Теория баз данных

Лекция 7. Нормализация

Е. П. Моргунов
Сибирский федеральный университет
г. Красноярск
Институт космических и информационных технологий
emorgunov@mail.ru

Версия от 12.04.2018

7.1. Введение

- **Нормализация.** Метод создания набора отношений с заданными свойствами на основе требований к данным, установленных в некоторой организации.
- Цель: идентифицировать подходящий набор отношений, который будет адекватно поддерживать требования к данным некоторого предприятия:
  - Минимальное количество атрибутов
  - Атрибуты, которые логически тесно связаны, должны находиться в одном отношении
  - Минимальная избыточность, т. е. каждый атрибут должен быть представлен только один раз, за исключением атрибутов внешних ключей

7.1. Введение (продолжение)

- Нормализация – это формальный метод, который можно использовать на любой стадии процесса проектирования.
- I подход – автономный метод проектирования базы данных «снизу-вверх».
- II подход – метод проверки структуры отношений, созданных с помощью одного из методов проектирования «сверху-вниз», например, ER-моделирования.
- Общее назначение процесса нормализации заключается в следующем:

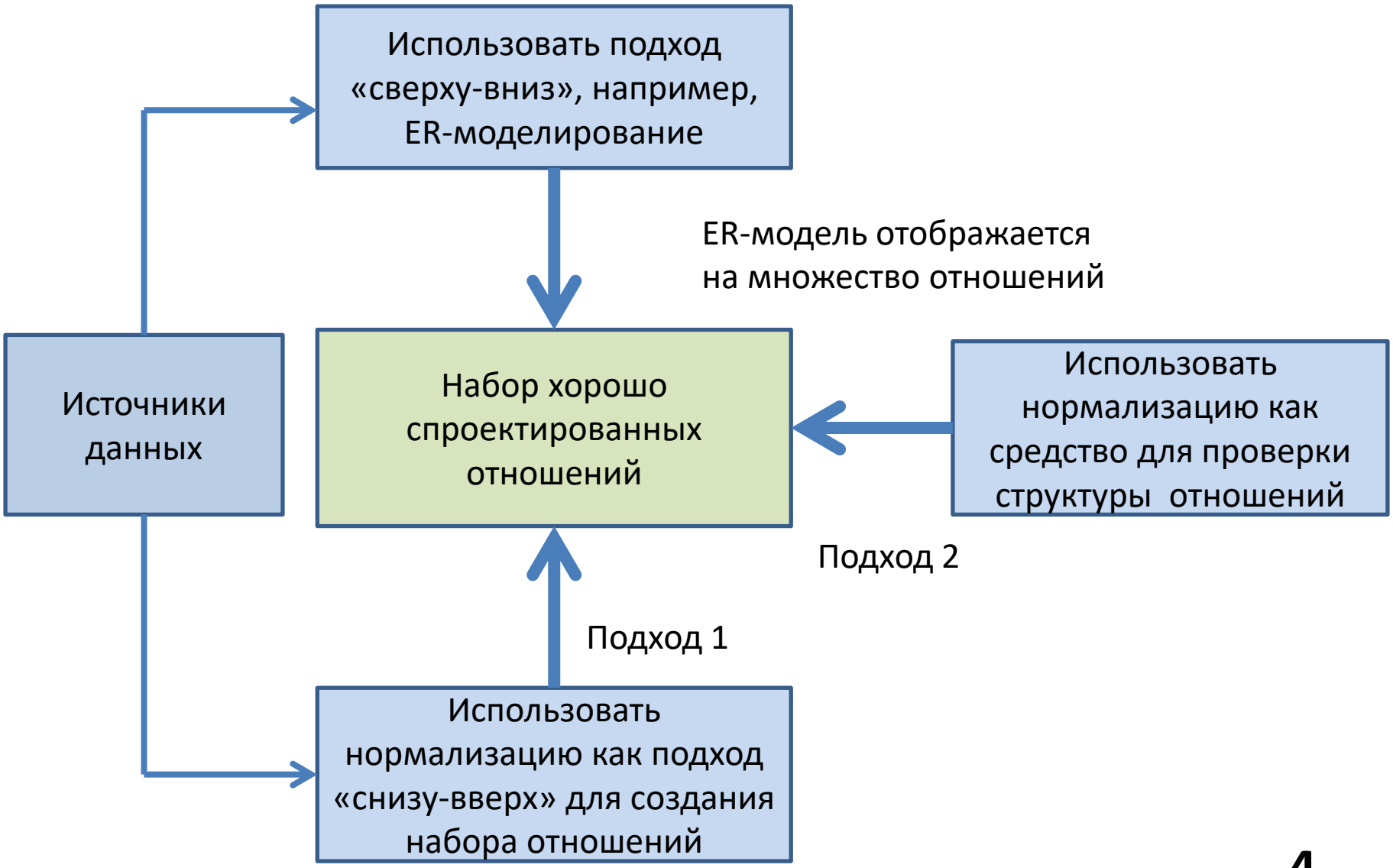
❑ исключение некоторых типов избыточности;

❑ устранение некоторых аномалий обновления (см. далее);

❑ разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;

❑ упрощение процедуры применения необходимых ограничений целостности.

7.1. Введение (продолжение)



7.2. Избыточность данных и аномалии обновления

- При проектировании реляционной базы данных важным является группирование атрибутов для минимизации избыточности данных.
- Потенциальные преимущества такой базы данных включают:
  - Обновления данных в БД будут производиться с использованием минимального числа операций, тем самым снижая возможности для возникновения несогласованности данных в БД
  - Снижение размера хранилища данных, следовательно снижение стоимости хранения данных
- Избыточность присутствует в виде внешних ключей, значения которых совпадают со значениями первичных ключей или потенциальных ключей в ссылочных таблицах. Таким образом моделируются взаимосвязи между данными.

7.2. Избыточность данных и аномалии обновления (продолжение)

Staff (staffNo, sName, position, salary, branchNo)
Branch (branchNo, bAddress)
StaffBranch (staffNo, sName, position, salary, branchNo, bAddress)

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

В отношении StaffBranch есть избыточность данных: сведения о филиале повторяются для каждого работника этого филиала.

При работе с отношениями, содержащими избыточные данные, могут возникать проблемы, которые называются *аномалиями обновления* и подразделяются на аномалии *вставки, удаления* и *модификации.*

7.2.1. Аномалии вставки

Тип 1. Дублирование сведений

- При вставке сведений о новых сотрудниках в отношение StaffBranch необходимо указать и сведения об отделении компании, в котором эти сотрудники работают. Например, при вставке сведений о новом сотруднике отделения 'B007' требуется ввести сведения о самом отделении 'B007', которые должны соответствовать сведениям об этом же отделении в других строках отношения StaffBranch.

Тип 2. Невозможность ввода сведений, т. к. появляются пустые значения

- Для вставки сведений о новом отделении компании, которое еще не имеет собственных сотрудников, требуется присвоить значение NULL всем атрибутам описания персонала отношения StaffBranch, включая и табельный номер сотрудника staffNo. Но поскольку атрибут staffNo является первичным ключом отношения StaffBranch, то попытка ввести значение NULL в атрибут staffNo вызовет нарушение целостности сущностей и потому будет отклонена. Следовательно, в отношение StaffBranch невозможно ввести строку о новом отделении компании, содержащую значение NULL в атрибуте staffNo.

Схема с двумя отношениями Branch и Staff избавляет нас от этих аномалий.

7.2.2. Аномалии удаления

- При удалении из отношения StaffBranch строки с информацией о последнем сотруднике некоторого отделения компании сведения об этом отделении будут полностью удалены из базы данных.
- Например, после удаления из отношения StaffBranch строки для сотрудника 'Mary Howe' с табельным номером ' SA9‘ из базы данных неявно будут удалены все сведения об отделении с номером B0071.
- Однако структура отношений Staff и Branch позволяет избежать возникновения этой проблемы, поскольку строки со сведениями об отделениях компании хранятся отдельно от строк со сведениями о сотрудниках. Связывает эти два отношения только общий атрибут branchNo. При удалении из отношения Staff строки с номером сотрудника ' SA9' сведения об отделении 'BO07 ' в отношении Branch останутся нетронутыми.

7.2.3. Аномалии модификации

- При попытке изменения значения одного из атрибутов для некоторого отделения компании в отношении StaffBranch (например, адреса отделения 'B003 ') необходимо обновить соответствующие значения в строках для всех сотрудников этого отделения.
- Если такой модификации будут подвергнуты не все требуемые строки отношения StaffBranch, база данных будет содержать противоречивые сведения.
- В частности, в нашем примере для отделения компании с номером 'B003' в строках, относящихся к разным сотрудникам, ошибочно могут быть указаны разные значения адреса этого отделения.

