

2018/2019 LEI, LEI-PL e LEI-CE (ECS) Bases de Dados

Fernanda Brito Correia

João Costa

Normalização - 1ªNF à BCNF

- ▶ Uma empresa pretende guardar informação sobre a quantidade total de peças fornecidas pelos seus fornecedores.

- ▶ Informação a registar:

S#, Sname, Status, City, P#, Pname, Color, Weight, City, QTY

- ▶ Consideremos a base de dados:

S <S#, Sname, Status, City>

P <P#, Pname, Color, Weight, City>

SP <S#, P#, QTY>, S# FK S, P# FK P

- ▶ Esta estrutura, intuitivamente parece-nos correcta.

- Suponhamos que o atributo **city** em vez de estar na relação (S) **Fornecedores** está na relação (SP) **Fornecimentos** (intuitivamente o lugar errado, uma vez que esse atributo “cidade dos Fornecedores” diz respeito aos **Fornecedores** e não aos **Fornecimentos**).

Relação SP' (Fornecimentos'):

| S# | CITY | P# | QTY |
|-----|--------|----|-----|
| S1 | LONDON | P1 | 300 |
| S1 | LONDON | P2 | 200 |
| S1 | LONDON | P3 | 400 |
| S1 | LONDON | P4 | 200 |
| S1 | LONDON | P5 | 100 |
| S1 | LONDON | P6 | 100 |
| S2 | PARIS | P1 | 300 |
| S2 | PARIS | P2 | 400 |
| ... | | | |

► 3

- A base de dados contém agora um elevado grau de redundância:
 - Todo o tuplo de SP' para o fornecedor S1 diz-nos que S1 está localizado em London.
 - Todo o tuplo de SP' para o fornecedor S2 diz-nos que S2 está localizado em Paris.
- Esta redundância conduz a vários problemas:
 - Por exemplo o Fornecedor S1 estar em London num tuplo e em Portugal noutro.

UM FACTO NUM SÓ LUGAR

Deve-se evitar a redundância sempre que possível.

► 4

Database Design Schema Design

- ▶ Quando fazemos o design de uma base de dados, estamos interessados nas propriedades dos dados que são sempre verdadeiras, não naquelas que são verdadeiras em determinados instantes.

▶ 5

Exemplo

- ▶ A propriedade “Toda a peça tem um peso” é sempre verdadeira.
- ▶ A propriedade “Toda a peça vermelha é armazenada em Londres” não é sempre verdadeira.
- ▶ **Objectivo do Schema**
 - ▶ Capturar aquelas propriedades que são sempre verdadeiras.

▶ 6

- ▶ Vamos estar mais preocupados com a parte do esquema (*HEADING*) de uma relação, **que é independente do tempo**, do que com a parte dos valores (*BODY*)
- ▶ As BD Relacionais estão sempre normalizadas no sentido de que são **atómicas**, mas apesar disso contém **propriedades indesejáveis** (caso de SP')
- ▶ A teoria da normalização permite-nos reconhecer tais casos e mostrar como tais relações podem ser convertidas para uma forma mais desejável.
- ▶ No caso de SP' a teoria diz-nos o que está de errado com essa relação e dir-nos-á para a dividir em duas relações mais desejáveis, nomeadamente as relações S e SP já nossas conhecidas.
- ▶ **Normalização** ☒ tipos de colunas e relação entre tabelas, etc. (esquema ou "Schema").

▶ 7

Cada "estádio de normalização" constitui uma "Forma Normal".

- ▶ Actualmente encontram-se definidas a **1ª, 2ª, 3ª, BCNF, 4ª e 5ª** formas normais. Haverá mais? Sim, baseadas noutros operadores.
- ▶ Quanto **maior for o grau de normalização** de uma BD, tanto melhor resulta a sua estrutura (**mais eficiente, menos redundante e maiores possibilidades de evolução**).
- ▶ O processo de normalização é **reversível**.
 - ▶ Podemos avançar até à 5ª e voltar à 1ª.
- ▶ A normalização, da 1ª à 5ª formas normais, baseia-se nas operações de **projectção** (criar novas tabelas relacionais) e **junção**.
- ▶ A normalização tem **regras**, mas é um processo baseado em pensamentos simples e muito **intuitivos**.

▶ 8

Questão

- ▶ Porque é que na base de dados exemplo se usaram três tabelas, S, P e SP?
- ▶ Porque **intuitivamente** nos pareceu que esta organização era a mais correcta. No fundo fez-se uma **normalização**.

▶ 9

Dependência Funcional

- ▶ Dada uma relação R, o atributo Y de R é **funcionalmente dependente** do atributo X de R- simbolicamente

$$R.X \rightarrow R.Y$$

(Y é funcionalmente dependente de X ou R.X funcionalmente determina R.Y)

se e só se cada valor de X em R tem associado precisamente um valor Y em R único (em qualquer instante).

Os atributos X e Y podem ser compostos.

