Fundamentos de Bancos de Dados

Aula 3

Prof. Me. Marco Aurelio M. Antunes

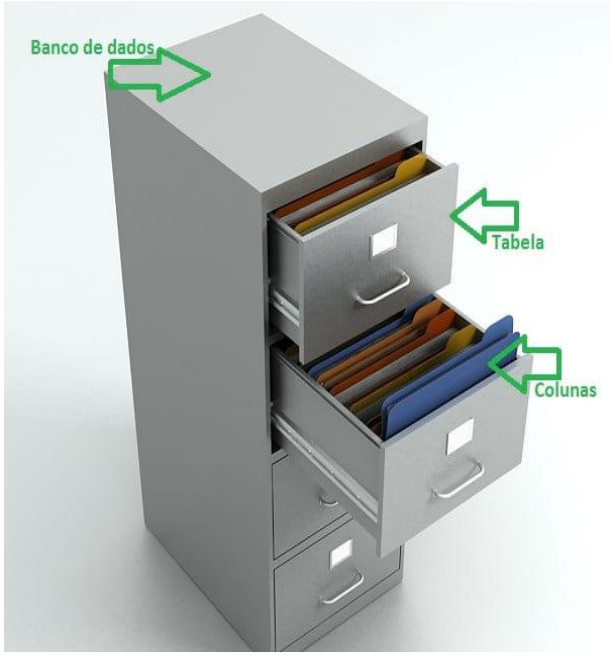
Objetivos da Modelagem de Dados

O principal objetivo da modelagem de dados é desenvolver um modelo que, contendo entidades e relacionamentos, seja capaz de representar os requerimentos das informações do negócio.

Entidade é um agrupamento lógico de informações inter-relacionadas necessárias para a execução das atividades do sistema. Quando transposta ao modelo físico chamamos a entidade de tabela.

Atributos são as informações básicas que qualificam uma entidade e descrevem seus elementos e características. Quando transpostos ao modelo físico chamamos os atributos de campos ou colunas.

Nome da Coluna	Tipo de Dados	Permitir Nul...
id_cliente	int	☐
nome	varchar(60)	☐
endereco	varchar(60)	☐
bairro	varchar(30)	☐
cidade	varchar(30)	☐
estado	varchar(2)	☐
fone	varchar(20)	☑
email	varchar(40)	☑



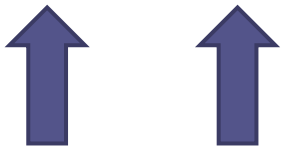
Objetivos da Modelagem de Dados

Registro ou linha é a estrutura de atributos intimamente relacionados e interdependentes que residem em uma entidade após serem transpostos ao modelo físico.

	codfunc	nome	cargo	fone	salario
1	1	Maria	Atendente	99754-8463	1500.00
2	2	João	Atendente	99624-8473	1500.00
3	3	Carlos	Bibliotecário	99614-5452	2500.00
4	4	Kleber	Estagiário	99124-3563	500.00
5	5	Maria	Estagiária	99734-8400	500.00
6	6	joao	atendente	996469113	1100.00
7	7	Guilherme	atendente	996548246	1100.00
8	8	Evandro	Bibliotecario	991411451	1200.00

Objetivos da Modelagem de Dados

	codfunc	nome	cargo	fone	salario
1	3	Carlos	Bibliotecário	99614-5452	2500.00
2	8	Evandro	Bibliotecario	991411451	1200.00
3	7	Guilherme	atendente	996548246	1100.00
4	6	joao	atendente	99646913	1100.00
5	2	João	Atendente	99624-8473	1500.00
6	4	Kleber	Estagiário	99124-3563	500.00
7	5	Maria	Estagiária	99734-8400	500.00
8	1	Maria	Atendente	99754-8463	1500.00



Definição de Dados - REVISÃO

Um banco de dados é composto por tabelas onde armazenamos registros catalogados em função de diferentes campos (características).

Um aspecto prévio a considerar, é a natureza dos valores que introduzimos nesses campos. Visto que um banco de dados trabalha com todo o tipo de informações, é importante especificar que tipo de valor estamos introduzindo, de maneira a, por um lado, facilitar a busca posteriormente, e por outro, otimizar os recursos de memória.

Cada banco de dados introduz tipos de valores de campo que não necessariamente estão presentes em outros. Entretanto, existe um conjunto de tipos que estão representados na totalidade destes bancos. Estes tipos comuns são os seguintes:

Tipos de Dados

Alfanuméricos	Contém cifras, números e letras. Apresentam uma longitude limitada (255 caracteres)
Numéricos	Existem de vários tipos, principalmente, inteiros (sem decimais) e reais (com decimais).
Booleanos	Possuem duas formas: Verdadeiro e falso (Sim ou Não)
Datas	Armazenam datas facilitando posteriormente sua exploração. Armazenar datas desta forma possibilita ordenar os registros por datas ou calcular os dias entre uma data e outra, com sua consistência automática.
Memos	São campos alfanuméricos de longitude ilimitada. Apresentam o inconveniente de não poder ser indexados.
Auto-incrementáveis	São campos numéricos inteiros que incrementam em uma unidade seu valor para cada registro incorporado.

