|  |
| --- |
| **Aula Prática SQL – Programas Armazenados II** |

Dado o esquema de tabelas, abaixo, para um sistema de vídeo locadora:



Usar o SCRIPT abaixo, para facilitar a criação das tabelas:

CREATE TABLE CATEGORIAS (

COD\_CATEGORIA NUMBER NOT NULL,

DESCRICAO VARCHAR2(80) NOT NULL,

PRECO\_LOCACAO NUMBER(5,2) NOT NULL

);

ALTER TABLE CATEGORIAS

ADD ( CONSTRAINT XPKCATEGORIAS PRIMARY KEY (COD\_CATEGORIA) ) ;

CREATE TABLE CLIENTE (

COD\_CLIENTE NUMBER NOT NULL,

NOME VARCHAR2(128) NOT NULL,

RUA VARCHAR2(100) NOT NULL,

NUMERO NUMBER NOT NULL,

COMPLEMENTO VARCHAR2(100) NULL,

BAIRRO VARCHAR2(100) NOT NULL,

CIDADE VARCHAR2(50) NOT NULL

);

ALTER TABLE CLIENTE

ADD ( CONSTRAINT XPKCLIENTE PRIMARY KEY (COD\_CLIENTE) ) ;

CREATE TABLE FILMES (

COD\_FILME NUMBER NOT NULL,

DESCRICAO VARCHAR2(128) NOT NULL,

DURACAO NUMBER NOT NULL,

COD\_CATEGORIA NUMBER NOT NULL

);

ALTER TABLE FILMES

ADD ( CONSTRAINT XPKFILMES PRIMARY KEY (COD\_FILME) ) ;

CREATE TABLE FITAS (

COD\_FITA NUMBER NOT NULL,

DESCRICAO VARCHAR2(60) NOT NULL,

DATA\_COMPRA DATE NOT NULL,

COD\_FILME NUMBER NOT NULL,

DISPONIVEL VARCHAR2(1) NOT NULL

);

ALTER TABLE FITAS

ADD ( CONSTRAINT XPKFITAS PRIMARY KEY (COD\_FITA) ) ;

CREATE TABLE LOCACAO (

COD\_CLIENTE NUMBER NOT NULL,

COD\_FITA NUMBER NOT NULL,

DATA\_LOCACAO DATE NOT NULL,

DATA\_DEVOLUCAO DATE NULL,

VALOR\_LOCACAO NUMBER(5,2) NOT NULL

);

ALTER TABLE LOCACAO

ADD ( CONSTRAINT XPKLOCACAO PRIMARY KEY (COD\_CLIENTE,

COD\_FITA, DATA\_LOCACAO) ) ;

ALTER TABLE FILMES

ADD ( CONSTRAINT FK\_CATEGORIA\_FILME

FOREIGN KEY (COD\_CATEGORIA)

REFERENCES CATEGORIAS ) ;

ALTER TABLE FITAS

ADD ( CONSTRAINT FK\_FILME\_FITA

FOREIGN KEY (COD\_FILME)

REFERENCES FILMES ) ;

ALTER TABLE LOCACAO

ADD ( CONSTRAINT FK\_FITA\_LOCACAO

FOREIGN KEY (COD\_FITA)

REFERENCES FITAS ) ;

ALTER TABLE LOCACAO

ADD ( CONSTRAINT FK\_CLIENTE\_LOCACAO

FOREIGN KEY (COD\_CLIENTE)

REFERENCES CLIENTE ) ;

**ATENÇÃO: Alguns dos exercícios abaixo deverão ter uma COMPROVAÇÂO de funcionamento. Na ausência dessa comprovação, os exercícios terão valor reduzido na avaliação.**

**1 -** Criar SEQUENCES que posteriormente serão usadas para geração de chave primária para as tabelas CLIENTE, FILMES, CATEGORIAS E FITAS, ou seja, deverão ser criadas 4 SEQUENCES. Todas as SEQUENCES deverão iniciar em 1, ser incrementada sempre de 1 em 1 e ter valor como valor máximo o maior valor permitido para o ORACLE. Por serem SEQUENCES para uso como chave primária da tabela, não poderão ser cíclicas.

