

# SISTEMAS DE BANCO DE DADOS 1

## AULA 7

### Controle de Consistência, Dependência Funcional e Descrição da Relação

Vandor Roberto Vilardi Rissoli



# APRESENTAÇÃO

- Controle de Consistência
- Anomalias
- Dependência Funcional
- Descrição da Relação
- Referências



# Controle de Consistência

No Modelo Relacional, cada esquema de uma relação possui um número de atributos (grau da relação), enquanto que o esquema de um BD relacional consiste de uma série de esquemas de relações (tabelas).

A avaliação de um mapeamento proposto por um projetista, ou resultante de um ME-R, deve respeitar algumas características que o conceitue como um **mapeamento eficiente**.

A indicação de que um esquema relacional está eficiente é identificada quando as relações produzidas representam os dados de forma consistente e com o mínimo possível de redundância controlada.



# Controle de Consistência

No intuito de orientar o projetista na definição e avaliação de um esquema relacional é aplicado um conjunto de critérios informais que o auxiliem, sendo alguns destes critérios apresentados a seguir:

## A) Semântica dos atributos de uma relação

Sempre que um conjunto de atributos é agrupado em uma relação, um determinado significado é associado a eles. Este significado (ou semântica) identifica como devem ser interpretados os atributos pertencentes a cada tupla da relação.

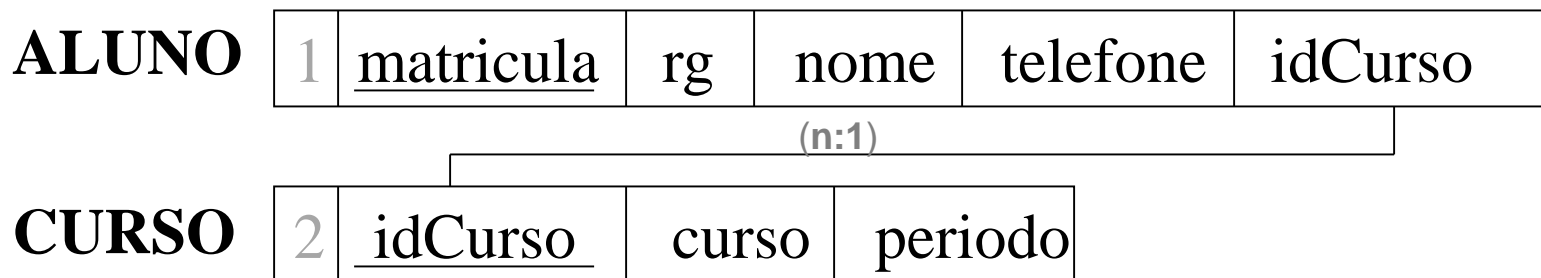
Por exemplo a relação ALUNO\_CURSO representada a seguir:

ALUNO\_CURSO (nome,rg,matricula,telefone,curso,periodo,idCurso)

# Controle de Consistência

O esquema anterior (ALUNO\_CURSO), apesar de representar claramente o seu significado (os alunos que fazem um curso), não é aconselhado, pois possui atributos de duas entidades diferentes sendo armazenados juntos.

A representação mais correta, por meio de esquemas, seria:



# Controle de Consistência

## B) Redundância de valores

Sempre deve-se evitar a definição de dados em mais que um local (esquema) no BD, ou seja, a duplicação (ou redundância) dos dados no BD.

Como exemplo, considere a relação ALUNO\_CURSO contendo os alunos e os seus respectivos cursos. Se mais de um aluno estudar em um mesmo curso, os valores dos atributos do curso deverão aparecer duplicados para cada tupla de aluno matriculado.

Com isso podem ocorrer **algumas anomalias** nas operações DML sobre o BD, tais como:



# Controle de Consistência

## ANOMALIA NO PROCESSO DE INSERÇÃO

Para se cadastrar um novo aluno se deve preencher os atributos do curso que ele faz, ou colocar atributos nulos, quando ele não fizer nenhum curso.

Para possuir um curso sem alunos, torna-se necessário colocar valores nulos para os atributos do aluno, o que significa que a chave da tupla será **NULA**, **violando** a restrição de integridade de chave.



