BANCO DE DADOS RELACIONAIS

Exercício Avaliativo

**Obs. Salvar os scripts e os modelos em um arquivo WORD e nomear o arquivo com o nome do aluno impreterivelmente.**

PARTE 1

1) Criar os modelos: lógico e físico para o cenário descrito abaixo:

1. Trata-se do clássico exemplo de uma locadora de filmes onde uma rede de lojas possui várias lojas que alugam filmes. Loja: id, gerente

Cliente: id, nome, email, ativo

Funcionario: id, nome, foto, email, ativo, login, senha

Ator: id, nome

Categoria: id, nome

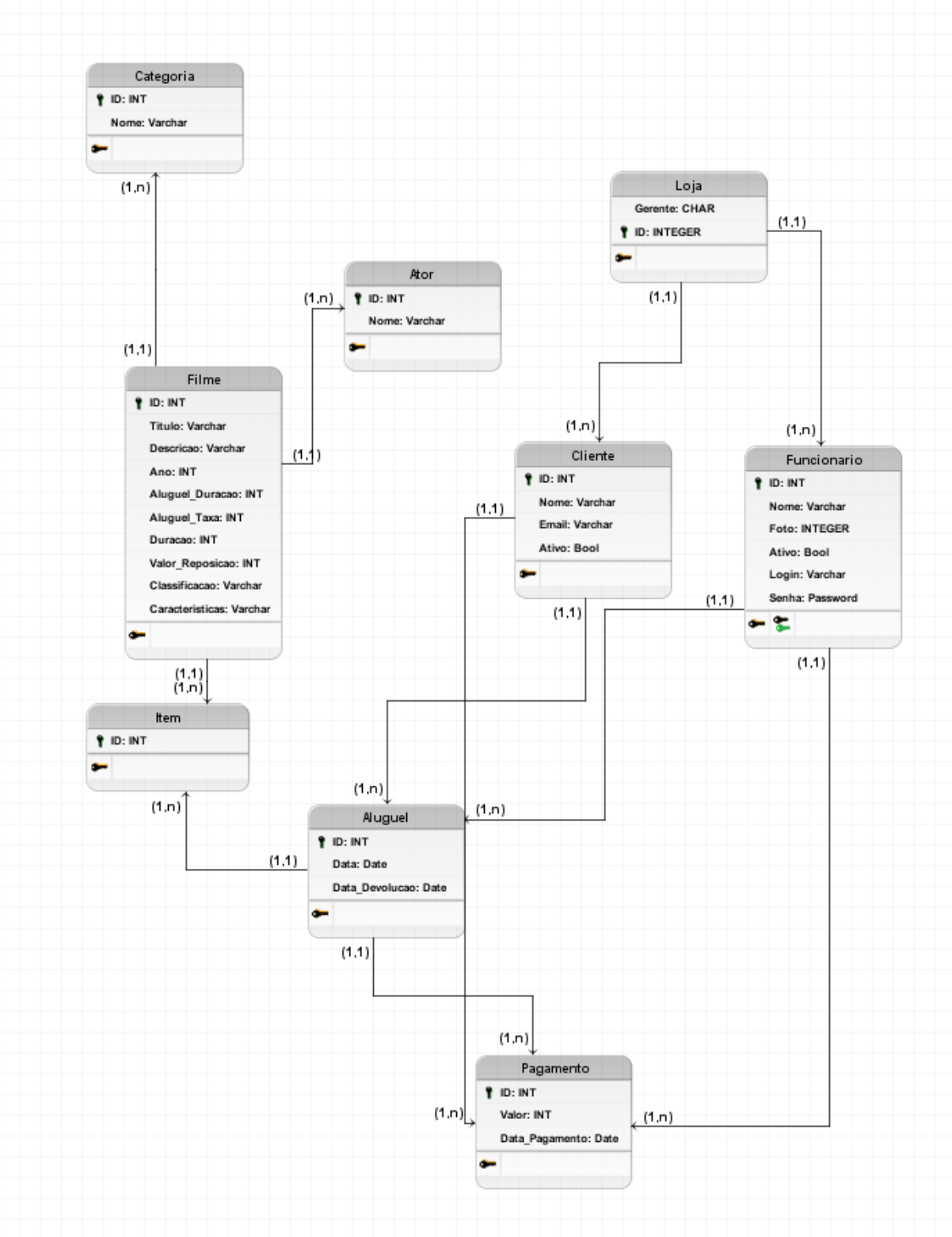
Filme: id, titulo, descricao, ano, aluguel\_duracao, aluguel\_taxa, duracao, valor\_reposicao, classificação, caracteristicas