▶ 10

Exemplo

- os atributos **SNAME**, **STATUS** e **CITY** da relação **S** são cada um **funcionalmente dependentes** do atributo **S#** da relação **S**, porque dado um valor particular de **S.S#** existe precisamente um valor correspondente para cada **S.SNAME**, **S.STATUS** e **S.CITY**.

$$S.S\# \rightarrow S.SNAME$$

$$S.S\# \rightarrow S.STATUS$$

$$S.S\# \rightarrow S.CITY$$

ou

$$S.S\# \rightarrow S.(SNAME, STATUS, CITY)$$

- Um contra-exemplo é:

P.COLOR não determina funcionalmente **P.WEIGHT**

(Na relação **P** não acontece uma côr um peso)

▶ 11

- Se o atributo **X** é uma **chave candidata** da relação **R** - em particular se é a **chave primária** - então **todos** os atributos **Y** de **R** devem ser **funcionalmente dependentes** de **X** (este facto segue da definição de chave candidata).
- Contudo **não existe** nenhum requisito na definição de dependência funcional (FD), de **X** ser de facto uma chave candidata de **R**,
- não existe nenhum requisito de que um dado valor **X** apareça somente num tuplo de **R**.

▶ 12

Definição Alternativa de DF

- ▶ Numa relação R, o atributo Y de R é **funcionalmente dependente** do atributo X de R, se e só se, sempre que dois tuplos de R concordam no valor de X devem também concordar no valor de Y.
- ▶ SP' satisfaz a FD

$$SP'.S\# \rightarrow SP'.CITY$$

(todo o tuplo com um dado valor **S#** deve ter o mesmo valor **CITY**)

mas o atributo **S#** não é chave candidata para a relação SP'.

▶ 13

Exemplo

- ▶ SP' satisfaz a FD

$$SP'.S\# \rightarrow SP'.CITY$$

(todo o tuplo com um dado valor **S#** deve ter o mesmo valor **CITY**)

mas o atributo **S#** não é chave candidata para a relação SP'.

▶ 14

FD Completa

- ▶ O atributo Y de R diz-se **totalmente dependente funcional** do atributo X da relação R se for **dependente funcional** de X e **não dependente funcional** de qualquer subconjunto próprio de X
- ▶ isto é não existe qualquer subconjunto próprio Z de atributos constituindo X tal que Y é **funcionalmente dependente** de Z.

Exemplo:

- ▶ Na relação SP' é certamente verdade que o atributo CITY é **funcionalmente dependente** do atributo composto (S#, P#):

$$SP'.(S\#, P\#) \rightarrow SP'.CITY$$

Contudo isto não é uma dependência funcional completa porque também temos a dependência funcional:

$$SP'.S\# \rightarrow SP'.CITY$$

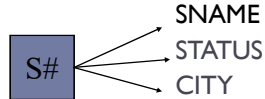
▶ 15

- ▶ Se Y é **dependente funcional** de X mas não **completamente** então X deve ser **composto**.
- ▶ Sendo assim normalmente “**dependência funcional**” significa “**dependência funcional completa**” a não ser que se explicita o contrário.

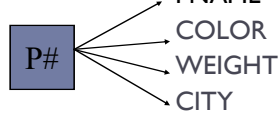
▶ 16

Diagrama de FD

S (FORNECEDORES)



P (PEÇAS)



SP (FORNECIMENTOS)



- ▶ Todas as setas partem das **chaves candidatas** da relação
 - ▶ na realidade a chave primária
- ▶ Por definição haverá sempre setas que partem da **chave candidata**
 - ▶ para um valor da chave candidata existe sempre um valor de tudo o resto
- ▶ Essas setas nunca podem ser eliminadas.
- ▶ Se existirem outras setas então é que surgem as dificuldades.

▶ 17

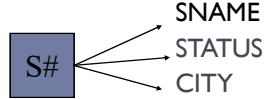
Observações

- ▶ Dependência Funcional (FD) é uma noção **semântica**.
- ▶ O reconhecimento das **FDs** é parte do processo de entendimento daquilo que os dados significam.
- ▶ Por exemplo:
 - ▶ O facto de **CITY** ser funcionalmente dependente de **S#**, significa que cada fornecedor está localizado em precisamente uma cidade.
 - ▶ Existe uma **restrição do mundo real** que a base de dados representa, nomeadamente que cada fornecedor está localizado numa só cidade.
- ▶ Uma vez que é parte da **semântica** da situação, essa **restrição** deve de alguma maneira ser observada na base de dados.
- ▶ A maneira disso ser assegurado é especificá-lo no **schema** de modo que o DBMS possa reforçá-lo.
- ▶ A maneira de especificá-lo no esquema é declarar a **FD**.

▶ 18

Diagrama de FD

S (FORNECEDORES)



P (PEÇAS)



SP (FORNECIMENTOS)



► O processo de **normalização** pode ser caracterizado, muito informalmente, como um processo para eliminar setas que não sejam setas que partem de chaves candidatas.

► Os conceitos de **normalização** conduzem a um meio muito simples de declarar as **dependências funcionais (FD)** (não necessariamente todas as FD, mas provavelmente as mais importantes na prática).

