ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2014. Т. 8. Вып. 4. С. 94–109

МЕТОДЫ РАЗРЕШЕНИЯ СУЩНОСТЕЙ И СЛИЯНИЯ ДАННЫХ В ETL-ПРОЦЕССЕ И ИХ РЕАЛИЗАЦИЯ В СРЕДЕ HADOOP*

А. Е. Вовченко[1], Л. А. Калиниченко[2], Д. Ю. Ковалев[3]

Аннотация: При интеграции данных из совокупности исходных коллекций важной задачей является извлечение сущностей, их трансформация и загрузка в интегрированное хранилище. Такие действия являются частью ETL-процесса (extract–transform–loading). Под сущностью здесь понимается некоторое цифровое представление объекта реального мира (например, информация о персонах). При извлечении сущностей возникает проблема их разрешения: из различных ресурсов можно извлечь различную информацию об одном и том же объекте реального мира. Проблема разрешения сущностей ориентирована на решение таких задач, как идентификация сущностей, выявление дубликатов, удаление дубликатов, установление связей между сущностями, сопоставление сущностей с некоторым шаблонным образцом и др. После разрешения сущностей следует этап их слияния — формирование интегрированных сущностей (содержащих информацию из всех связанных сущностей). Слияние сущностей является заключительным этапом интеграции данных. В работе дан обзор методов разрешения и слияния сущностей. Рассматриваются вопросы адаптации таких методов для применения в ETL-процессе при интеграции больших данных в Hadoop. Также рассматриваются способы программирования методов разрешения и слияния сущностей как частей ETL-процесса. В качестве языка программирования используется HIL (High-Level Integration Language) — декларативный язык, ориентированный на разрешение и интеграцию сущностей в Hadoop-инфраструктуре.

Ключевые слова: интеграция данных; ETL; разрешение сущностей; слияние сущностей; большие данные; Hadoop; Jaql; HIL

DOI: 10.14357/19922264140412

1 Введение

В течение нескольких последних лет информатика стала играть все возрастающую роль в широком наборе научных дисциплин, особенно заметную из-за существенных проблем, вызванных взрывоподобным ростом данных в таких науках. X-информатика образовалась как совокупность академических дисциплин, направленных на применение средств информатики для федерализации, организации и анализа данных в конкретных областях науки с интенсивным использованием данных (НИИД): X = астро-, био-, гео-, нейро- и пр.

Сложность использования данных в НИИД усугубляется еще и вследствие естественной разнородности моделей обрабатываемых данных, в которых представляют тексты, графы, структурированную и слабоструктурированную информацию и пр. Разнообразие обрабатываемой информации вызывается, в частности, не только большим числом источников поступления обрабатываемой информации, но и разнообразием объектов исследования, непрерывным и быстрым совершенствованием инструментов, вызывающим адекватные изменения структуры и содержания накапливаемой информации. Это приводит к необходимости использования неоднородной, распределенной информации, накопленной в течение значительного периода наблюдений технологически различными инструментами.

Для анализа больших объемов накапливаемых данных используются современные распределенные инфраструктуры обработки массивных данных (например, Hadoop [1, 2]). Основной особенностью подобных инфраструктур является почти линейная горизонтальная масштабируемость (производительность системы растет линейно относительно числа узлов кластера), а также высокая отказоустой-

* Работа выполнена при поддержке РФФИ (проекты 13-07-00579, 14-07-00548), ИПИ РАН (Тема 38.25 «Спецификация и решение задач анализа данных в концептуальных терминах предметных областей с интенсивным использованием данных» государственного задания ФГБУН ИПИ РАН) и Президиума РАН (Программа фундаментальных исследований Президиума РАН № 16 «Фундаментальные проблемы системного программирования»).

[1] Институт проблем информатики Российской академии наук, alexey.vovchenko@gmail.com

[2] Институт проблем информатики Российской академии наук; Московский государственный университет им. М. В. Ломоносова, факультет вычислительной математики и кибернетики, leonidk@synth.ipi.ac.ru

[3] Институт проблем информатики Российской академии наук, dm.kovalev@gmail.com

чивость (отказ любого узла кластера не должен влиять на работоспособность системы в целом).

Главным достоинством подобных инфраструктур является возможность анализировать и обрабатывать разноструктурированные данные, например реляционные, XML, JSON, NoSQL, текстовые и др. При этом возникает проблема интеграции информации, извлекаемой из таких разноструктурированных данных.

Процесс интеграции данных (рассматриваемый здесь как ETL-процесс) можно представить состоящим из следующих этапов:

- сопоставление схем;
- интеграция схем;
- трансформация данных;
- разрешение сущностей (Entity Resolution [3–5]);
- слияние сущностей (Data Fusion [6]).

В данной работе рассматриваются последние два этапа. При интеграции сырых разноструктурированных данных задачей ETL-процесса является извлечение сущностей из исходных коллекций, их разрешение, трансформация и загрузка в интегрированное хранилище. Под сущностью здесь понимается некоторое цифровое представление объекта реального мира (например, информация о персонах). При извлечении сущностей возникает проблема их разрешения: из разных ресурсов можно извлечь разную информацию об одном и том же объекте реального мира. В общем случае под термином разрешения сущностей (entity resolution (ER) [3–5, 7–10]) понимают извлечение информации об одной и той же сущности реального мира из разнообразных структурированных, слабоструктурированных и неструктурированных коллекций данных и приведение извлеченных данных к унифицированному представлению. При этом применяются методы извлечения, сопоставления, группирования, связывания устранения дублирования различных представлений информации. Подходы к разрешению сущностей рассматриваются в разд. 2. Слияние сущностей является заключительным этапом интеграции данных. Под слиянием сущностей [6, 11–13] понимается образование интегрированного представления информации об одной и той же же сущности реального мира, полученной из разных источников. Операции и процедуры, используемые при слиянии сущностей, приведены в разд. 3.

В разд. 4 приводится описание средств программирования в среде Hadoop, а также обоснование выбора в работе средств (языки Jaql и HIL). Наконец, в разд. 5 показаны примеры программирования методов разрешения сущностей и слияния данных как части ETL-процесса в Hadoop.

2 Методы разрешения сущностей

Этап разрешения сущностей важен для сохранения исходной информации в интегрированной коллекции. Кроме того, важно обнаруживать дубликаты сущностей, поскольку, например, увеличение числа узлов и ребер в сетевых задачах может существенно удлинять время работы простейших алгоритмов (например, поиска кратчайшего пути). Алгоритмы разрешения сущностей (включая поиск дубликатов — duplicate detection) часто используются и во всевозможных поисковых системах, таких как Google, Amazon и др. Аналогичная проблема встает остро в различных агрегаторах информации (например, новостных агрегаторах).

В общем случае процесс разрешения сущностей включает следующие этапы [10]:

- подготовка данных;
- выбор методов сопоставления значений;
- определение методов разрешения пар сущностей;
- определение зависимостей (constraints).

2.1 Методы сопоставления значений

Важным действием для успешного разрешения сущностей является подготовка данных, которая включает нормализацию схем и нормализацию данных. Нормализация схем — непростая задача, подробное ее рассмотрение выходит за рамки данной работы и содержится в работах по интеграции данных в традиционных архитектурах или в инфраструктурах больших данных. Ниже представлен пример списка действий, которые могут быть отнесены к нормализации схем:

- сопоставление атрибутов схем (например, «контактный телефон» и «мобильный телефон»);
- слияние атрибутов (например, «полный адрес» образуется из атрибутов «город», «индекс», «улица», . . .);
- слияние множественных значений и списков (например, «контактные телефоны» и «основной номер телефона», «дополнительный номер телефона») и др.

Нормализация данных может включать приведение к строчному или заглавному регистру; удаление разделителей; поиск и исправление опечаток; поиск сокращений и аббревиатур и замена их на

полные стандартные формы; использование словарей для нормализации строк и многое другое. Нормализация данных, так же как и нормализация схем, не рассматривается в данной статье.

