Связи между таблицами базы данных

### **1. Введение**

Связи — это довольна важная тема, которую следует понимать при проектировании баз данных. По своему личному опыту скажу, что осознав связи, мне намного легче далось понимание [нормализации](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0)базы данных.

### **1. Как организовываются связи?**

Связи создаются с помощью внешних ключей (foreign key).  
Внешний ключ — это атрибут или набор атрибутов, которые ссылаются на primary key или unique другой таблицы. Другими словами, это что-то вроде указателя на строку другой таблицы.

### **2. Виды связей**

Связи делятся на:

1. Многие ко многим.
2. Один ко многим.
   * с обязательной связью;
   * с необязательной связью;
3. Один к одному.
   * с обязательной связью;
   * с необязательной связью;

Рассмотрим подробно каждый из них.

### **3. Многие ко многим**

Представим, что нам нужно написать БД, которая будет хранить работником IT-компании. При этом существует некий стандартный набор должностей. При этом:

* Работник может иметь одну и более должностей. Например, некий работник может быть и админом, и программистом.
* Должность может «владеть» одним и более работников. Например, админами является определенный набор работников. Другими словами, к админам относятся некие работники.

Работников представляет таблица «Employee» (id, имя, возраст), должности представляет таблица «Position» (id и название должности). Как видно, обе эти таблицы связаны между собой по правилу многие ко многим: каждому работнику соответствует одна и больше должностей (многие должности), каждой должности соответствует один и больше работников (многие работники).

### **4. Как построить такие таблицы?**

Мы уже имеем две таблицы, описывающие работника и профессию. Теперь нам нужно установить между ними связь многие ко многим. Для реализации такой связи нам нужен некий посредник между таблицами «Employee» и «Position». В нашем случае это будет некая таблица «EmployeesPositions» (работники и должности). Эта таблица-посредник связывает между собой работника и должность следующим образом:

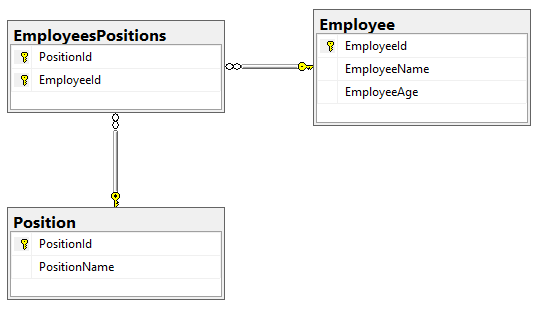
|  |  |
| --- | --- |
| **EmployeeId** | **PositionId** |
| 1 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 3 |

Слева указаны работники (их id), справа — должности (их id). Работники и должности на этой таблице указываются с помощью id’шников.  
  
На эту таблицу можно посмотреть с двух сторон:

1. Таким образом, мы говорим, что работник с id 1 находится на должность с id 1. При этом обратите внимание на то, что в этой таблице работник с id 1 имеет две должности: 1 и 2. Т.е., каждому работнику слева соответствует некая должность справа.
2. Мы также можем сказать, что должности с id 3 принадлежат пользователи с id 2 и 3. Т.е., каждой роли справа принадлежит некий работник слева.

### **4.1. Реализация**

Диаграмма



Код на T-SQL

create table dbo.Employee

(

EmployeeId int primary key,

EmployeeName nvarchar(128) not null,

EmployeeAge int not null

)

*-- Заполним таблицу Employee данными.*

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (1, N'John Smith', 22)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (2, N'Hilary White', 22)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (3, N'Emily Brown', 22)

create table dbo.Position

(

PositionId int primary key,

PositionName nvarchar(64) not null

)

*-- Заполним таблицу Position данными.*

insert into dbo.Position(PositionId, PositionName) values(1, N'IT-director')

insert into dbo.Position(PositionId, PositionName) values(2, N'Programmer')

insert into dbo.Position(PositionId, PositionName) values(3, N'Engineer')

