Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод фрактального аналізу реляційних баз даних

(тема)

Виконав:
студент II курсу, групи КСМзм-19-1
Міллер П.П.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Міллеру Паулу Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод фрактального аналізу реляційних баз даних

затверджена наказом по університету від " 23 " жовтня 2020 р. № 168 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи Бази даних «Деталь» та «Єдиний державний реєстр земельних ділянок»

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз предметної області;
Розробка алгоритму виділення доменів;
Критерії пошуку доменів;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайди презентації – 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд методів предметної області та	27.10.20-09.11.20	
	постановка задачі		
2	Вибір та обгрунтування загальних критерів	10.11.20-17.11.20	
	пошуку доменів		
3	Вибір інструментальних засобів	18.11.20-23.11.20	
4	Проведення експериментів	24.11.20-01.12.20	
5	Оформлення матеріалів атестаційної роботи	02.12.20-07.12.20	
6	Подання атестаційної роботи керівникові та	08.12.20-09.12.20	
	її попередній захист		
7	Подання атестаційної роботи на рецензування	10.12.20-11.12.20	

Дата видачі завдання 26 жовтня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 78 с., 22 рис., 7 табл., 1 дод., 18 джерел.

СКБД, FRACTAL DATA MINING, ФРАКТАЛ, ТАБЛИЦІ, ДОМЕНИ.

Основною метою є розгляд реляційних таблиць з точки зору теорії фракталів та застосування методів фрактального аналізу до реляційних баз даних.

Для досягнення даної мети необхідно вирішити такі завдання:

- вивчити наявні методи фрактального стиснення зображень;
- розглянути методи Fractal Data Mining;
- проаналізувати можливості застосування фрактального аналізу до реляційних баз даних;
- розробити алгоритм пошуку доменів в реляційних таблицях;
- оцінити можливість застосування отриманих результатів до: пошуку функціональних залежностей, стиску таблиць, проведення кластеризації.

ABSTRACT

Master’s thesis: 78 pages, 22 figures, 7 tables, 1 appendices, 18 sources.

DBMS, FRACTAL DATA MINING, FRACTAL, TABLES, DOMAIN.

The major goal of this thesis is consideration of relational tables from the point of view of fractal theory and application of fractal analysis methods to relational databases.

To achieve this goal it is necessary to solve the following tasks:

- to study the available methods of fractal compression of images;
- consider the methods of Fractal Data Mining;
- to analyze the possibilities of applying fractal analysis to relational databases;
- develop a domain search algorithm in relational tables;
- evaluate the possibility of applying the results to: search for functional dependencies, compression of tables, clustering.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АРМ – автоматизоване робоче місце

ІАЦ – інформаційно-аналітичний центр

СУБД – система управління базами даних

SQL – Structurized Query Language

XML – eXtended Markup Language

ВСТУП

Корпоративна база даних будь-якого сучасного підприємства зазвичай містить набір таблиць, що зберігають величезну кількість записів про ті чи інші факти або об'єкти. Як правило, кожен запис в подібній таблиці описує якийсь конкретний об'єкт або факт і, як правило, структури цих записів ідентичні. У зв'язку з тим, що сучасні бази даних містять величезну кількість даних, які необхідно не тільки компактно зберігати, а й аналізувати, здійснювати пошук, розробляються різні підходи до стиснення і аналізу інформації. Так, наприклад технологія Data Mining заснована на статистичних методах, служить для виявлення в «сирих» даних раніше невідомих, нетривіальних, практично корисних і доступних інтерпретацій знань, необхідних для прийняття рішень.

Для вилучення необхідної інформації розробляються різноманітні теорії і алгоритми. Однією з подібних теорія є фрактальний аналіз.

« Всі фігури, які я досліджував і називав фракталами, в моєму уявленні мали властивість бути нерегулярними, але самоподібними», - писав Бенуа Мандельброт, який в 1975 році ввів термін фрактал (від латинського fractus - дробовий). В даний час дана теорія розвивається і застосовується не тільки в комп'ютерній графіці, але і в архіваторах, в системах аналізу даних.

Основний принцип застосування теорії фракталів - це пошук простих частин, подібних цілому, застосувавши до яких певну ітераційну функцію, зможемо отримати всю систему. Виділення так званих самоподібних частин називають фрактальним аналізом.

Застосування теорії фракталів до графічної інформації цілком природно і зрозуміло, однак можна розглянути з точки зору даної теорії і системи баз даних.

Сукупність великої кількості записів таблиць, може стати джерелом додаткової, набагато більш цінної інформації, яку не можна отримати на

основі одного конкретного запису, а так же зайняти величезну кількість місця на накопичувачах інформації. Цілком можливо, що проаналізована інформація може бути самоподібна і може відображати певну залежність не тільки між записами таблиць, але і всередині самого запису.

Подібного роду інформація зазвичай використовується при прогнозуванні, плануванні, аналізі, і цінність її дуже висока, тому виявлення структури даних - ключовий аспект ефективного представлення і зберігання цих даних.

Основною метою є розгляд реляційних таблиць з точки зору теорії фракталів та застосування методів фрактального аналізу до реляційних баз даних.

Для досягнення даної мети необхідно вирішити такі завдання:

- вивчити наявні методи фрактального стиснення зображень;
- розглянути методи Fractal Data Mining;
- проаналізувати можливості застосування фрактального аналізу до реляційних баз даних;
- розробити алгоритм пошуку доменів в реляційних таблицях;
- оцінити можливість застосування отриманих результатів до: пошуку функціональних залежностей, стиску таблиць, проведення кластеризації.

1 ОСНОВНІ ТЕРМІНИ Й ПОНЯТТЯ ТЕОРІЯ ФРАКТАЛІВ

1.1 Загальні відомості про фрактали

Поняття "фрактал" було введено Бенуа Мандельброт в 1975 році. Про своєї теорії в книзі «Фрактальна геометрія природи» він відгукнувся так: «Ризикнувши відповісти на виклик, я задумав і розробив нову геометрію природи, а також знайшов для неї застосування в багатьох різноманітних областях. Нова геометрія здатна описати багато з неправильних і фрагментованих форм в навколишньому світі і породити цілком закінчені теорії, визначивши сімейство фігур, які я називаю фракталами. Найбільш корисні фрактали включають в себе елемент випадковості; як правильність, так і неправильність їх підпорядкування статистичним законам. Крім того, описувані тут фігури прагнуть до масштабної інваріантності, тобто ступінь їх неправильності і / або фрагментації незмінна в масштабах »[14]

Фрактал, у визначенні Мандельброта, це структура, що складається з частин, які в якомусь сенсі подібні цілому. Таке визначення дозволяє охопити найбільш широку множену об'єктів, які підпадають під поняття фрактал. Таким чином, самоподоба є одним з основних ознак фракталів, тобто невелика частина містить інформацію про всю систему. Варто відзначити і той факт, що строгого визначення терміна «фрактал» не існує.

Математичні фрактали поділяються на класи.

1. Геометричні фрактали. Фрактали цього класу - самі наочні, тому що в них відразу видно самоподобу. У двомірному випадку такі фрактали можна отримати, задавши деяку ламану, яка називається генератором. За один крок алгоритму кожен з відрізків, що становлять ламану, замінюється на ламану-генератор, у відповідному масштабі. В результаті нескінченного повторення цієї процедури (а точніше, при переході до межі) виходить фрак- 10 тальна крива. При видимій складності отриманої кривої, її загальний вигляд

задається тільки формою генератора. Прикладами таких кривих служать: крива дракона, крива Коха, крива Леві, крива Маньківського, крива Пеано;

2. Алгебраїчні фрактали. Для побудови алгебраїчних фракталів використовуються ітерації нелінійних відображень, що задаються простими алгебраїчними формулами. Найбільш вивчений двомірний випадок. Нелінійні динамічні системи можуть володіти декількома стійкими станами. Кожне стійкий стан (аттрактор) володіє деякою областю початкових станів, при яких система обов'язково в нього перейде. Таким чином, фазовий простір розбивається на області тяжіння аттракторов. Якщо фазовим є двомірний простір, то, фарбуючи області тяжіння різними кольорами, можна отримати колірний фазовий портрет цієї системи (ітераційного процесу). Приклади алгебраїчних фракталів: множина Мандельброта, множина Жюліа, басейни Ньютона, біоморфи;

3. Стохастичні фрактали. Ще одним відомим класом фракталів є стохастичні фрактали, які виходять в тому випадку, якщо в ітераційному процесі випадковим чином змінювати будь-які його параметри. При цьому виходять об'єкти дуже схожі на природні - несиметричні дерева, порізані берегові лінії і т.д. Двовимірні стохастичні фрактали використовуються при моделюванні рельєфу місцевості і поверхні моря.

Даними трьома класами фрактали не обмежуються. Потенційно найбільш корисним видом фракталів є фрактали на основі систем ітеративних функція (Iterated Function System - IFS). Метод IFS стосовно побудови фрактальних зображень, винайдений Майклом Барнслі і його колегами з Технологічного інституту штату Джорджія, базується на самоподобу елементів зображення і полягає в моделюванні малюнка кількома меншими фрагментами його самого. Спеціальні рівняння дозволяють переносити, повертати і змінювати масштаб ділянок зображення; таким чином, ці ділянки служать компонувальними блоками іншої частини картини.

У кожному класі фрактальних множин частина, яка є подібною цілому і містить інформацію про систему, називається по-різному. У геометричних фракталах - генератор, алгебраїчних - аттрактор, в IFS-фрактали - деякий параметр системи функцій. У даній роботі будемо називати доменом - частина об'єкта, яка є подібною до цілого об'єкта.

1.2. Фрактальний алгоритм стиснення зображень

1.2.1. Метрика Хаусдорфа

Для початку необхідно визначити простір, на якому будемо розглядати фрактали. Це простір називається «простір Хаусдорфа».

Спочатку визначимо метрику Хаусдорфа. Нехай *(X, d)* - повний метричний простір. Визначимо *H (X)* як простір, що складається з компактних підмножин *X*. Множена *C* в метричному просторі *(X, d)* називається компактною, якщо кожна нескінченна послідовність з *C* має сходимість в *C* підпослідовність. Таким чином, кожна точка *H (X)* - це компактне підмножина з *X*. Визначимо відстань між $x \in X$ і $B \in H(X)$ як найкоротша відстань між точкою x і довільної точкою $y \in B$:

$$d(x,B) = \min\{d(x, y) : y \notin B\}. \quad (1.1)$$

Зауважимо, що цей мінімум існує і кінцевий, так як *B* компактна і, отже, замкнута і обмежена. Тепер можемо визначити відстань між двома компактними множинами *A* і *B* як:

$$d(A,B) = max\left(d(x,B) : x \in A\right). \quad (1.2)$$

Компактність *A* забезпечує існування і кінечність цього максимуму.

Ясно, що в загальному випадку d (A, B) ≠ d (B, A) рис.1.1 то можемо поправити це, визначивши нову міру відстані h (A, B):

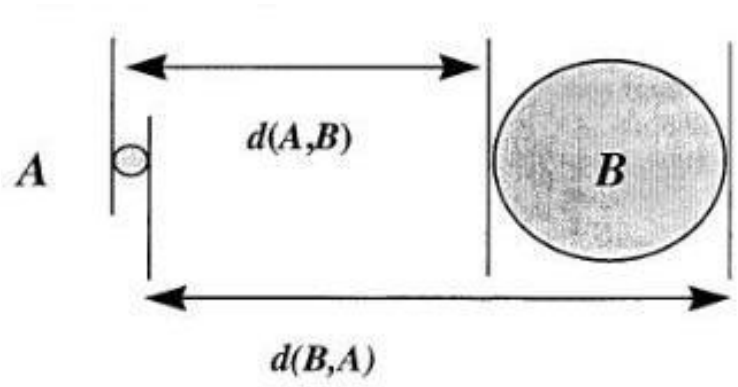

Рисунок 1.1 – Загальний випадок d(A,B) ≠ d(B, A)

Тепер $h\ (A, B) = h\ (B, A)$ і h є метрикою в $H\ (X)$. Метрика h називається метрикою Хаусдорфа, а метричної простір $(H\ (X), h)$ - метричних простором Хаусдорфа. Барнслі (1993) назвав $(H\ (X), h)$ «простором, де мешкають фрактали» [18].

1.2.2. Математичні основи фрактального стиснення.

Спочатку дамо визначення стискає перетворення. Перетворення $f: X \to X$ в метричному просторі (X, d) називається стискаючим відображенням, якщо існує константа s, 0 ≤ s < 1 така, що

$$d\ (f\ ((x_1), f\ (x_2) \leq\ sd(x_1, x_2)). \qquad (1.3)$$

Введемо визначення системи ітераційних функцій. Нехай { w_1, w_2, …, w_N} - кінцевий набір стискаючих відображень в (X, d) з коефіцієнтами стиснення $s_1, s_2, s_N, 0 \leq s < 1$. Визначимо відображення W, що впливає на компактні множени точок з X (тобто в просторі H (X)), в такий спосіб:

$$W(B) = w_1(B) \cup w2(B)...w_N(B) = \bigcup_{n=1}^{N} w_n(B) \quad (1.4)$$

Таким чином, *W* відображає *H (X)* в $_{H(X)}$ і є стискаючим відображенням на *(H (X), h)* з коефіцієнтами стиснення s, де $s = max\{s_1, s_2, ..., s_N\}$.

$$h(W(B),W(C)) \le sh(B,C), B,C \in H(X) \quad (1.5)$$

Система ітераційних функцій (IFS) складається з повного метричного простору *(X, d)* і кінцевого множени стискаючих відображень з коефіцієнтами стиснення s_N. Коефіцієнт стиснення IFS визначається як $s = max\{s_1, s_2, ..., s_N\}$. Введемо позначення для IFS: $\{X, w\ n: n = 1,2, ..., N\}$.

Як приклад можна розглянути перетворення, які використовував Барнслі для його IFS - це так звані афінні перетворення. Афінний перетворення

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}, \quad (1.5)$$

де a,b,c, d,e, f $\subset$ R . Афінний перетворення можуть здійснювати поворот, переміщення і масштабування рис 1.2. Визначимо матрицю

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad (1.6)$$

як матричну частину *T*. Якщо *S* - це множина точок в R^2, то площа перетвореної області *T (S)* в | det *A* | раз відрізняється від площі області *S*, де | *det A* | - це абсолютне значення визначника *A*. Таким чином T - це просторове стиснення і, отже, стиснення в H (X), якщо | det A | < 1 [18].

