

БАЗЫ ДАННЫХ

ТЕОРЕТИЧЕСКИЙ МИНИМУМ



ARCHITECT

Базы данных

Приведу основные **термины теории баз данных**, в дальнейшем мы будем расширять свой словарь:

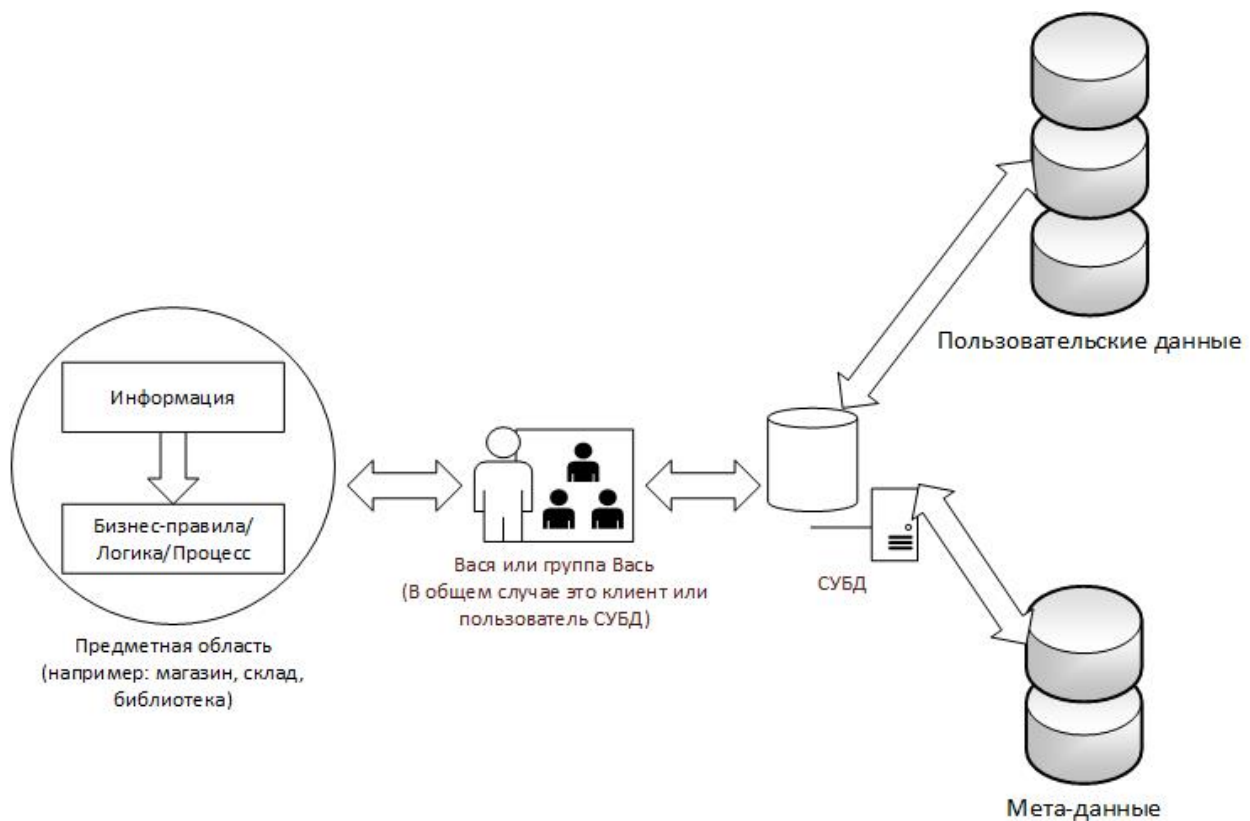
1. **SQL** – язык структурированных запросов. При помощи данного языка мы будем производить различные действия с нашими базами данных.
2. **Данные** – структурированная информация.
3. **База данных** – совокупность данных, определенных предметной областью. Проще говоря: набор таблиц
4. **Таблица** (сущность) – способ хранения информации в реляционной базе данных (минимальная единица измерения в базе данных). Таблицей упрощенно называют отношение в базе данных.
5. **СУБД** – система управления базами данных: программа, позволяющая производить различные действия с базами данных.
6. **Бизнес-правила** – формальные правила, которые учитываются при проектировании базы данных. Другими словами: это процесс или набор процессов, которые нам необходимо описать и автоматизировать при помощи базы данных.
7. **Информационная система** = **база данных** + **СУБД** и другое **ПО** + «**железо**». Некоторые расширяют данное понятие:
Информационная система = **база данных** + **СУБД** и другое **ПО** + «**железо**» + **человек**. Некоторые наоборот сужают:
Информационная система = **база данных** + **СУБД** и другое **ПО**.

