



DISCIPLINA: Banco de Dados 1

Prof. **GIOVANI** Volnei Meinerz

Aula 09 – Normalização

Objetivo da Aula

- Entender o conceito de **normalização** e sua aplicação
 - O que é Normalização?
 - Necessidade da Normalização
- Entender as anomalias
 - Inserção
 - Atualização
 - Exclusão
- Entender o processo e suas etapas de normalização
- Entender o conceito de **grupo de repetição**
- Entender o procedimento de passagem para a **1FN**
- Entender o conceito de
 - **Determinação**
 - **Dependência funcional**
 - **Diagrama de dependências**
- Entender o procedimento de passagem para a
 - Segunda Forma Normal (**2FN**)
 - Terceira Forma Normal (**3FN**)

O Que é Normalização?

É um processo para **avaliar** e **corrigir** estruturas e tabelas de modo a **minimizar** as **redundâncias** de dados, **reduzindo**, assim, a probabilidade de ocorrência de **anomalias**

- Atua por meio de uma série de etapas do processo de normalização, chamadas **Formas Normais**

Necessidade de Normalização

- Consideremos o domínio de negócio de uma empresa de desenvolvimento de software
- A empresa deseja armazenar alguns dados sobre os projetos executados
 - Número do projeto
 - Nome do projeto
 - Funcionários / Empregados que atuam no projeto (M:N)
 - Número do funcionário
 - Nome do funcionário
 - Cargo que ocupa
 - Tarifa (valor da hora, a depender do cargo)
 - Quantidade de horas trabalhadas no projeto

Necessidade de Normalização (cont.)

- ➔ Em um primeiro momento, e sem todos os conhecimentos necessários sobre a abordagem de modelagem relacional, é até aceitável chegarmos na solução a seguir

PROJETO						
PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS

- ➔ No entanto, a tabela encontra-se desnormalizada, podendo apresentar anomalias
- ➔ Vejamos algumas situações que tornam necessário o processo de normalização

Necessidade de Normalização (cont.)

→ Possibilidade de **dados** se tornarem **inconsistentes**

→ Exemplo: o valor de campo de **CARGO**, pode ser inserido como “**Projetista de BD**” ou “**Projetista de Banco de Dados**”

PROJETO

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de Banco de Dados	R\$105,00	56,3

Necessidade de Normalização (cont.)

- ➔ Relatórios podem apresentar **resultados incorretos**
 - ➔ Suponha que tenha sido solicitado um relatório que retorne o total de horas trabalhadas pelo **CARGO** “Projetista de BD”, em todos os projetos

PROJETO

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de Banco de Dados	R\$105,00	56,3

Necessidade de Normalização (cont.)

- A cada nova designação (funcionário a projeto), algumas entradas de dados são repetidas desnecessariamente
- Suponha que a intenção seja designar “Darlene” ao “Evergreen”
“15, Evergeen, 112, Darleni, Analista de Requisitos, R\$75,95, 0,0”

PROJETO

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de BD	R\$105,00	56,3

Necessidade de Normalização (cont.)

- A tabela apresenta redundância de dados, que resultam nas seguintes anomalias: **Inserção, Atualização, Exclusão**

PROJETO

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de BD	R\$105,00	56,3

Anomalias de Inserção

- ➔ **Anomalias de Inserção** – Quando dados não podem ser inseridos, a não ser que outros também sejam
- ➔ Exemplo: ao inserir um novo projeto, tem-se a necessidade de também inserir um funcionário

PROJETO

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de BD	R\$105,00	56,3
100	XYZ	E se ainda não tiver sido designado um funcionário?				

Anomalias de Atualização

- ➔ **Anomalias de Atualização** – modificar o valor de campo de um atributo em um registro pode nos obrigar a fazer a mesma modificação em um mesmo atributo, no entanto em outro registro (para não causar inconsistência)
- ➔ Exemplo – modificar o **valor de campo** do atributo **CARGO** para o funcionário de número **112** pode exigir, potencialmente, muitas alterações, uma para cada “**FUNC_NUM = 112**”

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de BD	R\$105,00	56,3

Anomalias de Exclusão

- ➔ **Anomalias de Exclusão** – dados não podem ser excluídos, a não ser que outros dados também sejam
- ➔ Exemplo – excluir um projeto, fará com que todos os funcionários designados a ele também sejam excluídos

PROJETO

PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
15	Evergreen	102	David	Analista de Sistemas	R\$96,00	23,8
		105	Alice	Projetista de BD	R\$105,00	35,7
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
22	Tide	106	William	Programador	R\$35,75	12,8
		113	Delbert	Projetista de Aplicações	R\$48,10	23,6
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de BD	R\$105,00	56,3

