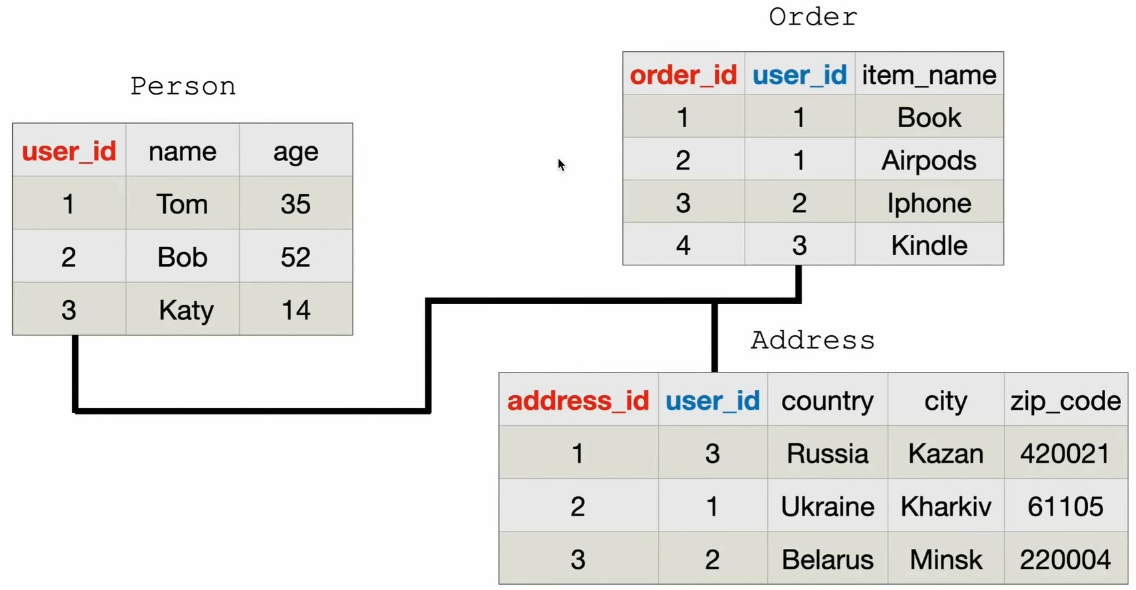
**Отношения связи в БД Внешний ключ Forein key**

**Primary key (первичный ключ)** — это колонка в таблице, которая уникально идентифицирует строку (другими словами — идентификатор строки в таблице).

**Foreign key (внешний ключ)** — это колонка которая позволяет устанавить связь мужду таблицами. Внешний ключ в одной таблице — это первичный ключ в другой таблице.

Как видно на картинке с помощью первичный и внешний ключей мы можем выстраивать связи между таблицами.

**Зачем разбивать данные на несколько таблиц ?**

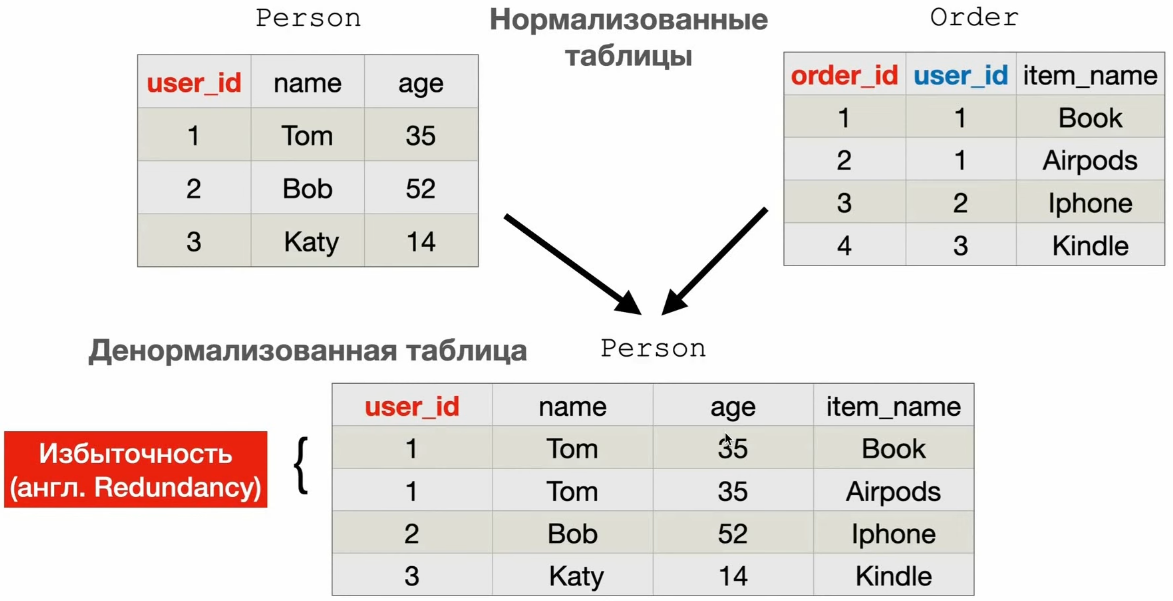
Это нужно для того, чтобы избежать избыточность данных и чтобы таблицы были нормализованными.

