Universidade do Minho

Ano Letivo: 2022/23
Turnos: PL1/PL8

Bases de Dados

PL08 – Exploração de Bases de Dados

Docente: Diana Ferreira
Email: diana.ferreira@algoritmi.uminho.pt

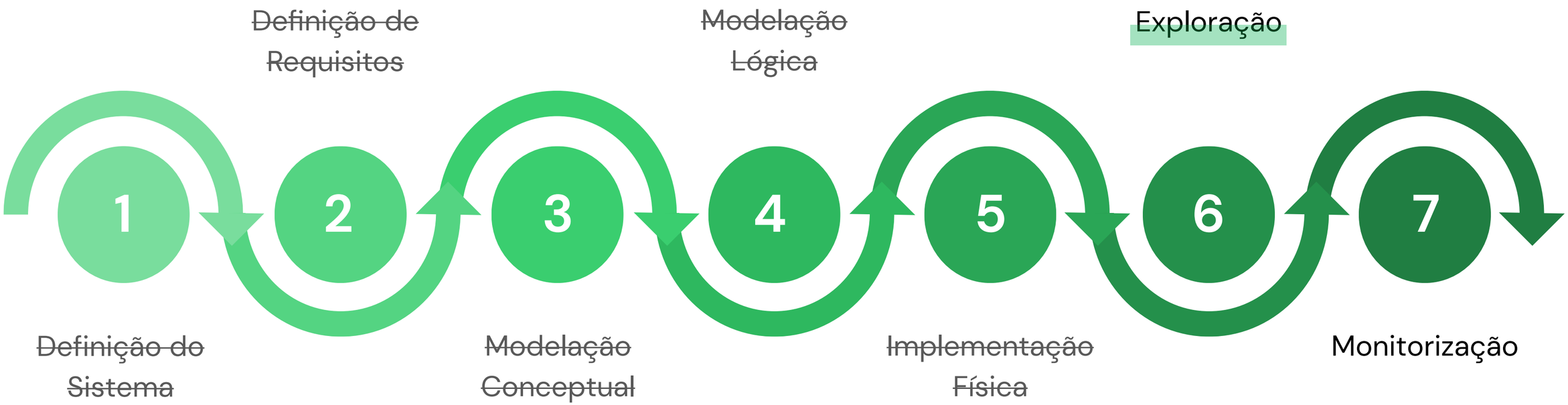
Horário de Atendimento:
4ª feira 18h-19h

Sumário

Bibliografia:

- Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management , Addison-Wesley, 4a Edição, 2004. **(Chapter 6 e 7)**
- Belo, O., "Bases de Dados Relacionais: Implementação com MySQL", FCA – Editora de Informática, 376p, Set 2021. ISBN: 978-972-722-921-5. **(Capítulo 4 e 5)**

Ciclo de vida de um SBD



FASE 6: Exploração

➡ Data Manipulation Language (DML)

Existem 4 instruções básicas para a manipulação de dados:

- **INSERT** → para inserir dados na BD;

```
INSERT INTO <nome_tabela> (<c1>,<c2>,...) VALUES (<v1>,<v2>,...);
INSERT INTO <nome_tabela> (<c1>,<c2>,...)
VALUES
  (<v11>,<v12>,...),
  ...
  (<vnn>,<vn2>,...);
```

- **DELETE** → para remover dados da BD;

```
DELETE FROM <nome_tabela> WHERE <condição>;
```

- **SELECT** → para consultar dados da BD;

```
SELECT [DISTINCT] {* | <nome_c1>, ...}
FROM <nome_tabela>,...
[WHERE <condição>]
[ORDER BY <c1> [ASC | DESC], ...];
```

- **UPDATE** → para atualizar dados da BD;

```
UPDATE <nome_tabela>
SET
   <c1> = <v1>,
   <c2> = <v2>,
   ...
[WHERE <condição>];
```

FASE 6: Exploração

➡ Data Manipulation Language (DML)

OPERADORES:

AND → para condições conjuntas.

