Centro Federal de Educação Tecnológica de Mato Grosso

UAB-CEFET-MT

CURSO DE TECNÓLOGO EM SISTEMAS PARA INTERNET
MODALIDADE A DISTÂNCIA

DISCIPLINA: FUNDAMENTOS DE BANCO DE DADOS

Professor Autor: Inara Aparecida Ferrer Silva

Unidade 6 - LINGUAGEM SQL- DDL (DATA DEFINITION LANGUAGE)

Introdução

VISÃO GERAL DA UNIDADE

Até agora você aprendeu a construir um projeto conceitual utilizando ferramentas gráficas para representá-lo, logo depois usou regras de transformação para chegar a outra fase do projeto de banco de dados. Dentro do modelo lógico você viu outra forma de corrigir o seu modelo lógico, utilizando a normalização. Neste momento você estudará uma linguagem padrão chamada SQL, muito usada por todos os fabricantes de SGBD do mundo.

O objetivo nesta unidade é auxiliar o aluno a compreender a linguagem SQL e sua função para Banco de Dados. Nesta unidade mostraremos a DDL(Data Definition Language), a primeira parte da linguagem SQL. Nela você perceberá que a linguagem SQL é muito mais do que uma linguagem de consulta, ela é também importante para a criação da estrutura do banco de dados.

Esperamos que você aproveite o máximo deste material e desejamos que continue as atividades com sucesso!

Objetivo Específico

Propomos, nesta unidade, levar você a:

- Entender a Linguagem SQL

- Utilizar os comandos da linguagem DDL- Linguagem de Definição de Dados para criar a estrutura do banco de dados.

UNIDADE VI - LINGUAGEM SQL- DDL (DATA DEFINITION LANGUAGE)

6. Linguagem SQL
6.1 Comandos DDL
6.1.1 Comando Create Table
6.1.2 Comando Alter Table
6.1.3 Comando Drop Table
6.1.4 Comando Create Índex
6.1.5 Comando Drop Index

6. LINGUAGEM SQL

A linguagem SQL (Structured Query Language ou Linguagem de Consulta Estruturada) representa um conjunto de comandos responsáveis pela definição das tabelas, comandos e atualização dos dados em um S.G.B.D. É a linguagem padrão de definição e manipulação de banco de dados.

Os fundamentos da linguagem SQL estão no modelo relacional de Codd (1970).

Segundo MACHADO (2004) seguem os fatos históricos do surgimento da linguagem SQL:

1. A primeira versão desta linguagem recebeu o nome de SEQUEL, entre 1976 e 1977, depois de revisada teve seu nome alterado para SQL por razões jurídicas.
2. Após a revisão foi colocado em prática um projeto ambicioso da IBM chamado System R e novas alterações foram introduzidas na SQL sugeridas pelos usuários do ambiente.
3. Com o sucesso desta nova forma de fazer consulta ao banco de dados, a utilização da SQL foi aumentando cada vez mais, com isso uma grande quantidade de SGBD's foi utilizando como

linguagem básica a SQL, DB2 da IBM , Oracle da Oracle Corporation, Sybase da Sybase Inc., Microsoft SQL Server da Microsoft, etc.

4. Assim a SQL tornou-se um padrão de fato no ambiente de banco de dados relacionais, faltava tornar-se de direito. Em 1982, o American National Standard Institute (ANSI) tornou a SQL o padrão oficial de linguagem em ambiente relacional.

Assim existem vários padrões para esta linguagem, nós utilizaremos o padrão ANSI da SQL. O que deve ficar claro para todos é que apesar de seguir um padrão pré-estabelecido temos pequenas diferenças de um fabricante para outro, nada que não consigamos compreender. Assim utilizaremos os comandos sem especificidades de cada fabricante.

