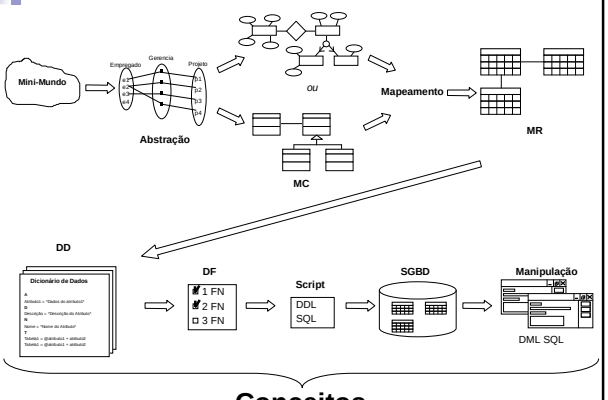
Banco de Dados I
Modelo Relacional

Osmar de Oliveira Braz Junior

Objetivos

- Conhecer os conceitos que regem o Modelo Relacional.
- Mapear o DER para o Modelo Relacional.
- Validar o Modelo Relacional utilizando as formas normais.



Conceitos

4. O Modelo Relacional

- O **modelo relacional** foi criado por Edgar Frank “Ted” Codd em 1970 e tem por finalidade representar os dados como uma coleção de relações, onde cada relação é representada por uma **tabela**, ou falando de uma forma mais direta, um arquivo. Porém, um arquivo é mais restrito que uma tabela. **Toda tabela** pode ser considerada um arquivo, porém, **nem todo arquivo** pode ser considerado uma **tabela**.

12 Regras de Codd

- **Codd** definiu **12 regras** para que um banco de dados, para que seja considerado "**totalmente relacional**".
- As doze regras estão baseadas na regra zero
  - "Qualquer sistema considerado, ou que deseja ser, um sistema gerenciador de banco de dados relacional deve ser capaz de gerenciar, por completo, bases de dados através de sua capacidade relacional".
- Essa regra determina que um SGBDR não permite exceções quanto ao modelo relacional de gerenciamento de bases de dados.
  - Algumas vezes as regras se tornam uma barreira e nem todos os SGBDs relacionais fornecem suporte a elas.

12 Regras de Codd

- **Regra 1** - **Representação de valores em tabelas:** Todas informações do banco de dados relacional são representadas de forma explícita no nível lógico e exatamente em apenas uma forma - por valores em tabelas. As informações devem ser apresentadas como relações (tabelas formadas por linhas e colunas) e o vínculo de dados entre as tabelas deve ser estabelecido por meio de valores de campos comuns. Isso se aplica tanto aos dados quanto aos metadados.
- **Regra 2** – **Acesso Garantido:** Cada um e qualquer valor atômico em um banco de dados relacional possuem garantia de ser logicamente acessado pela combinação do nome da tabela, do valor da chave primária e do nome da coluna.

12 Regras de Codd

- **Regra 3** – **Tratamento sistemático de nulos:** Valores nulos devem ser suportados de forma sistemática e independente do tipo de dado para representar informações inexistentes e / ou inaplicáveis, ou seja valores nulos devem ter um tratamento diferente de “valores em branco”.
- **Regra 4** – **Dicionário de dados ativo baseado no modelo relacional:** A descrição do banco de dados é representada no nível lógico da mesma forma que os dados ordinários, permitindo que usuários autorizados utilizem a mesma linguagem relacional aplicada aos dados regulares.

7

12 Regras de Codd

- **Regra 5** – **Linguagem Detalhada:** Um sistema relacional pode suportar várias linguagens e várias formas de recuperação de informações. Entretanto, deve haver pelo menos uma linguagem, com uma sintaxe bem definida e expressa por conjuntos de caracteres, que suporte de forma compreensiva todos os seguintes itens: definição de dados, definição de visões, manipulação de dados (interativa e embutida em programas), restrições de integridade, autorizações e limites de transações.
- **Regra 6** – **Atualização de Visões:** Todas as visões ("views") que são teoricamente atualizáveis devem também ser atualizáveis pelo sistema.

8

12 Regras de Codd

- **Regra 7** – **Atualização de alto nível:** A capacidade de manipular um conjunto de dados (relação) por um simples comando deve-se estender às operações de inclusão, alteração ou exclusão de dados.
- **Regra 8** – **Independência Física:** Programas de aplicação permanecem logicamente inalterados quando ocorrem mudanças no método de acesso ou na forma de armazenamento físico.
- **Regra 9** – **Independência Lógica:** Mudanças nas relações e nas visões provocam pouco ou nenhum impacto nas aplicações.

9

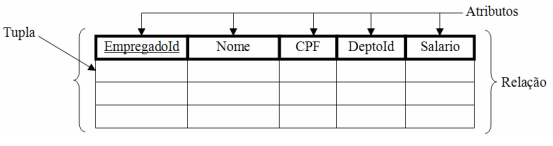
12 Regras de Codd

- **Regra 10** – **Independência de Integridade:** As aplicações não são afetadas quando ocorrem mudanças nas restrições de integridade.
- **Regra 11** – **Independência de Distribuição:** As aplicações não são logicamente afetadas quando ocorrem mudanças geográficas dos dados. Devem permanecer inalterados quando são distribuídos em meios ou máquinas diferentes.
- **Regra 12** – **Não Subversão:** Se um sistema possui uma linguagem de baixo nível, essa linguagem não pode ser usada para subverter as regras de integridades e restrições definidas no nível mais alto.

