Banco de Dados

Fundamentos e conceitos

Prof. Dr. Vladimir Costa Alencar

LANA - UEPB

https://www.valencar.com/

Atributos ou Campos (Variáveis)

É a menor unidade destinada ao armazenamento de valores em um arquivo ou tabela de um Banco de Dados.

Cada campo pode conter somente um tipo de dado (numérico, caractere, lógico, etc)

Exemplo:

Universidade Estadual da Paraíba

Rua Baraúnas, 351

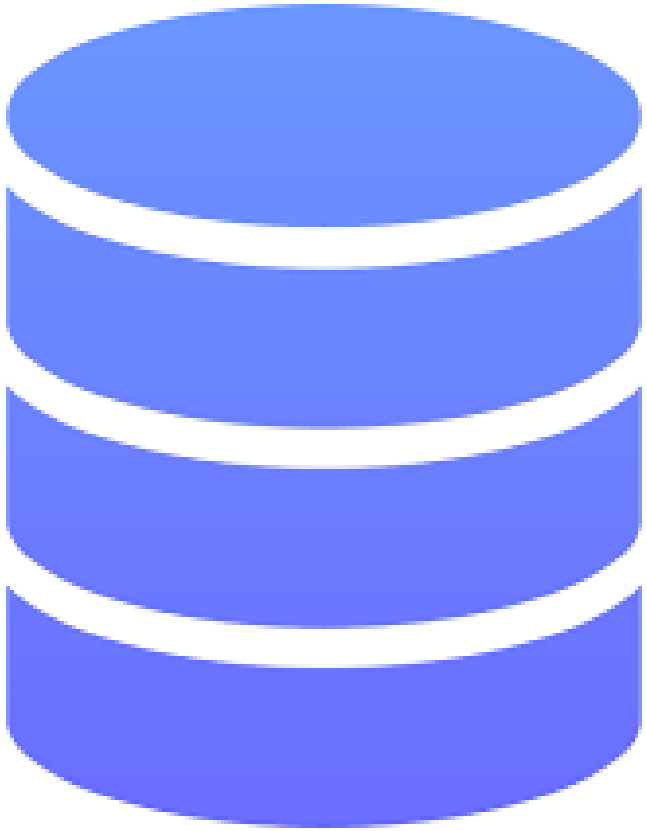
Bairro Universitário

Campina Grande

PB

CEP 58429-500

Fone: 83 3315.3300



Atributos ou Campos (Variáveis)

É a menor unidade destinada ao armazenamento de valores em um arquivo ou tabela de um Banco de Dados.

Cada campo pode conter somente um tipo de dado (numérico, caractere, lógico, etc)

Exemplo:

Universidade Estadual da Paraíba

Rua Baraúnas, 351

Bairro Universitário

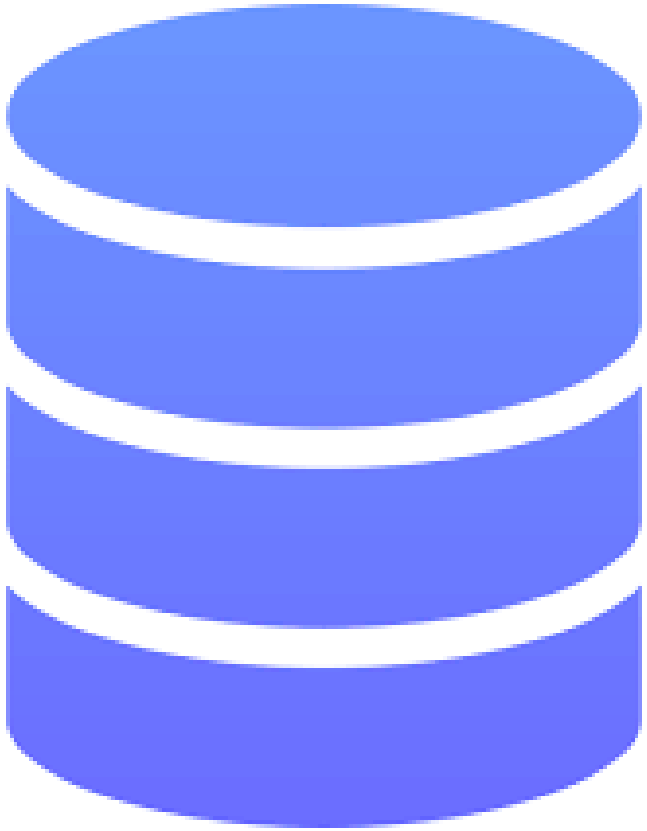
Campina Grande

PB

CEP 58429-500

Fone: 83 3315.3300

Atributos



Campos – Tipos Básicos

Caractere
Numérico
Data
Hora
Lógico - Booleano
Auto-Incremento
- Número decimal sequencial (registros)

