электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Конструктор баз данных на основе сущностей и их реквизитов с возможностью нормализации

77-30569/242645
№ 10, октябрь 2011
авторы: Виноградова М. В., Игушев Э. Г.

УДК 004.021

МГТУ им. Н.Э. Баумана
vinogradova.m@gmail.com
igushev@narod.ru

Введение

При разработке реляционной базы данных выполняется несколько задач. Во-первых, необходимо выделить сущности, их реквизиты и связи между ними. Во-вторых, на основе выделенных сущностей, их реквизитов и связей построить схемы отношений. В-третьих, выполнить нормализацию построенных схем отношений. И наконец, преобразовать схемы отношений ко множеству таблиц в терминах конкретной СУБД.

При этом возникают сложности, связанные с тем, что исходные и промежуточные данные при проектировании могут быть представлены в различных моделях: множеством сущностей, реквизитов и связей, множеством схем отношений, множествами атрибутов и функциональных зависимостей.

В процессе нормализации баз данных разработчику приходится работать с множеством атрибутов и функциональных зависимостей между ними. Как правило, такое множество слишком велико и, как следствие, с ним тяжело работать и обрабатывать. А само представление предметной области в виде множества атрибутов и связей между ними ненаглядно и, как следствие, неудобно для разработчика.

На практике, разработчик сознательно или неосознанно с начала процесса проектирования пользуется агрегированием, объединяя атрибуты в сущности. Однако, применяя стандартные алгоритмы и приемы проектирования, формальное агрегирование атрибутов в сущности происходит на последнем этапе.

Также в процессе проектирования баз данных немалое внимание уделяется нормализации. Существуют алгоритмы, которые позволяют нормализовать схему базы данных, однако, эти алгоритмы также работают с множеством атрибутов и функциональных зависимостей между ними. А алгоритмы, работающие со схемами отношений, неуниверсальны, поскольку для каждой нормальной формы требуется свой алгоритм. Такие алгоритмы работают методом декомпозиции схем отношений и не способны сделать композицию там, где это упростило бы общую схему базы данных.

Для повышения качества процесса проектирования баз данных необходимо создать конструктор баз данных, который позволяет вводить исходные данные и просматривать полученные результаты в наглядном и понятном для проектировщика виде, то есть в виде сущ-

ностей, их связей и реквизитов. При этом конструктор баз данных должен выполнять нормализацию структуры базы данных.

Нормализация

Нормализация – это процесс приведения схем отношений к нормальным формам [2].

Первая нормальная форма (1НФ). Согласно определению отношений, любое отношение уже находится в 1НФ. Перечислим свойства 1НФ:

1. Все атрибуты атомарны, различны (из определения множества) и не упорядочены;
2. В отношении нет одинаковых кортежей (из определения множества) и они не упорядочены.

При этом отношение обладает следующими аномалиями, которые рассмотрены на примере отношения, содержащего атрибуты «поставщик», «склад» и «поставка»:

1. Аномалия вставки. В данное отношение нельзя добавить информацию о новом поставщике, если он не осуществил ни одной поставки, или о новом складе, если на него не осуществлялась ни одна поставка.
2. Аномалия удаления. Например, при удалении всех поставок, которые осуществлял какой-либо поставщик, удалится информация о самом поставщике;
3. Аномалия обновления. Если, например, изменится адрес какого-либо склада, то изменения придется вносить сразу в несколько кортежей. Данную аномалию еще называют потенциальной противоречивостью;
4. Избыточность. Информация о наименованиях поставщиков, адресах складов и наименованиях поставок повторяются несколько раз.

Вторая Нормальная Форма (2НФ). Отношение находится во второй нормальной форме (2НФ) тогда и только тогда, когда отношение находится в 1НФ и нет неключевых атрибутов, зависящих от части сложного ключа.

Третья Нормальная Форма (3НФ). Отношение находится в третьей нормальной форме (3НФ) тогда и только тогда, когда отношение находится в 2НФ и все неключевые атрибуты взаимно независимы.

Нормальная форма Бойса-Кодда (НФБК). Отношение находится в нормальной форме Бойса-Кодда (НФБК) тогда и только тогда, когда детерминанты всех функциональных зависимостей являются потенциальными ключами.

Приведение схем отношений к 3НФ или к НФБК устраняет все перечисленные аномалии. Отсутствие аномалий при работе с базой данных является целью проведения нормализации.

Постановка задачи

Основываясь на вышесказанном, можно сформулировать задачу проектирования и разработки конструктора баз данных. К констуктору предъявляются следующие требования:

1. Наличие наглядного и интуитивно понятного интерфейса пользователя.
2. Возможность оперировать не только атрибутами и связывающими их функциональными зависимостями, но и сущностями, объединяющими атрибуты.
3. Наличие инструмента нормализации.
4. Инструмент нормализации должен быть универсальным.

