МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 23.00.00.000 ПЗ

Група ІПм-22-2

Ветошкін Олександр

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Ветошкін Олександр Олексійович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та засоби підготовки даних в системах електронного документообігу

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Ветошкін О.О.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Бестильний Михайло Ярославович, асистент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

доц.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Ветошкіну Олександру Олексійовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та засоби підготовки даних в системах електронного документообігу ”

керівник проекту (роботи) Бестильний Михайло Ярославович, асистент

затверджені наказом закладу вищої освіти від “ 18 ” грудня 2023 р. № 738/7

2. Строк подання студентом проекту (роботи) 15 січня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови іформаційних та програмних технологій електронного документообігу

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Огляд методів використання електронних документів для баз даних
2. Дослідження задачі класифікації електроних документів табличної структури
3. Методи роботи з базою даних на основі шаблонів електронних документів
4. Інформаційна технологія застосування шаблонів електронних документів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Додавання нового вузлу за правилом Koo (рис. 1.1)
2. Лінійки таблиць у нотації EMF файлу (рис. 1.2)
3. Представлення таблиці ЕД (рис. 1.3)
4. Таблиця у вигляді ASCII-тексту (рис. 1.4)
5. Плоске представлення табличного об’єкту (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц., к.т.н. Вовк Р.Б.	
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	01.10.2023	виконано
2	Аналіз концепцій та алгоритмів предметної області	25.10.2023	виконано
3	Огляд методів використання електронних документів для баз даних	10.11.2023	виконано
4	Дослідження задачі класифікації електронних документів табличної структури	22.11.2023	виконано
5	Методи роботи з базою даних на основі шаблонів електронних документів	01.12.2023	виконано
6	Реалізація функціональності запропонованої інформаційної технології	15.12.2023	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 97 с., 15 рис., 9 табл., 46 джерел.

Тема: Моделі, методи та засоби підготовки даних в системах електронного документообігу

Об’єкт дослідження: інформаційний процес обміну даними між системою електронного документообігу та базою даних інформаційної системи.

Мета роботи: дослідження процесу обміну даними між системою електронного документообігу та базою даних за рахунок застосування методів і засобів структурування шаблонів електронних документів.

Предмет дослідження: моделі, методи структурування та побудови шаблонів електронних документів, методи ведення баз даних.

Результати дослідження:

В роботі досліджена об’єктна модель електронного документа, яка на відміну від існуючих моделей, представляє у якості об’єкта електронного документа інформаційну область, отриману засобами графічного оформлення або об’єднанням певних елементів, що дозволяє вибудувати залежність одного елементу документу від іншого.

Висновок:

Запропоновано документно-орієнтований шаблон, як спеціальну інформаційну структуру, що формально описує зв’язок слабко структурованих елементів електронного документа та жорстко детермінованих реляційних таблиць бази даних і дозволяє встановити однозначні інформаційні потоки між ними за допомогою *SQL*-запитів.

ЕЛЕКТРОННИЙ ДОКУМЕНТООБІГ, БАЗА ДАНИХ, ШАБЛОН, ІЄРАРХІЯ, ЗАПИТ, СТРУКТУРОВАНИЙ ЕЛЕМЕНТ, ЕЛЕКТРОННИЙ ДОКУМЕНТ, ОБ’ЄКТНА МОДЕЛЬ, ТАБЛИЧНА СТРУКТУРА.

ABSTRACT

Master Thesis: 97 pp., 15 fig., 9 tab., 46 sources.

Thesis Subject: Models, methods and means of data preparation in electronic document management systems.

Object of research: information process of data exchange between the electronic document management system and the database of the information system.

Research goal: to study the process of data exchange between the electronic document management system and the database through the use of methods and tools for structuring electronic document templates.

Subject of research: models, methods of structuring and construction of templates of electronic documents, methods of maintaining databases.

The results:

The object model of the electronic document is investigated, which, unlike the existing models, represents as the object of the electronic document the information area received by means of graphic design or association of certain elements that allows to build dependence of one element of the document on another.

Conclusion:

A document-oriented template is proposed as a special information structure that formally describes the relationship of weakly structured elements of an electronic document and rigidly determined relational database tables and allows establishing unambiguous information flows between them using SQL queries.

ELECTRONIC DOCUMENT FLOW, DATABASE, TEMPLATE, HIERARCHY, REQUEST, STRUCTURED ELEMENT, ELECTRONIC DOCUMENT, OBJECT MODEL, TABLE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕДО	–	електронний документообіг
ПКД	–	паперова копія документу
СЕД	–	система електронного документообігу
ЕД	–	електронний документ
БД	–	база даних
ГЗ	–	генератор звітів
ІТ	–	інформаційні технології
ІС	–	інформаційна система
ІО	–	інформаційна область
ЕФ	–	екранна форма
ПЗ	–	програмне забезпечення
ПрОбл	–	предметна область

ВСТУП

Актуальність теми. Електронний документообіг (ЕДО) наразі є невід'ємним елементом інформаційної структури будь-якої організації. Наряду цим, існують корпоративні інформаційні системи (КІС) підтримки ЕДО, які поєднують три ключові ланки: первинні паперові документи, електронні документи (ЕД), які дублюють інформацію паперових та інформацію таблиць бази даних (БД). Системи ЕДО дозволяють підвищити якість та ефективність управління, що безпосередньо пов'язане з поняттям актуальності даних, циркулюючих у системі. Забезпечити актуалізацію інформації можна завдяки своєчасним, правильним та налагодженим способам обміну даними між БД КІС та ЕД. Тобто необхідно вирішувати задачу поєднання слабоструктурованого інформаційного простору ЕД із жорстко детермінованою реляційною структурою таблиць БД КІС. Таку задачу називають задачею поєднання гетерогенних або різнорідних схем даних – *schema matching* (*SM*).

В літературі описані різноманітні підходи до узгодження різних схем даних, що засновані на структурі, семантиці та синтаксисі даних ЕД. Але недоліком всіх зазначених робіт є односпрямованість на поєднання схем даних чітко структурованих таблиць гетерогенних БД і жодним чином не вказується можливість та способи поєднання слабких структур ЕД та строгих реляційних таблиць БД. Також слід зазначити, що існує ряд програмних засобів які вирішують задачу добування (*Extract*), перетворення (*Transform*) та завантаження (*Load*) даних в КІС – *ETL* технології.

Такі технології не вирішують всі проблеми обміну даними між ЕД та БД КІС та мають такі недоліки як:

- високі витрати на придбання, налаштування і супроводження програмного забезпечення (ПЗ);

- необхідність втручання спеціаліста високої кваліфікації для налаштування однозначних інформаційних потоків між ЕД та БД КІС;

- спрямованість на конкретні формати та форми представлення інформації в ЕД;

- відсутність урахування того, що операторам притаманно змінювати, можливо і несвідомо, структуру ЕД;

- відсутність методів автоматизованої генерацій інтерфейсу доступу до даних таблиць БД у вигляді документно-орієнтованих екранних форм та проміжних детермінованих *XML* структур та частин електронного документу – *ED* структур.

Всі перераховані недоліки призводять до виникнення неоднозначності інформаційних зв'язків між ЕД та відповідних таблиць БД КІС, що знижує продуктивність обміну даними між ЕД та БД та значно підвищує кількість операцій встановлення відповідних зв'язків за рахунок втручання відповідного спеціаліста (адміністратора). Тому створення інформаційної технології (ІТ), яка включає удосконалені моделі та методи представлення і класифікації ЕД табличної структури, модифіковані методи породжуючого програмування *XML/ED*-шаблонів та документно-орієнтованих екранних форм доступу до БД є актуальною науково-практичною задачею, що дозволить скоротити кількість операцій визначення та встановлення однозначних інформаційних зв'язків між ЕД та таблицями БД КІС та підвищити продуктивність обміну даними.

Мета і завдання дослідження.

Метою роботи є дослідження процесу обміну даними між системою електронного документообігу та базою даних за рахунок застосування методів і засобів структурування шаблонів електронних документів.

Досягнення поставленої мети здійснюється шляхом розв'язання таких **задач**:

– аналіз методів та технологій автоматизованого обміну даних;

– дослідження об'єктної метамоделі електронного документу табличної структури;

– опис методу структурно-семантичної класифікації об'єктної метамоделі ЕД табличної структури;

– розробка шаблону обміну даними між електронним документом до БД на основі аналізу структур БД.

Об'єктом дослідження є інформаційний процес обміну даними між системою електронного документообігу та базою даних інформаційної системи.

Предметом дослідження є моделі, методи структурування та побудови шаблонів електронних документів, методи ведення баз даних.

Методи дослідження. Наведені в роботі результати базуються на теорії класифікації та кластерного аналізу текстів в системах інформаційного пошуку, методи теорії множин, теорія породжуючого програмування (*generative programming*).

Наукова новизна отриманих результатів. Наукова новизна результатів магістерської роботи полягає у тому, що досліджена об’єктна модель електронного документа, яка на відміну від існуючих моделей, представляє у якості об’єкта електронного документа інформаційну область, отриману засобами графічного оформлення або об’єднанням певних елементів, що дозволяє вибудувати залежність одного елементу документу від іншого.

Практичне значення отриманих результатів полягає в реалізації документно-орієнтованого шаблону, як спеціальної інформаційної структури, що формально описує зв’язок слабко структурованих елементів електронного документа та жорстко детермінованих реляційних таблиць БД і дозволяє встановити однозначні інформаційні потоки між ними за допомогою *SQL*-запитів.

Структура магістерської роботи. Робота складається зі вступу, чотирьох розділів основного змісту, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 97 сторінок, і містить 15 рисунків, 9 таблиць, список використаних джерел із 46 найменувань та 2 додатки.

РОЗДІЛ 1
ОГЛЯД МЕТОДІВ ВИКОРИСТАННЯ ЕЛЕКТРОННИХ ДОКУМЕНТІВ ДЛЯ БАЗ ДАНИХ

1.1 Загальний опис систем електронного документообігу

На сьогоднішній день склалася певна прихильність в роботі з ЕД з використанням офісних систем типа – *MS Office* і *Open Office*. Така перевага пов’язана, перш за все, з тим, що програми, що входять до складу даних пакетів, дозволяють максимально наблизитися до вихідного вигляду паперових копій документів, створюючи для користувача найбільш сприятливі умови роботи з електронними документами.

Електронні документи (ЕД) – це основні інформаційні ресурси підприємства, робота з якими вимагає правильної організації. ЕД забезпечують інформаційну підтримку ухвалення управлінських рішень на всіх рівнях і супроводжують ведення всіх бізнес-процесів.

Електронний документообіг (ЕДО) – це безперервний процес руху ЕД, що об'єктивно відображає діяльність підприємства і дозволяє оперативно управляти ними. Великі масиви ЕД, тривалий пошук потрібного ЕД, дублюючі ЕД, затримки з відправкою і отриманням, помилки персоналу складають не повний перелік проблем, що виникають при поганій організації ЕДО.

Ефективний документообіг є обов'язковою складовою ефективного управління підприємством та багато в чому залежить від добре розвиненої інформаційної системи (ІС) – корпоративної ІС (КІС), яка у свою чергу не може існувати без єдиного сховища даних – бази даних (БД). Багато організацій в своєму управлінні вже використовують подібні системи, але їх робота залишає бажати кращого. Поняття КІС в контексті забезпечення зв’язку між ЕД та БД потребує визначення. КІС – ІС масштабу підприємства, головною задачею якої інформаційна підтримка виробничих, адміністративних та управлінських процесів (бізнес процесів), що формують продукцію або послуги підприємства.

Під ІС слід розуміти сукупність методів, виробничих процесів та програмно технічних засобів, об’єднаних у технологічний ланцюг, що забезпечує зберігання, збір, обробку, висновки та розповсюдження інформації для зниження трудомісткості процесів і використання інформаційних ресурсів, підвищення надійності та оперативності.

В більшості організацій існує безліч ЕД, які створюються і обробляються в різних відділах різними особами, за допомогою офісних програм. В процесі роботи ці ЕД, за допомогою систем електронного документообігу (СЕД), передаються між відділами, де до них вносяться зміни і поправки, таким чином накопичується масив таких ЕД. Керівникам цих підприємств, для ефективного управління, необхідно провести аналіз накопичених даних. Але зробити це буде неможливе без єдиної БД КІС. Таким чином існує проблема первинного створення БД КІС і проблема підтримки актуальності вмісту БД КІС.

Але, не зважаючи на це, функціональні можливості СЕД в своїй більшості збігаються і надають ряд переваг: скорочення витрат часу керівників співробітників, виключення несанкціонованого доступу, прозорість бізнес-процесів, підвищення виконавчої дисципліни, виконання вимог стандартів *ISO 9000*, легкість впровадження інновацій і навчання, розвиток корпоративної культури, зростання конкурентних переваг.

Тому для вирішення питання доцільності впровадження в організації СЕД користуються наступними критеріями:

- Якщо оперативність пошуку необхідного ЕД є важливою.
- Якщо необхідно визначити чи виконане одне з ваших доручень або воно на певній стадії виконання.
- Швидкість узгодження ЕД створює позитивний імідж підрозділу.
- Купа паперів на робочому столі не дає змогу ефективно працювати.
- Необхідна достовірна інформація про місцезнаходження ЕД.
- Якщо будь який з перелічених критеріїв є актуальним для певної організації, то з’являється необхідність у створені та введені СЕД в дію.

На даний час на ринку представлено багато зарубіжних та вітчизняних СЕД, таких як: *NauDoc*, *МОТИВ*, *ЕВФРАТ-Документооборот*, *БОСС-Референт*, *jDocFlow*, *DocsVision*, *Effect Ooffice* та *Lotus документообіг*, які модифікуються у напрямку управління різного вигляду контентом (мультимедіа) та використання технологій автопроцесингу. Перелічені СЕД виконують задачі створення різних ЕД та різних видів звітності, спостеріганням за всіма етапами переміщення ЕД. Однак слід зазначити, що виконання функції " *schema matching*" в достатньому, для однозначного та автоматизованого визначення і встановлення інформаційних потоків між ЕД та таблицями БД КІС, обсязі не підтримує жодна з них.

Проблеми ведення БД в КІС

Для КІС, яка автоматизує управлінську діяльність організації, лише завершення етапу первинного заповнення БД на основі аналізу вмісту масиву ЕД, створених в організації без використання КІС, починає характеризувати її як працюючу ІС, а підтримка актуальності БД дозволяє приймати ефективні рішення.

В більшості організацій до появи ІС все діловиробництво ведеться із використанням настільних офісних систем (текстові редактори та електронні таблиці – *MS Office* та *OOffice*) і вся необхідна для первинного заповнення інформація, в основному, вже міститься у електронному вигляді.

Автоматизація процесу обміну вмісту ЕД та таблиць БД дозволить скоротити час первинного заповнення БД КІС та підтримки актуальності БД під час функціонування системи. Зокрема у організації, в якій функціонує КІС із розвинутим комплексом управління ЕДО, внаслідок організаційних або тимчасових технічних проблем ЕД можуть створюватись за межами системи, що призводить до необхідності повторного внесення вмісту ЕД до БД для погодженої роботи КІС [1]. При функціонуванні такої ІС виникає ряд проблем:

- висока трудомісткість процесу первинного заповнення БД ІС на основі аналізу вмісту ЕД;

- висока трудомісткість процесу підтримка актуальності БД ІС в процесі функціонування ІС;
- зниження достовірності даних, що вводяться;
- обмеження доступу користувачів до даних БД;
- перевага у виборі офісного програмного забезпечення (ПЗ).

Вирішення описаних проблем потребує ґрунтовного аналізу форматів і способів представлення даних у ЕД, приведення до єдиної уніфікованої форми ЕД певного класу, систематизація ЕД за допомогою одного з методів класифікації, розробка єдиної документно-орієнтованної ІТ, що дозволить підвищити продуктивності процесу обміну даними між ЕД та БД КІС в СЕД.

1.2 Формати представлення електронних документів

Основні інформаційні потоки підприємства представляють ЕД різних форматів, що обробляються за допомогою різних офісних та спеціалізованих програм. Однак, найчастіше для обробки ЕД використовується безкоштовний пакет офісного ПЗ – *Open Office* та *Microsoft Office*. До складу представлених пакетів входить як текстові процесори – *MS Word, OOWriter*, табличні процесори – *MS Excel, OOCalc*, а також системи управління БД – *MS Access, OOAccess*. Офісні пакети такого типу підтримують наступні формати зберігання даних і їх структури:

- *RTF (Rich Text Format)* формат збагаченого тексту – вільний міжплатформений формат зберігання розмічених текстових документів. *RTF* ЕД підтримуються більшістю сучасних текстових редакторів та забезпечують роботу над текстом з багатьма іншими можливостями його обробки;

- *DOC* – формат зберігання тексту з розміткою або без, який найчастіше використовується для позначення простих текстових файлів без особливого форматування, а з часом можливості такого формату представлення даних значно збільшилася у напрямку форматування та представлення тексту, навіть з’являються елементи мультимедіа;

- *XLS* – формат зберігання даних, що представляють таблицю, з розміткою або без, має зручну структуру для представлення інформації у вигляді таблиці та можливості об'єднання та оформлення комірок, та можливості виконання різноманітних математичних операцій;

- *HTML (Hypertext Markup Language)* – це стандартна мова розмітки документів в Усесвітній мережі, більшість *WEB*-сторінок створюються за допомогою мови *HTML*, а мова *HTML* інтерпретується браузером і відображується у вигляді ЕД, зручному для сприйняття людиною;

- *XML (eXtensible Markup Language)* – зведення загальних синтаксичних правил. *XML* – текстовий формат, призначений для зберігання структурованих даних, для обміну інформацією між програмами. *XML* має ряд переваг у можливостях міжплатформеного та міжпрограмного обміну:

- *Містить дані, розмічені тегами* – *XML* містить дані, що робить можливим його використання для передачі і зберігання даних. Розмітка даних за допомогою спеціальних маркерів (тегів) дозволяє реалізувати засоби для розбору документа (парсери).

- *Документ має ієрархічну структуру* – ієрархічність *XML* документа дозволяє зручно представляти записи, списки, дерева, таблиці.

- *Немає заздалегідь визначеного набору тегів* – дозволяє сформувати набір тегів, який найбільш оптимально підходить до конкретного завдання.

- *Стандарти, що описують формат документа,* – відкритість стандартів забезпечує однакову обробку *XML* документа засобами різних виробників.

- *Наявність засобів розбору* – в разі використання *XML* ми отримуємо готові парсери для мов *C++*, *C#, Java*, *PHP, Python*, *Delphi*, *JavaScript, JScript*.

- *Наявність засобів опису структури документа для конкретної предметної області і засобів валідації* – для *XML* досить описати структуру ЕД – елементи, що використовуються їх атрибути і вкладеність, обмеження на значення і обмеження на значення елементів за допомогою мов *DTD, XML Schema*. Після чого можна використовувати стандартні засоби – *XML*-валідатори для перевірки коректності ЕД.

Наявність засобів перетворення документа – дозволяє привести дані до певного вигляду (наприклад, сформувати таблицю) або представити дані в декількох форматах (*html, pdf, rtf, xls*). Наявність стандартних засобів для проведення такого перетворення – *XSLT* перетворення дозволяє, як і у випадку з розбором і валідацією, не реалізовувати свій конвертер для кожного завдання і абстрагуватися від використовуваної мови програмування.

Отже *XML* дозволяє позбавитися від проблем, пов'язаних з розбором, перевіркою і перетворенням даних, полегшує взаємодію компонент різних виробників, дозволяє використовувати його як при обміні даними між програмами, так і в *Web*. В той же час, його істотним недоліком є його надмірність, що в сукупності з текстовим форматом підвищує витрати на обробку даних. Крім того, не слід використовувати *XML* там, де структура даних дозволяє комфортно використовувати реляційну структуру.

Мова *XML* спровокувало появу 2-х нових *XML*-орієнтованих мов:

- *Office Open XML (OOXML)* – серія форматів файлів для зберігання ЕД пакетів офісних доданків – зокрема, *Microsoft Office*. Формат являє собою *zip* - архів, що містить текст у вигляді *XML*, графікою і іншими даними.

- *Open Document Format (ODF)* – відкритий формат файлів ЕД для зберігання і обміну редагованими офісними ЕД, у тому числі текстовими ЕД, електронними таблицями, рисунками, базами даних, презентаціями.

1.3 Способи представлення інформації в електронному документі

Представлення даних у вигляді таблиць, графіків, діаграм і просто тексту важливою частиною процесу аналізу даних і сприйняття інформації. Інформація, представлена у вигляді лінійного тексту, дозволяє у розповідному стилі описати будь-яке явище, проте, як правило, ефективніший спосіб сприйняття інформації – табличний або графічний [36]. За допомогою таблиць і діаграм можна швидше донести до читача основні положення і тенденцій в даних. Таблиці і діаграми особливо корисні, коли дані представляються деякій аудиторії, тому що інформація має бути передана протягом обмеженого періоду часу. Також

відміною рисою таблично-представлених даних є змога агрегувати дані по деяких категоріях та виявляти необхідні залежності окремих елементів в достатньо малі проміжки.

Текстовий спосіб представлення даних це найпростіший спосіб подання, що описує нескладні залежності різноманітних об'єктів і процесів, що не вимагають аналітичної обробки. Наглядним прикладом текстового способу представлення інформації є художня література.

Таблична форма представлення даних впорядковує інформацію у вигляді рядків (по горизонталі) і колонок (по вертикалі). Перетин рядків і стовпців дозволяє утворити комірки.

Найбільш важлива перевага таблиць – організація даних для подальшої статистичної обробки і ухвалення рішення. Таблична форма подання дозволяє оцінити дані наступного характеру [37]:

- якісного – коли оцінюються об'єкти якісної природи соціальний статус, національність, характер;
- кількісного – коли характеристика того або іншого об'єкту може бути оцінена кількісно, наприклад: зріст, вага, прибуток;
- часового – коли об'єкт характеризується у відрізках часу: рік, місяць, година, секунда.
- просторового – об'єкт оцінювання зав'язаний на визначене місце розташування: місто, країна, вулиця.