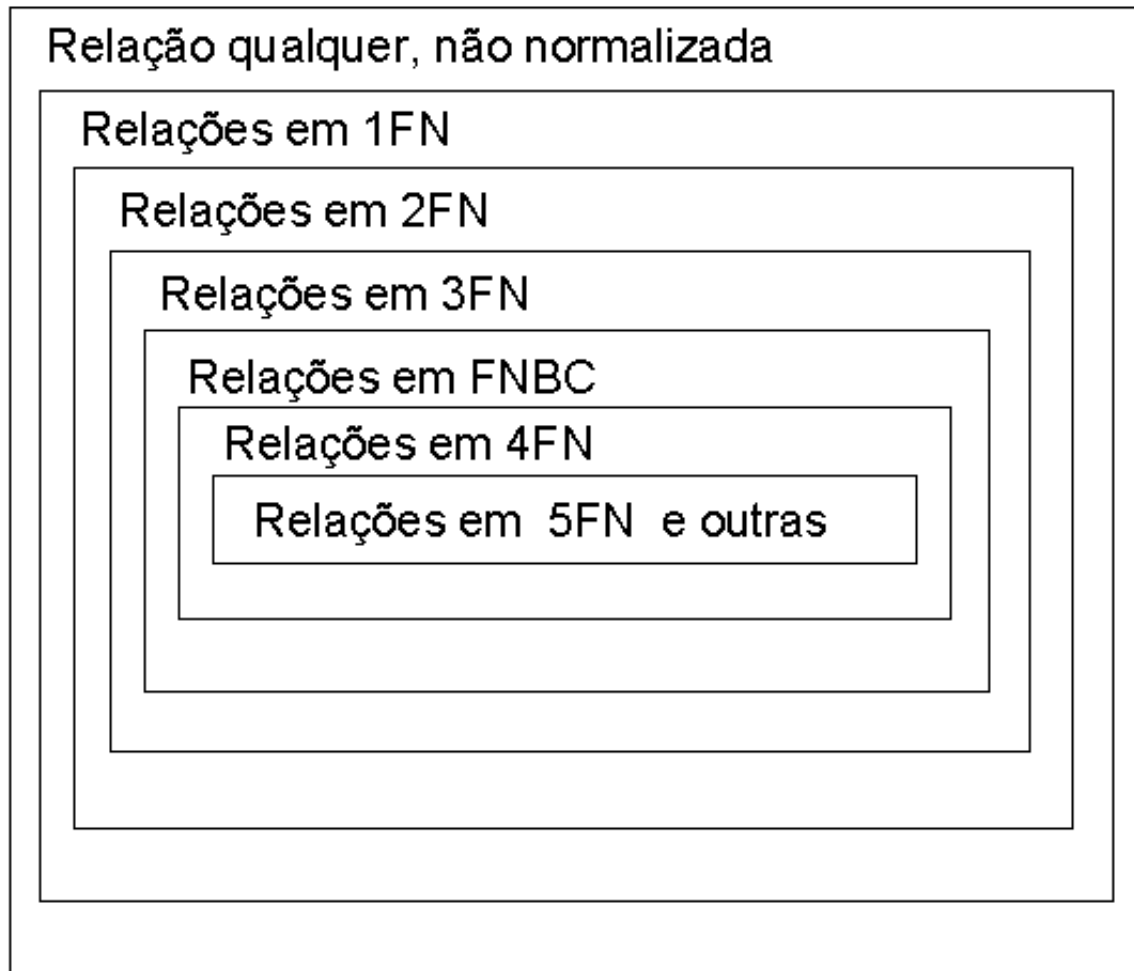
Processo de Normalização

- Baseia-se no conceito de **Forma Normal**, que são regras a serem obedecidas por uma tabela para que esta possa ser considerada bem projetada
- Deve garantir que...
 - ...cada tabela represente algo particular do mundo real
ALUNO, CURSO, DEPARTAMENTO, TURMA, DISCIPLINA,
 - ...nenhum item de dados seja armazenado desnecessariamente
 - ...todas as tabelas estejam livres de anomalias de inserção, atualização e exclusão

Processo de Normalização (cont.)

- Características gerais
- Trabalha em uma relação por vez
- Progressiva separação da relação (tabela) em um conjunto de novas relações (tabelas), baseado nas anomalias e/ou dependências identificadas

Etapas do Processo de Normalização



Para a maioria das finalidades dos projetos de bancos de dados, 3FN é o suficiente

Etapas do Processo de Normalização (cont.)

1FN

Uma relação está na 1FN se, e somente se, todos os seus atributos possuírem apenas **valores atômicos e monovalorados** (simples, indivisíveis).

2FN

Uma relação encontra-se na 2FN se, e somente se, estiver na **1FN** e não possuir **dependência funcional parcial**

3FN

Uma relação está em 3FN se, e somente se, estiver na **2FN** e não possuir **dependência funcional transitiva**

Etapas do Processo de Normalização (cont.)

→ Características gerais

→ Quanto mais alta a forma normal...

→ ...mais operações de junção relacional poderão ser necessárias para produzir a saída desejada, e...

→ ...mais recursos serão exigidos do SGBD para responder a consultas do usuário final

Grupo de Repetição

- Característica que descreve um grupo de **várias** entradas para uma **única ocorrência** de atributo de chave

PROJETO						
PROJ_NUM	PROJ_NOME	FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
18	Amber Wave	118	James	Suporte Geral	R\$18,36	45,3
		112	Darlene	Analista de Requisitos	R\$45,95	45,0
25	Starflight	112	Darlene	Analista de Requisitos	R\$45,95	41,4
		101	John	Projetista de BD	R\$105,00	56,3

- Para visualizarmos melhor o conceito acima, vamos reescrever a tabela

Grupo de Repetição (cont.)

→ Reescrevendo a tabela...

