Лекция № 10. Нормальные формы

1. Смысл нормализации схем баз данных

Понятие, которое мы будем рассматривать в данном разделе, связано с понятием функциональных зависимостей, т. е. смысл нормализации схем баз данных неразрывно связан с понятием ограничений, накладываемых системой функциональных зависимостей, и во многом следует из этого понятия.

Исходной точкой любого проектирования базы данных является представление предметной области в виде одного или нескольких отношений, и на каждом шаге проектирования производится некоторый набор схем отношений, обладающих «улучшенными» свойствами. Таким образом, процесс проектирования представляет собой процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает свойствами, в некотором смысле лучшими, чем предыдущая.

Каждой нормальной форме соответствует определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений. Примером может служить ограничение первой нормальной формы – значения всех атрибутов отношения атомарны.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

1) первая нормальная форма (1 NF);

2) вторая нормальная форма (2 NF);

3) третья нормальная форма (3 NF);

4) нормальная форма Бойса – Кодда (BCNF);

5) четвертая нормальная форма (4 NF);

6) пятая нормальная форма, или нормальная форма проекции-соединения (5 NF или PJ/NF).

(В данный курс лекций включается подробное рассмотрение первых четырех нормальных форм базовых отношений, поэтому мы не будем подробно разбирать четвертую и пятую нормальные формы.)

Основные свойства нормальных форм состоят в следующем:

1) каждая следующая нормальная форма в некотором смысле лучше предыдущей нормальной формы;

2) при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

В основе процесса проектирования лежит метод нормализации, т. е. декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, которые удовлетворяют требованиям следующей нормальной формы (с этим мы столкнемся, когда нам самим придется по мере прохождения материала проводить нормализацию того или иного базового отношения).

Как уже упоминалось в разделе, посвященном созданию базовых отношений, заданные множества функциональных зависимостей, накладывают соответствующие ограничения на схемы базовых отношений. Эти ограничения в общем случае реализуются двумя методами:

1) декларативно, т. е. с помощью объявления в базовом отношении различного вида первичных, кандидатных и внешних ключей (это метод, получивший наибольшее распространение);

2) процедурно, т. е. написанием программного кода (использованием упомянутых выше так называемых триггеров).

При помощи простой логики можно понять, в чем же заключается смысл нормализации схем баз данных. Нормализовывать базы данных или приводить базы данных к нормальному виду – это значит определять такие схемы базовых отношений, чтобы максимально уменьшить необходимость написания программного кода, увеличить производительность работы базы данных, облегчить поддержку целостности данных по состоянию и ссылочной целостности. То есть сделать код и работу с ним максимально простой и удобной разработчикам и пользователям.

Для того чтобы наглядно в сравнении продемонстрировать работу ненормализованной и нормализованной базы данных, рассмотрим следующий пример.

Пусть у нас имеется базовое отношение, содержащее информацию о результатах экзаменационной сессии. Такую базу данных мы уже рассматривали раньше.

Итак, **вариант 1** схемы базы данных.

> Сессия (***№ зачетной книжки***, Фамилия, Имя, Отчество,
> ***Предмет***, Оценка)

В этом отношении, как видно из изображения схемы базового отношения, задан составной первичный ключ:

Primary key (№ зачетной книжки, Предмет);

Также в этом отношении задана система функциональных зависимостей:

{№ зачетной книжки} > {Фамилия, Имя, Отчество};

Приведем табличный вид небольшого фрагмента базы данных с данной схемой отношения. Этот фрагмент мы уже применяли в рассмотрении ограничений функциональных зависимостей, поэтому на его примере нам будет довольно легко понять и данную тему.

Сессия					
№ зачетной книжки	**Фамилия**	**Имя**	**Отчество**	**Предмет**	**Оценка**
100	Тюряпина	Елизавета	Николаевна	Информатика	Удовлетворительно
…	…	…	…	…	…
1100	Михайлова	Анастасия	Александровна	Базы данных	Отлично
…	…	…	…	…	…
100	Тюряпина	Елизавета	Николаевна	Литература	Отлично

Здесь для поддержания целостности данных по состоянию, т. е. для выполнения ограничения системы функциональной зависимости {№ зачетной книжки} > {Фамилия, Имя, Отчество} при изменении, например, фамилии необходимо просматривать все кортежи этого базового отношения и последовательно вводить необходимые изменения. Однако так как это довольно громоздкий и трудоемкий процесс (особенно если мы имеем дело с базой данных большого учебного заведения), разработчики систем управления базами данных пришли к выводу, что этот процесс необходимо автоматизировать, т. е. сделать автоматическим. Теперь контроль выполнения этой (и любой другой) функциональной зависимости можно организовывать автоматически при помощи правильного объявления в базовом отношении различных ключей и так называемой декомпозиции (т. е. разбиения чего-либо на несколько самостоятельных частей) этого отношения.

