

Situação Problema

A empresa **MeuPortoSeguro** é uma plataforma online que conecta proprietários de imóveis a clientes interessados em alugar espaços para hospedagem, como casas, apartamentos, chalés, ou até mesmo quartos individuais. A plataforma permite que os proprietários cadastrem seus imóveis, definam preços e disponibilidade, enquanto os clientes podem pesquisar, alugar e avaliar as hospedagens.

Objetivo do Sistema:

O sistema foi projetado para gerenciar as seguintes operações:

- **Cadastro de Proprietários:** Proprietários de imóveis podem se cadastrar na plataforma fornecendo informações pessoais e de contato.
- **Cadastro de Clientes:** Clientes interessados em alugar hospedagens podem se cadastrar fornecendo seus dados pessoais.
- **Cadastro de Endereços:** Cada hospedagem precisa estar associada a um endereço completo (rua, número, bairro, cidade, estado e CEP).
- **Cadastro de Hospedagens:** Proprietários podem cadastrar suas propriedades para aluguel, informando o tipo de hospedagem (casa, apartamento, etc.) e associando-a a um endereço.
- **Registro de Aluguéis:** Clientes podem alugar hospedagens por um período específico, gerando um registro de aluguel com datas de início e fim, além do preço total.
- **Avaliações:** Após a estadia, os clientes podem avaliar a hospedagem, atribuindo uma nota e deixando um comentário.

Fluxo de Funcionamento:

Proprietário:

- Um proprietário se cadastra na plataforma fornecendo nome, CPF/CNPJ e contato.

- Ele cadastra um ou mais imóveis, informando o tipo de hospedagem e o endereço.
- O proprietário pode definir se a hospedagem está ativa ou não para aluguel.

Cliente:

- Um cliente se cadastra na plataforma fornecendo nome, CPF e contato.
- Ele pesquisa hospedagens disponíveis e realiza o aluguel de uma delas, definindo as datas de início e fim da estadia.
- Após a estadia, o cliente pode avaliar a hospedagem.

Endereço:

- Cada hospedagem deve estar associada a um endereço completo, que é cadastrado previamente.

Aluguéis:

- Cada aluguel é registrado com um ID único, associado ao cliente, à hospedagem, às datas de início e fim, e ao preço total.

Avaliações:

As avaliações são registradas com uma nota (de 1 a 5) e um comentário opcional, associadas ao cliente e à hospedagem.

Exemplo de Uso:

Cadastro de Proprietário:

Um proprietário chamado Carlos Almeida cadastra seu imóvel na plataforma. Ele fornece seu CPF, nome e contato.

Carlos cadastra um apartamento em São Paulo, fornecendo o endereço completo.

Cadastro de Cliente:

Um cliente chamado Maria Oliveira se cadastra na plataforma com seu CPF e contato.

Aluguel:

Maria aluga o apartamento de Carlos por 5 dias, pagando um total de R\$ 1.500,00.

Avaliação:

Após a estadia, Maria avalia o apartamento com nota 5 e deixa um comentário elogiando a localização e o conforto.

Tabelas e Relacionamentos:

proprietarios: Armazena os dados dos proprietários.

clientes: Armazena os dados dos clientes.

enderecos: Armazena os endereços das hospedagens.

hospedagens: Armazena as informações das hospedagens, relacionadas a um proprietário e um endereço.

alugueis: Registra os aluguéis realizados pelos clientes, relacionando cliente e hospedagem.

avaliacoes: Armazena as avaliações feitas pelos clientes sobre as hospedagens.

Criando Banco de Dados

```
create database meuportoseguro
```

```
USE meuportoseguro;
```

```
CREATE TABLE proprietarios(  
    proprietario_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(255),  
    cpf_cnpj VARCHAR(20),  
    contato VARCHAR(255)  
);
```

```
CREATE TABLE clientes (  
    cliente_id INT NOT NULL  
        AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(255),  
    cpf VARCHAR(14),  
    contato VARCHAR(255)  
);
```

```
CREATE TABLE enderecos (  
    endereco_id VARCHAR(255) PRIMARY KEY,  
    rua VARCHAR(255),  
    numero INT,  
    bairro VARCHAR(255),  
    cidade VARCHAR(255),  
    estado VARCHAR(2),  
    cep VARCHAR(10)  
);
```

```
CREATE TABLE hospedagens (  
    hospedagem_id INT NOT NULL  
        AUTO_INCREMENT PRIMARY KEY,  
    tipo VARCHAR(50),  
    endereco_id INT,  
    proprietario_id INT,  
    ativo bool,  
    FOREIGN KEY (endereco_id)  
        REFERENCES enderecos(endereco_id),  
    FOREIGN KEY (proprietario_id)  
        REFERENCES proprietarios(proprietario_id)
```

