Bases de Dados
SQL

Programação e Bases de Dados (CC4042)
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto

Adaptado de slides de
Eduardo R. B. Marques, Fernando Silva e Ricardo Rocha

Introdução ao SQL

SQL

- A **SQL (Structured Query Language)** é uma linguagem declarativa que permite definir e manipular bases de dados.
- É a linguagem standard para SGBDs baseados no modelo relacional.
- A SQL é uma **DDL ("Data Definition Language")** porque permite definir a estrutura da BD — o **esquema**.
- A SQL é uma **DML ("Data Manipulation Language")** porque permite manipular o conteúdo da BD — os **dados** que instanciam um **esquema**.

SQL — uma primeira introdução

- **Criação de tabelas com CREATE TABLE**
  - Definições essenciais de restrições de integridade
  - Tipos de valores associados a atributos de tabelas
- **Manipulação de dados**
  - Formas básicas das instruções **INSERT, SELECT, UPDATE,** e **DELETE**
  - Exemplos de violações de restrições de integridade em SQL
- Usaremos o exemplo da BD p/biblioteca (com ligeiras alterações), e a "shell" do MySQL para uma demonstração interactiva.
- Como suporte a estes "slides", **veja o código SQL para BD biblioteca disponível na página da cadeira.**

Criação de tabelas

Criação de uma tabela em SQL

```
CREATE TABLE <NOME DA TABELA>
(
    <NOME ATRIBUTO 1>   <TIPO 1> <RESTRIÇÕES DE DOMINIO 1> ,
    … ,
    <NOME ATRIBUTO N>   <TIPO N> <RESTRIÇÕES DE DOMINIO N>
    …
    PRIMARY KEY(<ATRIBUTOS QUE FORMAM CHAVE PRIMÁRIA>),
    …
    UNIQUE(<ATRIBUTOS QUE FORMAM UMA CHAVE (SECUNDÁRIA)>),
    …
    FOREIGN KEY(<ATRIBUTOS QUE FORMAM UMA CHAVE EXTERNA>)
    REFERENCES <NOME_DE_OUTRA_TABELA 1>(<CHAVE_PRIMÁRIA>)
    …
);
```

Exemplo da biblioteca — revisitado

UTENTE						
Num	CC	Nome	DataNasc	Sexo	Telefone	Email
1	10583212	João Pinto	1976-12-19	M	913 448 748	azulibranco@fcp.pt
2	12447555	Carlos Semedo	1985-06-02	M	223 774 327	NULL
3	16348500	Maria Silva	2005-11-17	F	939 939 939	NULL
4	11983516	Pedro Costa	1982-01-03	M	384 388 291	pc12345@xpto.com
…	…	…	…	…	…	…

- Variação do exemplo anterior
  - Utente tem um nº de utente por chave primária, em vez do CC.
  - são guardados dados pessoais adicionais como a data de nascimento e o sexo
  - o email é um atributo opcional
- Para definição de atributos de texto iremos convencionar:
  - Nomes de utentes, autores e títulos de livros: representados em até 64 caracteres
  - Email e nome de editora: até 32 caracteres.
  - Códigos de estante e secção até 3 caracteres.

Tabela UTENTE em SQL

```
CREATE TABLE UTENTE
(
  Num INT NOT NULL,
  CC INT NOT NULL,
  Nome VARCHAR(64) NOT NULL,
  DataNasc DATE NOT NULL,
  Sexo ENUM('M', 'F') NOT NULL,
  Telefone INT NOT NULL,
  Email VARCHAR(32),
  PRIMARY KEY(Num),
  UNIQUE(CC)
);
```

- **Chave primária**: Num; Outra chave (secundária): CC
- **Atributos não-opcionais indicados com** NOT NULL (todos excepto EMAIL)

Tabela UTENTE em SQL – Variação

```
CREATE TABLE UTENTE
(
  Num INT PRIMARY KEY,
  CC INT NOT NULL UNIQUE,
  Nome VARCHAR(64) NOT NULL,
  DataNasc DATE NOT NULL,
  Sexo ENUM('M', 'F'),
  Telefone INT NOT NULL,
  Email VARCHAR(32)
);
```

- PRIMARY KEY e UNIQUE junto à definição do atributo.
- NOT NULL pode ser omitida para atributos de chaves primárias, a restrição é implícita. UNIQUE NOT NULL define uma chave. **Caso particular**: UNIQUE apenas permite que o valor NULL seja usado para múltiplos registos , embora valores definidos continuem a ter de ser únicos (ver documentação de CREATE TABLE).

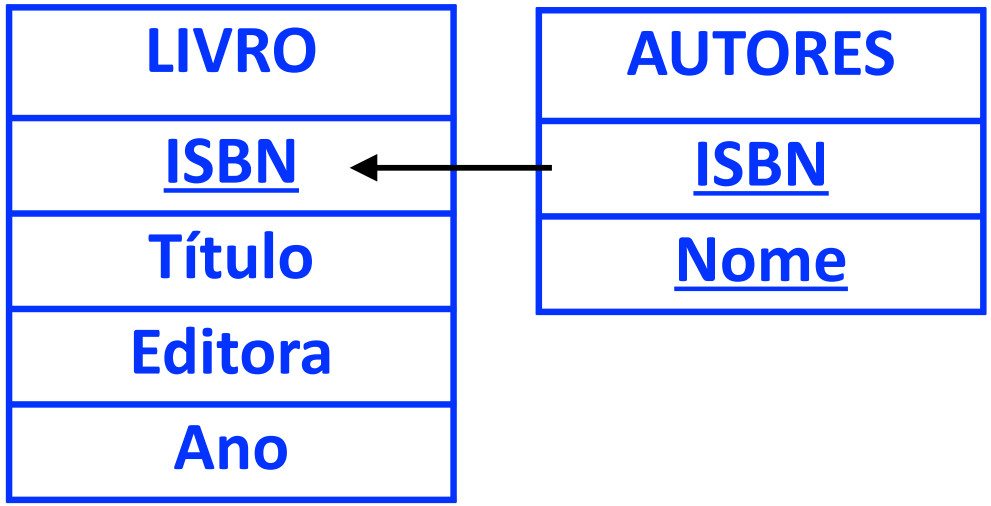
Tipos de dados para atributos (MySQL)

Números Inteiros	TINYINT (1 byte), SMALLINT (2) MEDIUMINT (3), INT (4), BIGINT (8) [INTEGER = INT]
Números de vírgula flutuante	DECIMAL, NUMERIC, FLOAT, DOUBLE
Sequências de caracteres	CHAR, VARCHAR, ENUM, …
Tempo	DATE, DATETIME, TIME, TIMESTAMP, YEAR

- Referência: MySQL 8.0 Reference Manual - Data Types
- Vamos olhar a referência "online" para tipos inteiros, VARCHAR, DATE, e ENUM rapidamente.

Tabela para livros e autores

```
CREATE TABLE LIVRO (
  ISBN BIGINT NOT NULL,
  Título VARCHAR(64) NOT NULL,
  Editora VARCHAR(32) NOT NULL,
  Ano YEAR NOT NULL,
  PRIMARY KEY(ISBN) );
```




```
CREATE TABLE AUTORES (
  ISBN BIGINT NOT NULL,
  Nome VARCHAR(64) NOT NULL,
  PRIMARY KEY(ISBN, Nome),
  FOREIGN KEY(ISBN) REFERENCES LIVRO(ISBN) );
```

- **AUTORES** : chave primária formada por 2 atributos, um deles chave externa para **LIVRO.ISBN**.

Tabelas para secção, estante e prateleira

```
CREATE TABLE SECÇÃO
( Código VARCHAR(3) NOT NULL,
  Descrição VARCHAR(32) NOT NULL,
  PRIMARY KEY(Código), UNIQUE(Descrição) );

CREATE TABLE ESTANTE (
  Código VARCHAR(3) NOT NULL,
  Secção VARCHAR(3) NOT NULL,
  PRIMARY KEY(Código),
  FOREIGN KEY(Secção) REFERENCES SECÇÃO(Código) );

CREATE TABLE PRATELEIRA (
  Estante VARCHAR(3) NOT NULL,
  Num TINYINT NOT NULL,
  PRIMARY KEY(Estante, Num),
  FOREIGN KEY(Estante) REFERENCES ESTANTE(Código)
);
```

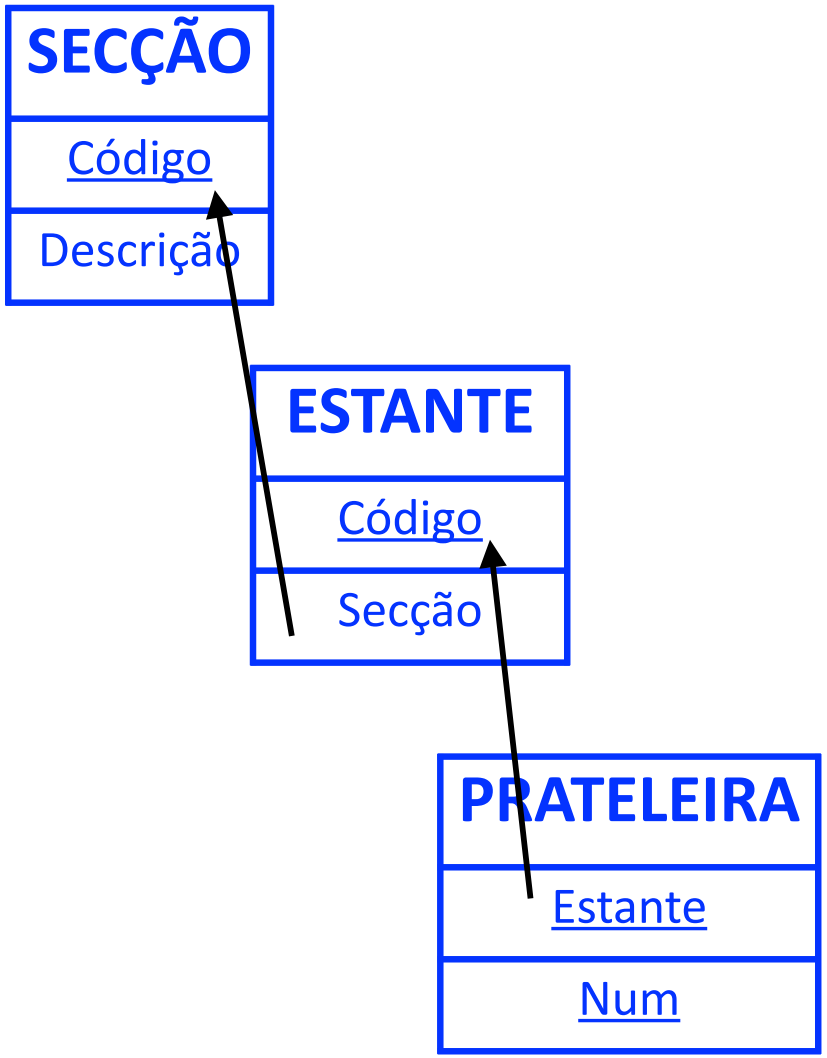
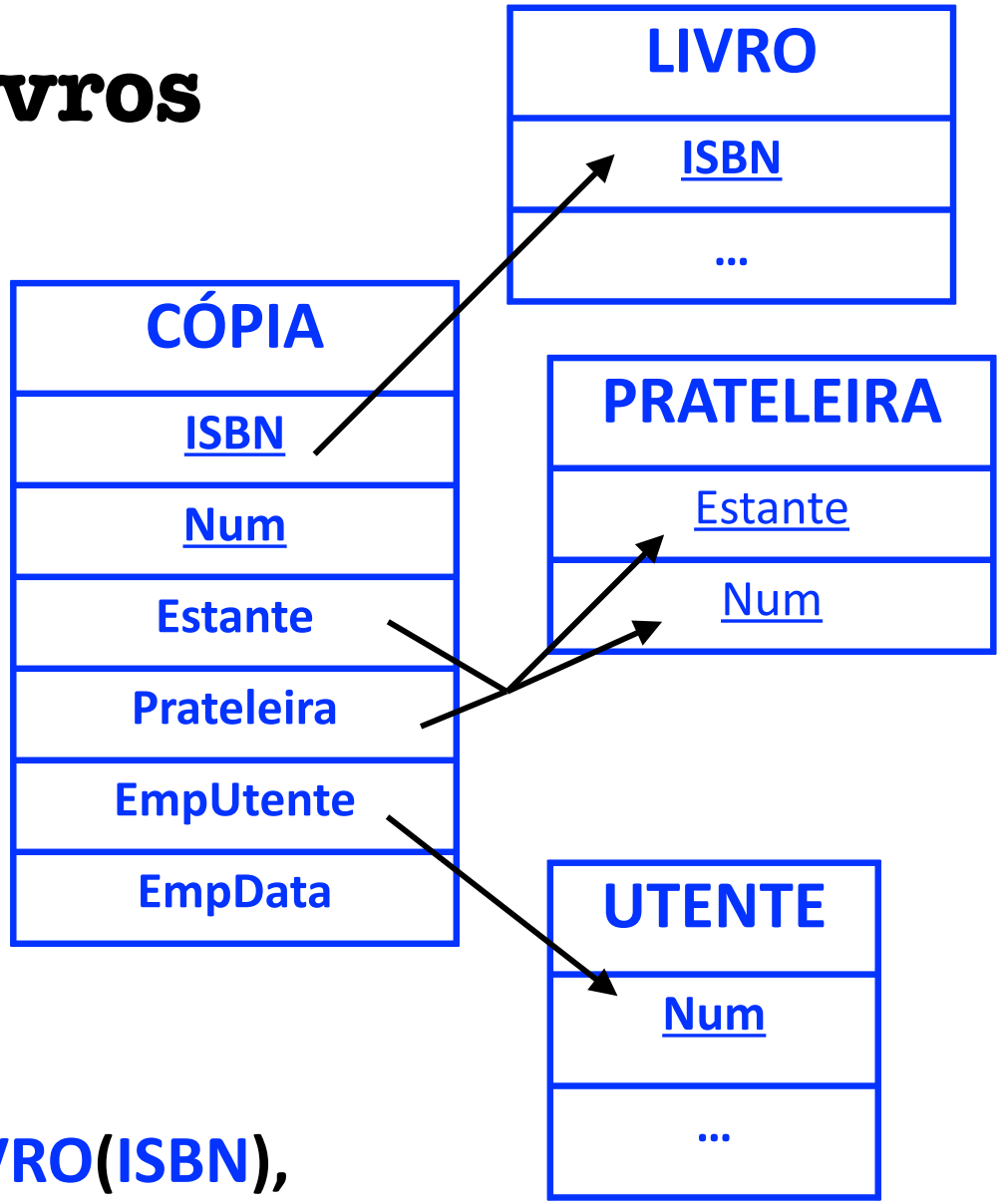


