



Data Base Design

Aprenda a modelar banco de dados relacionais

O que é um banco de dados?

- O **banco de dados** é uma entidade utilizada para gerenciar os dados de uma aplicação;
- Nos **DBs** temos as tabelas, que são responsáveis por categorizar os dados;
- Além disso, temos outros elementos importantes, como: **queries**, **views**, **stored procedures** e mais;
- Um **DB bem modelado** representa também o sucesso da aplicação;



Principais tipos de DBs

- **Relacional:** o que iremos utilizar no curso, é o mais comum nas aplicações (MySQL);
- **NoSQL:** DBs não relacionais, baseado em documentos (MongoDB);
- **Object Database:** DB orientado a objetos, integrado nas aplicações (ObjectBox);
- **Cloud Database:** Este DB fica em servidores cloud e tem alta escalabilidade, podemos utilizar SQL ou NoSQL;



O que é uma tabela?

- As tabelas são quem recebe a maioria das **operações** de banco de dados, o famoso **CRUD**;
- Elas **categorizam os dados**, os salvando em linhas;
- Os tipos dos mesmo são separados em colunas, como uma tabela de Excel;
- **Criar corretamente as tabelas** também faz parte do DB Design;
- As tabelas possuem características especiais chamadas **constraints**;



A importância do DB Design

- Realizar um bom planejamento do banco de dados e uma modelagem de dados é crucial para o sucesso da aplicação;
- **Reduz erros** no desenvolvimento da aplicação e do próprio DB;
- Melhora a **performance** da aplicação;
- Ajuda na **comunicação** do time de desenvolvimento;
- Consegue integrar pessoas não-técnicas no projeto, pois passa por várias fases: **conceitual, lógico e físico**;



Softwares para DB Design

- Temos diversos softwares para modelar o banco de dados, online e offline;
- Um bem famoso online é o **Lucidchart**, o problema dos online é que geralmente temos limitações pois são pagos;
- O mais utilizado offline é o **MySQL Workbench**, ele faz tudo que os outros fazem e ainda conseguimos transformar a modelagem em um banco real;
- Além disso, os recursos do software o tornam uma boa opção para uso geral, como fazer **queries** em banco de dados;



Instalando o Workbench

- XAMPP ou WAMPP não precisa instalar o server, veremos isso adiante;
- O server é interessante pois podemos **conectar no servidor do banco** e testar queries nos modelos desenvolvidos;
- Vamos lá!



Diagramas do curso

- Os diagramas estão separados por seções;
- Cada um tem um print screen, facilitando a visualização;
- É o material de apoio durante as atividades e projeto do curso;





Introdução

Conclusão da seção



Modelagem de dados

Introdução da seção

O que é modelagem de dados?

- A **modelagem de dados** faz parte do desenvolvimento do diagrama do banco de dados;
- A dividimos em três etapas: **conceitual, lógica e física**;
- Vamos precisar **entender melhor o que precisamos resolver**, qual software está sendo desenvolvido;
- Levantaremos os requisitos e então começamos as etapas descritas acima;



Tipos de modelagem

- **Conceitual:** modelagem de mais alto nível, basicamente definimos as entidades que estão envolvidas no projeto;
- **Lógica:** Nesta etapa podemos definir as relação entre as entidades, e também os seus atributos;
- **Física:** Aqui implementamos recursos mais técnicos, como: constraints, índices, tipos de dados, triggers e etc;
- Realizaremos todas estas etapas no projeto;



Levantamento de requisitos

- **Levantamento de requisitos** é a etapa onde pessoas técnicas e não técnicas se reúnem para falar sobre o produto/projeto;
- Nesta etapa vamos conseguir fazer duas etapas: **conceitual** e **lógica**;
- Precisamos identificar todas as **entidades** que compõem o sistema;
- E também todos os **dados que precisamos guardar** nestas entidades;
- Geralmente o levantamento é feito por uma **reunião**, no curso teremos uma etapa que vai simular isso;



Nosso estudo de caso

- Nosso projeto consiste em um **e-commerce de livros**;
- Vamos precisar passar por **todo o processo de design de DB**;
- Ou seja, pelas fases: **conceitual, lógica e física**;
- Depois nas últimas seções implementaremos **novas funcionalidades**, como se o projeto estivesse crescendo;
- Vamos ver o que foi definido!



Coral Store

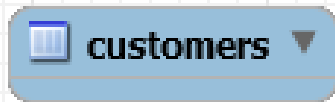
A Coral Store é uma pequena livraria de bairro, o dono nos contratou para desenvolver um e-commerce para sua empresa. A ideia central é captar os dados dos clientes, eles são: nome, email, senha, endereço e telefone. O sistema também precisa possibilitar cadastrar os produtos, como vendemos apenas livros o site a princípio não terá categorias. Os produtos precisam conter: nome, preço, descrição, número de páginas e ISBN. Precisamos também salvar as compras de cada usuário, que deve conter: o valor total e itens comprados. Este será o MVP da Coral Store, se o projeto der certo seremos contratados para expandir o mesmo.

Definindo entidades

- Agora iniciamos a **modelagem conceitual**;
- Vamos precisar ler o nosso documento novamente;
- Identificaremos as **principais entidades** que constituem o projeto que precisamos desenvolver;
- Após este passo criaremos a primeira versão do nosso **diagrama de banco de dados**;
- Vamos lá!



Definindo Entidades



Primary key

- Toda tabela precisa de uma **primary key**;
- Ela precisa ser **única**, ou seja, seu valor não pode se repetir;
- Para resolver esse problema, geralmente é criada uma coluna chamada **id**;
- Onde ela recebe duas características (**Constraints**):
- **Primary Key** – Valor único que não se repete ao longo da tabela como id, cpf, etc
- **Not Null** – Não admite registro vazio;
- Vamos aplicar isso também no nosso projeto!

Definindo atributos

- Agora que temos todas as entidades mapeadas entramos na **fase lógica**;
- Aqui **transformaremos em colunas para as tabelas todos os dados que precisamos salvar** das entidades;
- Isso já torna o nosso projeto muito mais próximo da **versão final**;
- Vamos lá!



Definindo atributos

customers
id INT
name VARCHAR(45)
email VARCHAR(45)
password VARCHAR(45)
address VARCHAR(45)
customerscol VARCHAR(45)
phone VARCHAR(45)
Indexes

products
id INT
name VARCHAR(45)
description VARCHAR(45)
price VARCHAR(45)
page_qty VARCHAR(45)
isbn VARCHAR(45)
Indexes

orders
id INT
customer VARCHAR(45)
items VARCHAR(45)
total VARCHAR(45)
Indexes

Sobre a física

- A próxima etapa seria a **fase física**, correto?
- Porém na verdade esta etapa seria já o banco **completamente desenhado e otimizado** para desenvolver a aplicação;
- Então normalmente é feito o processo de **normalização** antes, que é o vamos aprender nas próximas aulas;
- Com a normalização, teremos a fase física sendo aplicado aos poucos;
- E então poderemos desenvolver nossa aplicação!





Modelagem de dados

Conclusão da seção



Iniciando a normalização

Introdução da seção

O que é normalização?

- **Normalização** é um processo onde melhoramos a qualidade do nosso banco de dados, a fim de deixá-lo mais **otimizado**;
- Temos algumas formas normais para aplicar;
- As principais são três: **NF1**, **NF2** e **NF3**; (normal form)
- Ao longo do curso aplicaremos todas em nosso banco de dados;
- E com isso teremos também a modelagem física sendo concebida;



Forma normal 1

- Cada NF deve seguir um conjunto de requisitos para ser totalmente aplicada;
- As da NF1 são:
- Colunas devem ter apenas um valor em cada célula (atomicidade);
- Cada coluna deve salvar apenas um tipo de dado;
- O nome das colunas deve ser único;
- A ordem das colunas não importa;



Aplicando NF1 em clientes

- Agora vamos ter que **analisar todas as tabelas** e aplicar a normalização;
- É normal que ao aplicar as NFs seja necessário **criar uma nova tabela**;
- Alguns dos pontos a serem observados em customers são: **address**
e name;
- Ambas possuem a possibilidade de inserção de vários dados;
- Vamos lá!



