O sistema de consultório médico, onde há pacientes, médicos especialistas, convênios, consulta.

O objetivo do sistema é gerenciar: consultas e os CRUD

- Paciente (codigoPaciente, nome, codigoConvenio)

codigoPaciente é chave-primaria

codigoConvenio é chave-estrangeira referencia Convenio

- Medico (codigoMedico, nome)

codigoMedico é chave-primaria

- EspecialidadeMedica (codigoEspecialidade, descricao)

codigoEspecialidade é chave-primária

- MedicoEspecialista (codigoMedico, codigoEspecialidade)

codigoMedico e codigoEspecialidade são chaves-primárias

codigoMedico é chave estrangeira referencia Medico

codigoEspecialidade é chave estrangeira referencia EspecialidadeMedica

- Convenio (codigoConvenio, nome)

- MedicoConvenio (codigoMedico, codigoConvenio)

codigoMedico e codigoConvenio são chaves-primárias

codigoMedico é chave-estrangeira referencia Medico

codigoConvenio é chave-estrangeira referencia Convenio

- Consulta (codigoPaciente, codigoMedico, dataConsulta, codigoConvenio)

codigoPaciente, codigoMedico, dataConsulta são chaves primárias

codigoPaciente chave-estrangeira referencia Paciente

codigoMedico chave-estrangeira referencia Medico

codigoConvenio chave-estrangeira referencia Convenio

Medico 0..n ------- 1..n EspecialidadeMedica -> cardinalidade n x n -> cria-se uma 3º tabela/classe, e as chaves primarias das 2 tabelas vão para esta nova tabela como chaves primárias

Paciente 0..n -------- 0..1 Convenio -> cardinalidade 1 x n -> a chave estrangeira visita a tabela que tem o n

Medico 1..n ------ 0..n Convenio -> cardinalidade n x n -> cria-se uma 3º tabela/classe, e as chaves primarias das 2 tabelas vão para esta nova tabela como chaves primárias

Paciente 0..n ----- 1..n Medico -> cardinalidade n x n -> cria-se uma 3º tabela/classe, e as chaves primarias das 2 tabelas vão para esta nova tabela como chaves primárias

public class Convenio {

int codigoConvenio;

string nome;

//construtores

//getters e setters

}

public class Paciente {

int codigoPaciente;

string nome;

Convenio convenio;

//construtores

//getters e setters

}

public class Medico {

int codigo;

string nome;

//construtores

//getters e setters

}

public class EspecilidadeMedica {

int codigoEspecialidade;

string descricao;

//construtores

//getters e setters

}

public class MedicoEspecialista {

Medico medico;

EspecialidadeMedica especialidade;

//construtores

//getters e setters

}

Modelagem:

- Conceitual: livre de tecnologia (não pensa em banco ou SGBD, nem tipos)

- quando se utiliza?

- problema muito grande e a equipe de projeto tem várias pessoas

- quando não se tem o SGBD

- modelar entidades, atributos e relacionamentos

- Lógica: já se pensa na programação ou SQL

- quando se utiliza?

- começar a converter o modelo conceitual em físico

- por exemplo,

atributo 'nome do cliente' vira nomeCliente

atributo 'curso do aluno' vira cursoAluno

- Físico: dependente do banco e do SGBD, associado então à tecnologia