Что такое SQL? Structured Query Language, или язык структурированных запросов, был создан с одной-единственной целью: определять тип данных, предоставлять к ним доступ и обрабатывать их за достаточно короткие промежутки времени.

Самыми распространенными на сегодняшний день являются такие:

* DDL - является командой, которая определяет данные. Она используется для того, чтобы создавать, изменять и удалять самые разнообразные объекты в базе.
* DCL - является командой, которая управляет данными. Ее используют для предоставления доступа разным пользователям к информации в базе, а также чтобы использовать таблицы или представления.
* TCL - команда, которая управляет разнообразного рода транзакциями. Ее главной целью является определение хода транзакции.
* DML - манипулирует полученными данными. В ее задачу входит позволение пользователю перемещать различную информацию из базы данных или вносить туда ее.

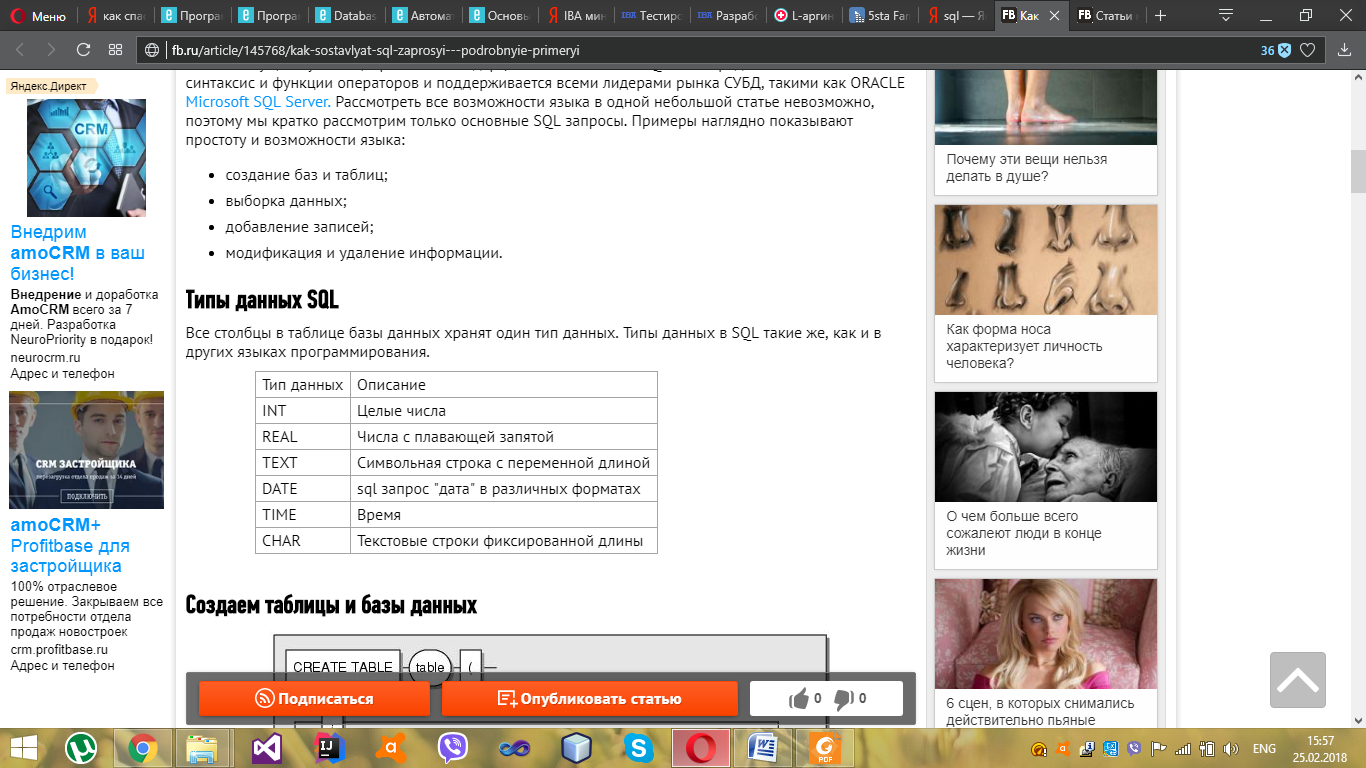
SQL состоит из четырех отдельных частей:

1. язык определения данных (DDL) используется для определения структур данных, хранящихся в базе данных. Операторы DDL позволяют создавать, изменять и удалять отдельные объекты в БД. Допустимые типы объектов зависят от используемой СУБД и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.
2. язык манипуляции данными (DML) используется для извлечения и изменения данных в БД. Операторы DML позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть DML, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер.
3. язык определения доступа к данным (DCL) используется для контроля доступа к данным в БД. Операторы DCL применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов DDL и DML к определенным объектам БД.
4. язык управления транзакциями (TCL) используется для контроля обработки транзакций в БД. Обычно операторы TCL включают commit для подтверждения изменений, сделанных в ходе транзакции, rollback для их отмены и savepoint для разбиения транзакции на несколько меньших частей.

Типы привилегий, которые существуют в этом сервере

Под привилегиями подразумеваются те действия, которые может выполнить тот или иной пользователь в соответствии со своим статусом. Самой минимальной, безусловно, является обычный вход в систему. Конечно же, со временем привилегии могут меняться. Старые будут удаляться, а новые добавляться. На сегодняшний день, все те, кто проходит курсы SQL Server "для чайников", знают, что существует несколько типов разрешенных действий.

1. Объектный тип - пользователю разрешается выполнять какую-либо команду только по отношению к определенному объекту, который находится в базе данных. При этом для разных объектов привилегии отличаются. Они также привязываются не только к тому или иному пользователю, но и к таблицам. Если кто-либо, пользуясь своими возможностями, создал таблицу, то он считается ее владельцем. Поэтому в его праве назначать новые привилегии для других пользователей, связанные с информацией в ней.
2. Системный тип - это так называемые авторские права на данные. Пользователи, которые получили такие привилегии, могут создавать различные объекты в базе



**Точные числовые типы**

Как вы, возможно, поняли из названия, точные числовые типы данных позволяют точно выразить значение числа. К этой категории относятся пять типов:

* INTEGER
* SMALLINT
* BIGINT
* NUMERIC
* DECIMAL

**Тип INTEGER**

В данных типа **INTEGER** (целый) нет дробной части, и их точность зависит от конкретной реализации SQL. Таким образом, точность не может быть установлена разработчиком базы данных.

***Помни:*** *Точностью числа является максимальное количество цифр, которое у него может быть*.

**Тип SMALLINT**

Тип **SMALLINT** (малый целый) также предназначен для целых значений, но его точность в конкретной реализации не может быть больше точности типа INTEGER, имеющейся в данной реализации. В реализациях, работающих на компьютерах IBM System /370, типы SMALLINT и INTEGER обычно представлены соответственно 16 – и 32-битовыми двоичными числами. Впрочем, во многих реализациях SMALLINT и INTEGER являются одним и тем же типом.

Если в таблице из базы данных вы определяете столбец для целых данных и известно, что значения в этом столбце никогда не превысят точность, установленную в вашей реализации для значений типа SMALLINT, то присвойте столбцу тип не INTEGER, a SMALLINT. Таким образом вы, возможно, позволите своей СУБД сэкономить место на диске.

### Тип BIGINT

Тип данных **BIGINT** (большой целый) – это новый тип данных, появившийся вместе с SQL:2003. Он также предназначен для целых значений и определяется как тип, точность которого может быть не намного больше, чем точность данных типа INTEGER, или сильно превышать ее. Предел точности данных типа BIGINT зависит от реализации.

### Тип NUMERIC

В данных типа **NUMERIC** (числовой), кроме целого компонента, может быть и дробный. Для этих данных можно указать точность и масштаб. Точность, как вы помните, – это максимально возможное количество цифр.

При определении типа NUMERIC необходимо указать требуемые значения точности и масштаба. В определении можно указать только NUMERIC и получить значения по умолчанию. А если вы укажете NUMERIC (p), то получите требуемую точность и значение масштаба по умолчанию. Выражение NUMERIC (p,s) позволяет непосредственно задать и точность, и масштаб. При определении данных вместо параметров p и s нужно ввести соответственно требуемые значения точности и масштаба.

