Теория проектирования реляционных баз данных

При проектировании базы данных решаются две основные проблемы:
1. Каким образом отобразить объекты предметной области в абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области, и было, по возможности, лучшим (эффективным, удобным и т. д.)? Часто эту проблему называют проблемой логического проектирования баз данных.
2. Как обеспечить эффективность выполнения запросов к базе данных? Эту проблему обычно называют проблемой физического проектирования баз данных.

В случае реляционных баз данных нет общих рецептов по части физического проектирования. Здесь слишком много зависит от используемой СУБД. Поэтому ограничимся только существенными вопросами логического проектирования реляционных баз данных. Более того, не будем касаться определения ограничений целостности общего вида, а ограничимся ограничениями первичного и внешнего ключей. Будем считать, что проблема проектирования реляционной базы данных состоит в обоснованном принятии решений о том, из каких отношений должна состоять базы данных, и какие атрибуты должны быть у этих отношений.
Классический подход к проектированию реляционных баз данных заключается в том, что сначала предметная область представляется в виде одного или нескольких отношений, а далее осуществляется процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает свойствами лучшими, чем предыдущая. Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений. Примером набора ограничений является ограничение первой нормальной формы – значения всех атрибутов отношения атомарны. Поскольку требование первой нормальной формы является базовым требованием классической реляционной модели данных, будем считать, что исходный набор отношений уже соответствует этому требованию.
В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

1. первая нормальная форма (1НФ или 1NF);
2. вторая нормальная форма (2НФ или 2NF);
3. третья нормальная форма (3НФ или 3NF);
4. нормальная форма Бойса-Кодда (НФБК или BCNF);
5. четвертая нормальная форма (4НФ или 4NF);
6. пятая нормальная форма, или нормальная форма проекции-соединения (5НФ или 5NF или PJ/NF).

Основные свойства нормальных форм такие:

1. каждая следующая нормальная форма в некотором смысле лучше предыдущей;
2. при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

Процесс проектирования реляционной базы данных на основе метода нормализации преследует две основные цели:
1. избежать избыточности хранения данных;
2. устранить аномалии обновления отношений.

Эти цели являются актуальными для информационных систем оперативной обработки транзакций (On-Line Transaction Processing – OLTP), которым свойственны частые обновления базы данных, и потому аномалии обновления могут сильно вредить

эффективности приложения. В информационных системах оперативной аналитической обработки (On-Line Analytical Processing – OLAP), в частности, в системах поддержки принятия решений, базы данных в основном используются для выборки данных. Поэтому аномалиями обновления можно пренебречь. Из этого не следует, что принципы нормализации непригодны при проектировании баз данных OLAP-приложений. Даже если схема такой базы данных должна быть денормализована по соображениям эффективности, то чтобы получить правильную денормализованную схему, нужно сначала понять, как выглядит нормализованная схема.

В основе метода нормализации лежит декомпозиция отношения, находящегося в предыдущей нормальной форме, в два или более отношения, удовлетворяющих требованиям следующей нормальной формы. Считаются правильными такие декомпозиции отношения, которые обратимы, т. е. имеется возможность собрать исходное отношение из декомпозированных отношений без потери информации.

Наиболее важные на практике нормальные формы отношений основываются на фундаментальном в теории реляционных баз данных понятии функциональной зависимости.

Декомпозиция без потерь

Декомпозицией отношения R называется замена R на совокупность отношений $\{R_1, R_2, \ldots, R_n\}$ такую, что каждое из них есть проекция R, и каждый атрибут R входит хотя бы в одну из проекций декомпозиции.

Например, для отношения R с атрибутами {a, b, c} существуют следующие основные варианты декомпозиции:

- {a}, {b}, {c}
- {a}, {b, c}
- {a, b}, {c}
- {b}, {a, c}
- {a, b}, {b, c}
- {a, b}, {a, c}
- {b, c}, {a, c}
- {a, b}, {b, c}, {a, c}

Рассмотрим теперь отношение R', которое получается в результате операции естественного соединения (NATURAL JOIN), применённой к отношениям, полученным в результате декомпозиции R.

Декомпозиция называется декомпозицией без потерь, если R' в точности совпадает с R.

Неформально говоря, при декомпозиции без потерь отношение «разделяется» на отношения-проекции таким образом, что из полученных проекций возможна «сборка» исходного отношения с помощью операции естественного соединения.

Далеко не всякая декомпозиция является декомпозицией без потерь. Проиллюстрируем это на примере отношения R с атрибутами {a, b, c}, приведённом выше. Пусть отношение R имеет вид:

a	b	c
Москва	Россия	столица
Томск	Россия	не столица
Берлин	Германия	столица

Декомпозиция $R_1 = \{a\}, R_2 = \{b, c\}$ имеет вид:

a
Москва
Томск
Берлин

b	c
Россия	столица
Россия	не столица
Германия	столица

Результат операции соединения этих отношений: $R' = R_1$ NATURAL JOIN R_2

a	b	c
Москва	Россия	столица
Москва	Россия	не столица
Москва	Германия	столица
Томск	Россия	столица
Томск	Россия	не столица
Томск	Германия	столица
Берлин	Россия	столица
Берлин	Россия	не столица
Берлин	Германия	столица

Очевидно, что R' не совпадает с R, а значит такая декомпозиция не является декомпозицией без потерь. Рассмотрим теперь декомпозицию $R_1 = \{a, b\}, R_2 = \{a, c\}$:

a	b
Москва	Россия
Томск	Россия
Берлин	Германия

a	c
Москва	столица
Томск	не столица
Берлин	столица

Такая декомпозиция является декомпозицией без потерь, в чём читатель может убедиться самостоятельно.

