і використання БД. За потреби з будь-якого етапу (крім першого) виконується повернення на кілька етапів назад, після чого уточнюються умови проходження відповідних етапів.

Рисунок 1.9: Основні етапи життєвого циклу БД

Розглянемо етапи життєвого циклу БД (рис. 1.9) детальніше.

1. **Етап «Збір і аналіз вимог до БД»** призначений для отримання й аналізу вимог до БД зі сторони користувачів (замовників) з урахуванням особливостей предметної області. При роботі з базами даних поняття «предметна область» означає частину реального середовища, яке описується та відображається в БД (підприємство, веб-сайт тощо). Тобто, предметна область окреслює сферу застосування конкретної БД. У багатьох випадках БД розробляються для збереження й аналізу даних певних підприємств (організацій), наприклад, інтернет-магазинів, навчальних закладів, бібліотек та ін., тому розглянемо етапи життєвого циклу БД підприємства (життєві цикли БД для інших предметних областей є подібними).

Збір і аналіз вимог до БД починається з аналізу існуючих інформаційних систем (ІС)-аналогів; на основі чого обґрунтовується доцільність зміни існуючої ІС шляхом створення БД. За потреби оцінюється обсяг потрібних для цього робіт і ресурсів, а також вартість проекту.

Необхідна для проектування БД інформація може бути зібрана такими способами:

• за допомогою опитування окремих співробітників підприємства, особливо фахівців у найбільш важливих областях його діяльності;

• за допомогою спостережень за діяльністю підприємства;

• за допомогою вивчення документів підприємства;

• за допомогою анкет, призначених для збору інформації з широкого кола користувачів;

• за рахунок використання досвіду проектування інших подібних систем.

• шляхом аналізу інтернет-ресурсів, пов'язаних з підприємством.

Отримана інформація може бути погано структурована і включати деякі неформальні заяви користувачів (на складі є велика кількість товару, попит на товар змінюється повільно і т.д.), які потрібно перетворити та представити у виді більш чітко сформульованих вимог. Зібрана інформація про предметну область повинна містити наступні компоненти:

- опис об'єктів предметної області;
- документацію підприємства (якщо це можливо);
- перелік функцій, які повинна виконувати БД;
- список вимог до БД (за потребою з указівкою їх пріоритетів).

На підставі цієї інформації для складних інформаційних систем складаються специфікації вимог (формальний перелік вимог) до БД. Збір і аналіз вимог є попереднім етапом перед проектуванням БД.

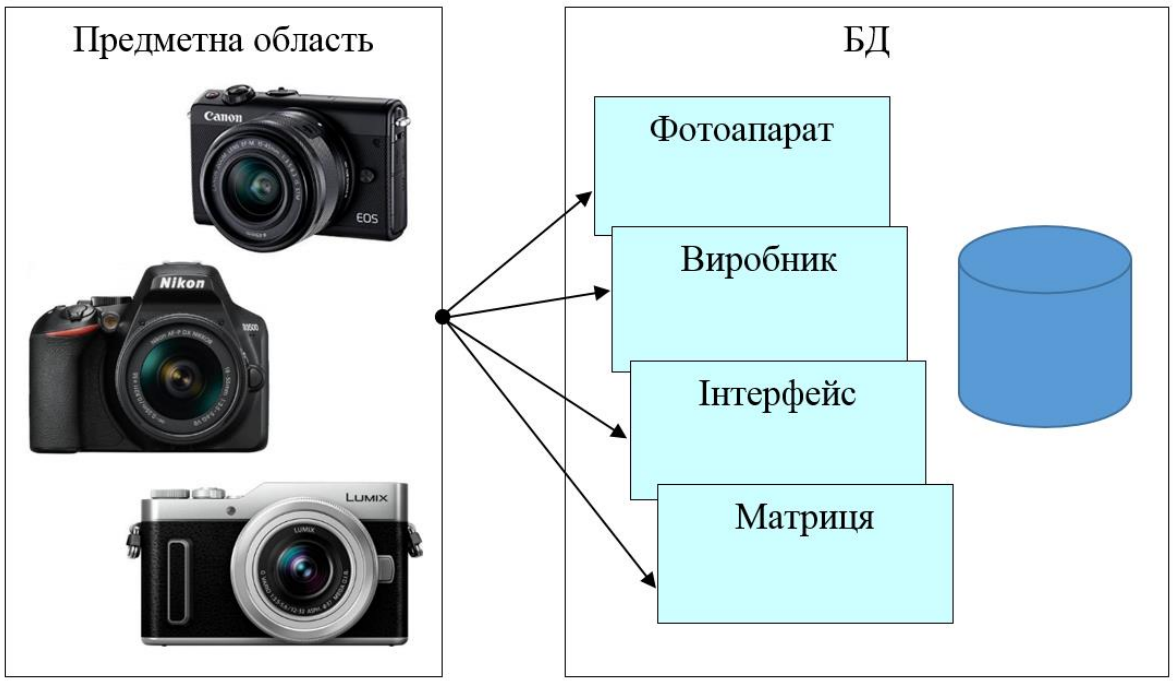
Розглянемо для прикладу життєвий цикл БД, призначеної для обліку фотоапаратів на складі магазину. У такому випадку етап збору і аналізу вимог до БД описується таким прикладом.

Приклад 1.1. Етап «Збір і аналіз вимог до БД»

Предметною областю є «Облік фотоапаратів на складі магазину».

Як інформаційні системи-аналоги розглянуто БД інтернет-магазинів фототехніки. Доцільність розробки БД для обліку фотоапаратів на складі магазину зумовлена величезною кількістю моделей фотоапаратів та їх параметрів, що робить обробку такої інформації у ручному режимі надзвичайно складною. Крім цього, існуючі БД використовують обмежений перелік параметрів фотоапаратів, який для деяких практичних задач потрібно змінити або розширити. Тобто, у БД повинен зберігатися достатньо широкий набір характеристик об'єктів предметної області (рис. 1.10).

Рисунок 1.10: Об'єкти предметної області та БД



Необхідна для проектування БД інформація зібрана шляхом аналізу веб-сайтів магазинів, які займаються продажем фотоапаратів. На основі такого аналізу виконано опис об'єктів предметної області та їх властивостей (атрибутів) (табл. 1.3).

Таблиця 1.3: Опис об'єктів предметної області «Облік фотоапаратів на складі магазину»

No	Об'єкт	Опис
1.	Фотоапарат	Характеризується властивостями: назва фотоапарату, назва об'єктиву, мінімальна світлочутливість, максимальна світлочутливість, клас, вага, вартість, дата отримання.
2.	Виробник	Означає фірму-виробника фотоапарату. Характеризується властивостями: назва фірми-виробника, спеціалізація фірми.
3.	Інтерфейс	Означає апаратний інтерфейс, за допомогою якого фотоапарат під'єднується до комп'ютера. Характеризується властивостями: назва інтерфейсу, максимальна швидкість передачі.
4.	Матриця	Означає фоточутливу матрицю (сенсор) фотоапарата. Характеризується властивостями: тип матриці, розміри матриці, кількість мегапікселів.

Сформовано список функцій, які повинна виконувати БД:

1. Додавати, видаляти та модифікувати інформацію про моделі фотоапаратів.

2. Виконувати пошук фотоапаратів за вказаними параметрами.

Сформовано список вимог до БД:

1. БД повинна міститися інформацію про основні параметри фотоапаратів.

2. У БД інформація не повинна дублюватися.

3. При введенні інформації про модель фотоапарату потрібно використовувати відому інформацію про виробників, інтерфейси та матриці фотоапаратів.

4. При введенні даних у таблиці БД по можливості застосовувати не введення з клавіатури, а вибір зі списків.

2. Етап «Проектування БД» (рис. 1.9) передбачає концептуальне, логічне та фізичне проектування. Проектування виконується на основі даних, зібраних на попередньому етапі, які подаються у вигляді опису об'єктів предметної області, документації підприємства, переліку функцій БД та списку вимог до БД.

2.1. Концептуальне проектування полягає в синтезі узагальненої структури (моделі) БД, яка не залежить від конкретних аспектів її реалізації (обраної обчислювальної платформи, типу моделі БД, СКБД, мови програмування та ін.). У даному випадку поняття «концепція» розуміється як система початкових положень, які у подальшому уточнюються та конкретизуються. З метою уникнення можливих конфліктів концептуальна модель БД повинна враховувати основні побажання всіх користувачів. У процесі моделювання визначаються найважливіші об'єкти предметної області та їх параметри (характеристики). Об'єкти, які мають однаковий набір параметрів, об'єднуються у множини. У відповідність таким множинам об'єктів ставляться сутності БД, оскільки у даному випадку сутність – множина однотипних об'єктів. Для кожної сутності визначаються операції, які виконує сутність, та операції, які виконуються над сутністю.

