Материал этой статьи напрямую не относиться к изучению языка SQL, так как имеет отношение к проектированию баз данных (БД), но для общего понимания взаимосвязи хранимой в системе информации она будет полезна.

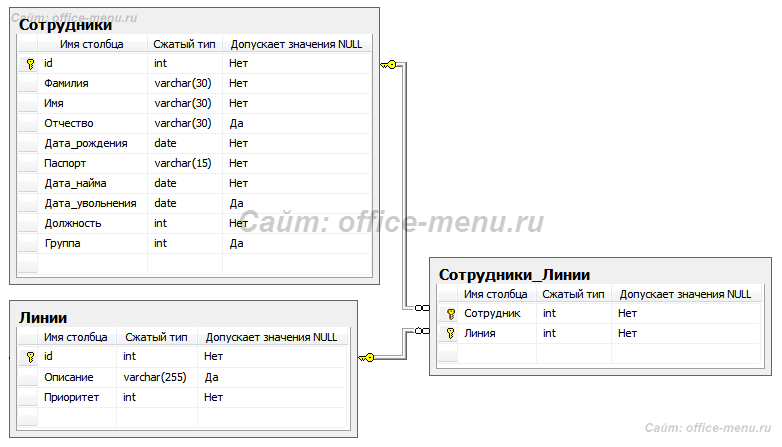
По поводу того, как должна быть спроектирована база нет 100% решения, потому что конкретный вариант может удовлетворять либо не удовлетворять различным бизнес-процессам и целям. Но не принимать во внимание элементарные правила нельзя, так как их соблюдение сохранит много времени, нервов и денег при работе с данными.

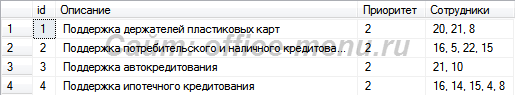
Нормализация баз данных заключается в приведении структуры хранения данных к нормальным формам (NF). Всего таких форм существует 8, но часто достаточным является соблюдение первых трех. Рассмотрим их более подробно на примере учебной базы данных. Примеры будут строится по принципу «что было бы, если было иначе, чем сейчас».

**Первая нормальная форма**

Основным правилом первой формы является необходимость неделимости значения в каждом поле (столбце) строки – **атомарность** значений.

Рассмотрим таблицы сотрудников и телефонных линий.

[](https://office-menu.ru/images/sql/many-to-many.png)

Чтобы избавиться от связывающей таблицы «Сотрудники\_Линии», мы могли бы записать идентификаторы сотрудников для каждой линии в виде перечня в дополнительном столбце:  
  
[](https://office-menu.ru/images/sql/notAtom.png)

Но подобная структура не является надежной. Представьте, что Вам необходимо поменять некоторым сотрудникам подключенные линии. Потребуется осуществить разбор составного поля, чтобы определить наличие id сотрудника в каждой записи линий, затем скорректировать перечень. Получается слишком сложный и долгий процесс для такой простой операции.

Организации структуры таблиц с применением дополнительной связывающей избавляет от подобных проблем.

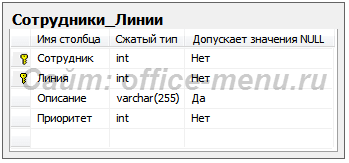
Помимо атомарности к первой нормальной форме относятся следующие правила:

* Строки таблиц не должны зависеть друг от друга, т.е. первая запись не должна влиять на вторую и наоборот, вторая на третью и т.д. Размещение записей в таблице не имеет никакого значения.
* Аналогичная ситуация со столбцами записей. Их порядок не должен влиять на понимание информации.
* Каждая строка должна быть уникальна, поэтому для нее определяется **первичный ключ**, состоящий из одного либо нескольких полей (**составной ключ**). Первичный ключ не может повторяться в пределах таблицы и служит идентификатором записи.

**Вторая нормальная форма**

Условием этой формы является отсутствие зависимости неключевых полей от части составного ключа.

Так как составной ключ в учебной базе наблюдается только в таблице «Сотрудники\_Линии», то рассмотрим пример на ней.

[](https://office-menu.ru/images/sql/2DNF.png)

На представленной диаграмме столбцы описания и приоритета зависят от столбца «Линия», входящего в составной ключ. Это значит, что для каждой линии, подключенной разным сотрудникам, потребуется повторно указывать описание и приоритетность. Подобная структура приводит к **избыточности данных**.

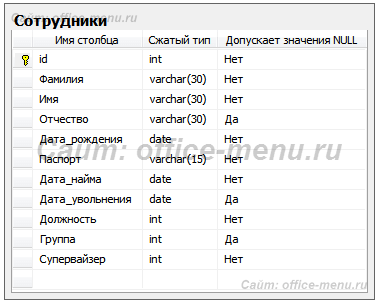
Также велика вероятность возникновения противоречивой информации. Изменяя приоритет или описание для линии, можно по ошибке оставить некоторые строки не обработанными. В таком случае, для одного и того же идентификатора линии значения зависимых полей будут различными.

Если соблюдены правила первой нормальной формы, то создание таблицы «Линии» и перенос в нее зависимых столбцов удовлетворяет второй нормальной форме.

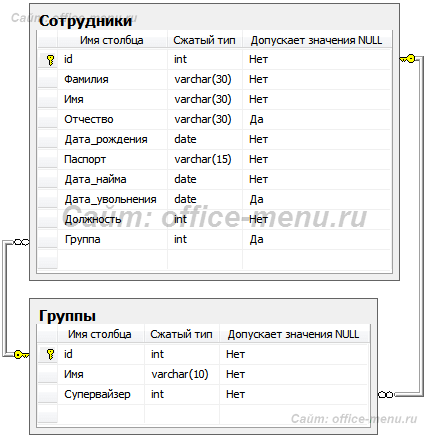
**Третья нормальная форма**

3NF схожа по логике с 2NF, но с некоторым отличием. Если 2 форма ликвидирует зависимости неключевых полей от части ключа, то третья нормальная форма исключает зависимость неключевых полей от других неключевых полей.

На приведенном примере таблицы сотрудников видно, что столбец «Супервайзер» имеет зависимость от столбца «Группа», а это значит, что при изменении значения поля группы, потребуется изменить значение поля супервайзера.

[](https://office-menu.ru/images/sql/3DNF.png)

Все риски, которые были рассмотрены для 2NF, так же относятся к 3NF и устраняются переносом зависимых полей в отдельную таблицу.

[](https://office-menu.ru/images/sql/3NF.png)

**Денормализация базы данных**

Теория нормальных форм не всегда применима на практике. Например, неатомарные значения не всегда являются «злом», а иногда наоборот. Связано это с необходимостью дополнительного объединения (следовательно, затрат производительности системы) при выполнении запросов, особенно когда производится обработка большого массива информации.

Для баз данных, предназначенных для аналитики, часто выполняют денормализацию, чтобы укорить выполнение запросов.