МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ
(МИИТ)

Кафедра Математического обеспечения
автоматизированных систем управления

М.А. ДАВЫДОВСКИЙ

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Рекомендовано редакционно-издательским советом университета в
качестве методических указаний
для студентов специальностей
«Программное обеспечение» и
«Администрирование информационных систем»

МОСКВА - 2008

УДК 681.3.06
Д13

Давыдовский М.А. Проектирование реляционных баз данных.: Методические указания к лабораторным работам. – М.:МИИТ, 2008– 32 с.

В работе рассматриваются вопросы построения схемы базы данных реляционной модели. Описаны основные понятия реляционной модели базы данных, даны определения нормальных форм отношений, приведены алгоритмы проектирования реляционной схемы базы данных.

Подписано к печати –	Изд №	Формат – 60x84/16
Усл.-п. л. –	Заказ №	Тираж – 150 экз

127994, Москва, ул. Образцова, 15
Типография МИИТа

Введение

Первым этапом построения базы данных является разработка ее концептуальной схемы. Схема базы данных строится на основе анализа предметной области, выделения основных сущностей предметной области их характеристик, зависимостей, которые существуют между различными характеристиками и сущностями предметной области. Большинство из современных систем управления базами данных работают с реляционной моделью данных. База данных не должна содержать избыточной информации, быть непротиворечивой, удовлетворять потребностям многих пользователей, позволять получать из базы данных ответы на запросы за приемлимое время. Для этих целей необходимо уметь правильно построить схему реляционной модели базы данных.

Реляционная модель позволяет определить, что одна схема базы данных лучше другой и как построить оптимальную схему базы данных.

В данном методическом указании определены основные понятия реляционной модели базы данных, функциональные и многозначные зависимости, нормальные формы отношений, дана постановка задачи проектирования реляционной схемы базы данных и описаны алгоритмы проектирования реляционной схемы базы данных.

1. Основные понятия реляционной модели базы данных

База данных представляет собой модель некоторой предметной области. При описании предметной области мы рассматриваем классы объектов, свойства этих объектов и связи между объектами.

Атрибут. Свойства объектов и связей в реляционной модели базы данных описываются атрибутами. Каждый атрибут имеет уникальное имя в базе данных. Например, атрибут ФИО_СТУДЕНТА. Атрибут может иметь некоторое значение, например «Иванов И.И.»

Домен. Атрибут принимает значения из множества допустимых значений, которое называется доменом. Каждый домен имеет имя. Например, ФАМИЛИЯ. Различные атрибуты могут быть определены на одном и том же домене. Например, атрибуты ФИО_СТУДЕНТА и ФИО_ПРЕПОДАВАТЕЛЯ определены на домене ФАМИЛИЯ.

Схема отношения. Классы объектов и связи между объектами предметной области описываются в реляционной модели некоторым множеством имен атрибутов $\{A_1, A_2, \ldots, A_n\}$, называемым схемой

отношения. Схема отношения обозначается как $R(A_1, A_2, ..., A_n)$, где R - это имя отношения. Количество атрибутов в схеме отношения называется степенью отношения. Множество всех атрибутов схемы отношения R обозначается U_R.

Например, класс объектов СТУДЕНТ описывается схемой отношения
СТУДЕНТ(НОМЕР_ЗАЧЕТКИ, ФИО_СТУДЕНТА, АДРЕС). Связь ЭКЗАМЕН между объектами классов ГРУППА, ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА описывается схемой отношения ЭКЗАМЕН (НОМЕР_ГРУППЫ, НОМЕР_УДОСТОВЕРЕНИЯ, НОМЕР _ДИСЦИПЛИНЫ, ДАТА, АУДИТОРИЯ).

Кортеж отношения. Набор значений атрибутов $(a_1, a_2, ..., a_n)$ схемы отношения $R(A_1, A_2, ..., A_n)$ называется кортежем. Некоторое a_i – это значение атрибута A_i. Например, кортеж (342, Иванов И.И., Москва ул.Строителей 12-15) отношения СТУДЕНТ.

Отношение. Множество кортежей вида $(a_1, a_2, ..., a_n)$ схемы отношения $R(A_1, A_2, ..., A_n)$ называется отношением. Если атрибут A_i определен на домене D_i, то отношение со схемой R - это подмножество декартова произведения доменов $D_1 \times D_2 \times ... \times D_n$

Ключ отношения. Ключ отношения со схемой R - это некоторое подмножество атрибутов $K = \{B_1, B_2, ..., B_n\}$, $K \subseteq U_R$, по значениям которых можно однозначно определить кортеж отношения. При этом никакое собственное подмножество ключа $K' \subseteq K$ этим свойством не обладает. Другими словами, в отношении не может быть двух кортежей с одинаковыми значениями из множества атрибутов K. Множество K называется ***суперключом***, если оно содержит в себе ключ.

Например, ключом отношения СТУДЕНТ является НОМЕР_ЗАЧЕТКИ, а суперключом – НОМЕР_ЗАЧЕТКИ, АДРЕС. Ключом отношения ЭКЗАМЕН является подмножество атрибутов НОМЕР_ГРУППЫ, НОМЕР_ДИСЦИПЛИНЫ в предположении, что экзамен принимает один преподаватель.

Одно отношение может иметь несколько ключей. Например, в схему отношения СТУДЕНТ можно добавить атрибут НОМЕР_ПАСПОРТА_СТУДЕНТА, который также будет являться ключом отношения СТУДЕНТ.

Атрибуты, которые принадлежат хотя бы одному из ключей, называются ***ключевыми*** или ***первичными атрибутами***. Атрибуты, которые не принадлежат ни одному из ключей, называются ***неключевыми*** или ***непервичными атрибутами***.

Ключ отношения не может меняться при изменении самого отношения, т.е. при изменении кортежей отношения. Определяя ключи отношения, мы тем самым накладываем некоторые ограничения целостности на базу данных.

Схема реляционной базы данных – это множество схем отношений**.**

Реляционной базой данных называют множество отношений.

2. Функциональные зависимости

Говорят, что в отношении со схемой $R(A_1, A_2, ..., A_n)$ существует ***функциональная зависимость*** $X \to Y$ между подмножествами атрибутами $X \subseteq U_R$ и $Y \subseteq U_R$ (Y функционально зависит от X), если для любой реализации этого отношения не существует двух кортежей с одинаковыми значениями из множества X и разными значениями из множества Y.

Функциональные зависимости определяются на основе семантики атрибутов, так как нет возможности проверить все реализации отношения. Они также как и ключи отношения накладывают некоторые ограничения целостности на базу данных.

Например, в отношении ЗАЧЕТКА (НОМЕР_ЗАЧЕТКИ, НОМЕР_ДИСЦИПЛИНЫ, ОЦЕНКА) функциональная зависимость НОМЕР_ЗАЧЕТКИ, НОМЕР_ДИСЦИПЛИНЫ $\to$ ОЦЕНКА означает, что в отношении ЗАЧЕТКА хранится только одна оценка, полученная студентом по дисциплине.

Используя понятие функциональной зависимости, можно дать следующее определение ключа отношения.

Ключ отношения со схемой $R(A_1, A_2, ..., A_n)$ - это некоторое подмножество атрибутов $K = \{B_1, B_2, ..., B_n\}$, $K \subseteq U_R$, которое удовлетворяет следующим свойствам:

1. каждый атрибут A_i функционально зависит от ключа, $K \to A_i$

2. никакое подмножество ключа $K' \subset K$ первым свойством не обладает.

То есть, если из ключа удалить хотя бы один атрибут, то он перестает быть ключом. Подмножество атрибутов K называется ***суперключом***, если оно удовлетворяет первому свойству, но не обязательно второму свойству.