Таким чином, у результаті концептуального проектування виділяються сутності предметної області та встановлюються зв'язки між ними («один до одного», «один до багатьох», «багато до одного», «багато до багатьох»). При цьому важливо не тільки формально описати сутності та зв'язки, але й розкрити їх семантику (пояснити значення).

Для побудови концептуальної моделі застосовується функціональний або предметний підходи. Функціональний підхід застосовує рух «від задач», коли заздалегідь відомі функції майбутніх користувачів БД, а також відомі всі задачі, для інформаційних потреб яких створюється БД. Тому при функціональному підході сутностями є об'єкти, які фігурують в таких задачах. Предметний підхід застосовується тоді, коли інформаційні потреби майбутніх користувачів чітко не визначені. Тоді як сутності вибираються об'єкти предметної області, що вважаються суттєвими. В практичній діяльності застосовується комплексний підхід, який враховує як відомі функціональні задачі, так і об'єкти предметної області.

Розроблена концептуальна модель документується, що дозволяє ефективно використовувати її на наступних етапах. Отримані сутності та їх взаємозв'язки можуть подаватися у табличній або графічній формах. Поширеним способом графічного подання моделей БД є ER (Entity-relationship) діаграми (діаграми сутність-зв'язок).

При розробці концептуальна модель даних постійно піддається тестуванню і перевірці на відповідність вимогам користувачів; за потреби сутності змінюються, додаються або видаляються, а відповідно до цього змінюються їх зв'язки.

Етап концептуального проектування моделі БД проілюструємо таким прикладом.

Приклад 1.2. Етап «Концептуальне проектування».

Модель БД для предметної області «Облік фотоапаратів на складі магазину» назвемо «db_Photocamera» (скорочення «db» означає «database» – база даних).

На основі даних, зібраних на попередньому етапі (див. приклад 1.1), вибрано такі сутності предметної області (табл. 1.4). Кожній сутності відповідає таблиця БД, яка буде реалізована на наступних етапах. Назви таблиць БД починаються з скорочення «t», яке означає «table» – таблиця.

Таблиця 1.4: Опис сутностей предметної області «Облік фотоапаратів на складі магазину»

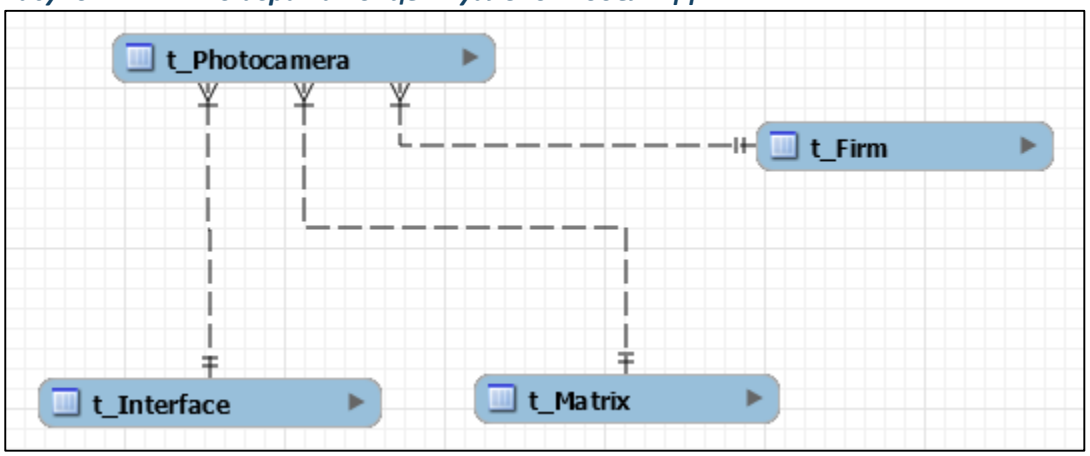
No	Сутність	Таблиця БД, яка відповідає сутності	Операції, які виконуються над сутністю
1.	Фотоапарат	t_Photocamera	Додавання, видалення, модифікація та пошук даних
2.	Виробник	t_Firm	-
3.	Інтерфейс	t_Interface	-
4.	Матриця	t_Matrix	-

Отриману концептуальну модель БД зобразимо у вигляді ER-діаграми (діаграми сутність-зв'язок) (рис. 1.11).

Між сутностями (а відповідно – таблицями) встановлено зв'язки «один до багатьох» у нотації «вороняча лапка» (Crow's Foot). Це означає, що одному запису таблиці «t_Firm» відповідає кілька записів таблиці «t_Photocamera». Проте, можливий випадок, коли певному запису таблиці «t_Firm» не відповідає жоден запис таблиці «t_Photocamera». Аналогічно виконується з'єднання таблиці «t_Interface» з «t_Photocamera» та таблиці «t_Matrix» з «t_Photocamera».

У даний час практично кожен виробник СКБД пропонує програмний продукт для для автоматизованого проектування БД, тобто засоби CASE (Computer-Aided Software Engineering – комп'ютерні засоби для проектування програмного забезпечення). Наприклад, це MySQL Workbench (СКБД MySQL), OracleDesigner (СКБД Oracle), PowerDesinger (Sybase) та ін.

Рисунок 1.11: ER-діаграма концептуальної моделі БД



Крім цього, існують програми для проектування БД третіх фірм, які не виробляють БД (наприклад, ERwin Data Modeler фірми Logic Works). У даній роботі проектування БД виконано засобами MySQL Workbench [9] (рис. 1.11), проте таке проектування можна виконувати іншими CASE-засобами.

2.2. Логічне проектування полягає в розробці логічної моделі БД на основі попередньо створеної концептуальної моделі. Логічна модель залежить від обраного типу моделі БД (наприклад, реляційної або об'єктно-орієнтованої), але не залежить від конкретної СКБД. Логічна модель БД містить наступні компоненти:

- назви сутностей (таблиць);
- назви атрибутів (полів);
- типи зв'язків між сутностями;
- первинні ключі;
- зовнішні ключі.

Розглянемо принципи побудови логічної моделі для реляційних БД, які на даний час є досить поширеними. Назви сутностей отримуються з концептуальної моделі, при цьому на етапі логічного проектування кожній сутності відповідає таблиця БД. Як атрибути сутностей вибираються параметри об'єктів, які є важливими для предметної області. У подальшому атрибути сутностей використовуються як поля (стовпці) таблиць.

Між отриманими сутностями встановлюються зв'язки відповідних типів: «один до одного», «один до багатьох», «багато до одного», «багато до багатьох». У випадку зв'язку «багато до багатьох» його потрібно замінити двома зв'язками типу «один до багатьох» шляхом створення проміжної сутності.

У кожної сутності повинен бути атрибут первинного ключа, за допомогою якого можна ідентифікувати (безпомилково розрізнити) записи в таблиці. Первинний ключ звичайно позначають англійською абревіатурою «id» (identifier – ідентифікатор) або PK (Primary Key). Зовнішні ключі FK (Foreign Key) у поєднанні з первинними ключами застосовуються для встановлення зв'язків між таблицями.

У процесі розробки логічна модель даних тестується і перевіряється на відповідність попередньо сформованим вимогам до БД. Для перевірки коректності логічної моделі даних використовується метод нормалізації. Завдяки нормалізації усувається дублювання даних і некоректні зв'язки між ними, створюються ширші можливості для маніпулювання даними. Наприклад, якщо в сутності є атрибут, який може існувати в предметній області у вигляді самостійного об'єкту, а кількість його допустимих значень є незначною, то такий атрибут доцільно виділити як окрему сутність (з'єднавши зв'язками з іншими сутностями).

Етап логічного проектування моделі БД проілюструємо таким прикладом.

Приклад 1.3. Етап «Логічне проектування».

