

Bases de Dados



Universidade do Porto
Faculdade de Engenharia
FEUP

1

- **INTRODUÇÃO**
- **MODELOS CONCEPTUAIS**
 - Diagrama de Classes UML
 - Modelo Entidade-Associação (E-A)
- **MODELO RELACIONAL**
- **LINGUAGEM DE DEFINIÇÃO DE DADOS**
- **INTERROGAÇÃO DE DADOS**
 - Álgebra relacional
- **LINGUAGEM DE MANIPULAÇÃO DE DADOS (LMD) SQL**

Observação: baseado em slides desenvolvidos pelo Prof. Gabriel David (FEUP)

Origem

2

- Introduzida em 1976 como Linguagem de Manipulação de Dados (LMD) para System R (IBM)
- Primeira implementação comercial em 1979 (Oracle)
- Linguagem padrão de acesso a BD relacionais
- Normalização ANSI e ISO: SQL89, SQL92, SQL3
- Objectivo principal
 - Tratamento unificado da definição, manipulação e controlo dos dados, sendo útil para todas as classes de utilizadores.

Instruções da LMD SQL

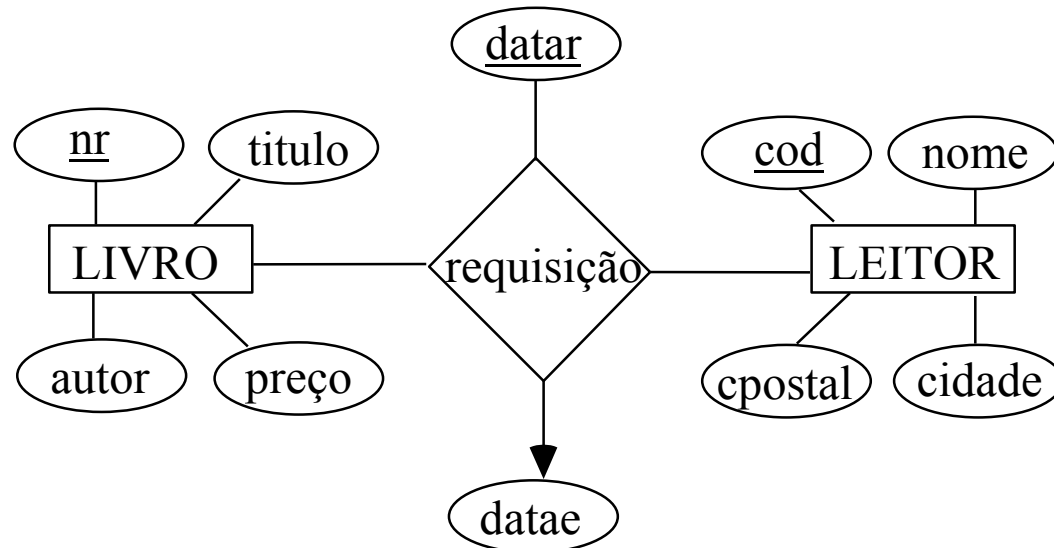
3

- INSERT: permite inserir registos;
- UPDATE: permite alterar valores de registos já existentes;
- DELETE: permite eliminar registos;
- SELECT: permite obter respostas a perguntas a partir dos dados existentes nas tabelas.
- Existem outras mas estas são as standard.

BD Biblioteca

4

- **esquema conceitual**



- Um leitor não pode requisitar o mesmo livro mais do que uma vez por dia.

- **esquema relacional**

- **livro**(nr, titulo, autor, preço)
- **leitor**(cod, nome, cpostal, cidade)
- **req**(liv → livro.nr, lei → leitor.cod, datar, datae)

Definição do esquema em SQL

5

create table livro

```
(  nr          number(4)  primary key,  
   titulo      varchar2(20) not null,  
   autor       varchar2(20),  
   preço       number(4) );
```

create table leitor

```
(  cod          number(4)  primary key,  
   nome        varchar2(20) not null,  
   cpost       number(4),  
   cidade      varchar2(20) );
```

create table req

```
(  liv          number(4)  references livro,  
   lei          number(4)  references leitor,  
   datar        date,  
   datae        date,  
   constraint req_ck check (datar<=datae),  
   constraint req_pk primary key(liv, lei, datar) );
```

Carregamento das tabelas

6

- **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 req VALUES (130, 6, '2013-06-15', null);**
equivalente a:
**INSERT INTO req (liv, lei, datar) VALUES
(130,6,'2013-06-15');**

