Design de Bases de Dados Relacionais

- Tópicos:
  - Objetivos com o Desenho de Bases de Dados
  - Dependências funcionais
  - 1ª Forma Normal
  - Decomposição
  - Forma Normal de Boyce-Codd
  - 3ª Forma Normal
  - Visão geral sobre o processo de design
- Bibliografia:
  - Capítulo 8 do livro recomendado
  - Capítulos 4, 5, 6 e 7 do livro *The theory of relational databases*
    - Neste último, a matéria está muito mais detalhada, e o livro está disponível online, gratuitamente, na página do autor.

Objetivos com o Desenho de BDs Relacionais

- Pretende-se encontrar “bons” conjuntos de esquemas de relações para armazenar os dados.
- Um “mau” design pode levar a:
  - Repetição de dados.
  - Impossibilidade de representar certos tipos de informação.
  - Dificuldade na verificação da integridade
- Objetivos do Design:
  - Evitar dados redundantes
  - Garantir que as relações relevantes sobre dados podem ser representadas
  - Facilitar a verificação de restrições de integridade.

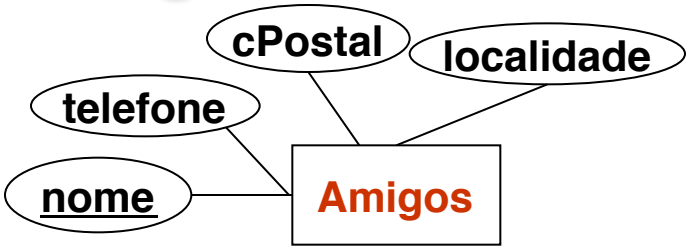
BDs Relacionais e o Modelo ER

- Vimos como é que se pode chegar a um desenho de uma BD relacional a partir de um modelo ER
  - Essa BD reflete na totalidade o que se especificou no modelo ER
  - Se o modelo ER estiver correto (e de acordo com o que se pretende dos dados em causa) então a BD relacional está bem desenhada
- E se o modelo ER não estiver correto?
  - A BD relacional não terá um bom design..
- E como é que se sabe se está correto ou não?
  - Desde logo, o que é que significa “está correto”?

- Vamos começar por ver em que termos se pode definir uma noção de correção, e depois ver como é que se pode garantir que a base de dados relacional obedece a essa noção de correção

Exemplo de mau design

- Este esquema dá origem à tabela:
  Amigos = (*nome*, *telefone, cPostal, localidade)*



- Está correta? É um bom design?

nome	telef	cPostal	localidade
Maria	1111	2815	Caparica
João	2222	1000	Lisboa
Pedro	1112	1100	Lisboa
Ana	3333	2815	Caparica

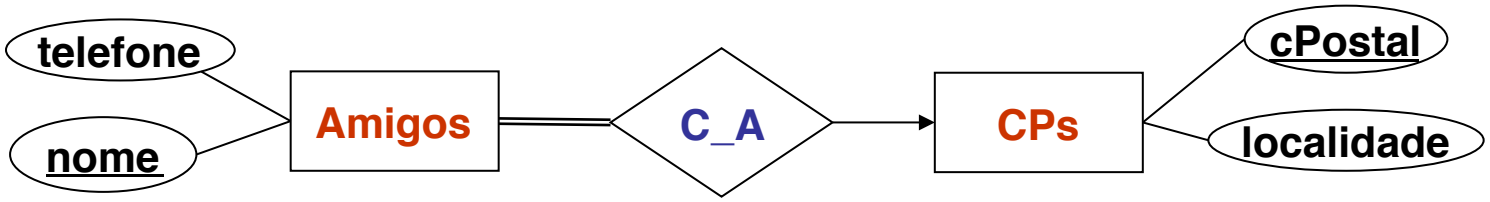
Repetido!

Redundante!

- Redundância:
  - Os valores de (*cPostal, localidade)* são repetidos para cada amigo com um mesmo código postal
  - Dá azo a inconsistências
  - Complica bastante a verificação da integridade dos dados
- Dificuldade de representar certa informação
  - Não se pode armazenar informação do código postal de uma localidade sem que existam amigos dessa localidade.
    - Por vezes podem usar-se valores nulos, mas estes são difíceis de gerir.