N1 - Customers

addresses	customers	products	orders
id INT	id INT	id INT	id INT
street VARCHAR(45)	first_name VARCHAR(45)	name VARCHAR(45)	customer VARCHAR(45)
number VARCHAR(45)	last_name VARCHAR(45)	description VARCHAR(45)	items VARCHAR(45)
neighborhood VARCHAR(45)	email VARCHAR(45)	price VARCHAR(45)	total VARCHAR(45)
city VARCHAR(45)	password VARCHAR(45)	page_qty VARCHAR(45)	
region VARCHAR(45)	phone VARCHAR(45)	isbn VARCHAR(45)	
complement VARCHAR(45)			
customer VARCHAR(45)			
Indexes	Indexes	Indexes	Indexes

Meta:

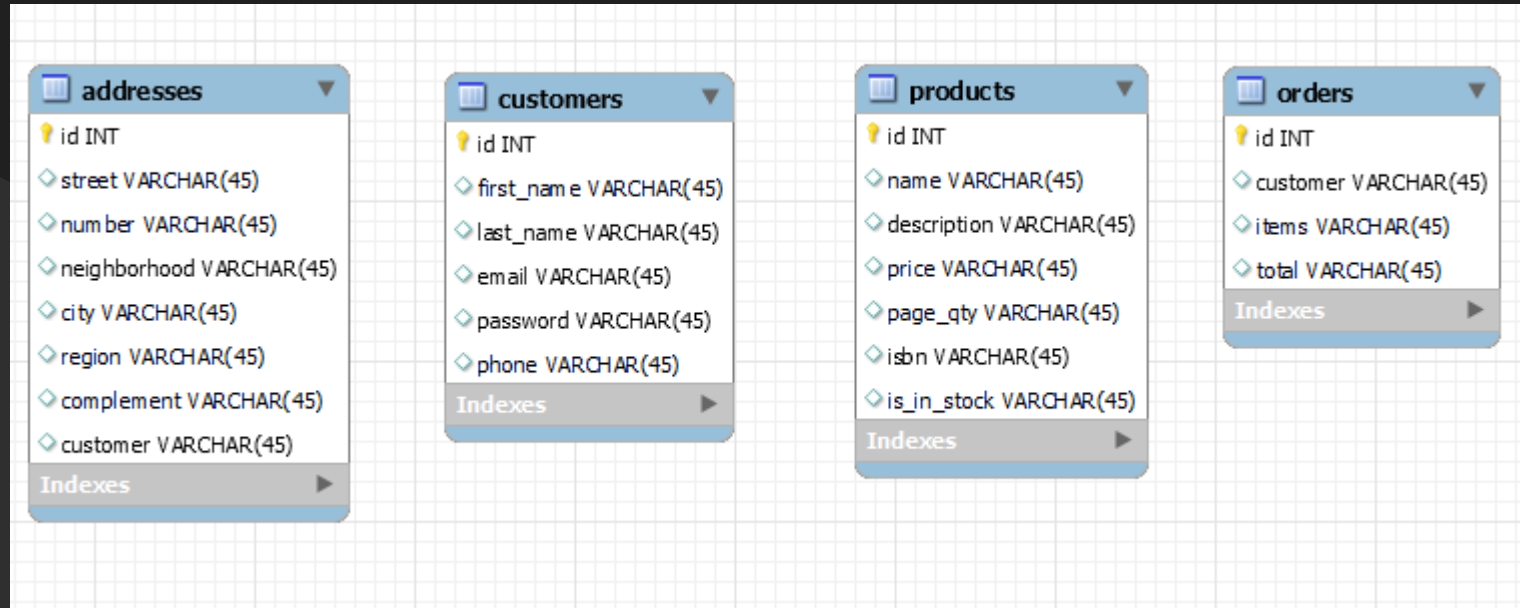
- 1 – Dividir nome em first_name e last_name
- 2 – Criar nova tabela addresses para os endereço

Aplicando NF1 em produtos

- Agora vamos analisar a **tabela de produtos**;
- Será que precisamos fazer alguma otimização para a NF1?
- Além disso, surgiu a necessidade de um novo campo, precisamos colocar se o produto **está ou não disponível em estoque**;
- Vamos lá!



N1 - Products



Meta:

1 – Em products criar o item is_in_stock para saber se o produto está em estoque

Aplicando NF1 em pedidos

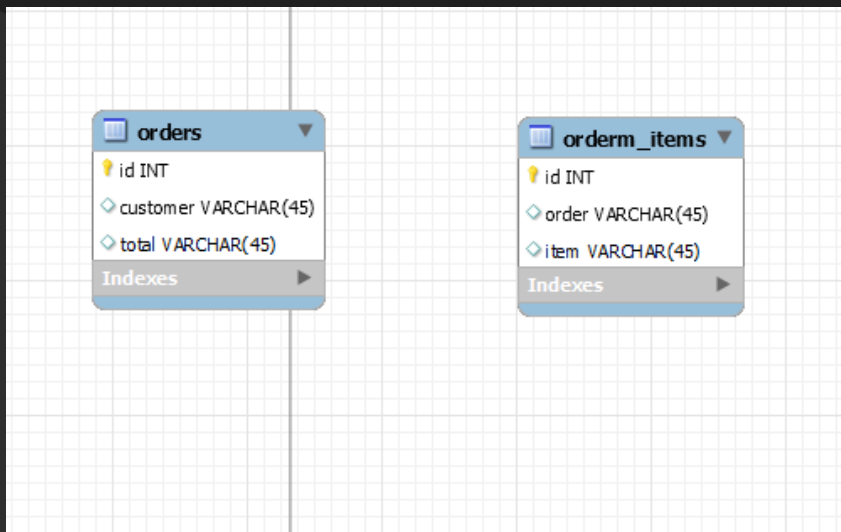
- Agora vamos analisar a **tabela de pedidos**;
- O campo que precisamos observar é items, ele também nos permite adicionar mais um item;
- Vamos lá!



N1 - Orders

Meta:

- 1 – Em orders criar retirar o itens.
- 2- Criar uma nova tabela “ordem_items” para armazenar os itens do pedido, com os campos com os campos id, order, item





Iniciando a normalização

Conclusão da seção



Aprimorando o banco

Introdução da seção

Forma normal 2

- Para aplicar a NF2 **todos os requisitos da NF1 precisam estar resolvidos**;
- E outro requisito da NF2 é que não tenhamos **dependências parciais** nas nossas tabelas;
- Essa dependência é caracterizada por **uma coluna que não tem relação direta com o intuito principal da tabela**;
- **Exemplo:** Produto e categoria não podem estar na mesma tabela;



Aplicando NF2 em endereços

- Vamos analisar uma nova tabela nossa, a de **endereços**;
- Temos a coluna de estado;
- Geralmente os estados possuem uma **sigla** (SC), e o seu **nome** (Santa Catarina);
- Isso é muita informação para uma coluna e não faz relação direta com o propósito da tabela: um endereço de um usuário;
- Vamos lá!