Dependências Funcionais

► As **dependências funcionais** representam relacionamentos de muitos para um.

► $S.S\# \rightarrow S.CITY$

pode ser lida como “Muitos fornecedores estão localizados na mesma cidade” (mas um fornecedor está localizado somente numa cidade).

1NF, 2NF e 3NF

- ▶ Vamos descrever as **três formas normais** de Codd.
- ▶ Primeiro vamos dar uma definição informal da **3NF** de modo a dar uma ideia do objectivo a alcançar.
- ▶ Repare-se que a **1NF**, **2NF** e **3NF** não são significantes em si a não ser como passos para a **BCNF** e seguintes.

▶ 21

3^aFN

Uma relação está na **3NF** se e só se os atributos que não são chave (se os houver) são:

- ▶ Dependentes **completamente** da chave primária.
- ▶ **Independentes** mutuamente
 - ▶ Dois ou mais atributos são **independentes mutuamente** se nenhum deles é dependente funcional de qualquer combinação dos outros.
 - ▶ Tal independência implica que cada um deles possa ser atualizado independentemente dos restantes.

Um atributo não chave é qualquer atributo que não participa na chave primária ou candidata da relação.

NOTA: Por uma questão de simplicidade supomos que cada relação tem exatamente uma **chave candidata**

O caso de uma relação com duas ou mais chaves candidatas é discutida mais à frente.

▶ 22

Exemplo

A relação **P (Peças)** está na **3NF** de acordo com a definição seguinte:

- ▶ Os atributos **PNAME**, **COLOR**, **WEIGHT** E **CITY** são certamente **independentes uns dos outros** (por exemplo, é possível mudar a cor de uma peça sem ter de mudar simultaneamente o seu peso) e claro são todos **totalmente dependentes** de **P#**, a chave primária (as dependências devem ser totais e são, porque a chave primária não é composta).

▶ 23

3^aFN

- ▶ Uma relação está na **3NF** se e só se em todos os instantes, cada tuplo consiste num valor de **chave primária** que identifica alguma entidade, juntamente com um conjunto de zero ou mais valores de **atributos independentes mutuamente** que descrevem essa entidade de alguma maneira.

▶ 24

Exemplo

- ▶ A relação **P (Peças)** está na **3NF** de acordo com a definição anterior
 - ▶ Cada tuplo de **P** consiste numa chave primária **P#**, valor que identifica alguma peça do mundo real, juntamente com **quatro valores** adicionais **PNAME**, **COLOR**, **WEIGHT** E **CITY** cada um deles servindo para descrever essa peça e **cada um deles independente dos outros**.

▶ 25

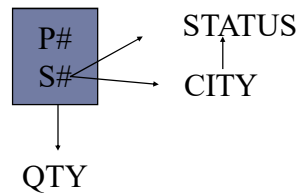
1ªFN

- ▶ Uma relação está na **primeira forma normal (1ªFN)** se e só se todos os domínios simples subjacentes contém somente **valores atômicos**, isto é todos os **atributos** são de **tipo simples** e **não** houver **nulos**.
- ▶ Uma relação que esteja somente na **1ªFN** tem uma **estrutura indesejável** por várias razões.

▶ 26

1ªFN para a 2ªFN Forma Normal

- Suponhamos que a informação sobre os fornecedores e fornecimentos em vez de estar dividida em duas relações **FORNECEDORES** e **FORNECIMENTOS** está agrupada numa só relação **FIRST** e que temos uma **restrição adicional** do status do fornecedor ser determinado pela cidade do fornecedor.



FIRST (S#, STATUS, CIDADE, P#, QTY)

Os atributos têm o significado habitual e com a restrição adicional já mencionada. É ignorado o atributo SNAME para maior simplicidade.

A **chave primária** de **FIRST** é a combinação dos atributos (S#, P#).

► 27

1ªFN para a 2ªFN Forma Normal

- O diagrama FD é mais complexo do que um diagrama FD na 3 NF.
- Um diagrama FD na 3ª NF tem setas a saírem somente da chave primária, enquanto que um diagrama que não esteja na 3ª NF (como o de FIRST) tem setas que partem da chave primária juntamente com outras **setas adicionais**.
- Essas **setas adicionais** é que causam todo o problema.
- De facto a relação **FIRST** viola ambas as condições a) e b) da definição da 3NF:
 - Os atributos não chave não são todos independentes mutuamente porque STATUS depende de CITY (uma seta adicional)
 - e não são todas dependentes funcionais da chave primária porque STATUS e CITY são cada um dependentes de S# (mais duas setas adicionais).