Итак, нашу имеющуюся схему отношения «Сессия» разобьем на две схемы: схему «Студенты», содержащую только информацию о студентах данного учебного заведения, и схему «Сессия», содержащую информацию о последней прошедшей сессии. А затем объявим ключи таким образом, чтобы можно было без труда получить любую необходимую информацию.

Покажем, как будут выглядеть эти новые схемы отношений со своими ключами.

Вариант 2 схемы базы данных.

> Студенты (***№ зачетной книжки***, Фамилия, Имя, Отчество),
>
> Primary key (№ зачетной книжки).
>
> Сессия (***№ зачетной книжки***, ***Предмет***, Оценка),
>
> Primary key (№ зачетной книжки, Предмет),
>
> Foreign key (№ зачетной книжки) references Студенты (№ номер зачетной книжки).

Что мы имеем теперь? В отношении «Студенты» первичный ключ «№ зачетной книжки» функционально определяет остальные три атрибута: «Фамилия», «Имя» и «Отчество». А в

отношении «Сессия» составной первичный ключ «№ зачетной книжки, Предмет» также однозначно, т. е. буквально функционально определяет последний атрибут этой схемы отношения – «Оценка». И связь между этими двумя отношениями налажена: она осуществляется посредством внешнего ключа отношения «Сессия» «№ зачетной книжки», который ссылается на одноименный атрибут отношения «Студенты» и при соответствующем запросе представляет всю необходимую информацию.

Покажем теперь, как будут выглядеть отношения, представленные таблицами, отвечающие второму варианту задания соответствующих схем баз данных.

Студенты			
№ зачетной книжки	Фамилия	Имя	Отчество
100	Тюряпина	Елизавета	Николаевна
…	…	…	…
1100	Михайлова	Анастасия	Александровна

Сессия		
№ зачетной книжки	Предмет	Оценка
100	Информатика	Удовлетворительно
…	…	…
1100	Базы данных	Отлично
…	…	…
100	Литература	Отлично

Таким образом, мы видим, что целью нормализации в аспекте ограничений, накладываемых функциональными зависимостями, является необходимость навязать любой базе данных требуемые функциональные зависимости при помощи объявлений различного вида первичных, кандидатных и внешних ключей базовых отношений.

2. Первая нормальная форма (1NF)

На ранних стадиях проектирования баз данных и разработки схем их управления использовались простые и однозначные атрибуты как наиболее продуктивные и рациональные единицы кода. Тогда применяли наряду с простыми и составные атрибуты, а также наряду с однозначными и многозначные атрибуты. Поясним значения каждого из этих понятий.

Составные атрибуты, в отличие от простых, – это атрибуты, составленные из нескольких простых атрибутов.

Многозначные атрибуты, в отличие от однозначных, – это атрибуты, представляющие множество значений.

Приведем примеры простых, составных, однозначных и многозначных атрибутов.

Рассмотрим следующую таблицу, представляющую отношение:

…	Адрес	Телефон
…	…	…

Здесь атрибут «Телефон» – простой, однозначный, а атрибут «Адрес» – простой, но многозначный.

Теперь рассмотрим другую таблицу, с другими атрибутами:

…	Адреса	Телефоны
…	…	…

В этом отношении, представленном таблицей, атрибут «Телефоны» – простой, но многозначный, а атрибут «Адреса» – и составной, и многозначный.

Вообще возможны различные комбинации простых или составных атрибутов. В разных случаях таблицы, представляющие отношения, могут выглядеть следующим общим образом:

Атрибут	Простой	Составной
Однозначный	Телефон	Адрес
Многозначный	Телефоны	Адреса

При нормализации схем базовых отношений программистами может быть использована одна из четырех наиболее распространенных видов нормальных форм: первая нормальная форма (1NF), вторая нормальная форма (2NF), третья нормальная форма (3NF) или нормальная форма Бойса – Кодда (NFBC). Поясним: сокращение NF – это аббревиатура от англоязычного словосочетания Normal Form. Формально, кроме вышеназванных, существуют и другие виды нормальных форм, но вышеназванные – одни из самых востребованных.