Meta:

- 1 – Em addresses altere o nome “region” para “region_id”
- 2- Criar uma nova tabela “regions” para armazenar code e name

addresses	
id	INT
street	VARCHAR(45)
number	VARCHAR(45)
neighborhood	VARCHAR(45)
region_id	VARCHAR(45)
complement	VARCHAR(45)
customer	VARCHAR(45)
Indexes	

regions	
id	INT
code	VARCHAR(45)
name	VARCHAR(45)
Indexes	

N2 - Adresses

Versão completa até aqui

addresses
id INT
street VARCHAR(45)
number VARCHAR(45)
neighborhood VARCHAR(45)
city VARCHAR(45)
region_id VARCHAR(45)
complement VARCHAR(45)
customer VARCHAR(45)
Indexes

customers
id INT
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(45)
password VARCHAR(45)
phone VARCHAR(45)
Indexes

products
id INT
name VARCHAR(45)
description VARCHAR(45)
price VARCHAR(45)
page_qty VARCHAR(45)
isbn VARCHAR(45)
is_in_stock VARCHAR(45)
Indexes

orders
id INT
customer VARCHAR(45)
total VARCHAR(45)
Indexes

order_items
id INT
order VARCHAR(45)
item VARCHAR(45)
Indexes

region
id INT
code VARCHAR(45)
name VARCHAR(45)
Indexes

Relacionamentos entre tabelas

- Quando aplicamos a NF2 o número de tabelas pode aumentar no banco;
- E há a necessidade de uma **conexão entre elas**, ou seja, relacionamentos;
- Temos três principais: **One to One**, **One to Many**, **Many to Many**;
- Com o relacionamento podemos identificar facilmente um dado mais completo e complexo de uma outra tabela;
- E ainda **mantendo os padrões de normalização** aplicados no banco;



Sobre os relacionamentos

- Vamos entender melhor os **relacionamentos**:
- **1:1** = Quando as duas tabelas possuem apenas um registro de ligação entre elas, exemplo: usuário e endereço;
- **1:n** = Quando uma tabela possui várias ligações na outra e o oposto não ocorre, exemplo: usuário e pedidos;
- **n:m** = Quando ambas tabelas tem múltiplos relacionamentos, exemplo: produtos e pedidos;



Chave estrangeira

- A **chave estrangeira** ou **foreign key (FK)** é um componente das relações entre tabelas;
- Basicamente consiste no **valor da chave primária onde a tabela tem ligação**;
- **Isso faz a conexão entre ambas**, conseguimos então saber qual registro possui ligação com qual;
- **Exemplo:** Qual estado o endereço do usuário pertence, id = 10 = SC;



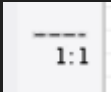
Aplicando relacionamentos

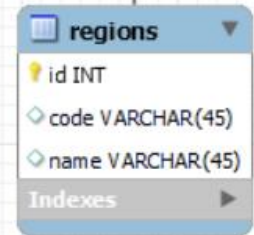
- Agora vamos **relacionar algumas tabelas no nosso design**;
- Sabemos que todo endereço tem um estado, e todo usuário tem pelo menos um endereço;
- Isso nos dá uma relação **1:1** em **endereço x estado**;
- E **1:n** em **usuário x endereço**, pois podemos cadastrar mais de um endereço de entrega;
- Vamos lá!



Relacionamento entre Addresses e Regions


Note que um endereço só pode estar associado a 1 Estado
então Relacionamento 1: 1

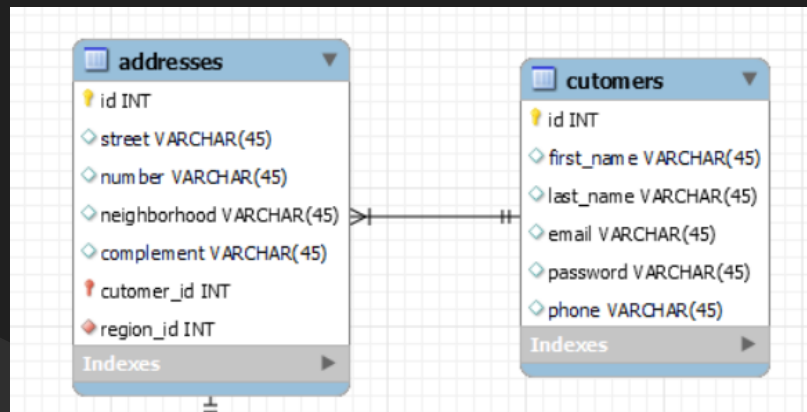
- 1 Localize o relacionamento 
- 2 Clique na tabela Addresses e ligue com a tabela regions
- 3 Renomeia para region_id
- 4 Em Addresses apague o item region_id criado anteriormente



Relacionamento entre Customers e Addresses 1:N

Note que na base de dados um usuário pode ter vários endereços de entrega de produtos, configurando uma relação de 1:N

- 1 Localize o relacionamento 
- 2 Ligue a tabela Addresses na Tabela Customers
- 3 Na tabela “Addresses” altere o nome para “region_id”
- 4 Em Addresses apague o item customer criado anteriormente



Aplicando relacionamentos

Versão completa até aqui

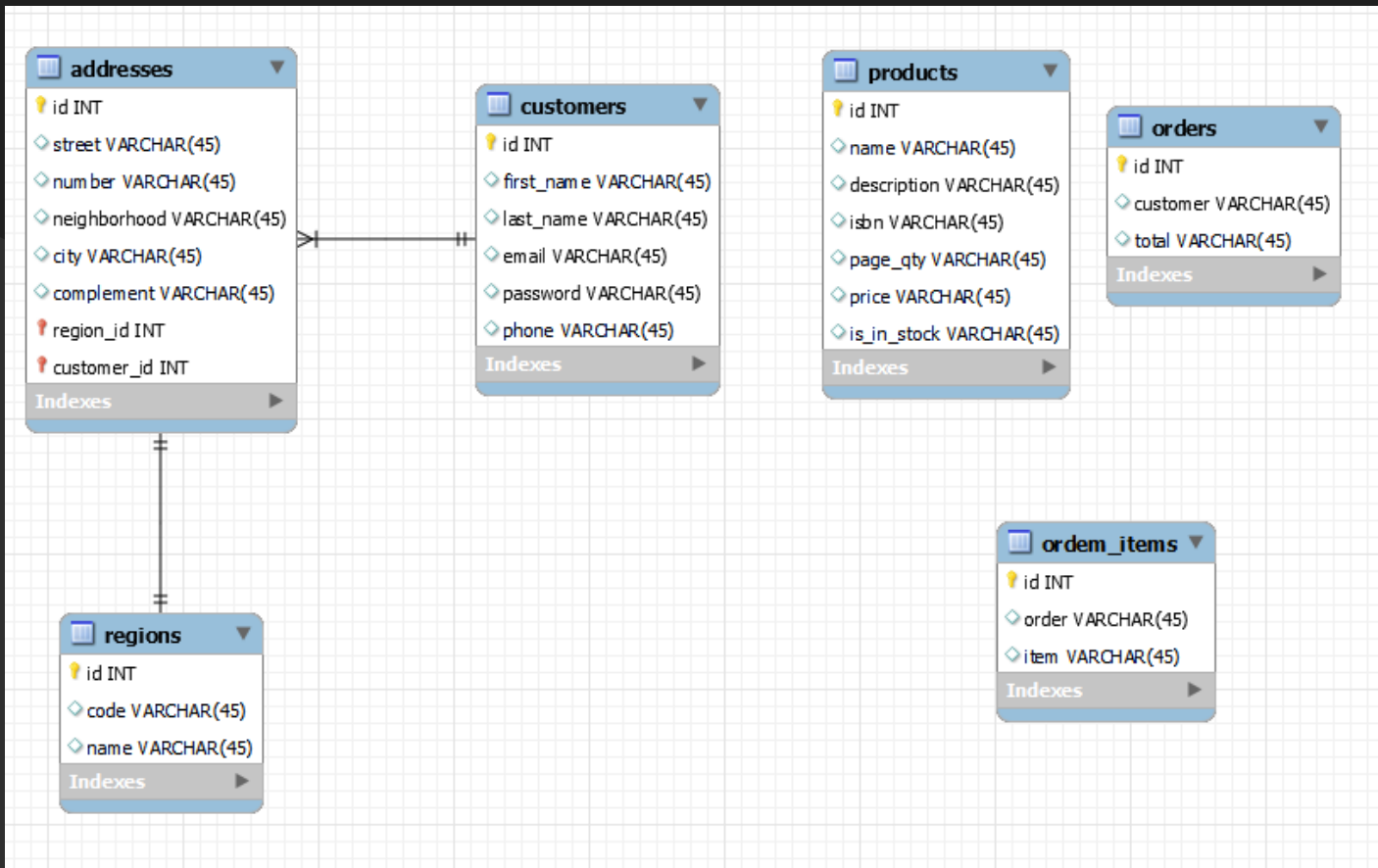


Tabela pivot

- **Tabela pivot** é uma estratégia utilizada para relação de muitos para muitos (n:m);
- **Ela concentra apenas ids** (chaves primárias) das outras tabelas, fazendo as relações;
- Se não tivéssemos esta tabela, **adicionaríamos vários dados redundantes em uma das tabelas**;
- Isso deixa nosso banco e design mais limpo;



