Моделирование динамических баз данных

Э. Э. Гасанов, А. А. Плетнев
(МГУ им. М. В. Ломоносова, Москва)

В статье рассматривается математическая модель динамических баз данных, которая обрабатывает три типа запросов: поиск, вставка и удаление. Она построена на взаимодействие информационного графа и конечного детерминированного автомата. Модель позволяет решать динамические задачи поиска и оценивать сложность их решения. Кроме этого, модель позволяет строить бесконечно распараллеливаемые алгоритмы решения.

Ключевые слова: динамические базы данных, информационный граф, автомат, потоки запросов, параллельная обработка данных.

Информационный граф [1] — это управляющая система, предназначенная для моделирования алгоритмов поиска информации. Сам граф описывает структуру хранения данных, а нагрузка ребер и вершин графа функциями и данными позволяет описывать алгоритм поиска. Но информационный граф применим только для статических баз данных, к которым поступают только запросы на поиск и нет(или почти нет) запросов на вставку и удаление данных.

В данной работе будем рассматривать только задачу поиска идентичных объектов, и будем считать, что к базе данных могут поступать запросы трех типов: на поиск, вставку и удаление. Запрос на поиск вызывает поиск в базе данных элемента, совпадающего с запросом, запрос на вставку добавляет в базу данных новый элемент, если его там не было, и запрос на удаление удаляет из базы данных элемент, совпадающий с запросом, если он там находился.

Для того, чтобы обрабатывать запросы на вставку и удаление данных надо иметь механизм для изменения структуры данных, то есть для изменения информационного графа. Предлагается в качестве такого механизма использовать конечный автомат. Как это можно было бы

делать? Можно было бы предположить, что автомат «ходит» по информационному графу и в каждый момент обозревает некоторую окрестность текущей вершины, и в зависимости от этой окрестности и своего внутреннего состояния как-то изменяет эту окрестность и переходит в новую вершину графа. Трудности данного подхода состоят в том, что во-первых, граф является потенциально бесконечным объектом, в частности, может иметь вершину со сколь угодно большой степенью инцидентности, а значит, сколь угодно большую окрестность, и во-вторых алфавит разметок является существенно бесконечным, поскольку множество данных может быть сколь угодно большим и каждому элементу данных нужна своя пометка.

Чтобы решить первую проблему, рассматривается класс графов, степень инцидентности каждой вершины которых, не больше некоторого параметра N. Это позволяет при фиксации радиуса окрестности ограничить ее размер.

Алфавит разметки содержит символы для обозначения данных (без ограничения общности можно считать, что это множество натуральных чисел) и символы функций. Каждая функция определена на множество запросов (то есть на натуральном ряде) и принимает конечное число значений. Также каждая функция имеет параметр (как правило из того же множества запросов) и тип, принимающий конечное число значений. Тип обозначает операцию, выполняемую функцией. Например, типом может быть операция сравнения на равенство, а параметром натуральное число, с которым запрос сравнивается. Другим примером типа является сравнение на неравенство запроса и параметра. Отсюда видно, что мы не можем ограничить алфавит разметки, поскольку данных может быть сколь угодно много, то есть база данных может быть сколь угодно большой. Поэтому, чтобы решить вторую проблему, считаем, что автомат «видит» не саму разметку графа, а некоторую ее проекцию. А именно, для вершин автомат видит тип вершины (корень, лист, внутренняя вершина), а вместо функций видит их тип и значение на запросе. Заметим, что автомат не видит значение запроса, а знает только его тип (поиск, вставка или удаление).

В этих предположениях входной алфавит автомата составляется из конечного множества подграфов, которые могут образовывать конечную окрестность текущей вершины, и конечной вариации проекций разметок этих подграфов.

Считаем, что имеется конечное множество преобразований окрестности. Под преобразованием понимается некоторое правило, как из исход-

ной окрестности и ее исходной разметке и по значению запроса получить новую размеченную окрестность. Самым простым преобразованием является тождественное. Тем самым, считаем, что преобразование в отличии от автомата видит и значение запроса, и саму разметку, а не ее проекцию.

Выходной символ автомата содержит информацию о следующей вершине, в которую переходит автомат, и о номере преобразования, производимого над текущей окрестностью, а так же информацию об окончании работы.

Автомат функционирует в дискретном времени. В начальный такт он находится в корне информационного графа. Далее каждый такт автомат анализирует окрестность текущей вершины и в зависимости от своего состояния сообщает о номере преобразования и новой позиции текущей вершины. После этого производится указанное преобразование окрестности и автомат помещается в новую текущую вершину, если он не сообщил об окончании работы. На этом такт заканчивается.

Информационный граф вместе с описанным автоматом будем называть *динамическим информационным графом* (ДИГ) [2, 3]. Как и в статическом случае информационный граф описывает структуру данных, тогда как автомат предназначен для осуществления поиска данных и для обеспечения актуальности базы данных в соответствии с запросами на вставку и удаление данных.

