Ficha 6 - Grupo 1

PROMPT Criação da tabela de funcionários

```
drop table funcionarios cascade constraints;
```

```
create table funcionarios(
            num_funcionario number(6) not null,
            nome varchar2(30) not null,
            sexo char(1) not null CHECK ( sexo IN ( 'F' , 'M' )  ),
            data_entrada date,
            cod_departamento number(3) not null,
            primary key (num_funcionario),
            foreign key (cod_departamento) references departamentos
                );
```

PROMPT E agora crie-se a sequencia

```
drop sequence seq_funcionario;

create sequence seq_funcionario
start with 1
increment by 1;
```

PROMPT E insiram-se alguns funcionarios

```
insert into funcionarios values(seq_funcionario.nextval,'Filipa
Reis','F',to_date('1995.10.15','YYYY.MM.DD'),1);
insert into funcionarios values(seq_funcionario.nextval,'Alexandre
Silva','M',to_date('1984.09.23','YYYY.MM.DD'),1);
```

PROMPT Testem-se as restrições

```
insert into funcionarios
values(seq_funcionario.nextval,'Mario','N',to_date('1985.10.15','YYYY.MM.DD')
,1);
insert into funcionarios
values(seq_funcionario.nextval,'Mario','F',to_date('1985.10.15','YYYY.MM.DD')
,26);
```

Grupo 2

```
drop table lecionacao cascade constraints;
```

```
create table lecionacao(
            cod_docente number(5) not null,
            cod_cadeira number(5) not null,
            primary key (cod_docente,cod_cadeira),
            foreign key (cod_docente) references docentes,
            foreign key (cod_cadeira) references cadeiras
                );
```

PROMPT Metam-se alguns dados permitidos

```
insert into lecionacao values (1,4);
insert into lecionacao values (1,5);
insert into lecionacao values (3,4);
```

PROMPT E agora uns que não podem ser

```
insert into lecionacao values (100,4);
insert into lecionacao values (1,500);
```

PROMPT Vamos agora meter os responsáve**is** pelas cadeiras

```
alter table cadeiras add cod_docente number(5);
```

PROMPT E preencha-se esse campo

```
update cadeiras
set cod_docente =
    (select min(cod_docente)
     from docentes
     where docentes.cod_departamento = cadeiras.cod_departamento
     group by cod_departamento);
```

PROMPT Vamos agora às restrições

PROMPT A relação é total do lado de cadeiras

```
alter table cadeiras add constraint cad_total_docentes
    check (cod_docente is not null);
```

PROMPT O docente tem que ex**is**tir na BD

```
alter table cadeiras add constraint cadeiras_fk_docentes
    foreign key (cod_docente) references docentes;
```

PROMPT E tem que ser do departamento da cadeira

```
alter table docentes add constraint doc_dep
    unique (cod_docente,cod_departamento);
```

```
alter table cadeiras add constraint cadeira_dep_docente
    foreign key (cod_docente,cod_departamento) references
docentes(cod_docente,cod_departamento);
```

PROMPT Teste-se então

```
update cadeiras
set cod_docente = 4
where cod_cadeira = some (select cod_cadeira from cadeiras where nome =
'Bases de Dados');
```

Grupo 3

PROMPT mudar o historico_categorias para ter data de início e data de fim

Adicionar um novo campo para a data de fim:

```
alter table historico_categorias add data_fim date;
```

Adicionar uma restrição:

```
alter table historico_categorias add constraint ordem_datas
    check (data_fim >= data);
```

Criar uma view temporária contendo a informação, para cada docente e categoria do passado, sobre a data de inicio e a data de fim. Esta view vai ter 4 atributos: docente, categoria, inicio e fim.
Assumimos que as categorias estão inseridas sequencialmente na BD.

```
drop view mudancas;

create view mudancas as
    (select cod_docente as docente, A.cod_categoria as categoria,
        A.data as inicio, B.data-1 as fim
    from historico_categorias A inner join historico_categorias B using
(cod_docente)
    where A.cod_categoria = B.cod_categoria - 1);
```

```
select * from mudancas;
```

PROMPT Preenchimento do atributo com as datas de cessação de funções.

```
update historico_categorias
set data_fim = (select fim from mudancas
            where mudancas.docente = cod_docente and
            mudancas.categoria = cod_categoria)
where exists
    (select * from mudancas
    where mudancas.docente = cod_docente and
        mudancas.categoria = cod_categoria
    );
```

```
select * from historico_categorias;
```

PROMPT Actualizar cod_categoria de docentes. Notem que aqui se assume que apenas uma das categorias está a NULL.
Porquê o distinct?

```
update docentes
set cod_categoria = (select distinct cod_categoria
                from historico_categorias
                where historico_categorias.cod_docente = docentes.cod_docente
and
                    data_fim is null);
```

PROMPT Sempre que se insere um docente há que inserir o respectivo no historico.

```
drop trigger novo_docente;

create trigger novo_docente
    after insert on docentes
    for each row
    begin
        insert into historico_categorias values
            (:new.cod_docente, :new.cod_categoria, sysdate, null);
    end;
/
```

PROMPT Experimente-se met**end**o um novo docente

```
insert into docentes
    (select seq_docente.nextval as cod_docente, 'João Nunes' as nome,
cod_departamento,  cod_categoria
    from departamentos, categorias
    where departamentos.nome like '%In**for**matica' and categorias.nome like
'%Catedratico');
```

PROMPT E verifiquemos

```
select cod_docente, historico_categorias.cod_categoria, data, data_fim
from docentes inner join historico_categorias using (cod_docente)
where nome = 'João Nunes';
```

PROMPT Agora o apagar

```
drop trigger elimina_docente;

create trigger elimina_docente
    after delete on docentes
    for each row
    begin
        delete from historico_categorias
        where cod_docente = :old.cod_docente;
    end;
/
```

PROMPT Experimente-se apag**and**o o tal João Nunes

```
delete from docentes where nome = 'João Nunes';

select * from historico_categorias;
```

PROMPT Sempre que se **alter**a a categoria dum docente, há que actualizar o historico

```
drop trigger promove_doc;

create trigger promove_doc
    after update of cod_categoria on docentes
    for each row
    begin
        update historico_categorias
        set data_fim = sysdate
        where cod_docente = :new.cod_docente and data_fim is null;
        insert into historico_categorias values
            (:new.cod_docente, :new.cod_categoria, sysdate, null);
    end;
/
```

PROMPT Vamos experimentar isto, promov**end**o os docentes 1 e 2

```
update docentes
set cod_categoria = cod_categoria + 1
where cod_docente < 3;
```

PROMPT E vejamos que tudo ficou bem no historico

```
select *
from historico_categorias
where cod_docente <= 2;
```