УДК 004.62

И.П. Карпова, канд. техн. наук (Московский институт электроники и математики Национального исследовательского университета «Высшая школа экономики»);
e-mail: karpova_ip@mail.ru

О РЕАЛИЗАЦИИ МЕТОДА ОБЕЗЛИЧИВАНИЯ ПЕРСОНАЛЬНЫХ ДАННЫХ

Рассмотрен метод обезличивания персональных данных на основе перемешивания данных. Предложены подходы к реализации этого метода при хранении данных в реляционной базе данных. Обоснованы технические решения, обеспечивающие защищенность данных от де-обезличивания и высокую скорость их обработки.

The method of a depersonalization of personal information on the basis of hashing of data is considered. Approaches to realization of this method are offered at data storage in a relational database. The technical solutions providing security of data from de depersonalization and high speed of data processing locate.

Ключевые слова: персональные данные; обезличивание персональных данных; перемешивание данных.

Keywords: Personal information; Depersonalization of personal information; Hashing of data.

Введение

Необходимость обеспечения защиты *персональных данных* (*ПД*) обусловлена в первую очередь требованиями Федерального закона о персональных данных [1]. Защищать персональные данные можно с помощью шифрования (криптографическая защита). Но применение шифрования очень дорого, поэтому большинство организаций (особенно бюджетных), занимающихся обработкой персональных данных, не могут себе позволить подобные расходы. С этой точки зрения более перспективным представляется использование различных процедур обезличивания персональных данных, применение которых не требует покупки дорогостоящей лицензии, как в случае криптографической защиты.

Под обезличиванием персональных данных в соответствии с законом [1] будем понимать действия, в результате которых по обработанным данным невозможно определить принадлежность персональных данных конкретному субъекту.

В качестве основных подходов к обезличиванию данных можно выделить следующие методы обезличивания персональных данных [2]:

– уменьшение перечня обрабатываемых сведений;
– замена части сведений идентификатором (-ами);
– замена численных значений минимальным, средним или максимальным значениями;
– понижение точности некоторых сведений;
– деление сведений на части и использование перекрестных ссылок.

Существуют и менее распространенные способы, например, использование внешнего ключа на физическом носителе для доступа к базе персональных данных [3].

Очевидно, что некоторые из этих подходов могут применяться только для статистической обработки. Использование перекрестных ссылок и замена части сведений идентификаторами уязвимы с точки зрения возможности восстановления исходных данных (*де-обезличивания*) при получении доступа ко всем частям данных. Применение внешнего носителя делает персональные данные недоступными для обработки в отсутствие владельца этих данных, что для большинства ситуаций неприемлемо.

Поэтому остается актуальным вопрос о методах обезличивания персональных данных, которые свободны от указанных недостатков и не требуют использования сертифицированных средств защиты.

Метод обезличивания персональных данных на основе перемешивания

В работе [4] предложен метод обезличивания персональных данных, основанный на их перемешивании. Суть метода заключается в следующем. Значения каждого атрибута нумеруются и разбиваются на подмножества. Внутри каждого i-го подмножества данные перемешиваются путем циклической перестановки (сдвига) на r_i позиций. Затем подмножества каждого j-го атрибута также перемешиваются путем циклического сдвига на C_j. Таким образом, для осуществления перемешивания достаточно знать пара-

метры перемешивания для каждого атрибута: $C_i(n_{i1}, r_{i1}; n_{i2}, r_{i2}; \ldots; n_{ik}, r_{ik})$, где C_i – величина сдвига подмножеств i-го атрибута; n_{ij} – начальный номер значения i-го атрибута, относящегося к j-му подмножеству; r_{ij} – величина сдвига внутри j-го подмножества i-го атрибута; k – количество подмножеств. Например, разобьем атрибуты данных на подмножества так, как это показано в табл. 1 исходных данных. (Каждое подмножество выделено шрифтом – обычный, курсив, полужирный.)

Для параметров перемешивания C (1(4, **–1**; 5, **3**; 3, **1**); –1(3, **1**; 5, **2**; 4, **2**); …; 2(5, **3**; 4, **–1**; 3, **2**)) будем иметь после обработки табл. 2. Запись 1(4, **–1**; 5, **3**; 3, **1**) относится к первому атрибуту (атрибуту *A*) и означает, что в первом подмножестве этого атрибута значения сдвигаются на **–1**, во втором – на **3**, в третьем – на **1**. На втором этапе все подмножества атрибута *A* сдвигаются на **1**. Из табл. 1 и 2 видно, что после перемешивания каждая запись, определяемая значением *идентификатора id*, содержит измененные данные.

