Criar um banco de dados utilizando a linguagem SQL e realizar operações de manipulação e acesso aos dados.

Programação e Desenvolvimento de Banco de Dados

Aluno: Luis Alexandre Bortoletti
Professor: Gilberto Fernandes Junior

AMPLI - 21/04/2024

1. Introdução

Este relatório apresenta o projeto de programação e desenvolvimento de um banco de dados.

2. Objetivos do Projeto

Fornecer scripts para a criação de tabelas no banco de dados e popular as tabelas.

3. Escopo do Projeto

Banco de dados de contas a receber com as tabelas necessárias para o controle.

4. Planejamento do Projeto

O planejamento do projeto incluiu as seguintes etapas:

- Definição dos requisitos do banco de dados
- Design das entidades e relacionamentos
- Script SQL para a criação do banco de dados

5. Implementação do Projeto

A implementação do projeto consistiu no desenvolvimento do banco de dados, detalhando as entidades, atributos e seus relacionamentos.

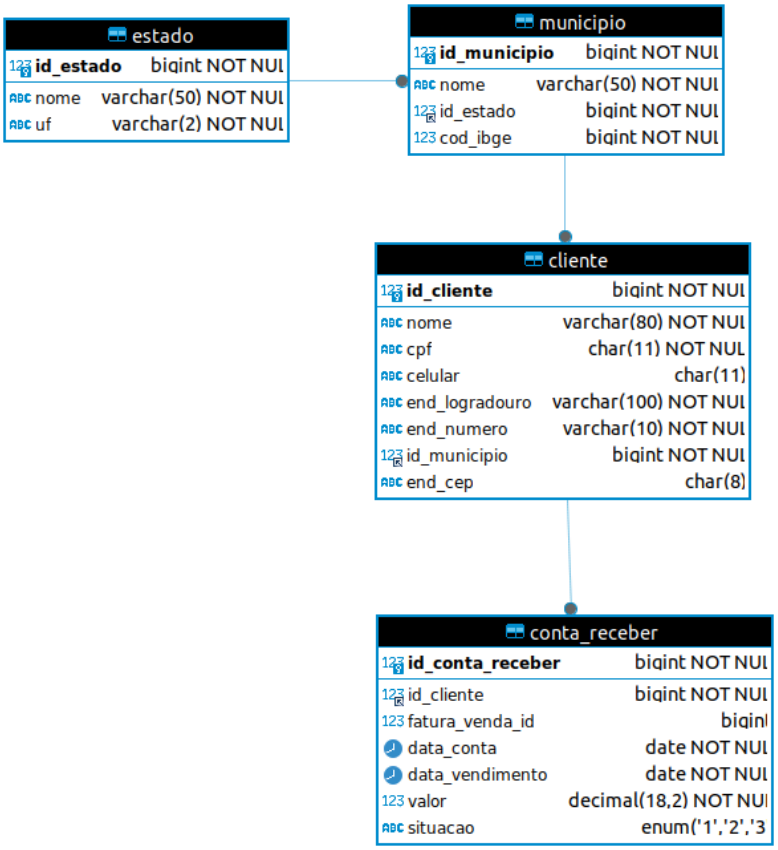
1. Definição dos requisitos do banco de dados

Partimos do contexto apresentado:

1. Estados, o banco de dados deve ter uma tabela para armazenar as informações de nome e sigla da UF que será associada ao município;
2. - Municípios, o banco de dados deve ter uma tabela para armazenar as informações sobre o município que será associado ao cliente;
3. - Clientes, o banco de dados deve ter uma tabela para armazenar informações sobre cliente em campos como: nome, cpf, endereço, incluindo associar o cliente ao município;
4. - Contas a receber, o banco de dados deve ter uma tabela para armazenar as informações sobre contas a receber ou cobrança, onde a coluna de situação controla os processos de cobrança, sendo:1 – Conta registrada, 2 – Conta cancelada, 3 – Conta paga, fazendo referência ainda ao cliente.
5. Visão de recebíveis, o banco de dados deve conter uma consulta específica sobre cobranças em aberto ou com campo situação = 1, oferecendo as colunas para a extração de dados nessa situação.

2. Design das entidades e relacionamentos

O Design das entidades abaixo foi realizado em ferramenta para modelagem de dados específica.



3. Dicionário de dados

O dicionário de dados reflete e formaliza como a situação problema será implementada no banco de dados.

TABELA	COLUNA	TIPO	TAMANHO	DESCRIÇÃO
cliente	celular	char	11	Numero do celular
cliente	cpf	char	11	CPF
cliente	end_cep	char	8	CEP do cliente
cliente	end_logradouro	varchar	100	Endereço do cliente
cliente	end_numero	varchar	10	numero associado ao endereço
cliente	id_cliente	bigint		id sequencial atribuido pelo banco de dados
cliente	id_municipio	bigint		id do municipio associado a tabela municipio
cliente	nome	varchar	80	Nome do cliente
conta_receber	data_conta	date		Data origem da divida
conta_receber	data_vendimento	date		Data de vencimento da dívida
conta_receber	fatura_venda_id	bigint		Id da Fatura que pertence a divida
conta_receber	id_cliente	bigint		id referencia da tabela cliente
conta_receber	id_conta_receber	bigint		id sequencial controlado pelo banco de dados

conta_receber	situacao	enum	1	controle da divida 1 - aberta 2 - cancelada 3 - paga
conta_receber	valor	decimal		valor da divida
estado	id_estado	bigint		id sequencial controlado pelo banco de dados
estado	nome	varchar	50	Nome do estado
estado	uf	varchar	2	Sigla da uf do estado
municipio	cod_ibge	bigint		codigo do igbe
municipio	id_estado	bigint		id referencia para a tabela estado
municipio	id_municipio	bigint		id sequencial controlado pelo banco de dados
municipio	nome	varchar	50	Nome do município
recebiveis	cpf	char	11	CPF do cliente
recebiveis	data_vendimento	date		Data do vencimento da divida
recebiveis	id_cliente	varchar	80	id referencia para tabela cliente
recebiveis	id_conta_receber	bigint		id da divida
recebiveis	valor	decimal		valor da divida

4. Script para criação do banco de dados

Os testes foram realizados com base na tabela abaixo

```
use trabalho;

create table estado(
 id_estado int8 auto_increment not null,
 nome varchar(50) not null,
 uf varchar(2) not null,
 primary key(id_estado)
);

drop table if exists municipio;

create table municipio(
 id_municipio int8 auto_increment not null,
 nome varchar(50) not null,
 id_estado int8 not null,
 cod_ibge int8 not null,
 primary key( id_municipio ) );
 ALTER TABLE municipio
ADD CONSTRAINT fk_estado
 FOREIGN KEY (id_estado)
 REFERENCES estado(id_estado);

create table cliente(
 id_cliente int8 auto_increment not null,
 nome varchar(80) not null,
 cpf char(11) not null,
 celular char(11),
 end_logradouro varchar(100) not null,
 end_numero varchar(10) not null,
 id_municipio int8 not null,
 end_cep char(8),
 primary key( id_cliente ) );
 ALTER TABLE cliente
ADD CONSTRAINT fk_cliente_municipio
 FOREIGN KEY (id_municipio)
 REFERENCES municipio(id_municipio);
```

```sql
create table conta_receber(
 id_conta_receber int8 auto_increment not null,
 id_cliente int8 not null,
 fatura_venda_id int8,
 data_conta date not null,
 data_vendimento date not null,
 valor decimal(18,2) default 0 not null,
 situacao enum('1','2','3'),
 primary key( id_conta_receber ));

ALTER TABLE conta_receber
ADD CONSTRAINT conta_receber_fk_cliente
 FOREIGN KEY (id_cliente)
 REFERENCES cliente(id_cliente);
 show tables;
```

5. Script de carga de dados

Para se ter uma massa de dados para testes, foi realizada a carga de dados a partir de uma tabela de municípios obtida no IBGE e as informações de clientes e contas a receber foram geradas a partir de dados randômicos, utilizando-se de funções do banco de dados.

```
use trabalho;

insert into estado( nome, uf )
select distinct Nome_UF, 'XX'
from importados
where Nome_UF in('Bahia','Minas Gerais', 'Paraiba');

update estado set uf = 'BA' where nome = 'Bahia';

update estado set uf = 'MG' where nome = 'Minas Gerais';

update estado set uf = 'PB' where nome = 'Paraiba';

select * from estado;

delete from municipio;
 INSERT INTO trabalho.municipio
( nome, id_estado, cod_ibge)
select distinct i.Nome_Município , e.id_estado, i.Município
from importados as i
 join estado e on( e.nome = i.Nome_UF );

select m.* , e.nome as estado, e.uf
from municipio as m
 join estado e on( e.id_estado = m.id_estado );


INSERT INTO trabalho.cliente
(nome, cpf, celular, end_logradouro, end_numero, id_municipio,
end_cep)
select
concat( 'CLIENTE-', ( floor( rand() * 10 ) +1 ) ) as nome
, ( floor( rand() * 14 ) + 1) as cpf
```

```
, concat( m.id_estado, ( floor( rand() * 14 ) + 1)  ) as celular
, concat( 'LOGRADOURO ', ( floor( rand() * 6 ) + 1) ) as
end_logradouro
, ( floor( rand() * 4 ) + 1) as end_numero
, m.id_municipio as id_municipio
, ( floor( rand() * 8 ) + 1) as end_cep
from municipio m;

select * from cliente


-- CONTAS A RECEBER
-- INCLUIR PARA CADA CLIENTE 3 PARCELAS.

INSERT INTO trabalho.conta_receber
( id_cliente, fatura_venda_id, data_conta, data_vendimento,
valor, situacao)
select c.id_cliente
, ( floor( rand() * 4 ) + 1) as  fatura_venda_id
, current_date as data_conta
, date_add( current_date, interval (floor(rand()*30)) day ) as
data_vendimento
, round( ( ( rand() * 10 ) + 1), 2)  as  valor
, '1' as situacao
from cliente as c
```

6. Encerramento do projeto

O projeto foi encerrado após a conclusão da criação das tabelas e cargas dos dados.

7. Conclusão

Este projeto permitiu o desenvolvimento de um diagrama entidade relacionamento(DER) a partir das informações de requisitos e escopo obtidos para sua realização.
O documento foi desenvolvido seguindo os padrões da norma ISO 21500:2012 e está em conformidade com os requisitos especificados.