

**طراحی پایگاه داده**

**دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف**

زمستان 1402



**معین آعلی - 401105561­**

**فهرست عناوین**

[**1. سوال شماره 1 2**](#_Toc161572506)

[**1.1. الف) 2**](#_Toc161572507)

[**1.2. ب) 2**](#_Toc161572508)

[**1.3. ج) 3**](#_Toc161572509)

[**1.4. د) 3**](#_Toc161572510)

[**1.5. ه) 3**](#_Toc161572511)

[**1.6. و) 4**](#_Toc161572512)

[**2. سوال شماره 2 5**](#_Toc161572513)

[**3. سوال شماره 3: 7**](#_Toc161572514)

## سوال شماره 1

### **الف)**

بله، در برخی موارد می توان یک موجودیت ضعیف را به موجودیت قوی تبدیل کرد.

موجودیت ضعیف موجودیتی است که بدون وابستگی به یک موجودیت دیگر (موجودیت قوی) نمی تواند وجود داشته باشد. به عنوان مثال، موجودیت جزئیات سفارش به موجودیت سفارش وابسته است. بدون وجود سفارش، جزئیات سفارش معنی ندارد.

موجودیت قوی موجودیتی است که به هیچ موجودیت دیگری وابسته نیست. به عنوان مثال، موجودیت مشتری یک موجودیت قوی است.

برای تبدیل موجودیت ضعیف به موجودیت قوی، باید:

* یک کلید اصلی منحصر به فرد برای موجودیت ضعیف تعریف کرد. به عنوان مثال، می توان به موجودیت جزئیات سفارش یک شناسه عددی منحصر به فرد اختصاص داد.
* تمام وابستگی‌های موجودیت ضعیف به موجودیت قوی را حذف کرد. به عنوان مثال، می‌توان اطلاعات مربوط به سفارش را از موجودیت جزئیات سفارش به موجودیت سفارش منتقل کرد.

**توجه:**

تبدیل موجودیت ضعیف به قوی همیشه امکان پذیر نیست. در برخی موارد، وابستگی بین موجودیت ضعیف و موجودیت قوی آنقدر قوی است که جداسازی آنها غیرممکن است.

تبدیل موجودیت ضعیف به قوی می‌تواند پیچیدگی مدل پایگاه داده را افزایش دهد.

فرض کنید موجودیت جزئیات سفارش داریم که شامل اطلاعاتی مانند نام محصول، تعداد و قیمت است. این موجودیت به موجودیت سفارش وابسته است. برای تبدیل موجودیت جزئیات سفارش به موجودیت قوی، می‌توان به هر جزئیات سفارش یک شناسه عددی منحصر به فرد اختصاص داد. همچنین می توان اطلاعات مربوط به سفارش را از موجودیت جزئیات سفارش به موجودیت سفارش منتقل کرد.

این کار ما مزایایی دارد:

* امکان جستجوی مستقیم جزئیات سفارش با استفاده از شناسه
* امکان حذف جزئیات سفارش بدون نیاز به حذف سفارش
* امکان به روز رسانی جزئیات سفارش بدون نیاز به به روز رسانی سفارش
* تبدیل روابط Many: Many به Many:1

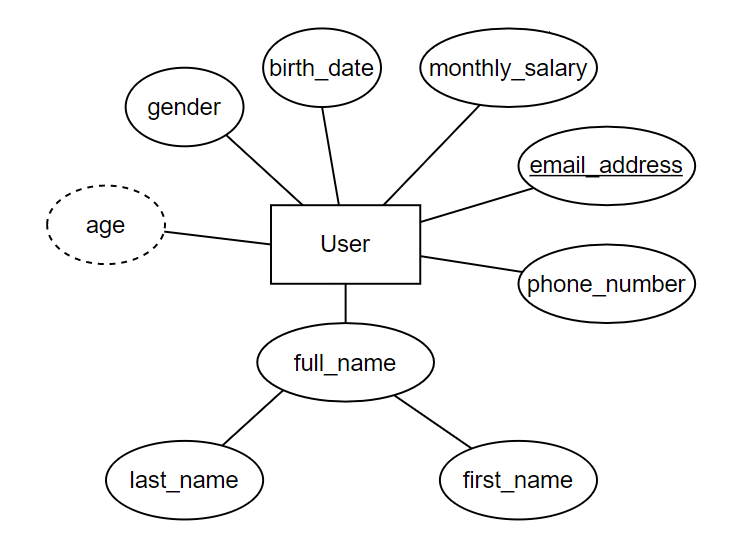
این کار ما معایبی دارد:

* افزایش پیچیدگی مدل‌سازی پایگاه داده

### **ب)**

صفت مشتق، صفتی است که در دیتابیس ما موجود و ذخیره نیست بلکه از طریق دیگر صفات موجود در دیتابیس محاسبه می‌شود.

نمونه صفت مشتق:

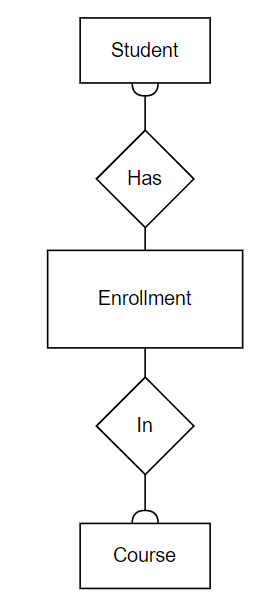
* **سن:** این صفت را می‌توان از صفت birth\_date با محاسبه اختلاف بین تاریخ تولد و تاریخ فعلی بدست آورد.

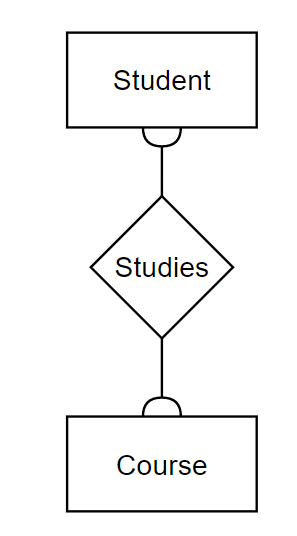
نمونه صفت ترکیبی:

* **نام کامل:** این صفت را می توان با ترکیب

صفات first\_name و last\_name با استفاده از یک فاصله ایجاد کرد.

### **ج)**

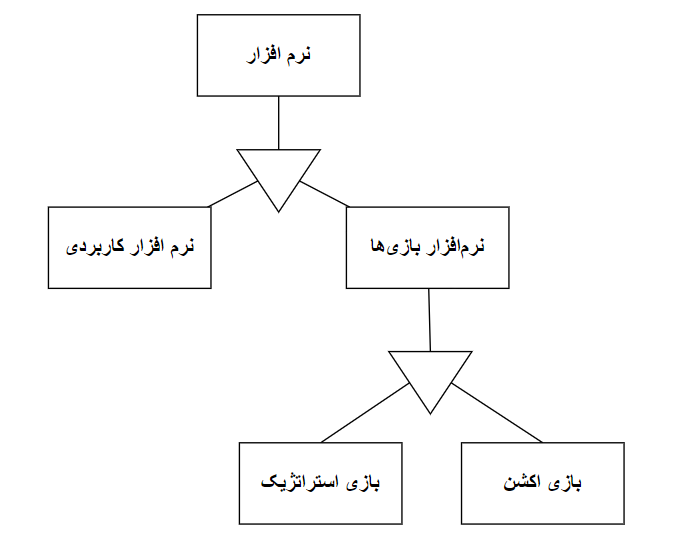
برای حذف روابط Many:Many می‌توانیم این روابط را به 2 رابطه‌ی 1:Many تبدیل کنیم. برای این کار یک موجودیت ضعیف جدید تعریف می‌کنیم که بین دو موجودیت ابتدایی ما قرار می‌گیرد. این موجودیت را Enrollment می‌نامیم.