Continuando nos relacionamentos


- Agora que já aprendemos como criar corretamente um relacionamento n:m, **vamos aplicar no nosso DB;**
- E também podemos **finalizar esta questão de relacionamento para todas as tabelas;**
- Assim teremos mais uma etapa cumprida;
- **Fomos informados também que todo pedido precisa conter o endereço de entrega, vamos lá!**

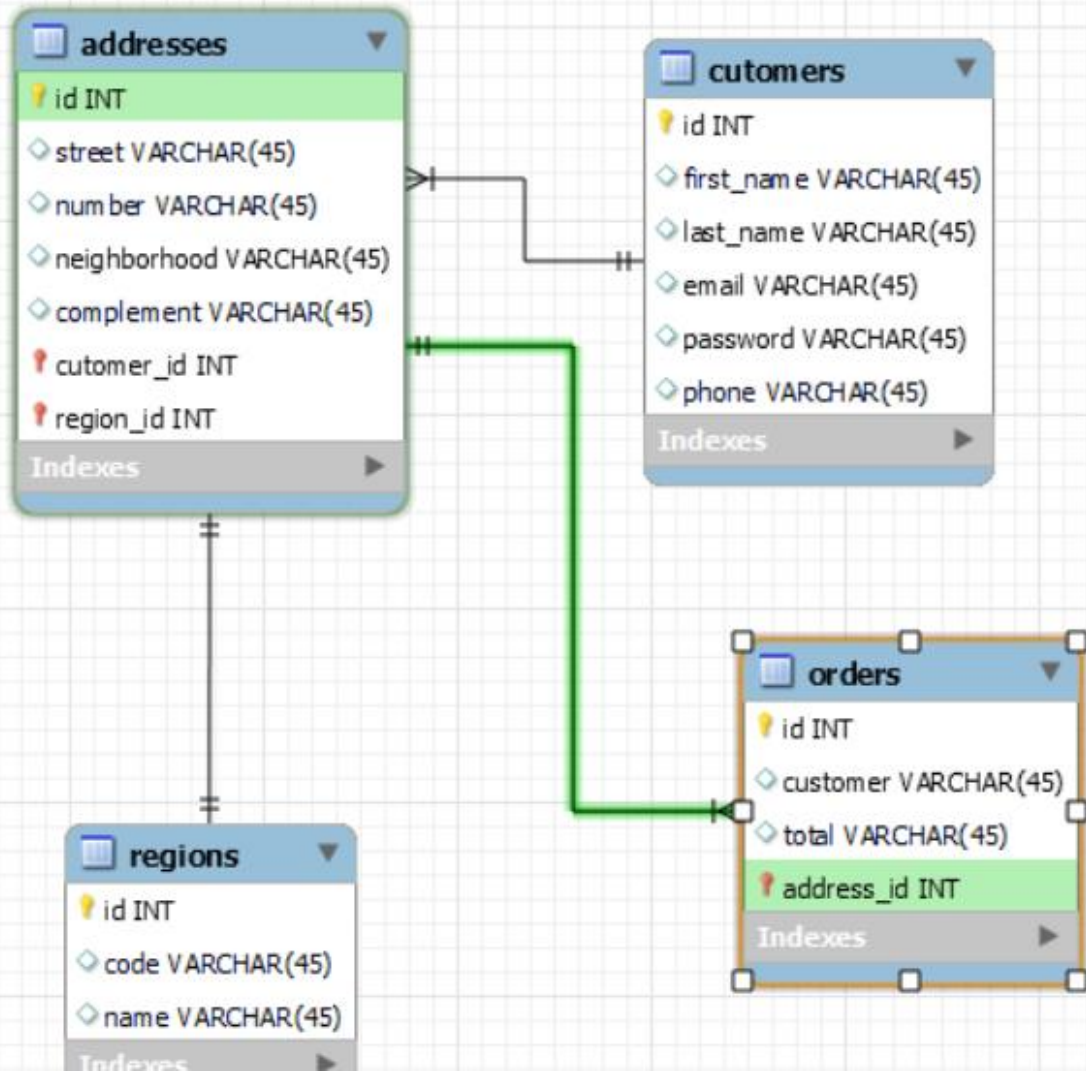


Relacionamento entre Endereços e Perdidos

1 pedido só pode ter 1 endereço de entrega, mas 1 endereço de entrega pode ser destino de muitos pedidos

Configura o relacionamento 1:N


- 1 Clique em Orders e traga mais para o meio da tela
- 2 Localize o relacionamento 
O ícone mostra uma seta apontando para a direita com o texto "1:n" abaixo dela, representando um relacionamento um-para-muitos.
- 3 Clique em Orders e em seguida em Addresses
- 4 Note que esse relacionamento trás chaves estrangeiras que não precisamos no momento. Apague "addresses_cutomer_id" e "addresses_region_id"
- 5 Altere o nome "addresses_id" para "address_id"

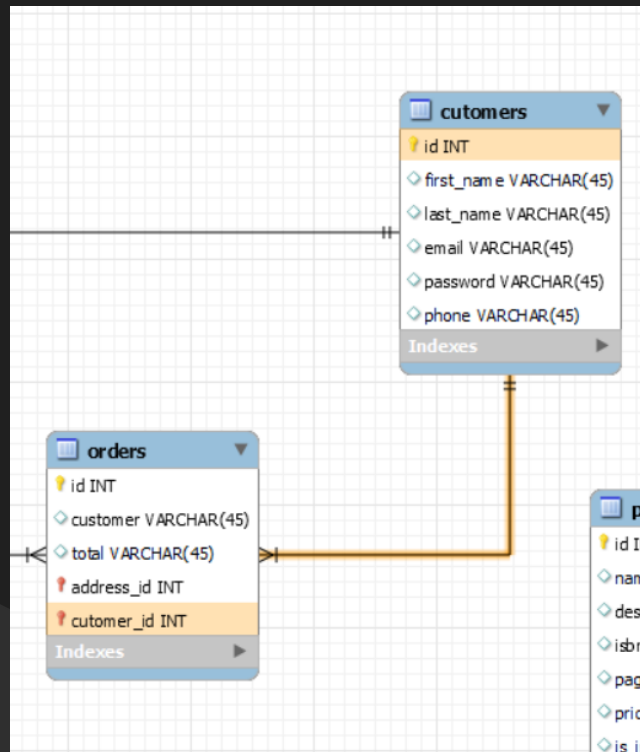


Relacionamento entre Cliente e Perdidos

1 pedido só pode ser direcionado a 1 Cliente, mas 1 Cliente pode ter vários pedidos ou compras associadas ao seu nome

Configura o relacionamento 1:N


- 1 Localize o relacionamento 
- 2 Clique em Orders e em seguida em Customers
- 3 Altere o nome “cutomers_id” para “cutomer_id”



