УДК 004.622:004.654

ВЫДЕЛЕНИЕ И ПОИСК ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ МЕЖДУ АТРИБУТАМИ В СИСТЕМАХ BIG DATA

А.А. Карпук
Профессор кафедры программного обеспечения сетей телекоммуникаций Белорусской государственной академии связи, кандидат технических наук, доцент
a_karpuk@mail.ru

Л.С. Лазута
Аспирант кафедры телекоммуникационных систем Белорусской государственной академии связи, магистр
lenya.lazuta@mail.ru

А.А. Карпук
Окончил Белорусский государственный университет, автор более 240 научных трудов, включая 2 единолично написанные монографии. Область научных интересов связана с моделированием и оптимизацией сложных систем, проектированием баз данных и хранилищ данных, оценкой качества радиосвязи и оптимизацией присвоения радиочастот в радиосетях и радиолиниях.

Л.С. Лазута
Окончил Белорусскую государственную академию связи. Область научных интересов связана с моделированием и оптимизацией сложных систем, проектированием баз данных и хранилищ данных.

Аннотация. Приведен обзор методов выделения и алгоритмов поиска функциональных зависимостей между атрибутами в системах Big Data. Рассмотрены основные алгоритмы поиска функциональных зависимостей на основе анализа решетки атрибутов, на основе анализа строк таблицы и гибридные алгоритмы. Определены области применения алгоритмов каждого типа и направления дальнейших исследований по совершенствованию и разработке алгоритмов поиска функциональных зависимостей между атрибутами в системах Big Data.

Ключевые слова: большие данные, функциональная зависимость, выделение функциональных зависимостей, алгоритмы поиска функциональных зависимостей.

Введение. Современные системы Big Data могут иметь архитектуру реляционного хранилища данных (Relational Data Warehouse, RDW), озера данных (Lake Data), современного хранилища данных (Modern Data Warehouse, MDW), фабрики данных (Data Fabric), озерного хранилища данных (Data Lakehouse), сетки данных (Data Mesh) [1]. Во всех перечисленных архитектурах, кроме архитектуры Lake Data, на одном или нескольких этапах обработки данных решаются задачи очистки данных, предварительной обработки данных и нормализации данных в виде приведения данных к третьей нормальной форме. Для решения задачи нормализации данных требуется знание системы образующих структуры функциональных зависимостей (ФЗ) между данными. В настоящей работе рассматриваются методы выделения и алгоритмы поиска ФЗ между данными в системах Big Data.

Методы выделения функциональных зависимостей между данными. Пусть $X \subset A$ – подмножество атрибутов (признаков) системы Big Data, $Z \in A$ – некоторый атрибут. Говорят, что существует ФЗ $X \to Z$, если любой комбинации значений атрибутов из X всегда соответствует единственное значение атрибута Z. Очевидно, что из $Z \in X$ следует, что $X \to Z$. Такая ФЗ, в которой зависимый атрибут входит в состав левой части ФЗ, называется тривиальной. В дальнейшем будем рассматривать только нетривиальные ФЗ между атрибутами. Структура ФЗ на множестве атрибутов удовлетворяет аксиомам У. Армстронга [2]. В работах по проектированию баз данных 80-х годов прошлого века структура ФЗ на множестве атрибутов предметной области задавалась путем постулирования некоторого множества ФЗ, которое называется системой образующих структуры ФЗ. П. Чен предложил описывать предметную область при проектировании баз данных в виде диаграммы «сущность-связь» (ER-диаграммы) [3]. Из ER-диаграммы можно выделить и включить в систему образующих структуры ФЗ зависимости всех атрибутов сущностей от первичных и потенциальных (альтернативных) ключей сущностей. Других нетривиальных ФЗ между атрибутами из ER-диаграммы получить нельзя. А.А. Карпук и В.М. Острейко предложили метод построения расширенной ER-диаграммы, отличающейся от диаграммы П. Чена тем, что сущности предметной области могут иметь сложную иерархическую структуру, и в диаграмме предусмотрено постулирование ФЗ и нефункциональных связей между атрибутами, существующих в отрыве от контекста [4]. Из расширенной ER-диаграммы можно дополнительно выделить и включить в систему образующих структуры ФЗ зависимости не ключевых атрибутов сложных объектов от сцепленных ключей и ФЗ между атрибутами, существующие в отрыве от контекста.

