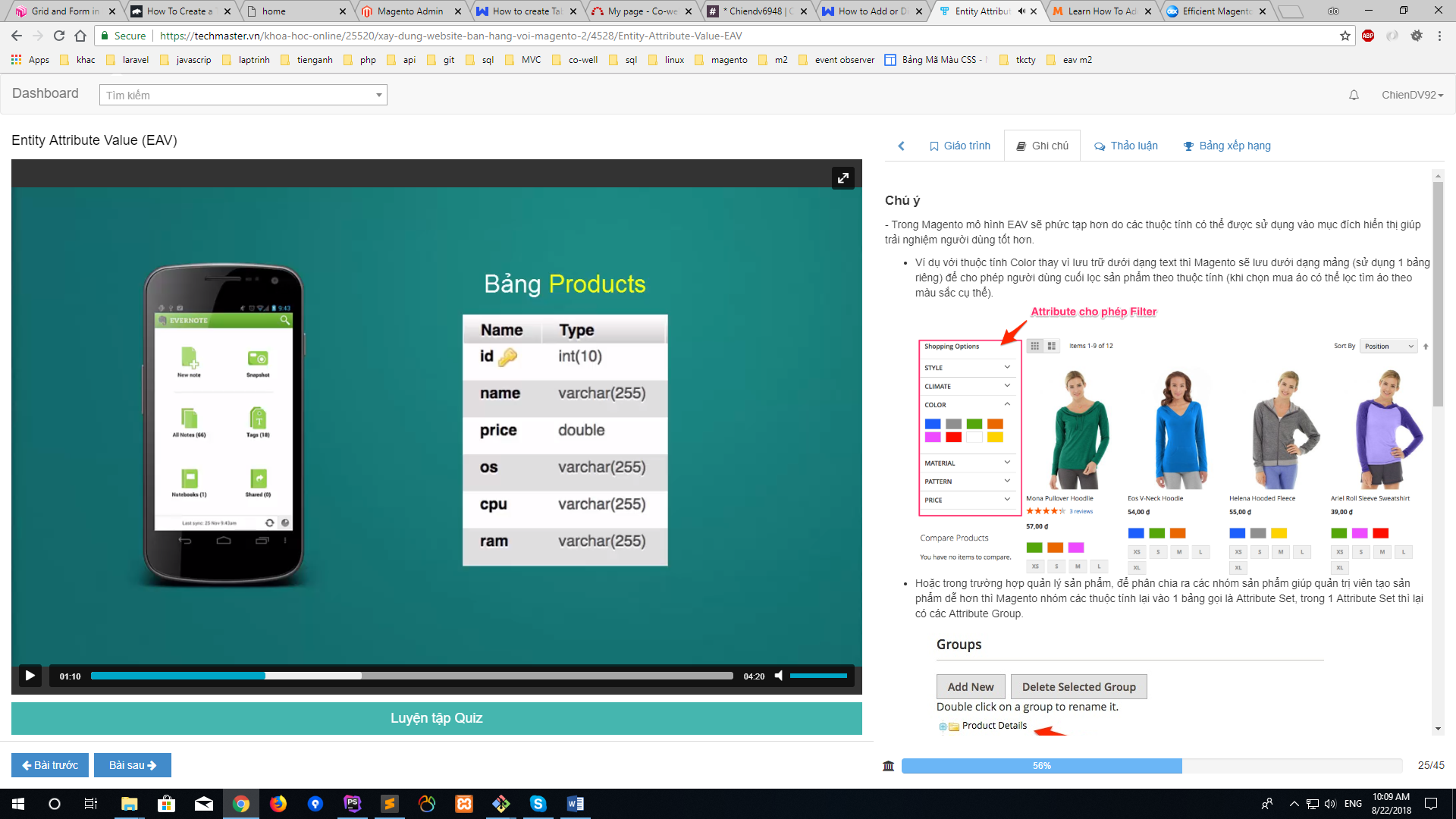
# EAV (Entity Attribute Value)

* Là kiểu cấu trúc cơ sở dữ liệu trong magento
* **Entity (thực thể) Attribute(thuộc tính) Value(giá trị)**
* Cho phép 1 đối tượng có thể mở rộng mà k cần can thiệp vào csdl

Vi dụ :

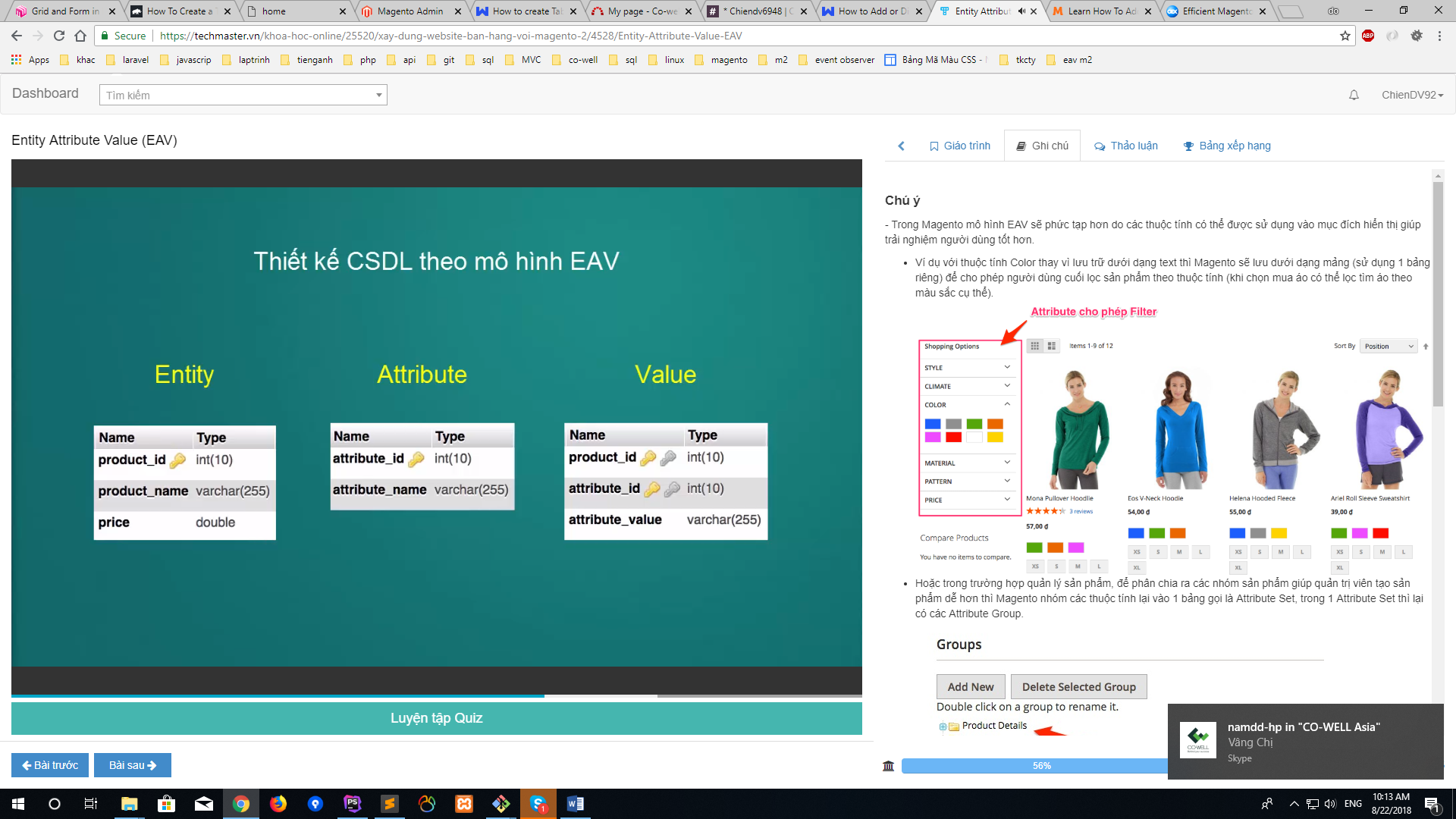
* **thiết kế csdl bình thường (theo hang ngang)**

