# Первая нормальная форма

Первая нормальная форма предполагает, что таблица не должна содержать повторяющихся столбцов или таких столбцов, которые содержат наборы значений. Ненормализованная таблица в этом случае может содержать одну или несколько повторяющихся групп данных. Повторяющаяся группа - это группа из одного или нескольких атрибутов таблицы, в которой возможно наличие нескольких значений для ключевого атрибута таблицы.

Итогом применения первой формы должно стать наличие для одного атрибута сущности только одного столбца в таблице, который при этом должен содержать скалярное значение.

Есть два похода к переходу к ненормализованной таблицы к первой нормальной форме. Первый способ называется выравниванием или flattaning. Он предполагает декомпозицию строки с повторяющимися группами данных, при котором для каждой повторяющейся группы создается своя строка. Полученная в результате таблица будет содержать атомарные значения для каждого из атрибутов. Хотя в то же время этот подход увеличит избыточность данных.

Второй подход предполагает, что один атрибут или группа атрибутов назначаются ключом ненормализованной таблицы, а затем повторяющиеся группы удаляются из таблицы и помещаются в отдельную таблицу вместе с копиями ключа из исходной таблицы.

Рассмотрим применение нормализации на примере. Пусть у нас есть система, которая описывается следующей информацией:

```
Том посещает курс по математике, который преподает профессор Смит. Дата записи 11/06/2017.
Сэм посещает курс по алгоритмам, которые преподает ассистент Адамс. Дата записи 12/06/2017.
Боб посещает курс по математике, который преподает профессор Смит. Дата записи 13/06/2017.
Том посещает курс по языку JavaScript, который преподает ассистент Адамс. Дата записи 14/06/2017.
Сэм имеет два электронных адреса: sam@gmail.com и sam@hotmail.com.
В университете может быть только один курс с определенным именем. Один преподаватель может
преподавать несколько курсов.
```

Вначале определим ненормализованную таблицу StudentCourses, которая содержит всю эту информацию:

| StudentId | Name | Emails | Course1 | Date1 | TeacherId1 | Teacher1 | Position1 | Course2 | Date2 | TeacherId2 | Teacher2 | Position2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Том | | Математика | 11/06/2017 | 1 | Смит | Профессор | JavaScript | 14/06/2017 | 2 | Адамс | Ассистент |
| 2 | Сэм | sam@gmail.com<br>sam@hotmail.com | Алгоритмы | 12/06/2017 | 2 | Адамс | Ассистент | | | | | |
| 3 | Боб | | Математика | 13/06/2017 | 1 | Смит | Профессор | | | | | |

Для каждого студента определен уникальный идентификатор StudentId, а также атрибут Name (имя), Emails (все электронные адреса), Course1 /Course2(курс), Date1/Date2 (дата поступления), Teacher1/Teacher2 (преподаватель), Position1/Position2 (должность преподавателя). Также чтобы различать преподавателей (так как теоретически могут быть преподаватели с одной и той же фамилией), добавлен атрибут TeacherId1/TeacherId2. Для курсов такой идентификатор не требуется, так как в нашем случае название курса уникально.

Поскольку Том записан сразу на два курса, то несколько атрибутов пришлось дублировать. Но что будет, когда Том в стремлении получить никому не нужные сертификаты запишется еще на десяток курсов?

Эта таблица представляет прекрасный пример отклонения от первой нормальной формы. В первую очередь мы видим группу повторяющихся атрибутов, которые представляют данные по одному курсу: Course, Date, TeacherId, Teacher, Position. Эти атрибуты представляют повторяющуюся группу, которую можно условно назвать StudentCourse.

StudentCourse = (Course, Date, TeacherId, Teacher, Position)

Вторая проблема - атрибут Emails содержит набор электронных адресов. Фактически этот атрибут также образует повторяющуюся группу.

Для избавления от первой повторяющейся группы атрибутов применим первый подход: создадим для каждой повторяющейся группы отдельную строку.

| StudentId | Name | Emails | CourseId | Course | Date | TeacherId | Teacher | Position |
|---|---|---|---|---|---|---|---|---|
| 1 | Том | | 1 | Математика | 11/06/2017 | 1 | Смит | Профессор |
| 1 | Том | | 2 | JavaScript | 14/06/2017 | 2 | Адамс | Ассистент |
| 2 | Сэм | sam@gmail.com<br>sam@hotmail.com | 3 | Алгоритмы | 12/06/2017 | 2 | Адамс | Ассистент |
| 3 | Боб | | 1 | Математика | 13/06/2017 | 1 | Смит | Профессор |

