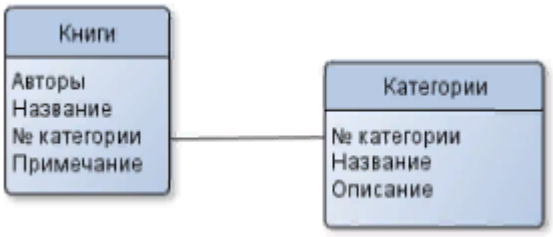
Связи между таблицами

Отношения один-ко-многим

В первых уроках курса, описывая структуру простой базы данных, мы говорили о том, что в ней используется связь между таблицами вида «один-ко-многим». Давайте вспомним эту структуру:

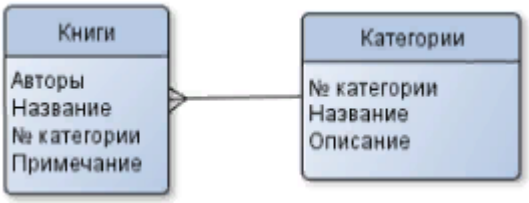


Поле **№ категории** в таблице **Категории** содержит уникальные значения, то есть не может быть двух категорий с одним и тем же номером.

На одноимённое поле **№ категории** в таблице **Книги** таких ограничений не накладывается: сколько угодно записей в этой таблице может иметь одно и то же значение этого поля.

В этом примере связь можно словами описать так: каждая книга может принадлежать только к одной категории, при этом к одной и той же категории может принадлежать множество книг.

На схемах баз данных используют обычно специальные обозначения для видов связей. Например, распространён такой вариант:

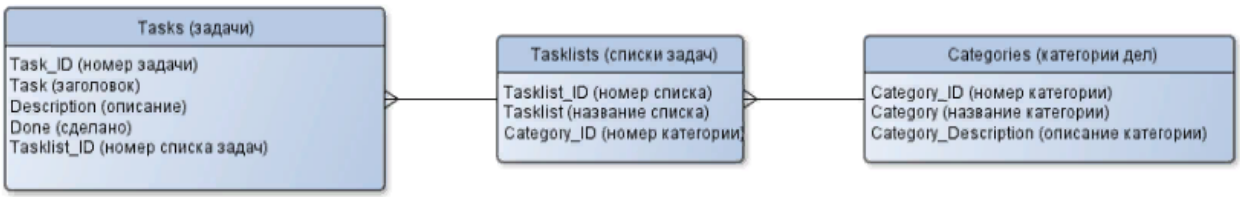


Это и есть отношение один-ко-многим. Слово «отношение» здесь используется в том же значении, что и слово «связь». В английском языке для этого применяется слово «relation». Благодаря этому слову прижилось и общее название баз данных, для работы с которыми предназначен язык SQL: *реляционные* базы данных.

В наших учебных базах данных тоже используются отношения этого вида. Например, в базе данных со списками дел таких отношений два:

1. Каждое дело относится только к одному списку (то есть существует отношение один-ко-многим между таблицами Tasklists и Tasks)
2. Каждый список относится только к одной категории (отношение один-ко-многим между таблицами Categories и Tasklists)

Схему базы данных можно представить так:



Как мы видим, для реализации такого отношения в таблицах достаточно выполнить два условия:

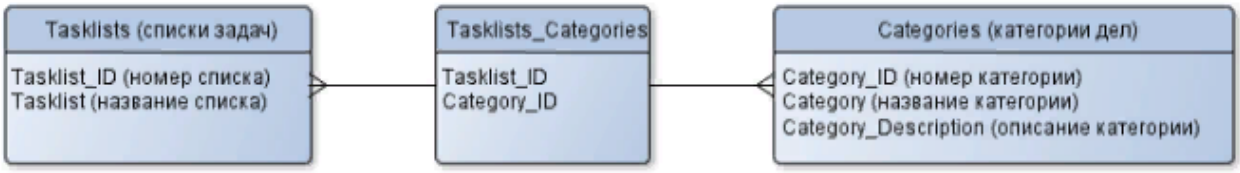
1. В таблице, находящейся со стороны «один», должно быть поле с уникальными значениями. Обычно это поле, которое используется в качестве идентификатора записи.
2. В таблице, находящейся со стороны «многие», должно быть поле, ссылающееся на этот идентификатор записи первой таблицы.