В некоторых случаях отношение вовсе невозможно декомпозировать без потерь. Существуют также примеры отношений, для которых нельзя выполнить декомпозицию без потерь на две проекции, но которые можно подвергнуть декомпозиции без потерь на три или большее количество проекций.

Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице. Например, есть таблица «Автомобили»:

Фирма	Модели
BMW	M5, X5M, M1
Nissan	GT-R

Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным. Преобразуем таблицу к 1НФ:

Фирма	Модели
BMW	M5
BMW	X5M
BMW	M1
Nissan	GT-R

Вторая нормальная форма

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК).

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

Например, дана таблица:

Модель	Фирма	Цена	Скидка
M5	BMW	5500000	5 %
X5M	BMW	6000000	5 %
M1	BMW	2500000	5 %
GT-R	Nissan	5000000	10 %

Таблица находится в первой нормальной форме, но не во второй. Цена машины зависит от модели и фирмы. Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

Модель	Фирма	Цена
M5	BMW	5500000
X5M	BMW	6000000
M1	BMW	2500000
GT-R	Nissan	5000000

Фирма	Скидка
BMW	5 %
Nissan	10 %

Третья нормальная форма
Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Рассмотрим таблицу:

Модель	Магазин	Телефон
BMW	Риал-авто	87-33-98
Audi	Риал-авто	87-33-98
Nissan	Некст-Авто	94-54-12

Таблица находится во 2НФ, но не в 3НФ.
В отношении атрибут «Модель» является первичным ключом. Личных телефонов у автомобилей нет, и телефон зависит исключительно от магазина.
Таким образом, в отношении существуют следующие функциональные зависимости: Модель → Магазин, Магазин → Телефон, Модель → Телефон.
Зависимость Модель → Телефон является транзитивной, следовательно, отношение не находится в 3НФ.
В результате разделения исходного отношения получаются два отношения, находящиеся в 3НФ:

Магазин	Телефон
Риал-авто	87-33-98
Некст-Авто	94-54-12

Модель	Магазин
BMW	Риал-авто
Audi	Риал-авто
Nissan	Некст-Авто

Нормальная форма Бойса-Кодда (НФБК) (частная форма третьей нормальной формы)
Определение 3НФ не совсем подходит для следующих отношений:
1) отношение имеет два или более потенциальных ключа;
2) два и более потенциальных ключа являются составными;
3) они пересекаются, т.е. имеют хотя бы один общий атрибут.

Для отношений, имеющих один потенциальный ключ (первичный), НФБК является 3НФ.
Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.
Предположим, рассматривается отношение, представляющее данные о бронировании стоянки на день:

Номер стоянки	Время начала	Время окончания	Тариф
1	09:30	10:30	Бережливый
1	11:00	12:00	Бережливый
1	14:00	15:30	Стандарт
2	10:00	12:00	Премиум-В
2	12:00	14:00	Премиум-В
2	15:00	18:00	Премиум-А

Тариф имеет уникальное название и зависит от выбранной стоянки и наличии льгот, в частности:

- «Бережливый»: стоянка 1 для льготников
- «Стандарт»: стоянка 1 для не льготников
- «Премиум-А»: стоянка 2 для льготников
- «Премиум-В»: стоянка 2 для не льготников.

Таким образом, возможны следующие составные первичные ключи: {Номер стоянки, Время начала}, {Номер стоянки, Время окончания}, {Тариф, Время начала}, {Тариф, Время окончания}.

Отношение находится в 3НФ. Требования второй нормальной формы выполняются, так как все атрибуты входят в какой-то из потенциальных ключей, а неключевых атрибутов в отношении нет. Также нет и транзитивных зависимостей, что соответствует требованиям третьей нормальной формы. Тем не менее, существует функциональная зависимость Тариф → Номер стоянки, в которой левая часть (детерминант) не является потенциальным ключом отношения, то есть отношение не находится в нормальной форме Бойса — Кодда.

Недостатком данной структуры является то, что, например, по ошибке можно приписать тариф «Бережливый» к бронированию второй стоянки, хотя он может относиться только к первой стоянки.

Можно улучшить структуру с помощью декомпозиции отношения на два и добавления атрибута Имеет льготы, получив отношения, удовлетворяющие НФБК (подчёркнуты атрибуты, входящие в первичный ключ.):

Тариф	Время начала	Время окончания
Бережливый	09:30	10:30
Бережливый	11:00	12:00
Стандарт	14:00	15:30
Премиум-В	10:00	12:00
Премиум-В	12:00	14:00
Премиум-А	15:00	18:00

Тариф	Номер	Имеет
Бережливый	1	Да
Стандарт	1	Нет
Премиум-А	2	Да
Премиум-В	2	Нет

Четвертая нормальная форма
Отношение находится в 4НФ, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей.

