

Banco de Dados

Sumário

- SQL - Structured Query Language
 - Definição de dados
 - Interrogação
 - Manipulação de dados

Origem

- Introduzida em 1976 como 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.

Como criar os bancos

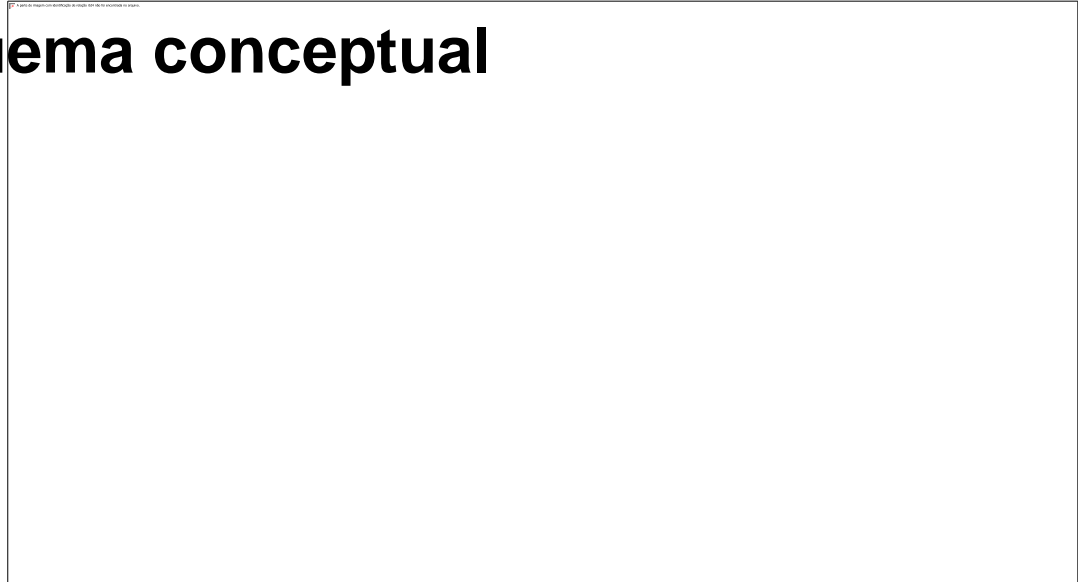


Acesso à BD



BD Biblioteca

□ esquema conceptual



- Cada leitor só pode requisitar cada livro uma vez por dia.

□ esquema relacional

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

Criação de uma tabela

- **create table** tabela

```
(  coluna      tipo      restrição_de_coluna,  
   coluna      tipo      restrição_de_coluna,  
   ...,  
   restrição_de_tabela,  
   ...  
);
```

- em Access, a criação de tabelas é feita pela interface gráfica

Tipos de dados em Oracle

- **char(n)**
 - cadeia de caracteres de comprimento fixo $1 \leq n \leq 255$
- **varchar2(n)**
 - cadeia de caracteres de comprimento variável, de valor máximo $1 \leq n \leq 4000$
- **number(p,s)**
 - número com precisão $1 \leq p \leq 38$ e escala $-84 \leq s \leq 127$
- **date**
 - data entre 4712-01-01 AC e 4712-12-31 DC
- **raw(n)**
 - dado binário de comprimento n
- **long**
 - cadeia de caracteres de comprimento variável até 2GB
- **long raw**
 - dado binário de comprimento variável até 2GB
- **rowid**
 - endereço de uma linha

Caracteres e números

□ em **varchar2**

- necessário especificar o comprimento máximo **n**
- erro se o comprimento efectivo ultrapassar **n**, mesmo que com um valor inferior a 4000

□ em **number**

- pode usar-se notação científica, ex. 6.34e-13, com expoente entre -130 e +125
- só **number** é equivalente a **float**
- **number(p)** é inteiro com p dígitos
- exemplos:

valor	tipo	memória
• 7456123.89	number	7456123.89
• 7456123.89	number(9)	7456124
• 7456123.89	number(9,2)	7456123.89
• 7456123.89	number(9,1)	7456123.9
• 7456123.89	number(6)	excede precisão
• 7456123.89	number(7,-2)	7456100