Tabela Aninhada / Coluna Multivalorada / Coluna Não Atômica

PROJETO		
PROJ_NUM	PROJ_NOME	FUNCIONARIO
18	Amber Wave	118, James, Suporte Geral, R\$18,36, 45,3; 112, Darlene, Analista de Requisitos, R\$45,95, 45,0.
25	Starflight	112, Darlene, Analista de Requisitos, R\$45,95, 41,4; 101, John, Projetista de BD, R\$105,00, 56,3.

Uma tabela aninhada é vista como uma tabela que aparece como valor de atributo multivalorado dentro de outra tabela

Grupo de Repetição (cont.)

- Antes da normalização, uma tabela relacional **eventualmente pode apresentar grupos de repetição**
- Grupos de repetição **devem ser eliminados** durante o processo da normalização, pois cada entidade deve representar um tipo particular de algo do mundo real

1FN

Uma relação está na 1FN se, e somente se, todos os seus atributos possuírem apenas **valores atômicos e monovalorados** (simples, indivisíveis).

1FN (cont.)

→ Procedimento em 2 etapas

- ETAPA 1 – Elimine o(s) grupo(s) de repetição, eventualmente existente(s), criando nova(s) tabela(s)
- ETAPA 2 – Identifique as chaves primárias das tabelas

1FN – Etapa 1 (cont.)

→ ETAPA 1 – Elimine os grupos de repetição

→ Em relação a tabela não-normalizada, criar:

Tabela Aninhada / Coluna Multivalorada / Coluna Não Atômica

PROJ_NUM	PROJ_NOME	FUNCIONARIO				
		FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS

→ A tabela (principal/inicial) contendo apenas as colunas com valores atômicos, isto é, sem a(s) tabela(s) aninhada(s)

Nome da tabela: PROJETO

PROJ_NUM	PROJ_NOME
----------	-----------

→ Uma nova tabela para cada tabela aninhada

Nome da tabela: ??? FUNCIONARIO ???

FUNC_NUM	FUNC_NOME	CARGO	TARIFA	HORAS
----------	-----------	-------	--------	-------

1FN – Etapa 2 (cont.)

→ ETAPA 2 – Identifique as chaves primárias

- Em relação a tabela principal/inicial, a chave primária deve ser formada pelo atributo que se constitui como melhor candidato para tal (lembrar de Restrição de Integridade de Chave)

Nome da tabela: PROJETO

PROJ_NUM (PK)	PROJ_NOME
---------------	-----------

- Em relação à nova tabela (a aninhada), sua chave primária deverá ser composta:
 - pelo atributo que se constitui como melhor candidato a chave primária da tabela aninhada, e
 - pelo atributo que forma a chave primária da tabela inicial

Nome da tabela: DESIGNACAO

PROJ_NUM (PK, FK)	FUNC_NUM (PK)	FUNC_NOME	CARGO	TARIFA	HORAS
-------------------	---------------	-----------	-------	--------	-------

1FN – Etapa 2 (cont.)

→ Resultado da conversão para a 1FN

Nome da tabela: PROJETO

PROJ_NUM (PK)

PROJ_NOME

Nome da tabela: DESIGNACAO

PROJ_NUM (PK, FK)

FUNC_NUM (PK)

FUNC_NOME

CARGO

TARIFA

HORAS

Determinação

No contexto de tabelas de bancos de dados relacionais, a afirmação “**A determina B**” indica que conhecer o valor do atributo **A** possibilita verificar/determinar/conhecer o valor de **B**

Determinação (cont.)

→ Exemplo (cont.)

ALUNO	
ra	nome
1	Tilo
2	Hanka
3	Lisa
4	Gerhard

→ Lê-se: **ra** determina **nome**

→ Escreve-se: **ra** → **nome**

**Atributo
Determinante**

**Atributo Dependente
(Determinado)**

Dependência Funcional

- Diz-se que o atributo **B** é funcionalmente dependente do atributo **A** se cada valor da coluna **A** determina um e somente um valor da coluna **B**
- No exemplo, **ra** é conhecido como atributo determinante e **nome** como atributo dependente/determinado
 - Lê-se:
 - **ra** determina funcionalmente **nome**, enquanto que
 - **nome** é funcionalmente dependente de **ra**

ALUNO	
ra	nome
1	Tilo
2	Hanka
3	Lisa
4	Gerhard

Dependência Funcional (cont.)

- ➔ Identificar dependência funcional faz parte da semântica dos dados (exige conhecimento do domínio de negócio)
- ➔ Explicando, por meio do exemplo abaixo...
- ➔ Mesmo que os valores do campo **nome** coincidam, o fato de existir dependência funcional em relação à **ra**, indica que é uma pessoa diferente (homônimo)

ALUNO	
ra	nome
1	Tilo
2	Hanka
3	Lisa
4	Gerhard
5	Hanka

Classificação das Dependências Funcionais

→ Dependência Parcial

- Ocorre quando um atributo é dependente apenas de parte de uma chave primária composta

→ Dependência Transitiva

- Ocorre quando um atributo não depende da chave primária e sim, de um outro atributo não-chave da tabela

Dependência Parcial

→ Aplicação exemplo – Depósitos em conta corrente

→ Identificar se...

- Há chave primária composta?
- Os atributos não chave dependem da chave como um todo, ou de apenas parte dela?

Table: **deposito**

Columns:

<u>idDeposito</u>	int(11) PK
<u>idContaCorrente</u>	int(11) PK
cpfCorrentista	int(11)
dataDeposito	date
valor	decimal(8,2)

→ Análise resultante...

