

# Создание структуры таблиц в СУБД PostgreSQL

Управление данными

Кафедра АПУ СПбГЭТУ «ЛЭТИ»

```
CREATE TABLE имя_таблицы  
( имя_поля тип_данных [ограничения_целостности],  
  имя_поля тип_данных [ограничения_целостности],  
  .....  
  имя_поля тип_данных [ограничения_целостности],  
  [ограничения_целостности],  
  [первичный_ключ],  
  [внешний_ключ]  
);
```

### *Замечания:*

- 1). Регистр символов значения не имеет. Однако традиционно для наглядности ключевые слова вводятся в верхнем регистре.
- 2). В квадратных скобках указаны необязательные элементы.
- 3). Символ ; в конце команды обязателен.

## 1. Числовые типы данных.

*Целочисленные* – smallint, integer, bigint.

Псевдонимы: int2, int4, int8. (Число байтов отражается в имени типа).

*Числа фиксированной точности* – numeric (decimal).

Выбирается для хранения денежных сумм с гарантированной точностью вычислений.

Пример: numeric(точность, масштаб)

numeric(6,2) – общее число цифр – 6, из них 2 после десятичной точки.

numeric(6) – целое число (масштаб = 0).

*Типы с плавающей точкой* – real, double precision.

real – от 1E-37 до 1E+37 с точностью не меньше 6 десятичных цифр.

double precision – от 1E-307 до 1E+308 с точностью не меньше 15 десятичных цифр.

*Последовательные типы* – serial, bigserial, smallserial.

Уникальные целые значения, например, значения суррогатного первичного ключа.

bigserial и smallserial соответствуют типам bigint и smallint.

## 2. Символьные типы данных.

Типы *character varying(n)* и *character(n)*, где n – максимальное число символов в строке.

Псевдонимы: *varchar(n)* и *char(n)*.

Если сохраняемая строка символов короче, чем указано (n), то значение типа *char* будет дополнено пробелами, а значение типа *varchar* будет сохранено так, как есть.

Тип *text* – сколь угодно большое символьное значение.

*Замечание:*

Значения символьных типов данных вводятся в апострофах : 'Москва'

## 3. Тип данных даты/времени.

*Тип date* – даты в формате уууу-мм-дд (год-месяц-день).

Используется по умолчанию.

*Тип time* и *time with time zone* – время суток и время с учетом часового пояса.

*Объединенный тип timestamp* и *timestamp with time zone* – объединение даты и времени (временная отметка).

*Замечание:* Значения дат и времени вводятся в апострофах (как символьные типы):  
'2020-12-30', '12:00', '2020-12-30 12:00'

## 4. Логический тип данных.

*Тип Boolean* – трехзначная логика: истина (TRUE), ложь (FALSE),  
неопределенное состояние (NULL).

В качестве истинного состояния могут быть: TRUE, 't', 'true', 'y', 'yes', 'on', '1'.

В качестве ложного состояния могут быть: FALSE, 'f', 'false', 'n', 'no', 'off', '0'.

# Основные ограничения целостности

Ограничения целостности могут быть заданы на уровне поля или на уровне таблицы.

CREATE TABLE имя\_таблицы

( имя\_поля тип\_данных [ограничения\_целостности],

имя\_поля тип\_данных [ограничения\_целостности],

.....

имя\_поля тип\_данных [ограничения\_целостности],

[ограничения\_целостности],

[первичный\_ключ],

[внешний\_ключ]

);



Ограничения уровня поля:

*DEFAULT значение* – значение по умолчанию.

*CHECK (условие)* – условие на вводимые данные.

*NOT NULL* – обязательность значения.

*UNIQUE* – уникальность значения.

*PRIMARY KEY* – первичный ключ.

*REFERENCES имя\_таблицы (имя\_поля)* – задание ссылочной целостности (упрощенное задание внешнего ключа).

*Пример:*

```
CREATE TABLE student_group  
( group_code serial PRIMARY KEY,      -- Первичный ключ  
  group_number varchar(4)  
);
```

```
CREATE TABLE student  
( id serial PRIMARY KEY,              -- Первичный ключ  
  name varchar(20) NOT NULL UNIQUE,  
  mark numeric(1) CHECK (mark >= 3 AND mark <=5) DEFAULT 3,  
  code integer REFERENCES student_group (group_code),      -- Внешний ключ  
);
```

*Замечание:*

При установке ограничений UNIQUE и PRIMARY KEY автоматически создается индекс на основе В-дерева для поддержки этого ограничения.