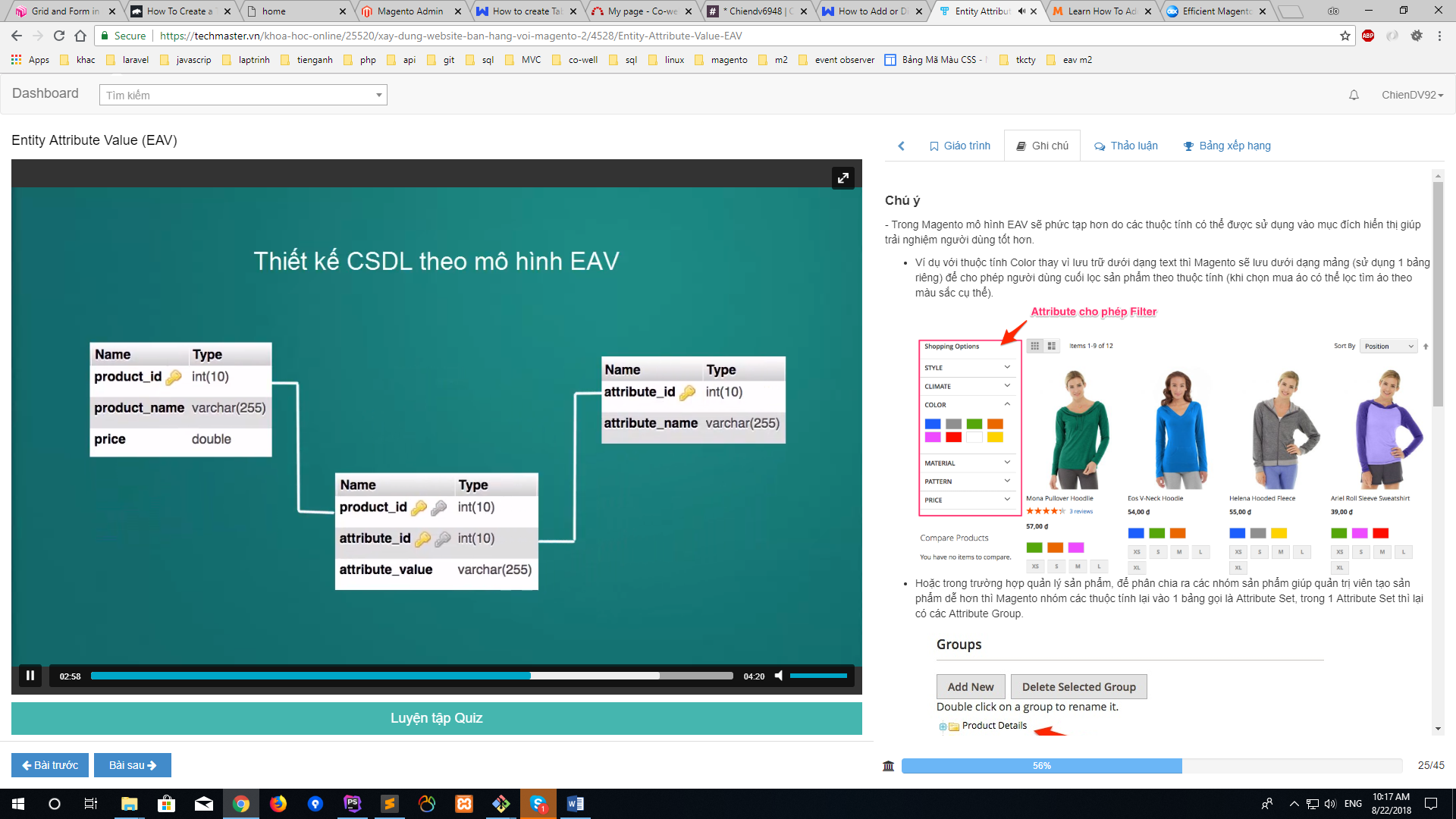
Product



Muốn bổ xung thêm thuộc tính sản phẩm thì

* Thêm cột vào csdl và xử lý code
* **Theo mô hình EAV**





* Entity : chứa thực thể product ()
* Attribute : thuộc tính product thông tin sản phẩm
* Value: Giá trị thuộc tính tương ứng với mỗi sản phẩm

