**Disciplina:** Banco de dados I

**Professor:** Rafael Variani

**O modelo ACID**

O modelo ACID é um dos mais velhos e importantes conceitos da teoria de Banco de Dados. Ele define quatro pontos que todos os SGBD devem ter: atomicidade, consistência, isolamento e durabilidade. Qualquer BD que não atinja algum desses pontos pode ser considerado não confiável. Vamos analisar cada um desses elementos com um pouco mais de detalhe.

**Atomicidade**: Estado em que as modificações no BD devem ser todas ou nenhuma feita. Cada transação é dita como “atômica”. Se uma parte desta transação falhar, toda transação falhará. É muito importante que o SGBD mantenha a natureza atômica das transações, diante de qualquer problema, seja do próprio SGBD, sistema operacional ou de hardware. Qualquer ação que constitui falha na unidade de trabalho, a transação deve ser desfeita (rollback). Quando todas as ações são efetuadas com sucesso, a transação pode ser efetivada (commit).

**Consistência**: Estado que garante que todos os dados serão escritos no BD. Se por alguma razão, uma transação que viole a consistência do BD, toda a transação será dada um “roll back”, desfazendo o que foi feito até aquele ponto, restaurando a consistência do BD. Por outro lado, se uma transação for executada com sucesso, atingirá um novo ponto de consistência.

**Isolamento**: Requer que múltiplas transações que estejam ocorrendo “ao mesmo tempo”, não interfiram nas outras. Por exemplo, se João realizasse uma transação ao mesmo tempo que Maria, mesmo que diferente, ambas transações devem operar de maneira isolada. O BD deve executar ambas as transações por completo e em fila, prevenindo que a transação do João leia informações parciais produzidas pela transação da Maria. Note que o isolamento em si, não garante quem será realizado primeiro, mas garante que nenhum irá interferir no outro.

**Durabilidade** Garante que toda transação cometida (commit) pelo BD não será perdida. A durabilidade é garantida através de backups e logs de transações que facilitam a restauração das transações cometidas, frente aos problemas de hardware ou software que possam ocorrer.

**Normalização de dados**

A normalização de dados é uma série de passos que se seguem no projeto de um [banco de dados](https://pt.wikipedia.org/wiki/Banco_de_dados), que permitem um armazenamento consistente e um eficiente acesso aos dados em bancos de dados relacionais. Esses passos reduzem a redundância de dados e as chances dos dados se tornarem inconsistentes.

No entanto, muitos [SGBDs](https://pt.wikipedia.org/wiki/SGBD) relacionais não têm separação suficiente entre o projeto lógico da base de dados e a implementação física do banco de dados, e isso tem como consequência que as consultas feitas a um banco de dados totalmente normalizado tenham mau desempenho. Nestes casos, usa-se por vezes a [desnormalização](https://pt.wikipedia.org/wiki/Desnormalização) para melhorar o desempenho, com o custo de menores garantias de consistência.

Diz-se que uma tabela num banco de dados relacional está numa certa forma normal se satisfaz certas condições. Basicamente possuímos três dessas formas, mas existem hoje outras formas normais geralmente aceitas. Cada forma normal listada abaixo representa uma condição mais forte que a precede na lista. Para a maioria dos efeitos práticos, considera-se que as bases de dados estão normalizadas se aderirem à terceira forma normal.