► 28

Tabela Exemplo de FIRST

2018/2019 - Bases de Dados - Fernanda Brito Correia

| S# | STATUS | CITY | P# | QTY |
|----|--------|--------|----|-----|
| S1 | 20 | London | P1 | 300 |
| S1 | 20 | London | P2 | 200 |
| S1 | 20 | London | P3 | 400 |
| S1 | 20 | London | P4 | 200 |
| S1 | 20 | London | P5 | 100 |
| S1 | 20 | London | P6 | 100 |
| S2 | 10 | Paris | P1 | 300 |
| S2 | 10 | Paris | P2 | 400 |
| S3 | 10 | Paris | P2 | 200 |
| S4 | 20 | London | P2 | 200 |
| S4 | 20 | London | P4 | 300 |
| S4 | 20 | London | P5 | 400 |



► 29

1ªFN: Redundância


2018/2019 - Bases de Dados - Fernanda Brito Correia

- As **redundâncias** são óbvias:
 - Cada tuplo do fornecedor S1 mostra que a cidade é LONDON.
 - Cada tuplo da cidade de LONDON mostra o STATUS igual a 20.
- As **redundâncias** desta relação conduzem a uma variedade das chamadas **anomalias de atualização** – isto é dificuldades com as operações **INSERT**, **DELETE** e **UPDATE**.
- Vamos concentrar-nos na redundância da cidade do fornecedor causada pela FD
FIRST.S# → FIRST.CITY
Ocorrem problemas com as três operações de atualização básicas.

► 30

INSERT

- ▶ Não podemos inserir o facto de um dado fornecedor estar localizado numa dada cidade até esse fornecedor fornecer pelo menos uma peça.



| S# | STATUS | CITY | P# | QTY |
|-----------|-----------|----------------|-----------|-----------|
| S1 | 20 | London | P1 | 300 |
| ... | ... | ... | ... | ... |
| S4 | 20 | London | P2 | 200 |
| S4 | 20 | London | P4 | 300 |
| S5 | 40 | Coimbra | ?? | ?? |

- ▶ Enquanto S5 não fornecer uma peça não temos um valor de chave primária apropriado.


Devido à regra de integridade da entidade do modelo relacional, **nenhuma componente de uma chave primária pode ser NULL**; na relação **FIRST** a chave primária consiste na combinação (S#, P#).

▶ 31

DELETE

Ao apagar um tuplo de **FIRST** para um dado fornecedor que **só forneça uma peça**

- ▶ e.x. tuplo com S# = S3 e P# = P2



| S# | STATUS | CITY | P# | QTY |
|---------------|---------------|------------------|---------------|----------------|
| S1 | 20 | London | P1 | 300 |
| ... | ... | ... | ... | ... |
| S2 | 10 | Paris | P1 | 300 |
| S2 | 10 | Paris | P2 | 400 |
| S3 | 10 | Paris | P2 | 200 |
| S4 | 20 | London | P2 | 200 |
| S4 | 20 | London | P4 | 300 |
| S4 | 20 | London | P5 | 400 |

- ❑ Destrói-se não só o fornecimento ligando esse fornecedor a uma peça
- ❑ mas também a informação de que um fornecedor está localizado numa determinada cidade.

▶ 32

UPDATE

- ▶ O valor CITY para um dado fornecedor aparece em FIRST muitas vezes. Esta **redundância** causa **problemas de atualização**.

| S# | STATUS | CITY | P# | QTY |
|-----|--------|--------|-----|-----|
| S1 | 20 | London | P1 | 300 |
| S1 | 20 | London | P2 | 200 |
| S1 | 20 | London | P3 | 400 |
| S1 | 20 | London | P4 | 200 |
| S1 | 20 | London | P5 | 100 |
| S1 | 20 | London | P6 | 100 |
| S2 | 10 | Paris | P1 | 300 |
| ... | ... | ... | ... | ... |

- ▶ Se o fornecedor S1 mudar de LONDON para LISBOA
- ▶ É necessário encontrar todo o tuplo de **FIRST** ligando S1 a LONDON (e alterá-lo)
- ▶ Caso contrário cria-se um **resultado inconsistente**
 - ▶ (CITY num tuplo S1 é LONDON e noutro LISBOA).

▶ 33

Solução: 2ªFN

- ▶ Uma relação está na **2ª NF** quando estiver na 1ª NF e todos os atributos que não são chave são **completamente dependentes** da chave primária

OU

- ▶ Extraia para outra tabela os atributos que dependem apenas de parte da chave primária.

▶ 34

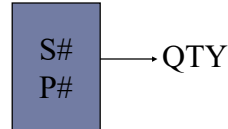
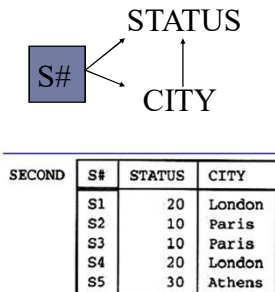
Solução: 2ªFN

- ▶ A **solução** para estes problemas consiste em substituir a relação **FIRST** pelas duas relações

SECOND(S#, STATUS, CITY)

SP(S#, P#, QTY)

S# é FK de SECOND



▶ 35

Solução: 2ªFN

- ▶ A informação para o **fornecedor S5** já foi incluída na relação **SECOND**.
- ▶ Esta **nova estrutura** ultrapassa todos os problemas relativamente às operações de **atualização** vistas atrás.
- ▶ O efeito desta revisão foi **eliminar as dependências não completas** e foi esta eliminação que resolveu estes problemas de atualização.
- ▶ Na relação **FIRST** o atributo **CITY** não descrevia a entidade identificada pela chave primária, nomeadamente um **fornecimento**. Em vez disso descrevia o **fornecedor** envolvido nesse fornecimento (de modo semelhante para STATUS). Misturar os dois tipos de informação causou problemas.