Ưu điểm :

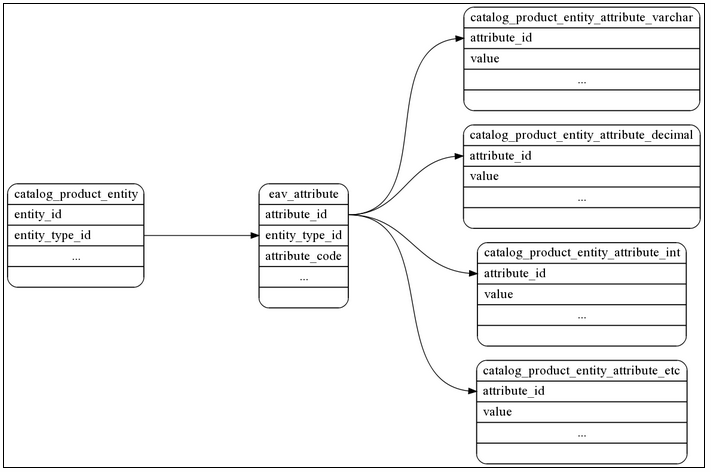
* Mỗi khi thêm thuộc tính mới cho sản phẩm chỉ cẩn insert data vào bảng thay vì thêm cột cho bảng
* Tạo các sp khác nhau có thể có thuộc tính chung

Nhược điểm

* Truy vấn phức tạp
* Time truy vấn lâu

Khắc phục nhược điểm

* Lưu toàn bộ data vào bảng phụ (FLAT table) lưu data vào join 3 bảng sau đó truy vấn lấy data từ bảng này
* Sử dụng cache lưu cache dữ liệu đã truy vấn

[](http://divante.co/blog/wp-content/uploads/2014/12/1.png)

Sơ đồ mô tả cấu hình bảng EAV mô tả hàng hóa trong cửa hàng.

Bảng đầu tiên chứa một tập các thực thể sản phẩm (entity\_id) là các chỉ mục sản phẩm cơ sở với kiểu (entity\_type\_id) cho biết loại thực thể trong cấu trúc EAV, cùng với các khóa cơ sở (ví dụ cho các sản phẩm này có thể là SKU của chúng).

Bảng ở giữa chứa tất cả các thuộc tính được tìm thấy trong Magento. Bằng cách lọc các bảng theo loại thực thể (entity\_type\_id), chúng ta có thể tìm hiểu tất cả các thuộc tính sản phẩm có thể có (ví dụ giá, kích thước, màu sắc vv).

Các bảng ở bên phải trình bày các giá trị cuối cùng của một thuộc tính đã cho. Cột nơi một giá trị đã cho được lưu tùy thuộc vào loại giá trị. Đó là khi bạn muốn ghi lại thuộc tính sản phẩm "màu", bạn nên ghi lại nó trong bảng catalog\_product\_entity\_varachar vì chúng ta biết rằng các giá trị thuộc tính "màu" thuộc loại chuỗi.

Để mô tả một sản phẩm trong cấu trúc bảng flat, chúng ta có thể chuyển đổi sơ đồ trên thành một bảng với số cột tương ứng với số lượng thuộc tính được yêu cầu. Vậy tại sao EAV?

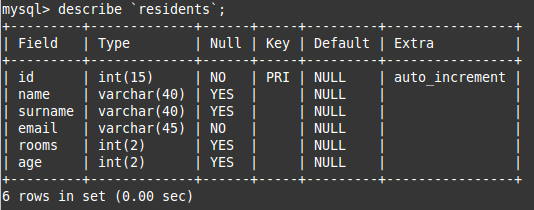
## Flat với EAV

Khi bạn nhìn vào tất cả những gì từ quan điểm Magento, EAV là một lựa chọn tuyệt vời vì các cửa hàng hoặc ứng dụng khác nhau có thể có các đặc điểm khác nhau và **nhờ EAV bạn có thể chơi với các thuộc tính mà không cần phải thiết kế lại cơ sở dữ liệu** .

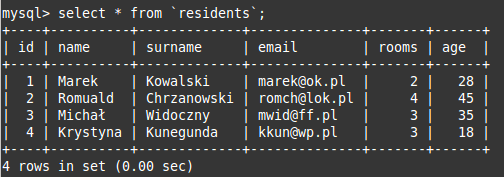
Hơn nữa, các bảng flat có số cột tối đa cụ thể (ví dụ: inoDB có thể có tối đa 1000 cột) giới hạn số lượng thuộc tính tối đa. Trong EAV, đây không phải là vấn đề.

**EAV, tuy nhiên, có một nhược điểm đáng kể, đó là hiệu suất** . Khi bạn muốn đến một thuộc tính cụ thể, bạn phải lấy dữ liệu từ một vài bảng được kết nối với nhu cầu thực hiện một vài truy vấn SQL trong Magento. Khi chúng ta cũng tiếp cận các thuộc tính bổ sung với một kiểu dữ liệu khác, các JOIN mới sẽ xuất hiện trong các truy vấn. Khi chúng ta thêm một nhà phát triển thiếu kinh nghiệm, người truy xuất nhiều dữ liệu theo lô thông qua một tải trong một vòng lặp, sự sụt giảm dung lượng tuyến tính là chắc chắn.

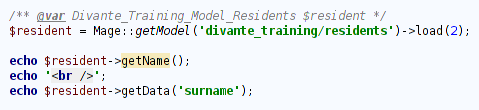
# ORM - cấu trúc bảng flat

Hãy tạo một bảng flat chứa thông tin về khách của khách sạn:  
[](http://divante.co/blog/wp-content/uploads/2014/12/2.png)

Bảng chứa thông tin cơ bản về khách và chúng tôi sẽ chỉ sử dụng hai loại dữ liệu, đó là INT và VARCHAR.