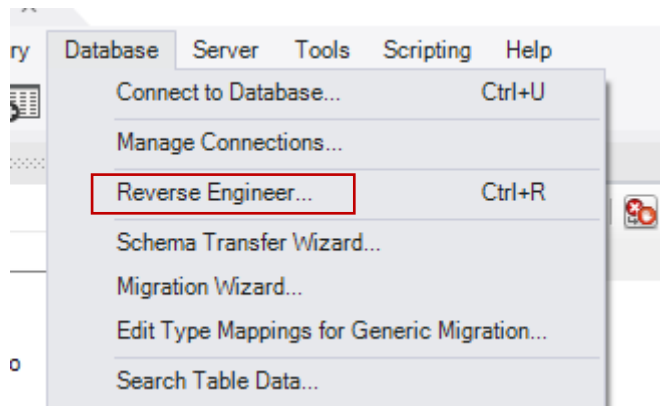
);

```
CREATE TABLE alugueis (  
    aluguel_id INT NOT NULL  
        AUTO_INCREMENT PRIMARY KEY,  
    cliente_id INT,  
    hospedagem_id INT,  
    data_inicio DATE,  
    data_fim DATE,  
    preco_total DECIMAL(10, 2),  
    FOREIGN KEY (cliente_id)  
        REFERENCES clientes(cliente_id),  
    FOREIGN KEY (hospedagem_id)  
        REFERENCES hospedagens(hospedagem_id)  
);
```

```
CREATE TABLE avaliacoes (  
    avaliacao_id VARCHAR(255) PRIMARY KEY,  
    cliente_id VARCHAR(255),  
    hospedagem_id VARCHAR(255),  
    nota INT,  
    comentario TEXT,  
    FOREIGN KEY (cliente_id) REFERENCES clientes(cliente_id),  
    FOREIGN KEY (hospedagem_id) REFERENCES  
        hospedagens(hospedagem_id)  
);
```

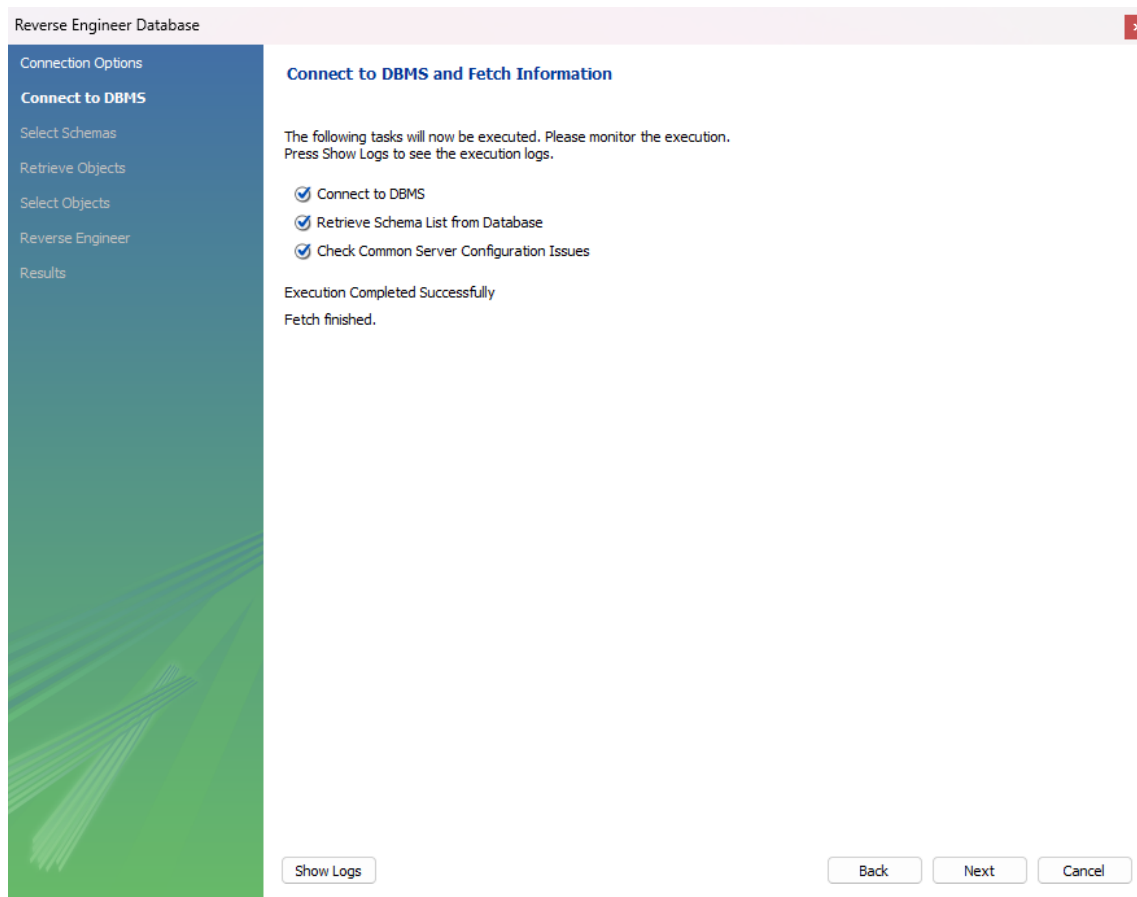

Gerando Diagrama no Workbench

Para gera o diagrama de clique no menu **Database – Reverse**.