Исследования по поиску ФЗ между атрибутами возобновились в начале 21-го века в связи с появлением онтологического подхода к описанию предметной области и развитием систем Big Data. А.А. Карпук разработал метод построения системы образующих структуры ФЗ между атрибутами предметной области на основе онтологии предметной области, описанной на языке OWL-2 [5]. Этот метод выделяет ФЗ между атрибутами, входящими в один класс, входящими в класс и его подкласс, входящими в функциональные отношения между классами, входящими в определение функций, заданных на атрибутах и классах онтологии предметной области.

Алгоритмы поиска функциональных зависимостей между данными. Ряд авторов предложили алгоритмы поиска ФЗ между атрибутами в таблицах систем Big Data, В общем случае эта задача является NP-трудной, поскольку для таблицы, содержащей m атрибутов и n строк, вычислительная сложность задачи оценивается величиной $O(2^{m-2}n^2m^2)$ [6]. Существующие алгоритмы поиска ФЗ между атрибутами в таблицах можно разделить на 3 класса: алгоритмы на основе анализа решетки атрибутов, алгоритмы на основе анализа строк таблицы и гибридные алгоритмы.

Решеткой атрибутов для таблицы, содержащей m атрибутов, называется ориентированный граф, состоящий из $m+1$ уровней. На уровне 0 расположена одна вершина графа, представляющая пустое множество атрибутов. На уровне 1 расположено m вершин, каждая из которых представляет 1 атрибут таблицы. На уровне 2 расположено $m(m-1)/2$ вершин, каждая из которых представляет сочетание двух атрибутов таблицы. В общем виде, на уровне $1 \le k \le m$ расположено $m!/k!(m-k)!$ вершин, каждая из которых представляет сочетание из k атрибутов таблицы. На уровне m расположена одна вершина, представляющая все множество атрибутов таблицы. В решетке атрибутов существует направленное ребро между вершинами соседних уровней от меньшего уровня к большему уровню, если все атрибуты вершины из меньшего уровня входят в состав атрибутов

вершины из большего уровня. Каждому ребру решетки атрибутов соответствует возможная ФЗ, в левой части которой содержатся атрибуты вершины меньшего уровня, а в правой части - атрибут, полученный удалением из состава атрибутов вершины большего уровня всех атрибутов вершины меньшего уровня. В левой части рисунка 1 показана решетка атрибутов для таблицы из четырех атрибутов A, B, C, D. Вершинам A из уровня 1 и AB из уровня 2 соответствует возможная ФЗ A→B, вершинам CD из уровня 2 и BCD из уровня 3 соответствует возможная ФЗ CD→B.

