УДК 004.932.2:681.3.06

В.А. Радченко, Ю.А. Мальков, С.А. Балюк, Ю.С. Горпиненко

Харьковский национальный университет радиоэлектроники, Харьков

СИНТЕЗ ЛОГИЧЕСКОЙ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ НА ОСНОВЕ ВЫЯВЛЕННОГО МНОЖЕСТВА ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ

В данной статье рассматривается способ проектирования логической схемы реляционной базы данных на основании функциональных зависимостей, полученных при анализе экземпляра реляционной БД. Применяя метод выявления функциональных зависимостей, становится возможным восстановить взаимосвязи данных, накладываемые предметной областью. Эти взаимосвязи представляют собой ограничения целостности в БД. Полученные зависимости являются входными данными для цепочки алгоритмов, реализующих метод синтеза логической схемы. Конечным результатом является логическая схема БД в третьей нормальной форме.

***Ключевые слова**: логическая схема, функциональная зависимость, предметная область, ограничение целостности, метод синтеза, третья нормальная форма.*

Введение

Постановка проблемы. Создание и внедрение в практику современных информационных систем и баз данных выдвигает новые задачи проектирования, которые невозможно решить традиционными приемами и методами. Большое внимание уделяется вопросам проектирования баз данных (БД). От того, насколько успешно будет спроектирована БД, зависит эффективность функционирования системы в целом, ее жизнеспособность и возможность расширения и дальнейшего развития. На сегодняшний день наиболее широкое распространение получили реляционные базы данных (РБД), которые обеспечивают наилучшее сочетание простоты, надежности и производительности для решения широкого класса задач хранения и обработки данных.

В процессе своего жизненного цикла даже “оптимально” спроектированная РБД неизбежно претерпевает модификации, связанные с непрерывными изменениями в предметной области (например, усовершенствование бизнес-процессов, различные требования к данным со стороны пользователей и т.п.). Поэтому одним из наиболее важных этапов жизненного цикла РБД является сопровождение, включающее в себя следующие аспекты:

- контроль производительности;
- модификация структуры РБД;
- изменение требований к РБД.

Ключевым компонентом, во многом определяющем характеристики функционирования РБД, является логическая схема, представляющая собой описание таблиц данных и связей между ними. Таким образом, выполнение задач сопровождения во многом невозможно без внесения изменений в логическую схему РБД. При этом необходимо сохранить взаимосвязи, накладываемые предметной областью. Для достижения этой цели можно использовать набор данных существующей РБД. Непосредственно метод заключается в выявлении множества функциональных зависимостей, описывающих логическую схему БД и построения новой схемы в третьей нормальной форме (3НФ), используя выявленные зависимости.

Анализ последних исследований и публикаций. Существующие подходы проектирования РБД основываются на использовании некоторого исходного множества функциональных зависимостей (ФЗ), сформированного на основании предположения о возможных взаимосвязях объектов реального мира. При этом существует большая вероятность того, что значимые связи будут упущены, что негативно повлияет (избыточность информации, неустранимые аномалии, противоречия) на дальнейшее проектирование БД и ее эксплуатацию.

В то же время в литературе [1, 2, 3] большое внимание уделяется методам выявления ФЗ из существующих экземпляров отношений РБД, что позволяет обнаружить скрытые взаимосвязи между данными, которые, в свою очередь, могут быть использованы для построения логической структуры, максимально приближенной к той, что обуславливается рассматриваемой предметной областью. Следует заметить, что рассматриваемые методы направлены, в основном, на использование в системах интеллектуального анализа данных (Data Mining) и ориентированы на выявление приближенных функциональных зависимостей (Approximate Functional Dependencies — AFD), которые имеют некоторую погрешность. В рамках данной работы использование таких методов позволяет также получать множество строгих ФЗ, то есть справедливых для всего набора входных данных на момент времени проведения обработки.

Наиболее широко известный и распространенный метод нормализации, используемый для проектирования логической схемы, позволяет перейти от первоначального отношения к набору отношений в 3НФ путем применения ряда правил декомпозиции. Ф. Бернштейн выдвинул альтернативную теорию: так как функциональные зависимости полностью определяют, находится ли отношение в 3НФ, то очевидно является возможным построить 3НФ на основании набора функциональных зависимостей, удовлетворяющих исходному отношению [6]. В работе [7] обоснован метод, позволяющий синтезировать логическую схему РБД. Автор предложил набор алгоритмов для синтеза логической схемы в 3НФ [8], которые будут описаны ниже.