Конструктор баз данных

На рис. 1 представлен пример схемы базы данных, спроектированной в данном конструкторе. Констуктор обладает визуальной средой, в которой пользователь может задать сущности, их атрибуты и функциональные зависимости. На рис. 1 видно, что разработчик

может агрегировать имеющиеся у него атрибуты в сущности. Сущности показаны серым цветом. Особый прием данного конструктора заключается в том, что стрелками показаны функциональные зависимости и они же показывают вхождение атрибутов в сущности (стрелка направлена от сущности к входящему в него атрибуту).

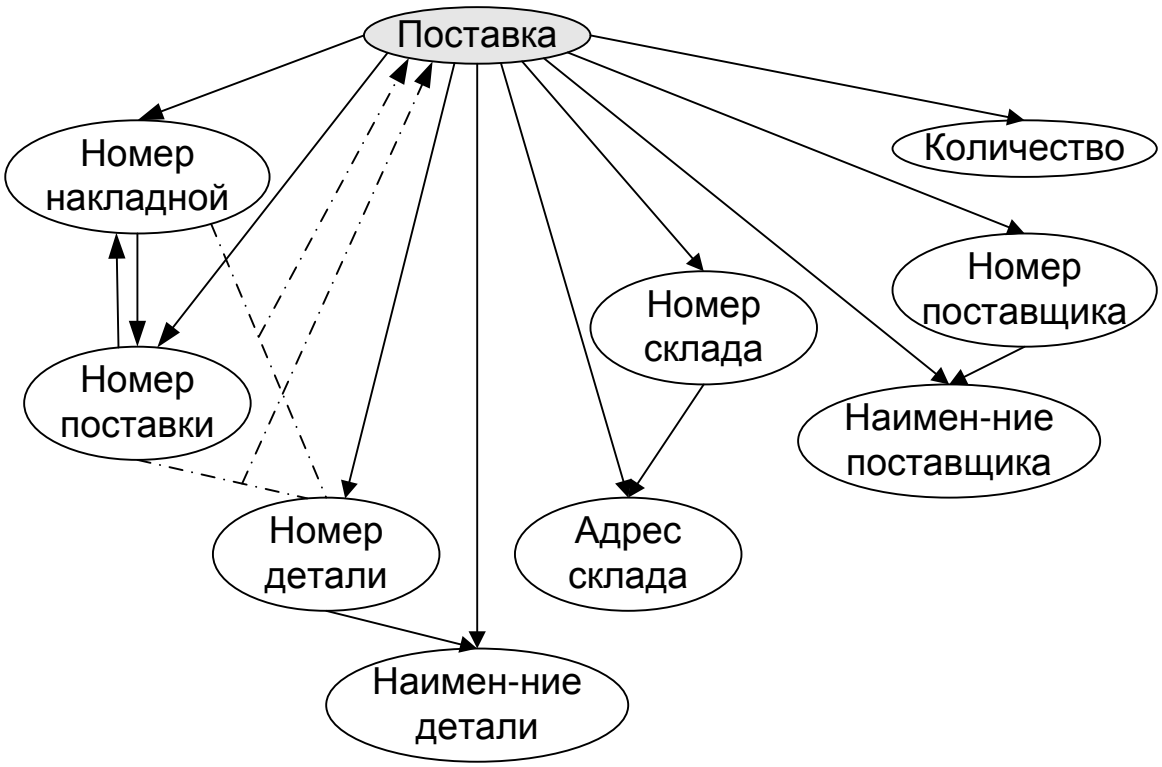

Рисунок 1. Пример схемы базы данных.

Это позволительно благодаря следующей интерпретации: каждая сущность на схеме для конструктора представляет собой атрибут с уникальным номером данной сущности, т.о. все стрелки действительно представляют собой функциональные зависимости, однако разработчик может видеть в них вхождение атрибута в сущность.

Пусть *A* – любой элемент структуры базы данных, атрибут или сущность. Как видно из устройства конструктора, атрибуты и сущности для конструктора не имеют различий.

Тогда *A.entity* – логическое свойство элемента, указывающее, является элемент сущностью или атрибутом.

Пусть $X \to Y$ - функциональная зависимость в структуре базы данных, где *X* и *Y* – это множество элементов.

Ключом сущности будет любая функциональная зависимость, детерминантом которой будут входящие в нее атрибуты, а в зависимой части будет сама сущность.

Ключи обозначаются как $X \xrightarrow{key} Y$, где *X* – множество атрибутов, входящих в ключ, а *Y* состоит из одного элемента со свойством *A.entity = true*.

