ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

Кафедра ЕОМ

Кваліфікаційна робота

Методи інтелектуального аналізу великих даних за допомогою машинного навчання

Виконала:
ст. гр. СПзм-23-1
Фатій Л.С.

Перевірив:
ас. Кравченко П.О.

Об’єкт дослідження та мета роботи

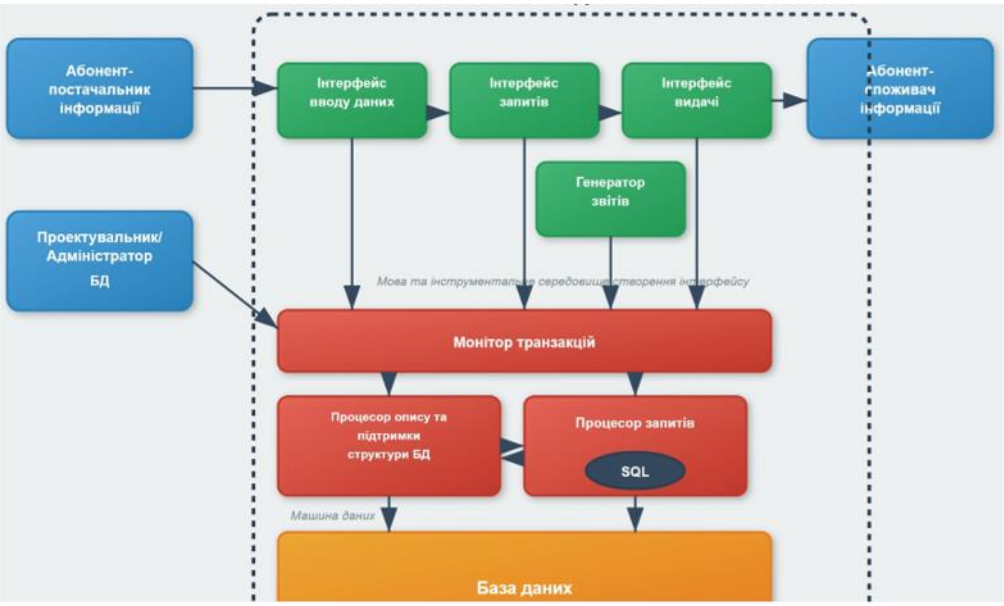
Метою кваліфікційної роботи є аналіз методів інтелектуального аналізу великих даних в паралельних системах керування базами даних.

Об’єктом дослідження є методи ІАД в реляційних СКБД.

Завдання:

- розглянути методи та алгоритми для впровадження інтелектуального аналізу великих даних в паралельну СКБД;
- проаналізувати паралельні алгоритми рішення завдань кластеризації великих даних засобами паралельної реляційної СКБД;
- розробити архітектуру паралельної СКБД з використанням модифікованого методу.

Функціональна структура системи керування базами даних [3]



Інтелектуальний аналіз великих даних [4]

```
Algorithm: Знайти найближчу пару з обмеженням відстані
Input: S_I^m - множина елементів, n - мінімальна відстань індексів
Output: {pos_bsf, dist_bsf} - позиція та відстань найкращої пари

// Ініціалізація змінних
dist_bsf ← ∞        // найкраща відстань
pos_bsf ← -1        // позиція найкращої пари

// Основний цикл по всіх елементах
for i = 1 to |S_I^m| do
    current_element ← S_I^m[i]
    dist_min ← ∞    // мінімальна відстань для поточного елемента

    // Пошук найближчого елемента з обмеженням
    for j = 1 to |S_I^m| do
        if |i - j| ≥ n then    // перевірка обмеження відстані індексів
            dist ← EditDistance(S_I^m[i], S_I^m[j])

            // Раннє припинення, якщо знайдено дуже близьку пару
            if dist < dist_bsf then
                break    // перервати внутрішній цикл
            end if

            // Оновити мінімальну відстань для поточного елемента
            if dist < dist_min then
                dist_min ← dist
            end if
        end if
    end for

    // Оновити глобальний мінімум, якщо знайдено кращу пару
    if dist_min < dist_bsf then
        dist_bsf ← dist_min
        pos_bsf ← i
    end if
end for

return {pos_bsf, dist_bsf}
```

