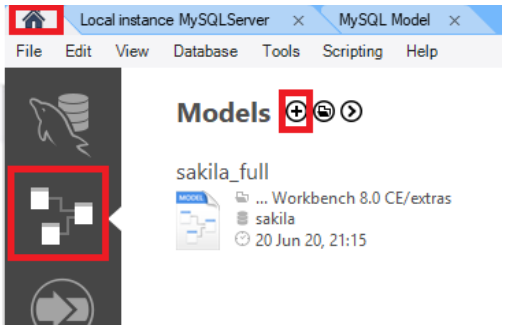
1. Veritabanı Diyagramı ile Çalışma

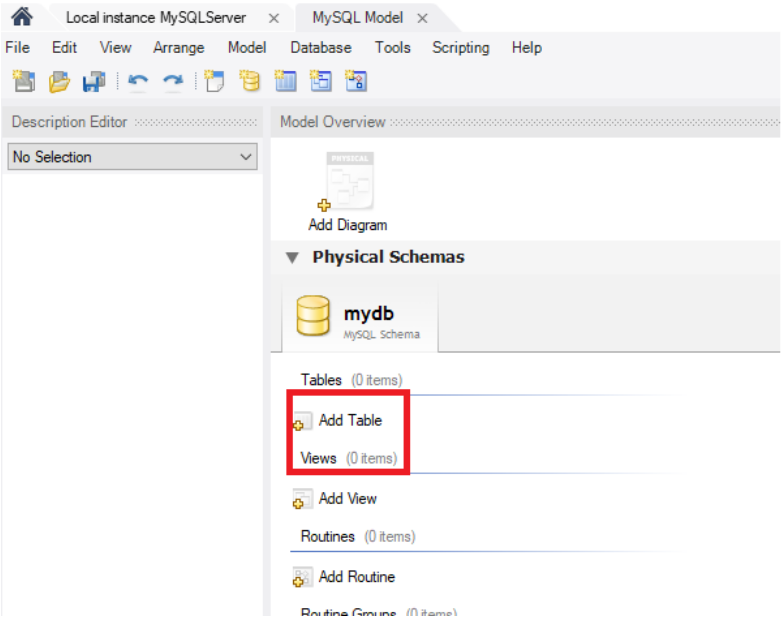
Biliyoruz ki Workbench IDE'si üzerindeki schemas kısmına sağ tıklayıp create new schema diyerek yeni bir database'i server'a ekleyebiliriz. Zaten shopapp database'ini böyle yaratmıştık.

Buna alternatif olarak database'i yaratmadan önce modelini yaratabiliriz ve bu modeli kullanarak bu database'in altındaki tablolar nelerdir, özellikleri nelerdir, aralarındaki bağlantılar nelerdir bunları tanımlarız, daha sonra modeli kullanarak database'i yaratabiliriz. Bunu yapalım.

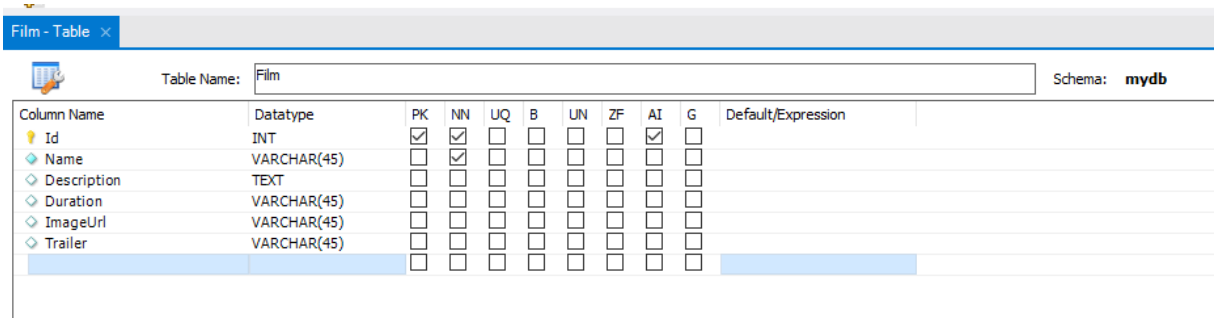


Home üzerinden models sekmesine tıklıyorum ve + butonu ile yeni bir model eklemek istediğimi belirtiyorum.

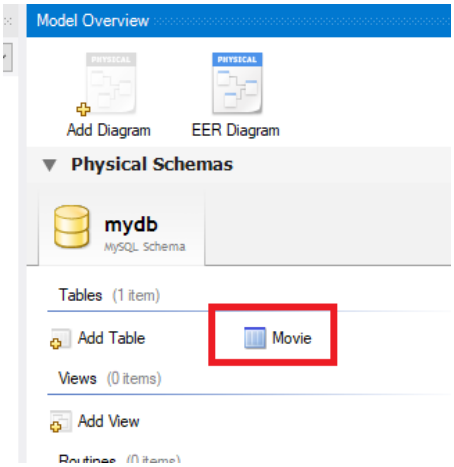
Daha sonra karşımıza aşağıdaki gibi bir sekme çıkacak buradan Add Table diyerek modelimize yeni tablo ekleyebiliriz.



Add Table dedikten sonra aşağıdaki gibi bir Film tablosu oluşturduk.



Bu tabloyu kapattığımızda modele kaydolmuş olacak.



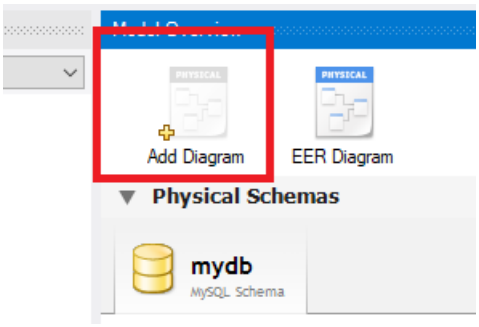
İstersek bu tabloya sağ tıklayıp ismini değiştirebiliriz, zaten yukarıdaki fotoğrafta Film ismi Movie olarak değiştirilmiş.

Buradan aynı şekilde add table ile modele yeni tablolar ekleyebiliriz ve onları güncelleyebiliriz.

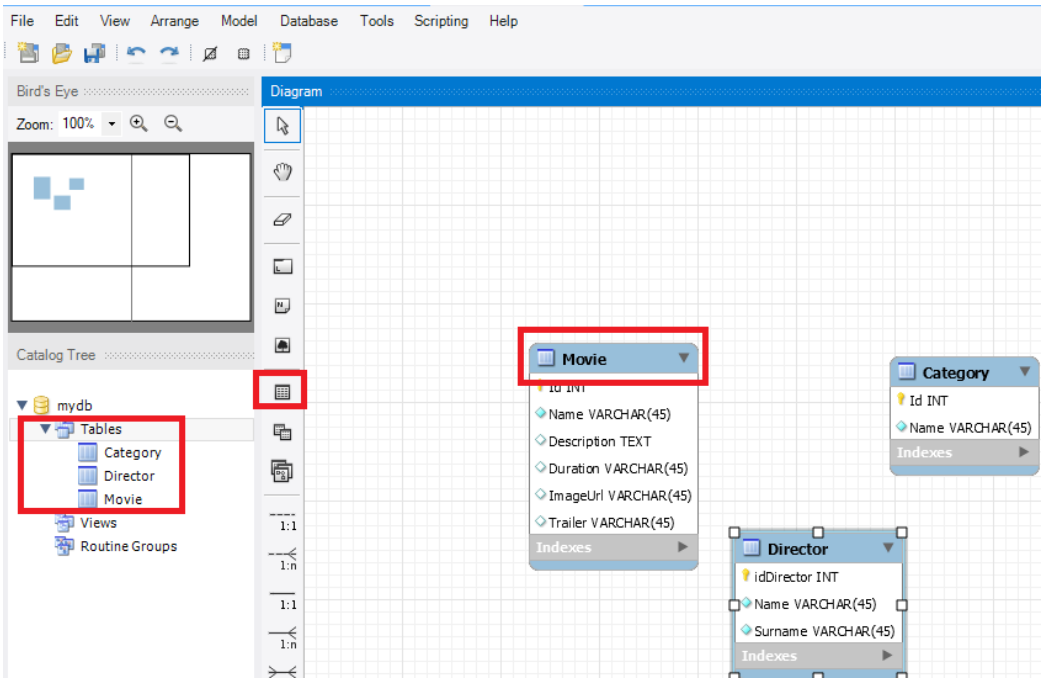
Ancak buna alternatif ve daha kolay bir yol da var ki o da Add Diagram'ı kullanmak.