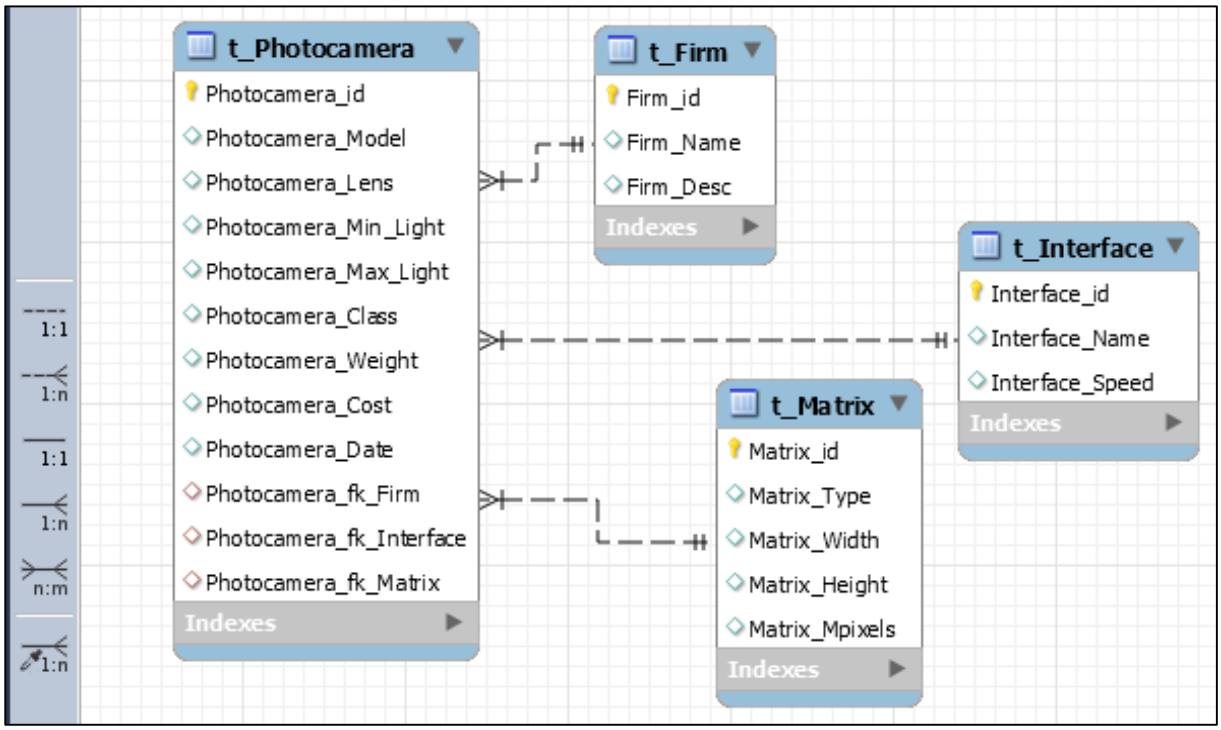
На основі розробленої концептуальної моделі (див. приклад 1.2) визначаються назви сутностей (таблиць), атрибутів (полів), первинні та зовнішні ключі (табл. 1.5). З врахуванням ER-діаграми концептуальної моделі БД (рис. 1.11) побудовано ER-діаграми логічної моделі (рис. 1.12). При цьому ER-діаграми на рис. 1.12а та на рис. 1.12б описують одну і ту ж логічну модель БД (з неідентифікуючими зв'язками), але з різними способами візуалізації (нотації). Зокрема, у нотації «вороняча лапка» (Crow's Foot) (рис. 1.12а) вказується, які таблиці з'єднуються між собою та типи зв'язків. У нотації «з'єднання стовпців» (Connect to Columns) (рис. 1.12б) вказується, які саме поля таблиць зв'язані (у такому випадку зв'язок «один до багатьох» позначається «1-∞», де знак безмежності «∞» означає кілька записів таблиці). Якщо зв'язки між таблицями позначені як ідентифікуючі (рис. 1.12в), то в такому випадку кожному запису дочірньої таблиці «t_Photocamera» повинен відповідати один запис батьківської таблиці «t_Firm», один запис батьківської таблиці «t_Interface» та один запис батьківської таблиці «t_Matrix».

Нотація «з'єднання стовпців» (Connect to Columns) є більш інформативною (порівняно з Crow's Foot), проте вибір конкретної нотації залежить від розробника. Вибір ідентифікуючих або неідентифікуючих з'єднань між таблицями залежить від особливостей предметної області.

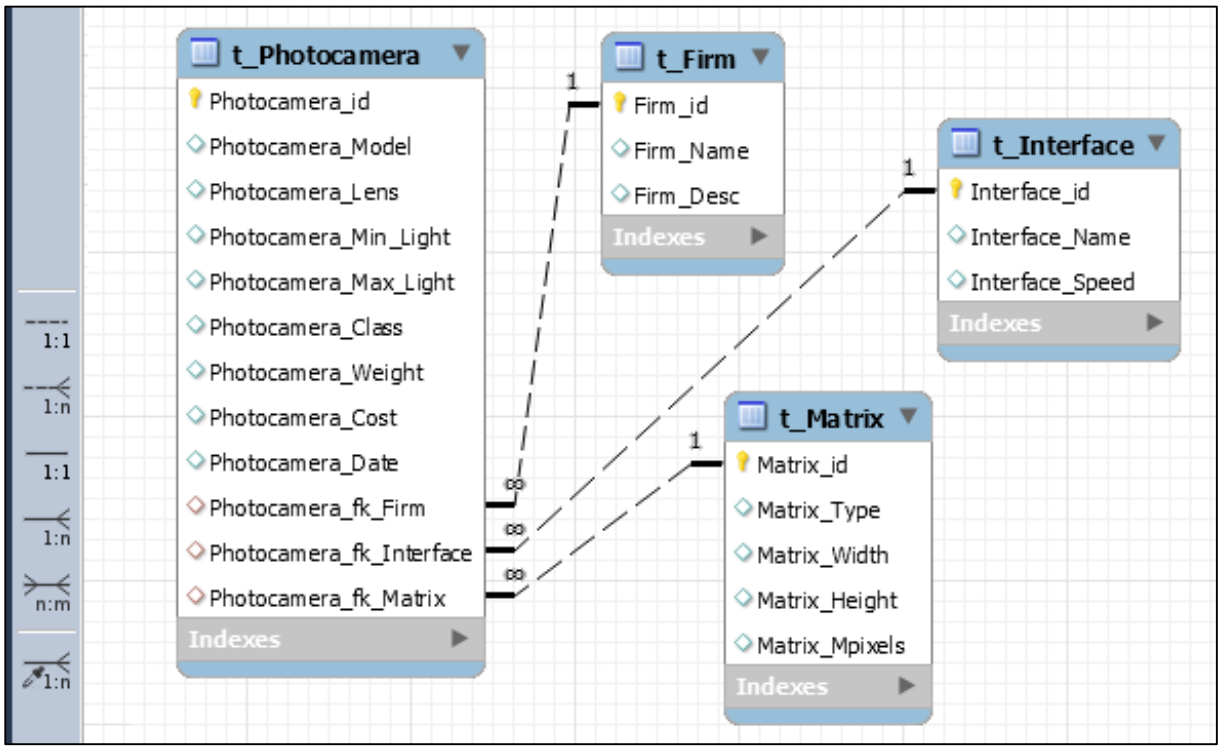
Таблиця 1.5: Опис сутностей та їх атрибутів для предметної області «Облік фотоапаратів на складі магазину» (№С – номер сутності, №А – номер атрибуту)

№С	Сутність	Таблиця	№А	Атрибут (поле)	Опис атрибуту
1.	Фотоапарат	t_Photocamera	1	Photocamera_id	Первинний ключ
			2	Photocamera_ Model	Назва фотоапарату
			3	Photocamera_Lens	Назва об’єктиву
			4	Photocamera_Min_Light	Мінімальна світлочутливість
			5	Photocamera_Max_Light	Максимальна світлочутливість
			6	Photocamera_Class	Клас
			7	Photocamera_Weight	Вага, г
			8	Photocamera_Cost	Вартість, грн
			9	Photocamera_Date	Дата отримання
			10	Photocamera_fk_Firm	Зовнішній ключ для зв’язку з таблицею «t_Firm»
			11	Photocamera_fk_Interface	Зовнішній ключ для зв’язку з таблицею «t_Interface»
			12	Photocamera_fk_Matrix	Зовнішній ключ для зв’язку з таблицею «t_Matrix»
2.	Виробник	t_Firm	1	Firm_id	Первинний ключ
			2	Firm_Name	Назва фірми
			3	Firm_Desc	Спеціалізація фірми
3.	Інтерфейс	t_Interface	1	Interface_id	Первинний ключ
			2	Interface_Name	Назва інтерфейсу
			3	Interface_Speed	Максимальна швидкість передачі, Мбіт/с
4.	Матриця	t_Matrix	1	Matrix_id	Первинний ключ
			2	Matrix_Type	Тип фоточутливої матриці
			3	Matrix_Width	Ширина матриці, мм
			4	Matrix_Height	Висота матриці, мм
			5	Matrix_Mpixels	Кількість мегапікселів