▶ 36

Observações

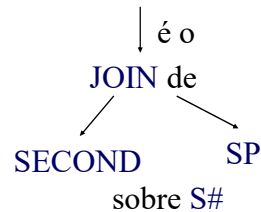
- ▶ A relação **FIRST** não está na 2NF.
- ▶ Uma relação na 1NF e não na 2NF pode ser sempre reduzida a uma coleção equivalente de relações na 2NF
- ▶ Repare que uma relação que está na 1NF e não está na 2NF tem uma chave primária composta.
- ▶ O processo de redução consiste em substituir a relação que está na 1NF por projeções adequadas.
 - ▶ A coleção de projeções obtidas é equivalente à relação original, no sentido de que a relação original pode ser sempre recuperada fazendo o JOIN (NATURAL) dessas projeções e sendo assim não há perda de informação (o que é muito importante!).

▶ 37

Observações

SECOND
SP }
}

São projeções de FIRST



(exceto se SECOND incluir tuplos que não existam em FIRST – p.e. S5)

- ▶ O processo de redução é um processo de tirar projeções, isto é o operador de decomposição é a projeção.
- ▶ Semelhantemente o operador de recomposição é o NATURAL JOIN.

▶ 38

2ªFN

- ▶ O 1º passo no processo de normalização é tomar projeções para eliminar as dependências funcionais não completas.
- ▶ Por outras palavras, dada uma relação R

R (A, B, C, D), Chave Primária (A, B)

R.A → R.D

- ▶ A normalização recomenda substituir R pelas projeções R1 e R2 :

R1 (A, D), Chave Primária (A)

R2 (A, B, C), Chave Primária (A, B)

Chave Estrangeira (A) referencia R1

A substituição de R por R1 e R2 – isto é a substituição de FIRST por SECOND e SP – são exemplos da chamada decomposição sem perda de informação.

▶ 39

2ªFN

- ▶ Uma vez que não é perdida informação nessa decomposição, qualquer informação que pode ser obtida a partir da estrutura original também pode ser obtida a partir da nova estrutura.
- ▶ O inverso contudo não é verdadeiro: a nova estrutura pode conter informação que não pode ser representada na estrutura original (tal como o facto do fornecedor S5 estar localizado em Athens).
- ▶ A nova estrutura que está na 2NF pode ser vista como uma representação mais credível do mundo real.

▶ 40

2ª para a 3ª Forma Normal

- ▶ Uma relação está na 2ª NF quando estiver na 1ª NF e todos os atributos que não são chave são completamente dependentes da chave primária

OU

- ▶ Extraia para outra tabela os atributos que dependem apenas de parte da chave primária.

▶ 41

2ª para a 3ª Forma Normal

As relações SECOND e SP estão ambas na 2ª NF.

- ▶ As chaves primárias são S# e (S#, P#) respetivamente.
- ▶ A estrutura SECOND e SP ainda causam problemas.
- ▶ A relação SP é satisfatória, de facto está na 3ª NF e vamos ignorá-la até ao fim.
- ▶ A relação SECOND ainda sofre da falta de independência mútua dos seus atributos que não são chave.
- ▶ O diagrama FD para SECOND é mais complexo que um diagrama da 3ª NF,
 - ▶ A dependência de STATUS sobre S# embora funcional e completa é transitiva. (via CITY).
 - ▶ Cada valor de S# determina um valor de CITY e cada valor de CITY por sua vez determina um valor de STATUS.

▶ 42

2ª para a 3ª Forma Normal

- ▶ Mais geralmente sempre que acontecem simultaneamente as FD

$$R.A \rightarrow R.B \text{ e } R.B \rightarrow R.C$$

Então é uma consequência lógica de que a **FD transitiva** acontece.

$$R.A \rightarrow R.C$$

- ▶ **Dependências transitivas** conduzem a anomalias de atualização.
- ▶ Vamos concentrar-nos na redundância CITY/STATUS, causada pela dependência funcional

$$\text{SECOND.CITY} \rightarrow \text{SECOND.STATUS}$$

▶ 43

2ª para a 3ª Forma Normal: Anomalias

INSERT

- ▶ Não podemos inserir o facto de uma cidade ter um dado status até termos um fornecedor localizado nessa cidade.

DELETE

- ▶ Se apagarmos o registo único onde aparece uma dada cidade perdemos a informação do STATUS dessa cidade bem como a informação desse fornecedor.

UPDATE

- ▶ O status de uma cidade aparece em **SECOND** em todos os registos onde aparecer essa cidade com o consequente problema de se quisermos atualizar o status ter de procurar todos os registos dessa cidade.