Alternatif Modele Tablo Ekleme Yolu

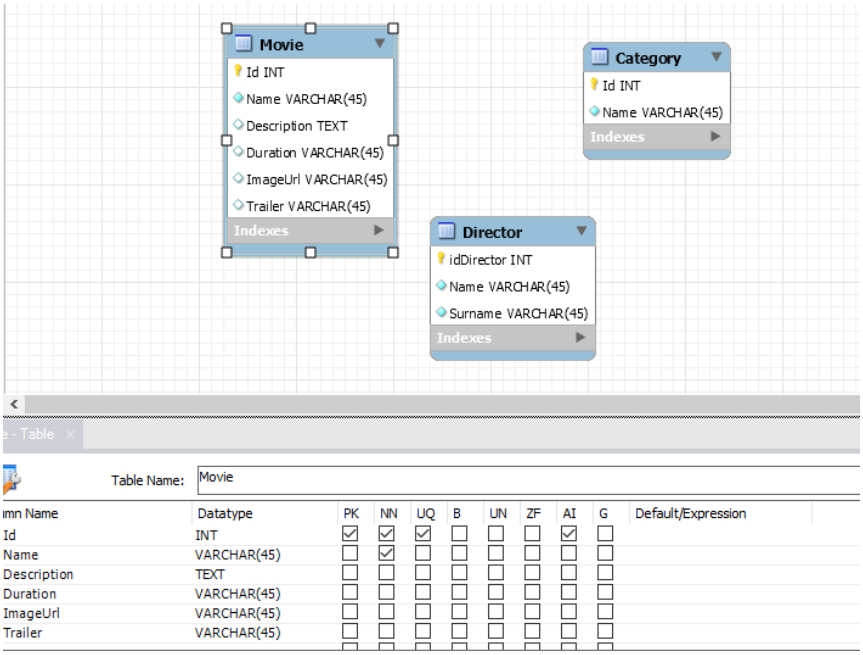
Home>Models>+ dedikten sonra add table yerine Add Diagram diyebiliriz böylece interactive bir diagram sekmesine gideriz. Bu sekmeden de tablo ekleme ve güncelleme işlerini rahatlıkla yapabiliriz. Üstelik tablolar arasındaki ilişkileri de bu sekmede tanımlayacağız.



Burada aşağıdaki gibi bir diagram sayfası göreceğiz, tables altından istediğimiz table'ı alana sürekleyip bırakabiliriz, eğer yeni tablo eklemek istersek paneldeki tablo işaretine tıklarız ve alana tıkladığımızda yeni tablo eklenir. Tabloyu doldurmak veya güncellemek istersek üzerine iki kez tıklamamız yeterli olacaktır.



Örneğin Movie tablosuna iki kere tıklayınca, aşağıdaki gibi tablo içeriği açılıyor ve burdan update işlemi yapılabilir.

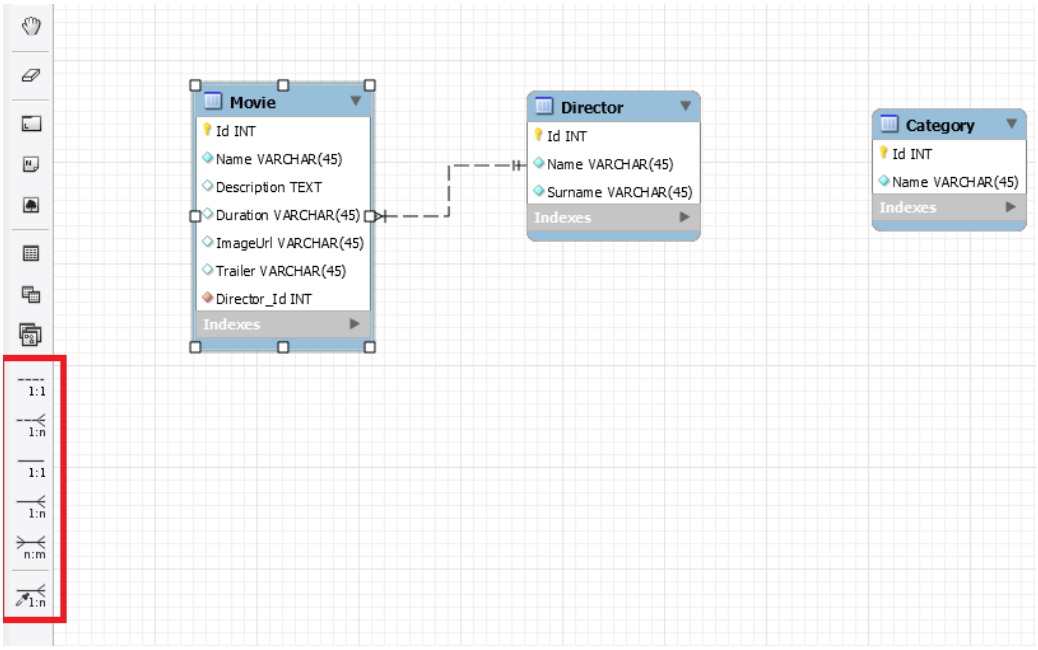


2. Tablolar Arası Fiziksel Bağlantıların Yapılması

Önceki kısımda bir diagram oluşturduk ve şuanda diagramda 3 farklı tablomuz var, elbette gerçek bir uygulamada çok daha fazla tablo olacaktır, bu sebeple farklı tablo bağlantılarını farklı diagramlarda gerçekleştirebiliriz.

Şimdilik önceki kısımda oluşturulan diagram üzerinde tablolar arası bağlantıları oluşturalım ve böylece bu bağlantıları daha iyi anlamaya çalışalım.

Bağlantıları oluştururken kullanılacak panel aşağıda kırmızı ile gösterilen panel.

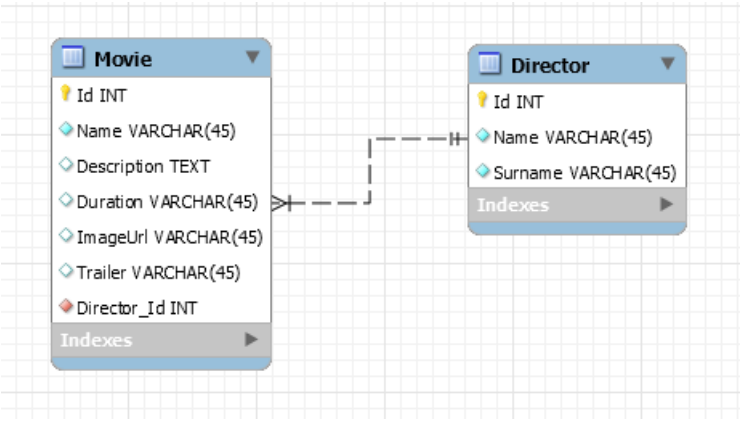


Bu panelden ilgili bağlantıyı seçeceğiz ve tablolara tıklayarak bu bağlantıları sağlayacağız.

1:n non identifying relationship

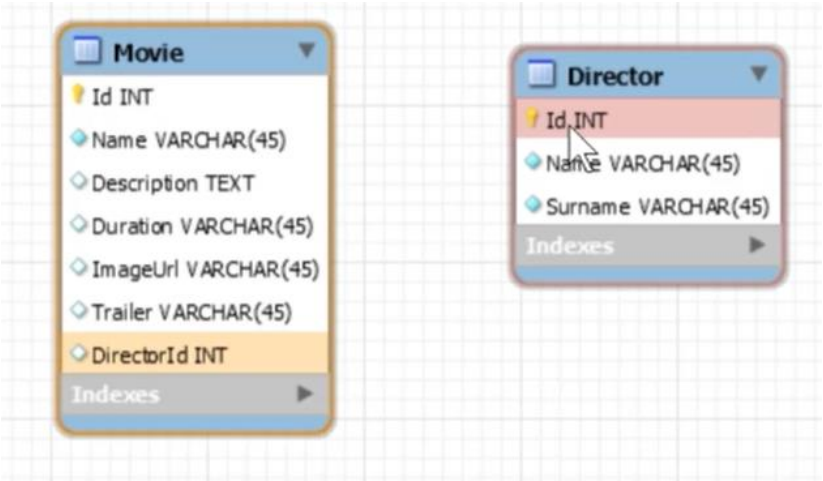
Diyelim ki bir filmin tek bir yönetmeni olsun ve elbette bir yönetmenin birden fazla filmi olabilir o halde bu ikisi arasında one to many yani 1:n bir bağlantı olacak.

Burada 1:n non identifying kısmına tıklayıp, önce movie tablosuna sonra director tablosuna tıkladığımızda movie tablosuna director_id şeklinde foreign key'i ekler ve iki tablo arasındaki 1:n ilişki tanımlanmış olur.