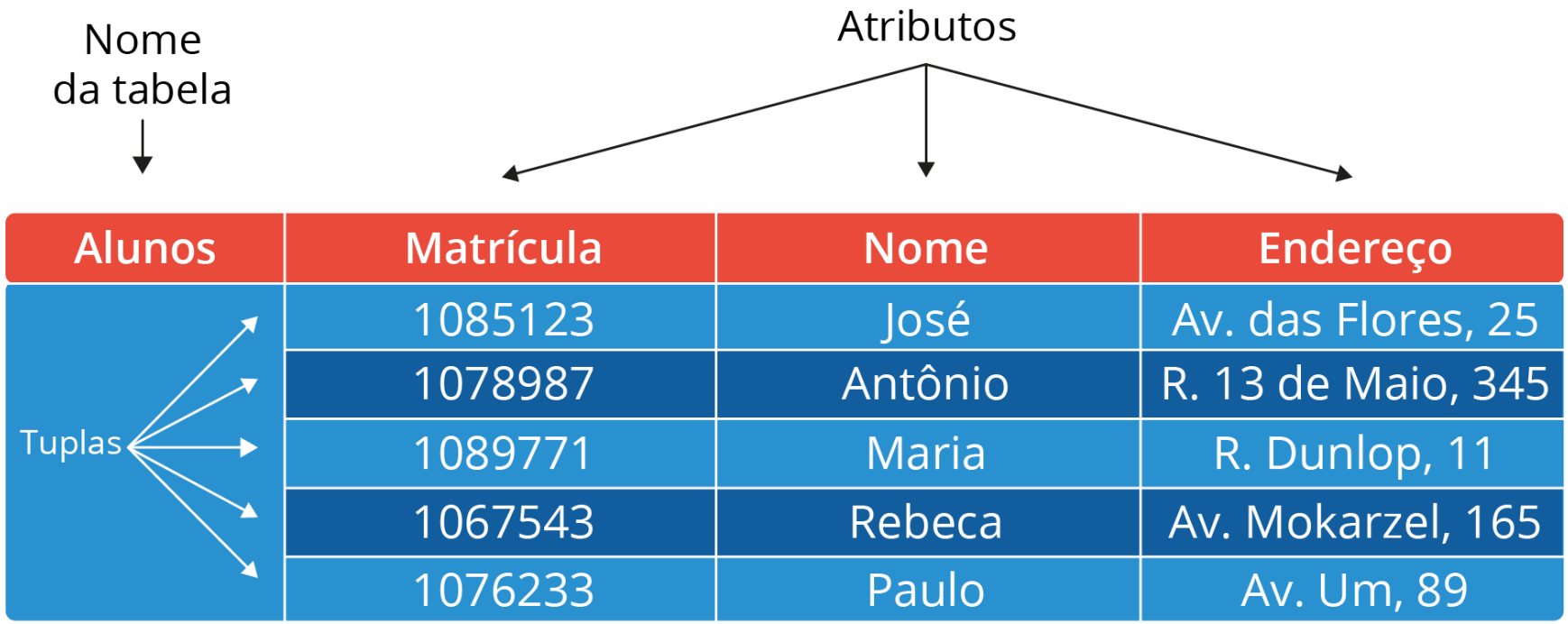
Tuplas - Registros - Linhas

É um conjunto de campos de uma tabela (com valores)

É a unidade básica de armazenamento e recuperação dos dados

Identifica a entrada de um único item de informação de uma tabela de banco de dados

Tuplas – Linhas ou Registros



Tabelas

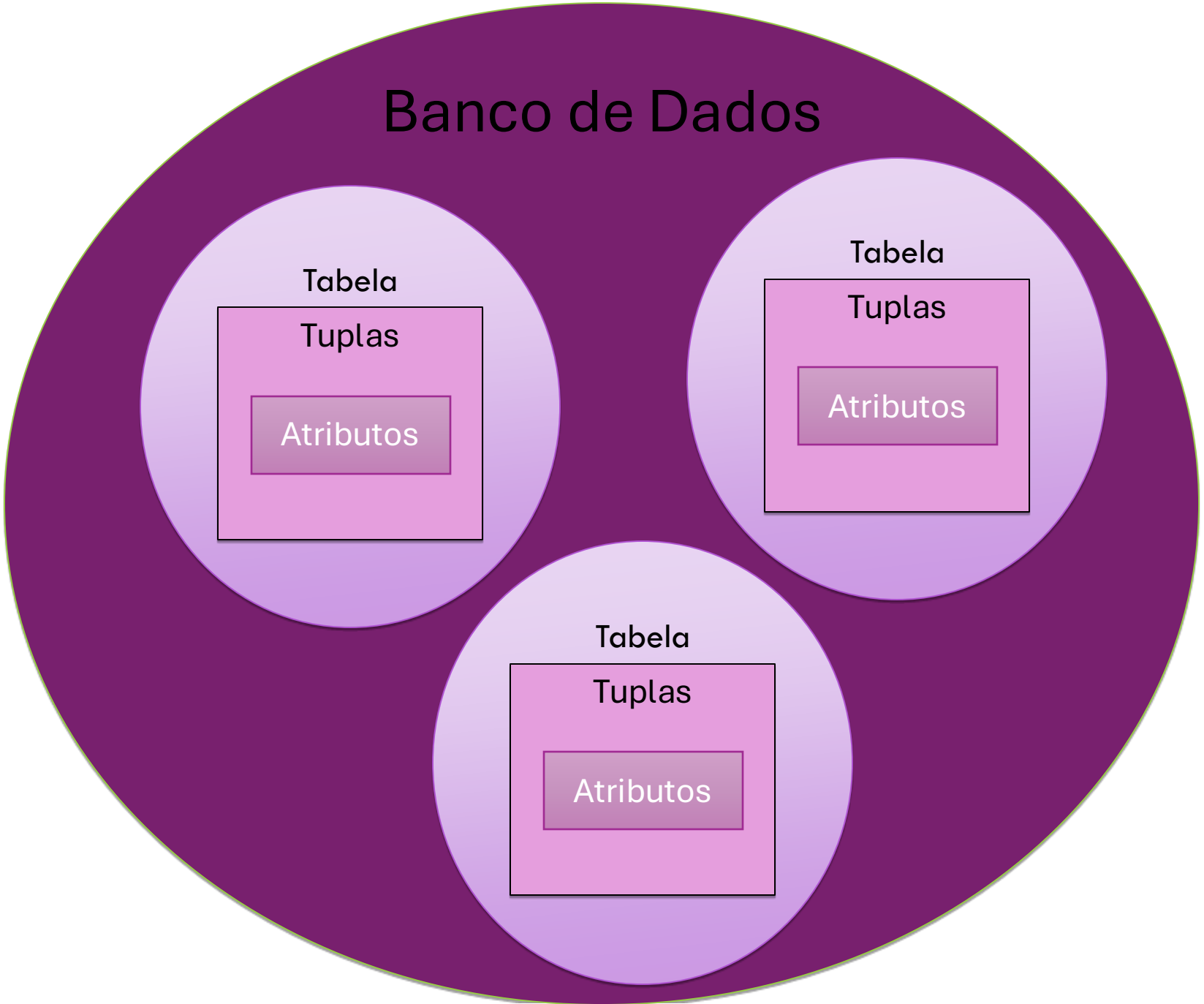
- É um conjunto de tuplas-registros (linhas)
- Um banco de dados pode ser formado por uma ou mais tabelas
- Cada tabela, tem seus campos específicios
- Ex. Tabela de alunos, funcionários, eventos, etc.