TIPOS DE DADOS

Considere os tipos de dados como regras que restringem o tipo de informação que pode ser inserida em cada coluna de uma tabela de um banco de dados. Se você quiser, por exemplo, garantir que ninguém insira um nome em um campo que só deve conter datas, defina o tipo de dados desse campo como data. Os tipos de dados são definidos para cada coluna no momento da criação de uma tabela. Abaixo segue uma relação dos tipos de dados básicos do SQL Server 2005:

BIGINT: Valores numéricos inteiros variando de –92.23.372.036.854.775.808 até 9.223.372.036.854.775.807

BINARY(N): Armazena N (até 8.000 bytes) dados no formato binário. Se a quantidade de dados binários armazenados no campo for menor que o tamanho total especificado em N, o resto do campo é preenchido com espaços em branco. Procure não utilizar este tipo de dado diretamente, pois existem funções específicas para trabalhar com este tipo de dado.

BIT: Somente pode assumir os valores 0 ou 1. Utilizado para armazenar valores lógicos e não pode conter valores nulos.

CHAR(N): Armazena N caracteres fixos (até 8.000). Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo é preenchido com espaços em branco.

DATETIME: Armazena hora e data variando de 1 de janeiro de 1753 até 31 de Dezembro de 9999. A precisão de hora é armazenada até os centésimos de segundos.

DECIMAL(I,D) e NUMERIC(I,D): Armazenam valores numéricos inteiros com casas decimais utilizando precisão. I deve ser substituído pela quantidade de dígitos total do número e D deve ser substituído pela quantidade de dígitos da parte decimal (após a vírgula).

Lembrando sempre que o SQL Server internamente armazena o separador decimal como ponto (.) e o separador de milhar como vírgula (,). Essas configurações INDEPENDEM de como o Windows está configurado no painel de controle e para DECIMAL E NUMERIC, somente o separador decimal (.) é armazenado

FLOAT: Valores numéricos aproximados com precisão de ponto flutuante, indo de -1.79E + 308 até 1.79E + 308

IMAGE: Armazena dados no formato binário (até 2,147,483,647 bytes). Se a quantidade de dados binários armazenados no campo for menor que o tamanho total especificado em N, o resto do campo não é preenchido. Costuma ser utilizado para armazenar grandes quantidades de dados como imagens ou arquivos de som. Campos definidos com este tipo de dado não podem ser indexados, pesquisados, agrupados ou ordenados.

INT: Valores numéricos inteiros variando de -2.147.483.648 até 2.147.483.647

MONEY: Valores numéricos decimais variando de -922.337.203.685.477,5808 até 922.337.203.685.477,5807. Os dados monetários são definidos com precisão de 4 dígitos.

NCHAR(N): Armazena N caracteres fixos (até 4.000). Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo é preenchido com espaços em branco.

NTEXT: Armazena caracteres (até 1.073.741.823). Se a quantidade de caracteres armazenada no campo for menor que 1.073.741.823, o resto do campo não é preenchido. Procure não utilizar este tipo de dado diretamente, pois existem funções específicas para trabalhar com este tipo de dado.

NVARCHAR(N): Armazena N caracteres (até 4.000). Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo não é preenchido.

NVARCHAR(MAX): serve para armazenar dados maiores que 8.000 caracteres (podem chegar até 2GB por registro) e fornecer as funcionalidades do tipo NVARCHAR como utilizar funções do tipo LEFT, RIGHT, SUBSTRING, etc.

REAL: Valores numéricos aproximados com precisão de ponto flutuante, indo de -3.40E + 38 até 3.40E + 38. Similar ao tipo de dado float, armazena com precisão de até 7 dígitos.

SMALLDATETIME: Armazena hora e data variando de 1 de janeiro de 1900 até 6 de junho de 2079. A precisão de hora é armazenada até os segundos.

SMALLINT: Valores numéricos inteiros variando de –32.768 até 32.767

SMALLMONEY: Valores numéricos decimais variando de -214.748,3648 até 214.748,3647

SQL_VARIANT: Permite o armazenamento de todos os tipos de dados em um mesmo campo de uma tabela com exceção dos tipos TEXT, NTEXT, TIMESTAMP e SQL_VARIANT.