Для сопоставления сущностей важно определиться с выбором метода оценки сходства (similarity) значений. Рассматриваются как булевы, так и вещественные меры сходства. Следующий список включает примеры часто используемых методов оценки сходства значений:

- эквивалентность булевых предикатов;
- вычисление функции сходства простых значений (расстояние Левенштейна [14], алгоритм Смита–Ватермана [14]);
- вычисление функции сходства множеств (коэффициент Жаккара [14], коэффициент сходства Дайса [14], коэффициент Адара [15]);
- вычисление функции сходства векторов (коэффициент косинусов [14], статистическая мера TFIDF (term frequency – inverse document frequency) [14]);
- оценка сходства на основе выравнивания (сходство Джаро–Винклера [14], статистическая мера Soft-TFIDF [16], расстояние Монг–Элкана [17]);
- оценка сходства фонетических данных (алгоритм сравнения двух строк по их звучанию Soundex [14]);
- оценка сходства, основанная на переводе (может использоваться для нормализации аббревиатур);
- оценка сходства, основанная на знаниях о предметной области.

Также существуют специальные методы для определения сходства отношений. Меры, используемые для отношений, обычно основаны на сходстве множеств и предполагают использование функций вычисления сходства множеств.

2.2 Методы сопоставления пар сущностей

2.2.1 Традиционные методы сопоставления пар сущностей

Пусть даны две коллекции объектов с атрибутами: author, venue, paper. Значение некоторой меры сходства (одной из приведенных в подразд. 2.1) для конкретного атрибута будем обозначать

$$\text{XX-match-score}(\text{author-match-score}, \text{venue-match-score}, \text{paper-match-score}).$$

Традиционным методом сравнения объектов является подсчет сходства некоторым алгоритмом (см. подразд. 2.1) для каждого из атрибутов независимо. Затем реализуется подсчет взвешенной суммы.

Например:

$$0{,}5\text{autor-match-score} + 0{,}2\text{venue-match-score} + 0{,}3\text{paper-match-score}.$$

Недостатком такого подхода является сложность выбора весов для каждого из атрибутов и сложность выбора порога сходства сущностей.

Другой метод предполагает задание булевого обобщенного правила, где условия накладываются на каждый атрибут независимо.

Например:

$$(\text{author-match-score} > 0{,}7 \text{ AND venue-match-score} > 0{,}8) \text{ OR } (\text{paper-match-score} > 0{,}9 \text{ AND venue-match-score} > 0{,}9).$$

Недостатком этого подхода является сложность формулирования подобных правил вручную.

2.2.2 Методы машинного обучения для сопоставления пар сущностей

Для сопоставления пар сущностей применяют также специальные методы машинного обучения, которые позволяют автоматизировать процесс формулирования критериев для сопоставления сущностей. Использование таких методов основано на применении теории Феллеги и Сантера [14] для связывания сущностей. Рассмотрим этот подход подробнее.

Пусть даны коллекции A и B.

Пусть r — это пара $r(x, y)$, где $x \in A$, $y \in B$.

Пусть $\gamma = \gamma(r)$ — это вектор сравнения, например:

$$\gamma(r) = \{x.\text{author} = y.\text{author}, x.\text{venue} = y.\text{venue}, x.\text{paper} = y.\text{paper}\},$$

$\gamma(r) = \{\text{true}, \text{false}, \text{true}\}$ — пример, в случае если $x.\text{author} = y.\text{author}$, $x.\text{venue} \neq y.\text{venue}$, $x.\text{paper} = y.\text{paper}$.

Пусть M — множество всех пар, являющихся дубликатами.

Пусть U — множество всех пар, не являющихся дубликатами.

Тогда правило для определения сходства сущностей можно описать следующей формулой:

$$R(r) = \frac{m(\gamma)}{u(\gamma)} = \frac{P(\gamma | r \in M)}{P(\gamma | r \in U)}.$$

Определим два порога T_I и t_U, такие что

- $R(r) \leq t_I$ — объекты (пара) не являются дубликатами;

– $R(r) > t_I$ AND $R(r) < t_u$ — невозможно определить, являются ли объекты (пара) дубликатами или нет;
– $R(r) \geq t_u$ — объекты (пара) являются дубликатами.

Правилом связывания, обозначаемым $L(t_I, t_U)$, называется пара порогов t_I и t_U.

При подобном подходе учитываются стандартные для задачи проверки статистических гипотез ошибки первого и второго рода. Ошибки первого рода — два объекта не являются дубликатами ($r(x, y) \in U$), однако правило L относит их к дубликатам. Ошибки первого рода обозначаются буквой μ и их можно описать формулой:

$$\mu = P(L_{\text{match}}|U) = \sum_{\gamma} u(\gamma) P\left(L_{\text{match}}|\gamma\right) .$$

Ошибки второго рода — два объекта являются дубликатами ($r(x, y) \in M$), однако правило L определяет, что это не дубликаты. Ошибки второго рода означаются буквой λ и их можно описать формулой:

$$\lambda = P(L_{\text{nonmatch}}|M) = \sum_{\gamma} m(\gamma) P\left(L_{\text{nonmatch}}|\gamma\right).$$

Оптимальным правилом связывания $L^*(t_I^*, t_U^*)$ называется такое правило, которое соответствует ограничениям на ошибки первого и второго рода для правила, а также ограничения на неопределенности. Эти ограничения выражаются следующими формулами:

– ограничения на ошибки:

$$P\left(L^*_{\text{match}}|U\right) \leq \mu\,; \qquad P\left(L^*_{\text{nonmatch}}|M\right) \leq \lambda\,;$$

– ограничения на неопределенности:

$$P\left(L^*_{\text{uncertain}}|U\right) \leq P\left(L_{\text{uncertain}}|U\right)\,;$$
$$P\left(L^*_{\text{uncertain}}|M\right) \leq P\left(L_{\text{uncertain}}|M\right)\,.$$

Нахождение оптимального правила является основной задачей при использовании теории Феллеги и Сантера. Классические (переборные) методы при таком подходе работают неэффективно, поэтому нахождение оптимального правила достигается с помощью средств машинного обучения, например можно использовать наивный байесовский классификатор. Одна из основных проблем при этом заключается в том, что для вычисления $P(\gamma|r \in M)$ и $P(\gamma|r \in U)$ необходимы знания о том, какие объекты являются дубликатами, а какие нет (знания о множествах M и U).

Для разрешения сущностей применяются различные реализации подходов, основанных на алгоритмах машинного обучения и использовании теории Феллеги и Сантера, например использование:

– деревьев решений [18];
– метода опорных векторов [19, 20];
– ансамблей классификаторов [21];
– метода условных случайных полей [22].

К недостаткам этих подходов можно отнести несбалансированность результирующих классифицированных множеств (так, в результате образуется значительно больше непохожих объектов, чем похожих), а также высокую вероятность того, что объект не будет причислен ни к какому классу (из-за неопределенности). Но оба этих недостатка могут быть устранены путем тонкой настройки алгоритмов. Ключевой проблемой при использовании методов машинного обучения при сравнении пар сущностей является выбор обучающего множества.

Выделяют следующие методы классификации сущностей, не требующие построения обучающей выборки:

– обучение без учителя или с частичным привлечением учителя [14, 23];
– методы с активным обучением;
– ансамбли классификаторов [24, 25];
– доказуемая оптимизация точности/полноты [26, 27];
– краудсорсинг [28, 29].

Подводя итог методам разрешения пар сущностей, выделим методы, основанные на мерах сходства, и методы, основанные на использовании машинного обучения. Общим недостатком первой группы методов является сложность формулирования критериев сходства (подбор весов или явных формул). Методы машинного обучения лишены этого недостатка в силу своей структуры, но при этом ключевой проблемой является выбор обучающего множества, да и сами методы значительно сложнее. Перспективными (но все еще мало изученными) представляются методы машинного обучения, не требующие изначального определения обучающего множества, такие как методы, основанные на активном обучении и краудсорсинге.

2.3 Использование ограничений

После того как определен метод разрешения конкретных пар сущностей, можно определить зависимости. Далее представлены примеры зависимостей, используемых для установления сходства сущностей:

– транзитивность: если M1 и M2 похожи и M2 и M3 похожи, то и M1 и M3 похожи;

– эксклюзивность: если M1 и M2 похожи, то M3 не может быть похож на M2;

– функциональные зависимости: если M1 и M2 похожи, то M3 и M4 должны быть похожи.

Транзитивность часто используется в методах удаления дубликатов, а эксклюзивность используется в методах установления связей между сущностями.

В заключение можно отметить, что разрешение сущностей является быстро развиваемой областью. Исследуются новые меры сходства [14], ведутся работы по применению перспективных методов машинного обучения [24–29]. Развивается применение функциональных зависимостей при очистке данных (data cleaning) [30–32]. Ведутся работы по построению сущностей с наиболее представительными данными (включающими данные из разнообразных дубликатов — методы канонизации сущностей [33]). Также исследуются методы, в которых решения по сходству двух сущностей принимаются на основе анализа совокупности сущностей, применения вероятностных логик сходства, латентной модели Дирихле [34–36].