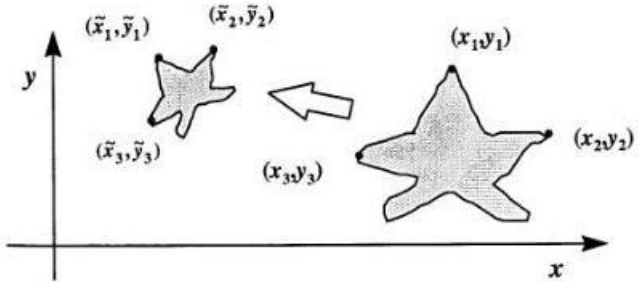

Рисунок 1.2 – Афінний перетворення множини точок в R^2

1.2.3. Системи ітераційних кусково-визначених функцій

Перш ніж приступити до опису фрактального кодування зображень в градації сірого - необхідно описати конкретне метричний простір, в якому будемо працювати.

Розглянемо зображення в градаціях сірого як дійсні функції $f(x, y)$, визначені на одиничному квадраті $I^2 = I \times I$. Тобто

$$f : I^2 \to \{1, 2, ..., N\} \in R, \tag{1.7}$$

де N - це число градацій сірого.

Визначимо простір F дійсних функцій, інтегрованих з квадратом на I^2 з введено. метрикою. Тоді простір F - повне, і в ньому виконується теорема про стискаючи відображення.

Розглянуті в цьому прикладі зображення - це цифрові зображення. Цифрове зображення $n \times m$ - це матриця значень $[f_{i,j}]$, $i = 1, ..., n$, $j = 1, ..., m$, де $f_{i,j} = f(x, y)$. Таким чином, це матриця фіксованих значень функції $f(x, y)$., взятих в фіксованих точках (x_i, y_j). У цьому випадку будемо говорити про середньоквадратичне метрire(скорочено rms - root mean square) [18]:

$$d_{rms}(f,g)=\left[\sum_{i=1}^{n}\sum_{j=1}^{m}\left|f(x_i,y_j)-g(x_i,y_j)\right|^2\right]^{1\backslash 2} \quad (1.8)$$

У фрактальному зображенні використовується IFS спеціального виду, а саме системи ітераційних кусочно-визначених функцій (partitiomed iterated function system - PIFS). PIFS складається з повного метричного простору X, набору під областей $D \epsilon X$ i $=1, \ldots, n$ набору стискаючих відображень w_i: $D_i \rightarrow X$, i $=1, \ldots, n$ рис. 1.3.

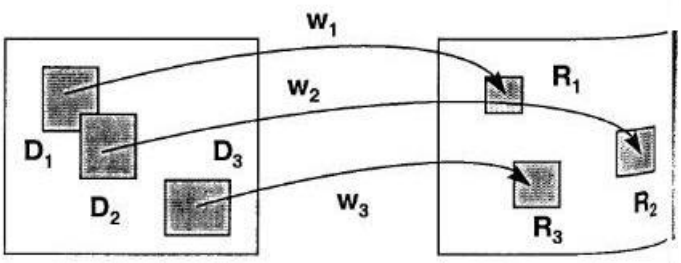

Рисунок 1.3 – Система ітераційних кусочно-визначених функці

Нехай $\tilde{w}_i(x,y)$ - це Афінне перетворення, що переводить в себе одиничний квадрат: $I^2 \rightarrow I^2$, іншими словами

$$\tilde{w}_i(x,y)=A_i\begin{pmatrix}x\\y\end{pmatrix}+b_i, \quad (1.9)$$

для деякої матриці A_i розміру 2×2 і вектора b_i розміру 2×1. Нехай $D_i \in I^2$ - деяка під область одиничного квадрата I^2 і нехай R_i - область значень перетворення $\tilde{w}_i$, що діє на множину D_i, так що $\tilde{w}_i(D_i)=R_i$. Тепер можемо визначити відображення $\tilde{w}_\varepsilon : F \rightarrow F$, що діє на зображення $f(x,y)$, у вигляді

$$\tilde{w}_i(f(x,y)) = s_i f(\tilde{w}_i^{-1}(x,y)) + o_i \qquad (1.10)$$

за умови, що зворотно і $(x,y) \in R_i$. Константа s_i розширює або звужує діапазон значень функції f, або управляє контрастністю. Аналогічно, константа o_i збільшує або зменшує значення градацій сірого, або управляє яскравістю. Перетворення $\tilde{w}_i$ називається просторовою складовою перетворення w_i. І так було задано простір і системи ітераційних функцій, далі опишемо процес фрактального кодування зображення.

1.2.4. Стиснення зображення в градаціях сірого

Опишемо процес стиснення зображення в градаціях сірого.

Розіб'ємо одиничний квадрат I^2 на множину рангових блоків, які утворюють покриття I^2:

$$I^2 = \bigcup R_i, R_i \cap R_j = 0 \qquad (1.11)$$

Нехай $\tilde{w}_i$ - PIFS виду $\tilde{w}_i : D_i \to R_i$ для деякої множини доменних областей $D_i \in I^2$ (області D_i можуть перекриватися і можуть не повністю покривати I^2) рис. 1.4. Для кожного $\tilde{w}_i$ визначимо відповідне стиснення w_i на просторі зображень F:

$$w_i(f)(x,y) = s_i f(\tilde{w}_i^{-1}(x,y)) + o_i, \qquad (1.12)$$

вибираючи s_i, так, щоб w_i було стисненням. Тепер визначимо W: F $\to$ F наступним чином:

$$W(f)(x,y) = w_i(f)(x,y) \text{ для } (x,y) \in R_i \qquad (1.13)$$

В силу того, що рангові області R_i покривають I^2, W визначено для всіх (x, y) з I^2 і, отже, W (f) є зображенням. Так як кожне відображення w_i є стисненням, то W є стисненням на F. Тому, відповідно до теореми про стискаючи відображення, W має єдину нерухому точку $f_\varepsilon \in F$, що задовольняє $W(f_w) = f_w$. Ітеративно застосовуючи W до довільного початкового зображення f_0, отримаємо нерухому точку f_w:

$$W^{0^n}(f_0), \text{ при } n \to \infty, \qquad (1.14)$$

де $W^{0^n}(f_0)$ - це $W(W(...W(f_0)))$ (n раз).

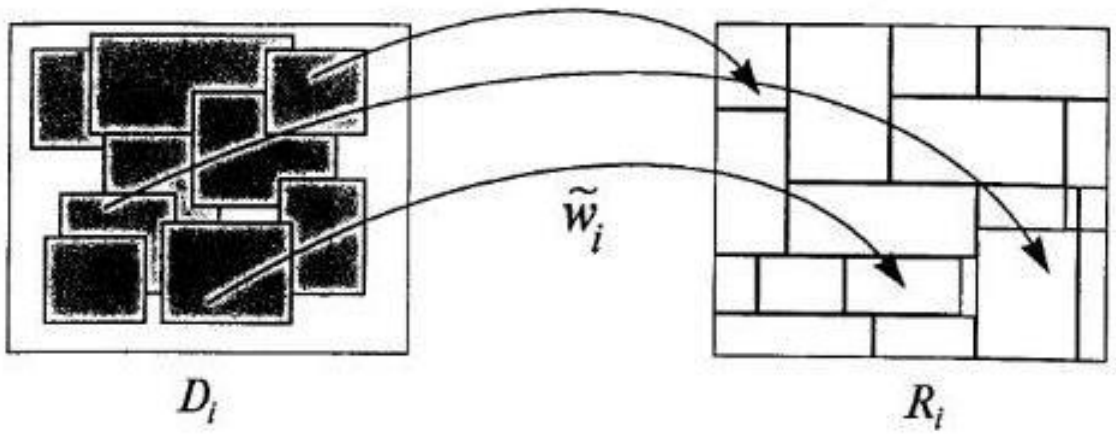

Рисунок 1.4 – Перетворення $\tilde{w}_i$ відображає домени D_i в рангові області R_i

Теорема про стискаючи відображення є базовою для методів фрактального кодування. Дійсно: маємо зображення в градаціях сірого *f*, намагаємося знайти стискаюче відображення *W*, таке, щоб зображення f_w, що є нерухомою точкою відображення *W*, було близько до *f*. Тоді *W* містить всю інформацію, необхідну для отримання f_w. Якщо для зберігання *W* потрібно

менше місця, ніж для зберігання зображення *f*, значить, отримали стиснення зображення [18].

1.2.5. Фрактальне кодування зображення

При фрактальному кодуванні зображень намагаємося знайти множину стискаючих перетворень, які відображають доменні блоки (які можуть перекриватися) в множині рангових блоків, які перекривають зображення. Рангові блоки можуть бути однакового розміру, але частіше використовується адаптивне розбиття зі змінним розміром блоків [18].

Базовий алгоритм фрактального кодування найбільш докладно викладено в [8].

1. Розбиваємо зображення f на непересічні рангові блоки $\{ R_i \}$. Припустимо - рангові блоки - це прямокутники, але можуть використовуватися і інші форми, наприклад, трикутники. Блоки R_i можуть бути рівними, але частіше використовується адаптивне розбиття зі змінним розміром блоків. Це дає можливість щільно заповнювати ранговими блоками маленького розміру частини зображення, що містять дрібні деталі.

2. Покриваємо зображення послідовністю доменних блоків, можливо перекриваючи. Домени можуть бути різних розмірів, і зазвичай їх кількість обчислюється сотнями і тисячами.

3. Для кожного рангового блоку знаходимо домен і відповідне перетворення, яке найкращим чином покриває рангові блок. Зазвичай це Афінне перетворення.

4. Якщо досить точної відповідності не вийшло, то розбиваємо рангові блоки на менші рангові блоки. Продовжуємо цей процес до тих пір, поки або не доб'ємося прийнятного відповідності, або розмір рангових блоків не досягне певної межі.

Одна з головних проблем фрактального кодування в тому, що велика кількість доменних та рангових областей уповільнює процес кодування.

1.3. Огляд методів Fractal Data Mining

1.3.1. Основні методи

Технологія Data Mining заснована на статистичних методах і служить для виявлення в «сирих» даних раніше невідомих, нетривіальних, практично корисних і доступних інтерпретацій знань, необхідних для прийняття рішень. Застосування фрактальних перетворень і самоподібності також відноситься до методів інтелектуального аналізу даних [1].

До фрактальним методам Data Mining відносяться:

- вилученняз баз даних правил асоціації, кластеризації, класифікації;
- зниження розмірності;
- моделі розподілу даних
- бази просторових даних, R-дерева, квадро-дерева
- прогнозуюче моделювання.

Далі розглянемо найбільш вивчені і описані приклади методів Fractal Data Mining.

1.3.2. Кластеризація

Кластеризація - це автоматичне розбиття елементів деякої множини на групи в залежності від їх схожості. Елементами множини може бути що завгодно, наприклад, дані або вектора характеристик. Самі ж групи прийнято називати кластерами. У кластеризації існує велика кількість практичних застосувань, як в інформатиці, так і в інших областях. Прикладами застосування можуть служити: аналіз даних, вилучення та пошук інформації, угруповання і розпізнавання об'єктів. Також кластеризація сама по собі є важливою формою абстракції даних.

Використання самоподібності в кластеризації забезпечує дуже природний спосіб визначення кластерів і не обмежується будь-якої

конкретної формою кластера. Алгоритм кластеризації на основі самоподібності, який в [2] названий «Фрактальна кластеризація» (ФК), поступово визначає місце точки в кластері, для якого може відбуватися зміна фрактальної розмірності після додавання точки. Це дуже природний спосіб кластеризації точок, так як точки в одному кластері мають велику ступінь самоподібності ніж поза ним. Алгоритм ФК є складає 2 кроки і докладно описаний в [2].

Перш ніж застосувати основний напрямок даної техніки, тобто, додавати точки поступово в існуючі кластери, засновані на «фрактальній розмірності», необхідно отримати їх початкову множину. Іншими словами, повинні «спочатку завантажити» наш алгоритм через процедуру ініціалізації, яка знаходить множина кластерів, в кожному з яких досить точок, для того, щоб було можливо обчислити його фрактальную розмірність рис. 1.5. Початкове розбиття на кластери реалізується на основі традиційної процедури обчислення відстані. Кластери формуються з вихідного зразка точок. Беремо випадковим чином крапку і рекурсивно шукаємо точки, близькі до неї. Коли близькі точки закінчуються, то починаємо формувати новий кластер, вибравши наступну точку з множини навмання.

1.Given an initial sample S of points $\{p_1, \cdots, p_M\}$
that fit in main memory, and a distance threshold κ.
(Initially $\kappa = \kappa_0$.)
2. Mark all points as unclustered, and make $k = 0$
3. Randomly choose a point P out of the set
of unclustered points
    4. Mark P as belonging to cluster C_k
    5. Starting at P and in a recursive,
    depth-first fashion,
        call $P' = NEAR(P, \kappa)$
    6. If P' is not NULL
        7. Put P' in cluster C_k
    8. else
        9. backtrack to the previous point
        in the search.
    10. Update $\hat{d}$, the average distance
    between pairs of points in C_k
    11. Make $\kappa = \kappa_0 \times \hat{d}$
12. If there are still unclustered points,
make $k = k + 1$ and go to 3.

$NEAR(P, \kappa)$
Find the nearest neighbor of P such that
$dist(P', P) \leq \kappa$.
If no such P' can be found return NULL
Otherwise return P'.

Рисунок 1.5 – Перший крок алгоритму «Фрактальна кластеризація»

Другий крок алгоритму - це розміщення новоприбулих точок по утвореним кластерам рис. 1.6.

