

MEET MIGRATION

- Allows you to **create/update/delete** a table in your database.
- Modify and share the application's **database schema**.
- It's make easier to maintain **relationship & foreign key** constraint



NAMING CONVENTION

- Create a new table

```
php artisan make:migration create_tableName
```

- Add a new column

```
php artisan make:migration add_columnName_to_tablename
```

- Update a column like rename column

```
php artisan make:migration update_columnName_to_tablename
```

- Remove a column

```
php artisan make:migration remove_columnName_from_tablename
```

- Drop a table

```
php artisan make:migration drop_tableName
```

MIGRATION STRUCTURE

- Up method is used to add new tables, columns, or indexes to your database
- Down method should reverse the operations performed by the up method.

2023_05_02_062343_create_profile_table.php

```
7     return new class extends Migration
8     {
9         public function up(): void
10        {
11            Schema::create('profile', function (Blueprint $table) {
12                $table->id();
13                $table->string('name');
14                $table->string('city');
15                $table->string('phone');
16                $table->timestamps();
17            });
18        }
19        public function down(): void
20        {
21            Schema::dropIfExists('profile');
22        }
23    };
```

AVAILABLE COLUMN TYPES

bigIncrements()	The bigIncrements method creates an auto-incrementing UNSIGNED BIGINT (primary key) equivalent	\$table->bigIncrements('id');
bigInteger()	The bigInteger method creates a BIGINT equivalent	\$table->bigInteger('votes');
binary()	The binary method creates a BLOB equivalent	\$table->binary('photo');
boolean()	boolean method creates a BOOLEAN equivalent column	\$table->boolean('confirmed');
char()	The char method creates a CHAR equivalent column with of a given length	\$table->char('name', 100);
dateTime()	The dateTime method creates a DATETIME equivalent column with an optional precision (total digits)	\$table->dateTime('created_at', \$precision = 0);
date()	The date method creates a DATE equivalent	\$table->date('created_at');
double()	The double method creates a DOUBLE equivalent column with the given precision (total digits) and scale (decimal digits)	\$table->double('amount', 8, 2);
enum()	The enum method creates a ENUM equivalent column with the given valid values	\$table->enum('difficulty', ['easy', 'hard']);

AVAILABLE COLUMN TYPES

float()	The float method creates a FLOAT equivalent column with the given precision (total digits) and scale (decimal digits)	\$table->float('amount', 8, 2);
foreignId()	The foreignId method creates an UNSIGNED BIGINT equivalent	\$table->foreignId('user_id')
foreignIdFor()	The foreignIdFor method adds a {column}_id UNSIGNED BIGINT equivalent column for a given model class	\$table->foreignIdFor(User::class);
geometryCollection()	The geometryCollection method creates a GEOMETRYCOLLECTION equivalent	\$table->geometryCollection('positions');
geometry()	The geometry method creates a GEOMETRY equivalent	\$table->geometry('positions');
id()	The id method is an alias of the bigIncrements method	\$table->id();
increments()	The increments method creates an auto-incrementing UNSIGNED INTEGER equivalent column as a primary key	\$table->increments('id');
integer()	The integer method creates an INTEGER equivalent	\$table->integer('votes')
ipAddress()	The ipAddress method creates a VARCHAR equivalent	\$table->ipAddress('visitor');
json()	The json method creates a JSON equivalent	\$table->json('options')
longText()	The longText method creates a LONGTEXT equivalent	\$table->longText('description');

AVAILABLE COLUMN TYPES

mediumIncrements()	The mediumIncrements method creates an auto-incrementing UNSIGNED MEDIUMINT equivalent column as a primary key	\$table->mediumIncrements('id');
mediumInteger()	The mediumInteger method creates a MEDIUMINT equivalent	\$table->mediumInteger('votes');
mediumText()	The mediumText method creates a MEDIUMTEXT equivalent	\$table->mediumText('description');
smallIncrements()	The smallIncrements method creates an auto-incrementing UNSIGNED SMALLINT equivalent column as a primary key	\$table->smallIncrements('id');
smallInteger()	The smallInteger method creates a SMALLINT equivalent	\$table->smallInteger('votes');
string()	The string method creates a VARCHAR equivalent column of the given length	\$table->string('name', 100);
text()	The text method creates a TEXT equivalent column	\$table->text('description');