В любом отношении существует ключ, так как множество всех атрибутов отношения удовлетворяет первому свойству. Далее можно, удаляя атрибуты, получить подмножество атрибутов, которое будет удовлетворять также и второму свойству.

Полная функциональная зависимость. Функциональная зависимость $X \to Y$ называется полной, если не существует такого атрибута $A \in X$, что имеет место функциональная зависимость $X \setminus A \to Y$. Например, в отношении ЭКЗАМЕН функциональная зависимость НОМЕР_ГРУППЫ, НОМЕР_ДИСЦИПЛИНЫ → ДАТА является полной, а функциональная зависимость НОМЕР_ГРУППЫ, НОМЕР_ДИСЦИПЛИНЫ, АУДИТОРИЯ → ДАТА является не полной, так как атрибут АУДИТОРИЯ можно удалить из левой части функциональной зависимости, а зависимость останется.

3. Аксиомы вывода функциональных зависимостей

Функциональные зависимости можно выводить на основе других фунциональных зависимостей с помощью аксиом Армстронга.

А1. Рефлексивность. Если $X \supseteq Y$, то $X \to Y$.

А2. Продолжение. Если $X \to Y$, $W \subseteq Z$, то $XZ \to YW$.

А3. Транзитивность. Если $X \to Y$, $Y \to Z$, то $X \to Z$.

А4. Псевдотранзитивность. Если $X \to Y$, $YW \to Z$, то $XW \to Z$.

А5. Аддитивность. Если $X \to Y$, $X \to Z$, то $X \to YZ$.

А6. Декомпозиция. Если $X \to Y$, $Y' \subseteq Y$, то $X \to Y'$.

Запись вида XY означает объединение множеств атрибутов X и Y. Для вывода функциональных зависимостей достаточно аксиом А1 – А3, так как аксиомы А4 – А6, могут быть выведены из них. Покажем это.

Доказательство псевдотранзитивности. Из $X \to Y$ и $W \subseteq W$ по аксиоме А2 выводится $XW \to YW$. Из $XW \to YW$ и $YW \to Z$ по аксиоме А3 выводится $XW \to Z$.

Доказательство аддитивности. Из $X \to Y$ и $Z \subseteq Z$ по аксиоме А2 выводится $XZ \to YZ$. Из $X \to Z$ и $XZ \to YZ$ по аксиоме А4 (мы можем ею воспользоваться, так как показали ее выводимость из аксиом А2 и А3) выводится $XX \to YZ$. Так как запись XX - это есть $X \bigcup X$, то получаем $X \to YZ$.

Доказательство декомпозиции. Из $Y' \subseteq Y$ по аксиоме А1 выводится $Y \to Y'$. Из $X \to Y$ и $Y \to Y'$ по аксиоме А3 выводится $X \to Y'$.

Замыканием множества функциональных зависимостей F называется множество всех функциональных зависимостей, выводимых из F с помощью аксиом Армстронга. Оно обозначается F^+.

Пример 3.1. Пусть задано множество функциональных зависимостей F = {A → B, B → C}. Применяя аксиомы вывода Армстронга можно построить следующее замыкание F^+ = {A → A, B → B, C → C, AB → A, AC → A, ABC → A, AB → B, BC → B, ABC → B, AC → C, BC → C, ABC → C, AB → AB, ABC → AB, AC → AC, ABC → AC, BC → BC, ABC → BC, ABC → ABC, A → B, AC → B, AC → BC, B → C, AB → C, AB → AC, B → BC, A → C, A → AC, A → AB, A→ BC, A →ABC, AB → C, AB → BC, AB → ABC, AC → AB, AC → ABC }.

Неизбыточным покрытием множества функциональных зависимостей *F* называется множество функциональных зависимостей H, которое удовлетворяет следующим условиям:

1. $H^+ = F^+$, т.е. множества всех функциональных зависимостей, выводимых из H и F, одинаковы.
2. Никакое подмножество $H' \subset H$, первому условию не удовлетворяет. Т.е., если из H удалить хотя бы одну функциональную зависимость, то из оставшегося множества уже нельзя будет вывести все те же функциональные зависимости, что и из множества F.

Для множества функциональных зависимостей F может существовать несколько неизбыточных покрытий, что видно из следующего примера.

Пример 3.2. Пусть задано множество функциональных зависимостей F = {A → B, B → A, B → C, A → C}. Тогда неизбыточным покрытием данного множества F будет множество H = {A → B, B → A, A → C}. Зависимость B → C выводится по аксиоме A3. Неизбыточным покрытием также является множество H = {A → B, B → A, B → C}. В этом варианте зависимость A → C выводится по аксиоме A3.

4. Поиск неизбыточного покрытия множества функциональных зависимостей

Замыканием множества атрибутов X на множестве функциональных зависимостей F называется множество таких атрибутов A, что функциональная зависимость $X \to A$ выводится из F с помощью аксиом. Оно обозначается X^+ и может быть вычислено на основе следующего алгоритма.

Алгоритм вычисления X^+

Пусть задано некоторое множество атрибутов U и множество функциональных зависимостей F на U. Построим замыкание множества атрибутов $X \subseteq U$ на множестве функциональных зависимостей F.

Будем последовательно строить подмножества атрибутов $X_0, X_1, ..., Xn$ по следующим правилам:

1. Положим $X_0 = X$
2. Пусть мы уже имеем множество X_i. Если в F существует функциональная зависимость вида $Y \to Z$, где $Y \subseteq X_i$, то построим множество $X_{i+1} = X_i \bigcup Z$.

Так как на каждом шаге алгоритма мы добавляем атрибуты и $X_i \subseteq U$, то в итоге на некотором шаге $X_{i+1} = X_i$ и алгоритм закончится.

Пример 4.1. Пусть на множестве U = {A, B, C, D, E, G} задано множество функциональных зависимостей F = {AB → C, D → EG,

C → A, BE → C, B → CD, CG → BD, ACD → B, CE → AG}. Построить замыкание атрибутов X = BD. Тогда по алгоритму получим:

1. X_0 = BD
2. D → EG, X_1 = BDEG
3. BE → C, X_2 = CBDEG
4. C → A, X_3 = ABCDEG

$(BD)^+$ = ABCDEG

На первом шаге алгоритма вычисления $X^+ X_0 = X$ и по аксиоме А1 имеет место функциональная зависимость $X \to X_0$. На следующем шаге алгоритма $Y \to Z$, где $Y \subseteq X_0$, то по аксиоме А1 имеет место функциональная зависимость $X_0 \to Y$, а по аксиоме А3 получаем $X_0 \to Z$. Тогда из $X_0 \to Z$ и $X \to X_0$, полученной на первом шаге, по аксиоме А3 имеем $X \to Z$. Так как $X_1 = X_0 \bigcup Z$, то по аксиоме А5 получим функциональную зависимость $X \to X_1$. Аналогично, так как $X \subseteq X_0, X_o \subseteq X_1, \ldots, X_{n-1} \subseteq X_n$, получим $X \to X^+$. Следовательно, по аксиоме А6 для подмножества $Y \subseteq X^+$ получим функциональную зависимость $X \to Y$. В результате алгоритм позволяет не только построить замыкание множества атрибутов X, но и определить что можно вывести функциональную зависимость $X \to Y$, где $Y \subseteq X^+$.

Пример 4.2. Рассмотрим тоже множество функциональных зависимостей, что и в примере 4.1. На каждом шаге алгоритма выводятся функциональные зависимости:

1. X_0 = BD; X → BD
2. D → EG, X_1 = BDEG; X → X_0, X_0 → D, D → EG, то X → EG и X → X_1
3. BE → C, X_2 = CBDEG; X → X_1, X_1 → BE, BE → C, то X → C и X → X_2
4. C → A, X_3 = ABCDEG; X → X_2, X_2 → C, C → A, то X → C и X → X_3

BD → ABCDEG

Используя понятие замыкания множества атрибутов и алгоритм его построения, можно построить неизбыточное покрытие множества функциональных зависимостей.