1.Given a batch S of points brought to main memory:
2.For each point $p \in S$:
3.For $i = 1, \cdots, k$:
4. Let $C_i' = C_i \bigcup \{p\}$
5. Compute $F_d(C_i')$
6.Find $\hat{i} = min_i(|F_d(C_i' - F_d(C_i)|)$
7.If $|F_d(C_{\hat{i}}') - F_d(C_{\hat{i}})| > \tau$
8.Discard p as noise
9.else
10. Place p in cluster $C_{\hat{i}}$

Рисунок 1.6 – Другий крок алгоритму «Фрактальна кластеризація»

При додаванні нових точок обчислюється нова фрактальная розмірність кластера. При цьому вже утворені кластери можуть розбиватися або з'єднуватися [2].

1.3.3. Зниження розмірності

Зниження розмірності варто розуміти як виняток корелюючих атрибутів у відношенні.

Реальні дані зазвичай включають шум, як правило, в малих кількостях, але, тим не менш, достатній, щоб ввести нестабільність в алгоритм, яка може бути подолана тільки тоді, коли використовуються великі обсяги даних. Таким чином, набори даних з високою природною розмірністю можуть бути проаналізовані, якщо вони мають велике число точок. У дослідженнях, в яких велика увага спрямована на кластеризацію даних, велика кількість кластерів означає меншу кількість точок в кожному кластері. Однак цікаво відзначити, що на відміну від більшості конкуруючих методів аналізу, в яких великі набори даних є проблемою, метод на фрактальної основі страждає тільки тоді, коли даних занадто мало.

Варто звернути увагу на те, що зазвичай відносини баз даних мають багато ознак, які корелюють з іншими. Атрибути, які корелюють з іншими, не вносять будь-які нові знання, і могли б бути видалені без втрати інформації.

Мета деяких досліджень зводиться до того, щоб знайти підмножину атрибутів, які можуть бути відкинуті при створенні індексів або застосування методів data mining на даних, без шкоди для результатів. Атрибути, які можуть бути розраховані з інших за відомим методом - можна виключити. Таким чином, завдання перетворюється в виявлення кореляції між атрибутами в наборі даних, і як багато надлишкових атрибутів знаходиться в наборі даних. Варто відзначити, що обережно обрання підмножини ознак підвищує продуктивність і ефективність різних алгоритмів. Це особливо вірно в надлишкових даних, так як багато наборів даних можна в значній мірі добре наблизити меншою розмірності. Даний підхід можна розглядати як спосіб стиснення: розглядати тільки ті атрибути, які підтримують основні характеристики збережених даних.

Основна ідея викладена в статті [3] і полягає в використанні «фрактальної розмірності» даних, і відмову від атрибутів, які не впливають на неї. Фрактальна розмірність (D) є відносно стійкою до впливу надлишкових атрибутів. Таким чином, пропонується свого роду алгоритм зворотньої ліквідації, який використовує швидке обчислення D. Цей алгоритм послідовно видаляє атрибути, які сприяють мінімуму на D.

Фрактальна розмірність, по суті, є оцінка ступеня свободи набору даних. Вона дає нам уявлення про те, яким чином дані поширюються в просторі даних. Поширення даних, як правило, пов'язано з кількістю інформації, яку можна отримати з них. Однак реалізація більшості методів інтелектуального аналізу даних є дорогим і вимагає великого часу обчислень. Використання фрактальної розмірності набору даних може скоротити час аналізу даних.

Фрактальна розмірність, по суті, є оцінка ступеня свободи набору даних. Вона дає нам уявлення про те, яким чином дані поширюються в просторі даних. Поширення даних, як правило, пов'язано з кількістю інформації, яку можна отримати з них. Однак реалізація більшості методів інтелектуального аналізу даних є дорогим і вимагає великого часу обчислень. Використання фрактальної розмірності набору даних може скоротити час аналізу даних.

1.3.4. Застосування методів в мультимедійних базах даних

Найбільш поширене застосування методів фрактального аналізу даних в мультимедійних базах даних.

Carnegie Mellon University є одним з університетів, який займається розробками в напрямку фрактального аналізу даних. Основна мета досліджень полягає в знаходженні закономірностей у великих наборах даних, таких як колекції фотографій і відеокліпів, або одного або декількох потоків даних (наприклад, залежність температури від датчиків), або моделі у вигляді графіків, або соціальні мережі. Фрактали і статичні закони використовуються для пошуку моделей і закономірностей, які традиційні методи ніколи не знайдуть, наприклад на типові запити: «знайти зображення, які схожі на захід». Методи для мультимедійного індексування, які дозволяють швидко здійснити пошук за подібністю на великих наборів даних. Ідея полягає в добуванні кількох особливостей, від кожного мультимедійного об'єкта (наприклад, Фур'є і вейвлет-коефіцієнтів з часових рядів, або колірні гістограми з образів), а потім використовувати поза готових методів бази даних для організації в результаті багатомірного точок.

В даний час в університеті Carnegie Mellon University ведеться створення біологічної бази даних зображень. Мета - використовувати сучасні методи пошуку за подібністю в зображення субклітинного білка, а також актуальність зворотного зв'язку [16].

1.3.5. Реалізація методів в СУБД Cross / Z

Однією з найбільш серйозних проблем аналізу та інтерпретації інформації є необхідність виділення підмножини даних. При побудові моделі необхідно шукати компроміс між числом записів (рядків) у вибірці даних і кількістю оцінюваних змінних.

Група продуктів компанії Cross / Z International, що отримала назву Fractal Data Mining System, здатна витягувати набори даних, розмір яких практично не залежить від кількості записів у вихідній БД. Однак число змінних не повинно перевищувати 12 (із сукупною комбінацією значень менше мільярда).

Дані виділяються з інформаційного сховища в файл, який містить результати запитів. Для цього задається максимальна кількість питань про змінні. Припустимо, у вас є база даних з таблицею, в якій містяться атрибути «стать», «місце проживання», «вік» і «дохід» - по одному запису на кожну людину. Система задає питання виду «Скільки чоловіків у віці 26 років проживає на Алясці?» і повторює його, збільшуючи вік (27, 28 років і т. д.). Значення кожної змінної змінюються до тих пір, поки вони не будуть вичерпані. Такі величини, як «дохід», використовуються в даному прикладі в якості базового показника і зберігаються у відповідях на питання як узагальнений результат (скажімо, як сума або середнє значення).

Отримана в результаті база даних, як правило, набагато компактніше, ніж вихідний набір даних, оскільки відповіді на питання зазвичай займають значно менше місця, ніж самі дані. У розглянутому прикладі існує близько 20 тис. Комбінацій віку, місця проживання і статі, тому незалежно від кількості записів (100 тис. або 100 млн.) Обсяг набору даних буде приблизно однаковий [6].

2 РОЗРОБКА АЛГОРИТМУ ВИДІЛЕННЯ ДОМЕНІВ

2.1 Подання таблиці як фрактала

Покажемо, що реляційна таблиця також є фракталом. Для початку наведемо основні визначення теорії реляційних баз даних. Схемою відношення R називається кінцеве множина імен атрибутів *{A_1, A_2, …, A_n}*. Кожному імені атрибута A_i ставиться у відповідність множина D_i - множина значень атрибута A_i, $1<i<n$. Нехай $D = D_1 \cup D_2 \cup ... \cup D_n$. Відношення r зі схемою R - це кінцева множина відображень $\{t_1, t_2, ..., t_p\}$ з R в D; причому кожне відображення t є r має задовольняти наступній обмеження: $t(A_i)$ належить D_i , $1<i<n$. Ці відображення називаються кортежами [17].

У попередньому розділі було відзначено, що основний принцип застосування теорії фракталів - це пошук простих частин, подібних цілому, застосувавши до яких певну ітераційну функцію, зможемо отримати всю систему. З визначення відносини r можна відзначити, що кортеж - це складова частина, яка несе в собі основну інформацію про структуру відносини. Додаючи кожен наступний кортеж - будуємо відношення рис. 2.1.

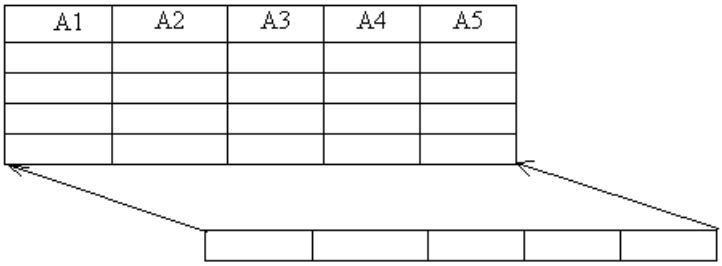

Рисунок 2.1 – Самоподібність відносин

З точки зору фрактального аналізу доменом для відносини може виступати кортеж цілком або деяка його частина. Таким чином, доменом

може бути деяка комбінація атрибутів одного кортежу, тим самим можемо розкласти відносини на ще більш дрібні частини, зберігши при цьому відомості про структуру таблиці рис. 2.2.

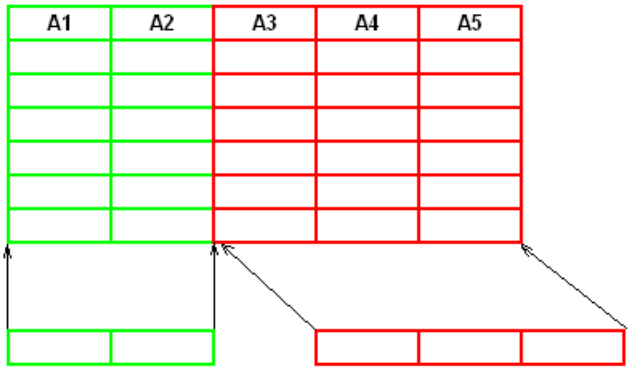

Рисунок 2.2 – Розбивка кортежу по доменах

Кожен атрибут обмежений певним множиною значень, значить, і пара атрибутів буде також обмежена деякою множиною. Тоді можна припустити, що домени в відношенні можуть бути не тільки ідентичними за структурою, але і мати однакові значення рис.2.3 [13].

Імя	Місто	Об'єм	Колір
Гайка	Харків	5	Червоний
Гайка	Харків	10	Синій
Болт	Київ	5	Червоний
Болт	Київ	10	Синій
Болт	Київ	20	Чорний

Рисунок 2.3 – Домени на рівні значень

Варто відзначити той факт, що первинний ключ відносини може виступати як деяка функція, яка дозволить визначити яке значення домену до якого кортежу відноситься.

Таким чином, виділяючи домени в відношенні, можна визначити взаємозв'язок атрибутів і фрактально закодувати таблицю, зберігаючи відомості про її структуру. Що дозволяє не тільки отримати нові, корисні і не тривіальні знання, але представити відношення в більш компактному вигляді [13].

2.2 Постановка задачі

Головна проблема подання таблиці як фрактала - це виділення доменів, тому основним завданням даної роботи є побудова алгоритму пошуку доменів.

Завдання побудови алгоритму пошуку доменів і їх використання розбивається на ряд підзадач.

1. Проектування загального алгоритму пошуку доменів.
2. Реалізація найпростішого алгоритму - пошук доменів повним перебором.
3. Виявлення критеріїв, що визначають структуру доменів, і побудова алгоритму пошуку доменів за отриманими критеріями.
4. Побудова прикладів алгоритмів вилучення нових знань на основі отриманого безлічі доменів.

Рішення вище описаних підзадач дозволить витягти залежності між атрибутами. Однак на основі побудованих доменів можна витягти не тільки відомості про взаємозв'язок атрибутів, але і про зв'язки між об'єктами. Крім того безліч можливих структур доменів дозволяє витягти нові, нетривіальні знання на основі побудови класифікації та кластеризації об'єктів.

2.3 Проектування алгоритму

2.3.1. Загальний вигляд алгоритму пошуку доменів

Основна ідея алгоритму полягає в підборі такої множини доменів, яка повністю опише структуру відносини і дозволить представити її вміст мінімально можливою кількістю записів.

У загальному вигляді алгоритм можна представити в такий спосіб рис. 2.4.

Рисунок 2.4 – Загальний вигляд алгоритму списку доменів

Кожна частина алгоритму являє собою окрему задачу, яку можна вирішити різними способами. Далі опишемо кожен крок загального алгоритму пошуку доменів більш докладно.

2.3.2 Список можливих структур доменів

Як було сказано, доменом будемо вважати деяку комбінацію атрибутів одного кортежу. Розміром домену будемо називати кількість вхідних в нього атрибутів.

Нехай n - це кількість атрибутів в таблиці, а k - це розмір домена, причому $1 \le k \le n$. Тоді кількість можливих доменів розміру k в таблиці становить:

$$C_n^k = \frac{n!}{k!(n-k)!} \tag{2.1}$$

Тоді загальна кількість всіляких структур доменів буде:

$$\sum_{k=1}^{n} C_n^k = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!} \tag{2.2}$$

При фрактальному кодуванні зображення виникало питання про оптимізацію пошуку і підборі доменних областей таким чином, щоб зберегти точність зображення. В результаті під структурою домену в завданні стиснення зображення розглядався не тільки розмір домену, але і його вигляд: прямокутник, трикутник, шестикутник та інші. В силу того, що в відношенні кортеж є конкретний об'єкт, то можна вважати, що інше визначення домену для відносини не представляється відповідним. Таким чином, кажучи про структуру домену, обмежуємося лише його розміром. Оцінимо приблизну кількість всіх можливих структур доменів відносини. При формуванні ставлення кількість вхідних в нього атрибутів намагаються обмежувати 15, тому кількість можливих структур доменів не повинно перевищувати 32767 штук. Однак якщо при проектуванні схеми бази даних не дотримуватися цього обмеження, то кількість атрибутів в таблиці може

бути набагато більше, отже, різко зросте кількість можливих структур доменів. Тим самим складність вирішення поставленого завдання багато в чому залежить від кількості атрибутів в таблиці, так як велика кількість атрибутів, тягне велику кількість можливих структур доменів, відповідно аналіз отриманого великого безлічі доменів тягне високі витрати на час виконання та зберігання.

У зв'язку з описаної вище проблемою, виникає питання про час пошуку і достатню множину структур доменів, які дозволять найбільш повно представити описувану систему. Ґрунтуючись на статистичних відомостях про дані, що містяться в базі, стає можливим скоротити список структур доменів. Чим менше кількість розглянутих структур доменів, тим менше часу необхідно на пошук різних значень і на формування оптимальної множини доменів.