Aluguel: id, data, data\_devolucao

Item: id

Pagamento: id, valor, data\_pagamento

1. São necessárias as seguintes relações:
   * Um filme possui várias categorias, e cada categoria possui vários filmes.
   * Um filme possui vários atores e cada ator atua em vários filmes.
   * Uma loja possui vários clientes e cada cliente é de uma loja.
   * Uma loja possui vários funcionários e cada funcionário é de uma loja.
   * O gerente da loja é um funcionário.
   * Um cliente realiza vários aluguéis e cada aluguel é de um cliente.
   * Um funcionário registra vários aluguéis e cada aluguel é registrado por um funcionário.
   * Um aluguel pode gerar vários pagamentos, e cada pagamento é de um aluguel.
   * Um funcionário emite vários pagamentos e cada pagamento é emitido por um funcionário.
   * Um cliente possui vários pagamentos e cada pagamento é de um cliente.
   * Um aluguel possui vários itens e cada item é de um aluguel.
   * Um item é um filme e cada filme pode ser vários itens.



PARTE 2

2) A partir do modelo físico criado na PARTE 1:

1. Criar as visões (VIEW) a seguir:
   * Listagem dos atores e para cada ator uma descrição contendo os filmes nos quais ele atuou.
   * Listar os filmes com suas categorias e atores.
   * Listar os 5 filmes mais alugados.
   * Listar os pagamentos em aberto e seus respectivos clientes.
   * Listagem das lojas, com seu gerente.
   * Listar os clientes das lojas em ordem alfabética por loja e cliente

Criação das tabelas + Views:

**CREATE** **TABLE** Categoria (

CategoriaID **INT** **PRIMARY** **KEY**,

Nome **VARCHAR**(255)

);

**CREATE** **TABLE** Ator (

AtorID **INT** **PRIMARY** **KEY**,

Nome **VARCHAR**(255)

);

**CREATE** **TABLE** Filme (

FilmeID **INT** **PRIMARY** **KEY**,

Título **VARCHAR**(255)

);

**CREATE** **TABLE** FilmeCategoria (

FilmeID **INT**,

CategoriaID **INT**,

**FOREIGN** **KEY** (FilmeID) **REFERENCES** Filme(FilmeID),

**FOREIGN** **KEY** (CategoriaID) **REFERENCES** Categoria(CategoriaID)

);

**CREATE** **TABLE** FilmeAtor (

FilmeID **INT**,

AtorID **INT**,

**FOREIGN** **KEY** (FilmeID) **REFERENCES** Filme(FilmeID),

**FOREIGN** **KEY** (AtorID) **REFERENCES** Ator(AtorID)

);

**CREATE** **TABLE** Loja (

LojaID **INT** **PRIMARY** **KEY**,

Nome **VARCHAR**(255)

);

**CREATE** **TABLE** Cliente (

ClienteID **INT** **PRIMARY** **KEY**,

Nome **VARCHAR**(255),

LojaID **INT**,

**FOREIGN** **KEY** (LojaID) **REFERENCES** Loja(LojaID)

);

**CREATE** **TABLE** Funcionario (

FuncionarioID **INT** **PRIMARY** **KEY**,

Nome **VARCHAR**(255),

LojaID **INT**,

**FOREIGN** **KEY** (LojaID) **REFERENCES** Loja(LojaID)

);

**CREATE** **TABLE** Gerente (

GerenteID **INT** **PRIMARY** **KEY**,

FuncionarioID **INT**,

**FOREIGN** **KEY** (FuncionarioID) **REFERENCES** Funcionario(FuncionarioID)

);

**CREATE** **TABLE** Aluguel (

AluguelID **INT** **PRIMARY** **KEY**,

ClienteID **INT**,

FuncionarioID **INT**,

**FOREIGN** **KEY** (ClienteID) **REFERENCES** Cliente(ClienteID),

**FOREIGN** **KEY** (FuncionarioID) **REFERENCES** Funcionario(FuncionarioID)

);

**CREATE** **TABLE** Pagamento (

PagamentoID **INT** **PRIMARY** **KEY**,

AluguelID **INT**,

FuncionarioID **INT**,

**FOREIGN** **KEY** (AluguelID) **REFERENCES** Aluguel(AluguelID),

**FOREIGN** **KEY** (FuncionarioID) **REFERENCES** Funcionario(FuncionarioID)

);