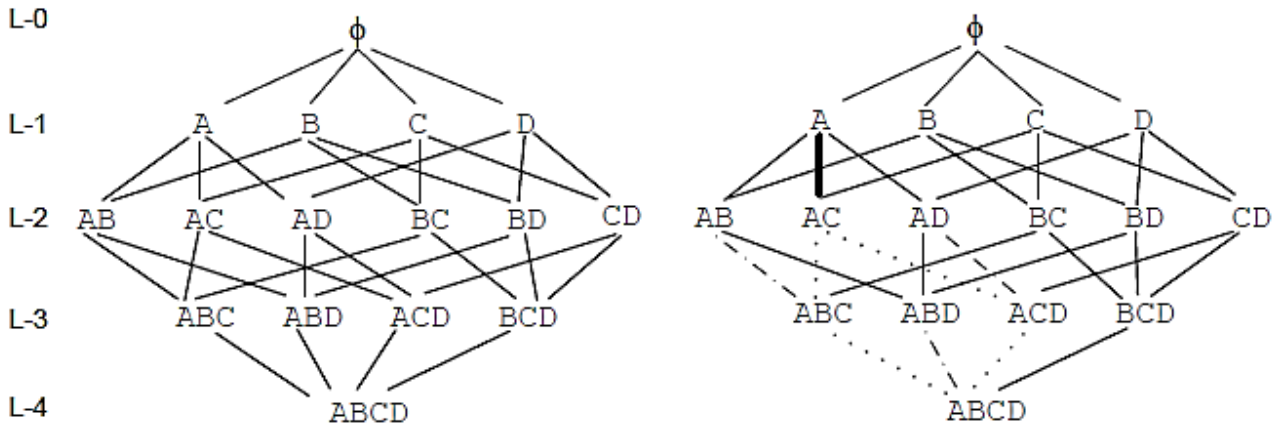

Рисунок 1. Решетка атрибутов для таблицы из четырех атрибутов

Идея алгоритмов поиска ФЗ между атрибутами в таблицах на основе анализа решетки атрибутов состоит в выделении возможной ФЗ, проверки возможной ФЗ и классификации ее на есть-ФЗ или нет-ФЗ, и сокращении области поиска на основе полученной классификации путем удаления некоторых возможных ФЗ или удаления некоторых вершин решетки. Если X и Y - 2 непересекающихся подмножества атрибутов, A и B - 2 атрибута, не принадлежащие X и Y, то в силу аксиом У. Армстронга справедливы следующие правила:

1) Если возможная ФЗ X→A классифицирована как есть-ФЗ, то все ФЗ XY→A тоже будут классифицированы как есть-ФЗ, и проверять их нет необходимости, а соответствующие ребра следует удалить из решетки атрибутов.

2) Если возможная ФЗ X→A классифицирована как есть-ФЗ, то вместо проверки возможной ФЗ XAY→B следует проверять возможную ФЗ XY→B, а соответствующие ребра следует удалить из решетки атрибутов.

3) Если X - ключ рассматриваемой таблицы, т. е. для любого атрибута Z, не принадлежащего X, возможная ФЗ X→A классифицируется как есть-ФЗ, то из решетки атрибутов следует удалить все вершины, в состав которых входят все атрибуты из X.

К примеру, пусть в решетке атрибутов, показанной на рисунке 1, возможная ФЗ A→C классифицирована как есть-ФЗ (соответствующее ребро выделено в правой части рисунка 1). Тогда то по правилу 1 из решетки надо удалить ребра между вершинами AB и ABC, AD и ACD, ABD и ABCD (это ФЗ AB→C, AD→C, ABD→C соответственно). По правилу 2 из решетки надо удалить ребра между вершинами AC и ABC, AC и ACD, ABC и ABCD, ACD и ABCD (это ФЗ AC→B, AC→D, ABC→D, ACD→B соответственно).

С использованием приведенных правил сокращения области поиска в решетке атрибутов работает алгоритм поиска ФЗ между атрибутами в таблицах систем Big Data, названный алгоритмом TANE [7]. В этом алгоритме обход решетки совмещен с ее

построением, начиная с уровня 1, сначала вправо, а затем вниз. За счет отсечения ребер решетки по правила 1 и 2, алгоритм находит минимальные ФЗ, не содержащие лишних атрибутов в левой части. Для классификации возможных ФЗ алгоритм TANE разбивает множество строк таблицы по классы эквивалентности с совпадающими значениями атрибутов левой части рассматриваемых возможных ФЗ, а затем проверяет значения атрибутов правых частей возможных ФЗ. Алгоритм FUN, предложенный в [8], отличается от алгоритма TANE тем, что для отсечения ребер решетки использует более жесткое правило. чем правило 2, сокращая тем самым объем вычислений.