Datas

□ em **date**

- este tipo destina-se a memorizar momentos, por exemplo, '12-Aug-1995 20:37:19'
- o formato por omissão é 'DD-MON-YY' mas é possível especificar outros
 - Comando Oracle:
 - ALTER SESSION
 - SET NLS_DATE_FORMAT= 'YYYY-MM-DD HH24:MI:SS'
- cadeias com o formato correcto são automaticamente convertidas para o tipo **date**
 - Para forçar conversões, usar TO_DATE(campo, formato) e TO_CHAR(campo, formato)

Tipos de dados em Access

□ **Text**

- cadeia de caracteres de comprimento variável até 255

□ **Memo**

- cadeia de caracteres de comprimento variável até 64Kbytes
- não indexado
- para documentos longos, Word, Excel, etc usar objectos OLE

□ **Number**

- números inteiros ou reais
- **Autonumber**, número incrementado automaticamente

□ **Date**

- data incluindo a hora

□ **Yes/No**

- Booleano

Restrições de coluna

- Estas restrições só se referem a uma coluna
- **not null** - obrigatório
 - **null** - opcional, i.e., aceita valores nulos
- **unique** - não admite valores repetidos na coluna
- **primary key** - chave primária
 - = **unique** + **not null** [pique a chave em Access]
- **references** tabela(coluna)
 - [definir no quadro “Relationship” em Access]
 - 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

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 em Access]
 - **foreign key**(col, col, ...) **references** tabela(col, col, ...)
- **unique** e **primary key** criam automaticamente índices
 - criação explícita, por exemplo por razões de eficiência
 - [indicar se se pretende indexar uma coluna na sua definição]
 - **create index** nome **on** tabela(col **asc**, col **desc**, ...)
- informação sobre tabelas existentes: **describe tabela**
 - comando do programa **sqlplus**, não da linguagem SQL

Dicionário de dados

- Descreve a estrutura da BD
- Contém vistas
 - *user_** (objectos pertencentes ao utilizador)
 - *all_** (objectos acessíveis ao utilizador)
 - *dba_** (todos os objectos na BD)
- Exemplos
 - *user_objects* - objectos de todos os tipos
 - *user_catalog, cat* - tabelas, vistas, sinónimos (resumida)
 - *user_tables, user_tab_columns* - tabelas e vistas, suas colunas
 - *user_indexes, user_ind_columns* - índices, colunas indexadas
 - *user_views* - definição das vistas
 - *user_source* - código de procedimentos, funções e pacotes
 - *user_constraints, user_cons_columns* - restrições, suas colunas
 - *user_dependencies* - dependências entre objectos

Carregamento das tabelas

- ❑ **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, '95-06-15', null);** equivale a **insert into req(liv, lei, datar) values(130,6,'95-06-15');**
- ❑ **drop tabela;**
 - ❑ elimina a tabela se não houver referências para essa tabela ou se estas especificarem **on delete cascade**

Definição do esquema em SQL

create table livro

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

create table leitor

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

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\$

BIBLIOTECA (2)

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)

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

Primeira pergunta

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**
 - quando não existe a tabela usa-se a pseudo-tabela **dual**, ex.
 - **select 25*363+8 from dual;**

Resposta 1

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\$

Seleccção simples

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)
- as perguntas (*Query*) ficam armazenadas na BD, de onde podem ser reutilizadas.

Resposta 2

COD	NOME
1	António
2	Chico
3	Marina
4	Zeca

Filtro mais elaborado

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;
```

NOME	CIDADE
Marina	Lisboa
Manuel	Gaia

```
select nome, cidade  
from leitor  
where nome like 'M%'  
and cod >= 2  
and cod <=5;
```

Pesquisa com cadeias

□ Comparação com uma cadeia usando like:

- % vale por qualquer sequência de 0 ou mais caracteres:

nome like 'M%' (Oracle)

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

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

```
select distinct cidade  
from leitor  
where cpost > 2000;
```

☐ forçar valores distintos tem
como efeito lateral a ordenação

CIDADE
Gaia
Matosinhos
Porto

Filtro complexo

5 Seleccionar os livros do Eça com preço superior a 1000\$00 e todos os livros de preço inferior a 750\$00 indicando o autor, o título, o preço e o número.