Relacionamento entre Pedido e produto

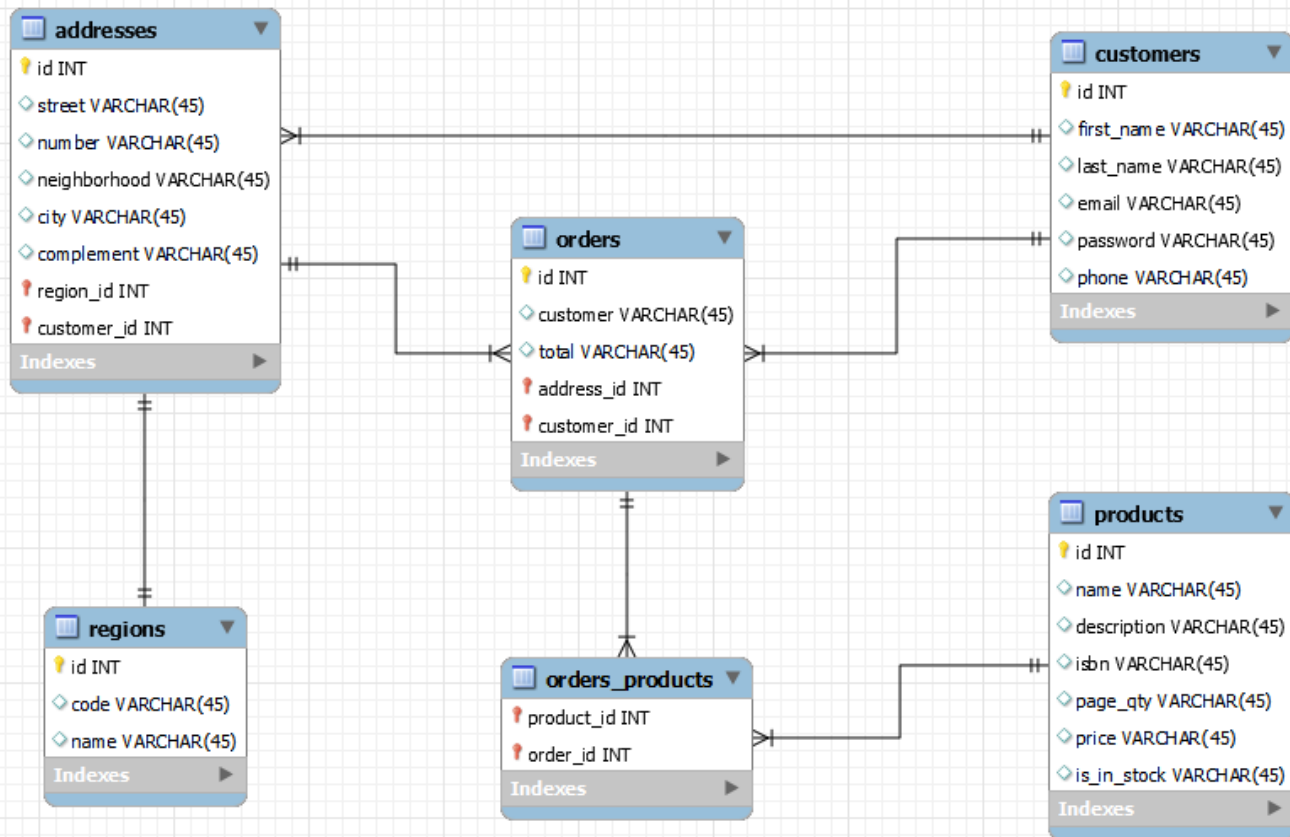
1 produto pode estar contido em vários pedidos e 1 pedido pode conter vários produto, ou seja

Relacionamento N:M

- 1 Localize o relacionamento 
- 2 Clique em Products e em seguida em Orders
- 3 Note que será gerada uma tabela de relacionamento products_has_Orders altere o nome para "orders_products"
- 4 Coloque no singular products_id e orders_id para product_id e order_id respectivamente
- 5 Exclua os campos orders_address_id e orders_customer_id

Finalizando Relacionamentos

Versão completa até aqui





Aprimorando o banco

Conclusão da seção



Concluindo normalização

Introdução da seção

Forma normal 3

- Para aplicar a NF3 todos os requisitos da NF2 precisam estar **resolvidos**;
- O requisito da NF3 é que seja feita a remoção de dependências transitivas;
- **Dependências transitivas: são colunas que tem relação com a chave primária**, mas podem ser desassociadas, criando outra tabela;



Parciais x transitivas

- As dependências **parciais** são os dados que não tem relação direta com a PK; Ex. Endereços e Estado – Conseguimos ter um endereço sem Estado mas por uma questão de normalização ajustamos
- As **transitivas** são as que têm relação, mas podem ser movidas (lembre-se das tabelas que retiramos ao longo do tempo);
- Com isso temos tabelas mais enxutas e facilitamos a expansão do sistema, **evitando dados repetidos**;



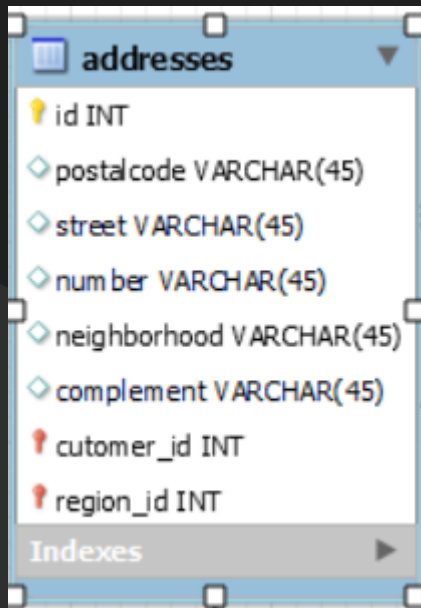
Nova demanda!

- Um dos nossos desenvolvedores notou a falta de um campo muito importante: **o CEP do endereço**;
- Porém com ambições maiores, temos que preparar a loja já para uma possível **internacionalização**;
- O campo vai se chamar **postalcode**, e também temos que prever um **país** para todo endereço cadastrado;
- Aproveitaremos isso para aplicar a **NF3**!



Adicionando PostalCode

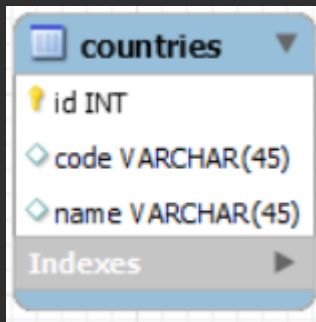
- 1 Clique em Addresses
- 2 Adicione o campo postcode e arraste para cima em baixo de Id



Adicionando Países

Note que Países estão associados ao endereços, sendo uma dependência parcial, pois precisamos ter ele registrado mas não está ligado ao propósito da tabela endereço.

- 1 Clique em adicionar tabela 
- 2 Nomeie como “countries” com os campos: id, code e name

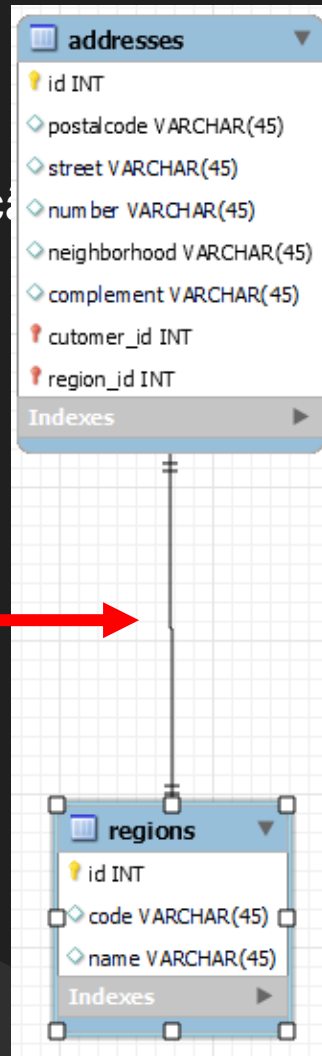
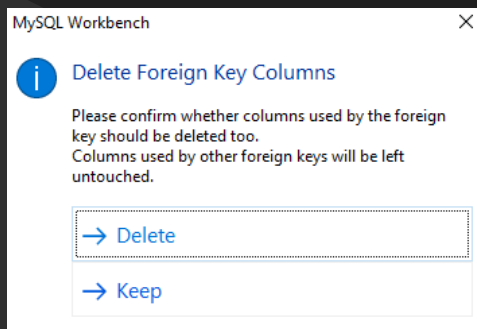


Parei no vídeo 42. Aplicando NF3 em 3:15s quando quebraríamos a ligação entre Addresses e regions criando uma nova tabela `adresses_details`

Para criar a dependência parcial vamos remodelar a relação **addresses** e **regions**.