**Нормализованные таблицы** — это таблицы в которых нет лишних данных.

**Денормализованная таблицы** — содержит избыточность данных.

Избыточноть данных это плохо. При проектирвоании таблицы баз данных нужно избегать любого дублирования данных. Чтобы избежать дублирования нужно применять подход с разбиением одной большой таблицы на несколько таблиц и между этими таблицами мы выстраиваем свзяи. Этот процесс называется **нормализация**.

**Нормализация** — это процесс когда мы одну денормализованную таблицу разбиваем на несколько маленьких таблиц и между этими маленькими таблицами выстраиваем связи

с помощью первичного ключа и внешнего ключа.  


**Отношение Один ко Многим (One to Many)**

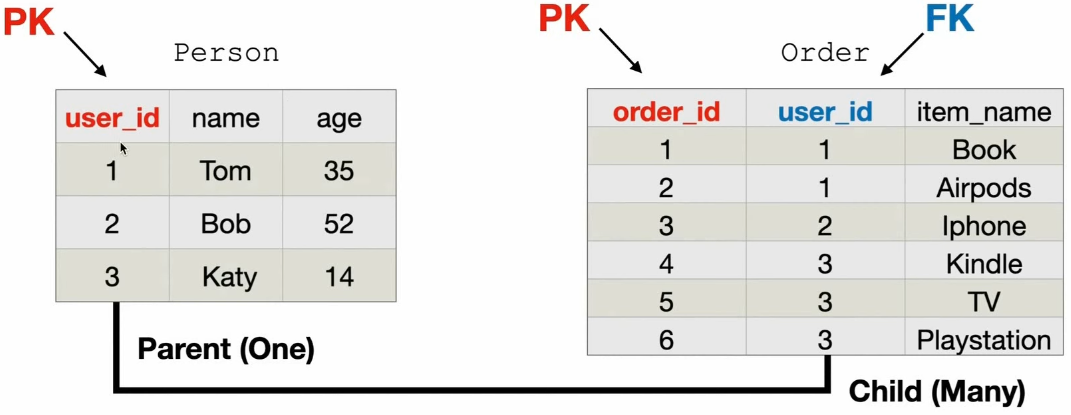
Отношение один ко многим это когда например - есть один покупатель и у этого покупателя есть много заказов (один покупатель ко многим заказам). Эти заказы храняться в отдельной таблице и каждый из этих заказов имеет ссылку на одного покупателя при этом покупатель не имеет никаких ссылок, покупатель просто содержится в какой то своей таблице и имеет у никальный идентификатор, а все заказы в себе содержат идентификатор этого покупателя.  
Другие примеры:

**1.** Учитель и ученики — один учитель и множество учеников.

**2.** Библиотека и книги — одна библиотека и множество книг, при этом каждая книга относится только к одной библиотеке.

**3.** Режиссер и его фильмы — один режиссер к его множеству фильмов при этом каждый из этих фильмов ссылается только на одного этого режиссера.

**Пример:**

У нас есть таблица Person, у неё есть первичный ключ(**PK**) — это у нас родительская таблица(**One**).