- Clique em Avançar na tela de Reverse Engineer Database

- Aguarde



- Clique em Avançar novamente
- Na próxima tela escolha seu banco de dados e clique em avançar

Reverse Engineer Database

Connection Options

Connect to DBMS

Select Schemas

Retrieve Objects

Select Objects

Reverse Engineer

Results

Select Schemas to Reverse Engineer

Select the schemas you want to include:

☐ bdecommerce
☐ db_editora
☒ db_meuportoseguro
☐ db_vendas
☐ dbescola
☐ dbserenatto
☐ testebd

Back

Next

Cancel

- Aguarde e avance novamente

Reverse Engineer Database

Connection Options

Connect to DBMS

Select Schemas

Retrieve Objects

Select Objects

Reverse Engineer

Results

Retrieve and Reverse Engineer Schema Objects

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

☒ Retrieve Objects from Selected Schemas
☒ Check Results

Retrieval Completed Successfully

Finished.

Show Logs

Back

Next


Cancel

- Na próxima tela clique em Execute

Reverse Engineer Database

Connection Options
Connect to DBMS
Select Schemas
Retrieve Objects
Select Objects
Reverse Engineer
Results

Select Objects to Reverse Engineer

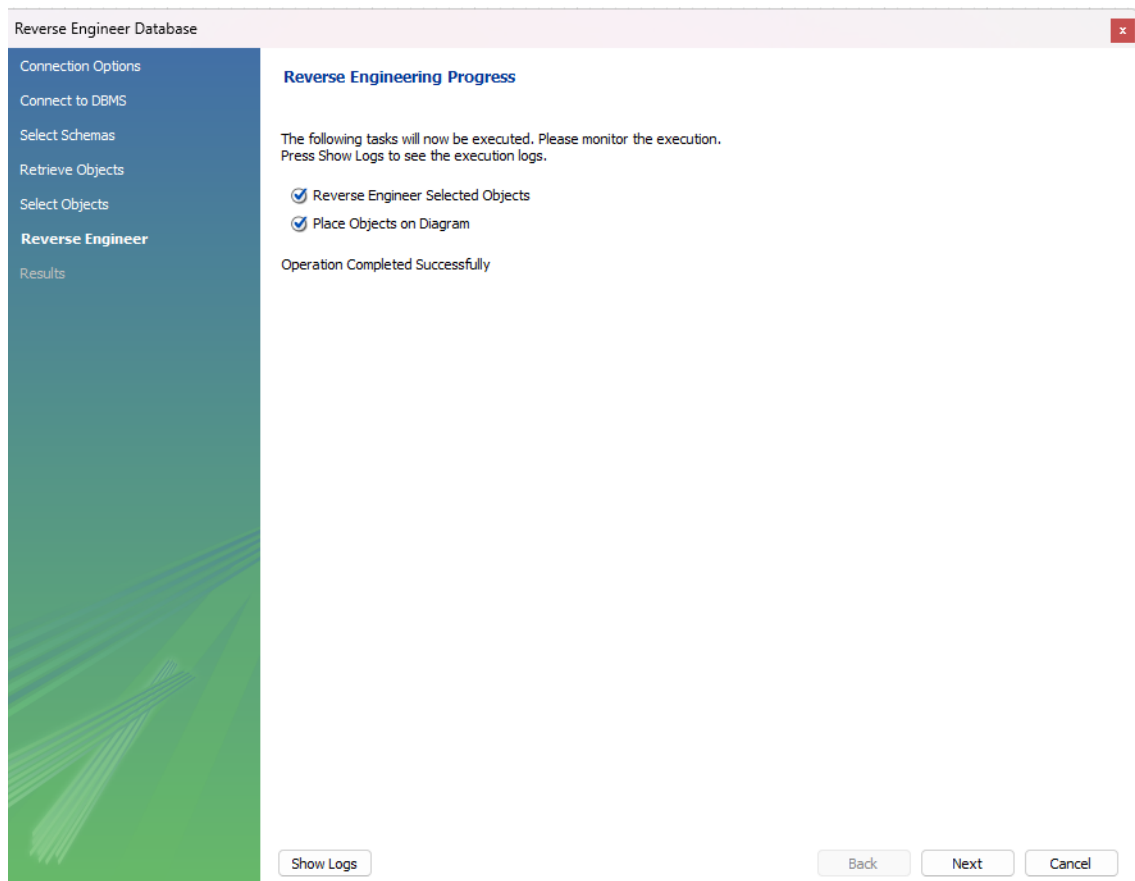
 ☒ Import MySQL Table Objects Show Filter