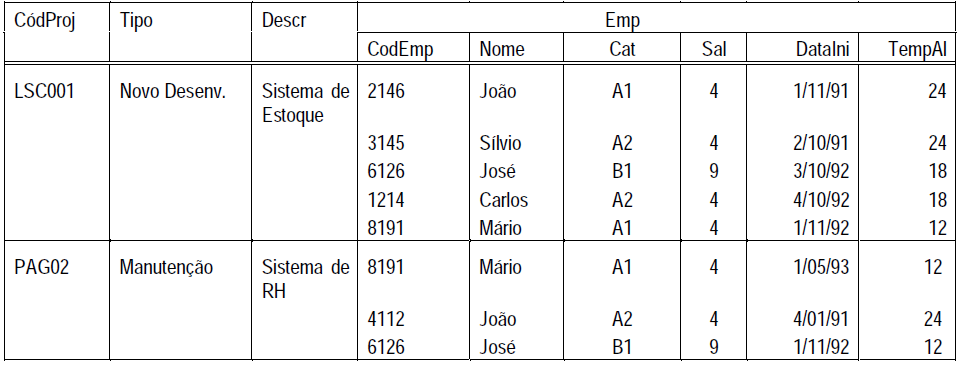
**1 – Primeira Forma Normal(ou 1FN)**

Requer que todos os valores de colunas em uma tabela sejam **atômicos** (exemplo: um número é um átomo, enquanto uma lista ou um conjunto não o são). A normalização para a primeira forma normal elimina grupos repetidos, pondo-os cada um em uma tabela separada, conectando-os com uma chave primária ou estrangeira;

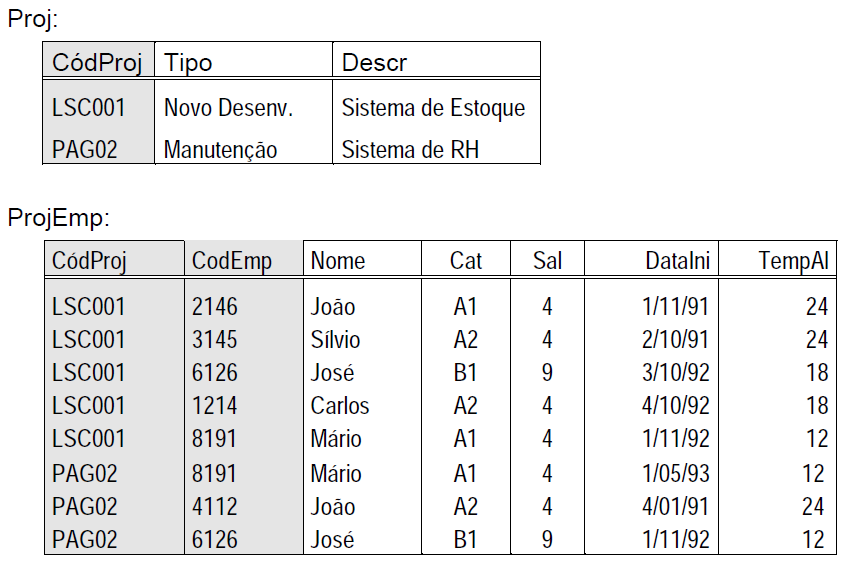
*Não pode conter atributos multivalorados.* **-> Multivalorado. telefone1 + telefone 2 em um único campo.**

*Todos os atributos devem possuir valores atômicos --> apenas um valor*

**Exemplo: Não normalizado**

****

**Normalizando para primeira forma normal**

****

**2 - Segunda Forma Normal** (ou **2FN**)

Requer que não haja \***Dependência funcional** não-trivial de um atributo que não seja a chave, em parte da chave candidata;

Tabelas com chaves primárias compostas devem sempre ter os demais atributos dependentes de toda essa **chave primária, e não de apenas parte dela.**

Um atributo pertence apenas aquela tabela, ou seja, atributo possuir dependência da chave primária.

Não pode haver informações repetidas para representar a mesmas coisas.