При построении структуры базы данных накладывается следующее ограничение: каждая сущность обязана иметь хотя бы один ключ.

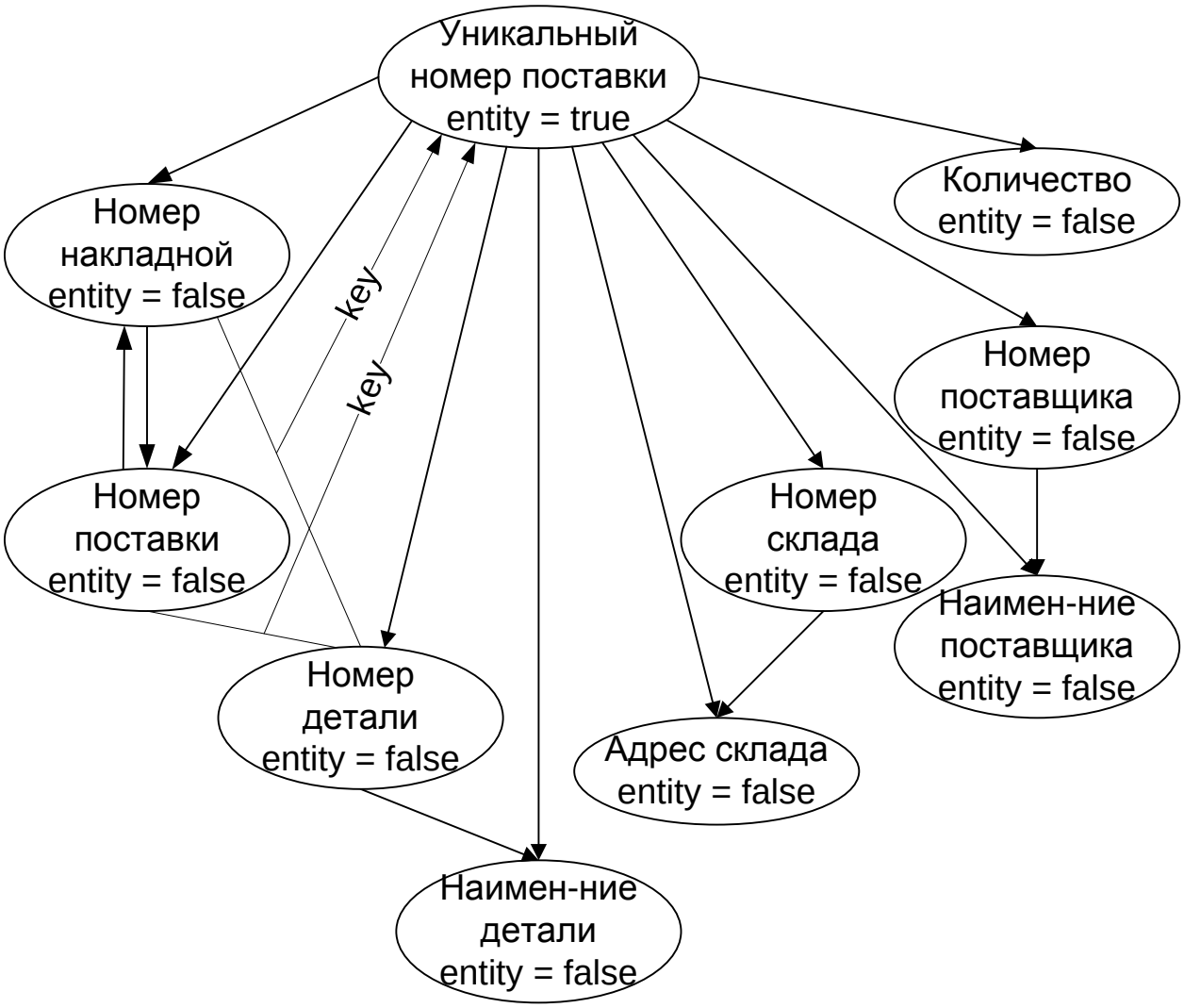

Рисунок 2. Схема базы данных с учетом ключей и свойств.

Таким образом, для конструктора рассматриваемая в качестве примера схема базы данных выглядит следующим образом (см. рис. 2).

Алгоритм нормализации структуры базы данных

Конструктор выполняет автоматическую нормализацию схемы базы данных. В основе приведенного ниже алгоритма лежит общеизвестный алгоритм Ульмана [1].

Вход:

1. Множество элементов базы данных *Q*.
2. Множество функциональных зависимостей *F*.

Выход:

1. Множество элементов базы данных *Q*.
2. Множество функциональных зависимостей *F*.