Алгоритм построения неизбыточного покрытия H.

Пусть задано множество функциональных зависимостей F. Будем последовательно строить множества функциональных зависимостей $H_0, H_1, ..., Hn$ по следующим правилам:

1. Положим $H_0 = F$
2. Пусть мы уже имеем множество H_i. Если в H_i существует функциональная зависимость $X \to Y$, где $Y \subseteq X^+$ и X^+ построено на множестве функциональных зависимостей $H_i \setminus (X \to Y)$, то перейдем к множеству $H_{i+1} = H_i \setminus (X \to Y)$.

Пример 4.3. Пусть задано множество функциональных зависимостей F = {A → BC, B → D, C → E, DE → G, AC → D, A → G}. Построим неизбыточное покрытие множества F.

1. H_0 = F
2. На множестве $H_0 \setminus$ (A → BC) замыкание множества атрибутов A^+ = AG, на $H_0 \setminus$ (B → D) B^+ = B, на $H_0 \setminus$ (C → E) C^+ = C, на $H_0 \setminus$ (DE → G) DE^+ = DE. Эти функциональные зависимости не выводимы, так как их правые части не принадлежат замыканиям множеств атрибутов их левых частей.
   На $H_0 \setminus$ (AC → D) AC^+ = ABCDEG, поэтому функциональную зависимость AC → D можно удалить из H_0. $H_1 = H_0 \setminus$ (AC → D).
3. На $H_1 \setminus$ (A → G) A^+ = ABCDEG, поэтому функциональную зависимость A → G можно удалить. $H_2 = H_1 \setminus$ (A → G)

В результате получаем неизбыточное покрытие H = {A → BC, B → D, C → E, DE → G}.

5. Многозначные зависимости

Рассмотрим отношение со схемой $R(X, Y, Z)$, где X, Y могут пересекаться, а $Z = U_R \setminus (X \cup Y)$. Обозначим $Y(x)$ множество всех таких значений y атрибутов Y, которые встречаются в кортежах отношения R вместе со значением x атрибутов X.

Говорят, что существует ***многозначная зависимость*** $X \to\to Y$ (Y многозначно зависит от X), если для любой пары значений xz атрибутов XZ верно $Y(xz) = Y(x)$. Другими словами,

каждому значению x атрибутов X соответствует некоторое множество значений Y и это множество не зависит от значений Z.

Можно определить многозначную зависимость также следующим образом. Обозначим $t_1(X)$ - множество значений атрибутов X в кортеже t_1. Тогда в отношении R существует **многозначная зависимость** $X \to\to Y$, если для любой пары кортежей t_1, t_2, таких что $t_1(X) = t_2(X)$, существует кортеж t_3, с $t_3(X) = t_1(X)$, $t_3(Y) = t_1(Y)$ и $t_3(Z) = t_2(Z)$. В силу симметрии определения относительно t_1 и t_2 следует, что в R существует кортеж t_4, с $t_4(X) = t_1(X)$, $t_4(Y) = t_2(Y)$ и $t_4(Z) = t_1(Z)$. То есть в отношении существуют следующие четыре кортежа:

t_1: x, y_1, z_1
t_2: x, y_2, z_2
t_3: x, y_1, z_2
t_4: x, y_2, z_1

Пример 5.1. В отношении R (НОМ_ЗАЧ, НОМ_ДИСЦ, ЯЗЫК_ПРОГР) хранится информация о дисциплинах, изучаемых студентами и языках программирования, которыми владеют студенты. Например, отношение может содержать следующие кортежи:

R (НОМ_ЗАЧ, НОМ_ДИСЦ, ЯЗЫК_ПРОГР)

123	1	C
123	2	Java
123	1	Java
123	2	C

Список дисциплин, изучаемых студентом 123, не изменится, если рассматривать кортежи отношения с конкретным языком программирования, например Java, которым владеет данный студент. Аналогично, список языков программирования, которыми владеет студент 123, не изменится, если рассматривать только кортежи с конкретной дисциплиной, изучаемой студентом, например 2. Поэтому в этом отношении существуют две многозначные зависимости: НОМ_ЗАЧ $\to\to$ НОМ_ДИСЦ, НОМ_ЗАЧ $\to\to$ ЯЗЫК_ПРОГР.

Многозначные зависимости, как и функциональные зависимости, определяются на основе семантики атрибутов. Так в примере 5.1 можно было бы интерпретировать ЯЗЫК_ПРОГР, как язык программирования, который студенты изучают на данной

дисциплине. В этом случае отношение R уже не будет содержать указанных многозначных зависимостей.

Многозначные зависимости вида $X \to\to \emptyset$, и $X \to\to Y$, если $Z = \emptyset$, т.е. $U_R = X \bigcup Y$, называются **тривиальными многозначными зависимостями**, так как они выполняются для любой реализации отношения.

Любая функциональная зависимость $X \to Y$ является частным случаем многозначной зависимости, так как единственное значение y, соответствующее значению x, ни от чего более не зависит.

Функциональная зависимость $X \to Y$ определяется только множествами атрибутов X, Y и не связана с другими атрибутами, многозначная же зависимость определяется всеми атрибутами отношения. Возможна ситуация, когда в отношении нет многозначной зависимости вида $X \to\to Y$, но она появляется в проекции этого отношения. Такие многозначные зависимости называются **встроенными многозначными зависимостями**.

Пример 5.2. Рассмотрим отношение

R 1 (НОМ_ЗАЧ, НОМ_ДИСЦ, ЯЗЫК_ПРОГР, НАИМ_ДИСЦ)

123	1	C	Математика
123	2	Java	Физика
123	1	Java	Математика
123	2	C	Физика

В нем отсутствует многозначная зависимость НОМ_ЗАЧ →→ НОМ_ДИСЦ, так как если зафиксировать пару значений Java Математика, то мы получим только одну дисциплину с кодом 1 для студента 123. После выполнения операции проекции мы получаем отношение R из примера 7.1, в котором эта многозначная зависимость присутствует, т.е. она является встроенной в отношение R1.

6. Аксиомы вывода многозначных зависимостей

Для многозначных зависимостей также как и для функциональных зависимостей существуют аксиомы вывода, позволяющие получать новые зависимости на основе имеющихся. Эти аксиомы были предложены Биэри, Фэджином и Ховардом. Они являются аналогами аксиом Армстронга, за исключением М0.

Рассмотрим отношение со схемой $R(U)$ и подмножества атрибутов данной схемы X, Y, Z. Имеются следующие аксиомы.

М0. Дополнение. Если $U = X \bigcup Y \bigcup Z$ и $Y \bigcap Z \subseteq X$, то $X \to\to Y$ имеет место тогда и только тогда, когда $X \to\to Z$.

М1. Рефлексивность. Если $X \supseteq Y$, то $X \to\to Y$.

М2. Продолжение. Если $X \to\to Y$, $W \subseteq Z$, то $XZ \to\to YW$.

М3. Транзитивность. Если $X \to\to Y$, $Y \to\to Z$, то $X \to\to Z - Y$.

М4. Псевдотранзитивность. Если $X \to\to Y$, $YW \to\to Z$, то $XW \to\to Z - (Y \bigcup W)$.

М5. Аддитивность. Если $X \to\to Y$, $X \to\to Z$, то $X \to\to YZ$.

М6. Декомпозиция. Если $X \to\to Y_1$, $X \to\to Y_2$, то $X \to\to Y_1 \bigcap Y_2$, $X \to\to Y_1 \setminus Y_2$, $X \to\to Y_2 \setminus Y_1$.

Как и для аксиом Армстронга некоторые из аксиом многозначных зависимостей могут быть выведены через другие аксиомы. Так из аксиом М0, М1, М3 можно вывести все остальные аксиомы.