OR → para condições disjuntas.

NOT → para negação de condições;

IS NULL → para verificar o preenchimento de uma coluna;

IN → para determinar se um valor especificado corresponde a qualquer valor de uma lista de valores.

BETWEEN → para determinar se um valor está contido num intervalo de valores.

LIKE → para consultar dados com base num padrão especificado (% → qualquer sequência de zero ou mais caracteres; _ → caracter único).

LIMIT → para limitar o número de instâncias retornadas.

FASE 6: Exploração

Data Manipulation Language (DML)

OPERADORES:

=	→	igual
<>	→	diferente
!=	→	diferente
<	→	inferior a
<=	→	igual ou inferior a
>	→	superior a
>=	→	igual ou superior a

FASE 6: Exploração

Funções de Strings

CONCAT - Concatena duas ou mais strings numa só;

INSTR - Retorna a posição da primeira ocorrência de uma substring numa string;

LENGTH – Devolve o comprimento de uma string em bytes e em caracteres;

LEFT - Retorna um número específico de caracteres mais à esquerda de uma string;

LOWER – Converte uma string para minúsculas;

LTRIM - Recebe um argumento de string e retorna uma nova string com todos os caracteres de espaço à esquerda removidos;

REPLACE – Procura e substitui o valor de uma substring numa string;

RIGHT - Devolve um número específico de caracteres mais à direita de uma string;

RTRIM - Recebe um argumento de string e retorna uma nova string com todos os caracteres de espaço à direita removidos;

FASE 6: Exploração

Funções de Strings

SUBSTRING - Extrai uma substring de uma string começando de uma posição com um comprimento especificado;

SUBSTRING_INDEX - Retorna uma substring de uma string antes de um número especificado de ocorrências de um delimitador;

TRIM - Remove caracteres indesejados de uma string;

FIND_IN_SET - Encontra uma string dentro de uma lista de strings separadas por vírgulas;

FORMAT(N, D, locale) - Formata um número N para D casas decimais. O locale é um argumento opcional que determina os separadores de milhar e o agrupamento entre os separadores.

UPPER – Converte uma string para maiúsculas.

FASE 6: Exploração

Funções de Datas

CURDATE() – Retorna a data atual;

NOW()/ SYSDATE() – Retorna a data e hora atuais;

DAY(d) – Obtém o dia do mês de um DATE/DATETIME;

DAYNAME(d) – Obtém o nome do dia da semana de um DATE/DATETIME;

DAYOFWEEK(d) – Obtém o índice do dia da semana de um DATE/DATETIME;
1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday.

LAST_DAY(d) – Retorna o último dia do mês de um DATE/DATETIME;

MONTH(d) – Retorna um inteiro que representa o mês de um DATE/DATETIME;

WEEK(d) – Retorna a semana de um DATE/DATETIME desde o ínicio do ano;

WEEKDAY(d) – Retorna um índice de dia da semana para um DATE/DATETIME;
0 = Monday, 1 = Tuesday, 2 = Wednesday, 3 = Thursday, 4 = Friday, 5 = Saturday, 6 = Sunday.

YEAR(d) – Retorna o ano de um DATE/DATETIME;

FASE 6: Exploração

Funções de Datas

HOUR(d/t) – Retorna a hora de um DATETIME/TIME;

MINUTE(d/t) – Retorna os minutos de um DATETIME/TIME;

SECOND(d/t) – Retorna os segundos de um DATETIME/TIME;

DATEDIFF(d2,d1) - Calcula o número de dias entre dois valores DATE/DATETIME;

DATE_ADD(d, INTERVAl value unit*) - Adiciona um valor de tempo a um valor DATE/DATETIME;

DATE_SUB(d, INTERVAL value unit*) – Subtrai um valor de tempo a um valor DATE/DATETIME;

DATE_FORMAT(d, format) - Formata um valor de data com base em um formato de data especificado;

EXTRACT(unit* FROM d) - Extrai uma parte de um DATE/DATETIME;

STR_TO_DATE(s, format) - Converte uma string num valor de data e hora com base num formato especificado;

TIMEDIFF(unit, t1, t2) - Calcula a diferença entre dois valores DATETIME/TIME;

TIMESTAMPDIFF(unit, d1, d2) - Calcula a diferença entre dois valores DATE/DATETIME.