1:n using existing columns relationship

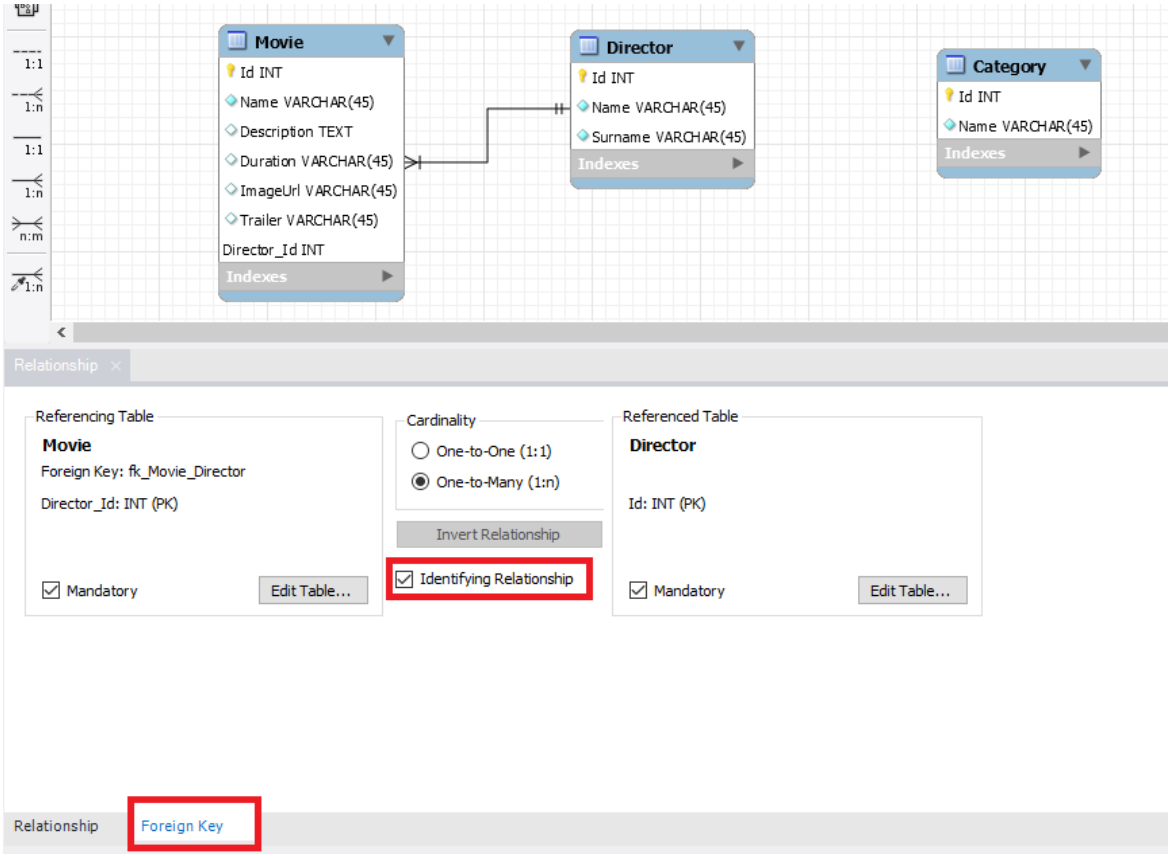
Foreign key'i kendimiz eklemek ve bağlantıyı ona göre yapmak istersek, panelin en altındaki 1:n using existing columns'u seçeriz ve movie tablosunda önceden yaratılmış foreign key'e tıklayıp sonra director tablosunda primary key'e tıklarız ve yine iki tablo arasındaki 1:n ilişki kurulmuş olur.



1:n identifying relationship

Panelde her ilişki için bir kesikli bir de kesikli olmayan çizgi olduğuna dikkat et, kesikli olmayan çizgi identifying relationship'i temsil eder.

Aşağıda bu örnek verilmiş. İlişki çizgisinin üstüne tıklayıp foreign key dersek buradan identifying check butonunu değiştirerek ilişkinin tipini de değiştirebiliriz.



Peki ama nedir bu identifying ve non-identifying relationship?

Non-Identifying İlişki Açıklama:

Diyelim ki bir kütüphane database'i içerisinde bir kitap tablosu var ve kitabın da bir okuyucusu var, ayrıca bir okuyucu tablosu da olsun aynı movie ve director tabloları gibi 1:n bir ilişki var.

Kitabı eğer biri okuyorsa kitabın okuyucuId kısmına okuyucunun id'si işaretleniyor ancak her kitabı her zaman biri okuyacak diye bir kaide yok.

Yani her kitabın okuyucuId kısmı her zaman bir okuyucuId ile dolu olmak zorunda değil bu kısım boş da olabilir. İşte bu ilişkinin tipi Non-Identifying olmalıdır.

Yani okuyucuId foreign key'i dolu olmasa da kitap varolabilir.

Identifying İlişki Açıklama:

Yine kitaptan yola çıkarsak bir kitabın mutlaka yazarının olması gerekir, kitabın yazarId foreign key'i boş olamaz çünkü mantık olarak kitabın mutlaka biri tarafından yazılması gerekmektedir.

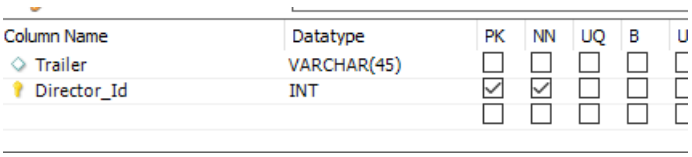
İşte bu da identifying ilişkiyi tanımlar.

Benzer şekilde eğer ben bir movie kaydı eklediğimde kesinlikle ve kesinlikle bir director'ı olsun istiyorsam identfying bir 1:n ilişkisi tanımlamalıyım.

Yok eğer bir film director'ı belirtilmeden de var olabilsin veya director tablosu dışındaki bir directorId de bir filme verilebilsin istiyorsak o halde non-identifying 1:n ilişki tanımlayamalıyız.

Identfying – Non Identifying ve NN alanı Açıklama

Şimdi örneğin movie tablomuzun director_id kısmı Not Null olarak belirtilmiş yani yeni bir movie kaydı oluşturulurken mutlaka bir directorId atanmalı.



Column Name	Datatype	PK	NN	UQ	B	U
Trailer	VARCHAR(45)	☐	☐	☐	☐	☐
Director_Id	INT	☑	☑	☐	☐	☐
		☐	☐	☐	☐	☐

Eğer movie-yönetmen ilişkisi non-identifying ise director_id'den NN kısmını kaldırrız ve bu kısma NULL atayabiliriz, veya NN seçiliyken de 0 gibi yönetmen tablosunda varolmayan bir id'yi buraya atayabiliriz.

Ancak anladığım kadarıyla aradaki ilişki identifying ise burası NN olsun olmasın bizim kesinlikle yeni bir movie kaydı eklerken yönetmen tablosunda var olan bir id kullanmamız şart!

Yani NN alanı ile Identifying-NonIdentifying ilişkisi aynı şey değil, NN denilen alana kesinlikle bir değer atanmalı diyoruz ancak iki tablo arasındaki identfying ilişkide bir değer atanması yetmez üzerine bir de bağlantılı olduğu tablodaki id'lerden biri atanmalı şartı koşuyoruz.

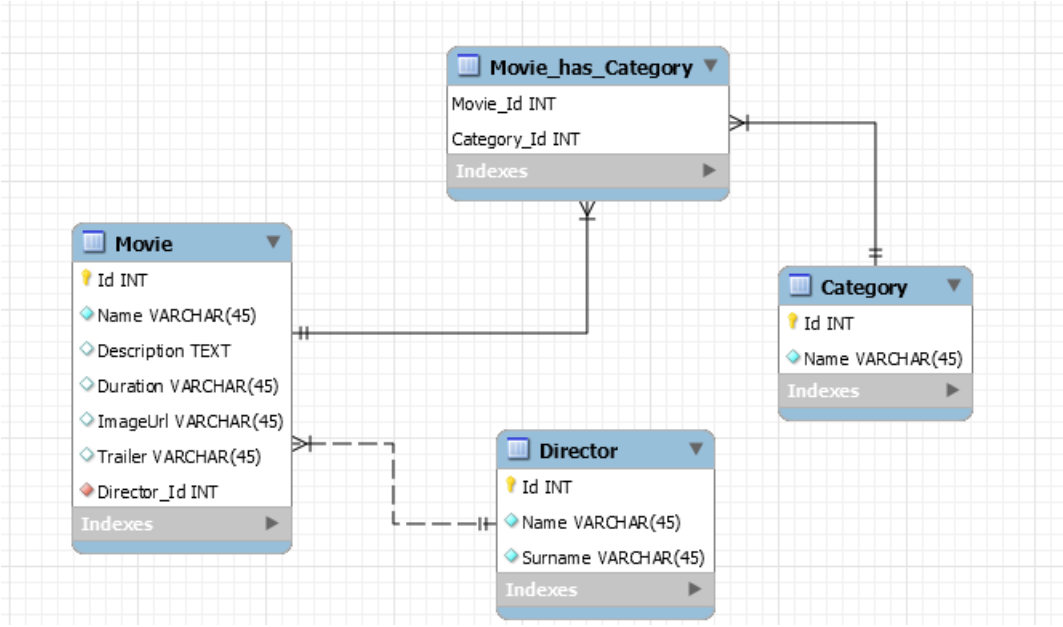
Bağlantıyı NonIdentfying yaparsak filmlere ilk başta rasgele yönetmen Id'ler atayıp daha sonra yönetmen tablosunu oluşturduktan sonra ilgili yönetmenlere atama yapabiliriz ancak identfying ilişki buna izin vermez, her bir movie yaratılırken mutlaka yönetmeni hazır olmalıdır.

n:m relationship

Şimdi movie tablosu ile category tablosu arasında bir ilişki kuralım, bu ilişki many to many olacak çünkü bir filmin birden fazla kategorisi olabilsin istiyoruz benzer şekilde zaten bir kategorinin birden fazla filmi olacak.

