Exercício Livraria 3

1. Iniciar o Oracle SQL Developer.
2. **Desativar** a opção **Autocommit** (Menu Tools > Preferences > Database > Advanced).
3. Criar/usar uma **ligação ao servidor** Oracle do DEI.
4. **Executar** os ***scripts*** disponibilizados para criar uma base de dados (BD) sobre uma livraria. A BD é implementada de acordo com o modelo relacional da Figura 1.

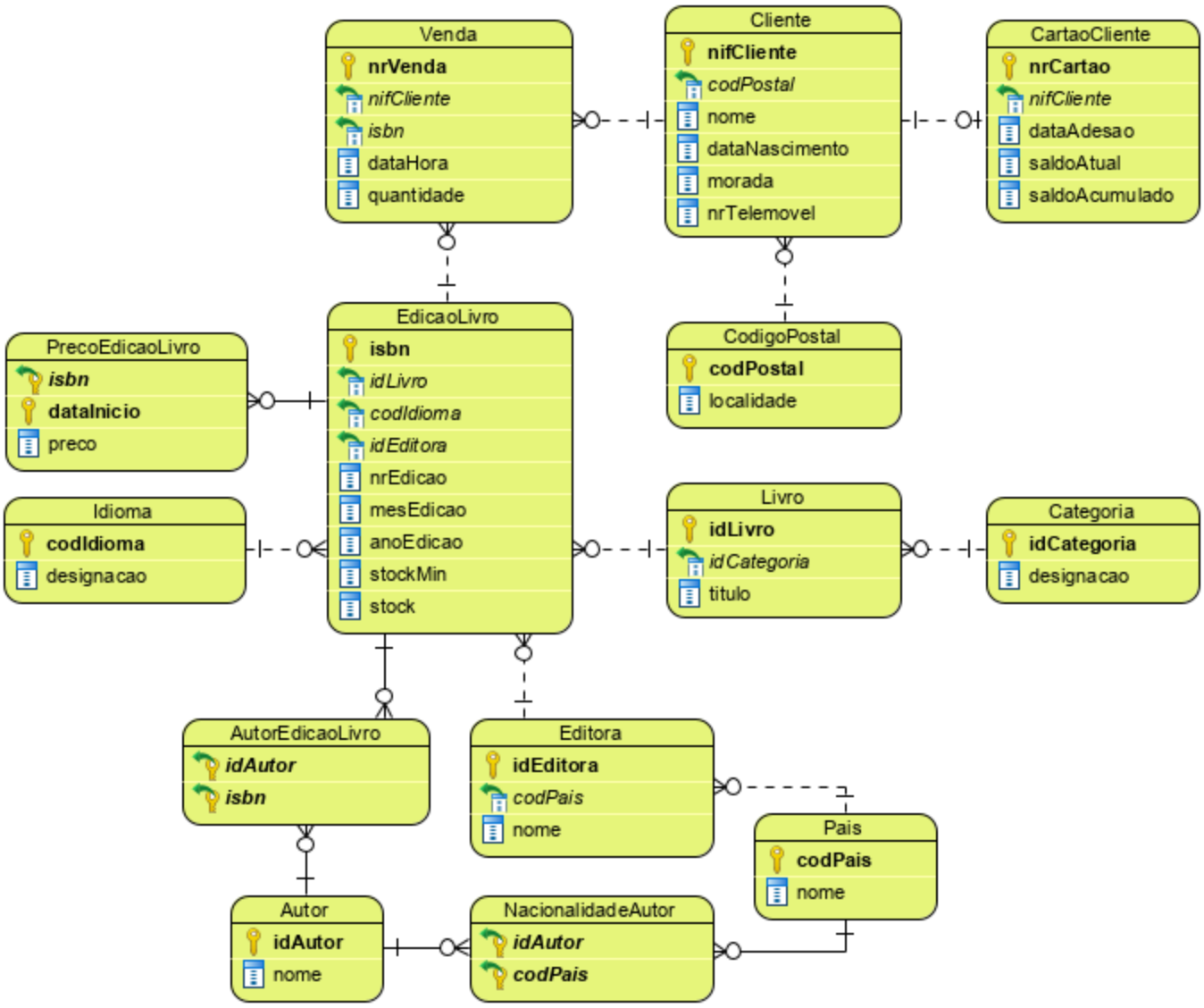

Figura 1 - Modelo Relacional

5. Notas prévias sobre ***triggers* DML**:
   1) Um *trigger* DML é um **programa** que é executado **automaticamente** quando executamos qualquer um dos **comandos DML** (insert, update ou delete) sobre uma **tabela**.
   2) Um *trigger* DML usa-se para:
      a) Implementar **restrições de integridade** que não conseguimos impor com as restrições básicas (PK, FK, UK, NN, CK);
      b) Implementar **regras de negócio** complexas.
   3) Um *trigger* DML pode ser um de um dos seguintes **tipos**:
      a) *Trigger* de **instrução** ("statement level trigger");
      b) *Trigger* de **linha** ("row level trigger").
   4) Um ***trigger* de instrução** é executado apenas **uma vez** por cada instrução DML que o iniciou.
   5) Um ***trigger* de linha** é executado **tantas vezes** quantos os registos afetados pela instrução DML que o iniciou.
   6) Num ***trigger* de linha** temos acesso aos registos afetados através das estruturas ":NEW" e ":OLD".
   7) Um *trigger* DML pode ter um ***timing*** "before" ou "after" a instrução DML que o iniciou, mas, o **código** do *trigger* nunca estará separado da instrução que o iniciou, logo:
      a) Um **"before" *trigger*** corre antes da instrução que o iniciou, mas, continua a fazer **parte integrante** dela;
      b) Um **"after" *trigger*** corre após a instrução que o iniciou, mas, continua a fazer **parte integrante** dela.
   8) Um *trigger* DML na altura da **criação** não verifica os registos **já existentes** nas tabelas.

6. **Implementar**, **compilar** e **testar** devidamente os *triggers* especificados nas alíneas seguintes. Os **testes** devem ser realizados através de **comandos SQL** implementados no mesmo *script* do *trigger*.

1. Pretende-se gerir **ordens de compra** para evitar roturas de stock.

   i. Adicione ao esquema uma tabela para registo de ordens de compra `OrdemCompra(`*`isbn`*`, data, pendente, qtd)`.