3

Vamos recriar a ligação entre **Addresses** e **regions**. Delete essa ligação




Recriando o relacionamento

4

Crie uma nova tabel chamada “adresses_detals” com os campos id, postalcode, street, neighborhood, city

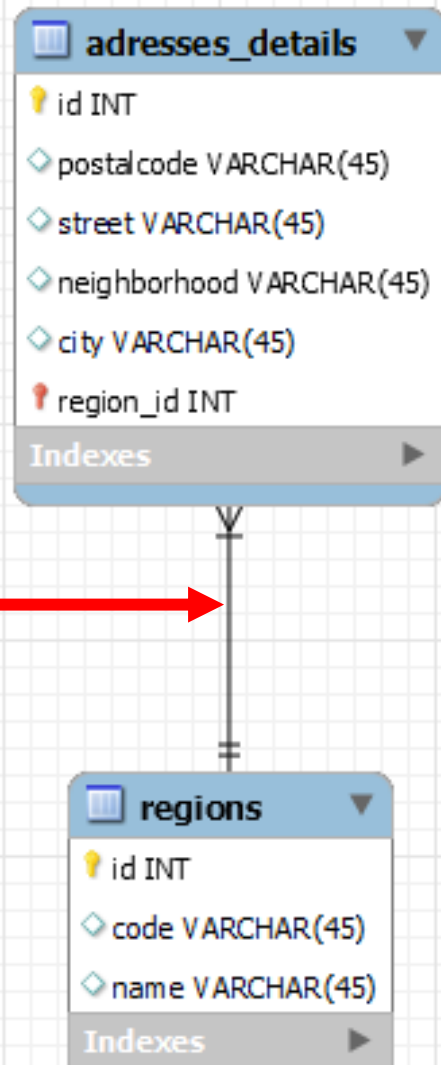
Relacionamento entre adresses_details e regions – Note que 1 região do estar associada a muitos endereços então o relacionamento será 1:n

5

Clique em  em seguida clique em adresses_details ligando regions

6

Em Addresses_details altere o nome regions_id para region_id

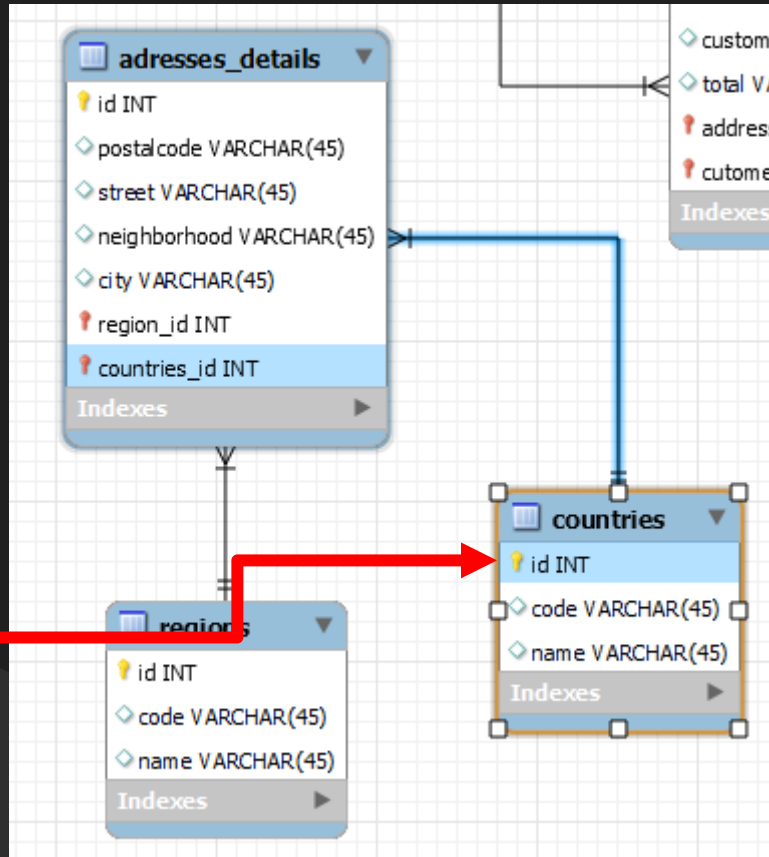


Relacionamento entre países e regiões

Podemos ter vários endereços de um mesmo país cadastrado. Temos o relacionamento 1:N


7 Clique em `addresses_details` em seguida clique em `addresses_details` ligando `countries`

8 Em `Addresses_details` altere o nome `countries_id` para `countrie_id`



9

Vamos agora apagar os campos que não precisamos mais da tabela address: postcode, street, neighborhood, city




addresses	
id	INT
number	VARCHAR(45)
complement	VARCHAR(45)
customer_id	INT
Indexes	

Relacionamento entre addresses e addresses_details

Note que 1 endereço pode ter muitos detalhes, cada detalhe é apenas par 1 endereço, sendo relacionamento 1:N

10

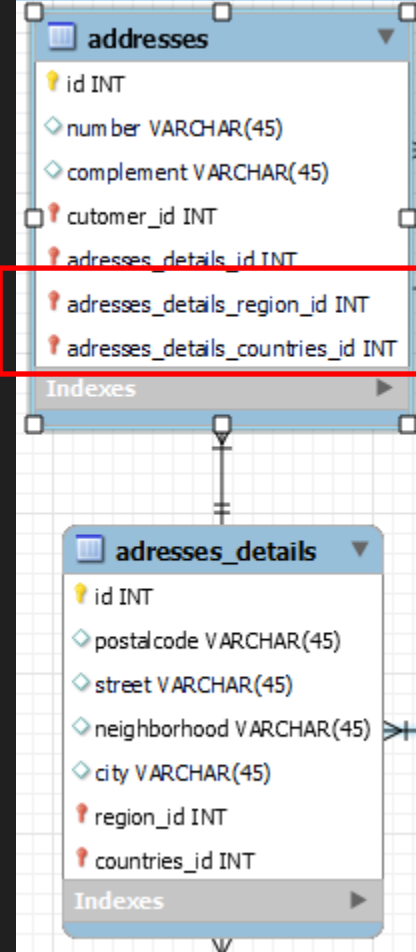
Selecione  em seguida, clique em
addresses e ligue com addresses_details