Приклади запитів

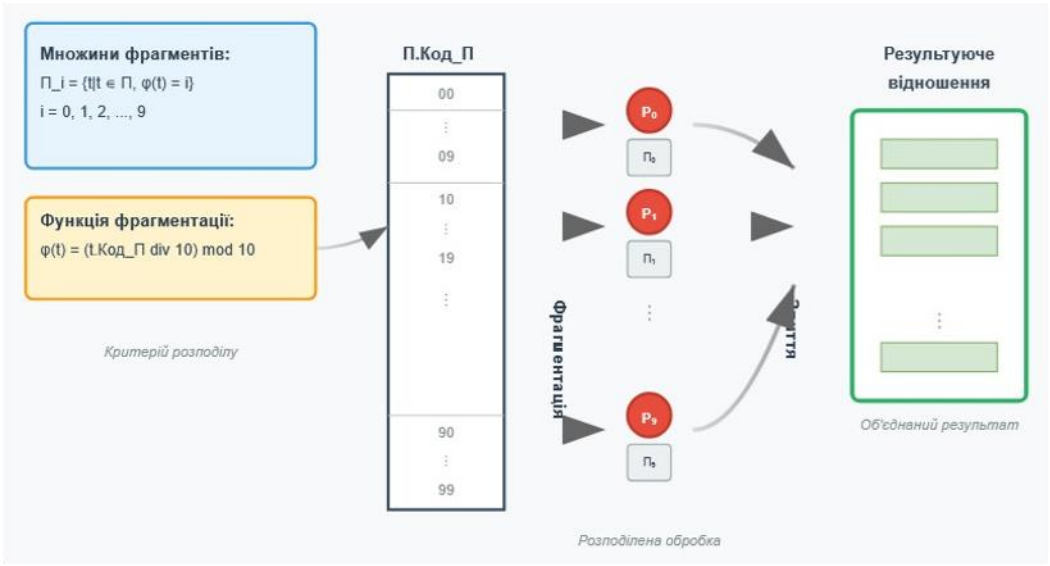
5

```
-- ============================================================
-- БЛОК 1: Пошук асоціативних правил для медичних даних
-- ============================================================

-- Знаходження асоціативних правил для пневмонії у пацієнтів старше 60
FIND ASSOCIATION RULES AS HealthRuleSet
    RELATED TO Salary, Age, isSmoker, Disease
    FROM HealthDB
    WHERE Disease = 'Pneumonia' AND Age > 60
    WITH SUPPORT_THRESHOLD = 0.05
    WITH CONFIDENCE_THRESHOLD = 0.07;

-- ============================================================
-- БЛОК 2: Видобуток специфічних правил з набору даних
-- ============================================================

-- Видобуток правил для аналізу зв'язку між курінням та пневмонією
MINE RULE HealthRuleSet AS
    SELECT DISTINCT
        1..n Disease AS antecedent,     -- передумова
        1..1 isSmoker AS consequent     -- наслідок
    FROM HealthDB
    WHERE antecedent.Disease = 'Pneumonia'
      AND antecedent.Age > 60
    EXTRACTING RULES WITH
        SUPPORT: 0.1,
        CONFIDENCE: 0.3;

-- ============================================================
-- БЛОК 3: Функція отримання правил з бази даних
-- ============================================================

FUNCTION GetRules(HealthDB)
RETURNS HealthRuleSet AS
BEGIN
    -- Отримання правил з підтримкою та довірою
    SELECT R INTO HealthRuleSet
    FROM HealthDB AS R
    WHERE R.Antecedent IN {
            (Disease = 'Pneumonia'),
            (Age > 60),
            (Salary > 1000)
        }
      AND R.Consequent IN {(isSmoker = TRUE)}
      AND R.Support > 0.1
      AND R.Confidence > 0.7;

    RETURN HealthRuleSet;
END;
```

```
-- ============================================================
-- БЛОК 4: Фільтрація та вибір найкращих правил
-- ============================================================

FUNCTION SelectRules(HealthRuleSet)
RETURNS FilteredRuleSet AS
BEGIN
    -- Вибір правил з високою якістю для пневмонії
    SELECT * FROM HealthRuleSet
    WHERE Antecedent HAS ((Disease = 'Pneumonia'))
      AND Consequent HAS {
            (Salary > 0),
            (Salary < 100000)
          }
      AND Support > 0.1
      AND Confidence > 0.7
      AND Lift > 1.0;  -- додатковий критерій якості
END;

-- ============================================================
-- БЛОК 5: Прогнозування асоціацій з використанням ML моделі
-- ============================================================

-- Прогнозування ризику пневмонії для конкретного пацієнта
SELECT
    PredictAssociation(
        [HealthMiningModel].[AssocLines],
        INCLUDE_STATISTICS,
        3  -- топ-3 асоціації
    ) AS PredictionResults
FROM [HealthMiningModel]
NATURAL PREDICTION JOIN (
    SELECT
        60 AS [Age],
        TRUE AS [isSmoker],
        'Pneumonia' AS [Disease]
) AS [PatientProfile];

-- ============================================================
-- БЛОК 6: Додатковий аналіз для покращення результатів
-- ============================================================

-- Аналіз кореляцій між факторами ризику
WITH RiskFactors AS (
    SELECT
        Age,
        isSmoker,
        Disease,
        Salary,
        COUNT(*) AS frequency,
        AVG(CASE WHEN Disease = 'Pneumonia' THEN 1.0 ELSE 0.0 END) AS pneumonia_rate
    FROM HealthDB
    WHERE Age > 60
    GROUP BY Age, isSmoker, Salary
)
SELECT
    *,
    pneumonia_rate * 100 AS risk_percentage
FROM RiskFactors
WHERE pneumonia_rate > 0.1
ORDER BY pneumonia_rate DESC;
```