Design alternativo

- E como seria correto?



- Assim a BD relacional teria:

  Amigos = (*nome, telefone, cPostal)* e *CPs(cPostal, localidade)*

- Já não há redundância!

nome	telef	cPostal
Maria	1111	2815
João	2222	1000
Pedro	1112	1100
Ana	3333	2815

cPostal	localidade
2815	Caparica
1000	Lisboa
1100	Lisboa

- Mas porquê? Como é que se sabe?
  - O que é que nos leva a dizer que no design anterior há redundância

Dependências Funcionais e Redundância

- Porque é que há redundância com o esquema?

  Amigos = (*nome, telefone, cPostal, localidade)*

- Porque *localidade* é univocamente determinado por *cPostal*, e *cPostal* não é chave de *Amigos*
  - Se houver dois amigos com o mesmo *cPostal*, o que pode acontecer porque *cPostal* não é chave, então a *localidade*, só pode ser a mesma, porque é univocamente determinada pelo *cPostal*
    - Redundância!
- **Dependência funcional** $\alpha \rightarrow \beta$ significa que β é univocamente determinado por α
  - Há que ver quais (todas) as dependências funcionais que se pretendem impor na BD
  - Se $\alpha \rightarrow \beta$, e houver uma tabela com atributos $\alpha\beta$, e α não é chave nessa tabela, então há redundância
  - Se houver redundância, decompor a tabela

Decomposição

- Decompor o esquema *Amigos* em:

  Amigos1 = (nome, telefone, código_postal)

  CPs = (código_postal, localidade)

- Todos os atributos do esquema original (*R*) devem aparecer na decomposição em (R_1, R_2):

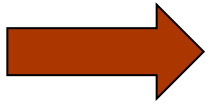
$$R = R_1 \cup R_2$$

- Decomposição sem perdas
  - Há que ter cuidado com a forma como se decompõe!
  - Se não se tem cuidado, a decomposição pode dar origem a perda de informação.

Exemplo de decomposição com perdas

- Decomposição de *CPs* em:
  - *CP1 = (código_postal)* e *Locs = (localidade)*

cPostal	localidade
2815	Caparica
1000	Lisboa
1100	Lisboa



cPostal
2815
1000
1100

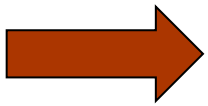
localidade
Caparica
Lisboa

- Perdeu-se a informação de quais os CPs das localidades!!!
  - Decompor parecia bom para evitar redundâncias
  - Mas decompor demais pode levar à perda de informação

Outro exemplo com perdas

- Decomposição de *Amigos* em:
  - *Amigos2 = (nome,telefone,localidade)* e *Loc = (código_postal,localidade).*

nome	telef	cPostal	localidade
Maria	1111	2815	Caparica
João	2222	1000	Lisboa
Pedro	1112	1100	Lisboa
Ana	3333	2815	Caparica



nome	telef	localidade
Maria	1111	Caparica
João	2222	Lisboa
Pedro	1112	Lisboa
Ana	3333	Caparica

cPostal	localidade
2815	Caparica
1000	Lisboa
1100	Lisboa

- Perdeu-se a informação de qual é o CP do João (e do Pedro)!!!
- O que torna esta decomposição diferente da primeira?
  - Na primeira, o atributo comum (*cPostal*) era chave numa das relações
  - Nesta, o atributo comum (*localidade*) não é chave em nenhuma
    - Logo, quando na 1ª se mete a localidade, não se consegue ir à 2ª ver qual é **o** cPostal que lhe corresponde

Objectivo: Desenvolver uma teoria para…

- Decidir se o esquema R já está num "bom" formato.
- Se não estiver, decompor R num conjunto de esquemas $\{R_1, R_2, ..., R_n\}$ tal que:
  - cada um deles esteja num "bom" formato
  - a decomposição seja sem perdas de informação
- A teoria é baseada em dependências funcionais

Dependências funcionais

- Restrições sobre o conjunto de relações possíveis.
- Exige que os valores num conjunto de atributos determinem univocamente os valores noutro conjunto de atributos.
- São uma generalização da noção de *chave.*

Dependências Funcionais (Cont.)