# Controle de Consistência

## ANOMALIA NO PROCESSO DE REMOÇÃO

A necessidade de apagar alunos cadastrados poderá chegar a situação da remoção do último aluno de um determinado curso, sendo então **todos os dados desse CURSO também apagados (perdidos)**, pois não estarão mais guardados no BD em decorrência de não existirem mais alunos matriculados (cadastrados) no referido curso.





# Controle de Consistência

## ANOMALIA NO PROCESSO DE ALTERAÇÃO

A necessidade de se alterar um atributo do CURSO deverá ser realizada em todas as tuplas da relação que possua aluno matriculado no referido curso, caso contrário o **BD se tornará inconsistente**.

O esforço computacional para se processar essa alteração também será maior, comprometendo a eficiência do BD.

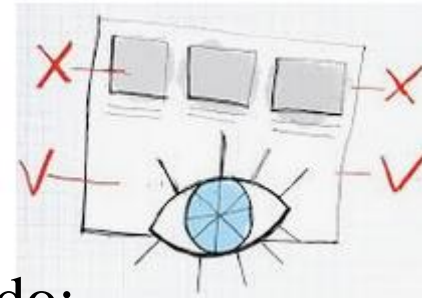


# Controle de Consistência

## C) Valores nulos em tuplas

Um atributo que possua valor **nulo** permite diferentes interpretações, tais como:

- 1- atributo não se aplica para esta tupla;
- 2- valor do atributo para esta tupla é desconhecido;
- 3- valor é conhecido, mas encontra-se ausente (**não foi inserido**).



Sempre procure **evitar atributos** que possam ser **nulos**. Quando for inevitável, verifique se estas tuplas acontecem em pequeno número na relação.

# Controle de Consistência

Uma das importâncias fundamentais na definição de esquemas relacionais é manter a CONSISTÊNCIA dos dados em todo o banco de dados.

O controle de consistência dos dados pode ser exercido **em três níveis:**

- 1) pelo gerenciador (SGBD);
- 2) pelo programa ou aplicativo;
- 3) pela própria construção de um sistema.



De uma forma geral o controle de consistência obtido pela própria construção do sistema é mais eficiente, pois normalmente não implica em perda de desempenho do SGBD durante a sua execução.

# Controle de Consistência

O controle por meio da própria construção do sistema é obtido, no Modelo Relacional, construindo-se as relações seguindo regras que garantem a manutenção de certas propriedades. As relações que atendem a determinadas regras diz-se estarem em uma determinada "**Forma Normal**".

O processo de **Normalização** pode ser considerado como a aplicação de uma série de regras e normas que são denominadas **Formas Normais**.

Algumas dessas regras fundamentais estão baseadas nas

**Dependências Funcionais.**



# Dependência Funcional

Baseia-se no reconhecimento que os valores de alguns atributos contribuem na identificação de outros. Uma dependência funcional significa que ao se conhecer o valor de um atributo, então pode-se sempre determinar o valor de outro. A notação usada na teoria relacional é:

$A \rightarrow B$  onde se lê que "A **determina funcionalmente** B",  
ou que "B **depende de** A".

Exemplo: conhecendo a matrícula de um aluno, então posso determinar o seu nome, ou conhecendo o código de um produto, então posso determinar o seu preço.



# Dependência Funcional

Uma dependência multivalorada significa que ao se conhecer o valor de um atributo, então pode-se sempre determinar os valores de um conjunto de outros atributos. A notação usada na teoria relacional é:

$A \twoheadrightarrow B$  onde se lê "A **determina muitos** B's"

Exemplo: conhecendo um curso posso determinar a lista de seus alunos.

Uma das maneiras de controlar a consistência é por meio das *Dependências Funcionais* existentes entre os atributos armazenados.



# Tipos de Dados

Para iniciar a definição de um BD é necessário conhecer os tipos de dados que o banco pode manipular. De forma geral, serão manipulados os tipos de dados já abordados:

- \* numéricos (-45 | 0 | 25.57)
- \* literais (José Roberto | porcelana | A)
- \* datas (dia, mês e ano | 23 / 12 / 01) - hora

A sintaxe que identifica estes tipos em **ORACLE** são, enquanto no **MySQL** foi abordado na **Aula 6** (*slide 15*):

- \* **number**( $n,d$ ) – numéricos (**int** e outros)
- \* **varchar2**( $n$ ) – caracteres variáveis até 4000 símbolos
- \* **date** – de 1/1/4712 A.C. até 31/12/9999 D.C.