Tabela ALUNO:

id_aluno	nome	email	dtcadastro
1	Douglas	douglas@targettrust.com.br	*NULL*
2	José Carlos	josecarlos@targettrust.com.br	2013-01-21
3	Douglas	dg@targettrust.com.br	2013-01-21
4	Marta	marta@targettrust.com.br	2013-01-21
5	Simone	simone@targettrust.com.br	2013-01-21
6	Rafael	rafael@targettrust.com.br	2013-01-21
7	ROCK	rock@targettrust.com.br	2013-01-21

Tabela CURSO:

id_curso	nome_curso	preco	duracao
1	Java	10.00	20
2	Oracle	30.00	20
3	.NET	60.00	20
4	PHP	40.00	20


Banco de Dados
Tabela
Tuplas
Atributos
Tabela
Tuplas
Atributos
Tabela
Tuplas
Atributos

Tabelas

Cada tabela deve ser identificada por um nome único no banco de dados

São as tabelas que possuem a estrutura/composição dos registros

-> Nomes, Tipos de dados, Tamanho

As aplicações de banco de dados geralmente utilizam várias tabelas

Índices

Em qual Capítulo do livro de Banco de Dados explica a linguagem SQL?

Índices

O Índice do livro contém a informação de qual capítulo fala sobre SQL

Índices

Permitem que os registros sejam encontrados com extrema rapidez

Tipos:

Simples (um só campo)

- Ex. CPF, Matrícula

Composto (vários campos da tabela)

- Ex. RG + UF

Chaves Primárias

- É um atributo ÚNICO da tabela que identifica de forma única os seu registros
- Ex. Matrícula, RG+UF, CPF, Passaporte, Cartão de Crédito, PIX são UNICOS. podem ser chaves primárias

Chaves Primárias

- Tem por função aplicar uma ordenação automática aos registros

- O seu funcionamento é similar ao de um índice

- Não é possível um registro ter o mesmo valor que o Atributo que compõe o índice

Chaves Primárias

Critérios de escolha

Escolher campos cujo tamanho não seja muito grande

Rapidez de acesso

O valor deve ser estável (não ser modificado)

Os campos são de preenchimento obrigatório

Chaves Candidatas

São Atributos que poderiam ser usados como chaves primárias

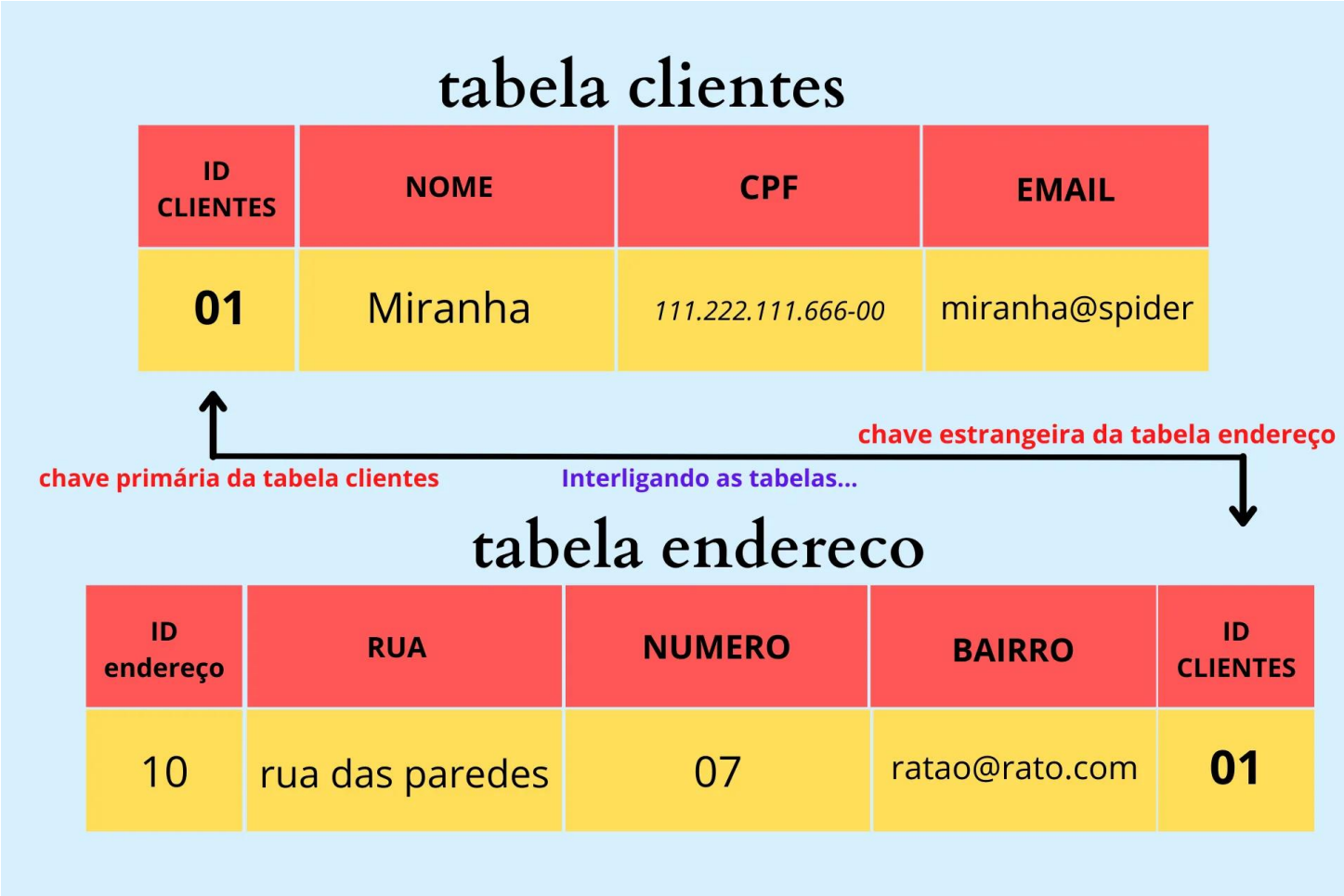
Considere a seguinte tabela `Aluno`:

Matricula	CPF	Nome	Email
123	111.222.333-44	João	joao@email.com
124	222.333.444-55	Maria	maria@email.com
125	333.444.555-66	Pedro	pedro@email.com

Os Atributos **Matricula**, **CPF** e **Email** são chaves candidatas

Chaves Estrangeiras

Permitem que os registros de uma tabela sejam relacionados com os de outra tabela pela sua chave primária



Integridade dos Bancos de Dados

INTEGRIDADE

→ Precisão, correção, validade

- Refere-se a evitar a perda acidental (não intencional) da consistência do BD

Os SGBDs possuem formas de restringir essa quebra de integridade

Integridade dos Bancos de Dados

A perda de integridade pode surgir de:

1. Quedas durante o processamento de uma Transação (Ti);

2. Anomalias motivadas pelo acesso concorrente ao BD;

3. Anomalias motivadas pela distribuição dos dados em vários computadores;

4. Erro lógico que viola a suposição de que as Tis devam preservar as restrições de integridade de BD

Perda de Integridade dos BD

Informação Incorreta;

Uma transação de venda ocorre mas o operador insere a data da transação de forma incorreta;

Um zero é esquecido de ser digitado ao se entrar um salário de um empregado;

Dados duplicados;

Perda de Integridade dos BD

Um novo departamento é criado, com codDepto = 200, e é inserido na tabela duas vezes;

Chaves estrangeiras inválidas;

Departamento codDepto = 300 é fechado, os empregados deste departamento recebem um novo codDepto e um empregado é esquecido de ser atualizado ficando com o extinto código 300.

Integridade dos Bancos de Dados

Para assegurar que os dados do BD estão válidos precisa-se formalizar verificações (**check**) e regras de negócio (**business rules**) que os dados devem aderir.

Esta forma de assegurar consistência é conhecida como restrições de integridade (**integrity constraints**).

Restrição de Integridade → Finalidade é fornecer um meio de assegurar que as mudanças feitas no BD, por usuários autorizados, não levam à perda de consistência dos dados.

Questão: Como garantir Integridade do BD?

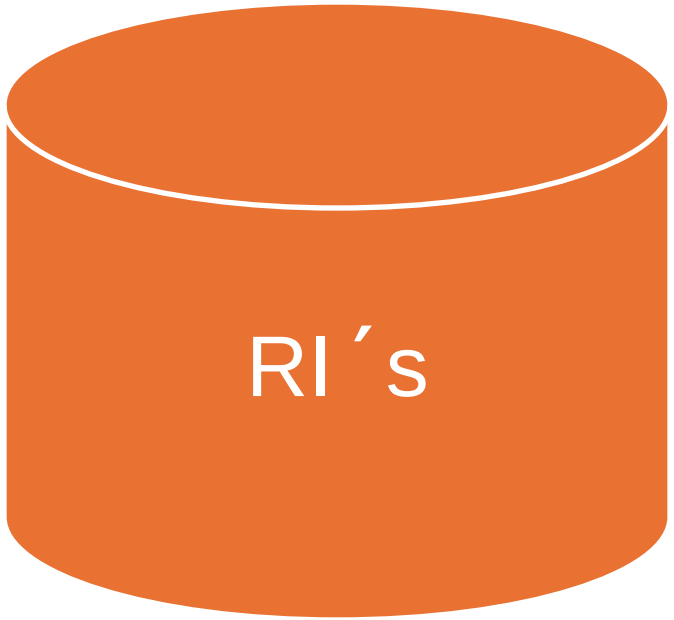
Integridade dos Bancos de Dados

- **Solução 1:** Inserir as restrições de integridade no código da aplicação

- Problemas:
  - **Sobrecarrega** o programador, pois ele precisa codificar todas as validações;
  - **Susceptível a erros** pode-se esquecer de fazer uma validação => BD inconsistente
  - **Difícil manutenção** Se RI mudar, então precisa mudar todas as aplicações que validam àquela RI

