# **Типы Данных**

**smallint = -+ 32768** целое число (2 Байта)

**integer = -+2147483648** целое число (4 Байта)

**bigint = -+9223372036854775808** целое число (8 Байт)

**smallserial = от 1 до 32767** целое число без минусных значений (2 Байт)

**serial = от 1 до 2147483647** целое число без минусных значений (4 Байт)

**bigserial = от 1 до 9223372036854775807** целое число без минусных значений (8 Байт)

**decimal / numeric = 5.3** числа с плавающей точкой с точным подсчетем -> денежные разчеты (Размер Измяется)

**real / float4 =** **5.3** числа с плавающей точкой с неточным подсчетом -> 6 знаков после запятой (4 Байт)

**double precision / float / float8 = 5.3** числа с плавающей точкой с неточным подсчетом -> 15 знаков после запятой (8 Байт)

**char(10) = “text”** строка с ограниченым количеством симолов, если не хватает символов тогда всеравно заполняет место пробелами (Размер Изменяется)

**varchar(10) = “text”** строка с ограниченым количеством символов, если не хватает символов, тогда не заполняет пустое пространство (Размер Изменяется)

**text = “text”** строка, которая подерживате любую длину символов (Размер Изменяется)

**Boolean / bool = True (False)** логический тип (1 Байт) **y , yes , 1 -> True** --- **n , no , 0 -> False**

**date = “1980 AD”** хранит даты (4 Байта)

**time = “12:32:16”** хранить время (8 Байт)

**timestamp = “1980 AD** или **12:32:16”** хранит и дату и время (8 Байт)

**interval = “-+ 13000”** хранит разницу между двумя timestamp (16 Байт)

**timestemptz = “timestamp + timezone”** хранит timestemp + часовой пояс (8 Байт)

**NULL** отсутствие данных (не пустая строка)

# **Основы**

**‘ ’** везде используються одинарные кавички

**\***  означает все

**pk\_name\_id** название для колонки из первичными ключами, нужно писывать pk в начале

**fk\_name\_id** название для колонки из внешними ключами , нужно писать fk в начале

# **CONSTRAINTS**

**CONSTRAINTS my\_name PRIMARY KEY - FOREIGN KEY - CHECK - DEFAULT - UNIQUE - NOT NULL** дает дополнительные названия для ограничений, создает именованые ограничения

**PRIMARY KEY** или **my\_col int PRIMARY KEY** первичный ключ,уникально индетифицирует каждую строку в таблице. Отсчет строк начинается с 1. Нельзя передвавать Null

**FOREIGN KEY**  или **REFERENCES my\_tab2 (my\_col2)** внешний ключ, подключает другие таблици к текущей. Значения в етом столбце могут повторяться. Не нужно прописывать FOREING KEY

**CHECK** или **my\_col int CHECK (my\_col>1)** проверка условия перед записью данних в таблицу

**DEFAULT** или **my\_col int DEFAULT 777** если не указываем данные для записи, то будет записываться указаное значение по умолчанию

**UNIQUE** или **my\_col varchar(5) UNIQUE** можно передавать только уникальные значения

**NOT NULL** или **my\_col varchar(5) NOT NULL** нельзя передавать пустое значение

# **Создание Базы Данных**

**CREATE DATABASE my\_database** создание базы данных

**CREATE DATABASE IF EXSIST my\_database** создание базы данных если уже существует с таким же названием

**CREATE DATABASE IF NOT EXSIST my\_database** создание базы данных если еще не существует с таким же названием

# **Создание Таблици**

**CREATE TABLE my\_table ( my\_name varchar(10) );** создание таблици

**pk\_table\_id integer PRIMARY KEY** указвает первичный ключ начальной колонкой. Значения в етой колонке не повторяються

**fk\_table\_id integer REFERENCES table2 (table2\_id)** добавление внешнего ключа при создании таблици. При создании таблици не нужно прописывать FOREIGN KEY

**CONSTRAINT my\_name PRIMARY KEY (my\_column1 , my\_column2)** отношение многие к многим, в создании третьей таблици

**my\_number INT AUTOINCREMENT** автоинкремент генерирует уникальный номер для каждой строки в таблице

# **Ключи**

**my\_primary INT PRIMARY KEY** создание колонки и назначение первичним ключем (нельзя вводить значения самостоятельно)

**PRIMARY KEY (my\_primary)** делаем уже созданую колонку первичным ключем

**my\_primary INT PRIMARY KEY GENERATED ALWAYS** такае же запись что и сверху, но с явным указанием что нельзя вводить значения самостоятельно

**my\_primary INT PRIMARY KEY GENERATED ALWAYS** **BY DEFAULT** создание первичного ключа, где можна вписивать значения самостоятельно