10

4. O Modelo Relacional

- Na terminologia do modelo relacional, cada tabela é chamada de **relação**; uma linha de uma tabela é chamada de **tupla**; o nome de cada coluna é chamado de **atributo**; o tipo de dado que descreve cada coluna é chamado de **domínio**.



11

4.1. Domínios, Tuplas, Atributos e Relações

- Um **domínio D** é um conjunto de valores atômicos, sendo que por atômico, podemos compreender que cada valor do domínio é indivisível. Durante a especificação do domínio é importante destacar o tipo, o tamanho e a **faixa do atributo** que está sendo especificado.

12

4.1. Domínios, Tuplas, Atributos e Relações

- Um **esquema de relação R**, denotado por *R(A1, A2, ... , An)*, onde cada atributo **A**i é o nome do papel desempenhado por um domínio **D** no esquema relação **R**, onde **D** é chamado domínio de **A**i e é denotado por *dom(Ai)*. O grau de uma relação **R** é o número de atributos presentes em seu esquema de relação

13

4.2. Atributo Chave de uma Relação

- Chave
- SuperChave
- Chave Composta
- Chave Candidata
- Chave Primária
- Chave Estrangeira

14

4.2. Atributo Chave de uma Relação

- Uma relação pode ser definida como um conjunto de tuplas **distintas**. Isto implica que a combinação dos valores dos atributos em uma tupla não pode se repetir na mesma tabela.
- Existirá sempre um subconjunto de atributos em uma tabela que garantem que **não** haverá valores **repetidos** para as diversas tuplas da mesma, garantindo que $t1[SC] \neq t2[SC]$.

15

4.2. Atributo Chave de uma Relação

- **SC** é chamada de **superchave** de um esquema de relação. Toda relação possui ao menos uma superchave - o conjunto de todos os seus atributos. Uma **chave C** de um esquema de relação **R** é uma superchave de **R** com a propriedade adicional que removendo qualquer atributo **A** de **K**, resta ainda um conjunto de atributos **K'** que não é uma superchave de **R**.

16

4.2. Atributo Chave de uma Relação

- Por exemplo, o conjunto: *(RegistroAcademico, Nome, Endereço)* é uma **superchave** para estudante, porém, não é uma chave pois se tirarmos o campo *Endereço* continuaremos a ter uma **superchave**.
- Já o conjunto *(Nome da Revista, Volume, No da Revista)* é uma **superchave** e uma chave, pois qualquer um dos atributos que retirarmos, deixaremos de ter uma **superchave**, ou seja, *(Nome da Revista, Volume)* não identifica uma única tupla.

17

4.2. Atributo Chave de uma Relação

- Em outras palavras, uma superchave é uma **chave composta**, ou seja, uma chave formada por mais que um atributo.

18

4.2. Atributo Chave de uma Relação

Relação DEPENDENTE

EmpregadoId	Nome	DtNascimento	Relacao	Sexo
10101010	Jorge	27/12/86	Filho	Masculino
10101010	Luiz	18/11/79	Filho	Masculino
20202020	Fernanda	14/02/69	Conjuge	Feminino
20202020	Angelo	10/02/95	Filho	Masculino
30303030	Fernanda	01/05/90	Filho	Feminino

19

4.2. Atributo Chave de uma Relação

- Quando uma relação possui mais que uma chave (não confundir com chave composta) - como por exemplo RG e CPF para empregados - cada uma destas chaves são chamadas de **chaves candidatas**. Uma destas chaves candidatas deve ser escolhida como **chave primária**

20

4.3. Chave Estrangeira

- Uma **chave estrangeira CE** de uma tabela **R1** em **R2** ou vice-versa, especifica um relacionamento entre as tabelas **R1** e **R2**.

21

Exemplo de Chave Estrangeira

Relação DEPARTAMENTO

DeptoId	Nome	GerenteId
1	*Contabilidade*	*10101010*
2	*Engenharia Civil*	*30303030*
3	*Engenharia Mecânica*	*20202020*

Relação EMPREGADO

EmpregadoId	Nome	CPF	DeptoId	SupervisorId	Salario
10101010	*João Luiz*	*11111111*	*1*	***NULO***	*3.000,00*
20202020	*Fernando*	*22222222*	*2*	*10101010*	*2.500,00*
30303030	*Ricardo*	*33333333*	*2*	*10101010*	*2.300,00*
40404040	*Jorge*	*44444444*	*2*	*20202020*	*4.200,00*
50505050	*Renato*	*55555555*	*3*	*20202020*	*1.300,00*

22

4.4. Restrições de Integridade

- A maioria das aplicações de banco de dados possui certas **restrições** de integridade que devem sustentar os dados. Um **SGBD** deve fornecer capacidades para definir e impor essas restrições.
  - **Integridade de Domínio**
  - **Integridade de Entidade**
  - **Integridade de Referência**