Integridade dos Bancos de Dados

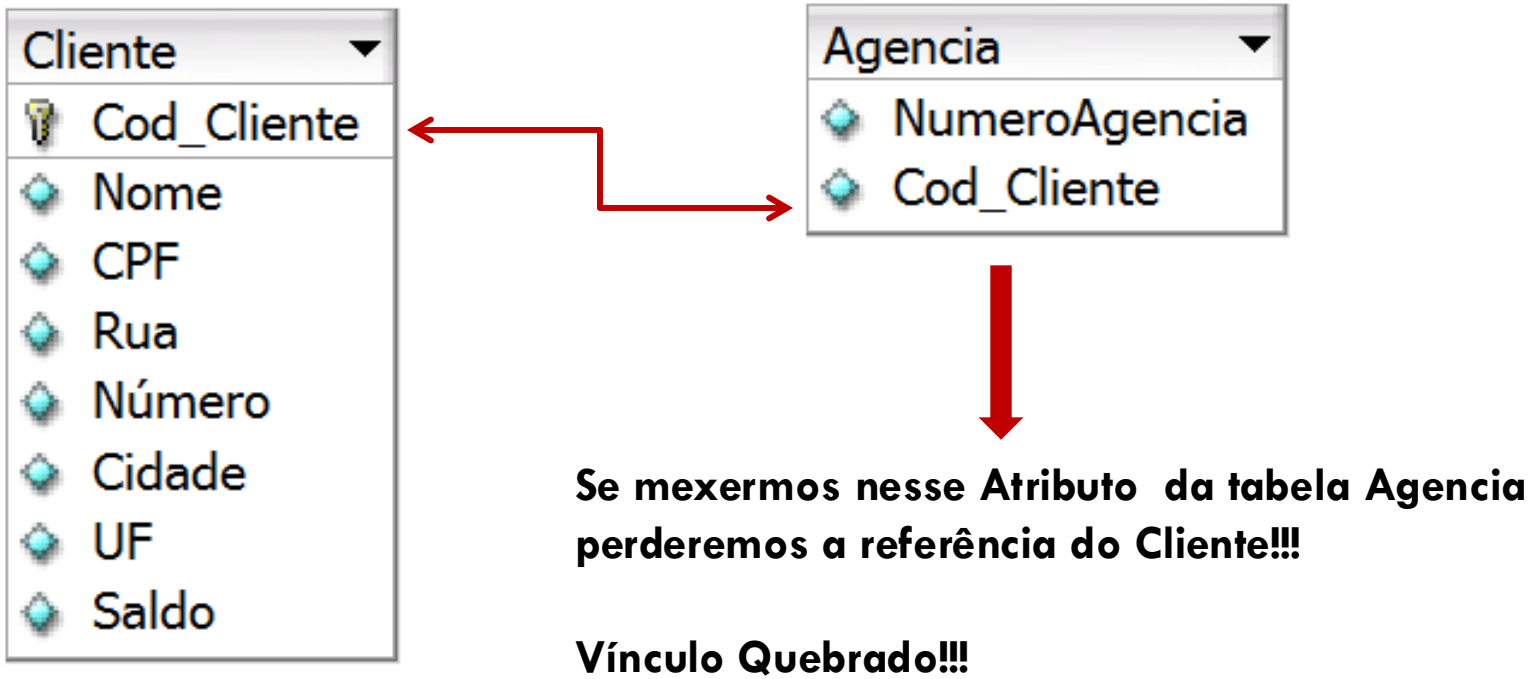
Solução 2: Inserir as RI no próprio BD, através do subsistema do SGBD.



Integridade dos Bancos de Dados

É a garantia que os dados armazenados não serão violados

Uma violação de integridade pode acarretar a perda de uma tabela.



Integridade dos Bancos de Dados

Table

Agencia

NumeroAgencia	Cod_Cliente
800	1
800	2
850	3
900	4
900	5

Se eliminarmos Cod_Cliente = 1 perderemos a referência do Cliente de Código número 1 !!!!

Vínculo Quebrado!!!

Table

Cliente

Database Connection

Agencia-Cliente

Cod_Cliente	Nome	CPF	Rua	Número	Cidade	UF	Saldo
1	Paula	22222222222	Av Chile	234	Campina Grande	PB	3000
2	João	56745454533	Rua 14	888	Patos	PB	1200
3	Maria	23412356709	Rua Floriano Peixoto	123	Campina Grande	PB	3900
4	Antonio	34567891234	Rua Y	902	João Pessoa	PB	3980
5	Marcos	29102034455	Rua H	321	João Pessoa	PB	4000

Integridade dos Bancos de Dados

Suponhamos uma implementação de um subsistema de integridade com as seguintes funções:

Monitorar transações;

Se violação → tomar ação apropriada

Para prover tais funções, o sistema contém **Regras de Integridade**, expressas numa linguagem de alto nível (Ex.: SQL DDL), que detecte uma tentativa de violação e tome a ação apropriada.

Regras de Integridade de BD

Exemplo:

```
R1: After Updating Conta.Saldo:
        IF Conta.Saldo > 0
                        ...
        Else
                Do;
                        Set return “regra R1 violada”
                        Reject
        End;
```

Regras de Integridade de BD

Importante:

As regras são compiladas e armazenadas no dicionário de dados;

Uma vez lançadas no sistema, as regras são utilizadas daquele estado do BD → definição deve ser rejeitada se for de encontro ao estado atual do BD

Regras de Integridade de BD

Vantagens:

- Validação é tratada pelo SGBD, ao invés de ser feita nas aplicações individuais;
- Melhor entendimento e manutenção devido à atualização das regras no catálogo;
- Deve existir um meio de atualizar regras com o sistema funcionando;
- Pode-se utilizar linguagem de consulta para as RI's

Exemplo: 'Quais as regras de integridade que se aplicam ao salário dos empregados?'