В отношении R (A, B, C) существует многозначная зависимость R.A -> -> R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

Предположим, что рестораны производят разные виды пиццы, а службы доставки ресторанов работают только в определенных районах города. Составной первичный ключ соответствующей переменной отношения включает три атрибута: {Ресторан, Вид пиццы, Район доставки}.

Ресторан	Вид пиццы	Район доставки
A1 Пицца	Толстая корка	Springfield
A1 Пицца	Толстая корка	Shelbyville
A1 Пицца	Толстая корка	Столица
A1 Пицца	Фаршированная корочка	Springfield
A1 Пицца	Фаршированная корочка	Shelbyville
A1 Пицца	Фаршированная корочка	Столица
Элитная пицца	Толстая корка	Столица
Элитная пицца	Фаршированная корочка	Столица
Пицца Винченцо	Толстая корка	Springfield
Пицца Винченцо	Толстая корка	Shelbyville
Пицца Винченцо	Толстая корка	Springfield
Пицца Винченцо	Толстая корка	Shelbyville

Такая переменная отношения не соответствует 4НФ, так как существует следующая многозначная зависимость:
{Ресторан} → {Вид пиццы}
{Ресторан} → {Район доставки}

То есть, например, при добавлении нового вида пиццы придется внести по одному новому кортежу для каждого района доставки. Возможна логическая аномалия, при которой определенному виду пиццы будут соответствовать лишь некоторые районы доставки из обслуживаемых рестораном районов.

Для предотвращения аномалии нужно декомпозировать отношение, разместив независимые факты в разных отношениях. В данном примере следует выполнить декомпозицию на {Ресторан, Вид пиццы} и {Ресторан, Район доставки}.

Однако, если к исходной переменной отношения добавить атрибут, функционально зависящий от потенциального ключа, например цену с учётом стоимости доставки ({Ресторан, Вид

пиццы, Район доставки} → Цена), то полученное отношение будет находиться в 4НФ и его уже нельзя подвергнуть декомпозиции без потерь.

Ресторан	Район доставки
A1 Пицца	Springfield
A1 Пицца	Shelbyville
A1 Пицца	Столица
Элитная пицца	Столица
Пицца Винченцо	Springfield
Пицца Винченцо	Shelbyville

Ресторан	Вид пиццы
A1 Пицца	Толстая корка
A1 Пицца	Фаршированная корочка
Элитная пицца	Толстая корка
Элитная пицца	Фаршированная корочка
Пицца Винченцо	Толстая корка

Пятая нормальная форма

Отношение находится в пятой нормальной форме (иначе — в проекционно-соединительной нормальной форме) тогда и только тогда, когда каждая нетривиальная зависимость соединения в нём определяется потенциальным ключом (ключами) этого отношения.

Это очень жесткое требование, которое можно выполнить лишь при дополнительных условиях. На практике трудно найти пример реализации этого требования в чистом виде.
Предположим, что нужно хранить данные об ассортименте нескольких продавцов, торгующих продукцией нескольких фирм (номенклатура товаров фирм может пересекаться):

Продавец	Фирма	Товар
Иванов	Рога и Копыта	Пылесос
Иванов	Рога и Копыта	Хлебница
Петров	Безенчук&Ко	Сучкорез
Петров	Безенчук&Ко	Пылесос
Петров	Безенчук&Ко	Хлебница
Петров	Безенчук&Ко	Зонт
Сидоров	Безенчук&Ко	Пылесос
Сидоров	Безенчук&Ко	Телескоп
Сидоров	Рога и Копыта	Пылесос
Сидоров	Рога и Копыта	Лампа
Сидоров	Геркулес	Вешалка

Если дополнительных условий нет, то данное отношение, которое находится в 4-й нормальной форме, является корректным и отражает все необходимые ограничения.

Теперь предположим, что нужно учесть следующее ограничение: каждый продавец имеет в своём ассортименте ограниченный список фирм и ограниченный список типов товаров и предлагает товары из списка товаров, производимые фирмами из списка фирм.

То есть продавец не имеет право торговать какими угодно товарами каких угодно фирм. Если продавец П имеет право торговать товарами фирмы Ф, и если продавец П имеет право

торговать товарами типа Т, то в этом случае в ассортимент продавца П входят товары типа Т фирмы Ф при условии, что фирма Ф производит товары типа Т.

Такое ограничение может быть вызвано, например, тем, что список типов товаров продавца ограничен имеющимися у него лицензиями, либо знаниями и квалификацией, необходимыми для их продажи, а список фирм каждого продавца определён партнёрскими соглашениями.

В рассматриваемом примере, в частности, предполагается, что продавец Иванов имеет право торговать товарами только фирмы «Рога и Копыта», продавец Петров — товарами только фирмы «Безенчук&Ко», зато продавец Сидоров не имеет права торговать хлебницами и сучкорезами и т. д.