7. Аномалии обновления

Рассмотрим отношение R(НОМ_ЗАЧ, НОМ_ДИСЦ, ОЦЕНКА, ФИО). Данное отношение имеет ряд недостатков.

Избыточность. Информация о фамилии студента дублируется столько раз, сколько он получил оценок на экзаменах.

Аномалия обновления. При изменении фамилии студента возможно появление противоречия в базе данных, если фамилия студента будет изменена не для всех кортежей, соответствующих студенту.

Аномалия удаления. При удалении всех оценок студента исчезнет информация о фамилии студента.

Аномалия включения. Пока не появится информация о сдаче студентом экзаменов не будет информации о фамилии студента.

Избыточности и указанных аномалий можно избежать, если вместо данного отношения хранить в базе данных две его проекции:

СТУД (НОМ_ЗАЧ, ФИО)

ЗАЧЕТКА (НОМ_ЗАЧ, НОМ_ДИСЦ, ОЦЕНКА)

Приведенный пример говорит о том, что один вариант базы данных может быть лучше, чем другой вариант базы данных.

8. Нормальные формы отношений

Для того, чтобы избавится от избыточности и аномалий нужно ввести некоторые ограничения на схему базы данных. Это делается за счет определения нормальных форм отношений.

8.1. Первая нормальная форма

Отношение находится в первой нормальной форме (1НФ), если каждый атрибут отношения определен на домене, принимающем только атомарные значения. Т.е. значения атрибутов представляют собой скалярные значения и не могут быть ни массивами, ни списками, ни более сложными структурами.

Пример 8.1. Рассмотрим отношение, содержащее сведения о языках программирования, которыми владеют студенты.

R (НОМ_ЗАЧ, ЯЗЫК_ПРОГР)

121	C, Java,
122	C, Pascal
123	C, C++

Данное отношение не находится в первой нормальной форме, так как значениями атрибута ЯЗЫК_ПРОГР являются массивы. Данное отношение не позволяет детально определить функциональные зависимости. Формально в этом отношении существует функциональная зависимость НОМ_ЗАЧ $\rightarrow$ ЯЗЫК_ПРОГР и тогда ее можно интерпретировать как знание студентом только одного языка программирования. Это, однако, противоречит данным, которые хранятся в этом отношении. Чтобы привести это отношение к 1НФ достаточно записать его в виде, где каждое значение атрибута ЯЗЫК_ПРОГР будет соответствовать только одному языку, которым владеет студент:

R (НОМ_ЗАЧ, ЯЗЫК_ПРОГР)

121	C
121	Java,
122	C

122 Pascal
123 C
123 C++

8.2. Вторая нормальная форма

Отношение R находится во второй нормальной форме (2НФ), если оно находится в 1НФ и каждый непервичный атрибут полно функционально зависит от каждого возможного ключа отношения R.

Пример 8.2. Рассмотрим отношение

R(НОМ_ЗАЧ, НОМ_ДИСЦ, ОЦЕНКА, ФИО)

В этом отношении имеют место следующие функциональные зависимости:

НОМ_ЗАЧ НОМ_ДИСЦ $\rightarrow$ ОЦЕНКА

НОМ_ЗАЧ НОМ_ДИСЦ $\rightarrow$ ФИО

НОМ_ЗАЧ $\rightarrow$ ФИО

Атрибут ФИО не полно зависит от ключа НОМ_ЗАЧ НОМ_ДИСЦ. Следовательно, данное отношение не находится во 2НФ. Проведем декомпозицию этого отношения на отношения:

СТУД (НОМ_ЗАЧ, ФИО)

ЗАЧЕТКА (НОМ_ЗАЧ, НОМ_ДИСЦ, ОЦЕНКА)

В первом отношении ключом является НОМ_ЗАЧ, а во втором - НОМ_ЗАЧ НОМ_ДИСЦ. Оба отношения находятся во 2НФ так как непервичный атрибут ФИО в отношении СТУД и атрибут ОЦЕНКА – в отношении ЗАЧЕТКА полно функционально зависят от ключа.

Из определения 2НФ вытекают два следствия, которые помогают в проверке определения нахождения отношения во 2НФ.

Следствие 1. Если в отношении все атрибуты ключевые, то оно находится в 2НФ.

Следствие 2. Если любой ключ отношения состоит из одного атрибута, то отношение находится во 2НФ.

8.3. Третья нормальная форма

Пусть X и Y подмножества атрибутов схемы R, а A – некоторый атрибут схемы R и $A \notin X, A \notin Y$. Говорят, что атрибут A транзитивно зависит от X, если выполняются следующие условия: $X \rightarrow Y$, $Y \rightarrow A$, не существует функциональной зависимости $Y \rightarrow X$.

Отношение R находится в третьей нормальной форме (3НФ), если оно находится в 1НФ и ни один из непервичных атрибутов A не является транзитивно зависимым от некоторого ключа X.

Теорема. Отношение, находящееся в 3НФ, также находится в 2НФ.

Доказательство. Предположим, что схема R не находится в 2НФ, тогда некоторый непервичный атрибут A неполно функционально зависит от некоторого ключа X, т.е. существует функциональная зависимость $Y \to A$, где $Y \subset X$. Функциональной зависимости $Y \to X$ не существует, так как в этом случае Y был бы тоже ключом, а это противоречит определению ключа (ключ не может содержать подмножества, которое само является ключом). Так как A – непервичный атрибут, то $A \notin X, A \notin Y$. Все атрибуты функциоанально зависят от ключа, поэтому имеет место зависимость $X \to Y$. В результате мы получили, что A транзитивно зависит от ключа X, а это противоречит тому, что R находится в 3НФ. Следовательно, наше предположение, что R не находится в 2НФ, не верно.

Теорема. В отношении в 3НФ не существует функциональных зависимостей между непервичными атрибутами.

Доказательство. Предположим, что существует функциональная зависимость между непервичными атрибутами $B \to A$. Тогда возьмем любой ключ X. Так как любой атрибут функционально зависит от ключа, то имеем $X \to B$. Так как В непервичный атрибут, то не существует зависимости $B \to X$. Получаем, что атрибут A транзитивно зависит от ключа, а это противоречит тому, что R находится в 3НФ.

Пример 8.3. Рассмотрим отношение

R (НОМ_ГРУП, НОМ_ДИСЦ, НОМ_УДОСТ, ДАТА. АУД, ФИО_ПРЕП)

В этом отношении имеют место следующие функциональные зависимости:

НОМ_ГРУП НОМ_ДИСЦ → НОМ_УДОСТ

НОМ_ГРУП НОМ_ДИСЦ → ДАТА

НОМ_ГРУП НОМ_ДИСЦ → АУД

НОМ_ГРУП НОМ_ДИСЦ → ФИО_ПРЕП

НОМ_УДОСТ → ФИО_ПРЕП

Ключом отношения R является множество атрибутов НОМ_ГРУП НОМ_ДИСЦ. Непервичный атрибут ФИО_ПРЕП транзитивно зависит от ключа, так как имеют место функциональные зависимости:

НОМ_ГРУП НОМ_ДИСЦ → НОМ_УДОСТ

НОМ_УДОСТ → ФИО_ПРЕП

Следовательно, отношение R не находится в 3НФ. В этом отношении существует избыточность и аномалии, так как фамилия преподавателя дублируется столько раз, сколько экзаменов принимает преподаватель. Проведем декомпозицию этого отношения на отношения:

ЭКЗ(НОМ_ГРУП, НОМ_ДИСЦ, НОМ_УДОСТ, ДАТА, АУД)

ПРЕП (НОМ_УДОСТ, ФИО_ПРЕП)

Эти отношения, как нетрудно заметить, уже будут в 3НФ.

