Bases de Dados

Universidade do Porto

Faculdade de Engenharia

FEUP

- INTRODUÇÃO
- MODELOS CONCEPTUAIS
  - Diagrama de Classes UML
  - Modelo Entidade-Associação (E-A)
- **MODELO RELACIONAL**
- LINGUAGEM DE DEFINIÇÃO DE DADOS (LDD)
- INTERROGAÇÃO DE DADOS
  - Álgebra relacional
  - Linguagem de Manipulação de Dados (LMD)

Observação: parcialmente baseado em slides desenvolvidos pelo Prof. Jeffrey D. Ullman

Índice

- Relações e atributos
- Dependências funcionais
- Chaves de relações
- Inferência de dependências funcionais
- Formas normais

Relações e atributos

O modelo relacional caracteriza-se por um conjunto de relações cada uma delas identificada por um nome e um conjunto de atributos

Exemplo

Valencia(idValencia, nome, descricao) - Relação Valencia com os atributos idValencia, nome e descricao.

Dependências funcionais

- Os atributos das relações relacionam-se entre si com base em regras a que se chamam dependências funcionais
- *X* ->*Y* é uma dependência funcional sobre a relação *R* dizendo que quaisquer dois tuplos de *R* que sejam iguais em todos os atributos de *X*, têm também de ser iguais em todos os atributos de *Y*.
  - Diz-se "*X* ->*Y* verifica-se em *R*."
  - Notação: ..., *X*, *Y*, *Z* representam conjuntos de atributos; *A*, *B*, *C*,... representam atributos isolados.

Dependências funcionais

Separação dos lados direitos

- $X->A_1A_2...A_n$ verifica-se em R exactamente quando cada um dos

  $X->A_1$, $X->A_2$,..., $X->A_n$ se verifica em R.
- Exemplo: $A->BC$ é equivalente a $A->B$ e $A->C$.
- Não há regras de separação para os lados esquerdos.
- Habitualmente expressam-se as DF's pondo um só atributo do lado direito.

Dependências funcionais

Exemplo

Encomendas(idEncomenda, dataEncomenda,
idProduto, nomeProduto, quantidade,
precoUnit, idCliente, nomeCliente)

DF's que se podem assumir:

- idEncomenda -> dataEncomenda
- idEncomenda -> idCliente
- idProduto -> nomeProduto
- idProduto -> precoUnit
- idCliente -> nomeCliente
- idEncomenda, idProduto -> quantidade

Chaves de relações

- *K* é uma *super-chave* da relação *R* se

  K determina funcionalmente todos os atributos de *R*.
- *K* é uma *chave (também se pode designar por chave candidata)* de *R* se

  K é super-chave, e nenhum subconjunto próprio de *K* é super-chave.
  - Um subconjunto diz-se próprio se estiver contido mas não for igual ao conjunto *K*

Chaves de relações

Super-chave: exemplo

Encomendas(idEncomenda, dataEncomenda,
idProduto, nomeProduto, quantidade,
precoUnit, idCliente, nomeCliente)

{idEncomenda, idProduto, quantidade} é uma super-chave porque estes atributos determinam todos os outros atributos.

- idEncomenda -> dataEncomenda idCliente nomeCliente
- idProduto -> nomeProduto precoUnit

Chaves de relações

Chave: exemplo

- {idEncomenda, idProduto, quantidade} não é uma chave porque existe um seu subconjunto próprio {idEncomenda, idProduto} que é uma super-chave.

- Não há mais chaves para a relação Encomendas, mas há muitas super-chaves.
  - Qualquer super-conjunto de {idEncomenda, idProduto}.
    - Um super-conjunto do conjunto S é um conjunto que contem S, i.e., possui todos os elementos de S e possivelmente mais alguns.

Chaves de relações

Chave primária

Designa-se por chave primária uma e uma só chave, de entre as chaves dessa relação.

- O conceito de chave primária não é importante no modelo relacional. No entanto, os SGBD costumam exigir a definição de uma chave primária. Por isso, é costume assinalar a chave primária de uma relação no modelo relacional (atributos sublinhados).
  - SGBD = Sistema Gestor de Bases de Dados (Ex.: MySQL, Oracle, SQLite, ...)

Encomendas(idEncomenda, dataEncomenda, idCliente)

Chaves de relações

Chave externa

Designa-se por chave externa o conjunto de atributos de uma relação que é chave de uma outra relação.

Encomenda(idEncomenda, dataEncomenda, idCliente -> Cliente)
Cliente(idCliente, nomeCliente)
Quants(idEncomenda -> Encomenda, idProduto -> Produto, quantidade)
Produto(idProduto, nomeProduto)

Inferência de DFs

- São-nos dadas as DFs X_1 -> A_1, X_2 -> A_2,..., X_n -> A_n, e queremos saber se uma DF Y -> B é garantida numa relação que satisfaça as DFs dadas.
  - Exemplo das encomendas: Se *idEncomenda* -> *idCliente* e *idCliente* -> *nomeCliente* se verificam, seguramente *idEncomenda* -> *nomeCliente* também se verifica, mesmo que tal não seja dito de forma explícita.

