**Генерація великої кількості даних у таблицях**

<https://oracleplsql.ru/random-postgresql.html>

В Таблиці 1 перераховані всі функції генерації випадкових чисел.

Таблиця 1. Функції генерації випадкових чисел

| **Функція** | **Тип результату** | **Опис** |
| --- | --- | --- |
| random() | dp | випадкове число в діапазоні 0.0 <= x < 1.0 |
| setseed(dp) | void | задає відправну точку для наступних викликів random() (значення між -1.0 та 1.0, включаючи границі) |

Функція random() повертає випадкове число в діапазоні від =>0 до <1. Вона використовує простий лінійний конгруентний алгоритм, працює швидко, але не підходить для криптографічних додатків; більш безпечна альтернатива є в модулі [pgcrypto](https://postgrespro.ru/docs/postgresql/12/pgcrypto). Якщо скористатися функцією setseed() и викликати її з одним і тим же аргументом, результати наступних викликів random() в поточному сеансі будуть повторюватися.

**Випадкове десятинне число**

Випадкове число в діапазоні від =>0 до <1

**SELECT** random()

Випадкове число в діапазоні від =>0 до <100

**SELECT** random()\*1000 або **SELECT** 1000\*random()

Випадкове число в діапазоні від =>10 до <25

**SELECT** random()\*(25-10)+10;

**Випадкове ціле число**

Стандартно ціле випадкове число отримують з функцією floor(). Вона повертає найближче ціле число менше або рівне заданому числу. Тип повертаємого значення - double precision, тому іноді його треба перетворювати до іншого типу, наприклад до int

Синтаксис функції floor():

floor (number or decimal)

Приклади:

Випадкове ціле число в діапазоні від =>1 до <=100

**SELECT** floor(random()\*1000+1)

**SELECT** floor(random()\*1000+1)::int

Випадкове ціле число в діапазоні від =>10 до <=25

**SELECT** floor(random()\*(25-10+1))+10;

**SELECT** (floor(random()\*(25-10+1)+10)::int;

*Примітка*. Ціле випадкове число можна отримати і з функцією trunk(). Вона повертає число, усічене до визначеної кількості десяткових знаків. Якщо кількості десяткових знаків не задана, то повертається ціле значення числа. Тип повертаємого значення - double precision, тому іноді його треба перетворювати до іншого типу, наприклад до int

Синтаксис функції trunc():

trunc( number, [ decimal\_places ] )

Приклади:

Випадкове ціле число в діапазоні від =>1 до <=100

**SELECT** trunc(random()\*1000+1)

**SELECT** trunc(random()\*1000+1)::int

Випадкове ціле число в діапазоні від =>10 до <=25

**SELECT** trunc(random()\*(25-10+1)+10);

**SELECT** trunc(random()\*(25-10+1)+10)::int;

**Випадкова позначка дата/час**

Випадкова позначка дата/час ([datetime/timestamp](https://stackoverflow.com/questions/22964272/postgresql-get-a-random-datetime-timestamp-between-two-datetime-timestamp)) між двома позначками дата/час

**SELECT** timestamp '2014-01-10 20:00:00' +

random() \* (timestamp '2014-01-20 20:00:00' –

timestamp '2014-01-10 10:00:00')

**Функції, які повертають множину значень**

<https://postgrespro.ru/docs/postgresql/9.5/functions-srf>

У цьому розділі описуються додаткові опції, які можуть повертати не одну, а безліч рядків. Найчастіше з їх числа використовуються функції, які генерують ряди значень, які перераховані в Таблиці 9.55 і Таблиці 9.56. Інші, більш спеціалізовані функції множин описані в інших розділах документації.