Предложенное выше отношение не может исключить ситуации, при которых данное ограничение будет нарушено. Ничто не препятствует занести данные о торговле товаром, который данная фирма вообще не выпускает, либо данные о торговле товарами той фирмы, которую данный продавец не обслуживает, либо данные о торговле таким типом товара, который данный продавец не имеет право продавать.

Отношение не находится в 5NF, поскольку в нём есть нетривиальная зависимость соединения *{{Продавец, Фирма}, {Фирма, Товар}, {Продавец, Товар}}, однако подмножества {Продавец, Фирма}, {Фирма, Товар}, {Продавец, Товар} не являются суперключами исходного отношения.

В данном случае для приведения к 5NF отношение должно быть разбито на три: {Продавец, Фирма}, {Фирма, Товар}, {Продавец, Товар}.

Продавец	Фирма
Иванов	Рога и Копыта
Петров	Безенчук&Ко
Сидоров	Безенчук&Ко
Сидоров	Рога и Копыта
Сидоров	Геркулес

Фирма	Товар
Рога и Копыта	Пылесос
Рога и Копыта	Хлебница
Рога и Копыта	Лампа
Безенчук&Ко	Сучкорез
Безенчук&Ко	Пылесос
Безенчук&Ко	Хлебница
Безенчук&Ко	Зонт
Безенчук&Ко	Телескоп
Геркулес	Вешалка

Продавец	Товар
Иванов	Пылесос
Иванов	Хлебница
Петров	Сучкорез
Петров	Пылесос
Петров	Хлебница
Петров	Зонт
Сидоров	Телескоп
Сидоров	Пылесос
Сидоров	Лампа
Сидоров	Вешалка

Шестая нормальная форма

Переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ.

Идея «декомпозиции до конца» выдвигалась до начала исследований в области хронологических данных, но не нашла поддержки. Однако для хронологических баз данных максимально возможная декомпозиция позволяет бороться с избыточностью и упрощает поддержание целостности базы данных.

Для хронологических баз данных определены U_операторы, которые распаковывают отношения по указанным атрибутам, выполняют соответствующую операцию и упаковывают полученный результат. В данном примере соединение проекций отношения должно производится при помощи оператора U_JOIN.

Таб.№	Время	Должность	Домашний адрес
6575	01-01-2000:10-02-2003	слесарь	ул.Ленина,10
6575	11-02-2003:15-06-2006	слесарь	ул.Советская,22
6575	16-06-2006:05-03-2009	бригадир	ул.Советская,22

Переменная отношения «Работники» не находится в 6НФ и может быть подвергнута декомпозиции на переменные отношения «Должности работников» и «Домашние адреса работников».

Таб.№	Время	Должность
6575	01-01-2000: 10-02-2003	слесарь
6575	16-06-2006: 05-03-2009	бригадир

Таб.№	Время	Домашний адрес
6575	01-01-2000: 10-02-2003	ул.Ленина,10
6575	11-02-2003: 15-06-2006	ул.Советская, 22

Функциональные зависимости

Для демонстрации основных идей данной темы, будет использоваться несколько измененная версия отношения поставок SP, содержащая атрибут City, представляющий город соответствующего поставщика. Это измененное отношение будет называться SCP.

Sno	City	Pno	Qty
1	Смоленск	1	100
1	Смоленск	2	100
2	Владимир	1	200
2	Владимир	2	200
3	Владимир	2	300
4	Смоленск	2	400
4	Смоленск	4	400
4	Смоленск	5	400

Следует четко различать:
1. значение этого отношения в определенный момент времени;
2. и набор всех возможных значений, которые данное отношение может принимать в различные моменты времени.

Примеры ФЗ, которым удовлетворяет отношение SCP в данном состоянии:
{ Sno } -> {City}
{Sno,Pno} -> {Qty}
{Sno,Pno} -> {City}
{Sno,Pno} -> { City, Qty }
{Sno,Pno} -> {Sno}
{Sno,Pno} -> { Sno, Pno, City, Qty }
{ Sno } -> {Qty}
{ Qty } -> {Sno}

Определение 1. Пусть R - это отношение, а X и Y - произвольные подмножества множества атрибутов отношения R. Тогда Y функционально зависимо от X, что в символическом виде записывается как X -> Y тогда и только тогда? Когда любое значение множества X связано в точности с одним значением множества Y.

Левая и правая стороны ФЗ будут называются детерминантом и зависимой частью соответственно.

Определение 2. Пусть R является переменной-отношением, а X и Y - произвольными подмножествами множества атрибутов переменной-отношения R. Тогда X -> Y тогда и только тогда, когда для любого допустимого значения отношения R любое значение X связано в точности с одним значением Y.

Определение 2а. Пусть R(A1, A2, ..., An) - схема отношения. Функциональная зависимость, обозначаемая X -> Y между двумя наборами атрибутов X и Y, которые являются подмножествами R определяет ограничение на возможность существования кортежа в некотором отношении r. Ограничение означает, что для любых двух кортежей t1 и t2 в r, для которых имеет место t1[X] = t2[X], также имеет место t1[Y] = t2[Y].

