УДК 004.043

ИНТЕГРАЦИЯ АЛГОРИТМА КЛАСТЕРИЗАЦИИ FUZZY C-MEANS В POSTGRESQL

Р. М. Миниахметов[1], **М. Л. Цымблер**[1]

Интеграция алгоритмов интеллектуального анализа данных (ИАД) в реляционные СУБД является актуальной задачей. Реализация алгоритмов ИАД на языке SQL позволяет существенно снизить накладные расходы по организации ИАД по сравнению с использованием внешних утилит. В настоящей статье предложена реализация алгоритма нечеткой кластеризации Fuzzy c-Means для реляционной СУБД PostgreSQL с открытым исходным кодом. Работа выполнена при финансовой поддержке РФФИ (проект № 09–07–00241-а) и Минобрнауки РФ (государственный контракт № 07.514.11.4036).

Ключевые слова: нечеткая кластеризация, интеграция алгоритма кластеризации, реляционные СУБД.

1. Введение. В настоящее время технологии интеллектуального анализа данных (ИАД) являются одной из интенсивно развивающихся областей аналитической обработки данных. Интеграция алгоритмов ИАД в реляционные системы управления базами данных (СУБД) представляет собой одну из актуальных задач этой области [1, 17, 18].

Сказанное обусловлено следующими основными причинами. С одной стороны, на сегодня СУБД активно используются во всех сферах деятельности человека, связанных с хранением и аналитической обработкой больших объемов данных. С другой стороны, большинство существующих алгоритмов ИАД предполагают размещение анализируемых данных в оперативной памяти [5, 6, 12], и их совместное использование с СУБД требует значительных накладных расходов, связанных с предварительным экспортом анализируемых данных из базы данных для внешней аналитической утилиты и импортом результатов работы этой утилиты обратно в базу данных.

В настоящее время СУБД с открытым исходным кодом (PostgreSQL, MySQL и др.) получили широкое распространение и являются надежной альтернативой коммерческим СУБД [13, 24, 25]. Кроме того, имеются проекты разработки параллельных СУБД на основе СУБД с открытым исходным кодом [21–23].

Исследования по интеграции алгоритмов интеллектуального анализа данных с реляционными СУБД представлены следующими работами. Ассоциативные правила рассмотрены в [15]. Обобщенные примитивы интеллектуального анализа предложены в работе [16]. Деревья решений рассмотрены в работе [17]. Настоящая работа посвящена проблеме интеграции алгоритма нечеткой кластеризации данных Fuzzy c-Means (FCM) [2–4] со свободной СУБД PostgreSQL [7]. Наше исследование основано на работах [1, 18], в которых описана интеграция алгоритма кластеризации K-Means с реляционной СУБД. В настоящей статье эти результаты расширены для случая нечеткой кластеризации, когда векторы данных принадлежат нескольким кластерам. Нечеткая кластеризация широко используется в интеллектуальном анализе реальных данных, например медицинских [19, 20]. Насколько нам известно, работы, посвященные интеграции алгоритмов нечеткой кластеризации в реляционную СУБД, по-видимому, отсутствуют.

Статья организована следующим образом. В разделе 2 вводятся базовые определения и приводится обзор алгоритма FCM. Раздел 3 описывает реализацию алгоритма FCM на языке SQL (Structured Query Language), названную pgFCM. Вычислительные эксперименты представлены в разделе 4. Раздел 5 содержит заключение и возможные направление дальнейшей работы.

2. Алгоритм Fuzzy c-Means. Алгоритм K-Means [8] является одним из самых известных алгоритмов кластеризации, он прост и достаточно быстр [9]. Алгоритм FCM обобщает K-Means для случая нечеткой кластеризации, когда векторы данных могут принадлежать нескольким группам (*кластерам*) одновременно с некоторым весом (*степенью принадлежности*). Для описания алгоритма FCM мы используем следующие обозначения: $d \in \mathbb{N}$ — размерность пространства векторов данных; $l \in \mathbb{N} : 1 \leqslant l \leqslant d$ — номер координаты вектора; $n \in \mathbb{N}$ — мощность обучающей выборки; $X \subset \mathbb{R}^d$ — обучающая выборка векторов данных; $i \in \mathbb{N} : 1 \leqslant i \leqslant n$ — номер вектора обучающей выборки; $x_i \in X$ — i-й вектор выборки; $k \in \mathbb{N}$ —