Tabela para cópias de livros




```
CREATE TABLE CÓPIA
(
  ISBN BIGINT NOT NULL,
  Num TINYINT NOT NULL,
  Estante VARCHAR(3) NOT NULL,
  Prateleira TINYINT NOT NULL,
  EmpUtente INT,
  EmpData DATE,
  PRIMARY KEY(ISBN, Num),
  FOREIGN KEY(ISBN) REFERENCES LIVRO(ISBN),
  FOREIGN KEY(EmpUtente) REFERENCES UTENTE(Num),
  FOREIGN KEY(Estante, Prateleira) REFERENCES PRATELEIRA(Estante, Num)
);
```

A instrução DROP TABLE

```
DROP TABLE IF EXISTS
  CÓPIA,
  PRATELEIRA,
  ESTANTE,
  SECÇÃO,
  AUTORES,
  LIVRO,
  UTENTE;
```

- DROP TABLE **remove do esquema da BD uma ou mais tabelas.**
  - A tabelas deixa de existir no esquema da BD.
  - E por inerência os dados associados às tabelas são removidos.
- **Nota**: a cláusula IF EXISTS verifica previamente se as tabelas existem para evitar erros. Analogamente CREATE TABLE suporta a cláusula simétrica IF NOT EXISTS.

A instrução CREATE DATABASE

```
CREATE DATABASE IF NOT EXISTS BIBLIOTECA;
USE BIBLIOTECA;
```

- Deve ser executada para inicializar o esquema .
- Sinónimo: **CREATE SCHEMA**
- Depois de criada uma BD empregamos a instrução USE para indicar qual o esquema activo p/alteração do mesmo ou p/manipulação de dados (como veremos a seguir).

Manipulação de registos

Formas básicas das instruções

INSERT, SELECT, UPDATE, e DELETE

Instruções essenciais de manipulação de dados

- **INSERT**
  - **Inserção** de registos
- **SELECT**
  - **Consulta** (leitura) de registos (não altera BD)
- **UPDATE**
  - **Actualização** de registos já existentes
- **DELETE**
  - **Remoção** de registos

Inserção de registos — forma básica

```
INSERT INTO
  <NOME DA TABELA>(<ATRIBUTO 1>, …., <ATRIBUTO N>)
VALUES
  (<VALOR 1.1>, …., <VALOR 1.N>),
  (<VALOR 1.2>, …., <VALOR 2.N>),
  … ,
  (<VALOR K.2>, …., <VALOR K.N>);
```

- É identificado o nome da tabela e os atributos a definir na inserção, seguido dos valores para um conjunto de registos.

Exemplos de inserção de registos

```
INSERT INTO UTENTE(Num, CC, Nome, DataNasc, Sexo, Telefone, Email)
VALUES
 (1, 10583212, 'João Pinto', '1976-12-19', 'M', 913448748, 'azulibranco@fcp.pt'),
 (2, 12447555, 'Carlos Semedo', '1985-06-02', 'M', 223774327, NULL),
 (3, 16348500, 'Maria Silva', '2005-11-17', 'F', 939939939, NULL),
 (4, 11983516, 'Pedro Costa', '1982-01-03', 'M', 384388291, 'pc12345@xpto.com');
```

UTENTE						
Num	CC	Nome	DataNasc	Sexo	Telefone	Email
1	10583212	João Pinto	1976-12-19	M	913 448 748	azulibranco@fcp.pt
2	12447555	Carlos Semedo	1985-06-02	M	223 774 327	NULL
3	16348500	Maria Silva	2005-11-17	F	939 939 939	NULL
4	11983516	Pedro Costa	1982-01-03	M	384 388 291	pc12345@xpto.com

Exemplos de inserção de registos (cont.)

```
INSERT INTO LIVRO(ISBN, Título, Editora, Ano)
VALUES
 (9789722709620, 'Os Lusíadas', 'INCM', 1999),
 (9789722526289, 'Sonetos', 'WOOK 11-17', 2013),
 (9780131103627,
  'The C Programming Language, 2nd edition',
   ‘Prentice Hall', 1988);

INSERT INTO AUTORES(ISBN, Nome)
VALUES
  (9789722709620, 'Luís de Camões'),
  (9789722526289, 'Luís de Camões'),
  (9780131103627, 'Brian W. Kernighan'),
  (9780131103627, 'Dennis M. Ritchie');
```

LIVRO			
ISBN	Título	Editora	Ano
9789722709620	Os Lusíadas	INCM	1999
9789722526289	Sonetos	WOOK 11-17	2013
9780131103627	The C Programming Language, 2nd edition	Prentice Hall	1988

AUTOR	
ISBN	Autor
9789722709620	Luís de Camões
9789722526289	Luís de Camões
9780131103627	Brian W. Kernighan
9780131103627	Dennis M. Ritchie

Consulta de registos — forma básica

```
SELECT
   <ATRIBUTO 1>, …., <ATRIBUTO N>
FROM
  <NOME DA TABELA>
WHERE
  <CONDIÇÃO>;
```

- Permite a consulta ("query") dos valores atributos da tabela indicada sob determinada **condição de selecção**, definida pelo que chama a **cláusula WHERE**.
- A "wildcard" * para os atributos define implicitamente **todos os atributos**. Na ausência da cláusula **WHERE** os valores são devolvidos para **todos os registos** da tabela. Vários operadores de seleção podem ser aplicados na cláusula WHERE. **Damos a seguir apenas alguns exemplos.**

Exemplos de consultas básicas

```
/* Consulta todos os atributos de todos os utentes. */
SELECT * FROM UTENTE;
```

```
+-----+----------+--------------+------------+------+-----------+--------------------+
| Num | CC       | Nome         | DataNasc   | Sexo | Telefone  | Email              |
+-----+----------+--------------+------------+------+-----------+--------------------+
|   1 | 10583212 | João Pinto   | 1976-12-19 | M    | 913448748 | azulibranco@fcp.pt |
|   2 | 12447555 | Carlos Semedo| 1985-06-02 | M    | 223774327 | NULL               |
|   3 | 16348500 | Maria Silva  | 2005-11-17 | F    | 939939939 | NULL               |
|   4 | 11983516 | Pedro Costa  | 1982-01-03 | M    | 384388291 | pc12345@xpto.com   |
+-----+----------+--------------+------------+------+-----------+--------------------+
```

```
/* Consulta nome, telefone, e email do utente com número 2.*/
SELECT Nome, Telefone, Email
FROM UTENTE
WHERE Num = 2;
```

```
+---------------+-----------+-------+
| Nome          | Telefone  | Email |
+---------------+-----------+-------+
| Carlos Semedo | 223774327 | NULL  |
+---------------+-----------+-------+
```

Exemplos de consultas básicas (cont.)

```
/* Consulta ISBN de todos os livros com autor Luís de Camões. */
SELECT ISBN FROM AUTORES
WHERE Nome = 'Luís de Camões';
```

```
+---------------+
| ISBN          |
+---------------+
| 9789722526289 |
| 9789722709620 |
+---------------+
```

```
/* Consulta ISBN e NUM de cópias na Estante 12 e prateleira 1 */
SELECT ISBN,Num FROM CÓPIA
WHERE Estante = 'E12'  AND Prateleira = 1;
```

```
+---------------+-----+
| ISBN          | Num |
+---------------+-----+
| 9789722709620 |   1 |
| 9789722709620 |   2 |
+---------------+-----+
```

Exemplos de consultas básicas (cont.)

```
/* Consulta ISBN, número, utente e data de empréstimo
   para todos os livros que estão emprestados. */
SELECT ISBN, Num, EmpUtente, EmpData
FROM CÓPIA
WHERE EmpUtente IS NOT NULL;
```

ISBN	Num	EmpUtente	EmpData
9789722709620	1	1	2019-01-17
9789722526289	1	2	2018-12-28
9789722709620	3	2	2018-12-27
9780131103627	2	4	2019-02-01

Actualização de registos — forma básica

```
UPDATE
  <NOME DA TABELA>
SET
  <ATRIBUTO 1> = <VALOR 1>,
  …
  <ATRIBUTO K> = <VALOR K>
WHERE
  <CONDIÇÃO>;
```

- Permite a actualização dos valores de um sub-conjunto de atributos para registos que verificam determinada condição.
- Na ausência da cláusula **WHERE** **todos os registos** são considerados para actualização.