* unit: DAY, HOUR, MONTH, YEAR, WEEK, SECOND, etc.

FASE 6: Exploração

Formatos de Datas

%W	Full name of weekday e.g., Sunday, Monday,..., Saturday
%a	Three-characters abbreviated weekday name e.g., Mon, Tue, Wed, etc.
%w	Weekday in number (0=Sunday, 1= Monday,etc.)
%D	Day of the month with English suffix e.g., 0th, 1st, 2nd, etc.
%d	Day of the month with leading zero if it is 1 number e.g., 00, 01,02, ...31
%e	Day of the month without leading zero e.g., 1,2,...31
%M	Full month name e.g., January, February,...December
%b	Three-characters abbreviated month name e.g., Jan, Feb, Mar, etc.
%m	Month with leading zero e.g., 00,01,02,...12
%c	Month in numeric e.g., 1, 2, 3...12
%Y	Four digits year e.g., 2000 and 2001.
%y	Two digits year e.g., 10,11, and 12.

FASE 6: Exploração

Formatos de Datas

%H	Hour in 24-hour format with leading zero e.g., 00..23
%h	Hour in 12-hour format with leading zero e.g., 01, 02...12
%i	Minutes with leading zero e.g., 00, 01,...59
%S/%s	Seconds with leading zero 00,01,...59
%f	Microseconds in the range of 000000..999999
%T	Time in 24-hour format hh:mm:ss
%r	Time in 12-hour format hh:mm:ss AM or PM

FASE 6: Exploração

Funções de Datas

EXEMPLOS:

Qual é a data da última prescrição emitida?

SELECT MAX(DATE(data_prescricao)) FROM prescricoes;

Quantos dias passaram desde que '2022-03-22'?

SELECT DATEDIFF(NOW(), ' 2022-03-22');

Liste as consultas que ocorreram no dia a seguir ao '2020-01-01'.

SELECT * FROM consultas WHERE DATE(hora_ini)=ADDDATE('2020-01-01', INTERVAL 1 DAY);

Liste as prescricoes emitidas à mais de um ano.

SELECT * FROM prescricoes WHERE DATE(data_prescricao)<ADDDATE(CURDATE(), INTERVAL -365 DAY);

FASE 6: Exploração

Funções Numéricas

ABS() Returns the absolute value of a number

CEIL() Returns the smallest integer value greater than or equal to the input number (n).

FLOOR() Returns the largest integer value not greater than the argument

MOD() Returns the remainder of a number divided by another

ROUND() Rounds a number to a specified number of decimal places.

TRUNCATE() Truncates a number to a specified number of decimal places

ACOS(n) Returns the arc cosine of n or null if n is not in the range -1 and 1.

ASIN(n) Returns the arcsine of n which is the value whose sine is n. It returns null if n is not in the range -1 to 1.

ATAN() Returns the arctangent of n.

ATAN2(n,m), ATAN(m,n) Returns the arctangent of the two variables n and m

FASE 6: Exploração

Funções Numéricas

CONV(n,from_base,to_base) Converts a number between different number bases

COS(n) Returns the cosine of n, where n is in radians

COT(n) Returns the cotangent of n.