[1] Южно-Уральский государственный университет, факультет вычислительной математики и информатики, просп. Ленина, 76, 454080, г. Челябинск; Р. М. Миниахметов, аспирант, e-mail: tavein@gmail.com; М. Л. Цымблер, доцент, e-mail: mzym@susu.ru

© Научно-исследовательский вычислительный центр МГУ им. М. В. Ломоносова

количество кластеров; $j \in \mathbb{N} : 1 \leqslant j \leqslant k$ — номер кластера; $C \subset \mathbb{R}^{k \times d}$ — матрица, содержащая центры кластеров (*центроиды*); $c_j \in \mathbb{R}^d$ — центр кластера j, вектор размерности d; $x_{il}, c_{jl} \in \mathbb{R}$ — l-е координаты векторов x_i и c_j соответственно; $U \subset \mathbb{R}^{n \times k}$ — матрица степеней принадлежности, где $u_{ij} \in \mathbb{R}$: $0 \leqslant u_{ij} \leqslant 1$ — степень принадлежности вектора x_i кластеру j; $\rho(x_i, c_j)$ — функция расстояния, определяющая степень принадлежности вектора x_i кластеру j; $m \in \mathbb{R} : m > 1$ — степень нечеткости целевой функции; J_{FCM} — целевая функция.

Алгоритм FCM основан на минимизации *целевой функции* J_{FCM}:

$$J_{\text{FCM}}(X, k, m) = \sum_{i=1}^{N} \sum_{j=1}^{k} u_{ij}^m \rho^2(x_i, c_j). \tag{1}$$

Нечеткое разбиение входного множества векторов достигается при минимизации целевой функции (1). На каждой итерации происходит обновление матрицы принадлежностей U и центроидов c_{ij} по формулам

$$u_{ij} = \sum_{t=1}^{k} \left(\frac{\rho(x_i, c_j)}{\rho(x_i, c_t)} \right)^{2/(1-m)}, \tag{2}$$

$$\forall j, l \quad c_{jl} = \sum_{i=1}^{n} u_{ij}^m \, x_{il} \left(\sum_{i=1}^{n} u_{ij}^m \right)^{-1}. \tag{3}$$

Пусть s — номер итерации, $u_{ij}^{(s)}$ и $u_{ij}^{(s+1)}$ — элементы матрицы U на шагах s и $s{+}1$ соответственно, а $\varepsilon \in (0, 1) \subset \mathbb{R}$ — критерий останова. Тогда условие завершения алгоритма выглядит следующим образом:

$$\max_{ij} \{ |u_{ij}^{(s+1)} - u_{ij}^{(s)}| \} < \varepsilon. \tag{4}$$

С каждой итерацией целевая функция (1) стремится к локальному минимуму (седловой точке) [10]. Ниже представлен базовый алгоритм FCM.

Алгоритм Fuzzy *c*-Means.

Вход: X, m, ε, k. Выход: U.

Шаг 1. $s := 0$.

Шаг 2. $U^{(0)} := (u_{ij})$.

Шаг 3 (вычисление новых координат центроидов). Вычислить $C^{(s)} := (c_j)$, используя формулу (3), где $u_{ij} \in U^{(s)}$.

Шаг 4 (обновление значений матриц). Вычислить $U^{(s)}$ и $U^{(s+1)}$ по формуле (2).

Шаг 5. $s := s + 1$.

Шаг 6. Если условие (4) не выполняется, то перейти на шаг 3.

Шаг 7. Стоп.

Таблица 1

Нумерация элементов данных

Номер	Интервал	Семантика
i	$\overline{1, n}$	номер вектора данных
j	$\overline{1, k}$	номер кластера
l	$\overline{1, d}$	номер координаты вектора

На вход алгоритма поступают множество векторов данных $X = (x_1, x_2, \ldots, x_n)$, количество кластеров k, степень нечеткости m и критерий останова ε. Результатом работы алгоритма является матрица степеней принадлежности U.