Постановка задачи. В работе рассматривается процесс проектирования реляционных БД с применением методов синтеза для построения оптимальной логической схемы. Предлагаемый подход основывается на методе выявления множества минимальных функциональных зависимостей в текущем наборе данных некоторой РБД. Полученное множество функциональных зависимостей используется в качестве входных данных для метода синтеза схемы БД. Результатом является отношение РБД в третьей нормальной форме. Описанный способ является вариантом автоматизированного решения для проектирования логической структуры РБД, непосредственно ориентированного на использование зависимостей в данных, порождаемых предметной областью.

Выявление функциональных зависимостей

Для большинства существующих методов базовыми являются две концепции – Tane [4] и FastFD [5]. Ключевым понятием данных методов является так называемая решетка атрибутов (set containment lattice, [4]).

Примем следующие условные обозначения:

R – множество атрибутов, входящих в отношение РБД;

r – экземпляр отношения (множество записей конкретной БД);

X – некоторое подмножество R.

Решетка атрибутов представляет собой направленный граф G={V, A}, где V – множество вершин графа, A – множество направленных ребер. $V = P(R) \setminus \varnothing$, где P(R) – булеан R.

Пусть $L_l = \left\{ X \in V \mid |X| = l, l = \overline{1,|R|} \right\}$ – некоторый уровень (совокупность множеств атрибутов, имеющих одинаковое количество элементов) решетки, тогда $A = \left\{ (v_1, w_{l+1}) \right\}, l = \overline{1,|R|-1}$, где $v_1 = X \in L_l \mid v_1 \subset w_{l+1}, l = \overline{1,|R|-1}$ – начальная вершина ребра графа, $w_{l+1} = Y \in L_{l+1}, l = \overline{1,|R|-1}$ – конечная вершина.

Приведем пример построения графа решетки для отношения R, содержащего атрибуты A, B, C. Начальными элементами графа являются одиночные атрибуты множества R, образующие первый уровень решетки: $L_1 = \{A,B,C\}$.

Следующие уровни представляют собой множества сочетаний элементов предыдущего: $L_2 = \{\{A,B\},\{B,C\},\{A,C\}\}$, $L_3 = \{\{A,B,C\}\}$, причем мощность каждого элемента уровня равна индексу этого уровня. Ребра графа строятся по следующему принципу: начальной вершиной ребра является элемент текущего уровня, конечной – элементы следующего уровня, являющиеся надмножеством начальной вершины ребра. После построения ребер от предпоследнего уровня к последнему процесс заканчивается. Сформированный граф будет иметь следующий вид (рис. 1):

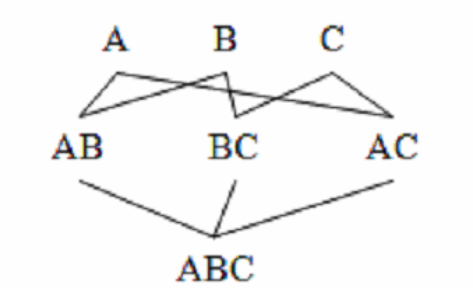

Рис. 1. Решетка для трех атрибутов

Основное различие методов выявления ФЗ – направление обхода представленной решетки атрибутов: Tane осуществляет проверку зависимостей уровень за уровнем, что гарантирует нахождение минимальных ФЗ (минимальная ФЗ – такая ФЗ, в которой никакое подмножество ее определяющей части не может сформировать новую ФЗ). FastFD, в свою очередь, проходит все уровни для каждого отдельного атрибута. Было принято решение реализовать метод Tane, поскольку на данный момент особенностями различных методов и скоростью выполнения можно пренебречь (важен результат работы реализации, который служит входными данными для метода синтеза).

Рассмотрим понятие класса эквивалентности в рамках данного метода. Пусть $t \in r$ – некоторая запись экземпляра отношения, тогда C(t) – класс эквивалентности, представляющий собой множество всех записей u, эквивалентных t: $C(t)=\{u \in r \mid u \sim t\}$. $C_X(t)=\{u \in \pi_X(R) \mid u \sim t\}$, где π_X – операция проекции реляционной алгебры по множеству атрибутов X. Операция объединения классов эквивалентности дает в результате исходный экземпляр отношения: $\bigcup C_i(t) = r$.

