FIPP – Faculdade de Informática de Presidente Prudente
Estruturas de Dados II – Trabalho 1º Bimestre – 2023

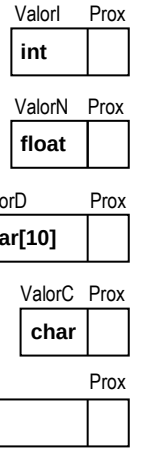
Professor: Francisco Assis da Silva

Banco de dados com Listas Encadeadas Dinâmicas

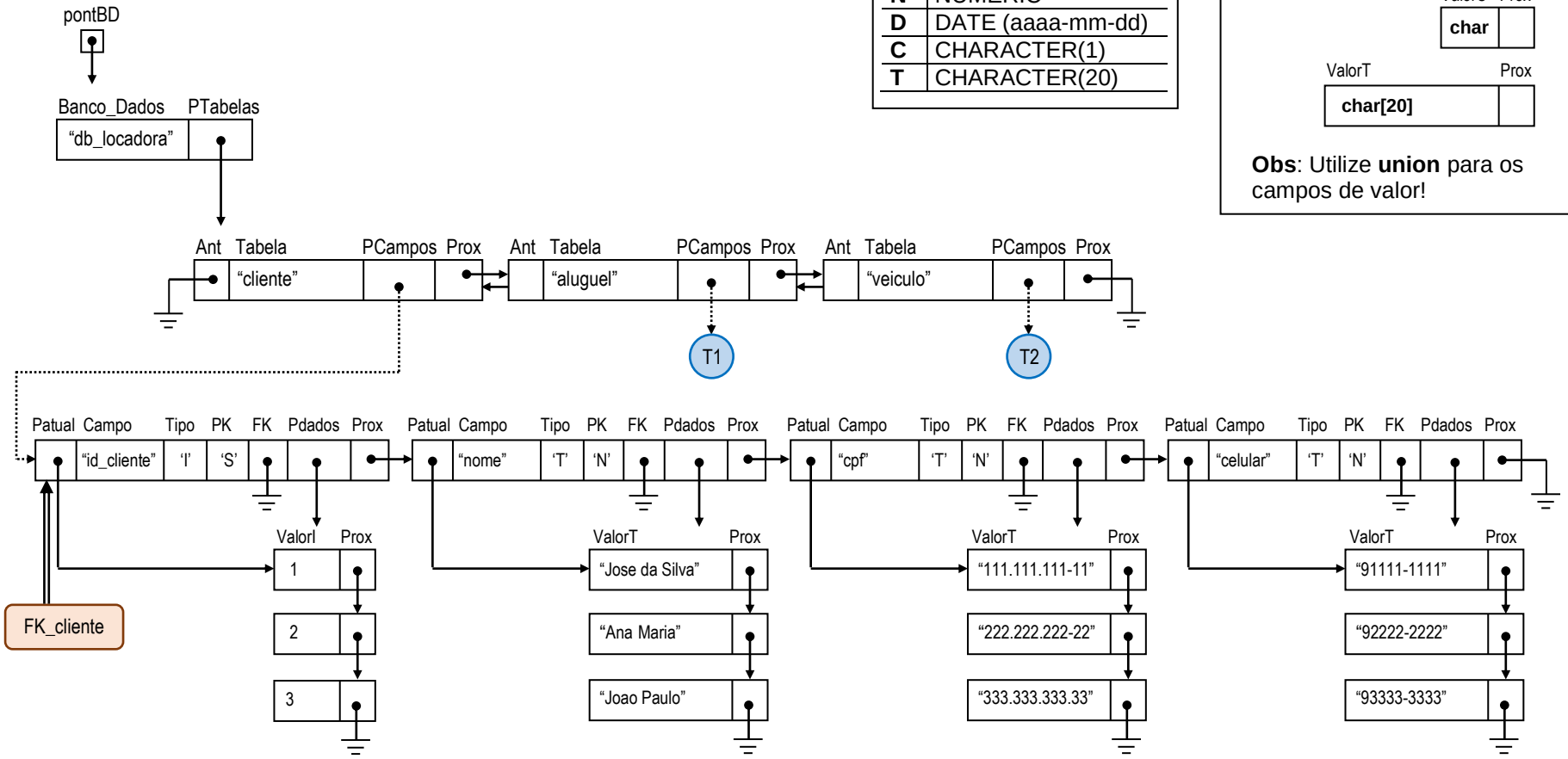
Considere as seguintes estruturas dinâmicas:

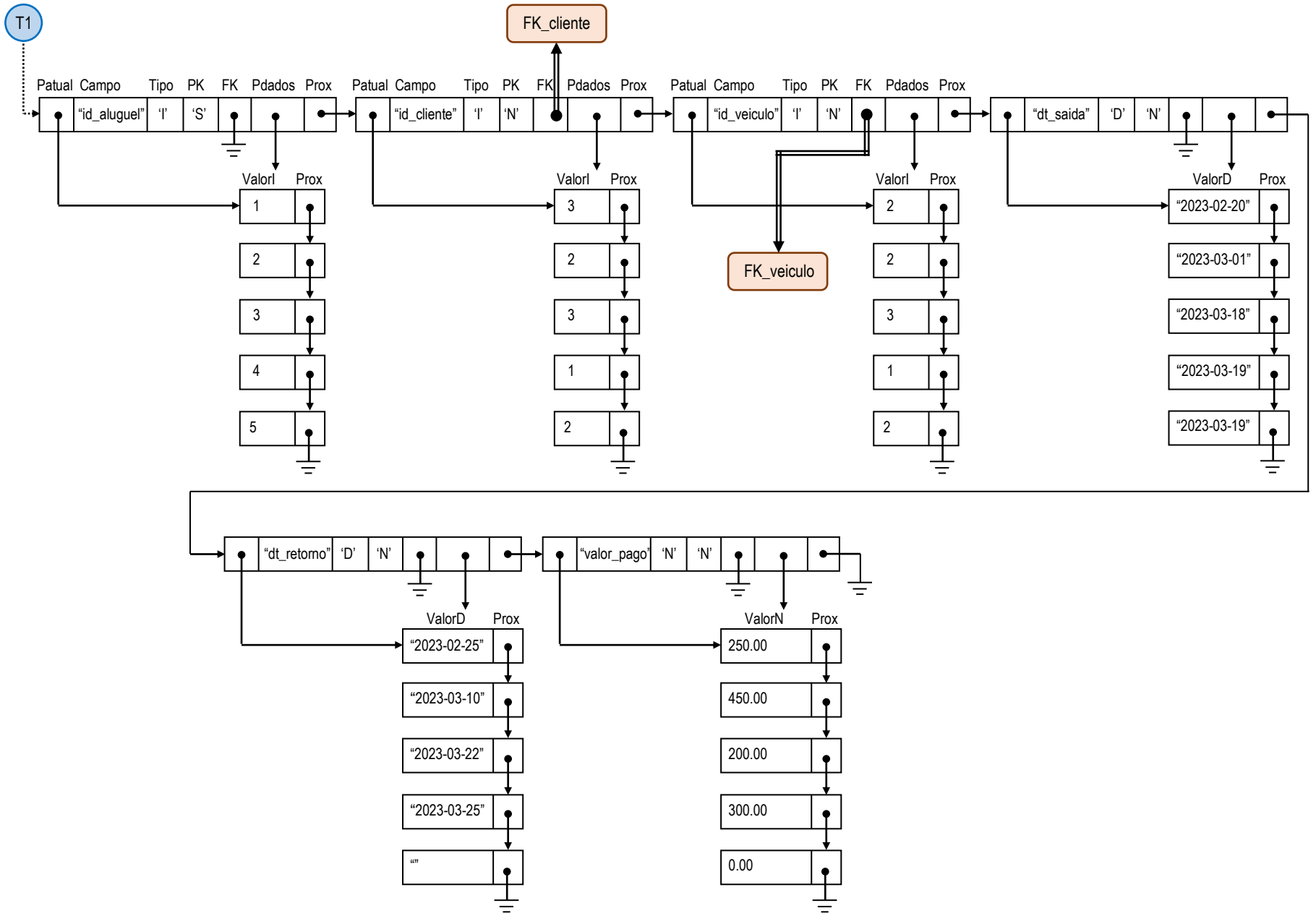
Tipos de Dados	
I	INTEGER
N	NUMERIC
D	DATE (aaaa-mm-dd)
C	CHARACTER(1)
T	CHARACTER(20)

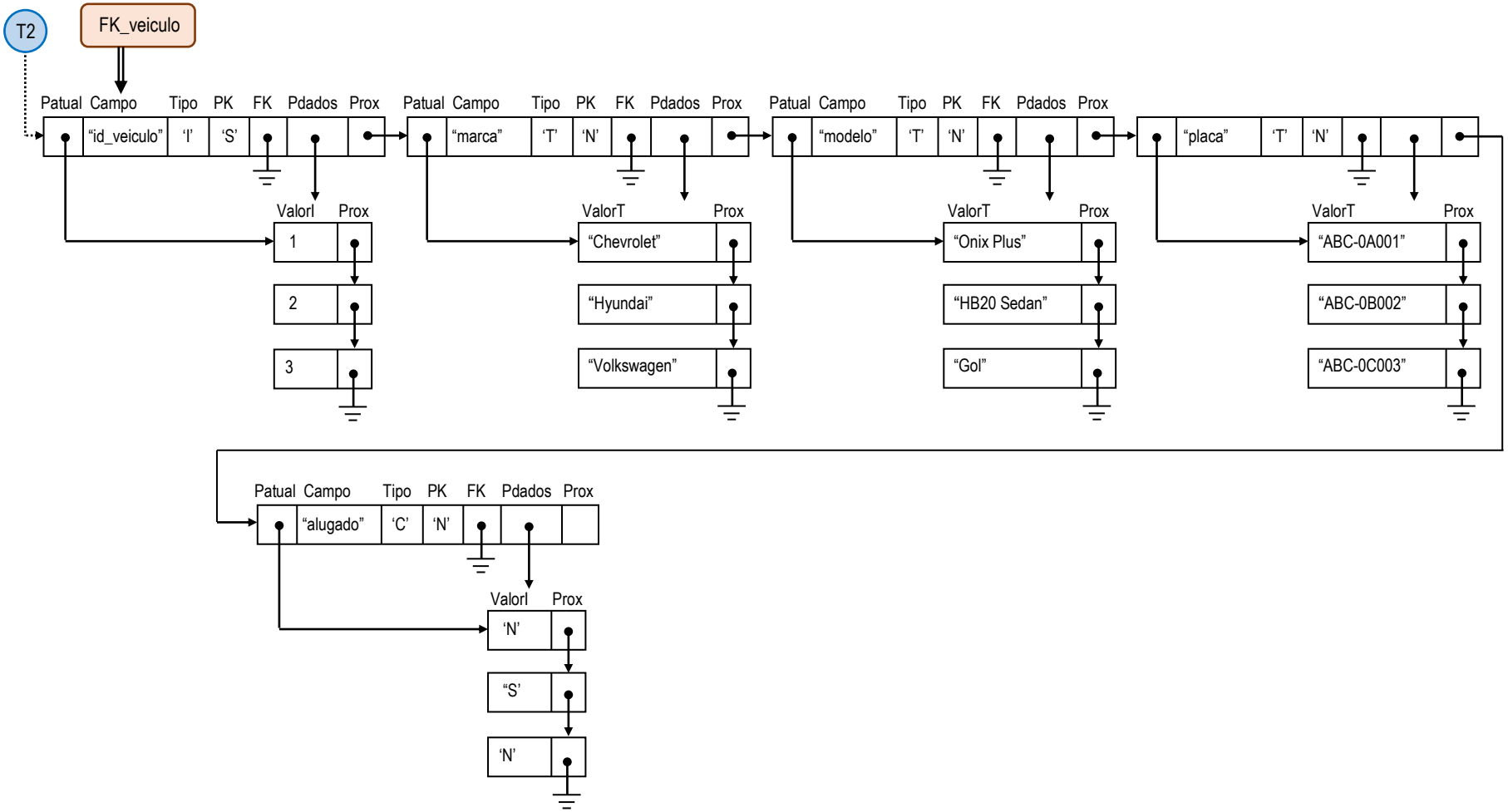
Exemplos de caixas de Dados



Obs: Utilize **union** para os campos de valor!




T1
FK_cliente
Patual Campo Tipo PK FK Pdados Prox
"id_aluguel" 'I' 'S'
"id_cliente" 'I' 'N'
"id_veiculo" 'I' 'N'
"dt_saida" 'D' 'N'
ValorI Prox
1
2
3
4
5
3
2
3
1
2
2
2
3
1
2
ValorD Prox
"2023-02-20"
"2023-03-01"
"2023-03-18"
"2023-03-19"
"2023-03-19"
FK_veiculo
"dt_retorno" 'D' 'N'
"valor_pago" 'N' 'N'
ValorD Prox
"2023-02-25"
"2023-03-10"
"2023-03-22"
"2023-03-25"
""
ValorN Prox
250.00
450.00
200.00
300.00
0.00


T2
FK_veiculo
Patual Campo Tipo PK FK Pdados Prox
"id_veiculo" 'I' 'S'
ValorI Prox
1
2
3
Patual Campo Tipo PK FK Pdados Prox
"marca" 'T' 'N'
ValorT Prox
"Chevrolet"
"Hyundai"
"Volkswagen"
Patual Campo Tipo PK FK Pdados Prox
"modelo" 'T' 'N'
ValorT Prox
"Onix Plus"
"HB20 Sedan"
"Gol"
Patual Campo Tipo PK FK Pdados Prox
"placa" 'T' 'N'
ValorT Prox
"ABC-0A001"
"ABC-0B002"
"ABC-0C003"
Patual Campo Tipo PK FK Pdados Prox
"alugado" 'C' 'N'
ValorI Prox
'N'
"S'
'N'

Na Figura 1 é apresentado o MER (Modelo Entidade Relacionamento) e na Figura 2 o script de criação do "banco de dados" mapeado nas listas encadeadas (páginas 1, 2 e 3):

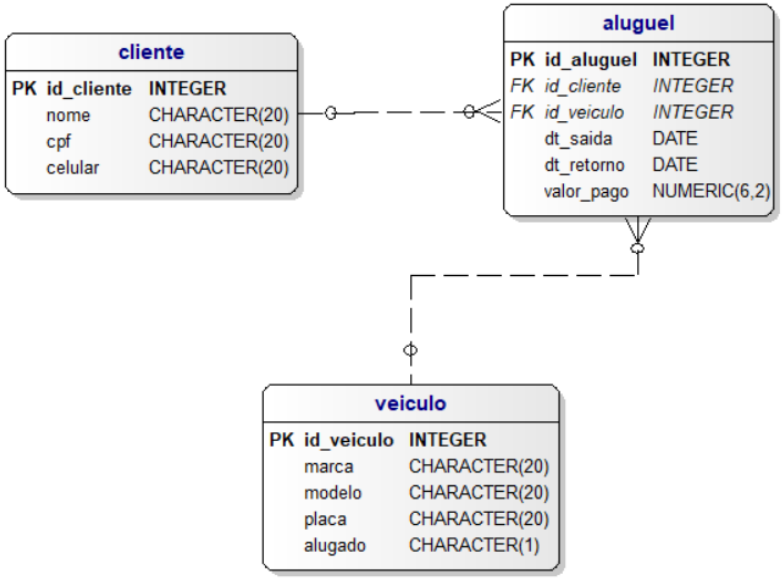


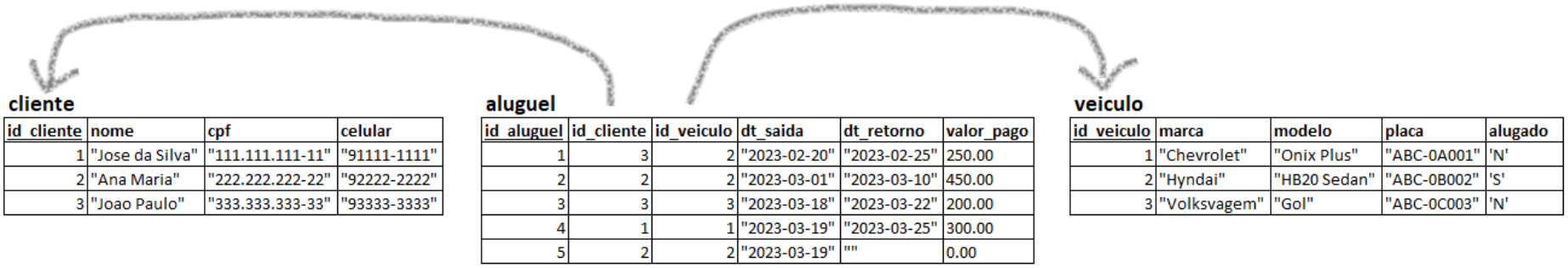
Figura 1. MER do banco de dados de exemplo.

```
CREATE DATABASE db_locadora;

CREATE TABLE cliente (
    id_cliente INTEGER   NOT NULL,
    nome CHARACTER(20) ,
    cpf CHARACTER(20) ,
    celular CHARACTER(20) ,
    CONSTRAINT PK_cliente PRIMARY KEY (id_cliente)
);

CREATE TABLE aluguel (
    id_aluguel INTEGER   NOT NULL,
    id_cliente INTEGER ,
    id_veiculo INTEGER ,
    dt_saida DATE ,
    dt_retorno DATE ,
    valor_pago NUMERIC(6,2) ,
    CONSTRAINT PK_aluguel PRIMARY KEY (id_aluguel)
);

CREATE TABLE veiculo (
    id_veiculo INTEGER   NOT NULL,
    marca CHARACTER(20) ,
    modelo CHARACTER(20) ,
    placa CHARACTER(20) ,
    alugado CHARACTER(1) ,
    CONSTRAINT PK_veiculo PRIMARY KEY (id_veiculo)
);

ALTER TABLE aluguel ADD CONSTRAINT cliente_aluguel
    FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente);

ALTER TABLE aluguel ADD CONSTRAINT veiculo_aluguel
    FOREIGN KEY (id_veiculo) REFERENCES veiculo (id_veiculo);
```

Figura 2. Script de criação do banco de dados.

Na Figura 3 são mostrados os mesmos dados contidos nas estruturas dinâmicas (páginas 1, 2 e 3), mas agora no formato de tabelas em memória.

Figura 3. Tabelas do banco de dados de exemplo com os mesmos dados contidos nas estruturas dinâmicas (páginas 1, 2 e 3).



cliente

id_cliente	nome	cpf	celular
1	"Jose da Silva"	"111.111.111-11"	"91111-1111"
2	"Ana Maria"	"222.222.222-22"	"92222-2222"
3	"Joao Paulo"	"333.333.333-33"	"93333-3333"

aluguel

id_aluguel	id_cliente	id_veiculo	dt_saida	dt_retorno	valor_pago
1	3	2	"2023-02-20"	"2023-02-25"	250.00
2	2	2	"2023-03-01"	"2023-03-10"	450.00
3	3	3	"2023-03-18"	"2023-03-22"	200.00
4	1	1	"2023-03-19"	"2023-03-25"	300.00
5	2	2	"2023-03-19"	""	0.00

veiculo

id_veiculo	marca	modelo	placa	alugado
1	"Chevrolet"	"Onix Plus"	"ABC-0A001"	'N'
2	"Hyndai"	"HB20 Sedan"	"ABC-0B002"	'S'
3	"Volksvagem"	"Gol"	"ABC-0C003"	'N'

Seu trabalho é implementar um simulador de Sistema Gerenciador de Banco de Dados, contemplando os seguintes comandos de DDL (CREATE TABLE e ALTER TABLE – apenas para adicionar a constraint de chave estrangeira), comandos SQL, mais especificamente de DML (INSERT, UDATE, DELETE) e de DQL (SELECT). O banco de dados não será constituído por tabelas em arquivos binários, índices de Árvore B+, Hashing entre outras estruturas de dados mais complexas..., e sim formado por listas encadeadas dinâmicas, conforme o desenho das estruturas dinâmicas mostrado nas páginas 1, 2 e 3.

Observe que o conteúdo **Valor** das **caixas de Dados**, representadas graficamente na Figura 4, pode ter valores de diferentes tipos, mas a caixa é de apenas um único tipo declarado. Nesse caso será necessário utilizar o recurso chamado **union** da linguagem C para resolver isso. Na Figura 4, em a) tem-se os tipos de dados possíveis a serem usados nos campos das tabelas do banco de dados, e em b) tem-se a representação gráfica das caixas de Dados, com seus tipos que devem ser usados na linguagem C. Na Tabela 1 tem-se o mapeamento dos Tipos de Dados do banco de dados com os tipos da linguagem C.

Figura 4. Tipos de Dados possíveis a serem usados no banco de dados e representação gráfica das caixas de Dados.

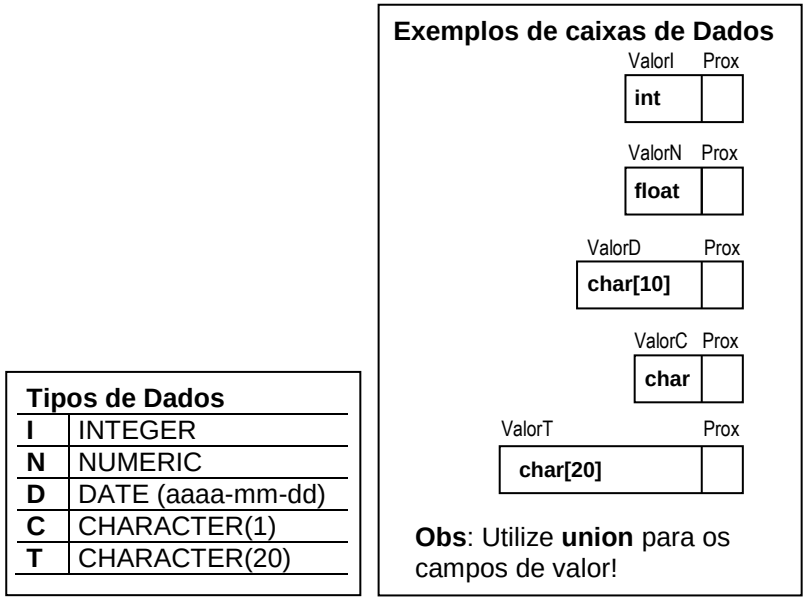


Tabela 1. Mapeamento dos Tipos de Dados possíveis do banco de dados com os tipos da linguagem C.

Tipos Banco de Dados		Linguagem C
I	INTEGER	int
N	NUMERIC	float
D	DATE (aaaa-mm-dd)	char[10]
C	CHARACTER(1)	char
T	CHARACTER(20)	char[20]

Seu programa deve permitir ao usuário realizar algumas tarefas:

a) abrir um arquivo contendo o script de criação do banco de dados (exemplo mostrado na página 4, Figura 2);
b) construir o "banco de dados", a partir do arquivo de script, que na verdade é a criação de todas as listas dinâmicas adequadas para o modelo de dados (tabelas, campos e relacionamentos). Nesse caso são utilizados os comandos de DDL;
c) permitir ao usuário executar os comandos de DML e DQL. No caso do comando SELECT, deve-se mostrar na tela do computador o resultado no formato de uma tabela.
   Exemplo:

```
+-----------+------------+------------+------------+
| id_tabela | coluna_1   | coluna_2   | coluna_3   |
+-----------+------------+------------+------------+
|         1 | dado_col_1 | dado_col_2 | dado_col_3 |
|         2 | dado_col_1 | dado_col_2 | dado_col_3 |
|         3 | dado_col_1 | dado_col_2 | dado_col_3 |
+-----------+------------+------------+------------+
```

Exemplos de comandos SQL que seu programa deve tratar:

```
INSERT INTO tabela (coluna_1, coluna_2, coluna_3, ...)
VALUES (valor_1, 'valor_2', valor_3, ...);

UPDATE tabela
SET coluna_1 = valor_1, coluna_2 = valor_2, ...
WHERE condição;

DELETE FROM tabela
WHERE condição;

SELECT coluna_1, coluna_2, coluna_3, ...
FROM tabela;

SELECT *
FROM tabela;

SELECT coluna1, coluna2, ...
FROM tabela
WHERE coluna_2 BETWEEN valor_inicial AND valor_final;

SELECT tabela_1.coluna_1, tabela_1.coluna_2, tabela_2.coluna_1, tabela_2.coluna_2, ...
FROM tabela_1, tabela_2
WHERE tabela_1.chave_primaria = tabela_2.chave_estrangeira;
```

Informações Importantes:

a) criar um modelo de banco de dados para teste, a seu critério, contendo pelo menos quatro tabelas com relacionamentos. Exemplos:

```
cliente, produto, venda, itens_venda
aluno, livro, emprestimo, itens_emprestimo
```

b) montar comandos SQL de tudo que foi pedido e envolva todas as tabelas, individualmente ou em conjunto;

c) não utilize nada pronto da linguagem C (funções para separar strings (strtok), arrays dinâmincos (vector), etc), todos os algoritmos devem ser implementados;

d) não precisa ter interfaces cheias de detalhes (quadros, posicionamento de cursor, etc), apenas ter uma interface com um terminal para que se possa abrir o arquivo de script (criação do banco de dados) (talvez usar uma tecla especial, como por exemplo, F2, para permitir a digitação do caminho + nome do arquivo de script). A interface também deve permitir ao usuário digitar e rodar os comandos SQL, mostrando os resultados na tela, no caso do comando SELECT;

e) utilize modularização para facilitar a implementação;

f) dica: para resolver a instrução SELECT utilize as estruturas de dados de campos e valores de dados que já estão definidas no trabalho (páginas 1, 2 e 3). Não há a necessidade de implementar novas estruturas para montar uma tabela em memória. Por isso, destaca-se novamente, a importância de fazer a modularização adequada das funções.