Алгоритм:

1. Каждую функциональную зависимость из *F* заменить на совокупность зависимостей, каждая из которых содержит один атрибут в правой части.
2. Полагаем Changed = true.
3. Пока *Changed = true*:
   3.1. *Changed = false*.

3.2. Для каждой зависимости $X \to Y$ из F проверить, принадлежит ли данная зависимость замыканию множества $\left(F - \left(X \to Y\right)\right)^+$. Если принадлежит, то (см. рис. 3):

3.2.1. Если в зависимости $X \to Y$ X является множеством из одного атрибута A со свойством *A.entity = true*, а Y является подмножеством какого-либо ключа $K \to A$, то:

3.2.1.1. Добавляем новую сущность B в множество Q.

3.2.1.2. Добавляем новую функциональную зависимость $B \to K$.

3.2.1.3. Добавляем новый ключ $K \to B$.

3.2.1.4. Добавляем новую функциональную зависимость $A \to B$.

3.2.1.5. Добавляем новый ключ $B \to A$.

3.2.1.6. Удаляем функциональную зависимость $A \to K$.

3.2.1.7. Удаляем ключ $K \to A$.

Примечание: Так как $A \to B$, $B \to K$, то $A \to K$ и значит, что функциональная зависимость из базы данных не исчезла. Так как $K \to B$, $B \to A$, то $K \to A$ и значит, что функциональная зависимость из базы данных не исчезла.

3.2.2. Удалить зависимость $X \to Y$ из F.

3.2.3. *Changed = true.*

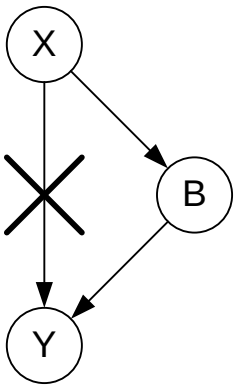

Рисунок 3. Удаление транзитивной зависимости $X \to Y$.

Примечание: для проверки того, принадлежит ли зависимость $X \to Y$ замыканию множества $\left(F - \left(X \to Y\right)\right)^+$ достаточно проверить принадлежность Y замыканию атрибутов X^+ на множестве функциональных зависимостей $F - \left(X \to Y\right)$.

3.3. Для каждой зависимости $X \to Y$ из F:

3.3.1. Для каждого собственного подмножества $Z \subset X$ проверить ли зависимость $Z \to Y$ замыканию F^+. Если принадлежит, то (см. рис. 4):

3.3.1.1. Заменить зависимость $X \to Y$ на $Z \to Y$.

3.3.1.2. *Changed = true.*

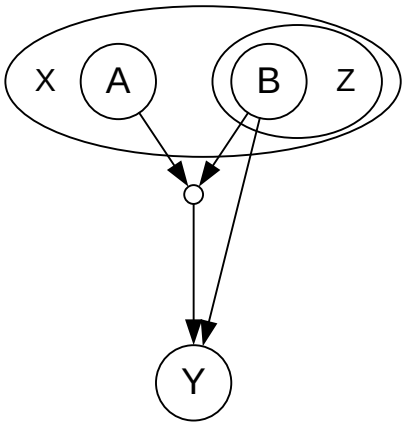

Рисунок 4. Замена зависимости $X \to Y$ на зависимость $Z \to Y$.

Примечание: для проверки того, принадлежит ли зависимость $Z \to X$ замыканию множества F^+ достаточно проверить принадлежность *X* замыканию атрибутов Z^+ на множестве функциональных зависимостей *F*.

3.4. Для каждой функциональной зависимости $X \to Y$: если *X* не является множеством из одного атрибута *E* со свойством *E.entity = true* и функциональная зависимость не является ключом, то (все зависимости $X \to Y_i$ с одинаковым детерминантом, удовлетворяющие данному условию рассматриваются вместе) выполнить (см. рис. 5):

3.4.1. Добавляем новую сущность *D* в множество *Q*.

3.4.2. Добавляем новую функциональную зависимость $D \to XY_1...Y_n$.

3.4.3. Добавляем новый ключ $X \to D$.

3.4.4. Удалить все рассматриваемые $X \to Y_i$ из *F*.