7.2.3. Аномалии модификации (продолжение)

- Отношение StaffBranch подвержено аномалиям обновления, но этих аномалий можно избежать путем декомпозиции первоначального отношения на отношения Staff и Branch.
- С декомпозицией крупного отношения на более мелкие связаны два важных свойства.
- Свойство 1. Соединение без потерь (lossless-join).
- Гарантирует, что любой экземпляр первоначального отношения может быть определен с помощью соответствующих экземпляров более мелких отношений
- Свойство 2. Сохранение зависимостей (dependency preservation).
- Гарантирует, что ограничения на первоначальное отношение можно поддерживать, просто применяя некоторые ограничения к каждому из более мелких отношений.
- Иными словами, для проверки того, не нарушается ли ограничение, которое распространялось на первоначальное отношение, нет необходимости выполнять операции соединения на более мелких отношениях.

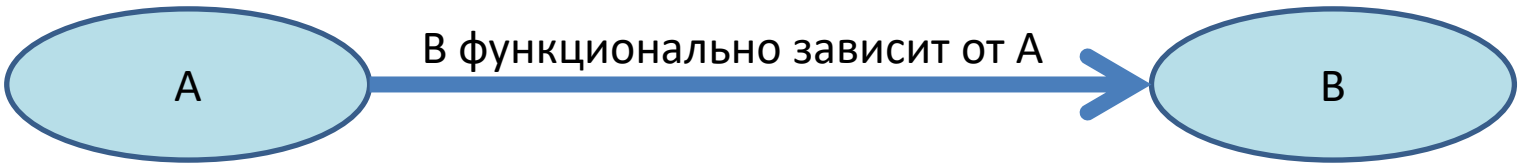
7.3. Функциональные зависимости
7.3.1. Характеристики функциональных зависимостей

- *Функциональная зависимость* (functional dependency) описывает связь между атрибутами и является одним из основных понятий нормализации.
- **Функциональная зависимость.** Описывает связь между атрибутами отношения. Например, если в отношении R, содержащем атрибуты A и B, атрибут B функционально зависит от атрибута A (что обозначается как A –> B), то каждое значение атрибута A связано только с одним значением атрибута B. (Причем атрибуты A и B могут быть и составными.)
- Если нам известно значение атрибута A, то при рассмотрении отношения с такой зависимостью в любой момент времени во всех строках этого отношения, содержащих указанное значение атрибута A, мы найдем одно и то же значение атрибута B. Таким образом, если две строки имеют одно и то же значение атрибута A, то они обязательно имеют одно и то же значение атрибута B. Однако для заданного значения атрибута B может существовать несколько различных значений атрибута A.

7.3.1. Характеристики функциональных зависимостей(продолжение)

- Функциональная зависимость является смысловым (или семантическим) свойством атрибутов отношения.
- Функциональная зависимость может быть транзитивной:

  если A –> B и B –> C, то A –> C.



- **Детерминант.** Детерминантом функциональной зависимости называется атрибут или группа атрибутов, расположенная на диаграмме функциональной зависимости слева от стрелки.
- Атрибут A является детерминантом атрибута B.

7.3.1. Характеристики функциональных зависимостей(продолжение)

Пример функциональной зависимости

- Рассмотрим атрибуты staffNo и position отношения Staff. Зная значение атрибута staffNo (например, 'SL21'), можно определить должность, занимаемую этим сотрудником ('Manager'). Иначе говоря, атрибут position функционально зависит от атрибута staffNo. Однако обратное утверждение неверно, поскольку атрибут staffNo функционально не зависит от атрибута position. Другими словами, каждый сотрудник может занимать только одну должность, однако не исключено, что несколько сотрудников могут иметь одинаковую должность.
- В данном примере атрибут staffNo является детерминантом функциональной зависимости staffNo –> position.

7.3.1. Характеристики функциональных зависимостей(продолжение)

Пример функциональной зависимости (продолжение)

- Еще одна функциональная зависимость staffNo –> sName
- Обратная зависимость sName –> staffNo может выполняться в частных случаях, но в общем случае ее рассматривать не следует, поскольку нельзя исключить вероятность того, что атрибут sName может содержать повторяющиеся значения при наличии в компании сотрудников с одинаковыми именами. Это означает, что при определенных обстоятельствах невозможно будет определить табельный номер сотрудника компании (staffNo) по его имени.
- Таким образом, между атрибутами staffNo и sName имеется связь «один к одному» (1:1), поскольку каждому табельному номеру соответствует только одно имя.
- С другой стороны, между атрибутами sName и staffNo имеется связь «один ко многим» (1:*), поскольку одинаковое имя могут иметь несколько сотрудников компании.

7.3.1. Характеристики функциональных зависимостей(продолжение)

- Для проведения нормализации необходимо прежде всего выявить функциональные зависимости между атрибутами отношения, которые участвуют в связи «один к одному» и остаются справедливыми при любых условиях.
- При выявлении функциональных зависимостей между атрибутами отношения необходимо прежде всего проводить четкое различие между значениями, хранящимися в атрибуте в *определенный момент времени*, и множеством *всех возможных значений*, которые могут храниться в атрибуте в тот или иной момент времени.
- Иными словами, функциональная зависимость является свойством реляционной схемы (т. е. абстрактной структуры), а не свойством конкретного экземпляра схемы (т. е. ее реализации).

7.3.1. Характеристики функциональных зависимостей(продолжение)

- Детерминанты функциональных зависимостей должны иметь *минимальное* число атрибутов, достаточное для того, чтобы поддерживать функциональную зависимость с атрибутами на правой стороне зависимости. Это требование называется **полной функциональной зависимостью**.
- **Полная функциональная зависимость.** Показывает, что, если A и B являются атрибутами какого-то отношения, то B *полностью* функционально зависит от A, если B функционально зависит от A, но не зависит от любого собственного подмножества A.
  ПРИМЕЧАНИЕ. Атрибуты A и B могут быть составными.
- Функциональная зависимость A –> B является **полной функциональной** зависимостью, если удаление какого-либо атрибута из множества A приводит к тому, что зависимость прекращает существовать.
- Функциональная зависимость A –> B является **частичной зависимостью**, если существует какой-то атрибут, который можно удалить из множества A и при этом зависимость сохранится.
- Пример частичной зависимости: staffNo, sName –> branchNo

7.3.1. Характеристики функциональных зависимостей(продолжение)

- В процессе нормализации должны учитываться следующие основные характеристики функциональных зависимостей:
  - определяют связь «один к одному» между атрибутами, приведенными в левой и правой частях выражения зависимости;
  - остаются справедливыми при любых условиях;
  - являются нетривиальными;
  - детерминант имеет минимальное число атрибутов, необходимых для поддержания зависимости с атрибутами правой части (т. е. должна существовать полная функциональная зависимость между атрибутами левой и правой частей зависимости).
- Зависимость называется **тривиальной,** если она не может не выполняться. функциональная зависимость является *тривиальной* тогда и только тогда, когда правая часть ее символической записи является подмножеством (не обязательно строгим подмножеством) левой части. С практической точки зрения подобные зависимости не представляют значительного интереса.
- Примеры тривиальных зависимостей:

  staffNo, sName –> sName

  staffNo, sName --> staffNo

7.3.1. Характеристики функциональных зависимостей(продолжение)

- **Транзитивная зависимость (Transitive dependency).** Это условие, когда A, B и C являются атрибутами отношения (возможно, составными), такими, что выполняются зависимости
  A –> B и B –> C, тогда C будет транзитивно зависеть от A через B (при условии, что A не является функционально зависимым от B или C).

ПРИМЕР.

Для отношения StaffBranch рассмотрим зависимости:

- staffNo –> sName, position, salary, branchNo, bAddress
- branchNo –> bAddress

Транзитивная зависимость branchNo –> bAddress существует для отношения staffNo через branchNo. Другими словами, атрибут staffNo функционально определяет атрибут bAddress через атрибут the branchNo, при этом ни branchNo, ни bAddress функционально НЕ определяют атрибут staffNo.

7.3.2. Выявление функциональных зависимостей

- Выявление функциональных зависимостей основано на знании предметной области.
- **Пример.** Выявление множества функциональных зависимостей для отношения StaffBranch.
- Прежде всего необходимо изучить семантику атрибутов отношения StaffBranch. Например, предположим, что зарплата сотрудника компании зависит от того, какую должность он занимает и в каком отделении работает. Исходя из такой трактовки атрибутов отношения, можно определить следующие функциональные зависимости:
  staffNo –> sName, position, salary, branchNo, bAddress
  branchNo –> bAddress
  bAddress –> branchNo
  branchNo, position –> salary
  bAddress, position –> salary
- Итак, мы определили пять функциональных зависимостей в отношении StaffBranch, детерминантами которых являются staffNo, branchNo, bAddress, (branchNo, position) и (bAddress, position). Применительно к каждой из этих функциональных зависимостей можно гарантировать, что все атрибуты, приведенные в правой части выражения, являются функционально зависимыми от детерминанта, который находится в левой части.