Tiếp theo, hãy điền vào bảng với dữ liệu:  
[](http://divante.co/blog/wp-content/uploads/2014/12/3.png)

và truy xuất dữ liệu cho khách thứ hai:

[](http://divante.co/blog/wp-content/uploads/2014/12/4.png)

Đoạn mã trên đề cập đến mô hình *divante\_training / resident* và truy xuất dữ liệu cho khách thứ hai bằng cách sử dụng phương thức tải. Kết quả của việc thực thi mã Magento ở trên là hiển thị tên và họ của khách:

[5](http://divante.co/blog/wp-content/uploads/2014/12/5.png)

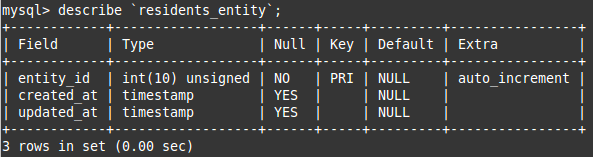
ORM đã làm mọi thứ cho chúng tôi. Hãy để chúng tôi không tin tưởng nó và kiểm tra những gì các truy vấn được thực hiện:

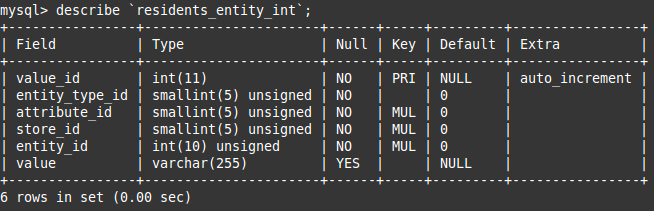
[6](http://divante.co/blog/wp-content/uploads/2014/12/6.png)

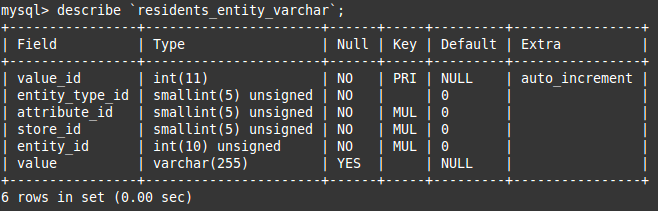
Như chúng ta thấy, ORM đã tạo ra một truy vấn cơ sở dữ liệu đơn giản, không có gì đáng ngạc nhiên.

# ORM - EAV

Bây giờ, chúng ta hãy tạo một cấu trúc bảng tương tự trình bày khách của khách sạn nhưng lần này trong bối cảnh EAV.

[](http://divante.co/blog/wp-content/uploads/2014/12/7.png)

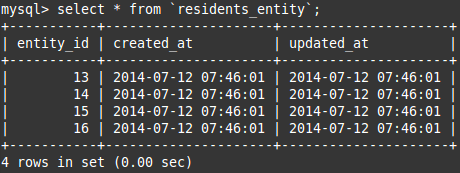
[](http://divante.co/blog/wp-content/uploads/2014/12/8.png)

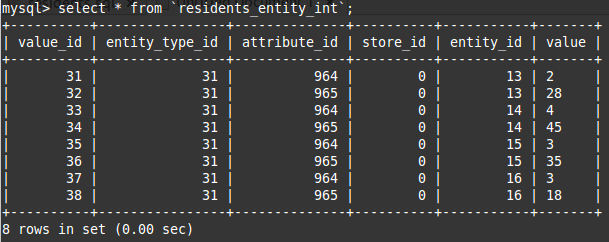
[](http://divante.co/blog/wp-content/uploads/2014/12/9.png)

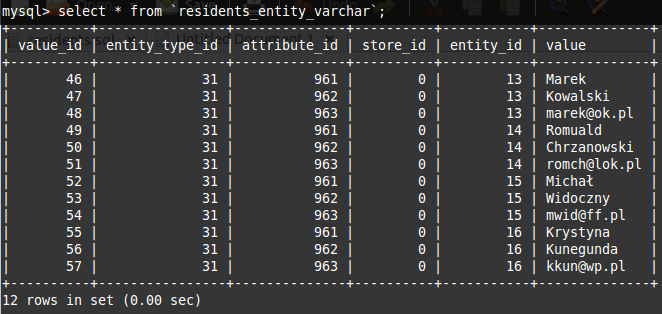
Như chúng ta đã thấy ở trên, lần này tôi phải tạo ba bảng: Đầu tiên sẽ chứa các thực thể của khách (các chỉ mục) trong khi thứ hai và thứ ba sẽ chứa các giá trị tùy thuộc vào kiểu dữ liệu (INT → resident\_entity\_int, và VARCHAR → resident\_entity\_varchar).

Một người đọc cảnh báo nên nhận thấy chúng tôi bỏ lỡ một bảng nữa với các thuộc tính (eav\_attribute - được mô tả trong khi thảo luận về lý thuyết EAV). Đó là, tuy nhiên, tạo ra trước đó cùng với toàn bộ cơ sở dữ liệu Magento như tất cả các loại bảng EAV Magento sử dụng nó (sản phẩm, đơn đặt hàng, vv).

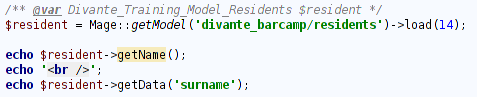
Hãy điền vào các bảng với dữ liệu tương tự với các bảng trong bảng flat.

[](http://divante.co/blog/wp-content/uploads/2014/12/10.png)

[](http://divante.co/blog/wp-content/uploads/2014/12/11.png)

[](http://divante.co/blog/wp-content/uploads/2014/12/12.png)

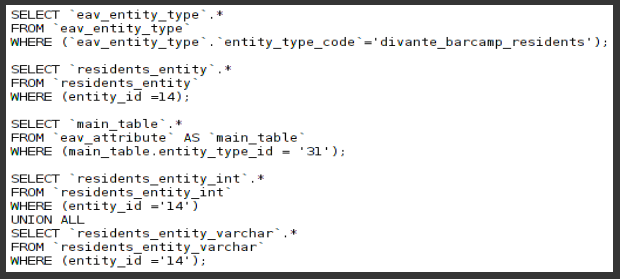
và hãy cùng một khách truy cập:

[](http://divante.co/blog/wp-content/uploads/2014/12/13.png)

Lần này chúng tôi sử dụng mô hình *divante\_barcamp / resident* và truy xuất dữ liệu cho khách thứ 14:

[14](http://divante.co/blog/wp-content/uploads/2014/12/14.png)

Như chúng ta thấy, cả ngữ cảnh tham chiếu và kết quả đều giống nhau nhưng những gì ORM đã làm là khác nhau:

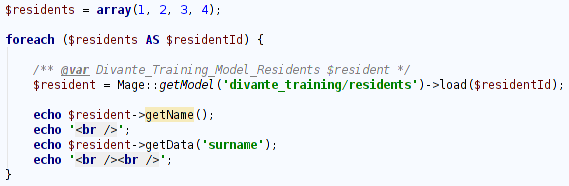
[](http://divante.co/blog/wp-content/uploads/2014/12/15.png)

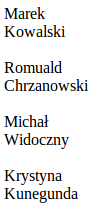
Truy xuất dữ liệu, ORM đã thực thi một số truy vấn và cuối cùng trong số chúng, chúng ta thấy hoạt động kết hợp các kết quả cho hai truy vấn bằng lệnh SQL UNION ALL.