→  $n$  corresponde ao comprimento/tamanho e  $d$  a quantidade de dígitos decimais (existem vários outros tipos em **ORACLE**).

# Dados Obrigatórios

O armazenamento de dados em uma relação, normalmente, necessita de alguns dados obrigatórios, enquanto que outros dados podem ser informados somente para algumas situações.

Exemplo: suponha o cadastro de um aluno na relação abaixo.

**ALUNO(matricula, nome, nascimento, rg, cpf, telefone)**

Para a efetivação da matrícula os dados coletados de um aluno deverão atender aos atributos especificados na relação acima, porém existiram situações em que um aluno ainda não fez o seu documento de CPF e também não possui telefone. **Isso seria motivo para não realizar a matrícula do mesmo?**





# Dados Obrigatórios

A obrigatoriedade de um atributo é identificado pela expressão **NOT NULL** que deve estar vinculada a todos os atributos que são obrigatórios em uma relação (**NÃO NULO** da descrição de esquemas).

No exemplo anterior todas as informações seriam obrigatórias, com exceção do telefone e dependendo das normas de tal escola o CPF.

## Descrição da Relação

Conhecendo os tipos de dados para cada atributo da relação (tabela), e algumas das restrições que as envolvem no momento de sua criação, é possível elaborar a descrição da relação que será **implementada** pela Linguagem SQL.

# Descrição da Relação

Exemplo: a descrição da relação a ser criada no BD **ORACLE**, por meio de uma instrução DDL seria:

```
CREATE TABLE ALUNO (  
    matricula          number(8)          NOT NULL,  
    nome               varchar2(30)       NOT NULL,  
    dtNascimento       date               NOT NULL,  
    rg                 varchar(10)        NOT NULL,  
    cpf                number(11),  
    telefone           number(12)  
);
```

Os dados a serem armazenados nesta relação teriam que atender as restrições informadas de tipo de dado, tamanho e obrigatoriedade.

Note também a pontuação necessária dessa instrução (**azul**).

# Descrição da Relação

Na relação apresentada como exemplo também existirá a identificação da restrição de **integridade referencial**, em que a chave primária simples é o atributo matricula.

Esta restrição (**PK** chave primária) será descrita no final da relação, em que a sintaxe, na forma geral, seria:

**CONSTRAINT <identificador> Primary Key (<atributo>)**

Para o exemplo anterior a restrição PK em SQL seria:

**CONSTRAINT Aluno\_PK Primary Key (matricula)**

Supondo que a restrição desta relação fosse uma chave primária **composta** por matricula e dtNascimento se teria:

**CONSTRAINT Aluno\_PK Primary Key (matricula, dtNascimento)**

# Descrição da Relação

Uma restrição de **integridade referencial** implanta a chave primária que permitirá um relacionamento com outras relações de forma consistente, por meio de uma **chave estrangeira**, também chamada de chave secundária (**FK**).

Este tipo de chave também deve ser identificada de forma clara na descrição da relação que será criada. Observe a sintaxe correta a ser efetuada na relação que possui o atributo (ou atributos) que será descrito como uma **chave estrangeira**.

## Forma Geral

**CONSTRAINT <identificador> Foreign Key (<atributo dessa relação>) REFERENCES <outra relação>(<atributo>)**



→ O uso de vírgulas também é necessário para separar uma atributo do outro, quando as chaves primárias envolvidas forem compostas.

# Descrição da Relação

Continuando o exemplo anterior, suponha a existência da relação CURSO que possui relacionamento com a relação ALUNO, tendo a cardinalidade identificada de que um aluno pode fazer só um curso e que um curso pode possui vários alunos (ALUNO **n : 1** CURSO) matriculados.

A representação dos esquemas em síntese então seriam:

**ALUNO**(matricula, nome, dtNascimento, rg, cpf, telefone, **idCurso**)

**CURSO**(idCurso, nome, periodo)

Estas relações possuem chaves primárias, além de uma chave estrangeira que implementa o seu relacionamento.

Observe a descrição das relações em seguida e a identificação coerente das restrições descritas a seguir.