Рисунок 1.12: ER-діаграми логічної моделі БД (побудовані засобами MySQL Workbench): а) неідентифікуючі зв'язки (позначаються пунктирної лінією) між таблицями у нотації «вороняча лапка» (Crow's Foot); б) неідентифікуючі зв'язки між таблицями у нотації «з'єднання стовпців» (Connect to Columns);

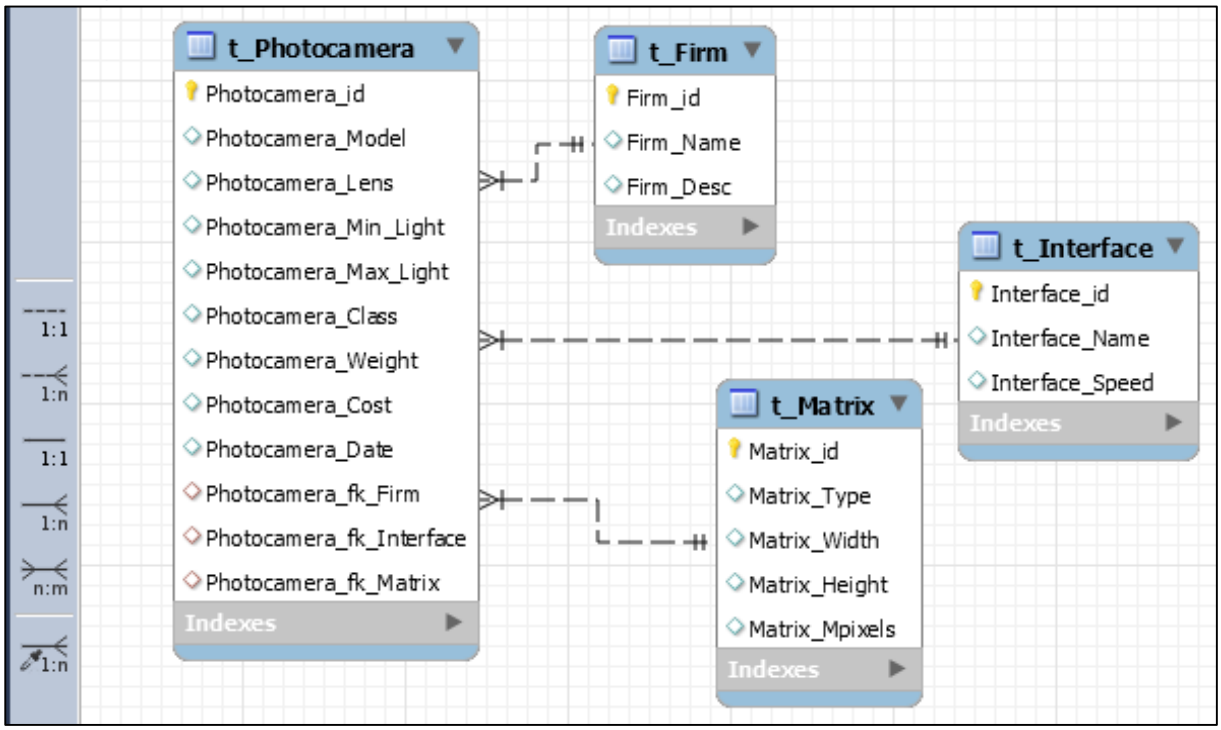


а)



б)

Продовження рисунку 1.12: ER-діаграми логічної моделі БД: в) ідентифікуючі зв'язки (позначаються суцільною лінією) між таблицями у нотації «з'єднання стовпців» (Connect to Columns)



в)

2.3. Фізичне проектування починається з вибору конкретної СКБД (наприклад, MySQL), за допомогою якої планується реалізувати БД. Основною метою фізичного проектування БД є опис способу фізичної реалізації логічної моделі БД. У випадку реляційної моделі даних фізичне проектування полягає у наступному:

1. Таблиці та їх поля (стовпці) отримуються з логічної моделі, при цьому як поля використовуються атрибути відповідних таблиць.

2. З урахуванням вимог СКБД виконується вибір типів полів таблиць.

3. Визначаються обмеження та діапазони значень полів.

4. Вибирається спосіб реалізації первинних та зовнішніх ключів.

Специфіка конкретної СКБД може включати в себе обмеження на іменування об'єктів бази даних, обмеження на підтримувані типи даних та ін. У процесі розробки фізична модель даних тестується і перевіряється на відповідність попередньо сформованим вимогам до БД.

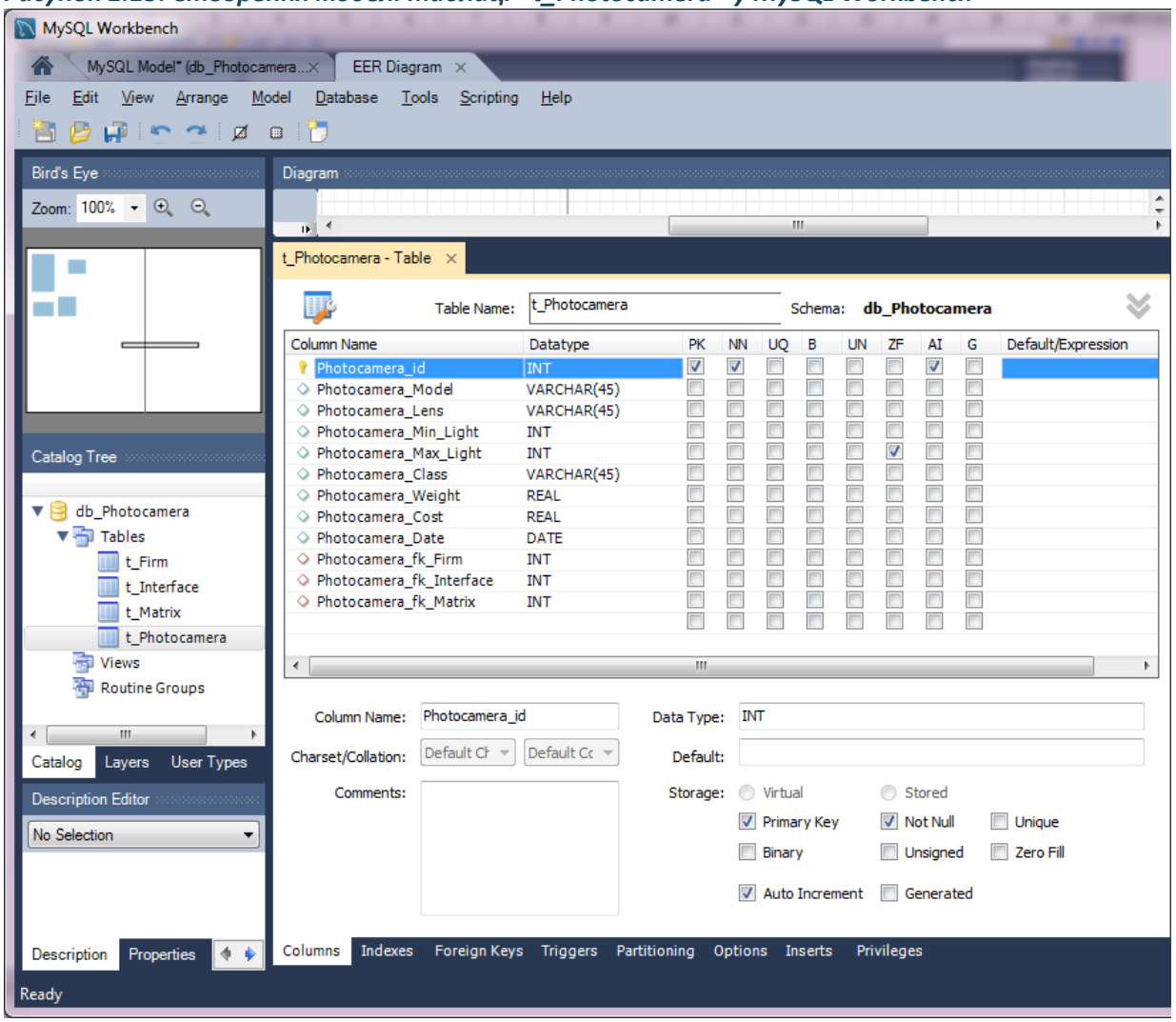
Процес проектування БД є ітераційним, оскільки у разі виявлення помилок у концептуальній, логічній або фізичній моделях такі помилки виправляються шляхом повторного виконання відповідних етапів.

