Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №2

Технології розроблення програмного забезпечення

Тема проєкту ‘Shell (total commander)’

Виконала: студентка групи ІА-33

Маляревич Анжела Валентинівна

Перевірив:

Мягкий Михайло Юрійович

Київ - 2025

Зміст

Тема: Основи проектування

Мета: Обрати зручну систему побудови UML-діаграм та навчитися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.

Тема роботи:

18. Shell (total commander) (state, prototype, factory method, template method, interpreter, client-server) Оболонка повинна вміти виконувати основні дії в системі – перегляд файлів папок в файлової системі, перемикання між дисками, копіювання, видалення, переміщення об'єктів, пошук.

Теоретичні відомості

UML та його діаграми

UML (Unified Modeling Language) - уніфікована мова візуального моделювання, що використовується для аналізу, проєктування та документування програмних систем. UML дозволяє описувати систему на різних рівнях: від концептуального до фізичного.

Основні діаграми UML

- Діаграма варіантів використання (Use Case Diagram) - показує вимоги до системи та взаємодію користувачів із нею.
- Діаграма класів (Class Diagram) - описує статичну структуру системи: класи, їх атрибути, методи та зв’язки.

Діаграма варіантів використання

Діаграма use case відображає функціональність системи з точки зору користувача.

Основні елементи:

- Актори (Actor) - користувачі або зовнішні системи.
- Варіанти використання (Use Case) - дії або послуги, які система надає актору (наприклад: вхід, перегляд даних, створення транзакції).

Типи відносин:

- Асоціація - прямий зв’язок актора з варіантом використання.
- Include - один сценарій завжди включає інший (обов’язковий).
- Extend - сценарій може бути розширений додатковим (необов’язковим).
- Узагальнення - спадкування ролей або функціоналу.

Для уточнення роботи системи складаються сценарії використання (use case scenarios), які описують:

- передумови та постумови;
- учасників;
- короткий опис;
- основний перебіг подій;
- винятки.

Діаграма класів

Діаграма класів показує структуру системи: класи, їх атрибути, методи та зв’язки між ними.

Клас містить:

- назву;
- атрибути (дані);
- методи (операції).

Види зв’язків:

- Асоціація - загальний зв’язок між класами.
- Узагальнення (успадкування) - зв’язок між батьківським і дочірнім класом.
- Агрегація - відношення «ціле-частина», де частини можуть існувати окремо.
- Композиція - сильне відношення «ціле-частина», де частини не існують без цілого.

Логічна структура бази даних

Проєктування бази даних часто виконується на основі діаграми класів. Виділяють:

- Фізичну модель - організація файлів і способів зберігання.
- Логічну модель - таблиці, атрибути, ключі, зв’язки.

Щоб уникнути надмірності даних застосовують нормалізацію:

- 1НФ - кожен атрибут має лише одне атомарне значення.
- 2НФ - усі неключові атрибути залежать від усього первинного ключа.
- 3НФ - немає транзитивних залежностей (атрибутів, що залежать від інших неключових атрибутів).
- НФ Бойса-Кодда (BCNF) - посилена форма 3НФ, кожна залежність визначається ключем.

Хід роботи

1. Розробка діаграми варіантів використання (Use Case Diagram)