# Descrição da Relação

Descrição do exemplo anterior de **ALUNO** e **CURSO** em SQL  
MySQL:

```
CURSO (  
    idCurso      int(2)          NOT NULL,  
    nome         varchar(20)     NOT NULL,  
    período      varchar(15)     NOT NULL,  
    CONSTRAINT CURSO_PK Primary Key (idCurso) );
```

```
ALUNO (  
    matricula    int(8)          NOT NULL,  
    nome         varchar(30)     NOT NULL,  
    dtNascimento date            NOT NULL,  
    rg           varchar(10)     NOT NULL,  
    cpf          int(11),  
    telefone     int(12),  
    idCurso      int(2),  
    CONSTRAINT ALUNO_PK Primary Key (matricula),  
    CONSTRAINT ALUNO_CURSO_FK Foreign Key (idCurso)  
    REFERENCES Curso(idCurso) );
```

# Descrição da Relação

A identificação das restrições (*CONSTRAINT*) corresponde a implementação de regras negócios relevantes relacionadas diretamente as tabelas do projeto de BD.

Por isso a implementação coerente, com a identificação adequada e significativa para cada uma dessas regras, deve acontecer em um projeto de BD respeitando as suas normas de definição, sendo as principais:

- Nome da restrição começa com o nome da tabela ao qual a regra de negócio pertence (foi implementada);
- Na chave estrangeira o nome da tabela referenciada vem em seguida e separada por um *underline* ( \_ );
- A identificação termina com a sigla que corresponde a ao tipo de restrição implementada, por enquanto: **PK** => chave primária) e **FK** => chave estrangeira.

# Descrição da Relação

Observe que relação ALUNO possui 3 atributos que NÃO são obrigatórios (cpf, telefone e **idCurso** - chave estrangeira).

A expressão **estrangeira** indica que o atributo (ou atributos para chave composta) são chaves primárias em outra relação, mas não nessa (ALUNO), por isso a ideia de ser uma chave **estrangeira**.

Como idCurso na tabela ALUNO NÃO é chave primaria, então ela não é obrigatória (sem NOT NULL). Porém, se existir a característica para este projeto de que só serão cadastrados nessa tabela alunos matriculados para a escola, então a descrição da tabela ALUNO deverá possuir a **obrigatoriedade** para **idCurso** (então terá NOT NULL).





# Descrição da Relação

Um aspecto relevante que colabora na implementação física de um projeto de BD está relacionado a organização de cada tabela e seus respectivos registros.

Por isso a descrição das relações deve possuir primeiro os atributos mais importantes e obrigatórios, ficando para o final a descrição dos atributos que **NÃO** são obrigatórios.

**ALUNO (**

matricula	int(8)	NOT NULL,
nome	varchar(30)	NOT NULL,
dtNascimento	date	NOT NULL,
rg	varchar(10)	NOT NULL,
<b>idCurso</b>	<b>int(2)</b>	<b>NOT NULL,</b>
cpf	int(11),	
telefone	int(12),	

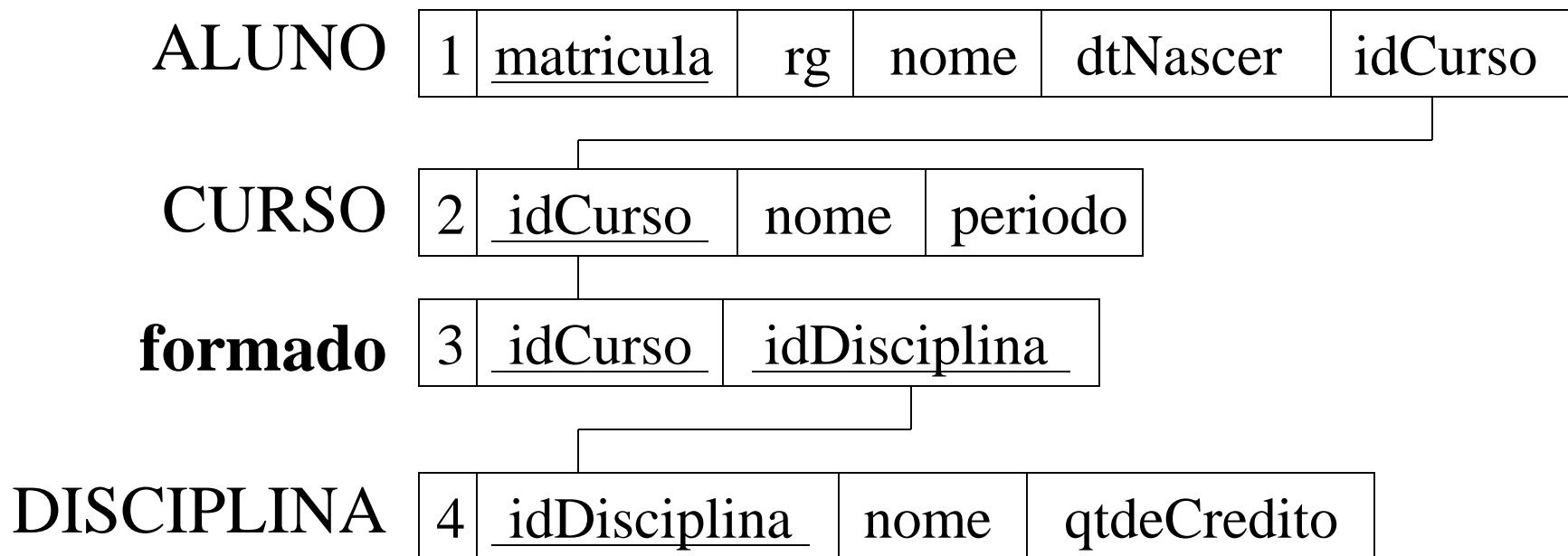
(MySQL)