TEXT: Armazena caracteres (até 2.147.483.647). Se a quantidade de caracteres armazenada no campo for menor que 2.147.483.647, o resto do campo não é preenchido. Este tipo de dado armazena grandes quantidades de textos. Colunas definidas com este tipo de dado não podem ser indexadas, pesquisadas, agrupadas ou ordenadas.

TIMESTAMP: Este tipo de dado permite a geração automática de um valor binário para um campo de uma tabela do SQL Server. Cada tabela pode possuir somente um campo com o tipo de dado TIMESTAMP.

TINYINT: Valores numéricos inteiros variando de 0 até 256.

UNIQUEIDENTIFIER: Este tipo de dado deve ser utilizado para a criação de um identificador global para uma coluna de uma tabela. Também podemos possuir somente um campo como tipo UNIQUEIDENTIFIER por tabela. Este identificador deve ser utilizado quando temos certeza absoluta que nenhum valor para o campo deve ser repetido.

VARBINARY(N): Armazena N (até 8.000 bytes) dados no formato binário. Se a quantidade de dados binários armazenados no campo for menor que o tamanho total especificado em N, o resto do campo não é preenchido. Procure não utilizar este tipo de dado diretamente, pois existem funções específicas para trabalhar com este tipo de dado.

VARBINARY(MAX): Dados de caracteres binários de comprimento variável n. Max indica o tamanho máximo de armazenamento, que é 2^31-1 bytes. O tamanho de armazenamento é o comprimento real dos dados inseridos + 2 bytes. Os dados inseridos podem ter 0 bytes de comprimento.

VARCHAR(N): Armazena N caracteres (até 8.000). Se a quantidade de caracteres armazenada no campo for menor que o tamanho total especificado em N, o resto do campo não é preenchido.

VARCHAR(MAX): Use varchar(max) quando os tamanhos das entradas de dados de coluna variarem consideravelmente e o tamanho puder exceder 8.000 bytes.

XML: O tipo de dados xml é um tipo de dados nativo para o armazenamento de documentos ou fragmentos XML, com tamanho máximo de até 2 gigabytes (GB). Você pode usá-lo como qualquer outro tipo de dados nativo do SQL Server: definição de colunas em tabelas, parâmetros para funções e stored procedures e na criação de variáveis.

Não existe regra para a escolha do tipo do dado e sim NECESSIDADE !!!

Chave Primária Composta

A chave primária composta é aquela que é criada em dois ou mais campos e desta forma passa a utilizar a junção dos dados de dois ou mais campos indicados para formar um valor único e assim aplicar o bloqueio de duplicidade.

Exemplo:

```
-- criando tabela de clientes2 com chave composta
create table clientes4
(
codigo_cli    int           not null,
nome_cli      varchar(40)   not null,
fone_cli      varchar(18)   null,
primary key(codigo_cli,nome_cli)
);
```

Exercícios - Lista 3

1) O banco de dados e a tabela do exemplo desta aula
2) Inserir 2 registros na tabela
3) Verificar os registros
4) Dropar a tabela criada
5) Inserir 2 campos na tabela – sendo 1 deles com a constraints NULL e criar a tabela novamente
6) Inserir um registro com todos os campos preenchidos
7) Inserir um registro com campo NULL
8) Testar chave primária forçando o erro
9) Testar chave primária composta forçando o erro – dica necessário dropar e criar a tabela com chave composta
10) Dropar a tabela
11) Modificar o campo codcli para IDENTITY e criar a tabela novamente
12) Inserir um registro com todos os campos preenchidos
13) Inserir um registro com campo NULL
14) Verificar os registros cadastrados
15) Forçar erro chave primaria
16) Criar o banco de dados HOSPITAL
17) Criar as tabelas MEDICO, PACIENTE, CONSULTA com campos a sua escolha – mínimo 6 em cada tabela – todas as tabelas devem ter chave primária e identity- mas uma delas deve ter chave composta
18) Inserir 2 registros em cada tabela
19) Testar chave primária forçando o erro em cada tabela que utiliza chave primária simples
20) Testar chave primária composta forçando o erro em cada tabela que utiliza chave primária composta
21) Verificar os registros de cada tabela

22 - Utilizando o banco de dados FIB e analisando o quadro abaixo, criar uma tabela com os campos necessários para inserir as informações apresentadas – inserir mais 4 registros além do registro proposto no exemplo. Verificar se os registros foram inseridos.

Ficha Cadastral de Atletas

CPF: 123456789-90 – Chave Primária

RG: 98657234-X

Nome: José da Silva Xavier

Idade: 20

Peso: 78

Altura 1,80

Chuteira: 48

Salário: 2738,90

Bônus: 1000,00

Desconto: 32,90

Data de Nascimento: 28/02/2019

Posição: Goleiro

Clube: Palmeiras

Número da Camisa: 1

Postar como lista2.sql - **NÃO haverá prorrogação de postagem de exercícios.**