1. Если ограничение на схеме отношения R утверждает, что не может быть более одного кортежа со значением атрибутов X в любом отношении экземпляре отношения r, то X является потенциальным ключом R. Это означает, что X -> Y для любого подмножества атрибутов Y из R. Если X является потенциальным ключ R, то X -> R.
2. Если X -> Y в R, это не означает, что Y -> X в R.

Функциональная зависимость является семантическим свойством, т. е. свойством значения атрибутов.
Примеры безотносительных ко времени ФЗ для переменной-отношения SCP:

{Sno,Pno} -> {Sno,Pno} -> {Sno,Pno} -> {Sno,Pno} -> {Sno,Pno} -> { Sno } ->
{Qty}
{City}
{ City, Qty }
{Sno}
{ Sno, Pno, City, Qty } {City}

Следует обратить внимание на ФЗ, которые выполняются для отношения SCP, но не выполняются «всегда» для переменной-отношения SCP:

{Sno} -> {Qty} {Qty} -> {Sno}
Если X является потенциальным ключом переменной-отношения R, то все атрибуты Y переменной-отношения R должны быть обязательно ФЗ от X (это следует из определения потенциального ключа). Если переменная-отношение R удовлетворяет ФЗ X -> Y и X не является потенциальным ключом, то R будет характеризоваться некоторой избыточностью. Например, в случае отношения SCP сведения о том, что каждый данный поставщик находится в данном городе, будут повторяться много раз (это хорошо видно из таблицы).
Эта избыточность приводит к разным аномалиям обновления, получившим такое название по историческим причинам. Под этим понимаются определенные трудности, появляющиеся при выполнении операций обновления INSERT, DELETE и UPDATE. Рассмотрим избыточность, соответствующую ФЗ (Sno -> City). Ниже поясняются проблемы, которые возникнут при выполнении каждой из указанных операций обновления.

1. Операция INSERT. Нельзя поместить в переменную-отношение SCP информацию о том, что некоторый поставщик находится в определенном городе, не указав сведения хотя бы об одной детали, поставляемой этим поставщиком.
2. Операция DELETE. Если из переменной-отношения SCP удалить кортеж, который является единственным для некоторого поставщика, будет удалена не только информация о поставке поставщиком некоторой детали, но также информация о том, что этот поставщик находится в определенном городе. В действительности проблема заключается в том, что в переменной-отношении SCP содержится слишком много собранной в одном месте информации, поэтому при удалении некоторого кортежа теряется слишком много информации.
3. Операция UPDATE. Название города для каждого поставщика повторяется в переменной-отношении SCP несколько раз, и эта избыточность приводит к возникновению проблем при обновлении.

Для решения всех этих проблем, как предлагалось выше, необходимо выполнить декомпозицию переменной-отношения SCP на две следующие переменные-отношения.
S' { Sno, City }
SP { Sno, Pno, Qty }

Даже если ограничиться рассмотрением ФЗ, которые выполняются «всегда», множество ФЗ, выполняющихся для всех допустимых значений данного отношения, может быть все еще очень большим. Поэтому встает задача сокращения множества ФЗ до компактных размеров. Важность этой задачи вытекает из того, что ФЗ являются ограничениями целостности. Поэтому при каждом обновлении данных в СУБД все они должны быть проверены. Следовательно, для заданного множества ФЗ S желательно найти такое множество T, которое (в идеальной ситуации) было бы гораздо меньшего размера, чем множество S, причем каждая ФЗ множества S могла бы быть заменена ФЗ множества T. Если бы такое множество T было найдено, то в СУБД достаточно было бы использовать ФЗ из множества T, а ФЗ из множества S подразумевались бы автоматически.

Очевидным способом сокращения размера множества ФЗ было бы исключение тривиальных зависимостей, т. е. таких, которые не могут не выполняться. Примером тривиальной ФЗ для отношения SCP может быть { Sno, Pno } -> { Sno }.

Определение 3. ФЗ (X -> Y) тривиальная тогда и только тогда, когда Y включается в X.
Одни ФЗ могут подразумевать другие ФЗ. Например, зависимость
{ Sno, Pno } -> { City, Qty }

подразумевает следующие ФЗ

{ Sno, Pno } -> City { Sno, Pno } -> Qty

В качестве более сложного примера можно привести переменную-отношение R с атрибутами A, B и C, для которых выполняются ФЗ (A -> B) и (B -> C). В этом случае также выполняется ФЗ (A -> C), которая называется транзитивной ФЗ.

Определение 4. Множество всех ФЗ, которые задаются данным множеством ФЗ S, называется замыканием S и обозначается символом S+.

Из сказанного выше становится ясно, что для выполнения сформулированной задачи следует найти способ вычисления S+ на, основе S.

Первая попытка решить эту проблему принадлежит Армстронгу (Armstrong), который предложил набор правил вывода новых ФЗ на основе заданных (эти правила также называются аксиомами Армстронга).

Пусть в перечисленных ниже правилах A, B и C – произвольные подмножества множества атрибутов заданной переменной-отношения R, а символическая запись AB означает { A, B }. Тогда правила вывода определяются следующим образом.

