Banco de Dados I
Unidade 3: Projeto de BD Relacional

Cláudio Baptista

4.1 Transformação de Diagramas MER em Diagramas DR

- Principais conceitos do MER:
  - Tipos de entidades (regular, fraca)
  - Graus de relacionamentos (binário, n-ário)
  - Atributos (simples, compostos, multivalorados)
  - Restrições (chave, cardinalidade, etc.)

- A seguir, mostraremos um conjunto de regras para efetuar o mapeamento entre modelo ER e modelo Relacional

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra 1: Entidades Regulares**
  - 1.1. Para cada entidade regular E no esquema E-R, criamos uma relação R que inclui os atributos simples de E.
  - 1.2. Para cada atributo composto de E incluímos somente os seus atributos simples.
  - 1.3. Escolhemos um dos atributos chaves de E para ser a chave primária de R.

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra 2: Entidades Fracas**
  - 2.1. Para cada entidade fraca W, com entidade forte E, no esquema E-R, criamos uma relação R e incluímos todos os atributos simples de W como atributos de R.
  - 2.2. Incluímos como atributos da chave estrangeira de R os atributos que compõem a chave primária da entidade forte E.
  - 2.3. A chave primária de R é a combinação da chave primária da entidade forte E e a chave da entidade fraca W.

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra 3: Relacionamentos 1:1**
  - 3.1. Identificamos as relações S e T que correspondem às entidades que participam do relacionamento.
  - 3.2. Escolhemos uma das relações, digamos S, e incluímos como chave estrangeira em S a chave primária de T. É melhor escolher para desempenhar o papel de S, a entidade que tenha participação total no relacionamento.
  - 3.3. Incluímos todos os atributos simples do relacionamento 1:1 como atributos de S.

3.1 Transformação de Diagramas MER em Diagramas DR

- **<u>Regra 4</u>: Relacionamentos 1:N que não envolvem entidades fracas**
  - 4.1. Identificamos a relação S que representa a entidade que participa do lado N do relacionamento.
  - 4.2. Incluímos como chave estrangeira em S, a chave primária da relação T que representa a outra entidade (lado 1) que participa do relacionamento.
  - 4.3. Incluímos qualquer atributo simples do relacionamento 1:N em S.

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra 5: Relacionamento N:M**
  - 5.1. Criamos uma nova relação S para representar o relacionamento.
  - 5.2. Incluímos como chave estrangeira em S as chaves primárias das relações que participam do relacionamento. A combinação destas chaves formará a chave primária da relação S.
  - 5.3. Incluímos qualquer atributo do relacionamento N:M em S.
  - Podemos mapear o relacionamento 1:1 ou 1:N de maneira similar ao M:N. Isto é usado quando poucas instâncias do relacionamento existe, evitando valores nulos nas chaves estrangeiras.

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra 6: Atributos Multivalorados**
  - 6.1. Criamos uma nova relação R que inclui o atributo multivalorado A mais a chave primária K da relação que representa a entidade (ou relacionamento) que tem A como atributo.
  - 6.2. A chave primária de R é a combinação de A e K.
  - 6.3. Se o atributo multivalorado é composto => incluir seus componentes atômicos

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra7: Especialização/Generalização**
  - 7.1. Converta cada especialização com m subclasses {S1,S2,...,Sm} e superclasse C, cujos atributos são {k, a1,..., an} onde k é a chave primária, em esquemas de relações usando uma das seguintes opções:

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra7: Especialização/Generalização**
  - A) Criar uma relação L para C com os atributos Atrib(L) = {k,a1, ... , an} e chave primária k. Criar também uma relação Li para cada subclasse Si, 1 <= i <= m, com os seguintes atributos: Atrib(Li) = {k} ∪ { atributos de Si}, k será a chave primária.
  - Ex.:Empregado(Matrícula,Nome, Salário,Endereço,TipoTrab), Secretária(Matr, VelocidadeDigitação), Técnico(Matrícula, Especialidade), Engenheiro(Matrícula, Tipo, CREA)