Етап фізичного проектування моделі БД проілюструємо таким прикладом.

Приклад 1.4. Етап «Фізичне проектування».

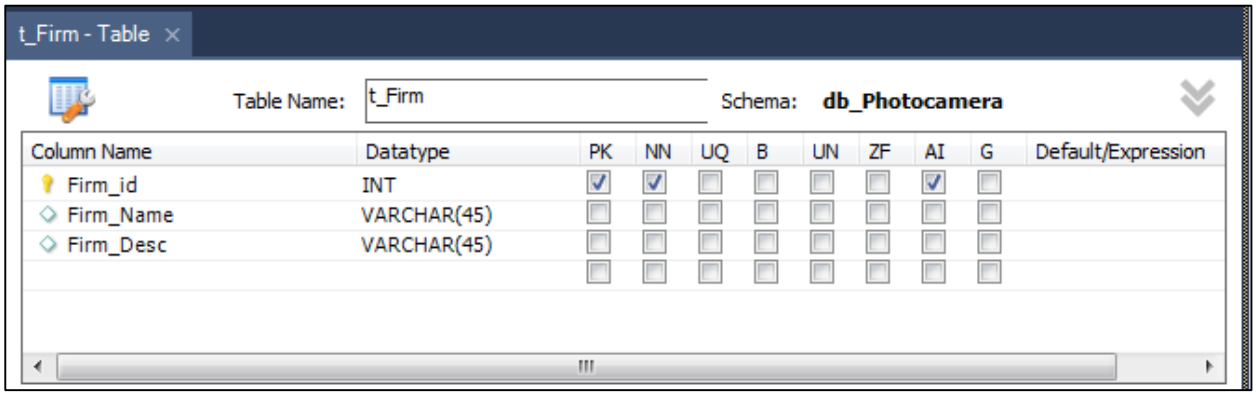
На основі розробленої логічної моделі (див. приклад 1.3) визначаються назви таблиць, їх полів, первинних та зовнішніх ключів. З урахуванням вимог СКБД (MySQL) виконується вибір типів полів таблиць, у середовищі проектування MySQL Workbench створюються моделі таблиць БД (рис. 1.13). Для первинного ключа потрібно вибрати тип даних INT (цілий) і встановити прапорці PK (первинний ключ), NN (не порожнє поле), AI (автоінкрементне). Для текстових полів вибрано тип VARCHAR (з вказанням максимальної кількості символів); для полів, які містять дійсні числа, вибрано тип REAL; для дати вибрано тип DATE. Для зовнішніх ключів вибрано тип даних INT.

Рисунок 1.13: Створення моделі таблиці «t_Photocamera» у MySQL Workbench

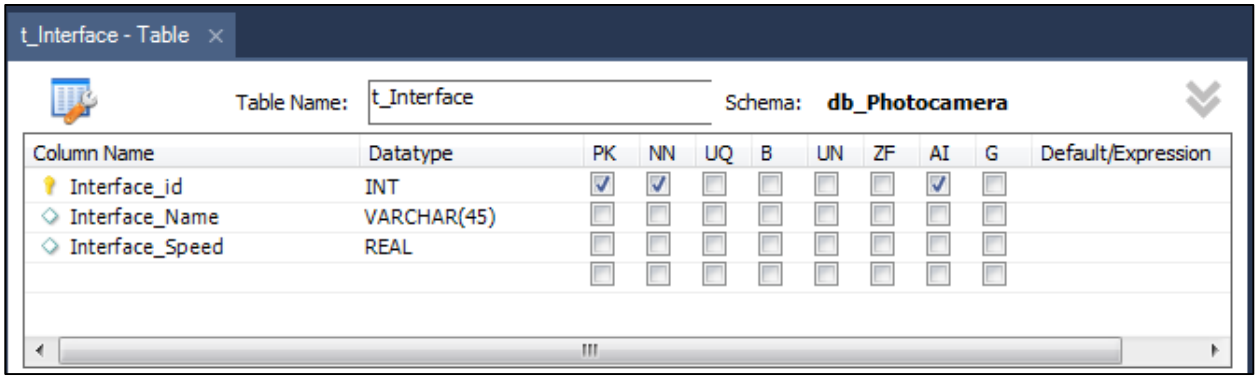


Аналогічно створено моделі таблиць «t_Firm», «t_Interface», «t_Matrix» (рис. 1.14). Створена модель БД зберігається у файл.

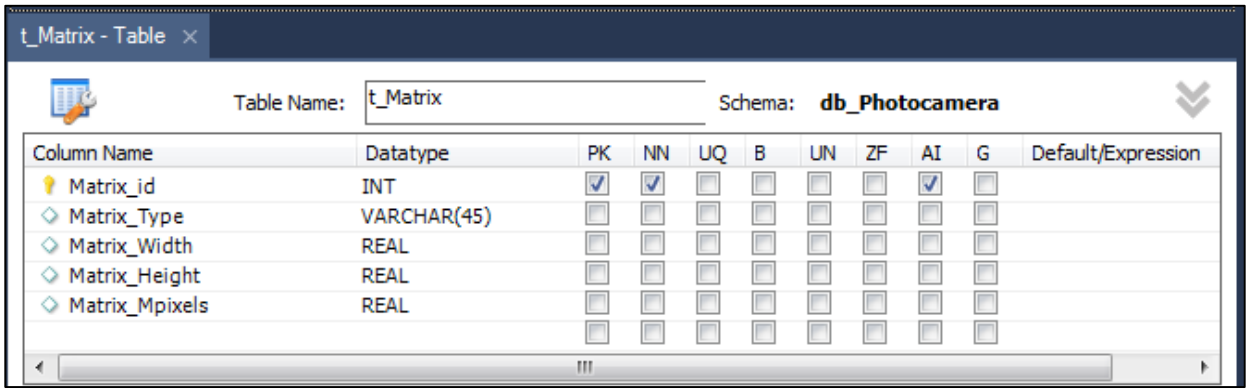
Рисунок 1.14: Створення моделей таблиць «t_Firm» (а), «t_Interface» (б), «t_Matrix» (в) у MySQL Workbench



а)



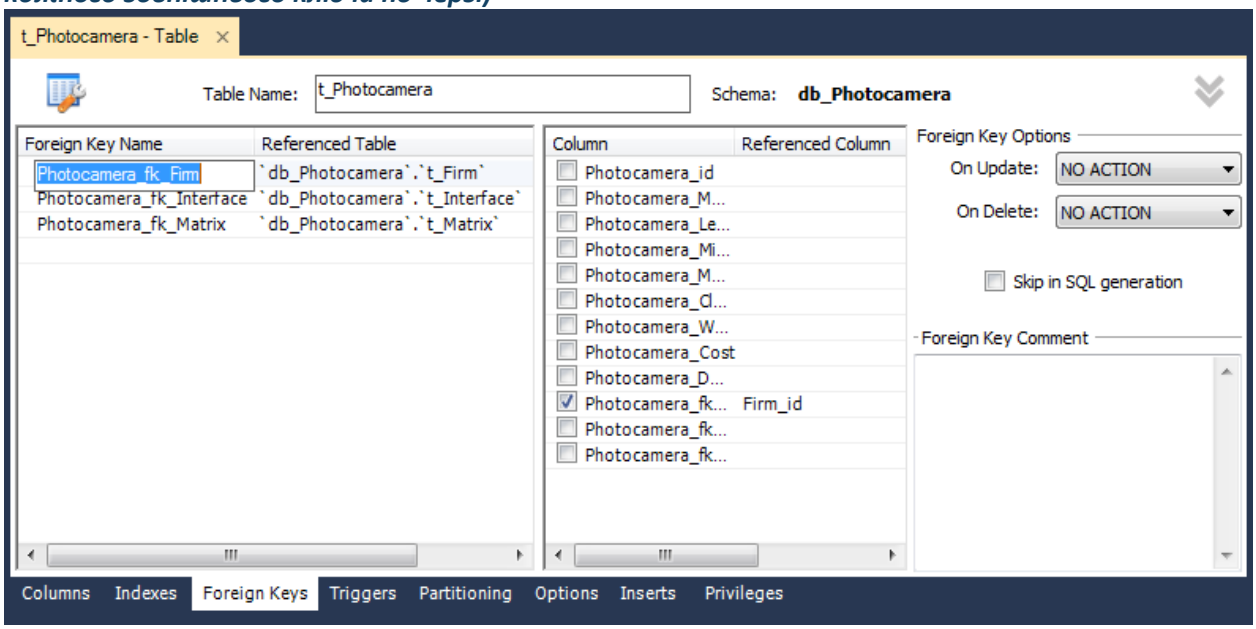
б)