CRC32() Computes a cyclic redundancy check value and returns a 32-bit unsigned value

DEGREES(n) Converts radians to degrees of the argument n

EXP(n) Raises to the power of e raised to the power of n

LN(n) Returns the natural logarithm of n

LOG(n) Returns the natural logarithm of the first argument

LOG10() Returns the base-10 logarithm of the argument

LOG2() Returns the base-2 logarithm of the argument

FASE 6: Exploração

Funções Numéricas

PI() Returns the value of PI

POW() Returns the argument raised to the specified power

POWER() Returns the argument raised to the specified power

RADIANS() Returns argument converted to radians

RAND() Returns a random floating-point value

SIGN(n) Returns the sign of n that can be -1, 0, or 1 depending on whether n is negative, zero, or positive.

SIN(n) Returns the sine of n

SQRT(n) Returns the square root of n

TAN(n) Returns the tangent of n

FASE 6: Exploração

Expressões Regulares

As expressões regulares diferenciam-se do operador LIKE por possuírem mais metacaracteres permitindo construir padrões mais flexíveis. No entanto, o tempo de consulta pode aumentar caso se usem padrões complexos. Estas expressões usam os operadores **RLIKE** ou **REGEXP**. Alguns metacaracteres comumente usados numa expressão regular:

^	Corresponde à posição no início da string pesquisada;
$	Corresponde à posição no final da string pesquisada;
.	Corresponde a qualquer caracter;
[...]	Corresponde a qualquer caracter específicado dentro dos parêntesis rectos;
[^...]	corresponde a qualquer caractere não especificado dentro dos parêntesis
p1|p2	corresponde a qualquer um dos padrões p1 ou p2
{n}	corresponde a n número de instâncias do caractere anterior
{m,n}	corresponde de m a n número de instâncias do caractere anterior

FASE 6: Exploração

Expressões Regulares

EXEMPLOS:

- **Retorna os fármacos cujo nome comece com a letra ‘o’ ou ‘a’:**

SELECT nome FROM farmacos WHERE nome REGEXP '^[oa]’;

SELECT nome FROM farmacos WHERE nome REGEXP '^o|^a';

- **retorna as especialidades cujo nome não termina com as letras ‘gia’**

SELECT des_especialidade FROM especialidades WHERE des_especialidade NOT REGEXP ‘gia$’;

- **retorna todos os fármacos que contêm os caracteres ‘ar’**

SELECT nome FROM farmacos WHERE nome REGEXP 'ar’;

FASE 6: Exploração

Expressões Regulares

- retorna as especialidades que contêm extamente 10 caracteres:

SELECT des_especialidade FROM especialidades WHERE des_especialidade REGEXP '^.{10}$';

SELECT des_especialidade FROM especialidades WHERE des_especialidade REGEXP '^..........$';

- retorna as especialidades que contêm entre 5 a 10 caracteres:

SELECT des_especialidade FROM especialidades WHERE des_especialidade REGEXP '^.{5,10}$';

- retorna os fármacos que contêm uma letra entre 'a' e 'c', seguidas por qualquer caracter, seguidas pela letra 'a'.

SELECT nome FROM farmacos WHERE nome REGEXP '[a-c].[a]';

- retorna todos os fármacos que comecem com vogal ou terminem em 'ol'

SELECT nome FROM farmacos WHERE nome REGEXP '^[aeiou]|ol$';

FASE 6: Exploração

➡ ORDER BY

A cláusula ORDER BY permite que as linhas sejam apresentadas por ordem ascendente (ASC) ou decrescente (DESC) de qualquer coluna ou combinação de colunas. A cláusula ORDER BY deve ser sempre a última cláusula da instrução SELECT.

EXEMPLOS:

Liste o nome dos administrativos por ordem crescente.

SELECT nome from funcionarios f, administrativos a WHERE f.nr_mec=a.nr_mec ORDER BY nome ASC;

Liste os procedimentos disponíveis no hospital, do maior custo para o menor.