- Seja *R* o esquema duma relação, $\alpha \subseteq R$ *e* $\beta \subseteq R$. A dependência funcional:

$$\alpha \rightarrow \beta$$

é verdadeira em *R* sse, para toda a relação possível (i.e. *"que faça sentido"*) *r*(R), sempre que dois tuplos t_1 *e* t_2 de *r* têm os mesmos valores em α, também têm os mesmos valores em β:

$$\forall\, t_1, t_2 \in r,\ t_1[\alpha] = t_2[\alpha] \ \Rightarrow\ t_1[\beta] = t_2[\beta]$$

- Exemplo: Seja R=(**nome**, **telef**, **cPostal**, **localidade**).

localidade → ***cPostal*** **não** é verdadeira

cPostal → ***localidade*** é verdadeira

nome	telef	cPostal	localidade
Maria	1111	2815	Caparica
João	2222	1000	Lisboa
Pedro	1112	1100	Lisboa
Ana	3333	2815	Caparica

Dependências Funcionais (Cont.)

- Casos extremos:
  - $\{\} \rightarrow \alpha$: Só se verifica se em qualquer relação r possível, todos os tuplos tiverem mesmo valor em α (nunca deve ser permitido!)
  - $\alpha \rightarrow \{\}$ *:* Verifica-se para toda a relação r e conjunto de atributos α
- Diz-se que uma dependência é trivial se é satisfeita por todas as relações (quer façam sentido ou não) sobre um esquema
  - *E.g.*
    - *customer-name, loan-number* → *customer-name*
    - *customer-name* → *customer-name*
  - Em geral, $\alpha \rightarrow \beta$ é trivial se $\beta \subseteq \alpha$

Dependências Funcionais e Chaves

- *K* é uma superchave no esquema *R* sse $K \rightarrow R$
- *K* é uma chave candidata em *R* sse
  - $K \rightarrow R$, e
  - para nenhum $\alpha \subset K$, $\alpha \rightarrow R$
- As dependências funcionais permitem expressar restrições que não se expressam apenas com os conceitos de chave. E.g:

  (customer-name, loan-number, branch-name, amount).

  Espera-se que as seguintes dependências sejam verdadeiras:

  loan-number $\rightarrow$ *amount*
  loan-number $\rightarrow$ *branch-name*

  mas não se espera que a dependência abaixo seja verdadeira:

  loan-number $\rightarrow$ *customer-name*

Uso de Dependências Funcionais

- Usam-se dependências funcionais para:
  - Especificar restrições sobre as relações
    - Diz-se que *F* é verdadeiro em *R* se todas as relações (possíveis) sobre *R* satisfazem as dependências em *F.*
  - Testar (instâncias de) relações, para verificar se "fazem sentido" de acordo com as dependências funcionais.
    - Se uma relação *r* torna verdadeiras todas as dependências de um conjunto *F*, então diz-se que *r* satisfaz *F.*

Nota: Uma instância particular duma relação pode satisfazer uma dependência funcional mesmo que a dependência não seja verdadeira no esquema. Por exemplo, uma instância particular (em que, por acaso, nenhum empréstimo tenha mais que um cliente) satisfaz:

loan-number → *customer-name.*

Implicação de Dependências Funcionais

- Dado um conjunto *F* de dependências, há **outras** dependências que são logicamente implicadas por *F*.
  - Se nome → cPostal e cPostal → localidade, então, obrigatoriamente, nome → localidade
  - Em geral, se A → B e B → C, então, obrigatoriamente, A → C
- Quando usamos dependências funcionais para decompor tabelas, o que queremos saber é se há alguma dependência funcional, especificada originalmente ou implicada, que causa redundância
  - Para isto o que interessa saber é o conjunto de **todas** as dependências funcionais implicadas – **Fecho de Dependências**
- Quanto precisamos testar dependências funcionais, quanto menos tivermos para testar melhor
  - Aqui o que dava jeito era encontrar um **conjunto o mais pequeno e mais simples possível**, que implicasse todas as dependências – **Cobertura Canónica de Dependências**

Fecho de um Conjunto de Dependências Funcionais

- Ao conjunto de todas as dependências funcionais implicadas por *F* chama-se *fecho* de *F* (denotado por F^+).
- Podem encontrar-se todas as dependências em F^+ por aplicação dos Axiomas de Armstrong:
  - Se $\beta \subseteq \alpha$, então $\alpha \rightarrow \beta$ **(reflexividade)**
  - Se $\alpha \rightarrow \beta$, então $\gamma\ \alpha \rightarrow \gamma\ \beta$ **(aumento)**
  - Se $\alpha \rightarrow \beta$, e $\beta \rightarrow \gamma$, então $\alpha \rightarrow \gamma$ **(transitividade)**
- Estes regras são
  - coerentes (só geram dependências que pertencem a F^+) e
  - completas (geram todas as dependências pertencentes a F^+).

Exemplo

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B, \quad A \rightarrow C, \quad CG \rightarrow H, \quad CG \rightarrow I, \quad B \rightarrow H \}$
- alguns elementos de F^+:
  - $A \rightarrow H$
    - *transitividade a partir de* $A \rightarrow B$ *e* $B \rightarrow H$
  - $AG \rightarrow I$
    - *aumento de* $A \rightarrow C$ *com G, obtendo-se* $AG \rightarrow CG$
      e depois transitividade com $CG \rightarrow I$
  - $CG \rightarrow HI$
    - Aumento de CG → I (com CG) inferindo CG → CGI,
      aumento de CG → H (com I) inferindo CGI → HI,
      e depois transitividade

Construção de F^+

- Para calcular o fecho de um conjunto de dependências F:

  $F^+ = F$
  Repetir
    Para cada dependência funcional $f \in F^+$
      aplicar reflexividade e aumento em f
      adicionar os resultados a F^+
    Para cada par de dependências f_1 e $f_2 \in F^+$
      Se f_1 e f_2 podem combinar-se para usar transitividade
        Então adicionar a dependência resultado a F^+
  Até F^+ não mudar mais

NOTA: Veremos, mais tarde, outro procedimento para este problema

Fecho de Dependências (Cont.)

- Podemos facilitar a construção (manual) de F^+ usando mais regras coerentes:
  - Se $\alpha \rightarrow \beta$ e $\alpha \rightarrow \gamma$, então $\alpha \rightarrow \beta\gamma$ **(união)**
  - Se $\alpha \rightarrow \beta\gamma$, então $\alpha \rightarrow \beta$ e $\alpha \rightarrow \gamma$ **(decomposição)**
  - Se $\alpha \rightarrow \beta$ e $\gamma\beta \rightarrow \delta$, então $\alpha\gamma \rightarrow \delta$ **(pseudotransitividade)**
- Todas estas regras podem ser derivadas a partir dos Axiomas de Armstrong:
  - **União:** se $\alpha \rightarrow \beta$, por aumento $\alpha \rightarrow \alpha\beta$; se $\alpha \rightarrow \gamma$, por aumento $\alpha\beta \rightarrow \gamma\beta$; se $\alpha \rightarrow \alpha\beta$ e $\alpha\beta \rightarrow \gamma\beta$, por transitividade $\alpha \rightarrow \gamma\beta$
  - **Decomposição:** Por reflexividade, $\beta\gamma \rightarrow \beta$; se $\alpha \rightarrow \beta\gamma$ e $\beta\gamma \rightarrow \beta$, por transitividade $\alpha \rightarrow \beta$ -- semelhante para $\alpha \rightarrow \gamma$
  - **Pseudotransitividade:** Se $\alpha \rightarrow \beta$, por aumento $\alpha\gamma \rightarrow \gamma\beta$; se $\alpha\gamma \rightarrow \gamma\beta$ e $\gamma\beta \rightarrow \delta$, por transitividade $\alpha\gamma \rightarrow \delta$