Exemplos de actualizações (teste na consola mysql)

```
/* Actualiza Ano com valor 2000 para livro com ISBN 9789722709620. */
UPDATE LIVRO
SET Ano = 2000
WHERE ISBN = 9789722709620;

/* Actualiza Secção com valor ’T' para estante com código E12. */
UPDATE ESTANTE
SET Secção = ’T'
WHERE Código = 'E12';

/* Actualiza a localização de todas as cópias na estante E12, prateleira 1
   para estante E13, prateleira 2. */
UPDATE CÓPIA
SET Estante = 'E13',
    Prateleira = 2
WHERE Estante = 'E12' AND Prateleira = 1;
```

Remoção de registos — forma básica

```
DELETE FROM
  <NOME DA TABELA>
WHERE
  <CONDIÇÃO>;
```

- Permite a remoção de registos de uma tabela indicada que verifiquem determinada condição.
- Na ausência da cláusula **WHERE todos os registos** são considerados para remoção.

Exemplos de remoções (teste na consola mysql)

```
/* Remove todos os registos da tabela CÓPIA */
DELETE FROM CÓPIA;

/* Remove a CÓPIA  número 2 do livro com ISBN 9789722709620 */
DELETE FROM CÓPIA
WHERE ISBN=9789722709620 AND Num = 2;

/* Remove todas os registos das prateleiras da Estante E14 */
DELETE FROM PRATELEIRA
WHERE Estante = 'E14';
```

Violação de restrições de integridade

```
INSERT INTO UTENTE(Num, CC, Nome, DataNasc, Sexo, Telefone, Email)
VALUES
 /* Já existe utente 3 - integridade de chave primária (PRIMARY KEY)
    violada. */
 (3, 17134500, 'Joana Silva', '2000-01-07', 'F', 90339001, NULL);
```

ERROR 1062 (23000): Duplicate entry '3' for key 'PRIMARY'

```
INSERT INTO UTENTE(Num, CC, Nome, DataNasc, Sexo, Telefone, Email)
VALUES
 /* Já existe utente com CC 16348500 - integridade de chave secundária
    (UNIQUE) violada. */
 (5, 16348500, 'Joana Silva', '2000-01-07', 'F', 90339001, NULL);
```

ERROR 1062 (23000): Duplicate entry '16348500' for key ‘CC'

Violação de restrições de integridade (cont.)

```
INSERT INTO UTENTE(Num, CC, Nome, DataNasc, Sexo, Telefone, Email)
VALUES
 /* NULL não permitido como chave primária - integridade de
    identidade violada. */
 (NULL, 17134500, 'Joana Silva', '2000-01-07', 'F', 90339001, NULL);
```

ERROR 1048 (23000): Column 'Num' cannot be null

```
UPDATE UTENTE
/* CC deve ser um número - Integridade de domínio violada. */
SET CC = 'CC123994'
WHERE Num = 2;
```

ERROR 1366 (HY000): Incorrect integer value: 'CC123994' for column 'CC' at row 1

Violação de restrições de integridade (cont.)

```
UPDATE CÓPIA
/* Utente 5 não existe - integridade referencial violada.  */
SET
   EmpUtente = 5,
   EmpData = '2019-01-18'
WHERE ISBN = 9789722709620 AND Num = 1;
```

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key
constraint fails (`biblioteca`.`cópia`, CONSTRAINT `cópia_ibfk_2`
FOREIGN KEY (`EmpUtente`) REFERENCES `utente` (`Num`))
```

Violação de restrições de integridade (cont.)

```
/* Há cópias que se referem a este livro - integridade referencial violada.  */
DELETE FROM LIVRO
WHERE ISBN = 9789722709620;
```

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`biblioteca`.`autores`, CONSTRAINT `autores_ibfk_1` FOREIGN KEY (`ISBN`) REFERENCES `livro` (`ISBN`))
```

Introdução a SQL – parte 2

- Tipos, operadores e funções
  - Tipos de dados em mais detalhe
  - Operadores e funções
  - Uso de operadores e funções em instruções SQL

- Definição de tabelas – aspectos complementares:
  - Uso de **DEFAULT**, **AUTOINCREMENT** e **CHECK**
  - A instrução **ALTER TABLE**
  - Uso de **ON UPDATE** e **ON DELETE** para configuração de acções de preservação de integridade referencial.

Tipos de dados

Tipos de dados principais para atributos (MySQL)

Números Inteiros	TINYINT (1 byte), SMALLINT (2) MEDIUMINT (3), INT (4), BIGINT (8)
Números de vírgula flutuante	DECIMAL, NUMERIC, FLOAT, DOUBLE
Sequências de caracteres	CHAR, VARCHAR, ENUM, TEXT / MEDIUMTEXT /LONGTEXT
Tempo	DATE, DATETIME, TIME, TIMESTAMP, YEAR

- Referência: MySQL Reference Manual - Data Types

Tipos inteiros — modificador UNSIGNED

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

- MySQL Reference Manual (11.2.1)
- O modificador **UNSIGNED** restringe o domínio a valores não negativos (>= 0), ex. **INT UNSIGNED.**
- **BOOL / BOOLEAN** são implementados como **TINYINT** em MySQL (**TRUE = 1, FALSE = 0**). **INTEGER** é sinómimo de **INT.**

Tipos de vírgula flutante

- **FLOAT**
  - Números reais representáveis em 32 bits com precisão arbitrária
- **DOUBLE**
  - Números reais representáveis em 64 bits com precisão arbitrária
- **DECIMAL(D, P)**
  - Números reais representáveis na escala de **D** dígitos decimais com precisão de **P** dígitos
  - Exemplo — valores correspondentes a DECIMAL(3,2) podem ser: 100.23 -2.35 2.1 -999.99
  - ... mas não 1234 1.234 -1234.567

Sequências de caracteres

- **CHAR(N)**
  - Sequência de caracteres de **tamanho fixo N**, onde **N <= 255 (2^8-1).** Para um valor de tamanho **L <= N** , a BD armazena sempre **N** caracteres usando o caracter para "padding" à direita.
- **VARCHAR(N)**
  - Sequência de caracteres de **tamanho variável até N caracteres**, onde **N <= 65535 (2^{16} - 1)**. Para um valor de tamanho **L <= N** , a BD só armazena **L** caracteres.
- **ENUM('VALOR 1', …, 'VALOR N')**
  - Tipo enumerado (como exemplificado na BD biblioteca)
- **TEXT / MEDIUMTEXT / LONGTEXT**
  - Texto de comprimento variável até 64 KB / 16 MB / 4GB

Tipos de dados temporais

- **YEAR**
  - Ano
- **DATE**
  - Representação de datas com a forma YYYY-MM-DD
- **TIME**
  - Representação de horas com a forma [H]HH:MM:SS
- **DATETIME, TIMESTAMP**
  - Representação de data e hora.
  - Valores com a forma YYYY-MM-DD [H]HH:MM:SS[.dddddd]
  - 6 dígitos de precisão (até micro-segundos)
  - **TIMESTAMP** guardado na BD com conversão para "UTC time"

Funções e operadores

Funções e operadores

- Em SQL estão definidas funções e operadores que podemos usar em associação às instruções em SQL.

- Os principais operadores/funções permitem
  - Operações tradicionais como as de índole aritméticas, relacional ou lógica
  - Outras operações específicas sobre tipos de dados como sequências de caracteres ou dados temporais

- Referência para MySQL: MySQL Reference Manual — 12.1 Function and Operator Reference

Alguns dos principais operadores

Operadores aritméticos	+ - * / % (os usuais)
Operadores relacionais	= (igualdade) <> (desigualdade) < <= > >= IS NULL IS NOT NULL
Operadores lógicos	AND / && OR / || NOT / !

Exemplo de uso em instruções

```
SELECT Título, Ano
FROM LIVRO
WHERE Ano >= 2000
AND Ano <= 2005;
```

```
+-------------------------------+------+
| Título                        | Ano  |
+-------------------------------+------+
| Astérix o Gaulês              | 2004 |
| A Volta à Gália de Astérix    | 2005 |
+-------------------------------+------+
```

```
SELECT Título, Ano
FROM LIVRO
WHERE  NOT
(Ano >= 2000
 AND Ano <= 2005);
```

```
+-----------------------------------------+------+
| Título                                  | Ano  |
+-----------------------------------------+------+
| The C Programming Language, 2nd edition | 1988 |
| Effective Java                          | 2008 |
| A Mensagem                              | 2017 |
| Sonetos                                 | 2013 |
| Os Lusíadas                             | 1999 |
+-----------------------------------------+------+
```

Exemplo de uso em instruções (cont.)

```
DELETE FROM LIVRO
WHERE Ano >= 2000
AND Ano <= 2005;
```

Título	Ano
Astérix o Gaulês	2004
A Volta à Gália de Astérix	2005

```
UPDATE LIVRO
SET
  ANO = ANO + 1
WHERE  NOT
(Ano >= 2000
  AND Ano <= 2005);
```

Título	Ano
The C Programming Language, 2nd edition	1988
Effective Java	2008
A Mensagem	2017
Sonetos	2013
Os Lusíadas	1999

Título	**Ano**
The C Programming Language, 2nd edition	**1989**
Effective Java	**2009**
A Mensagem	**2018**
Sonetos	**2014**
Os Lusíadas	**2000**

Algumas funções sobre datas

- **NOW()**
  - Devolve a data e tempo actual (DATETIME)
- **YEAR(x) MONTH(x) DAY(x)**
  - Resp. devolvem o ano, mês, dia de um valor DATE ou DATETIME
- **HOUR(x) MINUTE(x), SECOND(x)**
  - Análogo para horas, minutos e segundos para um valor DATETIME ou TIME
- **TIMESTAMPDIFF(ESCALA, A, B)**
  - Calcula diferença entre **B** e **A** em determinada escala (**ESCALA**=YEAR|MONTH|DAY|HOUR| MINUTE|SECOND|…)

Exemplo de uso em instruções

```
SELECT  Nome, DataNasc,
        (DATEYEAR(NOW()) - YEAR(DataNasc))
        AS Idade FROM UTENTE ;
```

Boa aproximação, mas cálculo não é rigoroso!

Utente pode ainda não ter feito aniversário no ano corrente (data destes slides: 8/3/2019).

Nome	DataNasc	Idade
João Pinto	1976-12-19	43
Carlos Semedo	1985-06-02	34
Maria Silva	2005-11-17	14
Pedro Costa	1982-01-03	37
Filipa Mendes	2002-05-03	17
Eva Mendes	1975-11-18	44
Pedro Simões	1993-02-22	26

- Empregamos neste caso para obter **valores de um atributos derivado.**
- Modificador (opcional) **AS Nome** serve para dar nome a um atributo derivado ou renomear um atributo nos resultados.

Exemplo de uso em instruções (cont.)

```
SELECT  Nome, DataNasc,
        TIMESTAMPDIFF(YEAR, DataNasc, NOW()) AS Idade
FROM UTENTE ;
```

Nome	DataNasc	Idade
João Pinto	1976-12-19	43
Carlos Semedo	1985-06-02	34
Maria Silva	2005-11-17	14
Pedro Costa	1982-01-03	37
Filipa Mendes	2002-05-03	17
Eva Mendes	1975-11-18	44
Pedro Simões	1993-02-22	26

Aproximação anterior

Nome	DataNasc	Idade
João Pinto	1976-12-19	42
Carlos Semedo	1985-06-02	33
Maria Silva	2005-11-17	13
Pedro Costa	1982-01-03	37
Filipa Mendes	2002-05-03	16
Eva Mendes	1975-11-18	43
Pedro Simões	1993-02-22	26

Cálculo correcto!

