УДК 681.518:681.327.8

Л.В. Чирун, Є. Я. Лещинський
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДЕРЕВА ПРИЙНЯТТЯ РІШЕНЬ В ІНФОРМАЦІЙНИХ СИСТЕМАХ СКРИНІНГОВОГО СПОСТЕРЕЖЕННЯ ЗА ІМУНОЛОГІЧНИМИ ПАЦІЄНТАМИ

Розглядається задача створення інформаційної системи для предметної області з ієрархічною організацією інформації. Визначається ряд функцій, які система покликана виконувати. Виходячи з поставлених задач, формуються вимоги до форми представлення даних. Значну увагу приділено збереженню ієрархічних даних, їх відтворенню, а також відтворенню їх структури.

Problem of creation of information system with hierarchical data structure is considered in this paper. Some functions, which have to be resolved by system, are defined. Desires to form of data representing based on those functions are made. Proper attention is paid to task of saving hierarchical data, their representation and representation of their structure as well.

Вступ

Інтелектуальний аналіз даних (DM, data mining) є складовою частиною процесу *видобування знань з баз даних* (KDD, knowledge discovery in data bases). Він дає змогу розкрити суть прихованих залежностей в даних, виявити взаємні впливи між властивостями об’єктів, інформація про які зберігається в базах даних, виділити закономірності, властиві певному набору даних. Актуальність проблеми дослідження та опрацювання даних підтверджується широким практичним та комерційним використанням систем інтелектуального аналізу. Найчастіше їх застосовують у науковій сфері та бізнесі.

Зв’язок висвітленої проблеми із важливими науковими та практичними завданнями

Важливе місце у житті людини посідають ієрархічно організовані дані та робота з ними. Структура таких даних характеризується специфікою відношень її елементів, а саме, особливостями наслідування та підпорядкованості між елементами ієрархії. Тому при створенні інформаційної системи для предметної області, що характеризується вказаними особливостями, важливим питанням, якому слід приділити увагу, є питання збереження не лише даних, але й їх структури.

У статті розглянуто аспекти проектування інформаційної системи скринінгового спостереження за імунологічними пацієнтами, що пов’язані з функціональними можливостями системи, а також запропоновано варіанти практичних рішень для розв’язання окремих задач функціонування системи.

Аналіз сучасних досліджень і публікацій

Відносно новітня методологія опрацювання наближених множин пристосована до роботи з недосконалими даними. Теорія наближених множин створена Ж. Павлаком (Z. Pawlak) [1] як математичний інструмент для подолання суперечностей даних і виявлення в даних прихованих закономірностей, або кліше (шаблонів). Фундаментальний принцип алгоритму навчання на шаблонах з використанням наближених множин полягає у виявленні надлишковості серед наявних атрибутів, що описують приклад. Так виявляють сильні атрибути, що впливають на класифікацію прикладу. Надалі вилучають стовпці, атрибути яких не впливають на класифікацію. Для аналізу залишаються лише атрибути, від яких залежить класифікація прикладів таблиці. Множину

атрибутів, що залишилися, називають *редуктом*. Іншими словами, редукт – це підмножина усіх атрибутів таблиці, які забезпечують один і той самий результат класифікації всіх шаблонів таблиці, як і вся множина шаблонів. Знаходження редукта є NP-складною задачею. Однак існують достатньо ефективні методи, які дають змогу розв'язувати цю задачу за прийнятний час. До таких методів належать, зокрема, *логічне виведення* (boolean reasoning) [2], алгоритм Джонсона [3].

Виділення проблем

Життєвий цикл інформації починається з її створення. Отже, як зазначалося в [1], відомості про пацієнта можна отримати і, попередньо систематизувавши, заповнити анкету встановленого взірця. Було здійснено огляд реальної анкети та визначено недоліки, що виникають при її заповненні. Було зазначено, що інформація, що вноситься в анкету, має чітко визначену ієрархічну структуру. Для автоматизації та спрощення процесу заповнення анкети було розроблено такі положення:

1) визначити відношення підпорядкованості одних пунктів анкети іншим (фактично визначити вигляд ієрархії);

2) для обмеження „вільного" введення даних для відповідних пунктів анкети визначити їхні домени.

Було запропоновано варіант практичного розв'язання задачі збирання та внесення інформації шляхом заповнення та збереження відповідного „електронного" аналогу анкети. Ці заходи дали змогу уникнути багатьох незручностей та невизначеностей і автоматизувати процес „перетворення" даних до електронного вигляду.

Зауважимо, що наведена в [1] схема мала декілька „структурних" недоліків, а саме:

1) структура збережених даних мала лінійний вигляд і жодним чином не відображала наявні ієрархічні залежності між відповідними пунктами анкети;

2) незалежно від кількості реально заповнених пунктів анкети для одного пацієнта зберігались значення всіх пунктів (навіть, якщо ці значення були порожніми). Очевидно, що такий підхід є неприпустимий при проектуванні інформаційних систем.

У [2] запропоновано варіант подання ієрархічної інформації з використанням фреймів. У [1] як практичне продовження ідеї, викладеної в [2], було розглянуто механізм опрацювання та збереження даних заповненої анкети, що оснований на використанні фреймів.

