Definição do esquema da base de dados

- o esquema da BD é composto pelas definições de todas as tabelas da BD.

- existem outros elementos (views, índices, triggers) que também fazem parte do esquema e que veremos mais tarde.

Definição de tabelas

```
CREATE TABLE <nome> (
   <lista de elementos>
);
```

- cada elemento consiste num atributo e no respectivo tipo de dados.

- os tipos mais comuns são: INTEGER, FLOAT, CHAR(n), VARCHAR(n), DATE, TIME.

- Exemplo:

```
CREATE TABLE Actores(
    nome            VARCHAR(50),
    morada          VARCHAR(70),
    sexo            CHAR,
    dataNascimento DATE
);
```

DATE e TIME

- o formato de DATE é 'yyyy-mm-dd'

- o formato de TIME é 'hh:mm:ss'. Poderá ter ainda fracções de segundo.

- exemplos:

  - DATE '2003-04-25'
    (25 de Abril de 2003)

  - TIME '15:32:04.5'
    (4 segundos e meio depois das 3:32 da tarde)

Declaração de chaves

- um atributo ou lista de atributos pode ser declarado como chave em SQL utilizando PRIMARY KEY ou UNIQUE.

- exemplo:

```
CREATE TABLE Actores(
    nome            VARCHAR(50) PRIMARY KEY,
    morada          VARCHAR(70),
    sexo            CHAR,
    dataNascimento DATE
);
```

Declaração de chaves (cont.)

- no caso da chave ser composta, temos de especificar um elemento à parte.

- exemplo:

```
CREATE TABLE Filmes(
    nome            VARCHAR(50),
    ano             INTEGER,
    duracao         INTEGER,
    aCores          BOOLEAN,
    nomeEstudio     VARCHAR(30),
    nomeRealizador VARCHAR(50),
    PRIMARY KEY (nome,ano)
);
```

PRIMARY KEY versus UNIQUE

- também se pode usar UNIQUE para especificar uma chave.

- cada tabela só pode ter uma PRIMARY KEY, mas pode ter vários UNIQUEs.

- os atributos declarados como PRIMARY KEY não podem ser NULL, mas os declarados como UNIQUE podem (e até podem ter vários NULLs).

Restrição NOT NULL

- podemos forçar um atributo a não poder ser NULL. Exemplo:

```
CREATE TABLE Alunos(
    nome         VARCHAR(50) NOT NULL,
    numero       CHAR(6) PRIMARY KEY,
    bi           CHAR(10) UNIQUE,
    numContrib   CHAR(10) UNIQUE,
    morada       VARCHAR(100)
);
```

```
-- 'bi', 'numContrib' e 'morada' ficam com NULL
INSERT INTO Alunos(nome,numero)
   VALUES ('Tó Tabelas','23384');

-- SGBD rejeita porque 'nome' não pode ser NULL
INSERT INTO Alunos(numero,bi)
   VALUES ('23384','7986866');
```

Chaves estrangeiras (integridade referencial)

- uma chave estrangeira é um conjunto de atributos que faz referência a um conjunto de atributos de uma outra tabela (eventualmente até poderia ser a própria tabela).

- os atributos referenciados têm de ser uma chave da outra tabela (declarados como PRIMARY KEY ou UNIQUE), daí o nome *chave estrangeira*.

- em SQL:

```
FOREIGN KEY( <lista de atributos> )
  REFERENCES <tabela> ( <lista de atributos> )
```

Exemplo

- `Participa(nomeFilme, anoFilme, nomeActor)`

- o valor do atributo `nomeActor` tem de existir na tabela de Actores.

- a mesma coisa para o par {`nomeFilme`,`anoFilme`} relativamente à tabela de Filmes.

```
CREATE TABLE Participa(
    nomeFilme   VARCHAR(50),
    anoFilme    INTEGER,
    nomeActor   VARCHAR(50),
    PRIMARY KEY (nomeFilme,anoFilme,nomeActor),
    FOREIGN KEY (nomeActor)
       REFERENCES Actores(nome),
    FOREIGN KEY (nomeFilme,anoFilme)
       REFERENCES Filmes(nome,ano)
);
```

Forçar a integridade referencial

Participa:

nomeFilme	anoFilme	nomeActor
...	...	...
The Fugitive	1993	Harrison Ford
Moulin Rouge	2001	Nicole Kidman
Indiana Jones	1981	Harrison Ford
...	...	...

Actores:

nome	morada	...
...	...	...
Nicole Kidman	26 Palm Dr., Hollywood	...
Harrison Ford	789 Palm Dr., Beverly Hills	...
Kevin Costner	88 Palm Dr., Hollywood	...
...	...	...

- vamos substituir Harrison Ford por Fernando Lobo no filme "The Fugitive". O que é que vai acontecer?

- e se apagarmos a Nicole Kidman da tabela de Actores?

Forçar a integridade referencial (cont.)

- um INSERT ou UPDATE na tabela Participa é rejeitado caso o actor não exista.
  - o SGBD rejeita a substituição de Harrison Ford por Fernando Lobo.

- um DELETE ou UPDATE à tabela de Actores pode fazer com que determindos valores na tabela Participa deixem de fazer sentido.
  - ex: Apagar a Nicole Kidman da tabela de Actores.
  - existem 3 alternativas possiveis de resolver o problema em SQL.

Forçar a integridade referencial (cont.)