В данном случае увеличилась избыточность данных, но тем не менее мы избавились от повторяющейся группы. Также следует отметить, что теперь атрибут StudentId не может использоваться в качестве первичного ключа. И в данном случае просматривается только один потенциальный ключ, который и будет использоваться в качестве первичного - это сразу два столбца StudentId и Course. Но название курса - не лучший ключ, если учитывать, что это название может редактироваться и изменяться. Поэтому для каждого курса добавлен еще один атрибут - CourseId - уникальной номер курса, который вместе с StudentId составляет первичный ключ. Хотя в принципе может было бы и оставить в качестве части первичного ключа имя курса с учетом, что оно уникально.

Для избавления от второй повторяющейся группы - атрибута Emails применим второй подход: вынесение этой группы с копией ключа в отдельную таблицу. Для этого определим таблицу Emails:

| Email | StudentId |
|---|---|
| sam@gmail.com | 2 |
| sam@hotmail.com | 2 |

Так как электронный адрес в принципе уникален, то его можно сделать первичным ключом.

Таким образом, таблицы Emails с таблицей StudentCourses будет связана связью один ко многим (один студент - много электронных адресов). И в этом случае таблица StudentCourses сократится следующим образом:

| StudentId | Name | CourseId | Course | Date | TeacherId | Teacher | Position |
|---|---|---|---|---|---|---|---|
| 1 | Том | 1 | Математика | 11/06/2017 | 1 | Смит | Профессор |
| 1 | Том | 2 | JavaScript | 14/06/2017 | 2 | Адамс | Ассистент |
| 2 | Сэм | 3 | Алгоритмы | 12/06/2017 | 2 | Адамс | Ассистент |
| 3 | Боб | 1 | Математика | 13/06/2017 | 1 | Смит | Профессор |

Теперь у нас нет повторяющихся столбцов, но увеличилась избыточность данных, так как для студента Том определено уже две строки в таблице, и соответственно Id повторяется. Но тем не менее 1-я нормальная форма применена.

В принципе можно отметить, что если повторяющиеся группы содержат уникальные значения для каждой строки таблицы (как в случае с электронными адресами), то мы имеем дело с потенциальной связью один ко многим. Если же повторяющиеся группы содержат неуникальные значения, которые могут иметь разные строки таблицы (как в случае с атрибутами курсов), то это скрывается потенциальная связь многие ко многим.

# Вторая нормальная форма

Во второй нормальной форме каждый столбец в таблице, который не является ключом, **должен зависеть от ключа**.

Ключевой момент второй нормальной формы - полная функциональная зависимость. Она предполагает, что атрибут B полностью функционально зависим от атрибута A, если атрибут B функционально зависит от полного значения атрибута A, а не от какого-либо подмножества значений из атрибута A. То есть, если атрибут A составляют несколько значений, скажем, A1 и A2, то атрибут B полностью функционально зависит от A, если он зависит и от A1 и от A2 (A1, A2 → B).

Если атрибут B зависит только от какого-либо подмножества из атрибута A, например, только от A1, то имеет место частичная функциональная зависимость.

Эта форма применяется к тем таблицам, которые имеют составной первичный ключ, то есть где первичный ключ состоит из нескольких атрибутов. Если в таблице **несоставной** первичный ключ, то в этом случае считается, что все остальные атрибуты автоматически находятся в полной функциональной зависимости от первичного ключа.

Вторая нормальная форма применяется только к тем таблицам, которые находятся в первой нормальной форме. После применения второй формы все столбцы таблицы зависят от первичного ключа.

Возьмем сформированную в прошлой теме таблицу StudentCourses после применения первой нормальной формы:

| StudentId | Name | CourseId | Course | Date | TeacherId | Teacher | Position |
|---|---|---|---|---|---|---|---|
| 1 | Том | 1 | Математика | 11/06/2017 | 1 | Смит | Профессор |
| 1 | Том | 2 | JavaScript | 14/06/2017 | 2 | Адамс | Ассистент |
| 2 | Сэм | 3 | Алгоритмы | 12/06/2017 | 2 | Адамс | Ассистент |
| 3 | Боб | 1 | Математика | 13/06/2017 | 1 | Смит | Профессор |

На данный момент эта таблица имеет составной первичный ключ StudentId+CourseId. Какие функциональные зависимости от ключевых атрибутов здесь можно выделить:

StudentId, CourseId → Date

StudentId → Name

CourseId → Course, TeacherId, Teacher, Position

От обоих частей составного ключа StudentId+CourseId зависит только арибут Date - дата, в которую студент с идентификатором StudentId поступил на курс с идентифкатором CourseId.