Математичний інструмент для подолання суперечностей даних і виявлення в даних прихованих закономірностей побудований на аналізі інфраструктури дерева прийняття рішень в інформаційних системах скринінгового спостереження за імунологічними пацієнтами. Інфраструктура дерева обчислювального пошуку подає схему для відображення проблеми пошуку підмножин як проблему пошуку дерева, дозволяючи визначати правила зменшення безпосередньо з метою зменшення розміру підмножин, що розглядаються. Основна ідея полягає в тому, щоб накласти впорядкування на список атрибутів, а потім перерахувати списки атрибутів за певним порядком. Так, список $U = \{1, 2, 3, 4\}$ з перерахованими атрибутами має таке повністю розширене обчислювальне дерево (рис. 1):

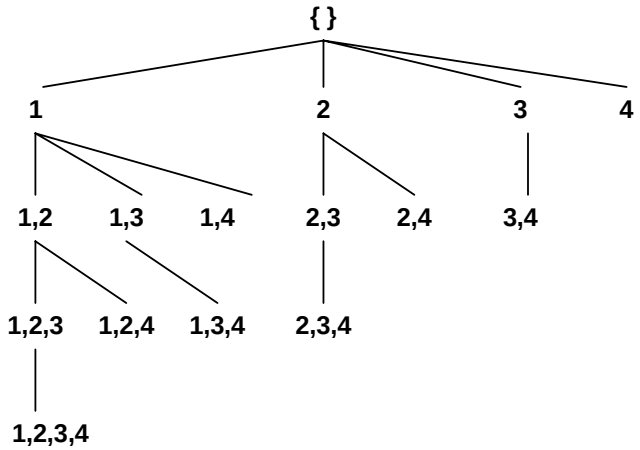

Рис. 1. Дерево пошуку підмножини атрибутів скринінгового спостереження за імунологічними пацієнтами

Дерево пошуку підмножини атрибутів скринінгового спостереження за імунологічними пацієнтами може зображатися графічно з використанням орієнтованих кореневих дерев (рис. 2). Ці дерева називають деревами виведення, або синтаксичного розбору.

Кореню цього дерева відповідає початковий символ (загальна назва списку атрибутів). Внутрішнім вершинам відповідають нетермінальні символи (групи або категорії атрибутів), що зустрічаються у виведенні. Листкам відповідають термінальні символи (атрибути).

Нехай w – назва атрибута і $A \to w$ – продукція, яка використана у виведенні. Тоді вершина, яка відповідає нетермінальному символу A, має синами вершини, які відповідають кожному символу w у порядку зліва направо.

Приклад: Визначити, чи група атрибутів $cbab$ належить списку атрибутів, породженому базою даних атрибутів $G = \{V, T, S, P\}$, де $V = \{a, b, c, A, B, C, S\}$, $T = \{a, b, c\}$, а множина продукцій $P = \{S \to AB,\ A \to Ca,\ B \to Ba,\ B \to Cb,\ B \to b,\ C \to cb,\ C \to b\}$.

Розв'язати цю задачу можна двома способами.

1. Розбір згори донизу.

Оскільки є лише одна продукція з S у лівій частині, то починаємо з $S \Rightarrow AB$.

Далі використаємо продукцію $A \to Ca$.

Отже, маємо

$$S \Rightarrow AB \Rightarrow CaB.$$

Оскільки $cbab$ починається з символів cb, то використовуємо продукцію $C \to cb$, це дасть

$$S \Rightarrow AB \Rightarrow CaB \Rightarrow cbaB.$$

Завершуємо виведення, використавши продукцію $B \to b$:

$$S \Rightarrow AB \Rightarrow CaB \Rightarrow cbaB \Rightarrow cbab.$$

Отже, характеристика пацієнта $cbab$ належить загальному списку атрибутів $L(G)$ скринінгового спостереження за імунологічними пацієнтами.

2. Розбір знизу догори.

Починаємо з рядка, який потрібно вивести: $cbab$.

Можна використати продукцію $C \to cb$, отже, $Cab \Rightarrow cbab$.

Далі використаємо продукцію $A \to Ca$, тоді матимемо

$$Ab \Rightarrow Cab \Rightarrow cbab.$$

Використавши продукцію $B \to b$, отримаємо

$$AB \Rightarrow Ab \Rightarrow Cab \Rightarrow cbab.$$

Нарешті, використаємо продукцію

$S \to AB$: $S \Rightarrow AB \Rightarrow Ab \Rightarrow Cab \Rightarrow cbab$ (рис. 2).

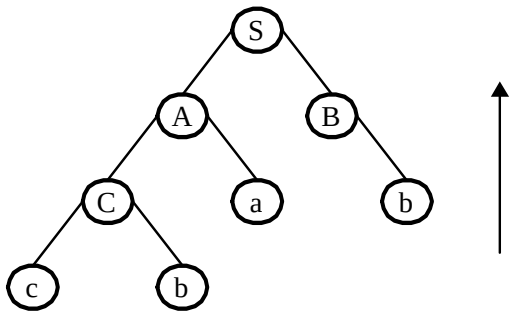

Рис. 2. Дерево виведення для групи атрибутів $cbab$ у базі даних G

Цілі статті

Автори статті мали на меті вирішити такі завдання:

1) визначити функціональні вимоги до проектованої інформаційної системи;

2) виходячи з окреслених функціональних потреб, визначити вигляд ядра інформаційної системи, а саме, у якому вигляді зберігатимуться введені дані;

3) запропонувати практичні рішення щодо окремих задач, що покладаються на інформаційну систему.

Основний матеріал

Визначення функціональних вимог

Введення інформації. Як зазначалося вище, життєвий цикл інформації починається з її створення. Зауважимо, що в анкеті передбачено введення інформації про етап звернення („первинний”, „повторний”). Отже, виділимо такі задачі:

1) введення даних про пацієнта, що обстежується вперше („первинний”);

2) введення даних про пацієнта, що прийшов на повторне обстеження („повторний”). До речі, таких повторних звернень може бути декілька.

У першому випадку необхідно заповнити анкету встановленого взірця. У другому — те саме, але можна передбачити, що „повторна” анкета не зазнає значних змін, тому б було зручно у повторній анкеті лише відкоригувати дані, внесені до первинної анкети. Отже, при внесенні „повторного” пацієнта було б зручно знайти відповідну первинну анкету, зробити її копію та відредагувати останню.

Отже, введені дані повинні зберігатися у вигляді, зручному для:

1) пошуку потрібної первинної (або останньої заповненої) анкети;

2) відображення введених даних у бланку повторної анкети.