Если запросы на вставку и удаление поступают достаточно редко, то ДИГ может справляться с этими запросами. Теперь рассмотрим случай, когда каждый такт может поступить один запрос на поиск, вставку или удаление, причем запросы поступают в произвольном порядке. Понятно, что один автомат не сможет справиться с таким потоком запросов (иначе он должен был бы осуществлять поиск и актуализировать базу данных за один такт). Поэтому будем считать, что у нас имеется сколь угодно большое количество экземпляров одного и того же автомата, и каждый запрос обрабатывается своим автоматом. Такой ДИГ с неограниченным числом экземпляров автомата назовем *потоковым динамическим информационным графом* (ПДИГ). ПДИГ в отличии от ДИГ моделирует параллельные алгоритмы, поскольку большое количество вычислителей работают на одной структуре данных. Как известно, если несколько вычислителей работают с одним массивом данных, то при их работе могут возникнуть конфликты доступа к данным. В данной работе считается, что разные вычислители могут читать из одних и тех же ячеек данных, но не могут одновременно писать в одни и те же ячейки данных. За-

дача синтеза ПДИГ, обслуживающего произвольный поток запросов, — это фактически задача построения бесконечно распараллеливаемого алгоритма обработки данных, не имеющего конфликтов при модификации структур данных.

ПДИГ называется типа (N, R), если любая его вершина имеет степень инцидентности не более N, а радиус обзора автомата равен R. ПДИГ называется *конечным*, если каждый из автоматов существует на графе конечное время. ПДИГ называется *селекторным*, если при формировании новой разметки информационного графа не используются никакие арифметические операции, кроме операции выбора. Под сложностью ПДИГ будем понимать время, необходимое на поиск данных в худшем случае.

В [4] показано, что существует конечный селекторный ПДИГ типа (2,2), обслуживающий произвольный поток запросов с линейной сложностью поиска. При этом при значениях параметров (8,4) можно построить ПДИГ, обслуживающий произвольный поток запросов с логарифмической сложностью поиска [5]. Если отказаться от свойства конечности, то можно построить селекторный ПДИГ типа (1,1) для произвольного потока запросов [6]. Кроме того в [6] показано, что существует конечный неселекторный ПДИГ типа (1,1), обслуживающий произвольный поток запросов, который отвечает на вопрос, имеется ли запрос в базе данных (такую задачу будем называть логической). Также существует конечный неселекторный ПДИГ типа (2,1), обслуживающий произвольный поток запросов, который находит запрос в базе данных [7].

Теперь приведем нижние оценки параметров ПДИГ. Показано, что ни для какого R не существует конечного ПДИГ типа $(1,R)$, обслуживающего произвольный поток запросов. Также показано, что если использовать только функции типа сравнения на равенство, то для любой

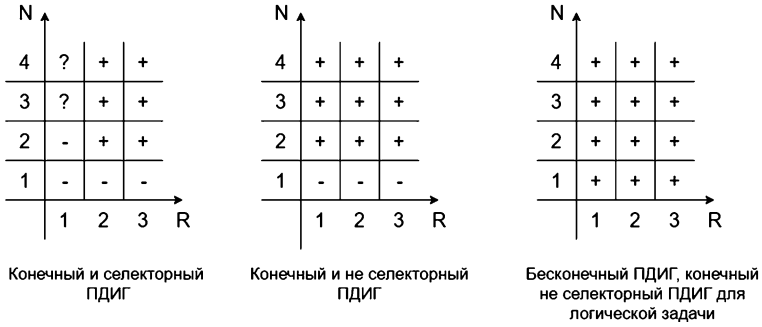

Рис. 1. Иллюстрация теорем в виде таблиц.

функции сложности L не существует ПДИГ типа (2,1), который обслуживает произвольный поток запросов со временем поиска, не превосходящим L [6].

На рис. **??** описанные выше результаты приведены в виде таблиц, где значок «+» означает, что ПДИГ с данными параметрами может обслуживать произвольный поток запросов, «−» означает, что не может, а значок «?» — что ответ не известен.

Список литературы

[1] Гасанов Э. Э., Кудрявцев В. Б. Интеллектуальные системы. Теория хранения и поиска информации. — М.: Юрайт, 2016.

[2] Плетнев А. А. Моделирование динамических баз данных // Интеллектуальные системы. — 2014. — Т. 17, вып. 1–4. — С. 75–79.

[3] Плетнев А. А. Информационно-графовая модель динамических баз данных и ее применение // Интеллектуальные системы. — 2014. — Т. 18, вып. 1. — С. 111–140.

[4] Плетнев А. А. Динамическая база данных, допускающая параллельную обработку произвольных потоков запросов // Интеллектуальные системы. Теория и приложения. — 2015. — Т. 19, вып. 1. — С. 117–142.

[5] Плетнев А. А. Логарифмическая по сложности параллельная обработка автоматами произвольных потоков запросов в динамической базе данных // Интеллектуальные системы. Теория и приложения. — 2015. — Т. 19, вып. 1. — С. 171–212.

[6] Плетнев А. А. Нижняя оценка на область видимости автомата, обрабатывающего произвольный поток запросов к динамической базе данных // Интеллектуальные системы. Теория и приложения. — 2015. — Т. 19, вып. 4. — С. 117–151.

[7] Плетнев А. А. Минимально возможный по степени ветвления информационный граф с радиусом видимости один, обрабатывающий произвольный поток запросов к динамической базе данных // Интеллектуальные системы. Теория и приложения. — 2016. — Т. 20, вып. 1. — С. 223–254.