Зная параметры перемешивания, из табл. 2 можно восстановить исходные данные, применяя перестановки в обратном порядке.

К достоинствам данного метода можно отнести простоту процедуры обезличивания данных и невозможность восстановления исходных данных без знания параметров перемешивания.

Сложности, возникающие при реализации данного метода, обусловлены двумя основными моментами. *Во-первых*, в реальных системах объем хранимых данных достигает $10^6 \ldots 10^8$ записей. Поэтому необходимо особое внимание уделять вопросам эффективности обработки данных. *Во-вторых*, надо учитывать, что данные не являются статичными: появляются новые записи, удаляются существующие.

1. Исходные данные

id	*A*	*B*	...	*Z*
1	*a1*	**b1**		*z1*
2	*a2*	**b2**		*z2*
3	*a3*	**b3**		*z3*
4	*a4*	b4		*z4*
5	a5	b5		*z5*
6	a6	b6		z6
7	a7	b7	…	z7
8	a8	b8		z8
9	a9	*b9*		z9
10	**a10**	*b10*		**z10**
11	**a11**	*b11*		**z11**
12	**a12**	*b12*		**z12**

2. Пример перемешивания исходных данных

id	*A*	*B*	...	*Z*
1	**a12**	b7		z7
2	**a10**	b8		z8
3	**a11**	b4		z9
4	*a2*	b5		z6
5	*a3*	b6		**z11**
6	*a4*	*b11*		**z12**
7	*a1*	*b12*	…	**z10**
8	a7	*b9*		*z3*
9	a8	*b10*		*z4*
10	a9	**b3**		*z5*
11	a5	**b1**		*z1*
12	a6	**b2**		*z2*

Хранение данных

Наиболее распространенной в настоящее время является *реляционная модель данных*, поэтому будем рассматривать реализацию данного метода средствами реляционной *системы управления базами данных* (*СУБД*).

Для каждой исходной таблицы, содержащей персональные данные, необходимо определить, какие данные (атрибуты) требуют перемешивания, а какие можно хранить в первоначальном виде. Не имеет смысла перемешивать все атрибуты: это не только требует времени, но и не является оправданным с точки зрения защиты данных. Данные, не позволяющие однозначно идентифицировать человека, не представляют интереса без идентификационной информации (разве только для статистической обработки). Таким образом, перемешиванию стоит подвергать только идентифицирующие атрибуты, но не описательные.

Состав идентифицирующих атрибутов и их количество N зависит от предметной области. Назовем процедуру перемешивания исходных данных *прямым преобразованием*, а процедуру восстановления исходных данных – *обратным преобразованием*.

Формирование таблицы с обезличенными данными

Постановка задачи: из таблицы с исходными (персональными) данными необходимо получить таблицу с обезличенными (деперсонифицированными) данными. В реляционной модели данных нет понятия «номер строки таблицы», и сдвиг на M позиций, как в массиве, невозможен. Поэтому для реализации процедуры перемешивания необходимо перенумеровать строки в таблице, т.е. добавить в таблицу суррогатный первичный ключ, который будет имитировать номер

строки. Для корректной работы алгоритма перемешивания строки должны быть перенумерованы без пропусков.

Для формирования таблицы с обезличенными данными необходимо применять обратное преобразование для каждого атрибута, подлежащего перемешиванию, в результате по номеру строки j получим номер той строки j^*, из которой нужно брать значение i-го атрибута, которое будет храниться в j-й строке после перемешивания данных.

Предлагаемая последовательность обратного преобразования для i-го атрибута j-й строки такова:

1) $j' = j - x_i$,

где x_i – сумма мощностей C_i последних подмножеств i-го атрибута;

C_i – величина сдвига подмножеств для i-го атрибута (например, для атрибута z и $C_z = 2$ в табл. 2 это будет 3 + 4 = 7);

2) определить n – номер подмножества, к которому относится строка с номером j' (например, для атрибута z и $j' = 5$ в табл. 2 $n = 2$);

3) $j^* = j' - y_{in}$,

где y_{in} – величина сдвига в n-м подмножестве i-го атрибута (например, для атрибута z в первом подмножестве табл. 2 сдвиг равен 3).