6

Обробка запиту великих даних

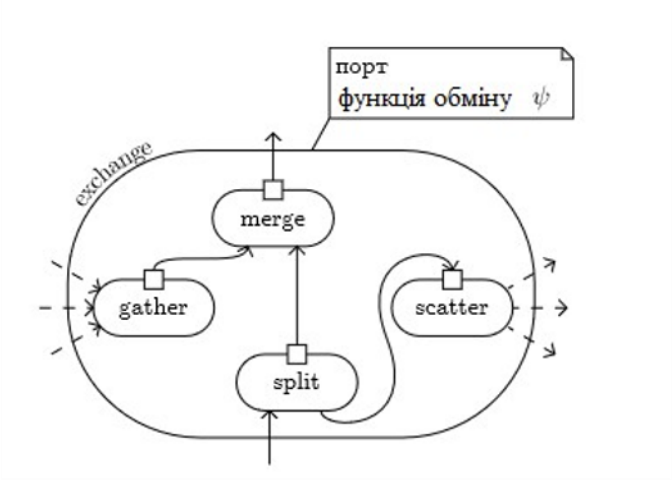


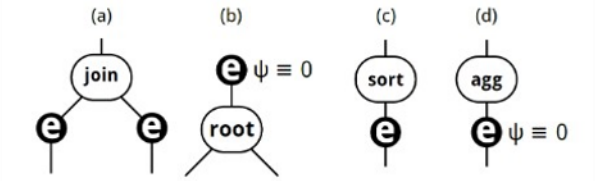
7

Схема обробки запиту в паралельній базі даних

8

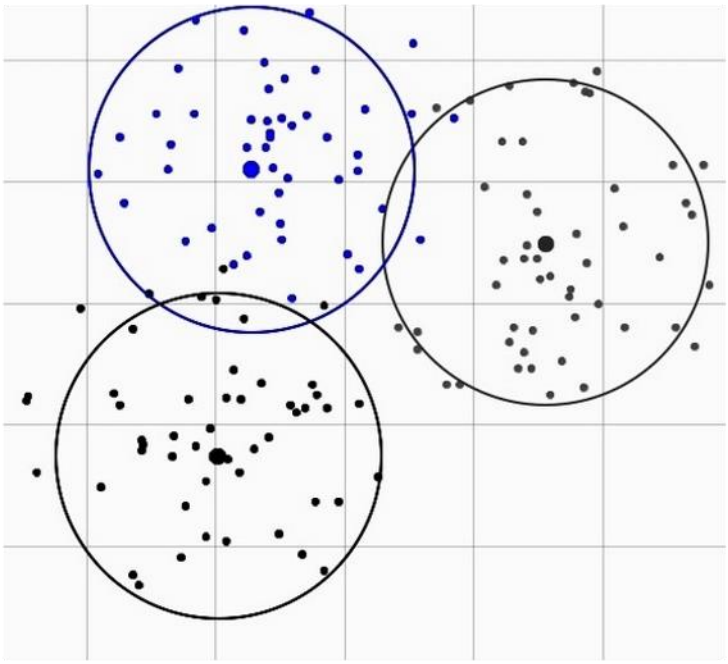
Структура оператора обміну та паралельний план запиту





