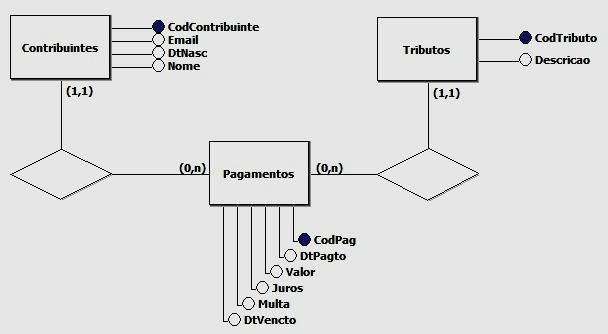
**1. Analise o seguinte modelo conceitual, crie o modelo físico correspondente (Banco de Dados) e faça:**

****

create database ExercSP01

go

use ExercSP01

CREATE TABLE Contribuinte (

CodContribuinte int CONSTRAINT pk\_contr primary key identity(1,1),

Nome varchar(70),

Email varchar(30),

DtNasc datetime

)

CREATE TABLE Tributo (

CodTributo int CONSTRAINT pk\_trib primary key identity(1,1),

Descricao varchar(50)

)

CREATE TABLE Pagamento (

CodPagto int CONSTRAINT pk\_pag primary key identity(1,1),

CodContribuinte int CONSTRAINT fk\_pag\_contr

FOREIGN KEY REFERENCES Contribuinte (CodContribuinte),

CodTributo int CONSTRAINT fk\_pag\_trib

FOREIGN KEY REFERENCES Tributo (CodTributo),

DtVencto datetime,

DtPagto datetime,

Multa money,

Valor money,

Juros money

)

1. Crie storeds procedures para inserir dados nas tabelas criadas.
2. Crie uma stored procedure para editar um determinado registro de pagamento (qualquer campo poderá ser editado).
3. Crie uma stored procedure para efetuar a liquidação de um pagamento. Ela deverá receber como parâmetros o Valor da Multa e dos Juros, além da data de pagamento.
4. Dado um código de contribuinte, mostre o total dos pagamentos liquidados e o total dos pagamentos em aberto para ele.
5. Receba o código de um tributo e liste a média dos pagamentos liquidados deste tributo.
6. Liste a soma dos pagamentos em aberto por contribuinte.
7. Qual foi o contribuinte que mais pagou tributos com multa?

**2. Usando o banco de dados criado anteriormente (BD do Power Point), crie stored procedures para:**

create database AulaStoredProc

go

use AulaStoredProc

-- CRIAR TABELAS

create table departamento(

idDepto int constraint pk\_depto primary key identity(1,1),

depto varchar(40) NOT NULL

)

create table funcionario(

idFun int constraint pk\_fun primary key identity(1,1),

nome varchar(50) NOT NULL,

cpf varchar(20),

rg varchar(50),

e\_mail varchar(50),

dt\_nasc datetime,

dt\_admissao datetime,

senhaDesc varchar(20),

idDepto int constraint fk\_fun\_depto foreign key references departamento(idDepto)

)

create table salario(

idSalario int constraint pk\_salario primary key identity(1,1),

valor money constraint chkValor check(valor > 0),

dtPagto datetime,

obs varchar(1000),

idFun int constraint fk\_sal\_fun foreign key references funcionario(idFun)

)

--INSERIR DADOS

set dateformat dmy

insert into departamento

values ('LOGISTICA')

insert into funcionario(nome, idDepto)

values ('AUGUSTO', 1)

insert into salario

values

(2050.49, '25/01/2014', 'Sem faltas no período', 1),

(2534.00, '15/02/2014', 'Inclui bônus por equipe - atingimento de meta', 1),

(1998.40, '18/03/2014', NULL, 1)

--###################################

create table MediaSalarial(

idVlMedio int

constraint pk\_mediaSal primary key

identity(1,1),

idFun int

constraint fk\_mediaSal foreign key

references funcionario(idFun),

VlMedio money

)

1. Dado um código de departamento, retornar a quantidade de funcionários deste departamento.
2. Inserir, alterar ou excluir um registro em uma tabela qualquer (usando uma **única** stored procedure).
3. Selecionar os departamentos cujo código seja igual a um parâmetro ou cujos nomes sejam PARECIDOS com um segundo parâmetro.
4. Gravar um log (nova tabela para guardar o cod consultado e a data e hora da pesquisa) toda vez que um departamento for consultado pelo código.

**3. Considerando um banco de dados representado pelo esquema a seguir, crie as stored procedures:**

**Aluno** (cod-aluno, nome, cod-curso)

**Notas** (cod-nota, cod-aluno, cod-disciplina, nota)

**Curso** (cod-curso, curso)

**Disciplina** (cod-disciplina, disciplina)

create database ExercSP03

go

use ExercSP03

create table Disciplina(

codDisciplina int constraint pk\_codDisci primary key identity(1,1),

disciplina varchar (30) not null

)

create table Curso(

codCurso int constraint pk\_codCurso primary key identity(1,1),

curso varchar (30) not null

)

create table Aluno(

codAluno int constraint pk\_codAluno primary key,

nome varchar (30) not null,

codCurso int

constraint fk\_Aluno\_Curso foreign key references Curso(codCurso)

)

create table Nota(

codNota int constraint pk\_codNota primary key identity(1,1),

nota numeric not null,

codAluno int

constraint fk\_Nota\_Aluno foreign key references Aluno(codAluno),

codDisciplina int

constraint fk\_Nota\_Disciplina foreign key references

Disciplina(codDisciplina)

)

1. Considere que o campo código da tabela **Alunos NÃO** está definido como autonumeração, crie uma stored procedure para inserir registros nesta tabela, sem que seja necessário informar o código do aluno como parâmetro.
2. Considere que os campos **códigos** das tabelas **Notas, Curso e Disciplina** estão definidos como autonumeração, crie stored procedures para inserir registros em cada tabela (uma stored procedure de inserção para cada tabela).
3. Usando as stored procedures criadas, insira pelo menos 5 registros em cada tabela.
4. Qual o nome de cada curso e a quantidade de alunos matriculados em cada curso?
5. Dado um código de aluno, retornar o nome do aluno e a média das disciplinas que este aluno está cursando.
6. Dado um código de curso, retornar o nome do curso, os nomes dos alunos matriculados neste curso (em ordem alfabética) e o nome das disciplinas, seguido de suas notas.