Exemplos de operações sobre strings

- **LENGTH(S)**
  - Devolve tamanho de **S**
- **CONCAT(S1,S2, …)**
  - Devolve concatenação de strings **S1, S2, …**
- **REPLACE(S, A, B)**
  - Obtém resultado de substituir **A** por **B** em **S**
- **REVERSE(S)**
  - Obtém **S** invertida.
- **S [NOT] LIKE <EXP REGULAR>**
  - Verifica S face a expressão regular SQL.
  - **_** : um qualquer caracter; **%**: qualquer sequência de 0 ou mais caracteres

Exemplo de uso em instruções (cont.)

```
SELECT ISBN, CONCAT(Título, ' (', Editora, ',', Ano, ')') AS Info
FROM LIVRO
WHERE Título LIKE '%Ast_rix%';
```

ISBN	Info
9789724138695	**Astérix** o Gaulês (Edições Asa,2004)
9789724138947	A Volta à Gália de **Astérix** (Edições Asa,2005)

```
UPDATE LIVRO
SET Editora= 'Assírio & Alvim'
WHERE Título LIKE '%Ast_rix%';
```

repetindo
a consulta acima
após UPDATE

ISBN	Info
9789724138695	**Astérix** o Gaulês (**Assírio & Alvim**,2004)
9789724138947	A Volta à Gália de **Astérix** (**Assírio & Alvim**,2005)

Funções de escolha de valores

- **IFNULL(A,B)**
  - Devolve **A** se **A não é NULL**, **B** caso contrário.
- **IF(COND, A, B)**
  - Devolve **A** se **COND = TRUE**, **B** caso contrário.

```
SELECT Nome,
       IFNULL(Email,'Sem email')
       AS 'Endereço Email'
FROM UTENTE;
```

```
+--------------+--------------------------+
| Nome         | Endereço Email           |
+--------------+--------------------------+
| João Pinto   | azulibranco@fcp.pt       |
| Carlos Semedo| Sem email                |
| Maria Silva  | Sem email                |
| Pedro Costa  | pc12345@xpto.com         |
| Filipa Mendes| filipa.mendes@gmail.com  |
| Eva Mendes   | Sem email                |
| Pedro Simões | simoes333@gmail.com      |
+--------------+--------------------------+
```

```
SELECT Num,
       IF(EmpUtente IS NULL,
          'Emprestada',
          'Disponível'),
FROM CÓPIA
WHERE ISBN = 9789722709620;
```

```
+---------------+-----+-------------+
| ISBN          | Num | Estado      |
+---------------+-----+-------------+
| 9789722709620 |   1 | Disponível  |
| 9789722709620 |   2 | Emprestada  |
| 9789722709620 |   3 | Disponível  |
| 9789722709620 |   4 | Disponível  |
+---------------+-----+-------------+
```

Definições de tabelas
— aspectos complementares —

Uso de DEFAULT

```
CREATE TABLE UTENTE
(
  …
  Email VARCHAR(32) DEFAULT NULL,
  …
);
/* Usa NULL para Email por omissão. */
INSERT INTO UTENTE(Num, CC, Nome, DataNasc, Sexo, Telefone)
VALUES
 (8, 12589212, 'Samuel Silva', '1986-12-19', 'M', 223774044);
```

- **DEFAULT** permite definir um valor a usar por omissão para um atributo.
- Uma instrução **INSERT** que omita o valor do atributo usa o valor especificado por **DEFAULT**.

Uso de AUTO_INCREMENT

```
CREATE TABLE UTENTE (
  Num INT NOT NULL AUTO_INCREMENT,
  …
  Email VARCHAR(32) DEFAULT NULL,   … );

/* Usa valor de sequência para Num e NULL para Email por omissão. */
INSERT INTO UTENTE(CC, Nome, DataNasc, Sexo, Telefone)
VALUES
 (12589212, 'Samuel Silva', '1986-12-19', 'M', 223774044);
```

- O uso de **AUTO_INCREMENT** permite que o valor de um atributo tenha um valor sequência associado que é automaticamente incrementado em cada inserção. O atributo tem de ser uma chave primária.
- É bastante conveniente em tabelas que usam um identificador de numeração como chave primária, uma situação frequente.

Uso de CHECK (parcialmente suportado em MySQL >=8.0)

```
CREATE TABLE PRATELEIRA (
  … Num TINYINT NOT NULL  CHECK(Num > 0), …
);
CREATE TABLE CÓPIA ( …
  EmpUtente INT,  EmpData DATE,
  CHECK(      ( EmpUtente IS NULL        && EmpData IS NULL        )
           ||  ( EmpUtente IS NOT NULL && EmpData IS NOT NULL) ), … );
```

- A cláusula **CHECK** permite definir restrições de domínio adicionais para um atributo. Podem ser associadas a um só atributo ou a uma tabela toda envolvendo vários atributos.
- **MySQL aceita sintaticamente restrições CHECK mas só valida de facto parte delas a partir da versão 8.0** e com "Strict SQL mode" habilitado – isto **embora o suporte de CHECK seja comum em outros SGBDs e parte da especificação SQL standard.**
- Para definir restrições deste tipo em MySQL será melhor recorrer a **"triggers"** (a ver mais à frente na cadeira).

Uso de ON UPDATE e ON DELETE — forma geral

```
CREATE TABLE <NOME DA TABELA>
(
    …
    FOREIGN KEY(<ATRIBUTOS QUE FORMAM UMA CHAVE EXTERNA>)
    REFERENCES <NOME_DE_OUTRA_TABELA>(<CHAVE_PRIMÁRIA>)
    [ON UPDATE <AÇÃO> ON DELETE <ACÇÃO>],
    …
    …
);
```

- Para **chaves externas** podemos especificar o que deve acontecer quando a entrada correspondente à chave primária referenciada é actualizada (**ON UPDATE**) ou apagada (**ON DELETE**).

Uso de ON UPDATE CASCADE

```
CREATE TABLE AUTORES (
  ISBN BIGINT NOT NULL,
  Nome VARCHAR(64) NOT NULL,
  PRIMARY KEY(ISBN, Nome),
  FOREIGN KEY(ISBN) REFERENCES LIVRO(ISBN)
  ON UPDATE CASCADE );
```

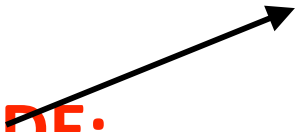
```
CREATE TABLE LIVRO (
  ISBN BIGINT NOT NULL,
  ...
  PRIMARY KEY(ISBN) );
```

- **ON UPDATE CASCADE**: substitui valor da nova chave primária em chaves externas.
  - **No exemplo:** se actualizarmos o **ISBN** para um registo em **LIVRO** este propagado em **AUTORES** **que** referenciavam o **ISBN** anterior.
  - Analogamente na BD biblioteca devemos ter esta configuração para **CÓPIA** que também tem uma chave externa p/**LIVRO**. Caso contrário, actualizações em **LIVRO** e **AUTORES** são rejeitadas (a não ser que não haja cópias do(s) livro(s) em causa).

Uso de ON DELETE CASCADE

```
CREATE TABLE AUTORES (
  ISBN BIGINT NOT NULL,
  Nome VARCHAR(64) NOT NULL,
  PRIMARY KEY(ISBN, Nome),
  FOREIGN KEY(ISBN) REFERENCES LIVRO(ISBN)
  ON UPDATE CASCADE ON DELETE CASCADE;
);
```

```
CREATE TABLE LIVRO (
  ISBN BIGINT NOT NULL,
  ...
  PRIMARY KEY(ISBN) );
```



- **ON DELETE CASCADE**: apaga entradas em outras tabelas que referenciem a chave primária do registo apagado.
- **No exemplo**: se apagarmos um registo em LIVRO, apagamos **também** todas os registos em AUTORES associados, isto novamente assumindo que há uma configuração similar em CÓPIA (se houverem cópias a referir os livros em causa)

Uso de ON UPDATE e ON DELETE — outras opções

- **ON UPDATE SET NULL / ON DELETE SET NULL**
  - Substitui valor da chave externa por **NULL.**
  - Válido desde que o domínio da chave externa admita o valor **NULL.**
- **ON UPDATE SET DEFAULT / ON DELETE SET DEFAULT**
  - Substitui valor da chave externa por valor **DEFAULT.**
  - Válido desde que o atributo da chave externa tenha associado um valor **DEFAULT.**
- **ON UPDATE RESTRICT / ON DELETE RESTRICT - a definição por omissão!**
  - **Operação de actualização / remoção é rejeitada.**

A instrução ALTER TABLE

Forma geral:

```
ALTER TABLE <NOME DA TABELA> [ADD|DROP|ALTER] <OPÇÕES>;
```

Exemplos:

```
ALTER TABLE UTENTE  DROP COLUMN Email;
ALTER TABLE UTENTE  ADD COLUMN
   Nacionalidade VARCHAR(16) DEFAULT 'Portuguesa';
ALTER TABLE UTENTE MODIFY Nome VARCHAR(128) NOT NULL;
```

- Permite alterar o esquema ou outras propriedades de uma tabela. Entre outros usos, a instrução permite adicionar / remover / rever atributos de tabelas.

Consultas SQL Avançadas

Consultas SQL avançadas

- Veremos agora uma série de consultas SQL "avançadas", i.e., além das básicas dos "slides" de introdução ao SQL.

- ... na verdade consultas que podem ser associadas também a alterações a uma BD como inserções, actualizações ou remoções.

- Continuaremos a usar a BD "biblioteca" para ilustração.

Conteúdo

- Ordenação e filtragem de resultados com ORDER BY, DISTINCT e LIMIT.
- Consultas "avançadas" de vários tipos:
  - com funções de agregação.
  - simples sobre múltiplas tabelas
  - encadeadas
  - usando operações de conjuntos.
  - usando operadores de junção ("joins")
- Definição e uso de vistas SQL.

Ordenação
e filtragem de resultados com
ORDER BY, DISTINCT e LIMIT

ORDER BY

- O uso de **ORDER BY** em uma consulta define um critério de ordenação para os resultados.

```
/* Obtém  Nome, DataNasc sem critério de
   ordenação. */
SELECT Nome, DataNasc FROM UTENTE;
```

João Pinto	1976-12-19
Carlos Semedo	1985-06-02
Maria Silva	2005-11-17
Pedro Costa	1982-01-03
Filipa Mendes	2002-05-03
Eva Mendes	1975-11-18
Pedro Simões	1993-02-22

```
/* Ordena por Nome. */
SELECT Nome, DataNasc FROM UTENTE
ORDER BY Nome;
```

Carlos Semedo	1985-06-02
Eva Mendes	1975-11-18
Filipa Mendes	2002-05-03
João Pinto	1976-12-19
Maria Silva	2005-11-17
Pedro Costa	1982-01-03
Pedro Simões	1993-02-22

```
/* Ordena por DataNasc de forma
   decrescente (utentes mais novos 1º). */
SELECT Nome, DataNasc FROM UTENTE
ORDER BY DataNasc DESC;
```

Maria Silva	**2005-11-17**
Filipa Mendes	**2002-05-03**
Pedro Simões	**1993-02-22**
Carlos Semedo	**1985-06-02**
Pedro Costa	**1982-01-03**
João Pinto	**1976-12-19**
Eva Mendes	**1975-11-18**

ORDER BY (cont.)