Fecho de um Conjunto de Atributos

- Dado um conjunto de atributos α, define-se o *fecho* de α sobre F (denotado por α^+) como sendo o conjunto de atributos que dependem funcionalmente de α dado F. ie.:

$$\alpha \rightarrow \beta \in F^+ \Leftrightarrow \beta \subseteq \alpha^+$$

Algoritmo para calcular α^+

result := α;
Enquanto (mudança em *result*)
  Para cada $\beta \rightarrow \gamma$ **em** F
    begin
      Se $\beta \subseteq$ *result* **Então** *result* := *result* $\cup \gamma$
    end

Exemplo de fecho de atributos

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. $result = AG$
  2. $result = ABCG$ $(A \rightarrow C$ e $A \rightarrow B)$
  3. $result = ABCGHI$ $(CG \rightarrow H$ e $CG \rightarrow I$ e $CG \subseteq ABCG)$

Algoritmo:

result := α;
  Enquanto (mudança em *result*)
    Para cada $\beta \rightarrow \gamma$ **em** *F*
      begin
        Se $\beta \subseteq$ *result* **Então** *result* := *result* $\cup \gamma$
      end

- Será que *AG* é chave candidata? **Sim**
  1. Será AG superchave? **Sim**
     1. $AG \rightarrow R$? **Sim**
  2. E algum subconjunto próprio de AG é superchave? **Não**
     1. $A^+ \rightarrow R$? **Não: $A^+ = ABCH$**
     2. $G^+ \rightarrow R$? **Não: $G^+ = G$**

Uso de fecho de atributos

O algoritmo pode ser usado para vários fins:

- Testar superchaves:
  - Para testar se α é superchave, calcular α^+, e verificar se α^+ contém todos os atributos de *R*.
- Testar dependências funcionais
  - Para ver se a dependência $\alpha \rightarrow \beta$ é verdadeira (i.e. pertence a F^+), basta verificar se $\beta \subseteq \alpha^+$.
  - Ou seja, basta calcular α^+, e verificar se contém β.
  - É um teste simples e muito útil
- Cálculo do fecho de F
  - Para cada $\gamma \subseteq R$, calcular γ^+. Para cada $S \subseteq \gamma^+$, devolver como resultado a dependência $\gamma \rightarrow S$.

Coberturas de Conjuntos de DFs

- Um conjunto de dependências, pode conter algumas que são redundantes (por se poderem inferir das outras)
  - Eg: $A \rightarrow C$ é redundante em: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C\}$
  - Partes de dependências também podem ser redundantes
    - E.g. : $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow CD\}$ pode ser simplificado para $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$
    - E.g. : $\{A \rightarrow B, \quad B \rightarrow C, \quad AC \rightarrow D\}$ pode ser simplificado para $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$
- Um conjunto de dependências funcionais F é uma cobertura de (outro) conjunto de dependências funcionais G sse $F^+ = G^+$
- Intuitivamente, F é uma cobertura canónica de G se é uma cobertura de G e, além disso, F é "minimal" e nenhuma das dependência em F tem partes redundantes

Atributos dispensáveis

- Considere o conjunto de dependências *F* e a dependência

$$\alpha \rightarrow \beta \in F$$

  - O atributo A é dispensável (à esquerda) em α se $A \in \alpha$ e *F* implica $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - O atributo A é dispensável (à direita) em β se $A \in \beta$ e o conjunto $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ implica *F.*

- *Nota:* a implicação na direção oposta é trivial em ambos os casos (uma dependência mais forte, implica sempre uma mais fraca)
- Exemplo: Dado $F = \{A \rightarrow C, AB \rightarrow C\}$
  - *B* é dispensável em $AB \rightarrow C$ porque $A \rightarrow C$ implica $AB \rightarrow C$.
- Exemplo: Dado $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - *C* é dispensável em $AB \rightarrow CD$ pois a partir de $A \rightarrow C$, $AB \rightarrow D$ podemos inferir $AB \rightarrow$ CD

