В даній роботі пропонується використати методологічний апарат, розроблений для задач періодичних часових рядів ([1]). Новизна пропонованого підходу полягає у пропозиції застосувати цей обчислювальний алгоритм для даних іншої природи.

Означення 1. Коефіцієнт подібності T(A, B) – це середнє геометричне усіх попарних коефіцієнтів кореляції між розподілами для A та B за всі представлені місяці.

Використання абсолютного значення забезпечує існування кореня.

Коефіцієнт T(A, B) набуває значень у проміжку [0; 1], досягаючи значення 0 у випадку повної незалежності та 1 у випадку повної лінійної статистичної залежності.

На нашу думку, говорити про подібність, міру подібності між розподілами доцільно лише у випадку, коли ці розподіли проявляють стійкість, відтворюючи певний "паттерн" впродовж спостереженого інтервалу часу.

Означення 2. Для вивчення стабільності пропонується коефіцієнт автоподібності T(A) = T(A,A).

Якщо коефіцієнт автоподібності є високим (близьким до 1), то можна вважати, що має зміст обчислення деякого усередненого розподілу — паттерну, — який характеризує даний продукт.

Після того, як визначено методологічні інструменти для обчислення міри подібності, можна розглянути і проблему класифікації продуктів. Для цього може бути залучений кластерний та дискримінантний аналіз. Для задач прогнозування та розпізнавання використовуються методи штучного інтелекту, наприклад, метод нейронних мереж, або метод опорних векторів.

Запропонований підхід може бути корисний у прикладних маркетингових дослідженнях.

Список використаних джерел:

1. Tetyana Mamchych, Fredrik Wallin. Looking for Patterns in Residential Electricity Consumption // Energy Procedia, Volume 61, 2014, Pages 1768-1771.

ДЕРЕВОВИДНІ СТРУКТУРИ В SQL

Маркітан В. О., Возняк М. А., Булатецька Л. В.

Волинський національний університет імені Лесі Українки

Деревом називається упорядкований граф, в якому від кожного вузла до вершини веде тільки один шлях. За допомогою дерева можна представити структуру каталогів і файлів на жорсткому диску, ієрархію підрозділів і співробітників компаній. Деревовидні структури в реляційних базах даних можна зберігати декількома способами. Одним із способів побудови дерева в реляційній базі даних передбачає зберігання батьківського елементу в полі parent_id. Кожен запис в таблиці відповідає вузлу дерева і зберігає його унікальний ідентифікатор (поле id), який являється первинним ключем та посилання на батьківський вузол (поле parent_id), який описується як зовнішній

ключ, що посилається на первинний ключ цієї ж таблиці. Використання зовнішніх ключів забезпечує збереження посилкової цілісності бази даних при зміні та видаленні записів. Тому виникає проблема при видаленні вузла дерева, яке має нащадків, що пов'язані з даним вузлом зовнішніми ключами. При видаленні вузла такого дерева, для забезпечення автоматичного видалення всього піддерева потрібно використати правило ON DELETE CASCADE.

SQL запит на створення таблиці test, яка представляє описану деревовидну структуру для бази даних Oracle, можна записати наступним чином:

```
CREATE TABLE test (
id INTEGER PRIMARY KEY,
parent_id INTEGER NOT NULL REFERENCES test ON DELETE CASCADE
);
```

При такій організації дерева, додавання нового вузла, переміщення вузла до іншого батьківського вузла, видалення вузла з усіма його нащадками відбувається з використанням простих команд SQL на додавання (INSERT), оновлення (UPDATE) та видалення (DELETE) даних. Вибірку дочірніх елементів батьківського вузла можна здійснити маючи значення parent_id. Виведення нащадків нащадка батьківського вузла здіснюється шляхом з'єднання таблиці самої на себе. Відображення всього дерева є досить складним, оскільки ми повинні об'єднати таблицю саму з собою, стільки разів, скільки рівнів має наше дерево. Таким чином, це не універсальний запит, кожен раз, коли змінюється кількість рівнів дерева, цей запит потрібно буде налаштовувати.

Якщо ж глибина дерева не відома, то потрібно створити рекурсивну процедуру. Для того, щоб уникнути рекурсії список вузлів можна зберігати в одній таблці, а зв'язки в іншій.

```
CREATE TABLE test (
 id INTEGER PRIMARY KEY
);
```

Друга таблиця зберігатиме зв'язки між кожним вузлом дерева і вузлами, що знаходяться на шляху від нього до вершини, включаючи саму вершину дерева. Зв'язок задається трьома числами: ідентифікатором вузла, від якого починається шлях, ідентифікатором кінцевого вузла на шляху та відстанню — цілим числом, що дорівнює кількості дуг між ними. Тобто, відстань між нащадком і батьком дорівнює 1, між нащадком і батьком батька дорівнює 2 і т.д. аж до самого кореня.

```
CREATE TABLE test_link (
Id_from INTEGER NOT NULL REFERENCES test ON DELETE CASCADE,
Id_to INTEGER NOT NULL REFERENCES test ON DELETE CASCADE,
distance INTEGER NOT NULL,
PRIMARY KEY (Id_from, Id_to)
);
```

При такій організації даних, виведення всіх вузлів піддерева не потребує рекурсії, але ускладнюється процес додавання нового вузла, переміщення вузла до іншого батьківського вузла. Для добавлення даних потрібно буде добавляти дані в дві таблиці. Крім того в базі даних потрібно створити тригери, які будуть

створювати зв’язки між доданим вузлом і нижчими вузлами, і всіма вузлами, що знаходяться вище батьківського.

Операція видалення вузла відбувається з усіма його нащадками так як використано правило ON DELETE CASCADE.

Недоліком такої організації є велика кількість записів у таблиці зв’язків через необхідність зберігати зв’язки кожного елемента дерева з усіма його предками.

Мова запитів SQL призначена для роботи з множинами, тому можна подати деревовидну структуру за допомогою множин, для цього необхідно для кожного вузла ввести два цілих параметри: ліву та праву межу. Для зберігання такої структури даних необхідно створити таблицю:

```
CREATE TABLE test(
id INTEGER PRIMARY KEY,
parent_id INTEGER NOT NULL REFERENCES test ON DELETE CASCADE
lb INTEGER,
rb INTEGER,
parent int references t,
CONSTRAINT lb_rb CHECK(lb<rb)
);
```

Описане таким способом дерево дозволяє досягти найкращих результатів при вилученні даних, але одночасно дуже вимогливе до обчислювальних ресурсів при вставці нового запису або переміщенні вузлів.

Список використаних джерел:

1. Голуб В.М. Ієрархічна модель вкладених множин у реляційних базах даних. ВІСНИК ЛЬВІВ. УН-ТУ. Серія прикл. матем. інформ. 2010. Вип. 16, с. 106-113.

2. Деревья в SQL. *Golden Software of Belarus, Ltd. Экономическое программное обеспечение на платформе Гедымин.* URL: http://gsbelarus.com/pw/articles/post/derev-ia-v-sql/#a1 (дата звернення: 13.05.2022).

ОПТИМІЗАЦІЯ ТЕРМІНАЛЬНОГО КЕРУВАННЯ ДИСКРЕТНИМИ СИСТЕМАМИ

Матвієнко В. Т., Демківський Є. О.

Київський національний університет імені Тараса Шевченка,
matvieenko.vt@gmail.com

Задача керування по переводу системи керування з початкової точки фазового стану у фінальний стан є основною задачею теорії керування (задача термінального керування). Для систем керування з дискретним аргументом описано повну множину [1] її розв’язків. Цікавішим з практичної точки зору є