3 Методы слияния данных

Под слиянием данных [6, 12, 13] понимается образование интегрированного представления информации об одной и той же сущности реального мира, полученной из разных источников данных. Процесс слияния данных включает следующие задачи: слияние записей о сущностях, разрешение возможных конфликтов, обнаружение и удаление ошибочных данных. Методы слияния данных, кратко рассмотренные в данном разделе, исследованы в Потсдамском университете [13]. Различные аспекты проблемы слияния данных представлены на рис. 1.

3.1 Типы конфликтов при слиянии данных

Различают два типа конфликтов: конфликты, вызванные неопределенными значениями, и конфликты, вызванные противоречивыми значениями.

Неопределенность означает, что в одном источнике данных содержатся неизвестные значения (null), а в другом — известные. Проблема заключается в том, что семантика неопределенных значений (null) может сильно отличаться. Различают три варианта: неизвестные значения, несуществующие значения (например, атрибут «имя супруга» всегда будет null для неженатых), скрытые значения (такие данные, которые по каким-то причинам не позволено видеть).

Противоречивость значений означает появление двух различных ненулевых (not null) значений. Возможны различные стратегии обработки подобных конфликтов, о чем рассказывается в следующем подразделе.

3.2 Стратегии разрешения конфликтов

Различают следующие подходы к разрешению конфликтов:

– игнорирование конфликтов;

– избегание конфликтов;

– разрешение конфликтов.

Стратегия игнорирования конфликтов предполагает извлечение всей доступной информации.

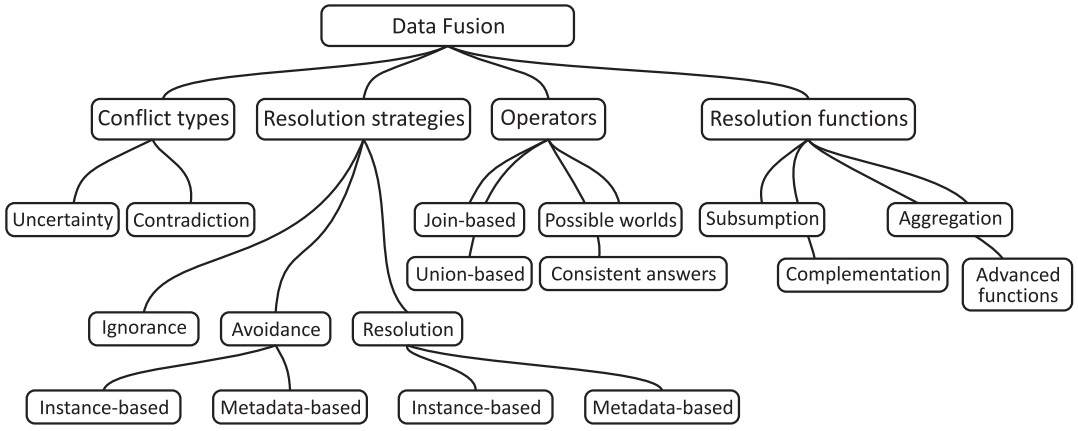

Рис. 1 Различные аспекты проблемы слияния данных

Примеры функций для разрешения конфликтов

Функция	Описание	Стратегия	Пример конфликта
Min, Max, Sum, Count, Avg	Обычная агрегация	Разрешение конфликта	Подсчет зарплаты (средней или максимальной), подсчет возраста или количества детей
Random	Случайное значение	Разрешение конфликта	Размер участка
Longest, Shortest	Самое короткое или длинное значение	Разрешение конфликта	Например, для имен
Choose (source)	Значение из конкретного ресурса	Избежание конфликта	Например, для финансовых данных, если принято решение доверять информации из Yahoo больше, нежели другим ресурсам
Choose Depending (val, col)	Выбирается значение в зависимости от значения в другом атрибуте	Избежание конфликта	Например, если выбран атрибут «город» из одного ресурса, то «почтовый индекс» разумно взять из того же самого ресурса
Vote	Голосование, решение по большинству	Разрешение конфликта	Например, для подсчета рейтинга
Coalesce	Выбор первого ненулевого значения	Избежание конфликта	Например, для имен
Group, Concat	Группировка или конкатенация всех значений	Избежание конфликта	Например, для отзывов о продуктах
MostRecent	Выбор наиболее свежего значения (недавно обновленного)	Разрешение конфликта	Например, если интересует последний адрес местожительства
Escalate	Сохранение всех конфликтующих значений, с тем чтобы пользователь сам решил, какое выбрать	Игнорирование конфликта	Например, для атрибута «пол» сложно придумать объективные причины выбора того или иного значения
...	...	...	...

Например, для строк это может быть обычная конкатенация строк, а пользователь уже сам решает, какие данные верны.

Стратегия избегания конфликтов предполагает выбор данных на основе самих данных (по некоторому алгоритму) или на основе метаданных. Примером функции на основе данных может служить функция coalesce (выбор первого ненулевого значения) или функция выбора самого длинного значения. Примером функций на основе метаданных может выступать выбор в зависимости от самого источника (например, известно, что один из источников наиболее достоверный). Другим примером является функция, выбирающая значение из того источника, в котором большее число значений было выбрано для других атрибутов.

Стратегии разрешения конфликтов учитывают все значения и выбирают из них «достоверное». Примером подобной функции могут выступать всевозможные функции голосования, функции выбора случайного значения, функции среднего значения, функции наиболее часто встречающегося значения и др.

В таблице представлены примеры функций для разрешения конфликтов.

3.3 Основные функции разрешения конфликтов

Вводится операция outer union [13], результатом которой является объединение двух отношений. Если схемы не совпадают, то результирующая схема является объединением двух исходных схем. Например, пусть даны два отношения: A с набором атрибутов $\{a, b, c, d\}$ и отношение B с набором атрибутов $\{c, d, e, f\}$. Результирующая схема будет содержать набор атрибутов = $\{a, b, c, d, e, f\}$. В результирующие кортежи для недостающих атри-

бутов помещаются нулевые значения. Эта операция не является стандартной и отсутствует в большинстве реляционных систем управления базами данных (СУБД). В реляционной алгебре подобная операция может быть представлена как

```
(SELECT a, b, c, d, NULL as e, NULL as f FROM A)
UNION
(SELECT NULL as a, NULL as b, c, d, e, f FROM B).
```

Вводится функция tuple subsumption [13]. Говорят, что кортеж t1 поглощает другой кортеж t2 (поглощаемый кортеж), если у них

- совпадают схемы;
- в t2 больше неизвестных (null) значений, чем в t1;
- в t2 все известные значения совпадают со значениями в t1.

Например, пусть даны кортежи $\mathrm{t1} = (5, \text{'text'}, \mathrm{null}, 7)$ и $\mathrm{t2} = (5, \mathrm{null}, \mathrm{null}, 7)$. Видно, что каждый атрибут в t2 либо совпадает с аналогичным атрибутом в t1, либо он null. Для этого примера кортеж t1 поглощает кортеж t2.

Вводится функция tuple complementation [13]. Говорят, что кортежи t1 и t2 дополняют друг друга, если

- у них совпадают схемы;
- они не совпадают;
- значения соответствующих атрибутов в t1 и t2 совпадают, либо одно из них не определено, либо оба не определены;
- t1 и t2 имеют как минимум один атрибут, значения которого совпадают.

Например, пусть даны кортежи $\mathrm{t1} = (5, \text{'text'}, \mathrm{null}, \mathrm{null})$ и $\mathrm{t2} = (5, \mathrm{null}, \mathrm{null}, 7)$. Видно, что кортежи дополняют друг друга. Результатом операции дополнения для этих двух кортежей будет новый кортеж $\mathrm{t} = (5, \text{'text'}, \mathrm{null}, 7)$.

3.4 Операторы слияния данных

Различают два основных подхода к слиянию данных. Эти подходы основаны на операции объединения (union-based) или на операции соединения (join-based). Различают следующие основные операции.

Minimum Union [13] (union-based). Операция представляет собой выполнение операции outer union, а затем удаление из результата всех поглощаемых (subsumed [13]) кортежей. Пример операции представлен на рис. 2.