- **idDeposito** não determina **cpfCorrentista**
- **cpfCorrentista** é dependente apenas de **idContaCorrente**
- **dataDeposito** e **valor** dependem da chave como um todo

	idDeposito	idContaCorrente	cpfCorrentista	dataDeposito	valor
	1	380	123	2017-02-28	5000.00
	2	380	123	2017-03-02	150.00
	3	380	123	2017-03-05	780.00
	1	79	456	2017-02-16	3700.00
	2	79	456	2017-02-26	1850.00
	1	211	789	2017-03-21	19000.00

- Logo, em **cpfCorrentista** está caracterizada uma dependência parcial

Dependência Parcial (cont.)

- Descrevendo, abaixo, as dependências funcionais identificadas na tabela **deposito** para facilitar a visualização de sua posterior decomposição

Table: **deposito**

Columns:

<u>idDeposito</u>	int(11) PK
<u>idContaCorrente</u>	int(11) PK
cpfCorrentista	int(11)
dataDeposito	date
valor	decimal(8,2)

	idDeposito	idContaCorrente	cpfCorrentista	dataDeposito	valor
	1	380	123	2017-02-28	5000.00
	2	380	123	2017-03-02	150.00
	3	380	123	2017-03-05	780.00
	1	79	456	2017-02-16	3700.00
	2	79	456	2017-02-26	1850.00
	1	211	789	2017-03-21	19000.00

idDeposito, idContaCorrente → dataDeposito, valor

idContaCorrente → cpfCorrentista

Dependência Parcial (cont.)

- Da decomposição da tabela **deposito** (que é mantida), é criada uma nova tabela: **correntista**

Table: **deposito**

Columns:

<u>idDeposito</u>	int(11) PK
<u>idContaCorrente</u>	int(11) PK
dataDeposito	date
valor	decimal(8,2)

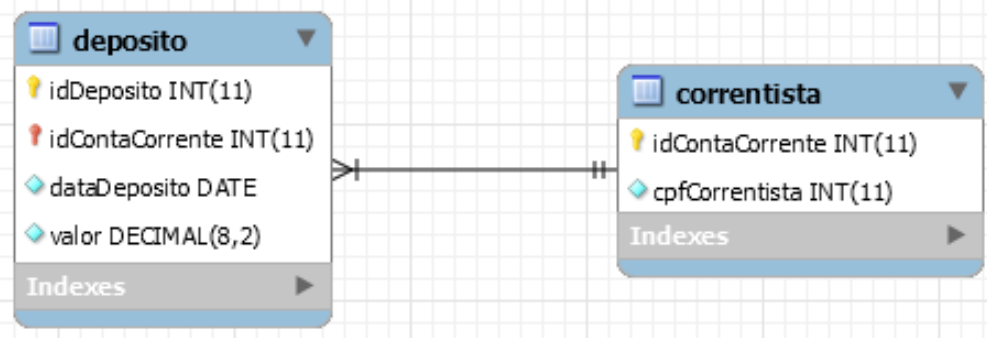
	idDeposito	idContaCorrente	dataDeposito	valor
	1	79	2017-02-16	3700.00
	2	79	2017-02-26	1850.00
	1	211	2017-03-21	19000.00
	1	380	2017-02-28	5000.00
	2	380	2017-03-02	150.00
	3	380	2017-03-05	780.00

Table: **correntista**

Columns:

<u>idContaCorrente</u>	int(11) PK
cpfCorrentista	int(11)

	idContaCorrente	cpfCorrentista
	79	456
	211	789
	380	123



Dependência Transitiva

→ Aplicação exemplo – TCC de aluno

→ Identificar se...

- Há dois ou mais atributos não-chave?
- Há algum atributo não-chave funcionalmente dependente de um outro atributo não-chave?

→ Análise resultante...

- **tccCodigo** não determina **nomeAluno**
- **nomeAluno** é dependente apenas de **raAluno**

Table: **tcc**

Columns:

<u>tccCodigo</u>	int(11) PK
raAluno	int(11)
nomeAluno	varchar(45)

	tccCodigo	raAluno	nomeAluno
	1	123	Tilo
	2	456	Hanka
	3	789	Lisa
	4	12	Tilo

- Logo, em **nomeAluno** está caracterizada uma dependência transitiva

Dependência Transitiva (cont.)

- Descrevendo, abaixo, as dependências funcionais identificadas na tabela **tcc** para facilitar a visualização de sua posterior decomposição

Table: **tcc**

Columns:

<u>tccCodigo</u>	int(11) PK
raAluno	int(11)
nomeAluno	varchar(45)

	tccCodigo	raAluno	nomeAluno
	1	123	Tilo
	2	456	Hanka
	3	789	Lisa
	4	12	Tilo

tccCodigo → raAluno

raAluno → nomeAluno

Dependência Transitiva (cont.)