Атрибут Name зависит только от части составного ключа - от атрибута StudentId, так как зная идентификатор студента, можно сказать, какое у него имя. В данном случае имеет факт частичной зависимости.

Атрибуты Course, TeacherId, Teacher, Position зависит от другой части ключа - от атрибута CourseId. Зная значение CourseId, можно сказать, как называется курс, какой у курса преподаватель, какую должность он занимает. Опять же здесь частичная зависимость.

Наличие частичных зависимостей говорит о том, что таблица не находится во второй нормальной форме. И для перехода к этой форме необходимо переместить атрибуты, которые не входят в первичный ключ, в новую таблицу вместе с копией части первичного ключа, от которой они функционально зависят.

В нашем случае из одной таблицы получатся три. Таблица Students:

| StudentId | Name |
|---|---|
| 1 | Том |
| 2 | Сэм |
| 3 | Боб |

Таблица Courses:

| CourseId | Course | TeacherId | Teacher | Position |
|---|---|---|---|---|
| 1 | Математика | 1 | Смит | Профессор |
| 2 | JavaScript | 2 | Адамс | Ассистент |
| 3 | Алгоритмы | 2 | Адамс | Ассистент |

И таблица StudentCourses:

| StudentId | CourseId | Date |
|---|---|---|
| 1 | 1 | 11/06/2017 |
| 1 | 2 | 14/06/2017 |
| 2 | 3 | 12/06/2017 |
| 3 | 1 | 13/06/2017 |

Итогом стало образование связи многие ко многим (много студентов - много курсов) между таблицами Students и Courses через таблицу StudentCourses .

Таким образом, база данных перешла во вторую нормальную форму.

# Третья нормальная форма

Третья нормальная форма предполагает, что каждый столбец, не являющийся ключом, должен зависеть **только** от столбца, который является ключом, то есть должна отсутствовать **транзитивная функциональная зависимость** (transitive functional dependency)

Транзитивная функциональная зависимость выражается следующим образом: A → B и B → C. То есть атрибут C транзитивно зависит от атрибута A, если атрибут C зависит от атрибута B, а атрибут B зависит от атрибута A (при условии, что атрибут A функционально не зависит ни от атрибута B, ни от атрибута C).

Если столбец зависит не только от первичного ключа, то данный столбец находится не в той таблице, в которой он должен находиться, либо же является производным от других столбцов.

Для нормализации из исходной таблицы те атрибуты, которые находятся в транзитивной зависимости от ключа, выносятся в отдельную таблицу с копией того атрибута, от которого они непосредственно зависят.

При применении третьей нормальной формы таблица должна находиться во второй нормальной форме. 3NF позволяет значительно снизить избыточность данных.

Для примера возьмем сформированную в прошлой теме таблицу Courses, которая содержит информацию о курсах и которая находится во второй нормальной форме:

| CourseId | Course | TeacherId | Teacher | Position |
|---|---|---|---|---|
| 1 | Математика | 1 | Смит | Профессор |
| 2 | JavaScript | 2 | Адамс | Ассистент |
| 3 | Алгоритмы | 2 | Адамс | Ассистент |

Какие функциональные зависимости здесь можно выделить:

CourseId → Course, TeacherId, Teacher, Position

Course → CourseId, TeacherId, Teacher, Position

TeacherId → Teacher, Position

Вторая зависимость фактически аналогична первой и говорит о том, что атрибут Course является потенциальным ключом.

Третья зависимость говорит о том, что, зная идентификатор преподавателя, мы можем узнать его фамилию и должность. То есть через атрибут TeacherId атрибуты Teacher и Position зависят от CourseId (CourseId→ TeacherId и TeacherId → Teacher, Position). И в данном случае мы можем говорить о транзитивной зависимости Teacher, Position от CourseId.

Для нормализации необходимо вынести в отдельную таблицу атрибуты TeacherId, Teacher, Position. Для этого пусть будет отдельная таблица Teachers:

| TeacherId | Teacher | Position |
|---|---|---|
| 1 | Смит | Профессор |
| 2 | Адамс | Ассистент |

А таблица Courses сократится следующим образом:

| CourseId | Course | TeacherId |
|---|---|---|
| 1 | Математика | 1 |
| 2 | JavaScript | 2 |
| 3 | Алгоритмы | 2 |