В настоящее время разработчики баз данных стараются избегать составных и многозначных атрибутов, чтобы не усложнять написание кода, не перегружать его структуру и не запутывать пользователей. Из этих соображений логически и вытекает определение первой нормальной формы.

Определение. Любое базовое отношение находится в **первой нормальной форме** тогда и только тогда, когда схема этого отношения содержит только простые и только однозначные атрибуты, причем обязательно с одной и той же семантикой.

Для наглядного объяснения различий нормализованных и ненормализованных отношений рассмотрим пример.

Пусть, имеется ненормализованное отношение, со следующей схемой.

Итак, **вариант 1** схемы отношения с заданным на ней простым первичным ключом:

> Сотрудники (***№ табельный***, Фамилия Имя Отчество, Код должности, Телефоны, Дата приема или увольнения);
>
> Primary key (№ табельный);

Перечислим, какие в этой схеме отношения имеются ошибки, т. е. назовем те признаки, которые и делают собственно эту схему ненормализованной:

1) атрибут «Фамилия Имя Отчество» является составным, т. е. составленным из разнородных элементов;

2) атрибут «Телефоны» является многозначным, т. е. его значением является множество значений;

3) атрибут «Дата приема или увольнения» не имеет однозначной семантики, т. е. в последнем случае не понятно, какая именно дата внесена.

Если, например, ввести дополнительный атрибут, чтобы поточнее определить смысл даты, то для этого атрибута значение будет семантически понятно, но тем не менее остается возможность хранения только какой-то одной из указанных дат для каждого сотрудника.

Что же необходимо сделать для приведения этого отношения к нормальной форме?

Во-первых, необходимо провести разбиение составных атрибутов на простые, для того, чтобы исключить эти самые составные атрибуты, а также атрибуты с составной семантикой.

А во-вторых, необходимо провести декомпозицию этого отношения, т. е. нужно разбить его на несколько новых самостоятельных отношений, с тем чтобы исключить многозначные атрибуты.

Таким образом, с учетом всего вышесказанного после приведения отношения «Сотрудники» к первой нормальной форме или 1NF путем его декомпозиции мы получим систему следующих отношений с заданными на них первичными и внешними ключами.

Итак, **вариант 2** отношения:

> Сотрудники (***№ табельный***, Фамилия, Имя, Отчество, Код должности, Дата приема, Дата увольнения);
>
> Primary key (№ табельный);
>
> Телефоны (***№ табельный***, ***Телефон***);
>
> Primary key (№ табельный, Телефон);
>
> Foreign key (№ табельный) references Сотрудники (№ табельный);

Итак, что мы видим? Составного атрибута «Фамилия Имя Отчество» больше в нашем отношении нет, вместо него присутствуют три простых атрибута «Фамилия», «Имя» и «Отчество», поэтому эта причина «ненормальности» отношения исключилась.

Кроме того, вместо атрибута с неясной семантикой «Дата приема или увольнения» у нас появилось два атрибута «Дата приема» и «Дата увольнения», каждый из которых имеет однозначную семантику. Следовательно, вторая причина того, что наше отношение «Сотрудники» не находится в нормальной форме, также благополучно устранена.

И, наконец, последняя причина того, что отношение «Сотрудники» не было приведено к нормальной форме, – это наличие многозначного атрибута «Телефоны». Чтобы избавиться от этого атрибута, и необходимо было провести декомпозицию всего отношения. Из исходного отношения «Сотрудники» в результате этой декомпозиции был исключен атрибут «Телефоны» вообще, но зато образовалось второе отношение – «Телефоны», в котором присутствуют два атрибута: «№ табельный» сотрудника и «Телефон», т. е. все атрибуты – опять-таки простые, условие принадлежности к первой нормальной форме выполняется. Эти атрибуты «№ табельный» и «Телефон» образуют составной первичный ключ отношения «Телефоны», а атрибут «№ табельный», в свою очередь, является внешним ключом, ссылающимся на одноименный атрибут отношения «Сотрудники», т. е. в отношении «Телефоны» атрибут первичного ключа «№ табельный» является одновременно внешним ключом, ссылающимся на первичный ключ отношения «Сотрудники». Таким образом, обеспечивается связь между этими двумя отношениями. Посредством этой связи можно по номеру табельному любого сотрудника без особого труда и затрат времени вывести весь список его телефонов, не прибегая к использованию составных атрибутов.