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra7: Especialização/Generalização**
  - B) Criar uma relação Li para cada subclasse Si, 1 <= i <= m, com os atributos Atrib(Li) = {atributos de Si} ∪ {k,a1,...,an} e chave primária (Li) = k.
  - Ex.:
    Carro (Identificação, Licença, Preço, VelMax,NumPassag),
    Caminhão(Identificação, Licença, Preço, NumEixos, Tonelag)

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra7: Especialização/Generalização**
  - C) Criar uma única relação L com atributos
    Atrib(L) = {k,a1,...,an} ∪ { atributos de S1} ∪ ... ∪ {atributos de Sm} ∪ {t} e chave primária k.
  - Onde t é um atributo de tipo que indica a subclasse a qual a tupla pertence. (opção usada para especialização cujas subclasses são disjuntas)
  - Ex.: Empregado(Matrícula, Nome, Salário, Endereço, TipoTrab, VelDatilog, EspTec, TipoEng, CREA)

3.1 Transformação de Diagramas MER em Diagramas DR

- **Regra7: Especialização/Generalização**
  - D) Criar uma única relação L com atributos Atrib(L) = {k,a1,...,an} ∪ { atributos de S1 } ∪ ... ∪ { atributos de Sm } ∪ {t1,t2,...,tm} e chave primária k.
  - Onde cada ti , 1 <= i <= m, é um atributo booleano que indica se uma tupla pertence a uma subclasse Si. (opção usada para especialização cujas subclasses são sobrepostas)
  - Ex.:Peça(Código,Descrição,MFLag,NDesenho,DataManufat,NLote,CFlag, Fornecedor, Preço)

3.2 Qualidade de Esquemas Relacionais: Normalização

- A normalização é necessária (embora não suficiente) a um bom projeto relacional. Felizmente, um bom projeto de um esquema de entidades, e sua consequente conversão para um esquema relacional, segundo as regras vistas, praticamente deixa o esquema relacional *normalizado*.
- Assim, utiliza-se a normalização somente para *validar* um projeto relacional.
- Para entender o que a normalização significa, vamos dar primeiramente um exemplo de motivação.

3.2 Qualidade de Esquemas Relacionais: Normalização

HABILIDADES-ESPORTIVAS

Identidade	*Nome*	*Endereço*	*Habilidade*
8795835	Édson Arantes	Ponta da Praia	Futebol
8795835	Édson Arantes	Ponta da Praia	Voleibol
8795835	Édson Arantes	Ponta da Praia	Basquete
8795835	Édson Arantes	Ponta da Praia	Atletismo
8795835	Édson Arantes	Ponta da Praia	Tênis

Esta tabela está mal projetada!

1) Se Pelé mudar de endereço ? (*anomalia de atualização*)
2)Um novo esporte para Pelé ? (*anomalia de inclusão*)
3) Retirar Pelé do Banco de Dados (*anomalia de remoção*)

3.2 Qualidade de Esquemas Relacionais: Normalização

Idealmente:

HABILIDADES-ESPORTIVAS

Identidade	*Nome*	*Endereço*	*Habilidade*
8795835	Édson Arantes	Ponta da Praia	{Futebol, Voleibol, Basquete, Atletismo, Tênis}

Mas isto não é uma tabela (atributo *habilidade* não é atômico)! O que é possível fazer, dentro do modelo relacional?

3.2 Qualidade de Esquemas Relacionais: Normalização

ESPORTISTAS

Identidade	*Nome*	*Endereço*
8795835	Édson Arantes	Ponta da Praia
...	...	...

HABILIDADES

Identidade	*Esporte*
8795835	Futebol
8795835	Voleibol
8795835	Basquetebol
8795835	Atletismo
8795835	Tênis

A repetição da coluna Identidade é uma redundância necessária

3.2 Qualidade de Esquemas Relacionais: Normalização

- **3.2.2 Primeira Forma Normal (1FN)**
  - Toda tabela deve ser “minimamente” normalizada (1FN).
  - Tabela em 1FN: O valor de uma coluna de uma tabela é indivisível.