6 Total Objects, 6 Selected

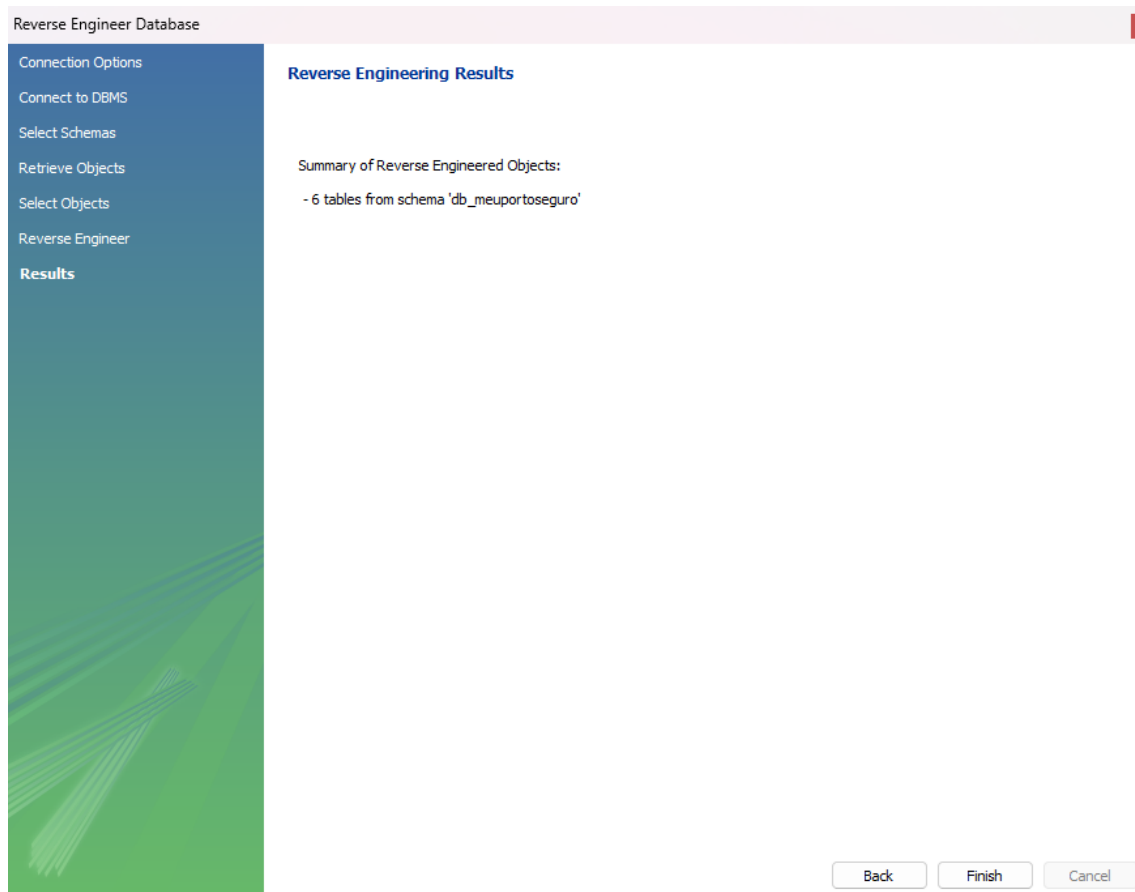
☒ Place imported objects on a diagram

Back Execute > Cancel

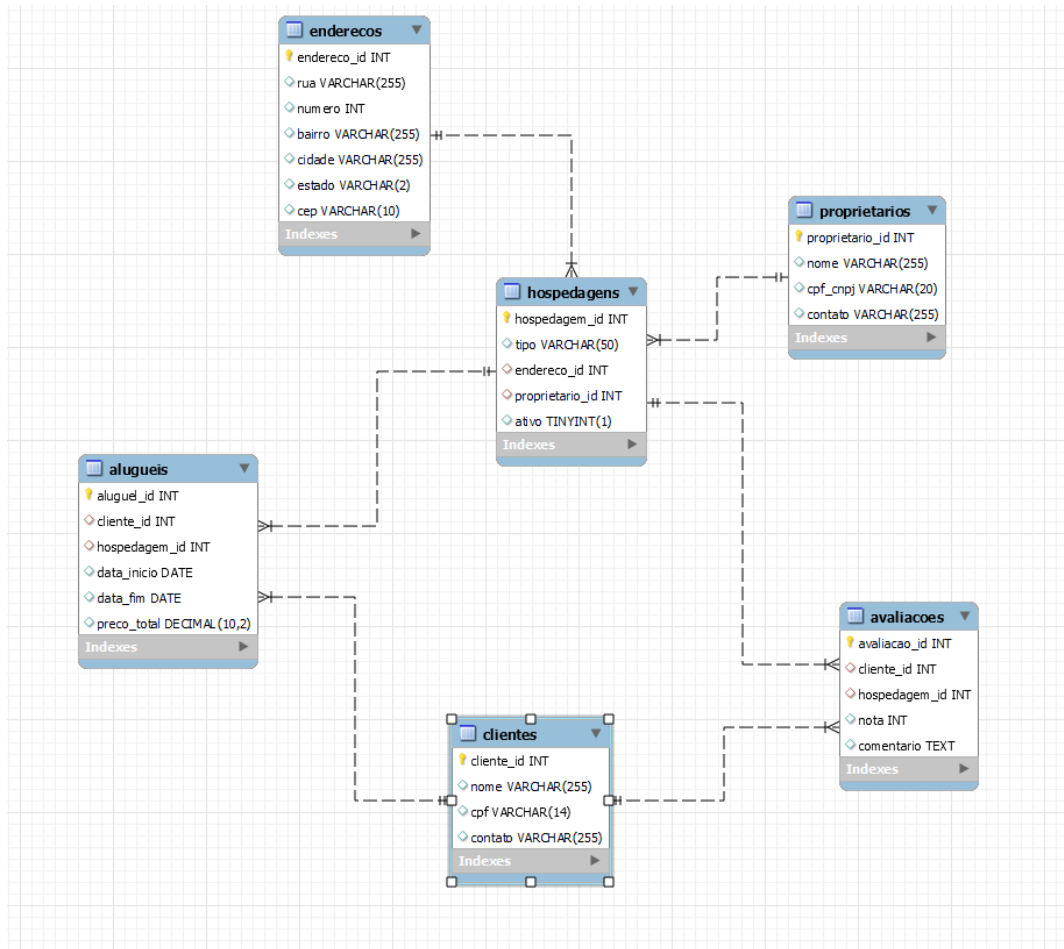
- Aguarde, e avance



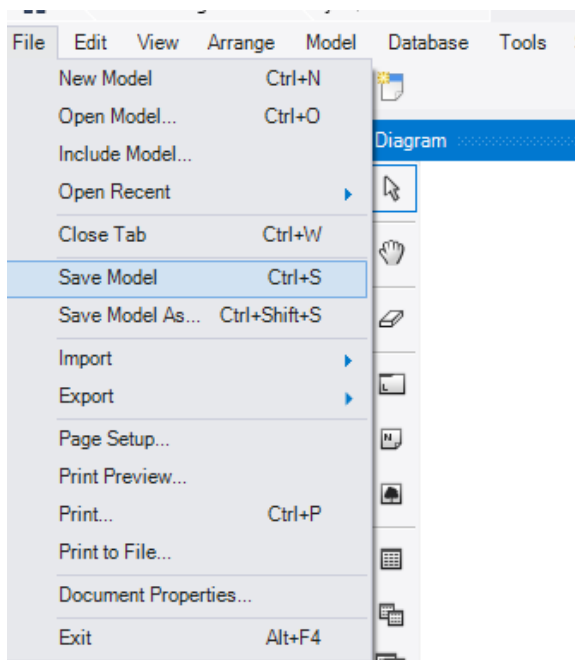