2.3.3 Кількість різних значень домену

Кількість різних значень домена є найважливішою характеристикою структури, по суті, вона є критерієм того, наскільки дана структура нам підходить. Чим менше кількість різних значень домену, тим краще, тим більше інформації про взаємозв'язок атрибутів структура домену відображає.

Алгоритм пошуку кількості різних значень досить простий: просто порівнюємо кожну комбінацію атрибутів з уже наявними значеннями рис. 2.5.

Пошук кількості різних значень домена є найбільш витратною частиною алгоритму, так як складність залежить від потужності віднощення. Тому дуже важливо заздалегідь відкинути деяку кількість структур доменів, щоб скоротити час виконання даної частини. Виняток невідповідних структур доменів буде проводитися на підставі відомостей, що зберігаються в словнику бази даних. Оцінюючи кількість різних значень доменів, можна виключити ті структури, які містять велику кількість відмінних даних.

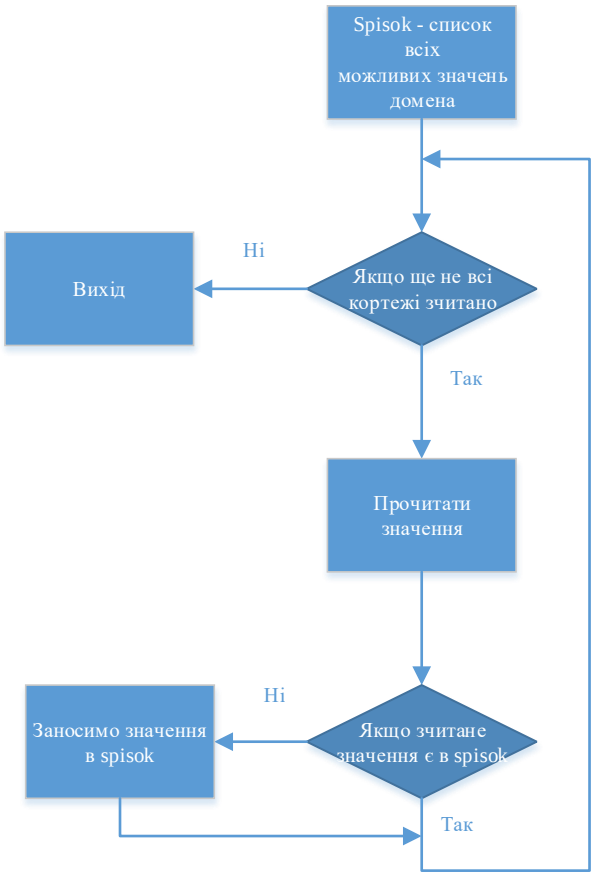

Рисунок 2.5 – Алгоритм формування списку різних значень домену

Варто відзначити той факт, що алгоритм пошуку кількості різних значень легко піддається розпаралелюванню на всій множині структур доменів, так як для кожної структури домену алгоритм може виконуватися незалежно. А так як робота з базою даної йде безпосередньо через СУБД, то використовуючи команди мови SQL, оптимізувати роботу алгоритму можна за рахунок зменшення кількості звернень до бази даних.

2.3.4 Оптимальна множина доменів і нові знання

Після того, як було отримано кількість різних значень доменів, можна сформувати множину доменів, які повністю описують таблицю. Зауважимо, що не для всіх структур доменів кількість різних значень може бути істотно менше кількості кортежів, тому необхідно вибрати одну з множин доменів,

що описують представлення, найбільш оптимальне. $D = D_1 \cup ... \cup D_m$, $|D|>1$,

Оптимальною множиною доменів D будемо називати сукупність структур доменів, яка містить в сукупності всі атрибути таблиці в єдиному екземплярі, і сума записів, що містяться в доменах, буде мінімальна. Оптимальна множина доменів дозволяє отримати список різних значень, яких достатньо, щоб представити таблицю, задавши певну систему функцій. Варто відзначити той факт, що найоптимальніша множина - це список всіх різних значень, що зберігаються в таблиці, тому накладається обмеження на розмір домена.

Крім того, отримавши відомості про кількість різних значень доменів, можна сформувати алгоритми для отримання нових, нетривіальних знань з даної таблиці, в тому числі:

- зниження розмірності. Отриманий набір різних значень дозволить визначити наявність функціональних залежностей в таблиці, тим самим з'являється можливість обчислити корелюють атрибути;
- кластеризація. Виділення доменів дозволяє виконувати послідовну кластеризацию об'єктів, тим самим з'являється можливість деякого регулювання розміру кластера і ознак розбиття.

Складність даної частини алгоритму визначається кількістю отриманих структур доменів - чим більше структур, тим більше інформації необхідно обробити і проаналізувати. Тому час виконання останньої частини багато в чому залежить від обсягу списку можливих структур доменів.

2.4 Алгоритм пошуку доменів повним перебором

Для того щоб бути впевненим в тому, що був знайдений відповідний набір доменів для кодування таблиці, необхідно розглянути всі можливі структури і всі можливі значення в таблиці.

Найбільш простий і очевидний спосіб пошуку потрібних доменів - це повний перебір. На відміну від перебору при фрактальному кодуванні

зображення, в реляційних таблицях структура домена обмежена в розмірі, отже, набір структур обмежується кількістю всіляких сполучень з $_n$ атрибутів по $k(k \leq n)$ штук, дорівнює виразу (2.1) , що дає можливість розглянути всі комбінації.

Сам алгоритм для фіксованого k можна представити у вигляді схеми рис. 2.6.

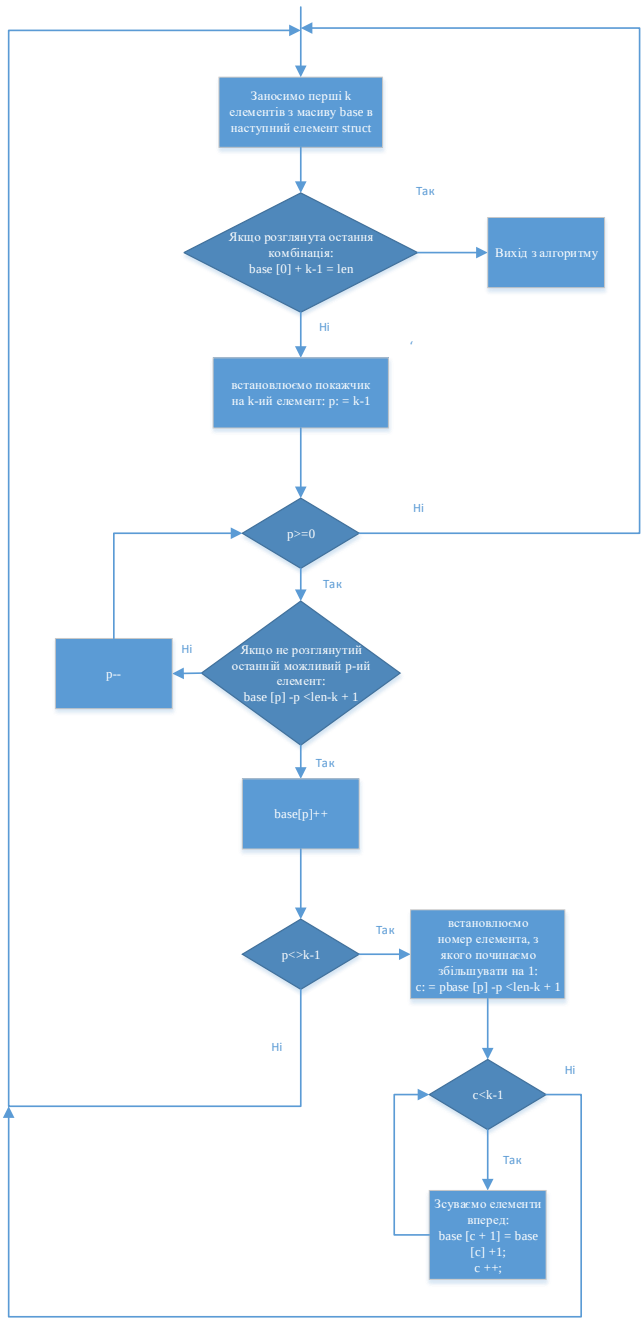

Рисунок 2.6 – Алгоритм складання всіляких поєднань по k.

Пронумеруємо всі стовпці від 1 до n, тобто кожному атрибуту аналізованого відносини дамо порядковий номер. Наведемо основні змінні:

base - масив елементів від 1 до n, кожен елемент є номер атрибуту відношення. Масив відсортований в порядку зростання;

struct - динамічний масив, кожен елемент якого є необхідною комбінацією атрибутів, що відображає одну з можливих структур домену.

len - розмір масиву base

p - номер елемента, що збільшується

c - номер елемента, що сдвигається

В результаті роботи алгоритму отримуємо список всіх сполучень атрибутів по k. Щоб отримати всі структури доменів алгоритм необхідно виконати для кожного k, $1 \leq k \leq n$.

Весь список структур доменів спочатку задає великий обсяг інформації, що обробляється, тим самим збільшується час виконання алгоритмів для вилучення кількості різних значень домену та побудови оптимальної множини.

2.5 Пошук доменів за критеріями

2.5.1 Зв'язок доменів і функціональних залежностей

Варто зазначити, що нормалізація баз даних переслідує дві основні мети: знизити надмірність даних і підвищити їх надійність. Будь-яке апріорне знання про різні обмеження, що накладаються на сукупність даних, може принести велику користь для досягнення зазначеної мети. Один із способів формалізації цих знань - встановлення залежностей між даними (функціональні залежності). Присутність функціональних залежностей в реляційній схемі означає необхідність декомпозиції схеми для зменшення надмірності. Тим самим розбиття таблиці на домени може допомогти обчислити ще невідомі функціональні залежності.

По суті, функціональна залежність є зв'язком типу «багато до одного» між множинами атрибутів всередині змінного-віднощення. Функціональна залежність усуває взаємозв'язок атрибутів у відношенні. Через такі залежності виникає надмірність даних. Однак структура домена, що складається з функціонально залежних атрибутів, дасть мінімальну кількість значень на атрибутах і буде найбільш привабливим для цих атрибутів, ніж їх наявність в інших структурах

Безліч різних значень доменів може явно відобразити наявні в таблиці функціональні залежності, а наявність інформації про відомі функціональні залежності - скоротити список можливих структур доменів, не втративши мінімально можливої кількості різних значень

Введемо визначення функціональної залежності. Нехай R є змінною-представлення, а X і Y - довільними підмножинами атрибутів змінного-відношення R. Тоді Y функціонально залежно від X (X $\rightarrow$Y) тоді і тільки тоді, коли для будь-якого допустимого значення змінного-відношення R кожне значення множини X відношення R пов'язано в точності з одним значенням множини Y відношення R. Інакше кажучи, для будь-якого допустимого значення змінного-відношення R, якщо два кортежи змінного-відношення R збігаються за значенням X, вони також збігаються і за значенням Y. [10] Дамо суворе визначення поняття, використовуючи реляційні оператори. Нехай r відношення зі схемою R, X і Y - підмножини R. Представлення r задовольняє функціональну залежності X $\rightarrow$Y, якщо $\pi_{\gamma}(\sigma_{X=x}(r))$ має не більше ніж один кортеж для кожного X-значення x. Один із способів інтерпретувати цей вислів - розглянути два кортежу t_1 і t_2 в r. Якщо $t_1(X)=t_2(X)$, то $t_1(Y)=t_2(Y)$ [17]. Розглянемо приклад представлення рис. 2.7. В даному відношенні явно міститься ФЗ: РЕЙС $\rightarrow$ЧАС_ВИЛЬОТУ.

У якості більш складного прикладу можна навести відношення R з атрибутами A, B і C, для яких виконуються ФЗ A $\rightarrow$B і B $\rightarrow$C. Неважко

помітити, що в цьому випадку також виконується ФЗ A →C, яка називається транзитивною ФЗ, тобто С залежить від А транзитивно через В.

Пілот	Рейс	Дата	Час_Вильоту
Кушинг	83	9.08	10:15
Кушинг	116	10.08	13:25
Кларк	281	8.08	5:50
Кларк	301	12.08	18:36
Кларк	83	11.08	10:15
Чин	83	13.08	10:15
Чин	116	12.08	13:25
Коупли	281	9.08	5:50
Коупли	281	13.08	5:50
Коупли	412	15.08	13:25

Рисунок 2.7 – Відношення ГРАФІК

ФЗ - це особливий вид обмежень цілісності, а тому вони, безсумнівно, є поняттям семантичним. Розпізнавання ФЗ являє собою частину процесу з'ясування сенсу тих чи інших даних [10]. Тим самим можемо прийняти ФЗ як особливий вид обмеження для домену. Якщо атрибути є функціонально залежними, то розглядати їх у складі інших структур доменів не має сенсу. Структура, що містить в собі ФЗ, дасть мінімальну кількість різних значень на множині можливих структур доменів, що містять один з атрибутів, що становлять ФЗ.

Розпізнавання ФЗ на етапі аналізу і пошуку структур доменів зменшить складність пошуку можливих структур доменів і скоротить їх список

2.5.2 Використання словника СКБД

Ще одним способом зменшення кількості можливих структур доменів є використання додаткових статистичних даних про розміщення значень в стовпцях таблиці. Основною інформацією для виключення невідповідних структур є кількість різних значень в стовпці.

Будь-яка СКБД має певний набір службових таблиць (словник бази даних), в яких зберігаються відомості про базу даних, з якого будь-який користувач в будь-який момент може дізнатися все необхідне про свої таблиці. Словник даних - це набір системних таблиць і представлень, модифікація яких заборонена.

Також засобами кожної СКБД визначаються можливості збору статистики про збережених даних, зокрема відомості про кількість різних значень стовпця певної таблиці. Дана статистика використовується оптимізатором виконання запитів для створення найбільш ефективного плану виконання запиту. Варто зазначити, що статистичні відомості повинні бути достовірними і повноцінні на момент проведення аналізу бази даних. Тому необхідно мати необхідні інструменти для отримання потрібних відомостей.