Complemetation Union [13] (union-based). Операция представляет собой выполнение операции outer union, а затем дополнение (complementation) всевозможных кортежей. Пример операции представлен на рис. 3.

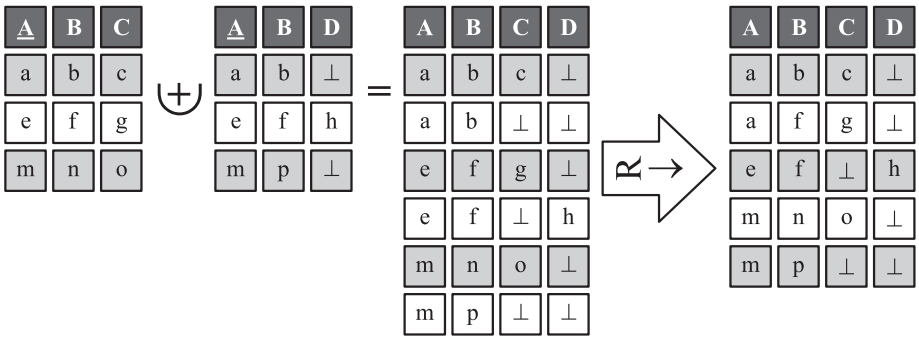

Рис. 2 Пример операции **Minimum Union**

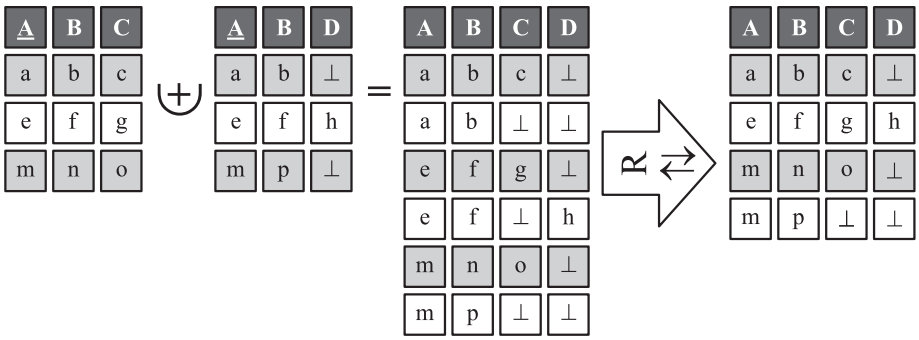

Рис. 3 Пример операции **Complementation Union**

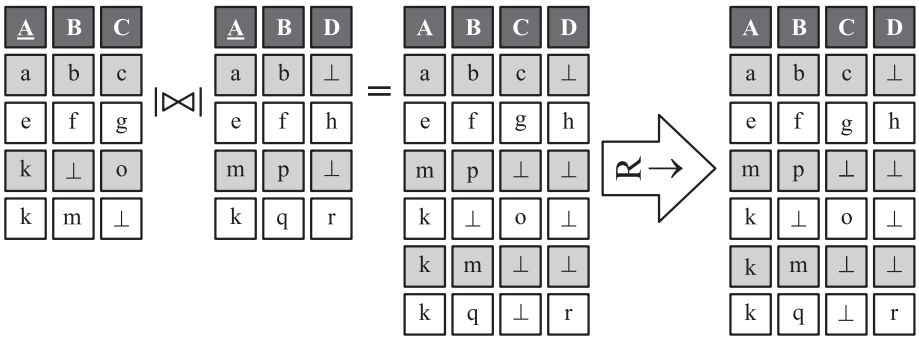

Рис. 4 Пример операции **Full Disjunction**

Grouping and Aggregation [13] (union-based). Операция предполагает выполнение outer union, а затем группировку по общему атрибуту и применение функции агрегации к остальным атрибутам. Пример операции на языке SQL представлен ниже.

```
WITH OU AS (
 (SELECT A, B, C, NULL AS D FROM U1)
   UNION (ALL)
 (SELECT A, B, NULL AS C, D FROM U2)),
SELECT A, MAX(B), MIN(C), SUM(D)
FROM OU
GROUP BY A
```

Full Disjunction [37] (join-based). Операция представляет собой full outer join (стандартную реляционную операцию), после чего применяется subsumption к результату. Пример представлен на рис. 4.

Match Join [13] (union- + join-based). В операции выбираются всевозможные комбинации значений атрибутов, после чего выполняется full outer join. Фактически реализуется outer union двух коллекций. Затем определяется $N-1$ вспомогательных отношений, где N — число атрибутов, а каждое из отношений содержит по два атрибута: один общий и какой-то другой. После чего происходит full outer join $(N-1)$-го отношения. Пример реализации операции на языке SQL представлен ниже.

```
WITH
 OU(A,B,C,D) AS (
 (SELECT A, B, C, NULL AS D FROM U1)
  UNION
 (SELECT A, B, NULL AS C, D FROM U2)),
   // ← Outer Union
 B_V(A,B) AS (SELECT DISTINCT A, B FROM OU),
   // ← 1-е отношение (N = 4)
 C_V(A,C) AS (SELECT DISTINCT A, C FROM OU),
   // ← 2-е отношение (N = 4)
 D_V(A,D) AS (SELECT DISTINCT A, D FROM OU),
   // ← 3-е отношение (N = 4)
SELECT A, B, C, D
FROM B_V FULL OUTER JOIN C_V FULL OUTER
    JOIN D_V // ← Full Outer Join
```

Merge (union- + join-based). Операция объединяет операции соединения и объединения. Для каждого общего атрибута формируются две версии значений, нулевые значения удаляются функцией COALESCE (выбор первого ненулевого значения). Пусть даны два отношения: A с набором атрибутов $\{a, b, c\}$ и B с набором атрибутов $\{a, b, d\}$. Пусть a — конфликтующий атрибут, b — атрибут с нулевыми значениями. Пример реализации операции на языке SQL представлен ниже, а результат показан на рис. 5.

```
(SELECT A.a, COALESCE(A.b, B.b), A.c, B.d
FROM A LEFT OUTER JOIN B ON A.a = B.a)
UNION
(SELECT B.a, COALESCE(B.b, A.b), A.c, B.d
FROM A RIGHT OUTER JOIN B ON A.a = B.a)
```

Grouping and Aggregation (union-based). Операция представляет собой группировку по некоторому атрибуту, а затем использование разнообразных агрегирующих функций. В качестве достоинства данного подхода можно выделить его реализацию в большинстве СУБД и эффективное выполнение. Пример реализации на SQL представлен ниже.

```
WITH OU AS (
 (SELECT A, B, C, NULL AS D FROM U1)
  UNION (ALL)
 (SELECT A, B, NULL AS C, D FROM U2)),
SELECT A, MAX(B), MIN(C), SUM(D)
FROM OU
GROUP BY A
```

Data Fusion оператор [11] — **Fuse By** (union-based). В некоторых системах пошли дальше использования стандартных операций группировки и агрегации. Ключевое слово FUSE BY используется вместо GROUP BY, и семантика у него аналогична. Вместо использования стандартных функций агрегации используется встроенная функция RESOLVE, которой параметром передается само значение и имя

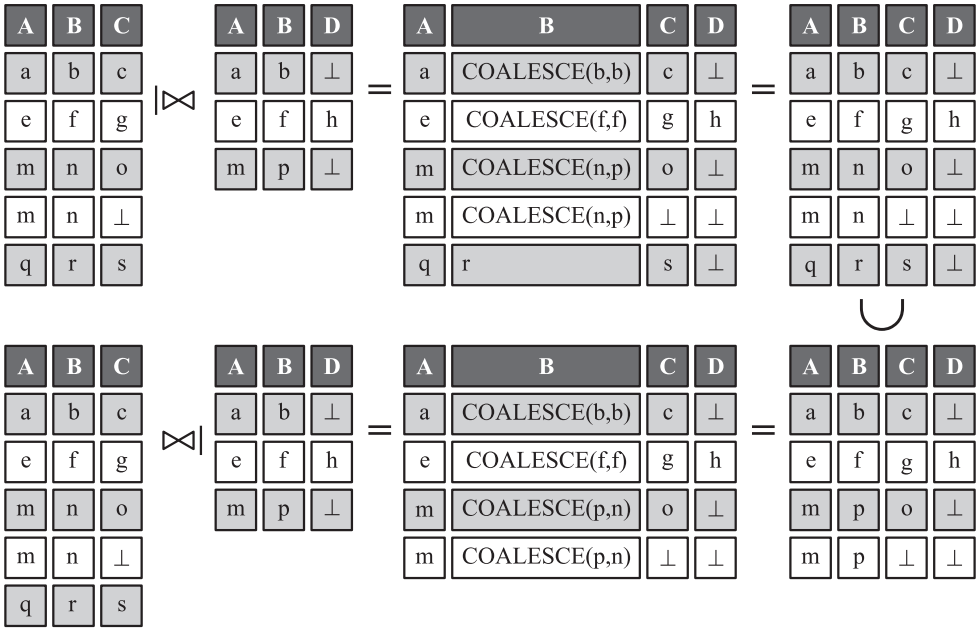

Рис. 5 Пример операции **Merge**

функции разрешения конфликтов. Пример реализации на SQL представлен ниже.

```
SELECT ID,
 RESOLVE(Title, Choose(IMDB)),
 RESOLVE(Year, Max),
 RESOLVE(Director, COALESCE),
 RESOLVE(Rating, COALESCE),
 RESOLVE(Genre, Concat)
FUSE FROM IMDB, Filmdienst
FUSE BY (ID)
ON ORDER Year DESC
```