▶ 44

2ª para a 3ª Forma Normal

- ▶ Mais uma vez a solução para estes problemas é substituir a relação original (**SECOND**) por duas projeções :

- ▶ **SC** (S#, CITY)
- ▶ **CS** (CITY, STATUS)

- ▶ **SC** e **CS** estão ambas na 3NF. Os diagramas funcionais para estas relações são:

$S\# \rightarrow CITY$
 $CITY \rightarrow STATUS$

Repare, que assim já podemos incluir a informação de STATUS para uma cidade que ainda não tem fornecedores.

- ▶ Uma relação que está na 2NF e não está na 3NF pode sempre reduzir-se a uma coleção equivalente de relações na 3NF.

▶ 45

3ª Forma Normal

- ▶ Uma relação está na **3ª NF** se e só se está na 2NF e todo o atributo que não é chave é dependente da chave primária **não transitivamente**.

OU

- ▶ Extrair para outra tabela todos os atributos que dependam não da chave primária mas de outro atributo qualquer.

▶ 46

3ª Forma Normal

Criar projeções para eliminar dependências transitivas.

- ▶ Dada uma relação **R**

R (A, B, C)
Chave Primária (A)
 $R.B \rightarrow R.C$

- ▶ Recomenda a substituição de R pelas duas projeções R1 e R2:

R1 (B, C)
Chave Primária (B)
R (A, B)
Chave Primária (A)
Chave Estrangeira (B) referencia R1

▶ 47

Nível de normalização

- ▶ é uma questão de semântica e não simplesmente uma questão de valores dos dados que aparecem nessa relação num dado instante.
- ▶ Não é possível olhar para a tabela de uma relação num dado instante e dizer se está ou não na 3ªFN
 - ▶ também é necessário conhecer o significado dos dados, isto é, as dependências, antes de tal julgamento ser feito.
- ▶ O DBMS não pode assegurar que uma relação está na 3NF (ou em qualquer outra forma normal sem ser a 1ª) sem estar informado de todas as dependências relevantes.
- ▶ A 3FN é a que se utiliza para bases de dados mais ou menos simples

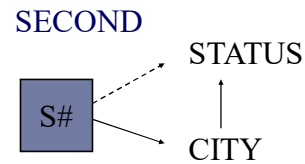
▶ 48

Boas e Más Decomposições

- ▶ Durante o processo de decomposição é frequente uma relação poder ser decomposta de várias maneiras diferentes. Tem de se ter o cuidado de escolher a boa decomposição.
- ▶ Consideremos a relação **SECOND** e as dependências funcionais:

SECOND.S# → SECOND.CITY
SECOND.CITY → SECOND.STATUS

E por transitividade
SECOND.S# → SECOND.STATUS



A dependência transitiva aparece como uma seta a tracejado.

▶ 49

Boas e Más Decomposições

- ▶ As anomalias de atualização encontradas em **SECOND** podiam ser ultrapassadas substituindo-a por duas projeções na 3ª NF

Decomposição A.

SC (S#, CITY)
CS (CITY, STATUS)

Decomposição B.

SC (S#, CITY)
SS (S#, STATUS)

A decomposição B também não perde informação e as 2 projeções estão na 3ª NF.

A decomposição B não é tão boa como a A.

Não é possível em B inserir a informação de que uma dada cidade tem um status a não ser que algum fornecedor esteja localizado nessa cidade.

▶ 50

Boas e Más Decomposições

- ▶ **Decomposição A** as 2 projeções são independentes uma da outra.
- ▶ **Decomposição B** as 2 projeções não são independentes uma da outra.
- ▶ Na decomposição B a FD

CITY → STATUS

Tornou-se uma **restrição inter-relacional**.

- ▶ Na **decomposição B** as atualizações nas projeções devem ser feitas de modo a assegurar a FD seguinte não seja violada

SECOND.CITY → SECOND.STATUS

O conceito de projeções independentes fornece um guia para a escolha de uma dada decomposição quando existe mais do que uma possibilidade.

▶ 51

Exemplo

- ▶ O Sr. Silva tem trabalhadores eventuais (“free-lancers”) que estão na cidade alojados em diversa residências e com diversas aptidões que queremos descrever e classificar. Cada residência tem um gerente. A informação a registar é então a seguinte:

Tabela trabalhadores

NOME IDADE RESIDÊNCIA GERENTE MORADA APTIDAO1 APTIDAO2
APTIDAO3 ...DESC1 DESC2... CLASS

- Um trabalhador só está numa residência que só tem 1 gerente
- Um trabalhador tem várias aptidões que quero descrever
- Quero classificar a aptidão para cada trabalhador

▶ 52

Está na 1ªFN?

NOME → IDADE, RESIDENCIA, GERENTE, MORADA

APTIDAO → DESC

NOME + APTIDAO → CLASS

RESIDENCIA → MORADA, GERENTE

- ▶ NÃO. O atributo APTIDAO pode ser 1 ou N e não é do tipo simples (quando muito seria um array, o que não é permitido nos sistemas relacionais e com a desvantagem de se ter um número limitado de aptidões.