به عنوان مثال در ابتدا نمودار روابط ما به این صورت است:

پس از تعریف موجودیت Enrollment، نمودار ما به این صورت می‌شود:

### **د)**

برای این کار 2 بار از Specialization استفاده می‌کنیم:



### **ه)**

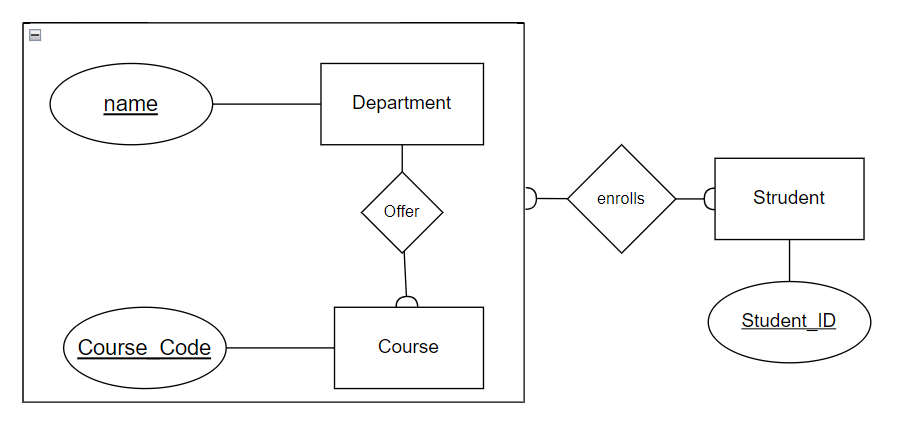
در مفاهیم مربوط به دیتابیس معمولاً از 3 نوع کلید استفاده می‌شود:

* کلید اصلی (Primary Key): به یکی از صفت‌های یک موجودیت قوی که آن موجودیت را برای ما Uniqe می‌کند، می‌گوییم. به عنوان مثال در تلگرام هر فرد یک آیدی عددی دارد که مخصوص به آن اکانت است و آن را از سایر اکانت‌ها Uniqe می‌کند. یا برای یک دانشجو، شماره‎ دانشجویی وی کلید اصلی است. این صفت در نمودار ER با یک خط زیر اسمش مشخص می‌شود.
* کلید جزئی (Composite Key): کلید جزئی یا کلید ترکیبی مجموعه‌ای از چندین فیلد است که به همراهٔ هم به عنوان کلید برای یک رکورد در جدول استفاده می‌شود. این کلید معمولاً زمانی استفاده می‌شود که یک کلید اصلی از یک فیلد واحد برای تشکیل شناسه یکتای یک رکورد کافی نباشد و نیاز به ترکیب چندین فیلد باشد. به عنوان مثال فرض کنید در یک فروشگاه، موجودیت سفارش و کالا را داریم و هرکدام یک کلید اصلی برای خود دارند. با ترکیب دو فیلد شناسه سفارش و شناسه محصول می‌توانیم رابطه بین سفارشات و محصولات را تعیین کنیم. با استفاده از این کلید جزئی، ما می‌توانیم مشخص کنیم که هر سفارش شامل کدام محصولات بوده است و همچنین هر محصول در چه سفارش‌هایی استفاده شده است. این اطلاعات می‌تواند برای ایجاد گزارشات، مدیریت موجودی و غیره بسیار مفید باشد. این صفت در نمودار ER با یک خط‌چین زیر اسمش مشخص می‌شود.
* کلید خارجی (Foreign Key): کلید خارجی یک یا چند ویژگی است که به کلیدهای اصلی موجودیت دیگری اشاره دارد و نشان‌دهنده‌ی ارتباط بین دو موجودیت در نمودار می‌باشد. این صفت در ER نمایش داده نمی‌شود. برای مثال در فروشگاه اینترنتی اگر موجودیت سبد‌خرید دارای صفت شناسه کاربر باشد، در اصل یک کلید خارجی به موجودیت کاربر است.

فرض کنید یک موجودیت به نام کاربر در شبکه اجتماعی داریم. هر کاربر دارای ویژگی‌های شماره تلفن و ایمیل و نام و... است. اگر به هر کاربر یک آیدی عددی منحصر به فرد بدهیم، به عنوان Primary Key شناخته می‌شود. در صورتی که این کار لزومی ندارد، چون ما می‌توانیم از کلید جزئی ایمیل و شماره تماس استفاده کنیم، چون این صفات هم می‌توانند هر کاربر را Uniqe کنند.