В работе [9] предложен алгоритм поиска ФЗ между атрибутами в таблицах систем Big Data на основе анализа решетки атрибутов, названный алгоритмом FD_Mine. Этот алгоритм также обходит решетку атрибутов, начиная с уровня 1, сначала вправо, а затем вниз. Для классификации возможных ФЗ алгоритм FD_Mine использует такой же метод, как алгоритм TANE. В отличие от алгоритмов TANE и FUN, алгоритм FD_Mine использует дополнительное правило отсечения вершин и ребер решетки, основанное на классах эквивалентности подмножеств атрибутов. Два набора атрибутов считаются эквивалентными, если они функционально зависят друг от друга. После проверки каждого уровня в решетке атрибутов алгоритм FD_Mine сканирует этот уровень и обнаруженные ФЗ на предмет эквивалентности. Проверка эквивалентности производится путем построения замыкания каждого из подмножеств атрибутов относительно найденных ФЗ. Если эквивалентные подмножества атрибутов найдены, то алгоритм оставляет в решетке атрибуты одного из эквивалентных подмножеств, удаляя из решетки вершины, содержащие остальные эквивалентные подмножества, и соответствующие им ребра.

В работе [10] предложен алгоритм поиска ФЗ между атрибутами в таблицах систем Big Data на основе анализа решетки атрибутов, названный алгоритмом DFD. На первом шаге алгоритма проверяется каждый атрибут, не является ли он ключом таблицы. Если атрибут является ключом, то фиксируются ФЗ, левая часть которых состоит из этого атрибута, а в правых частях находятся все остальные атрибуты таблицы, и этот атрибут исключается из решетки. Каждый оставшийся атрибут рассматривается как возможная правая часть некоторых ФЗ. Для каждой возможной правой части ФЗ алгоритм формирует множество источников для левой части ФЗ и обходит источники, выбирая очередной источник по определенным правилам, которые в некоторых ситуациях предусматривают случайный выбор. Каждый источник (вершину решетки) и соответствующее ребро алгоритм классифицирует как зависимость, минимальную зависимость, кандидат на минимальную зависимость, не зависимость, максимальную не зависимость или кандидат на максимальную не зависимость.

Зависимость и не зависимость представляют вершины левой части, которые не являются ни минимальными, ни максимальными соответственно. Кандидат на минимальную зависимость или/максимальную не зависимость – это комбинация атрибутов, которая все еще может быть левой частью минимальной ФЗ или максимальной не-ФЗ. В случае, если текущая комбинация атрибутов уже была пройдена на более раннем этапе процесса, алгоритм DFD снова проверяет только те комбинации атрибутов, которые классифицированы как кандидаты на минимальную зависимость или максимальную не зависимость. При этом алгоритм проверяет, можно ли изменить классификацию кандидата или нет. Может случиться так, что после повторного посещения кандидата на минимальную зависимость все подмножества этой вершины были классифицированы как не зависимость, что делает кандидата минимальной зависимостью. Вот почему алгоритм DFD сохраняет посещенные вершины кандидаты в стеке трассировки. Трассировка позволяет алгоритму

откатываться по решетке, повторно посещая вершины, которые в конечном итоге могут быть классифицированы на более позднем этапе. В случае, если вершина еще не была посещена, алгоритм DFD проверяет, является ли вершина надмножеством или подмножеством ранее обнаруженной зависимости или не зависимости, и обновляет ее классификацию соответствующим образом. В противном случае алгоритм определяет, какой тип кандидата представляет комбинация вершин. Если обнаруживается ФЗ, то вершина классифицируется как кандидат на минимальную зависимость, в противном случае – как кандидат на максимальную не зависимость.

К алгоритмам поиска ФЗ между атрибутами в таблицах на основе анализа строк таблицы относятся алгоритмы FDEP [11], Dep-Miner [12], FastFDs [13] и FSC [14]. В работе [11] П.А. Флах и И. Савник предложили 3 варианта алгоритма FDEP: алгоритм анализа сверху вниз, алгоритм двунаправленного анализа и алгоритм анализа снизу вверх. При анализе сверху вниз алгоритм сначала для каждого атрибута A определяет минимальные подмножества атрибутов X, для которых ФЗ X→A выполняется в первых двух строках таблицы. Затем выполнение каждой найденной ФЗ X→A проверяется для остальных строк таблицы. Если для очередной строки таблицы ФЗ X→A не выполняется, то эта ФЗ удаляется из рассматриваемых возможных ФЗ. В результате для каждого атрибута будет получено множество минимальных ФЗ, которые выполняются для всех строк таблицы.