4 Разрешение сущностей в больших данных

Для манипулирования большими разноструктурированными данными служат Hadoop-инфраструктуры [1, 2], предоставляющие масштабируемое хранилище и обеспечивающие высокую скорость анализа больших данных за счет распределенной их обработки. Для применения методов разрешения сущностей в такой среде нужна *адаптация* алгоритмов для их распределенного выполнения на различных узлах Hadoop-кластера.

В среде Hadoop реализована парадигма распределенного программирования для анализа данных Map-Reduce [38, 39], называемая по именам основных функций. Вначале на всех узлах кластера обрабатываются блоки данных независимо друг от друга (Map). После чего данные группируются по заранее выбранным для алгоритма ключам и поступают на выполнение на один или более узлов в зависимости от алгоритма (Reduce).

Таким образом, для реализации любого алгоритма в Hadoop-инфраструктуре требуется его адаптация к виду Map-Reduce. Другим вариантом является реализация алгоритма на одном из языков высокого уровня, таких как Pig [40], Hive [41], Jaql [42]. Все эти языки автоматически переписывают программы, реализованные на них, в Map-Reduce-приложения для выполнения на Hadoop-кластере.

В случае больших данных и распределенных инфраструктур традиционные подходы требуют доработок. Различают два основных метода разрешения сущностей над большими данными: разбиение данных на блоки (blocking [43, 44]) и распределенный метод разрешения сущностей.

Суть разбиения на блоки заключается в следующем. Пусть имеется 1000 компаний в 1000 городах. Требуется сравнить компании. Алгоритм полного попарного сравнения потребует 10^{12} сравнений. При этом если предположить, что компании из разных городов не могут совпадать, то потребуется 10^9 сравнений. Ключевой проблемой данного подхода является выбор критерия, по которому разбиваются данные.

Различают два основных метода: основанный на хэш-функции [10] и основанный на сходстве соседей [10]. Метод, основанный на хэш-функции, предполагает разбиение на блоки по хэш-ключу.

Основной проблемой алгоритма является выбор хэш-функции. Метод, основанный на сходстве соседей, предполагает, что совпадать могут только объекты, похожие по некоторой мере. Все объекты сортируются по какому-то признаку (ключу — простому или составному, уникальность ключа не требуется). После этого выбирается размер окна, внутри которого объекты сравниваются. Проблемой данного метода является выбор ключа сортировки.

Распределенный метод разрешения сущностей предполагает реализацию традиционных алгоритмов этого семейства в виде Map-Reduce-приложения, что требует зачастую полного пересмотра исходного алгоритма. Другой вариант — реализация алгоритма разрешения сущностей на специализированных языках, чему будет посвящен следующий раздел. Третий вариант — использование специализированных инструментов, направленных на распределенное выполнение методов разрешения сущностей над Hadoop [45].

5 Реализация операций разрешения сущностей и слияния данных в среде Hadoop

Язык HIL [46] — декларативный язык, ориентированный на разрешение и интеграцию сущностей в Hadoop инфраструктуре. HIL компилируется в язык Jaql [43, 44], который, в свою очередь, автоматически переписывается в Map-Reduce, если этого требует алгоритм.

5.1 Реализация методов разрешения сущностей

Пусть даны структуры данных, включающие три атрибута: id, value, name. Тогда простейшее правило разрешения сущностей на языке HIL будет выглядеть следующим образом:

```
declare Duplicated: ?;
declare Generated: ?;
declare Deduplicated: ?;

create link Deduplicated as
select
[gen: [id: g.id, name: g.name, value: g.value],
dup: [id: d.id, name: d.name, value: d.value]]
from Generated g, Duplicated d
match using
 rule_id: g.id = d.id,
 rule_name: g.name = d.name,
 rule_value: g.value = d.value;
```

В этом примере используется простое сопоставление сущностей по совпадению значений. Если требуется ввести какую-то функцию меры для значений, это можно реализовать внешней функцией Jaql:

```
@jaql{
compareValue =
    javaudf("org.ipiran.similarity.ValueSimilarity");
}
```

После этого такую функцию можно вызывать из языка HIL:

```
declare compareValue: function ? to ?;
declare Duplicated: ?;
declare Generated: ?;
declare Deduplicated: ?;

create link Deduplicated as
select
[gen: [id: g.id, name: g.name, value: g.value],
dup: [id: d.id, name: d.name, value: d.value]]
from Generated g, Duplicated d
match using
rule_id:
 compareValue(g.id, d.id) > 0.7,
rule_name:
 compareValue(g.name, d.name) > 0.7,
rule_value:
 compareValue(g.value, d.value) > 0.7;
```

Можно также ввести меру для сравнения объектов целиком.

Пусть описана функция compareObject, которая принимает на вход объекты. Тогда правило на языке HIL изменится, так как в этом случае используется другой вид правил:

```
insert into Deduplicated
select
[gen: [id: g.id, name: g.name, value: g.value],
dup: [id: d.id, name: d.name, value: d.value],
value: compareObject(g,d)]
from Generated g, Duplicated d
where compareObject(g, d) > 0.7;
```

Во всех этих случаях происходит сравнение всех объектов со всеми, сложность подобного сравнения $O(n^2)$. Несмотря на то что сравнения будут выполняться независимо и распределены на всех узлах кластера (так как HIL переписывается в Jaql, а тот, в свою очередь, в Map-Reduce), время их выполнения может быть довольно большим. Для уменьшения числа сравнений, как было описано в разд. 4, можно разбивать данные на блоки.

Пусть имеется функция calcHash, которая вычисляет хэш для объектов. В результате функция может выдавать столько уникальных значений, на сколько блоков требуется разбить данные. Тогда, объединив правила, рассмотренные выше, выбрав

вначале те объекты, что совпадают по хэш-функции, а далее, вычислив общую меру, можно получить результат за более короткое время:

```
declare calcHash: function ? to ?;
insert into GeneratedHash
select [$.*, hash: calcHash($.*)]
from Generated;
insert into DuplicatedHash
select [$.*,hash: calcHash($.*)]
from Duplicated;

create link Deduplicated as
select [
 gen: [id: g.id, name: g.name, value: g.value],
 dup: [id: d.id, name: d.name, value: d.value]]
from GeneratedHash g, DuplicatedHash d
match using
 rule_id: g.hash = d.hash;
insert into Measured
select [gen: dd.gen, dup: dd.dup, value:
  compareObject(dd.gen, dd.dup)]
from Deduplicated dd
where compareObject(dd.gen, dd.dup) > 0.8
```

5.2 Реализация методов слияния данных

Будем считать, что этап разрешения сущностей уже пройден и дана некоторая коллекция Deduplicated, где уже установлены соответствия одним из вышеперечисленных способов. Например, пусть имеются две коллекции: A (id, a, b, c) и B (id, a, b, d). Атрибуты a, b, c, d могут содержать null-значения, атрибуты id совпадают. Ниже дан пример подобных данных для коллекции A в формате JSON:

```
[{"a":null,"b":null,"c":"wmqhxfgmac",
"id":919132322},
{"a":null,"b":null,"c":"wmqhxfgmac",
"id":919132322}]
```

Тогда коллекция разрешенных сущностей может быть получена следующим образом:

```
create link Deduplicated as
select
[gen: [id: a.id, a:a.a, b:a.b, c:a.c],
dup: [id: b.id, a:b.a, b:b.b, d:b.d]]
from A a, B b
match using
 rule1: a.id = b.id;
```

Рассмотрим теперь реализацию Minimum Union и оператор Fusion [11] на языке HIL.