Графічний метод забезпечує найшвидше розуміння фактичної ситуації, яку пояснюють дані представлені таблицями або текстом, але менш точний.

У роботах [38, 39] представлені переваги від використання різного типу представлення інформації. Для визначення того, який тип найбільш придатний для представлення ЕД, по-перше, необхідно визначити для кого призначена інформація і з якою метою вона може бути використана. Якщо йдеться про графічне представлення інформації, то вона більш придатна для кінцевого користувача, але якщо інформація призначена для швидкого і точного аналізу, то таблична форма є найкращим варіантом. Проте у роботах [40, 41], вказано, що

взаємне використання 2-х способів представлення інформації може поліпшити як програмний аналіз даних, так і візуальне сприйняття кінцевим користувачем.

1.4 Моделі представлення електронних документів

Модель регулярного ортогонального чередування розбиттів. Традиційною моделлю для опису текстових ЕД є деревовидна структура аркуша: аркуш розбитий на текстові блоки, звані колонками, колонки розбиті на параграфи, параграфи складаються із слів, слова – з символів, символи – з однієї або декількох компонент зв'язності.

Для кожного об'єкту, що має “ спадкоємців”, зберігається інформація про відносне взаємне розташування його “ спадкоємців” на аркуші, що дозволяє у разі потреби зберегти у вихідному ЕД те ж взаємне розташування структурних елементів тексту в межах ієрархічного блоку більш високого рівня, що і у ЕД. Передбачені наступні варіанти взаємного розташування об'єктів: впорядкування по горизонталі та вертикалі, матричний порядок, індексний порядок, неврегульоване розташування.

Дерево регулярного ортогонального чередування зберігає лише деяку інформацію про взаємне розташування елементів таблиці, але цієї інформації недостатньо, щоб відтворити вихідну структуру таблиці у вихідному ЕД.

Графова модель Коо

Модель ЕД, що включає наступні складові:

- специфікація класу ЕД, що обробляються (наприклад, клас ЕД з табличною структурою);

- список початкових елементів ЕД, які будуть доповнені автором у повній версії ЕД (наприклад, початкові елементи таблиці);

- набір правил модифікації ЕД, тобто засоби за допомогою яких початкові елементи ЕД будуть змінені (наприклад, для табличних даних, порядок збільшення рядків чи стовпців).

Модель *Koo* представляє ЕД як направлений ациклічний граф, можливості якого більші ніж у абстрактних моделей, особливо коли їх семантичну властивість описують додатковими формалізмами. Під семантичною властивість розуміється не тільки текстова складова але й структурна організація ЕД.

Тобто *Koo* пропонує направлений ациклічний граф для зображення ЕД та ряд правил модифікації графової моделі. На рисунку 1.1 представлено граф ЕД та правило його модифікації – додавання нового вузлу ЕД.

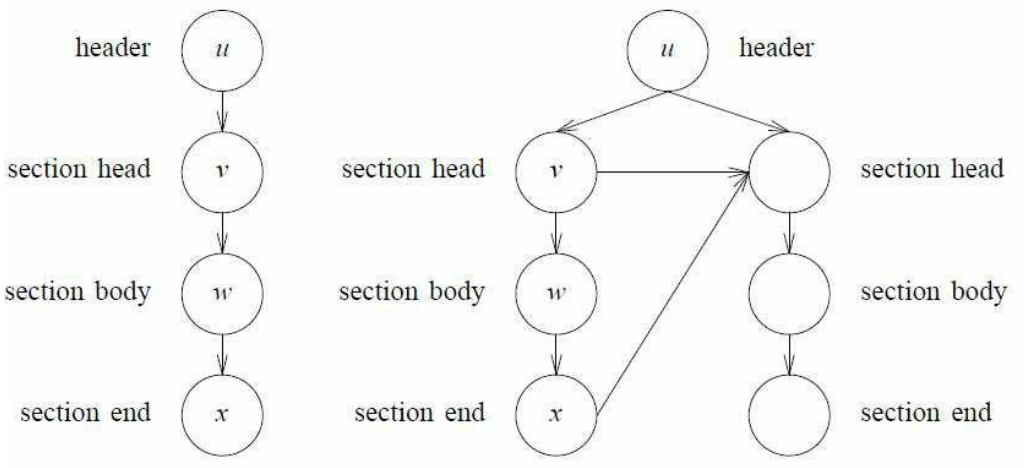


Рис. 1.1. Додавання нового вузлу за правилом *Koo*

Модель на основі метафайлів

Розглянемо модель представлення ЕД з табличною структурою, заснована на метафайлах. Метафайли використовуються в операційних системах *Microsoft Windows* при процесі друку у якості файлів спулінгу. Таким чином ЕД, що потрапляють принтеру на друк (у тому числі і віртуальному) передаються у якості команд *GDI*, які перетворюються у метафайли формату *EMF*. Проміжні файли у форматі *EMF*, можна перехопити у процесі спулінгу та продовжити їх обробку окремо.

Метафайл є бінарним файлом і впорядкований, як послідовність записів. Перший запис це заголовок, який вміщує різноманітні метрики файлу, такі як геометричні розміри аркушу ЕД, наділі слідує записи *EMR* (*Enhanced Metafile Record*), кожна з яких представляє одну команду *GDI*, що виводить графіку.

Інтерфейс *GDI* являє собою механізм інтерпретації метафайлів формату *EMF*. Таким чином за допомогою функцій *GDI* можна отримувати всю необхідну інформацію о ЕД: текст, позицію виводу тексту, інтервали, розміри шрифту, зазори між символами, тощо. Лінійки таблиць у метафайлах представляються за допомогою записів типу *EMR_BITBLT*, зображених на рисунку 1.2, що відображає формальний опис інформаційної області створеної лінійками розмітки у нотації *EMF* файлів.

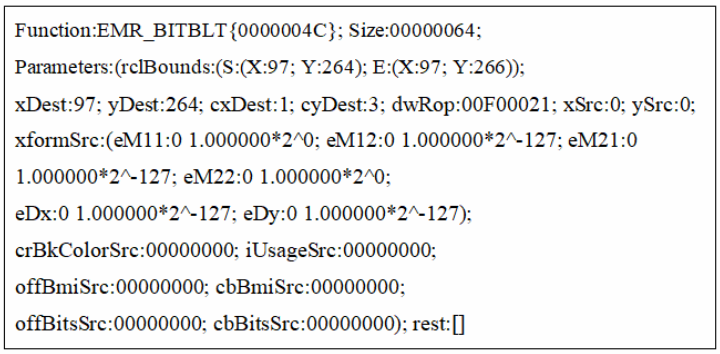


Рис.1.2. Лінійки таблиць у нотації *EMF* файлу

Модель множини слів – bag-of-words

Bag-of-words модель представлення ЕД отримала свою назву ще на початку 1950 року, коли з'явились перші інформаційно-пошукові системи. На той період часу проблема об'єму пам'яті стояла дуже гостро і зберігати повні об'єми ЕД не було можливим, не говорячи про структуру та оформлення ЕД. Тому ЕД документи зберігалися і оброблялися як набір визначень – *bag-of-words*. Але з плином часу текст розглядається вже як об'єкт зі складною структурою. Модель *bag-of-words* розглядає взаємне розташування слів у ЕД, а якості математичного апарату використовує функцію зважження. Однак слід зазначити, що запропонована модель довела свою ефективність в системах інформаційного пошуку.

Об'єктна модель ЕД (Document Object Mode - DOM) – є набором інтерфейсів для побудови об'єктного представлення у формі дерева аналізованого ЕД. Побудувавши *DOM*-модель, можна управляти нею за допомогою методів *insert* і *remove*, аналогічно роботі з деревами.

1.5 Методи структурування електронних документів

1.5.1 Таблична форма представлення інформації

На даний час в різних організаціях існує велика кількість документів, що зберігаються в різних електронних форматах. Зручним та наочним засобом відображення інформації є її таблична структура, яка дозволяє згрупувати дані по стовпцях та рядках. Такий засіб значно полегшує використання інформації, а також задачі її аналізу, упорядкування за деякими ознаками та зіставлення з даними з інших джерел. За допомогою таблиці зручно представляти такі ЕД: плани, розклади, журнали, телефонні довідники, відомості, звіти, анкети, тощо.

Згідно визначення таблиця – це особлива форма передачі вмісту, яку відрізняє від тексту організація слів та чисел у колонки (графи) та горизонтальні рядки таким чином, що кожний елемент є одночасно складовою частиною і рядку і колонки. Інформація у таблицях обов’язково впорядкована за визначеним принципом, наприклад у розкладі занять – за днями тижня та номером уроків. Між заголовком колонки, заголовком рядку та їх загальним елементом встановлюється безсловесний, графічно смисловий зв’язок, що розуміється читачем без перекладу у словесну форму. Така впорядкованість дозволяє швидко знаходити необхідні відомості. Однак, встановити смисловий зв'язок між елементами тексту можна і без використання таблиці, але для цього знадобиться багато додаткових надлишкових слів, в яких складно вловити закономірність або залежність, що виражена в таблиці. Таким чином інформація, що представлена таблицею, сприймається людиною найкраще.

1.5.2 Способи представлення таблиць в ЕД

Складність автоматичного видобування даних з табличною організацією обумовлена в першу чергу великою різноманітністю форм зображення таблиць. Відомі методи і технології обробки табличних даних орієнтовані на заздалегідь визначені структури і особливості таблиць, які пов'язані із стандартами вибраної предметної області.

Інформація в таблицях обов'язково впорядкована за якимсь принципом, наприклад в розкладі занять – по днях тижня і номерах уроків. Між заголовком колонки, заголовком рядка і їх спільним елементом встановлюється безсловесний, графічний смисловий зв'язок, що розуміється читачем без перекладу в словесну форму. Така впорядкованість дозволяє швидко знаходити таблиці потрібні відомості. Безперечно, встановити смисловий зв'язок між елементами тесту можна встановити і без використання таблиці, але для цього знадобилося б багато додаткових слів, в яких важче було б уловити закономірність або залежність, виражену в таблиці.

Огляд таких систем оптичного розпізнавання тексту як “ *OmniPage”* , “*FineReader*” і “ *Cuneiform*” показав, що основним їх завданням є розпізнавання тексту, а завдання розпізнавання таблиць орієнтоване на гратчасту структуру і погано справляється з таблицями складної організації (об'єднання рядків/стовпців). Також *OCR*-системи працюють виключно з графічними зображеннями, а більшість ЕД містять таблиці, організовані у вигляді комірок з візуально оформленими частинами, рисунок 1.3.

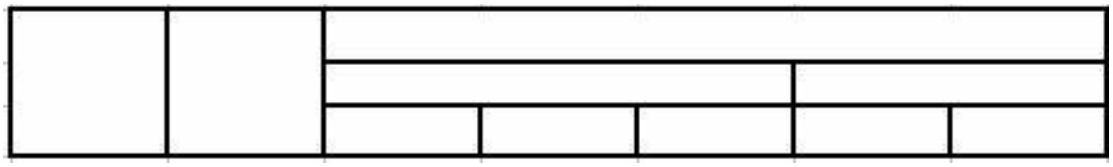


Рис. 1.3. Представлення таблиці ЕД

Часто в якості представлення ЕД використовується ASCII-текст, використовуючи для визначення таблиць спеціальні символи (- =, +, T, L), змальовані на рисунку 1.4.

```
T-------------------------------------------------------------------------------T
|       1               |                               2                       |
|-------------------------------------------------------------------------------|
|           3           |   4       |       5   |       6       |       7       |
|-------------------------------------------------------------------------------|
|                       |           |           |               |               |
|           9           |   10      |       11  |       12      |       13      |
|-------------------------------------------------------------------------------|
```

Рис. 1.4. Таблиця у вигляді ASCII-тексту

Також в якості представлення ЕД використовуються *HTML/XHTML/XML*.

1.5.3 Об’єктна модель таблиці

Важливою властивістю систем автоматизованого вводу документів з отсканованого графічного образу стає можливість “ розуміння” структури первинного ЕД і створення адекватного електронного уявлення. Істотну долю серед ЕД, що обробляються, складають табличні ЕД, для яких склад, розташування та структура інформації певним образом фіксовані. Для систем, які спеціалізуються на вводі і обробці табличних документів, апріорне знання структури та правил заповнення є важливою вхідною інформацією, істотно розширюючи технологічні можливості обробки документа. Серед технологій, які дозволяють створювати та редагувати таблиці виділяють наступні:

- в області генерації паперових та електронних документів – настільні видавницькі системи та генератори звітів.

- в області зберігання та маніпулювання даними – СУБД.

- в області розрахунків та візуального маніпулювання плоскими представленнями – табличні процесори.

- в області вводу – системи оптичного розпізнавання.

Всі запропоновані технології вирішують якісь приватні проблеми і не розглядають задачу обробки табличних документів як одне ціле.

В роботі [34] досліджуються моделі, методи и засоби обробки табличних документів в інформаційних системах і приводиться універсальна структурно орієнтована модель таблиці. Комплексна модель таблиці складається із наступних погоджених моделей:

- модель табличного інформаційного об’єкту, яка описує класи, які допускають плоске представлення інформаційного об’єкту;

- модель плоского представлення.

Таблиця розглядається, як частина електронного документу. Модель електронного документу

$$M_{DOC}=<M_D, M_I, M_V>$$

складається із моделі змісту M_D, взаємодії M_I і візуалізації M_V.

Загальна схема екземпляру плоского представлення табличного інформаційного об’єкту в ЕД представлена на рисунку 1.5 і складається з областей наступних типів: ідентифікаційна зона, кутова заголовочна зона, горизонтальна заголовочна зона, вертикальна заголовочна зона, врізана заголовочна зона, матричне ядро, інформаційна зона.

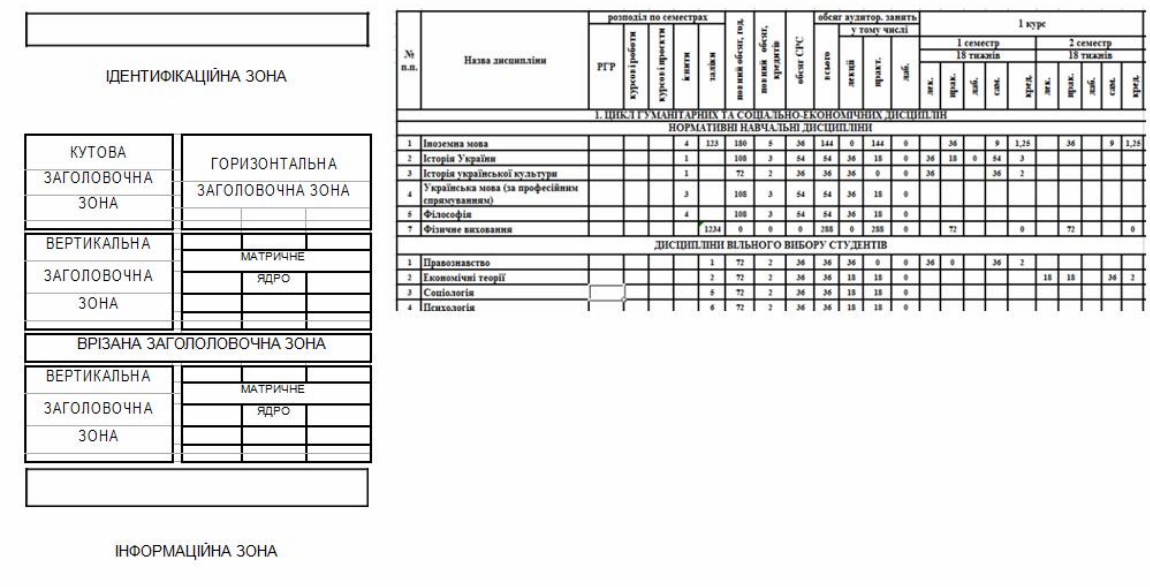


Рис. 1.5. Плоске представлення табличного об’єкту

Розглядаються вертикальні заголовочні зони 4-х типів: матриця (рисунок 1.6, а), Т-ієрархія (рисунок 1.6, б), М-ієрархія (рисунок 1.6, в), Г-ієрархія (рисунок 1.6, г).

а б в г

Рис. 1.6. Чотири типи заголовочних форм ЕД

Для горизонтальної заголовочної зони допустимі типи:

- матриця,

- Т-ієрархія,

- Г-ієрархія.

Заголовочна зона представлена лісом дерев, в яких шляхи від кореня до вершин визначають унікальний для зони ключ, який складається зі значення комірок заголовочної зони. Тіло таблиці в загальному випадку представляється матричною таблицею, яка погоджена із заготовочними областями геометрично. Врізка є способом компактного розміщення одного або декількох старших рівнів заголовків вертикальної зони. Ідентифікаційна і інформаційна зони містять загальну неструктуровану табличну інформацію.

1.5.4 Формування вектора ознак ЕД

Першочерговою в задачі класифікації є побудова класів електронних документів. Задача побудови класів ЕД є задача кластеризації або формування образів електронних документів. Образ електронного документу представляється у вигляді вектору у багатомірному просторі ознак, значимість яких заснована на розрахунку мір схожості.

Підходи до формування вектора ознак електронних документів можуть істотно розрізнятися. У простому випадку кожна ознака відповідає наявності в тексті однієї із словоформ, що зустрічається в даному наборі текстів. Величина компоненти вектора теж може визначатися по-різному: компонента може дорівнювати 1, якщо даний термін присутній, і 0 в протилежному випадку; може дорівнювати числу входжень терміну в ЕД – *term frequency* (*TF*) або інверсія частоти – *inverse document frequency* (*IDF*), або обчислюватися по формулам, що враховують середню зустрічаємість словоформи по всьому набору текстів – *term frequency*inverse document frequency* (*TFIDF*).

Мірою близькості між текстами вважається скалярний добуток між відповідними векторами. Більшість алгоритмів кластеризації оперують у якості вхідних даних прямокутною матрицею *T*, складеною з векторів електронних

документів, і симетричною квадратною матрицею $S=T\cdot T^t$, іменованою матрицею близькості. Матриця *T*, складена з векторів *TFIDF*. Основною проблемою методів, заснованих на такій матриці, виявляється дуже велика розмірність простору ознак, велика частина яких є надлишковий.

Для зменшення розмірності простору ознак можуть застосовуватися, зокрема, наступні прийоми: використання стоп-листів; використання методів лінгвістики, використання словників і тезаурусів для групування словоформ у нормальні форми і об'єднання нормальних форм в синонімічні групи.

Деякі алгоритми кластеризації використовують як елементи вектора *TF* не окремі словоформи, а іменні або дієслівні групи, імена власні, стійкі словосполуки. Для уточнення величин компонент вектора *TF* можуть застосовуватися всілякі методи омонімії та і полісемії.

При використанні описаного представлення текстів кластер виявляється набором векторів $V_1, \ldots, V_n$ в просторі ознак. Функція близькості між кластерами виводиться тим або іншим чином з функції близькості між векторами. Єдине обмеження – вимога монотонності функції близькості.

Важливою характеристикою кластера є положення його центроїда, тобто вектора центру мас, середнього арифметичного векторів.

1.5.5 Методи класифікації ЕД

Класифікація ЕД – процедура що відносить вхідний ЕД до певного класу електронних зважаючи на певні критерії, що його характеризує.

Імовірнісні класифікатори (Probabilistic classifiers). В імовірнісному підході функція приналежності ЕД до категорії $CSV_i(d_j)$ розглядається як імовірність $P(c_i|d_j)$ того, що ЕД $d_j=<w_{1j},\ldots, w_{|n|j}>$ потрапляє у категорію c_i, яка вираховується теоремою Байеса.

Нечислові класифікатори. Імовірнісний підхід має кількісну (чисельну) природу – складну для людського розуміння, тому існують символьні (не числові) методи, які в більшій мірі зрозумілі людині.

Класифікатори на основі правил і дерев прийняття рішень. Класифікатор на основі дерева прийняття рішень являє собою дерево, внутрішні вершини якого позначені термінами, а виходять з них ребра помічені перевірочними вагами. Вага терміна у векторному поданні тестового ЕД порівнюється з цими перевірочними вагами. Листя даного дерева розмічені значеннями категорій. Класифікатор подібного типу послідовно проходить по вершинах дерева, порівнюючи вагу терміна поточної вершини в представленні ЕД і визначаючи подальший напрям обходу, до тих пір поки не досягне кінця листа. Значення категорії даного листа і присвоюється ЕД. Існують ряд методів побудови дерева прийняття рішень на основі навчальної вибірки: *ID3, C4.5* та *C5*.

Класифікатор, побудований за методом правил прийняття рішень, складається з диз'юнктивних нормальних форм. У посилці затверджується наявність або відсутність терміна в ЕД, а у висновку міститься рішення про класифікацію ЕД з даної категорії.

Регресійні методи засновані на апроксимації бінарної цільової функції Φ, отриманої при аналізі навчальної множини, функцією Φ', значеннями якої є дійсні числа. Найбільш розповсюдженими представниками класу регресійних методів є модель лінійного підбору методом найменших квадратів – *Linear Least-Squares Fit* (*LLSF*).

В результаті задача класифікації полягає в визначені вихідного вектора для тестового ЕД. Звідси побудова класифікатора зводиться до прорахунку

$|C|\times|T|$ матриці $\hat{M}$ – такої, що $\hat{M}\, I(d_j)=O(d_j)$, де I – $|T|\times|T_r|$ матриця, в якій стовпцями є вхідні вектори навчальних документів, O – $|C|\times|T_r|$ матриця, в якій стовпцями є вихідні вектори навчальних документів. *LLSF* розраховує матрицю навчальних даних за допомогою лінійного підбору методом найменших квадратів (який мінімізує помилку на навчальних даних).

Нейроні мережі є потужним інструментом розпізнавання образів. Нейрона мережа – набор нейронів, певним чином пов'язаних між собою. Зв'язки між нейронами мають ваги, а самі нейрони – один з декількох стандартних алгоритмів

поведінки. Фактично навчання мережі зводиться до подачі на її входи спеціально підібраних сигналів, для яких заздалегідь відомий результат роботи мережі, і коригування коефіцієнтів кожного з нейронів мережі таким чином, щоб мережа видавала саме це значення.