3. Реализация алгоритма Fuzzy *c*-Means на языке SQL. В данном разделе мы опишем реализацию алгоритма Fuzzy c-Means на языке SQL в качестве подхода к интеграции алгоритма нечеткой кластеризации в СУБД PostgreSQL.

3.1. Общие определения. Для интеграции алгоритма FCM с реляционной СУБД необходимо обеспечить хранение данных, которыми оперирует алгоритм (U, X), в виде реляционных таблиц. Идентификация элементов реляционных таблиц осуществляется с использованием номеров, указанных в табл. 1 (где числа n, k и d определены ранее в разделе 2).

Без ограничения общности в качестве функции расстояния $\rho(x_i, c_j)$ используется евклидова метрика:

$$\rho(x_i, c_j) = \sqrt{\sum_{l=1}^{d} (x_{il} - c_{jl})^2}. \tag{5}$$

Для нахождения максимального значения $\left| u_{ij}^{(s+1)} - u_{ij}^{(s)} \right|$ определим функцию δ следующим образом:

$$\delta = \max_{ij} \left\{ \left| u_{ij}^{(s+1)} - u_{ij}^{(s)} \right| \right\}. \tag{6}$$

Значение функции δ используется при проверке условия завершения (4).

3.2. Схема базы данных. Опишем схему базы данных для алгоритма pgFCM. Краткое описание и семантика таблиц приведены в табл. 2. Атрибуты, являющиеся первичными ключами, подчеркнуты.

Таблица 2

Схема базы данных алгоритма

№	Таблица	Семантика	Атрибуты	Кол-во записей
1	SH	Выборка векторов данных	$\underline{i}, x1, x2, \ldots, xd$	n
2	SV	Выборка векторов данных	$\underline{i, l}, val$	$n\,d$
3	C	Координаты центроидов	$\underline{j, l}, val$	$k\,d$
4	SD	Расстояния между x_i и c_j	$\underline{i, j}, dist$	$n\,k$
5	U	Степени принадлежности вектора X_i кластеру j на шаге s	$\underline{i, j}, val$	$n\,k$
6	UT	Степени принадлежности вектора X_i кластеру j на шаге s+1	$\underline{i, j}, val$	$n\,k$
7	P	Значение функции (6) на текущей итерации	$\underline{d, k, n, s}, delta$	число итераций

Для хранения выборки векторов множества X требуется определить таблицу $SH(\underline{i}, x1, x2, \ldots, xd)$, каждая запись которой хранит вектор данных размерности d с номером i. Таблица SH имеет n записей и первичный ключ i.

В ходе выполнения вычислений, предусмотренных алгоритмом FCM, требуется выполнять агрегирование (подсчет суммы, максимума и др.) координат векторов множества X. Однако, в силу своего определения, таблица SH не позволяет применять функции агрегирования языка SQL. В соответствии с этим нами определяется таблица $SV(\underline{i, l}, val)$, состоящая из $n\,d$ записей и имеющая составной первичный ключ (i, l). Таблица SV представляет собой выборку данных из таблицы SH. Структура таблицы позволяет применять агрегирующие функции языка SQL, например функции max() и sum().

Для хранения данных о координатах центроидов кластеров необходимо создать отдельную временную таблицу $C(\underline{j, l}, val)$, которая имеет $k\,d$ записей и составной первичный ключ (j, l). Как и у таблицы SV, структура таблицы C позволяет применять агрегирующие функции.

Согласно алгоритму Fuzzy c-Means, на шаге 5 алгоритма требуется определять степень принадлежности вектора i кластеру j, что включает в себя вычисление расстояний $\rho(x_i, c_j)$. Для хранения расстояний используется таблица $SD(\underline{i, j}, dist)$ с количеством записей $n\,k$ и составным первичным ключом (i, j).

Таблица $U(\underline{i, j}, val)$ хранит степени принадлежности, полученные на шаге s. Для хранения степеней принадлежности на шаге s+1 потребуется еще одна, аналогичная по структуре, таблица $UT(\underline{i, j}, val)$. Обе таблицы имеют количество записей $n\,k$, а также составной первичный ключ (i, j).