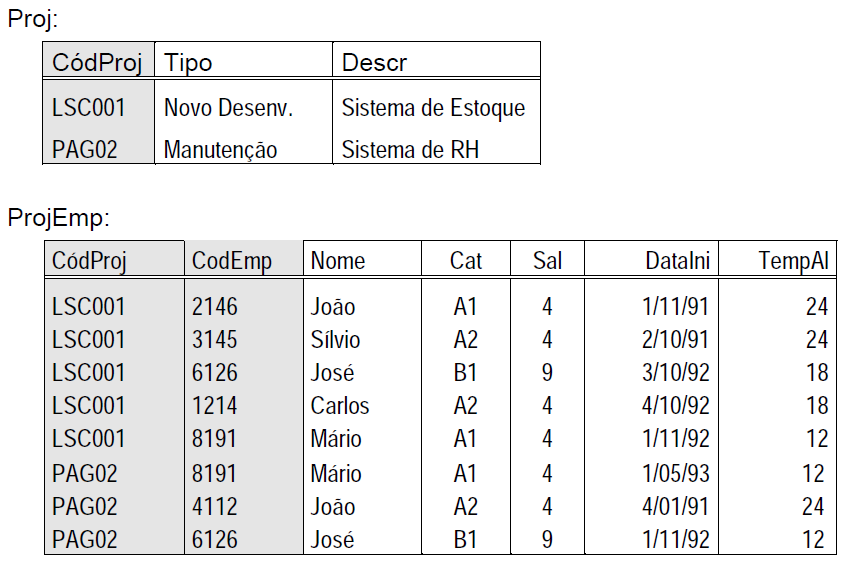
Exemplo: Aluno 1 telefone 1.

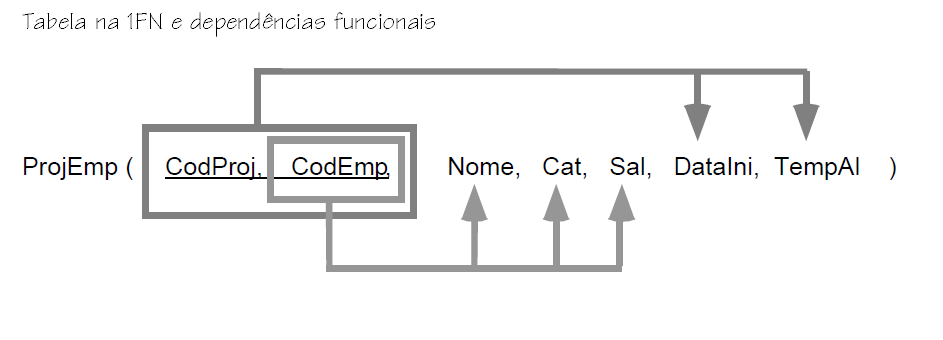
Aluno 1 telefone 2.

Claro que deve estar na primeira formula normal.

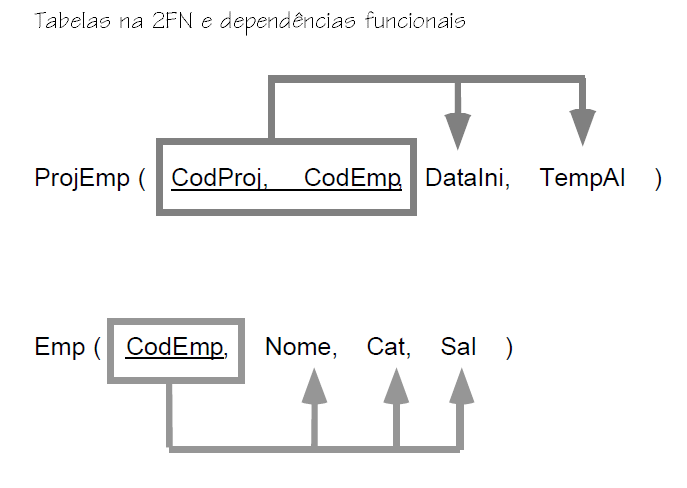
Objetivo não sobre carregar a tabela de atributos.

