**Реляционная база данных** — база данных, основанная на реляционной модели данных. Слово «реляционный» происходит от англ. *relation* («отношение», «зависимость», «связь»). Для работы с реляционными БД применяют реляционные СУБД.

Использование реляционных баз данных было предложено доктором Коддом из компании IBM в 1970 году.

### Нормализация

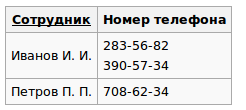
Целью нормализации реляционной базы данных является устранение недостатков структуры базы данных, приводящих к избыточности, которая, в свою очередь, потенциально приводит к различным аномалиям и нарушениям целостности данных.

Теоретики реляционных баз данных в процессе развития теории выявили и описали типичные примеры избыточности и способы их устранения.

#### 1ая нормальная форма

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия *отношение*. Что же касается различных *таблиц*, то они могут не быть правильными *представлениями отношений* и, соответственно, могут не находиться в 1НФ.



#### 2ая нормальная форма

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа.





(Наличие компьютера полностью зависит от должности)

#### 3ая нормальная форма

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых. (По сути повторяет вторую нормальную форму).

### Триггер

**Три́ггер** (англ. *trigger*) — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

### SQL

**SQL (*structured query language* — «язык структурированных запросов») —** формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных (СУБД).

SQL является прежде всего информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL можно назвать языком программирования, при этом он не является тьюринг-полным, но вместе с тем стандарт языка спецификацией [SQL/PSM](https://ru.wikipedia.org/wiki/SQL/PSM) предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Каждое предложение SQL — это либо *запрос* данных из базы, либо обращение к базе данных, которое приводит к изменению данных в базе. В соответствии с тем, какие изменения происходят в базе данных, различают следующие типы запросов:

* запросы на создание или изменение в базе данных новых или существующих объектов (при этом в запросе описывается тип и структура создаваемого или изменяемого объекта);
* запросы на получение данных;
* запросы на добавление новых данных (записей);
* запросы на удаление данных;
* обращения к СУБД.

Основным объектом хранения реляционной базы данных является таблица, поэтому все SQL-запросы — это операции над таблицами. В соответствии с этим, запросы делятся на:

* запросы, оперирующие самими таблицами (создание и изменение таблиц);
* запросы, оперирующие с отдельными записями (или строками таблиц) или наборами записей.

Каждая таблица описывается в виде перечисления своих полей (столбцов таблицы) с указанием

* типа хранимых в каждом поле значений;
* связей между таблицами (задание первичных и внешних ключей);
* информации, необходимой для построения индексов.

Запросы первого типа в свою очередь делятся на запросы, предназначенные для создания в базе данных новых таблиц, и на запросы, предназначенные для изменения уже существующих таблиц. Запросы второго типа оперируют со строками, и их можно разделить на запросы следующего вида:

* вставка новой строки;
* изменение значений полей строки или набора строк;
* удаление строки или набора строк.

Самый главный вид запроса — это запрос, возвращающий (пользователю) некоторый набор строк, с которым можно осуществить одну из трёх операций:

* просмотреть полученный набор;
* изменить все записи набора;
* удалить все записи набора.

Таким образом использование SQL сводится, по сути, к формированию всевозможных выборок строк и совершению операций над всеми записями, входящими в набор.

#### Преимущества

**Наличие стандартов**

Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка. Правда, стоит обратить внимание, что сам по себе стандарт местами чересчур формализован и раздут в размерах (например, базовая часть стандарта SQL:2003 состоит из более чем 1300 страниц текста).

**Декларативность**

С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных.

**Интересный факт**

Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что превратился в инструмент программиста.

### ORM

**ORM** (англ. ***o****bject-****r****elational* ***m****apping*, рус. *объектно-реляционное отображение*) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.

В объектно-ориентированном программировании [объекты](https://ru.wikipedia.org/wiki/Объект_(программирование)) в программе представляют объекты из реального мира. В качестве примера можно рассмотреть адресную книгу, которая содержит список людей с нулём или более телефонов и нулём или более адресов. В терминах объектно-ориентированного программирования они будут представляться объектами класса «Человек», которые будут содержать следующий список полей: имя, список (или массив) телефонов и список адресов.

Суть задачи состоит в преобразовании таких объектов в форму, в которой они могут быть сохранены в файлах или базах данных, и которые легко могут быть извлечены в последующем, с сохранением свойств объектов и отношений между ними. Эти объекты называют «хранимыми» (англ. *persistent*). Исторически существует несколько подходов к решению этой задачи.

Разработано множество пакетов, устраняющих необходимость в преобразовании объектов для хранения в реляционных базах данных.

Некоторые пакеты решают эту проблему, предоставляя библиотеки классов, способных выполнять такие преобразования автоматически. Имея список таблиц в базе данных и объектов в программе, они автоматически преобразуют запросы из одного вида в другой. В результате запроса объекта «человек» (из примера с адресной книгой) необходимый SQL-запрос будет сформирован и выполнен, а результаты «волшебным» образом преобразованы в объекты «номер телефона» внутри программы.

С точки зрения программиста система должна выглядеть как постоянное хранилище объектов. Он может просто создавать объекты и работать с ними как обычно, а они автоматически будут сохраняться в реляционной базе данных.

На практике всё не так просто и очевидно. Все системы ORM обычно проявляют себя в том или ином виде, уменьшая в некотором роде возможность игнорирования базы данных. Более того, слой транзакций может быть медленным и неэффективным (особенно в терминах сгенерированного SQL). Все это может привести к тому, что программы будут работать медленнее и использовать больше памяти, чем программы, написанные «вручную».

Но ORM избавляет программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки. Кроме того, большинство современных реализаций ORM позволяют программисту при необходимости самому жестко задать код SQL-запросов, который будет использоваться при тех или иных действиях (сохранение в базу данных, загрузка, поиск и т. д.) с постоянным объектом.