3.4.5. *Changed = true.*

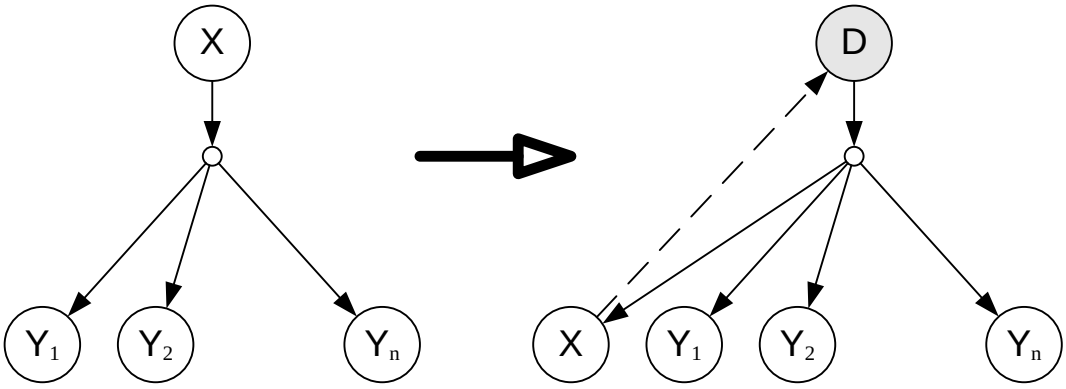

Рисунок 5. Образование сущности из функциональной зависимости.

Примечание: Так как $X \to D$, $D \to XY_1...Y_n$, то $X \to Y_i$ и значит, что функциональная зависимость из базы данных не исчезла.

3.5. Для каждого атрибута *A*, не входящего в зависимую часть ни одной функциональной зависимости (см. рис. 6):

3.5.1. Добавим новую сущность *D* в множество Q.

3.5.2. Добавляем новую функциональную зависимость $D \to A$.

3.5.3. Добавляем новый ключ $A \to D$.

3.5.4. Во всех функциональных зависимостях (в детерминантах и зависимых частях) заменяем *A* на *D*.

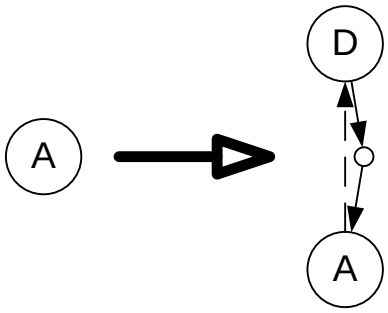

Рисунок 6. Образование сущности из атрибута, не зависящего ни от одного элемента.

3.6. Для каждой ключевой функциональной зависимости $X \to A$. Если *X* входит в детерминант или зависимую часть какой-либо другой функциональной зависимости, то (см. рис. 7):

3.6.1. Заменить вхождение *X* на *A*.

3.6.2. *Changed = true.*

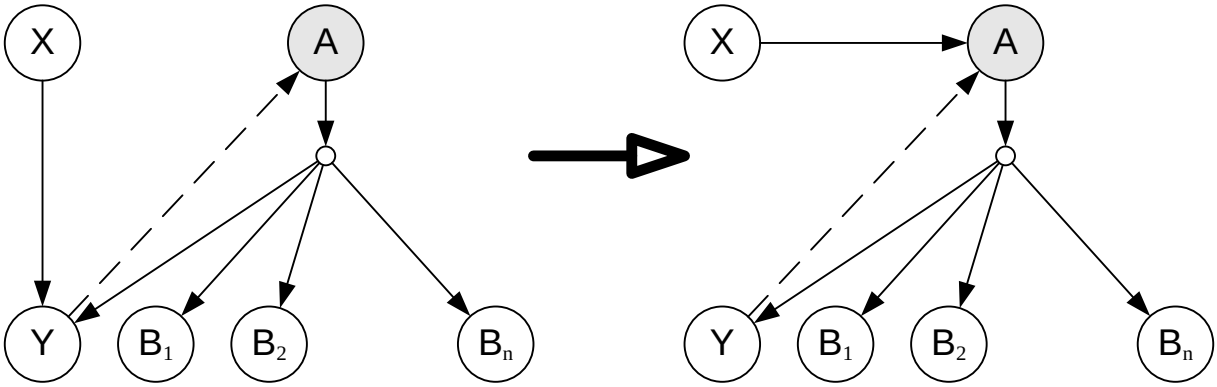

Рисунок 7. Анализ вхождения ключа в функциональную зависимость.

3.7. Для каждой пары элементов *A* и *B:* если *A.entity = true* и *B.entity = true* и $A \to B \in F^+$ и $B \to A \in F^+$, то (см. рис. 8):

3.7.1. Заменить во всех функциональных зависимостях *B* на *A*.

3.7.2. Удалить *B* из множества элементов.