Таблица $P(\underline{d, k, n, s}, delta)$ хранит номер итерации s и значение формулы (6) для этого номера. Количество записей в таблице зависит от числа итераций, которые понадобились для завершения алгоритма.

3.3. Алгоритм pgFCM. Выполнение алгоритма инициируется вызовом хранимой процедуры на языке PL/pgSQL. Ниже показаны основные шаги алгоритма pgFCM.

Вход: $m,\ eps,\ k,\ SH$. Выход: U.

Шаг 1 (инициализация таблиц). Создать и инициализировать временные таблицы U, P, SV и др.

Шаг 2 (вычисления). Вычислить координаты центроидов, обновить таблицу C; вычислить расстояния $\forall\ y_i\ c_j \|y_i - c_j\|$, обновить таблицу SD; вычислить $UT = (ut_{ij})$, обновить таблицу UT.

Шаг 3 (обновление таблиц). Обновить таблицы P и U.

Шаг 4 (проверка завершения). Если условие $\max\limits_{ij} \|ut_{ij} - u_{ij}\| < \varepsilon$ не выполняется, то перейти на шаг 2.

Шаг 5. Стоп.

Входное множество векторов данных X хранится в таблице SH. Степень нечеткости m, критерий останова eps и количество кластеров k являются входными параметрами функции pgFCM. Конечный результат работы алгоритма pgFCM находится в таблице U.

3.3.1. Реализация шага "Подготовка". Ниже приведены команды создания временных таблиц SV, U и P (табл. 2). Для создания таблиц используется ключевое слово `TEMP`, которое указывает, что создается специальная временная таблица. Временные таблицы создаются в отдельном табличном пространстве и уничтожаются по завершении SQL-сессии. Кроме того, временные таблицы с одинаковыми именами могут

независимо использоваться разными пользователями.

```
CREATE TEMP TABLE U (i int, j int, val numeric,
  PRIMARY KEY (i,j));

CREATE TEMP TABLE P (d int, k int, n int, s int,
  delta numeric, PRIMARY KEY (d,k,n));

CREATE TEMP TABLE SV (i int, l int, val numeric,
  PRIMARY KEY (i,l));
```

3.3.2. Реализация шага "Инициализация". Перед выполнением вычислительной части алгоритма необходимо проинициализировать таблицы SV, U и P.

Инициализация таблиц SV, U и P алгоритма pgFCM:

— *инициализация таблицы* SV

```
INSERT INTO SV
  SELECT SH.i, 1, x1 FROM SH;
...
INSERT INTO SV
  SELECT SH.i, d, xd FROM SH;
```

— *инициализация таблицы* P

```
INSERT INTO P(d, k, n, s, delta)
  VALUES (d, k, n, 0, 0.0);
```

— *инициализация таблицы* U

```
INSERT INTO U (i, j, val)
  VALUES (1, 1, random());
...
INSERT INTO U (i, j, val)
  VALUES (i, j, random());
...
INSERT INTO U (i, j, val)
  VALUES (n, k, random());
```

— *нормирование степеней принадлежности*

```
UPDATE U SET val = val / U1.tmp
FROM (SELECT i, sum(val) AS tmp
      FROM U
      GROUP BY i) AS U1
WHERE U1.i = U.i ;
```

Таблица SV формируется путем выборки записей из таблицы SH.

Существует несколько подходов к инициализации координат центроидов, в данном документе используется следующий. Для таблицы U за степень принадлежности вектора x_i кластеру j принимается случайное число, которое затем нормируется. Таким образом, после нормирования соблюдаются следующие свойства [4] алгоритма Fuzzy c-Means: $\forall\, i, j \quad u_{ij} \in [0;1], \quad \forall\, i \quad \sum\limits_{j=1}^{k} u_{ij} = 1.$

При инициализации таблицы P количество кластеров k задается процедурой pgFCM и является ее параметром. Размерность пространства векторов d и мощность обучающей выборки n задаются на этапе подготовки. Номер итерации s и $delta$ инициализируются нулевыми значениями.