*-- Заполним таблицу EmployeesPositions данными.*

create table dbo.EmployeesPositions

(

PositionId int foreign key references dbo.Position(PositionId),

EmployeeId int foreign key references dbo.Employee(EmployeeId),

primary key(PositionId, EmployeeId)

)

insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (1, 1)

insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (1, 2)

insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (2, 3)

insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (3, 3)

Объяснения

С помощью ограничения foreign key мы можем ссылаться на primary key или unique другой таблицы. В этом примере мы

* ссылаемся атрибутом PositionId таблицы EmployeesPositions на атрибут PositionId таблицы Position;
* атрибутом EmployeeId таблицы EmployeesPositions — на атрибут EmployeeId таблицы Employee;

### **4. Вывод**

Для реализации связи многие ко многим нам нужен некий посредник между двумя рассматриваемыми таблицами. Он должен хранить два внешних ключа, первый из которых ссылается на первую таблицу, а второй — на вторую.

### **5. Один ко многим**

Эта самая распространенная связь между базами данных. Мы рассматриваем ее после связи многие ко многим для сравнения.  
  
Предположим, нам нужно реализовать некую БД, которая ведет учет данных о пользователях. У пользователя есть: имя, фамилия, возраст, номера телефонов. При этом у каждого пользователя может быть от одного и больше номеров телефонов (многие номера телефонов).  
  
В этом случае мы наблюдаем следующее: пользователь может иметь многие номера телефонов, но нельзя сказать, что номеру телефона принадлежит определенный пользователь.  
  
Другими словами, телефон принадлежит только одному пользователю. А пользователю могут принадлежать 1 и более телефонов (многие).  
  
Как мы видим, это отношение один ко многим.

### **5.1. Как построить такие таблицы?**

Пользователей будет представлять некая таблица «Person» (id, имя, фамилия, возраст), номера телефонов будет представлять таблица «Phone». Она будет выглядеть так:

|  |  |  |
| --- | --- | --- |
| **PhoneId** | **PersonId** | **PhoneNumber** |
| 1 | 5 | 11 091-10 |
| 2 | 5 | 19 124-66 |
| 3 | 17 | 21 972-02 |

Данная таблица представляет три номера телефона. При этом номера телефона с id 1 и 2 принадлежат пользователю с id 5. А вот номер с id 3 принадлежит пользователю с id 17.  
Заметка. Если бы у таблицы «Phones» было бы больше атрибутов, то мы смело бы их добавляли в эту таблицу.

### **5.2. Почему мы не делаем тут таблицу-посредника?**

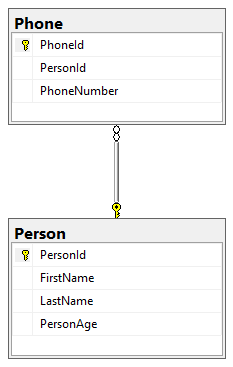
Таблица-посредник нужна только в том случае, если мы имеем связь многие-ко-многим. По той простой причине, что мы можем рассматривать ее с двух сторон. Как, например, таблицу EmployeesPositions ранее:

1. Каждому работнику принадлежат несколько должностей (многие).
2. Каждой должности принадлежит несколько работников (многие).

Но в нашем случае мы не можем сказать, что каждому телефону принадлежат несколько пользователей — номеру телефона может принадлежать только один пользователь.  
Теперь прочтите еще раз заметку в конце пункта 5.1. — она станет для вас более понятной.

#### **5.3. Реализация**

Диаграмма



Код на T-SQL

create table dbo.Person

(

PersonId int primary key,

FirstName nvarchar(64) not null,

LastName nvarchar(64) not null,

PersonAge int not null

)

insert into dbo.Person(PersonId, FirstName, LastName, PersonAge) values (5, N'John', N'Doe', 25)

insert into dbo.Person(PersonId, FirstName, LastName, PersonAge) values (17, N'Izabella', N'MacMillan', 19)

create table dbo.Phone

(

PhoneId int primary key,

PersonId int foreign key references dbo.Person(PersonId),

PhoneNumber varchar(64) not null

)

insert into dbo.Phone(PhoneId, PersonId, PhoneNumber) values (1, 5, '11 091-10')

insert into dbo.Phone(PhoneId, PersonId, PhoneNumber) values (2, 5, '19 124-66')

insert into dbo.Phone(PhoneId, PersonId, PhoneNumber) values (3, 17, '21 972-02')

Объяснения

Наша таблица Phone хранит всего один внешний ключ. Он ссылается на некого пользователя (на строку из таблицы Person). Таким образом, мы как бы говорим: «этот пользователь является владельцем данного телефона». Другими словами, телефон знает id своего владельца.