Inferência de DFs

Teste de inferência

1. Para testar se Y -> B, começar por assumir que dois registos têm todos os atributos de Y iguais.

← Y →

OOOOOOO. . . O

OOOOO?? . . . ?

2. Usar as DFs dadas para inferir se esses registos também têm de coincidir noutros atributos.
   - Se B é um desses atributos, então Y -> B é verdade.
   - Caso contrário, os dois registos, com algumas igualdades forçadas, formam uma relação de dois registos provando que Y -> B não provem das DFs dadas.

Inferência de DFs

Teste de inferência

Uma forma fácil de testar se Y-> B provem das DFs dadas é através do cálculo do *fecho* de Y (denota-se por Y^+).

- Base: $Y^+ = Y$.
- Indução: Procurar por lados esquerdos (X) das DFs que sejam sub-conjuntos do actual Y^+. Se a DF é X -> A, acrescentar A a Y^+.
- Conclusão: Se no final B pertencer a Y^+, então a DF Y -> B provem das DFs dadas sendo desnecessário acrescentá-la às existentes.

Inferência de DFs

Teste de inferência: indução de DFs para subconjuntos de atributos da relação original

Ideia base

1. Começar com as DFs dadas e procurar todas as DFs *não triviais* deduzíveis a partir das DFs dadas.
   - Não trivial = lado direito não está contido no lado esquerdo.
2. Utilizar apenas as DFs que envolvem só atributos do subconjunto pretendido.

Inferência de DFs

Teste de inferência: indução

Simples, Algoritmo Exponencial

1. Para cada conjunto de atributos X, calcular X^+.
2. Acrescentar $X \rightarrow A$ para todos os A em $X^+ - X$.
3. Apagar $XY \rightarrow A$ sempre que se descobrir $X \rightarrow A$.
   - Porque $XY \rightarrow A$ provém de $X \rightarrow A$ em qualquer projeção.
4. Finalmente, usar só DFs que envolvam atributos projetados.

Inferência de DFs

Teste de inferência: exemplo

- *ABC* com as DFs *A* ->*B* e *B* ->*C*.
- Objectivo: determinar as DFs para o subconjunto {A,C}
- Iterações:
  - $A^+ = ABC$; logo *A* ->*B*, *A* ->*C*.
    - Não é necessário calcular AB^+ ou AC^+.
  - $B^+ = BC$; logo *B* ->*C*.
    - Não é necessário calcular BC^+.
  - $C^+ = C$; não assegura nenhuma DF não trivial.
- DFs resultantes: *A* ->*B*, *A* ->*C* e *B* ->*C*.
- Nota: não se procurou AB^+ nem AC^+ porque pertencem a A^+. Nem BC^+ (porque ...), CA^+ (=AC^+), ou CB^+ (=BC^+).

Inferência de DFs

Teste de inferência: exemplo

- Projectar exemplo anterior em AC
- Projecção em AC : A ->C.
  - Só as DFs que envolvem subconjuntos de {A,C }.

Formas normais

Exemplo

Pretende-se armazenar a informação relativa a uma época do campeonato de Fórmula 1.

De cada marca participante no campeonato pretende-se armazenar o seu nome, país de origem, nº actual de pontos no campeonato de marcas e quais os carros inscritos. De cada carro interessa saber o seu peso, potência e velocidade máxima.