В даній роботі в якості СКБД для реалізації програми обрана СУБД Oracle 19g. Словник СУБД Oracle містить величезну кількість таблиць з різною інформацією про структуру бази даних, налаштування системи. Далі опишемо необхідні інструменти для збору статистичної інформації та таблиці словника даних, в яких збираються необхідні дані: USER_TAB_COL_STATISTICS, ALL_TAB_HISTOGRAMS.

USER_TAB_COL_STATISTICS - таблиця словника бази даних Oracle, яка містить статистику по стовпцях таблиць користувача, представлень і кластерів. Значення кожного з стовпців в USER_TAB_COL_STATISTICS описано в таблиці 3.1 [19].

Таблиця 2.1 – Структура таблиці USER_TAB_COL_STATISTICS

Ім'я стовпця	Опис
TABLE_NAME	Назва таблиці
COLUMN_NAME	Назва стовпця
NUM_DISTINCT	Число різних значень в стовпці
LOW_VALUE	Найменше значення в стовпці

Продовження таблиці 2.1

Ім'я стовпця	Опис
HIGH_VALUE	Найбільше значення в стовпці
DENSITY	Щільність в колонці
NUM_NULLS	Число значень null в стовпці
NUM_BUCKETS	Кількість сегментів в гістограмі стовпця
LAST_ANALYZED	Коли в останній раз колонка була проаналізована
SAMPLE_SIZE	Розмір вибірки, яка використовується при аналізі даної колонки
GLOBAL_STATS	Чи розраховується статистика без злиття основних розділів?
USER_STATS	Чи була статистика розрахована безпосередньо на користувача?
AVG_COL_LEN	Середня довжина стовпчика в байтах
HISTOGRAM	Гістограма

Таблиця USER_TAB_COL_STATISTICS містить інформацію про найменування таблиць користувача, найменування колонок таблиць, тип даних, і, що найголовніше, кількість різних значень в колонці, кількість значень NULL і чи побудована гістограма по стовпцю.

Засобами СКБД Oracle можливо в будь-який момент часу зібрати необхідну статистику для таблиці USER_TAB_COL_STATISTICS. Для того щоб видалити старі дані необхідно виконати команду: analyze table delete statistics;

Де table_name - ім'я таблиці користувача, статистику по якій необхідно оновити. Для збору загальної статистики виконується команда: analyze table compute statistics for all columns;

Тим самим можемо з'ясувати необхідні для аналізу дані: найменування стовпців, тип даних у відповідному стовпці, кількість різних значень в стовпці [7].

Отримавши необхідні статистичні дані, можна приблизно припустити яке поєднання атрибутів, дасть найменшу кількість різних значень.

Окремо розглянемо такий збір статистики як побудова гістограм. За визначенням з математичної статистики гістограма - наближення щільності ймовірності деякої випадкової величини, побудоване за вибіркою з її розподілу. Оптимізатор СКБД Oracle використовує гістограми для отримання точних оцінок розподілу даних в стовпці. При побудові гістограм значення в стовпці поділяються на смуги так, що всі значення стовпців в групі потрапляють в один і той же діапазон. Діапазон значень в гістограмі називається bucket. За замовчуванням максимально можливу кількість bucket для гістограми 75. Це значення забезпечує відповідний рівень деталізації даних. У СУБД Oracle гістограми діляться на два типи [5].

1. Value-Based Histograms або частотна гістограма створюється, коли число різних значень менше або дорівнює числу можливих bucket. Відповідно кожен bucket відображає точну кількість значень.

2. Height-Based Histograms або збалансована гістограма визначає приблизно однакову кількість значень в кожному діапазоні, так що кінці діапазону визначається тим, як багато значень в цьому діапазоні.

Частотні діаграми дають точну кількість значень в кожному bucket, тоді як по збалансованим гістограми можливо лише обчислити розподіл. Таким чином, стовпці за якими побудована частотна гістограма варто розглядати окремо, і порівнювати зі стовпцями, за якими побудована збалансована, тільки по відношенню між висотами смуг bucket.

Гістограми, як і всі інші дані оптимізатора, є статичними. Вони корисні тільки тоді, коли вони відображають поточний розподіл даних з даного стовпчика. Для того щоб побудувати гістограму в Oracle, необхідно скористатися такою процедурою: dbms_stats.gather_table_stats(, ‘’, method_opt => 'for columns size ');

Параметр name_columns задає список стовпців, за якими створюється гістограма. Особливо варто відзначити параметр «num_group», оскільки саме

він задає кількість частин, на які розбиваються значення в стовпці. Результат збору статистики - побудована гістограма, буде відображена в таблиці словника бази даних ALL_TAB_HISTOGRAMS [11].

Таблиця 2.2 – Структура таблиці ALL_TAB_HISTOGRAMS

Ім'я стовпця	Опис
OWNER	Власник таблиці
TABLE_NAME	Ім'я таблиці
COLUMN_NAME	Ім'я стовпця
ENDPOINT_NUMBER	Номер групи
ENDPOINT_VALUE	Нормалізоване значення кінцевої точки групи
ENDPOINT_ACTUAL_VALUE	Кінцеве, точне значення кінцевої точки групи

Такий інструмент аналізу як гістограма може допомогти у визначенні розподілу значень в стовпці і спробувати розпізнати аналогічні залежності в інших стовпцях.

Гістограма і інформація про кількість різних значень в стовпці в поєднанні дозволять сформувати критерії для виділення структур доменів, які будуть містити найменшу кількість різних значень.

2.5.3 Критерії пошуку доменів

Основою побудови критеріїв для пошуку потрібних структур доменів є теорія функціональної залежності і статистична інформація про дані у відносинах.

Раніше було відзначено, що структура домена, що містить в собі функціональну залежність, дасть мінімальну кількість різних значень на множині можливих структур доменів, що містять в своєму складі

функціонально залежні атрибути. Ґрунтуючись на статистичних даних, зібраних за допомогою засобів СКБД Oracle, можна виділити структури, які можуть бути функціонально залежні, використовуючи два критерії:

- кількість різних значень в двох стовпчиках має збігатися
- значення в стовпці повинні бути однаково розподілені.

Слід відзначити, що при початковому проектуванні схем баз даних не завжди вдається виявити всі функціональні залежності від самого початку, внаслідок проектування їх виділяє весь суб'єктивно. Однак не виключений той факт, що функціональні залежності будуть неявними: частина значень має явну залежність, а частина - ні.

При пошуку залежних атрибутів варто відзначити той факт, що не завжди можлива ситуація, коли виходить чітко виражена функціональна залежність. Цілком можливо наявність деякого невеликого шуму, який не дозволяє звести атрибути під визначення функціональної залежності. У подібній ситуації варто опустити критерій обов'язкового збігу кількості різних значень в стовпцях, залишивши критерій однакового розподілу значень. Розглядаючи можливі структури доменів за цим критерієм, ми можемо врахувати випадки, коли кількість значень в стовпцях приблизно однаково

Аналіз розподілу значень в стовпцях дозволяє приблизно визначити взаємозв'язок атрибутів відносини між собою. Однак необхідно розглянути ще один важливий фактор. Кількість різних значень в стовпці, може бути набагато менше, ніж загальна кількість кортежів в таблиці. Таким чином, ми не можемо виділити залежність навіть в деякій частині значень атрибутів, проте можлива кількість різних значень структур доменів може виявитися значно меншим, порівняно із загальним числом кортежів. Якщо розглядати з точки зору класифікації та побудови оптимального безлічі, то не можна не враховувати цей факт

Маючи відомості про кількість різних значень, завжди можемо максимально оцінити кількість різних значень структури домену. Кількість

різних значень домену (KD) завжди менше або дорівнює добутку кількостей різних значень атрибутів, що входять в домен. Значення KD може бути або більше кількості кортежів в таблиці, або менше. Якщо KD виявиться набагато більше кількості кортежів щодо, то структуру, кількість різних значень якої обмежує дане значення, розглядати сенсу не має. Однак якщо KD виявиться набагато менше кількості кортежів щодо, то структуру, кількість різних значень якої обмежує дане значення, варто розглядати, так як вона може дати малу кількість різних значень домену, структура якого дозволить підібрати оптимальний безліч для кодування відносини або розбити об'єкти таблиці на більші класи.

Виходячи з викладених вище тверджень, стає не зовсім зрозуміло, що саме може бути параметром обмеження величини KD. Використовувати цей критерій можливо з суб'єктивної сторони і вибирати параметр в залежності від того, яке завдання вирішується і яку максимальну кількість різних значень прийнятно для поставленого завдання. Наприклад, стиснення в певне число раз вимагатиме вибору параметра, який буде дорівнює частині кількості кортежів таблиці. Визначення значення параметра буде досліджено в практичних експериментах.

В цілому, формально представити здійснення вибору доменів за критеріями можна у вигляді схеми рис. 2.8.

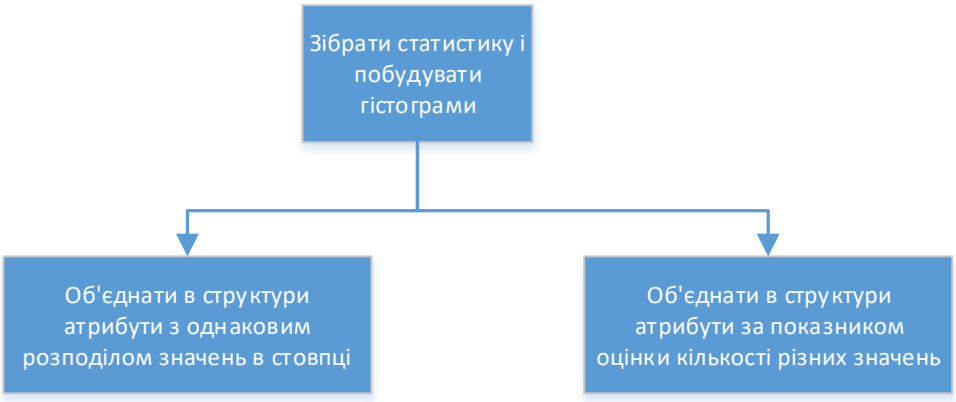

Рисунок 2.8 – Схема вибору структур доменів за критеріями

У підсумку, вибір структур доменів буде визначатися кількістю різних значень в стовпці і розподілом значень, що значно зменшить обсяг даної інформації і дозволить вже на цьому етапі виділити явні залежності.

2.5.4. Алгоритм пошуку доменів за критеріями

У попередньому розділі були описані основні критерії пошуку структур доменів, які можуть дати мінімальний набір безлічі різних значень. Весь алгоритм можна розділити на дві логічні частини.

1. Збір статистики та підготовка даних.

2. Складання структур доменів.

Збір статистики відбувається після виконання спеціалізованих функцій СУБД Oracle. З таблиці USER_TAB_COL_STATISTICS важливими даними є відомості з стовпців NUM_DISTINCT (кількість різних значень) і HISTOGRAM (вид побудованої гістограми).

Підготовка статистичних даних полягає у приведенні відомостей з таблиць словника до загального вигляду для порівняння і виявлення еквівалентних залежностей. Гістограми в СУБД Oracle представлені у вигляді функції розподілу і розділені на два типи. В незалежності від типу гістограми необхідно представити результат гістограми в вигляді співвідношення: «bucket» - «кількість значень» і виконати сортування отриманих значень по полю «кількість значень» в порядку убування. Відсортовані дані дозволять визначити відношення між смугами гістограми і тим самим оцінити, наскільки однаково розподілені значення в різних стовпцях. Окремо необхідно порівняти отримані розподілу за типом побудованої гістограми: збалансовані зі збалансованими, частотні з частотними

Варто відзначити, що при виявленні однакових закономірностей висоти смуг необхідно задати деяку можливу похибку. Нехай a характеризує відношення двох смуг першого атрибута, а b - другого, тоді будемо вважати,

що ставлення смуг однаково, якщо a / b = 1. Так як повна відповідність відносин смуг не завжди можливо в силу можливого шуму, тоді будемо вважати, що ставлення смуг однаково з похибкою e, якщо a / b ∈ (1-e, 1 +e). Значення параметра e залежить від задачі. Так, наприклад, якщо вирішується завдання про пошук функціональної залежності, то значення e повинно бути якомога менше, якщо ж вирішується завдання про стиснення таблиці, то значення e можна вибирати дещо більшим.

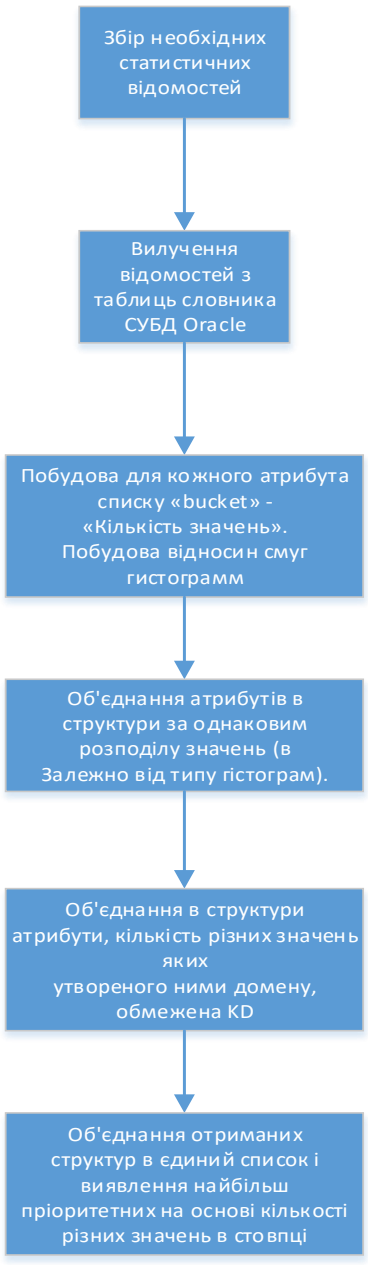

Рисунок 2.9 – Алгоритм пошуку доменів за критеріями

Після аналізу гістограм необхідно застосувати відомості про кількість різних значень в стовпці, оскільки ці відомості дозволять виділити найбільш пріоритетні структури. Найбільш пріоритетними структурами будемо вважати ті, в яких різні значення атрибутів, що входять до складу домену, збігаються.