Заметим, что в случае наличия в отношении системы ограничений функциональных зависимостей после всех вышеприведенных преобразований нормализация не была бы завершена. Однако в данном конкретном примере нет ограничений функциональных зависимостей, поэтому дальнейшая нормализация этого отношения не требуется.

3. Вторая нормальная форма (2NF)

Более сильные требования накладывает на отношения вторая нормальная форма, или 2NF.

Это происходит потому, что определение второй нормальной формы отношений предполагает, в отличие от первой нормальной формы, наличие системы ограничений функциональных зависимостей.

Определение. Базовое отношение находится во **второй нормальной форме** относительного заданного множества функциональных зависимостей тогда и только тогда, когда оно находится в первой нормальной форме и, кроме того, каждый неключевой атрибут полностью функционально зависит от каждого ключа.

В этом определении **неключевой атрибут** – это любой атрибут отношения, не содержащийся в каком-либо первичном или кандидатном ключе отношения.

Полная функциональная зависимость от ключа предполагает отсутствие функциональной зависимости от какой-либо части этого ключа.

Таким образом, теперь при нормализации отношения мы должны следить и за выполнением условий пребывания отношения в первой нормальной форме, т. е. следить, чтобы его атрибуты были простыми и однозначными, а также за выполнением второго условия, касающегося ограничений функциональных зависимостей.

Ясно, что отношения с простыми ключами (первичными и кандидатными) заведомо находятся во второй нормальной форме. Ведь в таком случае, зависимость от части ключа просто не представляется возможной, потому что никаких отдельных частей ключ банально не имеет.

Теперь, как и при прохождении предыдущей темы, рассмотрим пример ненормализованной схемы отношения и сам процесс нормализации.

Итак, **вариант 1** схемы отношения:

> Аудитории (***№ корпуса***, ***№ аудитории***, Площадь кв. м, № табельный коменданта корпуса);
>
> Primary key (№ корпуса, № аудитории);
>
> Кроме того, определена следующая система функциональной зависимости:
>
> {№ корпуса} > {№ табельный коменданта корпуса};

Что мы видим? Все условия пребывания этого отношения «Аудитории» в первой нормальной форме выполнены, ведь все до единого атрибуты этого отношения однозначны и просты. Но то условие, что каждый неключевой элемент должен полностью функционально зависеть от ключа, не выполняется. Почему? Да потому, что атрибут «№ табельный коменданта корпуса» функционально зависит не от составного ключа «№ корпуса, № аудитории», а от части этого ключа, т. е. от атрибута «№ корпуса». Действительно, ведь именно номер корпуса полностью определяет, какой именно комендант к нему приписан, а, в свою очередь, ни от каких номеров аудиторий табельный номер коменданта корпуса зависеть никак не может.

Таким образом, основной задачей нашей нормализации становится задача добиться того, чтобы ключи распределялись таким образом, чтобы, в частности, атрибут «№ табельный коменданта корпуса» полностью функционально зависел от всего ключа, а не от его какой-то части.

Для того, чтобы этого добиться, придется снова, как и в предыдущем параграфе, применить декомпозицию отношения. Итак, следующая система отношений, представляющая собой **вариант 2** отношения «Аудитории», как раз и получилась из исходного отношения путем его декомпозиции на несколько новых самостоятельных отношений:

Корпуса (***№ корпуса***, № табельный коменданта корпуса);

Primary key (№ корпуса);

Аудитории (***№ корпуса***, ***№ аудитории***, Площадь кв. м);

Primary key (№ корпуса, № аудитории);

Foreign key (№ корпуса) references Корпуса (№ корпуса);

Что мы видим теперь? В отношении «Корпуса» неключевой атрибут «№ табельный коменданта корпуса» полностью функционально зависит от первичного ключа «№ корпуса». Здесь условие нахождения отношения во второй нормальной форме полностью выполнились.

Теперь перейдем к рассмотрению второго отношения – «Аудитории». В отношении «Аудитории» атрибут первичного ключа «№ корпуса» является одновременно внешним ключом, ссылающемся на первичный ключ отношения «Корпуса». В этом отношении неключевой атрибут «Площадь кв. м» полностью зависит от всего составного первичного ключа «№ корпуса, № аудитории» и не зависит, даже не может зависеть ни от какой из его частей.

Таким образом, путем декомпозиции исходного отношения, мы пришли к тому, что все условия из определения второй нормальной формы полностью выполнились.