BIBLIOTECA

7

LIVRO

NR	TITULO	AUTOR	PREÇO
100	Os Maias	Eça de Queiroz	6.50€
110	Os Lusíadas	Luís de Camões	2.50€
120	A Selva	Ferreira de Castro	3.50€
130	A Capital	Eça de Queiroz	5.25€
140	Terra Fria	Ferreira de Castro	4.25€
150	A Relíquia	Eça de Queiroz	4.50€

BIBLIOTECA (2)

8

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

BIBLIOTECA (3)

9

REQ

LIV	LEI	DATAR	DATAE
100	1	2013-01-01	2013-02-06
110	2	2013-01-05	2013-03-05
120	2	2013-02-15	2013-02-25
100	3	2013-03-10	2013-03-20
130	6	2013-06-15	
140	5	2013-04-15	2013-05-02
100	1	2013-04-30	2013-05-08
110	4	2013-04-21	2013-04-26
150	6	2013-06-30	2013-07-08
130	5	2013-07-04	2013-07-12

Instrução UPDATE

10

- **UPDATE tabela SET col=val, col = val, ... [WHERE ...];**
 - Atribui valores às colunas.
 - [] indica que é opcional.
 - Caso não exista a cláusula WHERE, as alterações afectam todos os registos da tabela.
 - UPDATE req SET datae= '2013-06-22'
WHERE liv =130 AND datar= '2013-06-15';
 - ✦ As funções para lidar com datas variam entre Sistemas Gestores de Bases de Dados (SGBDs)
 - ✦ O SQLite tem 5 funções para lidar com datas. Ver:
 - http://www.sqlite.org/lang_datefunc.html

Instrução DELETE

11

- **DELETE FROM tabela [WHERE ...];**
 - Elimina registos de uma tabela.
 - [] indica que é opcional.
 - Caso não exista a cláusula WHERE, são eliminados todos os registos da tabela.
 - DELETE FROM leitor WHERE cod =7;

Instrução SELECT

12

- **A instrução SELECT:**

- É muito versátil permitindo responder a quase todas as perguntas;
- Não consegue resolver o fecho transitivo. Por exemplo: saber todos os descendentes de uma dada pessoa;
- Mas para ser assim tão versátil tem necessariamente uma sintaxe com alguma complexidade. Por isso, vamos aprender a instrução SELECT através da resolução de casos.

Primeira pergunta

13

1 Mostrar toda a informação sobre todos os livros.

SELECT * FROM livro;

- as perguntas podem ocupar mais do que uma linha e em formato livre, por uma questão de legibilidade
- fim da pergunta: ;
- sintaxe errada : < mensagem explicativa >
- todas as colunas da tabela : *
- obrigatório haver **select** e **from**
 - ✦ No SQLite é possível não colocar o FROM;
 - ✦ Ex.: `SELECT 25*363+8;`

Resposta 1

14

NR	TITULO	AUTOR	PREÇO
100	Os Maias	Eça de Queiroz	6.50€
110	Os Lusíadas	Luís de Camões	2.50€
120	A Selva	Ferreira de Castro	3.50€
130	A Capital	Eça de Queiroz	5.25€
140	Terra Fria	Ferreira de Castro	4.25€
150	A Relíquia	Eça de Queiroz	4.50€

Seleccção simples

15

2 Listar código e nome dos leitores cujo código é menor que 5.

```
SELECT cod, nome  
FROM leitor  
WHERE cod < 5;
```

- **select-from-where** assemelha-se ao cálculo relacional e faz uma escolha horizontal (selecção), seguida de uma escolha vertical (projecção)

COD	NOME
1	António
2	Chico
3	Marina
4	Zeca

Filtro mais elaborado

16

3 Listar o nome e a cidade dos leitores com nome a começar por 'M' e código entre 2 e 5.

```
SELECT nome, cidade  
FROM leitor  
WHERE nome LIKE 'M%'  
AND cod BETWEEN 2 AND 5;
```

=

```
SELECT nome, cidade  
FROM leitor  
WHERE nome LIKE 'M%'  
AND cod >= 2 AND cod <=5;
```

NOME	CIDADE
Marina	Lisboa
Manuel	Gaia

Pesquisa com cadeias

17

- Comparação com uma cadeia usando like:
 - % vale por qualquer sequência de 0 ou mais caracteres:
nome like 'M%' (Oracle, SQLite) **nome like 'M*'** (Access)
 - ✦ é comparação verdadeira com 'Marina', 'M'
 - O _ (?) vale por qualquer letra (uma e uma só);
nome like 'M_r%' **nome like 'M?r*'**
 - ✦ é comparação verdadeira com 'Mar', 'Maria', 'Moreira'
- Usando = faz-se a igualdade literal:
nome = 'M_r%'
 - ✦ só é verdade se nome for 'M_r%'

