SQL: Interrogações simples

Fernando Lobo

Base de Dados, Universidade do Algarve

Structured Query Language (SQL)

- É uma implementação da álgebra relacional incluindo os operadores extendidos.
- Contém um ou outro aspecto que não existe em álgebra relacional.
- Para além de operadores de interrogação, contém comandos de inserção, actualização e remoção de tuplos.
- Contém também comandos de definição de relações/tabelas.
- É uma linguagem usada em praticamente todos os SGBDs.

Porquê SQL?

- É uma linguagem de muito alto nível.
- Bem mais adequada para manipular relações do que linguagens como o C ou Java.
- O programador apenas se tem de preocupar com o conceito de relação/tabela.
- Não necessita de saber detalhes físicos sobre,
  1. o modo como as tabelas são implementadas, e
  2. o modo como as interrogações são executadas.
- As interrogações em SQL são optimizadas pelo SGBD.

SQL também tem limitações

- Não é uma linguagem completa.
- Não se consegue calcular o factorial de um número em SQL.
- Mas consegue-se fazer programas de 5 linhas que necessitariam de centenas de linhas de código em C.

Vamos usar este esquema para a BD de filmes

Estudios(nome, morada)

Realizadores(nome, categoria)

Filmes(nome, ano, duracao, aCores, nomeEstudio, nomeRealizador)

Actores(nome, morada, sexo, dataNascimento)

Participa(nomeFilme, anoFilme, nomeActor)

Interrogações simples: SELECT-FROM-WHERE

SELECT atributos
FROM uma ou mais relações
WHERE condição

```
SELECT nome
FROM Filmes
WHERE ano=1977;
```

SELECT-FROM-WHERE

- Não confundir SELECT com o operador de selecção da álgebra relacional.
- SELECT é equivalente à projecção (π).
- WHERE é equivalente à selecção (σ).

Operação em termos de álgebra relacional

```
SELECT nome
FROM Filmes
WHERE ano=1977;
```

1. Começar com a relação que aparece a seguir a FROM.
2. Aplicar o operador σ usando a condição do WHERE.
3. Aplicar o operador π usando os atributos de SELECT.

SQL versus Álgebra Relacional

Quais os filmes feitos em 1977?

SQL

```
SELECT nome
FROM Filmes
WHERE ano=1977;
```

Álgebra Relacional

$$\pi_{\text{nome}}\left(\sigma_{\text{ano}=1977}(\text{Filmes})\right)$$

Outra maneira de pensar

```
SELECT nome
FROM Filmes
WHERE ano=1977;
```

Imaginem uma variável que percorre todos os tuplos da relação.

- Verifica se satisfaz a condição WHERE.
- Se sim, envia os atributos especificados em SELECT para o output.

Asterisco dá todos os atributos

```
SELECT *
FROM Participa
WHERE nomeFilme='Moulin Rouge';
```

nomeFilme	anoFime	nomeActor
Moulin Rouge	2001	Nicole Kidman
Moulin Rouge	2001	Ewan McGregor
Moulin Rouge	2001	Jim Broadbent

Mudar o nome a colunas

- utilizar AS '<novo nome>'

```
SELECT nome AS 'nome do filme'
FROM Filmes
WHERE ano=1977;
```

nome do filme
Star Wars
Saturday Night Fever
...

Operadores lógicos

- a cláusula WHERE pode ter operadores lógicos (AND, OR, NOT)
- pode-se usar os operadores relacionais habituais
  (= , <> , < , > , <= , >=)

```
SELECT nome, duracao
FROM Filmes
WHERE estudio='Disney' AND ano=1977;
```


Expressões em SELECTs

- pode-se usar expressões como elementos da cláusula SELECT.

```
SELECT nome, ano,
       duracao/60 AS 'duração em horas'
FROM Filmes;
```

nome	ano	duração em horas
Star Wars	1977	2.07
Empire Strikes Back	1980	2.38
Return of the Jedi	1983	2.75
Moulin Rouge	2001	2.07

Padrões (pattern matching em strings)

- a condição WHERE pode especificar uma comparação entre uma string e um padrão.
- <atributo> LIKE <padrão>
  <atributo> NOT LIKE <padrão>
- um padrão é uma string em que % significa “0 ou mais caracteres” e _ significa “apenas 1 caracter”.

Exemplo

Quais são os filmes cujo nome contem a palavra King?

```
SELECT nome, ano
FROM Filmes
WHERE nome LIKE '%King%';
```

nome	ano
King Kong	1933
King Kong	1976
Lion King	1994

SQL trabalha com sacos

```
SELECT nomeEstudio
FROM Filmes;
```

nomeEstudio
Fox
Fox
Fox
Disney
Paramount
Disney
...

Usar DISTINCT para eliminar repetidos

```
SELECT DISTINCT(nomeEstudio)
FROM Filmes;
```

nomeEstudio
Fox
Disney
Paramount
Universal
Warner Brothers

Ordenação de tuplos

- **ORDER BY** <lista de atributos>
- ordem crescente por defeito
  (colocar **DESC** para ordem decrescente)
- ordem lexicográfica para strings.

Exemplos

Filmes da Disney ordenados por ano, e dentro de cada ano, ordenados por nome.

```
SELECT *
FROM Filmes
WHERE nomeEstudio='Disney'
ORDER BY ano, nome;
```

Filmes da Disney ordenados por ordem decrescente de duração, e em caso de empate, ordenados por ordem crescente de nome.

```
SELECT *
FROM Filmes
WHERE nomeEstudio='Disney'
ORDER BY duracao DESC, nome;
```