7.3.3. Выявление первичного ключа отношения с использованием функциональных зависимостей

- Выявление множества функциональных зависимостей для отношения осуществляется в целях определения множества ограничений целостности, которые должны распространяться на это отношение. Прежде всего необходимо рассмотреть такое важное ограничение целостности, как определение потенциальных ключей, один из которых должен быть выбран в качестве первичного ключа для отношения. Процесс определения первичного ключа для заданного отношения показан в следующем примере.

7.3.3. Выявление первичного ключа отношения с использованием функциональных зависимостей (продолжение)

ПРИМЕР.

- Чтобы определить потенциальный ключ (ключи) для отношения StaffBranch, необходимо установить, какой атрибут (или группа атрибутов) однозначно идентифицирует каждую строку в этом отношении. Если отношение имеет несколько потенциальных ключей, необходимо установить, какой потенциальный ключ должен применяться в качестве первичного для этого отношения. Все атрибуты, которые не входят в состав первичного ключа (называемые атрибутами, отличными от атрибутов первичного ключа), должны быть функционально зависимыми от этого ключа.
- Единственным потенциальным ключом отношения StaffBranch, и потому единственным первичным ключом, является staffNo, поскольку все прочие атрибуты этого отношения являются функционально зависимыми от атрибута staffNo.
- Несмотря на то что атрибуты branchNo, bAddress, (branchNo, position) и (bAddress, position) являются детерминантами в этом отношении, они не могут служить для него потенциальными ключами.

7.4. Процесс нормализации

- Нормализация — это формальный метод анализа отношений на основе их первичного ключа (или потенциальных ключей) и существующих функциональных зависимостей.
- Если некоторое требование не удовлетворяется, то противоречащее данному требованию отношение должно быть разделено на отношения, каждое из которых (в отдельности) удовлетворяет всем требованиям нормализации.
- Чаще всего нормализация осуществляется в виде нескольких последовательно выполняемых этапов, каждый из которых соответствует определенной нормальной форме, обладающей известными свойствами. В ходе нормализации формат отношений становится все более ограниченным (строгим) и менее восприимчивым к аномалиям обновления.
- Для создания отношений приемлемого качества обязательно только выполнение требований первой нормальной формы (1НФ). Все остальные формы могут использоваться по желанию проектировщиков. Как правило, добиваются третьей нормальной формы (3НФ), чтобы избежать аномалий обновления.
- Нормализация позволяет **избежать избыточности** и, следовательно, возникновения некоторых аномалий обновления.

7.4. Процесс нормализации (продолжение)

- Процесс нормализации заключается в замене данной переменной отношения некоторым набором ее **проекций,** составленным таким образом, чтобы обратное соединение этих проекций позволяло вновь получить исходную переменную отношения. Иначе говоря, этот процесс является обратимым, т. е. декомпозиция всегда выполняется без потерь информации.
- В дальнейших примерах предполагается, что множество функциональных зависимостей уже даны и первичные ключи определены.
- Важно, чтобы к началу выполнения нормализации были уже хорошо понятны значения всех атрибутов и взаимосвязей между ними. Эта информация используется для проверки того, находится ли некоторое отношение в конкретной нормальной форме.

7.5. Ненормализованная форма (ННФ)

- **Ненормализованная форма (ННФ).** Таблица находится в ненормализованной форме, если она содержит одну или несколько повторяющихся групп данных.
- *Повторяющейся группой* называется группа, состоящая из одного или нескольких атрибутов таблицы, в которой возможно наличие нескольких значений для единственного значения ключевого атрибута (атрибутов) таблицы.
- Два способа устранения повторяющихся групп:

1. При первом способе повторяющиеся группы устраняются путем ввода соответствующих данных в пустые столбцы строк с повторяющимися данными. Иначе говоря, пустые места при этом заполняются дубликатами неповторяющихся данных. Этот способ часто называют «выравниванием» («flattening») таблицы.
2. При втором способе один атрибут или группа атрибутов назначаются ключом ненормализованной таблицы, а затем повторяющиеся группы изымаются и помещаются в отдельные отношения вместе с копиями ключа исходной таблицы. Далее в новых отношениях устанавливаются свои первичные ключи.

7.5. Ненормализованная форма (ННФ) (продолжение)

- Иногда ненормализованная таблица может содержать *несколько повторяющихся групп* или включать повторяющиеся группы, содержащиеся в других повторяющихся группах. В таких случаях данный прием применяется до тех пор, пока повторяющихся групп совсем не останется.
- Полученный набор отношений будет находиться в первой нормальной форме только тогда, когда ни в одном из них не будет повторяющихся групп атрибутов.
- Хотя оба эти способа одинаково обоснованы, следует отметить, что при использовании второго подхода полученные отношения находятся как минимум в форме 1НФ и обладают меньшей избыточностью данных. При выборе первого подхода выровненное отношение 1НФ раскладывается в ходе дальнейшей нормализации на те же отношения, которые могли быть сразу же получены с помощью второго подхода.

7.5. Ненормализованная форма (ННФ) (продолжение)

ClientRental

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	50	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-11	10-June-12	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

Повторяющаяся группа = (propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

7.6. Первая нормальная форма (1НФ)

- **Первая нормальная форма (1НФ).** Отношение, в котором на пересечении каждой строки и каждого столбца содержится одно и только одно значение.
- Для преобразования ненормализованной таблицы в первую нормальную форму (1НФ) в исходной таблице следует найти и устранить все повторяющиеся группы данных.

7.6. Первая нормальная форма (1НФ) (продолжение)

Способ 1

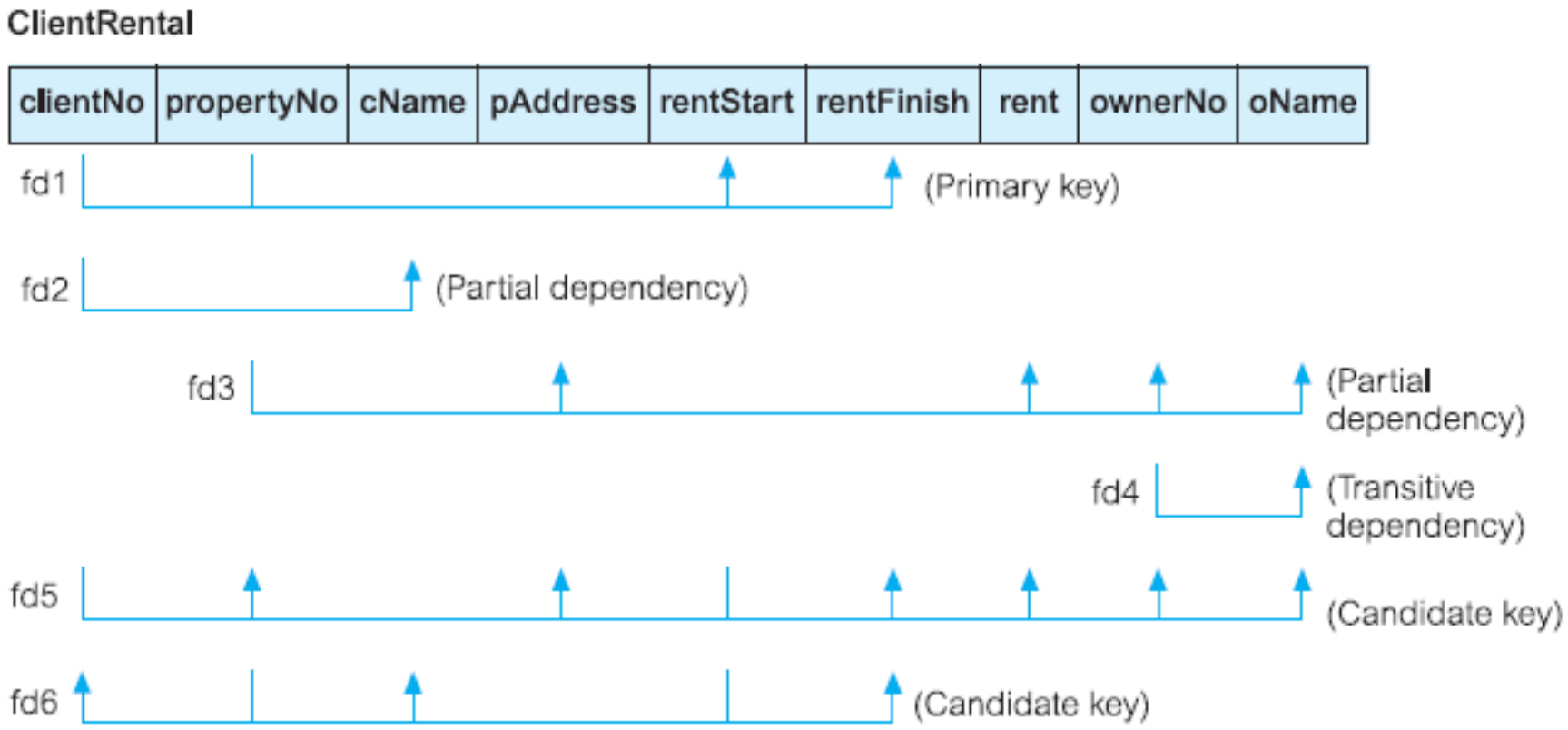
- Повторяющиеся группы устраняются путем ввода соответствующих данных в пустые столбцы строк с повторяющимися данными. Иначе говоря, пустые места при этом заполняются дубликатами неповторяющихся данных.
- Потенциальные ключи этого отношения являются составными: (clientNo, propertyNo), (clientNo, rentstart), (propertyNo, rentstart). В качестве первичного ключа выберем (clientNo, propertyNo) и разместим атрибуты первичного ключа в левой части отношения.

