Незнакомый знакомый Guid

Алексей Романовский

19 апреля 2017

t412q3bf-b36o-ba84-35c0-3c5fv6dmc4d8

f412b3bf-b366-ba84-35c0-3c5fc6ddc4d8

f412b3bf-b366-4a84-a5c0-3c5fc6ddc4d8
f5e26820-b7b8-11e6-9598-0800200c9a66

f412b3bf-b366-4a84-a5c0-3c5fc6ddc4d8
f5e26820-b7b8-11e6-9598-0800200c9a66

f412b3bf-b366-4a84-a5c0-3c5fc6ddc4d8
f5e26820-b7b8-11e6-9598-0800200c9a66

RFC 4122

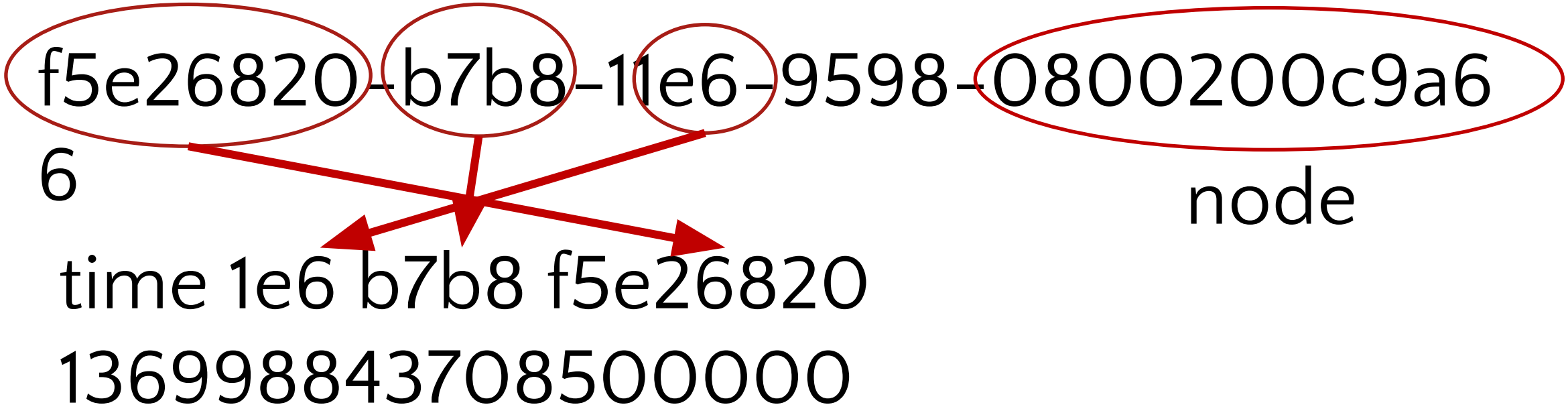
RFC 4122

- UUID – Universally Unique Identifier
- aka GUID – Globally Unique Identifier
- Нет единого центра генерации
- Уникален сквозь время и пространство
- Строковое представление в виде hex digits

Структура

- 128 bits / 16 bytes / 36 chars
  - time_low – 4 байта
  - time_mid – 2 байта
  - time_high_and_version – 2 байта
  - clock_seq_and_reserved – 1 байт
  - clock_seq_low – 1 байт
  - node – 6 байт
- f5e26820-b7b8-11e6-9598-0800200c9a66

Структура


f5e26820-b7b8-11e6-9598-0800200c9a66
node
time 1e6 b7b8 f5e26820
136998843708500000

Равенство и порядок UUID-ов

- UUID можно представить двумя числами unsigned integer – mostSignificantBits и leastSignificantBits
- Последовательно сравниваем соответствующие числа для двух UUID-ов
- Если все поля равны, то UUID-ы равны

- *можно сравнивать как 128-bit unsigned integers

Variants

- xxxxxxxx-xxxx-Mxxx-**N**xxx-xxxxxxxxxxxx
- Три значащих бита N
  - 0: 0xx (N=0…7) – для обратной совместимости с Apollo Network Computing System UUID format
  - 1: 10x (N=8…b) – RFC 4122
  - 2: 110 (N=c…d) – reserved, Microsoft Corporation backward compatibility
  - 3: 111 (N=e…f) – reserved for future definition

Versions

- xxxxxxxx-xxxx-**M**xxx-Nxxx-xxxxxxxxxxxx
- Четыре значащих бита M

Version	Msb0	Msb1	Msb2	Msb3	
1	0	0	0	1	The time-based version
2	0	0	1	0	DCE Security version, with embedded POSIX UIDs
3	0	0	1	1	The name-based version that uses MD5 hashing
4	0	1	0	0	The randomly or pseudo-randomly generated version
5	0	1	0	1	The name-based version that uses SHA-1 hashing

Version 1

- Timestamp – 60 бит
  - Количество 100 нс интервалов с полуночи 15 октября 1582 UTC
- Clock Sequence
  - Помогает избежать повторений
- Node
  - MAC-адрес или случайно сгенерированное значение

Version 4

- Timestamp – 60 бит
  - Случайно сгенерированное значение
- Clock Sequence
  - Случайно сгенерированное значение
- Node
  - Случайно сгенерированное значение

Реализации UUID в .NET и JAVA

Guid в .NET

- Состоит из полей:

```
int a
short b
short c
byte d
byte e
byte f
byte g
byte h
byte i
byte j
byte k
```