11

Em Addresses apague
addresses_details_countries_id e
addresses_details_region_id

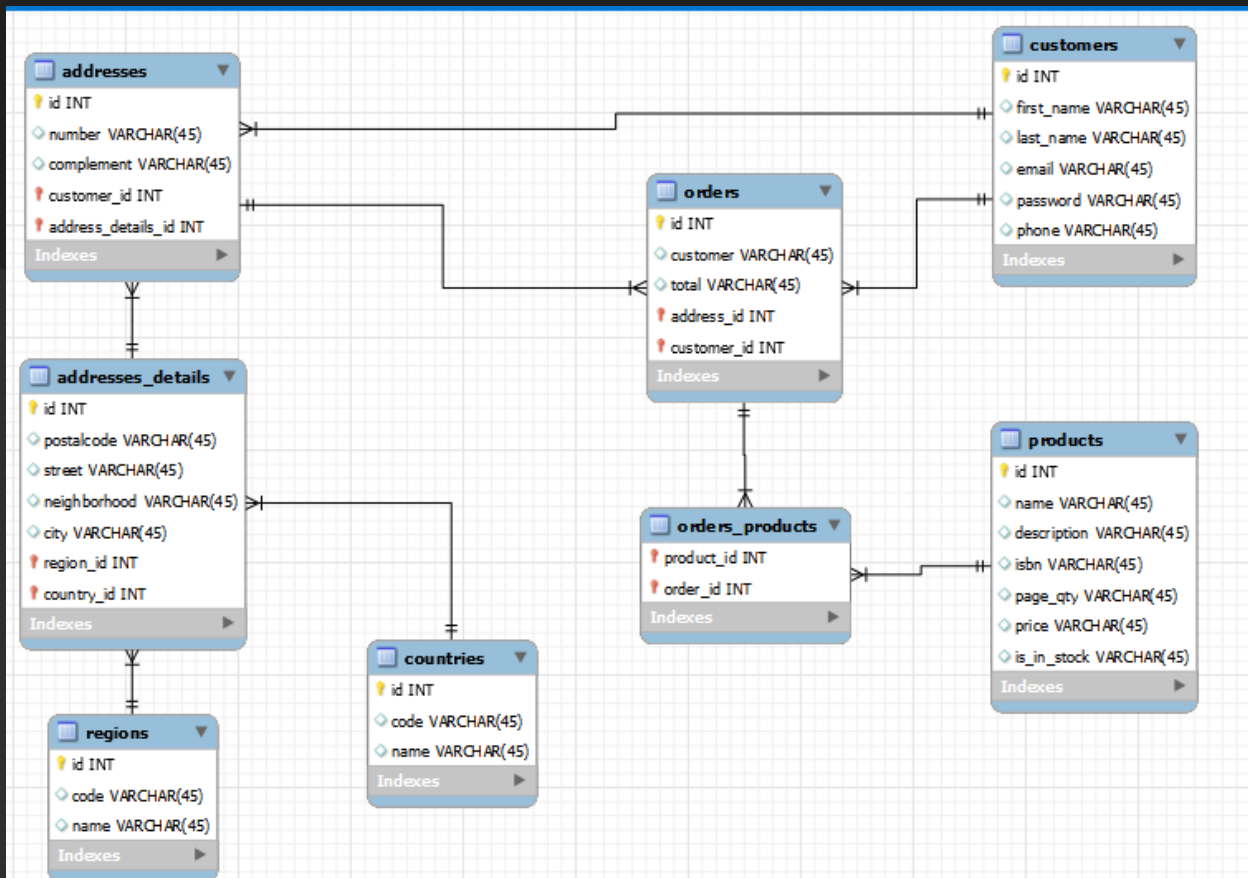
12

Em Addresses renomeie
addresses_details_id para
address_details_id



Adicionando recursos

Versão completa até aqui

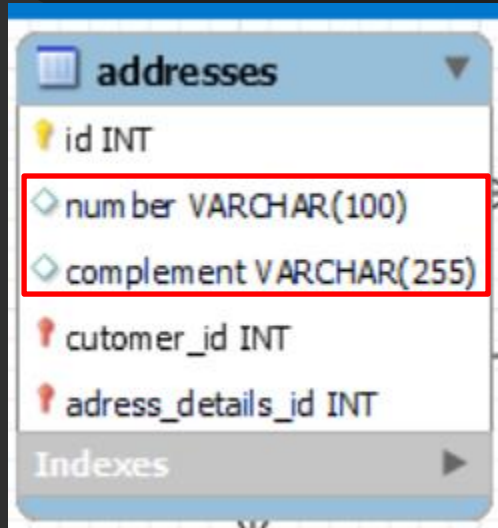


Polindo os campos das tabelas

- Agora vamos começar a melhorar os campos da tabela, já que o banco foi completamente desenvolvido;
- Precisamos identificar quais categorias encaixam melhor com os campos (**VARCHAR**, **FLOAT**, **DATE** e etc);
- Um outro requisito que nos foi pedido é adicionar as **colunas de pedido criado** e **pedido entregue** na tabela de pedidos;
- Vamos lá!



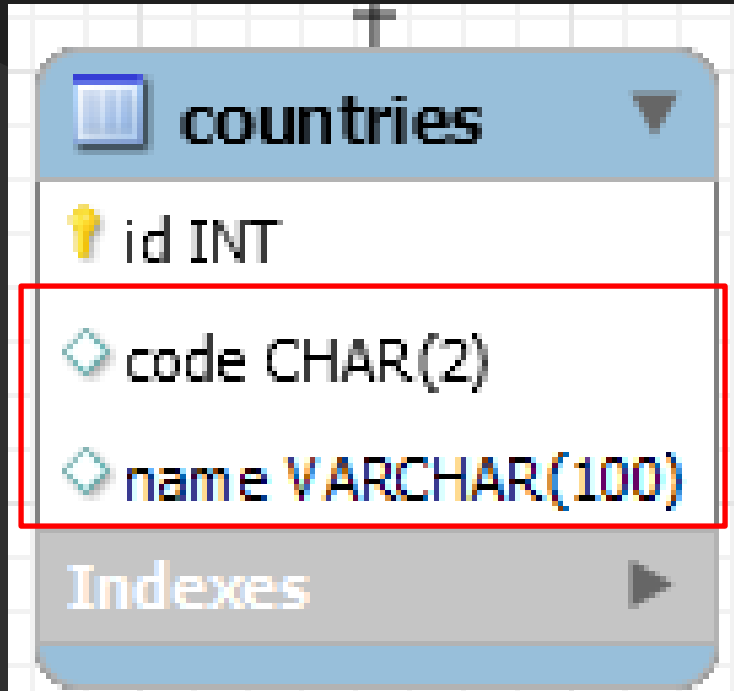
Ajustando os tipos de dado : Adreesses



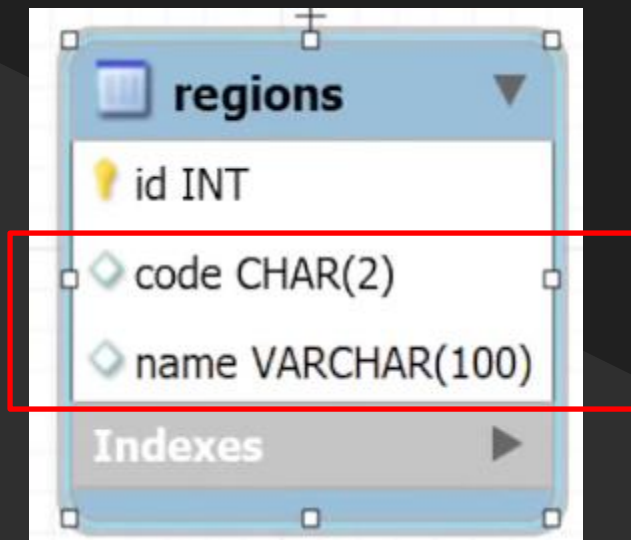
Ajustando os tipos de dado: Adreesses_details

addresses_details	
id	INT
postalcode	VARCHAR(10)
street	VARCHAR(255)
neighborhood	VARCHAR(100)
city	VARCHAR(100)
region_id	INT
country_id	INT
Indexes	