- Veja o resultado e clique em finish



- O Diagrama será criado



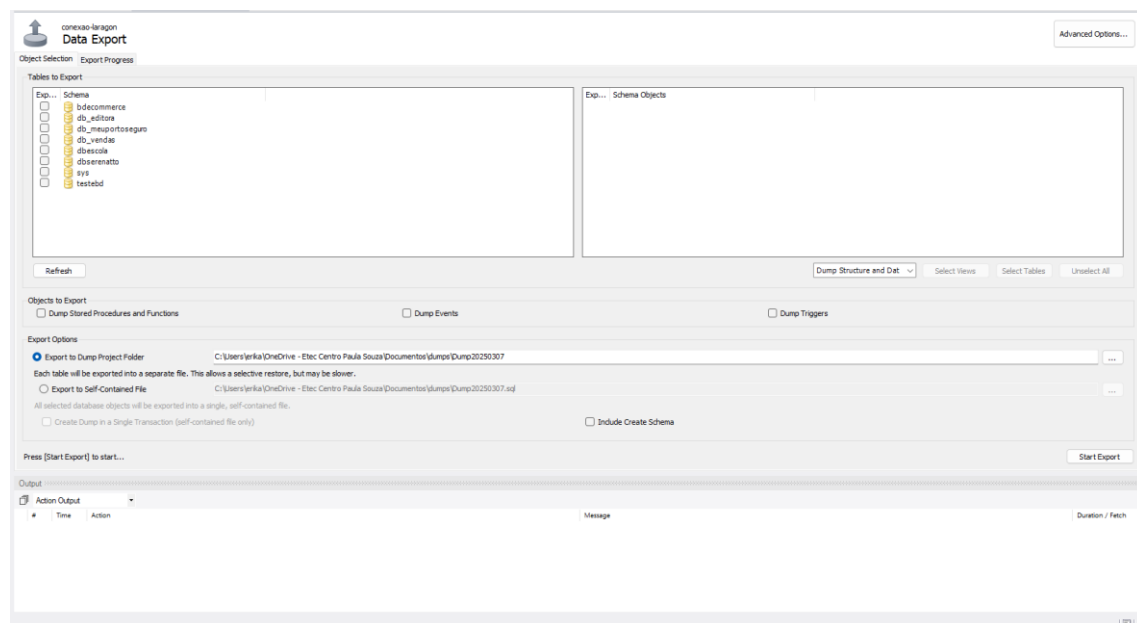
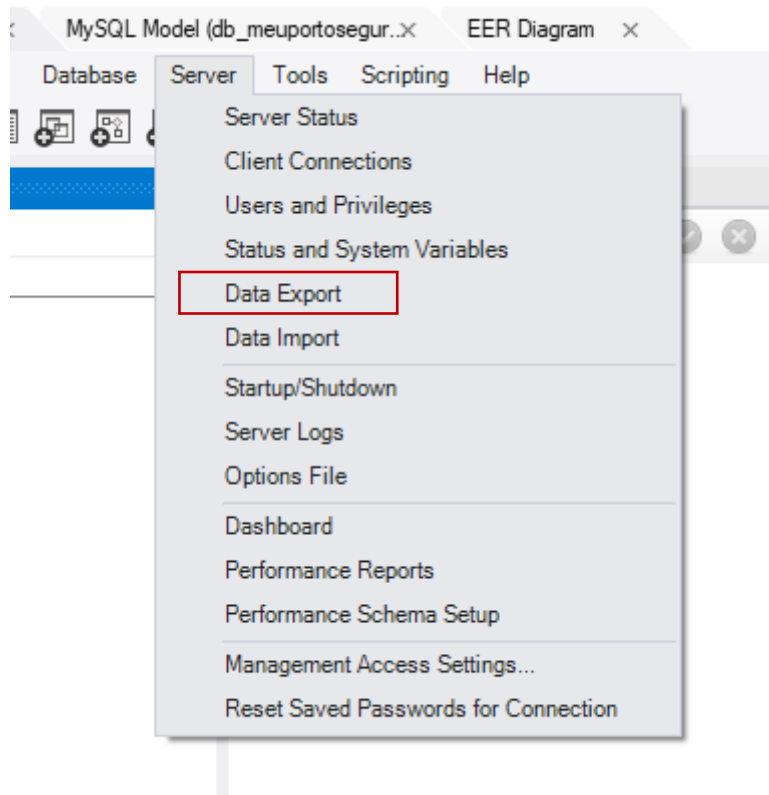
Para salvar o diagrama clique em File – Save Model