Аналогично: $\bigcup C_i(t \in \pi_X(R)) = \pi_X(R) = \rho_X$, где ρ_X – сегмент отношения (partition, [4]). Ранг сегмента $|\rho_X|$ – количество классов эквивалентности, которое он содержит. Также введем понятие сокращенного сегмента $\hat{\rho}$ (stripped partition), который является некоторым сегментом ρ с удаленными классами эквивалентности мощности, равной 1: $\hat{\rho} = \rho \setminus C(t) \,|\, |C(t)| = 1$. Отсюда можно вывести следствие: сокращенный сегмент, ранг которого равен 0, представляет собой ключ отношения, поскольку содержит в текущий момент только уникальные записи.

Каждый элемент, входящий в некоторый класс эквивалентности, удобно представить, связав с его порядковым номером в экземпляре отношения. Таким образом, становится возможным однозначно идентифицировать данную запись.

Например, для данных, приведенных в табл. 1 (атрибут «Фамилия» является ключевым), $\rho_{Фамилия}$ = = {{1}{2}{3}{4}}, а $\rho_{(Должность,Смена)}$ = {{1}{2,4}{3}}. Соответственно, $\hat{\rho}_{Фамилия} = \varnothing$, $\hat{\rho}_{(Должность,Смена)}$ = {{2,4}}.

Таблица 1

Пример данных экземпляра отношения

	Фамилия	Должность	Смена
1	Иванов	Инженер	1
2	Петров	Рабочий	2
3	Сидоров	Рабочий	1
4	Шолохов	Рабочий	2

Сегменты являются основой для проверки утверждения о корректности некоторой ФЗ: X→A соблюдается тогда, когда $|\rho_X| = |\rho_{(X \cup A)}|$. Например, для табл. 1, $\rho_{Фамилия}$ = 4. В то же время для X = Фамилия ∪ Должность $|\rho_X|$ также равно 4, поскольку наличие ключевого атрибута придает уникальность каждой записи в проекции по множеству X.

Следовательно, задача выявления ФЗ из некоторого экземпляра отношения сводится к проверке всех возможных вариантов зависимостей вида X→A для данного отношения, где $X \in V \setminus A$ (интерес представляют только минимальные нетривиальные зависимости).

Введем понятие ошибки e(X). Это величина, характеризующая количество записей, которые необходимо удалить из экземпляра отношения для того, чтобы строгая ФЗ X→A была соблюдена. Наличие данного параметра позволяет включать в рассмотрение нестрогие (приблизительные) зависимости, т.е., те, для которых e(X)>0. В этом заключается специфика этого и других методов обнаружения функциональных зависимостей применительно к обнаружению знаний в данных.

В статье [4] представлено, что e(X) может быть легко вычислена с использованием сокращенных сегментов:

$$e(X) = (\|\hat{\rho}_X\| - |\hat{\rho}_X|) / |r|, \quad (1)$$

где $\|\hat{\rho}_X\| = \sum |C_i|, C_i \in \hat{\rho}_X$ – сумма мощностей классов эквивалентности, входящих в сегмент.

Равенство $|\rho_X| = |\rho_{(X \cup A)}|$ не соблюдается для сокращенных сегментов, но, основываясь на том, что e(X) = e(Y) только тогда, когда $|\rho_X| = |\rho_Y|$, можно сделать вывод, что X→A является строгой ФЗ при следующем условии: $e(X) = e(X \cup A) = 0$.

Обоснование: строгая зависимость может быть соблюдена только тогда, когда X будет являться ключом (и, соответственно, обладать свойством уникальности). Таким образом, ρ_X будет содержать |r| классов эквивалентности мощности 1, следовательно, мощность сокращенного сегмента для данного случая будет равна 0. Подставив в (1) соответствующие значения, получим 0.

Для уменьшения количества сочетаний, подлежащих проверке на существование ФЗ, авторами метода были предложены пути сокращения пространства вариантов, если становится известным факт отсутствия в рассматриваемой ветке решетки атрибутов зависимостей, не являющихся минимальными. Нахождение ключевого атрибута отношения или их множества позволяет устранить из рассмотрения надмножества данного ключа.