Eliminação de repetidos

18

4 Seleccionar as cidades com código postal superior a 2000.

```
SELECT cidade  
FROM leitor  
WHERE cpost > 2000;
```

- como vários leitores são da mesma cidade vão aparecer valores repetidos no resultado

CIDADE
Porto
Porto
Gaia
Matosinhos

Resposta com conjunto

19

```
SELECT DISTINCT cidade  
FROM leitor  
WHERE cpost > 2000;
```

❑ Em Oracle, forçar valores distintos tem como efeito lateral a ordenação

CIDADE
Gaia
Matosinhos
Porto

Filtro complexo

20

5 Seleccionar os livros do Eça com preço superior a 5€ e todos os livros de preço inferior a 3.75 € indicando o autor, o título, o preço e o número.

SELECT autor, titulo, preco, nr

FROM livro

WHERE autor LIKE '%Eca%' AND preco > 5 OR preco < 3.75;

AUTOR	TITULO	PREÇO	NR
Eça de Queiroz	Os Maias	6.50€	100
Luís de Camões	Os Lusíadas	2.50€	110
Ferreira de Castro	A Selva	3.50€	120
Eça de Queiroz	A Capital	5.25€	130

Expressões aritméticas

21

6 Escrever o número de dias que durou cada requisição nos casos em que duraram até 10 dias.

```
SELECT liv, lei, julianday(datae) – julianday(datar) Duracao  
FROM req  
WHERE Duracao <= 10;
```

- para renomear uma coluna, indica-se o novo nome a seguir à especificação da mesma, como no exemplo.

Expressões lógicas

22

LIV	LEI	Duração
120	2	10
100	3	10
100	1	8
110	4	5
130	5	8

- operadores reconhecidos, por ordem de precedência:
- operadores aritméticos
 - + , - (binário); || (concatenação)
 - * , /
 - + , - (unário)
- operadores de comparação
- operadores lógicos
 - not
 - and
 - or

Operadores de comparação

23

=, <>, <, >, <=, >=	igual, diferente, menor, maior, menor ou igual, maior ou igual
[not] in	pertença a conjunto
[not] between x and y	$x \leq \text{valor} \leq y$
exists	Sub-pergunta com pelo menos uma linha no resultado
x [not] like y	compara com padrão
is [not] null	é valor nulo

Dificuldades com operadores

24

- **in**
select * from leitor
where cidade
in ('Lisboa','Porto')
- **not in** dá nulo (sem resultado) se algum dos elementos do conjunto for nulo

 cidade not in ('Lisboa','Porto', null) é equivalente a
 cidade != 'Lisboa' and cidade != 'Porto' and cidade != null
- qualquer comparação com nulo dá nulo, excepto a comparação **is null**.

Ordenação da saída

25

7 Obtenha uma lista com os autores, livros e preços ordenada decrescentemente por autor e decrescentemente por preço.

SELECT autor, titulo, preco
FROM livro
ORDER BY autor DESC, preco DESC;

AUTOR	TITULO	PREÇO
Luís de Camões	Os Lusíadas	2.50
Ferreira de Castro	Terra Fria	4.25
Ferreira de Castro	A Selva	3.50
Eça de Queiroz	Os Maias	6.50
Eça de Queiroz	A Capital	5.25
Eça de Queiroz	A Relíquia	4.50

Funções de agregação

26

8 Obtenha o preço médio, valor total e o número de livros da biblioteca, bem como o valor do livro mais caro e o do mais barato (ufff...).

```
SELECT AVG(preco), SUM(preco), COUNT(*),  
MAX(preco), MIN(preco)  
FROM livro;
```

avg(preco)	sum(preco)	count(*)	max(preco)	min(preco)
4.4167	26.5	6	6.5	2.5

Agrupamento de linhas

27

9 Calcule o preço médio dos livros de cada autor.

```
SELECT autor, AVG(preco)  
FROM livro  
GROUP BY autor;
```

	AUTOR	AVG(PREÇO)
}	Eça de Queiroz	5.4167
}	Luís de Camões	2.5
}	Ferreira de Castro	3.875

Agrupamento de linhas com filtro

28

10 Calcule o preço médio dos livros de cada autor, mas só para médias inferiores a 2.5€.

```
SELECT autor, AVG(preco)
FROM livro
WHERE AVG(preco) <= 2.5
GROUP BY autor;
```

ERRO!
misuse of aggregate: AVG()

having selecciona as
linhas da agregação
where selecciona as
linhas da tabela base

```
SELECT autor, AVG(preco)
FROM livro
GROUP BY autor
HAVING AVG(preco) <= 2.5;
```

certo!