**2 -** Criar 4 TRIGGER’s para geração automática de chave primária das tabelas CLIENTE, FILMES, CATEGORIAS E FITAS. Lembre-se de usar as SEQUENCES criadas no exercício anterior para as tabelas corretas.

**3 -** Criar os comandos INSERT para incluir valores nas tabelas CLIENTE, FILMES, CATEGORIAS E FITAS. Lembre-se que foram criadas TRIGGERS para geração de chaves primárias para essas tabelas e, portanto, o campo código dessas tabelas NÃO deverá ser usado no comando de inclusão. Lembre-se de demonstrar o funcionamento das Triggers, copiando os comandos usados para testes e os resultados apresentados.

**4 -** Criar uma VIEW que exiba o Código da Fita, a descrição da Fita, a descrição do Filme ao qual a fita pertence, a descrição da Categoria à qual pertence o filme e o preço para locação dessa Categoria APENAS das fitas que estejam disponíveis para locação. Pela lógica de negócio da locadora, fitas disponíveis para locação são aquelas cujo campo DISPONIVEL = ‘S’. Escreva o comando SQL DML capaz de demonstrar o funcionamento da VIEW.

**5 -** Criar uma FUNCTION que receba como parâmetro um código de Fita e retorne o PRECO\_LOCACAO dessa fita/filme. **DICA:** Usar a VIEW criada no exercício 4. É obrigatório que a FUNCTION capture a EXCEPTION de NO\_DATA\_FOUND, para os casos de o código passado como parâmetro não existir ou a fita não estiver disponível. Utilizar apenas o seguinte comando no tratamento dessa EXCEPTION ***Raise\_application\_error(-20000, 'Filta não encontrada ou indisponível')***.

**6 -** Criar uma TRIGGER que ANTES de INSERIR uma LOCACAO, busque automaticamente o VALOR\_LOCACAO de acordo com a categoria da fita/filme que está sendo alugada. **DICA:** Usar a FUNCTION criada no exercício 5. A TRIGGER também deverá verificar se a locação está sendo feita em final de semana (Sábado ou Domingo). Em caso afirmativo, o valor da locação deverá ser acrescido em 10%. Para os outros dias da semana o valor deverá permanecer inalterado. **Dica:** Os dias da semana referentes a Sábado e Domingo são 7 e 1, respectivamente. O ORACLE possui a função TO\_CHAR(DATA, ‘d’) que retorna o dia da semana para a data passada como parâmetro. Lembre-se de demonstrar o funcionamento da Trigger, copiando o comando usando para teste e o resultado apresentado.

**7 -** Criar uma TRIGGER que, após a inclusão de uma locação, faça com que a fita que está sendo locada passe para indisponível. Pela lógica de negócio da Locadora, fita indisponível possui campo DISPONIVEL = ‘N’.

**8 -**  Criar uma VIEW que contenha o código e o nome do Cliente e código e nome dos Filmes que este já assistiu. Usando a VIEW criada, crie uma consulta que retorne o nome do cliente e a quantidade de vezes que ele assistiu a determinado Filme.

**9 -** Dadas as tabelas com os registros especificados e a Trigger abaixo:

****

CREATE TRIGGER Resgate\_Poupanca

BEFORE UPDATE OF saldo ON Conta\_Corrente

FOR EACH ROW

BEGIN

IF :NEW.saldo < 0 THEN

UPDATE Poupanca SET saldo = saldo + :NEW.saldo

WHERE numero = :NEW.numero;

:NEW.saldo := 0;

END IF;

END;

O que acontece quando cada uma das expressões SQL abaixo é executada? Descreva em detalhes todas as ações que ocorrem em cada um dos casos e o valor dos saldos em cada uma das tabelas.

**a.** UPDATE Conta SET saldo = saldo - 3000 WHERE numero = 300;

**b.** UPDATE Conta SET saldo = saldo - 500 WHERE numero = 200;

**c.** UPDATE Conta SET saldo = saldo - 800 WHERE numero = 100;