### Тип DECIMAL

Тип данных **DECIMAL** (десятичный) похож на NUMERIC. В нем может быть дробная часть, и для него можно указать точность и масштаб. DECIMAL отличается от NUMERIC тем, что если точность имеющейся реализации SQL будет больше указанного значения, то в реализации будет использоваться большая точность. А если точность или масштаб не указаны, то, как и при использовании типа NUMERIC, в реализации применяются значения по умолчанию.

В столбце, тип которого NUMERIC (5.2), нельзя поместить числа, большие 999.99. Если же тип столбца DECIMAL (5.2), в него всегда можно поместить значения до 999.99. Кроме того, если точность реализации позволяет, то СУБД сможет поместить в этот столбец и значения, большие, чем 999.99.

Если в ваших данных имеются дробные части, тогда применяйте тип NUMERIC или DECIMAL, а если ваши данные всегда состоят из целых чисел, то используйте тип INTEGER или SMALLINT. Когда нужно добиться максимальной переносимости, следует использовать тип NUMERIC, потому что поле, тип которого вы определите, например, как NUMERIC (5.2), будет во всех системах иметь один и тот же диапазон значений.

### Тип REAL

Тип данных REAL (действительное число) дает возможность задавать числа однократной точности с плавающей запятой, точность которых зависит от реализации. Вообще-то, точность определяется используемым оборудованием. Например, 64-битовая машина дает большую точность, чем 32-битовая.

**Число с плавающей запятой** (floating – point number) – это число с десятичной запятой Десятичная запятая "плавает" или появляется в разных частях числа, в зависимости от значения этого числа. Примерами чисел с плавающей запятой являются 3.1, 3.14 и 3.14159.

### Тип DOUBLE PRECISION

Тип данных **DOUBLE PRECISION** (двойная точность) дает возможность задавать числа двойной точности с плавающей запятой, точность которых опять-таки зависит от реализации Удивительно, что само значение слова DOUBLE (двойной) также зависит от реализации Арифметика двойной точности в основном применяется в научных целях. А для разных научных дисциплин требуется разная точность. Некоторые реализации SQL обслуживают одну категорию пользователей, а другие реализации – соответственно другие категории.

В некоторых системах тип DOUBLE PRECISION имеет и для мантиссы, и для экспоненты как раз в два раза большую вместимость, чем тип REAL. (Если вы забыли то, что учили в средней школе, то вспомним, что любое число можно представить в виде **мантиссы**, умноженной на число десять, возведенное в степень, показатель которой является **экспонентой**. Например, 6626 можно написать в виде 6.626ЕЗ. Число 6.626 является мантиссой, которую вы умножаете на десять, возведенное в третью степень (в данном случае 3 является экспонентой).

### Тип FLOAT

Тип данных FLOAT (плавающий) является самым полезным, если вы считаете, что ваша база данных однажды должна перейти на аппаратную платформу, в которой размеры регистров отличаются от размеров регистров платформы, для которой вы первоначально спроектировали базу. Используя тип данных FLOAT, можно указывать точность– например, FLOAT (5). Если ваше оборудование поддерживает указанную точность, используя аппаратную одинарную точность, то ваша система использует арифметику с одинарной точностью. Если указанная точность требует арифметики с двойной точностью, то система использует ее.

***Совет:***Использование FLOAT вместо REAL или DOUBLE PRECISION облегчает перенос ваших баз данных на другие системы, так как тип данных FLOAT дает возможность указывать точность. Точность чисел типа REAL или DOUBLE PRECISION зависит от аппаратуры.

### Символьные строки

Есть три главных типа символьных данных: фиксированных символьных данных (CHARACTER или CHAR), переменных символьных данных (CHARACTER VARYING или VARCHAR) и данных для больших символьных объектов (CHARACTER LARGE OBJECT или CLOB). Кроме того, есть еще три варианта этих типов данных: **NATIONAL CHARACTER**(строка с национальными символами), **NATIONAL CHARACTER VARYING** (переменная строка с национальными символами) и **NATIONAL CHARACTER LARGE OBJECT** (большой объект с национальными символами).

### Тип CHARACTER

Если вы определяете для столбца тип данных CHARACTER или CHAR, количество символов, которое будет в нем находиться, можно указать, используя синтаксис CHARACTER (х), где х и является нужным вам количеством. Если, например, тип данных столбца был определен как CHARACTER (16), то максимальная длина любых данных, которые можно будет ввести в этот столбец, равна 16 символам. Если аргумент не указан (т.е. нет значения в скобках), тогда для SQL это означает, что длина поля равна одному символу. Если вы вводите данные в поле типа CHARACTER, имеющего определенную длину, и при этом вводите символов меньше, чем может поместиться в поле, то позиции, оставшиеся свободными, будут заполнены пробелами.