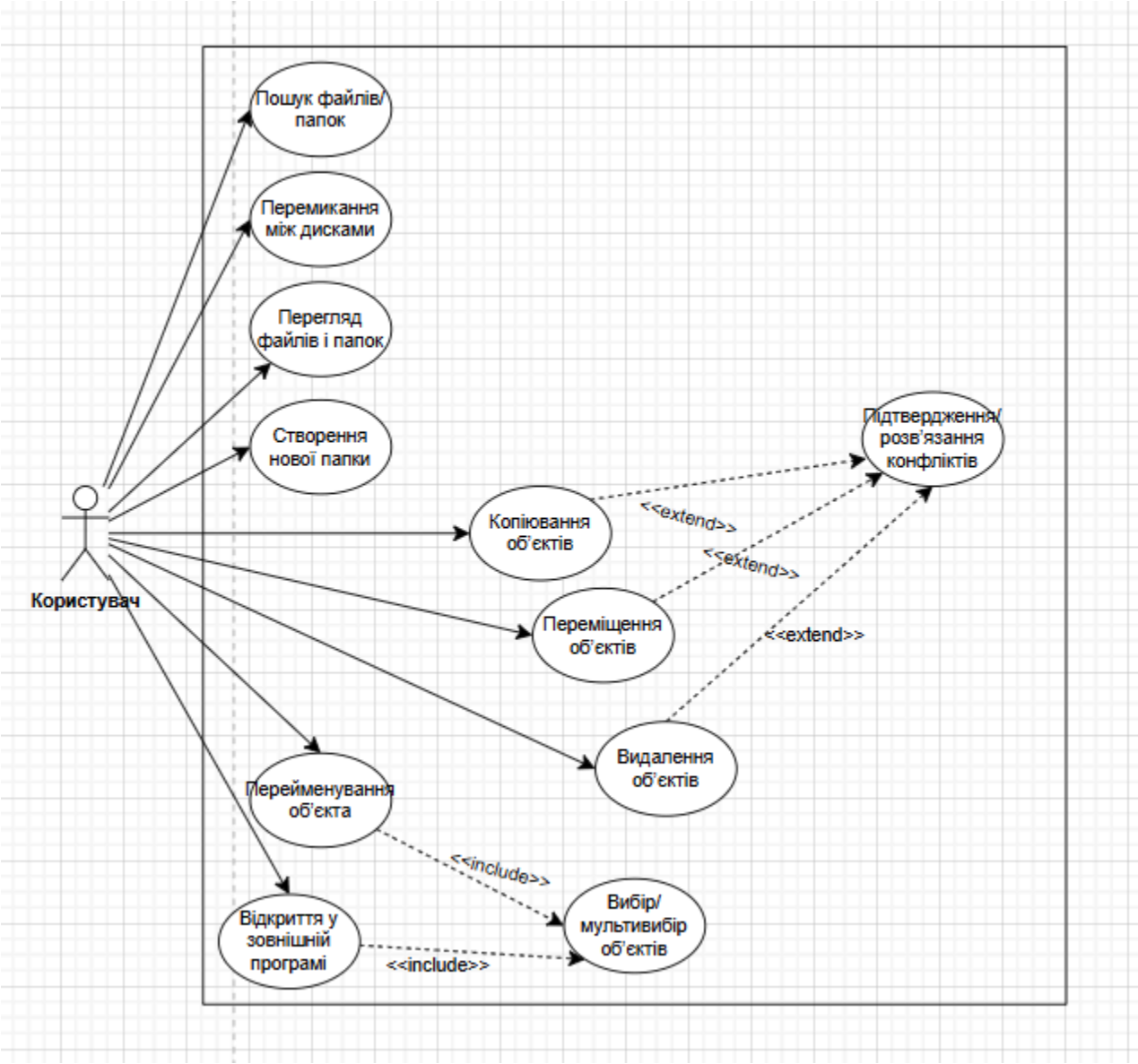

Рисунок 1 – Use case діаграма для звичайного користувача

2. Розробка діаграми класів предметної області

Діаграма:

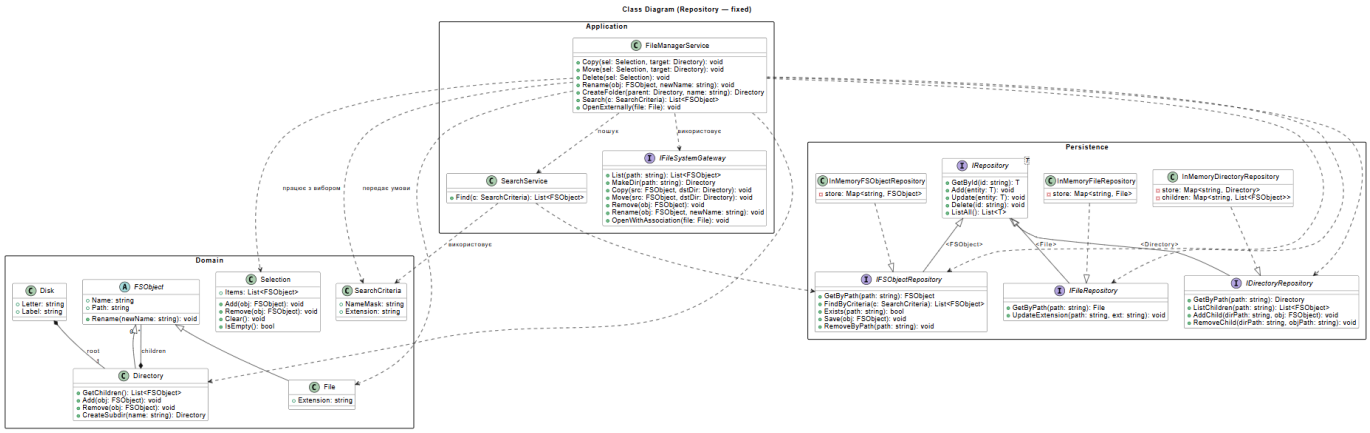

Рисунок 2 –Діаграма класів

Код:

@startuml

title Class Diagram (Repository — fixed)

skinparam classBackgroundColor #FFFFFF

skinparam classBorderColor #333333

skinparam packageStyle rectangle

package "Domain" {

abstract class FSObject {

+Name: string

+Path: string

+Rename(newName: string): void

}

class File {

+Extension: string

}

class Directory {

+GetChildren(): List<FSObject>

+Add(obj: FSObject): void

+Remove(obj: FSObject): void

+CreateSubdir(name: string): Directory

}

class Disk {

+Letter: string

+Label: string

}

class Selection {

+Items: List<FSObject>

+Add(obj: FSObject): void

+Remove(obj: FSObject): void

+Clear(): void

+IsEmpty(): bool

}

class SearchCriteria {

+NameMask: string

+Extension: string

}

FSObject <|-- File

```
  FSObject <|-- Directory


  Disk *-- "1" Directory : root
  Directory *-- "0..*" FSObject : children
}


package "Application" {
  class FileManagerService {
    +Copy(sel: Selection, target: Directory): void
    +Move(sel: Selection, target: Directory): void
    +Delete(sel: Selection): void
    +Rename(obj: FSObject, newName: string): void
    +CreateFolder(parent: Directory, name: string): Directory
    +Search(c: SearchCriteria): List<FSObject>
    +OpenExternally(file: File): void
  }

  interface IFileSystemGateway {
```

+List(path: string): List<FSObject>

+MakeDir(path: string): Directory

+Copy(src: FSObject, dstDir: Directory): void

+Move(src: FSObject, dstDir: Directory): void

+Remove(obj: FSObject): void

+Rename(obj: FSObject, newName: string): void

+OpenWithAssociation(file: File): void

}

class SearchService {

+Find(c: SearchCriteria): List<FSObject>

}

FileManagerService ..> IFileSystemGateway : використовує

FileManagerService ..> SearchService : пошук

FileManagerService ..> Selection : працює з вибором

FileManagerService ..> Directory

FileManagerService ..> File

```
  SearchService ..> SearchCriteria : використовує
  FileManagerService ..> SearchCriteria : передає умови
}

package "Persistence" as Persistence {
  interface IRepository<T> {
    +GetById(id: string): T
    +Add(entity: T): void
    +Update(entity: T): void
    +Delete(id: string): void
    +ListAll(): List<T>
  }

  interface IFSObjectRepository {
    +GetByPath(path: string): FSObject
    +FindByCriteria(c: SearchCriteria): List<FSObject>
    +Exists(path: string): bool
```