Приведем пример алгоритма процедуры формирования таблицы с обезличенными данными:

```
Цикл по строкам исходной таблицы
    Считать очередную строку, получить id
    строки и значения неперемешанных ат-
    рибутов.
    Цикл по i от 1 до N
       Для i-го атрибута путем обратного
       преобразования вычислить id' стро-
       ки, из которой будет взято значе-
       ние i-го атрибута для текущей
       строки.
       Считать из исходной таблицы строку
       с id' и получить значение i-го ат-
       рибута.
    к.ц. по i.
    Добавить сформированную строку в таб-
    лицу с обезличенными данными.
к.ц. по строкам исходной таблицы.
```

Здесь `к.ц.` – конец цикла.

Оценим вычислительную сложность этой процедуры. В таблице с исходными (персональными) данными присутствует индекс по атрибуту *id*, так как это первичный ключ. Для каждой строки таблицы с исходными (персональными) данными выполняется $(N + 1)$ операция чтения данных этой таблицы через индекс и одна операция добавления строки в таблицу с перемешанными (деперсонифицированными) данными. Таким образом, получим следующую оценку времени, необходимого для создания таблицы с обезличенными данными:

$$O_1 = U(M) + S(M) + S_i(MN) + I(M), \quad (1)$$

где M – число строк в исходной таблице;

N – число перемешиваемых атрибутов;

$U(M)$ – время на проставление значений *id* для M строк (с помощью команды `update`);

$S(M)$ – время на чтение M строк из таблицы (команда `select`);

$S_i(MN)$ – время на чтение MN строк из таблицы через индекс;

$I(M)$ – время на добавление M строк в таблицу.

После создания таблицы с обезличенными данными исходную таблицу необходимо удалить из базы данных.

Поиск данных

Организация поиска данных в таблице с обезличенными данными зависит от условий селекции. Поиск в базе данных в основном будет происходить по значению основного идентифицирующего атрибута, который определяется предметной областью. Например, в поликлинике – по номеру полиса, в налоговой инспекции – по ИНН и т.д. Но нельзя исключать и ситуации, когда потребуется поиск по значению произвольного атрибута, а не по идентификатору. Например, пришел человек в поликлинику, полис забыл дома, но может предъявить свой паспорт.

С формальной точки зрения можно выделить следующие ситуации поиска данных по значениям:

– уникального атрибута;
– неуникального атрибута;
– комбинаций значений неуникальных атрибутов.

Алгоритм поиска *по значению уникального атрибута*:

1) найти строку, в которой хранится искомое уникальное значение атрибута, считать *id* этой строки (значение суррогатного первичного ключа);

2) применить обратное преобразование и вычислить истинный *id*, т.е. значение *id*, соответствующее искомому значению атрибута;

3) считать из исходной таблицы строку с истинным *id*, извлечь из нее значения атрибутов, не подвергавшиеся перемешиванию;

4) применить к значению истинного *id* прямые преобразования для поиска оставшихся $(N - 1)$ перемешанных атрибутов, считать из таблицы $(N - 1)$ строку для извлечения значений этих атрибутов (если это нужно по условиям поиска).

Поиск значения уникального атрибута и *id* будет выполняться через индекс, так как для уникальных атрибутов СУБД автоматически создает индексы. Таким образом, при поиске по уникальному значению происходит $(N + 2)$ обращения к исходной таблице, оценка времени выполнения запроса:

$$O_2 = S_i(N + 2). \quad (2)$$

Поиск *по значению неуникального атрибута* потребует больше времени, но принципиально не отличается от поиска по значению уникального атрибута. Поиск по неуникальному атрибуту вернет на первом шаге K строк ($K \geq 0$) и все последующие шаги надо будет проводить для каждой из найденных строк. Оценка времени выполнения поискового запроса:

$$O_3 = S(M) + S_i(KN), \quad (3)$$

где $S(M)$ – полное чтение таблицы для поиска нужных значений неуникального атрибута;

$S_i(KN)$ – обращение к таблице через индекс для де-обезличивания данных в каждой из K найденных строк.