9

Кластеризація та пошук шаблонів



10

Метод pgFCM

```
-- ============================================================
-- ПОВНИЙ АЛГОРИТМ FUZZY C-MEANS
-- Математичні формули та SQL реалізація
-- ============================================================

/*
МАТЕМАТИЧНІ ОСНОВИ АЛГОРИТМУ:

1. Цільова функція (мінімізується):
   J_FCM(X,k,m) = Σᵢ Σⱼ u_ij^m · d²(x_i, c_j)
                  i=1 j=1

2. Оновлення центроїдів:
   ∀j, ℓ   c_jℓ = (Σᵢ u_ij^m · x_iℓ) / (Σᵢ u_ij^m)
                  i=1              i=1

3. Оновлення коефіцієнтів членства:
   u_ij = 1 / (Σₖ (d(x_i,c_j)/d(x_i,c_k))^(2/(1-m)))
          k=1

4. Критерій збіжності:
   max |u_ij^(w+1) - u_ij^(k)| ≤ ε
    ij

Де:
- X = {x_1, x_2, ..., x_n} - множина об'єктів
- k - кількість кластерів
- m - параметр нечіткості (зазвичай m = 2)
- u_ij - ступінь належності об'єкта i до кластера j
- c_j - центроїд кластера j
- d(x_i,c_j) - відстань між об'єктом i та центроїдом j
- ε - поріг збіжності
*/

-- ============================================================
-- СТРУКТУРА ДАНИХ: Створення тимчасових таблиць
-- ============================================================

-- Таблиця матриці членства U(i,j) = ступінь належності об'єкта i до кластера j
CREATE TEMP TABLE MEMBERSHIP_MATRIX (
    object_i INTEGER NOT NULL,        -- індекс об'єкта
    cluster_j INTEGER NOT NULL,       -- індекс кластера
    membership_value NUMERIC(10,6),   -- коефіцієнт u_ij
    PRIMARY KEY (object_i, cluster_j)
);

-- Таблиця параметрів алгоритму P(d,k,n,m,delta)
CREATE TEMP TABLE ALGORITHM_PARAMETERS (
    dimensions_d INTEGER,             -- розмірність простору ознак
    clusters_k INTEGER,               -- кількість кластерів
    objects_n INTEGER,                -- кількість об'єктів
    iteration_s INTEGER,              -- номер поточної ітерації
    convergence_delta NUMERIC(10,8), -- критерій збіжності
    PRIMARY KEY (iteration_s)
);

-- Допоміжна таблиця для збереження попередніх значень SV(i,l)
CREATE TEMP TABLE SAVED_VALUES (
    object_i INTEGER NOT NULL,        -- індекс об'єкта
    feature_l INTEGER NOT NULL,       -- індекс ознаки
    feature_value NUMERIC(12,6),      -- значення ознаки
    PRIMARY KEY (object_i, feature_l)
);
```

```
-- ІНІЦІАЛІЗАЦІЯ: Початкові значення та параметри
-- ============================================================

-- Збереження поточного стану матриці членства
INSERT INTO SAVED_VALUES (object_i, feature_l, feature_value)
    SELECT object_i, 1, membership_value
    FROM MEMBERSHIP_MATRIX;

-- Збереження додаткових параметрів (розширені ознаки)
INSERT INTO SAVED_VALUES (object_i, feature_l, feature_value)
    SELECT object_i, dimensions_d, extended_value
    FROM EXTENDED_FEATURES;

-- Ініціалізація параметрів алгоритму
INSERT INTO ALGORITHM_PARAMETERS (dimensions_d, clusters_k, objects_n, iteration_s, convergence_delta)
VALUES (d_value, k_value, n_value, 0, 0.0);

-- Генерація випадкових початкових значень матриці членства
INSERT INTO MEMBERSHIP_MATRIX (object_i, cluster_j, membership_value)
VALUES (1, 1, RANDOM()),
       (1, 2, RANDOM()),
       -- ... (продовжуємо для всіх комбінацій i,j)
       (n_value, k_value, RANDOM());

-- ============================================================
-- ОСНОВНИЙ ЦИКЛ АЛГОРИТМУ
-- ============================================================

-- КРОК 1: Оновлення матриці членства за формулою FCM
UPDATE MEMBERSHIP_MATRIX
SET membership_value = (
    SELECT
        new_membership.calculated_value / normalization.total_sum AS normalized_membership
    FROM (
        -- Обчислення чисельника формули оновлення
        SELECT
            um.object_i,
            um.cluster_j,
            SUM(um.membership_value) AS calculated_value
        FROM MEMBERSHIP_MATRIX AS um
        GROUP BY um.object_i, um.cluster_j
    ) AS new_membership,
    (
        -- Обчислення знаменника (нормалізація)
        SELECT
            object_i,
            SUM(membership_value) AS total_sum
        FROM MEMBERSHIP_MATRIX
        GROUP BY object_i
    ) AS normalization
    WHERE new_membership.object_i = normalization.object_i
      AND new_membership.object_i = MEMBERSHIP_MATRIX.object_i
      AND new_membership.cluster_j = MEMBERSHIP_MATRIX.cluster_j
);
```

