$$\frac{4767}{8461}=\frac{681}{\frac{1}{7}+\frac{4230}{\frac{1}{2}+3}}. \qquad (10)$$

Для (8), (9) и (10) получаем структурные схемы разложения чисел $\frac{379}{645}$, $\frac{583}{153}$, $\frac{4767}{8461}$ соответственно (рис. 4 а, б, в).

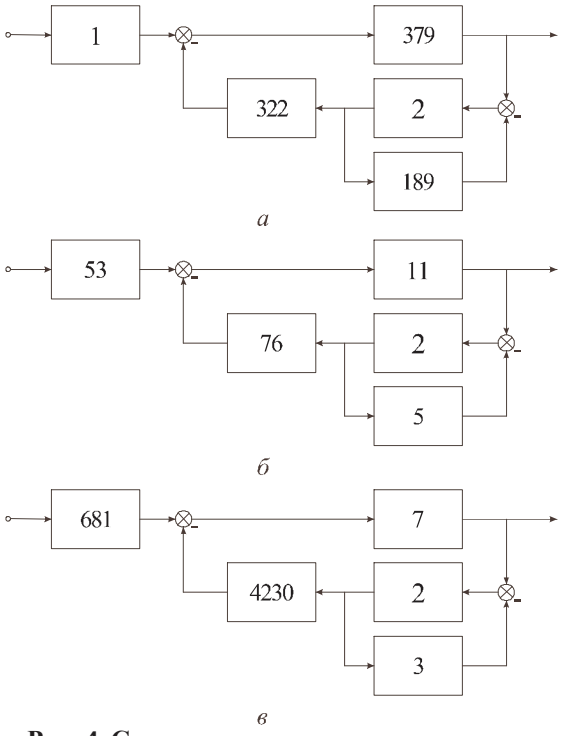

Рис. 4. Структурные схемы разложения чисел (7) - (9).

Таким образом, с помощью разработанного алгоритма, возможно представить вещественные коэффициенты $\frac{a}{b}$ (a – нечетное число, b – нечетное число) элементарных звеньев в виде определенной структуры из элементарных звеньев с целочисленными значениями параметров.

Литература

1. Алпатов Ю. Н. Синтез систем управления методом структурных графов. – Иркутск, Изд-во Иркут. ун-та, 1988. – 184 с.

2. Хинчин Д. Я. Цепные дроби. – М.: Наука, 1978. – 112 с.

3. Хованский А. Н. Приложение цепных дробей и их обобщений к вопросам приближенного анализа. – М.: Мир, 1973. – 368 с.

УДК 519.711.3, 681.51.015, 681.3.01,658.012.011.56:658.512

Ю.А. Шичкина

ПОСТРОЕНИЕ ИНФОРМАЦИОННОГО ГРАФА НОРМАЛИЗАЦИИ РЕЛЯЦИОННЫХ ОТНОШЕНИЙ

Системы баз данных являются ядром, движущей силой любой информационной системы. Обычно такие системы имеют дело с большими объемами информации, имеющей достаточно сложную структуру. Алгоритм приведения реляционного отношения к нормальной форме Бойса-Кодда основан на аппарате матричной алгебры, что позволяет его применять как на вычислительной технике с последовательной, так и параллельной архитектурой. В данной статье рассматривается процесс построения информационного графа алгоритма нормализации реляционной схемы, который явля-

ется ключевым при создании оптимальной по нескольким параметрам параллельной программы, автоматизирующей проектирование базы данных.

Ключевые слова: реляционная схема, отношения, нормальная форма, функциональная зависимость, параллельная форма, алгоритм, информационный граф.

Определение участков с внутренним параллелизмом

Для того, чтобы разработать оптимальный по времени и числу процессоров параллельный алгоритм, проведем согласно методу преобразования матрицы смежности орграфа функциональных зависимостей атрибутов реляционной схемы [2] декомпозицию последовательного алгоритма. С целью более наглядного представления последовательного алгоритма нормализации реляционной схемы поставим ему в соответствие синтаксическую диаграмму (рис. 1)

Первый блок «Стягивание матрицы» синтаксической диаграммы (рис. 1) отвечает за удаление транзитивных функциональных зависимостей в реляционном отношении.