### **6. Один к одному**

Представим, что на работе вам дали задание написать БД для учета всех работников для HR. Начальник уверял, что компании нужно знать только об имени, возрасте и телефоне работника. Вы разработали такую БД и поместили в нее всю 1000 работников компании. И тут начальник говорит, что им зачем-то нужно знать о том, является ли работник инвалидом или нет. Наиболее простое, что приходит в голову — это добавить новый столбец типа bool в вашу таблицу. Но это слишком долго вписывать 1000 значений и ведь true вы будете вписывать намного реже, чем false (2% будут true, например).  
  
Более простым решением будет создать новую таблицу, назовем ее «DisabledEmployee». Она будет выглядеть так:

|  |  |
| --- | --- |
| **DisabledPersonId** | **EmployeeId** |
| 1 | 159 |
| 2 | 722 |
| 3 | 937 |

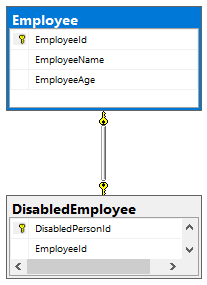
Но это еще не связь один к одному. Дело в том, что в такую таблицу работник может быть вписан более одного раза, соответственно, мы получили отношение один ко многим: работник может быть несколько раз инвалидом. Нужно сделать так, чтобы работник мог быть вписан в таблицу только один раз, соответственно, мог быть инвалидом только один раз. Для этого нам нужно указать, что столбец EmployeeId может хранить только уникальные значения. Нам нужно просто наложить на столбец EmloyeeId ограничение unique. Это ограничение сообщает, что атрибут может принимать только уникальные значения.  
  
Выполнив это мы получили связь один к одному.  
  
Заметка. Обратите внимание на то, что мы могли также наложить на атрибут EmloyeeId ограничение primary key. Оно отличается от ограничения unique лишь тем, что не может принимать значения null.

### **6.1. Вывод**

Можно сказать, что отношение один к одному — это разделение одной и той же таблицы на две.

### **6.2. Реализация**

Диаграмма



Код на T-SQL

create table dbo.Employee

(

EmployeeId int primary key,

EmployeeName nvarchar(128) not null,

EmployeeAge int not null

)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (159, N'John Smith', 22)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (722, N'Hilary White', 29)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (937, N'Emily Brown', 19)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (100, N'Frederic Miller', 16)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (99, N'Henry Lorens', 20)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (189, N'Bob Red', 25)

create table dbo.DisabledEmployee

(

DisabledPersonId int primary key,

EmployeeId int unique foreign key references dbo.Employee(EmployeeId)

)

insert into dbo.DisabledEmployee(DisabledPersonId, EmployeeId) values (1, 159)

insert into dbo.DisabledEmployee(DisabledPersonId, EmployeeId) values (2, 722)

insert into dbo.DisabledEmployee(DisabledPersonId, EmployeeId) values (3, 937)

Объяснения

Таблица DisabledEmployee имеет атрибут EmployeeId, что является внешним ключом. Он ссылается на атрибут EmployeeId таблицы Employee. Кроме того, этот атрибут имеет ограничение unique, что говорит о том, что в него могут быть записаны только уникальные значения. Соответственно, работник может быть записан в эту таблицу не более одного раза.

### **7. Обязательные и необязательные связи**

Связи можно поделить на обязательные и необязательные.