Після отримання необхідних даних, з'являється можливість вибрати поєднання атрибутів, які дадуть найбільш вдалі структури доменів. Якщо потенційна структура складається з більш ніж двох атрибутів, то необхідно додати в список можливих структур ті, які складаються з атрибутів вихідної комбінації. Дана дія необхідно для подальшого пошуку функціональних залежностей і проведення аналізу

Послідовність вибору структур доменів за складеними критеріями можна представити у вигляді блок-схеми рисунок 2.9.

В результаті виконання пошуку доменів за критеріями можна не тільки скоротити безліч доменів, які необхідно обробити, але і вже на початковому етапі виявити функціональні залежності.

2.6. Алгоритми пошуку оптимального множини і вилучення нових знань

2.6.1. Вибір оптимального множини доменів

Отримавши безліч можливих структур доменів, необхідно вибрати мінімальний набір значень доменів, застосувавши до якого деяку функцію, ми зможемо відновити всю таблицю. Для цього за списком можливих структур доменів потрібно прорахувати реальну кількість значень доменів з таблиці для кожного поєднання. Необхідно вибрати ті домени, мінімальна кількість значень яких дозволить зібрати всю таблицю

Для вирішення цього завдання можна запропонувати наступну послідовність дій. Отриманий список структур доменів розбиваємо на

множини, кожне з яких містить стільки доменів, скільки необхідно для відновлення таблиці. Головною характеристикою отриманого безлічі є кількість різних значень доменів, яке воно містить. Безліч, в яке входить мінімальна кількість різних значень доменів, - шукане

Алгоритм пошуку оптимальної множини доменів можна представити у вигляді рекурсивної функції. Будемо вважати, що список доменів представлений у вигляді масиву, в якому кожен елемент має свій індекс. Визначимо основні допоміжні змінні:

Мн - оптимальна множина.

М - множина, яку будуємо.

k - кількість атрибутів, необхідне для заповнення М.

num - індекс, з якого необхідно розглядати домени з масиву.

rz - кількість записів в М.

min - мінімальна кількість записів в множині Мн.

Параметри першого запуску функції optimal_collection: М - порожня множина, k = 0, num = 0, rz = 0, min = (кількість атрибутів в відношенні) * (кількість рядків в відношенні).

З отриманого списку множин необхідно вибрати оптимальний, а саме те безліч, сумарна кількість різних значень доменів в нього входять, буде мінімальним.

2.6.2. Виявлення функціональних залежностей

Визначення та необхідність пошуку ФЗ були описані вище. На етапі аналізу статистичних даних, що зберігаються в словнику СУБД, можливо найбільш ймовірно виділити прості залежності, проте необхідно мати можливість виділити на безлічі різних значень доменів і багатозначні залежності.

Пошук багатозначних залежностей також є одним з етапів нормалізації відносини. Визначимо поняття багатозначної залежності. Нехай A, B і C є

довільними підмножинами безлічі атрибутів змінної-відносини R. Тоді підмножина B багатозначно залежить від підмножини A (A $\rightarrow\rightarrow$ B), тоді і тільки тоді, коли безліч значень B, відповідне заданої парі (значення A, значення C) змінної-відносини R, залежить від A, але не залежить від C. [10] Дамо суворе визначення поняття, використовуючи реляційні оператори. Нехай R - реляційна схема, X і Y - підмножини R, і нехай Z = R- (XY). Ставлення r (R) задовольняє багатозначною залежності (MV - залежності) X $\rightarrow\rightarrow$Y, якщо для будь-яких двох кортежів t_1 і t_2 з r, для яких t_1 (X)= t_2 (X), в r існує кортеж t_3, для якого виконані співвідношення t_3 (X)= t_1 (X), t_3 (Y)= t_1 (Y) [17] Поняття багатозначної залежності узагальнює поняття функціональної залежності, пошук ФЗ буде будуватися на понятті багатозначною залежності.

Алгоритм:

Вхід: Відношення r і F-залежність X $\rightarrow$Y.

Вихід: істина, якщо r задовольняє X $\rightarrow$ Y, брехня - в іншому випадку. SATISFIES(r, X $\rightarrow$ Y);

1. Пересортируємо відношення r по X-стовпцях так, щоб зібрати кортежі з рівними X-значеннями разом.

2. Якщо кожна сукупність кортежів з рівними X-значеннями має також рівні Y-значення, то на виході отримуємо істину, в іншому випадку - брехня.

Розглянемо відношення ГРАФІК (Пілот, Рейс, Дата, Час-Вильоту) рис. 2.10

Пілот	Рейс	Дата	Час_Вильоту
Кушинг	83	9.08	10:15
Кушинг	116	10.08	13:25
Кларк	281	8.08	5:50
Чин	83	13.08	10:15
Чин	116	12.08	13:25
Коупли	281	9.08	5:50
Коупли	412	15.08	13:25

Рисунок 2.10 – Відношення Графік

Тоді результатом алгоритму для STATISFIES (графік, час вилета→рейс) буде наступне, дивись рис. 2.11

Пілот	Рейс	Дата	Час_Вильоту
Кларк	281	8.08	5:50
Коупли	281	9.08	5:50
Коупли	281	13.08	5:50
Кушинг	83	9.08	10:15
Кларк	83	11.08	10:15
Чин	83	13.08	10:15
Кушинг	116	10.08	13:25
Чин	116	12.08	13:25
Коупли	412	15.08	13:25
Кларк	301	12.08	18:36

Рисунок 2.11 – STATISFIES (графік, час вилета→рейс)

Якщо уявити сукупність різних значень доменів у вигляді таблиці і застосувати алгоритм STATISFIES, з'являється можливість виявити ФЗ, які присутні в аналізованої таблиці. При цьому вибір доменів за критеріями дозволяє обмежити набір структур «підозрюваних» на наявність ФЗ. Задоволення критерієм приносить структурі деякий пріоритет розгляду. Тим самим застосування алгоритму STATISFIES до найбільш пріоритетним структурам дозволить не тільки обмежити набір розглянутої інформації, а й остаточно упевнитися в наявності ФЗ.

2.6.3. Кластеризація

Розбиття таблиці на домени дозволить виділити основні ознаки збережених об'єктів і вибрати з них найбільш підходящі для кластеризації.

Кластеризація - це автоматичне розбиття елементів деякої множини на групи в залежності від їх схожості. Елементами безлічі може бути що завгодно, наприклад, дані або вектора характеристик. Самі ж групи прийнято називати кластерами. У Fractal Data Mining кластеризация виконується

природно, орієнтуючись на відстань між об'єктами, тобто вибір найбільш близьких один до одного. У зв'язку з тим, що розмірність (кількість вхідних в досліджуваний об'єкт атрибутів) об'єкта в таблиці може бути висока, обчислення відстані між об'єктами при високої розмірності вельми ускладнює процес виділення кластерів. Поділ набору атрибутів на частини - домени, дозволить провести кластеризацію на обмеженому наборі характеристик, працювати лише об'єктами відносини на рівні частин, які мають меншу розмірність, ніж об'єкт цілком. У загальному вигляді алгоритм можна представити у вигляді блок-схеми рис. 2.12.

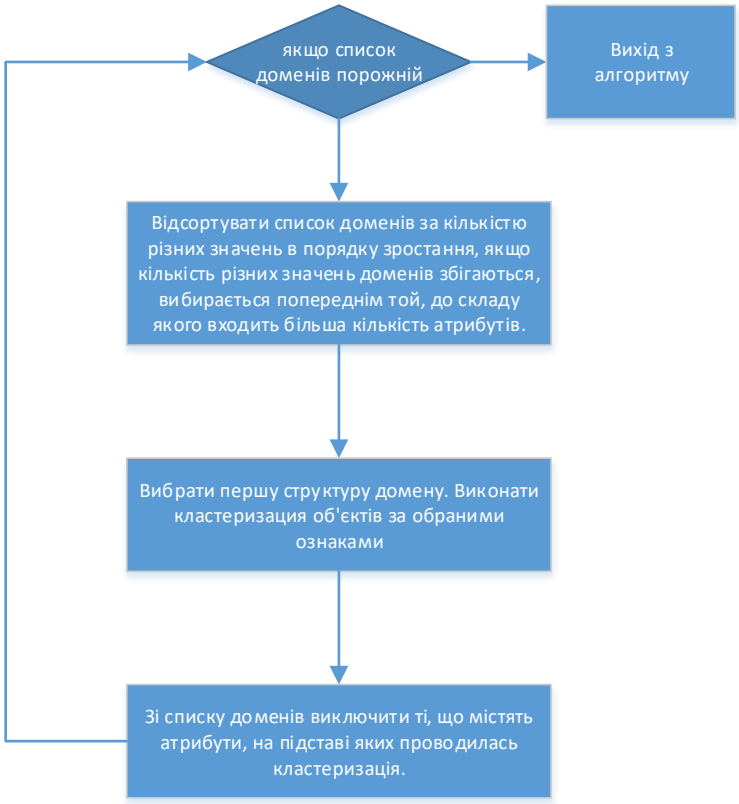

Рисунок 2.12 – Алгоритм кластеризації

Сам процес кластеризації будується на основі розробленого алгоритму «Фрактальна кластеризація» [2]. Єдиним внесеним зміна буде той факт, що кластер відразу будується на основі метрика Хаусдорфа, а не відстані між конкретно взятими значеннями. Побудований алгоритм кластеризації

дозволяє провести аналіз даних природним шляхом, грунтуючись на самоподобу об'єктів розглянутого відносини.

2.7. Структура програмної системи

Програмна система повинна не тільки реалізовувати описані вище алгоритми, а й здійснювати ефективну взаємодію з базою даних. Як один з розумних варіантів для реалізації програмної системи, була вибрана мова програмування Java. Мова програмування Java є об'єктно-орієнтованим, тому спроектуємо діаграму класів. Визначимо основні об'єкти, які визначають основну логіку програмної системи і відповідно реалізують описані вище алгоритми.

1. Клас «Domen» - описує структуру найменшого основного об'єкта - домену.

2. Клас «Domen_Collection» реалізує основні алгоритми для отримання списку доменів, який дозволить побудувати оптимальне безліч і виділити функціонально залежні атрибути.

3. Клас «Clusters» призначений для формування кластерів.

Далі опишемо допоміжні об'єкти, що описують структури для зберігання оброблюваних програмою даних.

1. Клас «Atribute» - задає структуру, яка описує стовпець відносини.

2. Клас «Table» - задає структуру, згідно з якою будуть зберігатися відомості про аналізованому відношенні.

3. Клас «Opt_collect» - задає структуру для зберігання оптимальної множини

Опишемо кожен об'єкт більш докладно. Спочатку розглянемо допоміжні об'єкти:

Клас «Atribute» представлено наступним набором атрибутів: ім'я стовпця; тип даних; тип гістограми; функція розподілу. У класу тільки один метод - побудова функції розподілу.

Клас «Table» представлено наступним набором атрибутів: список атрибутів відносини; потужність відносини; ім'я таблиці. У класі реалізовані наступні методи: побудова статистики, отримання відносини.

Клас «Opt_collection» представлено набором наступних атрибутів: список доменів, що становлять безліч; кількість записів; ознака повного заповнення. У класу тільки один метод - запис зберігає оптимального безлічі в файл.

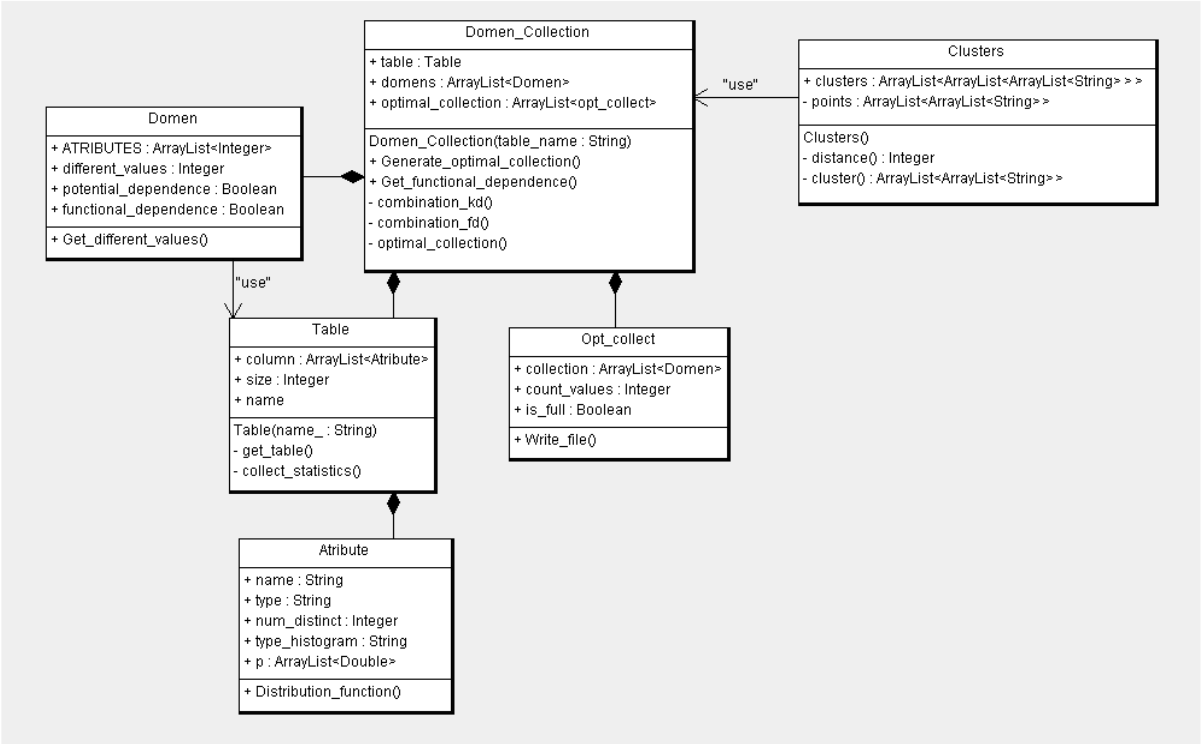

Рисунок 2.13 – Діаграма класів.