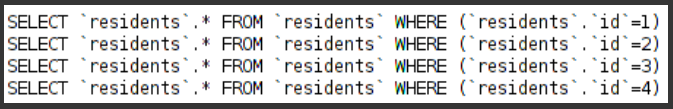
Nó không thể khác với ORM phải lấy kiểu thực thể EAV từ bảng *eaz\_entity\_type tabeli* , sau đó dữ liệu từ bảng *resident\_entity* chính của chúng ta *,* thông tin về các thuộc tính chúng ta cần ( *eav\_attribute* ), và cuối cùng là các giá trị của các thuộc tính đó ( *resident\_entity\_int* và *resident\_entity\_varchar* ).

# ORM - cấu trúc Flat (nhiều dữ liệu - tải trong một vòng lặp)

Trong bối cảnh của cấu trúc bảng flat được tạo trước đó, hãy thực thi mã dưới đây:

[](http://divante.co/blog/wp-content/uploads/2014/12/16.png)

Ở đây, chúng ta có thể thấy truy xuất dữ liệu bằng tải, như trong ví dụ trước, nhưng tất cả diễn ra trong một vòng lặp. Kết quả hiển thị các khách khác:  
[](http://divante.co/blog/wp-content/uploads/2014/12/17.png)

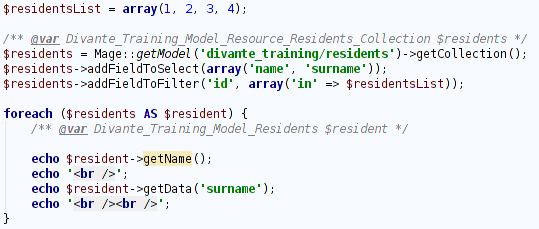
ORM đã thực hiện các hoạt động SQL sau đây:[](http://divante.co/blog/wp-content/uploads/2014/12/18.png)

Như có thể dễ dàng suy ra, cho mỗi lần lặp ORM gửi cùng một truy vấn một lần nữa và một lần nữa với một ID khác nhau trong điều kiện.

Ví dụ trên là sai lầm thường gặp nhất của các nhà phát triển Magento và một trong những ảnh hưởng đến hiệu suất. Nó nên được phổ biến kiến ​​thức rằng đây là nơi mà bộ sưu tập nên được sử dụng.

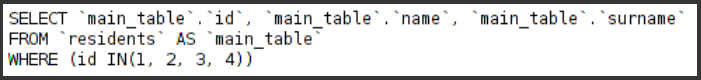
## ORM - cấu trúc bảng flat (nhiều dữ liệu - tập hợp)

Hãy sửa mã trước:

[](http://divante.co/blog/wp-content/uploads/2014/12/19.png)

Lần này sau khi đề cập đến mô hình *divante\_training / resident,* chúng tôi lấy bộ sưu tập của khách. Tiếp theo, chúng tôi tuyên bố rằng chúng tôi cần hai thuộc tính, tên và họ, và cuối cùng chúng tôi lọc bộ sưu tập của chúng tôi bằng ID.

Kết quả là giống hệt nhau (được hiển thị tên và họ của khách) nhưng những gì ORM đã khác với những gì chúng tôi đã có trước đây đến một mức độ đáng kể:

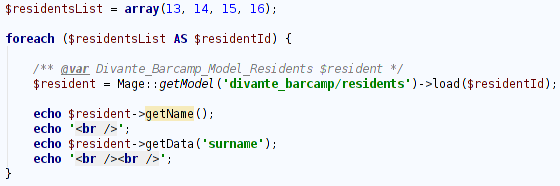
[](http://divante.co/blog/wp-content/uploads/2014/12/20.png)

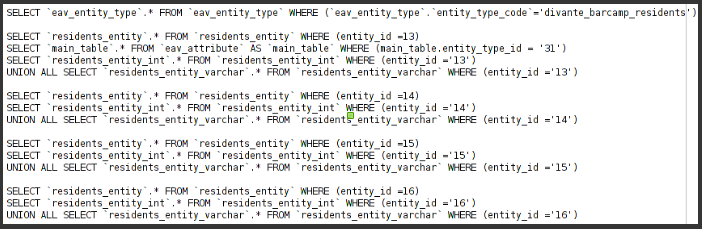
Chúng tôi thấy rằng thay vì cùng một truy vấn trùng lặp, chúng tôi có một truy vấn được kết hợp, trả về dữ liệu bắt buộc làm tăng hiệu suất của ứng dụng.

**Và điều gì xảy ra khi chúng ta lặp lại một lỗi dung lượng tương tự cho EAV?**

## ORM - EAV (nhiều dữ liệu - tải trong một vòng lặp)

Đây là ví dụ về mã tải dữ liệu của khách trong vòng lặp cho cấu trúc bảng EAV:

[](http://divante.co/blog/wp-content/uploads/2014/12/21.png)

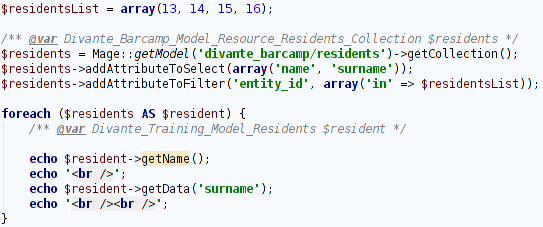
Và đây là kết quả của hoạt động ORM:[](http://divante.co/blog/wp-content/uploads/2014/12/22.png)

**Đây là kịch bản tồi tệ nhất cho mã Magento đối với hiệu suất** . Trong mỗi lần lặp lại, chúng ta tạo một số truy vấn bảng EAV một lần nữa. Chỉ truy vấn đầu tiên truy xuất loại thực thể EAV mới được thực hiện một lần, tất cả các truy vấn khác được kết hợp lại trong mỗi lần lặp vòng lặp.