3.3.3. Реализация шага "Вычисление". На шаге вычислений алгоритма pgFCM производятся вычисления степеней принадлежности, центров кластеров и расстояний по формулам (2), (3), и (5) соответственно. Ниже представлен соответствующий исходный код:

— *вычисление центров кластеров*

```
INSERT INTO C
  SELECT R.j, SV.l, sum(R.s * SV.val) / sum(R.s) AS val
  FROM (SELECT i, j, U.val^m AS s
        FROM U) AS R, SV
        WHERE R.i = SV.i
        GROUP BY j, l;
```

— *вычисление расстояний*

```
INSERT INTO SD
  SELECT i, j, sqrt(sum((SV.val - C.val)^2)) as dist
  FROM SV, C
  WHERE SV.l = C.l;
  GROUP BY i, j;
```

— *вычисление степеней принадлежности*

```
INSERT INTO UT
  SELECT i, j, SD.dist^(2.0^(1.0-m)) * SD1.den AS val
  FROM (SELECT i, 1.0 / sum(dist^(2.0^(m-1.0))) AS den
        FROM SD
        GROUP BY i) AS SD1, SD
  WHERE SD.i = SD1.i;
```

Согласно алгоритму FCM, вычисление степеней принадлежности производится по формуле (2). Поскольку числитель дроби в формуле не зависит от t, то для удобства использования формулу (2) можно переписать в следующем виде: $u_{ij} = \rho^{2/(1-m)}(x_i, c_j)\left(\sum\limits_{t=1}^{k} \rho^{2/(m-1)}(x_i, c_t)\right)^{-1}$.

3.3.4. Реализация шага “Обновление”. На шаге обновления алгоритма pgFCM происходят обновления таблиц P и U:

— *обновление служебной таблицы*

```
SELECT max(abs(UT.val - U.val)) INTO tmp
FROM U, UT
WHERE U.i = UT.i AND U.j = UT.j;

INSERT INTO P
  VALUES (d, k, n, steps, tmp);
```

— *обновление таблицы степеней принадлежности*

```
TRUNCATE U;
INSERT INTO U
  SELECT * FROM UT;
```

В таблице P обновляются значения номера итерации s и значение *delta* из формулы (6). Таблица UT хранит временные значения степеней принадлежности, которые затем вносятся в таблицу U. Для быстрого удаления всех записей таблицы U, полученных на предыдущем шаге, используется оператор `truncate`.

3.3.5. Реализация шага “Проверка”. Шаг проверки является заключительным этапом алгоритма pgFCM. На каждой итерации выполняется проверка условия завершения алгоритма (4):

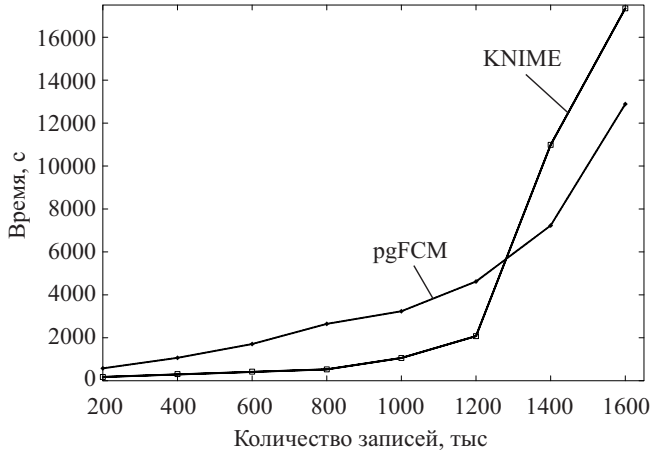
```
IF (tmp < eps) THEN
  RETURN;
END IF;
```

Для осуществления проверки используется выборка значения формулы (6) во временную переменную *tmp* процедуры pgFCM.

4. Вычислительные эксперименты. Данная статья продолжает исследование [11]. После доработки алгоритма нами были проведены вычислительные эксперименты. В этом разделе представлен график работы алгоритма pgFCM на различных наборах данных.