AUTOR	AVG(PREÇO)
Luís de Camões	2.5

Perguntas encaixadas

29

11 Obtenha o título e preço do livro mais caro dos autores que começam por E.

Pergunta (1ª tentativa, parece a mais lógica ... para alguns):

```
SELECT titulo, MAX(preco)  
FROM livro  
WHERE autor LIKE 'E%';
```

Funciona em SQLite, mas não funciona na maioria dos outros SGBDs.

Mas cuidado! Esta forma não faz muito sentido! Veja o que acontece se em vez de usar MAX usar AVG ou SUM. Nesses casos, o título apresentado é o último devolvido pela selecção antes da agregação.

Subpergunta

30

Na verdade, este pedido é constituído por duas perguntas:

- 1 Qual é o preço máximo dos livros escritos por autores que começam por E?
- 2 Qual o título do livro cujo preço é igual ao determinado acima e cujo autor começa por E? (esta condição de começar por E não é redundante...)

```
SELECT titulo, preco
FROM livro
WHERE preco = (
  SELECT MAX(preco)
  FROM livro
  WHERE autor LIKE 'E%' )
AND autor LIKE 'E%';
```

TITULO	PREÇO
Os Maias	6.5

Exagerando...

31

12 Seleccione o título do segundo livro mais caro.

```
SELECT titulo  
FROM livro  
WHERE preco = (  
    SELECT MAX(preco)  
    FROM livro  
    WHERE nr NOT IN (  
        SELECT nr  
        FROM livro  
        WHERE preco = (  
            SELECT MAX(preco)  
            FROM livro))));
```

TITULO
A Capital

É possível demonstrar teoricamente que qualquer relação que se consiga extrair da BD com SQL, extrai-se com uma única pergunta (nem sempre dá muito jeito...).

Análise

32

- 1) `preçomax :=`
SELECT MAX(preco)
FROM livro;
determina o preço máximo de todos os livros.
- 2) `numeromax :=`
SELECT nr
FROM livro
WHERE preco = preçomax;
números dos livros que custam o preço máximo.
- 3) `segundopreço :=`
SELECT MAX(preco)
FROM livro
WHERE nr NOT IN numeromax;
máximo preço dos livros cujo número é diferente do dos livros com preço máximo (ou seja, o segundo maior preço...).
- 4) `resultado :=`
SELECT titulo
FROM livro
WHERE preco = segundopreço;
determina o título dos livros com preço igual ao segundo maior preço. E já está!

Perguntas com várias tabelas

33

13 Escreva os títulos e datas de requisição dos livros requisitados depois de 2013-04-01.

```
SELECT titulo, datar  
FROM livro, req  
WHERE nr = liv  
AND datar >= '2013-04-01';
```

TITULO	DATAR
Os Maias	2013-04-30
Os Lusíadas	2013-04-21
A Capital	2013-07-04
A Capital	2013-06-15
Terra Fria	2013-04-15
A Relíquia	2013-06-30

Núcleo da álgebra relacional

34

- o conjunto de cláusulas **select-from-where** é equivalente a um produto cartesiano, seguido de uma selecção e de uma projecção:

select **campo**₁, ..., **campo**_n
from **tabela**₁, ..., **tabela**_m
where F;

<=>

$\Pi_{\text{campo}_1, \dots, \text{campo}_n}(\sigma_F(\text{tabela}_1 \times \dots \times \text{tabela}_m))$

<=>

$\Pi_{\text{campo}_1, \dots, \text{campo}_n}(\text{tabela}_1 \bowtie_{F'} \dots \bowtie_{F''} \text{tabela}_m)$

Inclusão em conjunto

35

14 Liste, para cada requisição, o título do livro e o nome do leitor, no caso de o código postal ser 1000, 4000 ou 4470.

- **SELECT** titulo, nome
FROM livro, req, leitor
WHERE nr = liv AND lei = cod
AND cpost IN (1000, 4000, 4470);
- Equivalente a:
SELECT titulo, nome
FROM livro, req, leitor
WHERE nr = liv AND
lei = cod AND
(cpost =1000 OR cpost = 4000 OR cpost = 4470);
parênteses obrigatórios, atendendo à precedência