- Podemos ter mais do que um critério de ordenação.

```
/* Ordena por data de aniversário
    (mês primeiro, dia depois). */
SELECT Nome, DataNasc FROM UTENTE
ORDER BY MONTH(DataNasc),
         DAY(DataNasc);
```

```
| Pedro Costa   | 1982-01-03 |
| Pedro Simões  | 1993-02-22 |
| Filipa Mendes | 2002-05-03 |
| Carlos Semedo | 1985-06-02 |
| Maria Silva   | 2005-11-17 |
| Eva Mendes    | 1975-11-18 |
| João Pinto    | 1976-12-19 |
```

```
/* Ordena por data de empréstimo decrescente
   e depois por nº de cópia.
   (obs. NULLs por último em DESC). */
SELECT Num, EmpUtente, EmpData
FROM CÓPIA
WHERE ISBN=9789720049759
ORDER BY  EmpData DESC, Num;
```

```
+-----+-----------+------------+
| Num | EmpUtente | EmpData    |
+-----+-----------+------------+
|   1 |         4 | 2019-02-01 |
|   3 |         3 | 2018-11-01 |
|   2 |      NULL | NULL       |
|   4 |      NULL | NULL       |
+-----+-----------+------------+
```

LIMIT

- O uso de **LIMIT n** devolve apenas os primeiros **n** resultados de uma consulta (com ou sem ORDER BY especificado).

```
/* Obtém uma cópia disponível
 do livro com ISBN 9789724138695. */
SELECT Num FROM CÓPIA
WHERE EmpUtente IS NULL
AND ISBN = 9789724138695
LIMIT 1 ;
```

```
+-----+
| Num |
+-----+
|   2 |
+-----+
```

1º resultado

```
/* Obtém 3 utilizadores mais novos
    da biblioteca. */
SELECT  Nome, DataNasc,
FROM UTENTE
ORDER BY DateNasc DESC
LIMIT 3;
```

```
+----------------+------------+
| Nome           | DataNasc   |
+----------------+------------+
| Maria Silva    | 2005-11-17 |
| Filipa Mendes  | 2002-05-03 |
| Pedro Simões   | 1993-02-22 |
+----------------+------------+
```

DISTINCT

```
/* Obtém ISBNs de livros que não se
encontrem requisitados. Poderão aparecer
ISBNs repetidos em correspondência a
várias cópias.*/
SELECT ISBN FROM CÓPIA
WHERE EmpUtente IS NULL;
```

```
+--------------+
| ISBN         |
+--------------+
| 9780131103627 |
| 9780321356680 |
| 9789720049759 |
| 9789722526289 |
| 9789722709620 |
| 9789724138695 |
| 9789724138695 |
| 9789724138947 |
+--------------+
```

duplicado

```
/* Uso de DISTINCT elimina duplicados. */
SELECT DISTINCT ISBN FROM CÓPIA
WHERE EmpUtente IS NULL;
```

```
+--------------+
| ISBN         |
+--------------+
| 9780131103627 |
| 9780321356680 |
| 9789720049759 |
| 9789722526289 |
| 9789722709620 |
| 9789724138695 |
| 9789724138947 |
+--------------+
```

valor único

- O uso de **DISTINCT** em associação a uma consulta filtra os resultados duplicados.

Consultas com funções de agregação

Agregações simples

- **Sintaxe — forma mais simples:**

```
SELECT
   <FUNÇÃO DE AGREGAÇÃO>(<ATRIBUTO>)  [AS NOME],
FROM  <TABELA>
[WHERE <CONDIÇÃO>];
```

- **Semântica:**
  - **1)** Agrupa todos os valores de <ATRIBUTO> ignorando valores NULL
  - **2)** Aplica sobre estes <FUNÇÃO DE AGREGAÇÃO>
  - **3)** Devolve o resultado final.
- **Nota:** Podemos especificar várias funções de agregação numa consulta só — veremos alguns exemplos a seguir.

Funções de agregação em SQL

- <FUNÇÃO DE AGREGAÇÃO> identifica **operação associativa e cumutativa** — ex. soma, máximo mas não subtração ou divisão.
- Alguns dos operadores mais comuns são listados abaixo
  - Ver por exemplo referência MySQL para estes e outros.

Função	**Operação**
COUNT	Contagem
COUNT(DISTINCT)	Contagem de elementos distintos
SUM	Soma
MIN	Mínimo
MAX	Máximo
AVG	Média
STDDEV	Desvio padrão

Agregação simples — exemplos (cont.)

- **COUNT(*) :** caso especial — conta o nº de registos.

```
/* Obtém nº total de utentes. */
SELECT COUNT(*) AS 'Nº de utentes'
FROM UTENTE;
```

```
+-----------------+
| Nº de utentes   |
+-----------------+
|               7 |
+-----------------+
```

```
/* Obtém mínimo, máximo e média
   do ano de nascimento. */
SELECT MIN(YEAR(DataNasc)) As Min,
       MAX(YEAR(DataNasc)) As Max,
       AVG(YEAR(DataNasc)) As Média
FROM UTENTE;
```

```
+------+------+------------+
| Min  | Max  | Média      |
+------+------+------------+
| 1975 | 2005 | 1988.2857  |
+------+------+------------+
```

Agregação simples — exemplos (cont.)

```
/* Obtém sumário de cópias para ISBN 9789722709620 */
SELECT
  COUNT(*) AS Total,
  COUNT(EmpUtente) AS Emprestadas,
  COUNT(*) - COUNT(EmpUtente)  AS Disponíveis
FROM CÓPIA
WHERE ISBN=9789722709620;
```

```
+-------+-------------+---------------+
| Total | Emprestadas | Disponíveis   |
+-------+-------------+---------------+
|     4 |           3 |             1 |
+-------+-------------+---------------+
```

- Observe que **COUNT(EmpUtente)** ignora valores NULL

Agregação com GROUP BY

- **Sintaxe — forma mais simples:**

```
SELECT   <ATRIBUTO DE AGRUPAMENTO>,
         <FUNÇÃO DE AGREGAÇÃO>(<ATRIBUTO>)  [AS NOME],
FROM  <TABELA> [WHERE <CONDIÇÃO>]
GROUP BY <ATRIBUTO PARA AGRUPAMENTO>;
```

- **Semântica:**
  - **1)** Agrupa todos os valores de <ATRIBUTO> **mas em grupos distintos por cada valor diferente em vez de um só (quando não temos GROUP BY).**
  - **2)** Aplica sobre **cada grupo** <FUNÇÃO DE AGREGAÇÃO>
  - **3)** Devolve o resultado final para **cada grupo**.
- **Nota:** Podemos especificar várias funções de agregação e vários atributos de agrupamento.

Agregação com GROUP BY (cont.)

```
SELECT
  ISBN,
  COUNT(*) AS Total,  COUNT(EmpUtente) AS Emprestadas,
  COUNT(*) - COUNT(EmpUtente)  AS Disponíveis
FROM CÓPIA
GROUP BY ISBN;
```

ISBN	Total	Emprestadas	Disponíveis
9780131103627	2	1	1
9780321356680	1	0	1
9789720049759	3	2	1
9789722526289	2	1	1
9789722709620	4	3	1
9789724138695	4	2	2
9789724138947	2	0	2

Agregação com GROUP BY (cont.)

```
/* Obtém nº de empréstimos e empréstimo mais antigo por
utente (utentes sem empréstimos não listados). */
SELECT
   EmpUtente AS Utente,  COUNT(EmpUtente) AS Empréstimos,
   MIN(EmpData) AS EmpMaisAntigo
FROM CÓPIA WHERE Utente IS NOT NULL
GROUP BY Utente;
```

Utente	Empréstimos	EmpMaisAntigo
1	2	2019-01-17
2	2	2018-12-27
3	1	2018-11-01
4	2	2019-02-01
5	1	2018-11-19
6	1	2018-02-01

Agregação com GROUP BY - HAVING

- **Sintaxe — forma mais simples:**

```
SELECT   [<ATRIBUTO DE AGRUPAMENTO>],
          <FUNÇÃO DE AGREGAÇÃO>(<ATRIBUTO>)  [AS NOME]
FROM  <TABELA> [WHERE <CONDIÇÃO>]
GROUP BY <ATRIBUTO DE AGRUPAMENTO>
HAVING <CONDIÇÃO SOBRE GRUPO>;
```

- **Semântica:**
  - <CONDIÇÃO SOBRE GRUPO> filtra resultados por grupo
  - **HAVING** funciona como cláusula **WHERE** ao nível de cada grupo

Agregação com GROUP BY — HAVING (cont.)

```
SELECT
  ISBN,
  COUNT(*) AS Total,  COUNT(EmpUtente) AS Emprestadas,
  COUNT(*) - COUNT(EmpUtente)  AS Disponíveis
FROM CÓPIA
GROUP BY ISBN
HAVING Emprestadas > 0;
```

ISBN	Total	Emprestadas	Disponíveis
9780131103627	2	1	1
9789720049759	3	2	1
9789722526289	2	1	1
9789722709620	4	3	1
9789724138695	4	2	2

Agregação com GROUP BY — HAVING (cont.)

```
/* Obtém nº de empréstimos e data do empréstimo mais antigo
para utentes com pelo menos uma cópia de um livro emprestado
há um mês ou mais. */
SELECT   EmpUtente AS Utente,
         COUNT(EmpUtente) AS Empréstimos,
         MIN(EmpData) AS EmpMaisAntigo
FROM CÓPIA  GROUP BY Utente
HAVING TIMESTAMPDIFF(MONTH, EmpMaisAntigo, NOW()) >= 1 ;
```

Utente	Empréstimos	EmpMaisAntigo
1	2	2019-01-17
2	2	2018-12-27
3	1	2018-11-01
4	2	2019-02-01
5	1	2018-11-19
6	1	2018-02-01

Agregação com GROUP BY — HAVING (cont.)

```
/* Exemplo anterior combinado com ORDER BY e LIMIT */
SELECT   EmpUtente AS Utente,
         COUNT(EmpUtente) AS Empréstimos,
         MIN(EmpData) AS EmpMaisAntigo
FROM CÓPIA  GROUP BY Utente
HAVING TIMESTAMPDIFF(MONTH, EmpMaisAntigo, NOW()) >= 1
ORDER BY Empréstimos DESC, EmpMaisAntigo
LIMIT  4;
```

```
+--------+--------------+----------------+
| Utente | Empréstimos  | EmpMaisAntigo  |
+--------+--------------+----------------+
|      2 |            2 | 2018-12-27     |
|      1 |            2 | 2019-01-17     |
|      4 |            2 | 2019-02-01     |
|      6 |            1 | 2018-02-01     |
+--------+--------------+----------------+
```

Consultas em múltiplas tabelas

Consultas em múltiplas tabelas

- **Forma geral**

```
SELECT
   <ATRIBUTOS>
FROM  <TABELA 1> [ID 1], …, <TABELA n> [<ID n>]
…
```

- De forma geral a cláusula **FROM** pode referir várias tabelas.
- Por forma a desambiguar atributos com o mesmo nome ou tornar a consulta mais legível podemos opcionalmente identificar cada tabelas com um identificador.
- **Nota**: Muitas vezes estas consultas correspondem implicitamente a operações de **junção ("join")** a ver mais tarde e para as quais há suporte SQL mais especializado.

Consultas em múltiplas tabelas (cont.)

```
/* Obtém título, editora e ano de livros de Luís de Camões. */
SELECT
   L.Título, L.Editora, L.Ano,
FROM
   LIVRO L,  AUTORES A
WHERE
  A.ISBN = L.ISBN AND A.Nome = 'Luís de Camões'
ORDER BY L.Título;
```

```
+---------------+-------------+------+
| Título        | Editora     | Ano  |
+---------------+-------------+------+
| Os Lusíadas   | INCM        | 1999 |
| Sonetos       | WOOK 11-17  | 2013 |
+---------------+-------------+------+
```

Consultas em múltiplas tabelas (cont.)

```
/* Obtém título de livro e nº de cópias por título. */
SELECT
   L.Título, COUNT(*)  AS Cópias
FROM  LIVRO L, CÓPIA C
WHERE  L.ISBN = C.ISBN
GROUP BY L.Título
ORDER BY Cópias, L. Título;
```

Título	Cópias
Effective Java	1
The C Programming Language, 2nd edition	2
Sonetos	2
A Volta à Gália de Astérix	2
A Mensagem	3
Os Lusíadas	4
Astérix o Gaulês	4

Consultas encadeadas

Consultas encadeadas

- **Forma geral**

```
SELECT … FROM … | DELETE FROM  … | UPDATE …
WHERE
  <EXPR> <OPERADOR> (
     SELECT …  /* Consulta encadeada */
  )
…
```

- Podemos **encadear consultas** usando na cláusula **WHERE** outra(s) consulta(s). Vários operadores podem ser usados para o encadeamento, como veremos a seguir.
- Podemos ter consultas encadeadas em associação a várias instruções como por exemplo **UPDATE** ou **DELETE** além de **SELECT**.