Следующая за блоком «Стягивание матрицы» группа из пяти блоков выполняется до тех пор, пока номер текущей строки матрицы смежности орграфа функциональных зависимостей не станет равным числу строк матрицы. Этот блок включает укрупненные операции:

– создание массива, содержащего атрибуты, входящие в имена строк. Все эти атрибуты представляют собой множество атрибутов P_name, входящих в левые части функциональных зависимостей реляционной схемы;

– проверка, существует ли непустое пересечение множества атрибутов строк P_name с множеством атрибутов столбцов G, соответствующих рассматриваемой строке;

– если пересечение этих множеств является пустым, т. е. $P\_name \cap G = \varnothing$, то атрибуты данной строки I_name и множества G образуют реляционное отношение, удовлетворяющее нормальной форме Бойса-Кодда: $R = I\_name \cup G$.

– создание отношения:
$R = I\_name \cup G$;

– удаление текущей строки и всех столбцов, соответствующих атрибутам, принадлежащим множеству G, которые уже вошли в отношение, созданное на предыдущем шаге, и в другое отношение входить не могут;

– стягивание матрицы. После удаления строк и столбцов могут возникнуть транзитивные зависимости, атрибуты которых ранее входили в другие зависимости и поэтому не подлежали стягиванию. При удалении атрибутов на предыдущем шаге зависимость транзитивных атрибутов от других атрибутов может исчезнуть. Поэтому матрица подвергается процессу стягивания после каждого очередного построения нового отношения.

По окончании работы данной группы из пяти блоков матрица может состоять из одной строки, нескольких строк или не содержать строк вообще. Обработку оставшейся матрицы производит блок «Создание последнего отношения».

Как уже было отмечено, каждый из входящих в диаграмму (рис. 1) блоков представляет собой укрупненную операцию, которую можно разбить на совокупность более мелких операций.

Данная синтаксическая диаграмма отражает порядок следования операций друг за другом. При этом следует отметить, что все блоки синтаксической диаграммы (рис. 1) являются информационно связанными в таком же порядке. Поэтому синтаксическая диаграмма укрупненных операций представляет собой последовательный алгоритм, который слабо подлежит распараллеливанию в данном виде. Параллельно можно выполнять только блоки «Создание отношения» и «Удаление выделенных строки и столбцов». Полученная параллельная синтак-

сическая диаграмма представлена на рис. 2.

Отрицательной характеристикой параллельной формы синтаксической диаграммы укрупненных операций является неравнозначность временных трудоемкостей блоков «Создание отношения» ($O(n^2)$) и «Удаление выделенных строки и столбцов» ($O(n^3)$). Но с учетом того, что больше нигде процесс, выделенный для блока «Создание отношения», не задействован, экономия времени при параллельном выполнении алгоритма будет равна $O(n^2)$ без учета временных затрат на передачу данных.

Больший объем внутреннего параллелизма можно выделить при детализации блоков рассмотренной синтаксической диаграммы (рис. 1).

Детализация операции стягивания матрицы

Синтаксическая диаграмма, соответствующая операции стягивания матрицы смежности орграфа функциональных зависимостей атрибутов реляционной схемы, представлена на рис. 3. В этой синтаксической диаграмме все блоки связаны информационно, поэтому диаграмма распараллеливанию не подлежит.

Детализация операции создания массива атрибутов строк

Информационный граф синтаксической диаграммы операции создания массива атрибутов строк показывает, что данную операцию можно распараллелить несколькими способами (рис. 3, рис. 4.). При этом распараллеливать лучше не по переменной цикла с пересчетом параметра (строкам матрицы смежности), как это делается чаще всего на практике (рис. 3.), а по операциям внутри цикла: работа со строками – первый процесс, работа с массивом P_name – второй процесс (рис. 4.).

В первом случае все процессы будут иметь равномерную загруженность и временную трудоемкость $O(k^2)$, где k – общее число различных атрибутов строк, что положительно характеризует данный способ Отрицательным моментом является большое количество операций обмена данными между процессами ($O(k^m)$). Помимо этого, при данном способе распараллеливания будет достаточно сложно отследить появление дубликатов атрибутов в массиве. Наличие дубликатов в массиве атрибутов строк не повлияет на качество работы алгоритма, но увеличит объем памяти.

Во втором способе предполагается использовать всего два процесса (рис. 4.).

Трудоемкость операции «Добавление атрибута в массив» меняется по мере увеличения номера строки матрицы смежности от 4 ед. до $k+3$. В то время как трудоемкость операции «Вырезания атрибута из имени» варьируется от 2 до 3 операций. Поэтому на первых шагах операции «Создания массива атрибутов строк» оба процесса будут работать сравнительно одинаково. При обработке последних строк первый процесс будет простаивать все больше. Обмен данными за время создания массива атрибутов составит k действий. Общая трудоемкость всей операции составит $O(k^2)$, как и в предыдущем варианте.