8.4. Нормальная форма Бойса-Кодда

Отношение R находится в нормальной форме Бойса-Кодда (БКНФ), если оно находится в 1НФ и для каждого подмножества атрибутов $X \subseteq U_R$ и каждого атрибута $A \in U_R$, $A \notin X$, таких что имеет место функциональная зависимость $X \to A$, подмножество атрибутов X является суперключом.

Теорема. Отношение, находящееся в БКНФ, также находится в 3НФ.

Доказательство. Предположим, что отношение R не находится в 3НФ, тогда найдется транзитивная зависимость $X \to Y$, $Y \to A$, $A \notin X, A \notin Y$, и не существует функциональной зависимости $Y \to X$. Так как имеет место функциональная зависимость $Y \to A$ и отношение R находится в БКНФ, то подмножество атрибутов Y является суперключом. Следовательно, должна существовать функциональная зависимость $Y \to X$, так как от суперключа функционально зависят все атрибуты. Получили противоречие с тем, что по условию транзитивной зависимости не существует функциональной зависимости $Y \to X$.

Пример 8.4. Рассмотрим отношение

R (ГОРОД, АДРЕС, ПОЧТОВЫЙ_ИНДЕКС)

В этом отношении, в предположении, что АДРЕС – это адрес без указания города, имеют место следующие функциональные зависимости:

ГОРОД АДРЕС → ПОЧТОВЫЙ_ИНДЕКС

ПОЧТОВЫЙ_ИНДЕКС → ГОРОД

Отношение R находится в 3НФ и имеет ключ, состоящий из пары атрибутов ГОРОД АДРЕС. Наличие функциональной зависимости ПОЧТОВЫЙ_ИНДЕКС → ГОРОД, в которой левая часть не является суперключом, означает, что отношение не находится в БКНФ. В данном случае с помощью декомпозиции мы не можем привести это отношение к отношениям в БКНФ.

8.5. Четвертая нормальная форма

Отношение R находится в четвертой нормальной форме (4НФ), если оно находится в 1НФ и для каждого подмножества атрибутов $X \subseteq U_R$ и $Y \subseteq U_R$, $Y \not\subset X$, таких что имеет место нетривиальная многозначная зависимость $X \to\to Y$, подмножество атрибутов X является суперключом.

Теорема. Отношение, находящееся в 4НФ, также находится в БКНФ.

Доказательство. Пусть для подмножества атрибутов $X \subseteq U_R$ и некоторого атрибута $A \in U_R$, $A \notin X$ имеет место функциональная зависимость $X \to A$. Так как функциональная зависимость является также и многозначной зависимостью и отношение R находится в 4НФ, то подмножество атрибутов X является суперключом.

Пример 8.5. Рассмотрим отношение

R (НОМ_ЗАЧ, НОМ_ДИСЦ, ЯЗЫК_ПРОГР)

В нем хранится информация о дисциплинах, изучаемых студентами и языках программирования, которыми владеют студенты.

В отношении R существуют две нетривиальные многозначные зависимости:

НОМ_ЗАЧ →→ НОМ_ДИСЦ, НОМ_ЗАЧ →→ ЯЗЫК_ПРОГР

Левые части этих зависимостей не являются суперключами, так как ключом являются все атрибуты отношения: НОМ_ЗАЧ, НОМ_ДИСЦ, ЯЗЫК_ПРОГР. Поэтому отношение R не находится в 4НФ. В отношении R имеет место избыточность и аномалии, от которых можно избавиться с помощью декомпозиции отношения на два отношения, находящиеся в 4НФ:

R1 (НОМ_ЗАЧ, НОМ_ДИСЦ)
R2 (НОМ_ЗАЧ, ЯЗЫК_ПРОГР)

Для приведения отношения к 4НФ с помощью декомпозиции можно воспользоваться следующей теоремой.

Теорема Фэджина. В отношении $R\,(X, Y, Z)$ имеет место многозначная зависимость $X \rightarrow\rightarrow Y$ тогда и только тогда, когда оно может быть представлено как естественное соединение своих проекций $R_1\,(X, Y)$ и $R_2(X, Z)$, т.е. $R(X, Y, Z) = R_1\,(X, Y) \triangleright\triangleleft R_2\,(X, Z)$

9. Задача проектирования реляционной схемы базы данных

9.1. Общая постановка задачи.

Введем новое определение схемы отношения.

Схемой отношения R называется описание отношения, состоящее из множества атрибутов U и множества зависимостей F:

$R = <U, F>$

Зависимости F могут быть как функциональными, так и многозначными.

Схемой базы данных называется множество схем отношений:

$S = \{Ri = <Ui, Fi>, i = 1,2, \ldots n\}$

Постановка задачи проектирования реляционной схемы базы данных заключается в следующем..

Пусть задана исходная схема базы данных, состоящая из одной схемы отношения:

$S_0 = \{R_0 = <U_0, F_0>\}$

Требуется построить в некотором смысле эквивалентную ей схему базы данных

$Sd = \{Ri = <Ui, Fi>, i = 1,2, \ldots n\},$

которая была бы лучше исходной схемы S_0.

В качестве критериев эффективности реализации реляционной схемы базы данных используются следующие два критерия:

- стремятся привести все отношения базы данных к наиболее «сильной» нормальной форме;
- стремятся построить количественно оптимальную схему базы данных, т.е. содержащую минимальное количество схем отношений.

Существуют и другие критерии, например, критерий минимизации времени выполнения запросов. Но эти критерии требуют рассмотрения дополнительных параметров описания отношений базы данных и знание характеристик выполняемых запросов.

9.2. Определения эквивалентности.

Существуют различные определения эквивалентности схем баз данных.

Определение 1. Схема базы данных Sd эквивалентна схеме базы данных So, если эти схемы имеют одинаковое множество атрибутов и сохранены все зависимости между атрибутами.

Декомпозицией схемы отношения Ro называется множество схем отношений $R_1, R_2, \ldots, R_n$ таких, что выполняется условие:

$$U_0 = \bigcup_{i=1}^{n} Ui,$$

где U_i - множество атрибутов схемы отношения R_i

Поэтому схема базы данных Sd эквивалентна схеме базы данных So, если эта схема получена из So с помощью декомпозиции, сохраняющей зависимости.

Если рассматривать только функциональные зависимости, то можно формализовать это определение.

Определение 1 (случай функциональных зависимостей). Схема базы данных Sd эквивалентна схеме базы данных S_0, если выполняются два условия:

1. $U_0 = \bigcup_{i=1}^{n} Ui$

2. $F_0^+ = \bigcup_{i=1}^{n} Fi^+$

Определение 2. Схема базы данных Sd эквивалентна схеме базы данных So, если эти схемы имеют одинаковое множество атрибутов и база данных со схемой Sd содержит те же данные, что и база данных со схемой S_0.

Это определение может быть формализовано, если ввести понятие соединения без потерь.

Свойство соединения без потерь. Декомпозиция схемы отношения R на схемы отношений $R_1, R_2, \ldots, R_n$ Обладает свойством соединения без потерь, если любая реализация отношения R может быть представлена как естественное соединение его проекций:

$R = R_1 \rhd\lhd R_2 \rhd\lhd \ldots \rhd\lhd R_n$, где $R_i = \pi_{U_i}(R)$

Другими словами, схемы Sd и S_0 эквиваленты, если схема Sd получена из схемы S_0 с помощью декомпозиции, обладающей свойством соединения без потерь.

Свойство соединения без потерь может быть проверено, используя теорему Фэджина.

Определения 1 и 2 не эквивалентны. Это видно из следующих примеров.