Teste para atributos dispensáveis

- Considere o conjunto *F* de dependências, e a dependência

$$\alpha \rightarrow \beta \in F.$$

- Para testar se $A \in \alpha$ é dispensável (à esquerda) em α
  1. calcular $(\alpha - \{A\})^+$ usando as dependências em *F*
  2. verificar se $(\alpha - \{A\})^+$ contém todos os atributos de β; se contém, então *A* é dispensável
- Para testar se $A \in \beta$ é dispensável (à direita) em β
  1. calcular α^+ usando as dependências em
     $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$,
  2. verificar se α^+ contém *A;* se contém, então *A* é dispensável

Cobertura Canónica

- Uma *cobertura canónica* de F é um conjunto de dependências F_c tal que
  - F implica todas as dependências em F_c, e
  - F_c implica todas as dependências em F, e
  - Nenhuma dependência em F_c contém atributos dispensáveis, e
  - O lado esquerdo de cada dependência em F_c é único.

Algoritmo para calcular uma cobertura canónica de F:

Repetir

Usar a regra da união para substituir as dependências em F

$\alpha_1 \rightarrow \beta_1$ e $\alpha_1 \rightarrow \beta_2$ por $\alpha_1 \rightarrow \beta_1 \beta_2$

Encontrar dependência $\alpha \rightarrow \beta$ com atributo dispensável (em α ou β)

Quando se encontra atributo dispensável, apaga-se de $\alpha \rightarrow \beta$

Até F não mudar

- Nota: A regra da união pode tornar-se aplicável depois de retirados alguns atributos dispensáveis. Por isso temos que a reaplicar.

Exemplo de cálculo de cobertura canónica

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$
- Combinar $A \rightarrow BC$ e $A \rightarrow B$ para $A \rightarrow BC$
  - Agora o conjunto é $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- A é dispensável em $AB \rightarrow C$ porque $B \rightarrow C$ implica $AB \rightarrow C$.
  - Agora o conjunto é $\{A \rightarrow BC, B \rightarrow C\}$
- C é dispensável em $A \rightarrow BC$ pois $A \rightarrow BC$ é implicado por $A \rightarrow B$ e $B \rightarrow$ C.
- Uma cobertura canónica é:

$$A \rightarrow B$$
$$B \rightarrow C$$

Conceitos da Teoria de Dependências Funcionais

- dependência funcional
- dependência funcional trivial
- superchave de uma relação dado um conjunto de dependências funcionais
- chave candidata de uma relação dado um conjunto de dependências funcionais
- fecho de um conjunto de dependências funcionais
- fecho de um conjunto de atributos sobre um conjunto de dependências funcionais
- cobertura de um conjunto de dependências funcionais
- atributos dispensáveis
- cobertura canónica de um conjunto de dependências funcionais

Teste de unicidade de chave candidata

- Existe uma condição necessária e suficiente para determinar a existência de chave candidata única:

  Dados R e F = $\{\alpha_1 \rightarrow \beta_1, \ldots, \alpha_p \rightarrow \beta_p\}$, o esquema de relação R tem chave única sse R – ($\gamma_1 \ldots \gamma_p$) é super-chave, onde

  - $\gamma_i = \beta_i - \alpha_i$ para $1 \leq i \leq p$.

- Exemplo:

  Considere-se o esquema de relação R(ABCDE) e o conjunto de dependências funcionais F = {AB → C,AC → B,D → E}.

  Para AB → C tem-se γ_1 = {C}

  Para AC → B tem-se γ_2 = {B}

  Para D → E tem-se γ_3 = {E}

  Logo $\gamma_1\gamma_2\gamma_3$ = {B,C,E}, e portanto R – $\gamma_1\gamma_2\gamma_3$ = {A,D}.

  Como, AD^+ = ADE ≠ R temos que AD não é super-chave e consequentemente existe mais de uma chave-candidata.