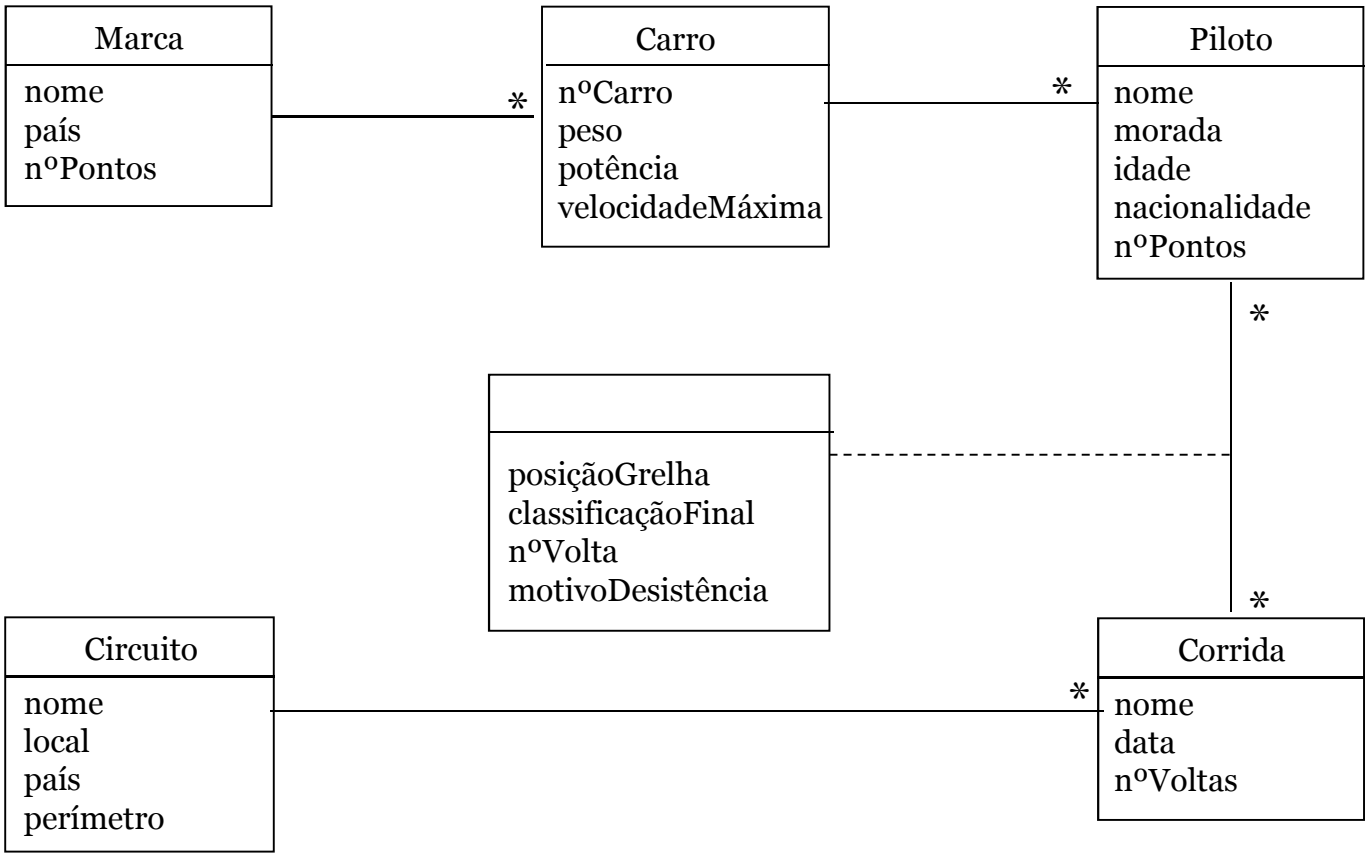
Relativamente aos pilotos participantes é necessário conhecer o seu nome, morada, idade, nacionalidade e nº actual de pontos no campeonato de pilotos. Um piloto só pode conduzir um carro ao longo da época, embora um determinado carro possa ser conduzido por mais de um piloto. Esta situação, embora não muito frequente, pode surgir, por exemplo, devido ao afastamento de um piloto ferido num acidente.

Cada época é constituída por um conjunto de corridas que se realizam em circuitos e em datas definidas no início da época. Para uma determinada corrida pode, ao longo da época e por razões várias, ser alterado o circuito onde esta se realiza. Em situações excepcionais pode acontecer também a realização de duas corridas no mesmo circuito. De cada circuito pretende-se saber o nome, local, país, nº de voltas e perímetro.

No que diz respeito à realização de uma corrida interessa saber quais os pilotos que participaram, as posições que ocuparam na grelha de partida e a classificação final. Relativamente à corrida interessa também saber quais os pilotos que desistiram, em que volta ocorreu e qual o motivo da desistência.

Formas normais

Exemplo: Diagrama de classes UML



Formas normais

1ª FN

O domínio de cada atributo contém apenas valores atômicos, e o valor de cada atributo contém apenas um único valor do domínio.

Exemplo

Classificação(nome_marca, país_origem, pontuação_marca, num_carro, peso_carro, potência_carro, vel_max, nome_piloto, morada_piloto, idade_piloto, nacionalidade_piloto, pontuação_piloto, nome_circuito, local_circuito, país_circuito, num_voltas_circuito, perímetro, nome_corrida, data, posição_grelha, classificação, motivo_desistência, num_voltas_realizadas)

Formas normais

2ª FN

Está na 1ª FN e nenhum atributo não primo da relação é funcionalmente dependente de um subconjunto próprio de uma chave candidata

- Atributo primo = pertence a alguma chave candidata da relação.

Exemplo

Piloto(nome_piloto, morada_piloto, idade_piloto, nacionalidade_piloto, pontuação_piloto, nome_marca, país_origem, pontuação_marca, num_carro, peso_carro, potência_carro, vel_max)

Corrida(nome_corrida, data, nome_circuito, local_circuito, país_circuito, num_voltas_circuito, perímetro)

Classificação(nome_piloto->Piloto, nome_corrida->Corrida, posição_grelha, classificação, motivo_desistência, num_voltas_realizadas)