Está na 1ªFN?

NOME → IDADE, RESIDENCIA, GERENTE, MORADA

APTIDAO → DESC

NOME + APTIDAO → CLASS

RESIDENCIA → MORADA, GERENTE

- ▶ NÃO. O atributo APTIDAO pode ser 1 ou N e não é do tipo simples (quando muito seria um array, o que não é permitido nos sistemas relacionais e com a desvantagem de se ter um número limitado de aptidões.

- ▶ Para ter valores atômicos teríamos de considerar a relação:

TRABALHADORES (NOME, IDADE, RESIDENCIA, GERENTE, MORADA, APTIDAO, DESC, CLASS)

Se considerarmos que queremos registar os trabalhadores dos quais ainda não sabemos qual é na sua aptidão, não está na 1NF, porque teríamos nulos.

Está na 1ªFN?

- ▶ Para estar na 1ªFN teríamos duas relações:
 - TRABALHADORES** (NOME, IDADE, RESIDENCIA, GERENTE, MORADA)
 - APTIDOES** (NOME, APTIDAO, DESC, CLASS)

NOME é FK de **TRABALHADORES**.
- ▶ Vantagens:
 - ▶ Cada trabalhador já pode ter um número quase ilimitado de aptidões e podemos ainda adicionar uma descrição e classificação (alta, média, baixa).

▶ 55

Está na 2ªFN?

- ▶ Qual o grau da aptidão e descrição de João como lenhador?

```
SELECT class, desc
FROM APTIDOES
WHERE nome='Joao' and aptidao='lenhador';
```

Resultado:

| CLASS | DESC |
|-------|------------------------------|
| boa | abate árvores, corta e serra |

▶ 56

Está na 2ªFN?

- ▶ Não. Na tabela **APTIDOES** vemos que a descrição da aptidão não depende do trabalhador mas só de **APTIDÃO** que é parte da chave. Por isso temos de dividir esta tabela em duas:

APTIDOES (APTIDAO, DESCRICAO)
APT_TRAB (NOME, APTIDAO, CLASS)
 APTIDAO é FK de **APTIDOES**
 NOME é FK DE **TRABALHADORES**

e também

TRABALHADORES (NOME, IDADE, RESIDENCIA, GERENTE, MORADA)

▶ 57

Está na 3ªFN?

- ▶ Na tabela **TRABALHADORES** o **GERENTE** e a **MORADA** dependem da **RESIDENCIA** que não é chave. Logo vamos colocar estes atributos noutra relação para alcançar a 3NF, obtendo:

RESIDENCIAS (RESIDENCIA, GERENTE, ENDERECO)
TRABALHADORES (NOME, IDADE, RESIDENCIA)
 RESIDENCIA é FK de **RESIDENCIAS**

e também

APTIDOES (APTIDAO, DESCRICAO)
APT_TRAB (NOME, APTIDAO, CLASS)
 APTIDAO é FK de **APTIDOES**
 NOME é FK DE **TRABALHADORES**

▶ 58

Boyce Codd Normal Form

- ▶ Uma relação está na BCNF se todos os determinantes são chaves candidatas.
- OU
- ▶ Todos os (atributos) determinantes são candidatos a chaves primárias ou são chaves primárias.
-
- ▶ Chave Candidata - é um atributo ou conjunto de atributos que poderão ser chave primária de uma tabela por terem um valor único.
 - ▶ Determinante - é um atributo do qual o outro é funcionalmente dependente completamente.

▶ 59

Boyce Codd Normal Form

- ▶ É uma evolução da 3NF pois esta não contempla todas as situações.
- ▶ A BCNF só interessa ser aplicada às tabelas que verificam:
 - ▶ Há múltiplas chaves candidatas.
 - ▶ Essas chaves são compostas.
 - ▶ As chaves sobrepõem-se, isto é têm pelo menos um atributo comum.
- ▶ Uma relação está na BCNF se nos diagramas funcionais as únicas setas partem de chaves candidatas. Não pode haver outras setas.

▶ 60

Exemplo

Considere a relação

FORNECEDORES (S#, SNAME, CITY, STATUS)

Admitindo que:

STATUS não depende de CITY.

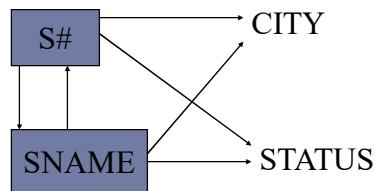
SNAME é único (não há dois fornecedores com o mesmo nome).

- ▶ A relação **FORNECEDORES** tem dois determinantes:
 - ▶ S# porque SNAME, CITY e STATUS dependem de S#.
 - ▶ SNAME porque S# , CITY e STATUS dependem de SNAME.
- ▶ Os únicos atributos que têm valores únicos são também S# e SNAME e portanto são duas chaves candidatas e não há outras

▶ 61

Exemplo

- ▶ Logo todos os determinantes são chaves candidatas. Está na BCNF.



- ▶ As únicas setas que há, partem das chaves candidatas.
- ▶ Nota:

Nesta relação não havia chaves candidatas sobrepostas, portanto não haveria necessidade de aplicar a BCNF.