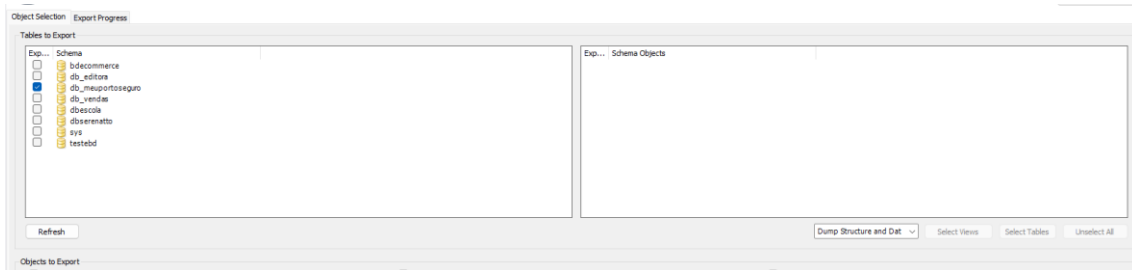
- Adicione um nome e salve

Salvando Script do Banco de Dados

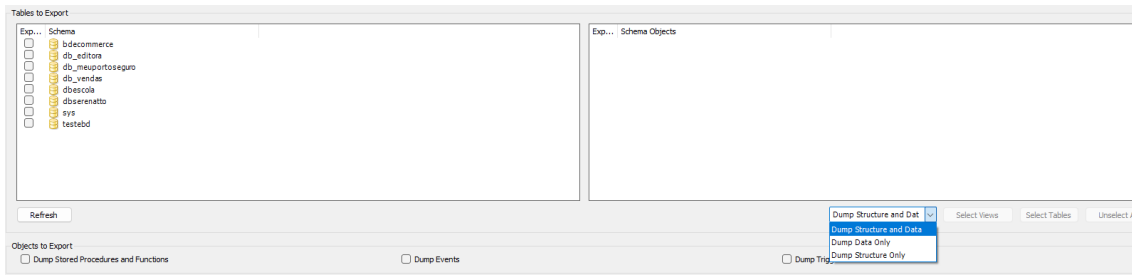
Para gerar os scripts do banco de dados criado, volte à janela de conexão e clique em Server – Data Export.



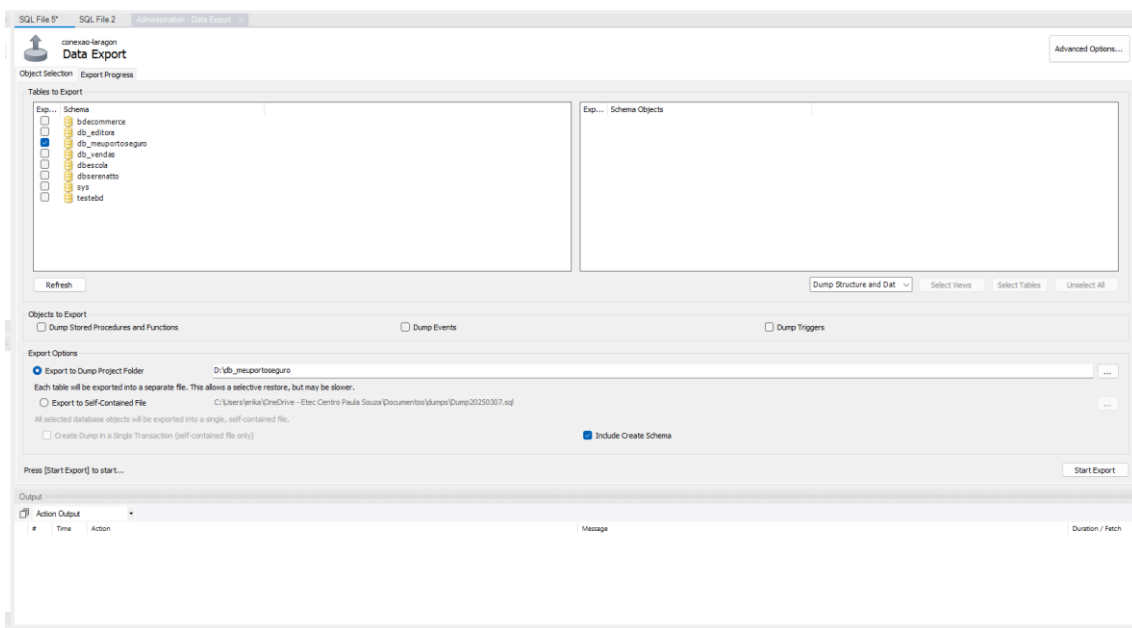
Na janela aberta selecione o banco de dados, db_meuportoseguro