**Тип CHARACTER VARYING**

Тип данных CHARACTER VARYING полезен тогда, когда вводимые в столбец значения имеют разную длину, но вы не хотите, чтобы поле заполнялось пробелами. Этот тип данных Дает возможность сохранять то количество символов, которое ввел пользователь. Для типа CHARACTER VARYING нет значения по умолчанию. Чтобы указать этот тип данных, используйте синтаксис CHARACTER VARYING (х) или VARCHAR (х), где х – это максимальное разрешенное количество символов.

**Тип CHARACTER LARGE OBJECT**

Тип данных CHARACTER LARGE OBJECT (CLOB) впервые появился в SQL: 1999. Как Указывает его имя (означает "большой символьный объект"), он используется вместе с громадными символьными строками, которые для типа CHARACTER слишком велики. Данные – LOB ведут себя во многом так же, как и обычные символьные строки, но на действия, которые можно с ними проводить, имеется ряд ограничений. Для типа CLOB нельзя использовать предикаты PRIMARY KEY (первичный ключ), FOREIGN KEY (внешний ключ), UNIQUE (уникальный). Более того, эти данные нельзя использовать для сравнения, за исключением равенства или неравенства. Из-за того, что у данных CLOB большие размеры, они как правило, всегда остаются на сервере. Вместо них на стороне клиента применяется специальный тип данных, который называется **локатор** LOB (LOB locator). Это параметр, значение которого идентифицирует большой символьный объект.

**Типы NATIONAL CHARACTER, NATIONAL CHARACTER VARYING и NATIONAL CHARACTER LARGE OBJECT**

В разных языках используются символы, которые отличаются от любых символов другого языка. Например, в немецком имеются символы, которых нет в наборе символов английского языка. Имеются языки, такие, например, как китайский или японский, в которых набор иероглифов очень сильно отличается от наборов символов других языков. Если вы в своей системе сделаете английский языком по умолчанию, то все равно сможете использовать и другие символьные наборы, так как типы данных NATIONAL CHARACTER, NATIONAL CHARACTER VARYING и NATIONAL CHARACTER LARGE OBJECT работают точно так же, как и CHARACTER, CHARACTER VARYING и CHARACTER LARGE OBJECT, за исключением того, что определяемый вами набор символов отличается от того, который вы используете по умолчанию. Набор символов можно указывать при определении столбца в таблице. Если нужно, то в каждом столбце можно использовать отдельный набор символов. В следующем примере инструкции, создающей таблицу, используется несколько таких наборов:

CREATE TABLE XLATE

LANGUAGE\_1 CHARACTER (40),

LANGUAGE\_2 CHARACTER VARYING (40) CHARACTER SET GREEK,

LANGUAGE\_3 NATIONAL CHARACTER (40),

LANGUAGE\_4 CHARACTER (40) CHARACTER SET KANJI

);

Столбец LANGUAGE\_1 (1-й язык) предназначен для символов из набора, используемого в данной реализации по умолчанию. В свою очередь, столбец LANGUAGE\_3 (3-й язык) предназначен для символов национального набора данной реализации. А для греческих символов предназначен столбец LANGUAGE\_2 (2-й язык). И, наконец, для иероглифов предназначен столбец LANGUAGE\_4 (4-й язык).

**Логические данные**

Тип данных BOOLEAN (булев) имеет два определенных логических значения, **true**(истина) и **false** (ложь), а также неопределенное значение **unknown** (неизвестное). Если любое из первых двух значений сравнить с NULL или со значением **unknown**, то в результате получится **unknown**.

**Данные типа даты-времени**

В SQL:2003 определяются пять различных типов данных, которые относятся к дате и времени. Они называются **типами данных даты-времени** или просто **датой-временем**. Некоторые из этих типов данных довольно сильно перекрывают друг друга, поэтому в отдельных реализациях, которые могут вам встретиться, все пять типов, возможно, и не поддерживаются.