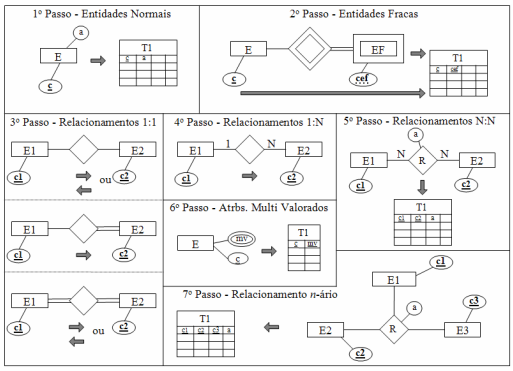
23

4.4. Restrições de Integridade

- **Integridade de Domínio**
  - Neste tipo de restrição de integridade, as validações ocorrem em cada campo da tabela, como por exemplo se um campo deve aceitar valores nulo ou apenas uma faixa de valores, por exemplo: Temos um campo Sexo, que só deve aceitar valores do tipo M ou F e, independente da aplicação, não deverá aceitar outros valores.
- **Integridade de Entidade**
  - Já na restrição de integridade de entidade, temos validações um nível acima da integridade de domínio, onde defini-se quais campos de nossa tabela são chaves primárias ou únicas. Um exemplo de chave única é o campo de CPF que nunca pode se repetir. Obs.: Geralmente chaves únicas são chamadas de chaves candidatas, pois são candidatas a chave primária, mas não são uma.
- **Integridade de Referência (ou Integridade Referencial)**
  - A restrição de integridade referencial, cria uma referência entre colunas tipo chave primária ou única de outras tabelas, as famosas chaves estrangeiras. Não se pode criar uma referência (relacionamento) de uma coluna se esta não for uma chave primária de outra tabela. Este tipo de integridade é fundamental para verificar se um dado será inserido de forma correta e, como é uma chave estrangeira, se este dado realmente existe.

24

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional



25

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- O mapeamento do modelo entidade relacionamento para o Modelo Relacional segue oito passos básicos a saber:
- 1. Para cada entidade **E** no modelo ER é criada uma tabela **T1** no Modelo Relacional que inclua todos os atributos simples de **E**; para cada atributo composto, são inseridos apenas os componentes simples de cada um; um dos atributos chaves de **E** deve ser escolhida como a chave primária de **T1**;

26

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 2. Para cada entidade fraca **EF** com entidade proprietária **E** no modelo ER, é criada uma tabela **T1** no Modelo Relacional incluindo todos os atributos simples de **EF**; para cada atributo composto , são inseridos apenas os componentes simples de cada um; a chave primária desta relação **T1** será composta pela chave parcial da entidade fraca **EF** mais a chave primária da entidade proprietária **E**;

27

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 3. Para cada relacionamento regular com cardinalidade 1:1 entre entidades **E1** e **E2** que geraram as tabelas **T1** e **T2** respectivamente, devemos escolher a chave primária de uma das relações (**T1**, **T2**)e inseri-la como chave estrangeira na outra relação; se um dos lados do relacionamento tiver participação total e outro parcial, então é interessante que a chave do lado com participação **parcial** seja inserido como chave estrangeira no lado que tem participação **total**;

28

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 4. Para cada relacionamento regular com cardinalidade 1:N entre entidades **E1** e **E2** respectivamente e que geraram as tabelas **T1** e **T2** respectivamente, deve-se inserir a chave primária de **T1** como chave estrangeira em **T2**;

29

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 5. Para cada relacionamento regular com cardinalidade N:M entre entidades **E1** e **E2**, cria-se uma nova tabela **T1**, contendo todos os atributos do relacionamento mais o atributo chave de **E1** e o atributo chave de **E2**; a chave primária de **T1** será composta pelos atributos chave de **E1** e **E2**;

30

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 6. Para cada atributo multivalorado **A1**, cria-se uma tabela **T1**, contendo o atributo multivalorado **A1**, mais o atributo chave **C** da tabela que representa a entidade ou relacionamento que contém **A1**; a chave primária de **T1** será composta por **A1** mais **C**; se **A1** for composto, então a tabela **T1** deverá conter todos os atributos de **A1**;

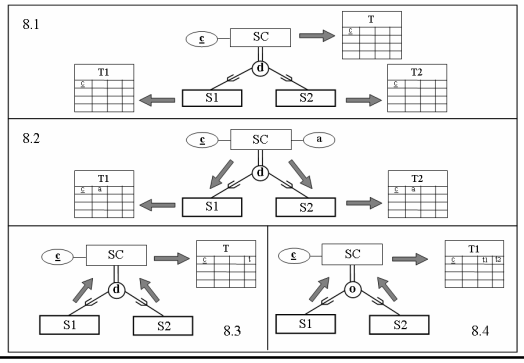
31

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 7. Para cada relacionamento n-ário, n > 2, cria-se uma tabela **T1**, contendo todos os atributos do relacionamento; a chave primária de **T1** será composta pelos atributos chaves das entidades participantes do relacionamento;

32

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional



33

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 8. Converta cada especialização com *m* subclasses **{S1, S2, ..., Sm}** e superclasse **SC**, onde os atributos de **SC** são **{c, a1, a2, ..., an}** onde **c** é a chave primária de **SC**, em tabelas utilizando uma das seguintes opções:

34

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

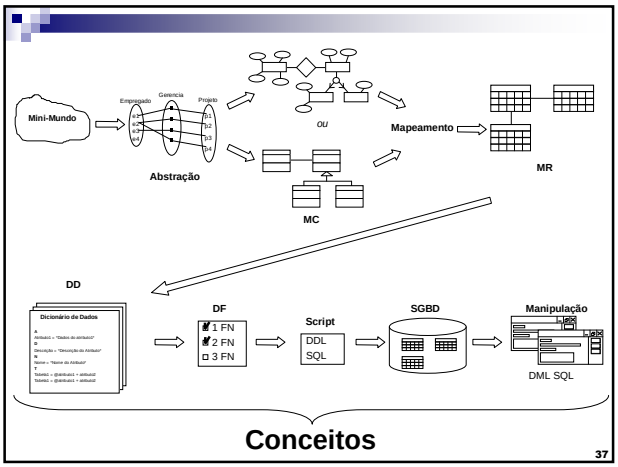
- 8.1. Crie uma tabela **T** para **SC** com os atributos **A(T) = {c, a1, a2, ..., an}** e chave **C(T)** = **c**; crie uma tabela **Ti** para cada subclasse **Si** , 1 ≤ i ≤ m, com os atributos **A(Ti)** = **{c}** ∪ **A(Si)**, onde **C(T)** = **c**;
- 8.2. Crie uma tabela **Ti** para cada subclasse **Si**, 1 ≤ i ≤ m, com os atributos **A(Ti) = A(Si)** ∪ **{c, a1, a2, ..., an}** e **C(Ti)** = **c**;

35

4.3. Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

- 8.3. Crie uma tabela **T** com os atributos **A(T) = {c, a1, a2, ..., an}** ∪ **A(S1)** ∪ **...** ∪ **A(Sm)** ∪ **{t}** e **C(T)** = **c**, onde **t** é um atributo **tipo** que indica a subclasse à qual cada tupla pertence, caso isto venha a ocorrer;
- 8.4. Crie uma tabela **T** com atributos **A(T) = {c, a1, a2, ..., an}** ∪ **A(S1)** ∪ **...** ∪ **A(Sm)** ∪ **{t1, t2, ..., tm}** e **C(T)** = **c**; esta opção é para generalizações com "overlapping", e cada **ti**, 1 ≤ i ≤ m, é um atributo "booleano" indicando se a tupla pertence ou não à subclasse **Si**; embora funcional, esta opção pode gerar uma quantidade muito grande de valores nulos;

36



4.4. Dicionário de Dados

- O dicionário de dados é um repositório de dados sobre os dados do software. Ele deverá conter a definição dos elementos que do Modelo de Dados, como Entidades e Atributos.

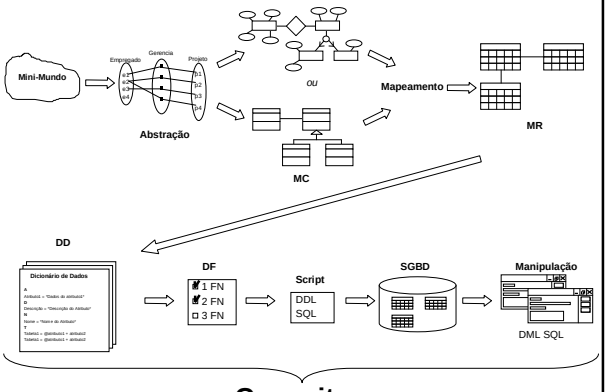
Símbolo	Significado
=	É composto de
+	E
[ ]	Seleção (Escolha uma das opções alternativas)
{ }	Iteração (indica ocorrência repetida, "zero ou mais ocorrências de")
( )	Opcional (pode estar presente ou ausente)
|	separa opções alternativas na construção [ ] (Seleção)
* *	Comentário
@	Identificador (campo chave) de uma entidade.

38

4.4. Dicionário de Dados
Exemplo

caracter_alfabético = [A-Z | a-z] * uma letra do alfabeto *
caracter_alfanumérico = [A-Z | a-z | 0-9 | ' | - | _] * um número, uma letra ou um sinal de pontuação *
caracter_numérico = [0-9] * um número decimal *
Cliente = @ClienteId + nome_cliente + endereço + limite_credito + sexo + data_nascimento * um cliente da empresa XYZ *
ClienteId = numero_chave * identificação de um cliente *
data = "DD/MM/YYYY" * uma data valida no formato dois dígitos numéricos para o dia, dois para o mês e quatro para o ano*
data_nascimento = data * data de nascimento do cliente, data superior a 01/01/1910 *
endereço = 1{caracter_alfanumérico}50 * endereço de um cliente para contato *
limite_credito = valor_monetário * valor do credito que será concedido a um cliente para pedidos não previamente pagos * * valor maior que zero *
nome_cliente = título_cortesia + primeiro_nome + (nome_intermediário) + último_nome * nome do cliente* * preenchimento obrigatório *
nome_intermediário = 1{caracter_alfabético}20
nome_produto = 1{caracter_alfabético}40
numero_chave = 1{caracter_numérico}12 * valor numérico para um campo chave *
preço = valor_monetário
primeiro_nome = 1{caracter_alfabético}20
Produto = @ProdutoId + nome_produto + preço + unidade * um produto da empresa XYZ *
ProdutoId = numero_chave * identificação de um produto *
sexo = ["M" | "F"] * Sexo M para Masculino e F para Feminino *
título_cortesia = ["Sr." | "Sra." | "Srta."] "Sras." | "Dr." | "Professor"]
último_nome = 1{caracter_alfabético}20
unidade = ["UN" | "DZ" | "CX" | "KG" | "GR"] * unidade do produto*
valor_monetário = 0{caracter_numérico}9, 0{caracter_numérico}2* valor numérico com casas decimais*