При двунаправленном анализе сначала выполняется анализ всех пар строк таблицы. Для каждой пары строк для каждого атрибута A алгоритм определяет максимальные подмножества атрибутов Y, для которых ФЗ Y→A не выполняется, т. е. находит максимальные нет-ФЗ. На основе найденных максимальных нет-ФЗ строится неизбыточное отрицательное покрытие ФЗ. Второй шаг алгоритма двунаправленного анализа похож на алгоритм анализа сверху вниз, и отличается от него тем, что выполнение каждой найденной по первым двум строкам ФЗ X→A проверяется не по остальным строкам таблицы, а по построенному не избыточному отрицательному покрытию ФЗ. На первом шаге алгоритма анализа снизу вверх также находятся максимальных нет-ФЗ и строится неизбыточное отрицательное покрытие ФЗ. На втором шаге алгоритма за одну итерацию на основе не избыточного отрицательного покрытия ФЗ строится положительное покрытие ФЗ, состоящее из минимальных ФЗ. Очевидно, что при достаточно большом количестве строк в таблице быстродействие алгоритма анализа снизу вверх будет выше, чем у алгоритма двунаправленного анализа.

В работе [12] С. Лопес и др. предложили алгоритм Dep-Miner, который выводит все минимальные ФЗ из наборов атрибутов, которые имеют одинаковые значения в определенных строках таблицы. Эти наборы атрибутов называются согласованными наборами, а наборы из оставшихся атрибутов называются дополнительными наборами. Алгоритм Dep-Miner состоит из пяти этапов. На этапе 1 алгоритм вычисляет урезанное разбиение таблицы для каждого атрибута, в котором каждый класс эквивалентности атрибута содержит более одной строки таблицы. Из урезанного разбиения таблицы строится множество максимальных классов эквивалентности для таблицы, которое состоит из тех классов эквивалентности для всех атрибутов, которые не являются подмножествами других классов эквивалентности. На этапе 2 алгоритма множество максимальных классов эквивалентности для таблицы используется для построения согласованных наборов атрибутов таблицы. Согласованный набор атрибутов таблицы состоит из одного или более атрибутов, которые имеют одинаковые значения в двух или более строках таблицы. На этапе 3 согласованные наборы атрибутов таблицы преобразуются в максимальные наборы, т. е. наборы атрибутов, которые не имеют

надмножества с одинаковыми значениями в двух строках таблицы. На этапе 4 алгоритм инвертирует максимальные согласованные наборы атрибутов в дополнительные наборы. На этапе 5 алгоритм вычисляет из дополнительных наборов атрибутов все минимальные ФЗ между атрибутами.

В работе [13] К. Вайс с соавторами предложили алгоритм FastFDs как усовершенствование алгоритма Dep-Miner. Алгоритм FastFDs также строится на использовании согласованных наборов атрибутов для вывода ФЗ между атрибутами. После вычисления согласованных наборов атрибутов алгоритм FastFDs следует другой стратегии для вывода минимальных ФЗ, поскольку максимизация согласованных наборов атрибутов на этапе 3 алгоритма Dep-Miner является дорогостоящей операцией, Алгоритм FastFDs вместо этого вычисляет все дополнительные наборы атрибутов непосредственно по согласованным наборам. На шаге 4 алгоритм для каждого атрибута вычисляет все минимальные дополнительные наборы атрибутов, на основе которых на шаге 5 алгоритм находит все минимальные ФЗ между атрибутами.