Отношения многие-ко-многим

Предположим, мы решили, что один список задач может относиться к нескольким категориям. Например, некоторые задачи мы хотим включить и в категорию «Дом», и в категорию «Семья». Существующая схема базы данных не позволит нам этого сделать, потому что в каждой записи таблицы **Tasklists** поле **Category_ID** может содержать только одно значение.

Добавить в таблицу Tasklists ещё одну запись для того же списка задач, но ссылающуюся на другую категорию, наша схема тоже не позволит: значение поля **Tasklist_ID** должно быть уникальным.

Правильным способом решения этой задачи является добавление ещё одной таблицы, единственное назначение которой состоит в связывании таблиц **Tasklists** и **Categories**. Схема этой связи может выглядеть так:



Из таблицы **Tasklists** убрано поле **Category_ID**, которое привязывало каждый список задач к одной категории. Вместо этого добавлена таблица **Tasklists_Categories**. В этой таблице два поля, которые ссылаются, с одной стороны, на записи таблицы **Tasklists**, и с другой — на записи таблицу **Categories**. В результате между таблицами Tasklists и Categories появляется отношение вида многие-ко-многим.

Теперь, чтобы указать, что список задач относится к определённой категории, нам нужно добавить запись в эту таблицу. Чтобы отнести один список к двум категориям, нужно добавить в таблицу две соответствующих записи.

Конечно, при использовании базы данных с новой схемой усложнятся запросы SELECT, которые используют связь между этими таблицами. Например, в старой схеме базы данных мы могли использовать такой запрос, чтобы выяснить, к какой категории относится какой список задач:

```
select Tasklist, Category
from Tasklists, Categories
where Tasklists.Category_ID=Categories.Category_ID
```

В новой схеме в запрос нужно добавить новую таблицу, реализующую отношение многие-ко-многим, а в условиях отбора указать, как эта таблица связывается с каждой из таблиц **Tasklists** и **Categories**:

```
select Tasklist, Category
from Tasklists, Categories, Tasklists_Categories
where Tasklists.Tasklist_ID=Tasklists_Categories.Tasklist_ID
and Categories.Category_ID=Tasklists_Categories.Category_ID
```

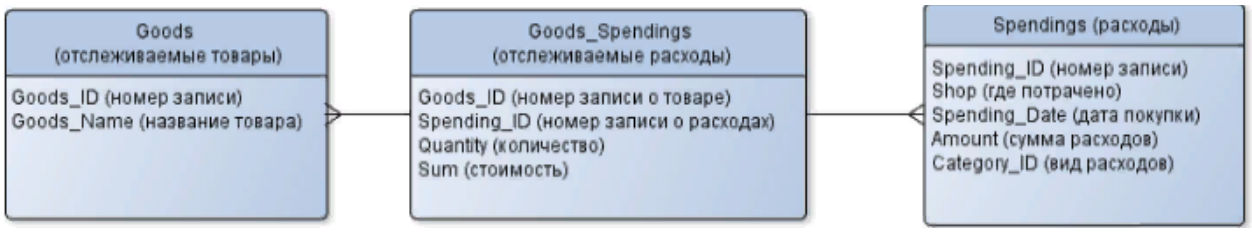
```
select Tasklist, Category
from Tasklists, Categories, Tasklists_Categories
where Tasklists.Tasklist_ID=Tasklists_Categories.Tasklist_ID
and Categories.Category_ID=Tasklists_Categories.Category_ID
```

	Tasklist	Category
1	Ремонт в ванной	Дом
2	Подготовка сына к школе	Семья
3	Запуск сайта	Своё дело
4	Регистрация ИП	Своё дело
5	Ремонт в ванной	Семья

Вы можете скачать изменённую базу данных и поэкспериментировать с ней.

Атрибуты отношения

Возможно, вы уже самостоятельно обнаружили отношение вида многие-ко-многим в нашей второй учебной базе данных, предназначенной для учёта расходов? Посмотрите на таблицу Goods_Spendings:



Она содержит поля **Goods_ID** и **Spending_ID**, которые ссылаются на идентификаторы таблиц **Goods** и **Spendings**, и, таким образом, реализует отношение многие-ко-многим между этими таблицами. Это отношение можно описать словами таким образом: можно многократно покупать товары, траты на которые мы хотим отслеживать, при этом в одну покупку могут входить несколько отслеживаемых товаров.