Середовище розробки. Реалізація прототипу інформаційної системи була здійснена за допомогою Web-інтерфейсу [1]. Нагадаємо, що в цьому випадку після заповнення анкети при натисненні кнопки „Зберегти” ініціюється процес оформлення введених даних у спеціальний об’єкт REQUEST та передача його на опрацювання серверу системи. Об’єкт REQUEST при цьому має вигляд хеш-таблиці [1].

Отже, вигляд даних, що зберігаються, має бути таким, щоб анкети зручно формувалися, просто обробляючи дані відповідної хеш-таблиці.

Облік інформації. З появою засобів для введення даних та їх зберігання система починає виконувати функції бази даних. Виникає потреба обліку даних, природна для баз даних, наприклад:

1) пошук карток пацієнтів за вказаними параметрами (можливість здійснити вибірку);

2) видалення та редагування заповнених анкет;

3) створення та збереження резервної копії анкети;

4) відтворення збережених даних у зручному для користувача вигляді;

5) отримання „твердої” копії картки пацієнта;

6) архівація даних.

Аналіз та статистика. Оскільки система проектується як система скринінгового спостереження, то вигляд даних має давати змогу легко знаходити потрібну інформацію. Це може використовуватись як для здійснення вибірок за заданими критеріями (для проведення наукових досліджень), так і підготовки звітів потрібної форми.

Здійснення трансформацій. Сьогодні класифікація захворювань визначена у документі про міжнародну класифікацію хвороб МКХ-10 [1]. Очевидно, що існує можливість перегляду і зміни існуючих положень у цій класифікації. Це призведе до зміни форми, в якій дані необхідно отримати (хоча зміст інформації може не зазнати змін). Логічним є припущення, що збережені дані також мають бути приведені до нової форми відповідно до вимог. Обґрунтуємо важливість цього пункту на прикладі.

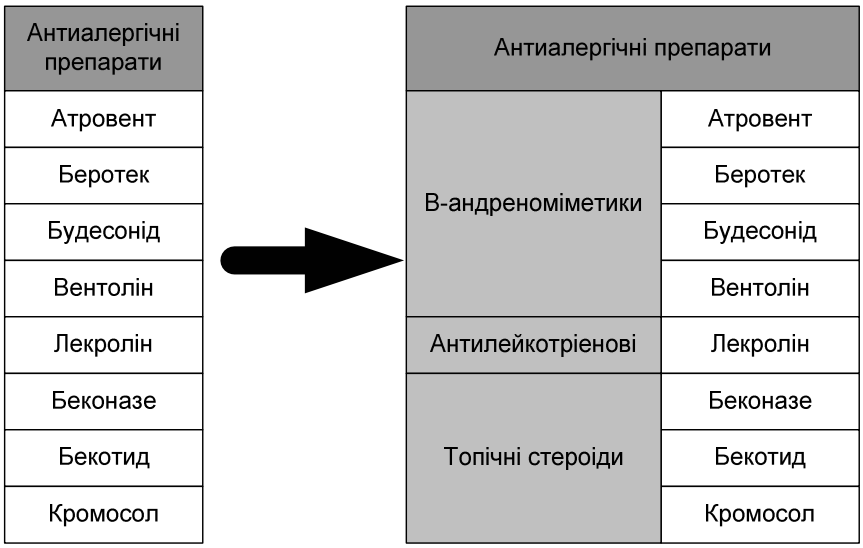

Рис. 3. Приклад здійснення трансформацій

Рис. 3 ілюструє необхідність здійснення різноманітних перетворень у структурі анкети. Другий стовпець у лівій частині містить перелік деяких препаратів, і, припустимо, з'являється необхідність ввести додаткову класифікацію цих препаратів, що призводить до появи додаткового стовпця у правій частині. Якщо пацієнту було призначено певні препарати і відповідні дані було внесено в анкету до здійснення структурних перетворень, то, очевидно, після трансформації анкети збережені дані мають бути модифіковані і приведені до відповідного вигляду.

Проілюструємо попередній випадок за допомогою дерева (див. рис. 4).

Отже, маючи вже заповнені анкети, необхідно переглянути їх і модифікувати ієрархічні залежності між даними. Очевидно, форма даних має давати змогу здійснювати це швидко і зручно.

Підсумовуючи, можемо сказати, що форма представлення даних у проектованій системі має надавати змогу ефективно та зручно здійснювати такі операції:

1) формування (створення) даних шляхом опрацювання хеш-таблиці;
2) здійснення пошуку для проведення аналізу та отримання статистики;
3) здійснення трансформацій;
4) інтерпретація введених даних з метою:

a) отримання „твердої" копії даних у зручному вигляді;
b) відображення даних з метою редагування;
c) відображення даних з метою введення „повторного" пацієнта.

Метод зберігання ієрархічних даних з використанням N-арного дерева

Розглянемо детальніше процес зберігання даних, який був реалізований до цього часу. Нехай на рис. 5 наведена форма, в якій необхідно отримати інформацію щодо наявності у пацієнта скерування на обстеження.