**Exemplo: Não atende a 2F**

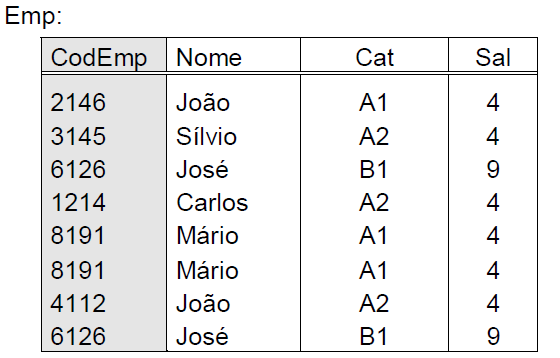
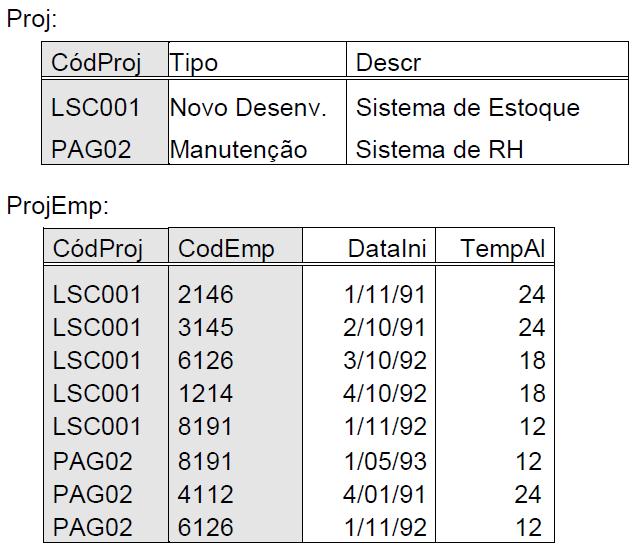


***Entendo um pouco melhor…..***

**Passando para 2F..**



***Resultado.***



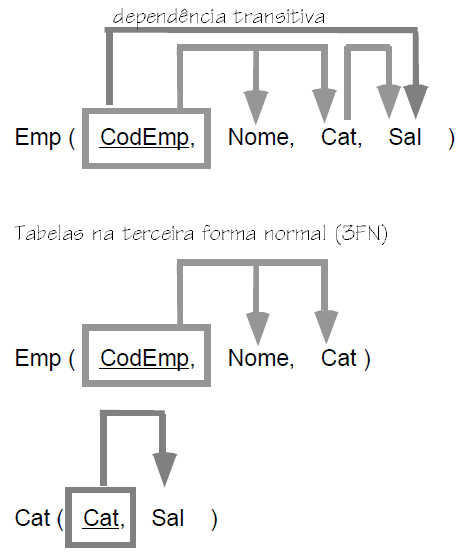
**3 - Terceira Forma Normal** (ou **3FN**).

Requer não haver dependências funcionais não-triviais de atributos que não sejam chave, em qualquer coisa exceto um superconjunto de uma chave candidata;

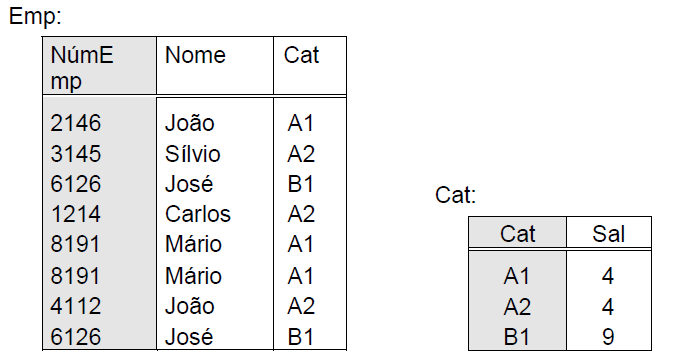
Os campos que não tem depedência com a chave primária não devem existir.

Uma tabela encontra-se na 3FN quando, além de estar na 2FN, toda coluna não chave depende diretamente de chave primária, isto é, quando não há dependências funcionais transitivas ou indiretas.