Еще одним способом распараллеливания является разбиение процесса добавления атрибута по n процессам, где n – произвольное число процессов, допускаемое вычислительной машиной и удовлетворяющее условию $n \le k/3$. В этом варианте при $n = k/3$ загруженность всех процессов будет равномерной и равной $3k$. Объем передаваемых данных будет равен $\approx \frac{1+n}{2}k$. Общая трудоемкость равна $\approx k^2/2$. Этот вариант будет, таким образом, наиболее экономным.

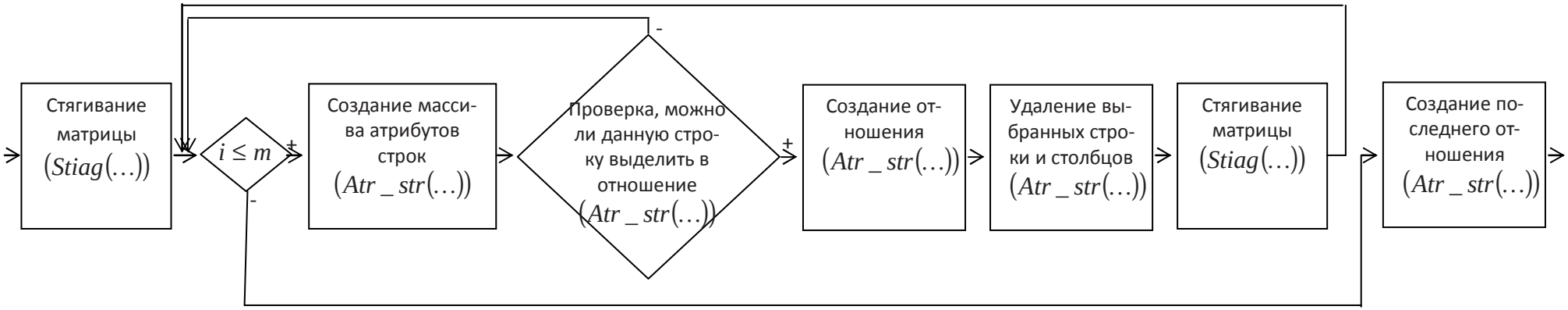

Рис. 1. Синтаксическая диаграмма последовательного алгоритма построения реляционных отношений.

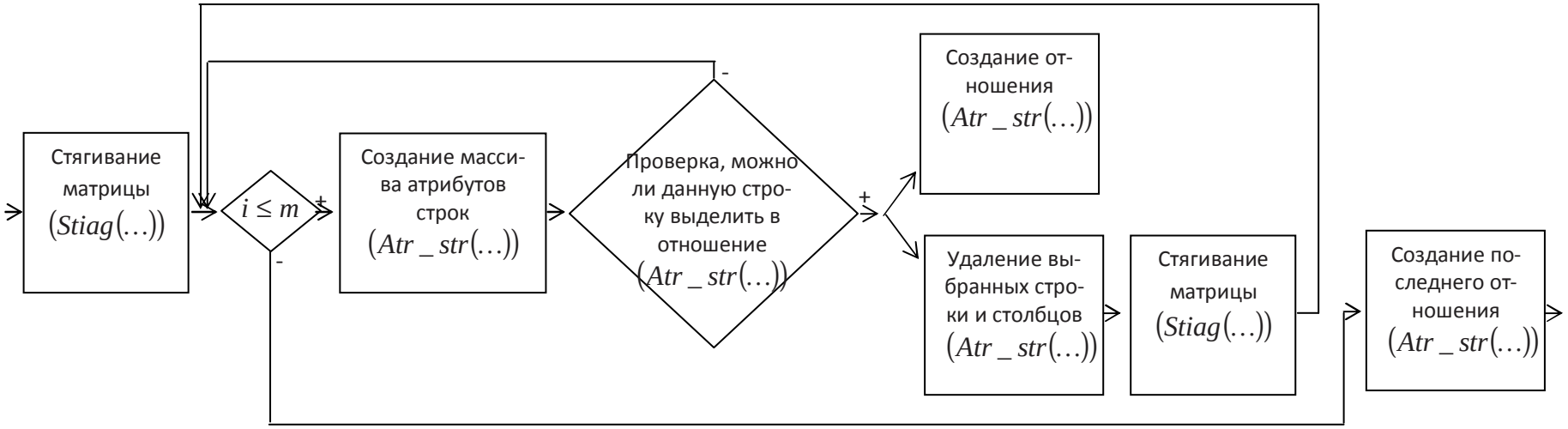

Рис. 2. Синтаксическая диаграмма параллельного алгоритма построения реляционных отношений.

.