Operadores de consultas encadeadas

- **<EXPR> IN <CONSULTA>**
  - Verdadeiro se **<EXPR>** é devolvido por **<CONSULTA>**
- **<EXPR> NOT IN <CONSULTA>**
  - Verdadeiro se **<EXPR>** não é devolvido por **<CONSULTA>**
- **EXISTS <CONSULTA>**
  - Verdadeiro se **<CONSULTA>** devolve pelo menos um resultado.
- **<EXPR> <OPERADOR DE COMPARAÇÃO> <CONSULTA>**
  - Operadores de comparação (os usuais): **= <> > < >= <= ...**
  - Verdadeiro se **<CONSULTA>** devolve um **único** resultado **R** e **<EXPR> <OPERADOR DE COMPARAÇÃO> R** se verifica.
- **<EXPR> <OPERADOR DE COMPARAÇÃO> ALL <CONSULTA>**
  - Similar ao caso anterior mas se a comparação é verdadeira para **todos** os resultados da consulta.
- **<EXPR> <OPERADOR DE COMPARAÇÃO> ANY <CONSULTA>**
  - Similar ao caso anterior mas se a comparação é verdadeira para **pelo menos um** os resultados da consulta.

Consultas encadeadas — exemplos

```
/* Obtém título de livros com alguma cópia emprestada. */
SELECT Título FROM LIVRO
WHERE ISBN IN
( SELECT  ISBN FROM CÓPIA
  WHERE  EmpUtente IS NOT NULL );
```

```
The C Programming Language, 2nd edition
A Mensagem
Sonetos
Os Lusíadas
Astérix o Gaulês
```

```
/* Obtém título de livros sem qualquer cópia emprestada. */
SELECT Título FROM LIVRO
WHERE ISBN  NOT IN
( SELECT  ISBN FROM CÓPIA
  WHERE  EmpUtente IS NOT NULL );
```

```
Effective Java
A Volta à Gália de Astérix
```

Consultas encadeadas — exemplos (cont.)

```
/* Actualiza prateleira de arrumação dos livros do Astérix. */
UPDATE  CÓPIA SET Prateleira = 2 WHERE   ISBN IN  (
    SELECT  ISBN FROM LIVRO
    WHERE  Título LIKE '%Astérix%'
  );

/* Apaga cópias não emprestadas excepto as dos livros do Astérix. */
DELETE FROM  CÓPIA
WHERE   EmpUtente IS NULL AND  ISBN NOT IN  (
    SELECT  ISBN FROM LIVRO
    WHERE  Título LIKE '%Astérix%'
  );
```

Consultas encadeadas — exemplos (cont.)

```
/* Obtém nomes de utentes com
   livros requisitados */
SELECT Nome FROM UTENTE U
WHERE EXISTS
(SELECT ISBN FROM CÓPIA
 WHERE EmpUtente = U.Num);
```

```
+---------------+
| Nome          |
+---------------+
| João Pinto    |
| Carlos Semedo |
| Maria Silva   |
| Pedro Costa   |
| Filipa Mendes |
| Eva Mendes    |
+---------------+
```

```
/* Obtém nomes de utentes
   sem livros requisitados */
SELECT Nome FROM UTENTE U
WHERE NOT EXISTS
(SELECT * FROM CÓPIA
 WHERE EmpUtente = U.Num);
```

```
+---------------+
| Nome          |
+---------------+
| Pedro Simões  |
+---------------+
```

Consultas encadeadas — exemplos (cont.)

```
/* Obtém prateleiras com pelo menos
    uma cópia de um livro emprestado.  Mesmo resultado poderia
    ser obtido com EXISTS ou IN */
SELECT *
FROM PRATELEIRA P WHERE (P.Estante, P.Num) = ANY
  (SELECT Estante,Prateleira FROM CÓPIA
  WHERE Estante = P.Estante  AND EmpUtente IS NOT NULL);

/* Obtém utentes que fizeram os último empréstimos
registados por data. */
SELECT DISTINCT U.Num, U.Nome
FROM UTENTE U, CÓPIA C WHERE C.EmpUtente = U.Num
AND C.EmpData >= ALL
  ( SELECT EmpData FROM CÓPIA
    WHERE EmpUtente IS NOT NULL);
```

Consultas encadeadas — exemplos (cont.)

```
/* Regista devolução de uma cópia de 'Os Lusíadas'
   pelo utente João Pinto. */
UPDATE CÓPIA SET EmpUtente = NULL, EmpData = NULL WHERE
 EmpUtente = (SELECT Num FROM UTENTE WHERE Nome = 'João Pinto')
AND ISBN = (SELECT ISBN FROM LIVRO WHERE Título = 'Os Lusíadas');

/* Empréstimo de (exactamente) 1 cópia disponível de 'Astérix o Gaulês'
pelo utente João Pinto. Obs.: note-se o uso de LIMIT 1. */
UPDATE CÓPIA SET
   EmpUtente = (SELECT Num FROM UTENTE WHERE Nome = 'João Pinto'),
   EmpData = NOW()
WHERE EmpUtente IS NULL
AND ISBN = ANY (SELECT ISBN FROM LIVRO
                WHERE Título = 'Astérix o Gaulês')
LIMIT 1;
```

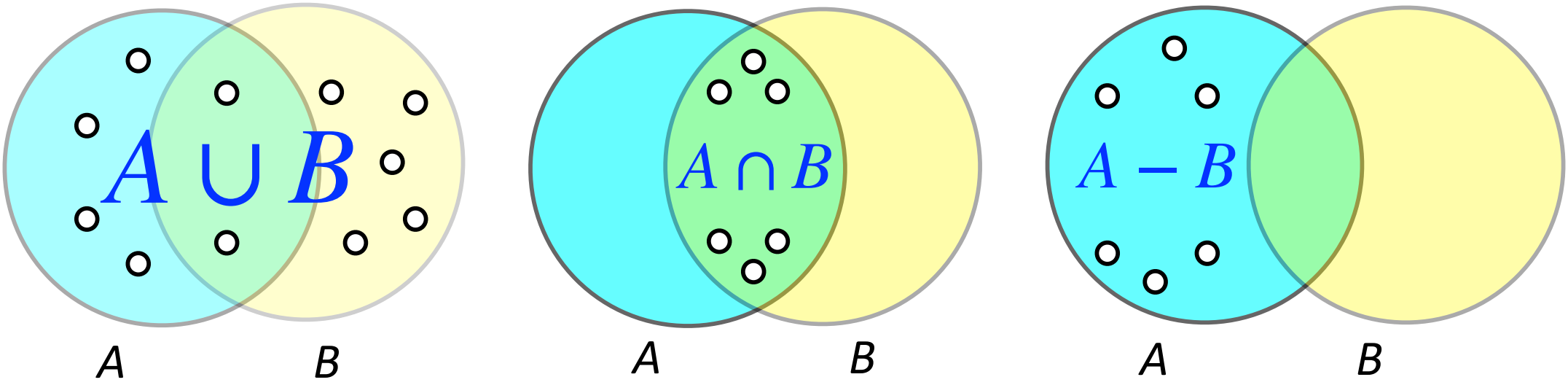
Consultas encadeadas — exemplos (cont.)

```
/* Insere cópia nº 5 de 'Os Lusíadas'. */
INSERT INTO CÓPIA
   (ISBN,
   Num, Estante, Prateleira,EmpUtente,EmpData)
VALUES
( (SELECT ISBN FROM LIVRO WHERE Título = 'Os Lusíadas'),
  5, 'E12', 1, NULL, NULL);

/* Seleciona nº de cópias de cada livro por título.
  Qual a consulta agregada "standard" equivalente? */
SELECT  Título,
          (SELECT COUNT(*) FROM CÓPIA WHERE ISBN = L.ISBN) AS Num
FROM LIVRO L
ORDER BY Título;
```

Consultas com operadores de conjuntos

Operações sobre conjuntos



- **Equivalente em SQL para consultas com mesmo nº de atributos iguais e respectivos domínios compatíveis:**
  - **<CONSULTA 1> UNION <CONSULTA 2>**
  - **<CONSULTA 2> INTERSECT <CONSULTA 2>**
  - **<CONSULTA 3> EXCEPT <CONSULTA 2>**
- **Nota:** MySQL suporta apenas **UNION,** mas consultas **INTERSECT** e **EXCEPT** podem ser reproduzidas recorrendo por ex. a consultas encadeadas do tipo **AND … IN <CONSULTA2>** e **AND … NOT IN <CONSULTA2>** .

Exemplo de uso de UNION

```
/* Obtém Num e Nome de utentes
que tenham mais de 3 cópias emprestadas OU
ainda não devolvidas há mais de 2 meses. */
SELECT U.Num, U.Nome
FROM UTENTE U
WHERE 3 <=
    ( SELECT COUNT(*) FROM CÓPIA
      WHERE EmpUtente = U.Num)
UNION
SELECT U.Num, U.Nome FROM UTENTE U
WHERE 2 <=
  (SELECT MAX(TIMESTAMPDIFF(MONTH, EmpData, NOW()))
  FROM CÓPIA WHERE EmpUtente = U.Num );
```

Num	Nome
4	Pedro Costa
2	Carlos Semedo
3	Maria Silva
5	Filipa Mendes
6	Eva Mendes

Num	Nome
4	Pedro Costa

U

Num	Nome
2	Carlos Semedo
3	Maria Silva
5	Filipa Mendes
6	Eva Mendes

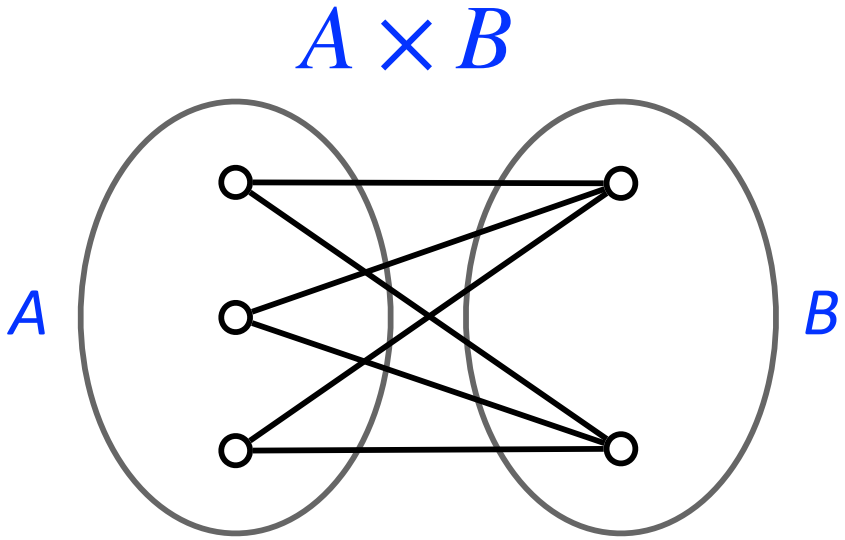
Consultas com junção de tabelas (“joins”)

Junções de tabelas (“joins”)

- Como vimos, as operações de consultas a múltiplas tabelas de uma BD pretendem “juntar” dados de forma correlacionada.
- Sendo esse um caso de uso bastante comum, a linguagem **SQL tem suporte explícito para operações de junção (“join”) de vários tipos.**
- Quando possível, o uso de operadores de junção pode ser preferível a consultas normais e/ou encadeadas sobre múltiplas tabelas, ao permitir uma melhor optimização das consultas ao nível do SGBD e/ou consultas mais sucintas.