Методи, засновані на прикладах, " паразитують" на рішеннях, прийнятих експертом при категоризації навчальних ЕД, діючи таким же чином на тестових ЕД. Тому такі методи ще називають " ледачими" системами (*lazy learners*), так як вони відкладають рішення про класифікацію до тих пір, поки не буде розглянуто кожен новий екземпляр ЕД.

Метод k-nearest neighbors приймає рішення про віднесення ЕД d_j до категорії c_i, *k-NN* проглядає *k* найбільш схожих на d_j ЕД і якщо більшість з них відноситься до тієї ж категорії, приймається позитивне рішення. Математично класифікація документа за допомогою *k-NN* зводиться до розрахунку:

$$CSV_i(d_j)=СУММ(RSV(d_j, d'_z) \times ca_{iz})$$

де $Tr_k(d_j)$ – множина *k* ЕД d'_z, на яких досягається максимум $RSV(d_j, d'_z)$, а ca_{iz} – значення з коректної матриці прийняття рішення. В свою чергу, $RSV(d_j, d'_z)$ – це деяка міра схожості між тестовим ЕД d_j та документом d'_z, що навчається; для цих цілей може бути використана будь-яка функція схожості, або імовірнісна або векторна міра з пошукової системи.

Метод моделювання максимальної ентропії (*maximum entropy modeling – EM*) – це інструмент об'єднання інформації з багатьох гетерогенних інформаційних джерел для класифікації. Дані на вході рішення класифікаційної проблеми описуються як безліч ознак, ці ознаки можуть бути досить складними дозволяють досліднику використовувати апріорні знання про те, які типи інформації можуть бути корисними для класифікації. Кожна ознака відповідає обмеженню моделі. Потім обчислюється модель з максимальною ентропією серед всіх моделей, які задовольняють обмеженням.

1.5.6 Методи кластеризації ЕД

Кластеризація ЕД – одне із завдань інформаційного пошуку. Метою кластеризації документів є автоматичне виявлення груп семантично схожих ЕД серед заданої фіксованої безлічі документів. Слід зазначити, що групи формуються тільки на основі попарної схожості описів документів, і ніякі характеристики цих груп не задаються заздалегідь, на відміну від класифікації документів, де категорії задаються заздалегідь.

Метод K-середніх (*K-means*) являє собою модифікацію EM-алгоритму для поділу суміші Гауссіан. Він розбиває множину елементів векторного простору на заздалегідь відоме число кластерів k. Дія алгоритму така, що він прагне мінімізувати середньоквадратичне відхилення на точках кожного кластера.

Графові алгоритми кластеризації засновані на представленні вибірки у вигляді графа. Вершинам графа відповідають об'єкти вибірки, а ребрам попарні відстані між об'єктами $p_{ij}=\rho(x_i, x_j)$.

Алгоритми FOREL використовують критерій *F*, заснований на гіпотезі компактності: в один таксон повинні збиратися об'єкти, “ схожі” за своїми властивостями на деякий “ центральний” об'єкт сферичної форми.

Ієрархічна кластеризація – таксономія. Метод розбиття заданої вибірки об'єктів на непересічні підмножини, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися.

EM-алгоритм. Алгоритм, який використовується в математичній статистиці для знаходження оцінок максимальної правдоподібності параметрів імовірнісних моделей, у випадку, коли модель залежить від деяких прихованих змінних. Кожна ітерація алгоритму складається з двох кроків. На *E*-кроці (*expectation*) обчислюється очікуване значення функції правдоподібності, при цьому приховані змінні розглядаються як спостережувані. На *M*-кроці (*maximization*) обчислюється оцінка максимальної правдоподібності, таким чином збільшується очікуване правдоподібність, що обчислюється на *E*-кроці. Потім це значення використовується для *E*-кроку на наступній ітерації. Алгоритм виконується до збіжності.

1.5.7 Латентно-семантичний аналіз ЕД

Метод визначення подібності значень слів і документів шляхом статистичних обчислень над великим корпусом текстів і не вимагає ніякої додаткової інформації (словники, семантичні мережі, бази знань).

В основі методу *LSA* лежить гіпотеза про те, що між словами і тим контекстом, в якому вони вживаються, існують неявні (латентні) взаємозв'язки. Передбачається, що семантичне значення документа може бути представлене як сума значень вхідних у нього слів:

значення(документ)=значення(слово 1)+ значення(слово 2)+…+ значення(слово m)

Метод дозволяє обчислити кореляції між парою термінів, між парою документів і між терміном і документом. Кожний рядок вихідної матриці C – вектор, що відповідає терміну t_i і показує його зв’язок з кожним з документів d_i корпусу ЕД. Кожен стовпець вихідної матриці – вектор, що відповідає ЕД і показує його зв'язок з кожним з термінів в корпусі ЕД.

Метод *LSA* полягає в сингулярному розкладанні матриці C (*SVD - singular value decomposition*) і апроксимації її матрицею меншого рангу.

Перевагами *LSA* методу кластеризації текстів є те, що він не потребує етапу навчання, тобто формується така структура кластерів, що залежить виключно від даних що обробляються. Крім того немає необхідності попереднього налаштування алгоритму.

1.6 Аналіз програмного забезпечення автоматизації процесу створення доступу до баз даних

1.6.1 Методи породжуючого програмування

Генеративні методи

Ідеї породжуючого програмування ґрунтуються на проектуванні і побудові породжуючих моделей для сімейств систем з метою генерування по цих моделях

конкретної системи. За наявності практичного досвіду при розробці сімейства систем в певній предметній області з'являється можливість розробляти серійне програмне забезпечення. Розробка серійних компонентів дозволяє виявити як спільності членів сімейства, так і наявність істотних параметрів мінливості. Виявлення особливостей предметної області з можливістю моделювання характеристик проекту надають можливість визначити, які характеристики і змінні параметри можуть бути реалізовані відразу, а які доцільно заплановані на майбутнє.

Генератори – різні технології, включаючи препроцесори, метафункції, компілятори, що генерують класи і процедури, генератори коду *CASE*-систем, трансформаційні компоненти, тощо. Наприклад, за допомогою генератора можна автоматично згенерувати реалізацію програми на машинній мові або у вигляді байтового коду, вихідний код якої був написаний на високорівневій мові програмування. Іншим прикладом генератора є обробник метафункцій в метапрограмуванні на основі шаблонів *C++* з генерацією класів і процедур.

Більшість *CASE*-систем містять генератори реалізацій графічних моделей вигляді описів на мові програмування. Деякі системи метапрограмування дозволяють використовувати генератори, маючи як вхідні специфікації конфігурації компонентів у вигляді предметно-орієнтованих нотацій і фрагменти коду на високорівневій мові.

Автоматні методи

Проте відомі інструменти не дозволяють повною мірою ефективно пов'язувати “ скелет” коду з моделлю поведінки, яку можна описувати за допомогою діаграм станів, діяльності, кооперації або послідовностей.

Відсутність однозначної операційної семантики, на думку авторів, при традиційному написанні програм наводить до відмінності опису поведінки в моделі і в програмі, а також до довільної інтерпретації програмістами поведінкових діаграм, а опис поведінки в моделі часто носить неформальний характер. Можлива ситуація, коли формальна модель поведінки будується архітектором, а програмісти при розробці вихідного коду її не використовують,

виконуючи розробку із застосуванням власного тлумачення однозначно описаної моделі. Поява операційної семантики в мові *UML* ідентифікує однозначність розуміння діаграм учасниками проектування ПЗ і дозволить створити виконуваний *UML*, а генерація коду при цьому може, зокрема, виконуватися безпосередньо при інтерпретації описаної моделі.

Інтенціальне програмування

Автоматизація застосування породжуючого програмування супроводжується дослідженнями по створенню спеціальних систем (середовищ) метапрограмування, які пропонують ширшу підтримку породжуючого програмування. Ці системи підтримують автоматичний рефакторінг і дозволяють зібрати більше знань по проектуванню, чим при традиційному програмуванні. Цим забезпечується вищий рівень автоматизації рефакторінга або іншого виду супроводу (розвитку) програмного забезпечення, оскільки зменшується кількість проектних рішень, які повинні передувати кодуванню. Застосування таких систем дозволяє знижувати обмеження, пов'язані з можливостями конкретних мов програмування, і дозволяють перенастроювати існуючі доданки на нові платформи з мінімізацією витрат за рахунок виключення етапів переписування програм на інші мови програмування. Яскравим і поки єдиним прикладом реалізацій таких систем є система *IP*, розроблена Чарльзом Симоні в період його роботи в корпорації *Microsoft*. Робота в середовищі метапрограмування *IP* (*Intentional Programming*) інколи називається ментальним або інтенціальним програмування.

1.6.2 Технології перенесення змістовної частини ЕД до БД

Технології взаємного відображення даних між ЕД та БД засновані на реалізації у складі систем керування БД (СУБД) прикладного ПЗ, що дозволяє налаштувати певний канал зв’язку між даними, що вносяться та структурами таблиць БД. Однак, як вже зазначалось, документно-орієнтований підхід вимагає наближення зовнішнього виду ПЗ до вихідних ЕД, що проблематично реалізувати

заздалегідь у зв’язку з безліччю можливих варіантів ЕД та їх представлень. Тому для вирішення таких проблем було створено технологію динамічного генерування звітів (ГЗ) або форм (ГФ).

ГФ – програма, що дозволяє представити інформацію в легкому для читання структурованому вигляді – ЕД-звіти, які можна роздрукувати або зберегти в різних електронних форматах.

На даний час існує велика кількість програмних комплексів, які підтримують механізм генерації звітів, серед них найбільш відомими є *Oracle Report, Centura Report Builder, Crystal Report, SoDA, Access, 1С:Предприятие, FastReport, LanDocs, NauDoc, Optima-WorkFlow, PayDox, Q&R, БОСС-Референт, Гран-Док, Дело, Documentum, CalliGraph, Cognitive Report.*

Функції будь-якого з перелічених ГЗ входить створення ЕД на основі вмісту БД, таким чином використовуючи алгоритми, закладені в них, можна створювати шаблони ЕД, та використовувати механізми обміну ЕД з БД.

Oracle Report. Потужний засіб, який включає вбудовану мову програмування для створення звітів, майстер запитів до БД, шаблони типових звітів у СУБД *Oracle*.

Crystal Report. Потужний ГЗ з багатьма можливостями але відсутній майстер таблиць, а також виникають труднощі з кількість символів, що відображаються на екрані.

Access. ГЗ компанії *Microsoft*, який вбудовано в СУБД *Access*. Добре продуманий майстер звітів з можливістю експорту *Word, Excel*. Можливості групування, форматування об’єктів, створення формул. Недоліки: відсутність майстра таблиць і взагалі можливість створення таблиць. Всі таблиці в кінцевому звіті створюються у табульованому вигляді.

1С:Підприємство (1С). Багатий набір звітів з можливістю налаштування і збереження цього налаштування. Користувач може самостійно задавати рівень деталізації, параметри угрупування і критерії відбору даних в звітах відповідно до специфіки вирішуваних завдань. Існує конвертор табличних документів 1С:Предприятия для *Microsoft Excel*.

CalliGraph. Самостійний генератор звітів зі своїм дизайнером і друком. Можливий експорт звіту в *Excel*. Підтримується робота з *ODBC*-базами даних через *ADO*. Можливе створення аналітичних звітів з подальшою деталізацією по колонках. Отриманий звіт можна експортувати в *Microsoft Excel*.

ГЗ існують в рамках достатньо потужних СУБД і поставляються разом з ними, але головним недоліком таких систем є максимальна залежність від дій користувача по створенню звіту, хоча слід зазначити, що можливості різного роду “ майстрів” зменшують цей показник.

1.6.3 Генератори на основі шаблонів

Apache Velocity – процесор шаблонів базується на *Java*, який забезпечує просту, але потужну шаблону мову, що не потребує попередньої підготовки моделі змінних для шаблону - в шаблон просто передаються посилання на *Java*-об'єкти, а обробник розбирає зазначені методи і за допомогою *Java reflection* отримує їх значення. Його мета полягає в тому, щоб гарантувати чисте поділ між рівнем представлення та бізнес рівнем в *WEB*-додатку.

Приклад використання шаблону і *Java* додатка. Шаблон - *template.vm*:

```
<HTML>
    <body>
      Відомость групи $group
    </body>
</HTML>
```

Java-код, що зв'язує змінну *"name"* в коді і змінну *"$ group"* в шаблоні:

```
public class First {
    VelocityContext vc = new VelocityContext(); // створення контексту Velocity
    String name = "Velocity";
    vc.put("group", name); // атрибут "name" отримує зв'язок зі змінною $group у
    шаблоні template.merge(vc, bw); }
```

FreeMarker. Компілюючий обробник шаблонів. Інструмент, що дозволяє відокремити бізнес-логіку і дані від подання в дусі концепції *Model-view-controller*. Використовується переважно при розробці web-додатків з використанням *Java*-сервлетів, також може використовуватися для виводу тексту: генерація *CSS*, вихідного коду *Java*.

JavaServer Pages (JSP) – технологія, що дозволяє *WEB*-розробникам створювати вміст, який має як статичні, так і динамічні компоненти. *JSP* є текстовим документом, що містить текст двох типів: статичні вихідні дані, оформлені в одному з форматів *HTML*, *SVG*, *WML*, *XML* і *JSP* елементи, що конструюють динамічний вміст. *JSP* – високопродуктивна технологія, що транслює код сторінки в *Java*-код сервлету за допомогою компілятора - *Jasper*, і потім компілюється в байт-код віртуальної машини *Java* (*JVM*).

WebMacro призначений для розробки *Java* сервлетів. *WebMacro* реалізує шаблон проектування типу *Model-View-Controller*, забезпечуючи відділення вихідного коду проекту від *HTML* коду представлення. *WebMacro* використовується також для генерації довільного текстового виводу на основі шаблону.

1.6.4 Підходи до автоматизації створення User Interface

Unified Modeling Language (*UML*) – уніфікована мова моделювання, розроблена Граді Бучем, Джимом Рембо та Іваром Якобсоном. *UML* – графічна мова, що дозволяє архітектору ПЗ візуалізувати, документувати, специфікувати та конструювати проекти ПЗ. Мова *UML* є формальною мовою специфікацій і відрізняється тим самим від синтаксису традиційних формально-логічних мов і мов програмування. Використання *UML* пов'язано з послідовністю ведення проектних робіт. Спочатку проект алгоритму описується на мові *UML*. Після цього алгоритм вручну переписується на мові програмування. Отриманий результат компілюється в машинний код і піддається тестуванню. Застосовуючи *CASE*-засоби, архітектор отримує можливість істотно знижувати рівень ручного програмування при формуванні макетів проектних рішень. Природний розвиток

рівня автоматизації процесу розробки проектів ПЗ, а також розвиток мови *UML* призвели до появи ідей *Model Driven Architecture* (*MDA*) – “ Архітектура на базі моделей”.

MDA архітектура

Концепцією *MDA* є опис алгоритмів на мові моделювання з наступним автоматичним перетворенням моделей в комп'ютерний код. Програмування на базі моделей припускає, що проектувальники ПЗ передусім створюють найбільш підходящу модель, не “ прив'язуючись” до платформи, на якій система буде реалізована.

Важливою перевагою *MDA* є те, що при розробці програм основні зусилля розробників ПЗ переносяться з етапу програмування на етап створення моделі ПЗ. Крім того, створивши модель один раз, розробники отримують принципову можливість генерації програм для різних апаратних і програмних платформ. Концентрація в моделі всієї логіки додатка і автоматична генерація коду та БД одне з найважливіших переваг *MDA*.

Підхід на основі онтологій заснований на наступних принципах:

- об'єднати однорідну за змістом інформацію у компоненти моделі інтерфейсу;
- звільнити розробника інтерфейсу від кодування, надавши йому візуальні редактори, що приховують синтаксис мов програмування;
- формувати інформацію для кожного компонента моделі інтерфейсу за допомогою редакторів, керованих відповідними моделями онтологій;
- автоматично генерувати код інтерфейсу за його моделлю;
- підтримувати різні типи взаємодії інтерфейсу з користувачем;
- забезпечити розробника інструментарієм, що розширюється.

Однорідна за змістом інформація, згідно даної концепції, об'єднується в компоненти моделі інтерфейсу. При цьому модель інтерфейсу повинна містити всю інформацію про цей призначений для користувача інтерфейс, яка може

піддатися зміні в його життєвому циклі, а також допускати автоматичну побудову призначеного для користувача інтерфейсу по його моделі.

1.7 Концептуальні поняття та визначення

Слабоструктуровані дані (*semi-structured*) – згідно визначення це форма структурованих даних, що не відповідає строгої структурі таблиць і відношень в моделях реляційних БД. Однак така форма даних вміщує теги та інші маркери для відокремлення семантичних елементів та для забезпечення ієрархічної структури записів та атрибутів в наборі даних. Таким чином, такий вид даних можна назвати безсхемним (*schemaless*), а структуру – самоописуваною.

В слабоструктурованих даних сутності, що належать одному і тому самому класу, можуть мати різні атрибути, навіть якщо класи належать до однієї і тієї ж самої групи. Порядок атрибутів не має значення.

Слабоструктуровані ЕД – ЕД, сформовані *semi-structured* даними, в яких семантична складова відокремлюється засобами графічної розмітки або об'єднанням певних елементів ЕД, що представляють одну інформаційну область, формуючи структурну залежність одного елемента від іншого.

Виділяють наступні *ознаки та властивості слабоструктурованих ЕД*: ЕД має внутрішню розмітку; вміст ЕД розбито внутрішнім форматуванням на окремі текстові фрагменти; кожен фрагмент об'єктивно являє собою або значення даних, або атрибут даних; внутрішня розмітка не має формальних ознак, що вказують на те, що є значенням, а що є атрибутом даних.

Для такого роду ЕД найбільш ефективним способом представлення є сукупність об'єктів з послідуючою можливістю ідентифікацією фрагментів ЕД як атрибутів та значень даних. Прикладом слабоструктурованих ЕД є всі ЕД які зберігаються у *xml, html* – подібні формати, ЕД форматів офісних систем типу *MS Office, OOffice, Libre*. Такі ЕД можуть мати форми анкет, відомостей, фінансової звітності, рапортів, повідомлень тощо.

Документно-орієнтована інформаційна технологія – сукупність програмних засобів та технологій, що поєднані в одну систему кругообігу даними

між учасниками інформаційного процесу, ключовими ланками яких є база даних та електронний документ. Також виділяють таке поняття як документно-орієнтована БД, що призначена для документно-орієнтованих додатків, таких як системи управління ЕД – *Data Management System* або корпоративних інформаційних систем організації. Ключовою ланкою документно-орієнтованих БД є ЕД. Відносини описуються двома полями БД, що в ЕД представлені певним одним значенням.

Документно-орієнтована екранна форма – програмним модуль, що імітує образ та роботу електронного докусменту, на основі якого відтворюється відображення ЕФ.

Детермінованість – під детермінованістю розуміють однозначний взаємозв'язок причини та наслідку. Детермінованість при вирішені будь-якої практичної задачі або у алгоритмі означає, що спосіб вирішення задачі визначено однозначно у вигляді послідовності кроків. На будь-якому кроці не допускається ніякі двозначності або невизначеності.

В загальному вигляді залежність майбутнього стану $x(t)$ від початкового $x(t_0)$ можна записати у вигляді $x(t)=F[x(t_0)]$, де F – детермінований закон, що визначає строге однозначне перетворення початкового стану $x(t_0)$ в майбутній стан $x(t)$ для будь-якого $t>t_0$.

Таблична структура даних ЕД – форма представлення даних, що впорядковує інформацію у вигляді довільно організованих рядків (по горизонталі) і довільно організованих стовпців (по вертикалі), де кожен фрагмент, що знаходиться на перетині рядку-стовпця (комірка) може являти або значення даних, або атрибут даних.

Структурна складність електронного документу (СС) – кількість інформаційних областей (ІО), отриманих засобами графічного оформлення або об'єднанням певних елементів електронного документу, що встановлюють відповідність з елементами таблиць БД КІС та дозволяють налагодити однозначні інформаційні потоки між ними.

1.8 Висновки до розділу

В розділі виконано аналіз способів представлення інформації в ЕД, що надав змогу видокремити ефективне поєднання форми її представлення та цільове призначення такої форми. Таблична форма представлення спрямована на швидкий і точний аналіз даних. Проаналізовані методи кластеризації та класифікації вирішують задачу систематизації ЕД з різною алгоритмічною складністю та відносно низьким рівнем похибок класифікації, але вони засновані на частотних характеристиках виразів в тексті і знаходженні відстані між ними, а метрики близькості не враховують пріоритетність структурних складових ЕД табличної структури. Проаналізовані підходи автоматизації створення інтерфейсів доступу до даних БД не надають змоги автоматичного встановлення відповідних зв’язків з даними ЕД.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ЕЛЕКТРОНИХ ДОКУМЕНТІВ ТАБЛИЧНОЇ СТРУКТУРИ

2.1 Об’єктна метамодель електронного документу табличної структури

Для забезпечення сумісності між різними платформами і мовами консорціумом *W3C* запропонував представляти ЕД у вигляді сукупності об'єктів – *DOM* (*Document Object Model).* Даний підхід дозволяє різним програмам отримувати доступ до вмісту ЕД, а також змінювати їх вміст, структуру і оформлення.

Модель *DOM* не накладає обмежень на структуру ЕД, тому будь-який ЕД відомої структури за допомогою *DOM* може бути представлений у вигляді дерева вузлів, кожен вузол якого є елементом або атрибутом, текстовий або графічний. Вузли зв'язані між собою стосунками “ батьківський-дочірній”.

Приклад дерева вузлів представлений на рисунку 2.1.