***Внимание:*** *В тех реализациях, в которых не полностью поддерживаются все пять типов данных для даты и времени, при переносе в них баз данных из других реализаций могут возникнуть проблемы. Если с переносом возникли сложности, то проверьте, какие имеются способы представления даты и времени в двух реализациях исходной и той, на которую требуется перейти*.

### Тип DATE

Тип DATE (дата) предназначен для хранения значений даты в следующем порядке: год, и день. Значение года занимает четыре цифры, а месяца и дня – по две. Значения этого могут представлять любую дату, начиная с 0001 года и заканчивая 9999 годом. Длина DAТЕ равна десяти позициям, как, например, для 1957-08-14.

Технические подробности: Так как язык SQL, используя тип DATE, представляет в явном виде все четыре цифры года, то данные SQL никогда не вызывали опасения из-за пресловутой проблемы 2000 года.

### Тип TIME WITHOUT TIME ZONE

Тип **TIME WITHOUT TIME ZONE** (время без часового пояса) предназначен для хранения значений времени в следующем порядке: час, минута и секунда. Значения часа и минуты занимают в точности по две цифры.

### Тип TIMESTAMP WITHOUT TIME ZONE

### Тип TIME WITH TIME ZONE

### Тип TIMESTAMP WITH TIME ZONE

### Типы коллекций

После того как с выходом SQL: 1999 была разрушена реляционная строгость, стало возможным использование таких типов данных, которые нарушают первую нормальную форму. Теперь поля могут содержать не один, а целую коллекцию объектов. Тип **ARRAY**(массив) впервые появился уже в SQL: 1999, а вот тип **MULTISET** (мультимножество) – это новинка SQL:2003.

Можно сравнить две коллекции между собой только в том случае, если они содержат один и тот же тип данных, либо ARRAY, либо MULTISET, и сравниваемые типы элементов. Поскольку массивы имеют определенный порядок элементов, соответствующие элементы массивов можно сравнить. Мультимножества такого порядка не имеют, однако их также можно сравнить, но только в том случае, если для одного из двух сравниваемых мультимножеств существует нумерация, которой должна соответствовать нумерация другого мультимножества.

**Типы ARRAY**

Тип данных **ARRAY** (массив) нарушает первую нормальную форму (1НФ), но делает это иначе, чем тип ROW. Тип ARRAY, относящийся к типу коллекций, в действительности не является отдельным типом данных в том же смысле, что и типы CHARACTER или NUMERIC. Тип ARRAY просто дает возможность одному из других типов иметь множество значений внутри одного поля в таблице. Скажем, например, что для вашей организации важно поддерживать контакт со своими клиентами, находятся ли они на работе, дома или в дороге. Поэтому среди данных о клиенте может потребоваться и несколько телефонных номеров. Такую возможность можно реализовать, объявив атрибут Phone (телефон) массивом, как показано в следующем коде:

|  |  |  |
| --- | --- | --- |
| CREATE TABLE CUSTOMER ( | | |
| CustID | INTEGER | PRIMARY KEY, |
| LastName | CHARACTER | VARYING (25), |
| FirstName | CHARACTER | VARYING (20), |
| Address | addr\_typ | |
| Phone | CHARACTER | VARYING (15) ARRAY [3] |
| ); | | |

Это дает возможность хранить в таблице **CUSTOMER** (клиент) до трех телефонных номеров каждого клиента. Три телефонных номера являются примером повторяющейся группы. Если следовать классической теории реляционных баз данных, то такие группы делать нельзя, однако они – это только один из нескольких примеров нарушения правил со стороны SQL: 1999. Когда доктор И.Ф. Кодд впервые провозгласил свои правила нормализации, то ради целостности данных он пожертвовал функциональной гибкостью. SQL: 1999 вернул назад часть этой гибкости – впрочем, за счет некоторого усложнения структуры. Такое усложнение может привести к ослаблению целостности данных. Это вам напоминание на тот случай, если вы еще не полностью осознаете последствия всех действий, выполняемых с базой данных. Массивы упорядочены таким образом, что каждый элемент массива связан только с одной порядковой позицией в массиве.

**Тип MULTISET**

Тип данных **MULTISET** (мультимножество) – это неупорядоченная коллекция. Определенные элементы мультимножества могут быть не связаны, поскольку им не назначается определенная порядковая позиция в мультимножестве.