Resposta 13

36

TITULO	NOME
Os Maias	Antonio
Os Lusíadas	Chico
A Selva	Chico
A Capital	Mafalda
A Reliquia	Mafalda

Condições sobre tuplos

37

15 Quantos Antónios moram em Lisboa e quantos Zecas moram no Porto?

```
SELECT nome, cidade, COUNT(*)                (Oracle)  
FROM leitor  
WHERE (nome, cidade) IN  
      ((('Antonio','Lisboa'),('Zeca', 'Porto'))  
GROUP BY nome, cidade;
```

- Equivalente a:

```
SELECT nome, cidade, COUNT(*)  
FROM leitor  
WHERE nome = 'Antonio' AND cidade = 'Lisboa'  
OR nome = 'Zeca' AND cidade = 'Porto'  
GROUP BY nome, cidade;
```

Resposta 14

38

NOME	CIDADE	COUNT(*)
Zeca	Porto	1

Agregação de agregação

39

16 Procure o livro cujas requisições têm maior duração média, exceptuando 'Terra Fria'.

```
SELECT titulo, AVG(julianday(datae) – julianday(datar))  
FROM livro, req  
WHERE nr = liv AND titulo != 'Terra Fria'  
GROUP BY titulo  
HAVING AVG(julianday(datae) – julianday(datar)) = (  
    SELECT MAX(media) FROM (  
        SELECT AVG(julianday(datae) – julianday(datar)) media  
        FROM req, livro  
        WHERE titulo != 'Terra Fria' AND nr = liv  
        GROUP BY titulo));
```

Autojunção

40

17 Obtenha a lista dos pares de pessoas que moram na mesma cidade.

```
SELECT p.nome, q.nome  
FROM leitor p, leitor q  
WHERE p.cod != q.cod AND p.cidade = q.cidade;
```

- para responder a esta pergunta, precisamos de duas *cópias* da tabela de leitores; como não temos duas cópias físicas, criamos duas cópias lógicas: **p** e **q**.
- Ex: **p.cidade = q.cidade** faz a junção das duas tabelas **p** e **q** sobre o atributo **cidade**.

Subtracção de conjuntos

41

18 Obtenha os leitores que não requisitaram o livro 150.

```
SELECT nome  
FROM leitor  
WHERE cod NOT IN  
  (SELECT lei  
    FROM req  
    WHERE liv = 150);
```

- Alternativa:
 SELECT cod FROM leitor
 MINUS
 SELECT lei FROM req
 WHERE liv = 150;

Reunião e intersecção

42

19 Quais os dias em que houve requisições ou entregas de livros?
E quais os dias em que houve requisições e entregas?

Pergunta da reunião:

```
SELECT datae FROM req  
UNION  
SELECT datar FROM req;
```

Pergunta da intersecção:

```
SELECT datae FROM req  
INTERSECT  
SELECT datar FROM req;
```

Operador all

43

20 Quais os livros mais caros do que (todos) os livros do Ferreira de Castro?

```
SELECT titulo      ¬SQLite  
FROM livro  
WHERE preco > ALL  
  (SELECT preco  
    FROM livro  
    WHERE autor =  
    'Ferreira de Castro');
```

- operador **all** exige que a comparação seja verdadeira para todos os valores do resultado da subpergunta.

Alternativa (viável em SQLite):

```
SELECT titulo  
FROM livro  
WHERE preco >  
  (SELECT MAX(preco)  
    FROM livro  
    WHERE autor =  
    'Ferreira de Castro');
```

Operador some (any).

44

21 Quais os livros mais baratos do que algum livro do Eça?

SELECT titulo
FROM livro
WHERE preco < SOME
 (SELECT preco
 FROM livro
 WHERE autor LIKE 'Eça%');

¬SQLite

ou ANY

Alternativa (viável em SQLite):

SELECT titulo FROM livro
WHERE preco <
 (SELECT MAX(preco)
 FROM livro
 WHERE autor LIKE 'Eça%');

- operador some (any) exige que a comparação seja verdadeira para pelo menos um dos valores do resultado da subpergunta

Operadores in e exists

45

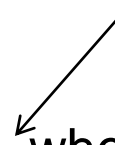
22 Quais os titulos dos livros requisitados depois de 2013-06-20?

```
SELECT titulo  
FROM livro  
WHERE nr IN  
  (SELECT liv FROM req  
    WHERE datar > '2013-06-20') ;
```