в)

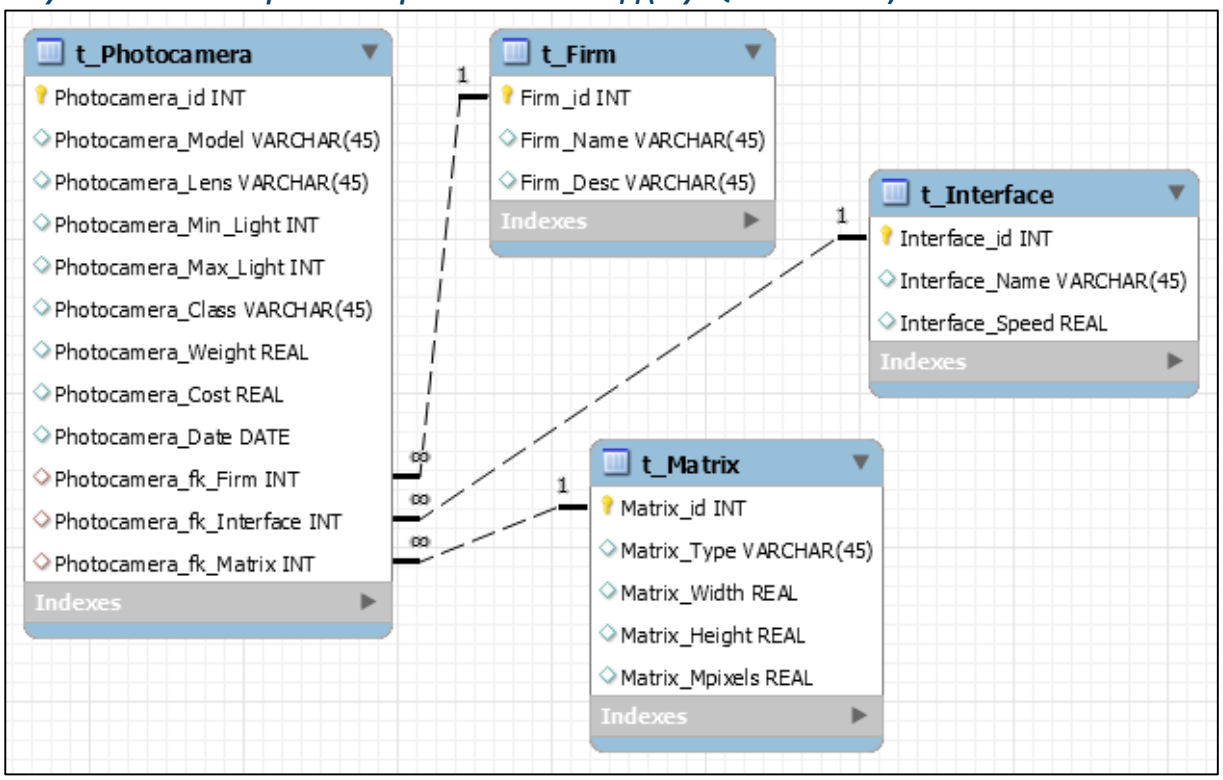
Для таблиці «t_Photocamera» для зовнішніх ключів вказано, з якими таблицями та полями вони пов’язані – на вкладці «Foreign Keys» у нижній частині вікна (рис. 1.15).

Рисунок 1.15: Встановлення параметрів зовнішніх ключів засобами MySQL Workbench (для кожного зовнішнього ключа по черзі)



Структуру створеної фізичної моделі БД показано у вигляді ER (Entity-relationship) діаграми (рис. 1.16). У порівнянні з попередніми (рис. 1.12) ER-діаграма фізичної моделі є найбільш детальною, оскільки на ній вказані типи полів.

Рисунок 1.16: ER діаграма для фізичної моделі БД (MySQL Workbench)



3. Етап створення (реалізація) БД (рис. 1.9) полягає у створенні структури БД на основі фізичної моделі БД засобами СКБД, програмуванні логіки роботи БД (наприклад, створення тригерів і збережених процедур), створенні інтерфейсу користувача БД за допомогою СКБД або середовищ розробки, створенні прикладних програм для роботи з БД, створенні засобів захисту системи (наприклад, паролів).

У створеній БД реалізується певний набір бізнес-правил (правил керування БД), які враховують обмеження БД:

1) допустимий діапазон значень полів;

2) значення полів за замовчуванням;

3) унікальність значень полів (заборону повторів);

4) заборону нульових значень;

5) обмеження цілісності за посиланнями (з урахуванням зв'язків між таблицями).

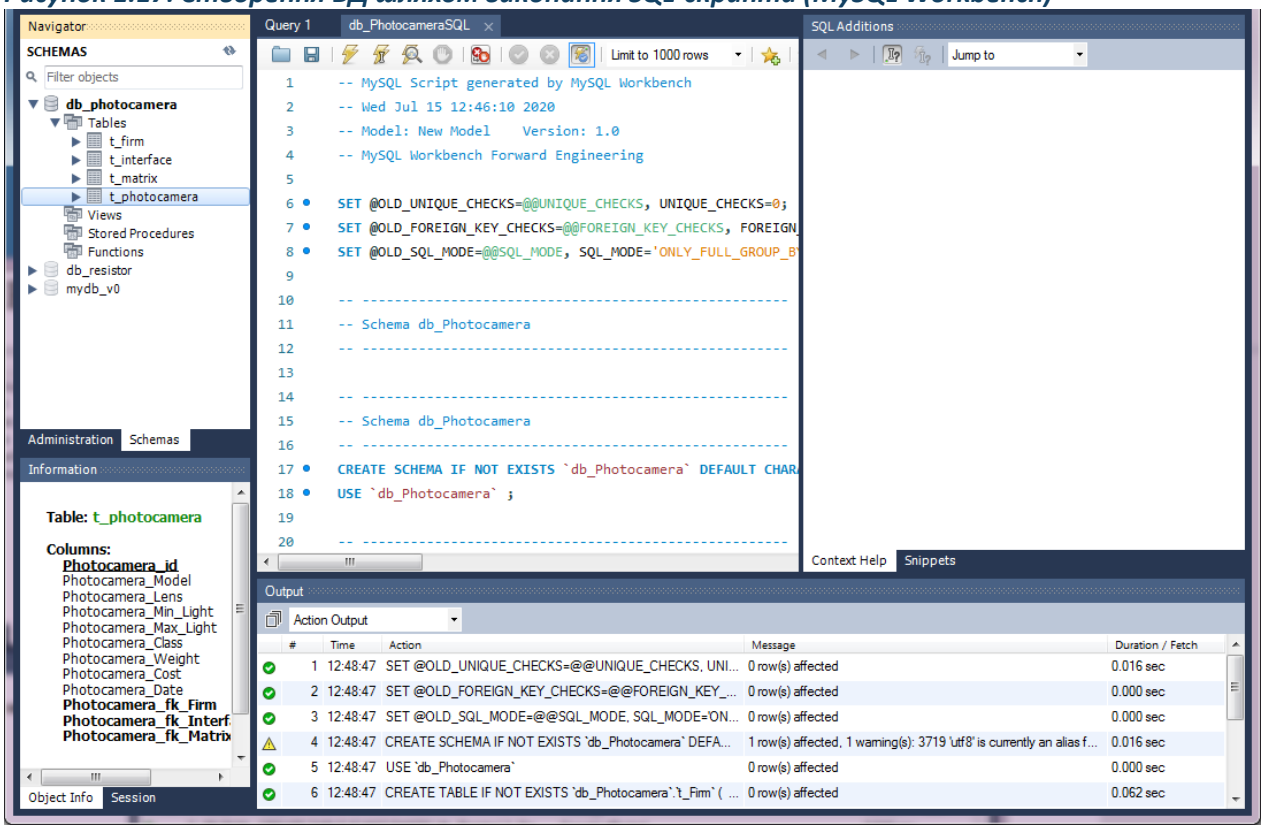
Етап створення БД проілюструємо таким прикладом.