Пример 9.1. Рассмотрим схемы S_0 и Sd:

$S_0 = \{R_0 = <(A, B, C), \{AB \rightarrow C, C \rightarrow A, C \rightarrow B\}>\}$

$Sd = \{R_1 = <(A, C), \{C \rightarrow A\}>, R_2 = <(C, B), \{C \rightarrow B\}>\}$

Так как имеет место функциональная зависимость $C \rightarrow A$, то по теореме Фэджина $R_0 = R_1 \rhd\lhd R_2$. Поэтому выполняется второе определение эквивалентности. Первое определение не выполняется, так как зависимость $AB \rightarrow C$ не выводима из зависмостей $C \rightarrow A$ и $C \rightarrow B$

Пример 9.2. Рассмотрим схемы S_0 и Sd:

$S_0 = \{R_0 = <(A, B, C, D), \{A \rightarrow B, BC \rightarrow D, D \rightarrow B, D \rightarrow C\}>\}$

$Sd = \{R_1 = <(A, B), \{A \rightarrow B\}>, R_2 = <(B, C, D), \{BC \rightarrow D, D \rightarrow B, D \rightarrow C\}>\}$

Схемы S_0 и Sd имеют одинаковое множество функциональных зависимостей, поэтому выполняется первое определение эквивалентности. Если предположить, что $R_0 = R_1 \rhd\lhd R_2$, то по теореме Фэджина должна существовать многозначная зависимость $B \rightarrow\rightarrow A$, что неверно. Следовательно, второе опередение эквивалентности не выполняется.

Можно также показать не выполнение второго определения эквивалентности на примере.

Пусть отношение R_0, содержит следующие кортежи:

R_0 (A, B, C, D),

a1, b1, c1, d1

a2, b1, c2, d2

тогда отношения R_1 и R_2, являющиеся проекциями отношения R_0, содержат следующие кортежи:

R_1 (A, B)	R_2 (B, C, D),
a1, b1	b1, c1, d1
a2, b1	b1, c2, d2

а их естественное соединение R_3 содержит следующие кортежи:

R_3 (A, B, C, D),
a1, b1, c1, d1
a1, b1, c2, d2
a2, b1, c1, d1
a2, b1, c2, d2

10. Алгоритмы проектирования реляционной схемы базы данных

10.1. Алгоритм Фэджина

Алгоритм начинает проектирование с исходной схемы базы данных

$S_0 = \{R_0 = <U_0, F_0>\}$, где

F_0 – множество многозначных зависимостей.

Результатом алгоритма является схема базы данных

$Sd = \{Ri = <Ui, Fi>, i = 1,2, \ldots n\}$, где

каждое Fi – множество многозначных зависимостей и каждое отношение Ri находится в 4НФ.

Алгоритм.

1. Положим первоначально $Sd = S_0$.

2. Если в отношении $Ri\ (X, Y, Z)$ существует нетривиальная многозначная зависимость $X \rightarrow\rightarrow Y$ и X не является суперключом, то заменим Ri в схеме Sd на две схемы отношений $R_{i+1}\ (X, Y)$ и $R'_{i+1}\ (X, Z)$

Шаг 2 алгоритма продолжается до тех пор, пока найдутся схемы отношений Ri с указанными свойствами.

Достоинства и недостатки алгоритма Фэджина.

Алгоритм строит схему базы данных из отношений в 4НФ.

Если предположить, что некоторое отношение Ri (X, Y, Z) в результирующей схеме не находится в 4НФ, то в нем найдется

нетривиальная многозначная зависимость X→→Y, в которой X не является суперключом, и это отношение будет использовано на шаге 2 для декомпозиции.

Схема базы данных, полученная по алгоритму, эквивалентна исходной схеме по второму определению. Это следует из того, что на каждом шаге алгоритма полученная декомпозиция согласно теореме Фэджина обладает свойством соединения без потерь.

Схема базы данных, полученная по алгоритму, не обязательно эквивалентна исходной схеме по первому определению.

Пример 10.1. Пусть задана схема

$S_0 = \{R_0 = <(A, B, C, D), \{A→B, B→C\}>\}$.

Применим к ней алгоритм Фэджина.

1. A→B. $R_0(A, B, C, D)$ заменим на $R_1(A, B)$, $R'_1(A, C, D)$

2. A→C. $R'_1(A, C, D)$ заменим на $R_2(A, C)$, $R'_2(A, D)$. Зависимость A→C не присутствует в схеме R_0, но она может быть получена по аксиоме транзитивности из зависимостей A→B, B→C.

На каждом шаге использовалась нетривиальная зависимость с левой частью, не являющейся суперключом, так в R_0 и R'_1 ключом является множество атрибутов AD.

В результате получаем схему базы данных

$Sd = \{R_1 = <(A, B), \{A→B\}>,$
$R_2 = <(A, C), \{A→C\}>,$
$R_3 = <(A, D), \{\}>\}.$

Полученная схема не эквивалентна исходной схеме по первому определению, так утеряна зависимость B→C.

Так как в алгоритме не определена последовательность выбора многозначных зависимостей, используемых для декомпозиции отношений, то в результате мы можем получать совершенно разные схемы.

Пример 10.2. Рассмотрим туже схему

$S_0 = \{R_0 = <(A, B, C, D), \{A→B, B→C\}>\}$.

Применим к ней алгоритм Фэджина, но теперь используем другую последовательность выбора зависимостей

1. B→C. $R_0(A, B, C, D)$ заменим на $R_1(B, C)$, $R'_1(A, B, D)$

2. A→B. $R'_1(A, B, D)$ заменим на $R_2(A, B)$, $R'_2(A, D)$.

В результате получаем схему базы данных

$Sd = \{R_1 = <(B, C), \{B→C\}>,$
$R_2 = <(A, B), \{A→B\}>,$
$R_3 = <(A, D), \{\}>\}.$

Схема Sd из примера 10.2 отличается от схемы Sd из примера 10.1 по составу атрибутов и зависимостей в схемах отношений. Схема

Sd из примера 10.2 эквивалентна исходной схеме S_0 по первому определению.

Сохранить функциональные зависимости удалось, благодаря изменению последовательности выбора зависимостей. В общем случае, если имеется цепочка функциональных зависимостей вида $A_1 \rightarrow A_2$, $A_2 \rightarrow A_3$, … $A_{n-1} \rightarrow A_n$, то в целях сохранения этих зависимостей следует применять их для декомпозиции в обратном порядке $A_{n-1} \rightarrow A_n$, $A_{n-2} \rightarrow A_{n-1}$, … $A_1 \rightarrow A_2$. Это, однако, не гарантирует от сохранения всех зависимостей, что следует из следующего примера.

Пример 10.3. Пусть задана схема

$S_0 = \{R_0 = <(A, B, C, D), \{A \rightarrow B, B \rightarrow C, D \rightarrow C\}>\}$.

Применим к ней алгоритм Фэджина, используя туже последовательность выбора функциональных зависимостей, что и в примере 10.2, чтобы не потерять зависимость B→C.

1. B→C. $R_0(A, B, C, D)$ заменим на $R_1(B, C)$, $R'_1(A, B, D)$
2. A→B. $R'_1(A, B, D)$ заменим на $R_2(A, B)$, $R'_2(A, D)$.

В результате получаем схему базы данных

$Sd = \{R_1 = <(B, C), \{B \rightarrow C\}>,$
$R_2 = <(A, B), \{A \rightarrow B\}>,$
$R_3 = <(A, D), \{\}> \}$.

Полученная схема совпадает со схемой примера 10.2, но она не эквивалентна исходной схеме по первому определению, так утеряна зависимость D→C.

Рассмотрим другой порядок выбора зависимостей

1. D→C. $R_0(A, B, C, D)$ заменим на $R_1(D, C)$, $R'_1(A, B, D)$
2. A→B. $R'_1(A, B, D)$ заменим на $R_2(A, B)$, $R'_2(A, D)$.

В результате получаем схему базы данных

$Sd = \{R_1 = <(D, C), \{D \rightarrow C\}>,$
$R_2 = <(A, B), \{A \rightarrow B\}>,$
$R_3 = <(A, D), \{\}>\}$.