- Selecione a opção: **Dump Structure and Data**









- Em **Export to Dump Project Folder**, selecione o caminho para salvar os scripts e selecione a opção **Include Create Schema**
- Clique em **Start Export**



- Clique em **OK** e aguarde

Será gerado um script para tabela do banco de dados

Nome	Data de modificação	tipo	tamanho
 db_meuportoseguro_alugueis	07/03/2025 17:24	Arquivo Fonte SQL	3 KB
 db_meuportoseguro_avaliacoes	07/03/2025 17:24	Arquivo Fonte SQL	3 KB
 db_meuportoseguro_clientes	07/03/2025 17:24	Arquivo Fonte SQL	3 KB
 db_meuportoseguro_enderecos	07/03/2025 17:24	Arquivo Fonte SQL	10 KB
 db_meuportoseguro_hospedagens	07/03/2025 17:24	Arquivo Fonte SQL	3 KB
 db_meuportoseguro_proprietarios	07/03/2025 17:24	Arquivo Fonte SQL	3 KB

Comandos SQL

use db_meuportoseguro;

```
INSERT INTO enderecos VALUES ('Trecho de Moraes', '5', 'Baleia',  
'Teixeira', 'RR', '62030-547');
```

```
SELECT * FROM enderecos;
```

```
INSERT INTO enderecos VALUES ('Trecho de Moraes', '5', 'Baleia',  
'Teixeira', 'RR', '62030-547');
```

```
SELECT * FROM endereços;
```

```
SELECT * FROM alugueis;
```

```
SELECT * avaliacoes;
```

```
SELECT * clientes;
```

```
SELECT * enderecos;
```

```
SELECT * hospedagens;
```

```
SELECT * proprietarios;
```

Consulta para filtrar as melhores hospedagens

```
SELECT * FROM avaliacoes
```

```
WHERE nota >= 4
```

Consulta por tipo de hospedagem

```
SELECT * FROM hospedagens
```

```
WHERE tipo = 'hotel' AND ativo = 1;
```

Ticket Médio


```
SELECT cliente_id, AVG(preco_total) AS ticket_medio  
  
FROM alugueis  
  
GROUP BY cliente_id;
```

Média de tempo de cada reserva

```
SELECT cliente_id, AVG(DATEDIFF(data_fim, data_inicio)) AS  
media_dias_estadia  
  
FROM alugueis  
  
GROUP BY cliente_id  
  
ORDER BY media_dias_estadia DESC;
```

Identificando propriedades mais ativas

```
SELECT p.nome AS nome_proprietario, COUNT(h.hospedagem_id) AS  
total_hospedagens_ativas  
  
FROM proprietarios p  
  
JOIN hospedagens h ON p.proprietario_id = h.proprietario_id  
  
WHERE h.ativo = 1  
  
GROUP BY p.nome  
  
ORDER BY total_hospedagens_ativas DESC  
  
LIMIT 10;
```

Identificando hospedagens inativas

```
SELECT p.nome AS nome_proprietario, COUNT(*) AS  
total_hospedagens_inativas  
  
FROM proprietarios p  
  
JOIN hospedagens h ON p.proprietario_id = h.proprietario_id  
  
WHERE h.ativo = 0  
  
GROUP BY p.nome;
```

Como Funciona

O JOIN conecta linhas de duas tabelas criando um conjunto de resultados que inclui todas as colunas das tabelas envolvidas. O tipo mais comum de JOIN é o INNER JOIN, que retorna apenas as linhas que têm correspondência em ambas as tabelas. Além do INNER JOIN, existem outros tipos, como LEFT JOIN, RIGHT JOIN, e FULL OUTER JOIN, cada um com uma maneira específica de tratar a inclusão de linhas sem correspondência.

Características e Tipos de JOIN

- **INNER JOIN:** Retorna linhas quando há pelo menos uma correspondência em ambas as tabelas.
- **LEFT JOIN** (ou LEFT OUTER JOIN): Retorna todas as linhas da tabela à esquerda e as linhas correspondentes da tabela à direita. Se não houver correspondência, o resultado é NULL no lado direito.
- **RIGHT JOIN** (ou RIGHT OUTER JOIN): O inverso do LEFT JOIN, retorna todas as linhas da tabela à direita e as correspondentes da tabela à esquerda.
- **FULL OUTER JOIN:** Combina LEFT JOIN e RIGHT JOIN, retornando todas as linhas quando há uma correspondência em uma das tabelas.