Приклад 1.5. Етап «Етап створення (реалізація) БД».

На основі розробленої фізичної моделі (див. приклад 1.4) створено SQL-скрипт засобами MySQL Workbench і збережено його у файл.

Виконано з'єднання фізичної моделі БД з сервером командою «Database → Manage Connections», при цьому назва з'єднання «db_Photocamera» співпадає з назвою БД. Далі на основі попередньо збереженого SQL-скрипта створено БД (рис. 1.17).

Рисунок 1.17: Створення БД шляхом виконання SQL-скрипта (MySQL Workbench)

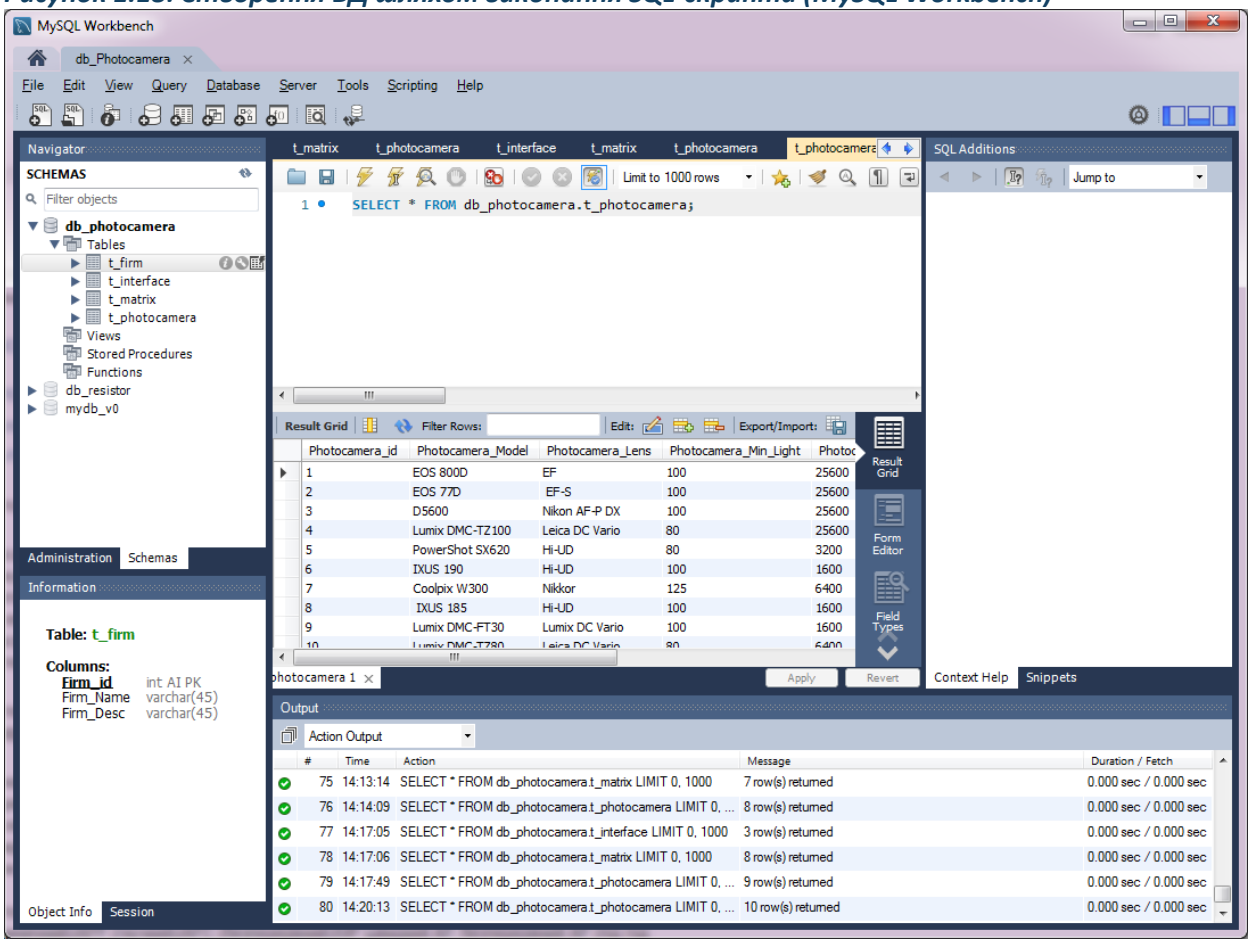


4. Етап введення даних (впровадження БД) (рис. 1.9) полягає у завантаженні (записі) даних у створену БД. Введення даних виконується в ручному режимі через інтерфейс БД або за допомогою спеціальних команд (наприклад, SQL-запитів). Також можливе зчитування даних з файлів певного формату та збереження їх у БД. Якщо існує попередня версія БД, то можливе зчитування, конвертація до потрібного формату та завантаження даних у нову БД. Етап введення даних проілюструємо таким прикладом.

Приклад 1.6. Етап «Етап введення даних (впровадження БД)».

Для додавання записів до таблиць (наповнення таблиць) створеної БД (див. приклад 1.5) потрібно відкрити відповідну таблицю засобами MySQL Workbench в режимі таблиці (рис. 1.18), заповнити всі поля (крім первинного ключа, оскільки значення первинного ключа обчислюються автоматично). При цьому дочірня таблиця «t_Photocamera», яка містить зовнішні ключі, заповнюється останньою, оскільки як значення зовнішнього ключа вказуються значення первинного ключа батьківської таблиці.

Рисунок 1.18: Створення БД шляхом виконання SQL-скрипта (MySQL Workbench)



5. Етап підтримки даних (супроводу БД) (рис. 1.9) полягає в тому, що у разі потреби у функціонуючу БД можуть вноситися зміни, що відповідають новим вимогам. Реалізація цих змін проводиться за допомогою повторного виконання деяких з перерахованих вище етапів життєвого циклу. Наприклад, для певної таблиці може бути додано нове поле.

Крім цього, періодично (або за вимогою) створюються резервні копії БД, з яких БД може бути відновлена (у випадку збою діючої БД). Виконується виявлення та усунення помилок у БД. Виконується контроль (моніторинг) продуктивності системи. Якщо продуктивність падає нижче прийнятного рівня, то може знадобитися додаткове налаштування чи реорганізація бази даних.

Сучасні СКБД надають різні утиліти для адміністрування БД, включаючи утиліти завантаження даних і контролю за функціонуванням системи. Подібні утиліти здатні відслідковувати роботу БД і надавати інформацію про її показники: рівень використання БД, ефективність системи блокувань (включаючи відомості про кількість взаємних блокувань) та ін. Адміністратор БД може використовувати цю інформацію для налаштування системи з метою підвищення її продуктивності (наприклад, за рахунок створення додаткових індексів), прискорення виконання запитів, об'єднання або розділення таблиць.

Підтримка даних (супровід БД) передбачає наступне:

• Забезпечення колективного доступу користувачів до БД з наданням паролів та відповідних прав доступу.

• Призначення прав доступу для нових користувачів.

• Зміна (модернізація) БД.

• Профілактичне обслуговування (наприклад, резервне копіювання).

• Відновлення БД з резервних копій.

• Ведення статистики доступу до БД для підвищення ефективності роботи системи.

• Періодична перевірка безпеки БД.

6. Етап експлуатації БД (пошуку інформації) (рис. 1.9) полягає у введенні нової інформації у БД, а також пошуку інформації в БД за вказаними критеріями (певними значеннями полів). Для пошуку інформації в БД часто застосовуються SQL-запити [13, 14].

Етап експлуатації БД та пошуку інформації проілюструємо таким прикладом.