Os comandos existentes nesta linguagem são subdivididos em dois grupos:

- **Comandos DDL (Data Definition Language)** - Conjunto de comandos responsáveis pela criação, alteração e deleção da estrutura das tabelas e índices de um sistema.
- **Comandos DML (Data Manipulation Language)** - Conjunto de comandos responsáveis pela consulta e atualização dos dados armazenados em um banco de dados. Esta parte da SQL nós veremos com detalhes no módulo 7.

6.1 - Comandos DDL

Relembrando o conceito de modelo relacional, temos que, ele é basicamente formado por tabelas, onde as linhas representam os registros (tuplas) e as colunas representam atributos (campos). Como já vimos cada registro deve ser único, eles são diferenciados por uma chave primária que só permite uma ocorrência de valor neste campo.

Na DDL (Linguagem de Definição de Dados) temos os comandos que nos permitem criar as tabelas do modelo relacional, comandos que identificam chaves, comando que apagam ou alterar estruturas das tabelas. Vamos conhecê-los!

6.1.1 - CREATE TABLE

Define a estrutura de uma tabela (arquivo), seu nome, suas colunas (campos) e as chaves primárias e estrangeiras existentes.

Sintaxe do comando **Create Table**:

CREATE TABLE <nome-tabela>(<nome-coluna> <tipo-dado> [NOT NULL],

PRIMARY KEY (<nome-coluna-chave>),

FOREIGN KEY (<nome-coluna-chave-estrangeira>) **REFERENCES** <nome-tabela-origem> **ON DELETE** [RESTRICT] [CASCADE] [SET NULL])

onde:

- **< nome-tabela>** - Representa o nome da tabela que será criada.
- **<nome-coluna>** - Representa o nome dos campos que serão criados. Coloca-se um campo após o outro.
- **<tipo-do-dado>** - Cláusula que define o tipo e tamanho dos campos definidos para a tabela.
- **NOT NULL** – No ato do preenchimento é obrigatório que possua um conteúdo.
- **NOT NULL WITH DEFAULT** - Preenche o campo com valores pré-definidos, de acordo com o tipo do campo, caso não seja especificado o seu conteúdo no momento da inclusão do registro. Os valores pré-definidos são:
    - e.1) Campos numéricos - Valor zero.
    - e.2) Campos alfanuméricos - Caracter branco.
    - e.3) Campo formato Date - Data corrente.
    - e.4) Campo formato Time - Horário no momento da operação.
- **PRIMARY KEY** (nome-chave) - Definir para o banco de dados a coluna que será a chave primária da tabela. Caso ela tenha mais de um coluna como chave, elas deverão ser relacionadas entre os parênteses.
- **FOREIGN KEY** (nome-chave-estrangeira) **REFERENCES** (nome-tabela-origem) -Definir para o banco de dados as colunas que são chaves estrangeiras, ou seja, os campos que são chaves primárias de outras tabelas.

Na opção REFERENCES escreve-se o nome da tabela de onde veio a chave estrangeira, ou seja a tabela de origem onde o campo era a chave primária.

➔ **ON DELETE** - Esta opção especifica os procedimentos que devem ser feitos pelo SGBD quando houver uma exclusão de um registro na tabela

de origem e existe um registro correspondente nas tabelas filhas. As opções disponíveis são:

- RESTRICT - Opção default (padrão). Esta opção não permite a exclusão na tabela de pai (tabela de origem) de um registro cuja chave primária exista em alguma tabela filha.
- CASCADE - Esta opção realiza a exclusão em todas as tabelas filhas que possua o valor da chave que será excluída na tabela pai.
- SET NULL - Esta opção atribui o valor NULO nas colunas das tabelas filha que contenha o valor da chave que será excluída na tabela pai.

Tipos de dados mais comuns:

Para dados do tipo:	Tipo definido	Tamanho
Caracteres	Char(n), varchar (n)	Armazena até n bytes
Numérico exato	Decimal (p,e) ou numeric (p, e)	Depende
Número aproximado	Float, real	8 bytes e 4 bytes
Número inteiro	Int, smallint, tinyint	4 bytes, 2 bytes e 1 byte
Data e hota	Date, Time, smalldatetime	8 bytes, 4 bytes
Texto e imagens	Text, image, ntext	Variável
Monetário	Money, smallmoney	8 bytes, 4 bytes

1) Numéricos (inteiros e exatos):

- Smallint - Armazena valores numéricos, em 2(dois) bytes, compreendidos entre o intervalo -32768 a +32767.