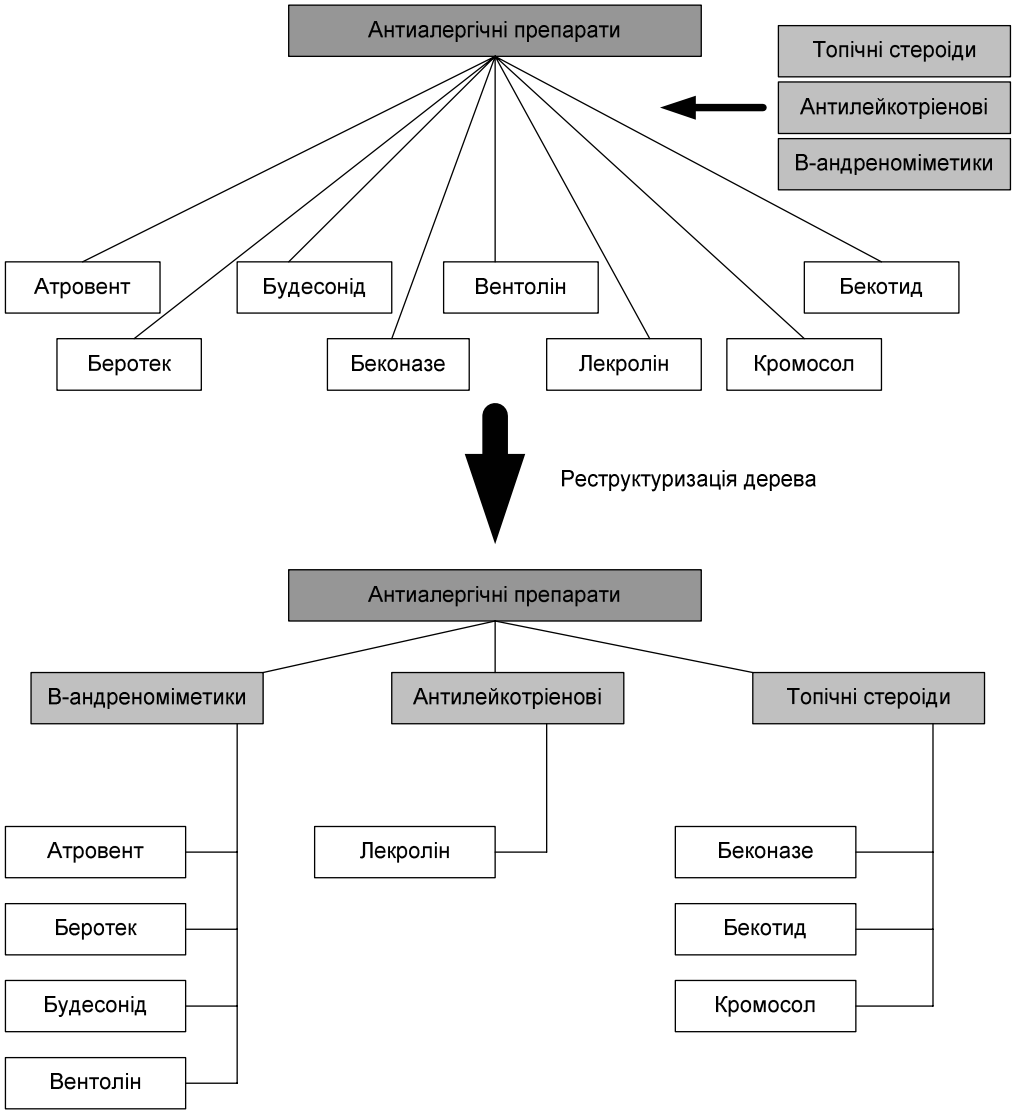

Рис. 4. Приклад здійснення трансформацій у структурі дерева

Скерування	є	Установа	дані
		Лікар	дані
		Діагноз при скеруванні	є
			немає
	немає		

Рис. 5. Форма, в якій необхідно отримати інформацію щодо наявності скерування

На рис. 6 вказано назви відповідних візуальних елементів керування форми.

Скерування	APPOINTMENT_7
Установа	INSTITUTION
Лікар	DOCTOR
Діагноз	DIAGNOSIS

Рис. 6. Назви відповідних візуальних елементів керування форми, що відповідає формі на рис. 5

За принципами об’єктно-орієнтованого програмування було створено клас “PATIENT”, атрибутами якого були усі передбачені в анкеті дані (див. рис. 7).

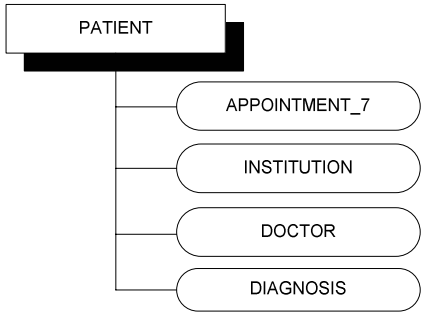

Рис. 7. Структура класу “PATIENT”

Як вже було зазначено, структура класу “PATIENT” є лінійною і жодним чином не відображає ієрархічної залежності, що існує між окремими пунктами анкети.

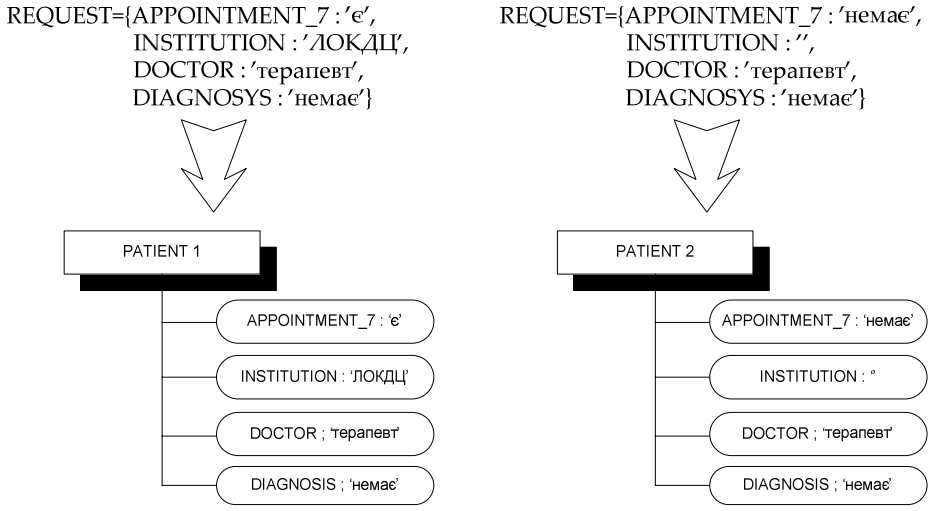

Рис. 8. Процес збереження даних

На рис. 8 зображено процес зберігання даних для двох різних пацієнтів. В обидвох випадках при спробі зберегти дані заповненої анкети вони формуються в об’єкт REQUEST і передаються для опрацювання серверу, після чого створюються фізичні об’єкти класу PATIENT і, оскільки структура класу є статичною, в об’єктах “PATIENT 1” та “PATIENT 2” зберігаються значення усіх властивостей REQUEST. Для першого пацієнта значення пункту „Скерування” (“APPOINTMENT_7”) позначено як „є”, отже, зберігати ієрархічно підпорядковані йому пункти INSTITUTION, DOCTOR та DIAGNOSYS є доцільним. Для другого пацієнта значення пункту „Скерування” позначено як „немає”, отже, збереження решти пунктів не є доцільним, але в силу статичності структури класу “PATIENT” їх значення зберігаються (навіть, коли вони є порожніми).

