**Реляционные базы данных**

Определение гласит «реляционная база данных – база данных, основанная на реляционной модели данных». Звучит так себе, попробую раскрыть.

Термин реляционный значит, что теория основана на математическом понятии «**отношение**» (relation). Не математики стараются уйти от теории, заменяя «отношение» на термин «**таблица**».

Понять не могу как они пришли к подобной замене, но в раскрываемом виде баз данных используются оба этих термина, так как реляционные базы данных состоят из таблиц и отношений между этими таблицами.

Как это выглядит?

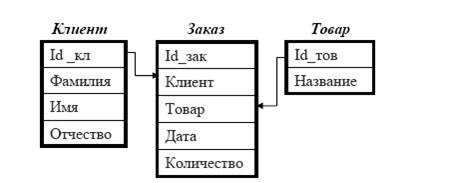


Рисунок 1. Пример реляционной базы данных

На рисунке 1 представлен простейший пример. Наша база данных состоит из трёх таблиц, где каждая таблица описывает небольшую сущность в контексте нашей предметной области (в данном случае, это похоже на ларёк или очень маленький магазин).

Как можно заметить из данного рисунка, помимо таблиц у нас есть стрелки, указывающие на отношения (связь) между этими таблицами.

Каждая таблица состоит из полей, вмещающих в себя определённую информацию. Как можно заметить, у каждой таблицы есть поле с префиксом «Id». Данное поле содержит в себе уникальное значение, которое позволяет разделять повторяющиеся записи в таблице, а также помогает с поиском определённой записи (пример – у нас может быть несколько клиентов с одинаковой фамилией, именем и отчеством, но они разные люди, в этом случае эту различность и покажет поле Id).

Данные таблицы представляют собой логическое представление хранения данных. При добавлении новой записи в таблицу «Клиент» мы получим следующее физическое представление данных:

**TABLE «Клиент» {**

**Id\_кл: 0,**

**Фамилия: «Иванов»,**

**Имя: «Иван»,**

**Отчество: «Иванович»**

**}**

Таким образом мы получаем, что таблица - это «разметка» для данных.

Связи же представляют из себя ссылку одной таблицы на другую. Ссылаются они с помощью значений поля Id. К примеру, у нас есть следующая запись в таблице «Заказ»:

**TABLE «Заказ» {**

**Id\_зак: 0,**

**Клиент: 0,**

**Товар: 0,**

**Дата: «03/12/2019»,**

**Количество: «1»**

**}**

Можно заметить, что значение в поле «Клиент» равно 0, что означает связь между записью в таблице «Клиент» с «Id\_кл» равной 0.

Почему существует подобный механизм связей?

Данный механизм нужен для защиты от ненавистного всеми программистами повторения одних и тех же записей. Попробую объяснить, что будет, если мы уберём механизм ссылок.

Допустим, у нас есть такая таблица заказа.

**TABLE «Заказ» {**

**Id\_зак: 0,**

**Фамилия: «Иванов»,**

**Имя: «Иван»,**

**Отчество: «Иванович»,**

**Товар: 0,**

**Дата: «03/12/2019»,**

**Количество: «1»**

**}**

Просто вставив вместо ссылки на другую таблицу сразу значения этой таблицы, можно сказать, что мы упростили понимание и хранение данных.

Хорошо, а что, если этот клиент решил заказать по одной единице КАЖДОГО товара в нашем магазине? Представим, что товара у нас действительно много.

Мы получим, допустим, пару сотен подобных записей, в каждой из которых будет фигурировать одинаковые значения фамилии, имени и отчества. А теперь представим, что таких клиентов у нас сотни и каждый совершает подобные заказы каждый день. Помимо того, что мы расходуем память зря, повторяясь в значениях, мы получаем ещё дополнительную проблему.

Представим себе, что наш постоянный клиент решил сменить фамилию. До этого он исправно делал покупки в нашем магазине почти каждый день и база имеет несколько десятков тысяч записей с его старыми данными. И нам нужно будет обойти КАЖДУЮ запись, чтобы ввести его новые данные. Либо забыть об этих данных, лишая себя достоверной информации о продажах.

Решением здесь и выступают связи – если каждая запись заказа будет содержать лишь указатель на запись с данными о клиенте, то мы не будем повторяться в значениях, расходуя память, а изменение данных у клиента не будут вызывать никаких проблем.