Одним из последних алгоритмов поиска ФЗ между атрибутами в таблицах на основе анализа строк таблицы является алгоритм FSC, предложенный С. Вань с соавторами [14] в 2024 г. Двухэтапное выполнение алгоритма FSC опирается на предварительно вычисленную вспомогательную структуру сопоставимых пар строк, которые отражают пары идентификаторов для строк, имеющих, по крайней мере, по одному атрибуту с одинаковыми значениями. На шаге 1 алгоритм FSC определяет нарушенные ФЗ и вводит выборочное сравнение с помощью сопоставимых пар строк, чтобы значительно сократить требуемое попарное сравнение. Для обработки атрибутов малой мощности разработана стратегия сжатия прямого значения и комбинации значений. На шаге 2 алгоритм FSC индуцирует требуемые ФЗ по результатам шага 1. Обширные экспериментальные результаты, полученные на синтетических и реальных наборах данных, показывают, что алгоритм FSC может эффективно обнаруживать ФЗ в таблицах систем Big Data.

Алгоритмы поиска ФЗ между атрибутами в таблицах систем Big Data на основе анализа решетки атрибутов критичны к росту количества атрибутов в таблицах и хорошо работают при небольшом количестве атрибутов (не более 20) для большого количества строк (до сотен тысяч). Алгоритмы поиска ФЗ между атрибутами в таблицах на основе анализа строк таблицы критичны к росту количества строк и хорошо работают при относительно небольшом количестве строк (не более 1000) для большого количества атрибутов (до нескольких сотен). В гибридных алгоритмах поиска ФЗ между атрибутами в таблицах систем Big Data предприняты попытки устранить эти недостатки.

Т. Папенброк и Ф. Науманн разработали гибридный алгоритм HyFD [15], который объединяет методы поиска ФЗ, эффективные для большого количества атрибутов и большого количества строк. На первом этапе алгоритм HyFD извлекает небольшое подмножество строк из входных данных и вычисляет ФЗ для этой неслучайной выборки. Поскольку на этом этапе используется только подмножество строк, он работает особенно эффективно по атрибутам. Результатом является набор ФЗ, которые либо действительны, либо почти действительны по отношению к полному входному набору данных. На втором этапе алгоритм HyFD проверяет обнаруженные ФЗ по всему набору строк и находит такие ФЗ, которые не выполняются. Этот этап эффективен по строкам, поскольку он использует ранее обнаруженные ФЗ для эффективного сокращения пространства поиска. Если проверка становится неэффективной, алгоритм может вернуться к первому этапу и продолжить работу со всеми результатами, обнаруженными на данный момент. Эта

чередующаяся двухфазная стратегия обнаружения явно превосходит все существующие алгоритмы с точки зрения времени выполнения и масштабируемости, при этом обеспечивает обнаружение всех минимальных ФЗ.

В работе [16] Ц. Вэй и С. Линк отметили, что в алгоритме HyFD переключение стратегий происходит, когда текущая стратегия работает некорректно, однако это переключение происходит без доказательств того, что другая стратегия будет работать хорошо. Более того, ни в одном из предыдущих алгоритмов нет механизма для балансировки эффективности времени выполнения и использования основной памяти. Авторы [16] предложили новый гибридный алгоритм DHyFD, в котором переход от подхода на основе атрибутов к подходу на основе строк происходит всякий раз, когда есть вероятность проверки многих ФЗ. Эта вероятность оценивается с помощью предложенной меры. Установка более низких пороговых значений для этой меры дает алгоритму возможность использовать больше ресурсов основной памяти всякий раз, когда это может повысить эффективность выполнения. Еще одним недостатком известных алгоритмов поиска ФЗ между атрибутами в таблицах систем Big Data Ц. Вэй и С. Линк считают слишком большое количество получаемых ФЗ и большое суммарное количество атрибутов в их левых частях. Действительно, ни в одном из известных алгоритмов не ставится задача минимизации количества получаемых ФЗ и количества атрибутов в них. В алгоритме DHyFD для уменьшения количества полученных ФЗ используется метод построения канонического покрытия структуры ФЗ, который фактически строит элементарный базис структуры ФЗ по алгоритму К. Делобеля и Р. Кейси [17]. В работе [18] А.А. Карпук показал, что в общем случае этот элементарный базис структуры ФЗ не является минимальным по количеству ФЗ и не является оптимальным по суммарному количеству атрибутов в ФЗ, и предложил методы построения минимального и оптимального элементарного базиса структуры ФЗ.

