# 3 – Introdução

- Ao contrário do que muitos profissionais pensam, para projetar um banco de dados não basta apenas utilizar o bom senso e a experiência do programador, pois em projetos mais complexos isso pode não ser suficiente.

- Existem uma série de boas práticas que podem ser empregadas para garantir que o modelo de banco de dados seja consistente e sem nenhum tipo de redundância.

- Uma forma de se projetar banco de dados consistentes é por meio da normalização. O procedimento de normalização consiste em aplicar uma série de boas práticas, que se forem corretamente aplicadas, garantem uma melhor modelagem dos dados.

# 3 – Introdução

- Com o objetivo de facilitar o entendimento das formas normais, alguns conceitos básicos devem ser sabidos, são eles:

1. **Chave Candidata**: Atributo ou grupo de atributos que têm a propriedade de identificar unicamente um registro, podendo vir a ser uma chave Primária. A chave candidata que não é chave primária também se chama chave alternativa. Por exemplo, digamos que em uma tabela temos os campos: código, nome e CPF. Tanto código quanto CPF são chaves candidatas, pois permitem identificar o registro. Se o projetista escolher o código como chave primária, o CPF será a chave alternativa (e vice-versa).

2. **Chave Primária**: É uma chave candidata, escolhida pelo projetista do banco de dados como significado principal para a identificação de registros dentro de uma tabela.

3. **Superchave**: É qualquer conjunto de atributos contendo uma chave, seja ela primária ou candidata.

4. **Dependência Funcional:** Uma dependência funcional é um relacionamento entre dois ou mais atributos de forma que o valor de um atributo identifique o valor para cada um dos outros atributos, ou seja, um atributo está relacionado a outro.

Por exemplo, existe uma dependência funcional entre os atributos Código do Cliente e Nome do Cliente.

Nesse exemplo, para descobrir o nome do cliente (dentro de um conjunto de clientes), primeiramente é preciso saber qual é o código dele. Assim, o campo/atributo nome é dependente do campo/atributo código.

Entretanto, não é possível obter o código a partir do nome do cliente, pois vários clientes podem ter o mesmo nome. Por isso que nome do cliente é dependente funcional do atributo código.

5. **Dependência Funcional Parcial:** Ocorre quando os atributos não chave depende funcionalmente de apenas parte da chave primária. Assim, nas tabelas onde a chave primária for composta, todos os atributos devem depender de toda a chave primária. Caso a dependência seja de parte da chave, verificamos a existência de dependência funcional parcial.

No exemplo abaixo, a chave primária é composta por “id_conta” e “id_agencia”. Analisando o atributo “nome_agencia”, é possível observar que é dependente funcional somente do atributo “id_agencia”, logo ela está em Dependência Funcional Parcial.

| **id_conta** | **id_agencia** | **nome_agencia** | **id_cliente** | **saldo** |
|---|---|---|---|---|
| 1213 | 2 | Centro | 4 | 300 |
| 1214 | 3 | Zona 7 | 8 | 200 |
| 1215 | 5 | UTFPR | 7 | 100 |

- 6. **Dependência Funcional Transitiva:** Ocorre quando um campo é dependente funcional de outro campo que não é chave da tabela. No exemplo abaixo, o campo “nome_cliente” depende funcionalmente do campo “id_cliente”, que não é chave da tabela.

| **id_conta** | **id_agencia** | **id_cliente** | **nome_cliente** | **saldo** |
|---|---|---|---|---|
| 1213 | 2 | 4 | Miguel Borja | 300 |
| 1214 | 3 | 8 | Roger Guedes | 200 |
| 1215 | 5 | 7 | Dudu | 100 |

7. **Atributos Multivalorados:** São atributos que podem conter mais de um valor para um mesmo registro. No exemplo abaixo é apresentado uma tabela de Cliente, onde o campo telefone é multivalorado.

| **id_cliente** | **nome_cliente** | **telefone** |
|---|---|---|
| 1213 | Miguel Borja | (44)999577675<br>(44)987876565<br>(44)986753242 |
| 1214 | Roger Guedes | (44)999577612 |
| 1215 | Dudu | (44)999572321<br>(44)987876509<br>(44)986753287 |

# 3.1 – Diretrizes Gerais de Projeto

**1. Garantir que a semântica dos atributos seja clara no esquema**

- A semântica de uma relação consiste em inserir em uma determinada tabela apenas informações que realmente fazem parte do contexto.