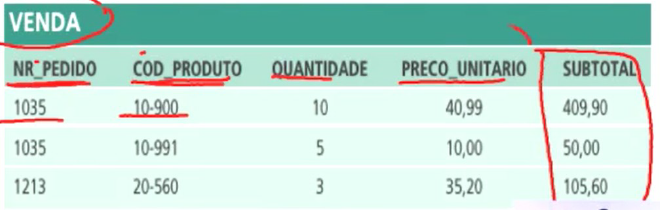
Passando para terceira forma normal



**Resultado Final**



**Outro exemplo…..**



### **Descrição de normalização**

A normalização é o processo de organizar os dados em um banco de dados. Isso inclui a criação de tabelas e estabelecer relacionamentos entre essas tabelas de acordo com as regras criadas para proteger os dados e para tornar o banco de dados mais flexíveis, eliminando a redundância e dependência inconsistente.   
 Dados redundantes desperdiçam espaço em disco e criam problemas de manutenção. Se os dados que existem em mais de um local devem ser alterados, os dados devem ser alterados da mesma maneira em todos os locais. Uma alteração de endereço de cliente é muito mais fácil de implementar se esses dados são armazenados somente na tabela clientes e lugar no banco de dados.   
  
 O que é uma "dependência inconsistente"? Embora seja intuitivo para um usuário examinar a tabela de clientes para o endereço de um cliente específico, pode não fazer sentido a procurar lá para o salário do funcionário que chama no cliente. Salário do funcionário está relacionado, ou dependentes, o funcionário e, portanto, deve ser movido para a tabela Funcionários. Dependências inconsistentes podem dificultar dados para acesso porque o caminho para localizar os dados pode estar ausente ou danificado.

Existem algumas regras de normalização de banco de dados. Cada regra é chamada de um "forma normal". Se a primeira regra é observada, o banco de dados é considerado na "primeira forma normalizada". Se as primeiras três regras forem observadas, o banco de dados é considerado na "terceira forma normalizada". Embora outros níveis de normalização são possíveis, a terceira forma normalizada é considerada o mais alto nível necessário para a maioria dos aplicativos.   
 Assim como acontece com muitas regras formais e especificações, cenários do mundo real não permitir sempre de conformidade perfeita. Geralmente, normalização requer tabelas adicionais e alguns usuários acham isso complicado. Se você decidir violar uma das primeiras três regras da normalização, certifique-se de que seu aplicativo prevê qualquer problema que possa ocorrer, como dados redundantes e dependências inconsistentes.

As descrições a seguir incluem exemplos.

### Primeira forma Normal

* Elimine grupos de repetição em tabelas individuais.
* Crie uma tabela separada para cada conjunto de dados relacionados.
* Identifique cada conjunto de dados relacionados a uma chave primária.
* Não use vários campos em uma única tabela para armazenar dados semelhantes.

### Segunda Forma Normal

### Crie tabelas separadas para conjuntos de valores que se aplicam a vários registros.

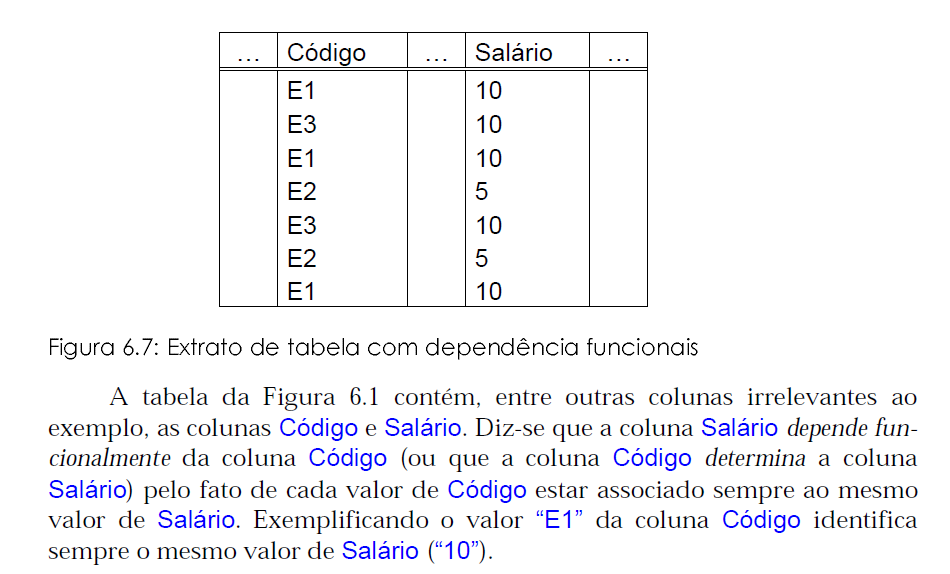
### Relacione essas tabelas com uma chave externa.