Integridade de Entidades

Define que as **chaves primárias** não podem ser nulas

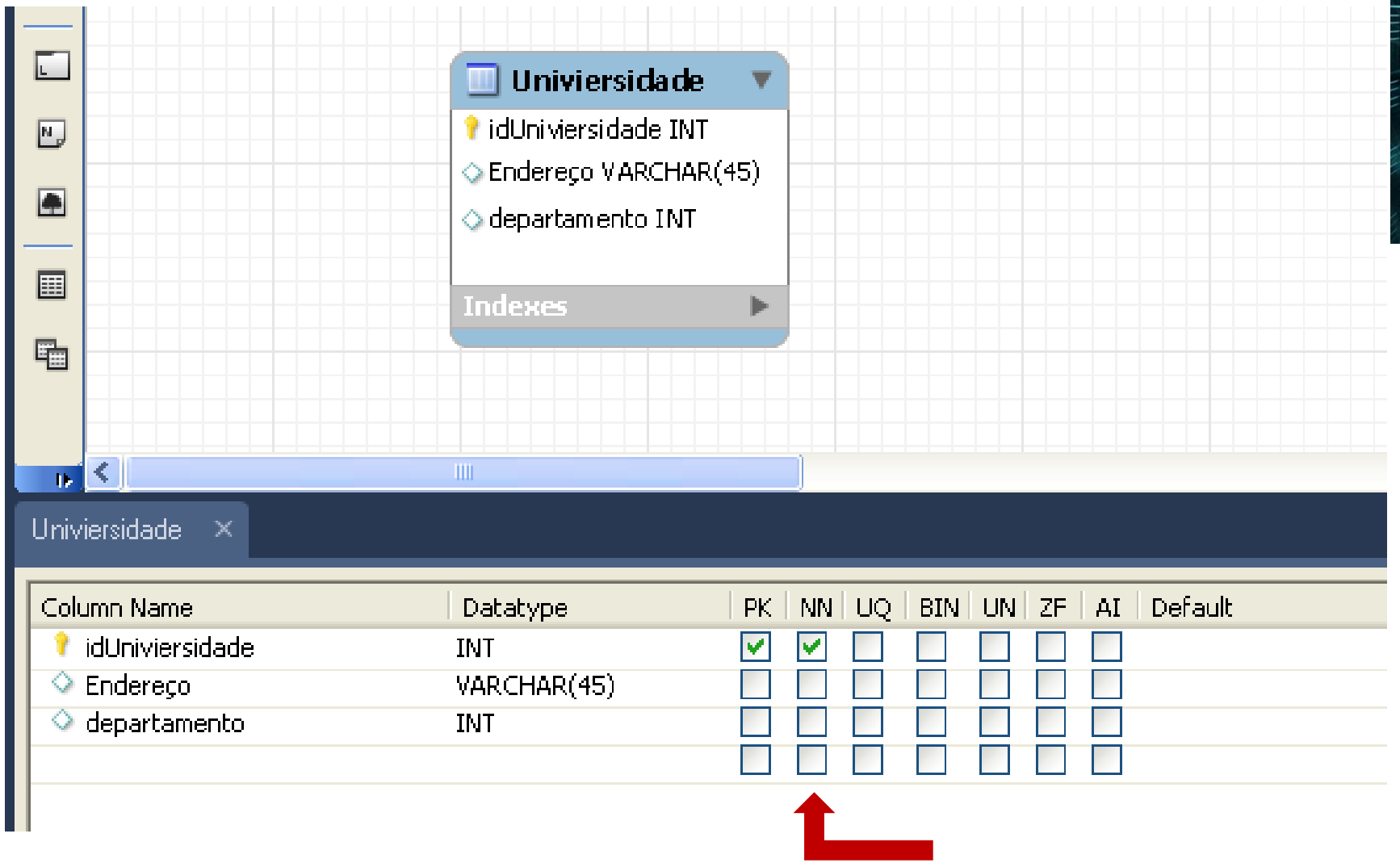
No caso de chaves primárias compostas, todos os campos devem ser não-nulos

Um valor nulo não é especificamente:

- O Valor Zero (0)
- O espaço em Branco

O valor nulo é um campo NÃO PREENCHIDO

Integridade de Entidades



Integridade de Campos

Algumas vezes há a necessidade de colocar restrições aos campos

Ex. Salário

Restrição: Entre R$ 500 e R$ 2.500

- O próprio SGBD executa a validação do campo

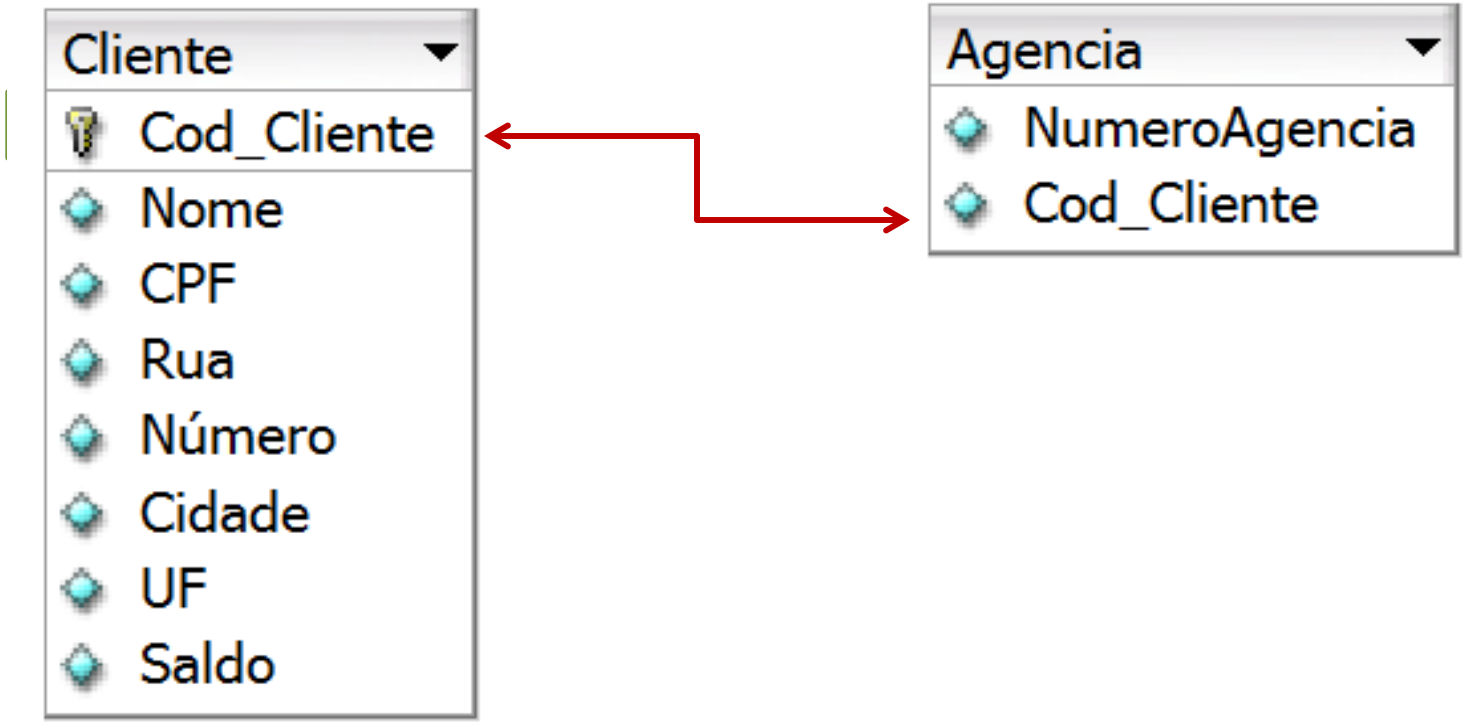
- Isso isola qualquer aplicativo de usar indevidamente o campo Salário

Integridade Referencial

Estabelece restrições ou bloqueios de algumas operações nos dados (chaves primárias)

Notadamente

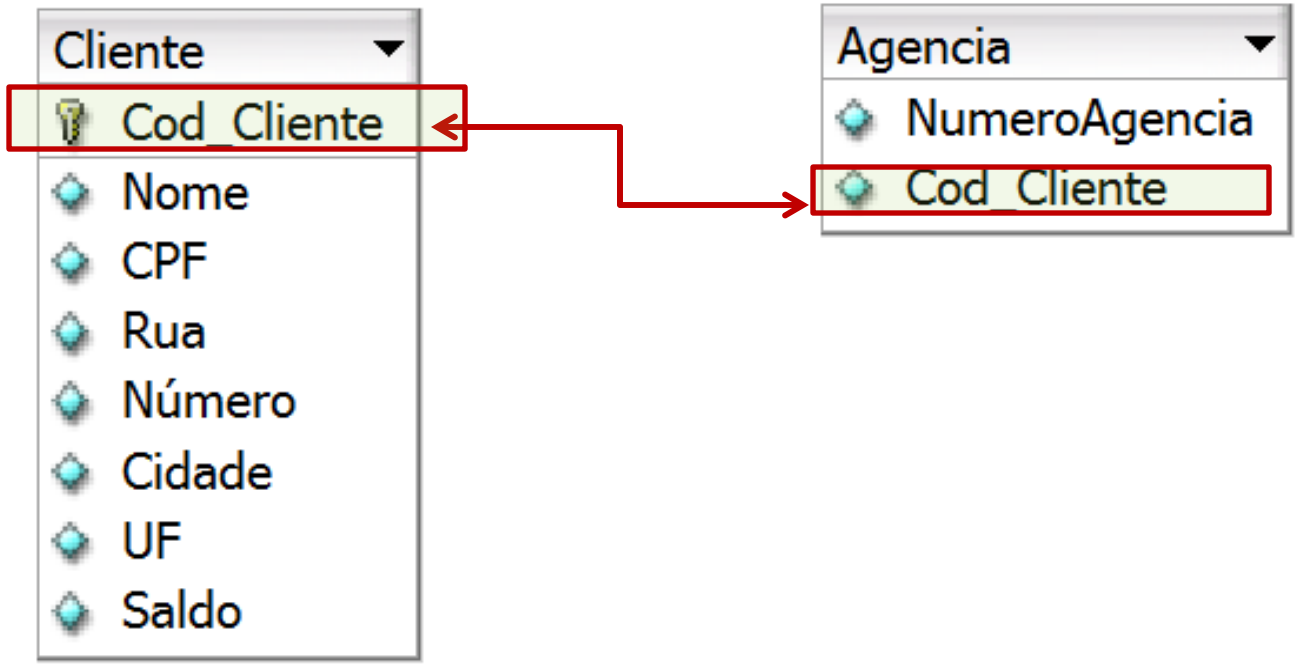
Alteração e Exclusão



Integridade Referencial

Esse tipo de inconsistência é proibido (excluir o campo agencia da tabela)

Assim é impedida que a relação entre a chave primária e a chave estrangeira seja quebrada.



Integridade de Campos/Atributos

Ainda Há regras que podem ser implementadas através de rotinas criadas pelo desenvolvedor

1. Stored Procedures
2. Triggers

Stored Procedures

Stored Procedures (ou Procedimentos Armazenados) em um banco de dados são blocos de código SQL que são armazenados e executados diretamente no servidor de banco de dados.