+Save(obj: FSObject): void

+RemoveByPath(path: string): void

}

interface IFileRepository {

+GetByPath(path: string): File

+UpdateExtension(path: string, ext: string): void

}

interface IDirectoryRepository {

+GetByPath(path: string): Directory

+ListChildren(path: string): List<FSObject>

+AddChild(dirPath: string, obj: FSObject): void

+RemoveChild(dirPath: string, objPath: string): void

}

class InMemoryFSObjectRepository {

-store: Map<string, FSObject>

```
  }
  class InMemoryFileRepository {
    -store: Map<string, File>
  }
  class InMemoryDirectoryRepository {
    -store: Map<string, Directory>
    -children: Map<string, List<FSObject>>
  }

  InMemoryFSObjectRepository ..|> IFSObjectRepository
  InMemoryFileRepository ..|> IFileRepository
  InMemoryDirectoryRepository ..|> IDirectoryRepository

  IFSObjectRepository -up-|> IRepository : <FSObject>
  IFileRepository     -up-|> IRepository : <File>
  IDirectoryRepository-up-|> IRepository : <Directory>
}
```

Application.FileManagerService ..> Persistence.IFSObjectRepository

Application.FileManagerService ..> Persistence.IDirectoryRepository

Application.FileManagerService ..> Persistence.IFileRepository

Application.SearchService ..> Persistence.IFSObjectRepository

@enduml

3. Варіанти сценарії використання

Сценарій 1: Копіювання вибраних файлів/папок (із розв’язанням конфліктів)

Передумови:

- Застосунок файлового менеджера запущено.
- Користувач має доступ до вихідної та цільової директорій.
- Є хоча б один вибраний об’єкт (файл/папка).

Постумови:

- Вибрані об’єкти скопійовано в цільову директорію (успішно, частково або з помилками — згідно з політикою конфліктів).
- У списку/панелі відображено актуальний стан (нові копії, повідомлення про пропуски/помилки).

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:
Користувач копіює один або декілька вибраних об’єктів у вказану ціль; система за потреби запитує, як діяти при конфліктах (перезаписати/пропустити/перейменувати).

Основний потік подій:

1. Користувач виділяє файли/папки (мультивибір).
2. Обирає дію «Копіювати» та вказує цільову директорію (друга панель або діалог вибору).
3. Система перевіряє доступність цільового шляху та вільне місце.
4. Система починає копіювання, показує індикатор прогресу.
5. Якщо зустрічається об’єкт із тим самим іменем у цілі — система відкриває діалог розв’язання конфлікту (перезаписати/пропустити/перейменувати/застосувати до всіх).
6. Система завершує копіювання та оновлює вміст цільової панелі.
7. Система показує підсумок (скільки успішно, скільки пропущено, скільки з помилкою).

Винятки:

- Недостатньо місця → повідомлення, пропозиція обрати інший диск/звільнити місце.
- Ім’я занадто довге/некоректні символи → пропозиція перейменувати.
- Файл зайнятий іншою програмою → пропозиція закрити програму або пропустити об’єкт.
- Переривання користувачем → часткове копіювання; показати підсумок.

Примітки:

- Перед копіюванням для папок може виконуватися перевірка загального розміру/кількості об’єктів.

Сценарій 2: Пошук файлів/папок за критеріями

Передумови:

- Застосунок запущено, відкрита вихідна директорія або корінь диска.
- Користувач знає базові критерії (маска імені, розширення, дата/розмір).

Постумови:

- Відображено список результатів, доступний перехід до знайдених об’єктів або подальші дії (копіювати/перемістити/видалити тощо).

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач задає умови пошуку; система виконує рекурсивний пошук у поточній директорії/диску та показує знайдені об’єкти.

Основний потік подій:

1. Користувач обирає «Пошук».
2. У формі пошуку користувач задає критерії: маска імені (наприклад, `*.docx`), розширення, діапазон дат, мін/макс розмір, пошук у підпапках.
3. Користувач підтверджує запуск пошуку.

4. Система виконує пошук (рекурсивно або за індексом, якщо є), показує прогрес.
5. Система відображає результати списком із шляхами та базовими метаданими.
6. Користувач може перейти до каталогу знайденого об’єкта або виконати над ним дію (відкрити/копіювати/перемістити/видалити).