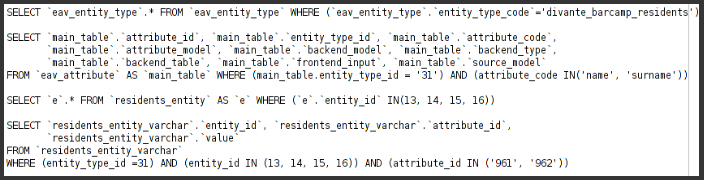
Nếu chúng tôi xử lý nhiều dữ liệu hơn (ví dụ 200 sản phẩm) trong mã của chúng tôi, hiệu suất giảm có thể được nhìn thấy bằng mắt trần, không cần chạy profiler.

Chúng ta không chờ đợi lâu hơn và sửa mã.

## ORM - EAV (nhiều dữ liệu - tập hợp)

[](http://divante.co/blog/wp-content/uploads/2014/12/23.png)

Đoạn mã trên đề cập đến mô hình *divante\_barcamp / cư dân* và truy lục một bộ sưu tập khách. Chúng tôi chỉ cần tên và họ của khách, và danh sách khách được yêu cầu sẽ được lọc theo ID sở hữu.

Kết quả được hiển thị không khác với kết quả trước đó, nhưng ORM có thể 'nghỉ' một chút.[](http://divante.co/blog/wp-content/uploads/2014/12/24.png)

Chúng ta có thể thấy rằng thay vì sao chép cùng các truy vấn cho mỗi lần lặp lại, các truy vấn được kết hợp và lọc dựa trên các phần tử cần thiết.

Tôi cũng muốn nhấn mạnh rằng truy vấn cuối cùng khác với truy vấn cuối cùng mà chúng tôi đã thực hiện trước đó bởi thực tế là chúng tôi chỉ tham khảo bảng *resident\_entity\_varchar* (không có ký tự nào sau bảng *resident\_entity\_int* ). Điều này xảy ra bởi vì cả hai loại dữ liệu cần thiết (tên và họ) là VARCHAR, và không cần phải đi đến các bảng khác vì chúng không chứa bất kỳ thuộc tính quan tâm nào đối với chúng tôi.

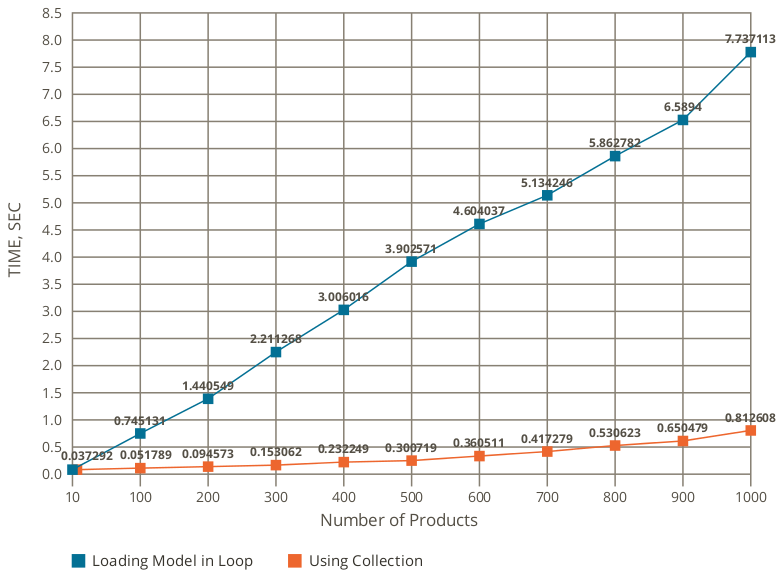
**Đối với bộ sưu tập, hãy nhớ nhập danh sách các thuộc tính được yêu cầu trừ khi chúng tôi chèn chúng một cách rõ ràng ORM sẽ không trả lại chúng.**

## Kết luận

Tôi hy vọng các ví dụ trên cho bạn thấy sự khác biệt giữa một bảng flat và EAV, không chỉ đối với giai đoạn thiết kế mà còn cho hiệu suất.