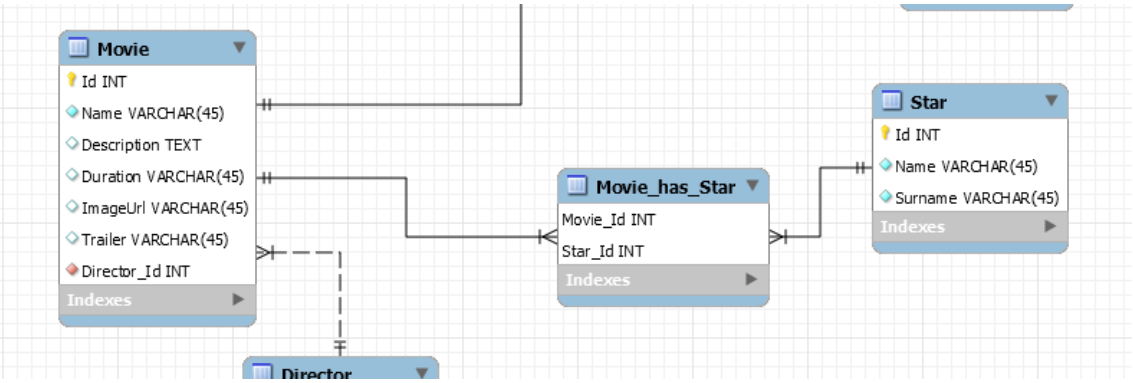
Hatırlarsak iki tablonun many to many ilişkisini tutabilmek için üçüncü bir tablo kullanıyorduk.

Burada diagram bizim için otomatik olarak 3. Tabloyu yaratacaktır.



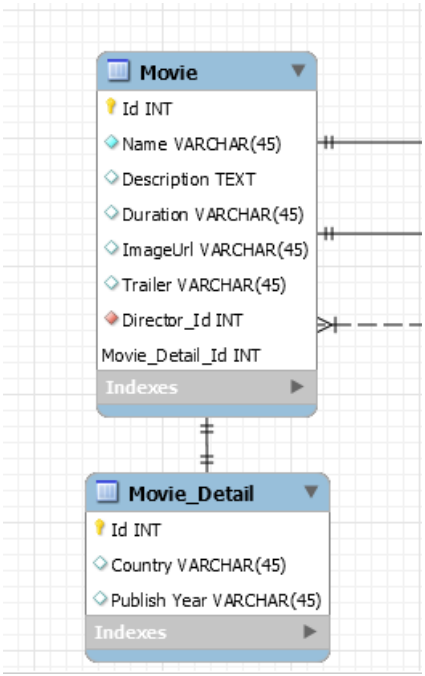
Buradaki ilişkinin **IDENTIFYING** olduğuna dikkat et, çünkü ben bu 3. Tabloda her zaman var olan bir movieId ile varolan bir CategoryId'yi kullanmak isterim, varolmayan bir movie ile varolmayan bir category'i eşleştirmek pek mantıklı değil.

Benzer şekilde oyuncuları tutan bir star tablosunu databse'e ekleyip movie ile many to many ilişkisi içine sokabiliriz:



1:1 relationship

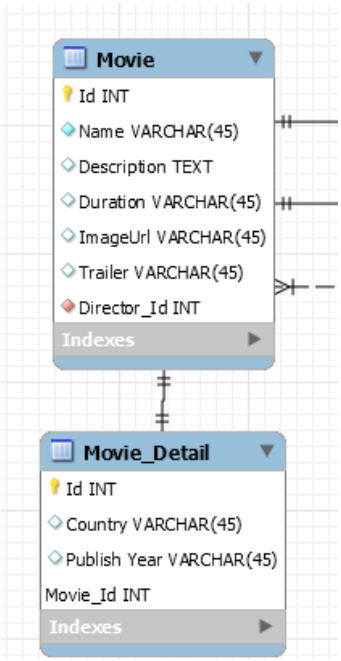
Şimdi bir movie detail tablosu oluşturduk ve bu tabloda detaylı movie bilgilerini tutmak istiyoruz. Her bir movie'ye karşılık bir movie_detail olacak, benzer şekilde her detail için de bir movie olacak aynı tablonun ikiye bölünmüş hali gibi.



Yukarıda böyle bir tablo oluşturuldu ve **1:1 IDENTIFYING** bir bağlantı yapıldı. Peki bu ilişkiyi yukarıdaki şekilde tanımlamak doğru mu?

Yanlış! Çünkü eğer böyle tanımlarsak, yeni bir movie kaydı tanımlarken kesinlikle bir movie_detail id'si movie'ye verilmeli iyi de movie'yi tanımlamadan detail'ini nasıl tanımlayayım bu biraz mantıksız.

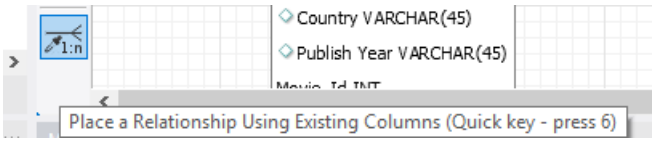
O halde bu ilişkiyi tersine çevirebiliriz, yani önce movie_detail tablosuna sonra movie tablosuna tıklarız ve foreign key movie tablosunda değil detail tablosunda oluşur. Böylece movie'yi detay tanımlamadan tanımlayabiliriz buna karşılık, bir movie detail tanımlandığında kesinlikle movie'sini de belirtmeliyizdir!



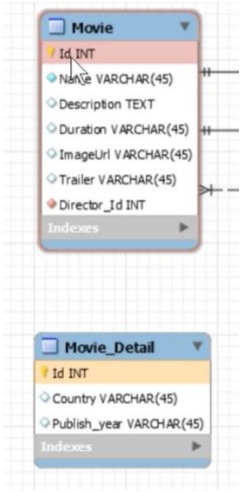
Artık movie detail tablosuna bir kayıt eklerken mutlaka movie tablosunda var olan bir movieId eklemeliyiz aksi halde hata alırız. Buna karşılık yeni bir movie tanımlarken bir detay tanımlamak zorunda değilizdir.

Ayrıca detail tablosunda MovieId foreign key'ini unique olarak ayarlarız ki her movie detail unique bir filmin detail'i olsun.

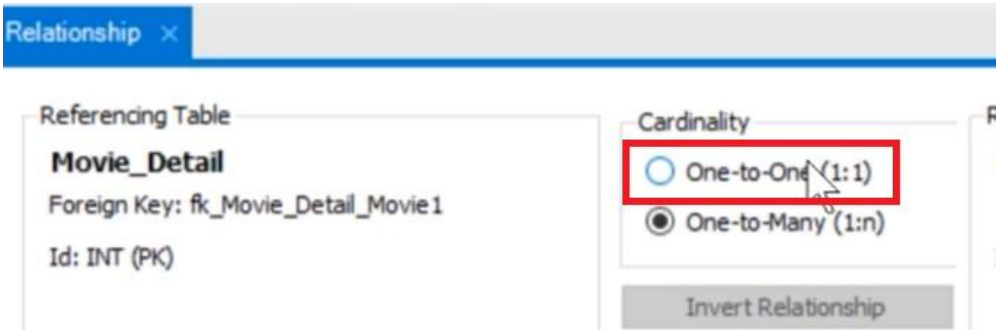
Son olarak yukarıda verilen 1:1 ilişkisi başka şekilde de verilebilir panelin en altındaki 1:n using existing columns seçilir



Daha sonra movie detail'in id'si ile movie id'si birbirine bağlanır.



Böylece movie detail'in id'si ile movie id'si birbirine bağlanır. Bu bağlılığın türü şuanda 1:n olduğu için ilişkiye tıklayıp bunu 1:1 olarak değiştiririz:

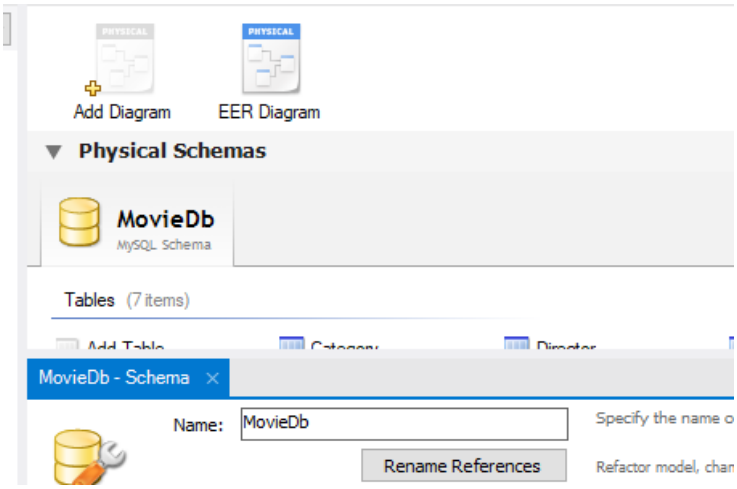


Bu yöntemle movie detail tablosuna ayrı bir foreign key belirtmeden 1:1 ilişki kurulmuş oldu, movie tablosuna eklenen bir filme detail girerken movie_detail tablosunda aynı id'yi kullanacağız.