- Da decomposição da tabela **tcc** (que é mantida), é criada uma nova tabela: **aluno**

Table: **tcc**

Columns:

<u>tccCodigo</u>	int(11) PK
raAluno	int(11)

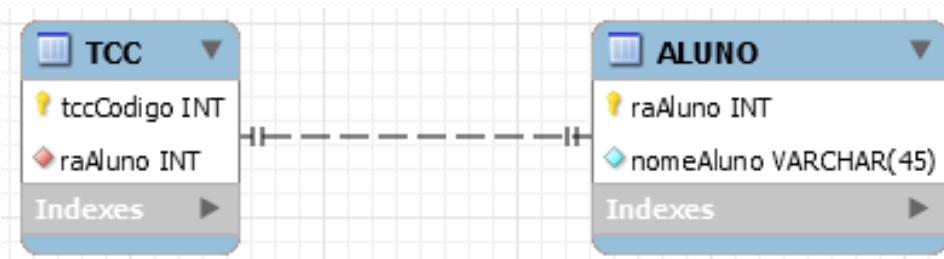
	tccCodigo	raAluno
	1	123
	2	456
	3	789
	4	12

Table: **aluno**

Columns:

<u>raAluno</u>	int(11) PK
nomeAluno	varchar(45)

	raAluno	nomeAluno
	12	Tilo
	123	Tilo
	456	Hanka
	789	Lisa



2FN

Uma relação encontra-se na 2FN se, e somente se, estiver na **1FN** e não possuir **dependência funcional parcial**

2FN (cont.)

- Características a serem observadas antes de iniciar a conversão para a 2FN
 - Se as tabelas resultantes da conversão para a 1FN tiverem chave primária simples, as mesmas já estarão automaticamente na 2FN
 - A conversão para a 2FN é realizada apenas quando alguma tabela, resultante da conversão para a 1FN, possuir chave primária composta e, algum atributo não primário depender de apenas parte dessa chave
 - Como identificar os atributos não primários que dependem de apenas parte da chave primária composta?
 - Requer análise das características da aplicação (regras de negócio)
 - Traçar diagrama de dependências
 - Descrever as dependências

2FN (cont.)

→ Resultado da conversão para a 1FN

1FN – Etapa 2 (cont.)

→ Resultado da conversão para a 1FN

Nome da tabela: PROJETO

PROJ_NUM (PK)	PROJ_NOME
---------------	-----------

Nome da tabela: DESIGNACAO

PROJ_NUM (PK, FK)	FUNC_NUM (PK)	FUNC_NOME	CARGO	TARIFA	HORAS
-------------------	---------------	-----------	-------	--------	-------

25/62

Banco de Dados 1 – Prof. Giovanni



→ Alguma das tabelas possui chave primária composta?

→ Sim! A tabela **DESIGNACAO**

2FN (cont.)

→ Traçando o diagrama de dependências

Nome da tabela: PROJETO

proj_num (pk)	proj_nome
---------------	-----------

**Tabela Normalizada
(não há dependências)**

Nome da tabela: DESIGNACAO

proj_num (pk, fk)	func_num (pk)	func_nome	cargo	tarifa	horas
-------------------	---------------	-----------	-------	--------	-------



— Dependência Parcial
— Dependência Total

→ Descrevendo as dependências

proj_num, func_num → horas

func_num → func_nome, cargo, tarifa

2FN (cont.)

→ Procedimento em 3 etapas

→ ETAPA 1

- Copiar para a 2FN a(s) tabela(s) que tiver(em) chave primária simples e/ou que, em relação as quais, não tenha sido identificada dependência parcial (no caso de tabela que tenha chave primária composta)

→ ETAPA 2

- Caso a tabela com chave primária composta tiver dependência parcial (requer que a dependência já tenha sido identificada), criar uma nova tabela que terá como chave primária a parte da chave composta da tabela original em relação a qual a dependência esteja ocorrendo, parte essa que também deverá permanecer na tabela original

→ ETAPA 3

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

2FN (cont.)

→ ETAPA 1

- Copiar para a 2FN a(s) tabela(s) que tiver(em) chave primária simples e/ou que, em relação as quais, não tenha sido identificada dependência parcial (no caso de tabela que tenha chave primária composta)

Nome da tabela: PROJETO

proj_num (pk)	proj_nome
---------------	-----------

2FN (cont.)

→ ETAPA 2

- Caso a tabela com chave primária composta tiver dependência parcial (requer que a dependência já tenha sido identificada), criar uma nova tabela que terá como chave primária a parte da chave composta da tabela original em relação a qual a dependência esteja ocorrendo, parte essa que também deverá permanecer na tabela original

Nome da tabela: DESIGNACAO

proj_num (pk, fk)	func_num (pk)	func_nome	cargo	tarifa	horas
----------------------	------------------	-----------	-------	--------	-------



— Dependência Parcial
— Dependência Total

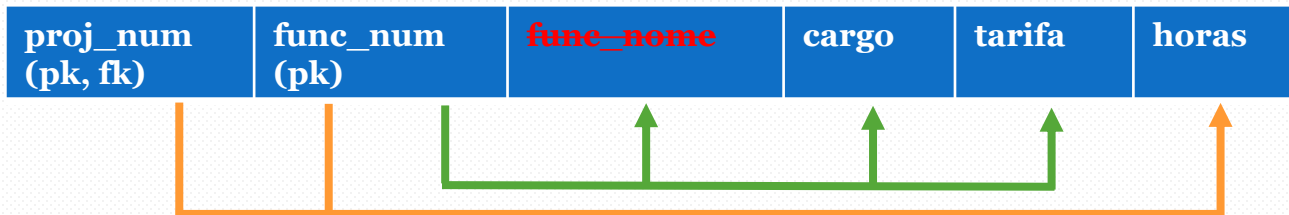
- < Nova Tabela > (func_num (pk), ???, ???, ???, ...)
- < Tabela Original > (proj_num (pk, fk), func_num (pk, fk), ???, ???, ???, ...)