В заключение нашего обзора основных алгоритмов поиска ФЗ между атрибутами в таблицах систем Big Data отметим последнюю работу Т. Блейфуса и Т. Папенброка с соавторами [19], в которой они предложили новый гибридный алгоритм FDhits. В этом алгоритме интегрировано несколько методов оптимизации для эффективного обнаружения ФЗ, а именно гибридная стратегия, поиск уникальных комбинаций атрибутов через перечисление наборов совпадений (Hiting Set Enumeration), однопроходная проверка всех правых частей кандидатов на ФЗ и распараллеливание вычислений. Авторы [19] утверждают, что алгоритм FDhits может за приемлемое время найти все минимальные ФЗ в таблице, содержащей до 100 атрибутов и более 5 миллионов строк, для которой все остальные алгоритмы вообще не могут найти решение. На наш взгляд, и этот алгоритм можно улучшить, применив для поиска уникальных комбинаций атрибутов результаты работы А.А. Карпука и В.В. Краснопрошина [20].

Заключение. По результатам проведенного анализа основных методов выделения и алгоритмов поиска ФЗ между атрибутами в таблицах систем Big Data можно сделать вывод, что исследования в этом направлении далеки от завершения, и их следует продолжить. Очевидно, что нет смысла улучшать существующие алгоритмы поиска ФЗ между атрибутами в таблицах на основе анализа решетки атрибутов и на основе анализа строк таблицы, поскольку область их применения существенно ограничена. Следует сосредоточить усилия на совершенствовании существующих и разработке новых гибридных алгоритмов поиска ФЗ между атрибутами в таблицах систем Big Data. В первую очередь требуется реализовать в этих алгоритмах методы минимизации количества получаемых ФЗ и количества атрибутов в них, а также более эффективные методы поиска уникальных комбинаций атрибутов.

Список литературы

[1] Serra James. Deciphering Data Architectures. Choosing Between a Modern Data Warehouse,Data Fabric, Data Lakehouse, and Data Mesh. O'Reilly 2024. – 252 p.

[2] Armstrong W.W. Dependency Structure of Data Base Relationships // Proc. IFIP Congress. – Geneva, Switzerland, 1974. – P. 580–583.

[3] Chen P.P. The Entity-Relationship Model-Toward a Unified View of Data // ACM Transactions on Database Systems. – March 1976. – Vol. 1, No 1. – P. 9–36.

[4] Карпук А.А., Острейко В.М. Построение информационно-логической модели предметной области при проектировании базы данных // Вопросы радиоэлектроники. Сер. ОТ. – 1981. – Вып. 12. – С. 23–28.

[5] Karpuk A. Bringing the Subject Domain Ontology to Optimal Canonical Form // Open Semantic Technologies for Intelligent Systems: Research Papers Collection. Issue 8 – Minsk: BSUIR, 2024. – P. 237–242.

[6] Discover Dependencies from Data - A Review / Liu J. [et al.] // IEEE Transactions on Knowledge and Data Engineering (TKDE). – Vol. 24, No 2. – 2012. – P. 251–264.

[7] TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies / Huhtala Y. [et al.] // Computer Journal. – Vol. 42, No 2. – 1999. – P. 100–111.

[8] Novelli N., Cicchetti R. FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies // Proceedings of 8th International Conference Database Theory, ser. ICDT. – 2001. – P. 189–203.

[9] Yao H., Hamilton H.J., Butz C.J. FD Mine: Discovering Functional Dependencies in a Database Using Equivalences // Proceedings of 2002 IEEE International Conference on Data Mining, ser. ICDM. – 2002. – P. 729–732.