Modeli Tamamladık

Modele bir çok tablo ekledik, tabloların içeriklerini ve birbirleri arasındaki bağlantılarını tanımladık bunu yeni yarattığımız EER ismindeki diagram üzerinde yaptık.

Bu modelin ismini de aşağıdaki sekmeden değiştirebiliriz, mesela MovieDb dedik.

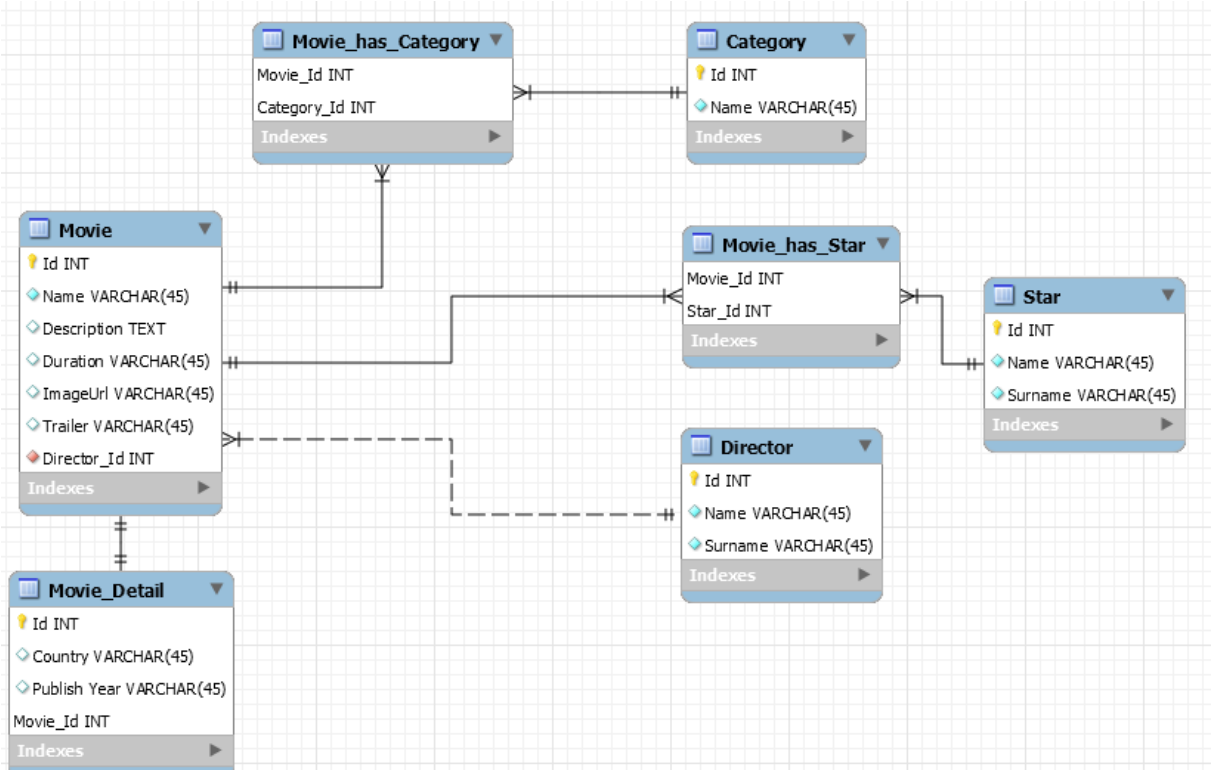


Ancak unutmayın bu model henüz hala bir model olarak tutuluyor, server üzerinde böyle bir database henüz oluşturulmadı.

Bu modeli File > Save diyerek kaydedelim ve sonraki derste bu modeli kullanarak database oluşturacağız.

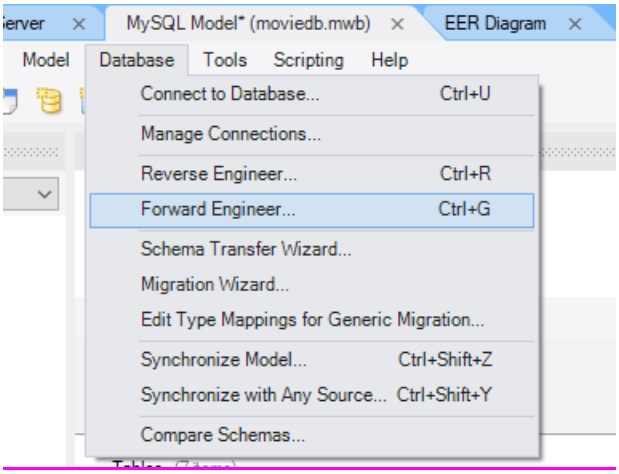
3. Veri Tabanı Şemasının Oluşturulması – Forward Engineering

Geçen kısımda bir model oluşturduk, model içerisinde tablolar oluşturduk, bu tablolar arası bağlantıları oluşturduk:

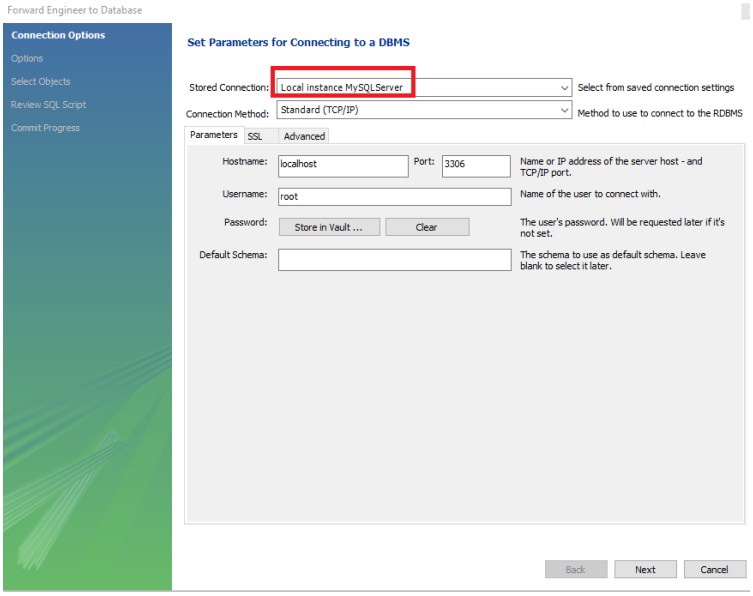


Şimdi bu modelden somut bir database oluşturalım.

Model sekmesinde iken Database > Forward Engineering seçeneğini seçeriz

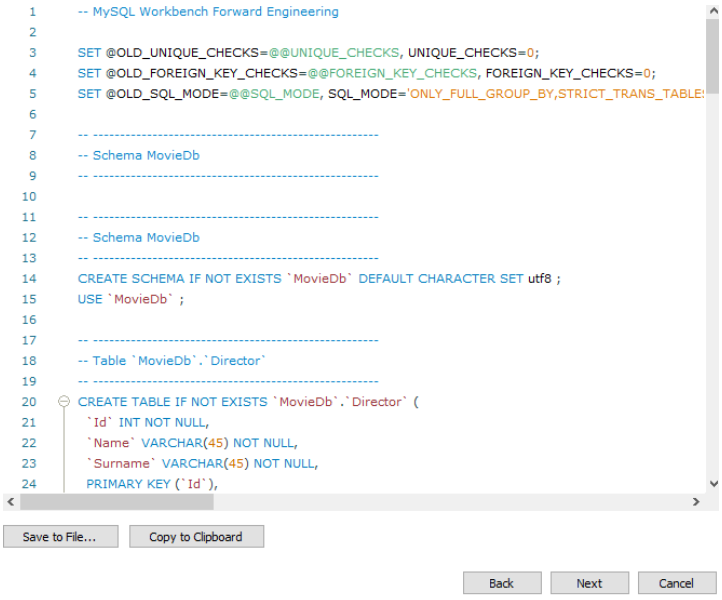


Aşağıdaki gibi bir sayfa açılacak, burada connection'u local instance olarak belirliyoruz, oluşacak database'imiz localhost altında oluşacak.

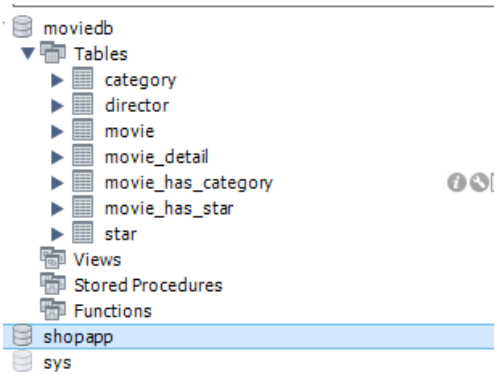


Daha sonra next diyerek, ilerliyoruz, bir sekmede aşağıdaki gibi modelde kaç tane tablo olduğunu görüyoruz.

Son olarak da ilgili modeli baz alarak serverde bir database yaratmak için gerekli kodlar IDE tarafından oluşturulur ve bize gösterilir:



Next diyerek ilgili database'i server üzerinde oluştururuz!!!



Böylece modelimizi kullanarak bir database oluşturmuş olduk!

