Universidade do Minho

Ano Letivo: 2023/24
Turno: PL9

Bases de Dados

PL08 – Introdução à SQL

Docente: Cristiana Neto
Email: cristiana.neto@algoritmi.uminho.pt

Horário de Atendimento:
6ª feira 09h-10h

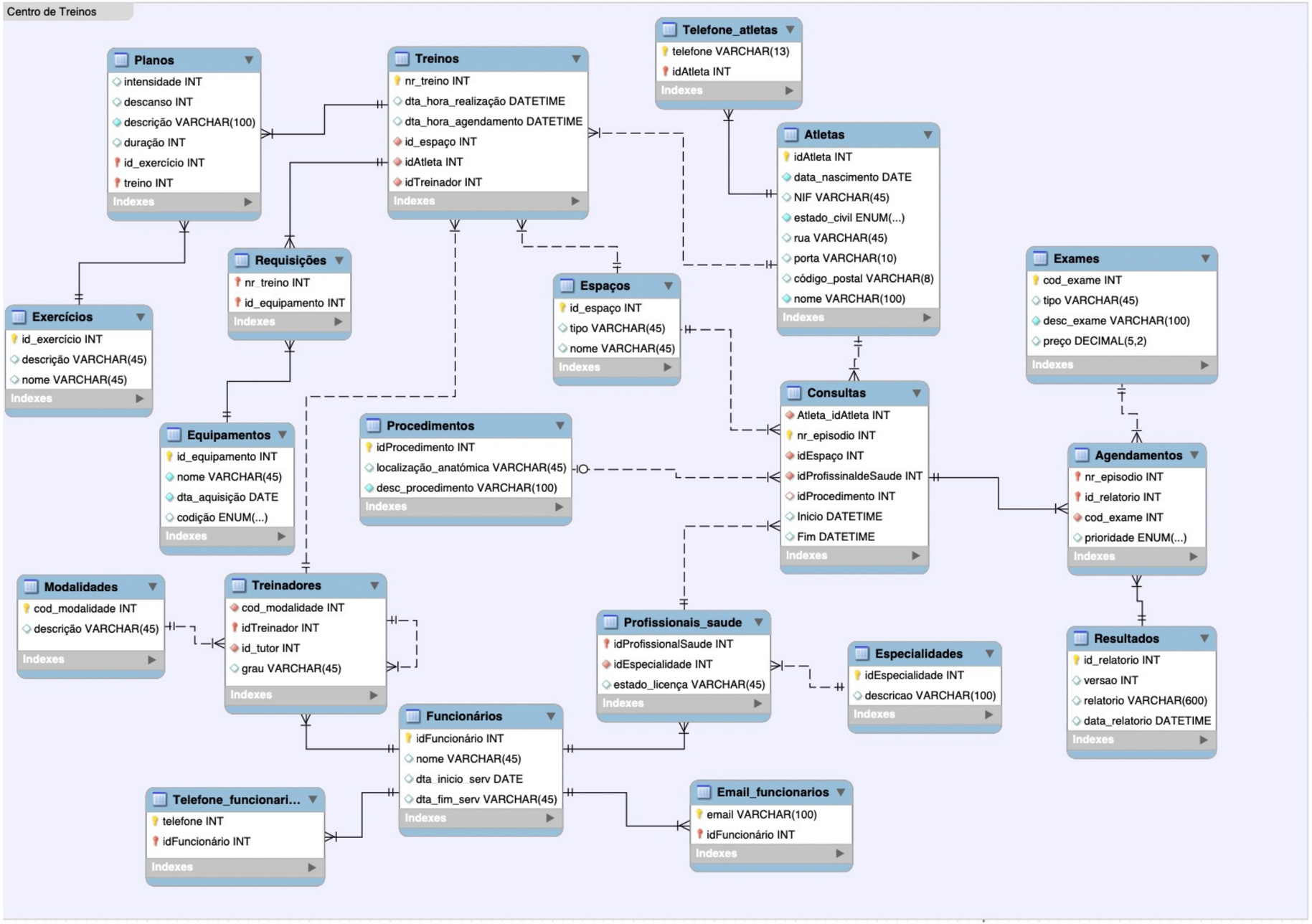
Sumário

Bibliografia:

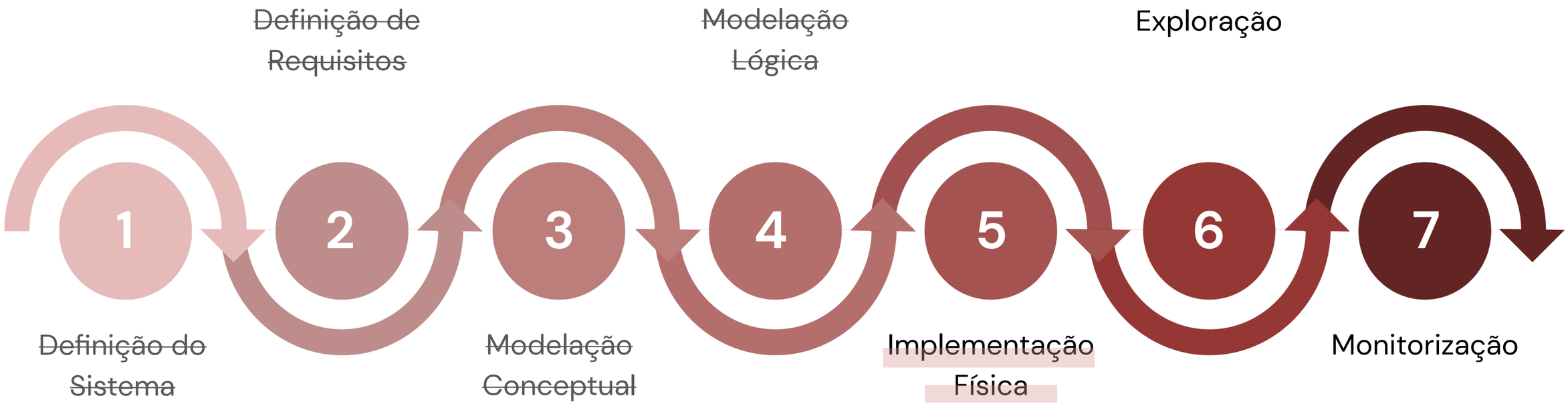
- Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management , Addison-Wesley, 4a Edição, 2004. **(Chapter 18)**
- Belo, O., "Bases de Dados Relacionais: Implementação com MySQL", FCA – Editora de Informática, 376p, Set 2021. ISBN: 978-972-722-921-5. **(Capítulo 2)**

Revisão da aula anterior:

Modelo Relacional



Ciclo de vida de um SBD



FASE 5: Modelação Física

➡ Data Definition Language (DDL)

→ cria uma base de dados física

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] <nome_BD>
[CHARACTER SET charset_name]
[COLLATE collation_name];
```

- Se não incluirmos as cláusulas CHARACTER SET e COLLATE, o MySQL usará os valores default/padrão.

→ listar as base de dados

```
SHOW DATABASES;
```

→ identificação da área de trabalho

```
USE <nome_BD>;
```

FASE 5: Modelação Física

Data Definition Language (DDL)

→ altera a base de dados com o nome especificado

```
ALTER {DATABASE | SCHEMA} <nome_BD>
alter_option: {
    [DEFAULT] CHARACTER SET [=] <charset_name>
  | [DEFAULT] COLLATE [=] <collation_name>
  | [DEFAULT] ENCRYPTION [=] {'Y' | 'N'}
  | READ ONLY [=] {DEFAULT | 0 | 1}
}
```

→ apaga a base de dados com o nome especificado

```
DROP {DATABASE | SCHEMA} [IF EXISTS] <nome_BD>
```

FASE 5: Modelação Física

Data Definition Language (DDL)

→ cria uma tabela com o nome escolhido e com as colunas especificadas

CREATE TABLE [IF NOT EXISTS] <nome_tabela> (

<nome_coluna> <tipo_coluna[tamanho]> [NOT NULL | NULL] [DEFAULT <value>] [AUTO_INCREMENT][UNIQUE],

…,

PRIMARY KEY (<nome_coluna_PK>,…)

[CONSTRAINT <constraint_name>] UNIKE {KEY | INDEX} (<nome_coluna>,…)

[CONSTRAINT <constraint_name>] FOREIGN KEY (<nome_coluna_FK>) REFERENCES <nome_tabela_FK> (<nome_coluna_FK>)

[ON UPDATE <referential_integrity_constraint>] [ON DELETE <referential_integrity_constraint>]

) [ENGINE=<storage_engine>];

Se o ENGINE não for declarado, o MySQL usará o InnoDB por padrão.

referential_integrity_constraint = {NO ACTION | RESTRICT | CASCADE | SET NULL | SET DEFAULT } Se não especificar a cláusula ON DELETE e ON UPDATE, o MySQL usará a definição padrão.

FASE 5: Modelação Física

Data Definition Language (DDL)