<table>
<tr><td>Дата обстеження</td><td>дані</td><td colspan="3"></td></tr>
<tr><td>Пацієнт</td><td>дані</td><td colspan="3"></td></tr>
<tr><td rowspan="6">Скерування</td><td rowspan="5">є</td><td>Установа</td><td>дані</td><td></td></tr>
<tr><td rowspan="3">Лікар</td><td>алерголог</td><td></td></tr>
<tr><td>терапевт</td><td></td></tr>
<tr><td>інше</td><td>дані</td></tr>
<tr><td>Діагноз при скеруванні</td><td>є</td><td></td></tr>
<tr><td>немає</td><td colspan="3"></td></tr>
</table>

Рис. 9. Фрагмент анкети

У [1] було запропоновано механізм збереження даних анкети за допомогою фрейма, проте недоліком такого варіанта є необхідність переглядати усі слоти фрейма, незалежно від того, потрібні були дані цього слота чи ні.

Розглянемо механізм збереження введених даних з використанням N-арного дерева. Для прикладу розглянемо фрагмент анкети, наведений на рис. 9.

Зобразимо тепер цей фрагмент у вигляді N-арного дерева (див. рис. 10).

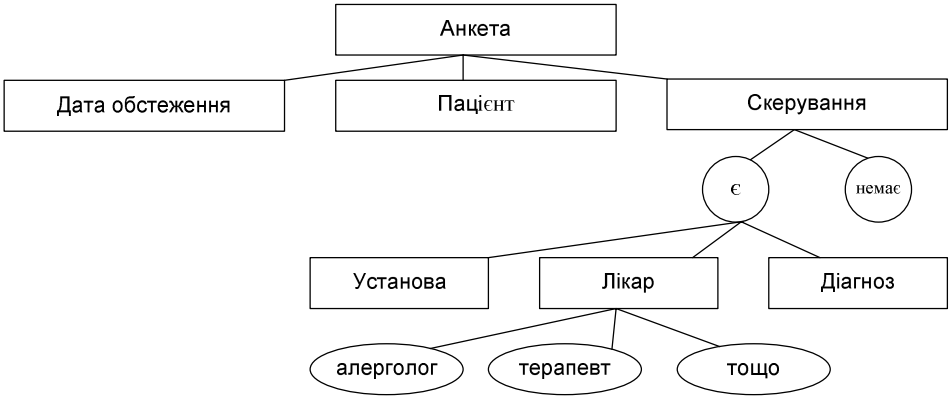

Рис. 10. Представлення анкети з рис. 9 у вигляді N-арного дерева

З рис. 11 видно, що певні значення окремих пунктів анкети дають змогу опуститися на наступний рівень ієрархії. Для пункту „Скерування” таким значенням є „є”, а для пункту „Лікар” — „інше”. Припустимо тепер, що в анкеті було набрано деякі дані, і відповідні об’єкти REQUEST було надіслано на опрацювання серверу.