3.7.3. *Changed = true.*

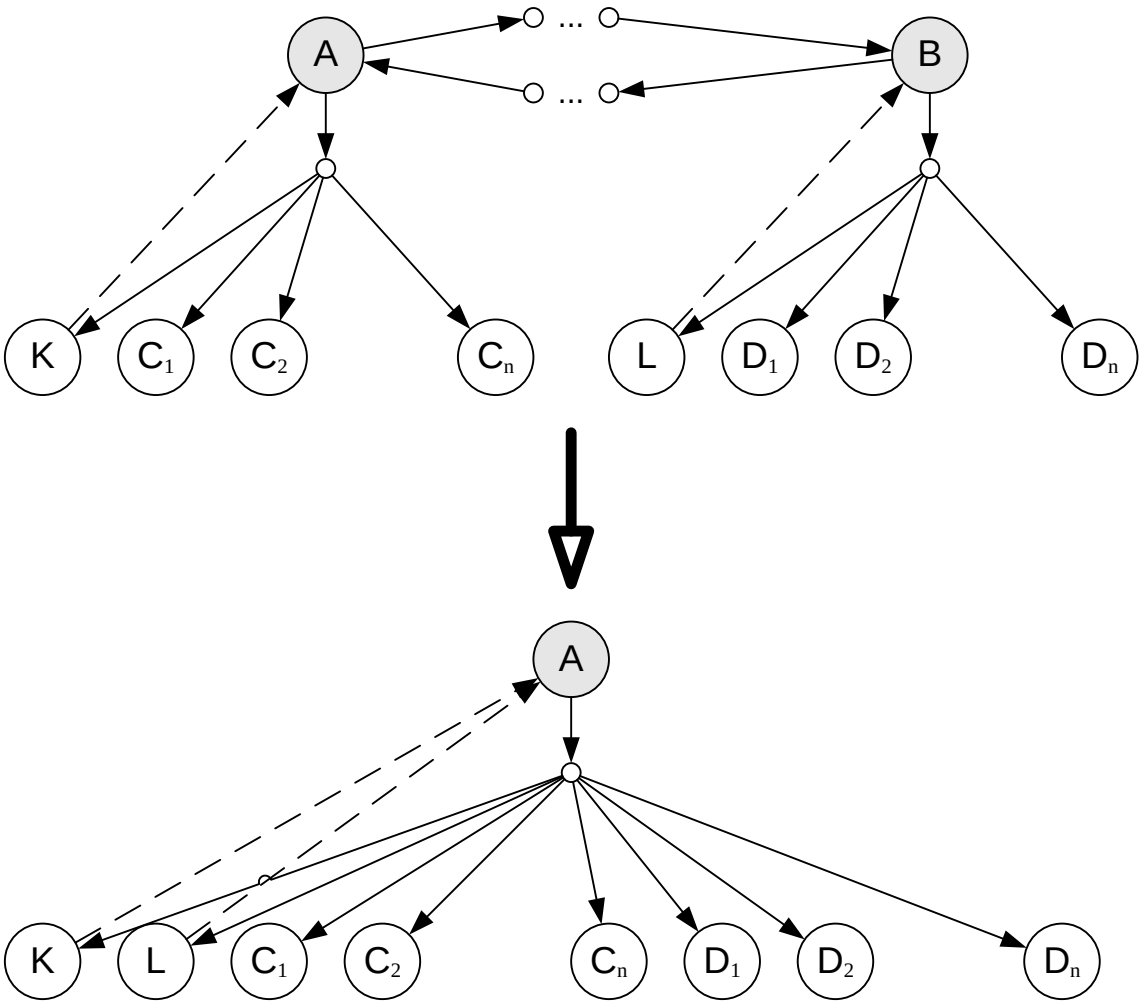

Рисунок 8. Объединение сущностей, взаимно однозначно определяющих друг друга.

Примечание: При выполнении условия, *A* и *B* будут сущностями, которые взаимно-однозначно определяют друг друга. На практике это означает, что это одна сущность, разделенная на две части.

Основные отличия данного алгоритма от алгоритма Ульмана:

1. Алгоритм адаптирован к наличию в структуре не только атрибутов, но и сущностей, объединяющей атрибуты. В связи с этим не из всех функциональных зависимостей появляются схемы отношений.
2. Добавлен анализ на вхождение ключей в детерминанты и зависимые части функциональных зависимостей.
3. Добавлен анализ на сущности, взаимно-однозначно определяющих друг друга.

Пример работы алгоритма

После преобразования структуры базы данных, изображенной на рис. 2, последняя будет выгладить следующим образом (см. рис. 9).