```
select autor, titulo, preco, nr  
from livro  
where autor like “%Eca%” and preco > 1000 or preco < 750;
```

AUTOR	TITULO	PREÇO	NR
Eça de Queiroz	Os Maias	1100\$	100
Luís de Camões	Os Lusíadas	490\$	110
Ferreira de Castro	A Selva	700\$	120
Eça de Queiroz	A Capital	1050\$	130

Expressões aritméticas

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

```
select liv, lei, datae - datar "Duracao"  
from req  
where (datae - datar) <= 10;
```

- para renomear uma coluna, indica-se o novo nome a seguir à especificação da mesma, entre aspas
- parâmetros: podia-se incluir na pergunta uma variável a preencher em tempo de execução

```
select liv, lei, datae - datar "Duracao"  
from req  
where (datae - datar) <= [intervalo];
```

Expressões lógicas

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
 - + , - (unário)
 - * , /
 - + , - (binário); || (concatenação)
- operadores de comparação
- operadores lógicos
 - not
 - and
 - or

Operadores de comparação

=, <>, <, >, <=, >=	igual, diferente, menor, maior, menor ou igual, maior ou igual
[not] in	pertença a conjunto
[not] between x and y	x <= valor <= 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

- **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 **is null**.

Ordenação da saída

7 Obtenha uma lista com os autores, livros e preço 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	490\$
Ferreira de Castro	Terra Fria	850\$
Ferreira de Castro	A Selva	700\$
Eça de Queiroz	Os Maias	1100\$
Eça de Queiroz	A Capital	1050\$
Eça de Queiroz	A Relíquia	900\$

Funções de agregação

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(preço), sum(preço), count(*),  
max(preço), min(preço)  
from livro;
```

avg(preço)	sum(preço)	count(*)	max(preço)	min(preço)
848.33	5090.00	6	1100.00	490.00

Agrupamento de linhas

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	1016.66
}	Luís de Camões	490.00
}	Ferreira de Castro	775.00

Agrupamento de linhas com filtro

9' Calcule o preço médio dos livros de cada autor, mas só para médias inferiores a 500\$.

```
select autor, avg(preco)  
from livro  
where avg(preco) < 500  
group by autor;  
errado!
```

```
select autor, avg(preco)  
from livro  
group by autor  
having avg(preco) < 500;
```

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

AUTOR	AVG(PREÇO)
Luís de Camões	490.00

Perguntas encaixadas

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

Pergunta (1ª tentativa, a mais lógica ...):

select titulo, max(preco)

from livro

where autor like 'E%';

TITULO	PREÇO
Os Maias	1100\$
A Capital	1100\$
A Relíquia	1100\$

?

Subpergunta

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	1100\$

Exagerando...

11 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

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

Análise

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

12 Escreva os títulos e datas de requisição dos livros requisitados depois de 95-01-01.

```
select titulo, datar  
from livro, req  
where datar >= '95-01-01' and nr = liv ;
```

TITULO	DATAR
Os Maias	95-01-01
Os Lusíadas	95-01-05
A Selva	95-02-15
Os Maias	95-03-10
A Capital	95-06-15
Terra Fria	95-04-15
Os Maias	95-04-30
Os Lusíadas	95-04-21
A Relíquia	95-06-30
A Capital	95-07-04

Núcleo da álgebra relacional

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

select campo₁, ..., campo_n
from tabela₁, ..., 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

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

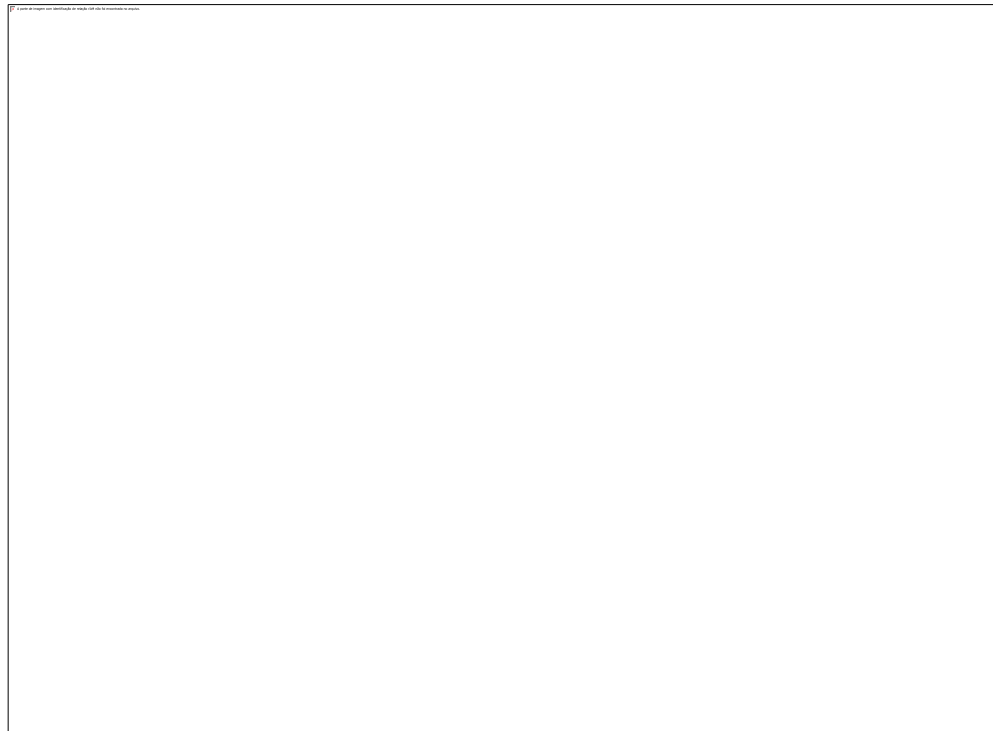
```
select titulo, nome  
from livro, req, leitor  
where nr = liv and lei = cod  
and cpost in (1000, 4000, 4470);
```