Реалізація обчислень методу та оновлення таблиць

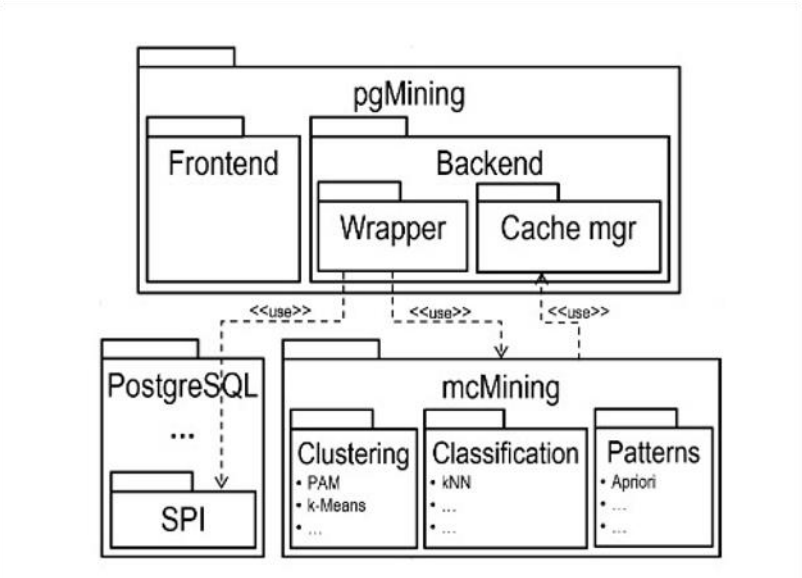
11

```sql
-- АЛГОРИТМ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ FUZZY C-MEANS
-- ============================================================

-- ============================================================
-- КРОК 1: Обчислення центроїдів кластерів
-- ============================================================
-- Розрахунок координат центрів кластерів на основі зважених середніх
INSERT INTO CENTROIDS (cluster_id, feature_id, centroid_value)
    SELECT
        cluster_j,
        feature_l,
        -- Формула зваженого середнього для центроїда
        SUM(membership.weight * feature_values.value) /
        SUM(membership.weight) AS centroid_value
    FROM (
        -- Підзапит для отримання ваг членства та значень ознак
        SELECT
            object_i,
            cluster_j,
            POWER(membership_matrix.value, fuzziness_m) AS weight
        FROM MEMBERSHIP_MATRIX AS membership_matrix
    ) AS membership,
    FEATURE_VALUES AS feature_values
    WHERE membership.object_i = feature_values.object_i
    GROUP BY cluster_j, feature_l;

-- ============================================================
-- КРОК 2: Обчислення стандартних відхилень (відстаней)
-- ============================================================
-- Розрахунок евклідової відстані від об'єктів до центроїдів
INSERT INTO STANDARD_DEVIATIONS (object_i, cluster_j, distance)
    SELECT
        object_i,
        cluster_j,
        -- Евклідова відстань до центроїда кластера
        SQRT(SUM(POWER(feature_values.value - centroids.centroid_value, 2))) AS distance
    FROM FEATURE_VALUES AS feature_values,
         CENTROIDS AS centroids
    WHERE feature_values.feature_l = centroids.feature_id
      AND feature_values.object_i = centroids.cluster_id
    GROUP BY object_i, cluster_j;
```

```sql
-- ============================================================
-- КРОК 3: Оновлення матриці членства (нечітких коефіцієнтів)
-- ============================================================
-- Обчислення нових значень функції членства для кожного об'єкта
INSERT INTO UPDATED_MEMBERSHIP (object_i, cluster_j, membership_value)
    SELECT
        current_cluster.object_i,
        current_cluster.cluster_j,
        -- Формула оновлення членства у Fuzzy C-Means
        POWER(current_cluster.distance, POWER(2.0, (1.0 - fuzziness_parameter))) *
        denominator.density_sum AS membership_value
    FROM (
        -- Підзапит для обчислення знаменника формули членства
        SELECT
            object_i,
            1.0 / SUM(POWER(distance, POWER(2.0, (fuzziness_parameter - 1.0)))) AS density_sum
        FROM STANDARD_DEVIATIONS
        GROUP BY object_i
    ) AS denominator,
    STANDARD_DEVIATIONS AS current_cluster
    WHERE denominator.object_i = current_cluster.object_i;

-- ============================================================
-- КРОК 4: Перевірка критерію збіжності
-- ============================================================
-- Обчислення максимальної зміни у матриці членства для критерію зупинки
SELECT
    MAX(ABS(new_membership.membership_value - old_membership.membership_value))
INTO convergence_threshold
FROM OLD_MEMBERSHIP AS old_membership,
     UPDATED_MEMBERSHIP AS new_membership
WHERE old_membership.object_i = new_membership.object_i
  AND old_membership.cluster_j = new_membership.cluster_j;

-- ============================================================
-- КРОК 5: Збереження параметрів ітерації та оновлення даних
-- ============================================================
-- Логування параметрів поточної ітерації
INSERT INTO ITERATION_LOG (
    dimensions,
    clusters_count,
    objects_count,
    iteration_step,
    convergence_value
) VALUES (
    dimension_d,
    clusters_k,
    objects_n,
    current_steps,
    convergence_threshold
);

-- Оновлення матриці членства для наступної ітерації
TRUNCATE TABLE MEMBERSHIP_MATRIX;
INSERT INTO MEMBERSHIP_MATRIX
    SELECT object_i, cluster_j, membership_value
    FROM UPDATED_MEMBERSHIP;
```