### **7.1. Один ко многим**

1. Один ко многим с обязательной связью:  
   К одному полку относятся многие бойцы. Один боец относится только к одному полку. Обратите внимание, что любой солдат обязательно принадлежит к одному полку, а полк не может существовать без солдат.
2. Один ко многим с необязательной связью:  
   На планете Земля живут все люди. Каждый человек живет только на Земле. При этом планета может существовать и без человечества. Соответственно, нахождение нас на Земле не является обязательным

Одну и ту же связь можно рассматривать как обязательную и как необязательную. Рассмотрим вот такой пример:

У одной биологической матери может быть много детей. У ребенка есть только одна биологическая мать.

А) У женщины необязательно есть свои дети. Соответственно, связь необязательна.  
Б) У ребенка обязательно есть только одна биологическая мать – в таком случае, связь обязательна.

### **7.2. Один к одному**

1. Один к одному с обязательной связью:  
   У одного гражданина определенной страны обязательно есть только один паспорт этой страны. У одного паспорта есть только один владелец.
2. Один к одному с необязательной связью:  
   У одной страны может быть только одна конституция. Одна конституция принадлежит только одной стране. Но конституция не является обязательной. У страны она может быть, а может и не быть, как, например, у Израиля и Великобритании.

Одну и ту же связь можно рассматривать как обязательную и как необязательную:

У одного человека может быть только один загранпаспорт. У одного загранпаспорта есть только один владелец.

А) Наличие загранпаспорта необязательно – его может и не быть у гражданина. Это необязательная связь.  
Б) У загранпаспорта обязательно есть только один владелец. В этом случае, это уже обязательная связь.

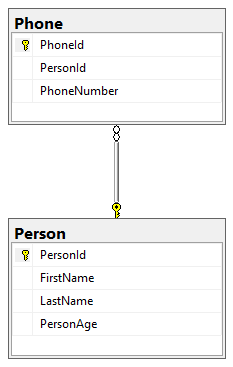
### **7.3. Многие ко многим**

Любая связь многие ко многим является необязательной. Например:

Человек может инвестировать в акции разных компаний (многих). Инвесторами какой-то компании являются определенные люди (многие).

А) Человек может вообще не инвестировать свои деньги в акции.  
Б) Акции компании мог никто не купить.

### **8. Как читать диаграммы?**

Выше я приводил диаграммы созданных нами таблиц. Но для того, чтобы их понимать, нужно знать, как их «читать». Разберемся в этом на примере диаграммы из пункта 5.3.  
  
  
  
Мы видим отношение один ко многим. Одной персоне принадлежит много телефонов.

1. Возле таблицы Person находится золотой ключик. Он обозначает слово «один».
2. Возле таблицы Phone находится знак бесконечности. Он обозначает слово «многие».

### **9. Итоги**

1. Связи бывают:
   * Многие ко многим.
   * Один ко многим.  
     1) с обязательной связью;  
     2) с необязательной связью.
   * Один к одному.  
     1) с обязательной связью;  
     2) с необязательной связью.
2. Связи организовываются с помощью внешних ключей.
3. Foreign key (внешний ключ) — это атрибут или набор атрибутов, которые ссылаются на primary key или unique другой таблицы. Другими словами, это что-то вроде указателя на строку другой таблицы.

### **10. Задачи**

Для лучшего усвоения материала предлагаю вам решить следующие задачи:

1. Описать таблицу фильм: id, название, длительность, режиссер, жанр фильма. Обратите внимание на то, что у фильма может быть более одного жанра, а к одному жанру может относится более, чем один фильм.
2. Описать таблицу песня: id, название, длительность, певец. При этом у песни может быть более одного певца, а певец мог записать более одной песни.
3. Реализовать таблицу машина: модель, производитель, цвет, цена
   * Описать отдельную таблицу производитель: id, название, рейтинг.
   * Описать отдельную таблицу цвета: id, название.

У одной машины может быть только один производитель, а у производителя — много машин. У одной машины может быть много цветов, а у одного цвета может быть много машин.

1. Добавить в БД из пункта 6.2. таблицу военно-обязанных по типу того, как мы описали отдельную таблицу DisabledEmployee.