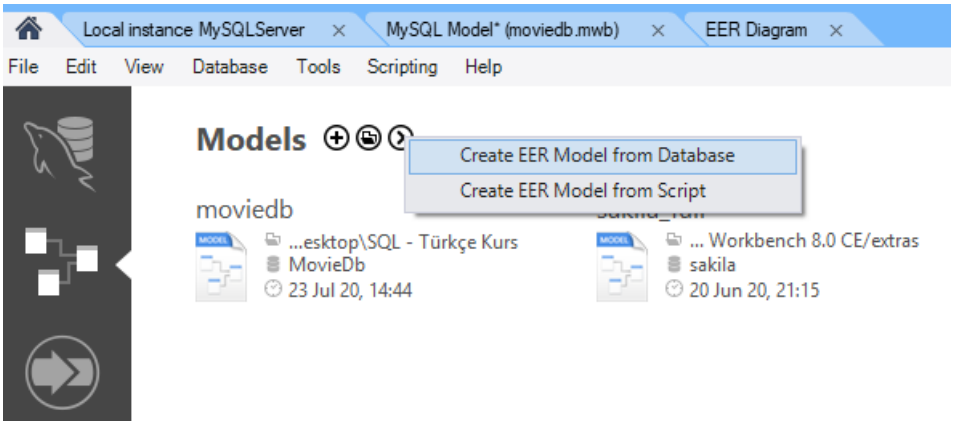
4. Reverse Engineering – Database'den Modele Geçmek

Modelden database oluşturmayı gördük, şimdi ise database'den nasıl model oluşturabileceğimize bakacağız. (Sadece Şema bilgisi taşınacak, data taşınmayacak)

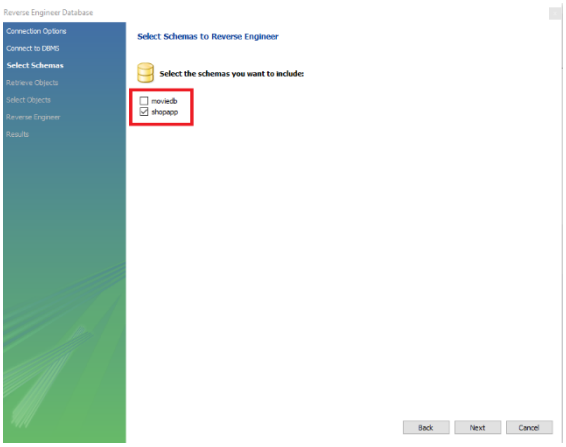
Amacımız daha önce yarattığımız shopapp database'inden bir model elde etmek.

Hatırlarsak shopapp database'inde yalnızca bir adet table'ımız var. Bunun da için 5-10 arası kayıt var.

Home > Models > Ok işareti 'ne tıklayıp Create EER Model from Database diyoruz.

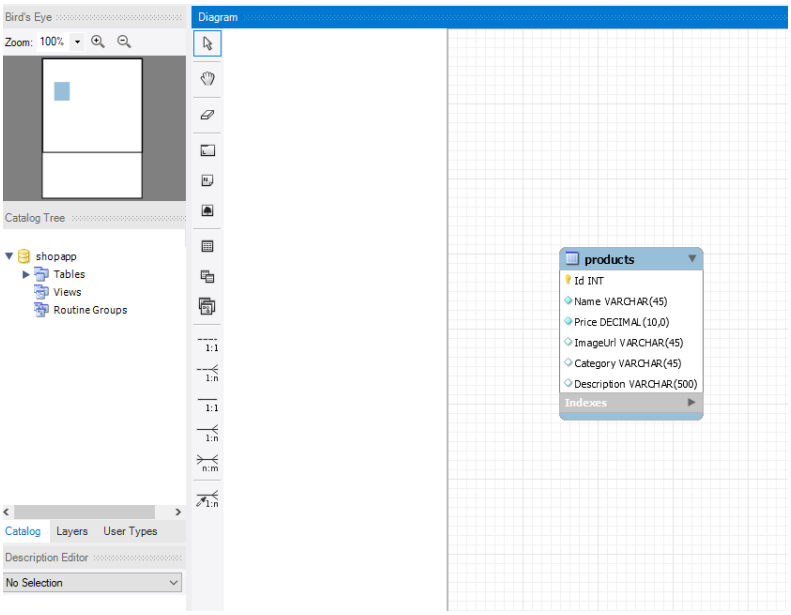


Açılan pencereden Local Instance'ı seçiyoruz çünkü local server üzerindeki bir database'i almak istiyoruz, next, next şeklinde ilerliyoruz

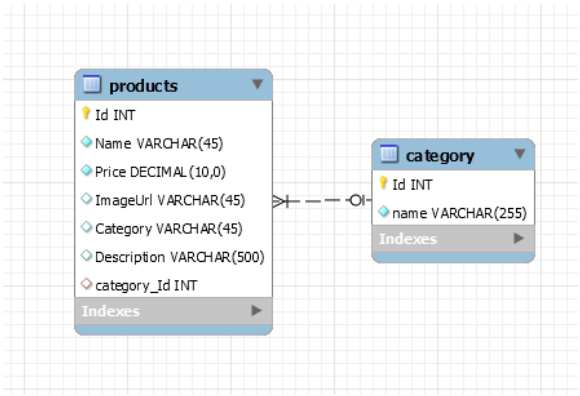


Yukarıdaki gibi bir menüden hangi databse'i modele çevirmek istediğimizi seçiyoruz.

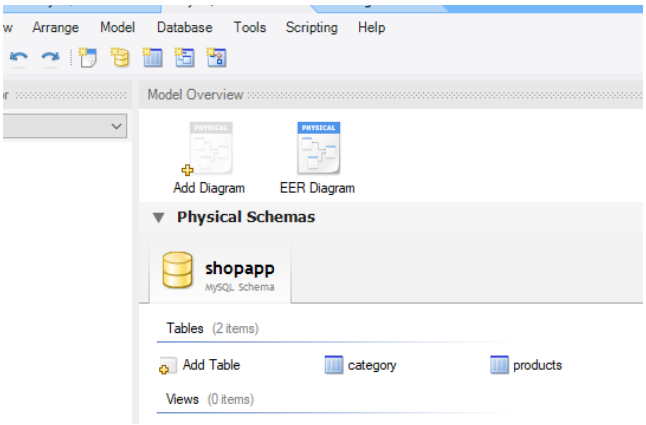
Next, next, next ilerliyoruz sonuçta aşağıdaki gibi bir model diagramı oluşturuluyor gördüğümüz gibi shopapp modeli içerisindeki products tablosu ekranda.



Şimdi diyagrama bir category tablosu daha ekledim, ikisi arasındaki ilişkiyi tanımladım:

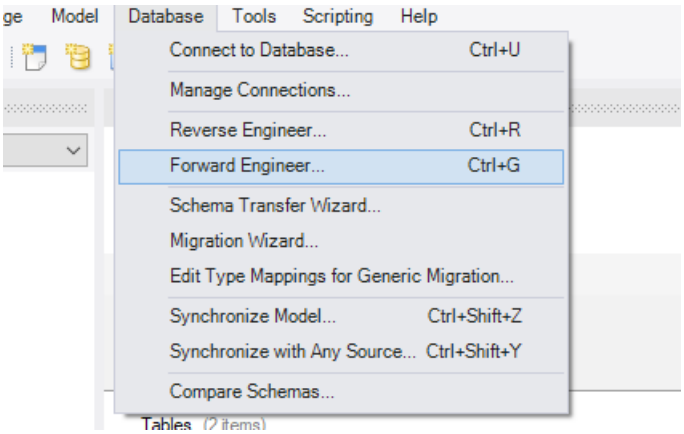


Şimdi modele baktığımızda isminin shopapp olduğunu görüyoruz.

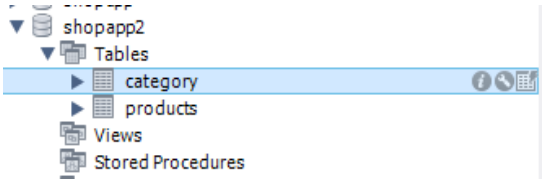


Ancak bu hali ile bu modele forward engineering uygularsam, yeni model eski shopapp database'ini override eder, biz şuan bunu istemiyoruz o yüzden bunun ismini değiştirip öyle forward engineering uygulayacağım.

Şimdi databse altından forward enginering'I seçerek önceki kısımda yaptığımız gibi bu modeli database olarak serverda oluşturacağım.



Sonuçta shopapp2 database'I server üzerinde oluşturulmuş olur, altında products ve category olarak 2 tablo olacak.



Products tablosunun yapısı direkt olarak daha önce oluşturulan shopapp database'inden reverse engineering yapılarak elde edildi.

ANCAK UNUTMA REVERSE ENGINEERING ILE ELDE EDILEN SEY DATABASE'IN ŞEMASIDIR. SHOPAPP DATABASE'İNİN İÇİNDE PRODUCTS TABLOSUNDA DAHA ÖNCE OLUŞTURDUĞUMUZ 10'A YAKIN KAYIT MODEL OLUŞTURURKEN TAŞINMAZ, YALNIZCA TABLOLARIN YAPILARI VE İLİŞKİLERİ TAŞINIR!