اگر بخواهیم یک مثال دیگر بزنیم، می‌توانیم به یک چاپ‌خانه اشاره کنیم که یک کتاب یا مقاله را کپی می‌کند! دو نوع موجودیت داریم، کتاب اصلی و کتاب کپی. در این موضوع به جای اینکه برای هر کپی یک آیدی درنظر بگیریم، یک کلید جزئی که شامل شناسه کتاب و نسخه‌ی کتاب است می‌دهیم تا میان بقیه‌ی کپی‌ها قابل شناسایی باشد.

### **و)**

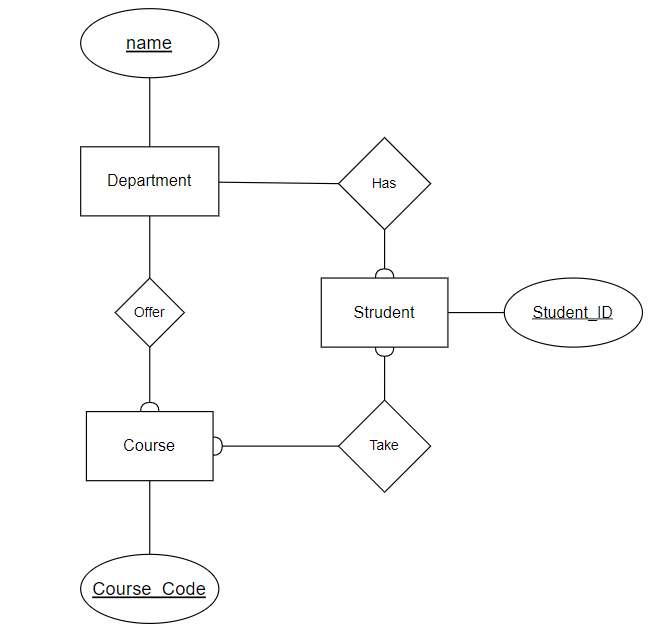
با توجه به توضیحات داده شده در صورت سوال، می‌توانیم دو موجودیت درس و دانشکده را با استفاده از مفهوم Aggregation با هم اجتماع کنیم و در اصل یک موجودیت در نظر بگیریم.

می‌توانیم رابطه‌ی Many:Many مقابل را به 2 رابطه‌ی

1:Many تبدیل کنیم که توضیحات آن در قسمت قبل

گفته شده است.

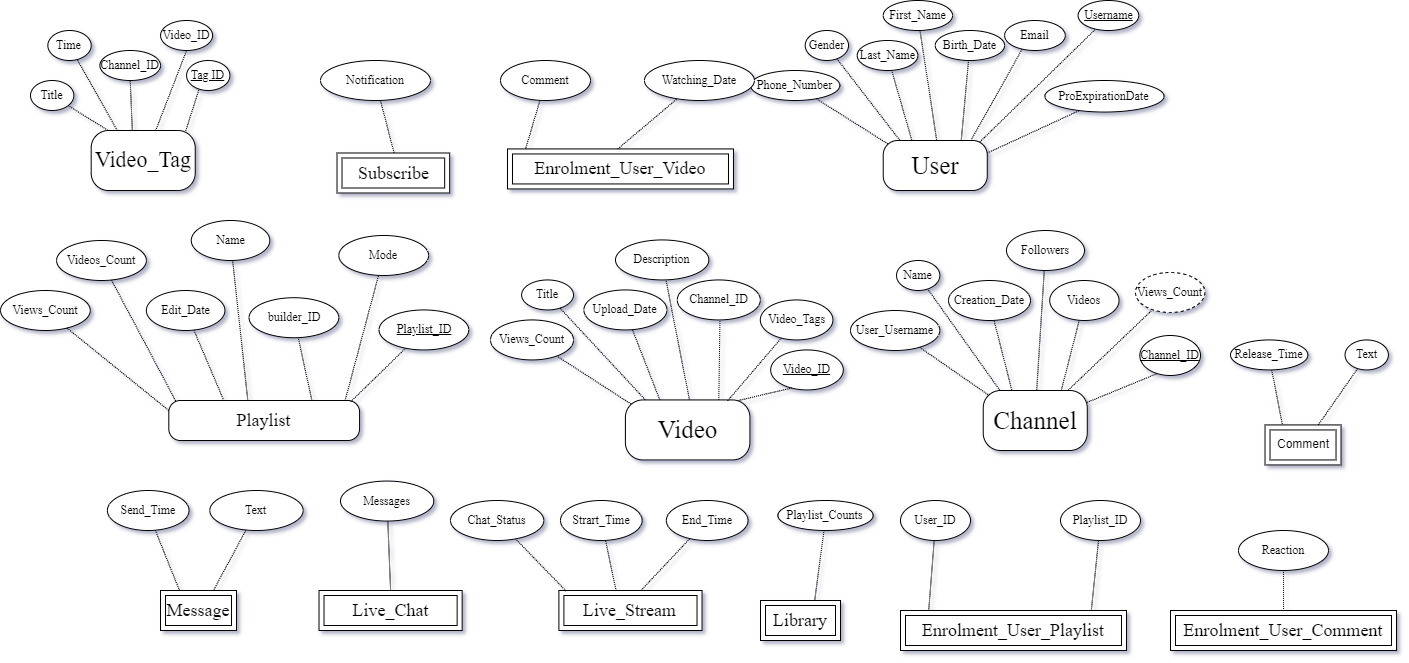
حال بدون استفاده از مفهوم اجتماع، نمودار را می‌کشیم:



## سوال شماره 2

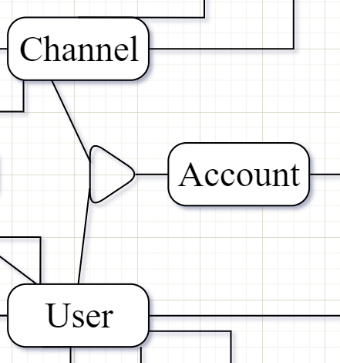
در این سوال قرار است ما یک پایگاه‌داده برای پلتفرم یوتیوب طراحی کنیم.

در ابتدا با توجه به سناریو توضیح داده شده در صورت سوال، Entityها را مشخص می‌کنیم(در این عکس فقط آن موجودیت‌هایی که دارای صفت هستند آورده شده است، برخی موجودیت‌های ضعیف داخل ERD استفاده شده‌اند که اینجا قرار ندارند.).

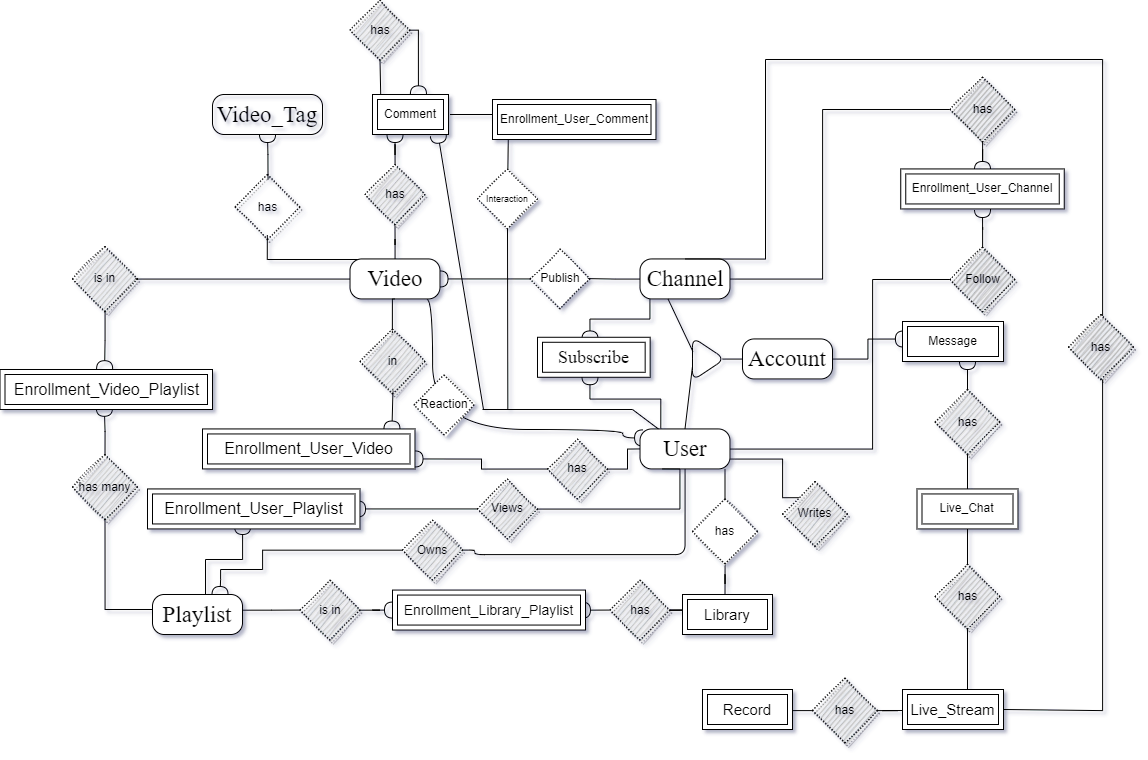


حال به سراغ روابط تعمیم و تخصیص(Specialization / Generalization) در بین موجودیت‌ها می‌رویم:

* **Account**
* **User**
* **Channel**

این موجودیت‌ها دارای رابطه Specialization هستند.