39



4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional

- Uma **dependência funcional** é uma restrição entre dois conjuntos de atributos de uma base de dados. Suponha que o esquema de uma base de dados **R** possua *n* atributos A1, A2, ..., An; pense em **R** = **{ A1, A2, ... , An }** como a representação universal da base de dados.

41

4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional

- representada por **X → Y**
  - entre dois conjuntos de atributos **X** e **Y** que são subconjuntos de **R**
  - especificam uma restrição nas tuplas que podem compor uma instância relação r de R
- Para **X → Y** lê-se: ***Y** é funcionalmente dependente de **X***, ou ***X** infere sobre **Y***.
- Os atributos **X** e **Y** podem ser compostos.

42

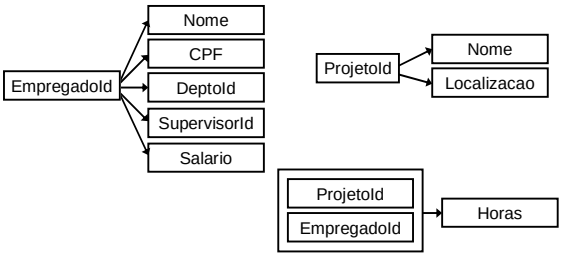
4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional

- Exemplos
  - EmpregadoId → { Nome, CPF, DeptoId, SupervisorId, Salario }
  - ProjetoId → { Nome, Localizacao }
  - { EmpregadoId, ProjetoId } → Horas

43

4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional

- Pode-se representar através de **diagramas de dependências**



44

4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional

- A **dependência 1** implica que o número de um EmpregadoId define de forma única o nome do empregado e o CPF do empregado.
- A **dependência 2** implica que o ProjetoId define de forma única o nome do projeto e sua localização.
- A **dependência 3** implica que o EmpregadoId mais o ProjetoId define de forma única o número de horas que o empregado trabalhou no projeto.

45

4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional

- A especificação das inferências deve ser elaborada pelo **projetista de banco de dados** em conjunto com o **analista de sistemas**, pois os mesmos deverão ter conhecimento da **semântica** da base de dados.

46

4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional Multivalorada

- Uma **dependência funcional multivalorada** (DMV) X→→Y especificada no esquema de relação R, em que tanto X como Y são subconjuntos de R, especifica a seguinte restrição em qualquer estado da relação *r* de R.
- Se duas tuplas *t*1 e *t*2, existirem em *r* de modo que *t*1[X] = *t*2[X], então duas tuplas *t*3 e *t*4 também deveriam existir em r com a seguintes propriedades(*t*1,*t*2,*t*3,*t*4 não são necessariamente **distintas**), onde utilizamos Z(é a abreviação para os atributos remanescentes em R depois que os atributos em (X∪Y) são **removidos** de R) para representar (R – (X ∪ Y)):

47

4.5. Dependência Funcional e Normalização
4.5.1. Dependência Funcional Multivalorada

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[X] = t_1[X]$ e $t_4[X] = t_2[X]$
- $t_3[X] = t_2[X]$ e $t_4[X] = t_1[X]$

- Sempre que X →→ Y se mantém, dizemos que X **multidetermina** Y. Devido à **simetria** na definição, sempre que X →→ Y se mantém em R, também o faz X →→ Z. Portanto, X →→ Y implica X→→ Z, e por isso é às vezes escrito como X →→ Y|Z.

48

4.5.2. Normalização

- Processo no qual são eliminados esquemas de relações (tabelas) **não satisfatórios.**
- Decompõe através da separação de seus atributos em esquemas de relações **menos complexas** mas que satisfaçam as propriedades desejadas.

49

4.5.2. Normalização

- Foi proposto inicialmente por Edgar Frank "Ted" Codd
- Conduz um esquema de relação através de um bateria de testes para certificar se o mesmo está na **1a**, **2a** e **3a Formas Normais**.
- Estas três Formas Normais são baseadas em dependências funcionais dos atributos do esquema de relação.

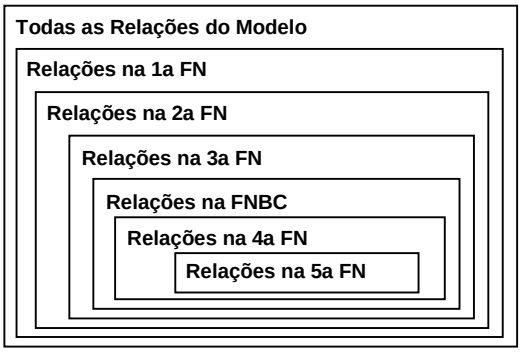
50

4.5.2. Normalização

- Ted Codd mais tarde propôs uma **3FN** mais sólida chamada de formal normal de **Boyce-Codd (FNBC)**
- Todas estas formas normais são baseadas nas dependências funcionais entre atributos de uma relação.
- Posteriormente, foram propostos uma **quarta forma normal (4FN)** e uma **quinta forma normal (5FN ou FNPJ -** Junção de Projeto**)**, baseadas nos conceitos de dependências multivaloradas e dependências de junção respectivamente.