▶ 62

Exemplo (BCNF)

- ▶ Considere a relação

SNP (S#, SNAME, P#, QTY),

com SNAME o nome único do fornecedor

- ▶ Determinantes:

- ▶ S# porque SNAME depende de S#.
- ▶ SNAME porque S# depende de SNAME.
- ▶ S# + P# -> QTY
- ▶ SNAME + P# -> QTY

- ▶ Chaves candidatas:

- ▶ Os únicos atributos que têm valores únicos são compostos:
 - ▶ (S#, P#)
 - ▶ (SNAME, P#)

▶ 63

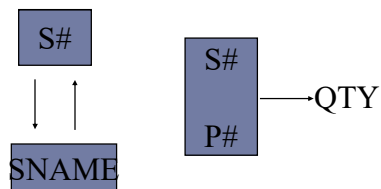
Exemplo (BCNF)

- ▶ Todos os determinantes não são chaves candidatas.
- ▶ Não está na BCNF.
- ▶ Decompondo fica: (Ambas na BCNF)

SN (S#, SNAME)
SP (S#, P#, QTY)
S# é FK de SN

OU

SN (SNAME, S#)
SP (SNAME, P#, QTY)
SNAME é FK de SN



▶ 64

Exemplo

A relação **Discip** contém dados sobre alunos (A) , disciplinas (D) e professores (P)

- ▶ **Restrição 1:** Cada professor só ensina uma disciplina
- ▶ **Restrição 2:** Para cada disciplina, cada aluno só tem um professor.

- ▶ **Restrição 1:**
Gomes só ensina Mat
Lopes só ensina Fisica
Alves só ensina Fisica

| A | D | P |
|------|--------|-------------|
| Luis | Mat. | Prof. Gomes |
| Luis | Fisica | Prof. Lopes |
| Ana | Mat | Prof. Gomes |
| Ana | Fisica | Prof. Lopes |

- ▶ **Restrição 2:**
Luis tem Fisica que é sempre ensinada pelo Prof. Lopes
(embora o Alves também ensine Fisica).

▶ 65

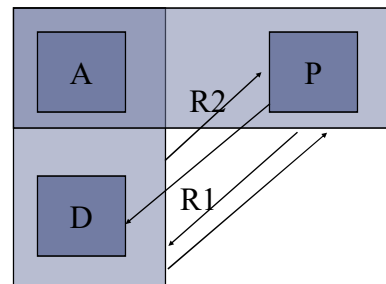
Dependências Funcionais

- ▶ Professor determina a Disciplina.
- ▶ Aluno + Disciplina determina Professor.
- ▶ Aluno + Professor também determina a disciplina, mas só Professor também determina a Disciplina.

Chaves Candidatas

Aluno + Disciplina porque o mesmo aluno tem várias disciplinas mas todas diferentes.

Aluno + Professor porque cada professor ensina apenas uma disciplina.



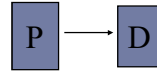
▶ 66

Está na BCNF?

- ▶ Não, porque nem todos os determinantes são chaves candidatas.
- ▶ Dividindo esta relação em duas:

Leciona (P, D)

Turma (A, P), P FK Leciona

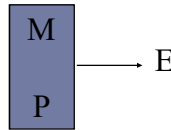
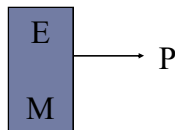


- ▶ **NOTA:** Esta **decomposição** cria outro **problema**.
 - ▶ Ex. se inserir o Prof. Alves (na tabela Turma) para o aluno Luis, este tuplo terá de ser rejeitado porque este aluno já tem Física com o Prof. Lopes.
 - ▶ Isto só pode ser validado acedendo à tabela Leciona. Isto acontece porque a relação original Discip é atômica, isto é não pode ser decomposta sem haver perda de informação.
- ▶ Portanto devemos parar a normalização na 3NF.

▶ 67

Exemplo

- ▶ A **relação EMP** regista que o estudante E fez exame na matéria M e obteve a posição P (1^a, 2^a, 3^a, ...).
- ▶ **Restrição:** Não há empates, ou seja para cada matéria cada aluno tem uma posição diferente.



| E | M | P |
|--------|---------|---|
| Joao | Matem | 1 |
| Joao | Fisica | 3 |
| Joao | Quimica | 2 |
| Manuel | Matem | 2 |
| Manuel | Fisica | 2 |
| Manuel | Quimica | 3 |
| Pedro | Matem | 3 |
| Pedro | Fisica | 1 |
| Pedro | Quimica | 1 |

▶ 68

Determinantes e Chaves Candidatas

- ▶ Determinante 1: $E + M \rightarrow P$
- ▶ Determinante 2: $M + P \rightarrow E$
- ▶ Chave candidata 1: $E + M$
- ▶ Chave candidata 2: $M + P$
- ▶ Os determinantes são todas chaves candidatas, logo está na **BCNF** (apesar de haver chaves sobrepostas no atributo M).