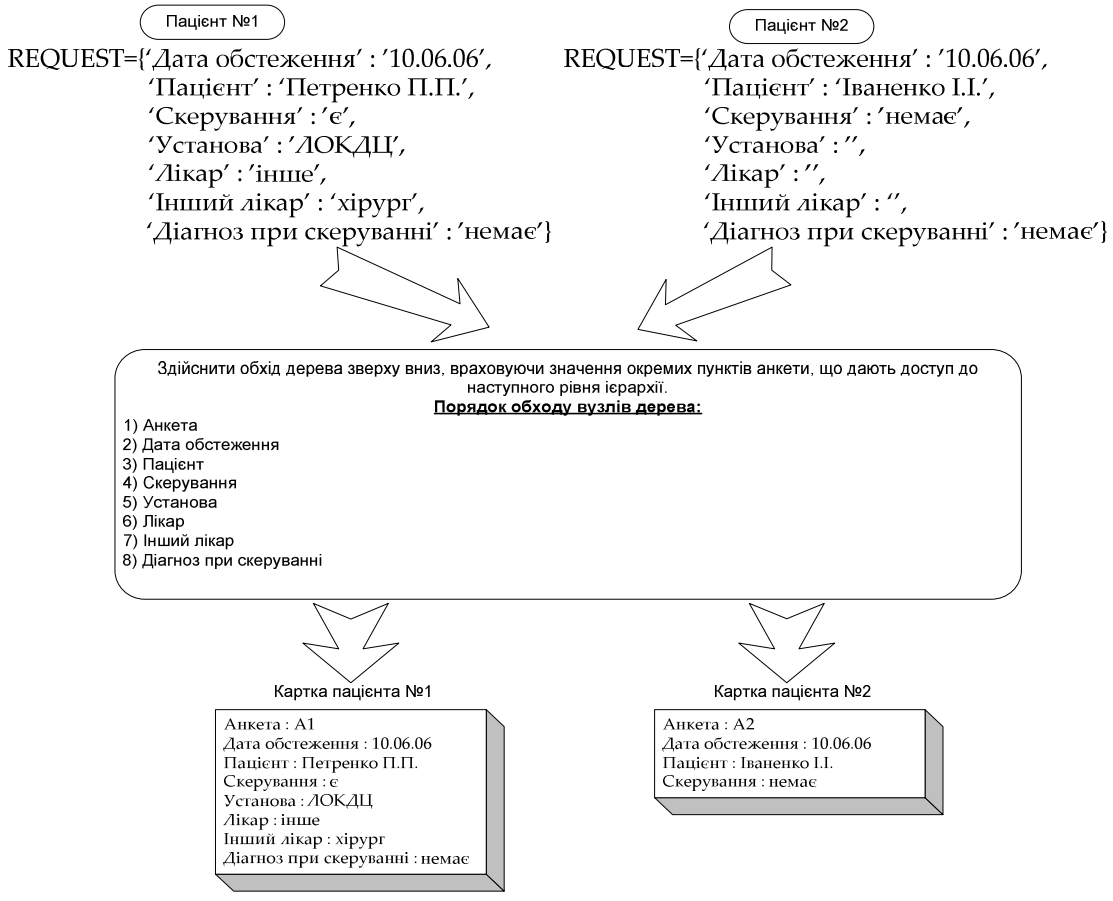

Рис. 11. Процес опрацювання об’єкта REQUEST за допомогою дерева

Як видно з рис. 11, обхід дерева виконується згори донизу, що відповідає ієрархічній структурі анкети. При цьому для окремих вузлів на наступний рівень ієрархії можна перейти лише при певному визначеному значенні батьківського вузла. Так, для першого пацієнта значення пункту „Скерування” визначене як „є”, отже, відбувається перехід на нижчий рівень і до розгляду беруться вузли „Установа”, „Лікар” та „Діагноз при скеруванні”. Оскільки для вузла „Лікар” задане значення „інше” — на підпорядкованому рівні розглядається значення пункту „Інший лікар”. Для другого пацієнта значення „немає” для пункту „Скерування” не дає доступу до наступного рівня дерева, отже, підпорядковані вузли взагалі не розглядаються. Тобто, для другого пацієнта в картці зберігатимуться лише ті дані, які мають зміст. Зауважимо, що при такому підході не лише економиться місце, але й взагалі відпадає необхідність розгляду „непотрібних” даних.

Отже, загальна ідея полягає в тому, що при опрацюванні кожної набраної анкети необхідно здійснювати обхід згори донизу заданого дерева, враховуючи „перепускні” значення для відповідних вузлів дерева. Виникає запитання: як побудувати таке дерево? Використаємо з цією метою фрейм. Для розглянутого вище прикладу фрейм матиме вигляд (див. рис. 12):

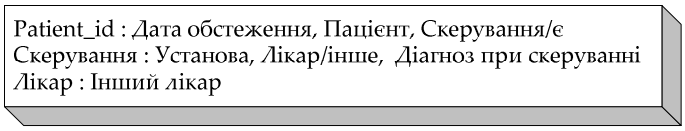

Рис. 12. Вигляд фрейму для анкети з рис. 9

Як видно з рисунка, фрейм – це фактично двовимірний список, в якому задано структуру дерева і „перепускні” значення для відповідних пунктів анкети. У кожному рядку фрейма перший елемент позначає батьківський вузол, а всі наступні елементи в рядку —вузли-нащадки.

Дерево потрібної структури будується шляхом послідовного перегляду відповідного фрейму. При цьому для першого елемента („Скерування”, „Лікар”) кожного рядка фрейму здійснюється пошук вже побудованого вузла, що відповідає цьому елементу, після чого для знайденого батьківського вузла будуються вузли-нащадки. Зауважимо, що в процесі побудови потрібно зберігати „перепускні” значення для елементів, в яких воно визначене.

Отже, схема збереження лише потрібних даних стає доволі зручною та прозорою. Послідовність цього процесу наведена на рис. 13.

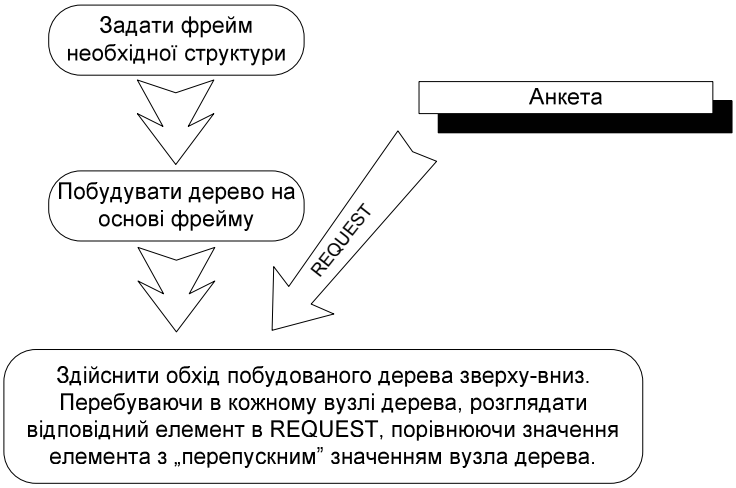

Рис. 13. Схема збереження даних

Щодо вигляду, в якому дані зберігатимуться на даний момент, скажемо: здійснюючи обхід дерева, виписуємо кожну розглянуту пару „елемент — значення елемента” в окремий рядок (подібно до того, як зображено на рис. 10).

Досі ми розглядали випадок, коли пункт анкети мав лише одне значення, яке давало змогу перейти на наступний рівень ієрархії. Проте можуть виникати ситуації, в яких декілька значень (або навіть кожне) призводять до переходу на нижчий рівень. Для прикладу розглянемо фрагмент анкети, зображений на рис. 14.

<table>
<tr><td rowspan="3">За етіологією</td><td rowspan="2">відомі</td><td>природжений (антенатальний)</td><td rowspan="2">інфекційні</td><td>ВІЛ/СНІД</td></tr>
<tr><td>набутий (постнатальний)</td><td>герпес віруси I, II (HerSVI,II)</td></tr>
<tr><td>невідомі</td><td></td><td></td></tr>
</table>

Рис. 14. Фрагмент анкети, що ілюструє „багатозначність” деяких пунктів анкети

Зауважимо, що значення „природжений” і „набутий” є взаємовиключними, причому вибір кожного з них призводить до переходу на наступний рівень ієрархії („інфекційні”). В даній ситуації рівень „інфекційні” та інші підпорядковані рівні є однаковими для двох значень, проте вони можуть бути і різними. Розглянемо, як працює попередньо описаний механізм збереження даних у цьому випадку. Але дещо модифікуємо фрейм, оскільки тепер деякі пункти анкети мають більше ніж одне „перепускне” значення.