- pergunta 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



Condições sobre tuplos

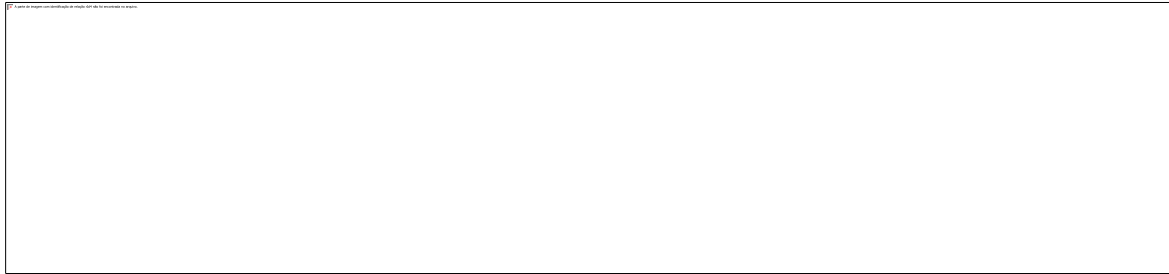
14 Quantos Antónios moram em Lisboa e quantos Zecas moram no Porto?

```
select nome, cidade, count(*)                (Oracle)  
from leitor  
where (nome, cidade) in  
      ((('Antonio','Lisboa'),('Zeca', 'Porto'))  
group by nome, cidade;
```

□ pergunta 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



Agregação de agregação

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

```
select titulo, avg(datae - datar)
from livro, req
where nr = liv and titulo ^= 'Terra Fria'
group by titulo
having avg(datae - datar) = (
    select max(avg(datae - datar))
    from req, livro
    where titulo ^= 'Terra Fria' and nr = liv
    group by titulo);
```

- só faz sentido ter até dois níveis de operadores de agregação

Autojunção

16 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** são aliás para a mesma tabela
- Ex: **p.cidade = q.cidade** faz a junção das duas tabelas **p** e **q** sobre o atributo **cidade**.

Subtracção de conjuntos

17 Obtenha os leitores que não requisitaram o livro 150.

```
select nome  
from leitor  
where cod not in  
  (select lei  
   from req  
   where liv = 150);
```

□ Pergunta alternativa:

```
(select cod  
from leitor)  
minus  
(select lei  
from req  
where liv = 150);
```


Reunião e intersecção

18 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

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

select titulo ¬Access

from livro

where preco > all

(select preco

from livro

where autor =

'Ferreira de Castro');

select titulo

from livro

where preco >

(select max(preco)

from livro

where autor =

'Ferreira de Castro');

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

Operador some (any).

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

select titulo

from livro

where preco < some -- any

(select preco

from livro

where autor like 'Eça%');

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

21 Quais os livros requisitados depois de 95-06-20?

```
select titulo  
from livro  
where nr in  
    (select liv  
        from req  
        where datar >  
            '95-06-20') ;
```

Perguntas alternativas:

```
select titulo  
from livro  
where exists -- where 0 <  
    (select * -- (select count(*)  
        from req  
        where livro.nr = liv  
        and datar > '95-06-20') ;
```

Sub-pergunta variável

- 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; esta referência tem que ter prefixo **livro.nr**.