Produto cartesiano — CROSS JOIN

SELECT … FROM
<TABELA 1> CROSS JOIN <TABELA 2>
…



- **<TABELA 1> CROSS JOIN <TABELA 2>** devolve **o produto cartesiano de registos das duas tabelas**, portanto expressa **todas as combinações de registos das duas tabelas**.
- **Pouco usado**: normalmente queremos fazer uma junção sujeita a condições. Podemos usar usa cláusula **WHERE** sobre o **CROSS JOIN**, mas serão mais apropriados outros tipos de junção.

CROSS JOIN — exemplo

```
/* Obtém todos os pares título de livro/autor sem
   qualquer correlação. */
SELECT L.Título, A.Nome
FROM LIVRO L CROSS JOIN AUTORES A;
```

Título	Nome
The C Programming Language, 2nd edition	Brian W. Kernighan
Effective Java	Brian W. Kernighan
A Mensagem	Brian W. Kernighan
Sonetos	Brian W. Kernighan
Os Lusíadas	Brian W. Kernighan
Astérix o Gaulês	Brian W. Kernighan
A Volta à Gália de Astérix	Brian W. Kernighan
The C Programming Language, 2nd edition	Dennis M. Ritchie
Effective Java	Dennis M. Ritchie
A Mensagem	Dennis M. Ritchie

. . . etc . . .

Junção interna

SELECT … FROM
<TABELA 1> INNER JOIN <TABELA 2>
ON(<CONDIÇÃO>)
…

- Junção das duas tabelas de acordo com **<CONDIÇÃO>** que especifica como os dados das 2 tabelas se correlacionam.
- Ou seja: **são devolvidos registos das duas tabelas que verificam a condição de correlacionamento <CONDIÇÃO>.**
- **O modificador INNER pode ser omitido** em MySQL e outros SGBDs SQL, é assumido um **INNER JOIN** por omissão quando especificamos apenas **JOIN.**

Junção interna — exemplo

```
SELECT C.ISBN,C.Num,U.Nome,C.EmpData
FROM CÓPIA C JOIN UTENTE U ON(C.EmpUtente = U.Num);
```

ISBN	Num	Nome	EmpData
9789720049759	2	João Pinto	2019-01-17
9789722709620	1	João Pinto	2019-01-17
9789722526289	1	Carlos Semedo	2018-12-28
9789722709620	3	Carlos Semedo	2018-12-27
9789724138695	3	Maria Silva	2018-11-01
9780131103627	2	Pedro Costa	2019-02-01
9789724138695	1	Pedro Costa	2019-02-01
9789720049759	1	Filipa Mendes	2018-11-19
9789722709620	4	Eva Mendes	2018-02-01

Junção interna — exemplo com 3 tabelas

```
SELECT L.Título,C.Num,U.Nome,C.EmpData
FROM CÓPIA C JOIN UTENTE U JOIN LIVRO L
ON(C.EmpUtente = U.Num AND L.ISBN=C.ISBN);
```

Título	Num	Nome	EmpData
A Mensagem	2	João Pinto	2019-01-17
Os Lusíadas	1	João Pinto	2019-01-17
Sonetos	1	Carlos Semedo	2018-12-28
Os Lusíadas	3	Carlos Semedo	2018-12-27
Astérix o Gaulês	3	Maria Silva	2018-11-01
The C Programming Language, 2nd edition	2	Pedro Costa	2019-02-01
Astérix o Gaulês	1	Pedro Costa	2019-02-01
A Mensagem	1	Filipa Mendes	2018-11-19
Os Lusíadas	4	Eva Mendes	2018-02-01

Equi-junção

```
SELECT S.Descrição,
       E.Código AS Estante
FROM SECÇÃO S JOIN ESTANTE E
ON (S.Código = E.Secção);
```

```
+------------------+---------+
| Descrição        | Estante |
+------------------+---------+
| Banda Desenhada  | E14     |
| Literatura       | E12     |
| Literatura       | E13     |
| Livros Técnicos  | E99     |
+------------------+---------+
```

```
SELECT L.Título, A.Nome
FROM LIVRO L JOIN AUTORES A  ON (A.ISBN = L.ISBN);
```

- **Equi-junção ("equi-join")** é um caso particular de junção bastante comum em que a **condição de junção é expressa pela igualdade de atributos (operador =)**, como nos exemplo acima.

Junção natural

```
SELECT L.Título, A.Nome
FROM LIVRO L JOIN AUTORES A  ON (A.ISBN = L.ISBN);

SELECT L.Título, A.Nome
FROM LIVRO L JOIN AUTORES A USING (ISBN);

SELECT L.Título, A.Nome
FROM LIVRO L NATURAL JOIN AUTORES A;
```

- **Junção natural**: Um caso particular de equi-junção, bastante comum, em que **os atributos em causa são os mesmo nas duas tabelas**.
- Podemos usar **USING(ATRIBUTOS)** para nomear esses atributos ou simplesmente **NATURAL JOIN** para usar todos os atributos comuns.

Junção natural — outro exemplo (cont.)

```
SELECT  L.Título,C.Num,U.Nome,C.EmpData
FROM CÓPIA C JOIN UTENTE U ON(C.EmpUtente = U.Num)
        NATURAL JOIN LIVRO L ;
```

Título	Num	Nome	EmpData
A Mensagem	2	João Pinto	2019-01-17
Os Lusíadas	1	João Pinto	2019-01-17
Sonetos	1	Carlos Semedo	2018-12-28
Os Lusíadas	3	Carlos Semedo	2018-12-27
Astérix o Gaulês	3	Maria Silva	2018-11-01
The C Programming Language, 2nd edition	2	Pedro Costa	2019-02-01
Astérix o Gaulês	1	Pedro Costa	2019-02-01
A Mensagem	[illegible]	[illegible]	[illegible]
Os Lusíadas	[illegible]	[illegible]	[illegible]

```
/* Versão anterior */
SELECT L.Título,C.Num,U.Nome,C.EmpData
FROM CÓPIA C JOIN UTENTE U JOIN LIVRO L
ON(C.EmpUtente = U.Num AND L.ISBN=C.ISBN);
```

Semi-junções

```
/* Semi-junção à esquerda. */
SELECT DISTINCT Título
FROM LIVRO JOIN CÓPIA USING(ISBN)
WHERE EmpData IS NOT NULL;

/* Semi-junção à direita. */
SELECT DISTINCT Título
FROM CÓPIA JOIN LIVRO USING(ISBN)
WHERE EmpData IS NOT NULL;
```

- Por vezes junções retêm apenas atributos de uma das tabelas na junção. São chamadas **semi-junções**.
- A junção pode ser "à esquerda" ou "à direita" consoante os atributos são retidos do 1º ou 2º operando da junção.

Outros exemplos de junção interna

```
/* Obtém título, editora e ano de livros de Luís de Camões. */
SELECT  Título, Editora, Ano
FROM   LIVRO NATURAL JOIN AUTORES
WHERE  Nome = 'Luís de Camões'
ORDER BY Título;
```

```
+---------------+-------------+------+
| Título        | Editora     | Ano  |
+---------------+-------------+------+
| Os Lusíadas   | INCM        | 1999 |
| Sonetos       | WOOK 11-17  | 2013 |
+---------------+-------------+------+
```

```
/* Versão anterior nestes slides */
SELECT  L.Título, L.Editora, L.Ano,
FROM  LIVRO L,  AUTORES A
WHERE  A.ISBN = L.ISBN
AND A.Nome = 'Luís de Camões'
ORDER BY L.Título;
```

Outros exemplos de junção interna (cont.)

```
/* Obtém título de livro e nº de cópias por título. */
SELECT  Título, COUNT(*)  AS Cópias
FROM  LIVRO NATURAL JOIN CÓPIA
GROUP BY Título
ORDER BY Cópias, Título;
```

```
/* Versão anterior nestes slides. */
SELECT  L.Título, COUNT(*)  AS Cópias
FROM  LIVRO L, CÓPIA C
WHERE  L.ISBN = C.ISBN
GROUP BY L.Título
ORDER BY Cópias, L. Título;
```

Título	Cópias
Effective Java	1
The C Programming Language, 2nd edition	2
Sonetos	2
A Volta à Gália de Astérix	2
A Mensagem	3
Os Lusíadas	4
Astérix o Gaulês	4

Junções externas — LEFT/RIGHT OUTER JOIN

```
SELECT
  C.ISBN,C.Num,U.Nome,C.EmpData
FROM  CÓPIA C LEFT OUTER JOIN UTENTE U
ON(C.EmpUtente = U.Num);

SELECT
  C.ISBN,C.Num,U.Nome,C.EmpData
FROM  CÓPIA C RIGHT OUTER JOIN UTENTE U
ON(C.EmpUtente = U.Num);
```

- **Junções externas** mantêm todos os registos da junção atribuindo valores **NULL** a atributos de registos **à esquerda ou à direita** que não obedecem à condição de junção.
- Vamos ver a diferença entre os dois exemplos acima.

Junções externas — LEFT OUTER JOIN

```
SELECT
  C.ISBN,C.Num,
   U.Nome,C.EmpData
FROM
  CÓPIA C
LEFT OUTER JOIN
  UTENTE U
ON(C.EmpUtente = U.Num);
```

ISBN	Num	Nome	EmpData
9780131103627	**1**	**NULL**	**NULL**
9780131103627	2	Pedro Costa	2019-02-01
9780321356680	**1**	**NULL**	**NULL**
9789720049759	1	Filipa Mendes	2018-11-19
9789720049759	2	João Pinto	2019-01-17
9789720049759	**3**	**NULL**	**NULL**
9789722526289	1	Carlos Semedo	2018-12-28
9789722526289	**2**	**NULL**	**NULL**
9789722709620	1	João Pinto	2019-01-17
9789722709620	**2**	**NULL**	**NULL**
9789722709620	3	Carlos Semedo	2018-12-27
9789722709620	4	Eva Mendes	2018-02-01
9789724138695	1	Pedro Costa	2019-02-01
9789724138695	**2**	**NULL**	**NULL**
9789724138695	3	Maria Silva	2018-11-01
9789724138695	**4**	**NULL**	**NULL**
9789724138947	**1**	**NULL**	**NULL**
9789724138947	**2**	**NULL**	**NULL**

- Atributos de **UTENTE** (**Nome** no exemplo) aparecem com valor **NULL** para cópias não emprestadas.

Junções externas — RIGHT OUTER JOIN

```
SELECT
   C.ISBN,C.Num,
    U.Nome,C.EmpData
FROM
   CÓPIA C
RIGHT OUTER JOIN
   UTENTE U
ON(C.EmpUtente = U.Num);
```

ISBN	Num	Nome	EmpData
9789720049759	2	João Pinto	2019-01-17
9789722709620	1	João Pinto	2019-01-17
9789722526289	1	Carlos Semedo	2018-12-28
9789722709620	3	Carlos Semedo	2018-12-27
9789724138695	3	Maria Silva	2018-11-01
9780131103627	2	Pedro Costa	2019-02-01
9789724138695	1	Pedro Costa	2019-02-01
9789720049759	1	Filipa Mendes	2018-11-19
9789722709620	4	Eva Mendes	2018-02-01
NULL	**NULL**	**Pedro Simões**	**NULL**

- Atributos de **CÓPIA** aparecem com valor **NULL** para utentes sem cópias emprestadas.