8. **Предметная область** – часть реального мира, которая описывается или автоматизируется при создании базы данных. Например: склад, магазин, библиотека, автозаправка.

9. **Клиент** – человек или программа, обращающийся(аяся) к базе данных.

Как связаны термины реляционной базы данных между собой

Попробуем отобразить графически **связи между терминами**, определение которым мы дали выше.



Терминология баз данных в «живом» мире

Слева в виде кружка я изобразил предметную область, которая состоит из информации и бизнес-правил. Например, к нам на склад завезли авторучки, другими словами появилась новая информация,

которая запустила некоторые процессы (бизнес-правила): грузчик стал разгружать, кладовщик считать, бухгалтер оприходовать.

Справа в прямоугольнике у нас находится информационная система, там есть какой-то сервер с установленной СУБД, а СУБД работает с базами данных, которые состоят из пользовательских данных (таблиц) и мета-данных (служебные данные, необходимые для работы с базой данных). Хочу обратить ваше внимание на то, что база данных должна обязательно в себе содержать, как пользовательские данные, так и служебные.

В центре всей схемы у нас находится клиент. Это может быть разработчик баз данных, в качестве клиента может выступать и приложение, а может это просто оператор. Клиент взаимодействует как с предметной областью, так и с информационной системой. Давайте представим, что в центре находится разработчик, который изучает предметную область слева, чтобы перенести ее в базу данных справа. Чем лучше разработчик будет понимать предметную область, тем качественнее он построит информационную систему (именно поэтому я бы не стал включать человека в информационную систему).

Виды связей между таблицами в базе данных. Связи в реляционных базах данных. Отношения, кортежи, атрибуты.

Сразу скажу, что **связей между таблицами в реляционной базе данных** всего три. Поэтому их изучение, понимание и восприятие пройдет быстро, легко и безболезненно. Приступим к изучению.

Термины кортеж, атрибут и отношение в реляционных базах данных

Я буду стараться объяснять теорию баз данных не с математической точки зрения, а на примерах. Грубо говоря, на пальцах. Во-первых, практические примеры позволяют легче усваивать материал. Во-вторых, с математической теорией проще разобраться, когда понимаешь суть происходящего.

Давайте разбираться с тем, что такое: **отношение, кортеж, атрибут** в реляционной базе данных.

	ID	Name	CountryCode	District	Population
1	1	Kabul	AFG	Kabol	1780000
2	2	Qandahar	AFG	Qandahar	237500
3	3	Herat	AFG	Herat	186800
4	4	Mazar-e-Sharif	AFG	Balkh	127800
5	5	Amsterdam	NLD	Noord-Holland	731200
6	6	Rotterdam	NLD	Zuid-Holland	593321
7	7	Haag	NLD	Zuid-Holland	440900
8	8	Utrecht	NLD	Utrecht	234323
9	9	Eindhoven	NLD	Noord-Brabant	201843
10	10	Tilburg	NLD	Noord-Brabant	193238
11	11	Groningen	NLD	Groningen	172701
12	12	Breda	NLD	Noord-Brabant	160398
13	13	Apeldoorn	NLD	Gelderland	153491

У нас есть простая таблица City из базы данных World, в которой есть строки и столбцы. Но термины: **таблица**, **строка**, **столбец** – это термины стандарта **SQL**.

Кстати: ни одна из существующих в мире **СУБД** не имеет полной поддержки того или иного стандарта **SQL**, но и ни один стандарт **SQL** полностью не реализует математику реляционных баз данных.

В терминологии реляционных баз данных: **таблица** – это отношение (принимается такое допущение), **строка** – это кортеж, а **столбец** – **атрибут**. Иногда вы можете услышать, как некоторые разработчики называют строки записями. Чтобы не было путаницы в дальнейшем предлагаю использовать термины SQL.

Если рассматривать таблицу, как объект (например книга), то столбец – это характеристики объекта, а строки содержат информацию об объекте.