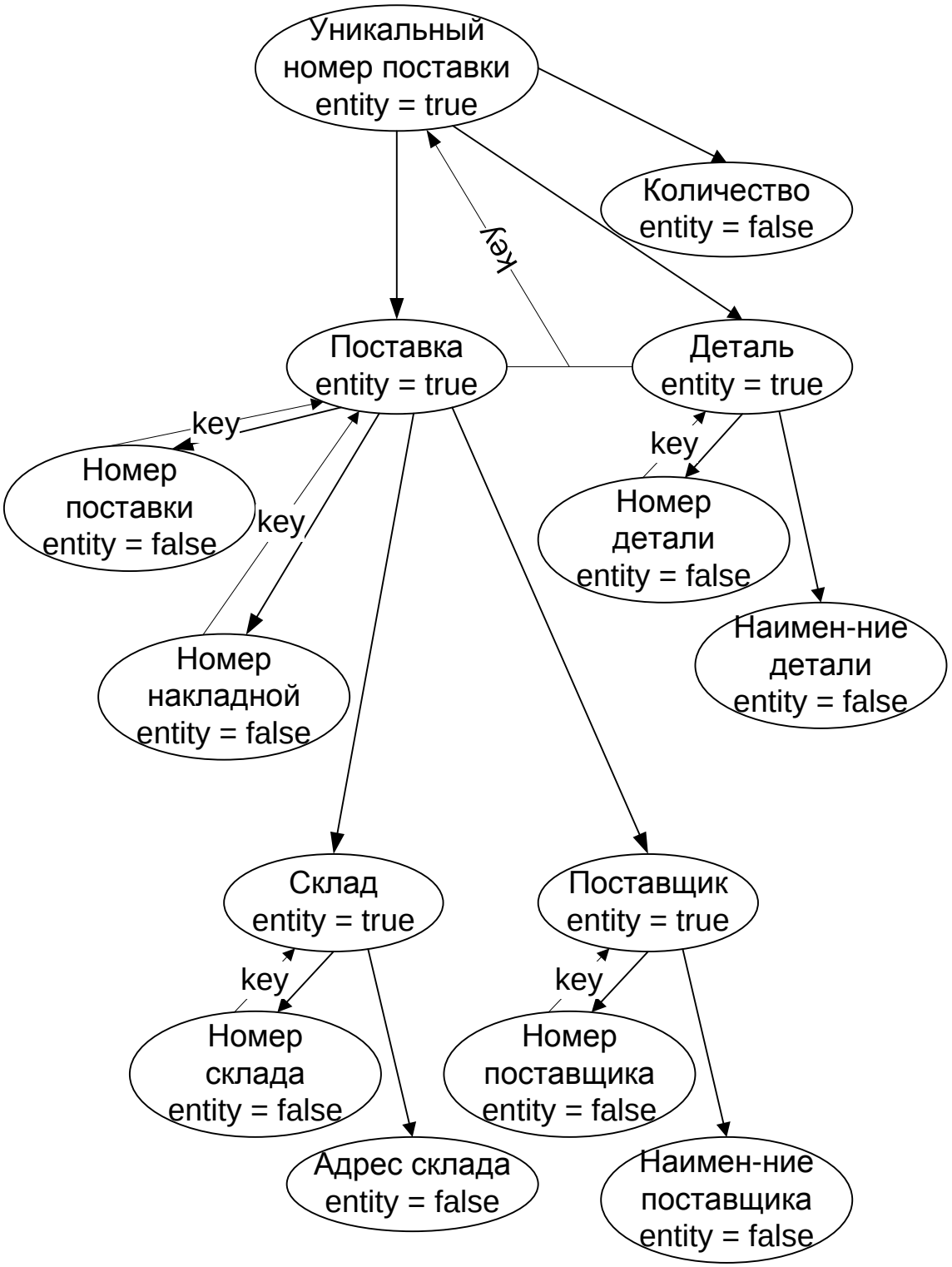

Рисунок 9. Нормализованная схема базы данных с учетом ключей и свойств.

Разработчик базы данных при этом будет видеть следующую схему (см. рис. 10).