Junções externas — FULL OUTER JOIN

```
SELECT
  C.ISBN,C.Num,
   U.Nome,C.EmpData
FROM
  CÓPIA C
FULL OUTER JOIN
  UTENTE U
ON(C.EmpUtente = U.Num);
```

ISBN	Num	Nome	EmpData
9780131103627	**1**	**NULL**	**NULL**
9780131103627	2	Pedro Costa	2019-02-01
9780321356680	**1**	**NULL**	**NULL**
9789720049759	1	Filipa Mendes	2018-11-19
9789720049759	2	João Pinto	2019-01-17
9789720049759	**3**	**NULL**	**NULL**
9789722526289	1	Carlos Semedo	2018-12-28
9789722526289	**2**	**NULL**	**NULL**
9789722709620	1	João Pinto	2019-01-17
9789722709620	**2**	**NULL**	**NULL**
9789722709620	3	Carlos Semedo	2018-12-27
9789722709620	4	Eva Mendes	2018-02-01
9789724138695	1	Pedro Costa	2019-02-01
9789724138695	**2**	**NULL**	**NULL**
9789724138695	3	Maria Silva	2018-11-01
9789724138695	**4**	**NULL**	**NULL**
9789724138947	**1**	**NULL**	**NULL**
9789724138947	**2**	**NULL**	**NULL**
NULL	**NULL**	Pedro Simões	**NULL**

- Alguns SGDBs SQL suportam também **FULL OUTER JOIN** , que é a composição (união) das junções à esquerda é à direita simultaneamente.

Junções externas — FULL OUTER JOIN (cont.)

```
(SELECT C.ISBN,C.Num,
   U.Nome,C.EmpData
FROM  CÓPIA C
LEFT OUTER JOIN UTENTE U
ON(C.EmpUtente = U.Num))
UNION
(SELECT C.ISBN,C.Num,
   U.Nome,C.EmpData
FROM  CÓPIA C
RIGHT OUTER JOIN UTENTE U
ON(C.EmpUtente = U.Num));
```

ISBN	Num	Nome	EmpData
9780131103627	**1**	**NULL**	**NULL**
9780131103627	2	Pedro Costa	2019-02-01
9780321356680	**1**	**NULL**	**NULL**
9789720049759	1	Filipa Mendes	2018-11-19
9789720049759	2	João Pinto	2019-01-17
9789720049759	**3**	**NULL**	**NULL**
9789722526289	1	Carlos Semedo	2018-12-28
9789722526289	**2**	**NULL**	**NULL**
9789722709620	1	João Pinto	2019-01-17
9789722709620	**2**	**NULL**	**NULL**
9789722709620	3	Carlos Semedo	2018-12-27
9789722709620	4	Eva Mendes	2018-02-01
9789724138695	1	Pedro Costa	2019-02-01
9789724138695	**2**	**NULL**	**NULL**
9789724138695	3	Maria Silva	2018-11-01
9789724138695	**4**	**NULL**	**NULL**
9789724138947	**1**	**NULL**	**NULL**
9789724138947	**2**	**NULL**	**NULL**
NULL	**NULL**	Pedro Simões	**NULL**

- **MySQL não suporta** **FULL OUTER JOIN** mas uma junção deste tipo pode ser obtida via **UNION** envolvendo um **LEFT OUTER JOIN** e um **RIGHT OUTER JOIN**.

Vistas SQL
(“views”)

Vistas SQL (“views”)

- Algumas consultas sobre múltiplas tabelas, empregando junções, consultas encadeadas, etc, permitem dar muitas vezes uma **“vista” conveniente e “amigável ao utilizador” da informação na BD.**
- O SQL suporta o conceito de **vista (“view”)**, que funciona como uma “tabela virtual”.
- **Uma vista pode ser consultada tal e qual uma tabela** e sob certas condições suportar também outras operações como inserções, remoções, ou actualizações.

Criação de vistas — CREATE VIEW

```
CREATE VIEW
   <NOME DA VISTA>(<ATRIBUTO 1>, …, <ATRIBUTO N>)
AS <CONSULTA>;
```

- Tal como uma tabela, uma vista tem um **nome e atributos** associados.
- Os dados são no entanto "virtuais", isto é, obtidos a partir de uma consulta que está associada à vita.
- Vamos ver alguns exemplos.

Criação de vistas — Exemplo

```
CREATE VIEW
   EMPRÉSTIMOS                                  ·····▸  Nome
   (Título, NC, Data, NU, Nome)                 ·····▸  Atributos
AS  (
    SELECT                                      ·····▸  Consulta
    L.Título, C.Num,C.EmpData, C.EmpUtente, U.Nome
    FROM LIVRO L NATURAL JOIN CÓPIA C
                JOIN UTENTE U ON (U.Num = C.EmpUtente)
     );
```

Consulta sobre vistas — exemplos

SELECT * FROM EMPRÉSTIMOS;

Título	NC	Data	NU	Nome
The C Programming Language, 2nd edition	2	2019-02-01	4	Pedro Costa
A Mensagem	1	2018-11-19	5	Filipa Mendes
A Mensagem	2	2019-01-17	1	João Pinto
Sonetos	1	2018-12-28	2	Carlos Semedo
Os Lusíadas	1	2019-01-17	1	João Pinto
Os Lusíadas	3	2018-12-27	2	Carlos Semedo
Os Lusíadas	4	2018-02-01	6	Eva Mendes
Astérix o Gaulês	1	2019-02-01	4	Pedro Costa
Astérix o Gaulês	3	2018-11-01	3	Maria Silva

- Criada uma vista, podemos executar consultas sobre a mesma, tal e qual sobre uma tabela.

Consulta sobre vistas — exemplos (cont.)

```
SELECT Nome, COUNT(*) AS NumEmp
FROM EMPRÉSTIMOS
GROUP BY Nome
ORDER BY Nome;
```

```
+---------------+---------+
| Nome          | NumEmp  |
+---------------+---------+
| Carlos Semedo |       2 |
| Eva Mendes    |       1 |
| Filipa Mendes |       1 |
| João Pinto    |       2 |
| Maria Silva   |       1 |
| Pedro Costa   |       2 |
+---------------+---------+
```

- Criada uma vista, podemos executar consultas sobre a mesma, tal e qual sobre uma tabela.

Alterações usando vistas

```
UPDATE
   EMPRÉSTIMOS
SET
   Data = NULL, NU = NULL
WHERE
        Título = 'A Mensagem'
AND Nome = 'João Pinto';
```

```
CREATE VIEW
   EMPRÉSTIMOS
   (Título, NC, Data, NU, Nome)
AS (
   SELECT
   L.Título, C.Num, C.EmpData, C.EmpUtente,
   U.Nome
   FROM LIVRO L NATURAL JOIN CÓPIA C
    JOIN UTENTE U ON (U.Num = C.EmpUtente)
    );
```

- A actualização regista a devolução de um livro (por título) por parte de um utente (por nome) i.e. de uma cópia do livro 'A Mensagem' por parte do utente 'João Pinto'.
- **Q: Quais os efeitos sobre a vista e as tabelas envolvidas?**

Alterações usando vistas (cont.)

```
UPDATE EMPRÉSTIMOS
SET Data = NULL, NU = NULL WHERE Título = 'A Mensagem'
AND Nome = 'João Pinto';
SELECT * FROM EMPRÉSTIMOS;
```

Título	NC	Data	NU	Nome
The C Programming Language, 2nd edition	2	2019-02-01	4	Pedro Costa
A Mensagem	1	2018-11-19	5	Filipa Mendes
~~A Mensagem~~	~~2~~	~~2019-01-17~~	~~1~~	~~João Pinto~~
Sonetos	1	2018-12-28	2	Carlos Semedo
Os Lusíadas	1	2019-01-17	1	João Pinto
Os Lusíadas	3	2018-12-27	2	Carlos Semedo
Os Lusíadas	4	2018-02-01	6	Eva Mendes
Astérix o Gaulês	1	2019-02-01	4	Pedro Costa
Astérix o Gaulês	3	2018-11-01	3	Maria Silva

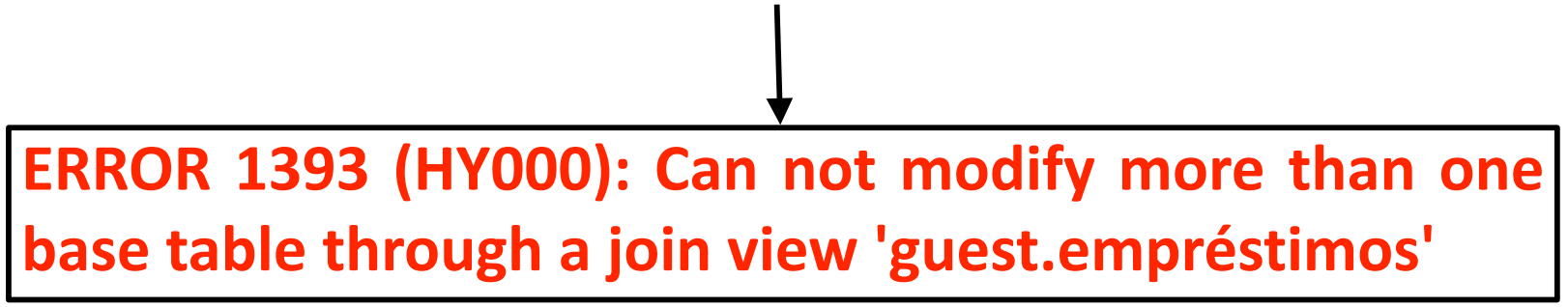
- Em EMPRÉSTIMOS: **registo deixa de constar na vista** — a actualização faz uma "remoção virtual" da entrada ao nível da vista!
- CÓPIA **actualizada**: EmpUtente e EmpData ficam a NULL para a cópia do livro em causa.

Alterações sobre vistas

- Demos apenas exemplo de uma actualização mas podem-se considerar também inserções ou remoções.
- No entanto alterações sobre vistas são sujeitas a várias restrições e o seu suporte pode variar muito entre SGBDs SQL.
- Por exemplo:
  - MySQL permite a actualização anterior porque afecta apenas uma tabela da junção associada à vista (experimente trocar C.EmpUtente por U.Num na consulta da vista e verá que a actualização já não é permitida!).
  - Mas por exemplo vistas resultantes de consultas agregadas ou fazendo uso de DISTINCT não podem ser alteradas. Ver: "Updatable and Insertable Views no manual MySQL.
  - Em alguns SGBDs, por exemplo SQLite, vistas só podem ser lidas (i.e. podemos apenas fazer consultas sobre vistas).

Alterações sobre vistas (cont.)

```
INSERT INTO EMPRÉSTIMOS(Título, NC, Data, NU, Nome)
VALUES('Os Lusíadas', 2, '2019-03-03', 4, 'Pedro Costa');
```



- Registo de um empréstimo?
- Esta alteração, apesar de poder parecer "natural", é ambígua.
- O MySQL não "infere" que a "intenção" é que só uma tabela (**CÓPIA**) seja afectada com uma actualização.

DROP VIEW

DROP VIEW EMPRÉSTIMOS ;

- Uma vista pode ser descartada tal e qual uma tabela com uma instrução **DROP VIEW**.

- No entanto, ao contrário de uma tabela, os dados não são descartados no caso geral. Alguns SGBDs suportam o modificador **CASCADE** para propagar a remoção de dados (MySQL ignora-o).