- Int- (Integer) - Armazena valores numéricos, em quatro bytes binários, compreendidos entre o intervalo -2.147.483.648 a +2.147.483.647.

- Tinyint- usa 8 bits (1 byte) números não negativos de 0 a 255.

- Decimal (p,e) - Armazena valores numéricos com no máximo 15 dígitos. Nesta opção deve ser definida a quantidade de dígitos inteiros (p) e casas decimais (e) existentes no campo. No exemplo decimal (7,2) permite que se armazene 5 dígitos antes da vírgula e dois dígitos depois, num total de 7 dígitos.

2) Alfanuméricos ou caracteres:

Os tipos Varchar (n) e Char (n) definem um campo alfanumérico de n caracteres, onde n deve ser menor ou igual a 254 caracteres.respectivamente. Ex: o tipo char (20) reserva na memória 20 espaços fixos. O tipo varchar (20) permite uma variação no tamanho, até o limite de 20 caracteres.

3) Campo data

O tipo Date define um campo que irá armazenar datas.

4) Campo Hora

O tipo Time define um campo que irá armazenamento de horário.

5) Números aproximados

O tipo real e float armazenam números inexatos, mas não tem precisão suficiente para vários dígitos.

6) Texto e imagem

Os tipos text, ntext e image armazenam respectivamente até 1 trilhão de caracteres, 2 trilhões de caracteres e imagem de tamanho variado.

Agora, vamos aplicar os comandos que aprendemos nos itens anteriores. Para isto escolhe-se um modelo conceitual ou as tabelas de um modelo lógico para escrever os respectivos comandos em SQL.