Так же есть таблица Order которая является дочерней(**Many**) — этих заказов может быть несколько. У таблицы Order(**Many**) помимо первичного ключа так же есть внешний ключ(**FK**), именно по внешнему ключу мы можем понять, что это у нас дочерняя таблица, **FK** их может быть много и они могут повторяться.

С помощью такой архитектуры мы достигаем то, что один человек может владеть несколькими заказами, а один заказ может принадлежать только одному человеку.

Итого:

**1. PK** содержит только уникальные значения, **FK** может содержать дубликаты.

**2.** Таблица содержащая **FK** – дочерняя. Таблица, содержащая только **PK** – родительская.

**3.** Один заказ не может приндлежать нескольким людям, так как на **order\_id** ограничние **PRIMARY KEY**.

**Отношение Один к Одному (One to One)**

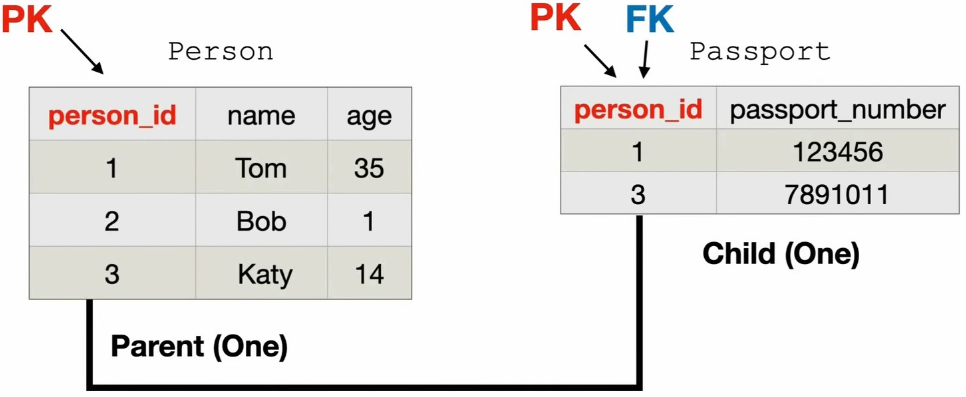
Пример отношения один к одному - у нас есть гражданин какой-то страны и у нас есть паспорт этой страны. У одного гражданина страны может быть только один паспорт этой страны и наоборот паспорт страны может принадлежать только одному человеку. У нас гражданин не может иметь несклолько паспортов, у него может быть только один паспорт и наоборот паспорт не может принадлежать нескольим гражданам, он может принадлоежать только одному гражданину.

Дргуие примеры:

**1.** презедент и страна

**2.** Директор и школа

**Пример отношения один к одному:**

  
У нас есть таблица **Person**, в этой таблице хранятся люди. У людей может быть только один паспорт. Так же есть таблица **Passport** (дочерняя таблица), у этой таблице есть колонка **person\_id** – она является одновременно **PK** и **FK,** тоесть колонка для внешнего ключа она совпадает с колонкой первичного ключа. Тот факт, что у таблицы **Passport** совпадают **PK** и **FK** не даст нам выдать одному человеку несколько паспортов, если мы захотим поместить в эту таблицу еще одну строку с существующим **person\_id** то у нас это сдлеать не получится потому-что на этой колонке есть ограничение **PK** и это ограничение не даст нам внедрить сюда дубликаты. Это значит, что мы одному человеку можем дать только один паспорт и наоборот, один паспорт может принадлежать только одному человеку.

Если в таблице **Passport** мы укажем **person\_id** который не существует в таблице **Person** – это будет некорректно, это будет означать, что этот пасопрт не принадлежит никакому человеку. Потомучто в таблице **Passport** мы можем указывать только те **id** которые есть в таблице Person. Именно по этому признаку можно определить какая таблица будет являться родительской, а какая дочерней потому-что у нас люди могут существовать без паспорта, но паспорта не могут существовать без людей.