Виды и типы связей между таблицами в реляционных базах данных

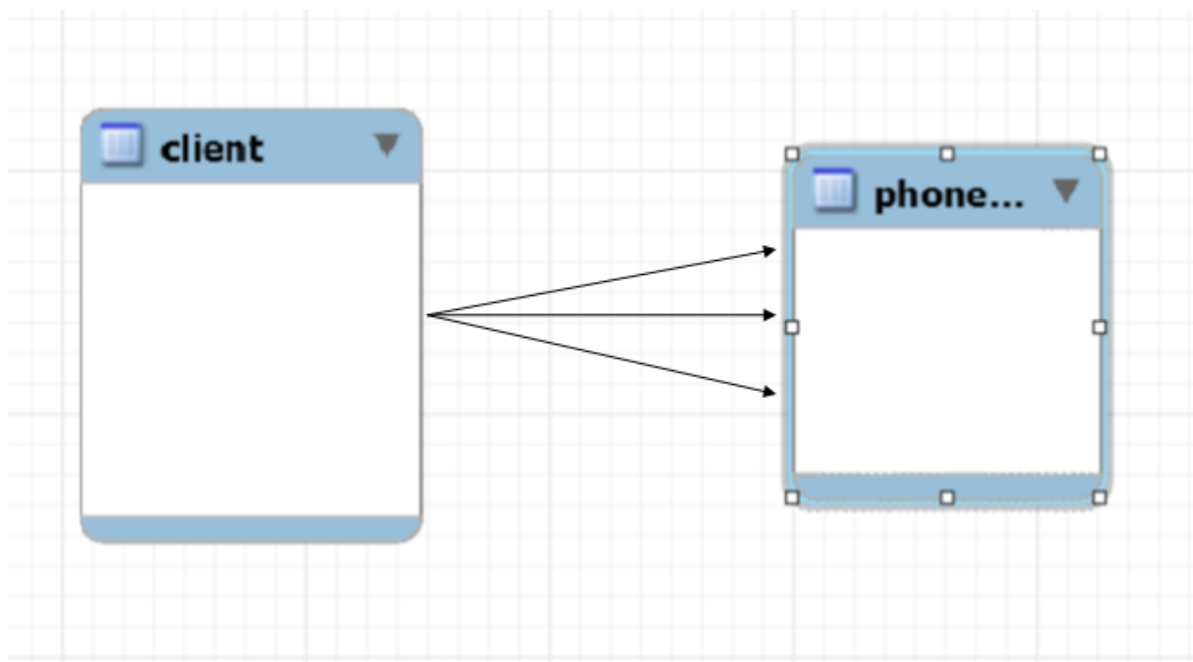
Давайте теперь рассмотрим то, как могут быть связаны таблицы в реляционных базах данных. Сразу скажу, что всего существует три вида связей между таблицами баз данных:

- связь один к одному (**one-to-one**);
- связь один ко многим (**one-to-many**);
- связь многие ко многим (**many-to-many**).

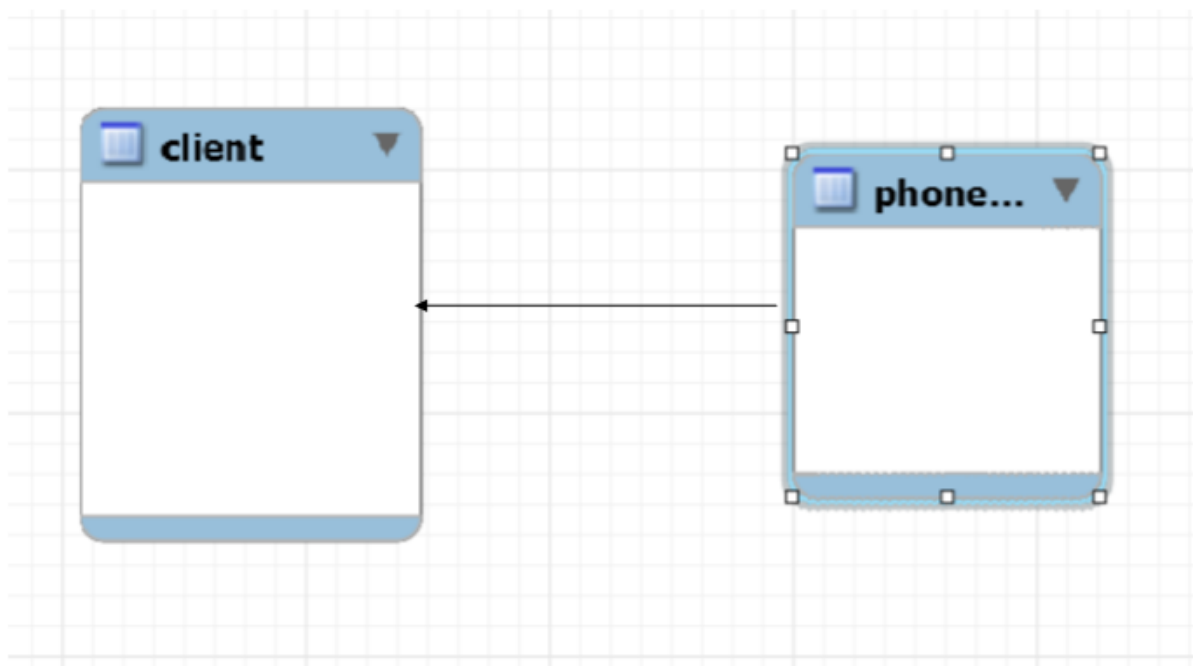
Рассмотрим, как такие связи между таблицами могут быть реализованы в реляционных базах данных.