Но, кроме этих полей, таблица также содержит поля **Quantity** и **Sum**, в которых хранится дополнительная информация: сколько товаров конкретного вида и на какую сумму было приобретено в рамках отдельной покупки. Таблица **Goods_Spendings**, таким образом, не является «чисто технической» реализацией отношения многие-ко-многим, а хранит ещё и полезную информацию о каждой отдельной связи между записями таблиц. Такие поля часто называют атрибутами отношения.

Отношения один-к-одному

Мы рассмотрели отношения видов один-ко-многим и многие-ко-многим. Возникает вопрос: а бывают ли отношения вида один-к-одному?

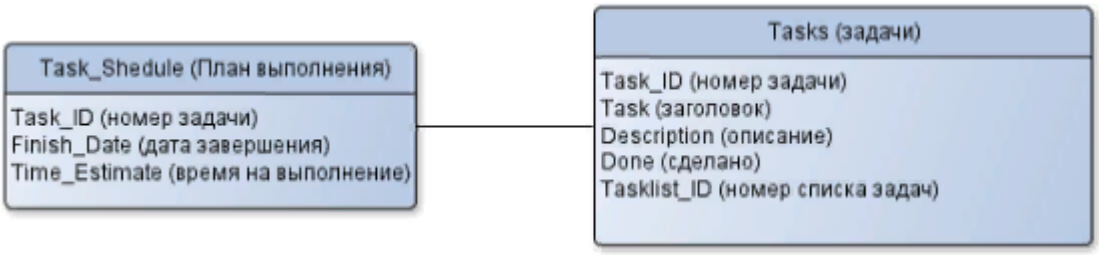
Да, бывают, но они не так интересны. Ведь если запись одной таблицы всегда однозначно соответствует записи другой таблицы, то почему бы просто не объединить эти две таблицы в одну? Тем не менее, такие отношения тоже используются.

Часто отношения вида один-к-одному появляются при разбиении одной большой таблицы на несколько частей. Обычно это делают для улучшения производительности или надёжности хранения данных.

Другой распространённый случай возникновения таких отношений между таблицами — это добавление новых функций в очередной версии программы, использующей базу данных. Особенно часто такие связи возникают, когда над программой работает множество программистов,

одновременно добавляющих разные функции. Тогда каждому из них может быть удобнее работать со своим набором полей в отдельной таблице, а потом их работу проще объединять в одну базу данных.

Например, мы могли бы добавить к нашей учебной базе данных списков задач новые поля: дату планируемого выполнения задачи и оценку времени, которое должно быть на неё потрачено. Можно было бы добавить эти поля прямо в таблицу Tasks, а можно создать отдельную таблицу:



Этот пример иллюстрирует, каким образом реализуются в таблицах связи вида один-к-одному. В каждой таблице есть поле Task_ID (номер задачи), по которому записи таблиц связываются между собой.

Такой же способ используется для реализации отношений многие-к-одному, и это неудивительно: ведь отношение один-к-одному фактически является его частным случаем. Но при этом на обе таблицы накладывается ограничение: поле Task_ID не должно содержать повторяющихся значений ни в одной, ни в другой таблице. В языке SQL есть средства, позволяющие поручить базе данных выполнение этого ограничения, и мы ещё познакомимся с ними в этом курсе.

Виды отношений — тест

Укажите, между какими таблицами учебной базы данных для учёта расходов существует отношение «один-ко-многим».

Выберите один или несколько ответов:

- [ ] a. Goods и Spendings
- [ ] b. Spendings и Categories
- [ ] c. Goods_Spendings и Goods

Укажите, между какими таблицами учебной базы данных для учёта расходов существует отношение «многие-ко-многим».

Выберите один или несколько ответов:

- [ ] a. Spendings и Categories
- [ ] b. Goods и Spendings
- [ ] c. Goods_Spendings и Goods

Какая таблица учебной базы данных для учёта расходов реализует отношение «многие-ко-многим»?

Выберите один ответ:

- ( ) a. Goods_Spendings
- ( ) b. Spendings
- ( ) c. Goods
- ( ) d. Categories