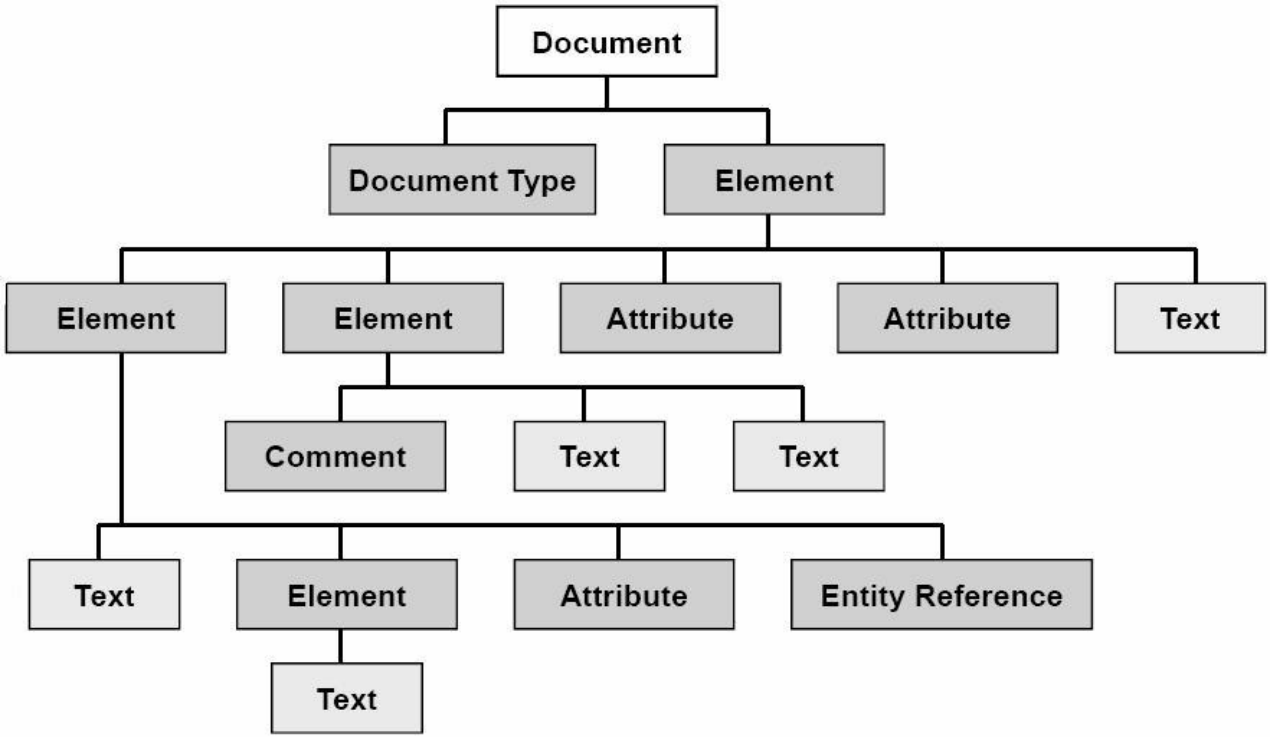


Рис. 2.1. *DOM*-дерево ЕД

Для представлення ЕД формату *XLS* з табличною структурою потрібно модифікувати *DOM*-модель у відповідності до формату, що враховує особливості оформлення табличних ЕД, пов'язаних з візуальним оформленням комірок засобами об'єднання або графічними примітивами. Модифіковану об'єктну модель ЕД (*ОМЕД*) *XLS* формату з табличною структурою представлено у вигляді двійки:

$$МОМД = head \cup body = \{n_i\},$$ де

head – заголовна частина ЕД;

body – змістовна частина ЕД;

n_i – об'єкт *ОМЕД*.

Під об'єктом розуміють інформаційну область (ІО), яка в початковому ЕД має вигляд об'єднаних комірок або вигляд комірок, границі яких визначаються наявністю ліній розмітки. Кожен об'єкт *ОМЕД* характеризується кортежем характеристик:

$$<pn, n, v, ix, l, m>,$$ де

pn – номер вузла батька;

n – номер вузла;

v – значення вузла;

ix – індекс значення вузлу *v* у словнику предметної області (СПО);

l – рівень схожості значення вузла *v* з еталоном в СПО;

$m \in \{0, 1, 2, 3\}$ – мітка приналежності вузлу *v* до визначеного типу (0, 1, 2, – статичний, крітеріальний, динамічний, невизначений тип, відповідно).

СПО – список записів, що вміщують всі поняття визначеної предметної області, оперуючи якими користувач отримує можливість отримати інформацію з інформаційної системи. СПО має вигляд множини значень, кожне з яких характеризує один запис предметної області –

$$Voc = \{voc_i\},$$ де $voc_i = <w, mw, intr, ix, dbr>,$ де

w – еталонна лексема для порівняння;

mw – можливі написання лексеми;

intr – інтерпретація лексеми;

ix – індекс лексеми у словнику;

dbr – зв’язок лексеми з таблицями у БД.

dbr =<*dbs, dbl*> – словник БД (СБД), множина записів, що описують зв'язок таблиць БД ІС. *dbs* = {<*r, a, t*>} – множина кортежів, які ставлять у відповідність вузол n_i таблицю *r* і атрибут таблиці *a*, *t* ∈{*vh, int, dt*} – тип атрибуту: рядковий, цілочисельний та дата відповідно; *dbl* = {<*t1, t2, a1, a2*>} – множина кортежів, що визначають зв’язок атрибута *a1* таблиці *t1* з атрибутом *a2* таблиці *t2*.

На рисунку 2.2 представлено дерево модифікованої об’єктної моделі ЕД.

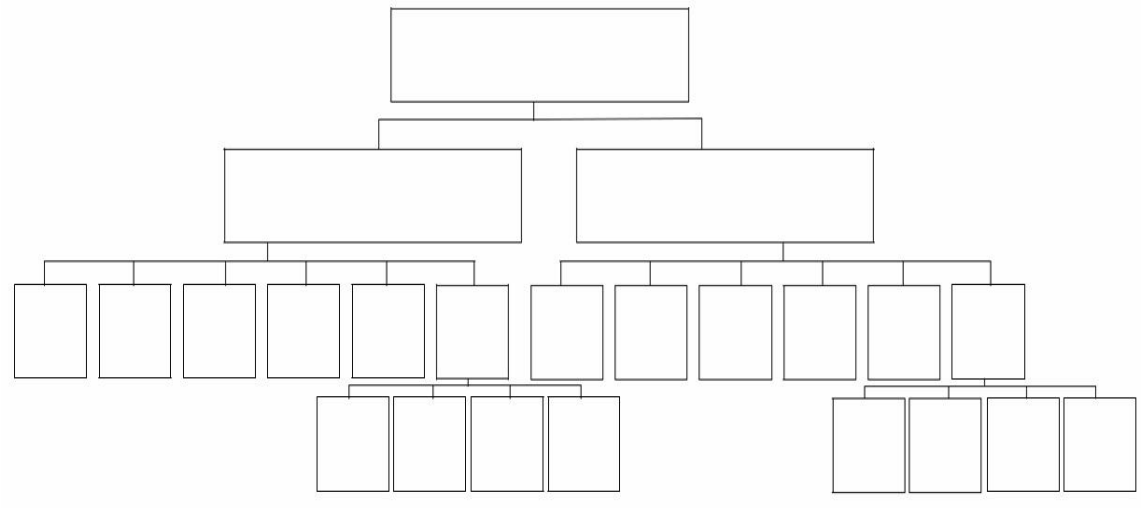
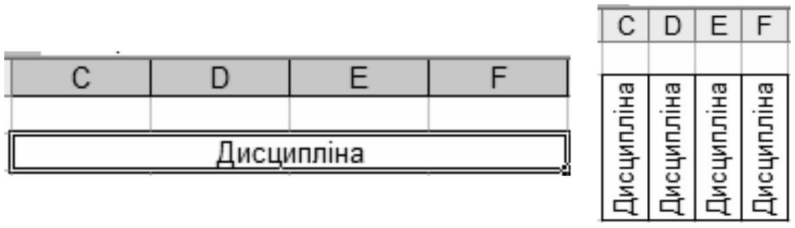


Рис. 2.2. *ОМЕД* дерево ЕД

2.2 Програмна модель доступу до електронного документу *XLS* формату

Як відомо, для виразного представлення структури ЕД оператори використовують механізм об’єднання комірок (рисунок 2.3, а), але згідно специфікації *Excel* значення комірок дублюється у кожну об’єднану комірку (рисунок 2.3, б), також кожен файл формату *XLS* розбитий на аркуші, що можуть виконувати окремі функції.



а) б)

Рис. 2.3. Внутрішнє представлення ЕД *XLS* формату

Враховуючі такі особливості, запропоновано модель доступу до ЕД формату *XLS*, що складається з множини об’єктів комірок $XLS=\{CL_i\}$, кожний елемент CL_i множини, описаний кортежем характеристик:

$<NS, NO, C, R, V, P>$, де

NS (number of sheet) – номер аркуша в ЕД;

NO (number of object) – номер комірки в ЕД;

C (column) – номер стовпця комірки;

R (row) – номер рядку комірки;

V (value) – значення комірки з номером стовпця *C* та номером рядку *R;*

P (property) – властивість комірки у рамках наявності лінії розмітки ліворуч, праворуч, зверху, знизу. *P* представлено як четвірка:

$P=<L, R, U, B>$, де

$L, R, U, B \in \{0, 1\}$ – наявність (1) або відсутність (0) лінії розмітки ліворуч, праворуч, зверху, знизу комірки CL_i, відповідно.

Алгоритм побудови *XLS* моделі аналізує кожну *i-my* комірку вихідного ЕД, визначає номер її стовпця C_i та рядку R_i, визначає значення комірки V_i та властивості P_i. Якщо ІО створено, як об’єднання сусідніх комірок, то призначити кожній комірці цього об’єднання один і той самий номер NO_i.

2.3 Програмна модель доступу до електронних документів *DOC* формату

Модель доступу до ЕД формату *DOC* формату представлена у вигляді впорядкованої множини параграфів $DOC=\{P_i\}$, кожний елемент P_i множини, описаний кортежем характеристик:

$<n, ts, tt, pe, pt>$, де

n – порядковий номер параграфу;

ts – дозвільний рядок;

$\in \{1, 0\}$ – тип тексту, де 1 - кінець комірки у таблиці, 0 – інакше;

$pe \in \{1, 2, 3, 4\}$ – тип параграфу, де 1,2,3,4 – параграф таблиці, списку, тексту, об’єкту, відповідно;

$pt \in \{1, 0\}$, де 1 – параграф є кінцем рядку, 0 – інакше.

На рисунку 2.4, представлено приклад електронного документу (ЕД) формату *DOC* з табличною структурою у вигляді множини впорядкованих параграфів.

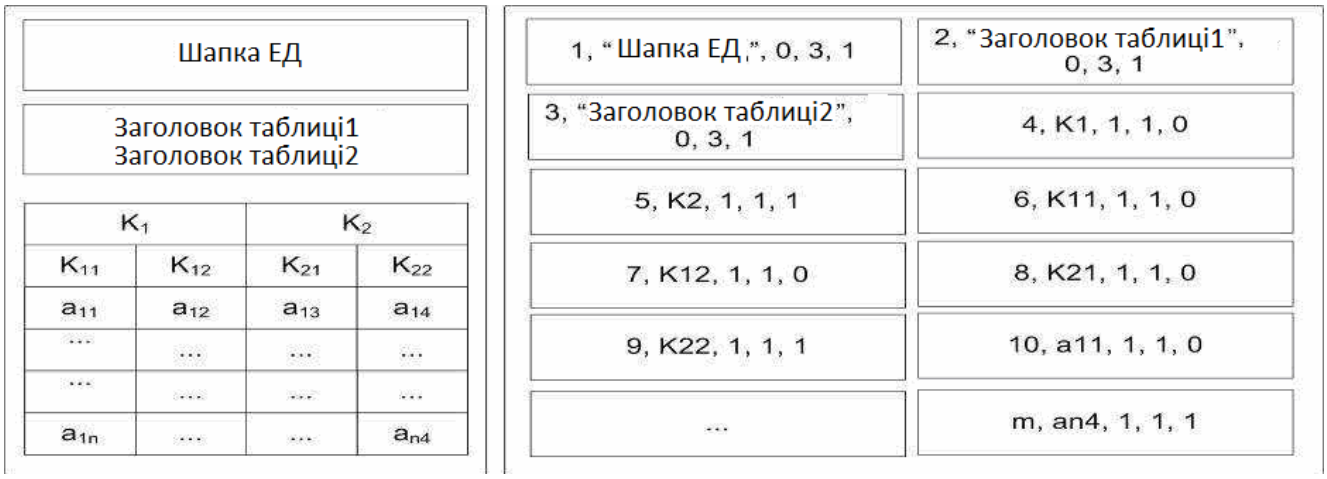


Рис. 2.4. Модель ЕД *DOC* формату у вигляді параграфів

2.4 Алгоритми та моделі створення об’єктної метамоделі електронного документу

2.4.1 Алгоритм пошуку прямокутних областей в ЕД XLS-формату

Складність автоматичного аналізу даних з табличною організацією обумовлена в першу чергу великою різноманітністю форм зображення таблиць. ЕД з однаковим змістом можуть мати різні структури, що значно ускладнює процедуру структурно-синтаксичної класифікації ЕД. Відомі методи і технології обробки табличних даних орієнтовані на заздалегідь певні структури та особливості таблиць, які пов'язані зі стандартами обраної ПрОбл. На рисунку 2.5 запропоновано фрагмент ЕД *XLS*-формату, в якому візуально можна підрахувати одинадцять елементів.

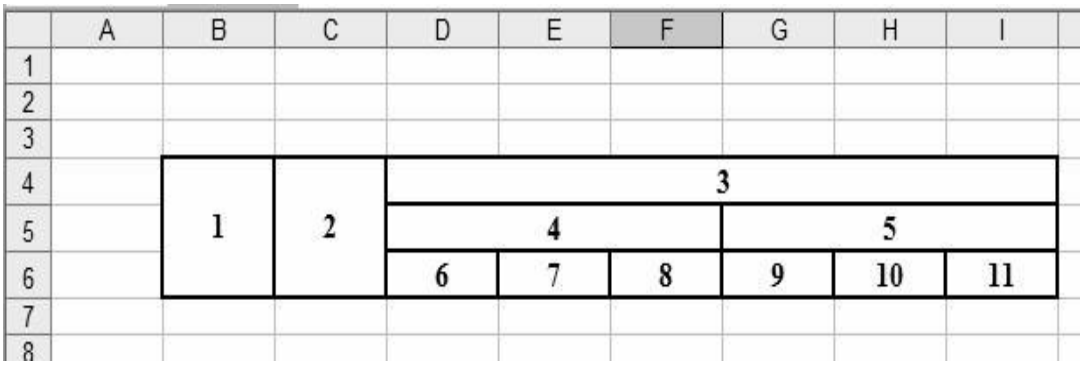
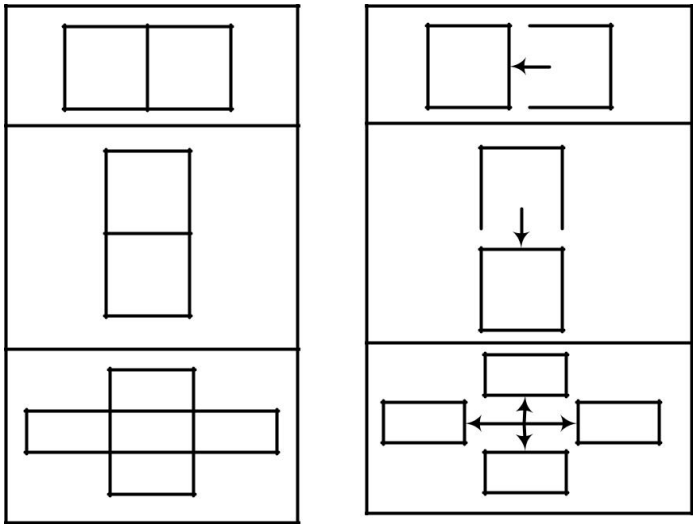


Рис. 2.5. Прямокутні області в ЕД

Однак це візуальне сприйняття прямокутних елементів. Внутрішній аналіз електронних документів з використанням програмної бібліотеки *Java Apache POI*, а також аналіз *XML*-представлення ЕД, отриманого засобами офісних пакетів, показав, що комірки, які мають суміжні кордони (рисунок 2.6, а), не завжди мають вигляд ІО (рисунок 2.6, б) і можуть утворюватися як об'єднання сусідніх комірок.

Враховуючи усі описані обмеження і особливості, потрібно застосовувати наступний алгоритм пошуку ІО в ЕД *XLS*-формату, вхідною множиною якого є множина *XLS*, і який складається з наступної послідовності кроків.



а) б)

Рис. 2.6. Внутрішнє представлення комірок у ЕД

Створення множини значень ІО – *Area={rect$_i$}*, кожен елемент якої представлено четвіркою:

<no, x1y1, x2y2, v>, де

no – номер знайденої комірки CL_i, що відповідає значенню *CL.NO*;

x1y1 та *x2y2* – значення *C* та *R* лівої верхньої комірки CL_i та правої нижньої комірки CL_i, що утворює ІО, відповідно;

v – значення в CL_i, що відповідає значенню *CL.V*.

Пошук ІО, які утворені об’єднанням комірок, але без лінії розмітки, тобто ті комірки, які мають єдиний номер *NO* та додавання їх до множини *Area*:

$$Area.add(Find(CL_i.NO = CL_{i+1}.NO)).$$

Пошук ІО з усіма лініями розмітки.

Пошук ІО, де відсутня одна лінія розмітки.

Пошук ІО, де відсутні дві лінії розмітки.

Пошук ІО, де відсутні три лінії розмітки.

Пошук ІО з усіма лініями розмітки.

Виділення всіх ІО, де комірка має всі лінії розмітки (рисунок 2.7), тобто $CL_i.P.R=CL_i.P.L=CL_i.P.U=CL_i.P.B=1$.

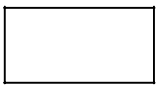


Рис. 2.7. Комірка з усіма лініями розмітки

Додавання у множину *Area* координат лівого верхнього кута *x1y1*, правого нижнього кута *x2y2* і значення в комірці *v*, відповідно. Виключення знайденої ІО зі списку комірок, що аналізуються, але не виключення зі списку ІО, за допомогою яких може бути створена інша ІО.

Пошук ІО, де відсутня одна лінія розмітки:

$$Area.add(Find(CL_i \in R \wedge U \wedge B) \vee (CL_i \in L \wedge U \wedge B) \vee (CL_i \in L \wedge R \wedge B) \vee (CL_i \in L \wedge R \wedge U))$$

Визначення комірок, де відсутня одна з ліній розмітки, або *R*, або *L*, або *U*, або *B*, рисунок 2.8.

Рис. 2.8. Комірка з трьома лініями розмітки

Пошук сусідньої комірки з відповідними координатами:

для 1-ого випадку – *x1+1*, *y1*, *x2+1*, *y2*;

для 2-ого випадку – *x1-1*, *y1*, *x2-1*, *y2*;

для 3-ого випадку – *x1*, *y1-1*, *x2*, *y2-1*;

для 4-ого випадку – *x1*, *y1+1*, *x2*, *y2+1*.

В процесі пошуку перевіряються наступні умови:

Умова 1. Якщо *CL.P.L=1*, *CL.P.R=1*, *CL.P.B=1*, *CL.P.U=1* сусідньої комірка для 1-ого, 2-ого, 3-ого, 4-ого випадку, відповідно, то додати знайдену комірку CL_i, як ІО до множини *Area*, інакше виконати умову 2.

Умова 2. Якщо відповідні властивості сусідньої комірки:

–для 1-ого випадку дорівнюють *CL.P.R=CL.P.B=CL.P.U=1;*

– для 2-ого випадку – *CL.P.L=CL.P.B=CL.P.U=1;*

– для 3-ого випадку – *CL.P.L=CL.P.R=CL.P.U=1*;

– для 4-ого випадку – *CL.P.L=CL.P.R=CL.P.B=1*, то додати координати лівого верхнього та правого нижнього куту, значення та номер отриманої ІО до множини *Area*, інакше виконати умову 3.

Умова 3. Якщо відповідні властивості сусідньої комірки:

– для 1-ого та 2-ого випадку дорівнюють *CL.P.B=CL.P.U=1*;

– для 3-ого та 4-ого *CL.P.L=CL.P.R=1*, то додаємо її як складову частину ІО і переходимо до аналізу наступної сусідньої комірки, починаючи з умови 1, інакше виконуємо умову 4.

Умова 4. Якщо відповідні властивості сусідньої комірки:

– для 1-ого та 2-ого випадку дорівнюють *CL.P.B=1*;

– для 3-ого та 4-ого *CL.P.R=1*, то аналізуємо для поточної комірки комірку з координатами для 1-ого і 2-ого випадку – *x1*, *y1-1*, *x2*, *y2-1*, для 3-ого і 4-ого – *x1+1*, *y1*, *x2+1*, *y2*, якщо для 1-ого і 2-ого випадку комірка, що аналізується має лінію розмітки – *B*, для 3-ого і 4-ого – *R*, то додаємо її як складову частину ІО і виконуємо умову 1, інакше область, що утворена сусідніми комірками не є прямокутником і виконуємо умову 5.

Умова 5. Якщо відповідні властивості сусідньої комірки:

– для 1-ого та 2-ого випадку дорівнюють *CL.P.U=1*;

– для 3-ого та 4-ого *CL.P.L=1*, то аналізуємо для поточної комірки клітинку з координатами для 1-ого і 2-ого випадку – *x1*, *y1+1*, *x2*, *y2+1*, для 3-ого і 4-ого – *x1-1*, *y1*, *x2-1*, *y2*, якщо для 1-ого і 2-ого випадку комірка, що

аналізується має лінію розмітки – *U*, для 3-ого і 4-ого – *L*, то додаємо її як складову частину ІО і виконуємо умову 1, інакше область, що утворена сусідніми комірками не є прямокутником і виконуємо умову 6.

Умова 6. Якщо *CL.P.R=1*, *CL.P.L=1*, *CL.P.U=1*, *CL.P.B=1* сусідньої комірки для 1-ого, 2-ого, 3-ого, 4-ого випадку, відповідно, то аналізуємо комірку з координатами щодо поточної для 1-ого і 2-ого випадку – *x1*, *y1-1*, *x2*, *y2-1*, для 2-ого та 3-ого – *x1-1*, *y1*, *x2-1*, *y2*, якщо у комірці, що аналізується для 1-ого і 2-ого випадку є лінія розмітки – *B*, для 3-ого і 4-ого – *R*, то виконуємо умову 5, інакше область, що утворена сусідніми комірками не є прямокутником і виконуємо умову 7.