1. Правило **рефлексивности**: (B принадлежит A) следовательно (A -> B)
2. Правило **дополнения**: (A -> B) следовательно AC -> BC
3. Правило **транзитивности**: (A -> B) и (B -> C) следовательно (A -> C)

Каждое и этих правил может быть непосредственно доказано на основе определения ФЗ. Более того, эти правила являются полными в том смысле, что для заданного множества ФЗ S минимальный набор ФЗ, которые подразумевают все зависимости из множества S, может быть выведен из S на основе этих правил. Они также являются исчерпывающими, поскольку никакие дополнительные ФЗ (т.е. ФЗ, которые не подразумеваются ФЗ множества S) с их помощью не могут быть выведены. Иначе говоря, эти правила могут быть использованы для получения замыкания S+.

Из трех описанных выше правил для упрощения задачи практического вычисления замыкания S+ можно вывести несколько дополнительных правил. (Примем, что D - это другое произвольное подмножество множества атрибутов R.)

4. Правило **самоопределения**: A -> A
5. Правило **декомпозиции**: (A -> BC) следовательно (A -> B) и (A -> C)
6. Правило **объединения**: (A -> B) и (A -> C) следовательно (A -> BC)
7. Правило **композиции**: (A -> B) и (C -> D) следовательно (AC -> BD)

Кроме того, Дарвен (Darwen) доказал следующее правило, которое назвал общей теоремой объединения:

8. (A->B) и (C->D) следовательно (A(C–B)->BD).

Упражнение. Пусть дана некоторая переменная-отношение R с атрибутами A, B, C, D, E, F и следующими ФЗ:
A -> BC
B -> E
CD -> EF
Показать, что для переменной-отношения R также выполняется ФЗ (AD -> F), которая вследствие этого принадлежит замыканию заданного множества ФЗ.

1. A -> BC (дано)
2. A -> C (следует из п. 1 согласно правилу декомпозиции)
3. AD -> CD (следует из п. 2 согласно правилу дополнения)
4. CD -> EF (дано)
5. AD -> EF (следует из п. 3 и 4 согласно правилу транзитивности)
6. AD -> F (следует из п. 5 согласно правилу декомпозиции)

Хотя эффективный алгоритм для вычисления замыкания S+ на основе множества ФЗ S так и не сформулирован, можно описать алгоритм, который определяет, будет ли данная ФЗ находиться в данном замыкании. Для этого, прежде всего, следует дать определение понятию **суперключ**.

Определение 5. Суперключ переменной-отношения R - это множество атрибутов переменной-отношения R, которое содержит в виде подмножества (но не обязательно собственного подмножества), по крайней мере, один потенциальный ключ.

Из этого определения следует, что суперключи для данной переменной-отношения R - это такие подмножества K множества атрибутов переменной-отношения R, что ФЗ (K -> A) истинна для каждого атрибута A переменной- отношения R.

Предположим, что известны некоторые ФЗ, выполняющиеся для данной переменной-отношения, и требуется определить потенциальные ключи этой переменной-отношения. По определению потенциальными ключами называются неприводимые суперключи. Таким образом, выясняя, является ли данное множество атрибутов K суперключом, можно в значительной степени продвинуться к выяснению вопроса, является ли K потенциальным ключом. Для этого нужно определить, будет ли набор атрибутов переменной-отношения R множеством всех атрибутов, функционально зависящих от K. Таким образом, для заданного множества зависимостей S, которые выполняются для переменной-отношения R, необходимо найти способ определения множества всех атрибутов переменной-отношения R, которые функционально зависимы от K, т.е. так называемое замыкание K^+ множества K для S.

Алгоритм вычисления этого замыкания имеет вид.

```
function Closure(K: SetOfAttr; S: SetOfFD): SetOfAttr;
var
        Jold, Jnew: SetOfAttr;
begin
        Jnew := K;
        repeat
                Jold := Jnew;
                foreach ((X -> Y in S) do
                        if (X принадлежит Jnew) then Jnew=Jnew + Y;
        until (Jold = Jnew);
        return Jold;
end; // of Closure
```

Определение. Суперключ в схеме отношения R(A1, A2, ... , An) - это набор атрибутов S из R со свойством, что ни для каких двух кортежей t1 и t2 в любом отношении над схемой R не будет выполняться равенство t1[S] = t2[S].

Определение. Потенциальный ключ K - это суперключ с дополнительным свойством: удаление любого атрибута из K приведет к тому, что K перестанет быть суперключом.

Определение. Атрибут в схеме отношения R называется первичным атрибутом R, если он является членом некоторого потенциального ключа R. Атрибут называется непервичным, если он не является первичным атрибутом, то есть, если он не является членом какого-либо потенциального ключа.

Алгоритм нахождения ключа K схемы отношения R для заданного множество функциональных зависимостей F

Вход: Схема отношения R и множество функциональных зависимостей F на атрибутах R.

1. Пусть K := R.

2. Для каждого атрибута из K
   {
   вычислить Closure({K - A), F);
   если замыкание содержит все атрибуты из R, то установите K: = K - {A}
   };

Заметьте, что алгоритм определяет только один ключ из множества возможных ключей на R; возвращаемый ключ зависит от порядка, в котором атрибуты удаляются из R на шаге 2.