حال در ادامه روابط بین موجودیت‌ها را نمایش می‌دهیم:



**موارد اصلاح شده در تحویل دوم:**

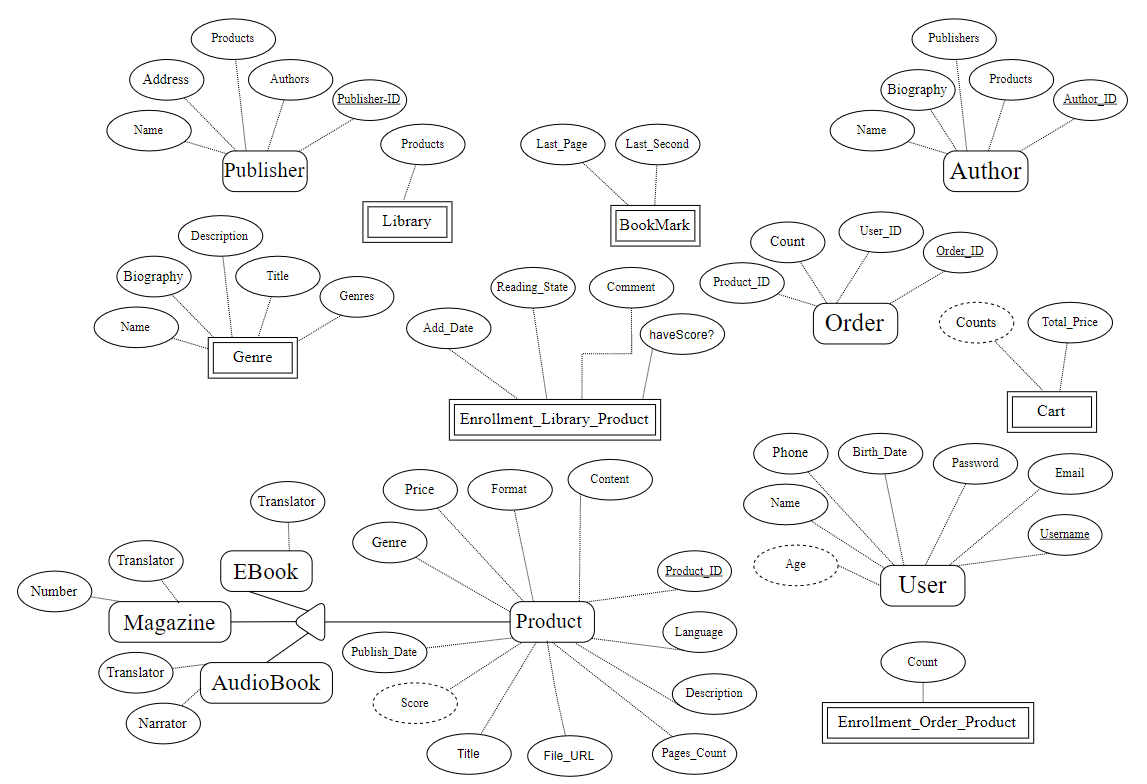
* کلیدهای خارجی مربوط به موجودیت‌های ضعیف به عنوان صفت نشان داده نشده‌اند و باید از طریق روابط موجود به آن‌ها پی برد.
* صفت مربوط به حساب کاربری ویژه به موجودیت User اضافه شد.
* آمار بازدید کانال به عنوان یک صفت مشتق به موجودیت Channel اضافه شد.
* رابطه‌ی اجتماع میان 3 موجودیت حذف شد.
* مالکیت Playlist و User با یک رابطه جدید هندل شد.
* یک رابطه برای Reaction بین User و Video برقرار شده، چون هر فرد می‌تواند فقط یک بار ری‌اکشن برود.