Эта схема также не эквивалентна исходной схеме по первому определению, так как утеряна зависимость B→C.

Количество схем отношений в результирующей схеме зависит от порядка выбора многозначных зависимостей, используемых для декомпозиции.

Пример 10.4.Пусть задана схема

$S_0 = \{R_0 = <(A, B, C, D), \{A \rightarrow B, A \rightarrow C, A \rightarrow BC\}>\}$.

Функциональная зависимость A→BC выводится из A→B, A→C, но исходная схема S_0 для алгоритма Фэджина не требует, чтобы F_0 было неизбыточным покрытием.

Рассмотрим следующую последовательность декомпозиции.

1. A→B. R_0(A, B, C, D) заменим на R_1(A, B), R'_1(A, C, D)

2. A→C. R'_1(A, C, D) заменим на R_2(A, C), R'_2(A, D).

В результате получаем схему базы данных

Sd = {R_1 = <(A, B), {A→B}>,

R_2 = <(A, C), {A→C}>,

R_3 = <(A, D), {}>}.

Полученная схема эквивалентна исходной схеме по первому и по второму определению.

На каждом шаге использовалась нетривиальная зависимость с левой частью, не являющейся суперключом, так в R_0 и R'_1 ключом является множество атрибутов AD.

Применим другой порядок выбора зависимостей.

A→BC. R_0(A, B, C, D) заменим на R_1(A, B, C), R'_1(A, D)

Функциональные зависимости A→B, A→C не могут быть использованы для декомпозиции схемы R_1(A, B, C), так как в этой схеме A является ключом.

В результате получаем схему базы данных

Sd = {R_1 = <(A, B, C), {A→BC}>,

R_2 = <(A, D), {}>}.

Эта схема также эквивалентна исходной схеме по первому и второму определению, но содержит меньше схем отншений, чем предыдущая схема.

Алгоритм не строит количественно оптимальную схему базы данных. Из примера 10.4 видно, что по алгоритму не всегда получается схема базы данных с минимальным количеством отношений.

Алгоритм Фэджина является не эффективным. На каждом шаге 2 надо решать задачу определения нахождения отношения Ri в 4НФ, а эта задача относится к классу NP-полных задач.

10.2. Алгоритм Делобеля-Кейси

Алгоритм начинает проектирование с исходной схемы базы данных

$S_0 = \{R_0 = <U_0, F_0>\}$, где

F_0 – множество функциональных зависимостей вида $X \rightarrow A$, где **X** - подмножество атрибутов U_0, а A – некоторый атрибут из U_0.

Результатом алгоритма является схема базы данных

$Sd = \{Ri = <Ui, Fi>, i = 1,2, \ldots n\}$, где

каждое Fi – множество функциональных зависимостей и каждое отношение Ri находится в 3НФ.

Алгоритм.

1. Найти все ключи схемы R_0 - $K_1, K_2, \ldots, K_m$

2. Найти неизбыточное покрытие H для множества функциональных зависимостей F_0.

2'. Дополнение Бернштейна . Прежде, чем строить неизбыточное покрытие H, необходимо удалить лишние атрибуты из левых частей функциональных зависимостей H.

Если $X \to Y, A \in X, X \setminus A \to Y$, то A – лишний атрибут.

3. Каждой функциональной зависимости $Xi \to Ai \in H$ ставим в соответствие схему $Ri\ (Xi, Ai)$. Если хотя бы один ключ Kj , найденный на 1 шаге, принадлежит множеству атрибутов хотя бы одной схемы Ri, то алгоритм закончен. Иначе добавляем схему $R_{n+1}\ (Kj)$ с произвольным ключом Kj.

Достоинства и недостатки алгоритма Делобеля-Кейси.

Алгоритм строит схему базы данных из отношений в 3НФ.

Каждая схема имеем вид $R\ (X,\ A)$, где $X \to A$ полная функциональная зависимость. В этой схеме X является ключом. Если в этой схеме существует еще один ключ $X1$, то $X1$ не принадлежит X, так как тогда зависимость будет не полной. Значит, $X1$ содержит A и в схеме все атрибуты являются ключевыми, т.е. она в 3НФ. Если X – это единственный ключ и это не 3НФ, то найдется множество Y, такое что $X \to Y, Y \to A, A \notin X, A \notin Y$. $Y \notin X$, так как зависимость $X \to A$ полная. Следовательно, $Y = A$, так как в схеме нет больше других атрибутов. Это противоречит тому, что $A \notin Y$.

Если не использовать дополнение Бернштейна, то полученная схема может получиться не в 3НФ. Это видно из следующего примера.

Пример 10.5. Пусть задана схема

S_0 = {R_0 = <(A, B, C), {AB→C, A→B, C→B}>}.

В отношении R_0 ключом является A, так как A→B и из A→B, AB→C по аксиоме псевдотранзитивности следует A→C.

Неизбыточное покрытие H совпадает с исходным множеством функциональных зависимостей F_0. Получаем схему

Sd = {R_1 = <(A, B, C), {AB→C}>,

R_2 = <(A, B), {A→B }>,

R_3 = <(C, B), {C→B }>}.

В отношении R_1 B – это неключевой атрибут, который транзитивно зависит от A (A→C, C→B), поэтому это отношение не находится в 3НФ.

Если в неизбыточном покрытии H из функциональной зависимости AB→C устранить лишний атрибут B, то получим множество функциональных зависимостей {A→C, A→B, C→B}, которое уже не является неизбыточным покрытием, так как зависимость A→B выводится по аксиоме транзитивности. Следовательно, в дополнении Бернштейна существенным является устранение лишних атрибутов до построения неизбыточного покрытия. В результате получим неизбыточное покрытие {A→C, C→B} и схему базы данных:

Sd = {R_1 = <(A, C), {A→C}>,
R_2 = <(C, B), { C→B }>}.

Схема базы данных, полученная по алгоритму, эквивалентна исходной схеме базы данных как по первому, так и по второму определению.

Эквивалентность по первому определению следует из того, что полученные схемы отношений содержат множество функциональных зависимостей из неизбыточного покрытия H. Эквивалентность по второму определению связана с добавлением на последнем шаге алгоритма схемы отношения, содержащей один из ключей.

По алгоритму может быть построено несколько разных схем баз данных, так как на шаге 2 алгоритма могут быть построены различные неизбыточные покрытия.

Пример 10.6. Рассмотрим схему

S_0 = {R_0 = <(A, B, C, D), {A→B, A→C, A→BC}>} из примера 10.4.

Построим схему базы данных, используя алгоритм Делобеля-Кейси.

Схема отношения R_0 имеет единственный ключ – A.

Неизбыточное покрытие H= {A→B, A→C}

Каждой зависимости ставим в соответствие отношение. В результате получаем схему базы данных

Sd = {R_1 = <(A, B), {A→B}>,
R_2 = <(A, C), {A→C }>}.

Ключ A содержится в схеме отношения R_1, поэтому нет необходимости добавлять схему отношения с ключом.

На шаге 2 алгоритма можно было построить неизбыточное покрытие H = {A→BC} и тогда результирующая схема базы данных будет иметь вид

Sd = {R_1 = <(A, B, C), {A→BC}>}.

Функциональные зависимости A→B, A→C не пропали, так как они выводятся из зависимости A→BC, используя аксиому декомпозиции.

Алгоритм не строит количественно оптимальную схему базы данных. Это следует из примера 10.5.

Алгоритм имеет неполиномиальную оценку сложности. Это связано с нахождением всех возможных ключей на шаге 1.

10.3. Алгоритм Бернштейна

Алгоритм начинает проектирование с исходной схемы базы данных

$S_0 = \{R_0 = <U_0, F_0>\}$, где

F_0 – множество функциональных зависимостей вида $X \to A$, где X - подмножество атрибутов U_0, а A – некоторый атрибут из U_0.