Упражнение. Пусть дана переменная-отношение R с атрибутами A, B, C, D, E и F и следующими ФЗ:
A -> BC
E -> CF
B -> E
CD -> EF
Вычислить замыкание $\{ A, B \}^+$ множества атрибутов { A, B }, исходя из заданного множества ФЗ.

1. Присвоим Jold и Jnew начальное значение - множество {A,B}.
2. Выполним внутренний цикл четыре раза - по одному разу для каждой заданной ФЗ. На первой итерации (для зависимости A -> BC) будет обнаружено, что левая часть действительно является подмножеством замыкания Jnew. Таким образом, к Jnew можно добавить атрибуты B и C. Jnew теперь представляет собой множество {A,B,C}.
3. На второй итерации (для зависимости E -> CF) обнаруживается, что левая часть не является подмножеством полученного до этого момента результата, который, таким образом, остается неизменным.
4. На третьей итерации (для зависимости B -> E) к Jnew будет добавлено множество E, которое теперь будет иметь вид {A, B, C, E}.
5. На четвертой итерации (для зависимости CD -> EF) Jnew останется неизменным.
6. Далее внутренний цикл выполняется еще четыре раза. На первой итерации результат останется прежним, на
   второй он будет расширен до {A, B, C, E, F}, а на третьей и четвертой - снова не изменится.
7. Наконец после еще одного четырехкратного прохождения цикла замыкание Jnew останется неизменным и весь
   процесс завершится с результатом $\{A, B\}^+ = \{A, B, C, E, F\}$. // Конец упр.

Выводы.

1. Для заданного множества ФЗ S легко можно указать, будет ли заданная ФЗ (X -> Y) следовать из S, поскольку это возможно тогда и только тогда, когда множество Y является подмножеством замыкания X^+ множества X для заданного множества S. Таким образом, представлен простой способ определения, будет ли данная ФЗ (X->Y) включена в замыкание S^+ множества S.
2. Напомним определение понятия суперключа. Суперключ переменной-отношения R - это множество атрибутов переменной-отношения R, которое в виде подмножества (но необязательно собственного подмножества) содержит по крайней мере один потенциальный ключ. Из этого определения прямо следует, что суперключи для данной пеменной-отношения R - это такие подмножества K множества атрибутов переменной-отношения R, что ФЗ (K -> A) будет истинна для каждого атрибута A переменной-отношения R. Другими словами, множество K является суперключ тогда и только тогда, когда замыкание K^+ для множества K в пределах заданного мноства ФЗ является множеством абсолютно всех атрибутов переменной-отношения R. (Кроме того, множество K является потенциальным ключом тогда и только тогда, когда оно является неприводимым суперключом).

Определение 6. Два множества ФЗ S1 и S2 эквивалентны тогда и только тогда, когда они являются покрытиями друг для друга, т. е. S1+ = S2+.

Упражнение.
Эквивалентны ли следующие множества ФЗ: F= {A ->C, AC ->D,E ->AD, E ->H} and G = {A ->CD, E ->AH}.
Проверим, что G покрывается множеством F и F покрывается множеством G

G покрывается множеством F:
{A}+ = {A, C, D} (относительно F), покрывается A ->CD из G
{E}+ = {E, A, D, H, C} (относительно F), покрывается E ->AH в G

F покрывается множеством G:
{A}+ = {A, C, D} (относительно G), покрывается A ->C в F

{A, C}+= {A, C, D} (относительно G), покрывается AC ->D в F
{E}+ = {E, A, H, C, D} (относительно G), покрывается E ->AD и E ->H в F

Каждое множество ФЗ эквивалентно, по крайней мере, одному неприводимому множеству.

Определение 7. Множество ФЗ является неприводимым тогда и только тогда, когда оно обладает всеми перечисленными ниже свойствами.
1. Каждая ФЗ этого множества имеет одноэлементную правую часть.
2. Ни одна ФЗ множества не может быть устранена без изменения замыкания этого множества.
3. Ни один атрибут не может быть устранен из левой части любой ФЗ данного множества без изменения замыкания множества.
   Если I является неприводимым множеством, которое эквивалентно множеству S, то проверка выполнения ФЗ из множества I автоматически обеспечит выполнение ФЗ из множества S.

Пример. Рассмотрим переменную-отношение деталей P с функциональными зависимостями, перечисленными ниже.

Pno -> Pname
Pno -> Color
Pno -> Weight
Pno -> City

Нетрудно заметить, что это множество функциональных зависимостей является неприводимым:
1. правая часть каждой зависимости содержит только один атрибут,
2. левая часть, очевидно, является неприводимой,
3. ни одна из функциональных зависимостей не может быть опущена без изменения замыкания множества (т. е. без
   утраты некоторой информации).

В противоположность этому приведенные ниже множества функциональных зависимостей не являются неприводимыми.

1. Pno -> { Pname, Color }
   Pno -> Weight
   Pno -> City
(Правая часть первой ФЗ не является одноэлементным множеством.)

2. { Pno, Pname } -> Color
   Pno -> Pname
   Pno -> Weight
   Pno -> City
(Первую ФЗ можно упростить, опустив атрибут Pname в левой части без изменения замыкания, т. е. она не является неприводимой слева.)