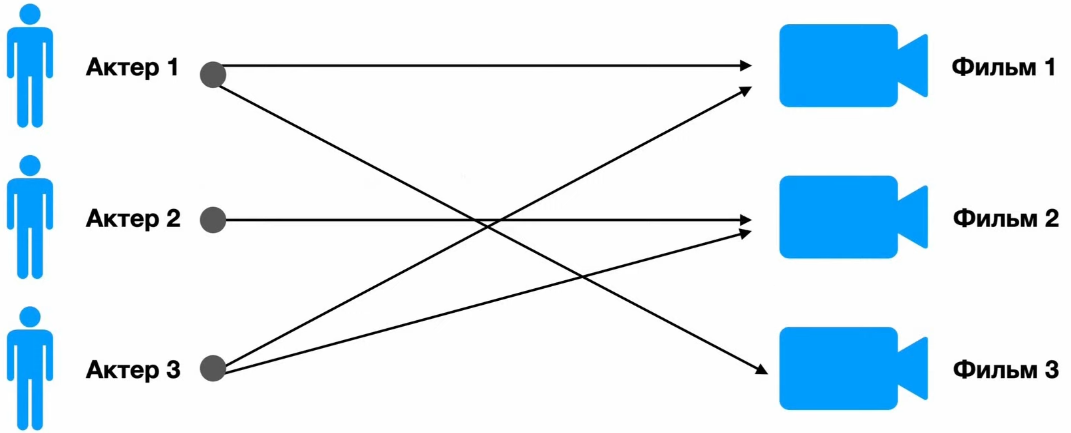
Итого:

**1.** В таблице Passport **FK** = **PK.**

**2.** Человек может существовать без паспорта (**Parent**), но паспорт не может сущесвовать без человека(**child**).

**3.** Колонка **person\_id** имеет ограничение **PRIMARY KEY**, поэтому у человека может быть только один паспорт.

**Отношение Многие ко Многим (Many to Many)**

Пример отношения многие ко многим — 1 актер за свою карьеру может сняться во многих фильмах и при этом в каком то фильме могут сняться множество актеров.  


Другие примеры:

1. Ученые и научные статьи - у ученного могут быть опубликованные несколько научных статей и у статьи могут быть несколько авторов.

2. Студенты и занятия — студент может посещать несколько занятий и при этом у занятия может быть несколько студентов.

**Как строиться отношение многие ко многим ?**

Для построения такой связи нам нужно создать еще одну таблицу, эта таблица называется связывающая таблица (**Join table**). Название такой таблицы обычно состоит из названия двух таблиц и эта связывающая таблица устанавливает свзяь многие ко многим между двумя таблицами.

В этой связывающей таблицы у нас только две колонки, колонка **actor\_id** и колонка **movie\_id.** Колонка **actor\_id** устанавливает связь с таблицей **Actor**, а колонка **actor\_id** устанавливает связь с таблией **Movie**. Эти две колонки помечаются как внешние ключи(**FK**).

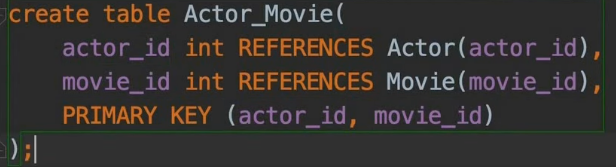
Так как в обоих колонках нахоядтся внешние ключи(**FK**), то в каждой колонки можем иметь дубликаты.

Здесь мы встрчаемся с новым понятием **Составной PRIMARY KEY.**

**Составной PRIMARY KEY –** говорит о том что мы можем в каждую из колонок помещать дубликаты, но мы не можем в этой таблице иметь строки которые полностью копируют уже существующую строку. Это нам позволит внедрять в эту связывающую таблицу значение дубликатов в каждую из колонок, но если вставляемая строка по двум колонкам сопадает со строкой которая уже есть в таблице, то база данных выдаст ошибку, так как этот **Соствной PRIMARY KEY** обеспечивает, что у нас в этой таблице не будет двух строк с одинаковыми значениями этих колонок.

**Составной PRIMARY KEY** обеспечивает уникалькальность двух колонок**.  
  
Пример создания связующей таблицы:**

Здесь actor\_id и movie\_id являются составным ключом.

  
  
  
  
  
  
  
  
  
  
  
**Пример создания связи двух таблиц через связующую таблицу:**