Умова 7. Якщо сусідня комірка для 1-ого, 2-ого, 3-ого і 4-ого випадку не має ліній розмітки, то аналізуємо комірку щодо поточної з координатами для 1-ого випадку – *x1+1*, *y1*, *x2+1*, *y2*, для 2-ого випадку – *x1-1*, *y1*, *x2-1*, *y2*, для 3-ого випадку – *x1*, *y1-1*, *x2*, *y2-1*, для 4-ого випадку – *x1*, *y1+1*, *x2*, *y2+1*, якщо у комірці, що аналізується для 1-ого випадку є лінія розмітки – *L*, для 2-ого – *R*, для 3-ого – *B*, для 4-ого – *U*, то виконати умову 6.

Пошук ІО, де відсутні дві лінії розмітки:

$$Area.add(Find(CL_i \in L \wedge R) \vee CL_i \in L \wedge U) \vee CL_i \in L \wedge B) \vee CL_i \in R \wedge U) \vee$$

$$\vee CL_i \in R \wedge B) \vee CL_i \in U \wedge B))$$

Визначення комірок, де відсутні дві лінії розмітки, або *R* і *U*, або *L* і *U*, або *L* і *B*, або *R* і *B*, або *L* і *R*, або *B* і *U*, рисунок 2.9.

Рис. 2.9. Комірка з двома лініями розмітки

Пошук сусідньої комірки з координатами:

для 1-ого, 2-ого і 6-ого випадку – *x1, y1-1, x2, y2-1*;

для 3-ого і 4-ого випадку – *x1, y1+1, x2, y2+1*;

для 5-ого випадку – *x1-1 , y1, x2-1, y2*.

В процесі пошуку перевіряються наступні умови:

Умова 1 Якщо *CL.P.B=1*, *CL.P.U=1*, *CL.P.R=1* сусідньої комірки для 1,2,6-ого, 3,4-ого, 5-ого випадку, відповідно, то додаємо її як складову частину ІО і виконуємо умову 2. Інакше, якщо *CL.P.U=CL.P.B=1*, *CL.P.L=CL.P.R=1* сусідньої комірки для 5-ого або для 6-ого випадку, відповідно, то додаємо знайдені комірки до ІО виконуємо умову 3. Інакше для поточної комірки аналізуємо комірку з координатами для 5-ого випадку – *x1, y1-1, x2, y2-1* і комірку – *x1, y1+1, x2, y2+1*, якщо у 1-ої комірки є лінія розмітки – *B* і у 2-ої – *U*, то додаємо знайдені комірки до ІО, для 6-ого – *x1-1, y1, x2-1, y2* і комірку – *x1+1, y1, x2+1, y2*, якщо у 1-ої комірки є лінія розмітки – *R* і у 2-ої – *L*, то додаємо знайдені комірки до ІО і виконуємо умову 3.

Умова 2 Пошук сусідньої комірки з координатами для 1-ого, 4-ого і 5-ого випадку – *x1+1, y1, x2+1, y2*, для 2-ого та 3-ого випадку – *x1-1, y1, x2-1, y2*, для 6-ого випадку – *x1 , y1 +1, x2, y2 +1*.

Умова 3 Якщо у знайдених сусідніх комірках є лінія розмітки для 1-ого, 4-ого і 5-ого випадку – *L*, для 2-ого та 3-ого – *R*, для 6-ого – *U*, то додаємо знайдену комірку до ІО і виконуємо умову 2.

Пошук ІО, де відсутні три лінії розмітки:

$$Area.add(Find(CL_i \in L) \vee CL_i \in R) \vee CL_i \in B) \vee CL_i \in U)).$$

Визначення комірок, де присутня тільки одна з ліній розмітки або *R*, або *L*, або *B*, або *U*, рисунок 2.10.

Рис. 2.10. Комірка з однією лінією розмітки

Пошук сусідніх комірок з координатами:

для 1-ого випадку:

x1, y1-1, x2, y2-1, або *x1, y1+1, x2, y2+1*, або *x1+1, y1, x2+1, y2*;

для 2-ого випадку:

x1, y1-1, x2, y2-1, або *x1, y1+1, x2, y2+1*, або *x1-1, y1, x2-1, y2*;

для 3-ого випадку:

x1, y1+1, x2, y2+1, або *x1-1, y1, x2-1, y2*, або *x1+1, y1, x2+1, y2*;

для 4-ого випадку:

x1, y1-1, x2, y2-1, або *x1-1, y1, x2-1, y2*, або *x1+1, y1, x2+1, y2*.

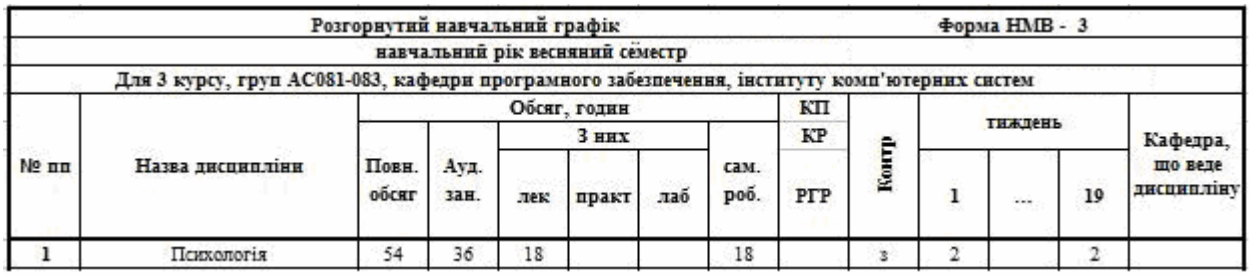
Якщо для 1-ого випадку у 1-й знайденій комірці є лінія розмітки – *B*, або у 2-й – *U*, або у 3-й – *L*, то виконати умову 3 з пункту “ *Пошук ІО, де відсутні дві лінії розмітки”*.

Якщо для 2-ого випадку у 1-й знайденій комірці є лінія розмітки – *B*, або у 2-й – *U*, або у 3-й – *R*, то виконати умову 3 з пункту “ *Пошук ІО, де відсутні дві лінії розмітки”*.

Якщо для 3-ого випадку у 1-й знайденій комірці є лінія розмітки – *U*, або у 2-й – *R*, або у 3-й – *L*, то виконати умову 3 з пункту “ *Пошук ІО, де відсутні дві лінії розмітки”*.

Якщо для 4-ого випадку у 1-й знайденій комірці є лінія розмітки – *B*, або у 2-й – *R*, або у 3-й – *L*, то виконати умову 3 з пункту “ *Пошук ІО, де відсутні дві лінії розмітки”* .

На рисунку 2.11 представлено фрагмент ЕД, на основі якого було побудована модель представлена в п. 2.2.



Розгорнутий навчальний графік										Форма НМВ - 3			
навчальний рік весняний сёместр													
Для 3 курсу, груп АС081-083, кафедри програмного забезпечення, інституту комп'ютерних систем													
№ пп	Назва дисципліни	Обсяг, годин						КП КР РГР	Контр	тиждень			Кафедра, що веде дисципліну
		Повн. обсяг	Ауд. зан.	з них лек	з них практ	з них лаб	сам. роб.			1	...	19	
1	Психологія	54	36	18			18		з	2		2	

Рис. 2.11. Фрагмент ЕД с табличною структурою “ Розгорнутий навчальний графік”

За допомогою програмної реалізації описаного алгоритму побудована множина ІО – *Area*, результати роботи представлені в таблиці 2.1.

Таблиця 2.1.

Множина прямокутних елементів ЕД “ Розгорнутий навчальний графік”

n	***x1:y1***	***x2:y2***	***v***
1	4:3	4:32	Розгорнутий навчальний графік Форма НМВ - 3
2	5:3	5:32	навчальний рік весняний семестр
3	6:3	6:32	Для 3 курсу, груп АС081-083, кафедри ПЗ, ІКС
4	7:3	7:3	№ пп
5	7:4	7:4	Назва дисципліни
6	7:5	7:5	Обсяг, годин
7	7:11	9:11	КП КР РГР
8	7:12	7:12	Контр
9	7:13	7:13	тиждень
10	7:32	7:32	Кафедра, що веде дисципліну

11	8:5	9:5	Повн. обсяг
12	8:6	8:6	Ауд. зан.
13	8:7	8:7	З них
14	8:10	8:10	сам. роб.

2.4.2. Приклад визначення індексів та рівня схожості лексем ОМЕД-моделі

Для визначення значень показників рівня схожості та індексу лексем *ОМЕД*-моделі розроблено 3 функції: *FuzzyMATCH*, *GetIndex*, *MaxFuzzy*.

Функція *FuzzyMATCH* використовує методику нечіткого порівняння двох лексем [1], та повертає коефіцієнт співпадіння в діапазоні від 0 до 1, де 0 – лексеми повністю не співпадають, 1- лексеми співпадають на 100%.

Функція *MaxFuzzy* повертає максимальний коефіцієнт співпадіння двох лексем. З метою визначення максимального коефіцієнту поточна лексема порівнюється з усіма лексемами в словнику предметної області. Визначення максимального коефіцієнту повинно задовольняти рівнянню 2.1.

$$\forall ОМЕД.v \; \exists ОМЕД.l = MaxFuzzy\left(ОМЕД.v_i, СПО.w_j\right) \tag{2.1}$$

Функція *GetIndex* – повертає значення індексу лексеми з СПО рівень схожості якої максимальний з лексемою з *ОМЕД*-моделі ЕД (рівняння 2.2).

$$\forall ОМЕД.v \; \exists \; ОМЕД.ix = GetIndex\left(MaxFuzzy\left(ОМЕД.v_i, СПО.w_j\right)\right) \tag{2.2}$$

На рисунку 2.12 зображено приклад ЕД, за яким визначались показники індексів та коефіцієнтів схожості *ОМЕД*-моделі ЕД. В таблицю 2.2 зведені загальні показники сформованої моделі.

ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ

Навчальний рік		Семестр	
Напрям підготовки	6050103	Курс	3
Спеціальність		Группа	АС091

Відомість обліку успішності № 1282

Дисципліна Конструювання Програмного Забезпечення ІСП

Підсумкову оцінку виставив n/d

Поточний контроль здійснив n/d

№ П/П	Прізвище, ініціали студента	№ інд. навч.	1 модуль	1 перескл.	Дата	2 перескл.	Дата	2 модуль
286590	АНАНЧЕНКО Н.О.		20	32		0		0
214529	АНДРЄЄВ П.П.							
214358	АРНАУТОВ О.В.							

Шкала оцінювання

Стобальна	Чотири бальна	ECTS	Назва
95-100	5	A	відмінно
85-94	4	B	дуже добре
75-84	4	C	добре
65-74	3	D	задовільно
60-64	3	E	достатньо
30-59	2	FX	не задовільно
< 30	2	F	не задовільно

Дата

Директор (Декан) _____

Рис. 2.12. Фрагмент ЕД “ Модульна відомість”

Таблиця 2.2.

Значення показників *ОМЕД*-моделі ЕД

ОМЕД.v	*ОМЕД.n*	*ОМЕД.pn*	*ОМЕД.ix*	*ОМЕД.l*
інститут комп'ютерних систем	4	0	22	1,00
навчальний рік	5	4	32	1,00
семестр	7	0	51	1,00
напрям підготовки	8	5	35	1,00
6050103	9	6	1	0,04
курс	10	7	29	1,00
група	13	10	12	1,00
ас091	14	11	52	0,13
відомість обліку успішності №	15	0	37	0,80
1282	16	0	3	0,10
дисципліна	17	0	15	1,00
конструювання програмного забезпечення	18	15	6	0,23
ісп	19	0	23	1,00
підсумкову оцінку виставив	20	17	43	1,00

поточний контроль здійснив	22	20	45	1,00
дата	24	0	13	1,00
директор (декан)	25	24	14	1,00
№ п/п	27	0	7	0,56
прізвище, ініціали студента	28	0	47	1,00
№ інд. навч. плану	29	0	6	1,00
1 модуль	30	0	1	1,00
1 перескл.	31	0	4	0,88
дата	32	0	13	1,00
2 перескл.	33	0	4	1,00
дата	34	0	13	1,00
2 модуль	35	0	3	1,00
стобальна	63	0	54	1,00
чотири бальна	64	0	58	1,00
ects	65	0	5	1,00
назва	66	0	33	1,00
95-100 85-94 75-84 65-74 60-64 30-59 < 30	68	63	57	0,09
5443322	69	64	3	0,16
a b c d e fx f	70	67	56	0,11
відмінно дуже добре добре задовільно достатньо не задовільно не задовільно	71	66	36	0,60

2.5 Висновки до розділу

Рішення задачі кластеризації принципове неоднозначно, і тому є декілька причин. По-перше, не існує однозначно найкращого критерію якості кластеризації. Відомий цілий ряд досить розумних критеріїв, а також ряд алгоритмів, що не мають чітко вираженого критерію, але що здійснюють досить розумну кластеризацію “ по побудові”. Всі вони можуть давати різні результати. По-друге, число кластерів, як правило, невідоме заздалегідь і встановлюється відповідно до деякого суб'єктивного критерію. По-третє, результат кластеризації істотно залежить від метрики, вибір якої, як правило, також суб'єктивний і визначається експертом.

РОЗДІЛ 3
МЕТОДИ РОБОТИ З БАЗОЮ ДАНИХ НА ОСНОВІ ШАБЛОНІВ ЕЛЕКТРОННИХ ДОКУМЕНТІВ

3.1 Документно-орієнтований *XML*-шаблон електронного документу

Поняття шаблону ЕД складається з двох частин:

- *XML* частина шаблону ЕД (рисунок 3.1), що описується множиною тегів

$$varDecl=\{vd_i\},$$

де

$vd_i=<N, dT, idObj, NameObj, TypeObj, idR, Sql, FillNumber, DeleteCol>$.

N – ім'я тегу *varDecl*;

dT – параметр, що описує тип даних змінної *N*, $dT \in \{int, string\}$ (*int*, *string* – цілочисельне та строкове значення);

$idObj \in \{1…K\}$ – порядковий номер елементу управління (ЕУ) ЕФ, де *K* – максимальна кількість елементів управління на екранній формі (ЕФ), що відповідає принципу *KISS* (*Keep It Simple, Stupid*) побудови інтерфейсів користувача;

$NameObj \in [\{a,A,...,я,Я\}\{a,A,...,z,Z\}\{0...9\}]$ – ім'я ЕУ;

$TypeObj \in \{C, T, X, B\}$ – тип ЕУ (*C* – ЕУ типу *ComboBox*; *T* – ЕУ типу *Text*; *X* – ЕУ типу *Table*; *B*– ЕУ типу *Button*);

$idR \in \{0…K\}$ – посилання на батьківський елемент *idObj*;

Sql – елемент зв'язку ЕУ та БД ІС;

$FillNumber[\{0…N\} \vee Null \vee *]$ – номер поля в запиті *select*, яке буде використовуватися для заповнення визначеного ЕУ;

$Delete \in \{0…N\}$ – номер поля ЕУ, яке слід приховати під час виконання програми, де *N* – кількість полів, що вертає *SQL*-запит.

```
<template formName= "C-21">
<varDecl
  type = "int"
  SQL = "select course from guide_courses order by course"
  idObj = "0"
  NameObj = "Курс"
  TypeObj = "C"
  idRef = "Null"
  FillNumber = "0"
>course</varDecl>
 <varDecl
  type = "String"
  SQL = "select distinct t.teach_year_id,teach_year from detailed_plan d, view_teach_year_graphic t where d.teach_year_id = t.teach_year_id and group_id =$1[0]"
  idObj = "2"
  NameObj = "Навчальний рік"
  TypeObj = "C"
  idRef = "1"
  FillNumber = "1"
>studyYear</varDecl>
```

Рис. 3.1. Представлення *XML*-частини шаблону ЕД

Опис *ED*-частини включає створення шаблону ЕД за наступним принципом: всі дані ЕД розділити на статичні, критеріальні та динамічні; замінити всі критеріальні та динамічні дані умовними позначеннями типу *$X*, де *X= vd.N* (рисунок 3.2).

ЗАБОРГОВАНІСТЬ СТУДЕНТІВ			
Навчальний рік		$studyYear	
Семестр		$semester	
Кількість боргів >=		$debtCount	
Курс		$course	
Група		$group	
Курс	Група	Студент	Кількість боргів
$courseInRow	$groupInRow	$studentName	$debtNumber

Рис. 3.2. Представлення частини шаблону ЕД

Аналіз структури та семантики ЕД у розрізі зв’язку з даними СПО та структури таблиць БД ІС дозволив зробити припущення, що всі данні ЕД можна розподілити на 3 типа:

1. Статичні дані (СД). Дані, що присутні у ЕД визначеного типу постійно і практично ніколи не змінюються, тобто ті, що описані множиною СПО: СД$\in$*Voc*. Приклад СД ЕД представлено на рисунку 3.3, сірим кольором.

ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ							
Навчальний рік	2011/2012					Семестр	
Напрям підготовки	6050103					Курс	3
Спеціальність						Група	АС091
	Відомість обліку успішності №		1282				
Дисципліна	Конструювання Програмного Забезпечення					ІСП	
Підсумкову оцінку виставив			n/d				
Поточний контроль здійснив			n/d				
№ П/П	Прізвище, ініціали студента	№ інд. навч. плану	Підсумкова оцінка за шкалою			Підпис викладача	
			Стобальною	Чотири бальн.	ECTS		
1	АНАНЧЕНКО Н.О.		85				
2	АНДРЄЄВ П.П.		0				
3	АРНАУТОВ О.В.		30				

Рис. 3.3. Статичні данні ЕД

2. Критеріальні данні (КД). Данні, що формуються для кожного екземпляру класу ЕД окремо. Приклад КД ЕД представлено на рисунку 3.4, сірим кольором.

ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ							
Навчальний рік	2011/2012					Семестр	
Напрям підготовки	6050103					Курс	3
Спеціальність						Група	АС091
	Відомість обліку успішності №		1282				
Дисципліна	Конструювання Програмного Забезпечення					ІСП	
Підсумкову оцінку виставив			n/d				
Поточний контроль здійснив			n/d				
№ П/П	Прізвище, ініціали студента	№ інд. навч. плану	Підсумкова оцінка за шкалою			Підпис викладача	
			Стобальною	Чотири бальн.	ECTS		
1	АНАНЧЕНКО Н.О.		85				
2	АНДРЄЄВ П.П.		0				
3	АРНАУТОВ О.В.		30				

Рис. 3.4. Критеріальні дані ЕД

3. Динамічні дані (ДД). Дані, що формуються на основі виконання *SQL*-запиту за критеріями вказаними КД. Приклад ДД ЕД представлено на рисунку 3.5, сірим кольором.

ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ

Навчальний рік	2011/2012	Семестр	
Напрям підготовки	6050103	Курс	3
Спеціальність		Группа	АС091

Відомість обліку успішності № 1282

Дисципліна	Конструювання Програмного Забезпечення
Підсумкову оцінку виставив	n/d
Поточний контроль здійснив	n/d

№ П/П	Прізвище, ініціали студента	№ інд. навч. плану	Підсумкова оцінка за шкалою			Підпис викладача
			Стобальною	Чотири бальн.	ECTS	
1	АНАНЧЕНКО Н.О.		85			
2	АНДРЄЄВ П.П.		0			
3	АРНАУТОВ О.В.		30			

Рис. 3.5. Динамічні дані

3.2 Опис методики автоматизованого створення шаблонів електронних документів

Серед методів породжуючого програмування *виділяють*:

- підстановки,
- підстановки з виконання коду,
- обробники даних регулярної структури.

Перший метод передбачає створення шаблону програми і набору даних в спеціальному форматі, але має недолік, що вимагає попередньої підготовки даних. Другий метод доповнює перший, дозволяючи вставляти в шаблон код, що оперує даними. У третьому методі використовується шаблон, який грає роль обробника даних і пишеться на спеціальній метамові, як приклад, *XSLT-обробник* даних, представлених в *XML-форматі*. Тому для автоматизованого створення *XML/ED*-шаблонів ЕД пропонується використовувати третій метод. метою автоматизованого створення *SQL*-запитів та складових елементів *XML/ED*-шаблонів ЕД використано модифікований метод породжуючого програмування – метод обробника шаблонів регулярних структур, що включає етап розмітки

ОМЕД-моделі ЕД та етапу генерації *SQL* запитів і схематично зображена у вигляді *IDEF0* діаграми на рисунку 3.6.

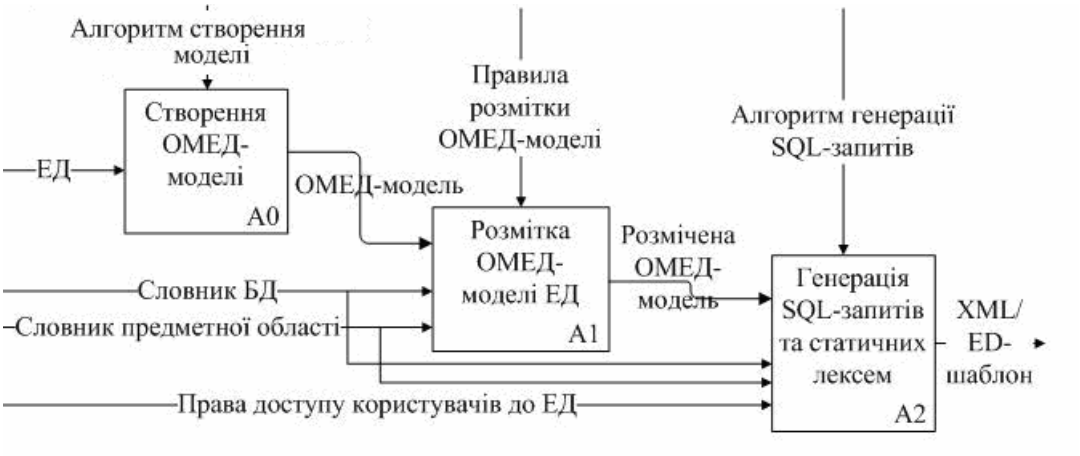


Рис. 3.6. *IDEF0* діаграма процесу створення *XML*-шаблонів

Вхідними даними методу є множини *N*, *Voc*, *A={<U, N>}* – множина правил доступу користувача *U* до *N* та *VocBD={vbd$_i$}* – множина записів словника БД, де *vbd$_i$=<t, f, v>* – кортеж, який встановлює відповідність таблиці *t* поля *f* та значення запису *v*.