Винятки:

- Недоступні підкаталоги → система пропускає їх і позначає у звіті результатів.
- Занадто загальні критерії → попередження про велику вибірку, пропозиція звузити пошук.
- Порожні критерії → підказка заповнити хоча б одну умову.

Примітки:

- Результати можна відфільтрувати/відсортувати (за іменем, датою, розміром).
- Можливе збереження шаблонів пошуку.

Сценарій 3: Створення нової папки

Передумови:

- Застосунок файлового менеджера запущено.
- Відкрита директорія, у якій користувач має право створювати об’єкти.

Постумови:

- У поточній директорії з’являється нова папка з обраною назвою.
- Вона відображається у списку/панелі, доступна для подальших дій (копіювання, перейменування тощо).

Взаємодіючі сторони:
Користувач, Система.

Короткий опис:
Користувач створює нову директорію у вибраному місці; система перевіряє назву та додає папку у файлову систему.

Основний потік подій:

1. Користувач у відкритій директорії обирає функцію «Створити папку».
2. Система пропонує ввести назву нової папки.
3. Користувач вводить назву та підтверджує.
4. Система перевіряє коректність імені (довжина, заборонені символи, відсутність дубліката).
5. Якщо перевірка успішна — система створює директорію у файловій системі.
6. Панель оновлюється, показуючи нову папку у списку.

Винятки:

- Назва порожня/містить недозволені символи → система просить ввести іншу.
- У директорії вже існує папка з такою назвою → система пропонує іншу назву або відміняє операцію.
- Немає прав доступу → повідомлення про помилку, операція не виконується.

- Недостатньо місця/помилка файлової системи → повідомлення, створення не відбулося.

Примітки:

- За замовчуванням система може пропонувати назву «New Folder» із можливістю редагування.
- Після створення папка автоматично стає виділеною для зручності користувача

4. Проєктування БД

Схема відображає зв’язки між таблицями: об’єкти файлової системи пов’язані ієрархічно, операції мають записи про хід виконання, події та оброблювані об’єкти.

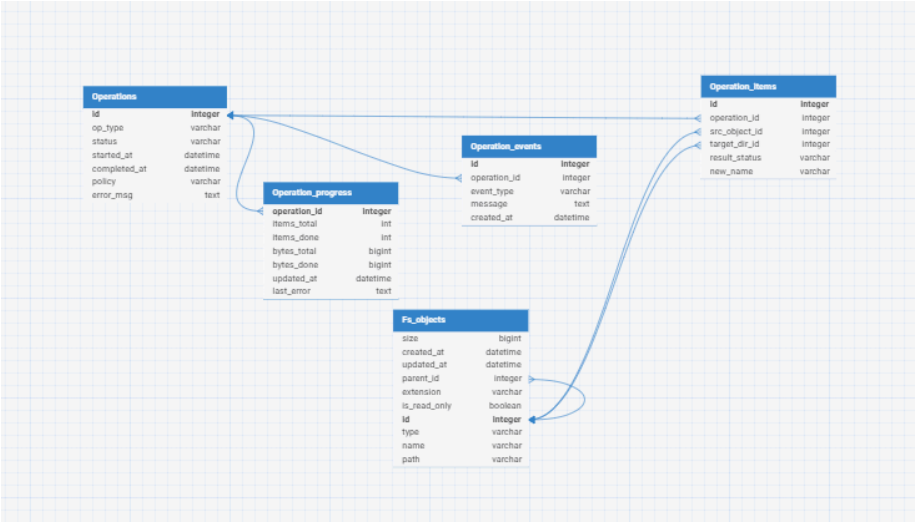

Рисунок 3 – Структура бази даних файлового менеджера

Висновок: Під час виконання лабораторної я спроєктувала систему Shell (аналог Total Commander): побудувала Use Case-діаграму з одним актором і варіантами використання для пошуку, копіювання, переміщення, видалення та перейменування об’єктів, використавши зв’язки include та extend. Узгодила її з діаграмою класів, де описала основні сутності файлової системи й операцій, а також перетворила модель на структуру бази даних із кількох таблиць і зовнішніми ключами. Реалізувала

основні класи домену й описала виконання операцій через збереження прогресу, подій і результатів, забезпечивши підтримку ключових функцій: робота з файлами й папками, історія дій, інтеграція із зовнішніми програмами. Я навчилася формулювати сценарії використання та переносити їх у модель даних, правильно задавати індекси й зв’язки між таблицями, а також працювати з нотаціями PlantUML для побудови діаграм.