ClientRental

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

7.6. Первая нормальная форма (1НФ) (продолжение)

- Функциональные зависимости для отношения ClientRental



7.6. Первая нормальная форма (1НФ) (продолжение)

Способ 2

- Один атрибут или группа атрибутов назначаются ключом ненормализованной таблицы, а затем повторяющиеся группы изымаются и помещаются в отдельные отношения вместе с копиями ключа исходной таблицы. Далее в новых отношениях устанавливаются свои первичные ключи.

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Client (<u>clientNo</u>, cName)

PropertyRentalOwner (<u>clientNo</u>, <u>propertyNo</u>, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

PropertyRentalOwner

clientNo	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

7.7. Вторая нормальная форма (2НФ)

- Вторая нормальная форма основана на понятии полной функциональной зависимости.
- **Вторая нормальная форма (2НФ).** Отношение, которое находится в первой нормальной форме и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа
- Нормализация отношений 1НФ с приведением к форме 2НФ предусматривает устранение частичных зависимостей.
- Если в отношении между атрибутами существует частичная зависимость, то функционально-зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.
- Вторая нормальная форма применяется к отношениям с составными ключами, т. е. к таким отношениям, первичный ключ которых состоит из двух или нескольких атрибутов. Дело в том, что отношение с первичным ключом на основе единственного атрибута всегда находится, по крайней мере, в форме 2НФ.

7.7. Вторая нормальная форма (2НФ)

Переход к второй нормальной форме (2НФ)

Функциональные зависимости в отношении ClientRental

- fd1 clientNo, propertyNo –> rentStart, rentFinish (первичный ключ)
- fd2 clientNo –> cName (частичная зависимость)
- fd3 propertyNo –> pAddress, rent, ownerNo, oName (частичная зависимость)
- fd4 ownerNo –> oName (транзитивная зависимость)
- fd5 clientNo, rentStart –> propertyNo, pAddress, rentFinish, rent, ownerNo, oName (потенциальный ключ)
- fd6 propertyNo, rentStart –> clientNo, cName, rentFinish (потенциальный ключ)
- После выявления функциональных зависимостей процесс нормализации отношения ClientRental предусматривает проверку его принадлежности ко второй нормальной форме. Для этого требуется найти хотя бы один случай частичной зависимости от первичного ключа. Нетрудно заметить, что атрибут имени клиента cName частично зависит от первичного ключа, иначе говоря, он зависит только от атрибута clientNo (эта зависимость представлена выше как fd2).

7.7. Вторая нормальная форма (2НФ)

Переход к второй нормальной форме (2НФ) (продолжение)

- Кроме того, атрибуты объекта недвижимости (pAddress, rent, ownerNo, oName) также частично зависят от первичного ключа, но на этот раз только от атрибута propertyNo (эта зависимость представлена выше как fd3). В свою очередь, атрибуты арендованных объектов недвижимости (rentstart и rentFinish) полностью функционально зависят от первичного ключа в целом, т.е. от атрибутов clientNo и propertyNo (эта зависимость представлена выше как fdl).
- Обратите внимание на наличие *транзитивной зависимости* (transitive dependence) от первичного ключа (эта зависимость представлена выше как fd4). Хотя транзитивная зависимость также может послужить причиной аномалий обновления, тем не менее ее присутствие в отношении не нарушает ограничений для формы 2НФ. Такие зависимости будут устранены при переходе к форме 3НФ.
- Итак, обнаружены частичные зависимости внутри отношения ClientRental, а это означает, что данное отношение не находится во второй нормальной форме.

7.7. Вторая нормальная форма (2НФ)

Переход к второй нормальной форме (2НФ) (продолжение)

- Для преобразования отношения ClientRental в форму 2НФ необходимо создать новые отношения, причем таким образом, чтобы атрибуты, не входящие в первичный ключ, были перемещены в них вместе с копией той части первичного ключа, с которой эти атрибуты связаны полной функциональной зависимостью. Применение такого процесса в нашем случае приведет к созданию трех новых отношений — Client, Rental и PropertyOwner.

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-12	31-Aug-13
CR76	PG16	1-Sep-13	1-Sep-14
CR56	PG4	1-Sep-11	10-Jun-12
CR56	PG36	10-Oct-12	1-Dec-13
CR56	PG16	1-Nov-14	10-Aug-15

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

Client (clientNo, cName)

Rental (clientNo, propertyNo, rentStart, rentFinish)

PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)

7.8. Третья нормальная форма (3НФ)

- **Третья нормальная форма (3НФ).** Отношение, которое находится в первой и во второй нормальных формах и не имеет атрибутов, не входящих в первичный ключ, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.
- Нормализация отношений 2НФ с образованием отношений 3НФ предусматривает устранение транзитивных зависимостей. Если в отношении существует транзитивная зависимость между атрибутами, то транзитивно зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.
- Хотя отношения 2НФ в меньшей степени обладают избыточностью данных, чем отношения 1НФ, они все еще могут быть подвержены аномалиям обновления. Так, при попытке обновления в отношении PropertyOwner имени владельца недвижимости (например, Tony Shaw с номером C093 (атрибут ownerNo)) потребуется обновить две строки отношения PropertyOwner. Если обновить только одну из этих двух строк, база данных попадет в противоречивое состояние. Эта аномалия обновления вызывается транзитивной зависимостью, присутствующей в данном отношении. Она может быть устранена путем приведения данного отношения к третьей нормальной форме.

7.8. Третья нормальная форма (3НФ) (продолжение)

Переход к третьей нормальной форме (3НФ)

- Нормализация отношений 2НФ с образованием отношений 3НФ предусматривает устранение транзитивных зависимостей. Если в отношении существует транзитивная зависимость между атрибутами, то транзитивно зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.
- Все не входящие в первичный ключ атрибуты отношений Client и Rental функционально зависимы только от их первичных ключей. Следовательно, отношения Client и Rental не имеют транзитивных зависимостей и поэтому они находятся в третьей нормальной форме (3НФ).

7.8. Третья нормальная форма (3НФ) (продолжение)

Переход к третьей нормальной форме (3НФ) (продолжение)

- Все не входящие в первичный ключ атрибуты отношения PropertyOwner функционально зависят от первичного ключа, за исключением атрибута oName, который зависит также и от атрибута ownerNo (зависимость fd4). Это типичный пример транзитивной зависимости, которая имеет место при наличии зависимости не входящего в первичный ключ атрибута (oName) от одного или нескольких других атрибутов, также не входящих в первичный ключ (ownerNo).
- fd3 propertyNo –> pAddress, rent, ownerNo, oNarne (Первичный ключ)
- fd4 ownerNo –> oName (Транзитивная зависимость)
- Для преобразования отношения PropertyOwner в третью нормальную форму необходимо прежде всего удалить упомянутую выше транзитивную зависимость путем создания двух новых отношений PropertyForRent и Owner.