- Por exemplo, a semântica de uma tabela de funcionários pode ser formada por campos como nome, cpf, endereço, etc. Como é possível observar, todos os atributos se referem diretamente a um funcionário.

- Entretanto, inserir dados de objetos diferentes em uma mesma entidade não é uma boa prática. Por exemplo, não é correto inserir em uma tabela de funcionários dados como nome do departamento, endereço do departamento, número de funcionários. Neste caso, uma mesma tabela armazena dados distintos.

# 3.1 – Diretrizes Gerais de Projeto

**2. Valores NULL nas tuplas**

- Embora os sistemas gerenciadores de banco de dados atuais sejam capazes de comprimir de forma eficiente os campos que possuem valor **NULL**, deve-se evitar ao máximo que os atributos fique nulos (sem valor).

- Os campos com valor NULL além de consumirem espaço de armazenamento, geram problemas em funções de **agregação** (SUM, MAX, AVG, etc.). Junções internas também podem gerar resultados imprevisíveis caso os campos utilizados para a junção possuírem valores nulos.

- Desta forma, os campos que permitem valores NULL devem ser evitados ao máximo, nem que para isso seja necessário a criação de outra tabela.

- Por exemplo, para armazenar diversos telefones para um cliente, é melhor criar um tabela "**telefone_cliente**" do que criar 3 ou 4 atributos "**telefone**" na tabela de cliente, sendo que no caso do

# 3.1 – Diretrizes Gerais de Projeto

**3. Geração de tuplas falsas**

- Devem ser usados para relacionamento somente atributos que são chaves primárias ou estrangeiras, para evitar que tuplas falsas sejam geradas.

- É uma boa prática deixar a chave estrangeira de uma tabela com o mesmo nome e tipo de sua respectiva chave primária, facilitando futuras junções.

# 3.2 – Normalização

- O **processo de normalização**, proposto por Codd em 1972, auxilia o desenvolvedor a criar um modelo de banco de dados relacional mais eficiente e sem redundâncias. O modelo deve ser passado por uma série de testes para certificar se ele satisfaz determinada **forma normal**.

- Inicialmente, Codd propôs três formas normais, que ele chamou inicialmente de **primeira, segunda e terceira forma normal**. Atualmente existem autores que propuseram outras formas normais.

- Todas estas formas normais estão baseadas em um único conceito, que são as dependências funcionais entre os atributos de uma relação.

# 3.2 – Normalização

- A normalização dos dados pose ser considerada um processo de analisar os esquemas de relação de dados com base em suas **dependências funcionais (DF)** e chaves primárias, com o objetivo de **minimizar redundância e anomalias de inserção, exclusão e atualização.**

- Quanto mais alto o grau de normalização que um modelo de banco de dados chegou, menor a probabilidade do banco apresentar inconsistências.

- Outro ponto que merece ser observado é que um projetista de banco de dados não precisa necessariamente normalizar até a forma normal mais alta possível.

  Muitas vezes, em nome do desempenho, pode ser uma boa estratégia não normalizar até determinado nível, esta técnica se chama **desnormalização**.

- Os projetistas mais experientes já criam o banco de dados normalizado, não necessitando efetuar o processo de normalização.

# 3.2.1 – Primeira Forma Normal (1FN)

- Para que uma tabela esteja na primeira forma normal, todos os seus atributos precisam ser **monovalorados** e **atômicos**.