Реализация связи один ко многим в теории баз данных

Связь один ко многим (one-to-many) в реляционных базах данных реализуется тогда, когда объекту А может принадлежать или же соответствовать несколько объектов Б, но объекту Б может соответствовать только один объект А. Не совсем понятно, поэтому посмотрим пример ниже.



У одного клиента может быть несколько номеров

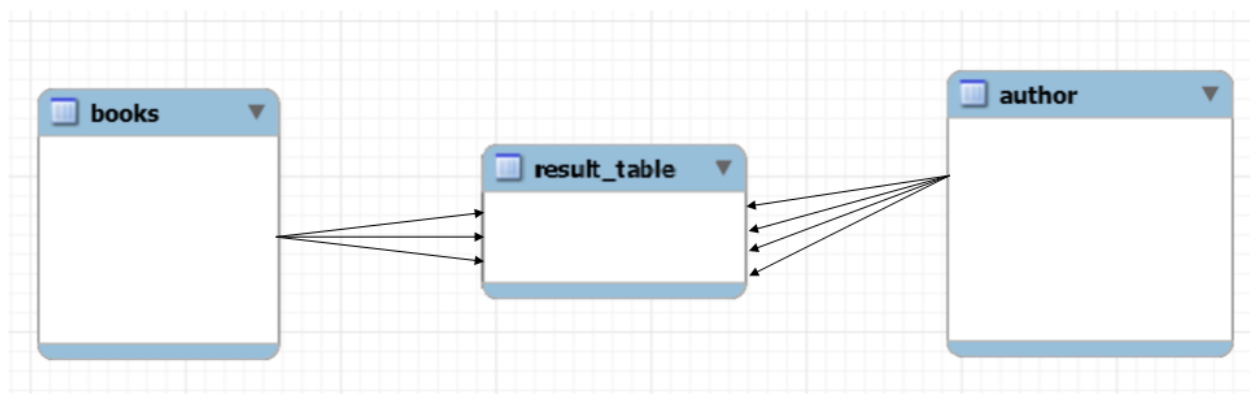


Но за номером закреплён лишь один клиент

У нас есть таблица, в которой содержатся данные о клиентах и у нас есть таблица, в которой хранятся их телефоны. Мы можем смело утверждать, что у одного клиента может быть несколько телефонов, но в тоже время мы можем быть уверены в том, что один конкретный номер может быть только у одного клиента. Это типичный **пример связи один ко многим**.

Связь многие ко многим (**many-to-many**)

Связь многие ко многим реализуется в том случае, когда нескольким объектам из таблицы **А** может соответствовать несколько объектов из таблицы **Б**, и в тоже время нескольким объектам из таблицы **Б** соответствует несколько объектов из таблицы **А**. Рассмотрим простой пример.