Для неуникальных атрибутов также можно создать индексы для ускорения поиска. Но в этом случае надо ориентироваться на предметную область: если значение атрибута часто меняется, то увеличение времени на внесение изменений в индекс может превысить выигрыш от ускорения поиска. При наличии индекса оценка времени выполнения будет другой:

$$O_4 = S_i(K(N+1)). \quad (4)$$

Поиск *по комбинации значений неуникальных атрибутов* может при реализации «в лоб» потребовать слишком много времени. Поэтому здесь предлагается использовать другой алгоритм:

1) выбрать самый высокоселективный атрибут;

2) по этому атрибуту найти и восстановить все строки, удовлетворяющие условию поиска для данного атрибута;

3) проверить для восстановленных строк остальные условия.

Если совместить восстановление строк и проверку условий, то можно дополнительно сократить время выполнения поиска.

Селективность каждого атрибута определяется процентом строк, удовлетворяющих условию поиска: чем выше этот процент, тем ниже селективность. Оценка селективности атрибута может проводиться средствами используемой СУБД (если это входит в состав ее функциональных возможностей), либо путем выполнения запроса типа:

```
select 100*count(*)/M from <таблица>
  where <условие на значение атрибута
  X>;
```

Удаление и добавление данных

При перемешивании атрибутов в расчетах используется количество записей в исходной таблице (для определения выхода за границы номеров строк таблицы при циклическом сдвиге). Следовательно, изменение этого количества, которое происходит при добавлении/удалении записей, потребует пересчета всех сдвигов в таблице, т.е. выполнения команды `update` на все записи. Если выполнять каждую такую операцию сразу, как только возникло соответствующее изменение в предметной области, то база данных бóльшую часть времени будет недоступна. Более целесообразно проводить добавление записей порциями и/или в ночное время, когда интенсивность обращения к данным минимальна.

Общая стратегия при добавлении данных такова: взять очередную порцию данных, объявить ее новым подмножеством и добавить это подмножество к существующей таблице. Добавление вызовет пересчет смещения всех элементов (значений атрибутов), т.е. реорганизацию таблицы.

Удалять данные из таблицы физически нельзя, так как это приведет к разрывам в нумерации строк, и алгоритм обезличивания перестанет работать. Поэтому удаление может быть только логическим: запись остается в таблице, но помечается как удаленная. Интерпретация данной ситуации зависит от предметной области, например, обладатель персональных данных перестал входить в число людей, чьи данные подлежат хранению и обработке в системе.

Если использовать только логическое удаление, то надо иметь в виду: если число логически удаленных данных станет слишком большим, то упадет эффективность обработки. В этом случае потребуется периодическая реорганизация таблицы с целью физического удаления логически удаленных записей. Физическое удаление не обязательно означает утрату старых данных: это может быть перенос удаленных записей в архивную таблицу, которая также хранит обезличенные данные.

Реорганизация таблицы подразумевает полную ее перестройку с целью добавления новых записей или физического удаления старых. Это можно провести путем восстановления исходной таблицы по таблице с обезличенными данными, добавления в нее новых записей и формирования новой таблицы с обезличенными данными. В процессе реорганизации можно изменить параметры перемешивания, что повысит защищенность системы от подбора параметров. Исходная таблица является временной и после завершения операции реорганизации удаляется.

Реорганизация таблицы займет много времени, оценить которое можно следующим образом:

$$O_5 = S(M) + S_i(MN) + I(M) + D(K_1) + I(K_2) + U(M') + O_1(M'), \quad (5)$$

где первые три слагаемых соответствуют формуле (1);

$D(K_1)$ – время на удаление K_1 записей ($K_1 < M$);

$I(K_2)$ – время на добавление K_2 записей;

$U(M')$ – время на проставление новых значений *id*;

$M' = M - K_1 + K_2$ – количество данных в новой таблице;

$O_1(M')$ – время на создание новой таблицы с обезличенными данными.

Если данные удаляются/добавляются в небольшом объеме ($K_1 << M$, $K_2 << M$), то приблизительная оценка вычислительной сложности

$$O_6 = 2S(M(N+1)) + 2I(M) + 2U(M). \quad (6)$$

В этой ситуации более рациональным представляется другой подход. Удаление данных происходит только логически, а новые данные образуют новое подмножество, которое хранится в отдельной области памяти (Partition), и значения данных в этом подмножестве перемешиваются на основе других параметров как отдельная таблица. Логически таблица остается единой, а при поиске данных добавляется одна проверка, которая по значению *id* определяет, в новой области хранятся данные или в старой. Таким образом, реорганизация будет затрагивать уже не всю таблицу, а только область добавляемых данных. И тогда с определенной периодичностью можно осуществлять слияние двух частей, причем периодичность зависит от интенсивности добавления данных.