Результатом алгоритма является схема базы данных

$Sd = \{Ri = <Ui, Fi>, i = 1,2, \ldots n\}$, где

каждое Fi – множество функциональных зависимостей и каждое отношение Ri находится в 3НФ.

Алгоритм.

1. Удаление лишних атрибутов. Каждую функциональную зависимость $X \to A \in F_0$ для каждого атрибута $B \in X$ заменим на функциональную зависимость $X \setminus B \to A$, если существует функциональная зависимость $X \setminus B \to A$.

2. *Поиск неизбыточного покрытия.* Найдем неизбыточное покрытие H для множества функциональных зависимостей, полученных на шаге 1.

3. Построение групп функциональных зависимостей. Все функциональные зависимости из H разобъем на группы функциональных зависимостей с одинаковыми левыми частями. В результате имеем набор групп $H_1, H_2, \ldots H_n$.

4. Объединение групп. Обозначим J - множество функциональных зависимостей. Первоначально $J = \varnothing$. Если X – левая часть функциональных зависимостей из группы Hi , а Y- левая часть функциональных зависимостей из группы Hj и $X \leftrightarrow Y \in H^+$ (т.е. $X \to Y \in H^+$ и $Y \to X \in H^+$ или, другими словами, зависимости

$X \to Y$ и $Y \to X$ выводимы из H), то объединим группы Hi и Hj. Добавим зависимости $X \to Y$ и $Y \to X$ в множество J. Удалим из множества H каждую зависимость вида $X \to A$, где $A \in Y$, и каждую зависимость вида $Y \to B$, где $B \in X$.

5. Устранение транзитивных зависимостей. Построим $H' \in H$ такое, что $(H' + J)^+ = (H + J)^+$ и никакое собственное подмножество H' этим свойством не обладает. Функциональные зависимости из множества J добавим в соответствующие группы $H_1, H_2, \ldots H_n$.

6. Построение схемы базы данных. Для каждой группы Hi построим схему отношения Ri, состоящую из всех атрибутов, входящих в функциональные зависимости этой группы. Каждое множество атрибутов, стоящее в левой части функциональной зависимости является ключом отношения Ri.

7. Дополнение Дрибаса. Слить попарно такие схемы отношений Ri и Rj, у которых $Uj \subseteq Ui$.

Достоинства и недостатки алгоритма Бернштейна

В большинстве случаев алгоритм Бернштейна строит схему базы данных из отношений в БКНФ.

Результирующая схема базы данных эквивалентна исходной схеме базы данных по первому определению но не всегда по второму.

Пример 10.7. Рассмотрим схему отношения

S_0 = {R_0 = <(ПРЕДМЕТ, АУД, ФАКУЛЬТЕТ, ДЕКАН),
{ ПРЕДМЕТ → АУД, ФАКУЛЬТЕТ→ ДЕКАН}>}

По алгоритму Бернштейна получим схему

Sd = {R_1 = <(ПРЕДМЕТ, АУД),
{ ПРЕДМЕТ → АУД }>,
R_2 = <(ФАКУЛЬТЕТ, ДЕКАН),
{ФАКУЛЬТЕТ→ ДЕКАН}>}

Эта схема базы данных не эквивалентна исходной схеме базы данных по второму определению.

Этот недостатот можно устранить, если ввести фиктивный атрибут А и функциональную зависимость
ПРЕДМЕТ ФАКУЛЬТЕТ → А. Тогда по алгоритму Бернштейна получим схему базы данных:

Sd = {R_1 = <(ПРЕДМЕТ, АУД),
{ПРЕДМЕТ → АУД }>,

R_2 = <(ФАКУЛЬТЕТ, ДЕКАН),
{ФАКУЛЬТЕТ→ ДЕКАН}>
R_3 = <(ПРЕДМЕТ, ФАКУЛЬТЕТ, А),
{ПРЕДМЕТ,ФАКУЛЬТЕТ → А }>}

Так как атрибут А – фиктивный, то его можно удалить из схемы отношения R_3.

Алгоритм не строит количественно оптимальную схему.

Алгоритм Бернштейна самый быстрый. Сложность алгоритма оценивается как $O(L^2)$, где L – длина строки для записи всех функциональных зависимостей из F_0.

11. Учет семантики атрибутов при проектировании реляционной схемы базы данных

При проектировании реляционной схемы базы данных необходимо обратить внимание на то, что одни и те же атрибуты, участвующие в разных зависимостях, могут иметь разное по смыслу значение.

Пример 11.1. Рассмотрим схему отношения

S_0 = {R_0 = <(НОМ_ЗАЧ, ФИО, НОМ_ГРУППЫ, НОМ_ИНСТ),
{НОМ_ЗАЧ → ФИО, НОМ_ЗАЧ → НОМ_ГРУППЫ,
НОМ_ГРУППЫ → НОМ_ИНСТ,
НОМ_ЗАЧ → НОМ_ИНСТ}>}

В этом отношении хранится информация о студентах институтов, их принадлежности к конкретной группе, принадлежности группы к некоторому институту, о посещении студентами семинаров других институтов.

Применив алгоритм Бернштейна, получим схему базы данных:

Sd = {R_1 = <(НОМ_ЗАЧ, ФИО, НОМ_ГРУППЫ),
{НОМ_ЗАЧ → ФИО, НОМ_ЗАЧ → НОМ_ГРУППЫ}>,
R_2 = <(НОМ_ГРУППЫ, НОМ_ИНСТ),
{НОМ_ГРУП → НОМ_ИНСТ }> }

В базе данных с этой схемой нельзя хранить данные о том, что студент учится в одном институте, а ходить на семинары может в другой институт. Это произошло, так как по аксиоме транзитивности функциональная зависимость НОМ_ЗАЧ → НОМ_ИНСТ может быть выведена из функциональных зависимостей НОМ_ЗАЧ → НОМ_ГРУППЫ и НОМ_ГРУП → НОМ_ИНСТ.

Чтобы этого не произошло нужно определить смысловое содержание атрибута НОМ_ИНСТ и ввести вместо одного атрибута

два разных атрибута: НОМ_ИНСТ_УЧИТСЯ и НОМ_ИНСТ_ПОСЕЩАЕТ. Тогда исходная схема базы данных будет выглядеть так:

S_0 = {R_0 = <(НОМ_ЗАЧ, ФИО, НОМ_ГРУППЫ,
НОМ_ИНСТ_УЧИТСЯ,
НОМ_ИНСТ_ПОСЕЩАЕТ),
{НОМ_ЗАЧ → ФИО, НОМ_ЗАЧ → НОМ_ГРУППЫ,
НОМ_ГРУППЫ → НОМ_ИНСТ_УЧИТСЯ,
НОМ_ЗАЧ → НОМ_ИНСТ_ПОСЕЩАЕТ}>}

По алгоритму Бернштейна получим схему базы данных:

Sd = {R_1 = <(НОМ_ЗАЧ, ФИО, НОМ_ГРУППЫ),
{НОМ_ЗАЧ → ФИО, НОМ_ЗАЧ → НОМ_ГРУППЫ}>,
R_2 = <(НОМ_ГРУППЫ, НОМ_ИНСТ_УЧИТСЯ),
{НОМ_ГРУП → НОМ_ИНСТ_УЧИТСЯ}>
R_3 = <(НОМ_ЗАЧ, НОМ_ИНСТ_ПОСЕЩАЕТ),
{НОМ_ЗАЧ → НОМ_ИНСТ_ПОСЕЩАЕТ}> }

В базе данных с этой схемой уже можно хранить информацию о том, что студент учится в одном институте, а ходить на семинары может в другой институт.

СОДЕРЖАНИЕ