В данном примере все требования функциональной зависимости навязаны объявлением первичных ключей (кандидатных ключей здесь нет) и внешних ключей. Поэтому дальнейшая нормализация не требуется.

4. Третья нормальная форма (3NF)

Следующей нормальной формой, которую мы подвергнем рассмотрению, является третья нормальная форма (или 3NF). В отличие от первой нормальной формы, так же как и вторая нормальная форма, третья – подразумевает задание вместе с отношением системы функциональных зависимостей. Сформулируем, какими свойствами должно обладать отношение, чтобы оно было приведенным к третьей нормальной форме.

Определение. Базовое отношение находится в **третьей нормальной форме** относительно заданного множества функциональных зависимостей тогда и только тогда, когда оно находится во второй нормальной форме и каждый неключевой атрибут полностью функционально зависит только от ключей.

Таким образом, требования, предъявляемые третьей нормальной формой, сильнее требований, накладываемых первой и второй нормальной формой, даже вместе взятых. Фактически в третьей нормальной форме каждый неключевой атрибут зависит от ключа, причем от всего ключа целиком и ни от чего другого, кроме как от ключа.

Проиллюстрируем процесс приведения ненормализованного отношения к третьей нормальной форме. Для этого рассмотрим пример: отношение, находящееся не в третьей нормальной форме.

Итак, **вариант 1** схемы отношения «Сотрудники»:

Сотрудники (***№ табельный***, Фамилия, Имя, Отчество, Код должности, Оклад);

Primary key (№ табельный);

Кроме того, над данным отношением «Сотрудники» задана следующая система функциональных зависимостей:

{Код должности} > {Оклад};

Действительно, как правило, от должности, а следовательно, от ее кода в соответствующей базе данных напрямую зависит размер оклада, т. е. размер заработной платы.

Именно поэтому это отношение «Сотрудники» и не находится в третьей нормальной форме, ведь получается, что неключевой атрибут «Оклад» полностью функционально зависит от атрибута «Код должности», хотя этот атрибут и не является ключевым.

Любопытно, что к третьей нормальной форме любое отношение приводится точно таким же методом, как и к двум формам до этой, а именно, путем декомпозиции.

Проведя декомпозицию отношения «Сотрудники», получим следующую систему новых самостоятельных отношений:

Итак, **вариант 2** схемы отношения «Сотрудники»:

Должности (***Код должности***, Оклад);

Primary key (Код должности);

Сотрудники (***№ табельный***, Фамилия, Имя, Отчество, Код должности);

Primary key (Код должности);

Foreign key (Код должности) references Должности (Код должности);

Теперь, как мы видим, в отношении «Должности» неключевой атрибут «Оклад» полностью функционально зависит от простого первичного ключа «Код должности» и только от этого ключа.

Заметим, что в отношении «Сотрудники» все четыре неключевых атрибута «Фамилия», «Имя», «Отчество» и «Код должности» полностью функционально зависят от простого первичного ключа «№ табельный». В этом отношении атрибут «Код должности» – внешний ключ, ссылающийся на первичный ключ отношения «Должности».

В данном примере все требования навязаны объявлением простых первичных и внешних ключей, поэтому дальнейшая нормализация не требуется.

Интересно и полезно знать, что на практике обычно ограничиваются приведением баз данных к третьей нормальной форме. При этом, возможно, не навязанными остаются некоторые функциональные зависимости ключевых атрибуты от других атрибутов этого же отношения.

Поддержка таких нестандартных функциональных зависимостей реализуется при помощи уже упоминаемых ранее триггеров (т. е. процедурно, путем написания соответствующего программного кода). Причем триггеры должны оперировать кортежами этого отношения.

5. Нормальная форма Бойса – Кодда (NFBC)

Нормальная форма Бойса – Кодда следует по «сложности» сразу после третьей нормальной формы. Поэтому нормальную форму Бойса – Кодда еще иногда называют просто **усиленной третьей нормальной формой** (или усиленной 3 NF). Почему же она именно усиленная? Сформулируем определение нормальной формы Бойса – Кодда:

Определение. Базовое отношение находится в **нормальной форме Бойса** – **Кодда** тогда и только тогда, когда она находится в третьей нормальной форме, и при этом не только любой неключевой атрибут полностью функционально зависит от любого ключа, но и любой ключевой атрибут должен полностью функционально зависеть от любого ключа.

Таким образом, требование о фактической зависимости неключевых атрибутов от всего ключа целиком и ни от чего другого, кроме как от ключа, распространяется и на ключевые атрибуты.