BU DATA TAŞIMA İŞLEMİNİ ÖNÜMÜZDEKİ BAŞLIKTA YAPACAĞIZ.

5. Veri Tabanındaki Bilgilerin Saklanması – Generate Scripts

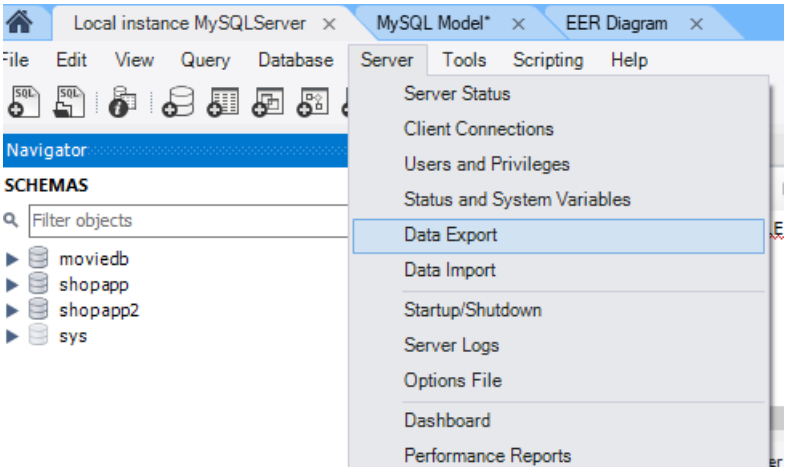
Önceki dersimizde reverse engineering ile varolan database'in model şemasını elde ettik ve bu model şemayı kullanarak forward engineering ile yeni bir database elde ettik.

Ancak bu yeni elde edilen shopapp2 database'inin içine herhangi bir kayıt girilmedi.

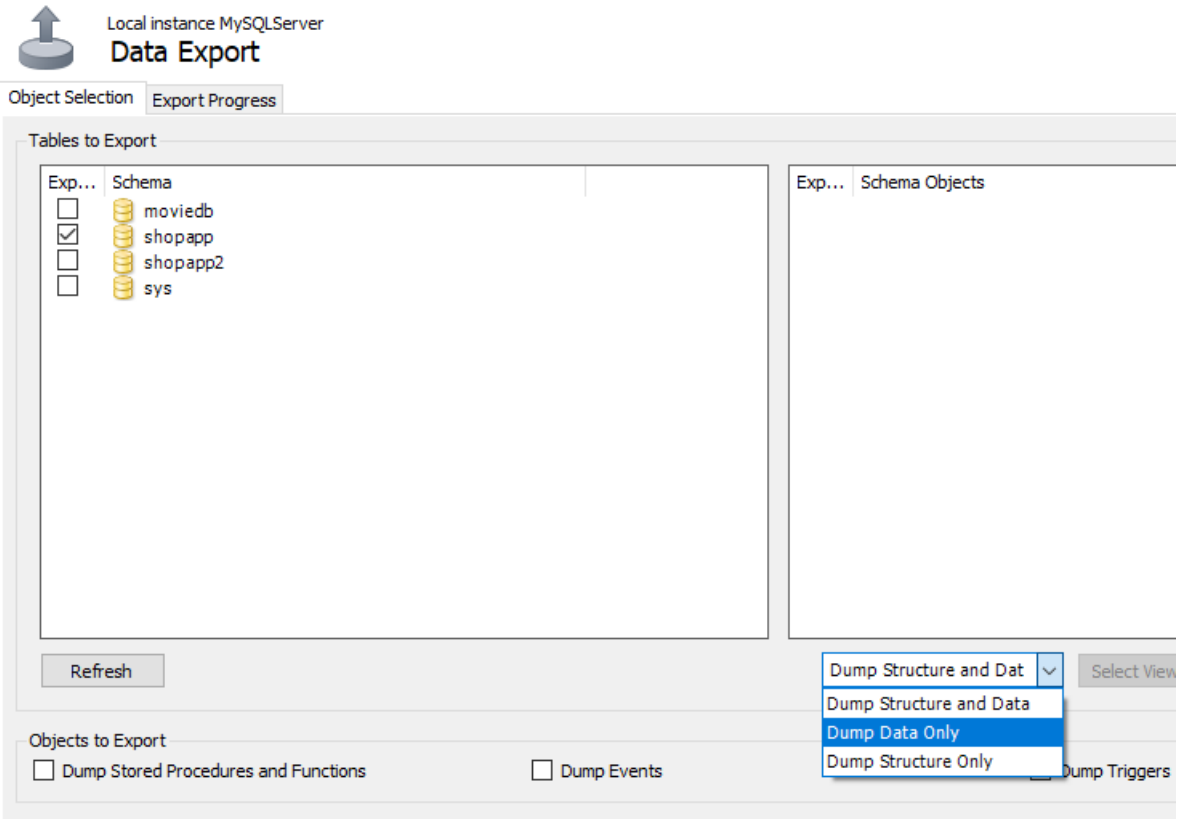
Bu kısımda amacımız, shopapp database'inin içindeki products tablosunda datayı almak ve bu datayı yeni shopapp2 database'inin içindeki product tablosuna aktarmak!

Bunu yapmak için script generate edeceğiz bu script ile de bir database'in verilerini .sql uzantılı bir dosya içerisinde kaydedebilirim.

Öncelikle Local Instance üzerindeyken > Server > Data Export diyoruz.



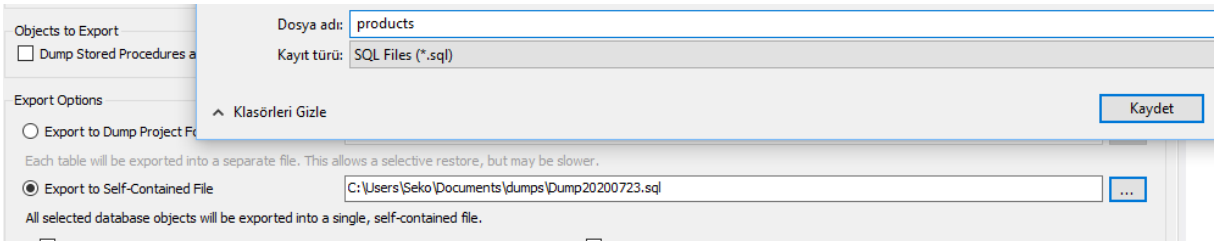
Burada data export'u seçtikten sonra aşağıdaki gibi hangi database'i export etmek istediğimizi seçiyoruz, bunun yanında bu database'in neyini ekport etmek istediğini de aşağıdan seçiyoruz.



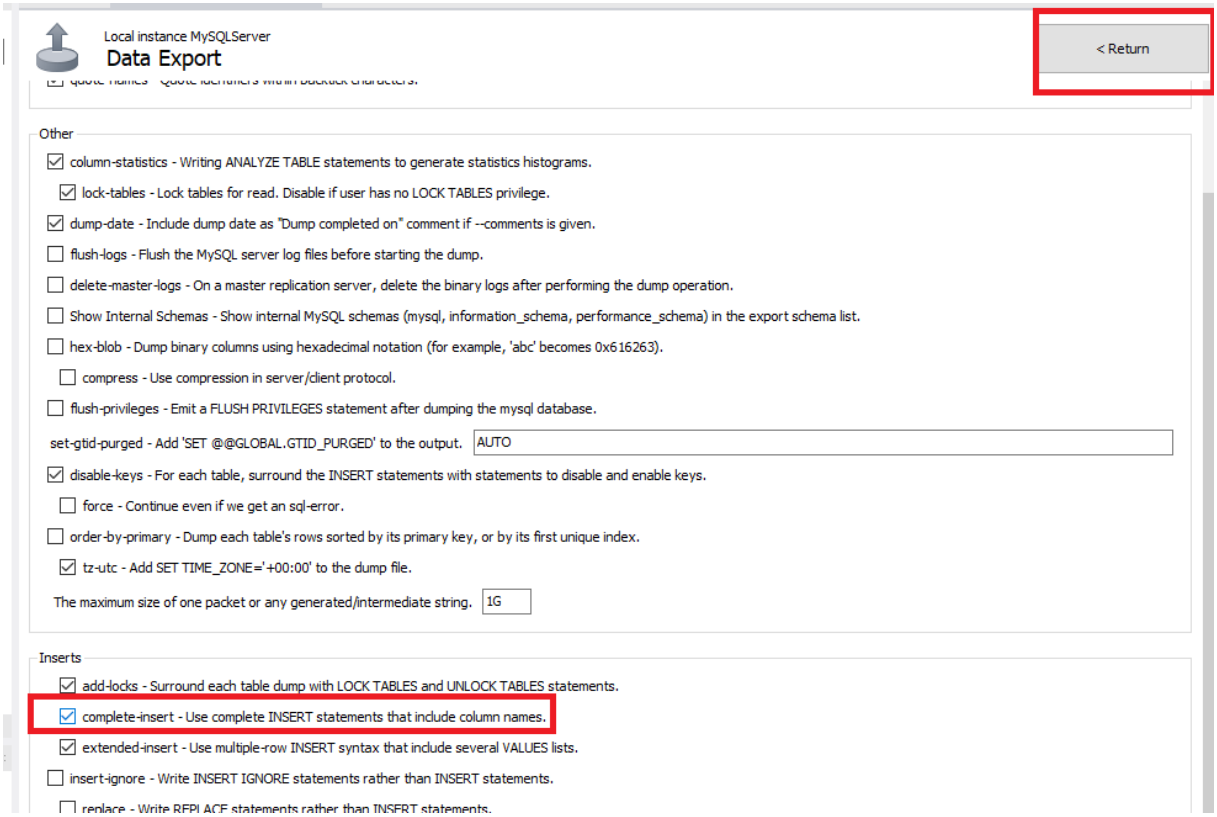
Biz şuan sadece data'yı export etmek istiyoruz ancak istersek structure'la birlikte datayı veya sadece structure'ı da export edebilirdik.