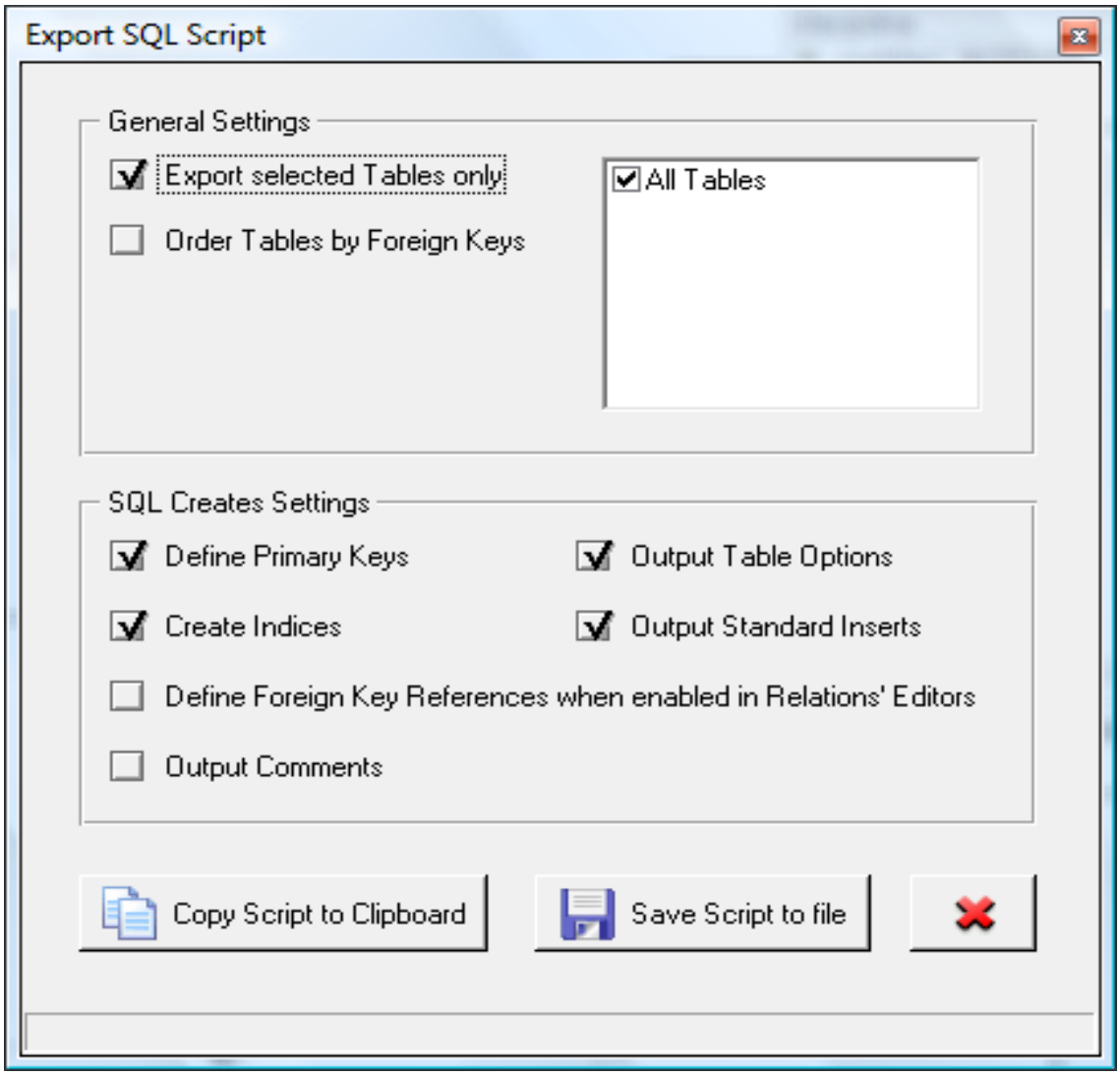
Neste exercício vamos utilizar alguns modelos conceituais desenhados na ferramenta DBdesigner 4 , realizados como atividade do capítulo 4, para reescrever as tabelas do modelo lógico em SQL.

O primeiro passo é verificar quantas entidades estão presentes no modelo Entidade-Relacionamento, se forem duas temos duas tabelas , se forem 3 temos três tabelas a serem escritas em SQL. Em seguida devem-se selecionar as entidades clicando na primeira, pressionando CTRL (no teclado) e com o mesmo pressionado selecionar as outras.