3.2 Qualidade de Esquemas Relacionais: Normalização

Ex.: Empregado

Matrícula	*Nome*	*Cod Cargo*	*NomeCargo*	*CodProj*	*DataFim*	*Horas*
120	João	1	Programador	01	17/07/95	37
120	João	1	Programador	08	12/01/96	12
121	Hélio	1	Programador	01	17/07/95	45
121	Hélio	1	Programador	08	12/01/96	21
121	Hélio	1	Programador	12	21/03/96	107
270	Gabriel	2	Analista	08	12/01/96	10
270	Gabriel	2	Analista	12	21/03/96	38
273	Silva	3	Projetista	01	17/07/95	22
274	Abraão	2	Analista	12	21/03/96	31
279	Carla	1	Programador	01	17/07/96	27
279	Carla	1	Programador	08	12/01/96	20
279	Carla	1	Programador	12	21/03/96	51
301	Ana	1	Programador	12	21/03/96	16
306	Manoel	3	Projetista	17	21/03/96	67

3.2 Normalização

- A chave primária para a tabela empregados é (Matrícula,CodProj)

- Vimos que um dos objetivos da normalização é reduzir a redundância de dados, porém com a tabela anterior aumentamos a redundância ?!?!

- Precisamos realizar outros passos de normalização para termos um bom projeto.
- A 1FN possui características indesejáveis!

3.2 Normalização

- **Anomalias da 1FN**

- Inserção: não podemos inserir um empregado sem que este esteja alocado num projeto, nem um projeto sem que haja um empregado trabalhando nele (integridade de entidade).

-

3.2 Normalização

- **Anomalias da 1FN**
- Remoção: se precisarmos remover um projeto, as informações de empregados que estiverem lotados apenas naquele projeto serão perdidas.
- Atualização: se um empregado for promovido de cargo teremos que atualizar os atributos CodCargo e NomeCargo em todas as tuplas nas quais aquele empregado está presente.

3.2 Normalização

- *Conclusão:*

- Uma tabela em 1FN não evita, porém, anomalias de inclusão, atualização, e remoção. É preciso uma normalização mais "fina" , ou outras formas formas normais:
  - Segunda Forma Normal (2FN)
  - Terceira Forma Normal (3FN)
- Esta normalização "fina" utiliza o conceito de *dependência funcional*

3.2.3 Dependências Funcionais

- A → B, lê - se:
  - A *funcionalmente determina* B
  - B é *funcionalmente dependente* de A
  - B é *função* de A

- Para cada valor de A só existe um valor de B.

  - A $\neg\rightarrow$ B, negação de A → B.

3.2.3 Dependências Funcionais

- A ou B podem ser um conjunto de atributos.
  - Identidade → Nome
  - Identidade → Endereço
  - Identidade ¬→ Habilidade
  - Nome ¬→ Identidade
  - Endereço ¬→ Identidade
  - Habilidade ¬→ Identidade
  - Identidade → Nome, Endereço

3.2.3 Dependências Funcionais

- A ou B podem ser um conjunto de atributos.
  - Identidade → Nome
  - Identidade → Endereço
  - Identidade ¬→ Habilidade
  - Nome ¬→ Identidade
  - Endereço ¬→ Identidade
  - Habilidade ¬→ Identidade
  - Identidade → Nome, Endereço

3.2.3 Dependências Funcionais

- Idéia de normalização “fina”: agrupar numa tabela somente dois conjuntos de atributos X e Y, com X → Y.
- X é então a *chave* da tabela, e Y é *complemento da chave*.
- Consequência das definições de dependência funcional e de chave:
  - se X é chave então cada valor de X é *único*, e, consequentemente, um valor de X identifica uma linha da tabela.

3.2.3 Dependências Funcionais

- É importante salientar que mais de um atributo (ou conjunto de atributos) pode ser chave, isto é, pode-se ter vários X → Y, cada X sendo uma *chave candidata*.

