Cuidado: Laravel 5.8 Adicionado bigIncrements como padrão

20 de março de 2019

Há uma alteração no Laravel 5.8 que não é mencionada no [Guia de atualização](https://laravel.com/docs/5.8/upgrade) oficial, mas me causou problemas - não consegui criar uma migração de chave estrangeira e passei meia hora até descobrir os motivos. Então, eu quero compartilhar, talvez você encontre a mesma coisa.

Exemplo e erro da minha situação

Enquanto trabalhava em um projeto, criei uma tabela simples **transaction\_types** :

php artisan make:migration create\_transaction\_types\_table

E adicionou apenas um novo campo **string ()** :

Schema::create('transactions\_types', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('name');

$table->timestamps();

});

E, em seguida, tentei criar uma tabela de migração para **transações** com uma chave estrangeira, com a mesma sintaxe que costumava fazer o tempo todo:

php artisan make:migration create\_transactions\_table

Conteúdo do arquivo de migração:

Schema::create('transactions', function (Blueprint $table) {

$table->bigIncrements('id');

$table->integer('transaction\_type\_id')->unsigned();

$table->foreign('transaction\_type\_id')->references('id')->on('transactions\_types');

$table->decimal('amount');

$table->timestamps();

});

E aqui está o que aconteceu ao executar o **php artisan migrate** :



Como de costume, o erro *"não é possível adicionar restrição de chave estrangeira"* não fornece muitos detalhes, então eu estava imaginando o que poderia dar errado:

* Erro de digitação no nome da tabela? Verificado duas vezes, não.
* Nome da chave estrangeira muito longo? (deve ter até 64 caracteres) Não parece.
* Incompatibilidade nos tipos de campo de chave estrangeira? Hmmm, então eu notei algo ...

Veja mais de perto o que o Laravel está gerando ao executar o comando **php artisan make: migration** :

Schema::create('transactions\_types', function (Blueprint $table) {

$table->bigIncrements('id');

// ...

bigIncrements. Incrementos GRANDES. E então, é claro, se você estiver fazendo uma chave estrangeira apenas do tipo "número inteiro não assinado" ( *como sempre fiz toda a minha vida* ), ela falhará. Digite incompatibilidade. Obviamente.

Como consertar

Bem, é fácil. Existem duas maneiras de fazê-lo funcionar:

* Altere a migração original de **bigIncrements()** para apenas **Increments ()**;
* Ou na coluna de chave estrangeira, faça **bigInteger() em** vez de **Integer ()**.

A escolha é sua.

PS Ah, a propósito, a migração de tabela de **usuários** padrão também vem com **bigIncrements ()** :

Schema::create('users', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('name');

$table->string('email')->unique();

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

Portanto, sempre que você criar novos campos com chave estrangeira para os usuários, não se esqueça de usar **bigInteger ()** .

Obs1: **Diferença entre número inteiro e bigint:**

**INT** [UNSIGNED] [ZEROFILL]

* **INT**: Um número inteiro de tamanho normal.
* **UNSIGNED** (NÃO ASSINADO): Esse atributo é usado para permitir somente valores não negativos em uma coluna, gerando uma faixa numérica maior de números positivos nessa coluna.

O intervalo **INT** normal é de -2147483648 a 2147483647.

O intervalo **INT** **UNSIGNED** (não assinado) é de **0 a 4294967295**.

* **ZEROFILL**: – O atributo opcional ZEROFILL, preenche espaços em uma coluna, substituindo por zeros. Por exemplo, uma coluna declarada como INT(4) ZEROFILL, o valor 5 é recuperado como 0005.

Obs.: Se você especificar para uma coluna numérica ZEROFILL, automaticamente o MySQL adiciona o atributo UNSIGNED a coluna.

**BIGINT** [UNSIGNED] [ZEROFILL]

* **BIGINT**: Um número inteiro grande.

O intervalo **BIGINT** normalé de -9223372036854775808 a 9223372036854775807.

O intervalo **UNSIGNED** (não assinado) é de **0 a 18446744073709551615**.

Obs2: **BigIncrements e Increments no Laravel:**

* Tanto um quanto o outro são colunas **‘auto-incrementáveis’** do tipo Integer ou BigInteger.
* Normalmente são usados para a criação da **chave primária** das migrations do Laravel.