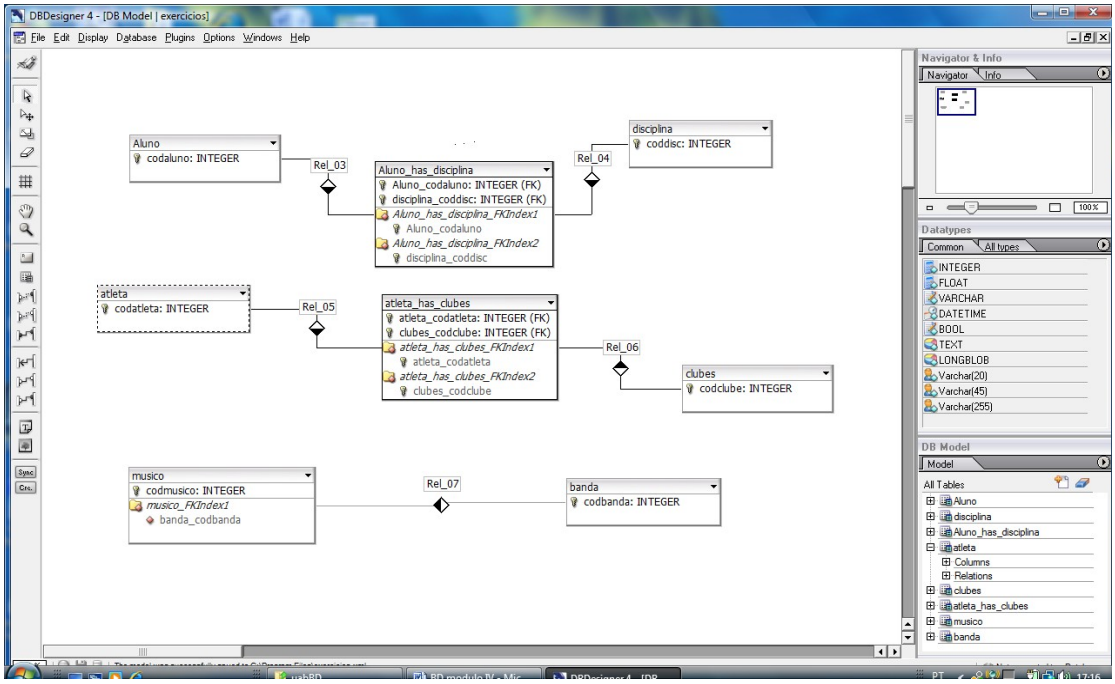
Aqui temos duas opções, podemos utilizar os códigos aprendidos em SQL para fazer este exemplo (digitando os mesmos), ou utilizar a opção File➔ Export➔ SQL create script. Após clicar neste comando temos a figura a seguir aparecendo na tela. Como sugestão, façam os exemplos a seguir utilizando os comandos e digitando-os, depois utilize a opção da ferramenta citada no parágrafo anterior e confira o resultado.

Mãos a obra, vamos em frente!

A opção **copy script to clipboard** copia os scripts em SQL para a área de transferência, isto faz você ter a opção de colar para um editor ou processador de textos todos os scripts em SQL das tabelas criadas. A outra opção **save script to file** cria um arquivo com extensão .sql .



Resolvendo os exemplos abaixo, olhe os comandos SQL utilizados para criar as referidas tabelas do modelo.



Para o modelo Entidade –Relacionamento Aluno-disciplina (cuja cardinalidade é N:N) da figura anterior temos as tabelas. Relembrando que as 3 tabelas do modelo Entidade-Relacionamento surgiram do relacionamento N:N e que a tabela aluno_disciplina carrega as duas chaves primárias das tabelas que estão relacionadas.

```
CREATE TABLE Aluno (
  codaluno INTEGER NOT NULL AUTO_INCREMENT,
  nomealuno VARCHAR(20) NULL,
  endereco VARCHAR(20) NULL,  PRIMARY KEY(codaluno));

CREATE TABLE Aluno_disciplina (
  codaluno INTEGER NOT NULL,
  coddisc INTEGER NOT NULL,
  PRIMARY KEY(codaluno, coddisc), FOREIGN KEY(codaluno) REFERENCES
aluno, FOREIGN KEY( coddisc) REFERENCES disciplina));

CREATE TABLE disciplina (
  coddisc INTEGER NOT NULL AUTO_INCREMENT,
  nomedisc VARCHAR(20) NULL,  PRIMARY KEY(coddisc));
```

Para o modelo Entidade –Relacionamento Atleta-clubes (cuja cardinalidade é N:N) da figura anterior temos as tabelas:

```
CREATE TABLE atleta (
  codatleta INTEGER NOT NULL AUTO_INCREMENT,
  nomeatleta VARCHAR(20) NULL,  modalidade VARCHAR(20) NULL,
  PRIMARY KEY(codatleta));

CREATE TABLE atleta_has_clubes (
  codatleta INTEGER UNSIGNED NOT NULL,
  codclube INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(codatleta, codclube), FOREIGN KEY(codatleta) REFERENCES
clubes, FOREIGN KEY(codclube) REFERENCES atleta);

CREATE TABLE clubes (
  codclube INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nomeclube VARCHAR(20) NULL,  endereco VARCHAR(20) NULL,
  PRIMARY KEY(codclube));
```

Para o modelo Entidade –Relacionamento banda e músico (cuja cardinalidade é 1:N) da figura anterior temos as tabelas:

```
CREATE TABLE banda (
  codbanda INTEGER NOT NULL AUTO_INCREMENT,
  nomebanda VARCHAR(20) NULL,  PRIMARY KEY(codbanda));


CREATE TABLE musico (
  codmusico INTEGER NOT NULL AUTO_INCREMENT,
  banda_codbanda INTEGER UNSIGNED NOT NULL,
  nomemusico INTEGER UNSIGNED NULL,
  tempo experiencia VARCHAR(20) NULL,  PRIMARY KEY(codmusico),
FOREIGN KEY(codbanda) REFERENCES banda);
```

6.1.2 - ALTER TABLE

Permite alterar a estrutura de uma tabela(arquivo) acrescentando, modificando e alterando nomes dos formatos das colunas de determinada tabela.

Sintaxe:

ALTER TABLE <nome-tabela>
ADD <nome-coluna> <tipo-do-dado>

ALTER TABLE <nome-tabela>
MODIFY <nome-coluna> <tipo-do-dado> [NULL]

ALTER TABLE <nome-tabela>
ADD PRIMARY KEY <nome-coluna>

ALTER TABLE <nome-tabela>
ADD FOREIGN KEY (nome-coluna-chave-estrangeira) **REFERENCES**
(nome-tabela-pai) **ON DELETE** [RESTRICT]
[CASCADE]
[SET NULL]

Onde já vimos que:

1. <nome-tabela> - Representa o nome da tabela que será atualizada.
2. <nome-coluna> - Representa o nome da coluna que será criada.
3. <tipo-do-dado> - Cláusula que define o tipo e tamanho dos campos definidos para a tabela.
4. ADD <nome-coluna> <tipo-do-dado> - Realiza a inclusão da coluna especificada na estrutura da tabela. Na coluna correspondente a este campo nos registros já existentes será preenchido o valor NULL (Nulo).
5. MODIFY <nome-coluna> <tipo-do-dado> - Permite a alteração na característica da coluna especificada.

6. ADD PRIMARY KEY <nome-coluna> - Esta opção é utilizada quando não especificamos a chave primária da tabela e precisamos adicioná-la posteriormente.
7. ADD FOREIGN KEY <nome-coluna> - Esta opção é utilizada quando não especificamos a chave estrangeira da tabela e precisamos adicioná-la posteriormente.

Exemplos do comando Alter Table:

Vamos supor que, na tabela aluno , eu tenha esquecido de adicionar o campo bairro.
Por coincidência também esqueci o campo nomeprofessor (que ministra a disciplina).
Assim temos:

ALTER TABLE aluno
ADD bairro char(20);

ALTER TABLE disciplina
ADD nomeprofessor char(20);

Nos exemplos anteriores os dois comandos acrescentam campos na estrutura da tabela.

Nos próximos exemplos temos que alterar as tabelas citadas para acrescentar uma chave primária e outra chave estrangeira que possivelmente foi esquecida. É uma outra maneira de inserir os dois tipos de chave. A primeira opção é definir no comando **create table** como já vimos.

ALTER TABLE disciplina

ADD PRIMARY KEY coddisc;

ALTER TABLE aluno_disciplina

ADD FOREIGN KEY (codaluno) **REFERENCES** aluno;

6.1.3 - DROP TABLE

Deletar a estrutura da tabela do Banco de Dados. Após a execução deste comando todos dados, estrutura e índices a ela associados estarão eliminados.

Sintaxe:

DROP TABLE <nome-tabela>

onde:

a) nome-tabela - Representa o nome da tabela que será deletada.

Exemplos do comando drop table, utilizando as tabelas dos exemplos do item 6.1.1.

Drop Table aluno;
Drop Table Disciplina;
Drop Table musico;
Drop Table banda;
Drop table aluno_disciplina;
Drop Table atleta;
Drop Table clubes;

6.1.4 - CREATE INDEX

O índice de acesso é um caminho de acesso mais rápido aos dados em uma operação de seleção. Os índices podem ser criados a partir de um ou mais campos de uma tabela. Normalmente as chaves primárias já são índices por definição, ou seja, através delas encontramos qualquer registro na tabela.

Para criar uma estrutura de índice de acesso para uma determinada coluna, em uma tabela, use a sintaxe:

CREATE INDEX <nome-índice>

ON <nome-tabela> (<nome-coluna>,<nome-coluna>);

onde:

a) nome-índice - Representa o nome do índice que será criado.

b) nome-tabela - Representa o nome da tabela que contem a coluna na qual será criado o índice de acesso.

c) nome-coluna - Representa o nome da coluna onde o índice será criado.

Exemplo:

Create Index buscadisciplina

On disciplina (nomedisciplina);

6.1.5 - DROP INDEX

Deletar uma estrutura de índice de acesso para uma determinada coluna em uma tabela.

Sintaxe:

DROP INDEX <nome-índice>

onde:

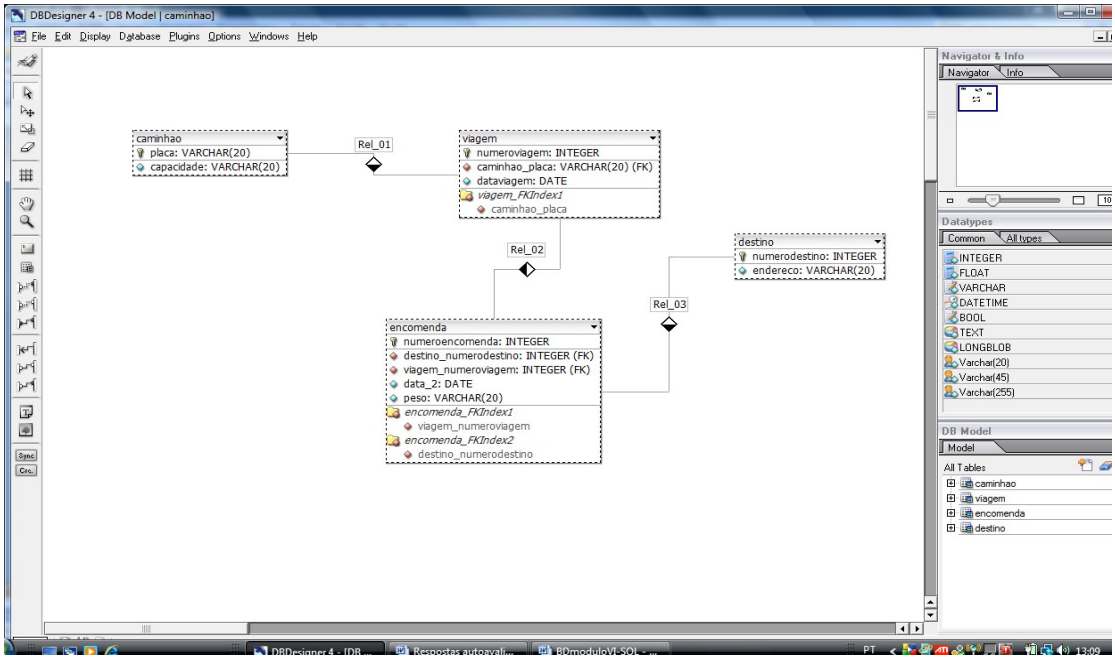
a) nome-índice - Representa o nome da estrutura de índice que será deletada.