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

7.8. Третья нормальная форма (3НФ) (продолжение)

Переход к третьей нормальной форме (3НФ) (продолжение)

- PropertyForRent (propertyNo, pAddress, rent, ownerNo)
- Owner (ownerNo, oName)

PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

7.8. Третья нормальная форма (3НФ) (продолжение)

Переход к третьей нормальной форме (3НФ) -- результат

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-12	31-Aug-13
CR76	PG16	1-Sep-13	1-Sep-14
CR56	PG4	1-Sep-11	10-Jun-12
CR56	PG36	10-Oct-12	1-Dec-13
CR56	PG16	1-Nov-14	10-Aug-15

PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

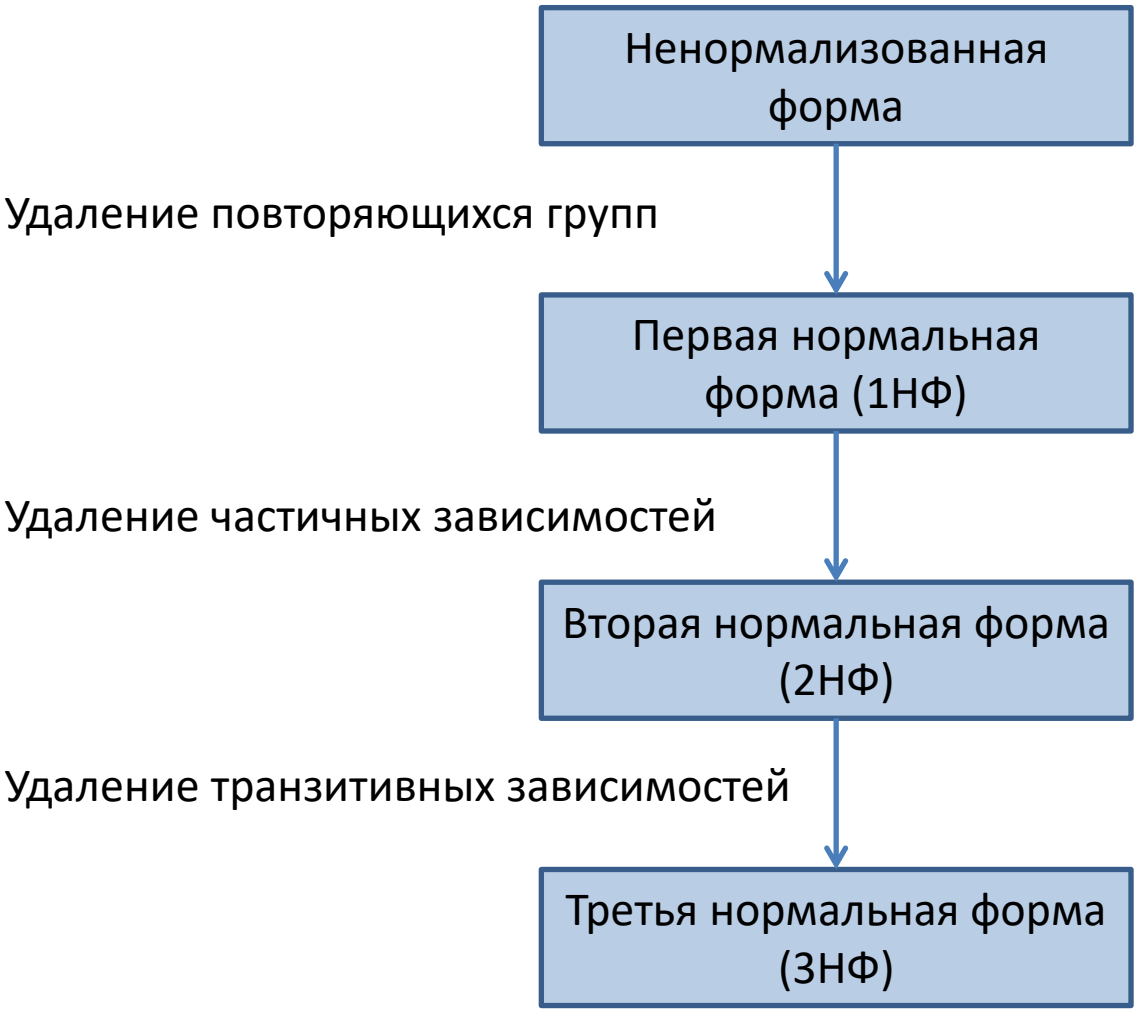
ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

Исходное отношение ClientRental может быть восстановлено путем соединения отношений Client, Rental, PropertyForRent и Owner. Данная цель достигается за счет использования первичных и внешних ключей. Данную процедуру иначе называют декомпозицией *без потерь (lossless).*

7.9. Общее определение второй и третьей нормальных форм

- Определения второй (2НФ) и третьей (3НФ) нормальных форм, приведенные выше, не допускают наличия частичных или транзитивных зависимостей от первичного ключа отношения. В общих определениях форм 2НФ и 3НФ учитываются потенциальные ключи отношения.
- **Вторая нормальная форма (2НФ)**. Отношение, находящееся в первой нормальной форме, в котором каждый атрибут, отличный от атрибута потенциального ключа, является полностью функционально зависимым от какого-либо потенциального ключа.
- **Третья нормальная форма (3НФ)**. Отношение, находящееся в первой и второй нормальной форме, в котором ни один атрибут, отличный от атрибута потенциального ключа, не является транзитивно зависимым ни от одного потенциального ключа.
- На практике чаще всего результаты декомпозиции являются одинаковыми, независимо от того, используются ли определения форм 2НФ и 3НФ, основанные на первичных ключах, или общие определения.

7.10. Общая схема нормализации



7.11. Еще о функциональных зависимостях

- Полный набор функциональных зависимостей для определенного отношения может оказаться слишком большим. Поэтому необходимо найти такой подход, который позволил бы уменьшить размеры этого множества функциональных зависимостей до приемлемого уровня. Необходимо стремиться к тому, чтобы было определено множество функциональных зависимостей (условно обозначенное как X) для отношения, которое меньше, чем полное множество функциональных зависимостей (условно обозначенное как Y) для этого отношения, но обладает тем свойством, что каждая функциональная зависимость в Y следует из функциональных зависимостей в X. Поэтому применение ограничений целостности, определяемых функциональными зависимостями из множества X, влечет за собой автоматическое применение ограничений целостности, определенных в более широком множестве функциональных зависимостей (Y). Из этого требования следует, что должны существовать функциональные зависимости, которые можно вывести из других функциональных зависимостей.

7.11. Еще о функциональных зависимостях (продолжение)

- Например, если в отношении имеются функциональные зависимости A –> B и B –> C, это означает, что в данном отношении соблюдается также функциональная зависимость A –> C. Зависимость A –> C является примером транзитивной функциональной зависимости.
- Процесс определения полезных функциональных зависимостей в отношении, как правило, начинается с определения функциональных зависимостей, которые представляются очевидными с точки зрения их семантики (смысла).
- Но в отношении может также существовать целый ряд других полезных функциональных зависимостей (неочевидных).
- Задача определения всех возможных функциональных зависимостей для «реальных» проектов с базами данных чаще всего является непрактичной.
- Тем не менее, существует подход, позволяющий определить полный набор функциональных зависимостей для отношения, а также существует способ получения минимального набора функциональных зависимостей, способного представить этот полный набор.

7.11.1. Правила Армстронга

- Множество всех функциональных зависимостей, которые могут быть выведены из заданного множества функциональных зависимостей X, называется *замыканием* (**closure**) X и записывается как X^+.
- Для успешной работы необходимо определить ряд правил, позволяющих вычислить X^+ на основе X. Набор правил вывода, называемый *аксиомами Армстронга,* показывает способы вывода новых функциональных зависимостей из заданных.
- Предположим, что A, B и C – подмножества атрибутов отношения R. Аксиомы Армстронга приведены ниже.

1. **Рефлексивность** (**Reflexivity**). Если B – подмножество A, то A –> B.
2. **Дополнение** (**Augmentation**). Если A –> B, то A,C –> B,C.
3. **Транзитивность** (**Transitivity**). Если A –> B и B –> C, то A –> C.

7.11.1. Правила Армстронга (продолжение)

- Каждое из этих трех правил можно обосновать, исходя непосредственно из определения понятия функциональной зависимости.
- Этот набор правил является полным (**complete**). Это означает, что если задано множество X функциональных зависимостей, то все функциональные зависимости, производные от X, можно вывести из X с помощью только этих правил.
- Такие правила являются также непротиворечивыми (**sound**), поскольку они не позволяют вывести какие-либо дополнительные функциональные зависимости, которые не следовали бы из X.
- Иными словами, эти правила могут применяться для получения замыкания X^+.

7.11.1. Правила Армстронга (продолжение)

- На основе трех правил, приведенных выше, можно вывести несколько дополнительных правил, позволяющих упростить практическую задачу вычисления X^+.
- Допустим, что D — еще одно подмножество атрибутов отношения R, и сформулируем следующие правила.

4. **Самоопределение** (**Self-determination**). A –> A.
5. **Декомпозиция** (**Decomposition**). Если A –> B,C, то A –> B и A –> C.
6. **Объединение** (**Union**). Если A –> B и A –> C, то A –> B,C.
7. **Композиция** (**Composition**). Если A –> B и C –> D, то A,C –> B,D.