Quando Utilizar

O JOIN é utilizado em diversas situações, como:

- Para combinar dados relacionados armazenados em tabelas separadas, como clientes e pedidos, onde cada pedido está associado a um cliente específico.
- Para realizar análises complexas que requerem a combinação de informações de múltiplas fontes de dados.
- Para enriquecer conjuntos de resultados com informações adicionais de tabelas relacionadas, melhorando a compreensão e a utilidade dos dados.

Particularidades no MySQL

No MySQL, o uso do JOIN segue os princípios padrões do SQL, mas com algumas particularidades que valem a pena destacar:

- **Performance:** O MySQL otimiza automaticamente as consultas com JOIN, mas o desempenho pode ser afetado pela estrutura das tabelas, índices, e pela complexidade da consulta. É essencial garantir que as colunas usadas para o JOIN estejam indexadas.
- **Sintaxe de Alias:** O MySQL permite o uso de aliases para tabelas e colunas nas consultas com JOIN, o que pode simplificar a consulta e melhorar a legibilidade, especialmente quando as tabelas ou colunas têm nomes longos ou quando a mesma tabela é usada múltiplas vezes na consulta.
- **Tipo de JOIN:** Embora o FULL OUTER JOIN não seja diretamente suportado pelo MySQL, é possível simular esse comportamento combinando LEFT JOIN e RIGHT JOIN com UNION.

Tipos Numéricos

1. **INT**: Um tipo de dado inteiro que pode ser assinado (negativo e positivo) ou não assinado (apenas positivo).
2. **TINYINT, SMALLINT, MEDIUMINT, BIGINT**: Variações do tipo inteiro que oferecem diferentes intervalos de valores, economizando espaço para dados com limites conhecidos.
3. **FLOAT, DOUBLE**: Tipos de dados de ponto flutuante para armazenamento de números reais, com DOUBLE oferecendo maior precisão.
4. **DECIMAL**: Utilizado para armazenar valores decimais exatos, como valores monetários, onde a precisão é crucial.

Tipos de Dados de Data e Hora

1. **DATE**: Armazena datas no formato AAAA-MM-DD.
2. **TIME**: Representa horários no formato HH:MM:SS.
3. **DATETIME**: Combinação de data e hora no formato AAAA-MM-DD HH:MM:SS, útil para registrar eventos exatos no tempo.
4. **TIMESTAMP**: Semelhante a DATETIME, mas usado para rastrear mudanças em registros. É automaticamente atualizado quando o registro é modificado.
5. **YEAR**: Armazena um ano no formato AA ou AAAA.

Tipos de Dados de String

1. **CHAR** e **VARCHAR**: Ambos são usados para armazenar cadeias de caracteres, com CHAR para comprimentos fixos e VARCHAR para comprimentos variáveis.
2. **TEXT**: Para textos longos, como comentários ou artigos. Existem variações como TINYTEXT, MEDIUMTEXT e LONGTEXT, diferenciando-se pelo tamanho máximo do texto que podem armazenar.
3. **BINARY** e **VARBINARY**: Semelhantes a CHAR e VARCHAR, mas para dados binários.

Tipos de Dados Lógicos

1. **BOOLEAN**: Representa valores verdadeiros ou falsos, internamente implementado como TINYINT(1), onde 0 significa falso e 1 significa verdadeiro.

Tipos de Dados Espaciais

MySQL também suporta tipos de dados espaciais, que são úteis para armazenar informações geográficas, como pontos, linhas e polígonos. Esses tipos são essenciais para aplicações que realizam cálculos geográficos ou armazenam dados de localização.

Conjuntos e Enumerações

1. **ENUM**: Permite definir uma lista de possíveis valores em uma coluna, e cada registro nessa coluna deve ser um desses valores ou NULL.
2. **SET**: Semelhante ao ENUM, mas permite armazenar zero ou mais valores de uma lista predefinida.

Listar todas as hospedagens ativas em uma cidade:

```
SELECT h.hospedagem_id, h.tipo, e.cidade, e.bairro, e.rua, e.numero  
FROM hospedagens h  
JOIN enderecos e ON h.endereco_id = e.endereco_id  
WHERE h.ativo = TRUE AND e.cidade = 'São Paulo';
```

Listar todos os aluguéis de um cliente:

```
SELECT a.aluguel_id, h.tipo, a.data_inicio, a.data_fim, a.preco_total  
FROM alugueis a  
JOIN hospedagens h ON a.hospedagem_id = h.hospedagem_id  
WHERE a.cliente_id = 'ID_DO_CLIENTE';
```

Calcular a média de avaliações de uma hospedagem:

```
SELECT AVG(av.nota) AS media_avaliacoes  
  
FROM avaliacoes av  
  
WHERE av.hospedagem_id = 'ID_DA_HOSPEDAGEM';
```

Listar todas as hospedagens de um proprietário:

```
SELECT h.hospedagem_id, h.tipo, e.cidade, e.bairro  
  
FROM hospedagens h  
  
JOIN enderecos e ON h.endereco_id = e.endereco_id  
  
WHERE h.proprietario_id = 'ID_DO_PROPRIETARIO';
```