## سوال شماره 3:

در این سوال قرار است ما یک پایگاه‌داده برای پلتفرم فیدیبو طراحی کنیم.

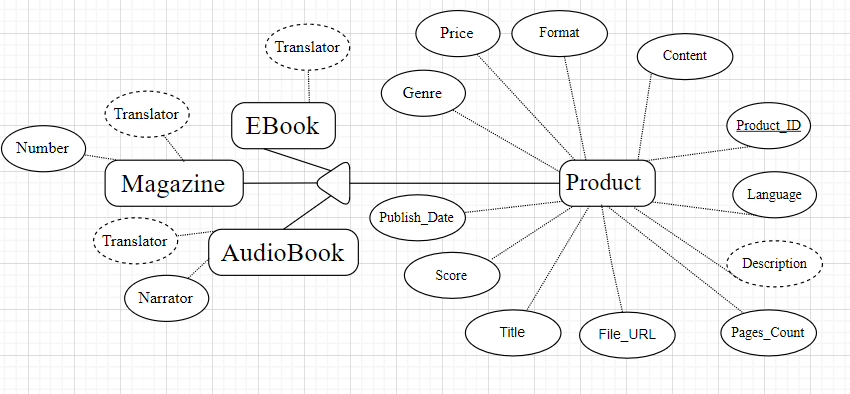
در ابتدا با توجه به سناریو توضیح داده شده در صورت سوال، Entityها را مشخص می‌کنیم(در این عکس فقط آن موجودیت‌هایی که دارای صفت هستند آورده شده است، برخی موجودیت‌های ضعیف داخل ERD استفاده شده‌اند که اینجا قرار ندارند.)

در عکس زیر تمامی Entityها به همراه Attributeهایشان آمده است:



حال به سراغ روابط تعمیم و تخصیص(Specialization / Generalization) در بین موجودیت‌ها می‌رویم:

* **Product**
* **EBook**
* **Magazine**
* **AudioBook**

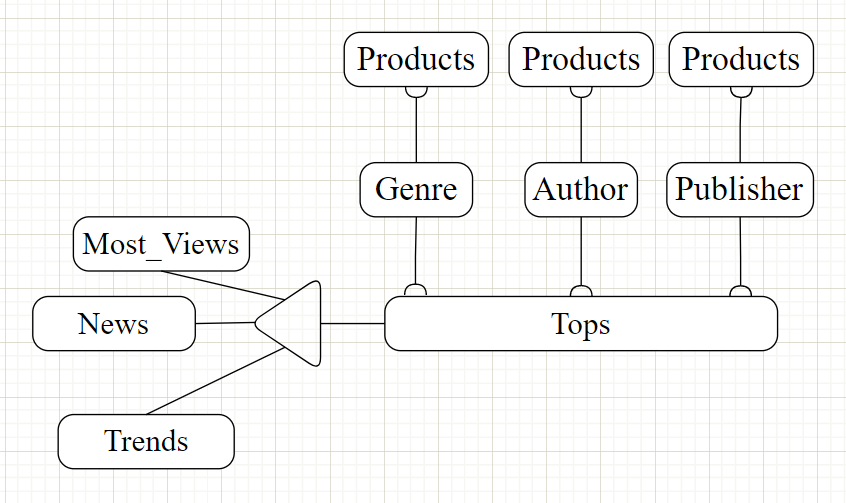
این موجودیت‌ها دارای رابطه‌ی تخصیص (Specialization) هستند.