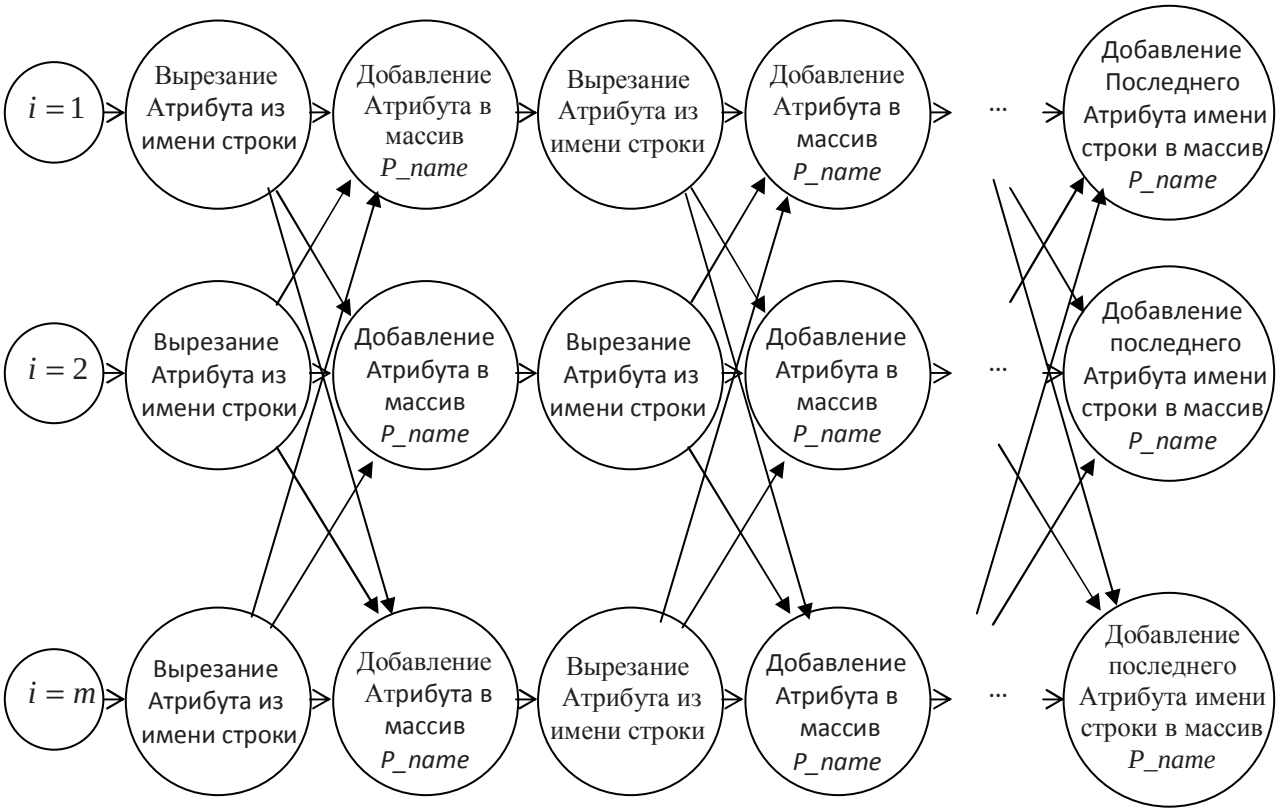

Рис. 3. Распараллеливание операции создания массива атрибутов строк по числу строк матрицы смежности.

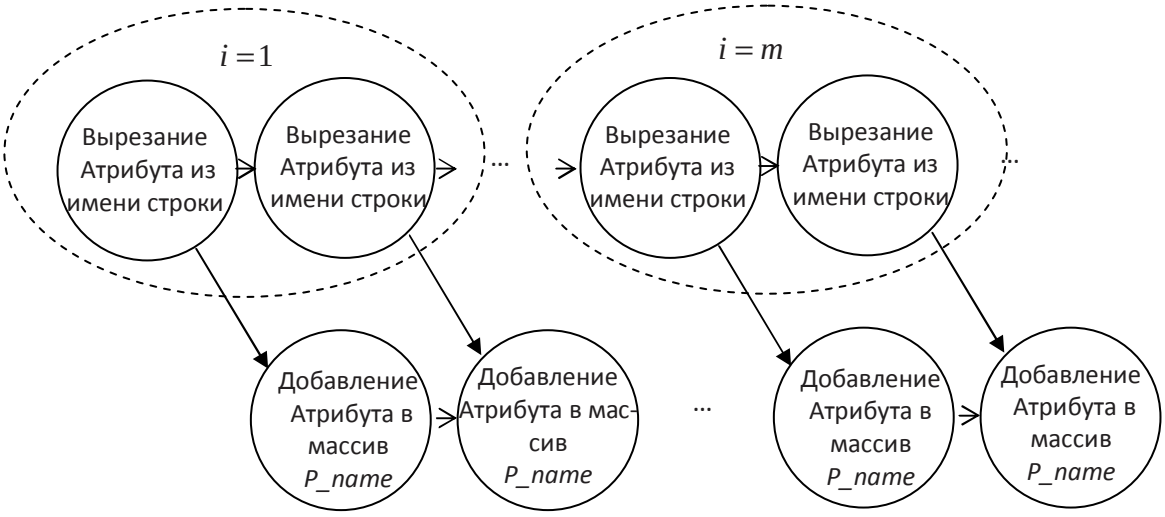

Рис. 4. Распараллеливание операции создания массива атрибутов строк по характеру операций.

Исходя из кода последовательной программы [] дальнейшая детализация данной операции не требуется.

Пересечение множеств атрибутов строк и столбцов текущей строки

Синтаксическая диаграмма операции проверки, можно ли атрибуты данной строки и соответствующих ей не нулевых столбцов выделить в отдельное отношение, представлена на рис. 5. Более наглядно характеризует эту операцию информационный граф (рис. 6.), на основании которого можно сделать вывод, что, перестроив структуру всей операции, разбив ее на два отдельных цикла, можно распараллелить алгоритм создания массива G и проверки пересечения множеств G и P_name по двум процессам (рис. 7.).

Оба процесса будут равны по загруженности. Трудоемкость всей операции равна $O(n)$, где n – число столбцов матрицы смежности. При распараллеливании операции создания массива атрибутов столбцов, соответствующих текущей строке, и проверке на его пересечение с массивом атрибутов строк трудоемкость полученного алгоритма вдвое меньше изначальной.

Примечание. Таким же образом распараллеливается последний блок «Сложение строк и удаление *i*-й строки» в синтаксической диаграмме «Стягивание матрицы».

Проверка, принадлежат ли единицы столбцов одной строке

Еще одной интересной операцией по части распараллеливания является операция проверки принадлежности единиц столбцов, соответствующих атрибутам массива I_name, одной строке. Это условие наряду с условием, что в каждом из этих столбцов должно быть всего по одной единице, является обязательным для стягивания матрицы и удаления транзитивных зависимостей. Синтаксическая диаграмма этой операции представлена на рис. 8а.

Так как первые четыре блока содержат вложенные друг в друга три цикла с пересчетом параметра, их можно распараллелить не одним способом. Наиболее экономным способом по числу передаваемых данных является распараллеливание по внешнему циклу (рис. 8б).

Последние три блока также представляют собой циклы с пересчетом параметра и содержат внутри цикла всего по одному действию. Их можно распараллелить по числу процессов $n \le v/2$, где v – верхняя граница цикла. Особое внимание следует обратить на то, что информационно три последних блока синтаксической диаграммы между собой не связаны.

Это видно также из информационного графа (рис. 8б).

Но они связаны логически. Если не выполняется условие равенства единице количества единиц хоть в одном из рассматриваемых столбцов, то выполнение последующих блоков излишне

Таким образом, наиболее логичной была бы их реализация на одном процессе, но это увеличит трудоемкость алгоритма в v^2 раз, где v – минимальная верхняя граница последних трех циклов диаграммы. Поэтому можно реализовать проверку всех трех условий параллельно, а результат собрать суммарно на одном процессе, что и представлено в информационном графе (рис. 8б).

Трудоемкость последовательного выполнения всей операции проверки принадлежности единиц столбцов, соответствующих атрибутам массива I_name, одной строке составляет $n^3 + 3n = O(n^3)$. Трудоемкость параллельного выполнения всей операции составляет $n^2 + n = O(n^2)$.

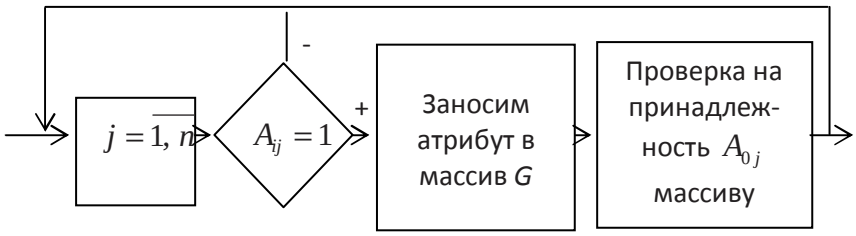

Рис. 5. Синтаксическая диаграмма операции проверки вхождения атрибутов строки в массив P_name.