7.11.1. Правила Армстронга (продолжение)

- Правило 1 (рефлексивность) и правило 4 (самоопределение) указывают, что множество атрибутов всегда определяет любое из своих подмножеств или само себя. Поскольку с помощью этих правил вырабатываются функциональные зависимости, которые всегда справедливы, они являются тривиальными и, как указано выше, обычно не представляют интереса или не позволяют узнать ничего нового.
- Правило 2 (дополнение) указывает, что добавление одного и того же множества атрибутов и к левой, и к правой частям зависимости приводит к получению еще одной действительной зависимости.
- Правило 3 (транзитивность) указывает, что функциональные зависимости являются транзитивными.

7.11.1. Правила Армстронга (продолжение)

- Правило 5 (декомпозиция) определяет, что можно удалять атрибуты из правой части зависимости. Повторное применение этого правила позволяет разложить функциональную зависимость A –> B,C,D на ряд функциональных зависимостей A –> B, A –> C и A –> D.
- Правило 6 (объединение) указывает, что в процессе проектирования может быть выполнена обратная операция, при которой ряд зависимостей A –> B, A –> C и A –> D объединяется в одну функциональную зависимость A –> B,C,D.
- Правило 7 (композиция) является более общим, чем правило 6, и указывает, что для получения еще одной действительной зависимости может объединяться ряд неперекрывающихся зависимостей.

7.11.1. Правила Армстронга (продолжение)

Методика определения набора функциональных зависимостей

- Все действия по определению набора функциональных зависимостей F для отношения, как правило, начинаются с выявления зависимостей, которые можно определить, исходя из семантики (смысла) атрибутов отношения.
- Затем применяются аксиомы Армстронга (правила 1–3) для вывода дополнительных функциональных зависимостей, которые также являются справедливыми для этого отношения.
- Систематический способ определения таких дополнительных функциональных зависимостей состоит в том, что вначале определяется каждое множество атрибутов A, которое присутствует в левой части некоторых функциональных зависимостей, а затем определяется множество всех атрибутов, зависимых от A.
- Поэтому для каждого множества атрибутов A можно определить множество A^+ атрибутов, функционально определяемых из A на основе F.
- A^+ называется **замыканием A в соответствии с F** (**the closure of A under F**).

7.11.2. Минимальное множество функциональных зависимостей

- В этом разделе дано определение понятия *эквивалентности* множеств функциональных зависимостей.
- Множество функциональных зависимостей Y *покрывается* множеством функциональных зависимостей X, если каждая функциональная зависимость из Y присутствует также в замыкании X^+; иными словами, каждая функциональная зависимость во множестве Y может быть выведена из X.
- Множество функциональных зависимостей X является минимальным, если оно удовлетворяет следующим условиям.

1. Каждая зависимость в X имеет единственный атрибут в правой части.
2. Ни одну зависимость A –> B в X нельзя заменить зависимостью C –> B, где C является собственным подмножеством A, и получить в результате множество зависимостей, эквивалентное X.

7.11.2. Минимальное множество функциональных зависимостей (продолжение)

3. Из множества X нельзя удалить ни одной зависимости и получить в результате множество зависимостей, эквивалентное X.

- Минимальное множество зависимостей должно быть представлено в стандартной форме, не допускающей избыточности. Минимальным покрытием множества функциональных зависимостей X называется минимальное множество зависимостей X_{min}, эквивалентное X. К сожалению, может существовать несколько разных минимальных покрытий для множества функциональных зависимостей.

7.11.2. Минимальное множество функциональных зависимостей (продолжение)

ПРИМЕР.

- В результате применения трех условий, описанных выше, к множеству функциональных зависимостей для отношения StaffBranch получено следующее множество функциональных зависимостей:

  staffNo –> sName

  staffNo –> position

  staffNo –> salary

  staffNo –> branchNo

  branchNo –> bAddress

  bAddress –> branchNo

  branchNo, position –> salary

  bAddress, position –> salary

7.11.2. Минимальное множество функциональных зависимостей (продолжение)

ПРИМЕР (продолжение).

- Эти функциональные зависимости удовлетворяют всем трем условиям получения минимального множества функциональных зависимостей и применяются к отношению StaffBranch.
- Условие 1 гарантирует, что каждая зависимость определена в стандартной форме с единственным атрибутом в правой части.
- Условия 2 и 3 гарантируют, что в зависимостях нет избыточности, поскольку в них либо удалены избыточные атрибуты в левой части зависимости (согласно условию 2), либо исключены зависимости, которые могут быть выведены из остальных функциональных зависимостей множества X (согласно условию 3).

7.12. Нормальная форма Бойса–Кодда (НФБК)

- Применение общих определений 2НФ и 3НФ может позволить выявить дополнительную избыточность, вызванную зависимостями от всех потенциальных ключей.
- Но даже после ввода этих дополнительных ограничений в отношениях все еще могут существовать зависимости, которые приводят к появлению избыточности в отношениях 3НФ.
- С учетом этого недостатка третьей нормальной формы была разработана более строгая нормальная форма, получившая название нормальной формы Бойса–Кодда (НФБК).
- Нормальная форма Бойса–Кодда (НФБК) основана на функциональных зависимостях, в которых учитываются все потенциальные ключи отношения. Кроме того, в НФБК предусмотрены более строгие ограничения по сравнению с общим определением 3НФ.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

- **Нормальная форма Бойса–Кодда (НФБК).** Отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом.
- Для проверки принадлежности отношения к НФБК необходимо найти все его детерминанты и убедиться в том, что они являются потенциальными ключами.
- Детерминантом является один атрибут или группа атрибутов, от которой полностью функционально зависит другой атрибут.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

- Различие между ЗНФ и НФБК заключается в том, что функциональная зависимость A –> B допускается в отношении ЗНФ, если B является атрибутом первичного ключа, а атрибут A не является потенциальным ключом. Тогда как в отношении НФБК эта зависимость допускается *только* тогда, когда атрибут A является потенциальным ключом.
- Следовательно, нормальная форма Бойса–Кодда является более строгой версией ЗНФ, поскольку каждое отношение НФБК является также отношением ЗНФ, но не всякое отношение ЗНФ является отношением НФБК.
- Нарушения требований НФБК происходят крайне редко, поскольку это может случиться только при следующих условиях:
  - в отношении имеются два (или несколько) составных потенциальных ключа или
  - эти потенциальные ключи перекрываются, т. е. ими совместно используется, по крайней мере, один общий атрибут.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

ПРИМЕР.

- В этом примере учебный проект *DreamHome* дополнен и в него включено описание собеседований сотрудников компании с клиентами. Информация, относящаяся к этим собеседованиям, приведена в отношении clientInterview.
- Сотрудникам компании, проводящим собеседования с клиентами, в этот день предоставляется специальное помещение. Однако в течение рабочего дня это помещение может использоваться несколькими разными сотрудниками.
- С клиентом проводится только одно собеседование в день, но он может участвовать в нескольких собеседованиях в разные дни.

ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13-May-14	10.30	SG5	G101
CR56	13-May-14	12.00	SG5	G101
CR74	13-May-14	12.00	SG37	G102
CR56	1-Jul-14	10.30	SG5	G102

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

Продолжение примера.

- Функциональные зависимости для отношения ClientInterview

  fd1 clientNo, interviewDate –> interviewTime, staffNo, roomNo (Первичный ключ)

  fd2 staffNo, interviewDate, interviewTime –> clientNo (Потенциальный ключ)

  fd3 roomNo, interviewDate, interviewTime –> staffNo, clientNo (Потенциальный ключ)

  fd4 staffNo, interviewDate –> roomNo

- Отношение ClientInterview обладает тремя составными потенциальными ключами, которые перекрываются, поскольку в них совместно используется один общий атрибут — interviewDate.
- ВОПРОС: в какой нормальной форме находится отношение ClientInterview?
- Поскольку функциональные зависимости fdl, fd2 и fd3 являются потенциальными ключами этого отношения, то они не вызовут никаких проблем.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

Продолжение примера.

- Нам потребуется рассмотреть только функциональную зависимость staffNo, interviewDate –> roomNo (зависимость fd4).
- Даже если комбинация атрибутов (staffNo, interviewDate} не является потенциальным ключом отношения ClientInterview, эта функциональная зависимость в 3НФ допускается, поскольку атрибут roomNo является частью потенциального ключа (roomNo, interviewDate, interviewTime).
- Так как в этом отношении нет никаких частичных или транзитивных зависимостей от первичного ключа (clientNo, interviewDate) и допускается наличие функциональной зависимости fd4, можно считать, что отношение ClientInterview находится в форме 3НФ.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

Продолжение примера.