[10] Abedjan Z., Schulze P., Naumann F. DFD: E_Cient Functional Dependency Discovery // Proceedings of the International Conference on Information and Knowledge Management (CIKM). – 2014. – P. 949–958.

[11] Flach A., Savnik I. Database Dependency Discovery: A Machine Learning Approach // AI Communications. – 1999. – Vol. 12., No 3. – P. 139–160.

[12] Lopes S., Petit J.M., Lakhal L. Efficient Discovery of Functional Dependencies and Armstrong Relations // Proceedings of the 7th International Conference on Extending Database Technology, ser. EDB. – 2000. – P. 350–361.

[13] Wyss C., Giannella C., Robertson E. FastFDs: A Heuristicdriven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances // Proceedings of International Conference on Data Warehousing and Knowledge Discovery. – 2001. – P. 101–110.

[14] Efficient Discovery of Functional Dependencies on Massive Data / Wan X. [et al.] // IEEE Transactions on Knowledge and Data Engineering. – Vol. 36, No 1. – 2024. – P. 107–121.

[15] Papenbrock T., Naumann F. A Hybrid Approach to Functional Dependency Discovery // Proceedings of the International Conference on Management of Data, ser. SIGMOD'16. – 2016. – P. 821–833.

[16] Wei Z., Link S. Towards the Efficient Discovery of Meaningful Functional Dependencies // Information Systems. – Vol. 116. – 2023. – P. 102–224.

[17] Delobel C., Casey R.G. Decomposition of a Data Base and the Theory of Boolean Switching Functions // IBM J. Res. And Dev. – 1973. – Vol. 17, No 5. – P. 374–386.

[18] Карпук А.А. Анализ структуры функциональных зависимостей между атрибутами реляционной базы данных // Экономика и менеджмент систем управления. – 2017. – № 3 (25). – С. 64–70.

[19] Discovering Functional Dependencies through Hitting Set Enumeration / Bleifuss T. [et al.] // Proceedings of the ACM on Management of Data (SIGMOD). – Vol. 2, No 1. – 2024. – Article 43, P. 43:1–43:24.

[20] Карпук А.А., Краснопрошин В.В. Циклы в структурах функциональных зависимостей // International Journal of Open Information Technologies. – 2017. – Vol. 5, No 7. – P. 38–44.

Авторский вклад

Карпук Анатолий Алексеевич – руководство исследованием методов выделения и алгоритмов поиска ФЗ между атрибутами в таблицах систем Big Data, постановка задач исследования, оценка результатов исследования и определение направлений дальнейшей работы по совершенствованию и разработке алгоритмов поиска ФЗ между атрибутами в системах Big Data.

Лазута Леонид Сергеевич – анализ методов выделения и алгоритмов поиска ФЗ между атрибутами в таблицах систем Big Data, определение области применения алгоритмов поиска ФЗ между атрибутами, разработка предложений по совершенствованию алгоритмов.

IDENTIFICATION AND SEARCH FOR FUNCTIONAL DEPENDENCIES BETWEEN ATTRIBUTES IN BIG DATA SYSTEMS

A.A. Karpuk
Professor, Department of Telecommunication Network Software, Belarusian State Academy of Communications, PhD of Technical sciences, Associate Professor

L.S. Lazuta
Postgraduate student, Department of Telecommunication Systems, Belarusian State Academy of Communications, Master´s Degree

Abstract. The article provides an overview of methods for identifying and algorithms for searching for functional dependencies between attributes in Big Data systems. The article considers the main algorithms for searching for functional dependencies based on the analysis of the attribute lattice, based on the analysis of table rows, and hybrid algorithms. The areas of application of each type of algorithms and directions for further research on improving and developing algorithms for searching for functional dependencies between attributes in Big Data systems are determined.

Keywords: big data, functional dependency, functional dependency extraction, functional dependency search algorithms.