Elas permitem encapsular um conjunto de instruções SQL em uma unidade reutilizável, que pode ser chamada quando necessário.

As **Stored Procedures** ajudam a reduzir a repetição de código, simplificar operações complexas e melhorar a segurança e a performance, já que o código é pré-compilado e executado diretamente no banco de dados.

Integridade de Campos/Atributos

Stored Procedures

```
CRIAR PROCEDIMENTO mostrar_clientes_especiais()
INICIO
SELECIONE Num_conta, Agencia, Saldo
DA TABELA Agencia
CONDIÇÃO saldo > 2000;
FIM
```

```
mysql> call mostrar_clientes_especiais;
+------------+---------+-------+
| Num_conta  | Agencia | Saldo |
+------------+---------+-------+
|        200 |      50 |  2500 |
|        250 |      50 |  3000 |
|        300 |      50 |  5000 |
+------------+---------+-------+
3 rows in set (0.00 sec)
```

Stored Procedures

```
delimiter //
CREATE PROCEDURE GetPaises()
BEGIN
   SELECT * FROM mundo;
END; //
delimiter ;

CALL GetPaises();
```

Stored Procedures

```
delimiter //
CREATE PROCEDURE GetPaisesCond(in taxa_alfabetizacao float)
BEGIN
   SELECT * FROM mundo
   WHERE Alfabetizacao  > taxa_alfabetizacao;
END; //
delimiter ;

CALL GetPaisesCond(70.0);
```

Stored Procedures

```
delimiter //
CREATE PROCEDURE GetPaisesCond_out(in tx_alfabetizacao float, out total_paises int)
BEGIN
 SET total_paises = (SELECT count(*) FROM mundo WHERE Alfabetizacao > tx_alfabetizacao)
END; //
delimiter ;

CALL GetPaisesCond_out(70.0, @total_paises);
SELECT @total_paises;
```

Integridade de Campos/Atributos – Triggers

- Uma TRIGGER ou gatilho são mecanismos que permitem a execução automática de um bloco de código quando um evento específico ocorre em uma tabela (como **INSERT**, **UPDATE** ou **DELETE**).

- Elas são muito usadas para garantir a **integridade dos dados**, realizar **auditorias** automáticas e **automatizar** certas tarefas que seriam difíceis ou repetitivas no nível da aplicação.

Integridade de Campos/Atributos – Triggers

Tipos de Triggers no MySQL:

BEFORE Trigger: Executa **antes** de a operação ocorrer.
AFTER Trigger: Executa **depois** da operação ocorrer.

Esses triggers podem ser associados a três eventos:
INSERT: Quando uma nova linha é inserida.
UPDATE: Quando uma linha existente é atualizada.
DELETE: Quando uma linha é excluída.

Integridade de Campos/Atributos – Triggers

```
CREATE TRIGGER nome_da_trigger
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON nome_da_tabela
FOR EACH ROW
BEGIN
   -- Ações que serão executadas
END;
```

Integridade de Campos/Atributos – Triggers

```
delimiter //
CREATE TRIGGER valida_salario_funcionario
BEFORE INSERT ON Funcionarios
FOR EACH ROW
BEGIN
   IF NEW.Salario < 1000 THEN
      SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = 'O salário deve ser maior que 1000';
   END IF;
END//
delimiter ;
```

```
INSERT INTO Funcionarios VALUES (4,  'Aline', 300.2);
```

Error Code: 1644. O salário deve ser maior que 1000

Integridade de Campos – Triggers

Gerar um **arquivo de Log** a cada remoção de produto:

```
DELIMITER |
CREATE TRIGGER criar_log
BEFORE DELETE ON produto
FOR EACH ROW
BEGIN
   INSERT INTO log_removidos (nome_produto, data_produto)
        VALUES (OLD.nome, NOW());
END;
|
DELIMITER ;
```

```
DELETE FROM produto where nome = 'Iphone';
SELECT * FROM log_removidos;
```

Integridade de Campos – Triggers

Os principais pontos positivos sobre os **triggers** são:

Parte do processamento que seria executado na aplicação passa para o banco de dados, poupando recursos da máquina cliente.

Facilita a manutenção, sem que seja necessário alterar o código fonte da aplicação.

Integridade de Campos – Triggers

Vantagens de Usar Triggers:

- **Automatização**: As triggers eliminam a necessidade de realizar certas tarefas manualmente, como manter logs ou atualizar campos relacionados.

- **Integridade dos Dados**: Elas ajudam a garantir a integridade e consistência dos dados aplicando regras de negócios diretamente no banco de dados.

- **Auditoria**: Permitem a criação automática de registros de auditoria, que são úteis para rastrear mudanças em sistemas críticos.

.

Integridade de Campos – Triggers

Desvantagens de Usar Triggers:

- **Dificuldade de Depuração**: Como as triggers são executadas automaticamente e "nos bastidores", elas podem ser difíceis de depurar, especialmente em sistemas grandes.
- **Impacto na Performance**: Triggers complexas ou que disparam frequentemente podem impactar o desempenho do banco de dados, especialmente se realizarem operações pesadas.
- **Complexidade Oculta**: Elas podem tornar o comportamento do sistema mais difícil de compreender, pois a lógica de negócios pode estar espalhada pelo banco de dados em vez de centralizada no código da aplicação.

Integridade de Campos – Triggers

Já contra sua utilização existem as seguintes considerações:

Alguém que tenha acesso não autorizado ao banco de dados poderá visualizar e alterar o processamento realizado pelos gatilhos.

Requer maior conhecimento de manipulação do banco de dados (SQL) para realizar as operações internamente.