Таблиця 2 - Функції, що генерують ряди значень

| **Функция** | **Тип аргумента** | **Тип результата** | **Описание** |
| --- | --- | --- | --- |
| *generate\_series(start, stop)* | int, bigint або numeric | setof int, setof bigint или setof numeric (визначається типом аргументу) | Видає ряд цілих чисел від *start* до *stop* з кроком 1 |
| *generate\_series(start, stop, step)* | int, bigint або numeric | setof int, setof bigint или setof numeric (определяется типом аргумента) | Видає ряд цілих чисел від *start* до *stop* с заданим кроком (*step*) |
| *generate\_series(start, stop, step interval)* | timestamp або timestamp with time zone | setof timestamp или setof timestamp with time zone (определяется типом аргумента) | Видає ряд цілих чисел від *start* до *stop* с заданим кроком (*step*) |

Якщо заданий крок (step) позитивний, а start виявляється більше stop, ці функції повертають 0 рядків. Той же результат буде, якщо step менше 0, а start менше stop, або якщо будь-якої аргумент дорівнює NULL. Якщо ж step буде дорівнює 0, відбудеться помилка.

**Генерація серії значень**

Генерація серії чисел в заданому діапазоні

**SELECT** \* **FROM** generate\_series(2, 4);

Генерація серії чисел в заданому діапазоні з заданим кроком

**SELECT** \* **FROM** generate\_series(5, 1, -2);

Генерація серії дат в заданому діапазоні з заданим кроком в 1 день. Тип даних date

**SELECT** gs::date **FROM** generate\_series('2018-01-31', '2018-05-31', interval '1 day') **AS** gs;

Генерація серії дат і часу в заданому діапазоні з заданим кроком в 1 хвилину. Тип даних timestamp

**SELECT** \* **FROM** generate\_series('2020-01-31 00:00:00', '2020-08-31 23:59:59', interval '1 minute') **AS** gs

Порівняти, вказано тип даних timestamp (без урахування часового поясу)

**SELECT** gs::timestamp **FROM** generate\_series('2020-01-31 00:00:00', '2020-08-31 23:59:59', interval '1 minute') **AS** gs

**Генерація серії псевдовипадкових чисел**

Генерація серії зі 100 псевдовипадкових цілих чисел

**SELECT** floor(random()\*1000+1)::int

**FROM** generate\_series(1, 100)

або

**SELECT** trunc(random()\*1000+1)::int

**FROM** generate\_series(1, 100)

Генерація серії псевдовипадкових цілих чисел в діапазоні 1-100 з кроком 5

**SELECT** floor(random()\*1000+1)::int

**FROM** generate\_series(1, 100, 5)

або

**SELECT** trunc(random()\*1000+1)::int

**FROM** generate\_series(1, 100, 5)

**Генерація псевдовипадкових рядків**

Генерація 5-ти псевдовипадкових рядків з 2-х символів

**SELECT** chr(floor(65 + random()\*25)::int) ||   
chr(floor(65 + random()\*25)::int)

**FROM** generate\_series(1, 5)

**SELECT** chr(trunc(65 + random()\*25)::int) ||   
chr(trunc (65 + random()\*25) ::int)

**FROM** generate\_series(1, 5)

**Функції, що генерують індекси масивів**

**Таблиця 2. Функції, що генерують індекси масивів**

| **Функция** | **Тип результата** | **Описание** |
| --- | --- | --- |
| generate\_subscripts(*array anyarray*, *dim int*) | setof int | Видає ряд значень для використання в якості індексу даного масиву. |
| generate\_subscripts(*array anyarray*, *dim int*, *reverse boolean*) | setof int | Видає ряд значень для використання в якості індексу даного масиву. Якщо параметр reverse дорівнює true, значення видаються від більшого до меншого. |

Функція generate\_subscripts дозволяє спростити отримання всього набору індексів для зазначеної розмірності заданого масиву. Вона видає 0 рядків, якщо в масиві немає зазначеної розмірності або сам масив дорівнює NULL (хоча для елементів, рівних NULL, індекси будуть видані, як і для будь-яких інших).