Exemplo:

Drop Index buscadisciplina;

Exercícios de auto-avaliação:

1. **Escreva os códigos da linguagem SQL para os exercícios do caminhão-viagem.**



2. **Escreva os códigos SQL para as tabelas abaixo:**

Tabela Cliente
Campos: codcli, nome, rua, cidade

Tabela Agencia
Campos: codagen (Chave primaria),nomeagen, cidadeagen,ativo

Tabela conta
Numeroconta (chave primaria), codagen (chave estrangeira), saldo

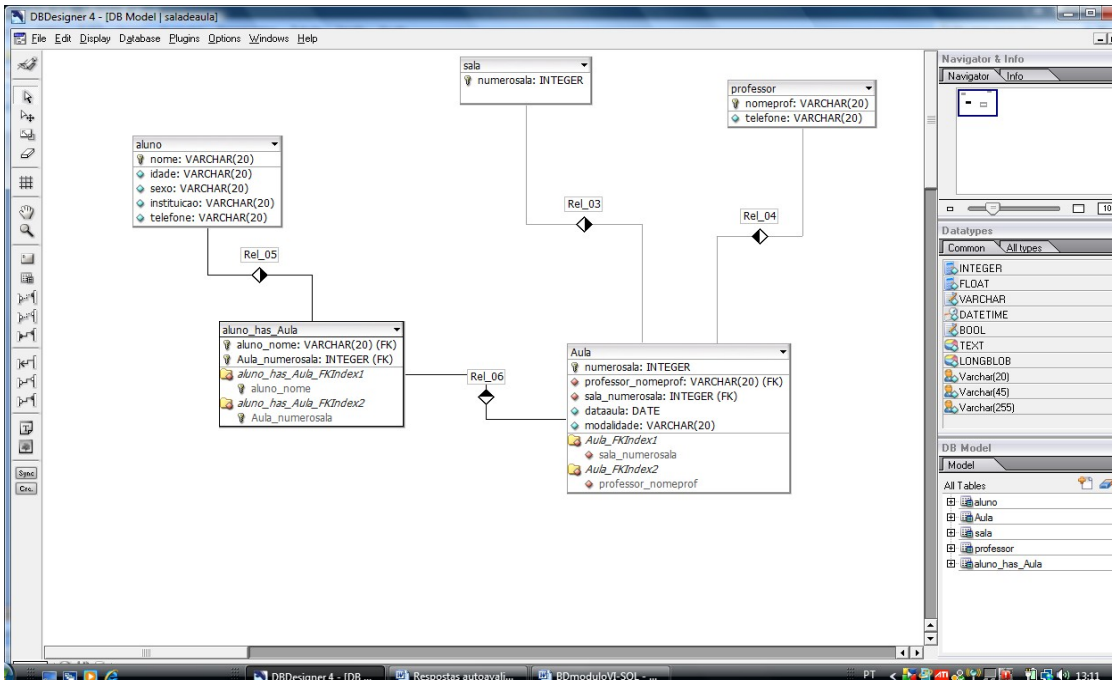
Tabela depositante
Codcli (chaveprimaria), numero conta (chave primaria), tabela que surgiu do relacionamento N:N entre cliente e conta

3. **Acrescente na tabela cliente o campo telefone e bairro.**

4. **Altere o tamanho do campo nomeagen de char(20) para char(30).**

5. **Altere o campo saldo de decimal(7,2) para decimal (9,2).**

6. **Escreva os códigos da linguagem SQL para os exercícios da aula-sala de aula.**



BIBLIOGRAFIA

COUGO, Paulo S. **Modelagem Conceitual e Projeto de Banco de Dados.** Rio de Janeiro: Campus, 1997.

MACHADO, Nery R.;ABREU, Maurício P. **Projeto de Banco de Dados: uma visão prática.** São Paulo: Editora Érica, 1996.

SILBERCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados- Tradução Daniel Vieira. Rio de Janeiro. Editora Elsevier, 2006.**

MACHADO, Felipe N. R. **Banco de Dados: Projeto e Implementação.** São Paulo, Érica : 2004.

TOREY, Toby J.; LIGHTSTONE, S.; NADEAU, T. **Projeto e Modelagem de Banco de Dados – Tradução de Daniel Vieira**. Rio de Janeiro: Elsevier, 2007.