AVAILABLE COLUMN TYPES

time()	The time method creates a TIME equivalent column with an optional precision (total digits)	\$table->time('sunrise', \$precision = 0);
timestamp()	The timestamp method creates a TIMESTAMP equivalent column with an optional precision (total digits)	\$table->timestamp('added_at', \$precision = 0);
timestamps()	The timestamps method creates created_at and updated_at TIMESTAMP equivalent columns with an optional precision (total digits)	\$table->timestamps(\$precision = 0);
tinyIncrements()	The tinyIncrements method creates an auto-incrementing UNSIGNED TINYINT equivalent column as a primary key	\$table->tinyIncrements('id');
tinyInteger()	The tinyInteger method creates a TINYINT equivalent	\$table->tinyInteger('votes');
tinyText()	The tinyText method creates a TINYTEXT equivalent	\$table->tinyText('notes')
unsignedBigInteger()	The unsignedBigInteger method creates an UNSIGNED BIGINT equivalent	\$table->unsignedBigInteger('votes');
unsignedInteger()	The unsignedInteger method creates an UNSIGNED INTEGER equivalent	\$table->unsignedInteger('votes');

AVAILABLE COLUMN TYPES

unsignedMediumInteger()	The unsignedMediumInteger method creates an UNSIGNED MEDIUMINT equivalent	\$table->unsignedMediumInteger('votes');
unsignedSmallInteger()	The unsignedSmallInteger method creates an UNSIGNED SMALLINT equivalent	\$table->unsignedSmallInteger('votes');
unsignedTinyInteger()	The unsignedTinyInteger method creates an UNSIGNED TINYINT equivalent	\$table->unsignedTinyInteger('votes');

AVAILABLE COLUMN ATTRIBUTES

nullable()	Accept null value	\$table->string('email')->nullable()
default(\$value)	Set default value if null	\$table->string('email')->default(\$value)
useCurrent()	Set TIMESTAMP columns to use CURRENT_TIMESTAMP as default value	\$table->timestamp('created_at')->useCurrent()
useCurrentOnUpdate()	Set TIMESTAMP columns to use CURRENT_TIMESTAMP when a record is updated	\$table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
collation()	Specify a collation for the column	\$table->string('email')->collation('utf8mb4_unicode_ci')
charset()	Specify a character set for the column (MySQL).	\$table->string('email')->charset('utf8mb4')
autoIncrement()	Set INTEGER columns as auto-incrementing (primary key).	\$table->increments('id')
first()	Place the column "first" in the table (MySQL).	\$table->increments('id')->first()
invisible()	Make the column "invisible" to SELECT * queries (MySQL).	\$table->string('email')->invisible()
unsigned()	Set INTEGER columns as UNSIGNED (MySQL)	\$table->integer('votes')->unsigned()
unique()	Ensure unique value	\$table->string('email')->unique()
change()	Allows you to modify the type and attributes of existing columns.	\$table->string('name', 50)->change();

CREATE RENAME AND DROP TABLES

● ● ● 2023_05_02_110102_drop_profile_table.php

```
12     public function up(): void
13     {
14         Schema::dropIfExists("user_profile");
15     }
```

● ● ● 2023_05_02_105819_rename_profile_table.php

```
12     public function up(): void
13     {
14         Schema::rename("profile", "user_profile");
15     }
```

● ● ● 2023_05_02_062343_create_profile_table.php

```
7     return new class extends Migration
8     {
9         public function up(): void
10        {
11            Schema::create('profile', function (Blueprint $table) {
12                $table->id();
13                $table->string('name');
14                $table->string('city');
15                $table->string('phone');
16                $table->timestamps();
17            });
18        }
19        public function down(): void
20        {
21            Schema::dropIfExists('profile');
22        }
23    };
```

ADD RENAME AND DROP COLUMN

● ● ● 2023_05_02_112754_modify_profile_table.php

```
12     public function up(): void
13     {
14         Schema::table('profile', function (Blueprint $table) {
15             $table->after('city', function ($table) {
16                 $table->string('address_line1');
17                 $table->string('address_line2');
18             });
19         });
20     }
```

ADD RENAME AND DROP COLUMN

```
Schema::table('users', function (Blueprint $table) {  
    $table->dropColumn('votes');  
});
```