Ідея задання структури дерева за допомогою фрейма залишилась, проте тепер вузол дерева додатково ідентифікується „перепускним” значенням.

Процес збереження даних відбувається за вже розглянутою схемою (див. рис. 14), проте можна зауважити, що значення елемента „ETIOLG_KNOW” зберігається в картці двічі (саме два взаємовиключних значення передбачені для цього пункту анкети). Це відбувається з такої причини: при обході дерева для першого пацієнта, перебуваючи у вузлі „ETIOLG_KNOW/антенатальний”, в REQUEST шукається ключ „ETIOLG_KNOW”, береться його значення („антенатальний”), порівняння останнього з перепускним значенням вузла („антенатальний”) дає позитивний результат і відбувається перехід на наступний рівень ієрархії; проте при подальшому обході, перебуваючи у вузлі „ETIOLG_KNOW/постнатальний”, знову в REQUEST шукається ключ „ETIOLG_KNOW”, береться його значення (знову ж таки „антенатальний”), порівняння останнього з перепускним значенням вузла („постнатальний”) дає негативний результат, переходу на наступний рівень не відбувається, але оскільки до вузла „ETIOLG_KNOW/постнатальний” заходили, це фіксується у картці пацієнта у вигляді ще одного рядка „ETIOLG_KNOW: антенатальний”. Аналогічна ситуація відбувається з другим пацієнтом, проте послідовність рядків у картці дещо інша, ніж в картці першого пацієнта (внаслідок обходу дерева). Виявлений недолік можна виправити, якщо перед обробкою даних анкети видалити з дерева непотрібну (непотрібні) гілки з коренем у відповідному „багатозначному” вузлі. Цей процес проілюстровано на рис. 15.

Інтерпретація збережених даних

Вище було визначено, що відображення введених даних потрібно для декількох цілей. Розглянемо процес інтерпретації даних з метою отримання „твердої” копії картки пацієнта. Зрозуміло, що таку копію можна отримати, роздрукувавши на папері відповідний текстовий файл. На цей момент дані про пацієнта зберігаються у вигляді послідовності рядків пар „елемент — значення елемента” і, очевидно, що така форма даних ніяк не відображає ієрархічних залежностей, які визначені між пунктами анкети. Прийнятним виглядом для друку могла б бути, наприклад, форма, зображена на рис. 16. Неважко помітити, що виписана послідовність рядків відповідає послідовності обходу відповідного дерева згори донизу.

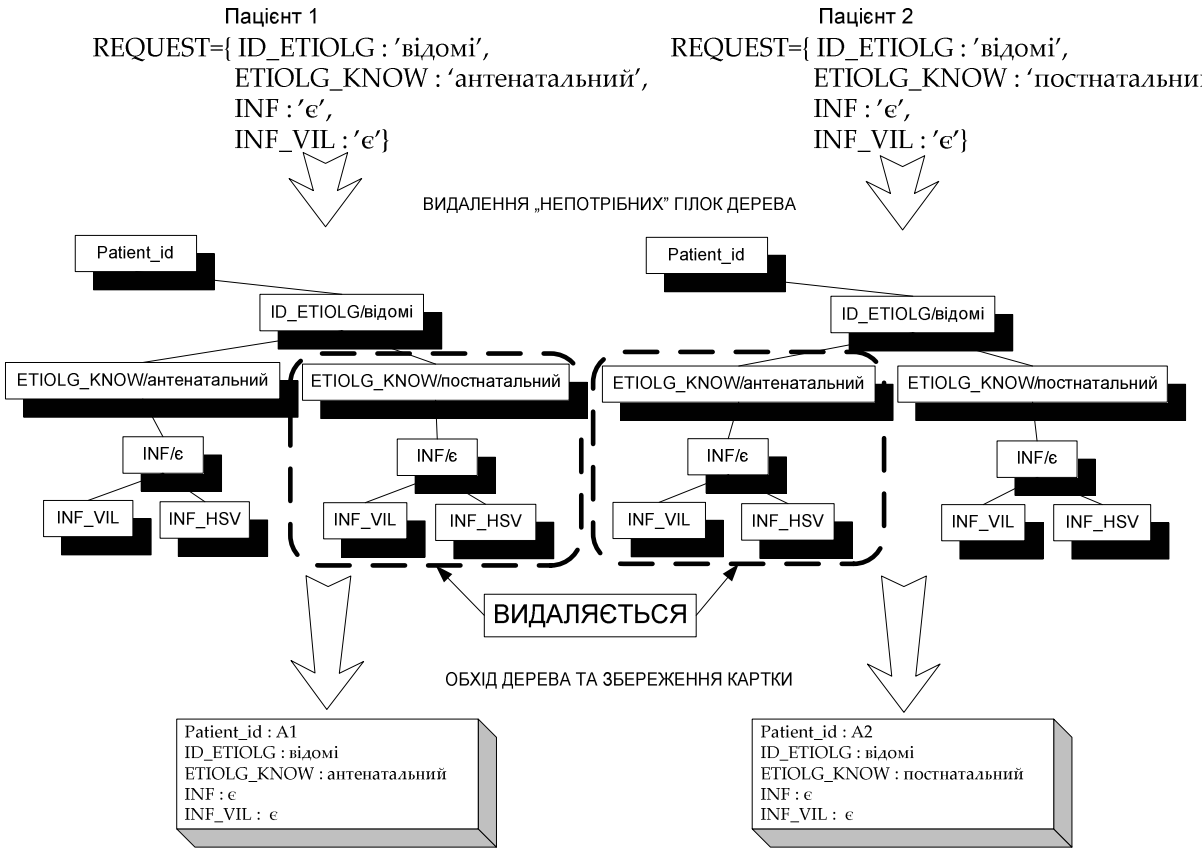

Рис. 15. Видалення непотрібної гілки дерева з коренем у „багатозначному вузлі"