Как было определено в разд. 3, **Minimum Union** — это последовательное применение операций outer union и subsumption [13]. Outer Union фактически реализуется с помощью индекса FusionIndex. Использование индекса оправдано, так как существует несколько записей, описывающих одну сущность. Ключом является атрибут id. Ниже представлена реализация операции Outer Union:

```
insert into FusionIndex![id: f.gen.id] select [a: f.gen.a,
b: f.gen.b, c: f.gen.c] from Deduplicated f;

insert into FusionIndex![id: f.dup.id] select [a: f.dup.a,
b: f.dup.b, d: f.dup.d] from Deduplicated f;
```

Далее для реализации subsumption требуется удалить все ненужные кортежи. Это делается на языке Jaql. Для этого нужна функция, которая бы определяла, поглощается ли один кортеж другим. К сожалению, в языке Jaql нет возможностей написания общих (generic) методов, универсальных для всех коллекций, поэтому функцию сравнения можно реализовать на Java и подключить к языку Jaql подобно тому, как демонстрировалось в подразд. 5.1 на примере функций вычисления меры. Либо же можно реализовать функцию для сравнения конкретных коллекций на языке Jaql, как показано ниже:

```
is_subsumed = fn(i,j) ((
isnull(j.a) or (i.a == j.a) ) and (isnull(j.b) or
 (i.b == j.b)) and (isnull(j.c) or (i.c == j.c)) and
 (snull(j.d) or (i.d == j.d)) and (i! = j));
```

Функция **is_subsumed(i,j)** проверяет, поглощает ли один кортеж другой при помощи попарного сравнения атрибутов или проверки на null.

```
removeSubsumed = fn (a) (b = a,
subs = for (i0 in b) [a → filter is_subsumed(i0,$)],
    s = subs → expand,
a → filter not $ in s);
```

Функция **removeSubsumed** удаляет все поглощенные записи из кортежа. Здесь реализован наивный алгоритм, который попарно для каждого кортежа находит все поглощенные им и удаляет их.

```
minUnion = fn(id,a) ( {id:id, minunion:
 removeSubsumed(a)});
```

Функция **minUnion** нужна для построения результирующих кортежей при реализации Minimum Union. С ее помощью операцию Minumum Union можно описать следующим образом на языке HIL:

```
insert into MinimumUnion
select minUnion(i.dup.id,
       FusionIndex![id : i.dup.id])
from Deduplicated i;
```

Для каждого id достаются все соответствующие записи и удаляются те, которые ими поглощаются.

Оператор **Data Fusion** [11] представляет собой особый вид функции, использующий группировку для преодоления конфликтов. Основная идея заключается в группировке различных представлений одной и той же сущности по общему атрибуту, а затем в применении функций разрешения конфликтов для всех остальных атрибутов, сливая данные

в одну сущность. Различают два вида стратегии для функций разрешения конфликтов:

(1) deciding-стратегия заключается в выборе какого-то одного значения каким-то способом (минимум, максимум, случайное значение);

(2) mediating-стратегия заключается в агрегации всех значений (среднее значение, сумма).

Пусть имеются две коллекции: A (id, name, age) и B (id, name, info), пример которых дан ниже:

A

```
[{"id":760046903,"name":null,"age":null},
{"id":15009544, "name": "zvqcsxkzxk",
"age":938781652}]
```

B

```
[{"id":15009544, "name":null, "info":null},
{"id":760046903, "name":"pjltaghyug","info":null}]
```

Пусть для них пройден этап разрешения сущностей и построена коллекция Deduplicated, как описано выше в этом разделе. Пусть также для этих данных построен индекс FusionIndex, как показано выше для операции Minimum Union. Тогда оператор Data Fusion на языке HIL может быть описан следующим образом:

```
@jaql{
average = fn($a) avg($a[*].age);
any = fn($a) any($a[*].name);
concat = fn ($a) strJoin($a[*].info,"_");
}
insert into Fused
select [
id: i.dup.id,
age:
 average(FusionIndex![id: i.dup.id]),
name:

any(FusionIndex![id: i.dup.id]),
info:
 concat(FusionIndex![id: i.dup.id])]
from Deduplicated i;
```

Функции вычисления среднего, выбора случайного ненулевого значения, а также конкатенации реализованы на Jaql. Данное правило образует коллекцию **Fused**, причем для атрибута age будет подсчитано среднее значение, для имени name выбрано любое ненулевое значение, а для атрибута info будет получена конкатенация всех доступных значений. Таким образом, в данном примере показана реализация обеих стратегий для функций разрешения конфликтов в операторе Data Fusion.

6 Заключение

Рассмотренные методы и операции извлечения и интеграции информации о сущностях реального мира, представленной сырыми разноструктурированными коллекциями данных, позволяют программировать интеграционные потоки вида ETL для образования интегрированных структурированных данных, которые могут быть использованы в приложениях для дальнейшего анализа и обработки. В статье рассмотрены методы разрешения сущностей и слияния данных. В статье показаны способы программирования методов и операций извлечения и интеграции информации о сущностях реального мира, включая методы слияния данных, на декларативном языке HIL.

Литература

1. *White T.* Hadoop: The definitive guide. — 3rd ed. — O'Reilly Media, 2012. 688 p.
2. Apache Hadoop 2.5.1. http://hadoop.apache.org.
3. *Naumann F., Herschel M.* An introduction to duplicate detection. Synthesis lectures on data management. — Morgan & Claypool, 2010. Lecture No. 3. 87 p.
4. *Christen P.* Data matching — concepts and techniques for record linkage, entity resolution, and duplicate detection. Data-centric systems and applications ser. — Springer, 2012. 272 p.
5. *Fan W., Geerts F.* Foundations of data quality management. Synthesis lectures on data management. — Morgan & Claypool, 2012. Lecture No. 29. 217 p.
6. *Bleiholder J., Naumann F.* Data fusion // ACM Computing Surveys (CSUR), 2009. Vol. 41. Iss. 1. Article No. 1. doi: 10.1145/1456650.1456651.
7. *Köpcke H., Thor A., Rahm E.* Evaluation of entity resolution approaches on real-world match problems // Proc. VLDB Endowment, 2010. Vol. 3. Iss. 1-2. P. 484–493.
8. *Köpcke H., Rahm E.* Frameworks for entity matching: A comparison // Data Knowledge Engineering, 2010. Vol. 69. Iss. 2. P. 197–210. doi: 10.1016/j.datak.2009.10.003.
9. *Ganti V., Das Sarma A.* Data cleaning, a practical perspective. Synthesis lectures on data management. — Morgan & Claypool, 2013. Lecture No. 36. 85 p.
10. *Getoor L., Machanavajjhala A.* Entity resolution for big data // KDD'13: 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining Proceedings, 2013. P. 1527–1527.
11. *Bleiholder J., Naumann F.* Declarative data fusion — syntax, semantics, and implementation // East European Conference on Advances in Databases and Information Systems (ADBIS) Proceedings, 2005. P. 58–73.
12. *Luna Dong X., Naumann F.* Data fusion — resolving data conflicts in integration // Proc. VLDB Endowment, 2009. Vol. 2. Iss. 2. P. 1654–1655.