Formas normais

3ª FN

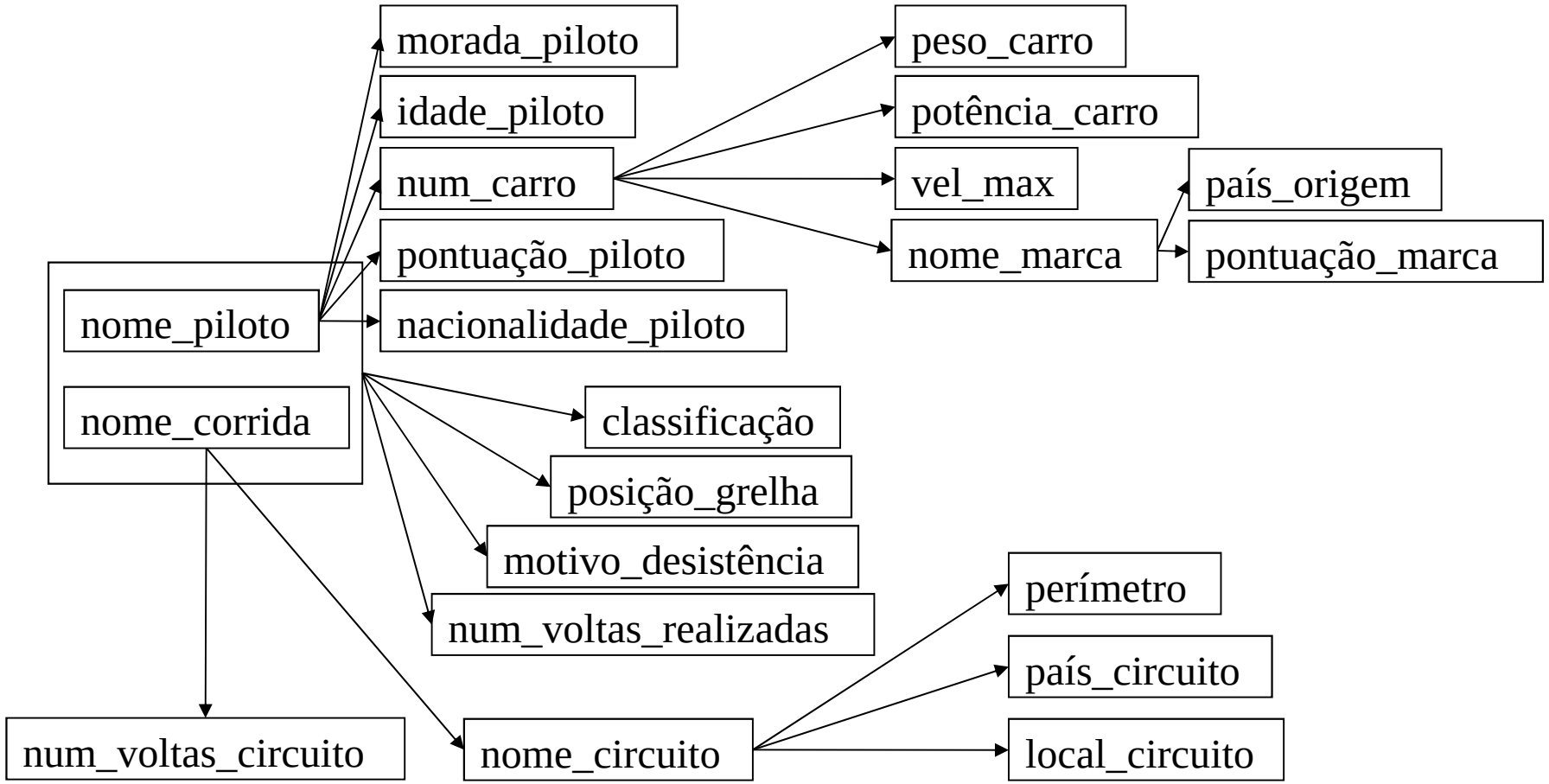
Está na 2ª FN e todos os atributos não primos da relação dependem funcionalmente de todas as chaves candidatas da relação de forma não transitiva

Exemplo

Piloto(nome_piloto, morada_piloto, idade_piloto, nacionalidade_piloto, pontuação_piloto, num_carro->Carro)
Carro(num_carro, peso_carro, potência_carro, vel_max, nome_marca->Marca)
Marca(nome_marca, país_origem, pontuação_marca)
Corrida(nome_corrida, data, nome_circuito->Circuito, num_voltas_circuito)
Circuito(nome_circuito, local_circuito, país_circuito, perímetro)
Classificação(nome_piloto->Piloto, nome_corrida->Corrida, posição_grelha, classificação, motivo_desistência, num_voltas_realizadas)

Formas normais

Diagrama de dependências funcionais: exemplo



Formas normais

Normalização vs. espaço ocupado pela BD

Sabendo que existem 11 equipas, 22 carros, 28 pilotos, 16 corridas e 16 circuitos e ainda que em cada corrida participam exactamente 22 pilotos, **estime o espaço ocupado pelas tabelas nas 1ª, 2ª e 3ª formas normais**.

Admita que o espaço ocupado por cada atributo é de 4 bytes para os atributos numéricos e para as datas e de 30 bytes para os atributos texto.

