

Normalização:

(Uma tabela muito Normalizada significa que ele é muito bem feita, e muito relacionado) 5FN (Somente 10% dos DBAs dos ADs conseguem)

A **normalização de dados** é uma série de passos que se seguem no projeto de um **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** 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** 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. O trabalho original de **Edgar F. Codd** definiu três dessas formas, mas existem hoje outras formas normais geralmente aceitas. Damos aqui uma curta panorâmica informal das mais comuns. 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.

- **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;
- **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;
- **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;
- **Forma Normal de Boyce-Codd (ou BCNF)** requer que não exista nenhuma dependência funcional não-trivial de atributos em algo mais do que um

superconjunto de uma chave candidata. Neste estágio, todos os atributos são dependentes de uma chave, de uma chave inteira e de nada mais que uma chave (excluindo dependências triviais, como $A \rightarrow A$);

- **Quarta Forma Normal (ou 4FN)** requer que não exista nenhuma dependência multi-valorada não-trivial de conjuntos de atributo em algo mais de que um superconjunto de uma chave candidata;
- **Quinta Forma Normal (ou 5FN ou PJ/NF)** requer que não exista dependências de *joins* (associações) não triviais que não venham de restrições chave;
- **Domain-Key Normal Form (ou DK/NF)** requer que todas as restrições sigam os domínios e restrições chave.

1 Forma Normal , Evitar Campos multivalorados

igual a Vetores (Atributo ele tem vários Valores ele é multivalorado ...
telefone[]={98199-0108, 3174666, 22629043 }

nome _ só temos um nome

email _ Um e-mail em qualquer site é sua indentificação

dataNascimento _ uma só ..

Estudo de Caso:

Armazenar um Aluno que possua, codigo, nome, email, cpf, disciplina, nota1, nota2, media, telefone, situacao ...

Tabelão Sem Normalização E regra _ Nomes que possuem valores preso a ele (Nomes transitivos)

Antes {Aluno_ codigo, nome,email,cpf, disciplina, nota1, nota2,media, telefone, situacao} de 1 tabela sem Normalização

1FN (Normalização)

Evitar redundância

3 tabelas na 1FN

{Aluno _ idAlunoPK, nome, email, cpf}

{Disciplina_ idDisciplinaPK, nome, nota1, nota2, media, id_ALunoFK}

{Telefone _ idTelefone, numero, operadora, tipo, Id_AlunoFK }

_ Vetores (Telefone) Acabando a Idéia de Vetores

_ Campos com vários Valores (Disciplina)

1ª Forma Normal {Pede não tenha campos mult valorados, não tenha campos de total, jamais armazene totais em tabela}

Porque não guardamos total:

disciplina
{nota1= 5.0, nota2=7.0, media=6}

Em tabelas as pessoas podem alterar
{nota1= 9.0, nota2=7.0, media=6}
por automação (Automaticamente a média é alterada) _ trigger
Trigger é um programa que se prende na tabela, para alterar, inserir, deletar

O Total é Gerado pelo Select ...

```
{Aluno _ idAlunoPK, nome, email, cpf}  
{Disciplina_ idDisciplinaPK, nome, nota1, nota2, media, id_ALunoFK}  
{Telefone _ idTelefone, numero, operadora, tipo,Id_AlunoFK }
```

#Aula2

```
create database modelo2;  
use modelo2;
```

```
select now();
```

```
+-----+  
| now() |  
+-----+  
| 2016-04-14 09:47:53 |  
+-----+
```

```
select user();
```

```
+-----+  
| user() |  
+-----+  
| root@localhost |  
+-----+
```

#CRUD em Uma Tabela (Inserir, Alterar, Excluir, Read (**select**)
#Create, Read, Update e Delete

#logo aluno estará preso a modelo
#Geralmente na WEB o Email obrigatório ...
#Auto_increment (significa automático)
#email não é obrigatorio

#CRUD
#Create, **insert** _ Read(**select**), u _ **update**, D _ **delete**

```
drop table if exists aluno;  
#ou  
drop table aluno;
```

```
create table Aluno (  
    idAluno int primary key auto_increment,  
    nome varchar (35) ,  
    email varchar (50),  
    cpf varchar (15) unique  
);
```

#Cpf (Campo **unique** não pode ser duplicado, ou seja a pessoa só pode ter um cpf)
#iDALuno (automaticamente) ou Não, que se você quiser inserir na mão você pode pode

```
insert into aluno values (null, 'jose',  
'jose@gmail.com', '2222');  
insert into aluno values (null, 'profirio',  
'profirio@gmail.com', '3333');  
insert into aluno values (null, 'lu', null, '4444');  
insert into aluno values (null, 'belem',  
'profedsonbelem@gmail.com' , '5555');
```