Alternativa:

```
SELECT titulo  
FROM livro  
WHERE EXISTS -- where 0 <  
  (SELECT * -- (SELECT COUNT(*)  
    FROM req  
    WHERE nr = liv  
    AND datar > '2013-06-20') ;
```

Símbolo que
significa
comentário



Sub-pergunta variável

46

- operador **exists** testa se o resultado da sub-pergunta não é vazio (o mesmo que **0 <**); **not exists** — **0 =**
- Sub-pergunta *constante*: na primeira versão, a sub-pergunta pode ser substituída pelo seu resultado (130,150) e este usado na pergunta exterior — avaliação de dentro para fora
- Sub-pergunta *variável*: na segunda versão, para cada tuplo da pergunta exterior (linha de livro), a sub-pergunta interior tem que ser reavaliada, pois contém uma referência a um atributo (**nr**) da tabela declarada na pergunta exterior — avaliação de fora para dentro.

Contagem

47

23 Obtenha os números e títulos dos exemplares que foram requisitados mais do que uma vez.

```
SELECT nr, titulo  
FROM livro, req  
WHERE nr = liv  
GROUP BY nr, titulo  
HAVING COUNT(*) > 1 ;
```

```
SELECT DISTINCT nr, titulo  
FROM livro, req r1, req r2  
WHERE nr = r1.liv AND nr = r2.liv  
AND r1.datar != r2.datar;
```

Alternativa (se em cada dia, cada exemplar for requisitado no máximo uma vez):

```
SELECT nr, titulo  
FROM livro, req  
WHERE nr = liv AND  
datar != SOME  
  (SELECT datar  
   FROM req  
   WHERE livro.nr = liv);
```

--→SQLite

Pertença de tuplos

48

24 Obtenha o número, título e preço das obras que têm mais do que um exemplar na biblioteca, com preço inferior a 4.50€.

```
SELECT I1.nr, I1.titulo, I1.preco  
FROM livro I1, livro I2  
WHERE I1.preco < 4.5 AND  
I1.preco < 4.5 AND  
I1.autor = I2.autor AND  
I1.titulo = I2.titulo AND  
I1.nr != I2.nr;
```

○ Utilizando uma sub-pergunta

```
SELECT nr, titulo, preco  
FROM livro I  
WHERE (titulo, autor) IN  
  (SELECT titulo, autor  
   FROM livro  
   WHERE nr != I.nr)  
AND preco < 4.5;
```

→SQLite

Sub-pergunta variável

49

- estratégia de *divisão e conquista*:

SELECT nr, titulo, preco

FROM livro l

WHERE 1 <

(SELECT COUNT(*)

FROM livro

WHERE titulo = l.titulo AND autor = l.autor AND preco < 4.5) ;

Quantificação universal

50

25 Quais os leitores que leram todos os livros?

```
SELECT DISTINCT nome      -- R
FROM leitor
WHERE cod NOT IN
  (SELECT cod AS cod1 --  $\Pi_{lei}(T-S)$ 
   FROM livro, leitor -- T
   WHERE NOT (nr IN
    (SELECT liv      -- S
     FROM req
     WHERE lei=cod1))));
```

○ Manipulação de conjuntos:

- $T = \Pi_{nr}(\text{livro}) \times \Pi_{cod}(\text{leitor})$
- $S = \Pi_{liv, lei}(req)$
- $R = \text{leitor} - \Pi_{lei}(T-S)$

Utiliza uma sub-
pergunta variável.

Formulações alternativas

51

select nome
from leitor
where not exists
 (select nr
 from livro
 where not exists
 (select lei
 from req
 where liv = nr and lei = cod));

- formulação directa das expressões de cálculo, usando uma sub-pergunta variável para cada elemento do produto cartesiano leitor x livro e o operador not exists

Estratégia da contagem

52

```
select nome
from leitor
where cod in
  (select lei
   from req
   group by lei
   having count(distinct liv) =
    (select count(*)
     from livro));
```

- operador **distinct** crucial para não contar duplicados

Inserção

53

26 Insira a requisição dos livros 100 e 120 feita pelo leitor 4 a 88-07-11.

```
insert into req(liv,lei,datar)  
values(120, 4, '88-07-11');
```

```
insert into req(liv,lei,datar)  
values(100, 4, '88-07-11') ;
```

- se se dessem valores a todos os atributos não era necessário indicar a lista de atributos a seguir ao nome da tabela
- os atributos não preenchidos ficam com os valores por omissão definidos para a coluna ou com valor nulo