51

4.5.2. Normalização



52

4.5.2. Normalização

- Quando definimos **cuidadosamente** um **Diagrama ER**, identificando todas as entidades corretamente, os esquemas de relação gerados do diagrama ER não precisam de muito mais normalização.
- Entretanto, pode haver dependências funcionais entre os atributos de uma entidade. Por exemplo:
  - Suponha que uma entidade ***funcionário*** tenha atributos *número_depto* e *endereço_depto* e que exista uma dependência funciona *número_depto*→*endereço_depto*.
  - Poderíamos, então normalizar a relação gerada de funcionário.

53

4.5.2. Normalização

- A maioria dos exemplos dessas dependências surge de um **projeto ruim** de diagrama ER.
- Nesse exemplo, se tivéssemos projetado o diagrama ER corretamente, teríamos criado uma entidade ***depto*** com o atributo *endereço_depto* e uma relação entre ***funcionário*** e ***depto***.
- Da mesma forma, uma relação envolvendo mais de duas entidades pode não estar em uma forma normal desejável.
  - Como a maioria dos relacionamentos é binária, esses casos são relativamente **raros**.
  - Na verdade, algumas variantes do diagrama ER realmente dificultam ou impossibilitam especificar relações não binárias.

54

4.5.2. Normalização

- As dependências funcionais **podem** ajudar a detectar um **mau projeto ER**.
- Se as relações geradas não estiverem na forma normal desejada, o problema **pode** (**e deve**) de ser corrigido no **diagrama E-R**.
- Ou seja, a normalização pode ser deixada por conta da intuição do projetista durante a modelagem ER, e pode ser feita formalmente nas relações geradas do modelo ER.

55

4.5.2.1. 1a Forma Normal

- A **1a Forma Normal** prega que todos os atributos de uma tabela devem ser **atômicos** (indivisíveis), ou seja, não são permitidos atributos multivalorados, atributos compostos ou atributos multivalorados compostos.

56

4.5.2.1. 1a Forma Normal

- **{ ClienteId } → { {Telefone}, Endereço (Rua Número, Cidade)}**

		Telefone 1	Endereco		
Cliente	ClienteId	Telefone *n*	Rua	No	Cidade

Aplicando a 1a FN

- **{ ClienteId } → { Rua Número, Cidade}**
- **{ ClienteId, Telefone } → { }**

Cliente	ClienteId	Rua	Número	Cidade

Cliente_Telefone	ClienteId	Telefone

57

4.5.2.2. 2a Forma Normal

- Prega o conceito da **dependência funcional total**.
- Uma dependência funcional **X → Y** é **total** se removemos um atributo **A** qualquer do componente **X** e desta forma, a dependência funcional deixa de existir.
- A dependência funcional **X → Y** é uma **dependência funcional parcial** se existir um atributo **A** qualquer do componente **X** que pode ser removido e a dependência funcional **X → Y** não deixa de existir.

58

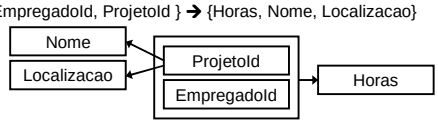
4.5.2.2. 2a Forma Normal

- Uma tabela **T** está na 2a Forma Normal se estiver na 1a Forma Normal e todo atributo que não compõem a chave primária **C** for totalmente funcionalmente dependente da chave primária **C**.
- Se uma tabela não está na 2a Forma Normal à mesma pode ser normalizada gerando outras tabelas cujos atributos que não façam parte da chave primária sejam totalmente funcionalmente dependente da mesma, ficando a tabela na 2a Forma Normal.

59

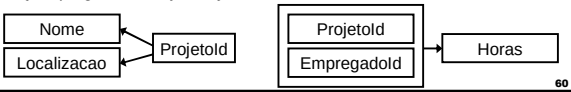
4.5.2.2. 2a Forma Normal

- { EmpregadoId, ProjetoId } → {Horas, Nome, Localizacao}



Aplicando a 2a FN

- ProjetoId → { Nome, Localizacao }
- { EmpregadoId, ProjetoId } → Horas



60

4.5.2.3. 3a Forma Normal

- Prega o conceito de **dependência transitiva**.
- Uma dependência funcional **X → Y** em uma tabela **T** (X ∪ Z ∪ Y) é uma dependência transitiva se existir um conjunto de atributos **Z** que não é um subconjunto de chaves de **T** e as dependências **X → Z**, **Z → Y**, são válidas. Y não pertence X e Z.

61

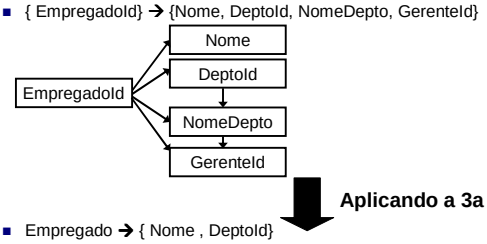
4.5.2.3. 3a Forma Normal

- Uma tabela está na 3a Forma Normal se estiver na 2a Forma Normal e **não houver dependência transitiva** entre atributos **não chave**.