**CREATE** **TABLE** Item (

ItemID **INT** **PRIMARY** **KEY**,

AluguelID **INT**,

FilmeID **INT**,

**FOREIGN** **KEY** (AluguelID) **REFERENCES** Aluguel(AluguelID),

**FOREIGN** **KEY** (FilmeID) **REFERENCES** Filme(FilmeID)

);

**CREATE** **TABLE** Estoque (

EstoqueID **INT** **PRIMARY** **KEY**,

LojaID **INT**,

FilmeID **INT**,

**FOREIGN** **KEY** (LojaID) **REFERENCES** Loja(LojaID),

**FOREIGN** **KEY** (FilmeID) **REFERENCES** Filme(FilmeID)

);

**CREATE** **VIEW** **AS**

**SELECT** A.AtorID, A.Nome, **GROUP\_CONCAT**(F.Título **SEPARATOR** ', ') **AS** FilmesAtuados

**FROM** Ator A

**JOIN** FilmeAtor FA **ON** A.AtorID = FA.AtorID

**JOIN** Filme F **ON** FA.FilmeID = F.FilmeID

**GROUP** **BY** A.AtorID, A.Nome;

**CREATE** **VIEW** FilmesCategoriasAtores **AS**

**SELECT** F.Título, **GROUP\_CONCAT**(C.Nome **SEPARATOR** ', ') **AS** Categorias, **GROUP\_CONCAT**(A.Nome **SEPARATOR** ', ') **AS** Atores

**FROM** Filme F

**JOIN** FilmeCategoria FC **ON** F.FilmeID = FC.FilmeID

**JOIN** Categoria C **ON** FC.CategoriaID = C.CategoriaID

**JOIN** FilmeAtor FA **ON** F.FilmeID = FA.FilmeID

**JOIN** Ator A **ON** FA.AtorID = A.AtorID

**GROUP** **BY** F.Título;

**CREATE** **VIEW** FilmesMaisAlugados **AS**

**SELECT** F.Título, **COUNT**(\*) **AS** TotalAlugué**is**

**FROM** Filme F

**JOIN** Item I **ON** F.FilmeID = I.FilmeID

**GROUP** **BY** F.Título

**ORDER** **BY** TotalAlugué**is** **DESC**

**LIMIT** 5;

**CREATE** **VIEW** ClientesPorLoja **AS**

**SELECT** L.LojaID, L.Nome **AS** Loja, C.Nome **AS** Cliente

**FROM** Loja L

**JOIN** Cliente C **ON** L.LojaID = C.LojaID

**ORDER** **BY** L.Nome, C.Nome;

**CREATE** **VIEW** LojasComGerente **AS**

**SELECT** L.LojaID, L.Nome **AS** Loja, F.Nome **AS** Gerente

**FROM** Loja L

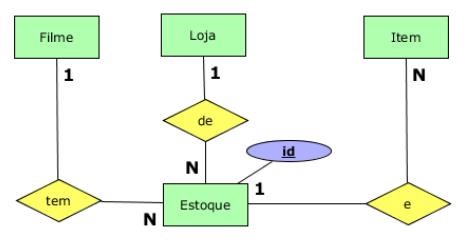
**JOIN** Funcionario F **ON** L.LojaID = F.LojaID

**JOIN** Gerente G **ON** F.FuncionarioID = G.FuncionarioID;

1. Acrescentar a tabela Estoque e os relacionamentos no modelo físico, seguindo o modelo abaixo:

Obs: Para cada filme em uma loja um novo registro é inserido na tabela. Por exemplo, se a loja A tem um estoque de 5 filmes X e a loja B tem um estoque de 3 filmes X, então haverá 8 registros na tabela de estoque do filme X.

Obs: Cada item alugado terá uma chave estrangeira correspondente a um item específico do estoque.



Modelo Físico:  
  
**CREATE** **TABLE** Estoque (

EstoqueID **INT** **PRIMARY** **KEY**,

LojaID **INT**,

FilmeID **INT**,

**FOREIGN** **KEY** (LojaID) **REFERENCES** Loja(LojaID),

**FOREIGN** **KEY** (FilmeID) **REFERENCES** Filme(FilmeID)

);

**ALTER** **TABLE** Item

**ADD** EstoqueID **INT**,

**ADD** **FOREIGN** **KEY** (EstoqueID) **REFERENCES** Estoque(EstoqueID);

1. Criar a seguinte Function:

• Inventario\_em\_Estoque (estoque\_id INT) RETURNS tinyint(1) – Retorna 1 quando o item está no estoque ou 0 quando o item está emprestado. Obs: Um item consta no estoque se não existir nenhum registro deste item alugado ou se os registros que existirem estiverem com a data de devolução preenchida.   
  