Рассмотрим пример удаления ребер из решетки отношения R_e = {A,B,C,D}. Предположим, что в данном отношении соблюдается следующая минимальная зависимость: AC→B. Построим решетку атрибутов для данного случая (рис. 2):

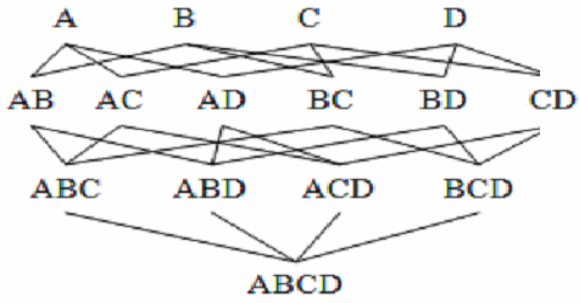

Рис. 2. Решетка для четырех атрибутов

Гипотетическая проверка зависимостей для первого уровня вида $B \rightarrow A \mid A,B \in R_e$ не выявит зависимостей в экземпляре отношения r_e. Будет сформирован следующий уровень $L_2 = \{X \in P(R_e) \mid |X| = 2\}$, для которого будет выявлена вышеуказанная зависимость. Соответственно, AC в данном отношении будет представлять ключ, и, поскольку минимальная зависимость для атрибута B была най-

дена, нет необходимости проверять зависимости, в которых определяющие части для B будут являться надмножествами ключа AC (суперключами). Следовательно, надмножества вида

$$X \in P(R_e) \mid |X| > 2,\ X \ni A,C$$

можно исключить из рассмотрения и удалить данные вершины из решетки атрибутов (выделены жирным остающиеся части, рис. 3):

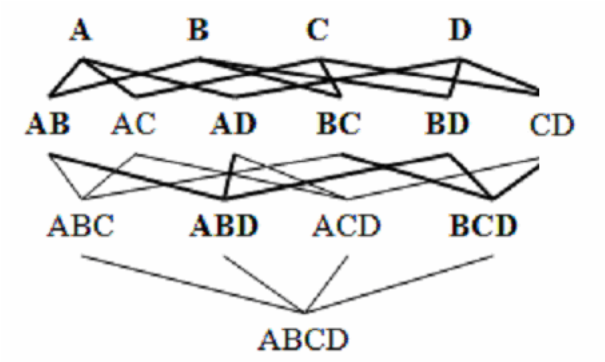

Рис. 3. Решетка с удаленными вершинами

Результатом выполнения данного метода над некоторым универсальным отношением является множество минимальных функциональных зависимостей, удовлетворяющих ему. Полученное множество служит входными данными для набора алгоритмов, составляющих метод синтеза.

Синтез логической схемы базы данных

Пусть задана схема БД – универсальное отношение $\sum = \{\sigma = (R;F)\}$, где F – множество функциональных зависимостей, удовлетворяющих R. Необходимо синтезировать схему $\sum' = \{R_i; K_i \mid i = \overline{1,\rho}\}$ (K_i – множество ключевых атрибутов отношения R_i) так, что:

а) все схемы σ_i находятся в 3НФ;

б) множество ограничений F_i представляется множеством ключей: $k_i=\{X_{i1},\ldots,X_{iqi}\}$, $X_{ij} \to R_i \in F^+$;

в) декомпозиция $\delta(\sigma) = \{(R_i; K_i) \mid i = \overline{1,p}\}$ имеет свойства соединения без потерь и сохранения функциональных зависимостей (F – зависимостей);

г) число схем отношения минимально.

Требования а), в) могут выполняться одновременно. Чтобы удовлетворить требование б), достаточно построить $R_i = X_{ij}\overline{X_{ij}}$. Требование г) является дополнительным критерием в задаче проектирования. Следует подчеркнуть важность этого требования: уменьшение числа отношений в декомпозиции универсального отношения означает следующее:

– уменьшение количества обращений к БД при введении каждого нового кортежа;

– уменьшение количества операций соединения кортежей производных отношений при возобновлении универсального.

Методы минимизации количества отношений в схеме БД базируются на понятии классов эквивалентных ключей базиса.

Алгоритм Бернштейна решает задачу построения 3НФ. Получаемая логическая схема БД характеризуется минимальным количеством отношений. В качестве сопровождающего примера будут приводиться промежуточные этапы выполнения алгоритма над следующими исходными данными:

R=ABCDEIJ;

$F = \{IE \to JC, AB \to IJ, IJ \to C, BC \to E, E \to IJ,$
$C \to A, AI \to DJ, D \to C\}$.