Contagem

22 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 nr, titulo  
from livro, req r1, req r2  
where nr = r1.liv and nr = r2.liv  
and r1.datar != r2.datar;
```

Pergunta 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);
```

Pertença de tuplos

23 Ache o número, título e preço das obras que têm mais do que um exemplar na biblioteca, com preço inferior a 900\$00.

```
select nr, titulo, preco  
from livro l  
where (titulo, autor) in  
  (select titulo, autor  
    from livro  
    where nr != l.nr)  
and preco < 900.00;
```

- substituindo a subpergunta por uma junção e mais restrições:

```
select l1.nr, l1.titulo, l1.preco  
from livro l1, livro l2  
where l1.preco < 900.00 and  
l1.autor = l2.autor and  
l1.titulo = l2.titulo and  
l1.nr != l2.nr;
```

- esta formulação é menos clara

Sub-pergunta variável

- estratégia de *divisão e conquista*
- uma formulação alternativa mais clara:

```
select nr, titulo, preco  
from livro l  
where preco < 900.00 and 1 <  
  (select count(*)  
   from livro  
   where titulo = l.titulo and autor = l.autor) ;
```


Quantificação universal

24 Quais os leitores que leram todos os livros?

```
select nome          -- R
from leitor          ¬Access
where cod not in
  (select cod        --  $\Pi_{lei}(T-S)$ 
   from livro, leitor -- T
   where not ((nr, cod) in
    (select liv, lei -- S
     from req)));
```

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

Expressões de cálculo

- $\{ \text{nome} \mid \forall \text{nr}, \neg \text{LIVRO}(\text{nr},,,) \vee$
 $\exists \text{cod}: \text{REQ}(\text{nr},\text{cod},,,) \wedge \text{LEITOR}(\text{cod},\text{nome},,,) \}$
 \Leftrightarrow
- $\{ \text{nome} \mid \neg \exists \text{nr} : \text{LIVRO}(\text{nr},,,) \wedge$
 $\neg \exists \text{cod} (\text{REQ}(\text{nr},\text{cod},,,) \wedge \text{LEITOR}(\text{cod},\text{nome},,,)) \}$
- Leis de De Morgan
 - $\neg(A \wedge B) = \neg A \vee \neg B$
 - $\neg(A \vee B) = \neg A \wedge \neg B$
 - $\forall x, P(x) = \neg \exists x: \neg P(x)$
 - $\exists x: P(x) = \neg \forall x, \neg P(x)$

Formulações alternativas

```
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

```
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

25 Faça uma requisição dos livros 100 e 120 pelo leitor 4 em 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

26 Insira, na tabela dos perdidos, os livros requisitados há mais de 300 dias.

create table perdidos

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

insert into perdidos

**(select * from livros where nr in
(select liv from req
where sysdate - datar > 300
and datae is null);**

□ a tabela **perdidos** tem
que já ter sido criada

□ existe a forma **create
table perdidos as
(select ...**

□ **sysdate** é uma função que
devolve a data do dia

Apagar

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

```
delete livro  
where nr in  
(select liv  
from req  
where sysdate - datar > 300  
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

28 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, inclusivé o resultado de uma pergunta, se retornar apenas um valor

Vistas

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

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

- se não se indicarem nomes para as colunas da vista ficam os das expressões do select
- só se podem alterar as vistas que assentem numa única tabela base
- o operador || concatena cadeias de caracteres

Junção externa

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

```
create view estatistica(cod,nome,total) as  
(select cod, nome, count(distinct liv)  
from req, leitor  
where lei (+) = cod  
group by cod, nome);
```

- o símbolo (+) indica que a junção implícita no = é externa na tabela de leitor; assim, mesmo o leitor 7, que não fez nenhuma requisição, aparece na vista.

Língua natural

31 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

- 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 língua natural é muito traiçoeira.

Vistas implícitas

32 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

33 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 **outer**, **inner**, **natural** (no caso do Access em vez de **outer**, usar **left** e **right**).

Outras instruções úteis

34 Coloque comentários na tabela e nas colunas de requisições.

comment on table req is 'Uma requisição é só de um livro por um autor'

comment on column req.datae is 'Valor nulo significa livro ainda não devolvido'

- Documentação Oracle em
 - <http://tahiti.oracle.com>

Restrições de integridade

35 Desactive a restrição de integridade referencial dos códigos de leitor, para poder colocar um código de um leitor não registado.

insert into req values (120, 10, '99-12-01', '99-12-25');

**ORA-02291: integrity constraint (GTD.SYS_C006905)
violated - parent key not found**

alter table req disable constraint SYS_C006905;

alter table req enable constraint SYS_C006905;

- Se introduzir um registo que viole uma restrição, depois não é possível reactivá-la.

Sequências

36 Crie uma sequência para gerar automaticamente números de livro.