Equals

```
public bool Equals(Guid g)
{
    if (g._a != _a)
        return false;
    if (g._b != _b)
        return false;
    if (g._c != _c)
        return false;
    if (g._d != _d)
        return false;
    …
    if (g._k != _k)
        return false;

    return true;
}
```

CompareTo

```
int GetResult(uint me, uint them) {
    if (me < them) {
        return -1;
    }
    return 1;
}

int CompareTo(Guid value) {
    if (value._a != this._a) {
        return GetResult((uint)this._a, (uint)value._a);
    }
    …
    if (value._k != this._k) {
        return GetResult((uint)this._k, (uint)value._k);
    }
    return 0;
}
```

ToByteArray

```
public byte[] ToByteArray() {
    byte[] g = new byte[16];
    g[0] = (byte)(_a);
    g[1] = (byte)(_a >> 8);
    g[2] = (byte)(_a >> 16);
    g[3] = (byte)(_a >> 24);
    g[4] = (byte)(_b);
    g[5] = (byte)(_b >> 8);
    g[6] = (byte)(_c);
    g[7] = (byte)(_c >> 8);
    g[8] = _d;
    g[9] = _e;
    g[10] = _f;
    g[11] = _g;
    g[12] = _h;
    g[13] = _i;
    g[14] = _j;
    g[15] = _k;
    return g;
}
```

UUID в JAVA

- Состоит из полей:

```
long mostSigBits
long leastSigBits
```

Equals

```
boolean equals(UUID id)
{
    return this.mostSigBits == id.mostSigBits
        && this.leastSigBits == id.leastSigBits;
}
```

CompareTo

```
int compareTo(UUID val)
{
    return (this.mostSigBits < val.mostSigBits ? -1 :
            (this.mostSigBits > val.mostSigBits ? 1 :
             (this.leastSigBits < val.leastSigBits ? -1 :
              (this.leastSigBits > val.leastSigBits ? 1 :
               0))));
}
```

ToByteArray

```
byte[] getGuidAsByteArray(UUID uuid) throws IOException {
   ByteArrayOutputStream ba = new ByteArrayOutputStream(16);
   DataOutputStream da = new DataOutputStream(ba);
   da.writeLong(uuid.getMostSignificantBits());
   da.writeLong(uuid.getLeastSignificantBits());

   return ba.toByteArray();
 }
```

Эксперимент

- JAVA
  - `8eacf48c-6750-401c-bea2-f89a8efc5bc1`
  - `jqz0jGdQQBy+oviajvxbwQ==`

- .NET
  - `8cf4ac8e-5067-1c40-bea2-f89a8efc5bc1`
  - `jPSsjlBnHEC+oviajvxbwQ==`

Что случилось

- Порядок байтов
- .NET
  - для первых трех частей: от младшего к старшему (little-endian)
  - для остальных: от старшего к младшему (big-endian)
- JAVA
  - от старшего к младшему (big-endian)

Как починить

```
var rfc4122bytes = Convert.FromBase64String("jqz0jGdQQBy+oviajvxbwQ==");

Array.Reverse(rfc4122bytes, 0, 4);
Array.Reverse(rfc4122bytes, 4, 2);
Array.Reverse(rfc4122bytes, 6, 2);

var guid = new Guid(rfc4122bytes);
```

Мораль

- RFC 4122 регламентирует строковое представление
- Передавать UUID между системами можно только в строковом представлении

Сравнение UUID-ов в разных БД

MS SQL

- NEWID() – UUID 4
- Байты сравниваются в другом порядке:
- 10, 11, 12, 13, 14, 15, 8, 9, 6, 7, 4, 5, 0, 1, 2, 3

```
3aaaaaaa-bbbb-cccc-dddd-2eeeeeeeeeee
2aaaaaaa-bbbb-cccc-dddd-1eeeeeeeeeee
1aaaaaaa-bbbb-cccc-dddd-3eeeeeeeeeee

2aaaaaaa-bbbb-cccc-dddd-1eeeeeeeeeee
3aaaaaaa-bbbb-cccc-dddd-2eeeeeeeeeee
1aaaaaaa-bbbb-cccc-dddd-3eeeeeeeeeee
```

MySQL

- UUID() – UUID 1
- varchar(36), то есть можно и UUID 4
- Сравнение лексикографическое

- Что если надо сравнивать, как UUID-ы?
  - например, вместо UUID-а сохранять hex-строку или blob с «правильным» порядком байтов, чтобы можно было сравнить лексикографически

```
65b2c6d8-29d5-4b0f-8aed-74dda3e8d27c

4b0f29d565b2c6d88aed74dda3e8d27c
```

Cassandra

- UUIDType – UUID x
- TimeUUIDType – UUID 1
  - Кейс: time serias – последовательная запись событий
  - Сравнение:
    - Сначала сравниваются timestamp-ы
    - Затем побайтовое сравнение хвоста

Как еще можно использовать UUID

Пространство для маневра

- 60 bit (timestamp)
- 1 byte (clock seq) – 256 значений
- 6 byte (node)

Идеи

- Все зависит от фантазии ☺️
- Важно соблюсти формальные требования

Выводы

- Для передачи UUID-а нужно использовать строковое представление
- Для консистентной работы надо реализовывать соответствующие БД алгоритмы сравнения
- Внутри вашего продукта можно отходить от стандарта при работе с UUID

Вопросы?

http://bit.ly/dotnet_feedback

Алексей Романовский

Контур.Диадок

logicman@skbkontur.ru

kontur.ru