Как было упомянуто ранее, для этого используется алгоритм Бернштейна (BSCHEMA). Общая структура приведена на рис. 4. На вход данного алгоритма подается $\sum = \{\sigma = (R;F)\}$ – начальная схема БД в виде множества минимальных функциональных зависимостей F, на выходе будет получена $\sum' = \{\sigma_i = (R_i; K_i) \mid i = \overline{1,p}\}$ – эквивалентная синтезированная схема.

Алгоритм BASIS принимает на вход $F = \{U_1 \to V_1, \ldots, U_p \to V_p\}$; результатом является $G = \{U_j \to V_j \mid \overline{1,q}, q \le p\}$ – базис для F^+, где F^+ – замыкание для F.

Метод определения базиса заключается в следующем: присваивается G:=F. Для каждого $i = \overline{1,p}$ формируется вспомогательное множество $F := G - \{U_i \to V_i\}$. Относительно F и зависимости $U_i \to V_i$ решается вопрос о ее принадлежности данному множеству. Если данная зависимость принадлежит F, модифицируется G:=F'. Задачу определения принадлежности решает алгоритм MEMBER.

На вход алгоритма MEMBER поступает F – множество зависимостей и $g = X \to Y$. Результатом выполнения алгоритма является логическое значение α. α = истина, если $g \in F^+$. MEMBER использует алгоритм LINCLOSURE для создания замыкания X^+ множества X. Если $Y \subseteq X^+$, тогда $\alpha :=$ истина; в противоположном случае $\alpha :=$ ложь.

Алгоритм LINCLOSURE на вход принимает X и $F = \{U_1 \to V_1, \ldots, U_p \to V_p\}$. Выполнение функции состоит из двух этапов:

Этап 1. Начальное состояние для массива LIST – пустое, COUNT[i] = 0 для всех i. Для каждой зависимости $f_i = U_i \to V_i$ определяется мощность $|U_i|$ и заносится в «счетчик» COUNT[i]. Одновременно для каждого атрибута A с U_i зависимость f_i заносится в массив LIST[A].