```
create sequence num_livro start with 200 increment by 10;  
insert into livro values( num_livro.nextval, 'Memorial do  
convento', 'José Saramago', 2000);  
select num_livro.currval from dual;
```

- ❑ Sequência é um contador autónomo que pode ser usado para gerar chaves primárias
- ❑ <seq>.currval dá o valor corrente
- ❑ <seq>.nextval dá o valor seguinte
- ❑ Garante-se que dois pedidos nextval concorrentes dão valores diferentes

Tratamento de valores nulos

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

- Utilizar a função

`NVL(col, valorSeNulo)`

- Devolve o valor *col* se não for nulo ou o *valorSeNulo* caso *col* seja nulo

`select liv, lei,`

`nvl(datae,sysdate)-datae duracao`

`from req`

- Existe uma `NVL2(T, S, N)` que, se o teste for positivo dá S, se não dá N

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

Tabela de conversão de valores

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

```
select nome,  
       decode(cidade, 'Lisboa', 'Alfacinha',  
              'Porto', 'Tripeiro',  
              'Gaia', 'Marroquino',  
              'Ignoto') origem  
from leitor
```

- Relativamente a cada valor da coluna Cidade, se coincidir com o 2º valor, mostra o 3º, se coincidir com o 4º, mostra o 5º, se não coincidir com nenhum, dá o último

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

Conversão de tipos de dados

39 Houve um bug do ano 2000 e as datas de requisição ficaram todas no século XXI. Reponha as datas a começar em 1999 .

update req

set datar =

to_date('19' || to_char(datar, 'YY-MM-DD'), 'YYYY-MM-DD')

- *To_char* converte números e datas para cadeias de caracteres
 - É possível indicar o formato dessa conversão como 2º argumento
- *To_date* e *To_number* fazem a operação inversa

Duas pseudo-colunas

40 Crie uma vista sobre os livros em que, para além das colunas respectivas se mostre também o número de linha do resultado e o endereço interno para pesquisas rápidas.

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

- ❑ **Rownum** – número de linha da tabela resultante da consulta, antes de eventual ordenação
 - ❑ Serve para limitar às primeiras n linhas um resultado muito extenso
- ❑ **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

Resultado

rownum	rowid	nr	titulo	autor	preço
1	AAA1q8AAHA AA31CAAA	100	Os Maias	Eca de Queiroz	1100
2	AAA1q8AAHA AA31CAAB	110	Os Lusíadas	Luis de Camoes	490
3	AAA1q8AAHA AA31CAAC	120	A Selva	Ferreira de Castro	700
4	AAA1q8AAHA AA31CAAD	130	A Capital	Eca de Queiroz	1050
5	AAA1q8AAHA AA31CAAE	140	Terra Fria	Ferreira de Castro	850
6	AAA1q8AAHA AA31CAAF	150	A Reliquia	Eca de Queiroz	900

Top-ten

41 Obtenha a lista dos três livros mais caros.

- Primeira solução

```
select titulo, preco  
from livro  
where rownum < 4  
order by preco desc
```

Titulo	Preço
Os Maias	1100
A Selva	700
Os Lusíadas	490

- Formulação correcta

```
select titulo, preco  
from (  
    select titulo, preco  
    from livro  
    order by preco desc)  
where rownum < 4
```

Titulo	Preço
Os Maias	1100
A Capital	1050
A Relíquia	900

Dados hierárquicos

42 Mostre os registos ordenados pela hierarquia.

Distrito(codigo, nome)

Concelho(codigo, nome, distrito) \Rightarrow Divisoies(codigo, nome, pai)

Freguesia(codigo, nome, concelho)

```
select codigo, nome from divisoies  
start with pai is null  
connect by prior codigo=pai
```

- Devolve as linhas ordenadas pela árvore implícita nos dados