### Registros não deveriam depender de algo que não seja a chave primária da tabela (uma chave composta, se necessário).

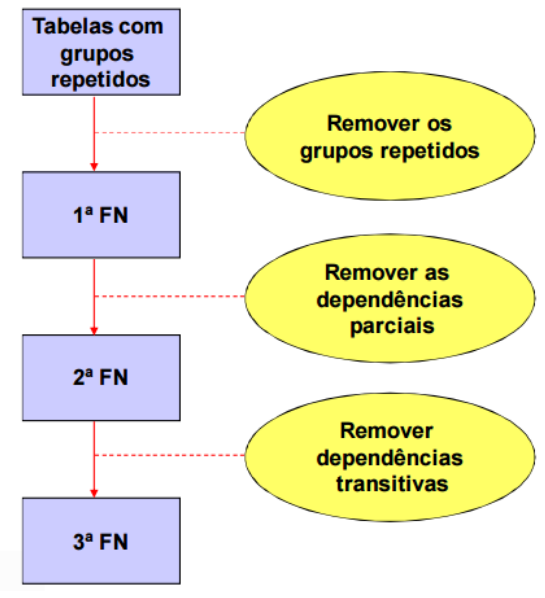
### Terceira Forma Normal

* Elimine campos que não dependem da chave.
* Valores em um registro que não são parte da chave do registro não fazem parte da tabela.   
    
  Por exemplo, em uma tabela de recrutamento de funcionário, nome da Universidade e o endereço do candidato podem ser incluídos.

\* **DEPENDÊNCIA FUNCIONAL**

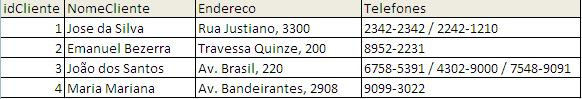


**ETAPAS DA NORMALIZAÇÃO**

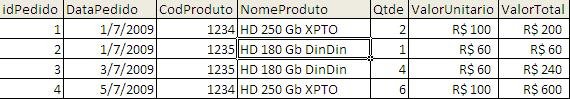


**ATIVIDADES**

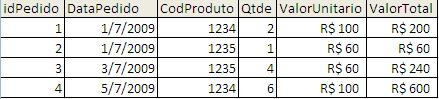
**Passar para primeira forma normal**



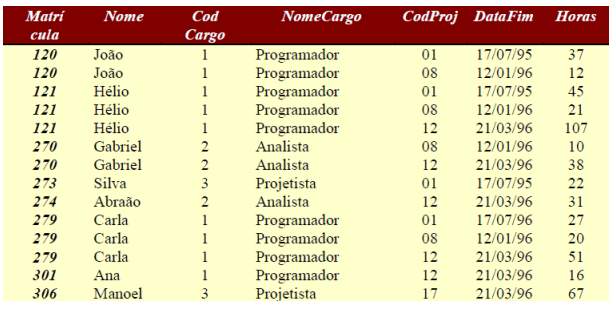
**Passar para segunda forma normal**



**Passar para terceira forma normal**



**Normalize a tabela a seguir:**

****

**Realizar uma pesquisa e citar problemas na normalização.**