62

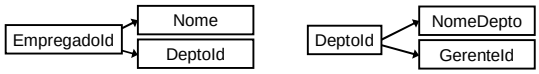
4.5.2.3. 3a Forma Normal

- { EmpregadoId} → {Nome, DeptoId, NomeDepto, GerenteId}



Aplicando a 3a FN

- Empregado → { Nome , DeptoId}
- DeptoId → { Nome, GerenteId}



63

4.5.3. Resumo das Formas

- Forma normal:
  - Primeira(**1FN**)
- Teste:
  - A relação não deve ter qualquer atributo não atômico nem relações agrupadas.
- Solução:
  - Forme novas relações para cada atributo não atômico ou relação aninhada.

64

4.5.3. Resumo das Formas

- Forma normal:
  - Segunda(**2FN**)
- Teste:
  - Para relações nas quais a chave primária contém múltiplos atributos, nenhum atributo não chave deve ser funcionalmente dependente de uma parte da chave primária.
- Solução:
  - Decomponha e monte uma relação para cada chave parcial com seu(s) atributo(s) dependente(s). Certifique-se de manter uma relação com a chave primária original e quaisquer atributos que sejam completamente dependentes dela em termos funcionais.

65

4.5.3. Resumo das Formas

- Forma normal:
  - Terceira(**3FN**)
- Teste:
  - A relação não deve ter um atributo não chave funcionalmente determinado por um outro atributo não chave (ou por um conjunto de atributos não chave). Ou seja, não deve haver dependências transitiva de um atributo não chave na chave primária.
- Solução:
  - Decomponha e monte uma relação que inclua o(s) atributo(s) não chave que funcionalmente determine(m) outros atributos não chave.

66

4.5.4. Outras Formas Normais

- Ted Codd mais tarde propôs uma 3FN mais sólida chamada de formal normal de **Boyce-Codd (FNBC)**
- Todas estas formas normais são baseadas nas dependências funcionais entre atributos de uma relação.
- Posteriormente, foram propostos uma **quarta forma normal (4FN)** e uma **quinta forma normal (5FN)**, baseadas nos conceitos de dependências multivaloradas e dependências de junção respectivamente.

67

4.5.4.1. Forma Normal de Boyce-Codd (FNBC)

- Proposta como uma forma mais **simples da 3FN**, mas foi considerada mais restrita do que ela, porque toda relação na FNBC também está na 3FN; entretanto, uma relação na 3FN não está necessariamente na FNBC.
- Um esquema de relação R está na FNBC se, sempre que uma **dependência funcional não-trivial** X→A se mantiver em R, X for uma superchave de R.
- A única **diferença** entre as definições da FNBC e da 3FN é que a condição da 3FN que diz que permite que A seja principal, está **ausente** da FNBC

68

4.5.4.1. Forma Normal de Boyce-Codd (FNBC)

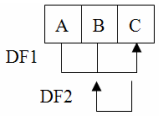
LECIONA		
Aluno	**Disciplina**	**Instrutor**
Naiana	Banco de Dados	João
Fernando	Banco de Dados	Pedro
Fernando	Sistemas Operacionais	Maria
Fernando	Teoria	Marco
Guilherme	Sistemas Operacionais	José
Guilherme	Banco de Dados	João
Márcia	Banco de Dados	Antonio
Zuleide	Banco de Dados	Pedro

A relação LECIONA esta na 3 Forma Normal mas não na FNBC.

69

4.5.4.1. Forma Normal de Boyce-Codd (FNBC)

- Na prática, a maioria dos esquemas de relação que **estão na 3FN** também estão na FNBC. Somente se X→A se mantiver em um esquema de relação R, não sendo X uma **superchave** e sendo A um atributo principal, R estará na 3FN mas não na FNBC.



DF1 : {aluno, disciplina}→instrutor

DF2 : instrutor→disciplina

70

4.5.4.1. Forma Normal de Boyce-Codd (FNBC)

- Note que (aluno, disciplina) é uma **chave candidata** para essa relação e que as dependências mostradas seguem o padrão da figura acima. Portanto essa relação está na 3FN, mas não está na FNBC.
- A decomposição desse esquema em dois esquemas não é simples porque pode ser decomposto em um de três pares possíveis:
  - 1. (aluno, instrutor) e (aluno, disciplina)
  - 2. (disciplina, instrutor) e (disciplina, aluno)
  - 3. (instrutor, disciplina) e (instrutor, aluno)

71

4.5.4.2. Quarta Forma Normal

- É violada quando uma relação que tem **dependências multivaloradas** indesejáveis, e portanto pode ser utilizada para identificar e decompor relações.
- Um esquema de relação R está na 4FN com relação a um conjunto de dependências F (que inclui dependências funcionais e dependências multivaloradas) se, para toda dependência multivalorada não-trivial X→→Y em F^+, X for uma superchave de R.