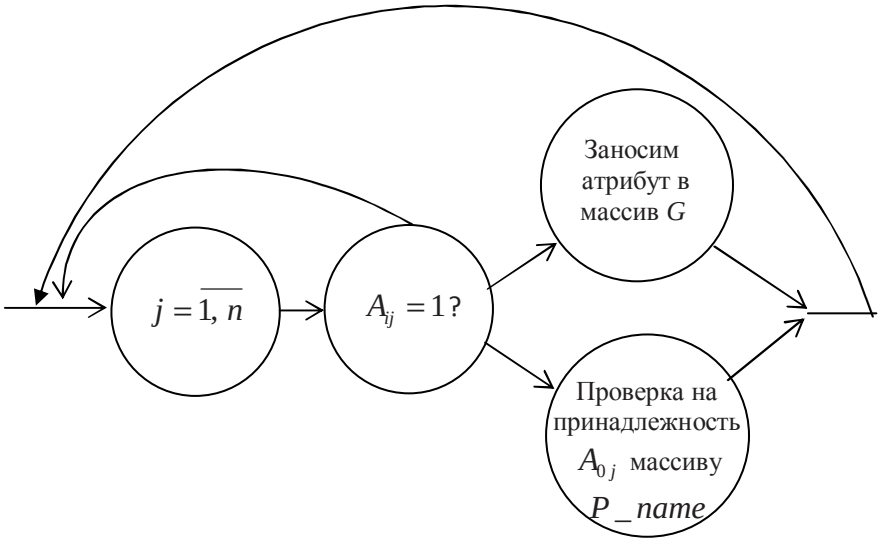

Рис. 6. Информационный граф операции создания проверки вхождения атрибутов строки в массив P_name.

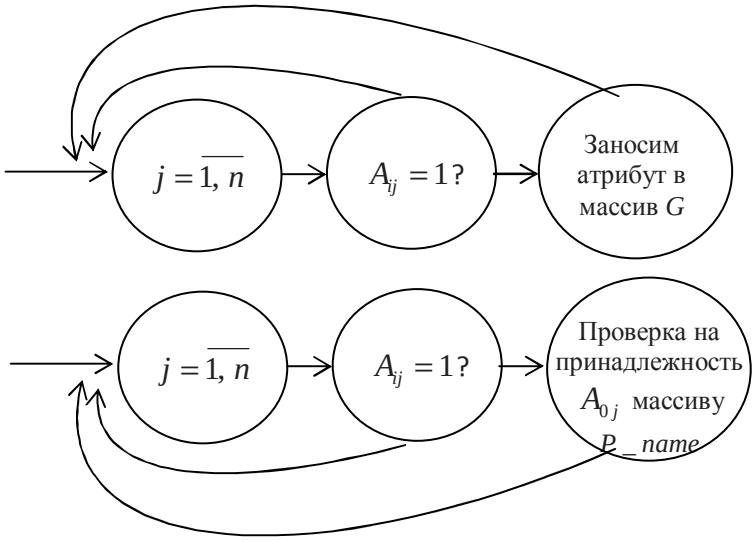

Рис. 7. Информационный граф операции создания проверки вхождения атрибутов строки в массив P_name.