Эксперименты проводились на следующей аппаратно-программной платформе: процессор AMD ATHLON 64 X2 2.8 ГГц; объем оперативной памяти 3.6 ГБайт; операционная система GNU/Linux 2.6.35 x86_64; СУБД PostgreSQL версии 9.0.4; фреймворк с открытым исходным кодом KNIME [12–14] версии 2.3.4. для проведения интеллектуального анализа данных.

Для исследования эффективности работы алгоритма использовались реальные наборы данных (графические изображения) с параметрами $d = 5$, $k = 3$, $n = \overline{200000, 1600000}$. На рисунке показаны результаты работы алгоритма на реальных наборах данных различных размеров. KNIME был настроен следующим образом: количество рабочих потоков для KNIME установлено равным 1; максимально доступная память java-машины — 2.5 ГБайт; при истечении объема доступной оперативной памяти — использовать жесткий диск; для доступа к СУБД PostgreSQL использовался оригинальный драйвер JDBC, который показывает лучшие характеристики времени доступа, чем драйвер ODBC, использованный в работе [1].



Производительность алгоритма pgFCM

Эксперименты показывают, что, начиная с определенного объема данных, время работы KNIME существенно превосходит время работы алгоритма pgFCM в СУБД PostgreSQL. При больших объемах исходных данных объем оперативной памяти, необходимый для работы KNIME, становится недостаточным и фреймворк начинает использовать жесткий диск для своппинга данных, что замедляет работу KNIME. В то же время, СУБД на уровне программной архитектуры предполагает возможность эффективной реализации обработки данных, размер которых превышает объем доступной оперативной памяти.

Таблица 3

Временны́е характеристики

N, тыс.	pgFCM	KNIME		
	Выполнение, с	Выполнение, с	Экспорт, с	Импорт, с
200	578	174	3	25
400	1067	310	9	49
600	1711	423	13	75
800	2648	529	28	100
1000	3238	1061	95	125
1200	4620	2078	123	152
1400	7229	10989	161	178
1600	12888	17347	216	223

С ростом объема исходных данных увеличивается время их экспорта из базы данных и время импорта результата кластеризации в базу данных. В табл. 3 представлены результаты исследования быстродействия, а также время на выгрузку исходного множества векторов из базы данных и время на загрузку ответа в базу данных.

5. Заключение. В работе предложен алгоритм нечеткой кластеризации pgFCM. Этот алгоритм реализует алгоритм кластеризации Fuzzy c-Means и работает с данными, которые размещены в реляционной СУБД с открытым исходным кодом PostgreSQL. Проведенные вычислительные эксперименты показали эффективность алгоритма на больших объемах данных по сравнению с традиционной реализацией, предполагающей использование оперативной памяти. Дальнейшие исследования могут быть направлены на разработку параллельной версии алгоритма pgFCM.

СПИСОК ЛИТЕРАТУРЫ

1. *Ordonez C.* Programming the K-means clustering algorithm in SQL // Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining. 2004. Seattle: ACM, 823–828.