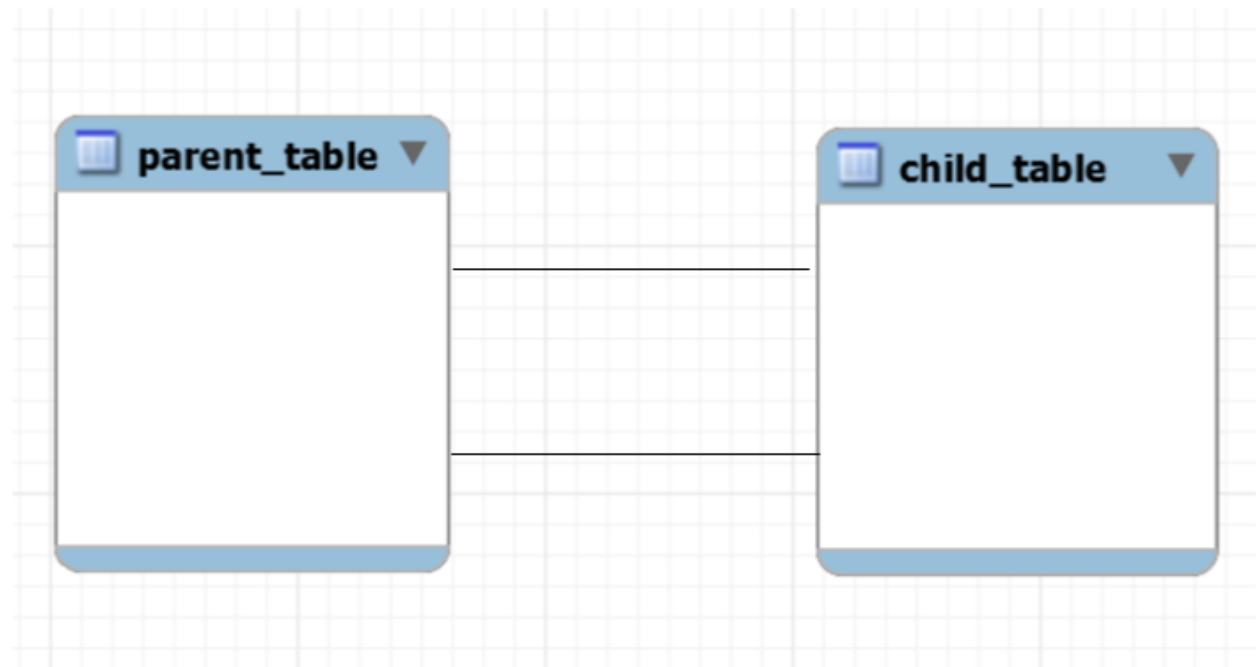
Одна книга могла быть написана несколькими авторами.
Автор мог написать несколько книг.

У нас есть таблица с книгами и есть таблица с авторами. Приведу два верных утверждения. Первое: **одну книгу может написать несколько авторов**. Второе: **автор может написать несколько книг**. Здесь мы наблюдаем типичную ситуацию, когда **связь между**

таблицами - многие ко многим (**many-to-many**). Такая связь (**связь многие ко многим**) реализуется путем добавления третьей таблицы.

Связь один к одному (**one-to-one**)

Связь один к одному – самая редко встречаемая связь между таблицами. В 97 случаях из 100, если вы видите такую связь, вам необходимо объединить две таблицы в одну.



Таблицы будут **связаны один к одному** тогда, когда одному объекту таблицы **А** соответствует один объект таблицы **Б**, и одному объекту таблицы **Б** соответствует один объект таблицы **А**. Как я уже говорил: если вы видите, что **связь один к одному** – смело объединяйте таблицы в одну, за исключением тех случаев, когда происходит модернизация базы данных.

Например, у нас была таблица, в которой хранились данные о сотрудниках компании. Но произошли какие-то изменения в бизнес-процессе и появилась необходимость создать таблицы с теми же

самыми сотрудниками, но не для всей компании, а разбив их по отделам. Таблицы отделов будут дочерними по отношению к таблице, в которой хранятся данные обо всех сотрудниках компании, и связаны такие таблицы будут связью **один к одному**.

Ключи и ключевые атрибуты в базах данных

Рассмотрим одну из самых простых, но очень важных тем в теории баз данных – **ключи и ключевые атрибуты**.

Давайте посмотрим, какие ключи и ключевые атрибуты бывают в таблицах баз данных:

1. **Ключи или ключевой атрибут** — атрибут (читай столбец) или набор атрибутов, который однозначно идентифицирует сущность/объект/таблицу в базе данных.
2. **Первичный ключ** — ключ, который используется для идентификации объекта.
3. **Ключ-кандидат** (альтернативный ключ) — ключ, по каким-либо причинам неиспользуемый как первичный.
4. **Составной ключ** — ключ, который использует несколько атрибутов.
5. **Суррогатный ключ** — ключ, значение которого генерируется СУБД.

Ключевые атрибуты или ключи по своему виду делятся на: простые и составные, естественные и суррогатные, первичные ключи и ключи кандидаты.

Рассмотрим различие между естественными и суррогатными ключами, естественно, на примере. Для этого обратимся к таблице с городами из базы данных World.