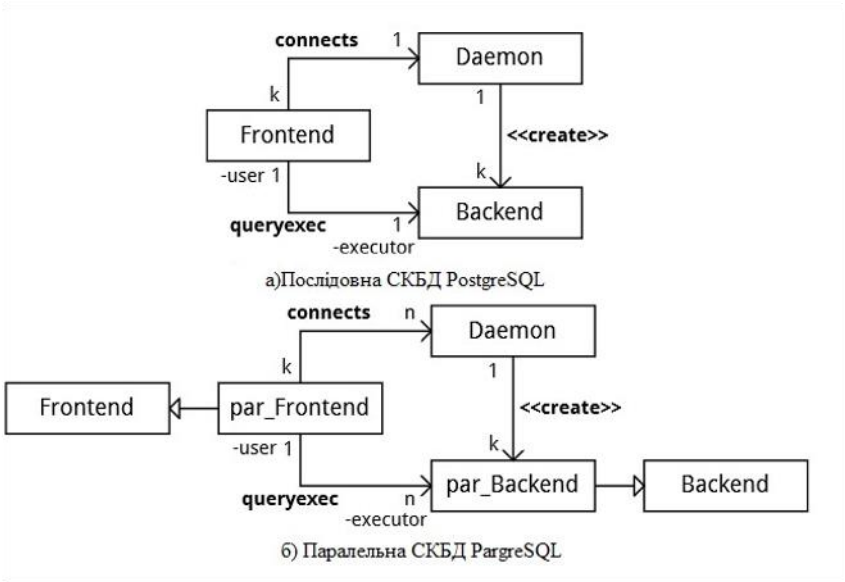
Системна архітектура інтеграції інтелектуального аналізу даних в СКБД PostgreSQL