1ª FN: **Classificação**(nome_marca, país_origem, pontuação_marca, num_carro, peso_carro, potência_carro, vel_max, nome_piloto, morada_piloto, idade_piloto, nacionalidade_piloto, pontuação_piloto, nome_circuito, local_circuito, país_circuito, num_voltas_circuito, perímetro, nome_corrida, data, posição_grelha, classificação, motivo_desistência, num_voltas_realizadas)

2ª FN: **Piloto**(nome_piloto, morada_piloto, idade_piloto, nacionalidade_piloto, pontuação_piloto, nome_marca, país_origem, pontuação_marca, num_carro, peso_carro, potência_carro, vel_max)
Corrida(nome_corrida, data, nome_circuito, local_circuito, país_circuito, num_voltas_circuito, perímetro)
Classificação(nome_piloto->Piloto, nome_corrida->Corrida, posição_grelha, classificação, motivo_desistência, num_voltas_realizadas)

3ª FN: **Piloto**(nome_piloto, morada_piloto, idade_piloto, nacionalidade_piloto, pontuação_piloto, num_carro->Carro)
Carro(num_carro, peso_carro, potência_carro, vel_max, nome_marca->Marca)
Marca(nome_marca, país_origem, pontuação_marca)
Corrida(nome_corrida, data, nome_circuito->Circuito, num_voltas_circuito)
Circuito(nome_circuito, local_circuito, país_circuito, perímetro)
Classificação(nome_piloto->Piloto, nome_corrida->Corrida, posição_grelha, classificação, motivo_desistência, num_voltas_realizadas)

Formas normais

Normalização vs. espaço ocupado pela BD

1ª Forma Normal

EspaçoOcupado =

22x16x(30+30+4+4+4+4+4+30+30+4+30+4+30+30+30+4+4+30+4+4+4+30+4)

= 22x16x352 = **123 904 bytes**

2ª Forma Normal

EspaçoOcupado = 28x(30+30+4+30+4+30+30+4+4+4+4+4)

+ 16x(30+4+30+30+30+4+4)+ 22x16x(30+30+4+4+30+4)

= 28x178 + 16x132 + 22x16x102 = **52 968 bytes**

3ª Forma Normal

EspaçoOcupado = 28x(30+30+4+30+4+4) + 22x(4+4+4+4+30) + 11x(30+30+4)

+ 16x(30+4+4+30) + 16x(30+30+30+4) + 22x16x(30+30+4+4+30+4)

= 28x102 + 22x46 + 11x64 + 16x68 + 16x94 + 22x16x102 = **43 068 bytes**

Nota: *os cálculos apresentados não são adequados para estimar o espaço efectivamente ocupado pelos dados. Realizam-se para fins exclusivamente pedagógicos.*

Formas normais

Normalização vs. facilidade e rapidez de consulta

O Comando SELECT da Linguagem SQL (a forma mais simples):

```
SELECT lista_de_campos
FROM lista_de_tabelas
[WHERE lista_de_condições];
```

- Os [] indicam que é opcional

EXEMPLO: Seleccionar o nome de todos os pilotos alemães

```
SELECT nome_piloto
FROM piloto
WHERE nacionalidade_piloto="Alemanha";
```

Formas normais

Normalização vs. facilidade e rapidez de consulta

Usando a linguagem SQL diga quais as marcas que já obtiveram o 1º lugar da grelha de partida utilizando, para tal, as tabelas na 1ª, 2ª e 3ª Forma Normal.

1ª Forma Normal

```
SELECT nome_marca FROM Classificação
WHERE posição_grelha=1;
```

2ª Forma Normal

```
SELECT nome_marca FROM Classificação, Piloto
WHERE posição_grelha=1
AND Classificação.nome_piloto = Piloto.nome_piloto;
```

3ª Forma Normal

```
SELECT nome_marca FROM Classificação, Piloto, Carro
WHERE posição_grelha=1 AND Classificação.nome_piloto = Piloto.nome_piloto
AND Piloto.num_carro = Carro.num_carro;
```

Formas normais

Exemplo

Cervejas(nome, empr, moradaEmpr)

DFs: nome->empr, empr->moradaEmpr

- A única chave é {nome}. Porquê?
- Em que forma normal se encontra esta relação?

Formas normais

Normalização

Actualmente não se desenham modelos relacionais recorrendo a este processo de normalização.

Porquê?

- Porque fazendo o mapeamento do modelo conceptual (por ex.: diagrama de classes UML ou modelo E-A) para o relacional, as relações já estão, tipicamente, na 3ª FN.
- Mas é importante verificar se o modelo relacional que resulta do mapeamento do modelo conceptual se encontra na forma normal pretendida.