CONSTRAINT **ALUNO\_PK** Primary Key (matricula),

CONSTRAINT **ALUNO\_CURSO\_FK** Foreign Key (**idCurso**)  
REFERENCES CURSO(idCurso) );

# Exercício de Fixação

1) De acordo com o Diagrama de Esquemas a seguir faça as descrições das relações que serão implementadas neste banco de dados, descrevendo corretamente as restrições de **integridade de entidade, referencial** e colocando como comentário ("--") os valores para a **integridade semântica**, quando ela existir sobre algum atributo.



# Exercício de Fixação

2) A Farmácia **AI-AI** fez a aquisição de alguns equipamentos de informática para conseguir gerenciar melhor os dados que são necessários ao seu funcionamento. Para o desenvolvimento de um banco de dados você foi contratado, e a especificação do problema foi sintetizada a seguir. Os produtos comercializados nesta farmácia são adquiridos de fornecedores únicos para cada produto, apesar de serem diversos os fornecedores (ou fabricantes). Cada fabricante é identificado por meio de seu CGC, nome, endereço completo e único da sede do distribuidor e telefones de distribuição. Os produtos comercializados possuem código de controle, nome comercial, tipo de embalagem, quantidade e preço unitário. A farmácia comercializa dois tipos de produtos: os medicamentos que apresentam uma fórmula e uma tarja e os itens de perfumaria que tem tipo e fragrância.

# Exercício de Fixação

... continuação do exercício 2.

A venda destes produtos também deverá ser acompanhada pelo banco de dados, onde cada venda deverá armazenar a data de compra, número da nota fiscal, nome do cliente, quantidade, preço total e o imposto a ser recolhido.

Somente para os medicamentos é necessário a apresentação da receita médica que terá o CRM do médico, a identificação do medicamento e a data da emissão da receita.

Baseado na especificação anterior elabore o projeto de banco de dados a partir do ME-R até a implementação física dessa solução para farmácia AI-AI. De acordo com o projeto elaborado realize o mapeamento deixando os esquemas coerentes e respeitando todas as principais restrições que devem ser observadas para o funcionamento adequado do Banco de Dados Relacional proposto para solução.



# Referência de Criação e Apoio ao Estudo

## Material para Consulta e Apoio ao Conteúdo

- ELMASRI, R. e NAVATHE, S. B., Fundamentals of Database Systems, Addison-Wesley, 3rd edition, 2000
  - Capítulo 14
- SILBERSCHATZ, A. & KORTH, H. F., Sistemas de Banco de Dados.
  - Capítulo 7
- HEUSER, C. A., Projeto de Banco Dados, 2001.
  - Capítulo 6
- Universidade de Brasília (UnB Gama)
  - <https://sae.unb.br/cae/conteudo/unbfga/>  
(escolha a opção **Sistemas Banco Dados 1** seguida da opção **Mod.Entid.Relacionamento**)