Алгоритм роботи нульового етапу (А0) аналізує вихідні ЕД *XLS*-формату та створює модифіковану об’єктну модель електронного документу. Алгоритм роботи першого етапу (А1) аналізує множину *N*, *Voc*, та *VocBD* помічає кожен *і-й* вузол n_i множини *N* відповідним значенням – 0 (статичний), 1 (крітеріальний), 2 (динамічний), 3 (невизначений). Розмітка ЕД представлена в термінах логіки предикатів:

Якщо існують такі вузли *n.v* , для яких існують такі елементи *voc.v* , тоді помітити *n* як 0.

Предикати: P_1^2 *(n.v,voc.w)* – вузол *n* знайдений у СПО. P_2^2 *(n,0)* – помітити вузол *n* як 0:

$$\exists_n \exists_{voc} \left(P_1^2 \left(n.v, voc.w \right) \right) \rightarrow \exists_n \left(P_2^2 \left(n,0 \right) \right).$$

Для всіх *n*, які не помічені як 0 та у яких є сусідній вузол з міткою 0 помітити як 1.

Предикати: $P_3^2 \left(n, P_2 \right)$ – вузол *n*, у якого є сусідній вузол з поміткою 0. $P_4^2 \left(n,1 \right)$ –помітити вузол *n* як 1:

$$_n \exists_{P_2} \left(P_3^2 \left(n, \neg P_2 \right) \right) \rightarrow \exists_n \left(P_4^2 \left(n,1 \right) \right).$$

Для всіх *n*, які не помічені як 0 та 1, помітити як 2.

Предикат: $P_5^2 \left(n,2 \right)$ – помітити вузол *n* як 2:

$$_n \exists_{P_4} \left(\neg P_2 \wedge \neg P_4 \right) \rightarrow \exists_n \left(P_5^2 \left(n,2 \right) \right).$$

Динамічні вузли утворюють регулярні структури – *Regular*. *Regular* – це регулярна структура, утворена появою в ЕД ряду однотипних значень, які належать значенням з словника БД і між кожною парою яких немає відмінних значень. Приклад регулярної структури представлений на рисунку 3.7.

	Прізвище, ініціали студента
1	АНАНЧЕНКО Н.О.
2	АНДРЄЄВ П.П.
3	АРНАУТОВ О.В.

Рис. 3.7. Приклад регулярної структури в ЕД

Надалі регулярні структури в *ED*-частини шаблону зливаються в одне значення.

Для всіх вузлів *n*, які не помічені як 0, 1, 2, помітити як 3. *Предикат:* $P_6^2 \left(n,3 \right)$ – помітити вузол *n* як 3:

$$\forall_n \exists_{P5} \left(\neg P_2 \wedge \neg P_4 \wedge \neg P_5\right) \rightarrow \exists_n \left(P_6^2 (n,3)\right).$$

Описана формалізація розмітки дерева ЕД з використанням логіки предикатів дозволяє спростити процес програмування методики автоматизованого створення *XML/ED*-шаблону, виключаючи процес ручного аналізу структури ЕД та виявлення залежності структурних елементів ЕД та структур БД.

Розмітка вузлів дерева *ОМЕД*-моделі дозволяє перейти до формування *SQL*-запитів для вузлів з критеріальним і динамічним типом і включення їх в тіло *XML/ED*-шаблону.

Алгоритм роботи другого етапу (А2) формує *SQL*-запит, який представлено як кортеж з трьох елементів:

SQL=<*Sl*, *F*, *W*>, де

Sl={a_i} – множина атрибутів фрази *select*, *F*={t_i} – множина таблиць фрази *from*, *W*={w_i} – множина умов фрази *where*.

Генерація виразу *Sl*. Для всіх *n*, де *n.m=1*, сформувати фразу *Select*, де кожен атрибут *Sl* дорівнює атрибуту *dbs.a* словника *Voc* кожного n_i, що є спадкоємцем поточного *n*:

Select{a_i=*voc.dbr.dbs.*a_j}.

Генерація виразу *F*. Для всіх *n*, де *n.m=1*, сформувати вираз *from*, де кожен атрибут *F* дорівнює *dbs.r* словника *Voc* кожного вузлу n_i, що є спадкоємцем поточного вузла:

from{a_i= *voc.dbr.dbs.*r_j}.

Генерація виразу *W*. Для всіх *n*, де *n.m=1*, сформувати вираз *where*, де кожен атрибут *W*, формується як відповідь на функцію *path(dbd)* + умова, яка сформована для кожного поточного вузла n_i для якого назва атрибуту a_i дорівнює назві відповідного атрибуту з $Voc{:}a_i=voc_{i.dbs.a}$, а значення a_i дорівнює конкретному значенню вузла $n_i.v$:

$$where\{path(dbr)+a_i=n_i.v\}.$$

Збірка цілісного *SQL*-запиту і включення його у тіло *XML/ED*-шаблону:

$$Select\{a_i=voc.dbr.dbs.a_j\}||from\{ai=voc.dbr.dbs.r_j\}||where\{path(dbr)+a_i=n_i.v\}$$

Створення віртуальної таблиці доступу користувача *U* до результатів *SQL*-запиту:

$$Create\ view\ V\ as$$

$$Select\{a_i=voc.dbr.dbs.a_j\}||from\{ai=voc.dbr.dbs.r_j\}||where\{path(dbr)+a_i=n_i.v\}$$

Описана методика автоматизованого створення *SQL-запитів* для генерації *XML/ED*-шаблонів дозволяє скоротити час аналізу структури ЕД і БД з метою генерації *SQL-запитів* і скорочує вірогідність виникнення помилок кодування.

Автоматизація процесу створення шаблонів ЕД скоротила час на ручний аналіз ЕД і виявлення залежностей між структурами даних ЕД та БД, що в свою чергу надало можливість використовувати шаблони для генерації документно-орієнтованих ЕФ, функціонально достатніх для виконання однієї операції читання – *select* та трьох операцій модифікації – *insert, update, delete*. БД ІС.

3.3 Методика визначення кількості операцій створення шаблону

Для створення *XML/ED*-шаблонів адміністраторові необхідно володіти знаннями в тієї предметної області, для якої створюється шаблон, а також знати

структуру таблиць, що забезпечують обмін даними між конкретним ЕД та конкретними таблицями. Таким чином необхідно виконувати наступні операції:

- аналіз структури і вмісту ЕД, з метою визначення типів даних. Число операцій ($|O_{ЕД}|$) залежить від кількості виявлених окремих блоків динамічного, критеріального і статистичного типу;

- пошук регулярності і об'єднання даних. Число операцій ($|O_{Reg}|$) залежить від вмісту ЕД і кількості блоків даних, що утворюють регулярні структури в ЕД;

- аналіз залежних таблиць БД ІС. Число операцій ($|O_{Ext}|$) залежить від числа таблиць, обслуговуючих конкретний тип ЕД;

- зіставлення даних ЕД і таблиць БД. Число операцій ($|O_{Match}|$) дорівнює числу зв'язків між таблицями БД та ЕД (кількість *SQL*-запитів у шаблоні).

Розрахунок кількості операцій автоматизованого створення *XML/ED*-шаблонів запропоновано обчислювати за формулою 3.1:

$$O_{заг.} = |O_{ЕД}| + |O_{Reg}| + |O_{Ext}| + |O_{Match}| \quad (3.1)$$

де $O_{заг.}$ – загальне число операцій, що виконується при генерації *XML/ED* - шаблону.

3.5 Висновки до розділу

В даному розділі досліджено документно-орієнтований *XML/ED*-шаблон дозволяє встановити однозначні інформаційні потоки між слабоструктурованими елементами ЕД та жорстко детермінованими структурами БД, за допомогою *SQL*-запитів. Автоматизація процесу створення шаблонів ЕД скоротила час на ручний аналіз ЕД і виявлення залежностей між структурами даних ЕД та БД.

РОЗДІЛ 4
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ЗАСТОСУВАННЯ ШАБЛОНІВ ДЛЯ ПІДГОТОВКИ ДАНИХ В ЕЛЕКТРОННИХ ДОКУМЕНТАХ

4.1 Технологія ведення баз даних на основі електронних документів

Документно-орієнтована інформаційна технологія ведення БД з використанням ЕД *XLS* формату табличної структури [7], зображена на рисунку 4.1 та складається з наступної послідовності кроків:

Параметризація ЕД.

Вихідними даними першого етапу технології є

$ED=\{ed_1, \ldots, ed_n\}$ – множина вхідних ЕД,

де $ed_i=<no, x1, y1, x2, y2, v, l, r, u, b>$ – кортеж характеристик ed_i, де *no* – номер знайденого вузлу ЕД, *x1, y1* – координати лівого верхнього куту вузлу, *x2, y2* – координати правого нижнього куту вузлу, *v* – значення вузлу, *l* – товщина лівої границі вузлу, *r* – товщина правої границі вузлу, *u* – товщина верхньої границі вузлу, *b* – товщина нижньої границі вузлу.

Задача першого етапу - представити ЕД у вигляді вектору його характеристик. *Результатом роботи етапу* є створена множина *N*. $N=\{n_1, \ldots, n_n\}$ – множина вузлів ЕД, де $n_i=<pn, n, v, ix, l, m>$ – кортеж характеристик n_i, де *pn* – номер вузлу батька, *n* – номер вузлу, *v* – значення вузлу, *ix* – індекс значення у СПО, *l* – рівень схожості значення вузлу, $m \in \{0, 1, 2, 3\}$ – мітка приналежності вузлу до визначеного типу (0 – статичний тип, 1 – крітеріальний тип, 2 – динамічний тип, 3 – невизначений тип).

Створення кластерів ЕД або класифікація ЕД. Вихідними даними другого етапу є результат роботи першого, а саме - множина *N*. Для формування класів ЕД з вхідного множини *N* або для визначення класу нового ЕД використовуються різноманітні методи кластеризації та класифікації: ієрархічні алгоритми, *k-Means* алгоритм, мінімальне покриваюче дерево, метод найближчого сусіда, нечітка

кластеризація, нейроні мережі, генетичні алгоритми, метод загартування, вірогідності та нечислові класифікатори, регресійні методи, нейроні мережі, метод Роккіо, *k*-найближчого сусіда, метод опорних векторів. *Результатом роботи етапу* є створена множина *CED*.

$CED=\{cl_1, \ldots, cl_n\}$ – множина класів ЕД,

де $cl_i=<ed_1, \ldots, ed_n>$ – множина однотипових ЕД.

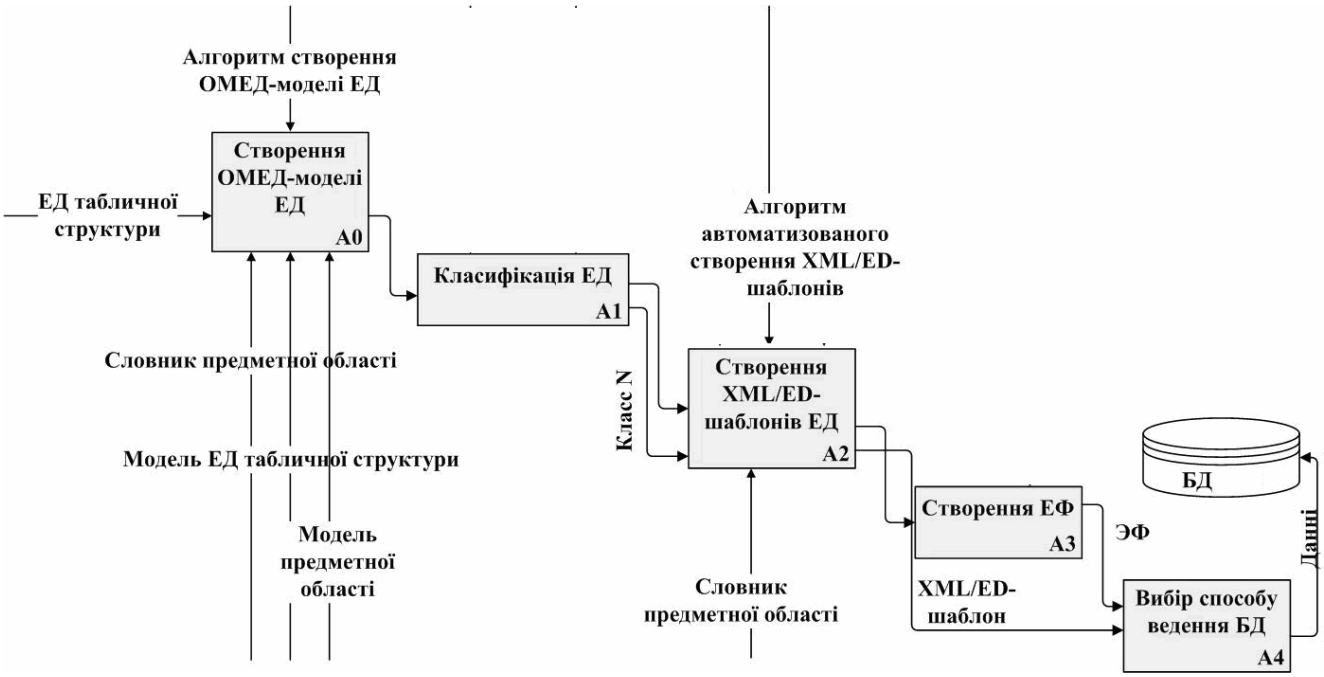


Рис. 4.1. Технологія ведення БД з використанням ЕД

Іменування класів ЕД

Вихідними даними третього етапу є результат роботи другого, а саме - множина *CED*. *Результатом роботи етапу* є створена множина *NCED*. $NCED=\{ncl_1, \ldots, ncl_n\}$ – множина іменованих класів ЕД, де $ncl_i=<ed_1, \ldots, ed_n>$ – множина однотипових ЕД.

Генерація шаблонів.

Вихідними даними четвертого етапу є результат роботи третього, а саме - множина *NCED*. Аналізуючи певний клас ЕД, створюється єдиний екземпляр класу – еталон. *Результатом роботи етапу* є створена множина *T*. $T=\{t_1, \ldots, t_n\}$

– множина шаблонів ЕД, де t_i – еталонний представник іменованого класу *NCED*, що включає елементи зв'язку ЕД та БД ІС.

Вибір способу ведення БД. Вихідними даними п'ятого етапу є результат роботи четвертого, а саме - множина *T*. Кожен шаблон t_i дозволяє виконувати операції вводу інформації до БД двома способами: за допомогою шаблону ЕД або ЕФ згенерованих за допомогою обраного t_i. *Результатом роботи етапу* є безпосередньо обраний спосіб або шаблон ЕД або екранні форми.

Перенесення даних до БД ІС. Вихідними даними п'ятого етапу є результат роботи четвертого. Автоматизоване перенесення даних з заповненого шаблону ЕД або за допомогою екранної форми до БД ІС ініціюється користувачем системи. *Результатом роботи етапу* є оновлена БД ІС.

4.2 Опис структур даних

Архітектура програмного продукту є реалізацією не функціональних вимог та встановлює глобальні обмеження. Зважаючи на це, встановлені наступні обмеження: ормат та структура ЕД – ЕД табличної структури різноманітних форматів офісних систем, типу *MS Office* та *Open Office*.

На рисунку 4.2 зображена діаграма концептуальних класів, що включає сім основних класів: *FileManagement*, *TreeGenerate*, *UIGenerate*, *DataBase*, *MarksOMED*, *CreateOMED*, *TemplateGenerate* та один клас агрегації *E-D*.

Клас *FileManagement.* Реалізовує підсистему управління ЕД та *XML/ED*-шаблонами: завантаження, обробку, видалення.

Клас *TreeGenerate.* Реалізовує підсистему генерації та навігації дерева ієрархії структурних підрозділів та залежних ЕД та *XML/ED*-шаблонів.

Клас *UIGenerate.* Реалізовує підсистему генерації документно-орієнтованих ЕФ на основі *XML/ED*-шаблонів.

Клас *CreateOMED.* Реалізовує методику створення ОМЕД-моделі ЕД з вхідного масиву ЕД табличної структури, СПО та словника бази даних.

Клас *MarksOMED.* Реалізовує механізм методику розмітки створеної ОМЕД-моделі ЕД.

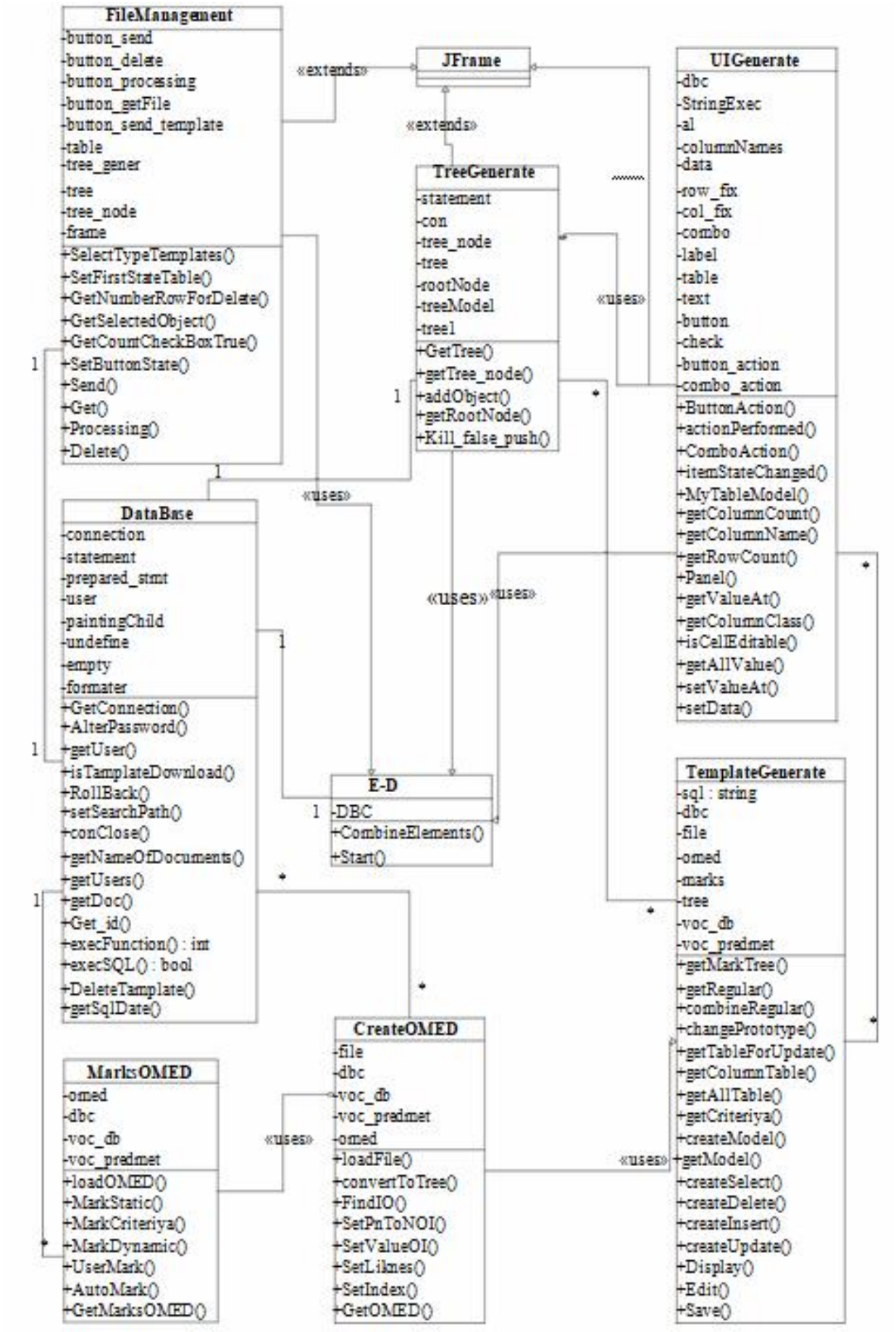


Рис. 4.2. Діаграма класів

Клас *TemplateGenerate*. Реалізовує методику автоматизованого створення *XML/ED*-шаблонів.

Клас *DataBase*. Реалізовує всі операції пов’язані з обміном даними між всіма підсистемами та ЕД.

Клас агрегації *E-D*. Реалізовує об’єднання всіх підсистем в єдину програмну систему.

4.3 Опис модулів генерації шаблонів електронних документів

4.3.1 Інтерфейс генератора XML/ED-шаблонів

Інтерфейс алгоритму створення *SQL*-запитів представлено в таблиці 4.1.

Таблиця 4.1

Інтерфейс алгоритму створення *SQL*-запитів

Назва методу	Опис
Клас *CreatePrototype*	
getMarkTree	Отримати розмічене дерево вузлів ЕД
getRegular	Визначити регулярні структури
combineRegular	Об'єднати регулярні структури
changePrototype	Змінити регулярні структури і Критичні блоки даних ЕД на умовні позначення типу “$name”
Клас *ExtTable*	
getTableForUpdate	Визначити таблицю для перенесення даних
getColumnTable	Визначити поля таблиці для перенесення даних
getAllTable	Визначити всі таблиці залежні від ЕД
getCriteriya	Визначити таблиці, обслуговуючі Критичні типи даних ЕД
Клас *CreateModel*	
createModel	Створити прототип *SQL*-запиту
getModel	Отримати прототип *SQL*-запиту
Клас *CreateSQL*	
createSelect	Згенерувати *SQL*-запит типу Select
createDelete	Згенерувати *SQL*-запит типу *Delete*
createInsert	Згенерувати *SQL*-запит типу *Insert*
createUpdate	Згенерувати *SQL*-запит типу *Update*
Клас *UserDisplay*	
Display	Відображує *SQL*-запит у візуальному редакторові
Edit	Задати нові елементи *SQL*-запиту
Save	Зберегти *SQL*-запит

Клас *CreatePrototype* створює прототип для інтерфейсної частини, а також створює чотири інтерфейсні вікна: для розмітки ЕД, для виявлення регулярних структур в електронному документі, для відображення зв'язаних таблиць, для генерації *SQL*-запитів.