Formas normais

Normalização

Muito importante:

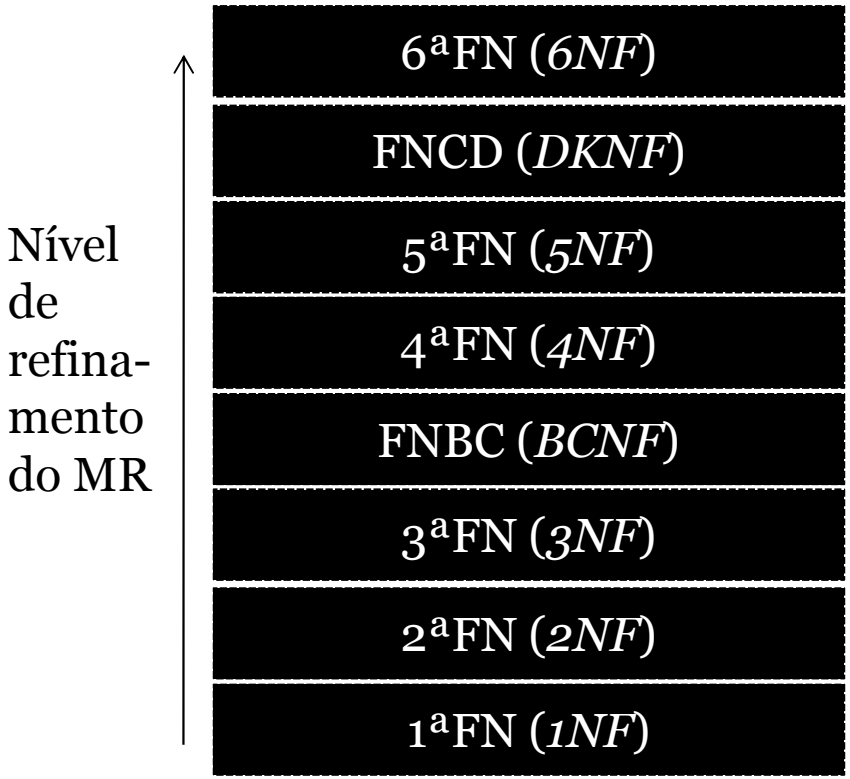
Uma relação na 2ª FN também está na 1ª FN.

Uma relação na 3ª FN também está na 2ª FN.

Mas garantir a 3ª FN pode não ser suficiente ...

Formas normais

Normalização



C.J. Date, H. Darwen e N. Lorentzos (2002)

Ronald Fagin (1981)

Ronald Fagin (1979)

Ronald Fagin (1977)

Raymond F. Boyce and E.F. Codd (1974)

E.F. Codd (1971)

E.F. Codd (1971)

E.F. Codd (1970), C.J. Date (2003)

Formas normais

Normalização

Garantindo que uma base de dados se encontra na forma normal de Boyce-Codd (FNBC) e na 3ª FN, cobre-se a maior parte dos problemas que se encontram na prática.

As formas normais da 4ª em diante não têm tanto interesse prático como a FNBC sendo, habitualmente, leccionadas em tópicos mais avançados sobre Bases de Dados.

Formas normais

Forma normal de Boyce-Codd

- Dizemos que uma relação está na *FNBC* (Forma Normal de *Boyce-Codd*) quando: sempre que existe alguma DF não trivial de *R* em que *X* ->*Y*, *X* é uma super-chave.
  - Lembrar: *não trivial* significa que *Y* não está contido em *X*.
  - Lembrar: uma *super-chave* é qualquer super conjunto de uma chave (não necessariamente um super-conjunto próprio).

Formas normais

Exemplo

CodPostais(localidade, ruaEnº, codPostal)

DFs: localidade ruaEnº -> codPostal, codPostal -> localidade

- Quais as chaves candidatas de *CodPostais*?
- Em que forma normal se encontra esta relação?
- O que devo alterar para garantir a forma normal de Boyce-Codd?

Formas normais

Exemplo

Clientes(nome, morada, cervQGosta, empr, cervFav)

DFs: nome->morada cervFav; cervQGosta->empr.

- A única chave é {nome, cervQGosta}.
- Em cada DF, o lado esquerdo *não* é uma super-chave.
- Qualquer uma destas DFs mostra que *Clientes* não está na FNBC.
- *Clientes* também não está na 3ª forma normal! Porque …
- Nem na 2ª forma normal! Porque …
- Mas está na 1ª FN! Porque …

Formas normais

Forma normal de Boyce-Codd: decomposição

- Dados de entrada: relação *R* com as DFs *F*.
- Procurar por entre as DFs dadas uma DF $X \rightarrow Y$ que viole a FNBC.
- Calcular X^+.
  - Se der todos os atributos é porque X é uma super-chave logo, essa DF não viola a FNBC, ou seja, qualquer coisa correu mal …
- Substituir R pelas seguintes relações:
  1. $R_1 = X^+$.
  2. $R_2 = R - (X^+ - X)$.
- Projectar as DFs F dadas em duas novas relações.

Formas normais

Forma normal de Boyce-Codd: exemplo

CodPostais(localidade, ruaEnº, codPostal)

DFs: localidade ruaEnº -> codPostal, codPostal -> localidade

- Verificar se há violação da FNBC em,
  localidade ruaEnº -> codPostal
  codPostal -> localidade.
- Calcular o fecho do lado esquerdo:
  $\{codPostal\}^+ = \{localidade, codPostal\}$.
- Relações decompostas:
  1. $R_1 = X^+$: CodPostais1(codPostal, localidade)
  2. $R_2 = R - (X^+ - X)$: CodPostais2(codPostal, ruaEnº)

Formas normais

Forma normal de Boyce-Codd: exemplo

- Ainda não acabou; precisamos de verificar se CodPostais1 e CodPostais2 estão na FNBC.
- Para CodPostais1(codPostal, localidade), a única DF relevante é codPostal->localidade.
  - Assim, {codPostal} é a única chave e CodPostais1 está na FNBC.
- Para CodPostais2(codPostal, ruaEnº), não há dependências funcionais.
  - Assim, {codPostal, ruaEnº} é a única chave e CodPostais2 está na FNBC.