Ограничения уровня таблицы:

- Ограничение уровня таблицы начинается с ключевого слова CONSTRAINT.
- Каждое ограничение имеет имя для отражения сути налагаемого ограничения.
- При нарушении ограничения выводится имя нарушенного ограничения.

CONSTRAINT [имя\_ограничения] ограничение,  
где ограничение – CHECK, UNIQUE

*Пример:*

```
CREATE TABLE student
( id serial PRIMARY KEY,
  name varchar(20) NOT NULL,
  mark numeric(1) DEFAULT 3,
  code integer REFERENCES student_group (group_code),
  CONSTRAINT unique_name UNIQUE (name),
  CONSTRAINT valid_mark CHECK (mark >= 3 AND mark <=5)
);
```



Ограничение NOT NULL напрямую через CONSTRAINT не задается. Для этого надо использовать CHECK:

```
CONSTRAINT not_null_name CHECK (name IS NOT NULL)
```

Ограничения на уровне таблицы можно задавать без ключевого слова CONSTRAINT. Тогда имя ограничения будет формироваться автоматически.

*Пример:*

```
UNIQUE (name),  
CHECK (mark >= 3 AND mark <=5),
```

*Замечание:*

Ограничения, в которых используются 2 и более полей, можно формировать только на уровне таблицы.

Задание маски ввода для значений символьных полей:

CONSTRAINT имя\_ограничения CHECK (имя\_поля SIMILAR TO '*шаблон*'),

где '*шаблон*' – регулярное выражение.

*Примеры:*

'[0-9][0-9][0-9]' – 3 цифры

'[A-Z][0-9][0-9]' – буква и 2 цифры

'\[A-Z\][0-9][0-9]' – буква в скобках и 2 цифры

'\[A-Z\][0-9]\-[0-9]' – буква в скобках и 2 цифры через дефис

*Замечание:*

Обычно регулярные выражения используются для запросов-выборок (для извлечения записей по некоторому шаблону).

## Основные ограничения целостности

Задание первичного и внешнего ключа на уровне таблицы:

*PRIMARY KEY (имя\_поля (или полей через запятую)),*

*FOREIGN KEY (имя\_поля (или полей через запятую))*

*REFERENCES имя\_таблицы (имя\_поля (или полей через запятую)),*

*Пример:*

```
CREATE TABLE student
```

```
( id serial,
```

```
  name varchar(20) NOT NULL UNIQUE,
```

```
  mark numeric(1) CHECK (mark >= 3 AND mark <=5) DEFAULT 3,
```

```
  code integer,
```

```
  PRIMARY KEY (id),
```

```
  FOREIGN KEY (code) REFERENCES student_group (group_code)
```

```
);
```

*Замечание:*

Задание первичного и внешнего ключа выполняется только на уровне таблицы, если они состоят из 2 и более полей.

Каскадное удаление и каскадное обновление при определении внешнего ключа:

*ON DELETE CASCADE* – задание каскадного удаления.

*ON DELETE RESTRICT* (*ON DELETE NO ACTION*) – запрет удаления, если есть хоть одна строка, ссылающаяся на удаляемую строку (применяется по умолчанию).

*ON DELETE SET NULL* – присваивание атрибутам внешнего ключа значения NULL.

*ON DELETE SET DEFAULT* – присваивание атрибутам внешнего ключа значения по умолчанию.

*ON UPDATE CASCADE* – задание каскадного обновления.

*ON UPDATE RESTRICT* – аналогично.

*ON UPDATE SET NULL* – аналогично.

*ON UPDATE SET DEFAULT* – аналогично.

*Пример:*

```
FOREIGN KEY (code) REFERENCES student_group (group_code)
                                ON UPDATE CASCADE
                                ON DELETE RESTRICT
```

## Удаление таблиц

---

DROP TABLE имя\_таблицы; – удаление таблицы, если нет зависимых объектов.

DROP TABLE имя\_таблицы CASCADE; – удаление таблицы вместе с зависимыми объектами.

DROP TABLE IF EXISTS имя\_таблицы CASCADE; – удаление таблицы, если она существует.

*Пример:*

DROP TABLE student\_group CASCADE;