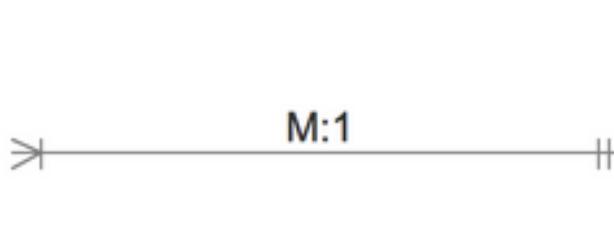
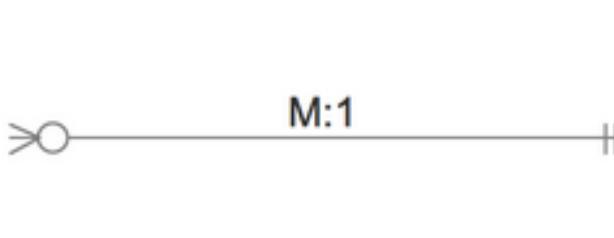
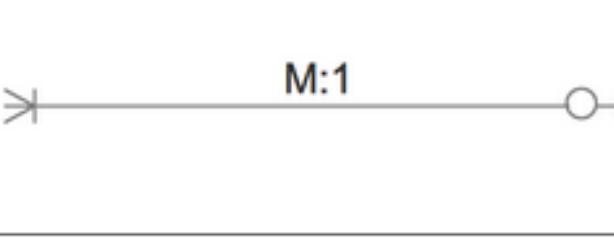
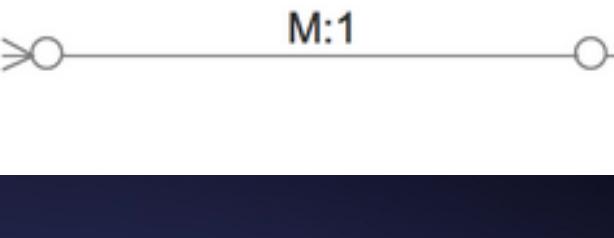
```
Schema::table('users', function (Blueprint $table) {  
    $table->renameColumn('from', 'to');  
});
```

```
Schema::table('users', function (Blueprint $table) {  
    $table->dropColumn(['votes', 'avatar', 'location']);  
});
```

RELATIONSHIP SIGN

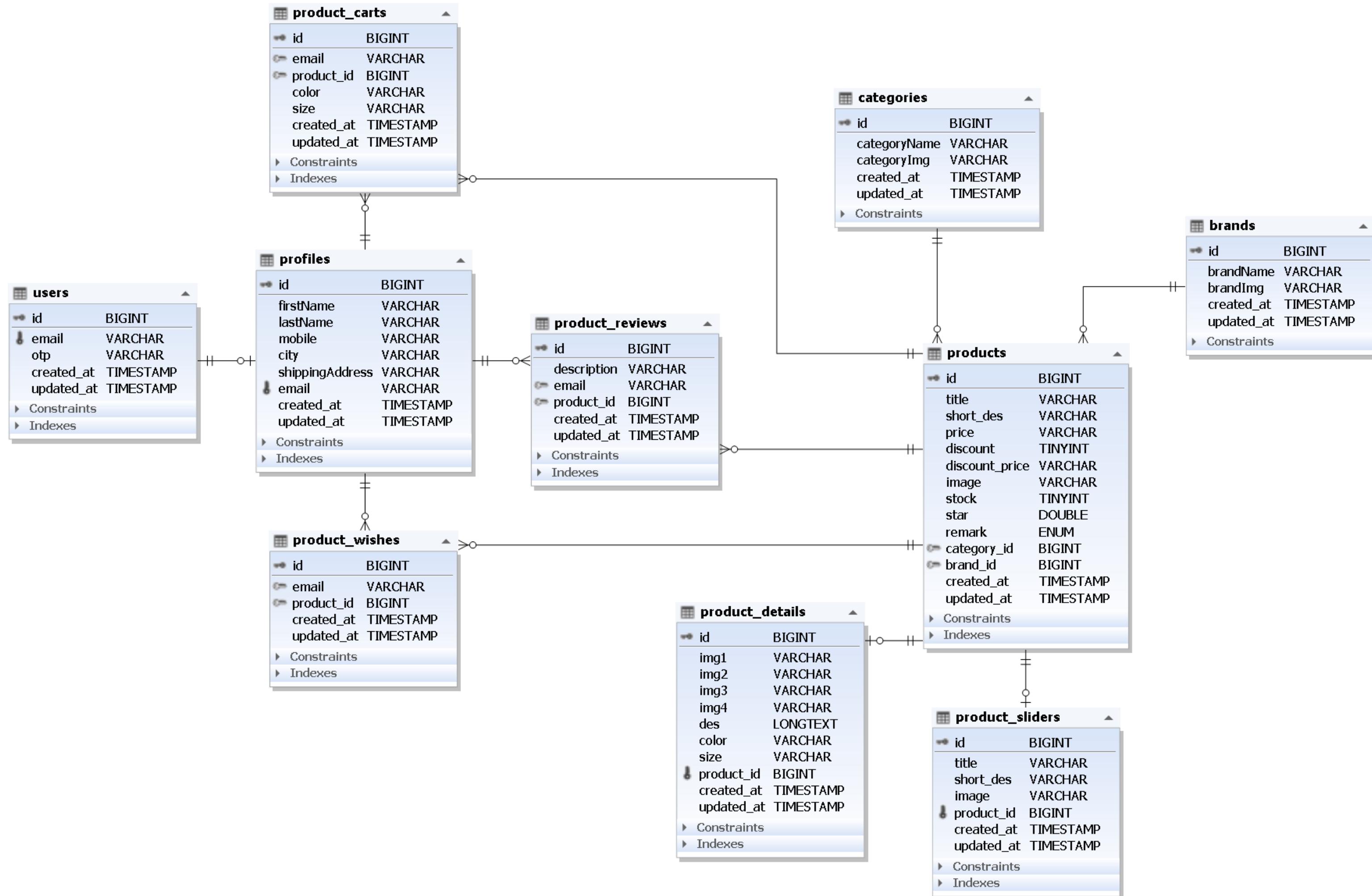
+○	Zero or One
✳	One or More
	One and only One
○○	Zero or More