Formas normais

Problema encontrado com a decomposição na FNBC

- A DF localidade ruaEnº -> codPostal deixou de ser garantida.
- Para que uma DF seja assegurada, todos os seus atributos têm de estar numa mesma relação.

Formas normais

Garantir a FNBC pode não ser suficiente: exemplo

FNBC:

ruaEnº	codPostal
Av. da liberdade, 265	1250-144
Av. da liberdade, 265	1250-145

localidade	codPostal
Lisboa	1250-144
Lisboa	1250-145

DF: codPostal -> localidade

Juntando registos com ruaEnº iguais dá:

ruaEnº	localidade	codPostal
Av. da liberdade, 265	Lisboa	1250-144
Av. da liberdade, 265	Lisboa	1250-145

Apesar de nenhuma DF ser violada nas relações decompostas em FNBC, a DF ruaEnº localidade -> codPostal é violada pela base de dados como um todo.

Formas normais

Garantir a FNBC pode não ser suficiente

- Existem conjuntos de DFs que criam problemas na decomposição.
- *AB* ->*C* e *C* ->*B*.
  - Exemplo: *A* = rua e nº, *B* = cidade, C = código postal.
- Existem duas chaves, {*A*,*B* } e {*A*,*C* }.
- *C* ->*B* é uma violação da FNBC, como tal temos de a decompor em *AC*, *BC*.

Formas normais

Garantir a FNBC pode não ser suficiente

- O problema é que se usarmos as relações *AC* e *BC*, não podemos garantir a DF *AB* ->*C* pela verificação das DFs nas relações decompostas.
- Exemplo com *A* = ruaEnº, *B* = cidade e *C* = codPostal tal como apresentado anteriormente.

Formas normais

Propriedades desejáveis da decomposição

- A decomposição deve ter duas propriedades importantes:
  1. *Junção sem perdas*: deverá ser possível projectar em relações decompostas, e depois reconstruir a original.
  2. *Preservação das dependências*: deverá ser possível verificar nas relações projectadas se todas as DF são satisfeitas.

Formas normais

Propriedades desejáveis da decomposição

- Podemos garantir *Junção sem perdas* com a decomposição FNBC.
- Podemos assegurar tanto *Junção sem perdas* como *Preservação das dependências* com a decomposição 3FN (tema a abordar já de seguida).
- Mas nunca conseguimos garantir *Junção sem perdas* e *Preservação das dependências* com a decomposição FNBC.
  - ruaEnº-localidade-codPostal é um exemplo.

Formas normais

Garantir a FNBC quando não há preservação das dependências

CodPostais(localidade, ruaEnº, codPostal)

DFs: localidade ruaEnº -> codPostal,
codPostal -> localidade

Neste exemplo, a decomposição na FNBC não garante a preservação das dependências: DF localidade ruaEnº -> codPostal

Nestes casos, a única maneira de ultrapassar o problema é redefinindo os atributos e, por inerência, as DFs.

CodPostais(localidade, ruaEnº, CP1, CP2)

DFs: CP1 -> localidade,
CP1, ruaEnº -> CP2

Exemplo: codPostal=4200-465;
Dá origem a: CP1=4200; CP2=465.

Exercício: Faça a decomposição desta relação na FNBC e verá que agora é possível decompor na FNBC garantindo a preservação das dependências.

Formas normais

Teste da caça: para aferir das junções sem perdas

- Exemplo:

Decomposição de R(A,B,C,D) em S1(A,D), S2(A,C) e S3(B,C,D).

DFs: A->B; A->C; CD->A.

- Como verificar se esta decomposição permite a *Junção sem perdas*?

Formas normais

Teste da caça: para aferir das junções sem perdas

Quadro inicial:
Põe-se uma linha por cada relação decomposta existente.
Coloca-se uma letra designativa do atributo por cada atributo na relação decomposta.
Coloca-se uma letra designativa do atributo com um posfixo designativo da linha por cada atributo que não esteja na relação decomposta.

Exemplo:

A	B	C	D	
a	b1	c1	d	→ S1(A,D)
a	b2	c	d2	→ S2(A,C)
a3	b	c	d	→ S3(B,C,D)

Formas normais

Teste da caça: para aferir das junções sem perdas

Iterações:
Sempre que o lado esquerdo de uma DF seja comum em duas linhas do quadro, substituem-se os atributos que aparecem no lado direito da DF preferencialmente por valores sem posfixo de forma a que as duas linhas com lados esquerdos iguais nessa DF fiquem também com lados direitos iguais

Exemplo: A->B; A->C; CD->A

A	B	C	D	
a	b1	c	d	⟶ A->C
a	b2	c	d2	
a	b	c	d	⟶ CD->A

Formas normais

Teste da caça: para aferir das junções sem perdas

Conclusão:
Se se conseguir obter uma linha sem posfixos significa que a decomposição garante junções sem perdas.
Caso todas as linhas tenham pelo menos um atributo com posfixo então a decomposição não garante que a junção seja feita sem perdas.