3. Pno -> Pno
   Pno -> Pname
   Pno -> Color
   Pno -> Weight
   Pno -> City
(Здесь первую ФЗ можно опустить без изменения замыкания.)

Можно сделать утверждение, что для любого множества ФЗ существует, по крайней мере, одно эквивалентное множество, которое является неприводимым. Это достаточно легко продемонстрировать на следующем примере. Пусть дано исходное множество ФЗ S. Тогда благодаря правилу декомпозиции можно без утраты общности предположить, что каждая ФЗ в этом множестве S имеет одноэлементную правую часть. Далее для каждой ФЗ f из этого множества S следует проверить каждый атрибут A в левой части зависимости f. Если

множество S и множество зависимостей, полученное в результате устранения атрибута A в левой части зависимости f, эквивалентны, значит этот атрибут следует удалить. Затем для каждой оставшейся во множестве S зависимости f, если множества S и S \ {f} эквивалентны, следует удалить зависимость f из множества S. Получившееся в результате таких действий множество S является неприводимым и эквивалентно исходному множеству S.

Алгоритм поиска минимального покрытия F для множества функциональных зависимостей E

Вход: Множество функциональных зависимостей E.

1. Пусть F: = E.
2. Заменить каждую функциональную зависимость X -> {A1, A2, ..., An} из F на n функциональных зависимостей X -> A1, X -> A2, ..., X -> An.
3. Для каждой функциональной зависимости X -> A из F
    1. Для каждого атрибута B, который является элементом X
        1. если { {F – {X -> A} } объединить { (X – {B}) -> A} } эквивалентно F заменить X -> A на (X – {B}) -> A в F.
4. Для каждой оставшейся функциональной зависимости X -> A из F
    1. если {F - {X -> A}} эквивалентно F, удалить X -> A из F.

Упражнение. Пусть дана переменная-отношение R с атрибутами A, B, C, D и следующими функциональными зависимостями.

A -> BC
B -> C
A -> B
AB -> C
AC -> D

Найти неприводимое множество функциональных зависимостей эквивалентное данному множеству.

1. Прежде всего, следует переписать заданные ФЗ таким образом, чтобы каждая из них имела одноэлементную правую часть.

A -> B
A -> C
B -> C
A -> B
AB -> C
AC -> D

Нетрудно заметить, что зависимость A -> B записана дважды, так что одну из них можно удалить.

2. Затем в левой части зависимости AC -> D может быть опущен атрибут C, поскольку дана зависимость A -> C, из которой по правилу дополнения можно получить зависимость A -> AC. Кроме того, дана зависимость AC -> D, из которой по правилу транзитивности можно получить зависимость A -> D. Таким образом, атрибут C в левой части исходной зависимости AC -> D является избыточным.
3. Далее можно заметить, что зависимость AB -> C может быть исключена, поскольку дана зависимость A -> C, из которой по правилу дополнения можно получить зависимость AB -> CB, а затем по правилу декомпозиции - зависимость AB -> C.
4. Наконец зависимость A -> C подразумевается зависимостями A -> B и B -> C, так что она также может быть отброшена. В результате получается неприводимое множество зависимостей.

A -> B
B -> C
A -> D

Множество ФЗ I, которое неприводимо и эквивалентно другому множеству ФЗ S, называется неприводимым покрытием множества S. Таким образом, с тем же успехом в системе вместо

исходного множества ФЗ S может использоваться его неприводимое покрытие I (здесь следует повторить, что для вычисления неприводимого эквивалентного покрытия I необязательно вычислять замыкание S+). Однако необходимо отметить, что для заданного множества ФЗ не всегда существует уникальное неприводимое покрытие.

Проектирование базы данных может быть выполнено с использованием двух подходов: снизу вверх (bottom-up) или сверху вниз (top-down).

1. Методология проектирования bottom-up (также называемая методологией синтеза) рассматривает основные связи между отдельными атрибутами в качестве отправной точки и использует их, чтобы построить схемы отношений схем. Этот подход не пользуется популярностью на практике, потому что она страдает от проблемы того, чтобы собрать большое число бинарных связей между атрибутами в качестве отправной точки. На пактике это сделать почти невозможно.
2. Методология проектирования top-down (также называемая методологией анализа) начинается с некоторого набора отношений, состоящих из атрибутов. Затем отношения анализируются отдельно или совместно, в результате чего происходит их декомпозиция до тех пор, пока не будут достигнуты все желаемые свойства.
   Неявными целями обеих методологий являются сохранение информации и минимизация избыточности.

Процесс нормализации схем отношений, основанный на операции декомпозии, должен обладать следующими свойства:

1. неаддитивностью JOIN или JOIN без потерь информации (NJP) , которая гарантирует, что в результате декомпозиции не появятся лишние кортежи.
2. свойством сохранения зависимостей (DPP), которое гарантирует, что каждая функциональная зависимость будет представлена в каком-либо отдельном отношении после декомпозиции.

Свойство NJP является очень критическим и должно быть достигнуто любой ценой. Свойство DPP желательно, но не всегда достижимо.