Далі більш детально опишемо основні об'єкти.

Клас «Domen». Атрибутами цього класу є: список індексів, які є посиланнями на атрибути таблиці, складових домен; кількість різних значень; ознака наявності потенційної функціональної залежності; ознака наявності функціональної залежності. У класу тільки один метод - підрахунок кількості різних значень. Клас «Domen_Collection». Атрибутами цього класу є: список доменів; інформація про таблиці (об'єкт «Table»);

список оптимальних множин (так як різний набір доменів може тиснути однакову кількість записів). Визначимо інтерфейсні методи класу:

- метод визначення функціональних залежностей;
- метод для побудови оптимальної множини.

Допоміжними методами класу є:

- побудова структур доменів, в які входять атрибути зі схожими розподілами даних.
- побудова структур доменів в залежності від оцінки параметра KD.
- реалізація рекурсивної функції побудови оптимального безлічі optimal_collection.

Клас «Clusters». Атрибути класу: список побудованих кластерів; набір точок, який необхідно розподілити по кластерам. Методи класу:

- розрахунок відстані між точками по метриці Дарбау-Левенштейна.
- побудова кластера на основі метрики Хаусдорфа.

Загальний вигляд діаграми класів представлений на рис. 2.13.

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

3.1. Тестова система

Тестування будемо проводити в два етапи:

1. Перевірка працездатності на прикладі простого відносини

2. Дослідження ефективності на прикладі структур реальних відносин.

Перше відношення, просте, взято виключно в цілях перевірки працездатності алгоритмів. Як простий приклад візьмемо таблицю «Деталі» (D) зі стандартної завдання «Постачальник-споживач», що розглядається в курсі баз даних [10]. Структура відносини D представлена в таблиці 3.1.

Таблиця 3.1 – Структура таблиці D

№	Ім'я атрибута	Опис атрибуту
1	KOD_D	Первинний ключ
2	NAME_D	Найменування деталі
3	CITY_D	Місто постачальника деталі
4	V_D	об'єм деталі
5	COLOR_D	Колір деталі

Заповнимо таблицю D таким чином, щоб була явна функціональна залежність City_D → Name_D таблиця 3.2.

Таблиця 3.2 – Дані в таблиці D

№	Name_D	City_D	Volume_D	Color_D
1	Гайка	Тула	5	червоний
2	Гайка	Тула	5	синій
3	Болт	Сургут	10	червоний
4	Болт	Сургут	10	синій
5	Гайка	Тула	10	червоний

Продовження таблиці 3.2

№	Name_D	City_D	Volume_D	Color_D
6	Гайка	Тула	10	синій
7	Болт	Сургут	20	черний
8	Шуруп	Челябінськ	5	черний
9	Шуруп	Челябінськ	20	черний
10	Болт	Сургут	5	черний
11	Болт	Сургут	20	черний
12	Гайка	Тула	5	черний
13	Гайка	Тула	20	черний
14	Шуруп	Челябінськ	5	червоний
15	Шуруп	Челябінськ	5	синій
16	Шуруп	Челябінськ	10	червоний
17	Шуруп	Челябінськ	10	синій
18	Шуруп	Челябінськ	6	зелений
19	Гайка	Тула	6	зелений
20	Болт	Сургут	6	зелений

Однак в реальних базах даних кількість кортежів набагато більше 20. Як було зазначено раніше, на відміну від більшості конкуруючих методів аналізу, в яких великі набори даних є проблемою, метод на фрактальної основі страждає тільки тоді, коли даних занадто мало.

В якості другого відносини будуть розглянуті структури таблиць з реального програмного комплексу. Як приклад розглянемо програмний комплекс «єдиного державного реєстру земель» (далі ПК ЕГРЗ) [18], призначений для ведення державного земельного кадастру на рівні кадастрового району. Структура бази даних ПК ЕГРЗ змінювалася залежно від змін в законі, появи наказів. Постійне внесення змін до структури бази даних привело до появи функціональних залежностей і збільшення кількості атрибутів в таблиці. В якості тестових відносин будуть розглянуті таблиці OBJLOT, що містить відомості про характеристики земельних ділянок, і OBJ, що містить перелік усіх об'єктів.

Таблиця OBJLOT складається з 53 атрибутів табл. 3.3, кількість кортежів 10568.

Таблиця 3.3 – Дані в таблиці D

№	Опис атрибута	Ім'я атрибута
1	Площа декларована	SQDECL_OBJLOT
2	Номер заявки	NUMZAYAV_OBJLOT
3	Дозволений вид використання	RAZRVID_OBJLOT
4	Площадь уточненная	SQTOCH_OBJLOT
5	Хто врахував уточнену площу	SQTOCHKTO_OBJLOT
6	Дата обліку уточненої площі	SQTOCHDATE_OBJLOT
7	Хто реєстрував уточнену площу	SQTOCHRKTO_OBJLOT
8	Коли зареєстрована уточнена площа	SQTOCHRDATE_OBJLOT
9	Тип земельної ділянки	TYPE_OBJLOT
10	Положення земельної ділянки на чергової кадастровій карті	POLDKK_OBJLOT
11	Найменування виду земельної ділянки	NAMEVID_OBJLOT
12	Фактичне використання земельної ділянки	FAKTISP_OBJLOT
13	Місцезнаходження в межах земельної ділянки	MESTOINGRAN_OBJLOT
14	Похибка визначення площі земельної ділянки	SQDELTA_OBJLOT
15	Кадастровий номер земельної ділянки, до складу якого входить об'єкт	KNSOSTAV_OBJLOT
16	Дата обліку раніше врахованого ділянки	DATE_RANEE_OBJLOT
17	Ділянка з оціночної опису земельних ділянок	LOTOCENKA_OBJLOT
18	Код Окатий земельної ділянки	OKATO_OBJLOT
19	Код КЛАДР земельної ділянки	KLADR_OBJLOT

Продовження таблиці 3.3

№	Опис атрибута	Ім'я атрибута
20	Умовний номер земельної ділянки	CONVNUM_OBJLOT
21	Повна адреса земельної ділянки по документу	FULLADR
22	Дата проведения кадастрових робіт	CADWORKDATE_OBJLOT
23	Первинний ключ	ID_OBJ
24	Вид дозволеного використання з довідника	ID_RAZRKLS
25	Вид фактичного використання з довідника	ID_FAKTKLS
26	Зовнішній ключ до таблиці документів	ID_RID
27	Зовнішній ключ до таблиці номерних характеристик адреси земельної ділянки	ID_ASNUM
28	Регіон за довідником КЛАДР	KLADR_REGION
29	Тип регіону за довідником КЛАДР	KLADR_TYPEREGION
30	Район за довідником КЛАДР	KLADR_RAYON
31	Тип району за довідником КЛАДР	KLADR_RAYONTYPE
32	Тип району за довідником КЛАДР	KLADR_CITY
33	Тип города по справочнику КЛАДР	KLADR_TYPECITY
34	Найменування населеного пункту по довідником КЛАДР	KLADR_NASELPUNCT
35	Тип населеного пункту за довідником КЛАДР	KLADR_TYEPNASELPUNCT
36	Найменування вулиці за довідником КЛАДР	KLADR_STREET
37	Тип вулиці за довідником КЛАДР	KLADR_TYPESTREET
38	Номер будинку за довідником КЛАДР	KLADR_HOME
39	Корпус за довідником КЛАДР	KLADR_KORPUS

Продовження таблиці 3.3

№	Опис атрибута	Ім'я атрибута
40	Будова за довідником КЛАДР	KLADR_STROENIE
41	Інша в адресі	KLADR_INOE
42	Рівень відповідності КЛАДР	KLADR_SOOTVETSTVIE
43	Повна адреса по КЛАДР	KLADR_FULLADR
44	Орієнтир	KLADR_ORIENTIR
45	Топоним	KLADR_TOPONIM
46	Індекс за довідником КЛАДР	KLADR_INDEX
47	Відстань від орієнтиру	KLADR_PRIMERNOV
48	Напрямок	KLADR_NAPRAVLENIE
49	Кадастровий інженер	ID_ENGINEER
50	Район міста	KLADR_CITYRAYON
51	Тип району міста	KLADR_TYPECITYRAYON
52	Найменування сільради	KLADR_SELSOVET

Таблиця OBJ складається з 16 атрибутів таблиця 3.4, кількість кортежів 63636.

Таблиця 3.4 – Структура таблиці OBJLOT

№	Опис атрибута	Ім'я атрибута
1	Первинний ключ об'єкта	ID_OBJ
2	Тип об'єкта (округ, ЗУ, нерухомість та інше)	TYPE_OBJ
3	Найменування об'єкта	NAME_OBJ
4	Графічний ідентифікатор об'єкта	GID_OBJ

Продовження таблиці 3.4

№	Опис атрибута	Ім'я атрибута
5	Кадастровий номер об'єкта	KN_OBJ
6	Частина Кадастрового номера поточного рівня і типу об'єкта	KNLEVEL_OBJ
7	Шаблон відображення кадастрового номера	KNMASK_OBJ
8	Примітка об'єкта	REM_OBJ
9	Статус об'єкта	STATUS_OBJ
10	Площа об'єкта	SQUARE_OBJ
11	Старий кадастровий номер об'єкта	OLDKN_OBJ
12	Номенклатура планшетів об'єкта	PLANSH_OBJ
13	Уровни КД интерфейс	KDLEVEL_OBJ
14	Умовний номер об'єкта	CONDNUM_OBJ
15	Ідентифікатор батьківського об'єкта	IDParent_OBJ
16	Зовнішній ключ до таблиці GEOMAP(Ідентифікатор положення об'єкта на ДКК)	ID_GEOMAP

Другий приклад, побудований на реальних таблицях з ПК ЕГРЗ, є показником того, що структура бази даних, яка схильна до постійних модифікацій може привести до збільшення функціональних залежностей, а так само до «громіздкість» таблиць.

3.2. Результати експериментів

В описаній вище системі тестів проводять такі експерименти:

1. Порівняння алгоритмів пошуку доменів.
2. Дослідження на функціональні залежності.
3. Побудова оптимального множини.
4. Побудова кластерів.

3.2.1. Порівняння алгоритмів пошуку доменів

У попередній частині для формування безлічі доменів було запропоновано два алгоритми побудови списку можливих структур доменів.

Перший алгоритм - повний перебір всіх можливих структур, дозволяє цілком точно оцінити кількість одержані доменів за формулою:

$$\sum_{k=1}^{n} C_n^k = \sum_{k=1}^{n} \frac{n!}{k!(n-k)} \tag{3.1}$$

Алгоритм пошуку доменів за критеріями не дає можливість точно обчислити кількість можливих структур доменів. Багато в чому кількість елементів множини доменів буде залежати від обраних параметрів e і KD, тому окремо проведемо дослідження з різними значеннями:

- параметра e;
- параметра оцінки KD.

Параметр e задає допустиму похибку при порівнянні відносин смуг гістограм двох стовпців. Чим менше параметр e, тим більше схоже розміщення даних в стовпчиках. Розглянемо різні значення параметра e на прикладі аналізу частотних гістограм, побудованих по відношенню OBJLOT таблиця 3.5.

Таблиця 3.5 – Результати для різного параметра e.

Значення e	Кількість доменів	Домени
0.01	1	«KLADR_REGION | KLADR_TYPEREGION»
0.02	2	«KLADR_RAYON | KLADR_RAYONTYPE»; «KLADR_REGION | KLADR_TYPEREGION»

Продовження таблиці 3.5

0.05	2	«KLADR_REGION | KLADR_TYPEREGION»; «LOTOCENKA_OBJLOT | KLADR_RAYON
0.09	3	KLADR_REGION | KLADR_TYPEREGION»; «LOTOCENKA_OBJLOT | KLADR_RAYON | KLADR_RAYONTYPE»; «TYPE_OBJLOT | NAMEVID_OBJLOT»
0.3	3	«TYPE_OBJLOT | NAMEVID_OBJLOT»; «LOTOCENKA_OBJLOT | OKATO_OBJLOT | KLADR_RAYON | KLADR_RAYONTYPE | ID_ENGINEER»; «KLADR_REGION | KLADR_TYPEREGION»
0.32	3	«TYPE_OBJLOT | NAMEVID_OBJLOT»; «KLADR_REGION | KLADR_TYPEREGION»; «LOTOCENKA_OBJLOT | OKATO_OBJLOT | KLADR_RAYON | KLADR_RAYONTYPE | KLADR_NASELPUNCT | ID_ENGINEER»
0.34	3	«TYPE_OBJLOT | NAMEVID_OBJLOT»; «KLADR_REGION | KLADR_TYPEREGION»; «LOTOCENKA_OBJLOT | OKATO_OBJLOT | KLADR_RAYON | KLADR_RAYONTYPE | KLADR_NASELPUNCT | ID_ENGINEER»; «MESTOINGRAN_OBJLOT | FULLADR»

Як видно з таблиці 3.5, чим більше значення параметра e, тим більше атрибутів можна позначити функціонально залежними. При найменшому значенні e перший домен, потенційно містить ФЗ, є «KLADR_REGION | KLADR_TYPEREGION », який містить значення« Челябінська | обл ». Отриманий домен дійсно містить функціональну залежність, так як на території України є єдиний суб'єкт Челябінська область. При збільшенні

значення параметра e додається домен «KLADR_RAYON | KLADR_RAYONTYPE », з наступними прийнятими значеннями: « Чебаркульском | район »,« Чебаркульском | р-н »,« р Чебаркуль | МО »,« Чебаркульском міської округ | тер ». Дійсно, цей домен містить неявно ФЗ, однак чистої ФЗ в домені не міститься, так як тип району було внесено двома різними способами. При подальшому збільшенні параметра e до домену «KLADR_RAYON | KLADR_RAYONTYPE »приєднався атрибут« LOTOCENKA_OBJLOT », але чи внесений земельну ділянку з оціночної опису чи ні, не залежить від місця розташування. Атрибут «LOTOCENKA_OBJLOT» потрапив в отриманий домен тільки тому, що має малу кількість різних значень і майже однаково розподілених по смугах, щоб потрапити в довірений інтервал. При подальшому збільшенні параметра e отримали наступний домен «TYPE_OBJLOT | NAMEVID_OBJLOT »зі значеннями:« 2 | Умовний ділянку »,« 0 | Землекористування »,« 2 | Відокремлений ділянку »,« 3 | Багатоконтурний »,« 1 | Єдине землекористування ». Як видно з даних, домен не містить чистої ФЗ, однак інформація, що міститься в ньому, має важливий сенс: земельні ділянки з типом «2» не вносяться в підсумкові звіти, так як інакше виходить подвійний облік. При подальшому збільшенні значення e атрибути додаються до домену «KLADR_RAYON | KLADR_RAYONTYPE », що не мають логічної залежності, тому для таблиці OBJLOT найбільш підходящим значенням параметра e = 0.09

Обчислення параметра оцінки KD (кількість різних значень домену) - це процес підрахунку унікальних значень для домену. Точне значення параметра KD можна отримати тільки через звернення до таблиці. Так як параметр KD буде обчислюватися для кожного домена, то в результаті вийде велика кількість звернень до бази даних, що сповільнить процес отримання безлічі доменів. Однак варто зазначити, що результат даного процесу збігається з результатом операції видалення кортежів-дублів в реляційної

алгебри. Відповідно за оцінку параметра KD можна взяти оцінку розміру результату виконання операції видалення дублів кортежів.