- Однако это отношение не находится в форме НФБК, поскольку в нем присутствует детерминант (staffNo, interviewDate), который не является потенциальным ключом этого отношения. В форме НФБК требуется, чтобы все детерминанты отношения были его потенциальными ключами.
- Вследствие этого отношение ClientInterview может быть подвержено аномалиям обновления. Например, при изменении номера комнаты, выделенной сотруднику 'SG5' на 13 мая 2014 года, потребуется обновить значения в двух строках. Если при этом будет обновлена только одна строка, то база данных перейдет в противоречивое состояние.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

Продолжение примера.

- Для преобразования отношения ClientInterview в форму НФБК необходимо устранить нарушающую это ограничение функциональную зависимость путем создания двух новых отношений — Interview и StaffRoom.

Interview

clientNo	interviewDate	interviewTime	staffNo
CR76	13-May-14	10.30	SG5
CR56	13-May-14	12.00	SG5
CR74	13-May-14	12.00	SG37
CR56	1-Jul-14	10.30	SG5

StaffRoom

staffNo	interviewDate	roomNo
SG5	13-May-14	G101
SG37	13-May-14	G102
SG5	1-Jul-14	G102

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

- Любое отношение, которое не находится в форме НФБК, можно преобразовать в отношения НФБК. Тем не менее преобразование отношений в форму НФБК не всегда приводит к желаемым результатам. Например, иногда после такой декомпозиции не сохраняется важная функциональная зависимость.
- Например, при создании двух новых отношений НФБК на основе исходного отношения ClientInterview «утрачивается» следующая функциональная зависимость: roomNo, interviewDate, interviewTime –> staffNo, clientNo (зависимость fd3), поскольку детерминант этой зависимости больше не будет находиться в том же отношении, что и определяемые им атрибуты.

7.12. Нормальная форма Бойса–Кодда (НФБК) (продолжение)

- Например, если всегда соблюдается условие, что сотрудники компании проводят в день только одно собеседование, то наличие зависимости fd4 в отношении ClientInterview не вызывает избыточности и поэтому декомпозиция этого отношения на два отношения НФБК не требуется.
- С другой стороны, если сотрудники компании проводят в день несколько собеседований, то наличие зависимости fd4 в отношении ClientInterview вызывает избыточность и можно порекомендовать нормализацию этого отношения до формы НФБК.
- Однако следует также рассмотреть, насколько значимой является потеря зависимости fd3.

7.13. Четвертая нормальная форма (4НФ)

- НФБК позволяет устранить любые аномалии, вызванные функциональными зависимостями.
- Однако в результате теоретических исследований был выявлен еще один тип зависимости — *многозначная зависимость* (Multi-Valued Dependency — MVD), которая при проектировании отношений также может вызвать проблемы, связанные с избыточностью данных.
- Возможность существования в отношении многозначных зависимостей возникает вследствие приведения исходных таблиц к форме 1НФ, для которой не допускается наличие некоторого набора значений на пересечении одной строки и одного столбца.

7.13. Четвертая нормальная форма (4НФ) (продолжение)

- Например, при наличии в отношении двух многозначных атрибутов для достижения непротиворечивого состояния строк необходимо повторить в них каждое значение одного из атрибутов в сочетании с каждым значением другого атрибута.
- Проблемы, которые связаны с переменными отношения в НФБК были замечены достаточно давно, и способы их разрешения также были вскоре после этого определены, по крайней мере, на интуитивном уровне.
- Однако эти идеи были сформулированы Фейгином (Fagin) в строгом теоретическом виде с использованием понятия **многозначной зависимости (МЗЗ)** только в 1977 году.

7.13. Четвертая нормальная форма (4НФ) (продолжение)

- Рассмотрим отношение BranchStaffOwner, в котором содержатся имена сотрудников (sName) и владельцев недвижимости (oName) определенного отделения компании (branchNo). В этом случае предположим, что имя сотрудника (sName) однозначно идентифицирует каждого члена персонала компании, а имя владельца (oName) однозначно идентифицирует каждого владельца.

BranchStaffOwner

branchNo	sName	oName
B003	Ann Beech	Carol Farrel
B003	David Ford	Carol Farrel
B003	Ann Beech	Tina Murphy
B003	David Ford	Tina Murphy

7.13. Четвертая нормальная форма (4НФ) (продолжение)

- Будем считать, что в данном отделении компании между сотрудниками и владельцами недвижимости нет никакой прямой связи. Поэтому необходимо создать строку для каждого сочетания данных о сотруднике и владельце для обеспечения гарантии, что отношение находится в непротиворечивом состоянии. Это требование отражает наличие в отношении BranchStaffOwner многозначной зависимости. Иначе говоря, в данном отношении существует многозначная зависимость, так как в нем содержатся две независимые связи типа 1:*.
- **Многозначная зависимость** . Представляет такую зависимость между атрибутами отношения (например, A, B и C), что для каждого значения A существует множество значений для B и множество значений для C. Однако множества значений для B и C не зависят друг от друга.
- Многозначные зависимости, имеющие место в отношении BranchStaffOwner (обозначается символом –>>):

  branchNo –>> sName

  branchNo –>> oName

7.13. Четвертая нормальная форма (4НФ) (продолжение)

- Многозначная зависимость может быть дополнительно определена как *тривиальная* или *нетривиальная.*
- Многозначная зависимость A –>> B некоторого отношения R определяется как тривиальная при выполнении одного из двух условий: либо B является подмножеством A, либо A ∪ B = R.
- Многозначная зависимость определяется как нетривиальная, если ни то ни другое условие не выполняется.
- Тривиальная многозначная зависимость не накладывает никаких ограничений на данное отношение, а нетривиальная — накладывает.
- Многозначная зависимость (МЗЗ) в представленном в отношении BranchStaffOwner является нетривиальной, так как в этом отношении ни то ни другое условие не удовлетворяется. Следовательно, на отношение BranchStaffOwner по причине наличия нетривиальной МЗЗ накладывается ограничение, которое приводит к появлению повторяющихся строк, необходимых для того, чтобы гарантировать непротиворечивость связи между атрибутами sName и oName.

7.13. Четвертая нормальная форма (4НФ) (продолжение)

- Например, если бы потребовалось зарегистрировать нового владельца недвижимости в отделении B003, то пришлось бы создать две новые строки, по одной для каждого сотрудника компании, чтобы обеспечить сохранение непротиворечивого состояния указанного отношения. Это — пример аномалии обновления, вызванной наличием нетривиальной многозначной зависимости.
- Хотя отношение BranchStaffOwner находится в форме НФБК, оно все еще остается плохо структурированным из-за избыточности данных, вызванной наличием нетривиальной МЗЗ.
- Очевидно, что целесообразнее было бы определить более строгую форму, чем НФБК, которая исключила бы создание таких реляционных структур, как отношение BranchStaf fOwner.
- **Четвертая нормальная форма (4НФ).** Отношение находится в 4НФ тогда и только тогда, когда для каждой нетривиальной многозначной зависимости A –>> B детерминант A является потенциальным ключом отношения.

7.13. Четвертая нормальная форма (4НФ) (продолжение)

- Для нормализации отношения, нарушающего 4НФ, требуется удалить многозначную зависимость из этого отношения. С этой целью нужно создать новое отношение и переместить в него атрибуты, которые участвуют в многозначной зависимости, а также поместить в него копии атрибутов детерминанта.
- Отношение BranchStaf fOwner следует преобразовать в отношения BranchStaff и BranchOwner. Оба новых отношения находятся в форме 4НФ, поскольку отношение BranchStaff содержит тривиальную М33 branchNo –≫ sName, а отношение BranchOwner – тривиальную М33 branchNo –≫ oName. Теперь возможность появления аномалий обновления исключена. Например, чтобы зарегистрировать в отделении B003 нового владельца объекта недвижимости, достаточно ввести единственную строку в отношение BranchOvmer.

BranchStaff

branchNo	sName
B003	Ann Beech
B003	David Ford

BranchOwner

branchNo	oName
B003	Carol Farrel
B003	Tina Murphy

7.13. Четвертая нормальная форма (4НФ) (продолжение)

Еще один пример.

- Соответствующий курс может читать любой из указанных преподавателей с использованием любого из указанных учебников.

HCTX

COURSE	TEACHERS	TEXTS
Physics	TEACHER Prof. Green Prof. Brown	TEXT Basic Mechanics Principles of Optics
Math	TEACHER Prof. Green	TEXT Basic Mechanics Vector Analysis Trigonometry

7.13. Четвертая нормальная форма (4НФ) (продолжение)

Продолжение примера.

- Исходное отношение разделено на два отношения.