Memorização de resultado

54

27 Insira, na tabela dos perdidos, os livros requisitados há mais de 30 dias.

create table perdidos
(nr number(4) primary key,
titulo varchar2(20) not null,
autor varchar2(20),
preco number(4));

- a tabela perdidos tem existir antes de fazer o insert
- “now” quando usado numa função de datas/tempo devolve o timestamp corrente.
- Podem-se fazer os dois passos num só através da forma create table perdidos as select ...

insert into perdidos
(select * from livro where nr in
(select liv from req
where julianday(“now”) –
julianday(datar) > 30
and datae is null);

create table if not exists perdidos as
select * from livro where nr in
(select liv from req
where julianday("now") –
julianday(datar) > 30
and datae is null);

Apagar

55

28 Retire os livros mencionados na pergunta anterior da tabela dos livros.

```
DELETE FROM livro  
where nr in  
  (select liv from req  
    where julianday("now") –  
      julianday(datar) > 30  
    and datae IS NULL);
```

- só se pode apagar numa tabela de cada vez
- pode ser usada qualquer pergunta para seleccionar os registos a apagar.

Modificar

56

29 Actualize o preço dos livros de código superior a 130 com 20% de inflação.

```
UPDATE livro  
SET preco = preco * 1.2  
WHERE nr > 130;
```

- só se pode actualizar numa tabela de cada vez, mas pode haver **set** para vários atributos em simultâneo
- a cláusula **set** admite qualquer expressão para modificar um campo, inclusive o resultado de uma pergunta, no caso de retornar apenas um valor

Língua natural

57

30 Quais os códigos dos leitores que requisitaram o livro 110 ou o livro 120? Quais os códigos dos leitores que requisitaram o livro 110 e o livro 120?

```
select lei  
from req  
where liv = 110 or liv = 120;
```

- para a disjunção, a resposta inclui os leitores 2 e 4; no caso da conjunção a pergunta

```
select lei  
from req  
where liv = 110 and liv = 120;
```

Conjunção

58

- dá um resultado vazio quando se estava à espera que desse 2! O problema é que não há nenhuma requisição que seja simultaneamente dos livros 110 e 120.
- reformulação, considerando que se tem que comparar duas requisições:
select a.lei
from req a, req b
where a.lei=b.lei and a.liv = 110 and b.liv = 120;
- a linguagem natural é muito traiçoeira.

Vistas implícitas

59

31 Quais os títulos dos livros que estão requisitados?

```
select titulo
from livro, (select liv, lei
              from req
              where datae is null) requisitados
where nr = requisitados.liv;
```

- Em SQL/92 é possível usar sub-perguntas como se fossem vistas, em várias situações, em especial na cláusula **from**.

Junções explícitas

60

32 Qual o número de livros requisitados por cada leitor, indicando o seu código e nome.

```
select cod, nome, count(liv)  
from req inner join leitor on lei=cod  
group by cod, nome;
```

- Em SQL/92 é possível usar junções explícitas como perguntas ou na cláusula **from**, com as variantes:
 - **cross** (produto cartesiano) (sem condição **on**)
 - **inner** (junção normal, por omissão)
 - **outer** (junção externa), ver próximo slide
 - **natural** (junção natural; pode também ser externa) (sem condição)

Junções externas

61

33 Qual o número de livros requisitados por cada leitor, indicando o seu código e nome.

```
select cod, nome, count(liv)
from leitor left join req on cod=lei
group by cod, nome;
```

- junção externa
 - **left** – todas as linhas da esquerda
 - SQLite ○ **right** – todas as linhas da direita
 - SQLite ○ **full** – todas as linhas de ambas
- Pode ser **natural left** ...

COD	NOME	COUNT(LIV)
1	Antonio	2
2	Chico	2
3	Marina	1
4	Zeca	1
5	Manuel	2
6	Mafalda	2
7	Rui	0

Vistas

62

34 Crie uma vista para os livros requisitados indicando o título do livro, o nome do leitor e a duração da requisição.

```
create view requisitado as  
select titulo, 'Sr. ' || nome, date('now')-datae  
from req, livro, leitor  
where liv=nr and lei=cod and datae is null;  
select * from requisitado;
```

- só se podem alterar as vistas que assentem numa única tabela base
- o operador || concatena cadeias de caracteres

Vista e junção externa

63

35 Crie uma vista com o código e o nome do leitor e o número de livros que já requisitou.

```
create view estatistica as  
select cod, nome, count(distinct liv)  
from leitor left join req  
on lei = cod  
group by cod, nome;
```

Tratamento de valores nulos

64

36 Calcule as durações de todas as requisições, contabilizando até à data actual os não entregues.

- Utilizar a função
`coalesce(col, valorSeNulo)`
 - Devolve o valor *col* se não for nulo ou o *valorSeNulo* caso *col* seja nulo

```
select liv, lei,  
coalesce(julianday(datae),julianday(date('now')))  
-julianday(datar) duracao  
from req;
```

- A função `coalesce` existe em quase todos os SGBDs.