**my\_foreign INT REFERENCES my\_tab2(my\_primary)** создание колонки и назчение внешним ключем

**FOREIGN KEY (my\_foreign) REFERENCES my\_tab2(my\_primary)** делаем уже созданую колонку внешним ключем

# **Измененние Данных в Строке**

**INSERT INTO my\_tab VALUES ( 1, ”name1” ) , (2 , “name2”);** добавление одной строки в таблицу (добавляет значения толко если поля пустые)

**INSERT INTO my\_tab (my\_col1 , my\_col2) VALUES (1,”name1”),(2,”name2”)**  можно указывать последователньость колонок при заполнении данными

**INSERT INTO my\_tab2 SELECT my\_col1, my\_col2 FROM my\_tab1** заполнение одной таблици с помозью виборки из другой таблици

**UPDATE my\_tab SET my\_col = 5** изменение данных во всех строках указаной колонки(связль колонок в одну строку)

**UPDATE my\_tab SET my\_col1 = 5 WHERE my\_col2 > 3** изменение данных в строках указаной колонки, где условие после WHERE будет True (связь колонок в одну строку)

**UPDATE my\_tab SET my\_col = my\_col + 10** **, my\_col2 = “text” , my\_col3 = NULL WHERE my\_col1 IS NULL** можно изменять по другому

**DELETE FROM my\_tab WHERE my\_col = 3** удаляет строку из таблици, где условие после WHERE будет True

# **Добавление Структуры в Таблицу**

**ALTER TABLE my\_tab ADD COLUMN name\_column INT NOT NULL** добавление колонки из другой таблици в текущую таблицу

**ALTER TABLE my\_table1 ADD CONSTRAINT fk\_new\_name\_id FOREIGN KEY (my\_table1\_my\_column1) REFERENCES my\_table2 (my\_table2\_my\_column2)** указываем добавленую колонку как внешний ключ

# **Простая Выборка**

**SELECT какие колонки FROM из каких таблиц**

**SELECT \* FROM my\_tab** выводит все колонки и все строки из указаной таблици

**SELECT \* FROM my\_tab1 , my\_tab2** выводит все строки и все колонки из указаных таблиц

**SELECT my\_col1 , my\_col2** выводит все строки только из указаных колонок

**SELECT my\_col1, my\_col2 +-\*/ my\_col3** выводим дополнительну дополнительну колонку с результатом матиматической операции с двумя указаными колонками

**SELECT DISTINCT my\_col** выводит указаную колонку только из уникальными значениями

**SELECT DISTINCT my\_col1, my\_col2** выводит первую колонку с уникальными сочетаниями, а вторую колонку с уникальным присоединением к первой

**FROM my\_tab new\_name** даем псевдоним для таблици, новое название

**my\_col AS new\_name** даем псевдоним для таблици, будет выводиться под новым названием (WHERE , HAVING не работает с псевдонимами)

**SELECT my\_tab.my\_col** возвращает колонку из указаной таблици

**SELECT my\_col1 || ‘ ’ || my\_col2** вывод двох колонок как одну

# **Операторы для Выборки**

**WHERE выражение (my\_col > 5)**

**> ( больше ) < ( меньше ) >= ( больше или равно ) <= ( меньше или равно )**

**= ( равно ) != (<>) ( не равно )**

**WHERE my\_col >= 3 , “my\_’name’ , ‘country’** выводит все строки из указаной таблици, где результат виражения вернеться True

**WHERE my\_price > 20**  выводит количество строк, где последнее выражение вернет True

**WHERE my\_date > ‘1996-07-08’** выводит все строки где дата больше указаной

**WHERE выражение1 AND( и ) OR( или ) выражение2** логические операторы и , или

**WHERE my\_col1 > 5 AND ( my\_col2 < 50 OR my\_col2 > 150 )** сортировка, где явно указываем значения которые або больше або меньше. возвращает строку если выражение1 True и одно из выражение2 тоже True

**WHERE my\_col1 BETWEEN 10 AND 20** или **WHERE my\_col1 >= 10 AND my\_col1 <= 20** выводит строки с указаным диапазоном значений. Between всегда включает указаные значения

**WHERE my\_col IN ( value1 , ‘value2’ )** или **WHERE my\_col NOT IN ( value1 , ’value2’ )** выводит все строки, где значения включение или не включены в указаные

↓ **ORDER BY** ↓

**SELECT \* FROM my\_table ORDER BY my\_col значение**

**ORDER BY my\_col** **ASC** соритировка по возростанию (от меньшего к большему)

**ORDER BY my\_col DESC** сортировка по убыванию (от большему к меньшему)

**ORDER BY my\_col1 DESC , my\_col2**  сортировка сразу нескольких колонок, можно использовать только одно из значений DESC - ASC