Function:

**CREATE** **FUNCTION** Inventario\_em\_Estoque (estoque\_id **INT**) **RETURNS** **TINYINT**

**BEGIN**

**DECLARE** item\_count **INT**;

**DECLARE** borrowed\_count **INT**;

**SELECT** **COUNT**(\*) **INTO** item\_count

**FROM** Item

**WHERE** FilmeID = estoque\_id;

**SELECT** **COUNT**(\*) **INTO** borrowed\_count

**FROM** Item

**JOIN** Aluguel **ON** Item.AluguelID = Aluguel.AluguelID

**WHERE** Item.FilmeID = estoque\_id **AND** Aluguel.DataDevolucao **IS** **NULL**;

**IF** item\_count = 0 **OR** item\_count = borrowed\_count **THEN**

**RETURN** 1; **ELSE**

**RETURN** 0; **END** **IF**;

**END** //

DELIMITER ;

1. Criar as seguintes Stored Procedures:
   * + Ator(IN quantidade INT) – Listar o nome dos atores limitando ao número de registros a serem retornados.
     + Filmes\_em\_Estoque(IN filme\_id INT, IN loja\_id INT, OUT filme\_qtde INT) – Retornar a quantidade de filmes no estoque, informando o id do filme e o id da loja.
     + Filmes\_alugados(IN filme\_id INT, IN loja\_id INT, OUT filme\_qtde INT) – Retorna a quantidade de filmes alugados, informados o id do filme e o id da loja. Um filme alugado tem a data da devolução NULL.

Todas as Procedures:

Ator:   
  
DELIMITER //

**CREATE** **PROCEDURE** Ator(**IN** quantidade **INT**)

**BEGIN**

**SELECT** Nome

**FROM** Ator

**LIMIT** quantidade;

**END** //

DELIMITER ;

Filmes\_em\_estoque:  
  
DELIMITER //

**CREATE** **PROCEDURE** Filmes\_em\_Estoque(**IN** filme\_id **INT**, **IN** loja\_id **INT**, **OUT** filme\_qtde **INT**)

**BEGIN**

**SELECT** **COUNT**(\*) **INTO** filme\_qtde

**FROM** Estoque

**WHERE** FilmeID = filme\_id **AND** LojaID = loja\_id;

**END** //

DELIMITER ;

Filmes\_alugados:  
  
DELIMITER //

**CREATE** **PROCEDURE** Filmes\_alugados(**IN** filme\_id **INT**, **IN** loja\_id **INT**, **OUT** filme\_qtde **INT**)

**BEGIN**

**SELECT** **COUNT**(\*) **INTO** filme\_qtde

**FROM** Item

**JOIN** Aluguel **ON** Item.AluguelID = Aluguel.AluguelID

**WHERE** Item.FilmeID = filme\_id **AND** Aluguel.LojaID = loja\_id **AND** Aluguel.DataDevolucao **IS** **NULL**;

**END** //

DELIMITER ;

1. Alterar a base de dados:

Alterar a tabela cliente e acrescentar um campo com o nome “criado\_em” do tipo “datetime”.

**ALTER** **TABLE** cliente

**ADD** criado\_em **DATETIME**;

1. Criar uma trigger “cliente\_BEFORE\_UPDATE”
   * A trigger deve atualizar a data de criação de cada registro automaticamente. g) Criar a tabela “log”
   * A tabela deve ter dois campos “reg” e “msg”, ambos do tipo varchar(255).

Trigger de cliente\_before\_UPDATE

DELIMITER //

**CREATE** **TRIGGER** cliente\_BEFORE\_UPDATE

**BEFORE** **UPDATE** **ON** cliente

**FOR** **EACH** **ROW**

**BEGIN**

**SET** **NEW**.criado\_em = **NOW**();

**INSERT** **INTO** **log** (reg, msg)

**VALUES** ('cliente', 'Record updated');

**END** //

DELIMITER ;

h) Criar uma trigger “cliente\_AFTER\_UPDATE”

• A trigger deve inserir um registro na tabela “log” contendo as informações de cada nova inserção na tabela cliente.

DELIMITER //

**CREATE** **TRIGGER** cliente\_AFTER\_UPDATE

**AFTER** **UPDATE** **ON** cliente

**FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** **log** (reg, msg)

**VALUES** ('cliente', **CONCAT**('Record updated - ClienteID: ', **NEW**.ClienteID, ', Nome: ', **NEW**.Nome, ', LojaID: ', **NEW**.LojaID));

**END** //

DELIMITER ;