13. *Bleiholder J.* Data fusion and conflict resolution in integrated information systems. — Potsdam: Hasso-Plattner-Institut, 2010. D.Sc. Diss. 184 p.
14. *Winkler W. E.* Overview of record linkage and current research directions. Research report ser. (Statistics # 2006-2). — Washington, DC: Statistical Research Division, U.S. Census Bureau, 2006. http://www.census.gov/srd/papers/pdf/rrs2006-02.pdf.
15. *Adamic L. A., Adar E.* Friends and neighbors on the Web // Social networks, 2003. Vol. 25. No. 3. P. 211–230.
16. *Bilenko M., Mooney R., Cohen W., Ravikumar P., Fienberg S.* Adaptive name matching in information integration // IEEE Intell. Syst., 2003. Vol. 18. No. 5. P. 16–23.
17. Monge–Elkan distance function. http://www.gabormelli.com/RKB/Monge-Elkan_Distance_Function.
18. *Cochinwala M., Kurienb V., Lalka G., Shasha D.* Efficient data reconciliation // Inform. Sci. Int. J., 2001. Vol. 137. Iss. 1-4. P. 1–15.
19. *Bilenko M., Mooney R.* Adaptve duplicate detecton using learnable string similarity measures // KDD'03: 9th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining Proceedings, 2003. P. 39–48.
20. *Christen P.* Automatic record linkage using seeded nearest neighbour and support vector machine classification // KDD'08: 14th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining Proceedings, 2008. P. 151–159.
21. *Chen Z., Kalashnikov D. V., Mehrotra S.* Exploiting context analysis for combining multiple entity resolution systems // SIGMOD'09: 2009 ACM SIGMOD Conference (International) on Management of Data Proceedings, 2009. P. 207–218.
22. *Gupta R., Sarawagi S.* Answering table augmentaton queries from unstructured lists on the Web // Proc. VLDB Endowment, 2009. Vol. 2. Iss. 1. P. 289–300.
23. *Ravikumar P., Cohen W.* A hierarchical graphical model for record linkage // UAI'04: 20th Conference on Uncertainty in Artificial Intelligence Proceedings, 2004. P. 454–461.
24. *Tejada S., Knoblock C. A., Minton S.* Learning object identification rules for information integration // Inform. Syst. Data Extraction Cleaning Reconciliation, 2001. Vol. 26. Iss. 8. P. 607–633.
25. *Sarawagi S., Bhamidipaty A.* Interactive deduplication using active learning // KDD'02: 8th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining Proceedings, 2002. P. 269–278.
26. *Arasu A., Götz M., Kaushik R.* On active learning of record matching packages // SIGMOD'10: 2010 ACM SIGMOD Conference (International) on Management of Data Proceedings, 2010. P. 783–794.
27. *Bellare K., Iyengar S, Parameswaran A. G., Rastogi V.* Active sampling for entity matching // KDD'12: 18th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining Proceedings, 2012. P. 1131–1139.
28. *Adam K., Wu E., Karger D., Madden S., Miller R.* Human-powered sorts and joins // Proc. VLDB Endowment, 2011. Vol. 5. Iss. 1. P. 13–24.
29. *Wang J., Kraska T., Franklin M. J., Feng J.* CrowdER: Crowdsourcing Entity Resolution // Proc. VLDB Endowment, 2012. Vol. 5. Iss. 11. P. 1483–1494.
30. *Ananthakrishna R., Chaudhuri S., Ganti V.* Eliminating fuzzy duplicates in data warehouses // VLDB'02: 28th Conference (International) on Very Large Data Bases Proceedings, 2002. P. 586–597.
31. *Fan W., Geerts F., Jia X., Kementsietsidis A.* Conditional functional dependencies for Data cleaning // ICDE'07: 23rd IEEE Conference (International) on Data Engineering Proceedings, 2007. P. 746–755.
32. *Fan W.* Dependencies revisited for improving data quality // PODS'08: 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems Proceedings, 2008. P. 159–170.
33. *Benjelloun O., Garcia-Molina H., Menestrina D., Su Q., Whang S. E., Widom J.* Swoosh: A generic approach to Entity Resolution // VLDB Int. J., 2009. Vol. 18. Iss. 1. P. 255–276.
34. *Bhattacharya I., Getoor L.* Collective Entity Resolution in relational data // ACM Transactions on Knowledge Discovery from Data (TKDD), 2007. Vol. 1. Iss. 1. Article No. 5. doi: 10.1145/1217299.1217304.
35. *Bhattacharya I., Getoor L.* A latent Dirichlet model for unsupervised Entity Resolution // 6th SIAM Conference (International) on Data Mining Proceedings, 2007. P. 47–58.
36. *Broecheler M., Getoor L.* Probabilistic similarity logic // UAI'10: 26th Conference on Uncertainty in Artificial Intelligence Proceedings, 2010. P. 73–82.
37. *Rajaraman A., Ullman J. D.* Integrating information by outerjoins and full disjunctions // PODS'96: 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems Proceedings, 1996. P. 238–248.
38. *Dean J., Ghemawat S.* MapReduce: Simplified data processing on large clusters // Comm. ACM, 2008. Vol. 51. Iss. 1. P. 107–113.
39. MapReduce Tutorial. http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html.
40. Apache Pig Project. http://pig.apache.org.
41. The Apache Hive data warehouse. http://hive.apache.org.
42. IBM InfoSphere BigInsights Version 3.0, Jaql reference. — 2014. http://www-01.ibm.com/support/knowledgecenter/SSPT3X_3.0.0/com.ibm.swg.im.infosphere.biginsights.jaql.doc/doc/c 0057749.html.
43. *Das Sarma A., Jain A., Machanavajjhala A., Bohannon P.* An automatic blocking mechanism for large-scale deduplication tasks // CIKM'12: 21st ACM Conference (International) on Information and Knowledge Management Proceedings, 2012. P. 1055–1064.
44. *Papadakis G., Ioannou E., Niederée C., Palpanas T., Nejdl W.* Beyond 100 million entities: Large-scale blocking-based resolution for heterogeneous data //

WSDM'12: 5th ACM Conference (International) on Web Search and Data Mining Proceedings, 2012. P. 53–62.

45. *Kolb L., Thor A., Rahm E.* Dedoop: Efficient deduplication with Hadoop // Proceedings of the VLDB Endowment, 2012. Vol. 5. Iss. 12. P. 1878–1881.

46. *Hernández M., Koutrika G., Krishnamurthy R., Popa L., Wisnesky R.* HIL: A high-level scripting language for entity integration // EDBT'13: 16th Conference (International) on Extending Database Technology Proceedings, 2013. P. 549–560.

Поступила в редакцию 09.11.14

METHODS OF ENTITY RESOLUTION AND DATA FUSION IN THE ETL-PROCESS AND THEIR IMPLEMENTATION IN THE HADOOP ENVIRONMENT

A. E. Vovchenko, L. A. Kalinichenko, and L. Yu. Kovalev

[1]Institute of Informatics Problems, Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation

[2]Faculty of Computational Mathematics and Cybernetics, M. V. Lomonosov Moscow State University, 1-52 Leninskiye Gory, GSP-1, Moscow 119991, Russian Federation

Abstract: Entities extraction, their transformation and loading in the integrated repository are the main problem of data integration. These actions are part of the ETL-process (extract–transform–loading). An entity is a digital representation of a real world object (for example, information about a person). Entity resolution takes care of duplicate detection, deduplication, record linkage, object identification, reference matching, and other ETL-related tasks. After the entity resolution step, entities should be merged into the one reference entity (containing information from all related entities). Data fusion is the final step in the data integration process. The paper gives an overview of the entity resolution and data fusion methods. Also, the paper presents the techniques for programming the entity resolution and data fusion methods for implementing the ETL-process in the Hadoop environment. High-Level Integration Language (HIL), a declarative language that focuses on resolution and fusion of entities in the Hadoop-infrastructure, is used in this part of the paper.

Keywords: data integration; ETL; entity resolution; data fusion; big data; Hadoop; Jaql; HIL

DOI: 10.14357/19922264140412

Acknowledgments

This work was supported by the Russian Foundation for Basic Research (projects 13-07-00579 and 14-07-00548), Institute of informatics Problems of the Russian Academy of Sciences (IPI RAN) (theme 38.25 "Specification and problem solving of data analysis in conceptual terms of subject areas with intensive use of data" of the state task for IPI RAN), and the Presidium of the Russian Academy of Sciences (Basic Research Program No. 16 "Fundamental problems of system programming").

References

1. White, T. 2012. Hadoop: The definitive guide. 3rd ed. O'Reilly Media. 688 p.
2. Apache Hadoop 2.5.1. Available at: http://hadoop.apache.org/ (accessed November 01, 2014).
3. Naumann, F., and M. Herschel. 2010. *An introduction to duplicate detection*. Synthesis lectures on data management. Morgan & Claypool. Lecture No. 3. 87 p.
4. Christen, P. 2012. *Data matching — concepts and techniques for record linkage, entity resolution, and duplicate detection*. Data-centric systems and applications ser. Springer. 272 p.
5. Fan, W., and F. Geerts. 2012. *Foundations of data quality management*. Synthesis lectures on data management. Morgan & Claypool. Lecture No. 29. 217 p.
6. Bleiholder, J., and F. Naumann. 2009. Data fusion. *ACM Computing Surveys (CSUR)* 41(1). Article No. 1. doi: 10.1145/1456650.1456651.
7. Köpcke, H., A. Thor, and E. Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endowment* 3(1-2):484–493.
8. Köpcke, H., and E. Rahm. 2010. Frameworks for entity matching: A comparison. *Data Knowledge Engineering* 69(2):197–210. doi: 10.1016/j.datak.2009.10.003.