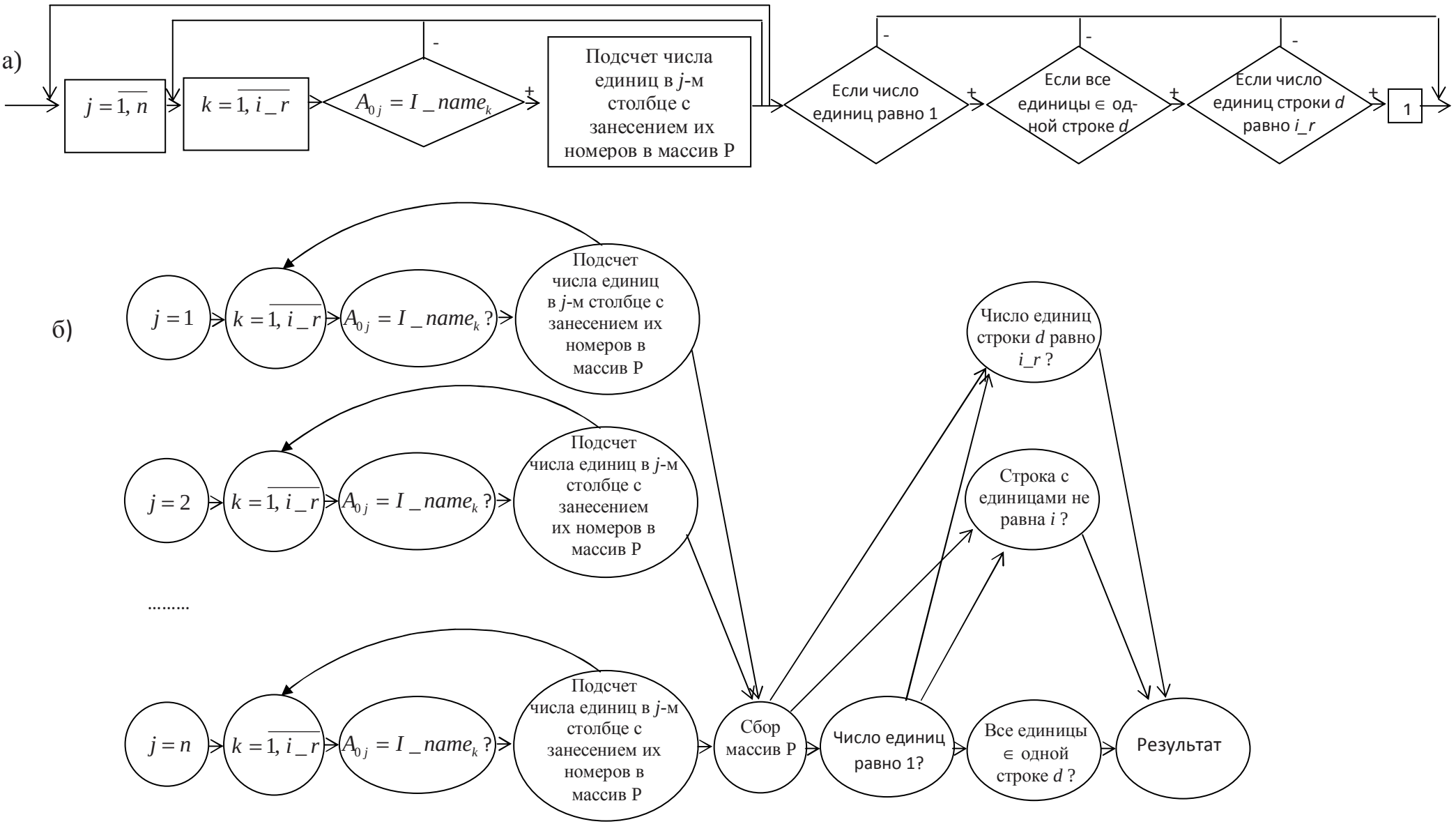

Рис. 8. Синтаксическая диаграмма (а) и информационный граф (б) операции принадлежности единиц столбцов, соответствующих атрибутам массива I_name, одной строке.