	ID	Name	CountryCode	District	Population
1	1	Kabul	AFG	Kabol	1780000
2	2	Qandahar	AFG	Qandahar	237500
3	3	Herat	AFG	Herat	186800
4	4	Mazar-e-Sharif	AFG	Balkh	127800
5	5	Amsterdam	NLD	Noord-Holland	731200
6	6	Rotterdam	NLD	Zuid-Holland	593321
7	7	Haag	NLD	Zuid-Holland	440900
8	8	Utrecht	NLD	Utrecht	234323
9	9	Eindhoven	NLD	Noord-Brabant	201843
10	10	Tilburg	NLD	Noord-Brabant	193238
11	11	Groningen	NLD	Groningen	172701
12	12	Breda	NLD	Noord-Brabant	160398
13	13	Apeldoorn	NLD	Gelderland	153491

В этой таблице ключом является столбец ID, данный столбец автоматически генерируется СУБД при добавлении новой записи в таблицу, следовательно, атрибут **ID–суррогатный ключ**. В данной таблице мы видим столбец с именем CountryCode, который может выступать в роли ключа для таблицы Country, **такой ключ будет естественным**.

Как нам определить, что столбец может быть ключом? Есть два очень простых признака того, что столбец является **ключом или ключевым атрибутом**: ключ уникален и ключ вечен. Но хочу отметить, что ключ – абстрактное понятие. Например, представим, что у нас есть таблица, в которой хранится информация о учениках класса, в принципе, ничего страшного не будет, если в такой таблице столбец ФИО будет выступать в роли ключа. Но, когда наша база данных работает в масштабах города, области, региона или страны,

то столбец ФИО никак не может выступать в роли ключа, даже номер паспорта – это не ключ, так как со временем мы меняем паспорт, а у несовершеннолетних его нет.

Поясню **принцип составного ключа**. Представим, что гражданин Петров был задержан сотрудниками полиции в нетрезвом виде за нарушение правопорядка. По факту задержания составляется рапорт (гражданин Петров не имеет при себе паспорта). Сотрудник полиции в рапорте укажет ФИО задержанного, но ФИО в масштабах города никак не идентифицируют Петрова Петра Петровича, поэтому сотрудник записывает дату рождения, если город большой, то Петр Петрович Петров, родившийся 14 февраля 1987 года в нем не один, поэтому записывается адрес фактического проживания и адрес прописки, для достоверности указывается время задержания. Сотрудник полиции составил набор характеристик, которые однозначно идентифицируют гражданина Петрова. Другими словами, все эти характеристики – **составной ключ**.

Типы данных в SQL

Типы данных в SQL важно знать, чтобы правильно и грамотно проектировать базы данных, правильно выбранный тип данных в SQL может очень сильно облегчить работу другим разработчикам. Поэтому не советую вам пропускать раздел о типах данных и всегда, когда вы знакомитесь с новой СУБД, обращать внимание на типы данных, которые поддерживает программа и на то, как она эти типы данных обрабатывает.

Типы данных в SQL

Рассмотрим типы данных, которые есть в языке SQL. У каждого столбца таблицы (у атрибута) должен быть тип данных для значений, которые хранятся в столбце. Тип данных для столбца определяется при создании таблицы, а еще лучше, когда он определяется на этапе проектирования баз данных.

Язык SQL делит данные на пять типов:

- Целочисленный тип данных SQL.
- Вещественный тип данных SQL.
- Типы данных даты и времени SQL.
- Строковый тип данных SQL.
- Строковый тип данных SQL в кодировки Юникод.

Давайте посмотрим, что включает в себя каждый из **типов данных SQL** и какие значения они позволяют нам хранить.

Целочисленный тип данных SQL

Целочисленный тип данных в SQL довольно таки широкий и зависит от реализации СУБД, снизу в таблицы лишь некоторые типичные примеры целочисленного типа данных SQL.

Тип данных SQL	от	до
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	$-10^{38} + 1$	$10^{38} - 1$
numeric	$-10^{38} + 1$	$10^{38} - 1$
money	-922,337,203,685,478	922,337,203,685,478
smallmoney	-214,748.3648	214,748.3647

Еще раз повторяюсь, что набор целочисленных **типов данных в SQL** зависит целиком и полностью от СУБД.

Вещественный тип данных SQL

Вещественный тип данных SQL, как и целочисленный тип данных, целиком и полностью зависит от реализации СУБД.