3.2.4 Segunda Forma Normal (2FN)

- *Uma tabela está na Segunda Forma Normal (2FN) se ela é 1FN e todo atributo do complemento de uma chave candidata é totalmente funcionalmente dependente daquela chave*.

- A, B, C => D (D é *totalmente funcionalmente dependente* de {A, B, C}) se para todo valor de {A, B, C} só existe um valor de D, e se D não é funcionalmente dependente de A, ou B, ou C.

3.2.4 Segunda Forma Normal (2FN)

- Exemplo 1: ESPORTISTA (Identidade, Nome, Endereço, Esporte)

Chaves Candidatas	***Complementos da Chave***
Identidade	Nome, Endereço, Esporte
{Nome, Endereço}	Identidade, Esporte

3.2.4 Segunda Forma Normal (2FN)

- Identidade → Nome
- Identidade → Endereço
- Identidade ¬→ Esporte

- {Nome, Endereço} => Identidade
- {Nome, Endereço} ¬=> Esporte

3.2.4 Segunda Forma Normal (2FN)

- Conclusão: O atributo Esporte deve ser retirado da relação ESPORTISTA.
- ESPORTISTA (Identidade, Nome, Endereço)
- PRATICA-ESPORTE (Identidade, Esporte)
- Um atributo sublinhado faz parte da chave.
- Atualizar o endereço de Pelé: sem anomalia.
- Incluir uma nova habilidade de Pelé: sem anomalia.

3.2.4 Segunda Forma Normal (2FN)

Exemplo 2:
ESTUDANTE-DISCIPLINA

E #	*Enome*	*Sexo*	*Idade*	*D #*	*Dnome*	*Opinião*
E 1	João	M	25	D 1	Mat	Boa
E 1	João	M	25	D 2	Quim	Má
E 1	João	M	25	D 3	Fis	Boa
E 2	Maria	F	22	D 2	Quim	Satisf.
E 2	Maria	F	22	D 3	Fis	Satisf.
E 2	Maria	F	22	D 4	Est	Má
E 3	João	M	27	D 2	Quim	Boa
E 3	João	M	27	D3	Fis	Boa

Chaves Candidatas	*Complementos da Chave*
{E#, D#}	Enome, Sexo, Idade, Dnome, Opinião
{E#, Dnome}}	Enome, Sexo, Idade, D#, Opin

3.2.4 Segunda Forma Normal (2FN

- **{E#, D# }:**
  - {E#, D#} ¬=> Enome (E# → Enome)
  - {E#, D#} ¬=> Sexo (E# → Sexo)
  - {E#, D#} ¬=> Idade (E# → Idade)
  - {E#, D#} ¬=> Dnome (D# → Dnome)
  - {E#, D#} => Opinião

3.2.4 Segunda Forma Normal (2FN

- **{E# , Dnome):**
- {E#, Dnome} ¬=> Enome (E# → Enome)
- {E#, Dnome} ¬=> Sexo (E# → Sexo)
- {E#, Dnome} ¬=> Idade (E# → Idade)
- {E#, Dnome} ¬=> D# (Dnome → D#)
- {E#, Dnome} => Opinião
- **Conclusão**: Enome, Sexo, Idade e Dnome devem ser retirados de ESTUDANTE-DISCIPLINA

3.2.4 Segunda Forma Normal (2FN)

ESTUDANTE

E #	*Enome*	*Sexo*	*Idade*
E1	João	M	25
E2	Maria	F	22
E3	João	M	27

DISCIPLINA

D #	*Dnome*
D1	Mat
D2	Quim
D3	Fis
D4	Est

ESTUDANTE-DISCIPLINA

E #	*D #*	*Opinião*
E1	D1	Boa
E1	D2	Pobre
E1	D3	Boa
E2	D2	Satisfatória
E2	D3	Satisfatória
E2	D4	Pobre
E3	D2	Boa
E3	D3	Boa

3.2.4 Segunda Forma Normal (2FN)

- Ex3:A tabela Empregado anterior após passarmos para 2FN resultaria em três tabelas:

Empregado

Matrícula	*Nome*	*CodCargo*	*NomeCargo*
120	João	1	Programador
121	Hélio	1	Programador
270	Gabriel	2	Analista
273	Silva	3	Projetista
274	Abraão	2	Analista
279	Carla	1	Programador
301	Ana	1	Programador
306	Manuel	3	Projetista

3.2.4 Segunda Forma Normal (2FN)

– Ex3:A tabela Empregado anterior após passarmos para 2FN resultaria em três tabelas:

Projeto

CodProj	*DataFim*
01	17/07/95
08	12/01/96
12	21/03/96

Alocação

Matrícula	*CodProj*	*Horas*
120	01	37
120	08	12
121	01	45
121	08	21
121	12	107
270	08	10
270	12	78
273	01	22
274	12	31
279	01	27
279	08	20
279	12	51
301	01	16
301	12	85
306	12	67

3.2.4 Segunda Forma Normal (2FN)

- Anomalias da 2FN:
  - Inserção: Só podemos criar cargos se houver empregados designados para ele.
  - Remoção: Se removermos um empregado que ocupa unicamente um cargo na empresa, perderemos a informação deste cargo.
  - Atualização: Se um cargo muda de nome precisaremos mudar todas as tabelas em que este cargo aparece.

3.2.5 Terceira Forma Normal (3FN)

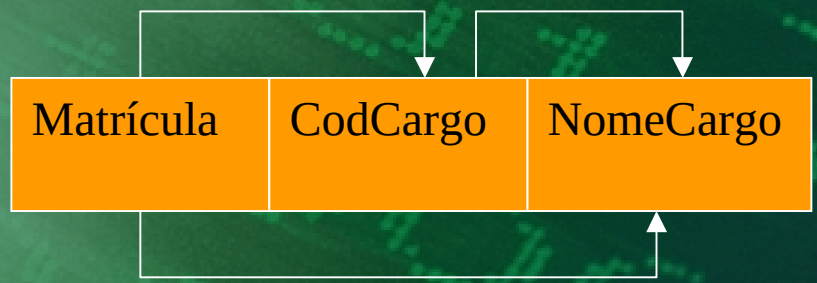
- Envolve o conceito de dependência transitiva. Suponha que tenhamos uma tabela com colunas A, B e C.

- Se a coluna C é funcionalmente dependente de B e B é funcionalmente dependente de A, então C é funcionalmente dependente de A.

3.2.5 Terceira Forma Normal (3FN)

- **Definição:** Uma relação está em 3FN se, e somente se, estiver em 2FN e todos os atributos não-chave forem dependentes não-transitivos da chave primária

- Ex.: Ao analisarmos a nova tabela empregado que está em 2FN temos:



3.2.5 Terceira Forma Normal (3FN)

- NomeCargo é dependente transitivo de Matrícula.
- Removendo esta dependência transitiva, obteremos,além das tabelas Projeto e Alocação, as seguintes tabelas:

Empregado

Matrícula	***Nome***	***CodCargo***
120	João	1
121	Hélio	1
270	Gabriel	2
273	Silva	3
274	Abraão	2
279	Carla	1
301	Ana	1
306	Manuel	3

Cargo

CodCargo	***Nome***
1	Programador
2	Analista
3	Projetista

3.2.5 Terceira Forma Normal (3FN)

- "Uma relação está em 3FN se todas as colunas da tabela são funcionalmente dependentes da chave inteira e nada além da chave".

- A 3FN elimina as características mais potencialmente indesejáveis dos dados que estão em 2FN ou 1FN.

- Existem outros casos especiais que requerem mais níveis de normalização: Boyce-Codd, 4FN e 5FN

3.2.6 Uma Metodologia de Normalização

- Passo 1: Tome projeções de tabelas 1FN para eliminar todas as dependências funcionais não-totais. O resultado é uma coleção de tabelas 2FN.

- Passo 2: Tome projeções das tabelas obtidas no passo 1 para eliminar todas as dependências transitivas. O resultado é uma coleção de relações 3FN.