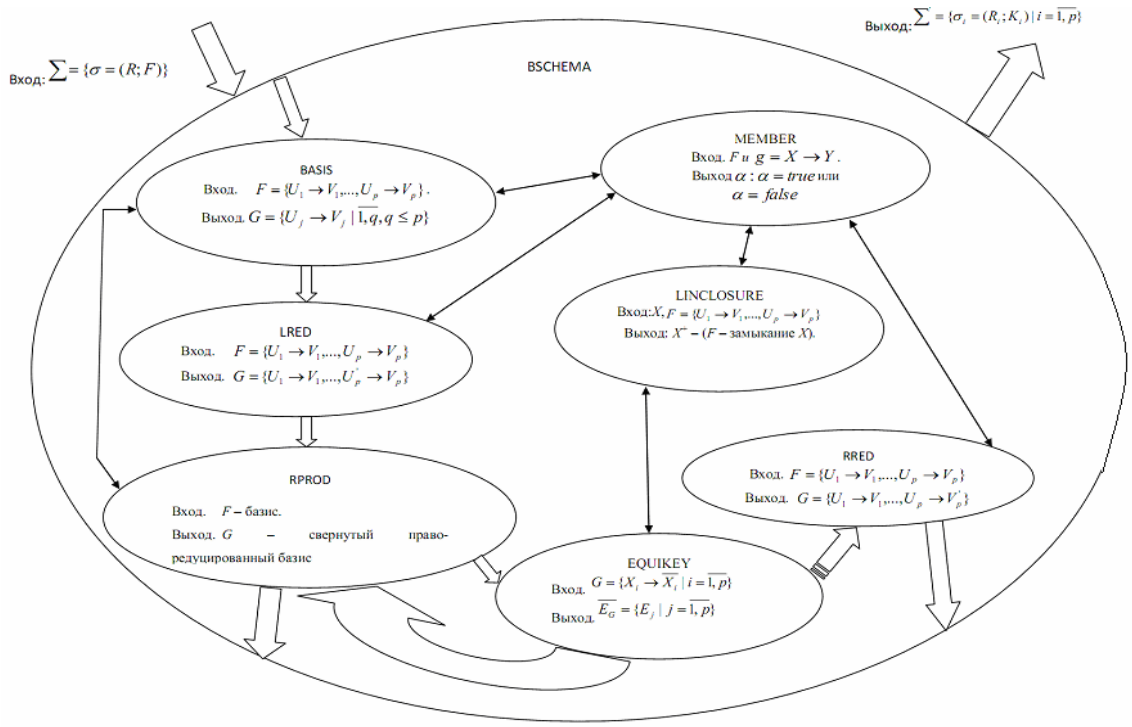

Рис. 4. Общая схема синтеза схемы базы данных

Этап 2. Используются вспомогательные множества атрибутов W, Y, $z : z := \emptyset$; W:=X; Y:=X. Выбирается очередной атрибут A из Y, модифицируется Y:=Y-{A}. Для каждой зависимости f_i из списка LIST[A] модифицируется счетчик COUNT[i]=0. Присваивается Z:=V_i-W_i; $W : W \bigcup Z; Y := Y \bigcup Z$, то есть к Y, а также к W добавляются новые атрибуты. Производится переход к следующей зависимости f_i из списка LIST[A]. Когда список исчерпан, производится переход к очередному атрибуту из Y. Когда $Y = \emptyset$ или все «счетчики» будут равняться нулю, X^+:=W. Производится останов. На выходе данного алгоритма будет получено X^+ – (F – замыкание X).

После обработки алгоритмом BASIS данных примера имеет место такой результат:

$$G = \{AB \to IJ, IJ \to C, BC \to E, E \to IJ, C \to A, AI \to DJ, D \to C\}.$$

На следующем шаге модифицируется $G := G \bigcup \{R \to O\}$, где O – фиктивный атрибут, что не принадлежит R:

$$G = \{AB \to IJ, IJ \to C, BC \to E, E \to IJ, C \to A, AI \to DJ, D \to C, ABCDEIJ \to O\}.$$

Применяя алгоритм LRED, выполняется левая редукция G; результатом является G_{lred}.

На вход алгоритма LRED поступает $F = \{U_1 \to V_1,..., U_p \to V_p\}$ – базис для F^+, на выходе $G = \{U_1 \to V_1,..., U'_p \to V_p\}$ – базис для F^+. Суть данного алгоритма заключается в следующем. Для каждого атрибута A_{ij} левой части U_i каждой зависимости из F выполняются действия:

– Присваивается вспомогательной переменной $Z := U_i - \{A_{ij}\}$.

– Проверяется, выводится ли $q = Z \to A_{ij}$ с F. Если результат выполнения MEMBER α = истина, тогда $U_i := Z$.

Для рассматриваемого примера G_{lred} будет иметь следующий вид:

$$G_{lred} = \{AB \to IJ, IJ \to C, BC \to E, E \to IJ, C \to A, AI \to DJ, D \to C, BIJ \to O\}.$$

Применяя алгоритм RPROD, выполняется правая редукция G_{lred}, результатом будет являться G_{rred}. На вход алгоритма подается F – базис, а на выходе получается G – свернутый базис, $G \equiv F$.

Для F строится сведенное покрытие F'.

Для F' вычисляется базис (используется алгоритм BASIS) G'.

Если свернуть каждое подмножество зависимостей из G' с одинаковыми левыми частями в одну зависимость, получится G.

Подав на вход G_{lred} рассматриваемого примера, получим:

$$G_{red} = \{AB \to I, IJ \to C, BC \to E, E \to IJ, C \to A, AI \to DJ, D \to C, BIJ \to O\}.$$

Применяя алгоритм EQUIKEY для G_{rred}, формируются классы зависимостей с эквивалентными ключами: $E_1 = \{X_{11} \to \overline{X}_{11},..., X_{1q1} \to \overline{X}_{1q1}\},...\}$ ..., $E_P = \{X_{P1} \to \overline{X}_{P1},..., X_{Pqp} \to \overline{X}_{Pqp}$. На вход алгоритма EQUIKEY подается $G = \{X_i \to \overline{X_i} \mid i = \overline{1,p}\}$ –

свернутый базис, в результате будет получено $\overline{E_G} = \{E_j \mid j = \overline{1,p}\}$ – разбиение G. Формирование $\overline{E_G}$ выполняется в два этапа:

Первый этап – для каждой зависимости $X_i \to \overline{X_i}$ из G вычисляется замыкание ее ключа; $W_i = X_i^+$. Второй этап – $\overline{E_1} := \{X_1 \to \overline{X_1}\}$. Для ключа X_i очередной зависимости из $G(i = \overline{2,p})$ выполняются следующие действия.