```
CREATE TABLE IF NOT EXISTS `Atletas` (
  `idAtleta` INT NOT NULL,
  `data_nascimento` DATE NOT NULL,
  `NIF` VARCHAR(45) NOT NULL UNIQUE,
  `estado_civil` ENUM('Casado', 'Solteiro') NOT NULL DEFAULT 'Solteiro',
  `rua` VARCHAR(45) NULL,
  `porta` VARCHAR(10) NULL,
  `código_postal` VARCHAR(8) NULL,
  `nome` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idAtleta`)
);
```

```
CREATE TABLE IF NOT EXISTS `Atletas` (
  `idAtleta` INT NOT NULL AUTO_INCREMENT,
  `data_nascimento` DATE NOT NULL,
  `NIF` VARCHAR(45) NOT NULL,
  `estado_civil` ENUM('Casado', 'Solteiro') NOT NULL DEFAULT 'Solteiro',
  `rua` VARCHAR(45) NULL,
  `porta` VARCHAR(10) NULL,
  `código_postal` VARCHAR(8) NULL,
  `nome` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idAtleta`),
 UNIQUE KEY (`NIF`),
);
```

FASE 5: Modelação Física

Data Definition Language (DDL)

Para lidarmos com as restrições de domínio no caso do nr_sequencial ter obrigatoriamente 6 dígitos:

A) Definir a coluna com o tipo CHAR(6) e aplicar *check constrainsts* para forçar o armazenamento de apenas 6 dígitos usando uma expressão regular;

```
CREATE TABLE Atletas (
idAtleta  CHAR(6) NOT NULL,
...
CONSTRAINT
      CHECK (idAtleta REGEXP '^[0-9]{6}$'));
```

B) Definir a coluna com o tipo INT e aplicar *check constrainsts* para forçar o armazenamento de exatamente 6 dígitos;