Тип	от	до
float	$-1.79E + 308$	$1.79E + 308$
real	$-3.40E + 38$	$3.40E + 38$

Тип данных даты и времени

Обычно в различных СУБД дата и время — это отдельно выделенный тип данных

Тип данных SQL	от	до
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079
date	Сохраняет дату как June 30, 1991	

time	Сохраняет время как 12:30 P.M.
------	--------------------------------

Строковый тип данных SQL

Строковых типов данных в SQL может быть гораздо больше, смотрите спецификацию своей СУБД, чтобы узнать какой тип данных может быть использован для строковых значений.

Тип данных SQL	Описание
char	Максимальная длина 8000 символов (все значения в столбце имеют фиксированный размер, указанный при объявлении столбца). Обратите внимание: единица измерения SQL типа данных char - символ.
varchar	Максимальная длина 8000 символов (все значения в столбце имеют различный размер в зависимости от количества символов, но не более того размера, что был указан при объявлении столбца). Обратите внимание: единица измерения SQL типа данных varchar - символ.

varchar (max)	Максимальная длина 231 символ. Обратите внимание: единица измерения SQL типа данных varchar (max) - символ.
text	Максимальная длина 2,147,483,647 символов. Обратите внимание: единица измерения SQL типа данных text - символ.

Строковый тип данных SQL в кодировке юникод

Да, некоторые СУБД выделяют строковый тип данных в кодировке юникод, поскольку в строках с кодировкой юникод на запись одного символа тратится, как минимум, 8 байт, поэтому мы и выделяем **строковый тип данных в SQL с кодировкой юникод**.

Тип данных SQL	Описание
nchar	Максимальная длина 4000 символов (все значения в столбце имеют фиксированный размер, указанный при объявлении столбца). Обратите внимание: единица измерения SQL типа данных nchar - символ.

nvarchar	Максимальная длина 4000 символов (все значения в столбце имеют различный размер в зависимости от количества символов, но не более того размера, что был указан при объявлении столбца). Обратите внимание: единица измерения SQL типа данных nvarchar - символ.
nvarchar (max)	Максимальная длина 231 символ. Обратите внимание: единица измерения SQL типа данных nvarchar (max) — символ.
ntext	Максимальная длина 1,073,741,823 символов. Переменная длина. Обратите внимание: единица измерения SQL типа данных ntext — символ.

Бинарный тип данных в SQL

Бинарный тип данных в SQL используется для хранения информации в бинарном виде (в виде последовательности байт).

Тип данных SQL	Описание
binary	Максимальная длина 8,000 байт (все значения в столбце имеют фиксированный размер, указанный при объявлении столбца). Обратите внимание: единицы измерения типа данных binary — байты.

varbinary	Максимальная длина 8,000 байт (все значения в столбце имеют различный размер в зависимости от количества символов, но не более того размера, что был указан при объявлении столбца). Обратите внимание: единицы измерения типа данных varbinary — байты.
varbinary (max)	Максимальная длина 231 байт. Обратите внимание: единицы измерения типа данных varbinary (max) — байты.
image	Максимальная длина 2,147,483,647 байт (все значения в столбце имеют различный размер в зависимости от количества символов, но не более того размера, что был указан при объявлении столбца). Обратите внимание: единицы измерения типа данных image — байты.

Хочу обратить ваше внимание на то, что каждая СУБД поддерживает свой набор типов данных и размерностей, мы рассмотрели базовые типы данных языка SQL.

Знаковые и без знаковые типы данных в SQL

Хочу обратить ваше внимание на то, что в некоторых СУБД числовые типы данных могут делиться на знаковые и без знаковые типы данных. И это совершенно разные типы данных.

Например, если у вас в одной таблице хранится id INTEGER UNSIGNED, а в другой id_table1 INTEGER SIGNED, то [связь между](#)

таблицами по этим двум столбцам вы реализовать не сможете, так как у них разные типы данных. Будьте внимательны и всегда читайте описания типов данных, когда начинаете использовать новую СУБД.

Приведем пример практического применения типов данных **SIGNED** и **UNSIGNED**. Например, у MySQL есть тип данных **TINYINT**, который занимает один байт и прекрасно подходит для хранения возраста, следовательно, в столбец с типом **TINYINT** можно записывать целые числа от 0 до 256 (два в восьмой степени, в одном байте 8 бит, а один бит может принимать два значения: ноль или единицу), если столбец без знаковый, если столбец знаковый, то в него можно записать числа от -128 до 127 (один бит уйдет на знак).