2FN (cont.)

→ ETAPA 3

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

Nome da tabela: DESIGNACAO



— Dependência Parcial
— Dependência Total

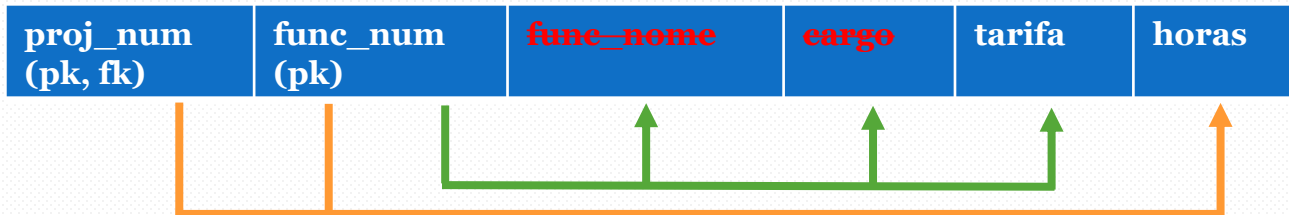
- < Nova Tabela > (func_num (pk), func_nome, ???, ???, ...)
- < Tabela Original > (proj_num (pk, fk), func_num (pk, fk), ???, ???, ???, ...)

2FN (cont.)

→ ETAPA 3 (cont.)

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

Nome da tabela: DESIGNACAO



— Dependência Parcial
— Dependência Total

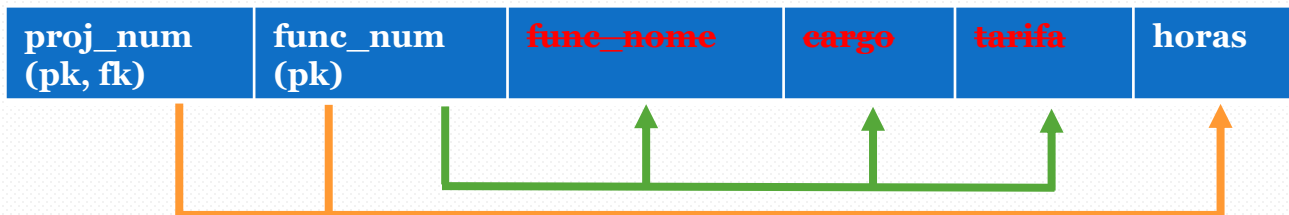
- < Nova Tabela > (func_num (pk), func_nome, cargo, ???, ...)
- < Tabela Original > (proj_num (pk, fk), func_num (pk, fk), ???, ???, ???, ...)

2FN (cont.)

→ ETAPA 3 (cont.)

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

Nome da tabela: DESIGNACAO



— Dependência Parcial
— Dependência Total

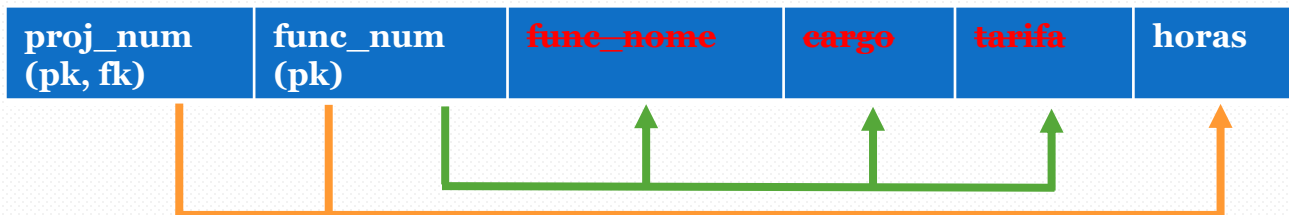
- < Nova Tabela > (**func_num (pk)**, func_nome, cargo, tarifa)
- < Tabela Original > (proj_num (pk, fk), func_num (pk, fk), ???, ???, ???, ...)

2FN (cont.)

→ ETAPA 3 (cont.)

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

Nome da tabela: DESIGNACAO



— Dependência Parcial
— Dependência Total

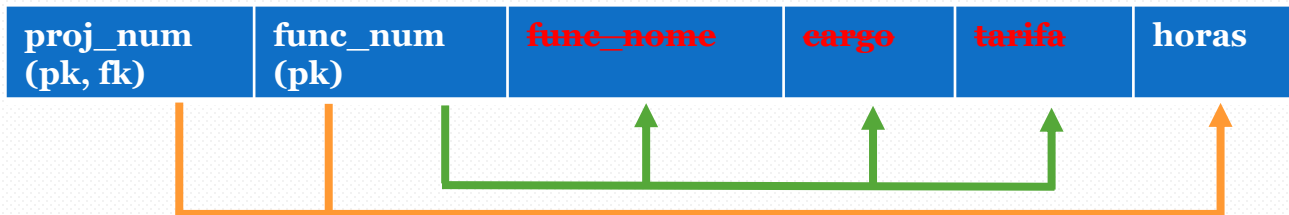
- < Nova Tabela > (**func_num (pk)**, func_nome, cargo, tarifa)
- < Tabela Original > (proj_num (pk, fk), **func_num (pk, fk)**, horas)

2FN (cont.)

→ ETAPA 3 (cont.)