در ERD موجود در صفحه‌ی بعد، Weak Entity **با مستطیل دولایه** نمایش داده شده‌اند.

**نکته‌ی مهم:**

در سناریوی صورت سوال، برخی صفات مد نظر است که نیاز نیست برایشان Attribute جدا تعریف کنیم. آن صفات، صفت مشتق(Derived attributes) هستند و از باقی موارد قابل بدست آمدن هستن.(برای اینکه نمودار شلوغ و کثیف نشود، این بخش جدا رسم شده است.)

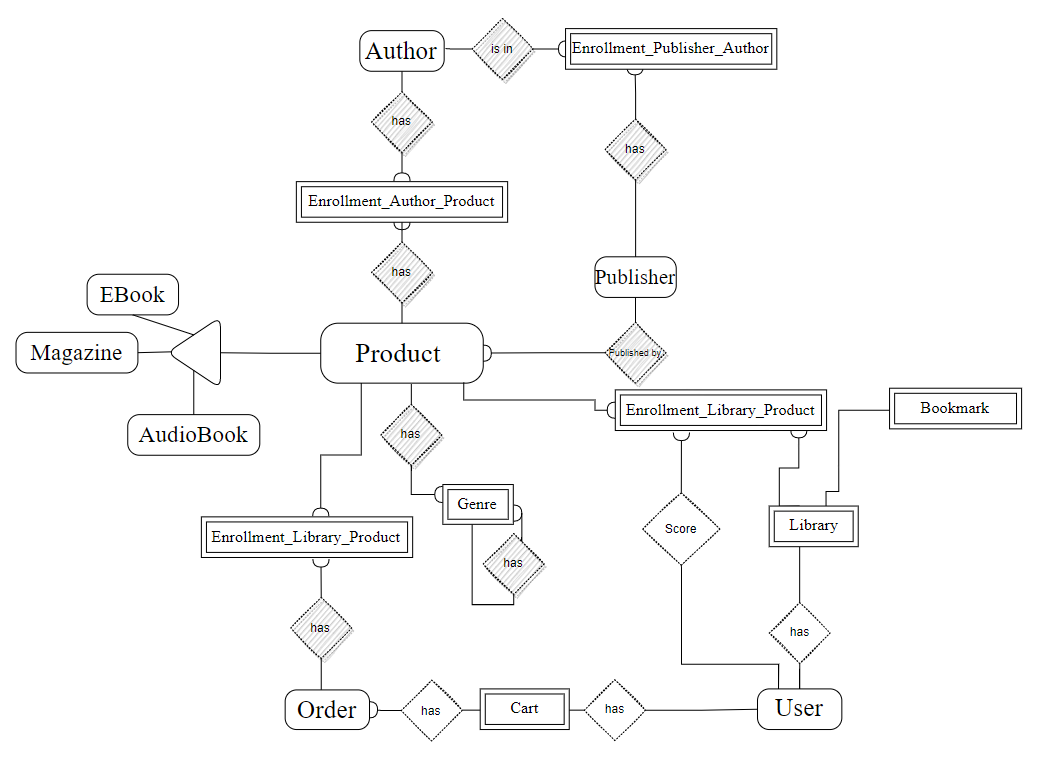
به عنوان مثال، لیست کتاب‌های محبوب این هفته نیاز به ذخیره‌سازی ندارند و از روی روابط زیر قابل دست‌یابی هستند:



اگر لیست کتاب‌های محبوب هر ژانر، نویسنده و یا انتشارات را بخواهیم، کافیست ازطریق آن به Productهایی که با آن رابطه دارند دست یابیم و از طریق Simple Attributeیی به نام Score، آن‌ها را شناسایی کنیم و به Tops بدهیم.

بعضی از صفت‌ها هم از نوع Composite هستند، می‌توان آن‌ها با ترکیب‌ کردن صفت‌های ساده‌ای که داریم، بدست آوریم.

به عنوان مثال: اسم کامل یک نویسنده.

حال در ادامه روابط بین موجودیت‌ها را نمایش می‌دهیم: