Universidade do Minho

Ano Letivo: 2022/23
Turnos: PL1/PL8

Bases de Dados

PL07 – Modelação Física

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

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

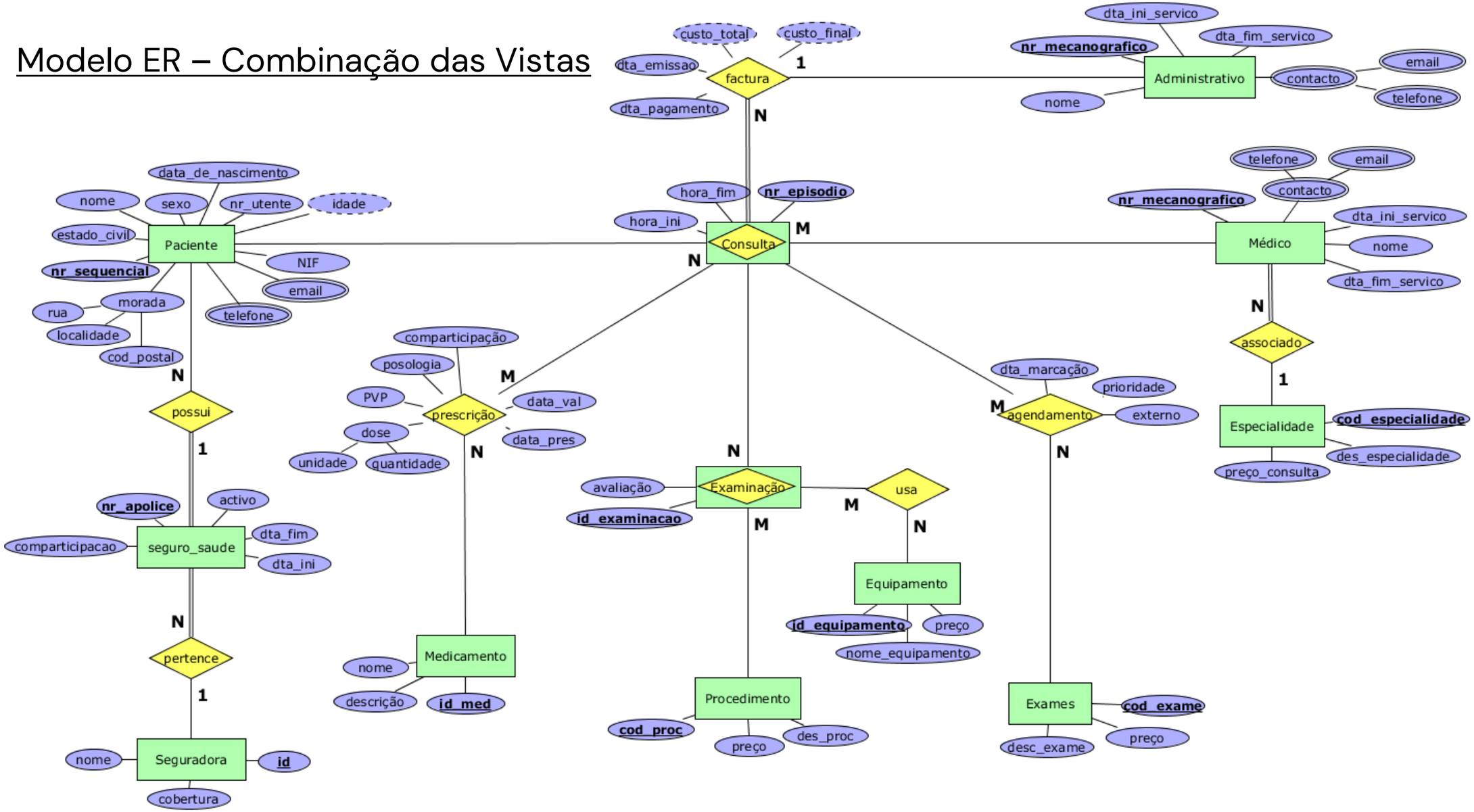
Sumário

Bibliografia:

- Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management , Addison-Wesley, 4a Edição, 2004. **(Chapter 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)**

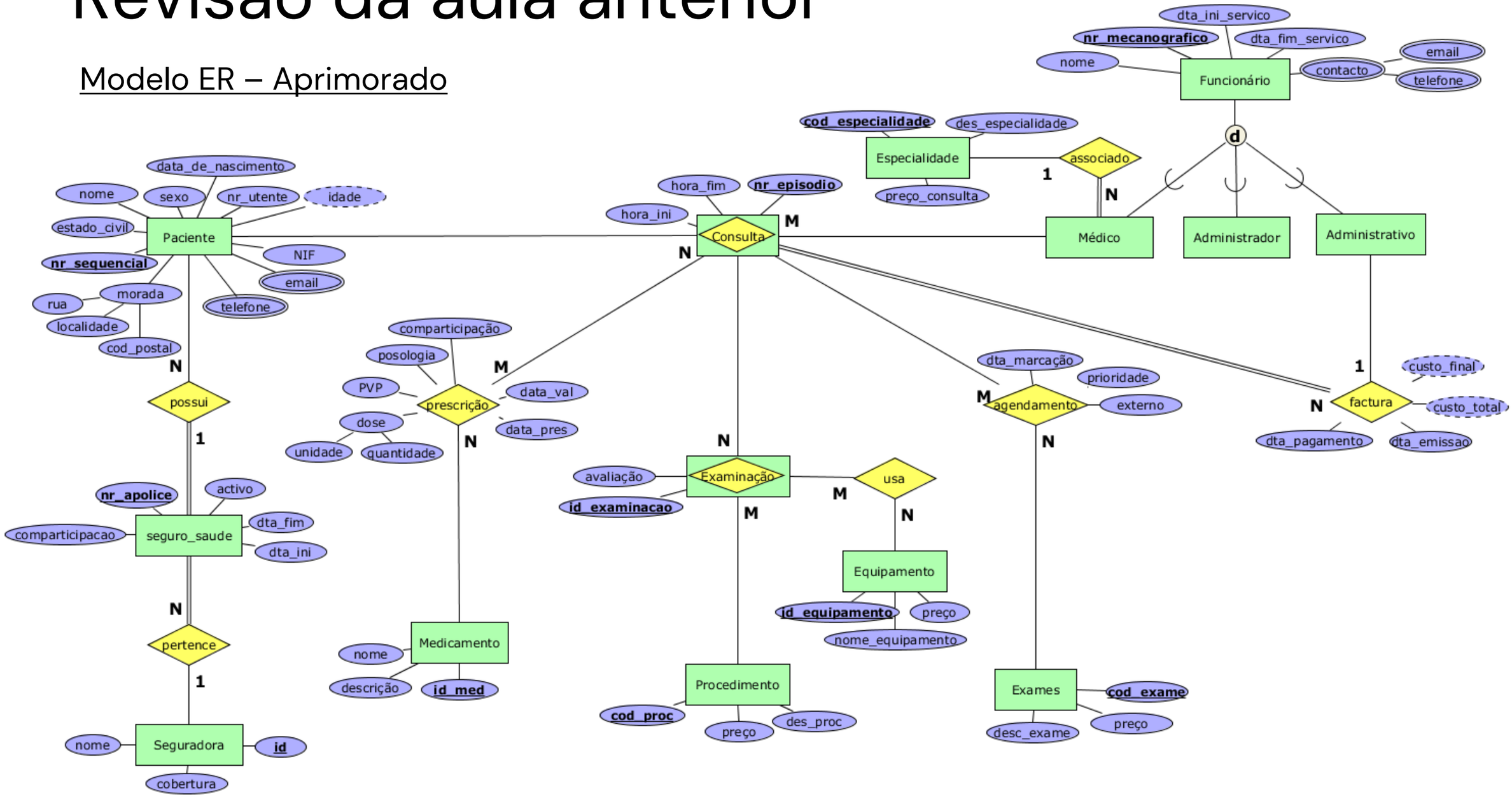
Revisão da aula anterior

Modelo ER – Combinação das Vistas



Revisão da aula anterior

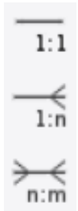
Modelo ER – Aprimorado



Modelação Lógica - MySQL

Quando estamos a construir o modelo lógico de dados no MySQL, é importante ter em consideração os seguintes aspetos:

- Tipo de relacionamento:



Relacionamentos identificadores (linha cheia) Quando a chave primária da entidade pai é incluída na chave primária da entidade filho.

Relacionamentos não identificadores (linha tracejada) Quando a chave primária da entidade pai é incluída na entidade filho, mas apenas como chave estrangeira.

- Chave estrangeira e chave primária.

- Chave estrangeira NOT NULL – participação obrigatória no modelo conceptual

- Chave estrangeira – participação opcional no modelo conceptual

Modelação Lógica - MySQL

- Valores padrão/por defeito: Devem ser usados caso se queira considerar um valor por *default*.

- PK (Primary Key), NN (Not Null), UQ (Unique Index), B (Binary), UN (Unsigned), ZF (Zero Fill), AI (auto increment), G (generated)

  - PK – deve ser usado para atributos que são chave primária;
  - NN – deve ser usado em todos os atributos de chave primária e todos os atributos que não possam ser NULL;
  - UQ - deve ser aplicado sempre que há chaves candidatas, faz com que não hajam valores duplicados na tabela;
  - UN - define que não podem ser inseridos valores negativos nessa coluna.
  - ZF - preenche o valor definido para o campo com zeros até a largura de exibição especificada na definição da coluna.
  - AI – deve ser usado para gerar automaticamente quando um novo registo é inserido numa tabela.
  - G – deve ser usado para gerar atributos a partir de outros usando uma expressão.

Tipos de Dados no MySQL

Dados Alfanuméricos

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

VARCHAR (strings de tamanho **variável**) **vs. CHAR** (strings de tamanho **fixo**)

- O comprimento de dados do tipo CHAR e VARCHAR indica o nº máximo de caracteres que é possível armazenar;
- Os dados do tipo CHAR são preenchidos à direita com **espaços em branco** para o comprimento especificado.

Valor	CHAR(4)		VARCHAR(4)	
''	'____'	4 bytes	''	1 byte
'AB'	'AB__'	4 bytes	'AB'	3 bytes
'ABC'	'ABC_'	4 bytes	'ABC'	4 bytes
'ABCD'	'ABCD'	4 bytes	'ABCD'	5 bytes

O VARCHAR usa 1 ou 2 bytes de memória adicionais para tamanho ou para marcar o fim dos dados.

Para armazenar textos mais longos:
- TEXT
- TINYTEXT
- MEDIUMTEXT
- LONGTEXT

Para armazenar dados JSON:
- JSON

Tipos de Dados no MySQL

Dados Alfanuméricos

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

- O tipo **ENUM** é um objeto de string cujo valor é seleccionado a partir de um conjunto de valores permitidos que são definidos explicitamente no momento de criação da coluna.

EXEMPLO:

prioridade ENUM('Não Urgente', 'Pouco Urgente', 'Urgente', 'Muito Urgente', 'Emergente') NOT NULL);

A coluna prioridade aceitará apenas a inserção de um dos cinco valores definidos. O MySQL mapeia cada membro de enumeração para um índice numérico. Neste caso, 'Não Urgente', 'Pouco Urgente', 'Urgente', 'Muito Urgente' e 'Emergente' são mapeados para 1, 2, 3, 4 e 5 respectivamente.

- O tipo **SET** é um objeto string que pode ter zero ou mais valores, cada um dos quais deve ser escolhido a partir de um conjunto de valores especificados quando a tabela é criada.

EXEMPLO:

tipo SET('A', 'B') NOT NULL);

A coluna tipo aceitará a inserção de '', 'A', 'B' ou 'A,B'. O MySQL armazena valores SET numericamente, com o bit de ordem inferior do valor armazenado correspondendo ao primeiro membro do conjunto.

Tipos de Dados no MySQL

Dados de Data/Hora

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

Tipo de Dados	Notação
DATE	YYYY-MM-DD
TIME	hh:mm:ss
DATETIME*	YYYY-MM-DD hh:mm:ss
TIMESTAMP**	YYYY-MM-DD hh:mm:ss
YEAR	YYYY

* O intervalo suportado varia de '1000-01-01 00:00:00' a '9999-12-31 23:59:59'.
** O intervalo suportado varia de '1970-01-01 00:00:01' a '2038-01-19 03:14:07'.

Tipos de Dados no MySQL

Dados Numéricos

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

- Fixed-Point Types (Exact Value) - DECIMAL, NUMERIC

Os tipos DECIMAL e NUMERIC armazenam valores de dados numéricos exatos. Estes tipos de dados são usados quando é importante preservar a precisão exata, por exemplo, com dados monetários.

DECIMAL(n,m)

n – precisão - representa o número de dígitos significativos que são armazenados.
m – escala - representa o número de dígitos que podem ser armazenados após o ponto decimal.

Exemplo: 105,98€ -> DECIMAL (5,2)

* DEC e FIXED são sinónimos de DECIMAL

FASE 4: Modelação Lógica

Verificar as restrições de integridade

Devem ser considerados os seguintes tipos de restrições de integridade:

- **Restrições de domínio de atributos**: Cada atributo tem um domínio, ou seja, um conjunto de valores que são possíveis. Por exemplo, o sexo de uma pessoa ou é 'M' ou 'F' ou 'I'. Estas restrições deveriam ter sido identificadas quando escolhemos os domínios de atributos para o modelo de dados (Aula 3 - Fase 4).

- **Integridade de entidade:** O valor da chave primária de uma tabela/relação não pode ser "Null" nem igual a outro já existente (caso contrário não conseguiríamos identificar registos). Estas restrições deveriam ter sido consideradas quando identificamos as chaves primárias para cada tipo de entidade (Aula 3 – Fase 5).

- **Restrições gerais/regras de negócio:** As atualizações de entidades podem ser controladas por restrições que regem as transações "do mundo real" que são representadas pelas atualizações. Por exemplo: Uma receita não pode conter mais do que 5 medicamentos.

FASE 4: Modelação Lógica

Verificar as restrições de integridade

- **Integridade referencial**: Um valor definido (diferente de "Null") para um atributo que seja chave estrangeira deve referir-se a uma chave primária da tabela a que a chave estrangeira se refere, ou seja, a uma tupla existente na relação pai. Há duas questões que devem ser abordadas:

  - A primeira considera se os nulos são permitidos para a chave estrangeira. Em geral, se a participação do filho na relação for:
    - obrigatória –> nulos não são permitidos;
    - opcional -> nulos são permitidos.

  - A segunda define como garantir a integridade referencial. Para fazer isso, especificamos restrições de existência que definem as condições sob as quais uma chave estrangeira pode ser inserida, atualizada ou excluída.

    - Inserção ou atualização de uma tupla na relação filha - Para garantir a integridade referencial, verifique se o atributo de chave estrangeira da nova tupla está definido como nulo ou com um valor de uma tupla existente.

FASE 4: Modelação Lógica

Verificar as restrições de integridade

• Remoção de uma tupla da relação pai - Se uma tupla pai é excluída, a integridade referencial é perdida se existir uma tupla filho referenciando a tupla pai. Existem várias estratégias que podemos considerar:

- NO ACTION — Impede a remoção da tupla se houver alguma tupla filho referenciada.

- SET NULL — Quando uma tupla pai é excluída, os valores de chave estrangeira em todas as tuplas filho correspondentes são automaticamente definidos como nulos. Esta estratégia só pode ser aplicada se os atributos que constituem a chave estrangeira aceitarem nulos.

- SET DEFAULT — Quando uma tupla pai é excluída, os valores de chave estrangeira em todas as tuplas filho correspondentes devem ser automaticamente configurados para os seus valores padrão. Esta estratégia só pode ser aplicada se os atributos que constituem a chave estrangeira tiverem valores padrão definidos.

- CASCADE — Quando a tupla pai é excluída, exclui automaticamente todas as tuplas filhas referenciadas. Se qualquer tupla filha excluída atuar como pai noutro relacionamento, a operação de exclusão deverá ser aplicada às tuplas nessa relação filha e assim por diante em cascata.

- NO CHECK — Quando uma tupla pai é excluída, nada é feito para garantir a integridade referencial.

FASE 4: Modelação Lógica

➡ Verificar as restrições de integridade

Paciente				
nr_sequencial	nome	sexo	dta_nascimento	...
323431	Ana Luísa Dias Gomes	F	20/12/1990	
453347	José da Costa Silva	M	03/05/1975	
212423	Maria Leonor Ribeiro Barbosa	Fem	12/07/2000	

✗ Integridade Referencial ?

✗ Integridade de Domínio

Consulta							
nr_episodio	id_pac	id_med	hora_ini	hora_fim	id_agenda	cod_proc	id_sec
12345678	212423	3456	2022-01-23 10:18:17	2022-01-23 10:38:27	123456789	P22	1212
14451643	453347	3224	2022-01-25 08:35:23	2022-01-25 09:00:12	223212434	P23	1598
14451643	212423	3371	2022-02-02 09:00:33	2022-02-02 09:15:20	345567811	NULL	1479
13415324	123456	3834	2022-02-04 12:34:11	2022-02-04 13:00:00	433212456	P22	1234
NULL	323431	3456	2022-02-12 11:20:23	2022-02-12 11:52:33	387612392	P24	1176
...	...	...	...	...	...	...	...

✗ Integridade de Entidade

✗ Integridade de Entidade

Revisão da aula anterior:

Modelo Relacional

Para lidarmos com as restrições de domínio, como por exemplo no caso do sexo e do estado civil, temos 3 opções possíveis:

A) Definir a coluna com o tipo ENUM;

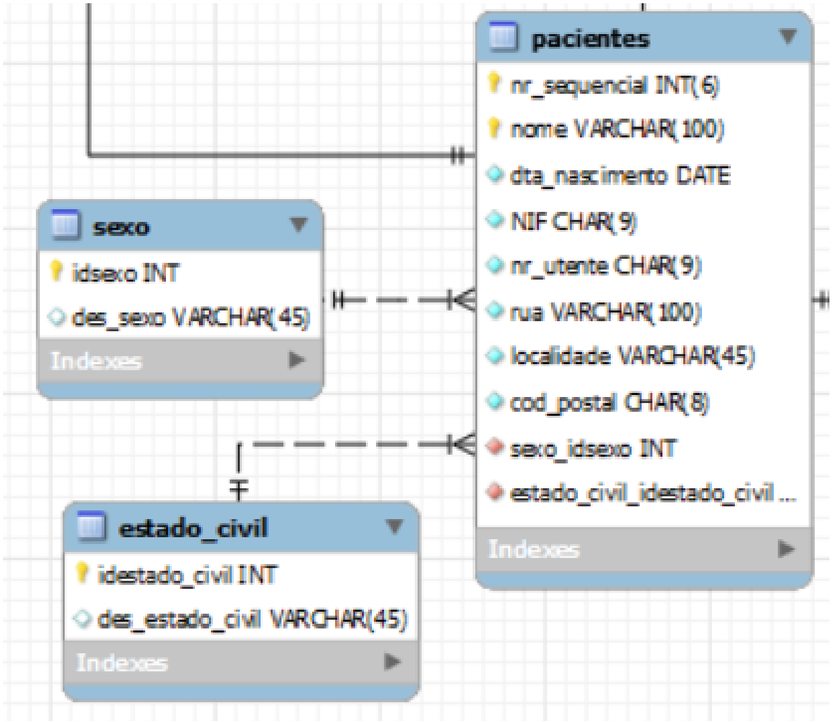
```
CREATE TABLE pacientes (
nr_sequencial  INT NOT NULL AUTO_INCREMENT,
...
sexo ENUM('F', 'M', 'I') NOT NULL,
estado_civil ENUM('S', 'C', 'D', 'V') NOT NULL,
...
);
```

Revisão da aula anterior:

Modelo Relacional

Para lidarmos com as restrições de domínio no caso do sexo e do estado civil temos 3 opções possíveis:

B) Criar uma tabela à parte para definir as opções possíveis;



Revisão da aula anterior:

Modelo Relacional

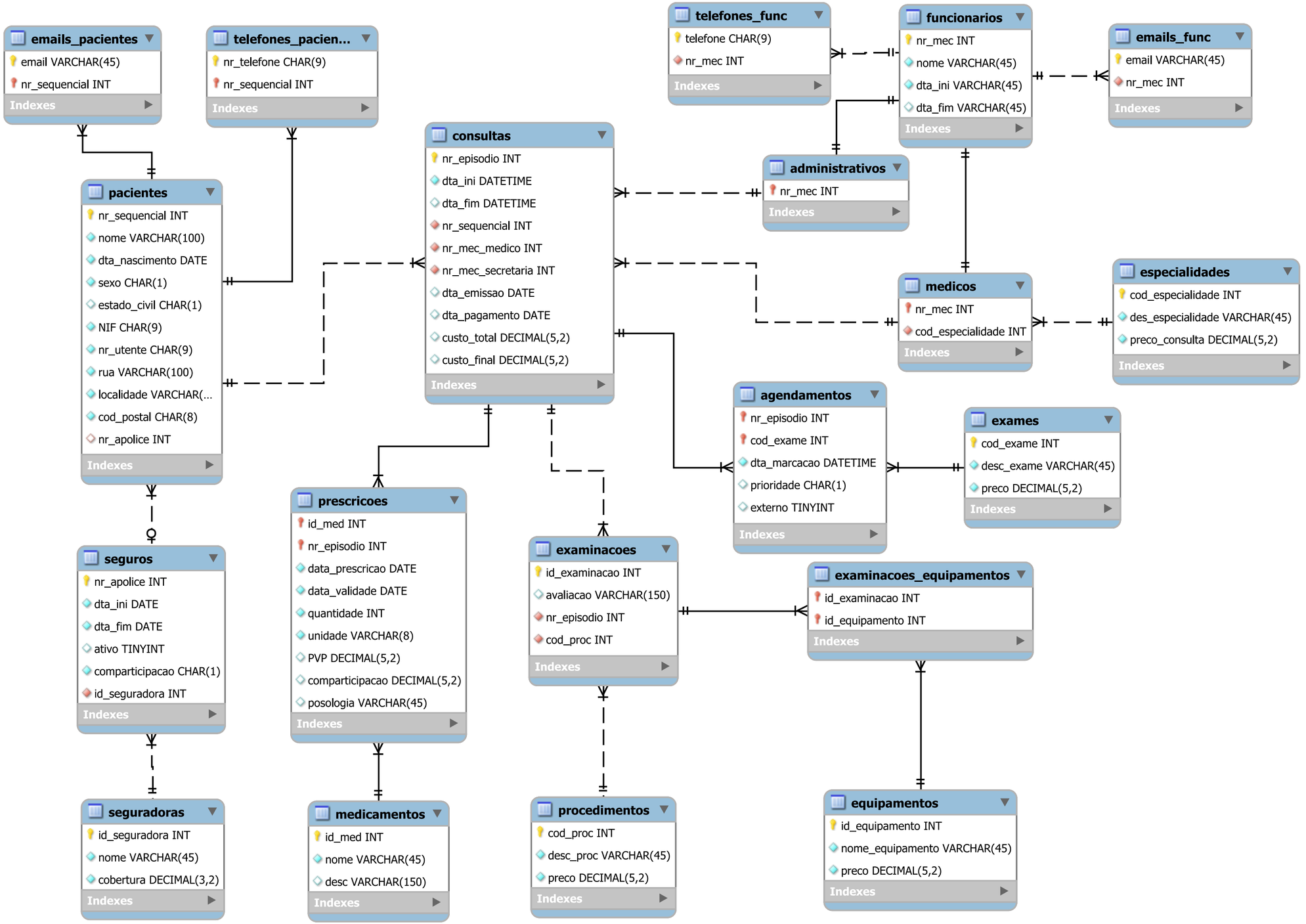
Para lidarmos com as restrições de domínio no caso do sexo e do estado civil temos 3 opções possíveis:

C) Definir a coluna com o tipo CHAR(1) e aplicar *check constrainsts* para limitar os valores inseridos;

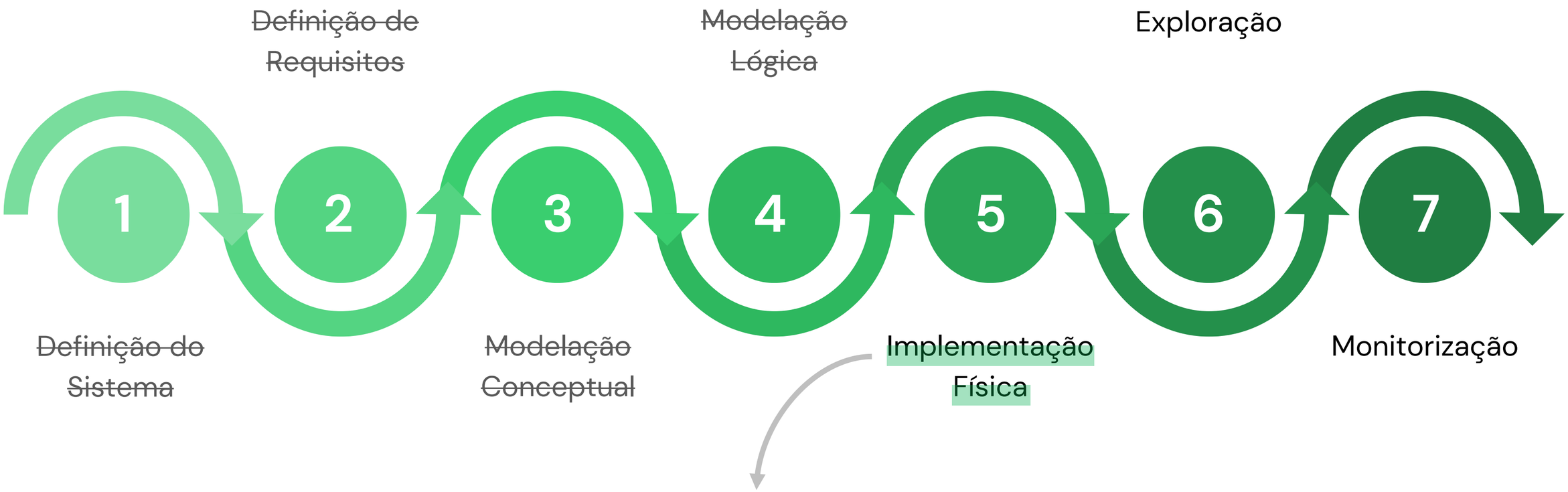
```
CREATE TABLE pacientes (
nr_sequencial  INT NOT NULL AUTO_INCREMENT,
...
sexo CHAR(1) NOT NULL,
estado_civil CHAR(1) NULL,
CONSTRAINT chk_sexo
      CHECK(sexo = 'F' OR sexo = 'M' OR sexo = 'I')
CONSTRAINT chk_estado_civil
      CHECK(estado_civil IN ('S', 'C', 'D', 'V')
);
```

Revisão da aula anterior:

Modelo Relacional



Ciclo de vida de um SBD



Decidir como traduzir o projeto de base de dados lógico (ou seja, as entidades, atributos, relacionamentos e restrições) num projeto de base de dados físico que pode ser implementado usando o SGBD de destino.

FASE 5: Modelação Física

Traduzir o modelo lógico para o SGBD de destino

A **primeira fase** do projeto de BD físico envolve a **tradução** das relações no modelo de dados lógico num formato que possa ser implementado no SGBD relacional de destino.

Esta fase divide-se em:

Fase 1.1 - Representar relações de base

Fase 1.2 - Representar os dados derivados

Fase 1.3 - Representar restrições gerais

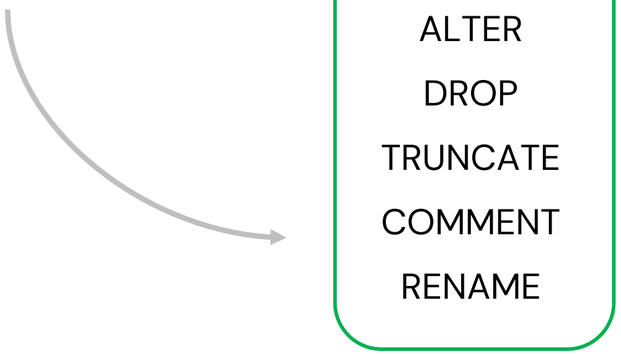
FASE 5: Modelação Física

Data Definition Language (DDL)

Fase 1.1 – Representar relações de base

DDL (Data Definition Language) – Linguagem usada para especificar a informação acerca de cada relação, incluíndo:

- O esquema de cada relação;
- O domínio de valores associados a cada atributo;
- As restrições de integridade;
- O conjunto de índices a manter para cada relação;
- As informações de segurança e autorização para cada relação;
- As estruturas de armazenamento físico em disco de cada relação.



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.

→ para consultar os valores suportados podemos executar a instrução:

SHOW CHARACTER SET;

→ 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)

Exemplo:

```
CREATE TABLE IF NOT EXISTS `Paciente` (
`nr_sequencial` INT(6) NOT NULL AUTO_INCREMENT,
`nome` VARCHAR(100) NOT NULL,
`sexo` CHAR(1) NOT NULL,
 `dta_nascimento` DATE NOT NULL,
 `NIF` CHAR(9) NOT NULL UNIQUE,
`nr_utente` CHAR(9) NOT NULL UNIQUE,
`estado_civil` CHAR(1) NULL,
`rua` VARCHAR(100) NULL,
`localidade` VARCHAR(45) NULL,
`cod_postal` CHAR(8) NULL,
PRIMARY KEY (`nr_sequencial`)
);
```

```
CREATE TABLE IF NOT EXISTS `Paciente` (
`nr_sequencial` INT(6) NOT NULL AUTO_INCREMENT,
`nome` VARCHAR(100) NOT NULL,
`sexo` CHAR(1) NOT NULL,
 `dta_nascimento` DATE NOT NULL,
 `NIF` CHAR(9) NOT NULL,
`nr_utente` CHAR(9) NOT NULL,
`estado_civil` CHAR(1) NULL,
`rua` VARCHAR(100) NULL,
`localidade` VARCHAR(45) NULL,
`cod_postal` CHAR(8) NULL,
PRIMARY KEY (`nr_sequencial`),
UNIQUE KEY (`NIF`),
UNIQUE KEY (`nr_utente`)
);
```

Revisão da aula anterior:

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 pacientes (
nr_sequencial  CHAR(6) NOT NULL,
...
CONSTRAINT chk_nr_sequencial
      CHECK (nr_sequencial 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 pacientes (
nr_sequencial  INT NOT NULL,
...
CONSTRAINT chk_nr_sequencial
      CHECK (LENGTH(nr_sequencial)=6));
```

FASE 5: Modelação Física

Data Definition Language (DDL)

→ mostra informação sobre os elementos do esquema criado (metadados)

{DESC | DESCRIBE} <nome_tabela>;

→ mostra a definição da tabela com o nome especificado

SHOW COLUMNS FROM <nome_tabela>;

→ mostra a instrução de criação da tabela com o nome especificado

SHOW CREATE TABLE <nome_tabela>;

→ mostra a definição das chaves e dos índices de uma tabela

SHOW KEYS FROM <nome_tabela>;

→ apaga a tabela com o nome especificado

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

FASE 5: Modelação Física

Data Definition Language (DDL)

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 .

FASE 5: Modelação Física

Traduzir o modelo lógico para o SGBD de destino

Fase 1.2 – Representar os dados derivados

A decisão entre armazenar um atributo derivado na BD ou calculá-lo sempre que for necessário, deve ter em consideração:

- o custo adicional para armazenar os dados derivados e mantê-los consistentes com os dados operacionais dos quais são derivados;
- o custo para calculá-lo cada vez que for necessário.

No caso de estudo do Hospital Portucalense, existem os seguintes atributos derivados:

- **Idade (Pacientes)**
- **Ativo (Seguros)**
- **Custo total (Consultas)**
- **Custo final (Consultas)**

FASE 5: Modelação Física

Traduzir o modelo lógico para o SGBD de destino

Fase 1.2 – Representar os dados derivados

No caso de estudo do Hospital Portucalense, existem os seguintes atributos derivados:

- **idade (Pacientes)**
- **ativo (Seguros)**

Ambos os atributos dependem do valor da **data atual**. Se decidíssemos armazenar os atributos na BD, estes precisariam de ser atualizados todos os dias para todos os pacientes e seguros do hospital. Por isso, deve ser calculado cada vez que for necessário.

✓ ***Criação de Functions***

- **custo total (Consultas):** preço consulta + preço do procedimento + preço de equipamentos
- **custo final (Consultas):** **se** comparticipação = co-pagamento **então** custo final = custo total * (1-cobertura)
  caso contrário custo final = custo total

Ambos os atributos dependem de valores de outras colunas. Por isso devem ser calculados sempre que os valores das outras colunas forem atualizados, através de **triggers**. Também poderá ser criada uma **view**.

✓ ***Criação de Trigger/View***

FASE 5: Modelação Física

Traduzir o modelo lógico para o SGBD de destino

Fase 1.2 – Representar os dados derivados

Supondo que tínhamos uma tabela sobre vendas com a quantidade, o custo e o valor total (atributo derivado):

- **valor = custo * quantidade**

```
CREATE TABLE IF NOT EXISTS vendas (
id_venda INT NOT NULL AUTO_INCREMENT,
quantidade INT NOT NULL,
custo DECIMAL(5,2) NOT NULL,
valor DECIMAL(5,2) AS (quantidade*custo),
PRIMARY KEY (id_venda)
)
```

FASE 5: Modelação Física

Traduzir o modelo lógico para o SGBD de destino

Fase 1.3 – Representar restrições gerais

As atualizações das relações podem ser limitadas por restrições de integridade que governam as transações do "mundo real". Na Etapa 1.1, projetamos várias restrições de integridade: dados necessários, restrições de domínio e integridade de entidade e referencial. Nesta etapa, é necessário considerar as restrições gerais restantes.

Exemplo: Uma receita não pode conter mais do que 5 fármacos.

- **Instruções para criar regras de negócio/restrições gerais:**

✗ CONSTRAINT MaximoMeds

CHECK (NOT EXISTS (SELECT nr_episodio FROM prescricoes GROUP BY nr_episodio HAVING COUNT(*) >= 5))

Error Code: 1146. Table prescricoes doesn't exist

✗ CONSTRAINT MaximoMeds

CHECK (SELECT COUNT(*) FROM prescricoes GROUP BY nr_episodio) < 5

✓ ***Criação de Trigger***

FASE 5: Modelação Física

Definir organizações de ficheiros e índices

- Um índice é uma estrutura de dados que melhora a velocidade de recuperação dos dados de uma tabela. Os índices podem ser usados para localizar dados rapidamente sem precisar de "varrer" cada linha de uma tabela para uma determinada consulta.
- Quando se cria uma tabela com uma chave primária ou chave candidata (Unique Constraint), o MySQL cria automaticamente um índice chamado PRIMARY.
- Por padrão, o MySQL cria o índice B-Tree se este não for especificado. Os tipos de índices permitidos variam com base no mecanismo de armazenamento da tabela:

Storage Engine	Allowed Index Types
InnoDB	BTREE
MyISAM	BTREE
MEMORY/HEAP	HASH, BTREE

FASE 5: Modelação Física

Data Definition Language (DDL)

→ cria índice

CREATE [UNIQUE] INDEX <nome_índice>

ON <nome_tabela> (<nome_campo> [ASC | DESC],…);

→ mostra os índices de uma tabela

SHOW {KEYS | INDEXES} FROM <nome_tabela>;

→ mostra todos os índices criados sobre as tabelas de uma base de dados

INFORMATION_SCHEMA.STATISTICS

```
SELECT DISTINCT TABLE_NAME, INDEX_NAME
        FROM INFORMATION_SCHEMA.STATISTICS
        WHERE TABLE_SCHEMA = <nome_tabela>;
```

→ apaga índice

DROP INDEX <nome_índice> ON <nome_tabela>

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>];
```

Próxima aula: Exploração da BD