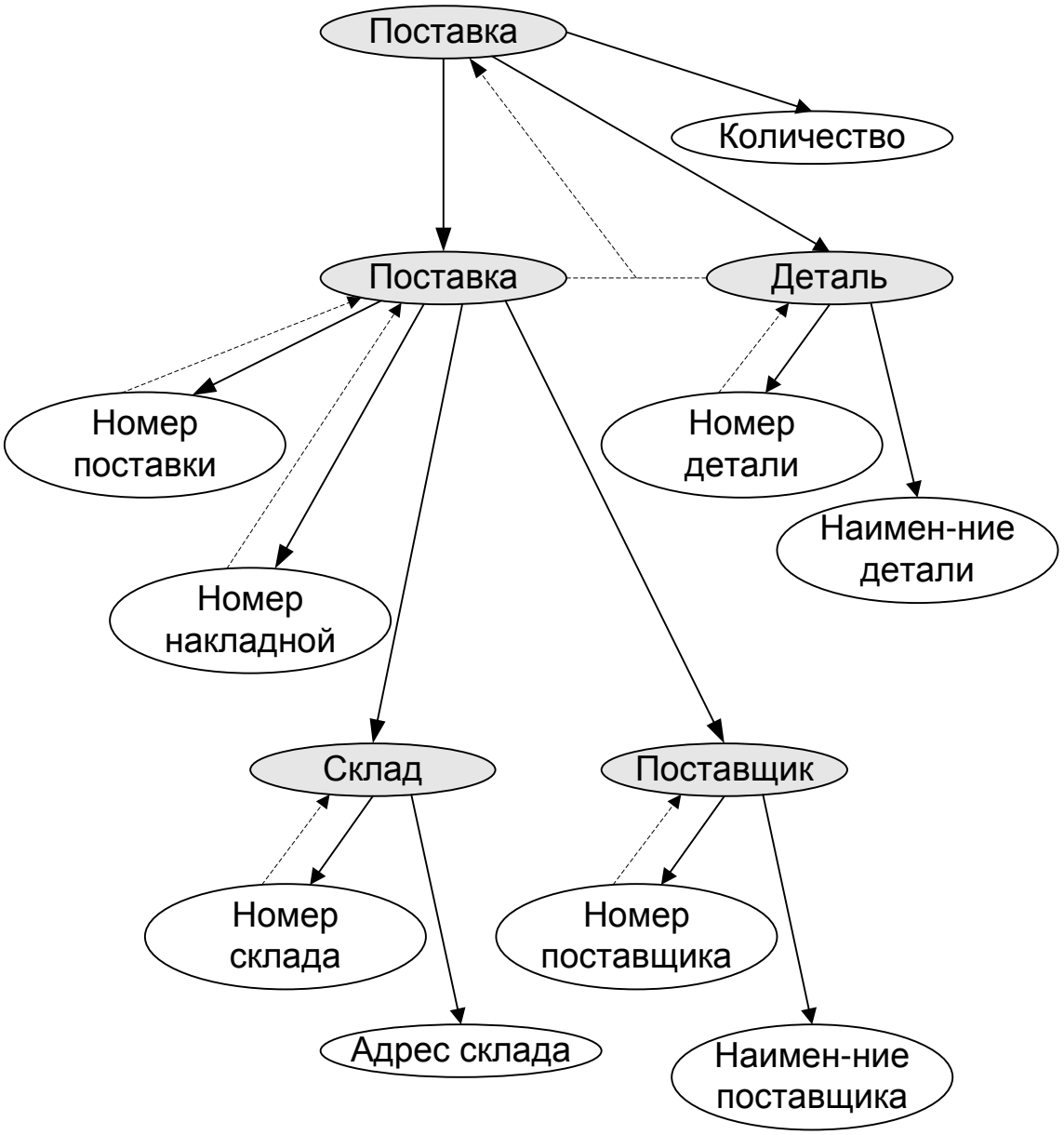

Рисунок 10. Нормализованная схема базы данных.

Выводы

В результате работы получена модель конструктора баз данных, который удовлетворяет следующим требованиям:

1. Наличие наглядного и интуитивно понятного интерфейса пользователя.
2. Возможность оперировать не только атрибутами и связывающими их функциональными зависимостями, но и сущностями, объединяющими атрибуты.
3. Наличие инструмента нормализации.
4. Инструмент нормализации является универсальным.

Предложенный констуктор баз данных является эффективным, эргономичным и удобным средством автоматизированного проектирования баз данных.

Список литературы

1. Гарсиа-Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс: Пер. с англ. – М.: Издательский дом «Вильямс», 2004 г.
2. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: Издательский дом «Вильямс», 2006.

electronic scientific and technical periodical
SCIENCE and EDUCATION
EL № FS 77 - 30569. №0421100025. ISSN 1994-0408

Database constructor on basis of entities and their properties with possibility of normalization
77-30569/242645

 # 01, January 2012
authors: M.Vinogradova, E.Igushev

Bauman Moscow State Technical University
vinogradova.m@gmail.com
igushev@narod.ru

The authors consider principles of creating database constructors which have a convenient graphical user interface and a universal normalization tool. The constructor allows operating not only with attributes and functional dependences, but also with entities which unite attributes. The database structure normalization algorithm is based on Ullman algorithm which additionally carries out analysis of keys entry into determinants and dependant parts of functional dependences, and which also reveals entities unambiguously determining one another. The algorithm is adapted to the presence - in the structure - of not only attributes but also entities that unite attributes.

Publications with keywords: functional dependence, normal form
Publications with words: functional dependence, normal form

Reference

1. Garsia-Molina G., Ul'man D., Uidom D., Database system. A full course of, Moscow, Izdatel'skii dom «Vil'iams», 2004.
2. Deit K. Dzh., An introduction to database systems, Moscow, Izdatel'skii dom «Vil'iams», 2006.