CTX

COURSE	TEACHER	TEXT
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principles of Optics
Math	Prof. Green	Basic Mechanics
Math	Prof. Green	Vector Analysis
Math	Prof. Green	Trigonometry

CT

COURSE	TEACHER
Physics	Prof. Green
Physics	Prof. Brown
Math	Prof. Green

CX

COURSE	TEXT
Physics	Basic Mechanics
Physics	Principles of Optics
Math	Basic Mechanics
Math	Vector Analysis
Math	Trigonometry

7.14. Пятая нормальная форма (5НФ)

- При любой декомпозиции отношения на два других отношения полученные отношения обладают свойством соединения без потерь. Это означает, что полученные отношения можно снова соединить и получить прежнее отношение в исходном виде. Однако бывают случаи, когда требуется выполнить декомпозицию отношения более чем на два отношения. В таких (достаточно редких) случаях возникает необходимость учитывать зависимость соединения без потерь, которая устраняется с помощью пятой нормальной формы (5НФ).
- **Зависимость соединения без потерь (Lossless-join dependency).** Свойство декомпозиции, которое гарантирует отсутствие фиктивных строк при восстановлении первоначального отношения с помощью операции естественного соединения.
- **Зависимость соединения (Join dependency)**. Представляет собой одну из разновидностей зависимости. Например, если рассматривается отношение R с подмножествами атрибутов R, обозначенными как A, B, ..., Z, то отношение R удовлетворяет зависимости соединения, если и только если каждое допустимое значение R равно соединению его проекций по атрибутам A, B, ..., Z.

7.14. Пятая нормальная форма (5НФ) (продолжение)

ПРИМЕР. Взят из статьи: William Kent. A Simple Guide to Five Normal Forms in Relational Database Theory // Communications of the ACM 26(2), Feb. 1983, 120-125.

- В примере участвуют агенты, компании и продукты. Если агенты представляют какие-то компании, компании изготавливают некоторые продукты, а агенты продают продукты , тогда мы, возможно, захотели бы фиксировать записи о том, какой агент продал тот или иной продукт, произведенный той или иной компанией. Эта информация могла бы храниться в одном отношении с тремя атрибутами:

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	GM	truck

Эта форма нужна в общем случае. Например, хотя агент Smith продает автомобили, сделанные компанией Ford, и грузовики, сделанные компанией GM, но он не продает грузовики, сделанные компанией Ford, или автомобили, сделанные компанией GM. Поэтому нам нужна комбинация всех трех атрибутов, чтобы узнать, какие комбинации имеют место, а какие нет.

7.14. Пятая нормальная форма (5НФ) (продолжение)

ПРОДОЛЖЕНИЕ ПРИМЕРА.

- Добавим в таблицу больше данных.
- Предположим, что имеет место следующее правило: если агент продает определенный продукт, и этот агент представляет компанию, производящую этот продукт, тогда он продает данный продукт этой компании.

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car

7.14. Пятая нормальная форма (5НФ) (продолжение)

ПРОДОЛЖЕНИЕ ПРИМЕРА.

- В этом случае оказывается, что мы сможем реконструировать все истинные факты, используя нормализованную форму, состоящую из трех отдельных отношений, каждое из которых содержит по два атрибута.

AGENT	COMPANY
Smith	Ford
Smith	GM
Jones	Ford

COMPANY	PRODUCT
Ford	car
Ford	truck
GM	car
GM	truck

AGENT	PRODUCT
Smith	car
Smith	truck
Jones	car

7.14. Пятая нормальная форма (5НФ) (продолжение)

ПРОДОЛЖЕНИЕ ПРИМЕРА.

- Эти три отношения находятся в 5НФ, а исходное отношение – нет.
- Упрощенно, мы можем сказать, что отношение находится в 5НФ, когда его информационное содержание нельзя восстановить на основе нескольких более мелких отношений, т. е. отношений, каждое из которых имеет меньше атрибутов, чем исходное отношение. При этом случай, когда все эти более мелкие отношения имеют один и тот же ключ, исключается.
- Если отношение можно разбить только на такие более мелкие отношения, все из которых имеют один и тот же ключ, тогда считается, что исходное отношение уже находится в 5НФ и без декомпозиции.
- Отношение, находящееся в 5НФ, находится также в 4НФ, 3НФ, 2НФ, 1НФ.
- Если бы в нашем примере не существовало правила относительно агентов, компаний и продуктов, тогда исходное отношение сразу было бы в 5НФ.

7.14. Пятая нормальная форма (5НФ) (продолжение)

ПРОДОЛЖЕНИЕ ПРИМЕРА.

- Одно из преимуществ 5НФ заключается в том, что может быть устранена некоторая избыточность. В нормализованной форме тот факт, что Smith продает автомобили, записан только один раз, а в ненормализованной форме он может быть повторен много раз.
- Важно, что для восстановления информации требуются все три отношения, полученные в результате нормализации. Например, из первых двух из них мы узнаем, что Jones представляет компанию Ford, и что компания Ford делает грузовики. Но мы не сможем определить, продает ли Jones грузовики компании Ford, до тех пор пока не посмотрим на третье отношение, чтобы определить, продает ли вообще Jones грузовики.

7.15. Общая схема процедуры нормализации

1. Переменную отношения в 1НФ следует разбить на такие проекции, которые позволят исключить все функциональные зависимости, не являющиеся неприводимыми. В результате будет получен набор переменных отношения в 2НФ.
2. Полученные переменные отношения в 2НФ следует разбить на такие проекции, которые позволят исключить все существующие транзитивные функциональные зависимости. В результате будет получен набор переменных отношения в 3НФ.
3. Полученные переменные отношения в 3НФ следует разбить на проекции, позволяющие исключить все оставшиеся функциональные зависимости, в которых детерминанты не являются потенциальными ключами. В результате такого приведения будет получен набор переменных отношения в НФБК.

- *Примечание.* Правила 1—3 могут быть объединены в одно: «Исходную переменную отношения следует разбить на проекции, позволяющие исключить все функциональные зависимости, в которых детерминанты не являются потенциальными ключами».

7.15. Общая схема процедуры нормализации (продолжение)

4. Полученные переменные отношения в НФБК следует разбить на проекции, позволяющие исключить все многозначные зависимости, которые не являются также функциональными. В результате будет получен набор переменных отношения в 4НФ.

- *Примечание.* На практике такие многозначные зависимости обычно исключаются *перед* выполнением этапов 1-3.

5. Полученные переменные отношения в 4НФ следует разбить на проекции, позволяющие исключить все зависимости соединения, которые не определяются потенциальными ключами (хотя в данном случае в определение следовало бы добавить фразу «если удастся их выявить»). В результате будет получен набор переменных от ношения в 5НФ.

(К. Дейт)

7.16. Полезные ссылки

- William Kent. A Simple Guide to Five Normal Forms in Relational Database Theory // Communications of the ACM. – 1983. – 26(2), Feb. – P. 120–125. http://www.bkent.net/Doc/simple5.htm

Литература

1. Гарсиа-Молина, Г. Системы баз данных. Полный курс : пер. с англ. / Гектор Гарсиа-Молина, Джеффри Ульман, Дженнифер Уидом. – М. : Вильямс, 2003. – 1088 с.
2. Грофф, Дж. SQL. Полное руководство : пер. с англ. / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. – 3-е изд. – М. : Вильямс, 2015. – 960 с.
3. **Дейт, К. Дж. Введение в системы баз данных : пер. с англ. / Крис Дж. Дейт. – 8-е изд. – М. : Вильямс, 2005. – 1328 с.**
4. **Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика : пер. с англ. / Томас Коннолли, Каролин Бегг. – 3-е изд. – М. : Вильямс, 2003. – 1436 с.**
5. Кузнецов, С. Д. Основы баз данных : учеб. пособие / С. Д. Кузнецов. – 2-е изд., испр. – М. : Интернет-Университет Информационных Технологий ; БИНОМ. Лаборатория знаний, 2007. – 484 с.
6. Лузанов, П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Лёвшин ; Postgres Professional. – М., 2017. – 146 с.
7. Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие. / Е. П. Моргунов ; под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с.
8. PostgreSQL [Электронный ресурс] : официальный сайт / The PostgreSQL Global Development Group. – http://www.postgresql.org.
9. Postgres Professional [Электронный ресурс] : российский производитель СУБД Postgres Pro : официальный сайт / Postgres Professional. – http://postgrespro.ru.

Задание

Для выполнения практических заданий необходимо использовать книгу:

Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие / Под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с. https://postgrespro.ru/education/books/sqlprimer

1. Изучить материал глав 9–10. Запросы к базе данных выполнять с помощью утилиты psql, описанной в главе 2, параграф 2.2.