As 3 alternativas são:

1. rejeitar modificações (opção por defeito em SQL).

2. SET NULL: nomeActor em Participa fica com NULL.

3. CASCADE: propagar as alterações.

   - apagar actor da tabela de Actores $\Longrightarrow$ SGBD apaga todos os tuplos de Participa que tenham esse actor.

   - actualizar actor na tabela de Actores $\Longrightarrow$ SGBD actualiza todos os tuplos de Participa que tenham esse actor.

Forçar a integridade referencial (cont.)

- quando declaramos uma chave estrangeira, podemos especificar SET NULL ou CASCADE, quer para "DELETEs" quer para "UPDATEs".

- se não o fizermos, o SGBD rejeita as modificações.

- sintaxe:

```
FOREIGN KEY( <lista de atributos> )
  REFERENCES <tabela> ( <lista de atributos> )
  ON UPDATE [SET NULL | CASCADE]
  ON DELETE [SET NULL | CASCADE]
```

Exemplo

```
CREATE TABLE Participa(
    nomeFilme   VARCHAR(50),
    anoFilme    INTEGER,
    nomeActor   VARCHAR(50),
    PRIMARY KEY (nomeFilme,anoFilme,nomeActor),
    FOREIGN KEY (nomeActor)
       REFERENCES Actores(nome)
       ON UPDATE CASCADE
       ON DELETE SET NULL,
    FOREIGN KEY (nomeFilme,anoFilme)
       REFERENCES Filmes(nome,ano)
);
```

- Se actualizarmos o nome de um actor na tabela de Actores, essa modificação é propagada para todos os tuplos da tabela Participa que fazem referência a esse actor.

- e se apagarmos um actor na tabela de Actores?

- SET NULL só faz sentido se a chave for declarada como UNIQUE.

Modificações à BD

- existem comandos para alterar o estado da BD.

- ao contrário dos SELECTs, os comandos de modificação não retornam uma tabela.

- existem 3 tipos de modificações:

  1. INSERT
     insere um ou mais tuplos.

  2. DELETE
     apaga um ou mais tuplos.

  3. UPDATE
     actualiza os valores de um ou mais tuplos.

INSERT

- para inserir um tuplo:

```
INSERT INTO <tabela>
VALUES( <lista de valores> );
```

- exemplo:

```
INSERT INTO Actores VALUES (
   'Nicole Kidman',
   '26 Palm Dr., Hollywood',
   'f',
   '1969-11-14'
);
```

INSERT (cont.)

- pode-se especificar uma lista de atributos.

- dois motivos para o fazer:

  1. quando não nos lembramos da ordem pela qual os atributos foram definidos na criação da tabela.

  2. quando não temos todos os valores e queremos que o SGBD preencha os atributos em falta com NULL (ou com um valor por defeito).

- para especificar um valor por defeito faz-se:

```
<atributo> <tipo> DEFAULT <valor>
```

Exemplo

Se a tabela de filmes for definida assim:

```
CREATE TABLE Filmes(
    nome            VARCHAR(50),
    ano             INTEGER,
    duracao         INTEGER,
    aCores          BOOLEAN DEFAULT true,
    nomeEstudio     VARCHAR(30),
    PRIMARY KEY (nome,ano)
);
```

e se fizermos,

```
INSERT INTO Filmes(nome,ano,nomeEstudio)
VALUES('Lion King',1994,'Disney');
```

`duracao` fica com `NULL`.
`aCores` fica com `true`.

Inserir vários tuplos de uma vez só

- pode-se inserir o resultado de uma query numa tabela.

```
INSERT INTO <tabela> ( <subquery> );
```

- exemplo: Inserir na tabela de actores todos os actores que tenham participado nalgum filme.

```
INSERT INTO Actores(nome) (
   SELECT DISTINCT nomeActor
   FROM Participa
);
```

- morada dos actores fica com NULL.

DELETE

- para apagar tuplos que satisfaçam uma determinada condição deve fazer-se:

```
DELETE FROM <tabela>
WHERE( <condição> );
```

- exemplo 1: Apagar todos os filmes cuja duração seja inferior a 120 minutos.

```
DELETE FROM Filmes
WHERE duracao < 120;
```

- exemplo 2: Apagar todos os filmes.

```
DELETE FROM Filmes;
```

UPDATE

- para actualizar tuplos de uma tabela deve fazer-se:

```
UPDATE <tabela>
SET <lista de atribuições>
WHERE( <condição> );
```

- exemplo 1: Actualizar a duração do King Kong de 1976 para 127 minutos.

```
UPDATE Filmes
SET duracao = 127
WHERE nome = 'King Kong'
  AND ano = 1976;
```

- exemplo 2: Passar todos os filmes anteriores a 1970 para preto e branco.

```
UPDATE Filmes
SET aCores = false
WHERE ano < 1970;
```

Modificar a definição de uma tabela

- retirar a coluna `aCores` da tabela de Filmes.

```
ALTER TABLE Filmes DROP aCores;
```

- voltar a colocar a coluna `aCores`.

```
ALTER TABLE Filmes ADD aCores BOOLEAN;
-- aCores fica com NULL
```

Eliminar tabelas

- utilizar DROP TABLE.

```
-- apaga a tabela de filmes
DROP TABLE Filmes;
```

- qual a diferença entre a

```
DROP TABLE Filmes;
```

  e

```
DELETE FROM Filmes;
```