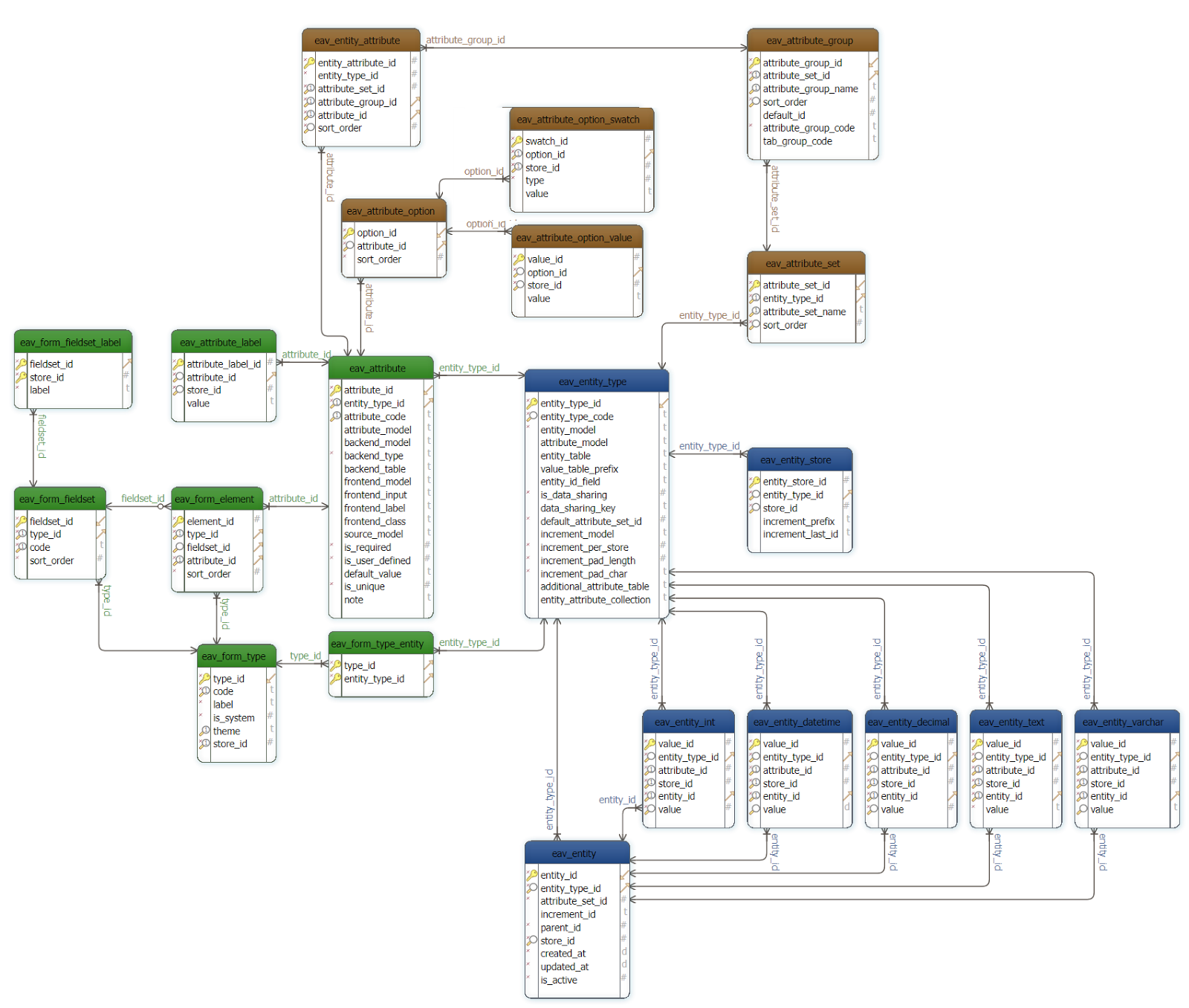
Trong mọi mô-đun Magento, bạn nên chú ý đến cấu hình bạn làm việc với và phương pháp xử lý dữ liệu.

Để tăng cường độ nhạy của bạn với hiệu suất, hãy để tôi sử dụng sơ đồ bên dưới:

[](http://divante.co/blog/wp-content/uploads/2014/12/25.png)

Sơ đồ thể hiện sự khác biệt về xử lý sản phẩm (EAV) trong một vòng lặp bằng tải và bằng các phương tiện thu thập (đó là các ví dụ được cung cấp ở trên).

**Sơ đồ với bảng eav\_ \*:**

[](https://belvg.com/blog/wp-content/uploads/2018/01/EAV-in-Magento-2.png)

**Những thực thể EAV nào có trong Magento 2:**

Danh sách các thực thể có thể được tìm thấy trong bảng eav\_entity\_type:

* khách hàng
* địa chỉ của khách hàng
* catalog\_category
* catalog\_product

**Những loại thực thể EAV nào có trong Magento 2:**

* eav\_entity\_int
* eav\_entity\_varchar
* eav\_entity\_text
* eav\_entity\_decimal
* eav\_entity\_datetime

## **Entities catalog\_product in Magento 2**

Để hiểu mô hình EAV trong Magento 2, hãy xem xét ví dụ “catalog\_product”, loại bảng được sử dụng để tạo các thuộc tính, các hàm của các thuộc tính trong nhóm thuộc tính và nhóm thuộc tính. Bên cạnh đó, chúng tôi có một cái nhìn sâu hơn về các bảng được sử dụng để gán sản phẩm với một thuộc tính. Ở mỗi bước, có mã SQL được tạo bởi Magento.

1. Tạo thuộc tính cho từng loại có dữ liệu mặc định.
2. Tạo thuộc tính được thiết lập với các thuộc tính mới.
3. Thêm nhóm thuộc tính mới trong tập thuộc tính.
4. Xem xét các bảng được sử dụng để gán sản phẩm với một thuộc tính.

**Trước hết, chúng tôi tạo một thuộc tính cho từng loại với dữ liệu mặc định. Bạn có thể thực hiện nó trong bảng quản trị: Cửa hàng - Thuộc tính - Thuộc tính sản phẩm - Thêm thuộc tính mới**

* Text Field
* Text Area
* Date
* Yes/No
* Multiple Select
* Dropdown
* Price
* Media Image
* Fixed Product Tax
* Visual Swatch
* Text Swatch
  1. **Tạo thuộc tính sản phẩm với kiểu trường văn bản**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `frontend\_class`, `source\_model`, `is\_required`, `is\_user\_defined`, `default\_value`, `is\_unique`)

VALUES (4, 'text\_field', NULL, 'varchar', 'text', 'Text Field', NULL, NULL, 0, 1, NULL, 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='159')

Nếu chúng ta quên gán nhãn cho thuộc tính, sẽ có một dòng được thêm vào:

INSERT INTO `eav\_attribute\_label` (

`attribute\_id`,

`store\_id`,

`value`)

VALUES

(

'159',

1,

'Label Name'

)

INSERT INTO `catalog\_eav\_attribute` (

`attribute\_id`,

`is\_global`,

`is\_searchable`,

`is\_filterable`,

`is\_comparable`,

`is\_visible\_on\_front`,

`is\_html\_allowed\_on\_front`,

`is\_filterable\_in\_search`,

`used\_in\_product\_listing`,

`used\_for\_sort\_by`,

`apply\_to`,

`is\_visible\_in\_advanced\_search`,

`is\_used\_for\_promo\_rules`,

`is\_used\_in\_grid`,

`is\_visible\_in\_grid`,

`is\_filterable\_in\_grid`

)

VALUES (159, 0, 0, 0, 0, 0, 1, 0, 0, 0, NULL, 0, 0, 1, 1, 1)