RELATIONSHIP SIGN

	a one through many notation on one side of a relationship and a one and only one on the other
	a zero through many notation on one side of a relationship and a one and only one on the other
	a one through many notation on one side of a relationship and a zero or one notation on the other
	a zero through many notation on one side of a relationship and a zero or one notation on the other

RELATIONSHIP SIGN

	a zero through many on both sides of a relationship
	a zero through many on one side and a one through many on the other
	a one through many on both sides of a relationship
	a one and only one notation on one side of a relationship and a zero or one on the other
	a one and only one notation on both sides

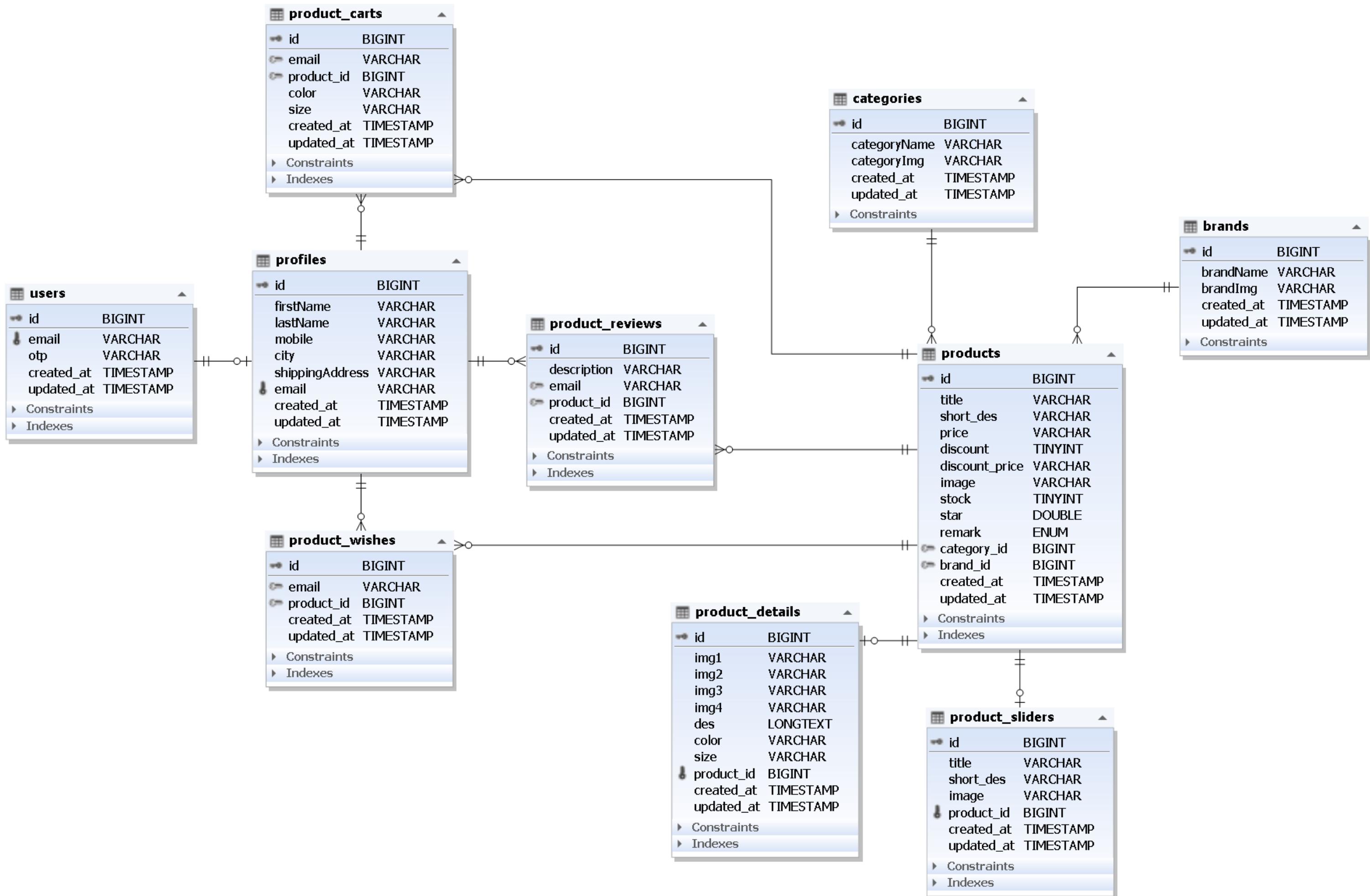


RELATIONSHIP CONSTRAINT

Method	Description
<code>\$table->cascadeOnUpdate();</code>	Updates should cascade.
<code>\$table->restrictOnUpdate();</code>	Updates should be restricted.
<code>\$table->cascadeonDelete();</code>	Deletes should cascade.
<code>\$table->restrictonDelete();</code>	Deletes should be restricted.
<code>\$table->>nullonDelete();</code>	Deletes should set the foreign key value to null.

LETS DO A DATABASE PROEJCT

For Better Understanding



LETS DO A DATABASE PROEJCT

user migration

● ● ● 2023_02_16_065502_create_users.php

```
12     public function up(): void
13     {
14         Schema::create('users', function (Blueprint $table) {
15             $table->id();
16             $table->string('email',50)->unique();
17             $table->string('otp',10);
18             $table->timestamp('created_at')->useCurrent();
19             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
20         });
21     }
```

LETS DO A DATABASE PROEJCT

profiles migration

● ● ● 2023_02_16_065520_create_profiles.php

```
12     public function up(): void
13     {
14         Schema::create('profiles', function (Blueprint $table) {
15             $table->id();
16             $table->string('firstName',50);
17             $table->string('lastName',50);
18             $table->string('mobile',50);
19             $table->string('city',50);
20             $table->string('shippingAddress',1000);
21             $table->string('email',50)->unique();
22             $table->foreign('email')->references('email')->on('users')
23             ->restrictOnDelete()
24             ->cascadeOnUpdate();
25             $table->timestamp('created_at')->useCurrent();
26             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
27         });
28     }
```

LETS DO A DATABASE PROEJCT

categories migration

● ● ● 2023_02_16_065529_create_categories.php

```
12     public function up(): void
13     {
14         Schema::create('categories', function (Blueprint $table) {
15             $table->id();
16             $table->string('categoryName',50);
17             $table->string('categoryImg',300);
18             $table->timestamp('created_at')->useCurrent();
19             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
20         });
21     }
```