   ii. Deve ser emitida uma ordem de compra sempre que se venda um livro cujo stock fique igual ou inferior ao seu stock mínimo após a venda e desde que não haja nenhuma ordem de compra pendente para esse mesmo livro/edição. O campo `OrdemCompra.pendente` é igual a 1 se a ordem de compra estiver pendente e igual a 0 caso contrário. As ordens de compra são emitidas no estado "pendente".

2. Criar um trigger sobre a tabela Cliente, designado trgClienteImpedirEliminacaoRegistos, para **impedir a eliminação de registos de clientes durante o fim de semana**. Numa perspetiva de legibilidade do código, todas as mensagens de erro devem ser definidas na secção EXCEPTION.

3. Criar um novo script para implementar um trigger sobre a tabela Venda, designado trgVendaImpedirAlteracoesForaExpediente, para **impedir a adição ou atualização de registos com dataHora fora do horário de funcionamento** da livraria. A livraria funciona de segunda a sexta, das 9H00 às 19H00. A mensagem de erro deve ser: 'Horário de fim de semana' ou 'Fora do horário permitido'. Numerar adequadamente as mensagens de erro (não esquecer que já foi usado um valor no trigger implementado anteriormente). Compilar e testar adequadamente o trigger implementado.

4. Implementar um trigger sobre a tabela PrecoEdicaoLivro, designado trgPrecoEdicaoLivroImpedirRegisto, para **impedir o registo ou a alteração de um preço** de uma edição de um livro, se a data inicial (dataInicio) for anterior ou igual à data atual. Todas as eliminações devem ser impedidas. Testar adequadamente o trigger implementado.

5. Acrescentar um novo atributo à tabela PrecoEdicaoLivro que permita **registar o instante da criação/alteração de preços**. Esse novo atributo é representado por uma nova coluna (a criar) chamada `dataHora`. Alterar o trigger desenvolvido na alínea anterior 5) para que passe a registar automaticamente o instante da criação/alteração de preços. Se um comando DML que inicia o trigger explicitar um valor para o instante da criação/alteração de preços, este deve ser ignorado pelo trigger, associando-lhe sempre a data do sistema. Testar adequadamente o trigger implementado.

6. Criar um novo script para implementar um trigger, designado trgVendaAtualizarSaldosCartaoCliente, para **atualizar o saldo atual e o saldo acumulado do cartão de cliente**, quando é adicionado ou eliminado um registo na tabela Venda. Adicionar/Reduzir 5% do valor da venda a cada um dos saldos do cartão. Compilar e testar adequadamente o trigger implementado.