```
CREATE TABLE Atletas (
idAtleta  INT NOT NULL,
...
CONSTRAINT
      CHECK (LENGTH(idAtleta)=6));
```

FASE 5: Modelação Física

Data Definition Language (DDL)

→ apaga a tabela com o nome especificado

DROP TABLE <nome_ tabela> [RESTRICT | CASCADE];

→ altera a tabela com o nome especificado

ALTER TABLE <nome_tabela_antigo> RENAME TO <nome_tabela_novo>; → altera o nome da tabela.

ALTER TABLE <nome_tabela> ADD <nome_campo> <domínio_campo>; → cria um novo atributo na tabela com o nome e domínio especificados. Todos os tuplos existentes ficam com NULL no novo atributo.

ALTER TABLE <nome_tabela> DROP <nome_campo>; → apaga o atributo com o nome especificado da tabela.

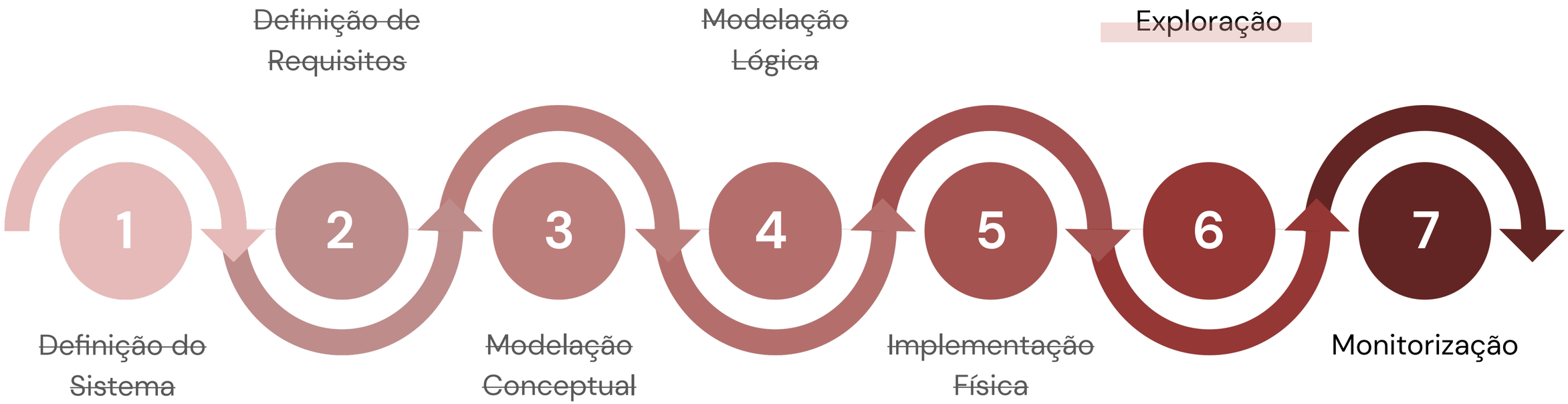
ALTER TABLE <nome_tabela> MODIFY <nome_campo> <domínio_campo>; → modifica o atributo com o nome especificado da tabela.

ALTER TABLE <nome_tabela> ALTER < nome_campo> SET DEFAULT <value>; → modifica uma coluna de uma tabela para lhe atribuir valores padrão.

ALTER TABLE <nome_tabela> ALTER < nome_campo> DROP DEFAULT; → remove os valores padrão de uma coluna de uma tabela.

ALTER TABLE <nome_tabela> ADD CONSTRAINT <nome_constraint> UNIQUE KEY(column_1,column_2,...); → modifica uma tabela para lhe atribuir uma indexação única .

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

Operadores

= → igual

<> → diferente

!= → diferente

< → inferior a

<= → igual ou inferior a

> → superior a

>= → igual ou superior a

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

➡ Projeção e Filtragem

EXEMPLOS:

Quais os funcionários inativos?

SELECT * FROM Funcionários WHERE dta_fim_serv IS NOT NULL;

Liste o nome e id dos espaços no interior e exterior;

SELECT id_espaço,nome FROM Espaços where tipo IN (‘Interior,Exterior’);

Liste os as piscinas interiores.

SELECT * FROM Espaços WHERE tipo = 'Interior' AND nome='Piscina';

FASE 6: Exploração

➡ Funções de Datas

CURDATE – Retorna a data atual;

NOW/ SYSDATE – Retorna a data e hora atuais;

DAY - Obtém o dia do mês de um DATE/DATETIME;

DAYOFWEEK – Obtém o índice do dia da semana de um DATE/DATETIME;

MONTH - Retorna um inteiro que representa o mês de um DATE/DATETIME;

WEEK - Retorna um número de semana de um DATE/DATETIME;

WEEKDAY - Retorna um índice de dia da semana para um DATE/DATETIME;

YEAR - Retorna o ano de um DATE/DATETIME;

HOUR – Retorna a hora de um DATETIME/TIME;

MINUTE – Retorna os minutos de um DATETIME/TIME;

SECOND – Retorna os segundos de um DATETIME/TIME;

FASE 6: Exploração

Funções de Datas

DATEDIFF - Calcula o número de dias entre dois valores DATE/DATETIME;

DATE_ADD - Adiciona um valor de tempo a um valor DATE/DATETIME;

DATE_SUB – Subtrai um valor de tempo a um valor DATE/DATETIME;

DATE_FORMAT – Formata um valor de data com base em um formato de data especificado;

STR_TO_DATE - Converte uma string num valor de data e hora com base num formato especificado;

TIMEDIFF - Calcula a diferença entre dois valores DATETIME/TIME;

TIMESTAMPDIFF – Calcula a diferença entre dois valores DATE/DATETIME.

FASE 6: Exploração

➡ Funções de Datas

EXEMPLOS:

Qual é a data do ultimo despedimento?

SELECT MAX(DATE(dta_fim_serv)) FROM Funcionários;

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(Inicio)=ADDDATE('2024-01-20', INTERVAL 1 DAY);

Liste os functionários que trabalham há mais de 2 anos.

SELECT * FROM Funcionários WHERE DATE(dta_inicio_serv)<ADDDATE(CURDATE(), INTERVAL -365*2 DAY);

FASE 6: Exploração

➡ ORDER BY

EXEMPLOS:

Liste o nome dos atletas por ordem crescente.

SELECT nome from Atletas ORDER BY nome ASC;

Liste os planos de maior duração custo para o menor.

SELECT * FROM Planos ORDER BY duração DESC;

FASE 6: Exploração

➡ GROUP BY

EXEMPLOS:

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

- SELECT DISTINCT código_postal FROM Atletas;
- SELECT código_postal FROM Atletas GROUP BY código_postal;

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

O valor máximo de duração de um exercício.

SELECT max(duração) from Planos GROUP BY id_exercício

O valor médio de duração de um exercício.

SELECT avg(duração) from Planos GROUP BY id_exercício

FASE 6: Exploração

GROUP BY

EXEMPLOS:

D) Uso com a cláusula HAVING

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

O nº total de treinos por ano após 2019.

```
SELECT YEAR(dta_hora_realização) as ano, COUNT(*) as total_treinos
FROM Treinos GROUP BY YEAR(dta_hora_realização) HAVING ano > '2023';
```

Resolução de Exercícios

Ficha de Excercícios PL08

Universidade do Minho

Ano Letivo: 2023/24
Turno: PL9

Bases de Dados

PL08 – Introdução à SQL

Docente: Cristiana Neto
Email: cristiana.neto@algoritmi.uminho.pt

Horário de Atendimento:
6ª feira 09h-10h