12



Архітектура паралельної СКБД на базі PostgreSQL. Процеси СКБД

13

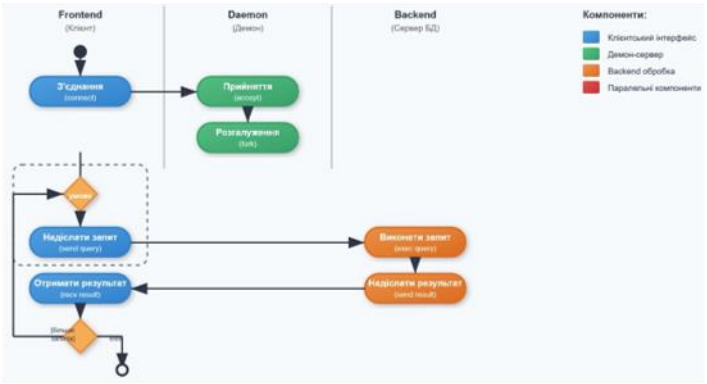


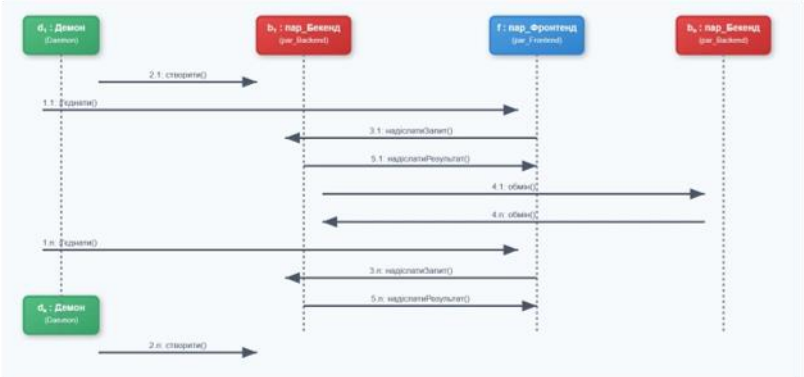
а)Послідовна СКБД PostgreSQL

б) Паралельна СКБД PargreSQL

Архітектура паралельної СКБД на базі PostgreSQL. Клієнт-серверна взаємодія

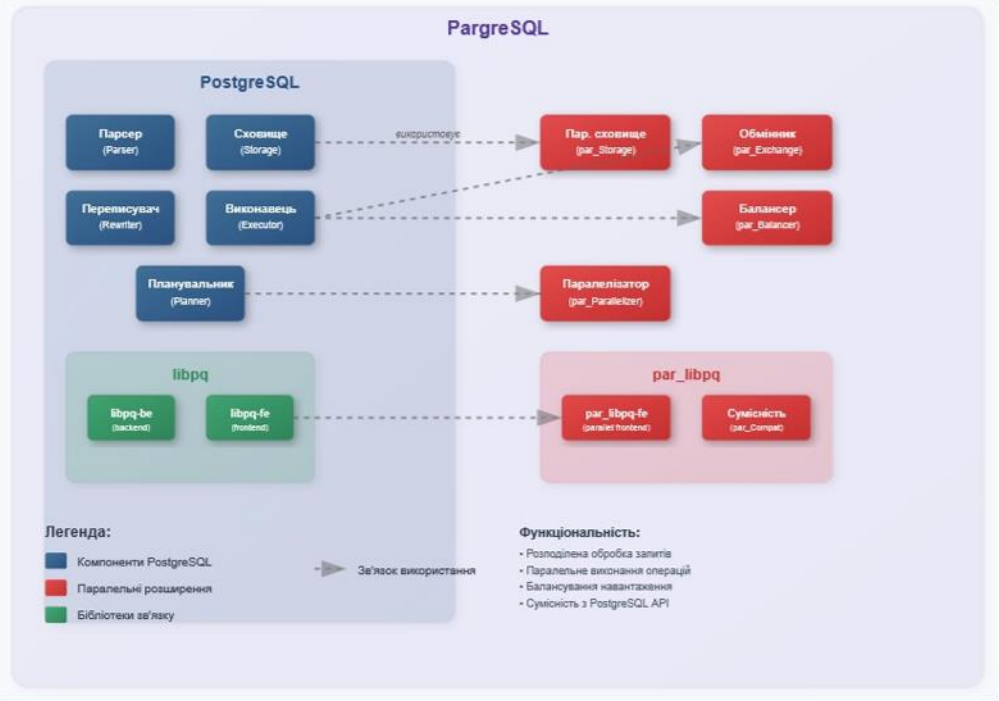
14





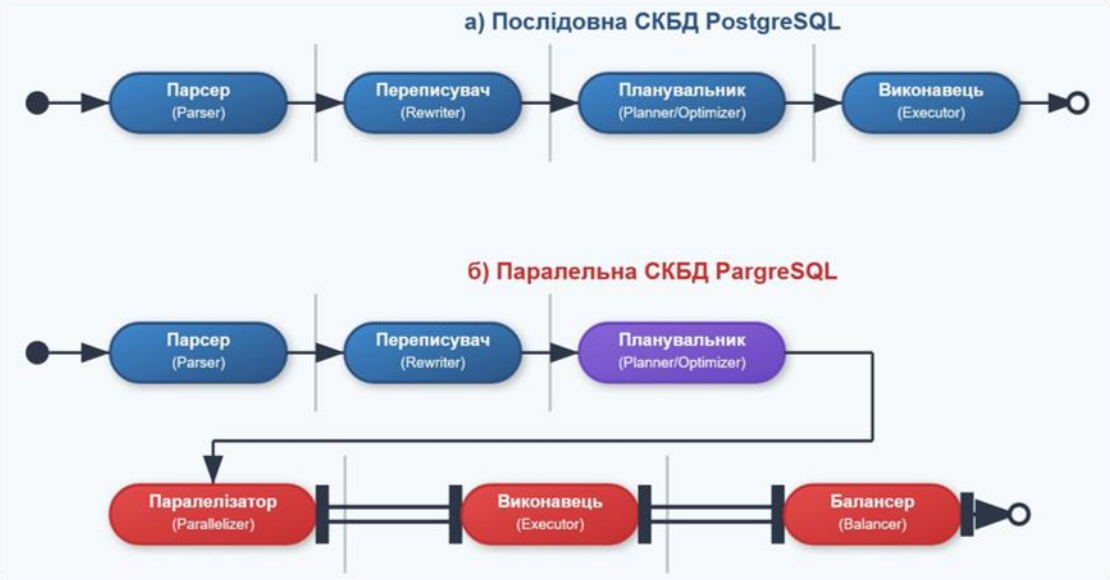
Архітектура СКБД на PargreSQL

16



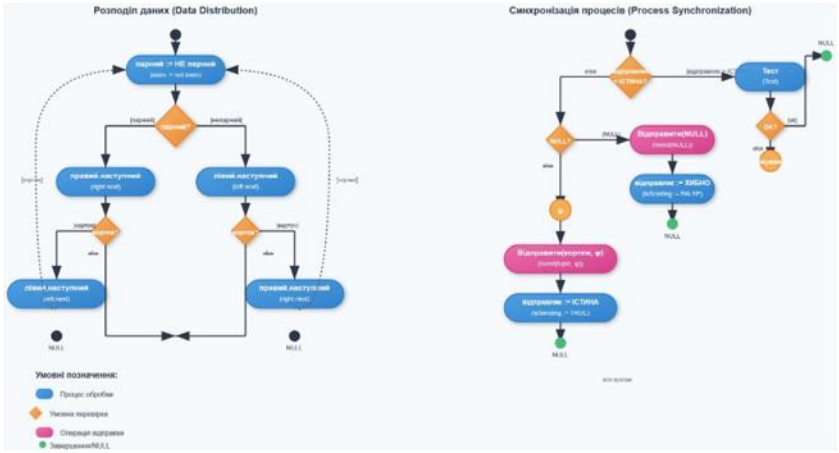
Архітектура паралельної СКБД на базі PostgreSQL. Обробка запиту

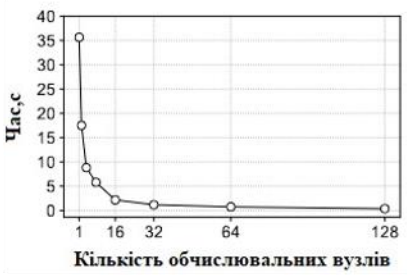
15

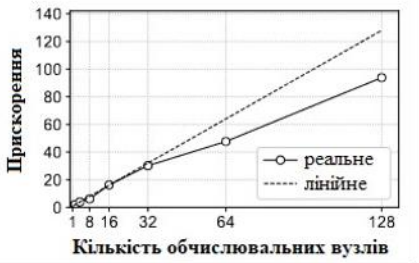


17

Реалізація деяких методів та результати







18

Висновки

У дослідженні здійснено глибокий аналіз інтеграції методів інтелектуального аналізу даних у середовище реляційних систем управління базами даних. Розглянуто концептуальні засади та практичні підходи до створення паралельних алгоритмів, орієнтованих на виконання у кластерних обчислювальних середовищах, що базуються на сучасних багатоядерних прискорювачах. Особливу увагу приділено архітектурним рішенням і принципам побудови інтегрованої платформи, реалізованої на основі відкритої СКБД PostgreSQL з використанням апаратного забезпечення типу Intel Many Integrated Core.