2. *Jain A.K., Murty M.N., Flynn P.J.* Data clustering: a review // ACM Computing Surveys. 1999. **31**, N 3. 264–323.
3. *Dunn J.C.* A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters // J. of Cybernetics. 1973. **3**. 32–57.
4. *Bezdek J.C.* Pattern recognition with fuzzy objective function algorithms. Norwell: Kluwer Acad. Publ., 1981.
5. *Dimitriadou E., Hornik K., Leisch F., Meyer D., Weingessel A.* Machine Learning Open-Source Package "r-cran-e1071" (http://cran.r-project.org/web/packages/e1071/index.html).
6. *Drost I., Dunning T., Eastman J., Gospodnetic O., Ingersoll G., Mannix J., Owen S., Wettin K.* Apache Software Foundation. Apache Mahout. 2010 (http://cwiki.apache.org/confluence/display/MAHOUT/Fuzzy+K-Means).
7. *Stonebraker M., Rowe L.A., Hirohama M.* The implementation of POSTGRES // IEEE Trans. on Knowledge and Data Engineering. 1990. **2**. 125–142.
8. *MacQueen J.B.* Some methods for classification and analysis of multivariate observations // Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability. Vol. 1. Berkeley: Univ. of Calif. Press, 1967. 281–297.
9. *Bradley S., Fayyad U.M., Reina C.* Scaling clustering algorithms to large databases // Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining. Menlo Park: AAAI Press, 1998. 9–15.
10. *Bezdek J., Hathaway R., Sobin M., Tucker W.* Convergence theory for fuzzy *c*-means: counterexamples and repairs // IEEE Trans. on Systems, Man, and Cybernetics. 1987. N 17. 873–877.
11. *Miniakhmetov R.* Integrating fuzzy c-means clustering with PostgreSQL // Proc. of SYRCoDIS 2011: The Seventh Spring Researchers Colloquium on Databases and Information Systems. Moscow: Moscow State Univ., 2011. 6–10.
12. *Berthold M.R., Cebron N., Dill F., et al.* KNIME — the Konstanz Information Miner: Version 2.0 and Beyond // SIGKDD Explorations Newsletter. 2009. **11**. 26–31.
13. *Chen X., Ye Y., Williams G., Xu X.* A survey of open source data mining systems // Proc. of the 2007 Int. Conf. on Emerging Technologies in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science. Vol. 4819. Berlin: Springer, 2007. 3–14.
14. *Tiwari A., Sekhar A.K.* Workflow-based framework for life science // Informatics Computational Biology and Chemistry. 2007. **31**, N 5-6. 305–319.
15. *Sarawagi S., Thomas S., Agrawal R.* Integrating association rule mining with relational database systems: alternatives and implications // Proc. of the 1998 ACM SIGMOD Int. Conf. on Management of Data. Seattle: ACM, 1998. 343–354.
16. *Clear J., Dunn D., Harvey B., Heytens M., Lohman, Mehta A., Melton M., Rohrberg L., Savasere A., Wehrmeister R., Xu M.* Nonstop SQL/MX primitives for knowledge discovery // Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. New York: ACM, 1999. 425–429.
17. *Graefe G., Fayyad U.M., Chaudhuri S.* On the efficient gathering of sufficient statistics for classification from large SQL databases // Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining. 1998. Menlo Park: AAAI, 204–208.
18. *Ordonez C.* Integrating k-means clustering with a relational DBMS using SQL // IEEE Trans. on Knowledge and Data Engineering. 2006. **18**, N 2. 188–201.
19. *Shihab A.I.* Fuzzy clustering algorithms and their applications to medical image analysis. London: Univ. of London, 2000.
20. *Zhang D., Chen S.* A novel kernelized fuzzy *c*-means algorithm with application in medical image segmentation // Artificial Intelligence in Medicine. 2004. **32**. 37–50.
21. *Пан К.С., Цымблер М.Л.* Архитектура и принципы реализации параллельной СУБД PargreSQL // Параллельные вычислительные технологии (ПаВТ-2011): труды международной научной конференции (Москва, 28 марта–1 апреля 2011 г.). Челябинск: Издательский Центр ЮУрГУ, 2011. 577–584.
22. *Paes M., Lima A.A.B., Valduriez P., Mattoso M.* High-performance query processing of a real-world OLAP database with ParGRES // VECPAR 2008. Proc. of 8th Int. Conf. (Toulouse, France, June 24–27, 2008). Lecture Notes in Computer Science. Vol. 5336. Berlin: Springer, 188–200.
23. *Kotowski N., Lima A.A.B, Pacitti E., Valduriez P., Mattoso M.* Parallel query processing for OLAP in grids // Concurrency and Computation: Practice and Experience. 2008. **20**, N 17. 2039–2048.
24. *Golfarelli M.* Open source BI platforms: a functional and architectural comparison // Proc. of the 11th Int. Conf. on Data Warehousing and Knowledge Discovery. DaWaK '09. Berlin: Springer, 2009. 287–297.
25. *Thomsen C., Pedersen T.B.* A survey of open source tools for business intelligence // Int. J. of Data Warehousing and Mining. 2009. **5**, N 3. 56–75.

Поступила в редакцию
11.04.2012