В отношении, находящемся в нормальной форме Бойса – Кодда, все функциональные зависимости в пределах отношения навязаны объявлением ключей. Однако при приведении отношений баз данных к форме Бойса – Кодда, возможны ситуации, при которых не навязанными функциональными зависимостями оказываются зависимости между атрибутами различных отношений. Поддержка таких функциональных зависимостей при помощи триггеров, оперирующих кортежами различных отношений, сложнее, чем в случае третьей нормальной формы, когда триггеры оперируют кортежами единственного отношения.

Кроме всего прочего, практика проектирования систем управления базами данных показала, что не всегда удается привести базовое отношение к нормальной форме Бойса – Кодда.

Причиной отмеченных аномалий является то, что в требованиях второй нормальной формы и третьей нормальной формы не требовалась минимальная функциональная зависимость от первичного ключа атрибутов, являющихся компонентами других возможных ключей. Эту проблему и решает нормальная форма, которую исторически принято называть нормальной формой Бойса – Кодда и которая является уточнением третьей нормальной формы в случае наличия нескольких перекрывающихся возможных ключей.

Вообще нормализация схемы базы данных способствует более эффективному выполнению системой управления базами данных операций обновления базы данных, поскольку сокращается число проверок и вспомогательных действий, поддерживающих целостность базы данных. При проектировании реляционной базы данных почти всегда добиваются второй нормальной формы всех входящих в базу данных отношений. В часто обновляемых базах данных обычно стараются обеспечить третью нормальную форму отношений. На нормальную форму Бойса – Кодда внимание обращают гораздо реже, поскольку на практике ситуации, в которых у отношения имеется несколько составных перекрывающихся возможных ключей, встречаются нечасто.

Все вышеназванное делает нормальную форму Бойса – Кодда не слишком удобной в использовании при разработке программного кода, поэтому, как уже было сказано ранее, на практике разработчики обычно ограничиваются приведением своих баз данных к третьей нормальной форме. Однако здесь тоже есть своя довольно любопытная особенность. Дело в том, что ситуации, когда отношение находится в третьей нормальной форме, но не находится в нормальной форме Бойса – Кодда крайне редки на практике, т. е. после приведения к третьей нормальной форме обычно все функциональные зависимости оказываются навязанными объявлениями первичных, кандидатных и внешних ключей, так что необходимость в триггерах для поддержки функциональных зависимостей отпадает.

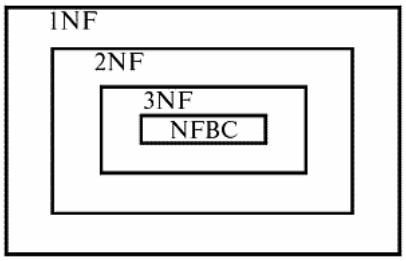
Однако необходимость в триггерах остается для поддержки ограничения целостности, не связанных функциональными зависимостями.

6. Вложенность нормальных форм

Что означает вложенность нормальных форм друг в друга?

Вложенность нормальных форм – это отношение понятий ослабленной и усиленной формы по отношению друг к другу.

Вложенность нормальных форм полностью следует из их соответствующих определений. Представим диаграмму, иллюстрирующую отношение вложенности известных нам нормальных форм:



Поясним понятия ослабленной и усиленной нормальной формы по отношению друг к другу на конкретных примерах.

Первая нормальная форма является ослабленной по отношению ко второй нормальной форме (да и по отношению ко всем остальным нормальным формам тоже). Действительно, вспоминая определения всех пройденных нами нормальных форм, можно заметить, что требования каждой нормальной формы включали в себя требование принадлежности именно к первой нормальной форме (ведь она входила в каждое последующее определение).

Вторая нормальная форма является усиленной по отношению к первой нормальной форме, но ослабленной по отношению к третьей нормальной форме и нормальной форме Бойса – Кодда. На самом деле принадлежность второй нормальной форме включается в определение третьей, а сама вторая форма, в свою очередь, включает в себя первую нормальную форму.

Нормальная форма Бойса – Кодда является усиленной не только по отношению к третьей нормальной форме, но также и по отношению ко всем остальным, предшествующим ей.

А третья нормальная форма, в свою очередь, является ослабленной только по отношению к нормальной форме Бойса – Кодда.

Оглавление

Главная | В избранное | Наш E-MAIL | Добавить материал | Нашёл ошибку | Другие сайты | Наверх