72

4.5.4.2. Quarta Forma Normal

- Uma tupla nessa relação EMPREGADO representa que um empregado cujo nome seja NomeEmpregado trabalha no projeto cujo nome é NomeProjeto e tem um dependente cujo nome é NomeDependente.
- Um empregado pode trabalhar em diversos projetos e pode ter diversos dependentes, e os projetos e dependentes do empregado são independentes um do outro.
- ER – Entidade Fraca ou Multivalorado

Relação EMPREGADO

NomeEmpregado	NomeProjeto	NomeDependente
Joao	X	Pedro
Joao	Y	Maria
Joao	X	Maria
Joao	Y	Pedro

Relação EMPREGADO_PROJETOS

NomeEmpregado	NomeProjeto
Joao	X
Joao	Y

Relação DEPENDENTES

NomeEmpregado	NomeDependente
Joao	Pedro
Joao	Maria

4.5.4.2. Quarta Forma Normal

- A relação EMPREGADO não está na **4FN** porque nas **DMVs** não triviais
  - NomeEmpregado→→NomeProjeto e
  - NomeEmpregado→→NomeDependente,
- NomeEmpregado não é a **superchave** de EMPREGADO.
- Decompomos EMPREGADO em:
  - EMPREGADO_PROJETOS e
  - DEPENDENTES.
- Tanto EMPREGADO_PROJETOS como DEPENDENTES são DMVs triviais. Nenhuma outra DMV não trivial se mantém em EMPREGADO_PROJETOS e DEPENDENTES.

74

4.5.4.3. Quinta Forma Normal

- Também chamada ***Forma Normal de Junção de Projeto***.
- Um esquema de relação R está na **quinta forma normal** (5FN) (ou forma normal de projeção de junção[FNPJ]) com respeito a um conjunto F de **dependências funcionais**, **multivaloradas** e de junção se, para toda dependência de junção não trivial DJ(R_1,R_2,...,R_n) em F^+ (ou seja, implicada por F), todo R_i for uma superchave de R.

75

4.5.4.3. Quinta Forma Normal

- sempre que um fornecedor *f* fornece a peça *p*, e um projeto *j* utiliza a peça *p*, e o fornecedor *f* fornece pelo menos uma peça para o projeto *j*, então o fornecedor *f* também estará fornecendo a peça *p* para o projeto *j*
- A Figura mostra como a relação FORNECIMENTO *com a* ***dependência de junção*** é decomposta em três relações R1, R2 e R3, que estão cada qual na 5FN.
- Note que aplicar a **JUNÇÃO NATURAL** a quaisquer duas dessas relações produz tuplas inválidas, mas com uma JUNÇÃO NATURAL aplicada as três conjuntamente não o fazem

76

4.5.4.3. Quinta Forma Normal

Relação FORNECIMENTO

NomeFornecedor	NomePeca	NomeProjeto
Joao	Quadrado	X
Joao	Circulo	Y
Antonio	Quadrado	Y
Wilson	Circulo	Z
Antonio	Triângulo	X
Antonio	Quadrado	X
Joao	Quadrado	Y

Relação R1

NomeFornecedor	NomePeca
Joao	Quadrado
Joao	Circulo
Antonio	Quadrado
Wilson	Circulo
Antonio	Triângulo

Relação R2

NomeFornecedor	NomeProjeto
Joao	X
Joao	Y
Antonio	Y
Wilson	Z
Antonio	X

Relação R3

NomePeca	NomeProjeto
Quadrado	X
Circulo	Y
Quadrado	Y
Circulo	Z
Triângulo	X

77

Conseqüências da Normalização

- Como conseqüência do processo de normalização temos:
  - problemas com anomalias e inconsistências diminuem,
  - relações são simplificadas e estrutura mais regulares,
  - aumento da integridade dos dados e
  - eventual queda de performance.

78

Bibliografia

- **Principal**
  - CHEN, P. **The Entity-Relationship Model - Toward a Unified View of Data**. TODS, 1976.
  - DATE, C. J. **Introdução a Sistemas de Banco de Dados.** 7. ed. Rio de Janeiro: Campus, 2000. 803 p.
  - ELMASRI, S. N.; NAVATHE, B.S.. **Sistemas de Banco de Dados:** Fundamentos e Aplicações. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos, 2002. 837 p.
  - SILBERSCHATZ, A. ; KORTH, H.F. ; SUDARSHAN, S. **Sistema de Banco de Dados.** 5. ed. Rio de Janeiro: Elsevier, 2006.
- **Complementar**
  - BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML.** 1. ed. Rio de Janeiro: Campus, 2002. 286 p.
  - CHIAVENATO, I. **Introdução à Teoria Geral da Administração.** 7. ed. Rio de Janeiro: Campus, 2004. 634 p.
  - DAUM, B. **Modelagem de Objetos de negócio com XML:** abordagem com base em XML Schema. Rio de Janeiro: Elsevier, 2004.
  - KROENKE, D.M. **Banco de Dados:** Fundamentos, Projeto e Implementações. 6. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1998. 382 p.
  - OZSU, M.T.; VALDURIEZ, P. **Princípios de sistemas de banco de dados distribuídos.** 1. ed. Rio de Janeiro: Campus, 2001.
  - YOURDON, E. **Análise Estrutura Moderna.** 3. ed. Rio de Janeiro: Campus, 1992. 836 p.

Fim