Exemplo: A->B; A->C; CD->A

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

Uma linha com todos os atributos sem posfixo => junção sem perdas

Formas normais

Decomposição 3FN

- Podemos sempre construir uma decomposição em relações 3FN sem perdas nas junções e com preservação das dependências.
- Requere *base mínima* para as DFs:
  1. Os lados direitos têm um só atributo.
  2. Nenhuma DF pode ser removida.
  3. Nenhum atributo pode ser removido dos lados esquerdos.

Formas normais

Decomposição 3FN: construção da base mínima

1. DFs com lados direitos de um só atributo.
2. Repetidamente tentar remover uma DF e ver se as DFs restantes são equivalentes à original.
3. Repetidamente tentar remover um atributo do lado esquerdo e ver se as DFs resultantes são equivalentes à original.

Formas normais

Decomposição 3FN: síntese

- Uma relação por cada lado esquerdo diferente nas DFs na base mínima.
  - Cada relação é a união dos lados direito e esquerdo das DFs com lados esquerdos iguais.
- Se nenhuma das relações criadas tiver uma super-chave, acrescentar uma relação com uma chave da relação R original.

Formas normais

Decomposição 3FN: exemplo

- Relação R = ABCD.
- DFs *A->B* e *A->C*.
- Decomposição: R1=AB e R2=AC a partir das DFs, mais R3=AD para a chave.

Formas normais

Decomposição 3FN

Garante:

- Preservação das dependências funcionais;
- Junção sem perdas;
- 3FN.

Formas normais

Exemplos

1. Faça a decomposição para garantia da FNBC do exemplo:

Clientes(nome, morada, cervQGosta, empr, cervFav)

DFs: nome->morada cervFav; cervQGosta->empr.

2. Faça a decomposição 3FN para o mesmo exemplo.

Formas normais

Decomposição FNBC: exemplo

Clientes(<u>nome</u>, morada, <u>cervQGosta</u>, empr, cervFav)

F = nome->morada; nome -> cervFav;
cervQGosta ->empr.

- Verificar se há violação da FNBC em, nome -> morada.
- Calcular o fecho do lado esquerdo: $\{nome\}^+$ = {nome, morada, cervFav}.
- Relações decompostas:
  1. Clientes1(<u>nome</u>, morada, cervFav)
  2. Clientes2(<u>nome</u>, <u>cervQGosta</u>, empr)

Formas normais

Decomposição FNBC: exemplo

- Ainda não acabou; precisamos de verificar se Clientes1 e Clientes2 estão na FNBC.
- Para este caso a projecção das DFs é fácil.
- Para Clientes1(nome, morada, cervFav), as DFs relevantes são nome->morada e nome->cervFav.
  - Assim, {nome} é a única chave e Clientes1 está na FNBC.

Formas normais

Decomposição FNBC: exemplo

- Para Clientes2(nome, cervQGosta, empr), a única DF é cervQGosta->empr, e a única chave é {nome, cervQGosta}.
  - Violação da FNBC.
- cervQGosta$^+$ = {cervQGosta, empr}, por isso decompomos *Clientes2* em:
  1. Clientes3(cervQGosta, empr), com a DF cervQGosta->empr
  2. Clientes4(nome, cervQGosta), sem DFs

Formas normais

Decomposição FNBC: exemplo

- A decomposição resultante de *Clientes*:
  1. Clientes1(nome, morada, cervFav)
  2. Clientes3(cervQGosta, empr)
  3. Clientes4(nome, cervQGosta)
- Nota: *Clientes1* é sobre clientes, *Clientes3* sobre cervejas, e *Clientes4* sobre a relação entre os clientes e as cervejas de que eles gostam.

Formas normais

Decomposição 3FN: exemplo

Clientes(nome, morada, cervQGosta, empr, cervFav)
F = nome->morada, nome -> cervFav, cervQGosta ->empr

1. DFs com lados direitos de um só atributo: ok.
2. Repetidamente tentar remover uma DF e ver se as DFs restantes são equivalentes à original: ok.
3. Repetidamente tentar remover um atributo do lado esquerdo e ver se as DFs resultantes são equivalentes à original: ok.
4. Uma relação por cada lado esquerdo diferente nas DFs:
   a) Clientes1(nome, morada, cervFav), com as DFs nome->morada e nome->cervFav.
   b) Clientes2(cervQGosta, empr), com a DF cervQGosta->empr.
5. Se nenhuma das relações criadas tiver uma super-chave, acrescentar uma relação com uma chave da relação R original:
   c) Clientes3(nome, cervQGosta), sem DFs.