**1.2. Tạo thuộc tính sản phẩm với loại vùng văn bản**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `default\_value`, `is\_unique`) VALUES (4, 'text\_area', NULL, 'text', 'textarea', 'Text Area', NULL, 0, 1, NULL, 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='160')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_wysiwyg\_enabled`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (160, 0, 0, 0, 0, 0, 1, 0, 0, NULL, 0, 0, 0, 1, 1, 1)

**1.3. Tạo thuộc tính sản phẩm với loại ngày**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_model`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `default\_value`, `is\_unique`) VALUES (4, 'date', 'Magento\\Eav\\Model\\Entity\\Attribute\\Backend\\Datetime', 'datetime', 'Magento\\Eav\\Model\\Entity\\Attribute\\Frontend\\Datetime', 'date', 'Date', NULL, 0, 1, NULL, 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='161')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `used\_for\_sort\_by`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (161, 0, 0, 0, 0, 0, 1, 0, 0, 0, NULL, 0, 0, 1, 1, 1)

**1.4. Tạo thuộc tính sản phẩm với loại có / không**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `default\_value`, `is\_unique`) VALUES (4, 'yes\_no', NULL, 'int', 'boolean', 'Yes No', 'Magento\\Eav\\Model\\Entity\\Attribute\\Source\\Boolean', 0, 1, '0', 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='162')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `used\_for\_sort\_by`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (162, 0, 0, 0, 0, 0, 1, 0, 0, 0, NULL, 0, 0, 1, 1, 1)

**1.5. Tạo thuộc tính sản phẩm với nhiều loại lựa chọn**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `is\_unique`) VALUES (4, 'multiple\_select', 'Magento\\Eav\\Model\\Entity\\Attribute\\Backend\\ArrayBackend', 'varchar', 'multiselect', 'Multiple Select', NULL, 0, 1, 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='163')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (163, 0, 0, 0, 0, 0, 1, 0, 0, NULL, 0, 0, 1, 1, 1)

INSERT INTO `eav\_attribute\_option` (`attribute\_id`, `sort\_order`) VALUES ('163', '1')

DELETE FROM `eav\_attribute\_option\_value` WHERE (option\_id = '213')

INSERT INTO `eav\_attribute\_option\_value` (`option\_id`, `store\_id`, `value`) VALUES ('213', '0', 'Option one')

INSERT INTO `eav\_attribute\_option` (`attribute\_id`, `sort\_order`) VALUES ('163', '2')

DELETE FROM `eav\_attribute\_option\_value` WHERE (option\_id = '214')

INSERT INTO `eav\_attribute\_option\_value` (`option\_id`, `store\_id`, `value`) VALUES ('214', '0', 'Option two')

UPDATE `eav\_attribute` SET `default\_value` = '' WHERE (attribute\_id = '163')

**1.6. Tạo thuộc tính sản phẩm với loại thả xuống**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `is\_unique`) VALUES (4,  'dropdown',  NULL, 'int', 'select', 'Dropdown', 'Magento\\Eav\\Model\\Entity\\Attribute\\Source\\Table', 0,  1,  0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='164')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `used\_for\_sort\_by`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (164, 0, 0, 0, 0, 0, 1, 0, 0, 0, NULL, 0, 0, 1, 1, 1)

INSERT INTO `eav\_attribute\_option` (`attribute\_id`, `sort\_order`) VALUES ('164', 1)

DELETE FROM `eav\_attribute\_option\_value` WHERE (option\_id = '215')

INSERT INTO `eav\_attribute\_option\_value` (`option\_id`, `store\_id`, `value`) VALUES ('215', '0', 'Option one')

INSERT INTO `eav\_attribute\_option` (`attribute\_id`, `sort\_order`) VALUES ('164', '2')

DELETE FROM `eav\_attribute\_option\_value` WHERE (option\_id = '216')

INSERT INTO `eav\_attribute\_option\_value` (`option\_id`, `store\_id`, `value`) VALUES ('216', '0', 'Option two')

UPDATE `eav\_attribute` SET `default\_value` = '' WHERE (attribute\_id = '164')

**1.7. Tạo thuộc tính sản phẩm với loại giá**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `default\_value`, `is\_unique`) VALUES (4, 'price\_example',  'Magento\\Catalog\\Model\\Product\\Attribute\\Backend\\Price',  'decimal',  'price', 'Price Example', NULL, 0, 1, NULL,  0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='165')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `used\_for\_sort\_by`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (165, 0, 0, 0, 0, 0, 1, 0, 0, 0, NULL, 0, 0, 1, 1, 1)

**1.8. Tạo thuộc tính sản phẩm với loại hình ảnh phương tiện**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `backend\_type`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `is\_unique`) VALUES (4, 'media\_image', NULL, 'varchar', 'media\_image', 'Media image', NULL, 0, 1, 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='166')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `used\_for\_sort\_by`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (166, 0, 0, 0, 0, 0, 1, 0, 0, 0, NULL, 0, 0,  1, 1, 1)

**1.9. Tạo thuộc tính sản phẩm với loại thuế sản phẩm cố định**

INSERT INTO `eav\_attribute` (`entity\_type\_id`, `attribute\_code`, `backend\_model`, `frontend\_input`, `frontend\_label`, `source\_model`, `is\_required`, `is\_user\_defined`, `is\_unique`) VALUES (4, 'fixed\_product\_tax', 'Magento\\Weee\\Model\\Attribute\\Backend\\Weee\\Tax', 'weee', 'Fixed Product Tax', NULL, 0, 1, 0)

DELETE FROM `eav\_attribute\_label` WHERE (attribute\_id ='167')

INSERT INTO `catalog\_eav\_attribute` (`attribute\_id`, `is\_global`, `is\_searchable`, `is\_filterable`, `is\_comparable`, `is\_visible\_on\_front`, `is\_html\_allowed\_on\_front`, `is\_filterable\_in\_search`, `used\_in\_product\_listing`, `used\_for\_sort\_by`, `apply\_to`, `is\_visible\_in\_advanced\_search`, `is\_used\_for\_promo\_rules`, `is\_used\_in\_grid`, `is\_visible\_in\_grid`, `is\_filterable\_in\_grid`) VALUES (167, 0, 0, 0, 0, 0, 1, 0, 0, 0, 'simple,virtual,bundle,downloadable,configurable', 0, 0, 1, 1, 1)

## **EAV attribute**