Клас *Exttable* визначає залежні таблиці для побудови кожного типа запиту: *select, insert, update, delete*.

Клас *CreateModel* забезпечує створення прототипів запитів згідно описаних вище моделей.

Клас *CreateSQL* дозволяє створити *SQL*-запити чотирьох типів : *select, insert, update, delete* для подальшого редагування.

Клас *UserDisplay* визначає інтерфейс доступу до редагування і збереження *SQL*-запитів.

4.3.2 Інтерфейс генератора дерева ієрархії

Інтерфейс генератора дерева ієрархії структурних підрозділів представлено в таблиці 4.2.

Таблиця 4.2.

Інтерфейс генератора дерева ієрархії структурних підрозділів

Назва методу	Опис
Клас *TreeGenerate*	
GetTree	Побудувати дерева ієрархії з нульовим рівнем
getTree_node	Отримати поточний стан дерева
addObject	Додати новий вузол до дерева
getRootNode	Отримати дані визначеного вузла дерева
Kill_false_push	Видалення некоректних натискань на вузли дерева
treeNodesChanged	Отримати індексу вузлу що змінюється

Клас *TreeGenerate*, вміщує набір методів для побудови дерева ієрархії та маніпуляції з вузлами дерева.

4.3.3 Інтерфейс генератора екранних форм

Клас *Panel* – головний клас проекту, який створює і розміщує елементи управління на батьківській формі, а також створює масив обробників для певних ЕУ.

Клас *ButtonAction* – клас проекту, який описує можливі дії по обробці натискань на кнопку.

Клас *ComboAction* – клас проекту, який описує можливі дії по обробці натискань на елемент керування випадаючий список.

Клас *MyTableModel* – клас проекту, який описує інтерфейс доступу до ЕУ таблиця.

Клас *DatabaseConnection* – клас проекту, який описує інтерфейс доступу до БД.

Інтерфейс генератора ЕФ представлено в таблиці 4.3.

Таблиця 4.3

Інтерфейс генератора ЕФ

Назва	Опис
Клас *Panel*	
main	Головний метод проекту
Panel	Конструктор класу *Panel*
Клас *ButtonAction*	
ButtonAction	Конструктор обробника натискань на кнопку
actionPerformed	Обробник натискань на кнопку
Клас *ComboAction*	
ComboAction	Конструктор обробника натискань на ЕУ випадаючий список
itemStateChanged	Обробник натискань на ЕУ випадаючий список
Клас *MyTableModel*	
MyTableModel	Конструктор класу роботи з таблицями – *MyTableModel*
getColumnCount	Отримати кількість стовпців таблиці
getColumnName	Отримати імена стовпців таблиці
getRowCount	Отримати кількість рядків таблиці
getValueAt	Отримати значення комірки
getColumnClass	Отримати клас стовпця
isCellEditable	Можливість редагування комірки
getAllValue	Отримати всі значення таблиці
setValueAt	Встановити значення у комірку таблиці
setData	Встановити масив значень в таблицю
Клас *DatabaseConnection*	
DabaseConnection	Конструктор класу *DatabaseConnection*
GetConnection	Отримати поточне з’єднання
getData	Отримати масив значень з БД, як відповідь на виконання *SQL*-запиту
getTable	Отримати масив значень з БД, як відповідь на виконання *SQL*-запиту

Покроковий опис роботи програмних класів

Клас Panel створює елементи управління і розташовує їх на координатній сітці. Покрокове представлення роботи класу:

1. Підготовка *XPath* запитів на отримання кількості елементів управління, які необхідно розташувати на формі:

2. Створення підключення до БД.

3. Визначення кількості елементів управління на формі за допомогою підготовлених *XPath* запитів.

4. Створення координатної сітки.

5. Створення елементів управління *JComboBox*, *JButton*, *JTable*, *JText* в кількості визначеним на кроці 3.

6. Розташування елементів керування на координатній сітці.

7. Створення подій для елементів управління *JComboBox*, *JButton* - *ComboAction, ButtonAction*.

Клас ComboAction забезпечує обробку події натискання на елемент управління *JComboBox*:

1. Створення об'єкта класу.

2. Підготовка *XPath* запитів на отримання властивостей елементів управління *JComboBox*, *JButton*, *JText*, у яких батьківський елемент дорівнює поточному значенню циклу по і.

3. Виконання запитів в циклі по і та отримання списку параметрів елемента управління.

4. Вибір SQL-запиту з умовними значеннями типу $ [X].

5. Заміна умовностей $[X] на реальні значення з батьківських елементів управління.

6. Виконання змінених SQL-запитів і створення масивів значень.

7. Заповнення відповідних елементів управління із створених масивів.

Клас *ButtonAction* забезпечує обробку події натискання на елемент управління JButton:

1. Створення об'єкта класу.

2. Підготовка *XPath* запитів на отримання властивостей елементів управління *JButton*, *JTable*, у яких батьківський елемент дорівнює поточному значенню циклу по і.

3. Підготовка XPath запитів на отримання властивостей елементів управління JButton, JTable, у яких номер елемент керування дорівнює поточному значенню циклу по і.

4. Виконання запитів підготовленого на кроці 2 в циклі по і та отримання списку параметрів залежних елементів керування, якщо запит повернув нульовий результат, то виконуємо запит підготовлений на кроці 3 і переходимо на крок 5, інакше на крок 6.

5. В залежності від типу поточного об'єкта (delete, insert, update) виконуємо процедуру заміни умовностей $[X] на реальні значення з батьківських елементів управління.

6. Виконуємо процедуру заміни умовностей $[X] на реальні значення з батьківських елементів керування в SQL-запиті.

7. Виконуємо змінені SQL-запити.

Клас DabaseConnection описує операції пов'язані з виконанням запитів в БД і створення підключення до БД. Методи класу:

- *String [] getData (String sql)* - метод повертає масив записів, як відповідь на *SQL*-запит.

- *boolean sqlExecute (String sql)* - метод виконує *SQL*-запит і вирощує *true* якщо запит виконаний вдало або *false*, якщо запит виконаний невдало.

- *ArrayList <String[]> getDataTable (String sql)* - метод повертає список рядків як результат виконання *SQL*-запиту, причому перший рядок *ArrayList.get(0)* містить назви полів таблиці.

- *String [][] getTable (String sql)* - метод повертає двовимірний масив рядків як результат виконання *SQL*-запиту, кожен запис *String[i][j]* відповідає запису в таблиці.

- *void RollBack (SQLException ex)* - метод робить відкат транзакції в разі помилкового виконання запиту.

Клас StringToArrayString виконує перетворення одного типу даних *ArrayList <String[]>* до іншого *String[][]*.

Клас MyTableModel є спадкоємцем класу *AbstractTableModel* і реалізує інтерфейс *TableModel*. У класі перевизначені наступні методи:

Три конструктора: *MyTableModel ()*, *MyTableModel (String [] [] str)*, *MyTableModel (Object [] [] str)*;

- Методи: *getColumnClass (int col)*, *isCellEditable (int row, int col)*, *setValueAt (Object value, int row, int col)*, *setColName (String [] str)*;

removeAllRow () - видаляє всі рядки в моделі таблиці;

removeRow (int row []) - видаляє зазначені рядки таблиці, в якості параметрів вказується масив номерів рядків, які необхідно видалити;

addRow (Object [] info) - додає новий рядок в модель таблиці, в якості параметра вказується масив об'єктів рядки;

addRow (String [] info) - додає новий рядок в модель таблиці, в якості параметра вказується масив рядків;

setData (Object [][] obj) - встановити нову модель таблиці, в якості параметрів вказується двовимірний масив об'єктів;

setData (String [][] str) - встановити нову модель таблиці, в якості параметрів вказується двовимірний масив рядків;

setData (ArrayList <String[]> str) - встановити нову модель таблиці, в якості параметра вказується список строкових одновимірних масивів, де перший масив списку є рядком заголовків полів таблиці;

Vector <Object[][]> getUpdatedRowIndexes () - метод оновлює модель таблиці у випадку якщо вносяться нові дані в комірки;

setColumnFix (int col []), *setRowFix (int row [])* - встановлює блокування на зазначені в якості параметра масив стовпців.

4.3.4 Інтерфейс управління XML/ED-шаблонами ЕД

Інтерфейс управління *XML/ED*-шаблонами ЕД – *FileManagement* представлено в таблиці 4.4.

Таблиця 4.4.

Інтерфейс підсистеми *FileManagement*

Назва методу	Опис
Клас *FileManagement*	
SelectTypeTemplates	Визначає тип шаблону, що оброблюється
SetFirstStateTable	Визначає початковий стан таблиці завантажених шаблонів
GetNumberRowForDelete	Повертає порядкові номери рядків з назвами шаблонів які необхідно видалити
GetSelectedObject	Визначає обрані шаблони для обробки
GetCountCheckBoxTrue	Визначає кількість активованих елементів *CheckBox*
SetButtonState	Встановлює стан кнопок обробки шаблонів, в залежності від заданих умов
Send	Визначає дії по завантаженню *XML/ED*-шаблонів ЕД та звичайних ЕД на сервер обробки
Get	Визначає дії по завантаженню *XML/ED*-шаблонів ЕД та звичайних ЕД на визначений комп'ютер
Processing	Визначає дії по обробці ЕД у відповідності з *XML/ED*-шаблоном
Delete	Визначає дії по видаленню *XML/ED*-шаблонів ЕД та звичайних ЕД з серверу.

Клас *FileManagement* вміщує набір методів для обробки *XML/ED*-шаблонів ЕД та звичайних ЕД табличної структури, а саме: завантаження, видалення, отримання та обробка.

4.4 Висновки до розділу

В даному розділі описана технологія ведення баз даних на основі електронних документів, наведено опис структур даних та детальний опис модулів генерації шаблонів електронних документів.

ВИСНОВКИ

В магістерській роботі досліджено моделі, методи та засоби підготовки даних в системах електронного документообігу. Розглянута проблема автоматизації процесу обміну даними між електронними документами з табличною структурою та базою даних інформаційної системи з метою підвищення продуктивності цього процесу та зменшення часу на ручні операції вводу та зменшення вірогідності помилок.

Порівняльний аналіз форматів представлення електронних документів, які використовуються у інформаційних процесах, та аналіз їх структури показав, що найбільш поширенішими форматом електронних документів є формат з табличною структурою. Аналіз існуючих способів ведення баз даних визначив, що вони не враховують можливість взаємного відображення даних у електронних документах з табличною структурою.

В роботі описана об’єктна метамодель електронного документа яка виконує приведення до єдиного формату електронних документів табличної структури.

Досліджено метод оброблювача даних регулярних структур, що дозволяє автоматизувати механізм генерації *XML*- шаблонів електронних документів. Наведений метод також дозволяє генерувати окремі документно-орієнтовані екрані форми зв’язку електронних документів з базою даних на основі *SQL*-запитів, функціонально достатніх для виконання основних операцій ведення бази даних, і тим самим виключити етап створення нових програмних модулів засобами мов програмування.

На основі методики вибору документно-орієнтованого способу ведення БД вдалось визначити, що електронний документ, у якому переважають рядкові дані, раціонально обробляти за допомогою відповідної форми доступу до бази даних, а електронний документ, в якому переважають числові типи, раціонально обробляти за допомогою *XML*-шаблонів.

Представлена інформаційна технологія ведення бази даних, яка суттєво скорочує час вводу даних з електронного документа до бази даних інформаційної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Automatic Digital Document Processing and Management: Problems, Algorithms and Techniques, Advances in Pattern Recognition, ISBN 978-0-85729.
2. Statistics in Education and the Science (With application to research). Laurentina Paler-Calmorin, Melchor A. Calmorin, RBS, First Edition 1997, ISBN 971-23-2232-7.
3. Klass, Gary M. 2008. Creating Good Charts." Chapter 3 in Just Plain Data Analysis: Finding, Presenting, and Interpreting Social Science Data.
4. David Meyer. OOXML ratification faces delay after objection / David Meyer 2008. – 840c.
5. Cong Yu ; Zhihong Yao; Mapping DICOM to OpenDocument format. Proc. SPIE 7264, Medical Imaging 2009: Advanced PACS-based Imaging Informatics and Therapeutic Applications, 72640Y.
6. Carpenter, Arthur L. and Dennis G. Fisher, 2011, “R eading and Writing RTF Documents as Data: Automatic Completion of CONSORT Flow Diagrams”, presented at the Western Users of SAS Software Conference, WUSS. http://www.wuss.org/proceedings11/Papers_Carpenter_A_74920.pdf.
7. Microsoft Office for the Older and Wiser: Get Up and Running with Office 2010 and Office 2007, Sean McManus, John Wiley / University of the Third Age (U3A), 978-0470711965, 308, full colour.
8. Klyahzkin, E. Shchepin, K. Zingerman. Hierarchical analysis of multi-column texts, Pattern Recognition and Image Analysis, Vol.5, No.1, 1995,
9. V. Kliatskine, G. Thorvaldsen, K. Zingerman, V. Lazarev, E. Shchepin, “A structural method for the recognition of complex historical tables”, History & Computing, 9, no. 3, Edinburg University Press, 1997, 58–77.
10. Shigarov A.O. A method for table detection in metafiles [Текст] /Shigarov A.O., Bychkov I.V., Khmel’nov A.E., Ruzhnikov G.M. // Pattern Recognition andImage Analysis. – 2009. – Vol. 19, No 4. P. 693–697.
11. Ng H.T., Lim C.Y., Li Teng Koo J. Learning to recognize tables in free text In Proc. 37th Annual Meeting of the Association for Computational Linguistics. USA. 1999. P. 443-450.

12. Embley D.W., Tao C., Liddle S. Automating the extraction of data from HTML tables with unknown structure // Data & Knowledge Engineering. Elsevier Science Publishers. 2005. Vol. 54, No 1. P. 3-28.
13. Fuhr, N., Hartmann, S., Knorz, G., Lustig, G., Schwantner, M., Tzeras, K., AIR/X – a rulebased multistage indexing system for large subject fields. In
14. Proceedings of RIAO-91, 3rd International Conference “Recherche d’Information Assistee par Ordinateur” (Barcelona, ES, 1991), pp. 606–623., 1991
15. Cohen, W. W., Hirsch, H., Joins that generalize: text classification using Whirl.In Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining (New York, US, 1998), pp. 169–173., 1998], [Cohen, W. W, Singer, Y., Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems 17, 2, 141–173. , 1999
16. T. Joachims “A probabilistic analysis of the rocchi o algorithm with TFIDF for text categorization.”, In Proc. of the ICML'97, 143-151, 1997.
17. Shapire R.E./ Singer Y., Singhal A. Boosting and Rocchio applied to text filtering // Proceedings of SIGIRS-98 / 21st ACM International Conferece on Research and Development in Information Retrieval. – New York: ACM Press, 1998. – P. 215-223.
18. D. T. Pham and E. Oztemel, Control chart pattern recognition using combinations of multi-layered perceptrons and learning-vector quantisation neural networks, Proc. IMechE, Part E, Journal of Process Mechanical Engineering, 207, pp.113-118, 1994.
19. Y. Solano and H. Ikeda, A comparative study of eight learning algorithms for arti_cial neural networks based on a real application, IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, E81A, pp.355-357, 1998.
20. Larkey, L. S., Croft, W. B., Combining classifiers in text categorization. In Proceedings of SIGIR-96, 19th ACM International Conference on Research and Developmentin Information Retrieval (Zurich, CH, 1996), pp. 289–297., 1996
21. Yang, Y., Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In Proceedings of SIGIR-94, 17th ACM

International Conference on Research and Development in Information Retrieval (Dublin, IE, 1994),pp. 13–22., 1994

22. P. Pinheiro da Silva and N. Paton. User Interface Modelling with UML. In Proceedings of the 10th European-Japanese Conference on Information Modelling and Knowledge Representation, Saariselk¨a, Finland, May 2000. IOS Press.
23. Pinheiro da Silva, P., 2000. User interface declarative models and development environments: A survey. In Interactive Systems - Design, Specification, and Verification: 7th International Workshop, DSV-IS 2000, Limerick, Ireland, June 2000. Revised Papers, Springer Berlin / Heidelberg, Lecture Notes in Computer Science vol. 1946, pp. 207–226.
24. F. Moussa, C. Kolski, M. Riahi, A Model Based Approach to Semi-Automated User Interface Generation for Process Control Interactive Applications. Interacting with Computers 12, 3 (2000) 245–279.
25. F. Moussa, M. Moalla, Toward a user centered system design methodology: Application to the graphical interfaces design, AMSE Periodicals Advanced in modelling and analysis 35 (1) (1995) 1– 10.
26. F. Moussa, C. Kolski, Vers une formalisation d'une de´marche de conception de synoptiques industriels:application au syste`me ERGO-CONCEPTOR, Proceedings Colloque ERGO-IA Ergonomie et Informatique Avance´e, 7–9 October, Biarritz, France, 1992.
27. Silva, A.R., Saraiva, J., Silva, R., Martins, C. (2007). XIS - UML profile for extreme modeling interactive systems. In Proceedings of the 4th International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2007). IEEE, March.
28. Pastor, O., Molina, J. (2007). Model-driven Architecture in Practice – A software production environment based on Conceptual Modeling. Springer-Verlag.
29. Pastor, O., Insfrán, Pelechano, V., Romero, J., Merseguer, J. (1997). OO-METHOD: An OO software production environment combining conventional and formal methods. In CAiSE '97: Proceedings of the 9th International Conference on Advanced Information Systems Engineering, pages 145-158, London, UK. Springer-Verlag.

30. Pastor, O., Insfrán, E. (2003). OO-Method, the methodological support for OlivaNova model execution system. Technical report, Care Technologies. White paper. Available at http://www.care-t.com.
31. Molina, P., Pastor, O., Marti, S., Fons, J., Insfrán, E. (2001). Specifying conceptual interface patterns in an object-oriented method with automatic code generation. In Proceedings Second International Workshop on User Interfaces in Data Intensive Systems, UIDIS 2001.
32. Jia, X., Steele, A., Liu, H., Qin, L., Jones, C. (2005). Using ZOOM approach to support MDD. In Proceedings of the 2005 International Conference on Software Engineering Research and Practice (SERP'05), Las Vegas, USA.
33. Jia, X., Steele, A., Qin, L., Liu, H., Jones, C. (2007). Executable visual software modelling – the ZOOM approach. Software Qu ality Control, 15(1):27-51.
34. Puerta, Angel R., Eriksson, Henrik, Gennari, John H. and Musen, Mark A. (1994): Beyond Data Models for Automated User Interface Generation. In: Cockton, Gilbert, Draper, Steven and Weir, George R. S. Proceedings of the Ninth Conference of the British Computer Society Human Computer Interaction Specialist Group - People and Computers IX August 23-26, 1994, Glasgow, Scotland, UK. pp. 353-366.
35. Gennari, J.H. 1993. A Brief Guide to Maître and MODEL: An Ontology Editor and a Frame-Based Knowledge Representation Language. Stanford University, Knowledge Systems Laboratory, Report KSL-93-46, Stanford, California. June 1993.
36. Eriksson, H., Puerta, A.R. and Musen, M.A. 1994. Generation of Knowledge-Acquisition Tools from Domain Ontologies. In Proceedings of the Eighth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff, Alberta, Canada. pp. 7.1–7.20.
37. Kleshchev Alexander, Gribova Valeriya. From an Ontology-Oriented Approach Conception to User Interface Development // Information Theories & Using Database Metadata and its Semantics toGenerate Automatic and Dynamic Web Entry Forms. Mohammed M. Elsheh and Mick J. Ridley
38. K. Alsabti, S. Ranka, V. Singh. An Efficient k-means Clustering Algorithm, Proc. First Workshop High Performance Data Mining, Mar. 1998.

39. V. Faber. Clustering and the Continuous k-means Algorithm, Los Alamos Science, vol. 22, pp. 138-144, 1994.
40. S.Z. Selim and M.A. Ismail, K-means-type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality, IEEE Trans. Pattern
41. G. Celeux and J. Diebolt. The SEM algorithm : A probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. Computational Statistics Quarterly, 2:73–82, 1985.
42. G.J. MacLachlan and T. Krishnan. The EM Algorithm and Extensions, pages 120–211. Wiley, New York, 1997.
43. Yiming Yang, Xin Liu A re-examination of text categorization methods //Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. – New Yor k: ACM Press, 1999. – pp. 42-49.
44. Cachero, C., Gòmez, J., Pastor, O., “Object-Oriente d Conceptual Modeling of Web Application Interfaces: the OO-H Method Abstract Presentation Model”, 2000
45. Damiano Distante, Paola Pedone, Gustavo Rossi, Gerardo Canfora. Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces, Web Engineering, 7th International Conference, ICWE 2007, Como, Italy, July 16-20, 2007, Proceedings. Volume 4607 of Lecture Notes in Computer Science, pages 457-472, Springer, 2007.
46. Christopher Scaffidi, Allen Cypher, Sebastian G. Elbaum, Andhy Koesnandar, Brad A. Myers.Scenario-Based Requirements for Web Macro Tools. In 2007 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007), 23-27 September 2007, Coeur d Alene, Idaho, USA. pages 197-204, IEEE Computer Society, 2007.

ДОДАТКИ

Додаток А

Опис процесу управління електронними документами та XML-шаблонами

На рисунку А.1 представлена діаграма прецедентів управління ЕД, що складається з п’яти прецедентів:

1. “ Відіслати файл на сервер”.
2. “ Відіслати шаблон на сервер”.
3. “ Отримати файл з серверу”.
4. “ Видалити файл з серверу”.
5. “ Обробити файл”.