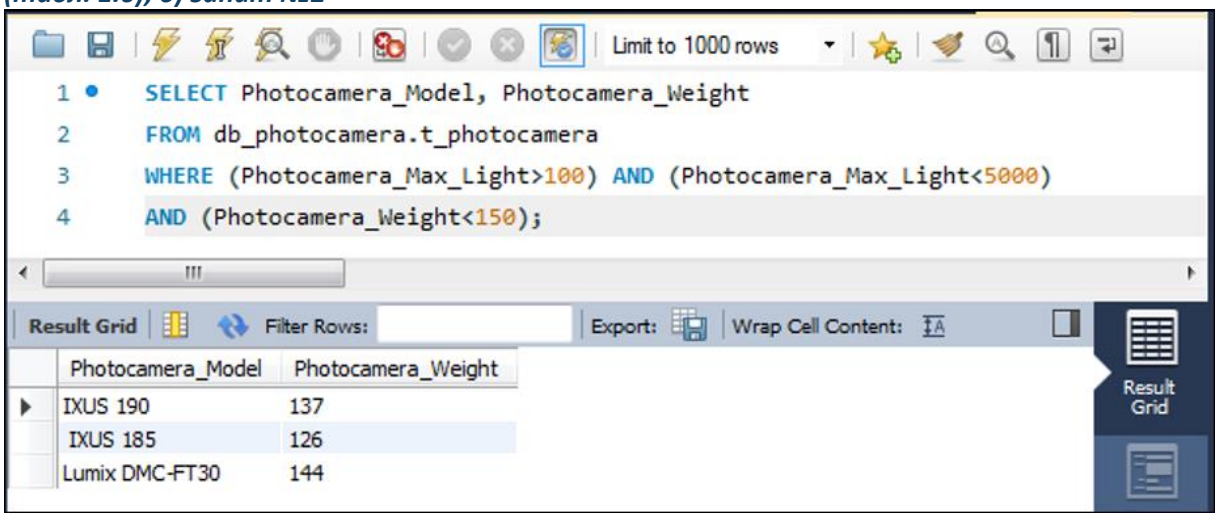
Приклад 1.7. Етап «Етап експлуатація БД (пошук інформації)».

Залежно від завдання пошуку інформації (табл. 1.6) формується відповідний код SQL запиту. Після виконання такого SQL запиту користувачу надається відібрана інформація (рис. 1.19). Використання SQL запитів для пошуку інформації значно підвищує ефективність роботи користувача (наприклад, з існуючих 10000 записів таблиці користувачу автоматично повертаються тільки 3 потрібних).

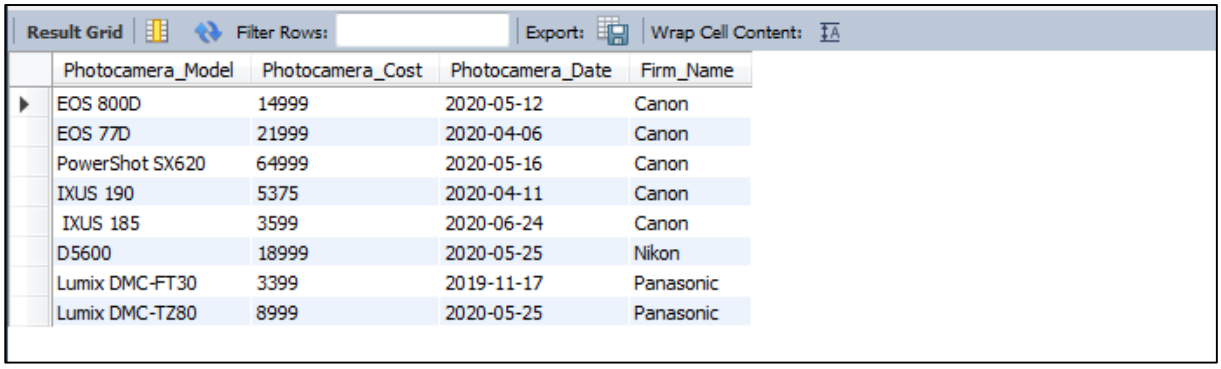
Таблиця 1.6: Завдання пошуку інформації та код SQL-запитів

No	Завдання пошуку інформації	Код запиту
1.	Вивести назви і вагу моделей із заданими максимальною світлочутливістю та вагою.	SELECT Photocamera_Model, Photocamera_Weight FROM db_photocamera.t_photocamera WHERE (Photocamera_Max_Light>100) AND (Photocamera_Max_Light<5000) AND (Photocamera_Weight<150);
2.	Вивести назви, вартості та дати отримання моделей, назви фірм із заданою датою отримання.	SELECT t_Photocamera.Photocamera_Model, t_Photocamera.Photocamera_Cost, t_Photocamera.Photocamera_Date, t_Firm.Firm_Name FROM db_photocamera.t_photocamera, db_photocamera.t_Firm WHERE (Photocamera_fk_Firm= Firm_id) AND (t_Photocamera.Photocamera_Date between '2019-05-19' and '2020-07-20');

Рисунок 1.19: SQL-запити до створеної БД (рис. 1.18) та результати їх виконання: а) запит №1 (табл. 1.6); б) запит №2



а)



Photocamera_Model	Photocamera_Cost	Photocamera_Date	Firm_Name
EOS 800D	14999	2020-05-12	Canon
EOS 77D	21999	2020-04-06	Canon
PowerShot SX620	64999	2020-05-16	Canon
IXUS 190	5375	2020-04-11	Canon
IXUS 185	3599	2020-06-24	Canon
D5600	18999	2020-05-25	Nikon
Lumix DMC-FT30	3399	2019-11-17	Panasonic
Lumix DMC-TZ80	8999	2020-05-25	Panasonic

б)

На всіх етапах життєвого циклу БД складається з двох компонентів: структури та даних. При цьому використовують 3 рівні опису структури БД:

1. Інфологічний (БД є сукупністю сутностей та зв'язків між ними), такому опису відповідають етапи 1 та 2.1 життєвого циклу (рис. 1.9).

2. Датологічний (БД описується моделлю певного типу: ієрархічною, мережевою, реляційною, об'єктно-орієнтованою), такому опису відповідає етап 2.2 життєвого циклу (рис. 1.9).

3. Фізичний (БД є сукупністю файлів даних і допоміжних файлів), такому опису відповідають етапи 2.3-6 життєвого циклу (рис. 1.9).

1.1.3 Загальні бізнес-програми з використанням баз даних: динамічні веб-сайти, системи управління відносинами з клієнтами, системи планування ресурсів підприємства, системи управління вмістом веб-сайтів

Text.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дейт К.Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. / К.Дж. Дейт. – М.: Вильяме, 2005. – 1328 с.

2. Фаронов В.В. Программирование баз данных в Delphi 7. Учебный курс / В.В. Фаронов. – СПб.: Питер, 2006. – 459 с.

3. ISO/IEC 2382:2015. Information technology – Vocabulary. [Electronic resource]. – Access mode: *https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html*.

4. Хомоненко А.Д. Базы данных: учебник для высших учебных заведений / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – СПб.: Корона, 2009. – 736 с.

5. Аткинсон Л. MySQL. Библиотека профессионала: пер. с англ. / Л. Аткинсон. – М.: Вильямс, 2002. – 624 с.

6. Пасічник В.В. Організація баз даних та знань / В.В. Пасічник, В.А. Резніченко. – К.: BHV, 2006. – 384 с.

7. Берко А.Ю. Системи баз даних та знань : підруч. для студентів ВНЗ. [Кн. 1. Організація баз даних та знань] / А. Ю. Берко, О. М. Верес, В. В. Пасічник; за заг. ред. В. В. Пасічника. – Львів : Магнолія 2006, 2016. – 438 с.

8. Гайна Г.А. Основи проектування баз даних : навч. посіб. для студ. вищ. навч. закл. / Г. А. Гайна. – К. : Кондор, 2008. – 199 с.

9. Elmasri R. Fundamentals of Database Systems / R. Elmasri, S.B.Navathe. – Hoboken, USA: Pearson, 2016. – 1273 p.

10. Perkins L. Seven Databases in Seven Weeks. A Guide to Modern Databases and the NoSQL Movement / L. Perkins, E. Redmond, J.R. Wilson. – Raleigh, North Carolina, USA: The Pragmatic Bookshelf, 2018. – 354 p.

11. Роб П. Системы баз данных: проектирование, реализация и управление: пер. с англ. / П. Роб, К. Коронел. – СПб.: БХВ-Петербург, 2004. – 1040 с.

12. Швецов В.И. Базы данных. Учебное пособие / В.И. Швецов, А.Н. Визгунов, И.Б. Мееров. – Нижний Новгород: Изд-во ННГУ, 2004. – 217 с.

13. MySQL. [Electronic resource]. – Access mode : *https://www.mysql.com*.

14. Microsoft SQL documentation. [Electronic resource]. – Access mode : *https://docs.microsoft.com/uk-ua/sql/t-sql/functions/functions?view=sql-server-ver15*.

Annexes

Annex 1. Name