Оценка защищенности данных от де-обезличивания

В статье [4] приводятся оценки высокой степени защищенности обезличенных данных от попыток подбора параметров перемешивания и от атак, использующих внешние данные, имеющие наборы атрибутов и совпадающие с некоторыми атрибутами в исходной таблице. Наличие набора записей из исходной таблицы не позволяет провести процедуру де-обезличивания для других записей из таблицы обезличенных данных.

Покажем практический аспект обеспечения защиты данных. Особенность рассматриваемого метода обезличивания данных – его ориентация на номера строк (значения идентификатора *id*). Если возрастание значений идентификатора совпадет с возрастанием значений одного из уникальных атрибутов, можно провести независимое упорядочение этих значений и сопоставить эти два списка друг с другом. Тогда, зная значение уникального атрибута, можно определить принадлежность неперемешанных атрибутов конкретному лицу.

Исходя из вышеизложенного, нельзя допускать, чтобы порядок назначения значений суррогатного ключа совпадал с упорядоченностью какого-либо уникального ключа исходной таблицы.

Заключение

Подход к реализации метода обезличивания персональных данных на основе перемешивания данных, способы организации хранения данных и алгоритмы их обработки позволяют реализовать данный метод и могут обеспечить защиту данных от де-обезличивания и достаточно высокую эффективность работы с персональными данными.

Следует отметить, что остается открытым вопрос о защите параметров перемешивания. Естественно, что доступ к этим параметрам должен быть запрещен для любых категорий пользователей. Они должны назначаться автоматически и храниться в зашифрованном виде.

Очевидно, что от характеристик параметров зависит вычислительная сложность данного метода. В процессе реорганизации таблицы временно восстанавливается исходная таблица, что создает потенциальную угрозу безопасности персональных данных. В связи с этим в дальнейшем актуальной является разработка метода (алгоритма) реорганизации таблицы без восстановления исходных данных даже на ограниченный период времени.

Библиографический список

1. **О персональных** данных: федер. закон РФ от 27 июля 2006 г. № 152-ФЗ. 2-е изд. М.: Ось-89, 2008. 32 с.

2. **McCallister E., Grance T., Scarfone K.** Guide to Protecting the Confidentially of Personally Identifiable Information (PII): Recommendations of the National Institute of Standard and Technology (NIST) [Электронный ресурс] // U.S. Department of Commerce. 2010. URL: http://csrc.nist.gov/publications/nistpubs/800-122/sp800-122.pdf (дата обращения 20.02.2013).

3. **Уникальная** методика обезличивания персональных данных: предложения на рынке [Электронный ресурс] URL: http://secandsafe.ru/stati/predlojeniya_na_rynke/unikalnaya_metodika_obezlichivaniya_personalnyh_dannyh (дата обращения 23.11.12).

4. **Саксонов Е.А., Шередин Р.В.** Процедура обезличивания персональных данных [Электронный ресурс] // Наука и образование: электрон. научн.-техн. изд. 2011. № 3. URL: http://technomag.edu.ru/doc/173146.html (дата обращения 20.02.2013).

Статья поступила в редакцию 04.12.2012 г.

ООО «Издательский дом «Спектр», 119048, Москва, ул. Усачева, д. 35, стр. 1. Http://www.idspektr.ru
Учредитель ООО «Издательский дом «Спектр», **E-mail: vkit@idspektr.ru; vkitpost@rambler.ru**
Интернет: Http://www.vkit.ru
Телефон редакции журнала: (495) 589-56-41, (495) 514-76-50
Корректор *А.И. Евсейчев.* Инженер по компьютерному макетированию *М.А. Евсейчева*.
Сдано в набор 00.00.11 г. Подписано в печать 00.00.11 г. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 0,00. Уч.-изд. л. 0,00. Заказ 00. Тираж 00 экз.
Оригинал-макет и электронная версия подготовлены в ООО «Издательский дом «Спектр».
Отпечатано в ОАО «Подольская фабрика офсетной печати», 142100, Московская область, г. Подольск, Революционный проспект, д. 80/42