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

Nome da tabela: DESIGNACAO



— Dependência Parcial
— Dependência Total

→ < Nova Tabela > (func_num (pk), func_nome, cargo, tarifa)

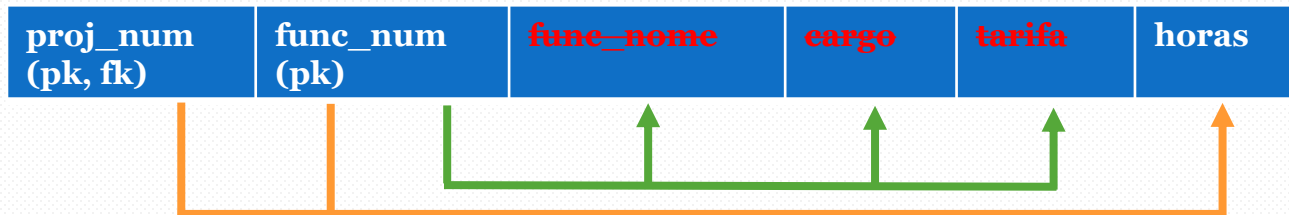
DESIGNACAO (proj_num (pk, fk), func_num (pk, fk), horas)

2FN (cont.)

→ ETAPA 3 (cont.)

- Remover da tabela original todos os atributos dependentes (parciais) que foram identificados, criando-os na nova tabela

Nome da tabela: DESIGNACAO



— Dependência Parcial
— Dependência Total

FUNCIONARIO (**func_num (pk)**, func_nome, cargo, tarifa)

DESIGNACAO (**proj_num (pk, fk)**, **func_num (pk, fk)**, horas)

2FN – Etapa 2 (cont.)

→ Resultado da conversão para 2FN

PROJETO	
proj_num (pk)	proj_nome

FUNCIONARIO			
func_num (pk)	func_nome	cargo	tarifa

DESIGNACAO		
proj_num (pk, fk)	func_num (pk, fk)	horas

2FN (cont.)

- A tabela está na 2FN quando
 - Está na 1FN
 - Não possui dependência funcional parcial

3FN

Uma relação está em 3FN se, e somente se, estiver na **2FN** e não possuir **dependência funcional transitiva**

3FN (cont.)

- ➔ Para estar na 3FN, a relação não deve ter um atributo não-chave funcionalmente dependente de outro atributo não-chave
- ➔ Análise das características da aplicação
 - ➔ Se mantido o cenário abaixo, caso a tarifa por hora for alterada para um cargo mantido por muitos funcionários, essa alteração deverá ser realizada para cada um desses funcionários. Caso esqueça de atualizar alguma tarifa, funcionários diferentes com a mesma descrição de cargo gerarão tarifas horárias diferentes

PROJETO	
proj_num (pk)	proj_nome

FUNCIONARIO			
func_num (pk)	func_nome	cargo	tarifa

DESIGNACAO		
proj_num (pk, fk)	func_num (pk, fk)	horas

Dependência
Transitiva



3FN (cont.)

→ Procedimento em 3 etapas

→ ETAPA 1

- Copiar para a 3FN a(s) tabela(s) que tiver(em) menos que dois atributos não-chave

→ ETAPA 2

- Para todas as dependências transitivas identificadas, apresente o atributo determinante como chave primária de uma nova tabela a ser criada. O atributo determinante, em relação ao qual a dependência esteja ocorrendo, deve permanecer também na tabela original

→ ETAPA 3

- Remover da tabela original todos os atributos dependentes (transitivos) que foram identificados, criando-os na nova tabela

3FN (cont.)

→ ETAPA 1

- Copiar para a 3FN a(s) tabela(s) que tiver(em) menos que dois atributos não-chave

PROJETO	
proj_num (pk)	proj_nome

DESIGNACAO		
proj_num (pk, fk)	func_num (pk, fk)	horas

3FN (cont.)

→ ETAPA 2

- Para todas as dependências transitivas identificadas, apresente o atributo determinante como chave primária de uma nova tabela a ser criada. O atributo determinante, em relação ao qual a dependência esteja ocorrendo, deve permanecer também na tabela original

FUNCIONARIO			
func_num (pk)	func_nome	cargo	tarifa

Dependência
Transitiva

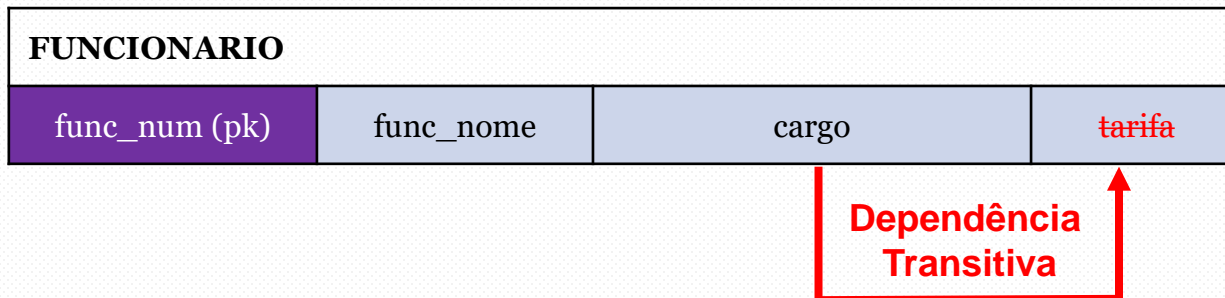


- < Nova Tabela > (cargo (pk), ???, ...)