Сопоставляется X_i с замыканием $W_1,\ldots,W_{i-1}$ уже рассмотренных ключей. Если $X_i \le W_j$ для какого-либо $j(j = 1, i-1)$, также происходит проверка, соответствует ли $X_j \le W_i$. Если соответствует, то $E(X_j) := E(X_j) \cup \{X_i \to \overline{X_i}\}$, выполняется останов, сопоставляется X_i и производится переход к следующей зависимости G. Если нет, то сопоставление продолжается; при этом, если j=i, а ключ, эквивалентный X_i, не найден, создается новое множество $\overline{E_k} := \{X_i \to \overline{X_i}\}(\overline{E_k} = E(X_i))$, выполняется переход к следующей зависимости G. Последнее значение k=q. Применяя EQUIKEY, для рассматриваемого примера будет получено такое множество ключей:

$$E_1 = \{AB \to I, BC \to E, BIJ \to O\},$$
$$E_2 = \{IJ \to C, AI \to DJ\},$$
$$E_3 = \{E \to IJ\},$$
$$E_4 = \{C \to A\},$$
$$E_5 = \{D \to C\}.$$

Для каждого класса E_j формируется множество зависимостей между ключами

$$E_j' = \{X_{j1} \to X_{j2}, X_{j2} \to X_{j3}, \ldots, X_{j1j} \to X_{j1}\}.$$

То есть, для множества ключей, содержащих более одного элемента, составляются зависимости между ключами:

$$E_1' = \{AB \to BC, BC \to BIJ, BIJ \to AB\},$$
$$E_2' = \{IJ \to AI, AI \to IJ\}.$$

Применяя RPROD или RRED, выполняется правая редукция зависимостей $\bigcup_{i=1}^{P} E_j$ относительно множества $\bigcup_{i=1}^{P}(E_j \cup E_j')$, вместо каждого E_j будет получен класс E_j''.

Сущность алгоритма RRED – вычисление правой редукции базиса. На вход принимается $F = \{U_1 \to V_1, \ldots, U_p \to V_p\}$ – базис для F^+, на выходе образуется $G = \{U_1 \to V_1, \ldots, U_p \to V_p'\}$ – правая редукция базиса для F^+.

Принцип работы заключается в следующем: для каждого атрибута A_{ij} левой части V_i каждой зависимости с F выполняются следующие действия. Вспомогательной переменной присваивается $Z := U_i - \{A_{ij}\}$. Относительно множества $F' = (F - \{U_i \to V_i\}) \cup \{U_i \to Z\}$ и зависимости $U_i \to A_{ij}$ выполняется проверка принадлежности с помощью алгоритма MEMBER. Если результат выполнения MEMBER α = истина, выполняется присвоение $V_i := Z$.

Применяем алгоритм RPROD:

$$E_1'' = \{BC \to EIJ, BIJ \to OA, AB \to C\},$$
$$E_2'' = \{IJ \to CA, AI \to DJ\}.$$

Последний этап состоит в формировании $R_j := [E_j' \cup E_j'']$, причем, если $O \in R_j$, то $R_j := Rj - \{O\}$ и множества выделенных ключей $K_j := \{X_{j1}, \ldots, X_{jqi}\}$,

Таким образом, искомая схема примет вид $\sum' = \{\sigma_1 = (R_1; K_1), \ldots, \sigma_P = (R_P; K_P)\}$.

В результате работы данного метода синтеза для рассматриваемого примера будет получена следующая схема:

$$\sum\nolimits_1 = \{\sigma_1 = (ABCIJE; AB, BC, BIJ),$$
$$\sigma_2 = (IJACD; IJ, AI), \sigma_3 = (EIJ; E),$$
$$\sigma_4 = (CA; C), \sigma_5 = (DC; D)\}.$$

Выводы

Применение рассмотренного подхода на этапе проектирования РБД позволяет построить логическую схему, наиболее точно соответствующую взаимосвязям между данными на момент проведения обработки. Использование метода выявления функциональных зависимостей дает возможность включить в рассмотрение те взаимосвязи предметной области, которые не были включены в исходный процесс проектирования, но были образованы и поддерживаются со стороны программного обеспечения, оперирующего целевой БД. Это позволяет гарантировать оптимальность конечной логической схемы, полученной с помощью метода синтеза, и, как следствие – минимизацию избыточности хранимых данных и уменьшение затрат ресурсов на хранение и извлечение данных в процессе функционирования информационной системой в целом.