- Analisando a figura abaixo, na tabela cliente, o campo “telefone” está multivalorado. E o campo “endereco” não é atômico, pois possui várias informações (rua, número e bairro). Por isso não está na 1FN:

| **id_cliente** | **nome_cliente** | **telefone** | **endereco** |
|---|---|---|---|
| 1213 | Miguel Borja | (44)999577675<br>(44)987876565<br>(44)986753242 | Rua Central, 343.<br>Bairro Santana |
| 1214 | Roger Guedes | (44)999577612 | Rua Matrix, 12.<br>Bairro Central |
| 1215 | Dudu | (44)999572321<br>(44)987876509<br>(44)986753287 | Rua Manoel, 45.<br>Bairro Industrial |

# 3.2.1 – Primeira Forma Normal (1FN)

- Para deixar a tabela na 1FN é preciso criar uma tabela para armazenamento de telefone, e riar novos campos para deixar o “endereco” atômico, como mostra a figura abaixo:

| id_cliente | nome_cliente | endereco | numero | bairro |
|---|---|---|---|---|
| 1 | Miguel Borja | Rua Central | 343 | Bairro Santana |
| 2 | Roger Guedes | Rua Matrix | 12 | Bairro Central |
| 3 | Dudu | Rua Manoel | 45 | Bairro Industrial |

| id_telefone | id_cliente | telefone |
|---|---|---|
| 1213 | 1 | (44)999577675 |
| 1214 | 1 | (44)999577612 |
| 1215 | 2 | (44)999572323 |

# 3.2.2 – Segunda Forma Normal (2FN)

- A segunda forma normal é baseada no conceito de dependência funcional parcial, ou seja, todos os atributos não-chave devem depender funcionalmente de toda a chave primária (e não de parte dela). Além disso, o modelo já deve estar na 1FN.

- A figura abaixo mostra uma tabela que está na 1FN, mas não está na 2FN, porque o atributo "nome_agencia" não depende de toda a chave primária, mas somente de parte dela (apenas id_agencia).

| **id_conta** | **id_agencia** | **nome_agencia** | **id_cliente** | **saldo** |
|---|---|---|---|---|
| 1213 | 2 | Centro | 4 | 300 |
| 1214 | 3 | Zona 7 | 8 | 200 |
| 1215 | 5 | UTFPR | 7 | 100 |

# 3.2.2 – Segunda Forma Normal (2FN)

- Para colocar na 2FN, novas relações precisam ser feitas, como mostra a figura:

| id_conta | id_agencia | id_cliente | saldo |
|---|---|---|---|
| 1213 | 2 | 4 | 300 |
| 1214 | 3 | 8 | 200 |
| 1215 | 5 | 7 | 100 |

| id_agencia | nome_agencia |
|---|---|
| 2 | Centro |
| 3 | Zona 7 |
| 5 | UTFPR |

# 3.2.3 – Terceira Forma Normal (3FN)

- A terceira forma normal é baseada no conceito de dependência transitiva. Um dependência transitiva ocorre quando um atributo (não principal) possui dependência funcional de um atributo que não é chave primária.

- Para obter a 3FN, o modelo já deve estar na 2FN e não pode existir nenhuma dependência funcional transitiva. A figura abaixo não está na 3FN, pois “nome_cliente” é dependente funcional de “id_cliente”, que não é chave da tabela.

| **id_conta** | **id_agencia** | **id_cliente** | **nome_cliente** | **saldo** |
|---|---|---|---|---|
| 1213 | 2 | 4 | Miguel Borja | 300 |
| 1214 | 3 | 8 | Roger Guedes | 200 |
| 1215 | 5 | 7 | Dudu | 100 |

# 3.2.3 – Terceira Forma Normal (3FN)

- Para que a 3FN seja obtida é preciso eliminar a dependência funcional transitiva, como mostra a figura abaixo:

| id_conta | id_agencia | id_cliente | saldo |
|---|---|---|---|
| 1213 | 2 | 4 | 300 |
| 1214 | 3 | 8 | 200 |
| 1215 | 5 | 7 | 100 |

| id_cliente | nome_cliente |
|---|---|
| 4 | Miguel Borja |
| 8 | Roger Guedes |
| 7 | Dudu |