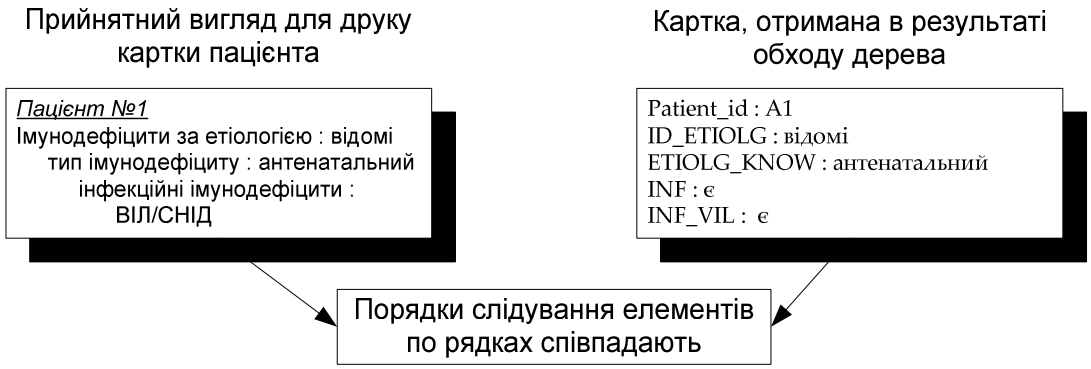

Рис. 16. Відповідність послідовності обходу дерева „згори донизу" послідовності відображення даних

Отже, картку для друку можна отримати зі збереженої картки шляхом додавання відступів з лівого боку. Кількість відступів визначається заданим ієрархічним рівнем елемента. Загальну схему цього процесу наведено на рис. 17.

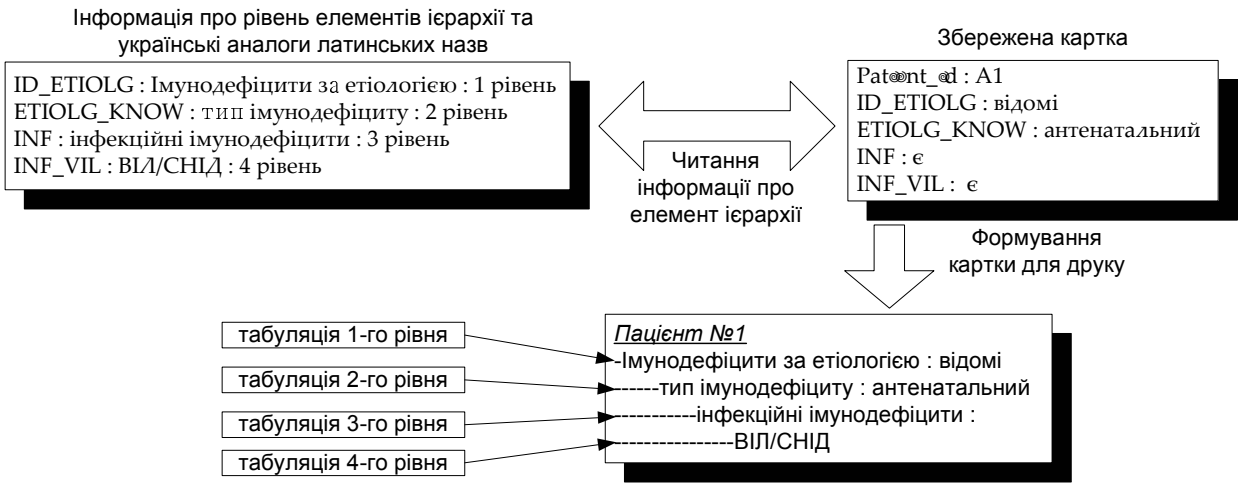

Рис. 17. Отримання картки для друку шляхом додавання табуляції до збереженої картки

Висновки

Для проектованої інформаційної системи було визначено необхідні виконувані функції, на основі яких було сформовано функціональні вимоги до ядра системи і поставлено питання: у якому вигляді повинні зберігатися дані, щоби система ефективно вирішувала необхідні завдання. Основними невирішеними проблемами до цього моменту були такі: як зберігати лише потрібні дані і як зберегти інформацію про ієрархічну структуру цих даних? Для розв'язання першої задачі було запропоновано підхід, оснований на використанні n-арних дерев та фреймів, а також було розглянуто деякі специфічні ситуації, що виникають при цьому. Щодо другої задачі, то вона була розв'язана частково на прикладі отримання „твердої копії" картки пацієнта. На практичному рівні залишаються нерозглянутими проблеми редагування ієрархічних даних та здійснення трансформацій. Отже, можна зробити висновок, що описані у статті підходи є загальними і їх можна використовувати до будь-яких предметних областей з ієрархічною організацією інформації.

1. Лещинський Є.Я. Моделювання процесу збирання та подання ієрархічно організованої інформації на прикладі спостережень за імунологічними пацієнтами // Вісник Національного університету „Львівська політехніка". — 2004. — № 519. — С. 214–224. 2. Нікольський Ю.В. Моделі даних у структурах, що трансформуються // Вісник Національного університету „Львівська політехніка". — 2004. — № 519. — С. 233–243. 3. Стив Спикльмайр, Кевин Фридли, Джерри Спикльмайр, Ким Брэнд. Zope. Разработка Web-приложений и управление контентом: Пер. с англ. — М.: ДМК Пресс, 2003. — 464с. 4. Лесса Андре. Python. Руководство разработчика: Пер. с англ. /Андре Лесса — СПб.: ООО «ДиаСофтЮП», 2001. — 688 с. 4. http://www.zope.org. 5. http://www.python.org.