SELECT * FROM procedimentos ORDER BY preco DESC;

FASE 6: Exploração

➡ ORDER BY

Liste as prescrições de maior prazo de validade para o menor.

```
SELECT *, GREATEST(datediff(data_validade, curdate()), 0) as dias_validade from prescricoes ORDER BY dias_validade DESC;
```

ou

```
SELECT *,
        CASE
                WHEN datediff(data_validade, curdate()) < 0 THEN 0
                ELSE datediff(data_validade, curdate())
        END as dias_validade
FROM prescricoes  ORDER BY dias_validade DESC;
```

FASE 6: Exploração

GROUP BY

A cláusula GROUP BY pode ser usada em diferentes contextos:

A) Uso em alternativa ao SELECT DISTINCT(<nome coluna>)

- SELECT DISTINCT localidade FROM pacientes;
- SELECT localidade FROM pacientes GROUP BY localidade;

B) Uso com funções de agregação (AVG, COUNT, SUM, MAX, MIN, etc.)

O valor máximo de consulta cobrado por especialidade.

SELECT e.des_especialidade, MAX(c.custo_final) as preco_max FROM consultas c, especialidades e, medicos m WHERE m.nr_mec = c.nr_mec_medico AND m.cod_especialidade = e.cod_especialidade GROUP BY e.des_especialidade;

FASE 6: Exploração

GROUP BY

A cláusula GROUP BY pode ser usada em diferentes contextos:

C) Uso com outras funções

O valor médio de consulta por ano.

SELECT YEAR(c.dta_ini) as ano, AVG(c.custo_final) as preco_medio FROM consultas c GROUP BY YEAR(c.dta_ini);

D) Uso com a cláusula HAVING

Para filtrar os valores retornados pela cláusula GROUP BY, usa-se uma cláusula HAVING.

O nº total de consultas por ano após 2019.

SELECT YEAR(dta_ini) as ano, COUNT(*) as total_consultas FROM consultas GROUP BY YEAR(dta_ini) HAVING ano > '2019';

FASE 6: Exploração

SUBQUERIES

Uma instrução SELECT pode ser usada dentro de outra instrução SELECT, é chamada de SELECT interna (ou subselect) e tem de estar entre parêntesis curvos. Por sua vez, um subselect pode ser usado dentro de outro subselect.

A) subselect com operadores de comparação

Qual é o procedimento mais caro?

SELECT * FROM procedimentos WHERE preco = (SELECT MAX(preco) FROM procedimentos);

B) subselect com operadores IN e NOT IN

Listar os administrativos que ainda não faturaram uma consulta.

SELECT * FROM administrativos WHERE nr_mec NOT IN (SELECT nr_mec_secretaria FROM consultas WHERE nr_mec_secretaria IS NOT NULL)

FASE 6: Exploração

SUBQUERIES

C) subselect na cláusula FROM - o conjunto de resultados retornado de um subselect é usado como uma tabela temporária.

Listar o número máximo e o número mínimo de medicamentos receitados.

```
SELECT max(total), min(total) FROM (SELECT id_med, COUNT(*) as total FROM prescricoes GROUP BY id_med) AS sub;
```

D) subselect com operadores EXISTS e NOT EXISTS

Listar os médicos que deram consultas.

```
SELECT * FROM medicos m WHERE EXISTS (SELECT * FROM consultas c WHERE c.nr_mec_medico = m.nr_mec);
```

FASE 6: Exploração

SUBQUERIES

E) subselect com operadores ANY e ALL: É usado para efetuar uma comparação entre o valor de uma coluna e uma range de valores. Ambos retornam um valor booleano como resultado. O ANY retorna TRUE se QUALQUER um dos valores da subconsulta atender à condição. O ALL retorna TRUE se TODOS os valores da subconsulta atenderem à condição .

Listar os médicos que deram consultas.

```
SELECT * FROM medicos WHERE nr_mec =  ANY(SELECT nr_mec_medico FROM consultas);
```

Próxima aula: Exploração da BD

- Operações do tipo JOIN
- Operações para combinar resultados (UNION, INTERSECT, EXCEPT)