Розмір підсумкового відносини R, що повертається оператором δ (R), який застосовується до відношенню R (D) (відносини, до складу якого входять атрибути домена D) дорівнює кількості різних значень, що входять в домен атрибутів. Найчастіше подібна статистична характеристика просто недоступна, що змушує користуватися наближеними оцінками. В екстремальних випадках розмір δ (R) або збігається з кількістю кортежів в R (тобто відношення не містить дублікатів), або дорівнює 1 (всі 63 кортежі відносини однакові). Як інший варіант значення верхньої межі кількості кортежів в δ (R) доречно використовувати максимально допустиму кількість різних кортежів, виражене у формі твору різних значень атрибутів, що входять в домен. Ця величина може бути нижче в порівнянні з іншими оцінками кількості кортежів δ (R). Існує ряд правил прогнозування значень кількості кортежів δ (R), але зазвичай рекомендують наступне: обчислити мінімум з половини кількості всіх кортежів відносини і твори всіх різних значень атрибутів, що входять у відношення [9].

Так як в нашому випадку видалення дублів відбувається в одній таблиці, то виведену оцінку можна уточнити: вибрати мінімум з твору всіх різних значень атрибутів, що входять у відношення і максимуму з половини кількості всіх кортежів відносини і максимуму кількості різних значення атрибутів, що входять в домен.

Оцінивши значення параметра KD, з'являється можливість залишити тільки ті домени, які нам підходять. Умова максимально допустимого значення KD буде залежати від розв'язуваної задачі. Так, щоб розглянути всі домени значення параметра не повинно перевищувати половини кількості всіх кортежів відносини. Якщо ж вирішується завдання стиснення відносини, то значення параметра KD можна обмежити меншими значеннями, наприклад: T (R)/3, T (R)/4. Однак варто зазначити, що вирішуючи завдання

стиснення відносини, ми ніколи не зможемо отримати кількість різних записів менше, ніж є насправді.

Порівняємо результати, отримані після виконання алгоритмів повного перебору і пошуку списку доменів за критеріями.

Уявімо результати за кількістю доменів в залежності від зміни оцінки параметра KD. Як видно з таблиці 3.6, якщо KD половиною кортежів, то спостерігається великий стрибок, тому графіки представимо до значення 0,45 від загального числа кортежів (0.5 дає не розумний результат)

Таблиця 3.6 – Результати для різного параметра e.

Умова	D	OBJLOT	OBJ
Алгоритм повного перебору	31	≈ 9*10	65535
Пошук за критеріями: KD оцінено половиною кортежів відносини	31	≈11*10	8195
Пошук за критеріями: KD оцінено третину кортежів відношення	7	920516	241
Пошук за критеріями: KD оцінено чверть кортежів відносини	7	712477	227

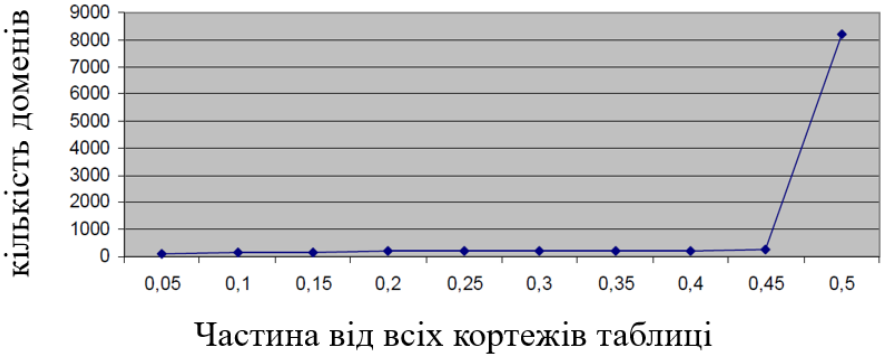

Рисунок 2.14 – Графік для таблиці OBJLOT.

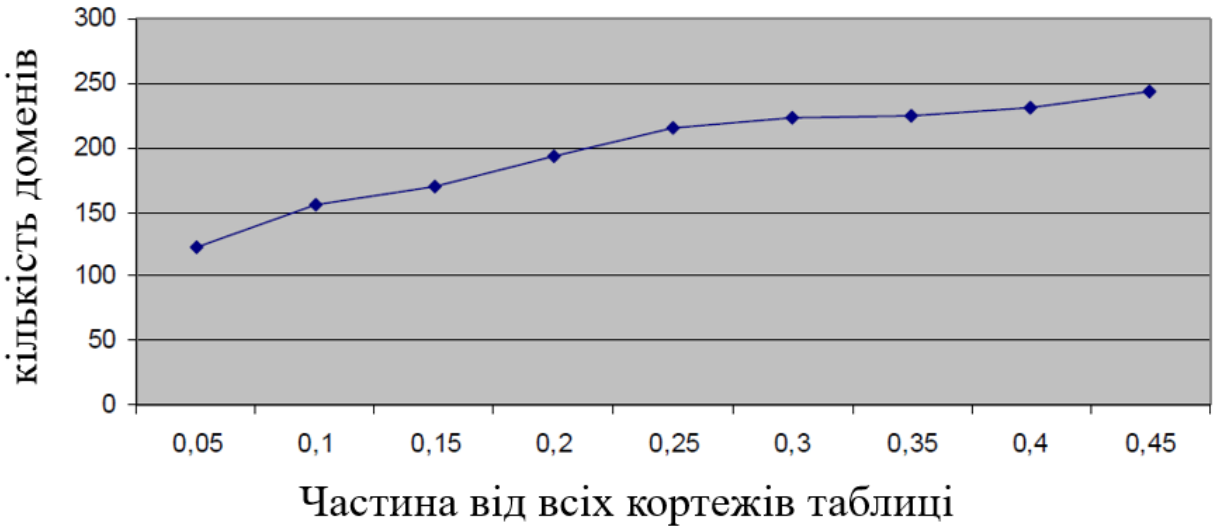

Рисунок 2.15 – Графік для таблиці OBJ.

Для того, щоб оцінити на скільки великий скачок в кількості побудованих доменів відобразимо графік для таблиці OBJ, з урахуванням оцінки KD половиною кількості кортежів відносини.

Як видно з таблиці 3.6, алгоритм пошуку доменів за критеріями дає меншу кількість можливих структур, ніж алгоритм повного перебору. Однак зменшення кількості можливих структур може відбитися на якості аналізу, тому розглянемо порівняння двох алгоритмах на завданнях Data Mining.

3.2.2. Дослідження на функціональні залежності

У тестових прикладах тільки дві таблиці містять потенційні функціональні залежності: D (таблиця «Деталі»), OBJLOT (таблиця «характеристики земельних ділянок»).

Після аналізу таблиці D була виявлена функціональна залежність City_D → Name_D. Дана функціональна залежність була закладена в таблицю, тому проводити аналіз цієї залежності не викликає особливого інтересу.

Розглянемо потенційні функціональні залежності, які були виявлені програмою в таблиці OBJLOT при проведенні аналізу:

1. TYPE_OBJLOT → NAMEVID_OBJLOT,
2. KLADR_RAYON → KLADR_RAYONTYPE,
3. KLADR_REGION → KLADR_TYPEREGION.

Проаналізуємо отриманий результат:

Розглянемо першу потенційну залежність TYPE_OBJLOT NAMEVID_OBJLOT: «2 | Умовний ділянку »,« 0 | Землекористування »,« 2 | Відокремлений ділянку »,« 3 | Багатоконтурний »,« 1 | Єдине землекористування ». Дане поєднання атрибутів не є чистою функціональної залежністю, але відмінність лише в тому, що тип «2» відповідає двом найменуванням видів земельних ділянок, таким чином, ми отримали цілком хороший результат. Якщо відзначити той факт, що при формуванні звітності види земельних ділянок «Умовні» і «Відокремлені» дільниці не включаються, то фактично ми отримали повну функнальних залежність.

KLADR_RAYON → KLADR_RAYONTYPE повинен бути представлений функціонально залежним, так як ПК ЕГРЗ-Т розрахований для ведення кадастрового обліку тільки на рівні кадастрового району. Домен містить значення: «Чебаркульском | район »,« Чебаркульском | р-н »,« р Чебаркуль | МО »,« Чебаркульском міської округ | тер ». Відсутність чистої функціональної залежності в базі даних пов'язано з різним введенням.

KLADR_REGION → KLADR_TYPEREGION дійсно містить функціональну залежність: «Челябінська | обл »,« null | null ».

Виявлені функціональні залежності дійсно мають місце бути в таблиці, проте не можна назвати їх чистими через некоректності введення даних або визначення стовпців і їх призначення. Однак не менш важливим залишається той факт, що обчислити функціональні залежності, нехай навіть і множинні, вдалося і при наявності некоректного введення даних

ВИСНОВКИ

У ході проробленої роботи по розробці алгоритмів фрактального аналізу реляційних баз даних були отримані наступні результати:

- вивчено відомі методи Fractal Data Mining.
- розглянуто уявлення таблиці як фрактала, яка не порушує структуру і зміст.
- розроблено алгоритм пошуку доменів за критеріями, побудованих на основі статистичної інформації зі словника СУБД.
- розроблено алгоритм пошуку функціональних залежностей.
- на основі фрактального подання розроблений алгоритм кодування таблиці.
- розроблено алгоритм кластеризації на множину отриманих доменів.
- реалізована програмна система для перевірки розроблених алгоритмів.
- система протестована на моделі реальних даних.

Проведені експерименти показали:

- незалежно від того, як було отримано множину доменів (алгоритмом повного перебору або за критеріями), результати інтелектуального аналізу даних мають незначні відмінності.
- наявність помилок введення даних (помилки, пропуск символу) не робить великого впливу на результат, що показує, що запропоновані методи стійкі.
- проведення кластеризації на безлічі доменів дозволяє виділити об'єкти в кластери за загальними ознаками і об'єднати їх на основі подібності.

Головним результатом магістерської роботи є не тільки побудовані алгоритми, а виявлена можливість іншого погляду на реляційну базу даних і демонстрація того, що деякі алгоритми фрактального аналізу для роботи з зображеннями можуть бути застосовні і для таблиць.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Елементи інформатики [Текст] : довідник / В. С. Височанський, А. І. Кардаш, В. С. Костєв, В. В. Черняхівський. – К. : Наук. думка, 2003. – 192 с.

2. Barbara D., Chen P.: Using the fractal dimension to cluster datasets. // KDD. 2000. С. 260-264.

3. Caetano Traina Jr., Agma J. M. Traina, Leejay Wu, Christos Faloutsos. Fast Feature Selection Using Fractal Dimension. // SBBD. 2000. С. 158- 171.

4. Kumaraswamy K., Megalooikonomou V., Faloutsos C. Fractal dimension and vector quantization. // Inf. Process. Lett. (IPL) 91(3). 2004. С. 107- 113.

5. Oracle Documentation. Database Performance Tuning Guide and Reference. Oracle9i. Release 2 (9.2). October 2002. URL: http://download.oracle.com/docs/cd/B10501_01/server.920/a96533.pdf

6. Брил Б., Луни К. Oracle Database 10g. Настольная книга администратора баз данных. Издательство «Лори», 2008. 752 с.

7. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: ДИАЛОГ-МИФИ, 2003, 384 с.

8. Гарсиа-Молина Г., Уидом Д., Ульман Д. Системы баз данных. Полный курс. Издательский дом «Вильямс», 2003, 1088 с.

9. Дейт К. Дж. Введение в системы баз данных. М.: Издательский дом «Вильямс», 2001, 1072 с.

10. Джонатан Л. Oracle. Основы стоимостной оптимизации. - Издательский дом «Питер», 2007, 528 с.

11. Кайт Т. Oracle для профессионалов. 2 том. СПб.: ООО «ДиаСофтЮП», 2003, 672 с.

12. Лымарь Т.Ю., Староверова Н.Ю. Параллельный алгоритм

фрактального поиска в базе данных // Параллельные вычислительные технологии (ПаВТ’2011): труды международной научной конференции (Москва, 28 марта - 1 апреля 2011 г.). Челябинск: Издательский центр ЮУрГУ, 2011. С. 703.

13. Мандельброт Б. Фрактальная геометрия природы. М.: Институт компьютерных исследований, 2002, 656 с.

14. Мейер Д. Теория реляционных баз данных. М.: Мир, 1987, 608 с.

15. Официальный сайт Carnegie Mellon University. Computer Science department. URL: http://www.csd.cs.cmu.edu/

16. Техническая документация по ПК ЕГРЗ. URL: http://ufo.fccland.ru/doc.aspx?doc_id=724&id=509

17. Уэлстид С. Фракталы и вейвлеты для сжатия изображений в действии. М.: Издательство Триумф, 2003, 320с

18. Міллер П.П., Мартовицький В.О. Аналіз методів редукування простору пошуку в базах знань великої розмірності. Збірник тез доповідей восьмої міжнародної науково-технічної конференції «Проблеми інформатизації»: с. 49.