

O que é uma Base de Dados?

Conjunto de informação armazenada de forma permanente num sistema informático que se encontra dividida em ficheiros interligados entre si.

Cada ficheiro de uma BD é constituído por um conjunto de **registos** estruturados em **campos**.

Sistemas de Gestão de Base de Dados (SGBD)

Programas ou conjunto integrado de programas, que permitem criar e manipular a BD, em que os dados são estruturados com independência relativamente aos programas de aplicação que os manipulam.

Como é constituído um SGBD?**Um SGBD pode ser visto com uma arquitectura de 3 níveis:**

- **Nível Físico** - Forma como os dados são armazenados em suportes informáticos e como são organizados internamente no sistema informático.
- **Nível Conceptual** – Constitui a estrutura da BD. Forma como os dados são estruturados ao nível da sua concepção lógica, ou seja, número e tipos de campos em que a informação é estruturada, as relações entre os dados, etc.
- **Nível de Visualização** – Modo como os dados são apresentados aos utilizadores finais, através de interfaces gráficos.

Funções de um SGBD

1. **Definição e alteração da estrutura da BD:**
(LDD – Linguagem de Definição de Dados)
 - Criação de uma nova BD
 - Criação de um novo ficheiro ou tabela
 - Alteração da estrutura de campos de uma tabela
 - Eliminação de ficheiros ou tabelas
2. **Manipulação da BD:**
(LMD – Linguagem de Manipulação de Dados)
 - Consulta ou pesquisa de dados
 - Inserção de novos dados
 - Alteração de dados existentes (campos e registos)
 - Eliminação de dados (registos)
3. **Controlo dos dados**
 - Atribuição ou supressão dos direitos de acesso aos dados, em relação aos utilizadores ou grupo de utilizadores
4. **Mecanismos de recuperação de falhas**
 - Cópias de segurança, registo de alterações

Esquema de uma Base de Dados

Consiste no **design ou estrutura lógica** com que a BD é definida. O esquema é concebido segundo um modelo conceptual e implementado através da LDD.

Instância de uma Base de Dados

Consiste nos **dados concretos que a BD contém** em cada momento. É feita através da LMD.

Modelo Relacional

O Modelo Relacional tornou-se no modelo mais utilizado nos sistemas de BD actuais que por isso se designam de Sistemas de Bases de Dados Relacionais.

Uma Base de Dados Relacional é um conjunto de relações ou tabelas.

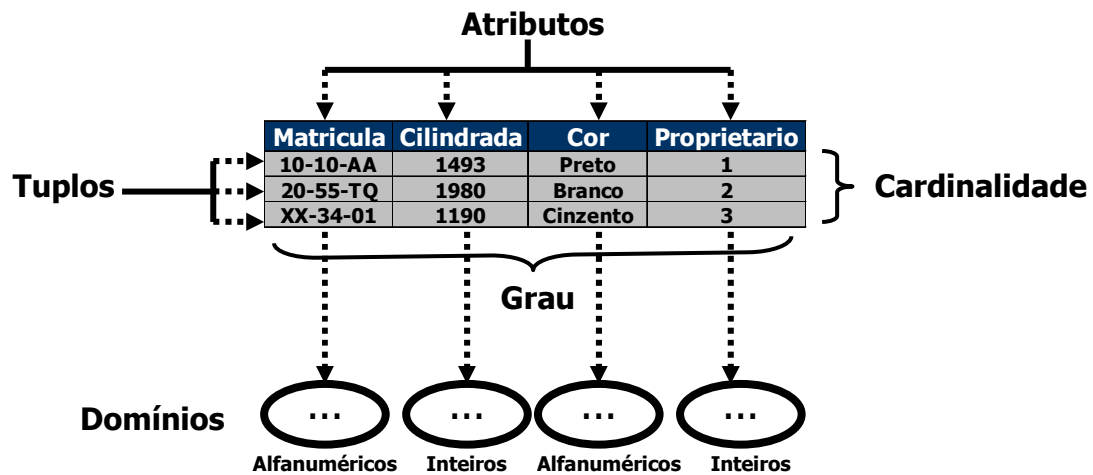
O modelo relacional tem como fundamento teórico a álgebra relacional (modalidade da teoria dos conjuntos).

Relação

É a estrutura fundamental do modelo relacional. **Qualquer subconjunto do produto cartesiano de uma lista de domínios.**

Esquema de Dados Relacional

O conjunto de tabelas de uma base de dados relacional.



Domínio

O domínio de um atributo é o conjunto de valores permitidos para esse atributo.

Ex: o domínio do campo NOME é o conjunto de todas as strings com um máximo de 50 caracteres.

O grau de uma relação é o número de atributos que ela contém. (**colunas**)

A cardinalidade de uma relação é o número de tuplos que ela contém. (**linhas**)

Tuplos/Registos

Linhas de cada relação.

Os tuplos de uma relação ou tabela não têm ordem.

Os valores das componentes de um tuplo são atômicos.

O número total de tuplos possíveis de uma relação é igual ao produto cartesiano dos domínios dos seus atributos.

Ex: $R(A, B)$
 $A \in \{5, 20\}$
 $B \in \{x, y, z, w\}$

Número total de tuplos possíveis de R são 8 ($2 * 4$)

O valor de cada atributo num tuplo é **atômico**, pois numa linha (**tuplo**) com uma coluna (**atributo**) só é possível encontrar um valor.

No modelo relacional não pode haver:

- Atributos compostos
- Atributos multivalor

Tabelas

Todos os dados (assim como o relacionamento existente entre os dados) são representados por tabelas relacionadas entre si:

- Cada tabela tem um **nome único** pela qual é referenciada.
- Cada **linha da tabela (registo)** representa uma entidade única ou um relacionamento entre entidades.

- Cada **coluna (atributo)** tem um nome e refere-se a um dado aspecto do objecto representado na tabela.
- Cada tabela pode conter muitos registos (ou nenhum: tabela vazia).

Tabela: Cliente_banco

Atributo	Nome	Cidade	BI	Data_nascimento	N_de_conta
	António	Porto	123133	12-12-1981	100
	Joaquim	Lisboa	4244442	12-12-1981	200
Registo	Luís	Coimbra	2343423	12-12-1981	300
	Pedro	Aveiro	8787967	12-12-1981	234
	Maria	Beja	4334343	12-12-1981	222

Domínio

Num esquema de dados relacional todas as tabelas devem ter nomes diferentes (isto é, não pode haver duas tabelas com o mesmo nome). O nome de uma tabela deve ser sugestivo. Ou seja, deve indicar para que serve a tabela. Por exemplo, a tabela Cliente_banco serve para guardar dados sobre os clientes de um banco.

Cada registo/tuplo (linha) da tabela Cliente_banco guarda informação sobre um cliente do banco.

Cada atributo (coluna) representa um aspecto dos clientes. Por exemplo, o atributo Nome regista os nomes dos clientes, Cidade regista a cidade onde cada cliente mora, etc. O número de atributos que uma tabela pode ter depende das necessidades.

Uma tabela pode estar vazia (isto é, não tem ainda registos) ou pode ter qualquer número de registos. O importante é perceber que quando se cria uma tabela não há um limite máximo de registos à partida.

Atributos

Cada atributo possui um conjunto de valores. O conjunto de todos os valores para um dado atributo constitui o **domínio do atributo**.

Dependendo do domínio, assim o atributo é representado por números, texto, datas, etc.

É muito importante que todos os atributos de uma tabela sejam não decomponíveis. Diz-se então que os atributos são **atómicos**, ou que são atributos elementares.

Exemplo para o atributo Cidade:

- Domínio é o conjunto formado pelos nomes de todas as cidades de Portugal.
- O atributo é representado por texto (cadeia de caracteres).
- É atómico pois cada atributo representa uma cidade (e não várias).

Os atributos de uma tabela têm um determinado conjunto de valores possível, que se designa por **domínio**.

Um exemplo de um atributo cujo domínio é um conjunto de valores muito pequeno seria o atributo estado civil, que só tem como valores possíveis casado, solteiro, viúvo, divorciado ou separado. Quando se define uma tabela é preciso escolher o tipo de dados adequado para o domínio do atributo. Por exemplo, o atributo Nome terá de ser do tipo texto (cadeia de caracteres). O atributo Data_nascimento tem de ser do tipo data. E assim sucessivamente. Os atributos devem representar dados elementares. Por exemplo, o atributo Cidade permite registar uma cidade em cada linha da tabela (e não várias cidades). Por isso diz-se que **os atributos são atómicos** (porque não se podem decompor).

Quando se cria uma tabela **a ordem pela qual definimos os atributos não é importante** (o que é importante são os atributos e não a ordem pela qual estes aparecem).

A ordem pela qual estão guardados os registos também não é relevante (quando muito pode determinar a velocidade de acesso à tabela).

Ao consultar uma tabela podemos definir a ordem pela qual queremos que sejam mostrados os atributos (por exemplo, primeiro o Nome e depois o BI, ou ao contrário). Também

No exemplo temos os seguintes erros de integridade:

- Há dois registos com o valor do atributo da chave primária (BI) igual. Isto nunca poderá acontecer se BI for definido como sendo a chave primária da tabela.
- Há um registo com o atributo nome não preenchido (diz-se que um atributo está a **Null** quando isso acontece). Como não faz sentido ter um registo de um professor sem ter o nome desse professor este campo deve estar obrigatoriamente preenchido.
- Há duas datas erradas. Se a tabela tem professores que estão presentemente a dar aulas então não podem ter nascido nem em 12-03-2060 nem em 18-01-1878.
- Há vários erros no atributo Salario. O salário não pode ser negativo e o salário da professora Joana Varela é demasiado alto, pelo que não poderia estar correcto.

Restrições de integridade

Restrições de integridade são regras que definem que dados são válidos. Vejamos alguns exemplos para a tabela Professor:

- O salário tem de ser maior ou igual a zero (não pode ser negativo)
- O salário tem de ser menor do que 5 mil dólares
- A data de nascimento tem de estar compreendida entre 01-01-1900 e o dia em que se está a inserir o registo
- O campo Nome não pode ser nulo (preenchimento obrigatório)
- O campo BI é chave primária (não pode haver duplicações)

Estas regras fazem parte da definição da estrutura da tabela (são indicadas normalmente quando se define a tabela).

O Sistema de Gestão de Bases de Dados (SGBD) verifica se os dados respeitam as regras definidas de cada vez que são executadas operações que alteram os dados (inserir, actualizar ou apagar registos).

Integridade de Entidade

Impõe que **os valores dos atributos chave não podem ser nulos nem iguais a outros já existentes na tabela. Não pode haver duas chaves primárias iguais.**

Integridade Referencial

Impõe que **o valor de uma chave externa tem que existir como elemento da chave primária da tabela relacionada** com aquela chave externa.

Verificações de integridade referencial

A maneira como o SGBD vai reagir quando há uma operação que leva à violação da integridade referencial pode ser configurada. Há as seguintes possibilidades:

- **Não deixa fazer qualquer operação que viole a integridade referencial**
- **Cascata**
 - **Apagamento em cascata.** Por exemplo, para cada registo que apaga na tabela Conta apaga também todos os registos a ele associados na tabela Cliente_banco (onde está definida a chave externa)
 - **Actualização em cascata.** Por exemplo, para cada registo que actualiza na tabela Conta actualiza também todos os registos a ele associados na tabela Cliente_banco (onde está definida a chave externa)

Integridade de Domínio

Força a que os valores de atributos estejam dentro de certos valores, sejam de preenchimento obrigatório, etc. Estas regras referem-se apenas a atributos de uma dada tabela (mas excluem a chave primária).

Ex:

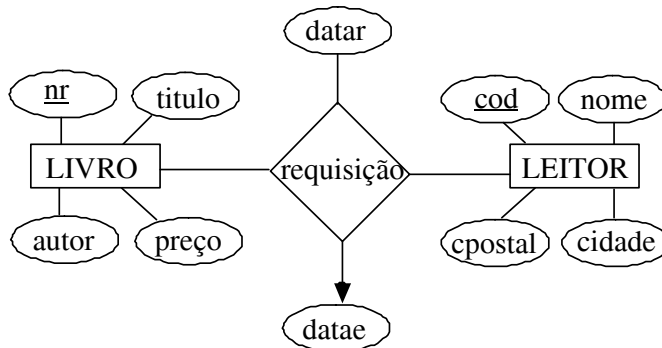
- Há um registo com o atributo nome não preenchido (diz-se que um atributo está a **Null** quando isso acontece). Como não faz sentido ter um registo de um professor sem ter o nome desse professor este campo deve estar obrigatoriamente preenchido.
- Há duas datas erradas. Se a tabela tem professores que estão presentemente a dar aulas então não podem ter nascido nem em 12-03-2060 nem em 18-01-1878.
- Há vários erros no atributo Salario. O salário não pode ser negativo e o salário da professora Joana Varela é demasiado alto, pelo que não poderia estar correcto.

SQL

Tratamento unificado da definição, manipulação e controlo dos dados, sendo útil para todas as classes de utilizadores.

BD Biblioteca

Esquema conceptual:



Esquema relacional:

- Livro (nr, titulo, autor, preço)
- Leitor (cod, nome, cpostal, cidade)
- Req (liv, lei, datar, datae)

BIBLIOTECA

Livro

NR	TITULO	AUTOR	PREÇO
100	Os Maias	Eça de Queiroz	1100
110	Os Lusíadas	Luís de Camões	490
120	A Selva	Ferreira de Castro	700
130	A Capital	Eça de Queiroz	1050
140	Terra Fria	Ferreira de Castro	850
150	A Relíquia	Eça de Queiroz	900

Req

LIV	LEI	DATAR	DATAE
100	1	95-01-01	95-02-06
110	2	95-01-05	95-03-05
120	2	95-02-15	95-02-25
100	3	95-03-10	95-03-20
130	6	95-06-15	
140	5	95-04-15	95-05-02
100	1	95-04-30	95-05-08
110	4	95-04-21	95-04-26
150	6	95-06-30	95-07-08
130	5	95-07-04	95-07-12

Leitor

COD	NOME	CPOST	CIDADE
1	António	1000	Lisboa
2	Chico	4000	Porto
3	Marina	1100	Lisboa
4	Zeca	4100	Porto
5	Manuel	4400	Gaia
6	Mafalda	4470	Matosinhos
7	Rui	1200	Lisboa

Criação de uma tabela

- **create table** tabela(
 - coluna tipo restrição_de_coluna,
 - coluna tipo restrição_de_coluna,
 - ...
 - restrição_de_tabela,
 - ...
 -);

Tipo de dados

- **Nvarchar** - Cadeia de caracteres de comprimento variável até 4096
- **Int** - Numeros inteiros
- **Real** - números reais
- **Datetime** - data incluindo a hora

Restrições de coluna

- **primary key** - chave primária = **unique** + **not null**
 - **unique** - não admite valores repetidos na coluna
- **DEFAULT constant_expression**
- **IDENTITY (seed , increment)**
- **not null** - obrigatório
- **null** - opcional, i.e., aceita valores nulos.
- **references tabela(coluna)**
identifica a chave primária ou alternativa de uma tabela referida por uma restrição de integridade referencial; a opção **on delete cascade** faz com que apagar uma chave referida apague automaticamente todos os respectivos referentes
- **check(condição)**
especifica uma condição que todas as linhas da tabela têm que satisfazer; a condição só se refere a valores na linha corrente

Definição do esquema em SQL

```
create table livro
(
  nr          int identity(1,1)          primary key,
  titulo     nvarchar(20) not null unique,
  autor      nvarchar(20),
  preço     money          default 0      );

create table leitor
(
  cod      int          primary key,
  nome     nvarchar(20) not null,
  cpost   int,
  cidade  nvarchar(20) );

create table req
(
  liv      int          foreign key references livro(nr),
  lei      int          foreign key references leitor(cod),
  datar    datetime,
  datae    datetime,
  constraint req_ck check datar<=datae,
  constraint req_pk primary key(liv, lei, datar) );
```

Restrições de tabela

- Podem referir-se a mais do que uma coluna, como:
 - **primary key** (col, col, ...) [seleccionar todas as colunas da chave, antes de picar a chave]
 - **foreign key**(col, col, ...) **references** tabela(col, col, ...)
- **unique** e **primary key** criam automaticamente índices
 - Criação explícita, por exemplo por razões de eficiência.

Comando Alter Table

```
Alter table <tabela> [<campos>]
  Add column <definição da coluna> |
  Alter column <definição da coluna> |
  Drop column column_name
```

Exemplo:

```
ALTER TABLE doc_exe ADD column_b INT IDENTITY
CONSTRAINT column_b_pk PRIMARY KEY
```

Comando Drop

```
Drop tabela
```

Elimina a tabela se:

Não houver referências para essa tabela
Estas especificarem **on delete cascade**

Exemplo:

```
drop livros
```

Comando Insert

INSERT INTO <tabela> [<campos>]
VALUES <valores> | <cmd select >

Insert into tabela values(val, val, ...);

Adiciona uma linha com todos os valores e pela ordem correcta

insert into tabela(col, col, ...) values(val, val, ...);

Adiciona uma linha só com os valores das colunas referidas

Insert into tabela(col, col, ...) select...

Adiciona uma ou mais linhas com os registos resultantes do comando select especificado.

insert into req values(130, 6, '95-06-15', null)

= equivale a =

insert into req(liv, lei, datar) values(130,6,'95-06-15')

Comando DELETE

DELETE FROM <tabela>

[WHERE <condições>]

Exemplo:

Delete from livro

where titulo = 'Os maias'

Comando UPDATE

UPDATE <tabela>

SET <atribuições>

[WHERE <condições>]

Exemplo:

Update livro set preço=preço *1,1

Where cod=150

SQL**Consultar vários dados específicos:**

Select nome, nacionalidade
from pilots where nome = 'shumacker';

Consultar todos os dados

select * from pilotos
where nome = 'shumaker';

%shumacker% - o símbolo % procura o que está antes e depois do nome

Apagar registos específicos de uma tabela

Delete from pilotos
Where nome = 'shumacker';

Apagar todos os registos de uma tabela

Delete from pilotos;

Actualizar registos numa tabela

Update piloto set nacionalidade = 2
Where nome = 'shumacker';

Actualizar todos os registos numa tabela

Update piloto set nacionalidade = 1;

TESTE EXEMPLO**1. Para que serve o comando alter table?**

Para modificar a definição de uma tabela.

2. Uma chave externa:

Pode não pertencer à chave primária.

3. Afirmações verdadeiras:

Numa relação R, não existem dois tuplos de R com valores iguais nos atributos que fazem parte de qualquer chave candidata.

Sendo R uma relação, uma instância de R consiste num conjunto de tuplos de R.

4. Qual dos seguintes comandos elimina todos os registos da tabela Vendas?

DELETE FROM Vendas

5. Qual o comando que insere dados numa tabela chamada Projects?

INSERT INTO Projects (ProjectName, ProjectDescription) VALUES ('Content Development', 'Website content development project')

6. Ao inserir dados numa tabela é sempre necessário especificar o nome dos campos a que os dados se destinam.

Falso

7. O número total teórico dos possíveis tuplos de uma relação é:

O produto cartesiano dos domínios dos atributos

8. O que é a chave primaria de uma tabela?

Coluna ou conjunto de colunas, irreduzível, que identificam de forma única os registos da tabela.

9. create table produtos(

produto_id int primary key,

nome_produto nvarchar(20) not null,

cor nvarchar(20)

)

create table catalogo(

fornecedor_id int foreign key references fornecedores(fornecedor_id) on delete set NULL,

produto_id int foreign key references produtos(produto_id) on delete set NULL,

preco money,

primary key(fornecedor_id, produto_id)

)

create table fornecedores(

fornecedor_id int primary key,

nome_fornecedor nvarchar(20) not null,

endereco nvarchar(20)

)

Indique o tipo de restrição de integridade que os seguintes comandos violam.

delete from fornecedores

Nenhuma

insert into fornecedores (nome_fornecedor, endereco) values ('Adidas','França')

Integridade de entidade

insert into fornecedores values(100,'Nike','Portugal'); insert into produtos values(1,'Sapatilha','Preto') insert into catalogo values(100,2,30)

Integridade referencial

insert into catalogo values(100,2,30); insert into catalogo values(100,1,30)

Integridade de domínio

insert into produtos values('a','Sapatilha','Preto')

Integridade de domínio

insert into produtos values(2,'Sapatilha Conforto com protecção extra e forro não sintético','Azul')

Integridade de domínio

10. A associação primary key-foreign key é utilizada para ...

Relacionar tabelas

11. O esquema de uma relação consiste:

Na especificação do nome da relação, do nome e domínio dos seus atributos

12. Considere a relação R(a ,b), sendo o domínio dos seus atributos respectivamente:a $\in \{ 5,20 \}$;b $\in \{ x,y,z,w \}$.**Qual o numero máximo possível de tuplos de R?**

8

13. create table produtos(**produto_id int primary key,****nome_produto nvarchar(20) not null,****cor nvarchar(20)****)****create table catalogo(****fornecedor_id int foreign key references fornecedores(fornecedor_id) on****delete set NULL,****produto_id int foreign key references produtos(produto_id) on delete set NULL,****preco money,****primary key(fornecedor_id, produto_id)****)****create table fornecedores(****fornecedor_id int primary key,****nome_fornecedor nvarchar(20) not null,****endereco nvarchar(20)****)****Qual a ordem correcta para criação das tabelas?**

produtos, fornecedores, catalogo

14. Faça a correspondência correcta:

Conjunto de atributos de uma relação que determina univocamente um tuplo da relação

Chave primária

Conjunto de atributos de uma relação que são chave de uma outra relação

Chave estrangeira

A definição é idêntica à de chave primária mas sem a restrição de só poder haver uma. Isto é, podem existir várias chaves candidatas mas só pode haver uma chave primária por relação.

Chave candidata

15. create table produtos(**produto_id int primary key,****nome_produto nvarchar(20) not null,****cor nvarchar(20)****)****create table catalogo(****fornecedor_id int foreign key references fornecedores(fornecedor_id)on delete set NULL,****produto_id int foreign key references produtos(produto_id) on delete set NULL,****preco money,****primary key(fornecedor_id, produto_id)****)****create table fornecedores(****fornecedor_id int primary key,****nome_fornecedor nvarchar(20) not null,****endereco nvarchar(20)****)****a) insert into fornecedores values(100,'Nike','Portugal')****b) insert into fornecedores values(110,'Adidas','França')****c) insert into produtos values(1,'Sapatilha','Preto')****d) insert into produtos values(2,'Sapatilha','Azul')**

- e) insert into catalogo values(110,1,30)
- f) insert into catalogo values(100,2,30)

Qual a ordem correcta para estes comandos?

b) c) e)

16. Considere as tabelas R e S, bem como os comandos I, II, e III, especificados em SQL do seguinte modo:

CREATE TABLE R (a int PRIMARY KEY, b int);

CREATE TABLE S (a int PRIMARY KEY, b int foreign key REFERENCES R (a) ON UPDATE CASCADE);

I: DELETE FROM R WHERE a=3;

II: UPDATE R SET a=5 WHERE a=3;

III: UPDATE S SET b=5;

Especifique qual (ou quais) dos comandos poderão dar uma mensagem de erro devido a uma violação da integridade referencial.

Apenas I e III

Exemplos de SQL

```

create database arvore

use arvore

create table pessoa(
bi bigint primary key,
nome nvarchar (50) not null,
datanascimento datetime,
sexo char (1) not null check (sexo='m' or sexo='f'),
bipai bigint,
bimae bigint)

create table casamento(
bi_marido bigint foreign key references pessoa (bi),
bi_mulher bigint foreign key references pessoa (bi),
data_inicio datetime,
data_fim datetime,
numero_filhos int default 0,
constraint pk_casamento primary key (bi_marido, bi_mulher),
constraint fk_datas check(data_fim > data_inicio))

drop table casamento

alter table casamento
add constraint verificadatas check (data_fim > data_inicio)

alter table casamento
add constraint verificabi check (bi_marido<>bi_mulher)

alter table casamento
add constraint fk1 foreign key (bi_marido) references pessoa on update
cascade

alter table casamento
add constraint numero default 0 for numero_filhos

```

```

create database biblioteca;

drop database biblioteca;

use biblioteca;

create table livro

(
nr int identity(1,1) primary key,
titulo nvarchar(20) not null unique,
autor nvarchar(20),
preço money default 0 );

create table leitor
(
cod int primary key,
nome nvarchar(20) not null,
cpost int,
cidade nvarchar(20) );

create table req

(
liv int foreign key references livro(nr),
lei int foreign key references leitor(cod),

```

```

    datar datetime,
    datae datetime,
    constraint req_ck check (datar <= datae),
    constraint req_pk primary key(liv, lei, datar) );

```

```
create database linhas
```

```
use linhas
```

```

create table comboios(
    numero int identity(2,2) primary key, -- este campo é gerado
    automaticamente
    categoria nvarchar (50) not null,
    designacao nvarchar (50) not null);

```

```

create table estacoes (
    cidade nvarchar (30) primary key,
    telefone char (20));

```

```

create table linha (
    cidade nvarchar (30) foreign key references estacoes (cidade) on
    update cascade,
    linha nvarchar (30),
    nordem int,
    primary key (cidade, linha)) -- constraint pk porque é chave
    composta
);

```

```
drop table horario
```

```

create table horario(
    cidade nvarchar (30) ,
    ncomboio int foreign key references comboios (numero) on update
    cascade,
    hora datetime
--constraint pk_horario primary key (cidade, comboio, hora),
--constraint fk1_horario_comboios foreign key (ncomboio) references
comboios(numero) on update cascade,
--constraint fk2_horario_estacoes foreign key (cidade) references
estacoes (cidade) on update cascade
);

```

```

select * from comboios
select * from estacoes
select * from horario
select * from linha

```

```

insert into comboios (categoria, designacao) values ('beta','beta
perpendicular')
insert into estacoes values ('Porto','227872328')
insert into estacoes values ('Aveiro','227872328')
insert into estacoes values ('Viseu','227872328')
insert into estacoes (cidade) values ('Evora')
insert into estacoes values ('Braga', null)
insert into horario (cidade, ncomboio, hora) values ('Porto','2',
'12:30')
insert into linha (cidade, linha, nordem) values ('Porto', 'A', '1')
insert into linha (cidade, linha, nordem) values ('Lisboa', 'C', '3')
insert into linha (cidade, linha, nordem) values ('Braga', 'B', '123')
insert into linha (cidade, linha, nordem) values ('Coimbra', 'A', '1')

```

```
insert into horario (cidade, ncomboio, hora) values ('Braga','4',
'17:30')
update estacoes set cidade = 'Porto 1' where cidade ='porto'
alter table comboios add ncarruagens int default 3
alter table comboios add constraint ncdefault default 3 for
ncarruagens
alter table comboios add combustivel int
insert into comboios (designacao, categoria)
values ('c11', 'alfa')
insert into comboios (designacao, categoria)
values ('c12', 'alfa')
insert into comboios (designacao, categoria)
values ('abc', 'beta')
alter table comboios drop ncarruagens
delete from comboios where designacao = 'beta perpendicular'
```