Ki sadece structure'ı export etme işlemini zaten biz reverse engineering ile model oluşturmak ile de yapabiliyorduk, böylece structure model içerisine alınmış oluyordu, forward enginerring ile elde edilen structure'dan geri bir database elde edebiliyorduk.

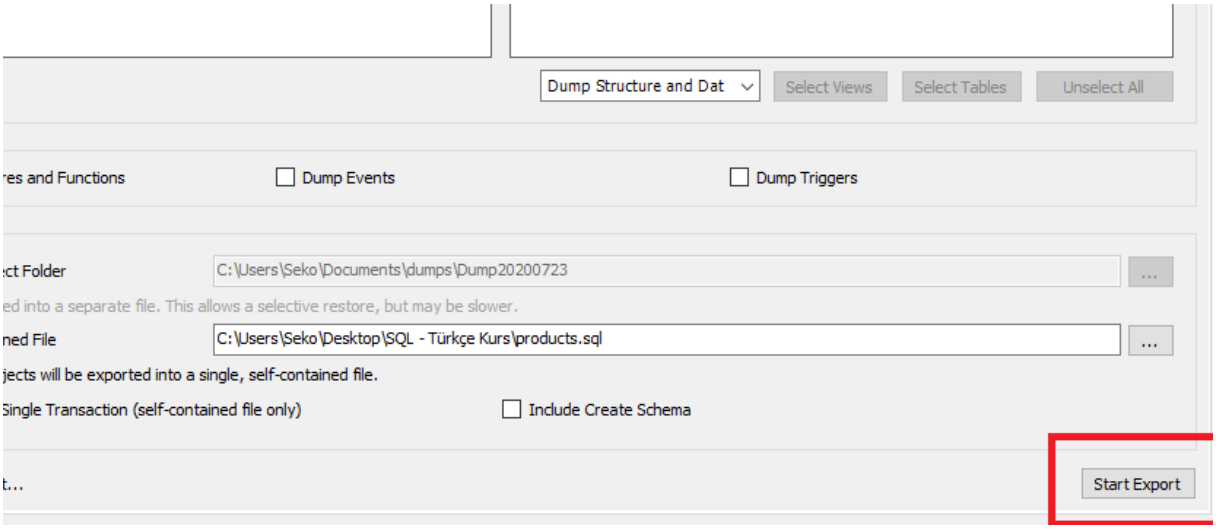
Daha sonra export edilecek data file'ını nereye kaydedeceğimizi seçiyoruz, ve bir .sql file'ı olarak belirttiğimiz yere dosyayı kaydediyoruz.



Daha sonra sağ üstte Adanced'i seçip aşağıda görülen complete-insert checkbox'ını da seçiyorum, bunu seçince insert sorguları içerisinde kolon özellikleri de tanımlanacak.

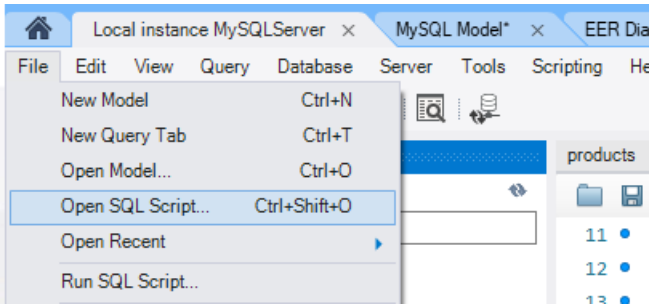


Daha sonra return deriz ve aşağıdan start export diyerek script'imizi belirtilen konuma kaydederiz:

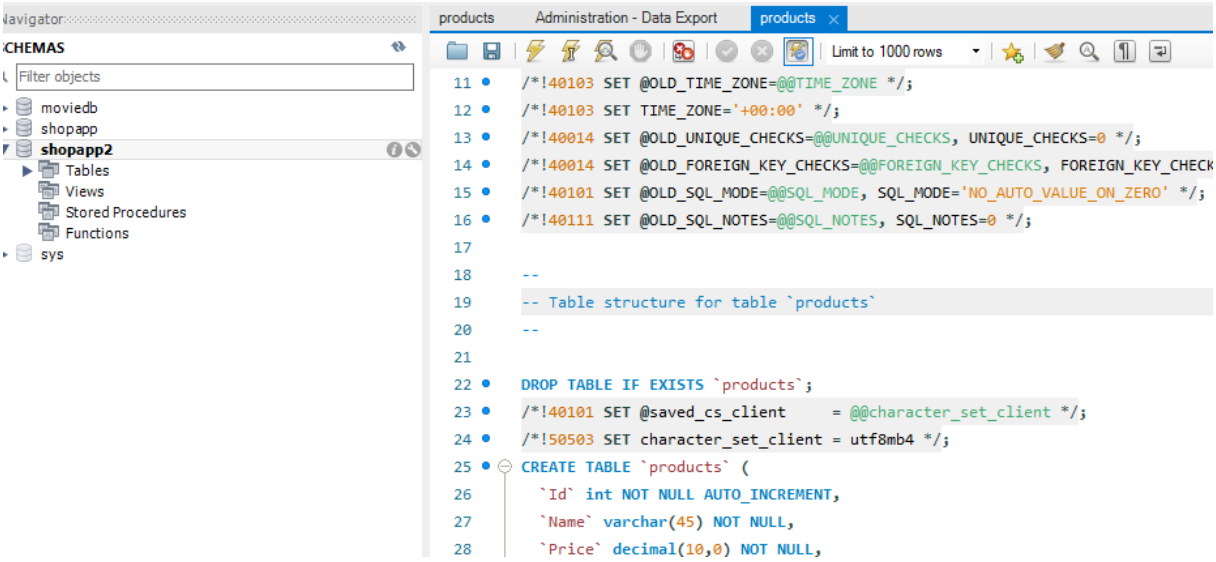


Sonuçta products.sql dosyası istediğimiz yere kaydedildi, bu dosyayı açalım:

File > Open SQL Script > Dosyayı seçiyoruz.



Aşağıdaki gibi products.sql dosyasının içini görebiliyoruz. Bu script temelde shopapp içerisindeki dataya sahip.



Bizim amacımız bu datayı shopapp2 içerisine aktarmak, bu amaçla soldaki panelden shopapp2'yi focus hale getiriyoruz yani çift tıklıyoruz daha sonra da bu products script'ini çalıştırıyoruz ve products içerisindeki veriler shopapp2'ye aktarılmış oluyor.

6. Hazır Veritabanı Kullanımı

Şimdi intenet üzerinden indireceğimiz hazır bir veritabanını local server'ımızda oluşturacağız böylece önümüzdeki kısımda advanced SQL commands'i kullanabileceğiz.

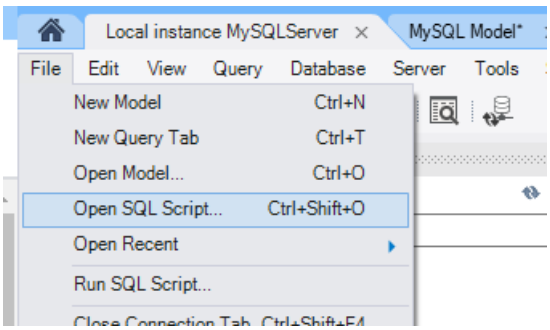
Hazır veritabanını kullanmak için ihtiyacımız olan 2 farklı .sql dosyası var, bir tanesi hazır veritabanının şemasını oluşturmak için gerekli .sql dosyası diğeri ise bu database'in içeriğini doldurmak için gerekli datayı tutan .sql dosyası.

Bu iki script'i github üzerinden indirebiliriz: https://github.com/dalers/mywind

Yukarıdaki github linkinden aşağıdaki iki raw dosya linkine geçebiliriz ve bunları bilgisayarımıza indirebiliriz.

https://raw.githubusercontent.com/dalers/mywind/master/northwind.sql
https://raw.githubusercontent.com/dalers/mywind/master/northwind-data.sql

Daha sonra bu scriptleri File > Open SQL Script ile açarız



Ve çalıştırırız, çalıştırınca önce database oluşur sonra tabloların içleri doldurulur.

Database şeması üzerinde değişiklik yapmak istersek de yapmamız gereken reverse engineering ile database'in modelini elde etmek.

Bu noktadan sonra bir sonraki word dosyasında hazır nortwind dataseti üzerinde advanced sql sorgularını göreceğiz!