LIV	LEI	DURACAO
100	1	36
110	2	59
120	2	10
100	3	10
130	6	5974,0265625
140	5	17
100	1	8
110	4	5
150	6	8

CASE

65

37 Substitua Lisboa por Alfacinha, Porto por Tripeiro, e Gaia por Marroquino, deixando os outros casos com Ignoto.

```
SELECT nome,  
       CASE cidade  
         WHEN 'Lisboa' THEN 'Alfacinha'  
         WHEN 'Porto' THEN 'Tripeiro'  
         WHEN 'Gaia' THEN 'Marroquino'  
         ELSE 'Ignoto'  
       END AS origem  
FROM leitor;
```

Nome	Origem
Antonio	Alfacinha
Chico	Tripeiro
Marina	Alfacinha
Zeca	Tripeiro
Manuel	Marroquino
Mafalda	Ignoto
Rui	Alfacinha

Uma pseudo-colunas

66

38 Crie uma vista sobre os livros em que, para além das colunas respectivas se mostre também o número de linha do resultado.

```
select rowid, livro.*  
from livro;
```

- **Rowid** – endereço interno da linha na BD
 - Permite acessos muito rápidos mas é afectado por operações de exportação/importação, pelo que não pode ser usado de forma geral

Top-ten

67

39 Obtenha a lista dos três livros mais caros.

```
SELECT titulo, preco  
FROM livro  
ORDER BY preco DESC LIMIT 3;
```

Titulo	Preço
Os Maias	1100
A Capital	1050
A Relíquia	900

Álgebra relacional vs. LMD:SQL

68

Vamos considerar as tabelas:

Aluno(idAluno, nomeAluno, idade, idCurso->Curso)

AlunoExt(idAluno, nomeAluno, idade, idCurso->Curso)

Curso(idCurso, nomeCurso)

Professor(idProfessor, idade)

Operadores	Álgebra relacional	LMD:SQL
União	$\text{Aluno} \cup \text{AlunoExt}$	SELECT * FROM aluno UNION SELECT * FROM AlunoExt;
Intersecção	$\text{Aluno} \cap \text{AlunoExt}$	SELECT * FROM aluno INTERSECT SELECT * FROM AlunoExt;
Diferença	$\text{Aluno} - \text{AlunoExt}$	SELECT * FROM aluno EXCEPT SELECT * FROM AlunoExt;
Produto cartesiano	$\text{Aluno} \times \text{Curso}$	SELECT * FROM aluno, Curso;
Projecção	$\pi_{\text{idAluno}, \text{nomeAluno}}(\text{Aluno})$	SELECT idAluno, nomeAluno FROM aluno;
Seleccção	$\sigma_{\text{idAluno} < 5}(\text{Aluno})$	SELECT * FROM ALUNO WHERE idAluno < 5;

Álgebra relacional vs. LMD:SQL

69

Vamos considerar as tabelas:

Aluno(idAluno, nomeAluno, idade, idCurso->Curso)

AlunoExt(idAluno, nomeAluno, idade, idCurso->Curso)

Curso(idCurso, nomeCurso)

Professor(idProfessor, idade)

Operadores	Álgebra relacional	LMD:SQL
Divisão	Ver questão 24	Ver questão 24
θ -junção	Aluno \bowtie Professor Aluno.idade > Professor.idade	SELECT * FROM Aluno, Professor WHERE Aluno.idade > Professor.idade;
Junção natural	Aluno \bowtie Curso	SELECT * FROM Aluno NATURAL JOIN Curso;
Junção externa	Aluno \bowtie^+ Curso	SELECT Aluno.*, Curso.* FROM Aluno LEFT JOIN Curso ON Aluno.idCurso=Curso.idCurso UNION ALL SELECT Aluno.*, Curso.* FROM Curso LEFT JOIN Aluno ON Curso.idCurso = Aluno.idCurso WHERE Aluno.idCurso IS NULL;
Semi-junção	Curso \bowtie Aluno	SELECT * FROM Curso WHERE idCurso IN (SELECT idCurso From Aluno);