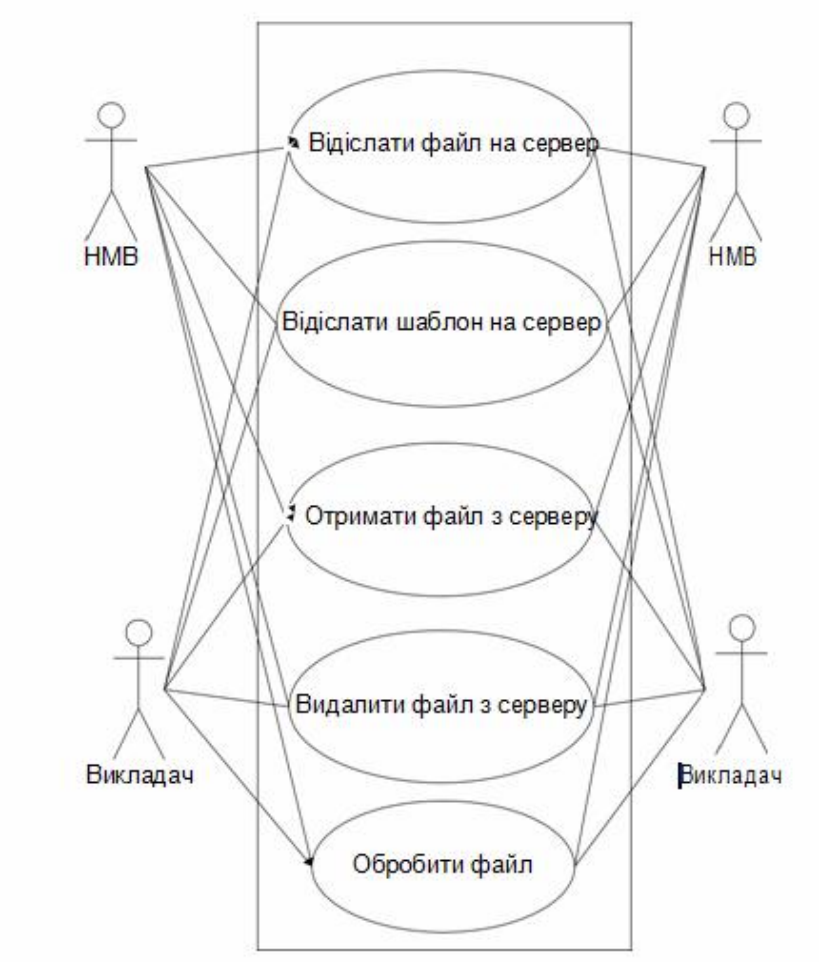


Рис. А.1. *Use Case* діаграма

Для кожного прецеденту в таблиці А.1, А.2, А.3, А.4, А.5 описано передумова, тригер, постумова сценарій його використання та альтернативний сценарій.

Таблиця А.1 – Описання прецеденту “ Відіслати файл на сервер”

Опис	Для відсилки певних файлів на сервер, з ціллю їх подальшого використання іншими акторами системи
Актор	Навчально методичний відділ, Викладач
Передумова	1. Користувач успішно авторизований (визначені права доступу). 2. Користувач перемістився на кінцевий вузол ієрархії, відповідний документу що завантажується. 3. В програмі з’являється кнопка “ Відіслати файл на сервер”.
Тригер	Натискання на кнопку “ Відіслати файл на сервер”
Сценарій	1. З’являється вікно вибору файлу. 2. Користувач натискає обирає необхідний файл і натискає кнопку “ Обрати”. 3. Обраний файл перевіряється на відповідність розширенню *.*doc, *.xls*. 4. Файл завантажується на сервер. 5. Файл з’являється не в першому рядку списика файлів. В полі “ Ім’я” з’являється ім’я файлу, в полі “ Дата” – дата завантаження файлу, в полі “ Стан” – напис “ Завантажено”, в полі “ Власник” з’являється ім’я особи, що завантажила файл, поле “ Відмітка” залишається порожнім.
Альтернативний сценарій №1	1. Користувач натискає кнопку “ Відіслати файл на сервер”. 2. З’являється вікно вибору файлу. 3. Користувач натискає обирає необхідний файл і натискає кнопку “ Обрати”. 4. Обраний файл перевіряється на відповідність розширенню *.*doc, *.xls*. 5. Файл не завантажується на сервер – відсутній зв'язок або інші неполадки зі з’єднанням. 6. На екран користувачу виводиться відповідне повідомлення.
Альтернативний сценарій №2	1. Користувач натискає кнопку “ Відіслати файл на сервер”. 2. З’являється вікно вибору файлу. 3. Користувач натискає обирає необхідний файл і натискає кнопку “ Обрати”. 4. Обраний файл перевіряється на відповідність розширенню *.*doc, *.xls*. 5. Файл не проходить перевірку на відповідність розширенню *.*doc, *.xls*. 6. На екран користувачу виводиться відповідне повідомлення з проханням обрати файл який відповідає формату *.*doc, *.xls*.

Таблиця А.2 – Описання прецеденту “ Відіслати шаблон на сервер”

Опис	Для відсилки певних шаблонів на сервер, з ціллю їх подальшого використання іншими акторами системи
Актор	Навчально методичний відділ, Викладач
Передумова	1. Користувач успішно авторизований (визначені права доступу). 2. Користувач перемістився на кінцевий вузол ієрархії, відповідний документу що завантажується. 3. В програмі з’являється кнопка “ Відіслати шаблон на сервер”
Тригер	Натискання на кнопку “ Відіслати шаблон на сервер”
Сценарій	1. З’являється вікно вибору типу документу, що завантажується: *D* – документ-зразок, *T-XLS/DOC* – шаблон для автоматичної генерації документа-зразка. 2. Користувач обирає необхідний тип документу і натискає кнопку “ Обрати”. 3. З’являється вікно вибору файлу. 4. Користувач обирає необхідний файл і натискає кнопку “ Обрати”. 5. Обраний файл перевіряється на відповідність розширенню **.doc, *.xls*. 6. Файл завантажується на сервер. 7. Файл з’являється в першому рядку списика файлів. В полі “ Ім’я” з’являється ім’я файлу + напис “ Шаблон”, в полі “ Дата” – дата завантаження файлу, в полі “ Стан” – напис “ Завантажено”, в полі “ Власник” з’являється ім’я особи, поле “ Відмітка” залишається порожнім.
Альтернативний сценарій №1	1. Користувач натискає кнопку “ Відіслати шаблон на сервер”. 2. З’являється вікно вибору файлу. 3. Користувач натискає обирає необхідний файл і натискає кнопку “ Обрати”. 4. Обраний файл перевіряється на відповідність розширенню **.doc, *.xls*. 5. Файл не завантажується на сервер – відсутній зв'язок або інші неполадки зі з’єднанням. 6. На екран користувачу виводиться відповідне повідомлення.

продовження таблиці А.2

Альтернативний сценарій №2	1. Користувач натискає кнопку “ Відіслати шаблон на сервер ”. 2. З’являється вікно вибору файлу 3. Користувач натискає обирає необхідний файл і натискає кнопку “ Обрати”. 4. Обраний файл перевіряється на відповідність розширенню *.doc, *.xls. 5. Файл не проходить перевірку на відповідність розширенню *.doc, *.xls. 6. На екран користувачу виводиться відповідне повідомлення з проханням обрати файл який відповідає формату *.doc, *.xls.
Альтернативний сценарій №3	1. Користувач натискає кнопку “ Відіслати шаблон на сервер ”. 2. З’являється вікно вибору файлу 3. Користувач натискає обирає необхідний файл і натискає кнопку “ Обрати”. 4. Обраний файл перевіряється на відповідність розширенню *.doc, *.xls. 5. Обраний файл вже існує на сервері. 6. На екран користувачу виводиться відповідне повідомлення з інформацією о том, що файл на сервері буде замінено обраним. 7. Користувач погоджується з повідомленням. 8. Файл завантажується на сервер.

Таблиця А.3 – Описання прецеденту “ Отримати файл з серверу”

Опис	Для отримання шаблонів та файлів з подальшим їх використання
Актор	Навчально методичний відділ, Викладач
Передумова	1. Користувач успішно авторизований (визначені права доступу). 2. Користувач перемістився на кінцевий вузол ієрархії, відповідний документу що завантажується. 3. В таблиці списку файлів з’являються файли які можна звантажити для обраного типу документу в дереві ієрархії. 4. В програмі з’являється кнопка “ Отримати файл з серверу” 5. В полі “ Відмітка” таблиці списку файлів зроблено одну або більше відміток.
Тригер	Натискання на кнопку “ Отримати файл з серверу”

продовження таблиці А.3

Сценарій	1. З’являється вікно вибору місця для завантаження файлу(ів). 2. Користувач визначає місце завантаження обраних файлів і натискає кнопку “ Обрати”. 3. Файл завантажується на локальну машину користувача. 4. Користувачу видається повідомлення про успішне завантаження певної кількості файлів.
Постумова	
Альтернативний сценарій №1	1. Користувач натискає кнопку “ Отримати файл з серверу ”. 5. З’являється вікно вибору місця для завантаження файлу(ів). 2. Користувач визначає місце завантаження обраних файлів і натискає кнопку “ Обрати”. 3. Файли що обрані користувачем для завантаження мають однакові імена. 4. Операційна система забороняє такі дії а користувачу повідомляє про неможливість такої операції.
Альтернативний сценарій №2	1. Користувач натискає кнопку “ Отримати файл з серверу ”. 2. З’являється вікно вибору місця для завантаження файлу(ів). 3. Користувач визначає місце завантаження обраних файлів і натискає кнопку “ Обрати”. 4. Файли не завантажують з причини недостатнього об’єму вільного місця на диску. 5. Операційна система забороняє такі дії а користувачу повідомляє про неможливість такої операції.

Таблиця А.4 – Описання прецеденту “ Видалити файл з серверу”

Опис	Для видалення завантажених на сервер файлів
Актор	Навчально методичний відділ, Викладач
Передумова	1. Користувач успішно авторизований (визначені права доступу). 2. Користувач перемістився на кінцевий вузол ієрархії, відповідний документу що завантажується. 3. В таблиці списку файлів з’являються файли які можна видалити. 4. В програмі з’являється кнопка “ Видалити файл з серверу ” 5. В полі “ Відмітка” таблиці списку файлів зроблено одну або більше відміток.
Тригер	Натискання на кнопку “ Видалити файл з серверу ”

продовження таблиці А.4

Сценарій	1. З’являється повідомлення про видалення файлів з серверу. 2. Користувач підтверджує дію. 3. Відмічені файли видаляються з сервера. 4. Користувачу видається повідомлення про те, що файли успішно видалені.
Постумова	
Альтернативний сценарій №1	1. З’являється повідомлення про видалення файлів з серверу. 2. Користувач підтверджує дію. 3. Обрані файли не видаляються – відсутній зв'язок з БД. 4. Користувачу видається відповідне повідомлення.

Таблиця А.5 – Описання прецеденту “ Обробити файл”

Опис	Для обробки певних файлів на сервері
Актор	Навчально методичний відділ, Викладач
	1. Користувач успішно авторизований (визначені права доступу). 2. Користувач перемістився на кінцевий вузол ієрархії, відповідний документу що завантажується. 3. В таблиці списку файлів з’являються файли які можна Передумова обробити. 4. В полі “ Стан” таблиці списку файлів стоїть значення яке не дорівнює – “ Помилка” або “ Оброблений” 5. В програмі з’являється кнопка “ Обробити файл” 6. В полі “ Відмітка” таблиці списку файлів зроблено одну відмітку файлу що буде оброблюватися.
Тригер	Натискання на кнопку “ Обробити файл”
Сценарій	1. З’являється повідомлення про те, що обраний файл буде оброблено. 2. Користувач підтверджує дію. 3. Обраний файл оброблюється, данні потрапляють до таблиць БД. 4. Користувачу видається повідомлення про те, що файл успішно оброблено. 5. В полі “ Стан” таблиці списку файлів з’являється напис “ Оброблений”.

продовження таблиці А.5

Альтернативний сценарій №1	1. В полі “ Відмітка” таблиці списку файлів зроблено не одну відмітку. 2. Користувачу видається повідомлення з проханням обрати тільки один файл для обробки. 3. Користувач обирає один файл для обробки. 4. З’являється повідомлення про те, що обраний файл буде оброблено. 5. Користувач підтверджує дію. 6. Обраний файл оброблюється, данні потрапляють до таблиць БД. 6. Користувачу видається повідомлення про те, що файл успішно оброблено. 7. В полі “ Стан” таблиці списку файлів з’являється напис “ Оброблений”.
Альтернативний сценарій №2	1. З’являється повідомлення про те, що обраний файл буде оброблено. 2. Користувач підтверджує дію. 3. Обраний файл оброблюється, але виникає помилка обробки. 4. Користувачу видається повідомлення про те, що виникла помилка при обробці файлу. 5. Користувачу пропонується вибрати інший файл на обробку. 6. Користувач обирає інший файл. 7. З’являється повідомлення про те, що обраний файл буде оброблено. 8. Користувач підтверджує дію. 9. Обраний файл оброблюється, данні потрапляють до таблиць БД. 10. Користувачу видається повідомлення про те, що файл успішно оброблено. 11. В полі “ Стан” таблиці списку файлів з’являється напис “ Оброблений”.

Додаток Б

Б1. Приклад створення та заповнення структури

Приклад створення таблиці *tree_node*:

```
CREATE TABLE tree_node

(
id integer NOT NULL,

"name" character varying NOT NULL,
parent_id integer,
sql_def character varying,
CONSTRAINT tree_node_pkey PRIMARY KEY (id)
)
```

Таблиця Б.1 – Приклад заповнення структури *tree_node*

Id	Name	Parent_id	Sql_def
1	ІФНТУНГ	0	""
2	Адмін. підрозділ	1	select 13 as id_form, * from admin_department_list
3	Ін-т/Фак-т	1	select 14 as id_form,* from faculty_list
4	Склад співробітників	3	select 15 as id_form,dep_id, fio||' '||position as fio from employee_list where dep_id = @3@
5	Документи навчального процесу	3	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @3@ and dh.id in (101,102,106,110,111,112,113,500,209) order by order_id
6	Кафедри	3	select 17 as id_form,* from department_list where p_id = @3@
7	Склад співробітників	6	select 18 as id_form,dep_id, fio||' '||position as fio from employee_list where dep_id = @6@
8	Документи навчального процесу	6	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @6@ and dh.id in (101,102,106,110,208) order by order_id
9	Склад студентів	6	select 20 as id_form,course,course from guide_courses order by course
10	Лабораторії	6	""
11	Студ.групи	9	select 21 as id_form,gr_name,gr_name from group_list where dep_id = @6@ and course = @9@

продовження таблиці Б.1

12	Студенти групи	11	select 22 as id_form,pers_id,full_name from view_student where gr_name = '@11@' order by foreign_student,full_name
13	Контроль якості навчання	3	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @3@ and dh.id in (123,105,300,400) order by order_id
14	Аналіз якості навчання	3	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @3@ and dh.id in (201,202,203,205,206,207,210) order by order_id
15	Контроль якості навчання	6	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @6@ and dh.id in (123,108,109,225,226,301,104) order by order_id
16	Аналіз якості навчання	6	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @6@ and dh.id in (221,222,223,225,227) order by order_id
17	Журнал змін в документах	3	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @3@ and dh.id in (700,702,703) order by order_id
18	Журнал змін в документах	6	select dh.id, dh.id,dh.name,dh.id from doc_hierarchy dh, department d, struct_hierarchy s where dh.struct_id = s.id and s.short_name = d.dep_type and d.id = @6@ and dh.id in (701) order by order_id

Б.2. Приклад *XML*-шаблону ЕД

XML частина шаблону

```
<varDecl
  type="int"
  varType_DBRelation="SELECT"
  varType_InputRelation="NEVER_IN_INPU
  T" varType_XLSVisisbility="VISIBLE"
  fromDBQuery="select course from
          onpu.department_group
          where dep_group_id =
          (select group_id from
          onpu.detailed_plan where id =
          $detailed_plan_id);"
  idObj   ="0"
  NameObj ="Курс"
  TypeObj ="C"
  idRef   ="Null"
  SQL   ="select course from guide_courses order by course"
  FillNumber="0"
 >course</varDecl>
 <varDecl
  type="String"
  varType_DBRelation="SELECT"
  varType_InputRelation="NEVER_IN_INPU
  T" varType_XLSVisisbility="VISIBLE"
  fromDBQuery="select gr_name from
          onpu.view_vedomost_marks where
          plan_id = $detailed_plan_id;"
  idObj ="1"
  NameObj ="Группа"
  TypeObj ="C"
  idRef   ="0"
  SQL     ="select  dep_group_id,gr_name
                          from
   department_group  where
  course =$0[0] and dep_id =73"
  FillNumber  ="1"
 >groupName</varDecl>
 <varDec
  type="String"
  varType_DBRelation="SELECT"
  varType_InputRelation="NEVER_IN_INPUT"
  varType_XLSVisisbility="VISIBLE"
  fromDBQuery="select cast(teach_year as
  char(9)) from
          onpu.view_vedomost_marks where
          plan_id = $detailed_plan_id;"
  idObj   ="2"
  NameObj ="Навчальний рік"
  TypeObj ="C"
     idRef ="1"
  SQL   ="select distinct t.teach_year_id,teach_year from
        detailed_plan d, view_teach_year_graphic t where
        d.teach_year_id = t.teach_year_id and group_id =$1[0]"
```

```
        FillNumber="1"
       >studyYear</varDecl>
      <varDecl
        type="String"
        varType_DBRelation="SELECT"
        varType_InputRelation="NEVER_IN_INPU
        T" varType_XLSVisisbility="VISIBLE"
        fromDBQuery="select teach_subject
        from
                onpu.view_vedomost_marks where
                plan_id = $detailed_plan_id;"
        idObj  ="3"
        NameObj ="Дисциплина"
        TypeObj ="C"
           idRef ="2"
        SQL    ="select id as detailed_plan_id, cathedra_own_id,
        teach_subject ||
''-'' || control as teach_subject from
              view_detailed_plan where
              cathedra_id =73 and teach_year_id= $2[0] and group_id
              =$1[0]"
        FillNumber="2"
      >teachSubject</varDecl>
      <varDecl
        type       ="String"
        varType_DBRelation  ="SELECT"
        varType_InputRelation
        ="NEVER_IN_INPUT"
        varType_XLSVisisbility ="VISIBLE"
        fromDBQuery="select position || '' ''
        || fio from
                onpu.teacher_subject_link t, onpu.employee_list e
                where
                det_plan_id = $detailed_plan_id and t.teacher_id =
                e.emp_id and work_id = 1;"

        TypeObj ="C"
           idRef ="3"
        SQL   ="select emp_id as first_teacher_id, position ||
              '' '' || fio from teacher_subject_link t,
              employee_list e where
              det_plan_id =$3[0] and t.teacher_id = e.emp_id"
        FillNumber="1"
      >firstTeacher</varDecl>
       <varDecl
        type="String"
        varType_DBRelation="SELECT"
        varType_InputRelation="NEVER_IN_INPU
        T" varType_XLSVisisbility="VISIBLE"
        fromDBQuery="select position || ''
        '' || fio from
                onpu.teacher_subject_link t,
                onpu.employee_list e where det_plan_id
                = $detailed_plan_id and t.teacher_id =
                e.emp_id and work_id !=1;"
                idObj    ="5"
                NameObj ="Викладач 2"
```

```
     FillNumber="1"

>secondTeacher</varDecl>
<varDecl
type="int"
varType_DBRelation="SELECT"

varType_InputRelation="NEVER_IN_INPUT"
varType_XLSVisisbility="VISIBLE" fromDBQuery="select id
from
onpu.detailed_plan where
id = $detailed_plan_id;"
idObj  ="6"
NameObj ="Номер відомості"
TypeObj ="T"
idRef ="3"

SQL  ="select id,vedomost_number from
view_vedomost_marks_info where
plan_id =$3[0]"
FillNumber="0"
>vedomostNumber</varDecl>
<varDecl
idObj  ="7"

NameObj ="Дата"
TypeObj ="T"
idRef ="3"

SQL  ="select to_char(create_date,''dd.mm.yyyy'') as
create_date from view_vedomost_marks_info where

plan_id =$3[0]"

FillNumber="0"
>Date</varDecl>
<varDecl
idObj  ="8"
NameObj ="Отримати відомість"
TypeObj ="B"

idRef ="3"

SQL  ="select student_id,student_name,mark from
view_vedomost_marks where

plan_id =$3[0]"
FillNumber="NULL"
>GetVedomost</varDecl>
<varDecl

idObj  ="9"
NameObj ="Table1"
TypeObj ="X"
idRef ="8"
SQL  ="$8[*]"
DeleteCol ="1"
```

```
>Table1</varDecl>
<varDecl
idObj  ="10"
NameObj ="Змінити відомість"
TypeObj ="B"
idRef ="10"

SQL  ="update vedomost_marks set mark =$8[3] from
vedomost_marks_info v where

vedomost_marks.id = v.id and v.plan_id = $3[0] and
student_id =$8[0]"

>ChangeVedomost</varDecl>
<varDecl
idObj  ="11"

NameObj ="Видалити відомість"
TypeObj ="B"
idRef     ="3"
SQL  ="delete from
vedomost_marks using vedomost_marks_info v where
vedomost_marks.id = v.id and v.plan_id = $3[0]"
>DeleteVedomost</varDecl>
```

							Форма	С-21
ІФНТУНГ								
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ								
Навчальний рік	$studyYear						Семестр	
Напрям підготовки	$code						Курс	$course
Спеціальність							Группа	$groupName
	Відомість обліку успішності №			$vedomostNumber				
Дисципліна	$teachSubject						$controlType	
Підсумкову оцінку виставив	$firstTeacher							
Поточний контроль здійснив	$secondTeacher							
№ П/П	Прізвище, ініціали студента		№ інд. навч. плану	Підсумкова оцінка за шкалою				Підпис викладача
				Стобальною		Чотири бальн.	ECTS	
$stNum	$studName			$studMark		0	Не определено	

Рис. Б.1. Частина шаблону електронного документу