↓ **LIKE** ↓

**WHERE LIKE ‘ значение ’**

**LIKE ‘a %’** строки, которые начинаються с ‘а’

**LIKE ‘% a’** строки, которые кончаються на ‘a’

**LIKE ‘% a %’** строки, которые содержат ‘a’

**LIKE ‘a % b’** строки, которые начаниються на ‘a’ и заканчуються на ‘b’

**LIKE ‘\_ab\_’** строки, где 2 и 3 символ - ‘ab’. 1 и 4 символы любые. Длина строки ограниченая и указаная

**LIKE ‘\_ab%’** строки , где 1 символ любой , 2 и 3 - ‘ab’ и продолжение из любих симолов любого количества

**LIMIT 10** показывает первые 10 строк. Пишется в конце запроса

**LIMIT 3 , 5** или **LIMIT 5 OFFSET 3** пропускает первые 3 строки и выводит слудующие 5 строк

**WHERE my\_col IS NULL** возвращает все значения колонки, где значение равно NULL

**WHERE my\_col IS NOT NULL** возвращает все значения колонки, где значение не равно NULL

↓ **GROUP BY** ↓

**SELECT my\_col1, my\_col2 FROM my\_tab WHERE my\_col3>10**

**GROUP BY my\_col1**

**ORDER BY my\_col2 DESC**

**SELECT my\_col1, COUNT (\*) WHERE my\_col3 > 10 GROUP BY my\_col1** Групировака всех строк только по первому указаному столбцу my\_col1.

**HAVING my\_col > 5** выводит значения которые возвращают True, применяеться после сортировки с WHERE. Ограничивает отобращение отсортированых колонок

**SELECT my\_col1 , SUM(my\_col2) WHERE my\_col3 > 5 GROUP BY my\_col1 HAVING > 100** применение having

# **Функции**

**FUNCTION ( передача аргументов )**

**SELECT COUNT( \* )** или **SELECT COUNT( my\_col )** возвращает количество строк в таблице или указаном столбце

**SELECT COUNT( DISTINCT my\_col )** возвращает количество уникальных строк в указаной колонке

**SELECT MIN ( my\_col )** возвращает минимальное знеачение из указаной колонки

**SELECT MAX ( my\_col )** возвращает макисамальное значение из указаной колонки

**SELECT AVG ( my\_col )** или **SELECT AVG (my\_col1 - my\_col2)** возвращает среднее арифметическое из всех значений указаной колонки

**SELECT SUM ( my\_col )** возвращает суму всех значений из указаной колонки

**SELECT CONCAT (my\_col1 , ‘ ’ , my\_col2) AS new\_col** обьединение двох колонок в одну

# **Выборка сразу с несколькими таблицами**

**SELECT my\_col1 FROM my\_tab1 UNION SELECT my\_col2 FROM my\_tab2**

**UNION** выводит только уникальные значения из двох таблиц без дубликатов

**UNION ALL** выводит все значения (могут повторяться) из двох таблицс дуликатами

**INTERSECT**  пересичения, только те значения которые есть и в первом и во втором - без дубликатов. **UNTERSECT ALL** с дубликатами

**EXCEPT** только те значения, которые есть в и первом и нету во втором без дубликатов.

**EXCEPT ALL** с дубликатами

# **Соединения**

↓ **Основные Соединения** ↓

**SELECT** название колонок **FROM** название таблици **INNER JOIN** (название соединения) **ON** уcловие обьединения **( my\_id1 = my\_id2 )** соединяем первичный ключ из левой таблици из внешний ключем ил правой

**SELECT FROM** my\_tab1 **INNER JOIN** my\_tab2 **ON** my\_tab1. my\_col1 = my\_tab2 = my\_col2 **(условие)** внутрешнее соединение.

Независимо от какой выборки , записи которые попадают в виборку определяет **ON** и совпадние значений по связаным двум колонком обеих таблиц



**INNER JOIN** или **JOIN** ето одно и тоже , нужно писать просто JOIN

**FOREIGN KEY(my.tab1.mycol2) REFERENCES my\_tab1(my\_col1)** SELF JOIN, внешний ключ в таблице ссылаеться на первичный ключ в етой же таблице

**JOIN my\_tab USING (my\_col1)** укороченая запись, если таблици соединяються через колонку с одним названием

**NATURAL JOIN my\_tab** тоже самое что и INNER JOIN, но он сам находит внешние ключи на другие таблици и заполняет вывод (не рекомендуеться использовать)

# **Удаление**

**DROP DATABASE my\_database** удаление базы данных

**DROP TABLE my\_table** удаление таблици

**DROP TABLE IF EXISTS** **my\_table** удаление указаной таблици если существует

# **Другое**

**-- text** коментраии к коду

**Everyman‘’s** позволяет вписывать спецальные символы в редактор