Следует отметить, что представленный подход не рассматривает перенос данных из исходной БД в конечную. Этот вопрос является темой для дальнейших исследований в данной области. Также возможным направлением является включение в рассмотрение иных типов зависимостей (например, многозначных) для получения возможности применения предложенного подхода при создании логических схем в нормальных формах более высокого порядка.

Список литературы

1. Qiang Wei. Efficient discovery of functional dependencies with degrees of satisfaction / Qiang Wei, Guoqing Chen // International Journal of Intelligent Systems. – 2004. – P. 1089-1110.

2. Wenfei Fan. Discovering Conditional Functional Dependencies / Wenfei Fan, Floris Geerts, Jianzhong Li, Ming Xiong // IEEE Trans. on Knowledge and Data Engineering, 2010.

3. Aravind Krishna Kalavagattu. Mining approximate functional dependencies as condensed representations of association rules [Электронный ресурс] – 2008. // Режим доступа к ресурсу: rakaposhi.eas.asu.edu – Загл. с экрана.

4. Ykä Huhtala. Tane: An Efficient Algorithm For Discovering Functional and Approximate Dependencies [Электронный ресурс] / Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, Hannu Toivonen // The Computer Journal 42 (2). – 1999. – P. 100-111. Режим доступа к ресурсу: http://www.oxfordjournals.org/our_journals/computer_journal/hdb/ Volume_42/ Issue_02/ – Загл. с экрана.

5. Catharine Wyss. FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances [Электронный ресурс] / Catharine Wyss, Chris Giannella, Edward Robertson. – 2001. – P. 101-110.

6. Bernstein P.A. A unified approach to functional dependencies and relations / P.A. Bernstein, J.R. Swenson, D.C. Tsichritzis // Proc. ACM 1975 SIGMOD Conf., San Jose, Calif. – P. 237-245.

7. Bernstein P.A. Synthesizing Third Normal Form Relations from Functional Dependencies / P.A. Bernstein // ACM Transactions on Database Systems (TODS) – 1976. – Volume 1, Issue 4. – С. 277-298.

8. Буслік М.М. Оптимальні зображення реляційних баз даних / М.М. Буслік. – К.: ІСДО, 1993. – С. 48-55.

Поступила в редколлегию 25.07.2011

Рецензент: д-р техн. наук, проф. С.Г. Удовенко, Харьковский национальный университет радиоэлектроники, Харьков.

СИНТЕЗ ЛОГІЧНОЇ СХЕМИ РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ З ВИЯВЛЕНОЇ МНОЖИНИ ФУНКЦІОНАЛЬНИХ ЗАЛЕЖНОСТЕЙ

В.О. Радченко, Ю.А. Мальков, А.С. Балюк, Ю.С. Горпиненко

У даній статті розглядається спосіб проектування логічної схеми реляційної бази даних на основі існуючої РБД. Вхідними даними для методу є інформація, що зберігається у вихідній БД. Застосовуючи метод виявлення функціональних залежностей з набору даних, що містяться у БД, набуває можливості відновити взаємозв'язки між даними, що накладаються предметною областю. Ці взаємозв'язки представляють собою обмеження цілісності у РБД. Отримані залежності є вхідними даними для ланцюга алгоритмів, що представляють метод синтезу логічної схеми. Кінцевим результатом є логічна схема БД у третій нормальній формі.

Ключові слова: *логічна схема, функціональна залежність, предметна область, обмеження цілісності, метод синтезу, третя нормальна форма.*

RELATIONAL DATABASE LOGICAL SCHEMA SYNTHESIS FROM THE SET OF DISCOVERED FUNCTIONAL DEPENDENCIES

V.A. Radchenko, Y.A. Malkov, A.S. Balyuk, U.S. Horpinenko

This article presents the way of relational database logical schema engineering on the basis of existing RDB. Information stored in initial DB is the input data for the method described. Applying the method of functional dependencies' discovery from the set of data that DB contains, it becomes available to restore relations in data that are caused by universe of discourse. These relations appear as integrity constraints for the chain of algorithms that form the logical schema synthesis method. It produces final result as RDB logical schemae in third normal form.

Keywords: *logical schemae, functional dependency, universe of discourse, integrity constraint, third normal form.*