3FN (cont.)

→ ETAPA 3

- Remover da tabela original todos os atributos dependentes (transitivos) que foram identificados, criando-os na nova tabela




- < Nova Tabela > (**cargo (pk)**, tarifa)

3FN (cont.)

→ ETAPA 3

- Remover da tabela original todos os atributos dependentes (transitivos) que foram identificados, criando-os na nova tabela

FUNCIONARIO			
func_num (pk)	func_nome	cargo	tarifa



A red L-shaped arrow points from the 'cargo' attribute to the 'tarifa' attribute, with the text 'Dependência Transitiva' written in red inside the corner of the arrow.

- CARGO (**cargo (pk)**, tarifa)

3FN (cont.)

- Resultado da conversão para a 3FN
 - Os nomes das tabelas devem refletir seu conteúdo e função

PROJETO	
proj_num (pk)	proj_nome

FUNCIONARIO		
func_num (pk)	func_nome	cargo (fk)

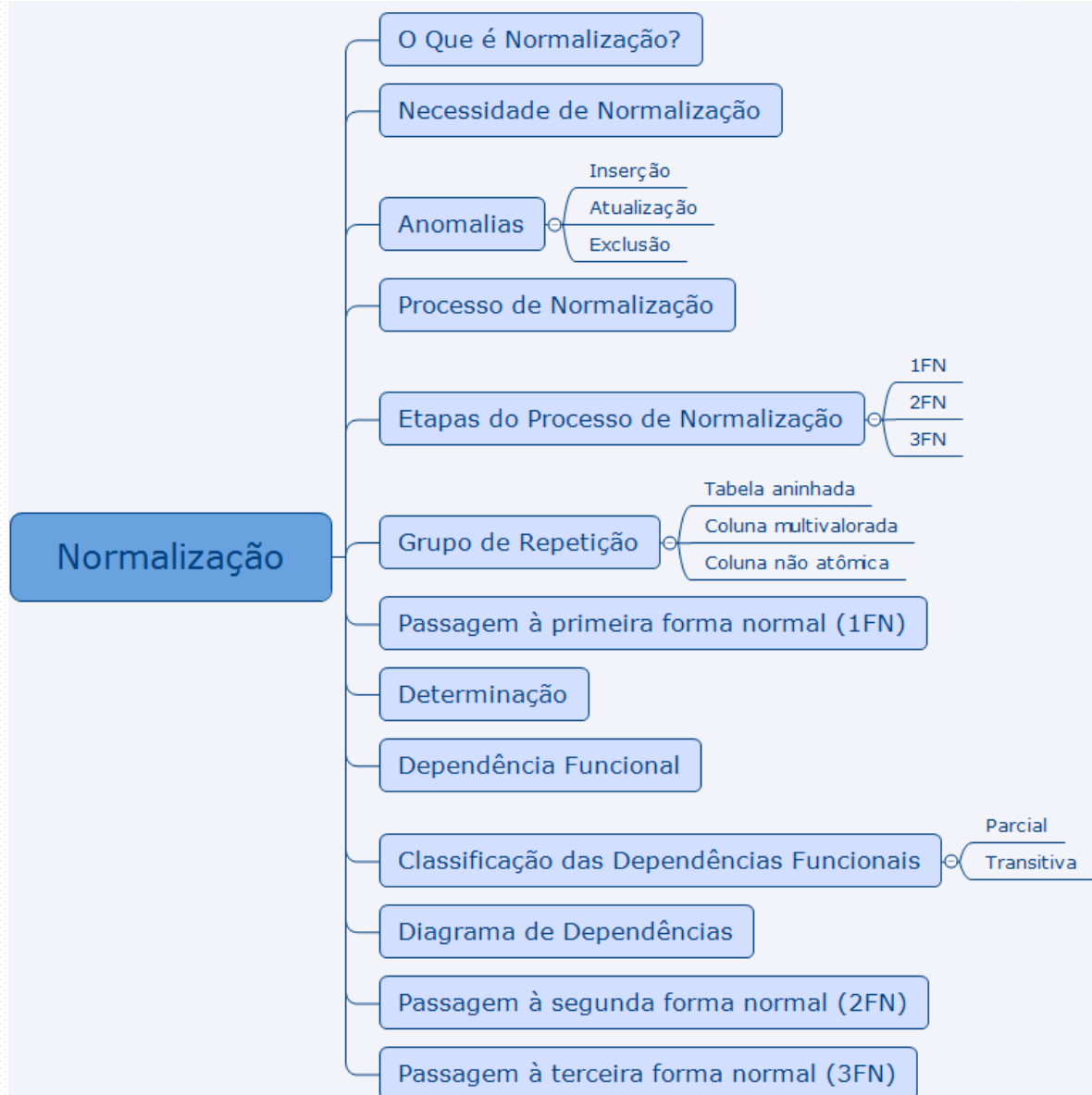
CARGO	
cargo (pk)	tarifa

DESIGNACAO		
proj_num (pk, fk)	func_num (pk, fk)	horas

3FN (cont.)

- A tabela está na 3FN quando
 - Está na 2FN
 - Não possui dependência funcional transitiva

Resumo da Aula





DISCIPLINA: Banco de Dados 1

Prof. **GIOVANI** Volnei Meinerz

Aula 09 – Normalização