Запитання на відповіді

1. Що таке UML?
UML (Unified Modeling Language) — це уніфікована мова моделювання, яка використовується для специфікації, візуалізації, проєктування та документування програмних систем. Вона дозволяє описати як статичну структуру, так і динамічну поведінку системи, використовуючи набір діаграм.

2. Що таке діаграма класів UML?
Це діаграма, яка відображає статичну структуру системи: класи, їх атрибути, методи та зв’язки між ними. Вона показує архітектуру системи з точки зору об’єктно-орієнтованого підходу.

3. Які діаграми UML називають канонічними?
До канонічних належать основні діаграми, що охоплюють ключові аспекти моделі: діаграма варіантів використання, діаграма класів, діаграма послідовностей, діаграма станів, діаграма діяльності, діаграма компонентів та діаграма розгортання.

4. Що таке діаграма варіантів використання?
Це концептуальна UML-діаграма, яка відображає зовнішню функціональність системи, показуючи, як актори (користувачі чи інші системи) взаємодіють із системою через різні варіанти використання.

5. Що таке варіант використання?
Варіант використання (use case) — це опис послідовності дій, які виконує система для досягнення певної цілі користувача. Він формалізує вимоги та показує, який сервіс система надає актору.

6. Які відношення можуть бути відображені на діаграмі використання?
Основні: асоціація (зв’язок актора з варіантом використання), узагальнення (успадкування акторів або варіантів), залежність (include, extend).

7. Що таке сценарій?
Це текстовий опис варіанта використання у вигляді покрокової інструкції, де зазначаються передумови, основні дії, винятки та постумови. Він деталізує варіант використання і робить його зрозумілим для реалізації.

8. Що таке діаграма класів?
Діаграма класів — це UML-діаграма, що моделює структуру системи у вигляді класів із їх атрибутами та методами, а також зв’язки між ними: асоціації, агрегації, композиції, наслідування.

9. Які зв’язки між класами ви знаєте?
Основні: асоціація (зв’язок між об’єктами), агрегація (частина-ціле зі слабким зв’язком), композиція (частина-ціле з сильним зв’язком, життєвий цикл залежить), успадкування (узагальнення).

10. Чим відрізняється композиція від агрегації?
Композиція означає сильний зв’язок: частини не можуть існувати без цілого (наприклад, квартира без будинку). Агрегація — слабший зв’язок, коли частини можуть існувати окремо (наприклад, група студентів, які можуть існувати незалежно від групи).

11. Чим відрізняються зв’язки типу агрегації від зв’язків композиції на діаграмах класів?
На діаграмі агрегація позначається порожнім ромбом біля цілого, композиція — зафарбованим ромбом. Агрегація показує незалежність життєвого циклу частини, композиція — повну залежність.

12. Що являють собою нормальні форми баз даних?
Це правила структуризації таблиць у реляційній БД, які зменшують надмірність і забезпечують логічну цілісність. Існують 1НФ (атоми даних), 2НФ (повна функціональна залежність від ключа), 3НФ (відсутність транзитивних залежностей), а також форма Бойса–Кодда для усунення аномалій.

13. Що таке фізична модель бази даних? Логічна?
Фізична модель — це опис бази даних з урахуванням конкретної СУБД: таблиці, індекси, типи даних, обмеження. Логічна модель — це абстрактний опис сутностей, атрибутів і зв’язків, незалежний від конкретної реалізації.

14. Який взаємозв'язок між логічною та фізичною моделлю бази даних?
Логічна модель є проміжним етапом між концептуальною моделлю та фізичною. Вона описує дані з точки зору бізнесу. Фізична модель реалізує цю структуру у вибраній СУБД. Тобто логічна модель трансформується у фізичну при переході від проєктування до реалізації.