9. Ganti, V., and A. Das Sarma. 2013. Data cleaning: A practical perspective. Synthesis lectures on data management. Morgan & Claypool. Lecture No. 36. 85 p.
10. Getoor, L., and A. Machanavajjhala. 2013. Entity resolution for big data. *19th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining (KDD'13) Proceedings*. Chicago. 1527–1527.
11. Bleiholder, J., and F. Naumann. 2005. Declarative data fusion — syntax, semantics, and implementation. *East European Conference on Advances in Databases and Information Systems (ADBIS) Proceedings*. Tallinn. 58–73.
12. Dong, L. X., and F. Naumann. 2009. Data fusion — resolving data conflicts in Integration. *Proc. VLDB Endowment* 2(2):1654–1655.
13. Bleiholder, J. 2010. Data fusion and conflict resolution in integrated information systems. Potsdam. D.Sc. Diss. 184 p.
14. Winkler, W. E. 2006. Overview of record linkage and current research directions. Research report ser. No. 2006-2. Washington, DC: Statistical Research Division, U.S. Census Bureau. 44 p. Available at: http:// www.census.gov/srd/papers/pdf/rrs2006-02.pdf (accessed November 01, 2014).
15. Adamic, L. A., and E. Adar. 2003. Friends and neighbors on the Web. *Social Networks* 25:211–230.
16. Bilenko, M., R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. 2003. Adaptive name matching in information integration. *IEEE Intell. Syst.* 18(5):16–23.
17. Monge–Elkan distance function. Available at: http:// www.gabormelli.com/RKB/Monge-Elkan_Distance_ Function (accessed November 01, 2014).
18. Cochinwala, M., V. Kurienb, G. Lalka, and D. Shasha. 2001. Efficient data reconciliation. *Inform. Sci. Int. J.* 137(1-4):1–15.
19. Bilenko, M., and R. Mooney. 2003. Adaptve duplicate detecton using learnable string similarity measures. *9th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining (SIGKDD 2003) Proceedings*. Washington. 39–48.
20. Christen, P. 2008. Automatic record linkage using seeded nearest neighbour and support vector machine classification. *14th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining (KDD'2008) Proceedings*. Las Vegas. 151–159.
21. Chen, Z., D. V. Kalashnikov, and S. Mehrotra. 2009. Exploiting context analysis for combining multiple entity resolution systems. *2009 ACM SIGMOD Conference (International) on Management of Bata (SIGMOD 2009) Proceedings*. Providence. 207–218.
22. Gupta, R., and S. Sarawagi. 2009. Answering table augmentaton queries from unstructured lists on the Web. *Proc. VLDB Endowment* 2(1):289–300.
23. Ravikumar, P., and W. Cohen. 2004. A hierarchical graphical model for record linkage. *20th Conference on Uncertainty in Artificial Intelligence (UAI 2004) Proceedings*. Virginia. 454–461.
24. Tejada, S., C. A. Knoblock, and S. Minton. 2001. Learning object identification rules for information integration. *Inform. Syst. Data Extraction Cleaning Reconciliation* 26(8):607–633.
25. Sarawagi, S., and A. Bhamidipaty. 2002. Interactive deduplication using active learning. *8th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining (KDD 2002) Proceedings*. Edmonton. 269–278.
26. Arasu, A., M. Götz, and R. Kaushik. 2010. On active learning of record matching packages. *2010 ACM SIGMOD Conference (International) on Management of Data Proceedings*. Indianapolis. 783–794.
27. Bellare, K., S. Iyengar, A. G. Parameswaran, and V. Rastogi. 2012. Active sampling for entity matching. *18th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining (KDD 2012) Proceedings*. Beijing. 1131–1139.
28. Adam, K., E. Wu, D. Karger, S. Madden, and R. Miller. 2011. Human-powered sorts and joins. *Proc. VLDB Endowment* 5(1):13–24.
29. Wang, J., T. Kraska, M. J. Franklin, and J. Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *Proc. VLDB Endowment* 5(11):1483–1494.
30. Ananthakrishna, R., S. Chaudhuri, and V. Ganti. 2002. Eliminating fuzzy duplicates in data warehouses. *28th Conference (International) on Very Large Data Bases (VLDB 2002) Proceedings*. Hong Kong. 586–597.
31. Fan, W., F. Geerts, X. Jia, and A. Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. *2007 IEEE 23rd Conference (International) on Data Engineering Proceeding*. Istanbul. 746–755.
32. Fan, W. 2008. Dependencies revisited for improving data quality. *27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2008) Proceedings*. Vancouver. 159–170.
33. Benjelloun, O., H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. 2009. Swoosh: A generic approach to Entity Resolution. *VLDB Int. J.* 18(1):255–276.
34. Bhattacharya, I., and L. Getoor. 2007. Collective Entity Resolution in relational data. *ACM Trans. Knowledge Discovery Data (TKDD)* 1(1). Article No. 5. doi: 10.1145/1217299.1217304.
35. Bhattacharya, I., and L. Getoor. 2007. A latent Dirichlet model for unsupervised Entity Resolution. *6th SIAM Conference (International) on Data Mining Proceedings*. Maryland. 47–58.
36. Broecheler, M., and L. Getoor. 2010. Probabilistic similarity logic. *26th Conference on Uncertainty in Artificial Intelligence Proceedings*. Corvallis. 73–82.
37. Rajaraman, A., and J. D. Ullman. 1996. Integrating information by outerjoins and full disjunctions. *15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS1996) Proceedings*. Montreal. 238–248.
38. Dean, J., and S. Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Comm. ACM* 51(1):107–113.

39. MapReduce tutorial. Available at: http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html (accessed November 01, 2014).
40. Apache Pig Project. Available at: http://pig.apache.org/ (accessed November 01, 2014).
41. The Apache Hive data warehouse. Available at: http://hive.apache.org/ (accessed November 01, 2014).
42. IBM InfoSphere BigInsights Version 3.0, Jaql reference. Available at: http://www-01.ibm.com/support/knowledgecenter/SSPT3X_3.0.0/com.ibm.swg.im.infosphere.biginsights.jaql.doc/doc/c0057749.html (accessed November 01, 2014).
43. Sarma, D.A., A. Jain, A. Machanavajjhala, and P. Bohannon. 2012. An automatic blocking mechanism for large-scale de-duplication tasks. *21st ACM Conference (International) on Information and Knowledge Management Proceedings*. Maui. 1055–1064.
44. Papadakis, G., E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. 2012. Beyond 100 million entities: Large-scale blocking-based resolution for heterogenous data. *5th ACM Conference (International) on Web Search and Data Mining Proceedings*. Seattle. 53–62.
45. Kolb, L., A. Thor, and E. Rahm. 2012. Dedoop: Efficient deduplication with Hadoop. *Proceedings of the VLDB Endowment* 5(12):1878–1881.
46. Hernández, M., G. Koutrika, R. Krishnamurthy, L. Popa, and R. Wisnesky. 2013. HIL: A high-level scripting language for entity integration. *16th Conference (International) on Extending Database Technology (EDBT'13) Proceedings*. Genoa. 549–560.

Received November 9, 2014

Contributors

Vovchenko Alexey E. (b. 1984) — Candidate of Science (PhD) in technology, senior researcher, Institute of Informatics Problems, Russian Academy of Sciences; 44-2 Vavilov Str., Moscow 119333, Russian Federation; alexey.vovchenko@gmail.com

Kalinichenko Leonid A. (b. 1937) — Doctor of Science in physics and mathematics, professor; Head of Laboratory, Institute of Informatics Problems, Russian Academy of Sciences; 44-2 Vavilov Str., Moscow 119333, Russian Federation; professor, Faculty of Computational Mathematics and Cybernetics, M.V. Lomonosov Moscow State University, 1-52 Leninskiye Gory, GSP-1, Moscow 119991, Russian Federation; leonidk@synth.ipi.ac.ru

Kovalev Dmitry Yu. (b. 1988) — junior researcher, Institute of Informatics Problems, Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russian Federation; dm.kovalev@gmail.com