Запросы MySQL

MySQL запрос – это обращение к базе данных MySQL, с помощью которого мы можем реализовать: получение, изменение, удаление, сортировку, добавление, и другие манипуляции с данными базы.

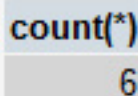
Зная структуру БД, таблиц в БД и полей, можно посылать следующие запросы в MySQL.

Select запросы

пользуясь данными запросами, мы будем выбирать (читать) информацию из БД.

SELECT count(*) **FROM** table_name;

Выведет количество всех записей в таблице



count(*)
6

SELECT * FROM table_name;

Выбирает все записи из таблицы БД

id	site	description
1	sitear.ru	SiteAR
2	sitear.ru	SiteAR
3	yaveterinar.ru	Ветеринария
4	wi-korporaciya.ru	О корпорации
5	sitear.ru	пример 1
6	sitear.ru	пример 2

SELECT * FROM table_name **LIMIT** 2,3;

Выбирает 3 записи из таблицы, начиная с 2 записи. Этот запрос полезен при создании блока страниц навигации.

id	site	description
3	yaveterinar.ru	Ветеринария
4	wi-korporaciya.ru	О корпорации
5	sitear.ru	пример 1

SELECT * FROM person **ORDER BY** number;

Выберет все записи из таблицы person в порядке возрастания значений поля number.

number	name	last_name	age
1	Anna	Moroz	12
2	Anka	Moroz	15
3	Anna	Cool	16
4	Anko	Second	18
5	Polina	First	13
6	Polianna	Second	18
7	Vanna	Third	9
8	Anno	Wow	10

SELECT * FROM person ORDER BY number DESC;

Выбирает все записи из person, но уже в порядке убывания (т.е. в обратном порядке).

number	name	last_name	age
8	Anno	Wow	10
7	Vanna	Third	9
6	Polianna	Second	18
5	Polina	First	13
4	Anko	Second	18
3	Anna	Cool	16
2	Anka	Moroz	15
1	Anna	Moroz	12

SELECT * FROM person ORDER BY number LIMIT 5;

Выбирает 5 записей из таблицы person, в порядке возрастания.

number	name	last_name	age
1	Anna	Moroz	12
2	Anka	Moroz	15
3	Anna	Cool	16
4	Anko	Second	18
5	Polina	First	13

SELECT * FROM person WHERE name='Anna';

Выбирает все записи из таблицы person, где поле name соответствует значению Anna.

number	name	last_name	age
1	Anna	Moroz	12
3	Anna	Cool	16

SELECT * FROM person WHERE name LIKE 'An%';

Выбирает все записи из таблицы person, в которой значения поля name начинаются с An.

number	name	last_name	age
1	Anna	Moroz	12
2	Anka	Moroz	15
3	Anna	Cool	16
4	Anko	Second	18
8	Anno	Wow	10

SELECT * FROM person WHERE name LIKE '%na' ORDER BY number ;

Выбирает все записи из таблицы person, где name заканчивается на **na**, и упорядочивает записи в порядке возрастания значения number.

number	name	last_name	age
1	Anna	Moroz	12
3	Anna	Cool	16
5	Polina	First	13
6	Polianna	Second	18
7	Vanna	Third	9

SELECT name, last_name FROM person;

Выбирает все значения полей name и last_name из таблицы person.

name	last_name
Anna	Moroz
Anka	Moroz
Anna	Cool
Anko	Second
Polina	First
Polianna	Second
Vanna	Third
Anno	Wow

SELECT DISTINCT site FROM table_name;

Выбирает **уникальные** (DISTINCT) значения поля site из таблицы **table_name**. Например, при 5 значениях поля site: sitear.ru, sitear.ru, sitear.ru, yaveterinar.ru, wi-korporaciya.ru; выведет только 3 уникальные значения: sitear.ru, yaveterinar.ru, wi-korporaciya.ru;

site
sitear.ru
yaveterinar.ru
wi-korporaciya.ru

SELECT * from person **where** age in (12,15,18);

Выведет все записи таблицы person в которых значения поля age будет равно 12 или 15 или 18.

number	name	last_name	age
1	Anna	Moroz	12
2	Anka	Moroz	15
4	Anko	Second	18
6	Polianna	Second	18

select max(age) **from** person;

Выберет максимальное значение age из таблицы person.

max(age)
18

select name, min(age) **from** person;

Выберет минимальное значение age из таблицы person.

min(age)
9