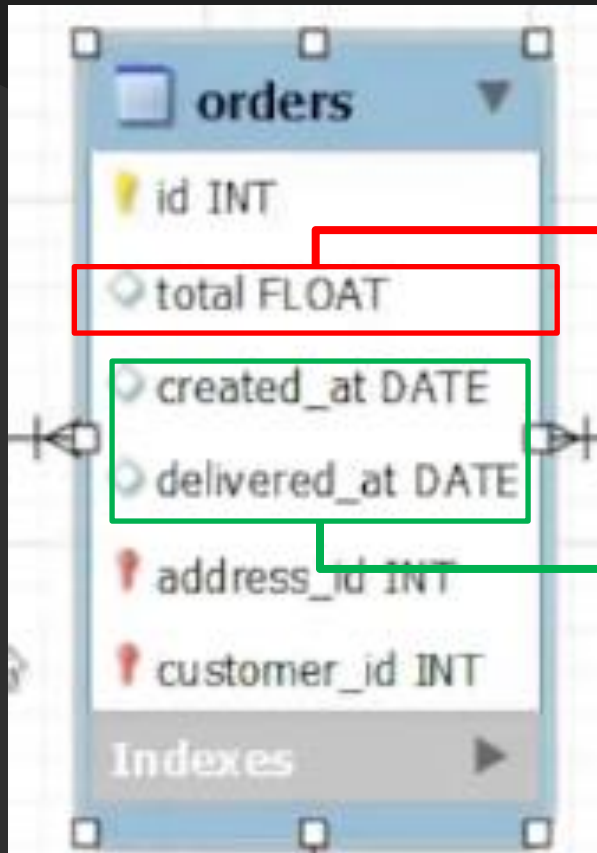
Ajustando os tipos de dado: countries



Ajustando os tipos de dado: regions



Ajustando os tipos de dado: orders



1

Apague o campo de customer

2

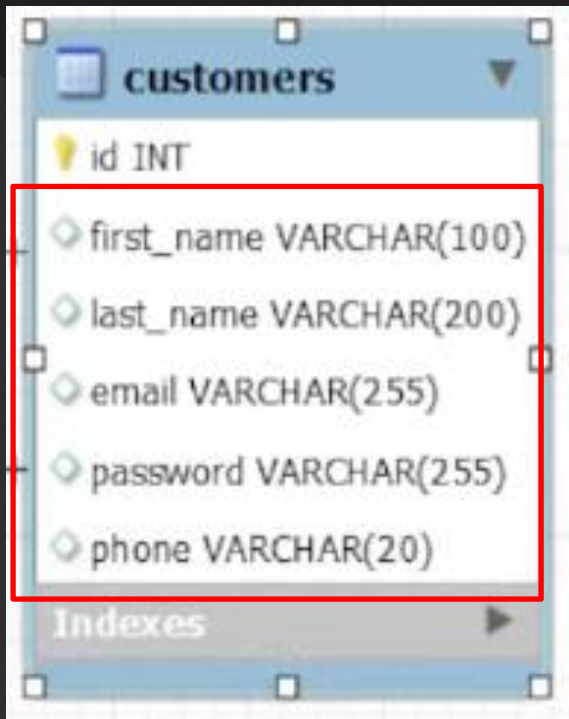
No campo total, precisamos utilizar número com virgula então vamos alterar o tipo de dado para **float**

3

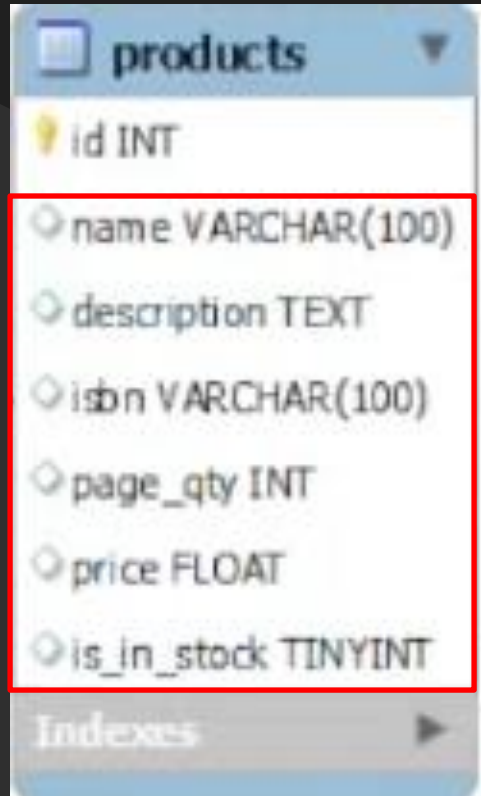
Vamos adicionar 2 colunas que foram solicitadas

Created_at do tipo date
Devlivered_at do tipo date

Ajustando os tipos de dado: customers

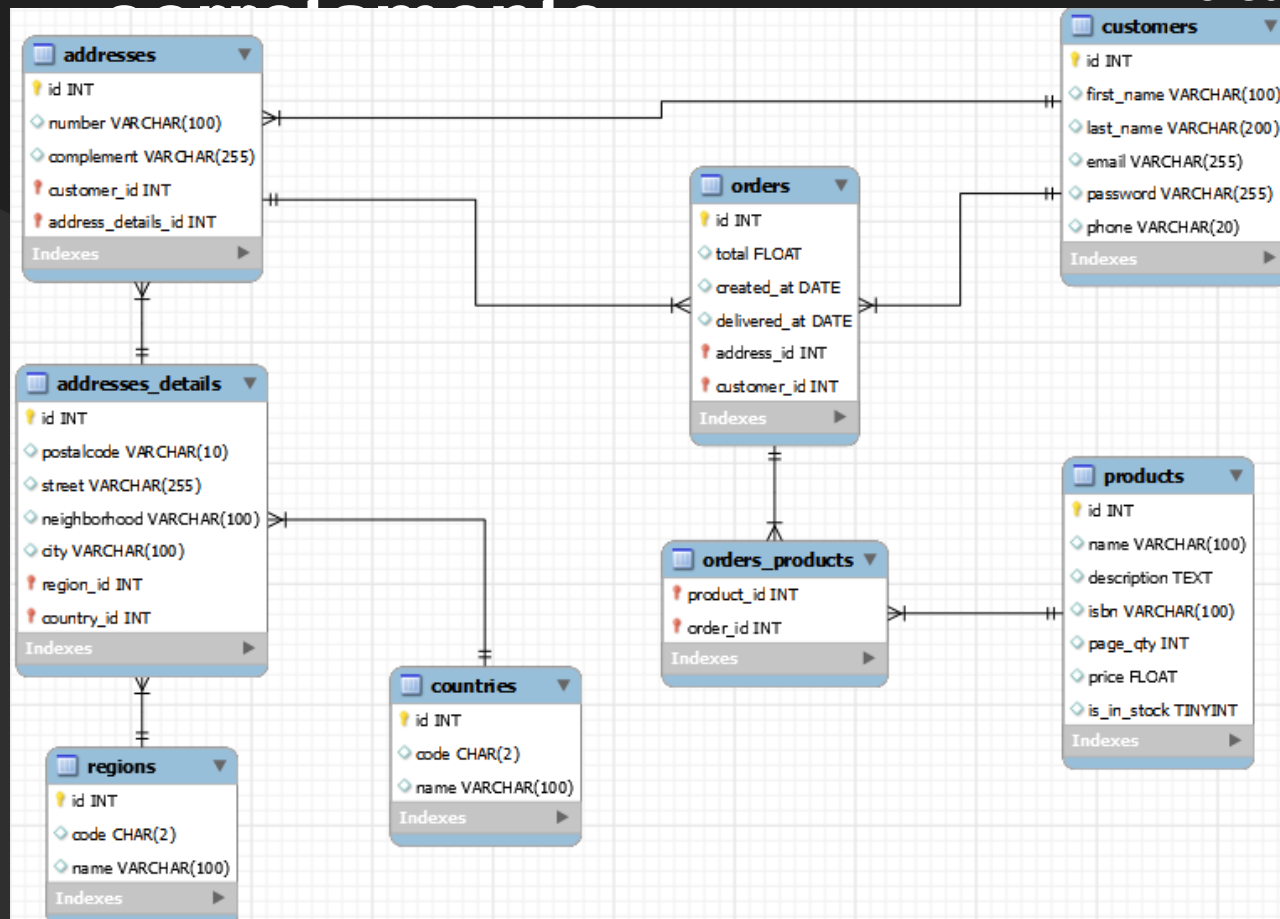


Ajustando os tipos de dado: products



Escolhendo os tipos de dado

Versão completa até aqui





Concluindo normalização

Conclusão da seção