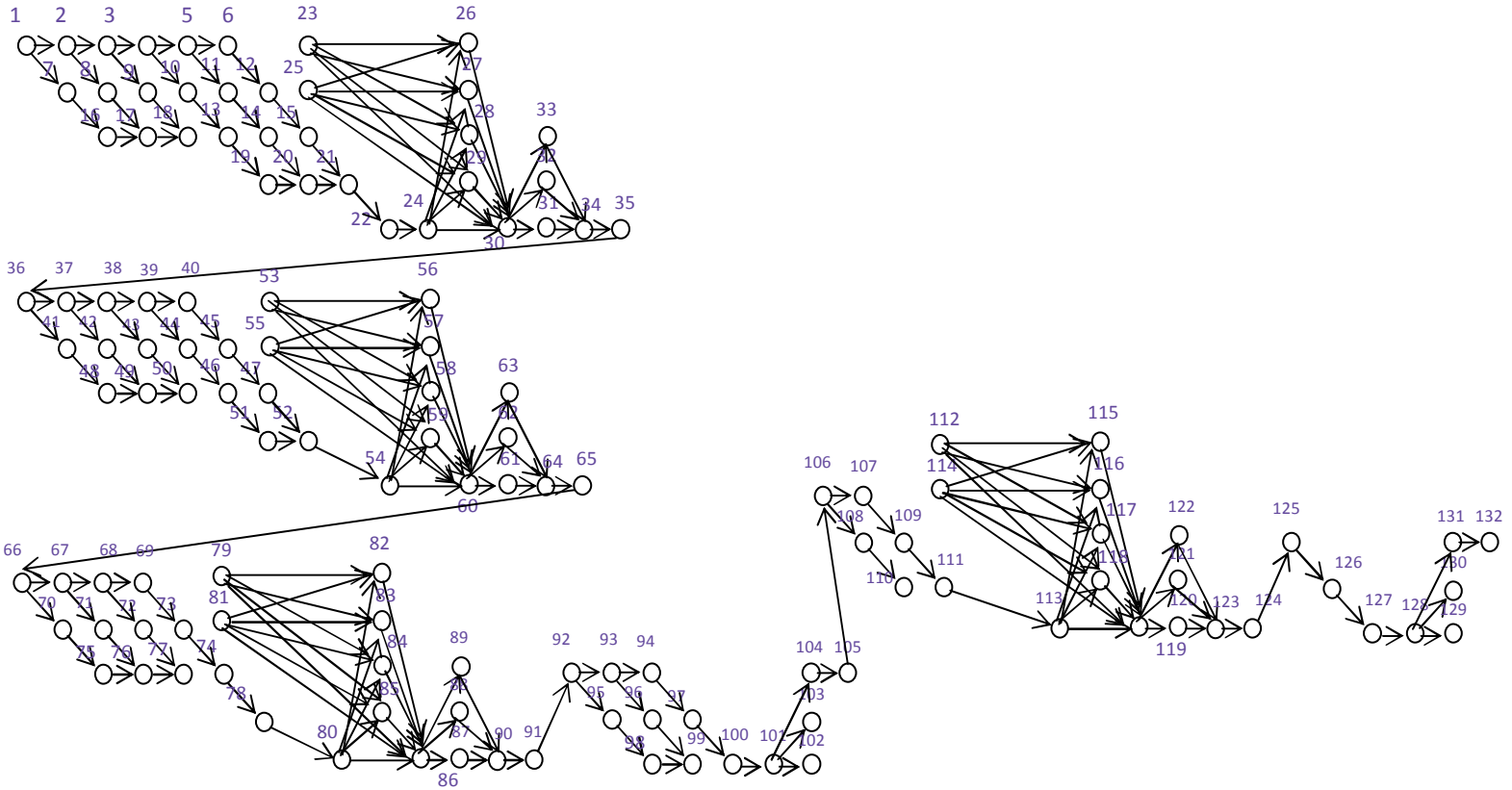

Рис. 9. Информационный граф алгоритма нормализации реляционных отношений.

Увеличив число передаваемых между процессами данных, можно уменьшить трудоемкость алгоритма до $O(n)$.

Загруженность процессов на протяжении всего времени выполнения параллельного алгоритма является достаточно равномерной.

Информационный граф параллельного алгоритма по ширине

Выше была построена синтаксическая диаграмма алгоритма нормализации реляционных отношений и рассмотрены все ее отдельные блоки. Фактически, весь алгоритм был разбит на сравнительно небольшие операции, и между ними установлены логические и информационные зависимости. Собрав все информационные подграфы блоков синтаксической диаграммы, получаем информационный граф алгоритма нормализации реляционных отношений при $m = 6, n = 11$ (рис. 9)

Заключение

С помощью построенного информационного графа можно осуществить:

– построение параллельного алгоритма нормализации реляционных отношений;

– оптимизацию параллельного алгоритма по ширине;

– оптимизацию параллельного алгоритма по высоте;

– подбор параллельного алгоритма в соответствии с характеристиками вычислительной системы и заданными условиями выполнения алгоритма.

Построение информационного графа алгоритма нормализации реляционных отношений и его дальнейшая оптимизация позволяют сделать следующие выводы:

1. В зависимости от информационного графа, все алгоритмы могут давать различные по качеству результаты. Следовательно, имея в наличии информационный граф и его матрицу смежности, желательно применить все алгоритмы оптимизации и на основе сравнительного анализа полученных результатов выбрать оптимальный способ решения задачи распараллеливания последовательного алгоритма.

2. Время, затрачиваемое на построение информационного графа, значительно превышает время проведения анализа способов его оптимизации и получения информационного графа по заданным параметрам вычислительной системы, на основе которого и будет записана параллельная программа.

3. Экономия времени при выполнении параллельной программы, по сравнению с ее последовательным аналогом, с учетом времени, затраченного на построение информационного графа и выбора способа распараллеливания, зависит от объема данных и структуры последовательного алгоритма.

Литература

1. Шичкина Ю.А. Автоматизация процесса нормализации реляционных баз данных. Наука. Технологии. Инновации// Материалы всероссийской научной конференции молодых ученых в 6-ти частях. Новосибирск: Изд-во НГТУ, 2004. Часть 1. - 233с.

2. Шичкина Ю.А., В.И.Воробьев Оптимизация параллельного алгоритма по числу процессов. Вестник гражданских инженеров.– 2008. - № 2(15). – С. 92-97

3. Шичкина Ю.А. Сокращение высоты информационного графа параллельного алгоритма. Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. -2009. – №3(80). – с.148-152.