LETS DO A DATABASE PROEJCT

brands migration

● ● ● 2023_02_16_065654_create_brands.php

```
12     public function up(): void
13     {
14         Schema::create('brands', function (Blueprint $table) {
15             $table->id();
16             $table->string('brandName',50);
17             $table->string('brandImg',300);
18             $table->timestamp('created_at')->useCurrent();
19             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
20         });
21     }
```

LETS DO A DATABASE PROEJCT

products migration

```
12     public function up(): void
13     {
14         Schema::create('products', function (Blueprint $table) {
15             $table->id();
16             $table->string('title', 200);
17             $table->string('short_des', 500);
18             $table->string('price', 50);
19             $table->boolean('discount');
20             $table->string('discount_price', 50);
21             $table->string('image', 200);
22             $table->boolean('stock');
23             $table->float('star');
24             $table->enum('remark', ['popular', 'new', 'top', 'special', 'trending', 'regular']);
25
26             $table->unsignedBigInteger('category_id');
27             $table->unsignedBigInteger('brand_id');
28
29             $table->foreign('category_id')->references('id')->on('categories')
30                 ->restrictonDelete()
31                 ->cascadeOnUpdate();
32
33             $table->foreign('brand_id')->references('id')->on('brands')
34                 ->restrictonDelete()
35                 ->cascadeOnUpdate();
36
37             $table->timestamp('created_at')->useCurrent();
38             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
39         });
40     }
```

LETS DO A DATABASE PROEJCT

product_reviews

2023_02_17_144756_create_product_reviews.php

```
public function up(): void
{
    Schema::create('product_reviews', function (Blueprint $table) {
        $table->id();
        $table->string('description', 1000);

        $table->string('email', 50);
        $table->foreign('email')->references('email')->on('profiles')
            ->restrictOnDelete()
            ->restrictOnUpdate();

        $table->unsignedBigInteger('product_id');
        $table->foreign('product_id')->references('id')->on('products')
            ->restrictOnDelete()
            ->restrictOnUpdate();

        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
    });
}
```

LETS DO A DATABASE PROEJCT

product_details

● ● ● 2023_02_17_164424_create_product_details.php

```
7     public function up(): void
8     {
9         Schema::create('product_details', function (Blueprint $table) {
10             $table->id();
11             $table->string('img1',200);
12             $table->string('img2',200);
13             $table->string('img3',200);
14             $table->string('img4',200);
15             $table->longText('des');
16             $table->string('color',200);
17             $table->string('size',200);
18
19             $table->unsignedBigInteger('product_id')->unique();
20             $table->foreign('product_id')->references('id')->on('products')
21                 ->restrictOnDelete()
22                 ->restrictOnUpdate();
23
24
25             $table->timestamp('created_at')->useCurrent();
26             $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
27         });
28     }
```

LETS DO A DATABASE PROEJCT

product_sliders

2023_02_17_184723_create_product_sliders.php

```
public function up(): void
{
    Schema::create('product_sliders', function (Blueprint $table) {
        $table->id();
        $table->string('title',200);
        $table->string('short_des',500);
        $table->string('image',200);
        $table->unsignedBigInteger('product_id')->unique();
        $table->foreign('product_id')->references('id')->on('products')
            ->restrictOnDelete()
            ->restrictOnUpdate();
        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
    });
}
```

LETS DO A DATABASE PROEJCT

product_wishes

```
public function up(): void
{
    Schema::create('product_wishes', function (Blueprint $table) {
        $table->id();
        $table->string('email',50);
        $table->unsignedBigInteger('product_id');

        $table->foreign('product_id')->references('id')->on('products')
            ->restrictOnDelete()
            ->restrictOnUpdate();

        $table->foreign('email')->references('email')->on('profiles')
            ->restrictOnDelete()
            ->restrictOnUpdate();

        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
    });
}
```

LETS DO A DATABASE PROEJCT

product_carts

```
public function up(): void
{
    Schema::create('product_carts', function (Blueprint $table) {
        $table->id();

        $table->string('email',50);
        $table->unsignedBigInteger('product_id');

        $table->string('color',200);
        $table->string('size',200);

        $table->foreign('product_id')->references('id')->on('products')
            ->restrictonDelete()
            ->restrictOnUpdate();

        $table->foreign('email')->references('email')->on('profiles')
            ->restrictonDelete()
            ->restrictOnUpdate();

        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
    });
}
```