#Parte 1 (inserção)
#Parte 2 leitura (**select**)
#buscar todos os dados do ALuno

```
select * from aluno;
```

idAluno	nome	email	cpf
1	jose	jose@gmail.com	2222
2	profirio	profirio@gmail.com	3333
3	lu	NULL	4444
5	belem	profedsonbelem@gmail.com	5555

```
select count(*) as quantidade from aluno;
```

quantidade
4

#Alterar o Nome da Luciana de Lu para lucina

#geralmente eu faço tudo por código

#UPDATE

```
update aluno set nome='luciana' where idAluno=3;
```

```
select * from aluno;
```

idAluno	nome	email	cpf
1	jose	jose@gmail.com	2222
2	profirio	profirio@gmail.com	3333
3	luciana	NULL	4444
4	belem	profedsonbelem@gmail.com	5555

alterar dois campos de Aluno

#nome=belem para =Edson

#cpf=5555 cpf=6666

```
update aluno set nome='Edson',cpf='6666' where idAluno=5;
```

#Geralmente a condição é feita pelo Código ...

#D

#estou apagando o José ...

```
delete from aluno where idAluno=1;
```

```
select * from aluno;
```

idAluno	nome	email	cpf
2	profirio	profirio@gmail.com	3333
3	luciana	NULL	4444
5	belem	profedsonbelem@gmail.com	5555

```
#-----
#Vamos mudar essa idéia
# Essa idéia de apagar é desativar ..
#-----
```

```
drop table if exists aluno;
```

```
create table alunoNovo(
    idAluno int primary key auto_increment,
    nome varchar (50),
    email varchar (50),
    cpf varchar (20),
    ativo int
);
```

```
insert into alunoNovo values
(null,'porfirio','porfirio@gmail.com', '2222',1);
insert into alunoNovo values (null,'belem','belem@gmail.com',
'3333',1);
insert into alunoNovo values (null,'lu','lu@gmail.com',
'4444',1);
insert into alunoNovo values (null,'garra','garra@gmail.com',
'5555',1);
```

```
select * from alunoNovo;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1
4	garra	garra@gmail.com	5555	1

#apaga o aluno garra
#A idéia se apagar deve possuir uma tabela para recebe a lixeira...

```
update alunoNovo set ativo=0 where idALuno=4;
```

```
select * from alunoNovo where ativo=1;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1

```
select * from alunoNovo where ativo=0;
```

idAluno	nome	email	cpf	ativo
4	garra	garra@gmail.com	5555	0

#Criando uma tabela de BACKUP
#BACKUP #ELE iRÀ CRIAR A TABELA ALUNOEXCLUIDO EM BRANCO ...

```
create table alunoExcluido as select * from alunoNovo  
where idAluno<0;
```

#Qual a Intenção (criar a tabela com nenhum registro)
#criar uma tabela em branco ...
Pergunta absurda para criar essa tabela
#Importante verificar se são iguais

```
desc alunoNovo;
```

Field	Type	Null	Key	Default	Extra
idAluno	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(50)	YES		NULL	
email	varchar(50)	YES		NULL	
cpf	varchar(20)	YES		NULL	
ativo	int(11)	YES		NULL	

desc AlunoExcluido;

Field	Type	Null	Key	Default	Extra
idAluno	int(11)	NO		0	
nome	varchar(50)	YES		NULL	
email	varchar(50)	YES		NULL	
cpf	varchar(20)	YES		NULL	
ativo	int(11)	YES		NULL	

select * from alunoNovo;

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1
4	garra	garra@gmail.com	5555	0

select * from alunoExcluido;

#Quando apagar do AlunoNovo e ir para Excluído utilizarei uma **trigger**
 #delimiter espaço o que você quer
 #Toda vez que eu deletar de alunoNovo ele irá para Aluno_Excluído ...
 #Quando eu apagar de AlunoNOvo ele irá para Aluno Excluído

Gatilho ou trigger é um recurso de programação executado sempre que o evento associado ocorrer. Trigger é um tipo especial de procedimento armazenado, que é executado sempre que há uma tentativa de modificar os dados de uma tabela que é protegida por ele.

É muito utilizada para ajudar a manter a consistência dos dados ou para propagar alterações em um determinado dado de uma tabela para outras. Um bom exemplo é um gatilho criado para controle de quem alterou a tabela, nesse caso, quando a alteração for efetuada, o gatilho é "disparado" e grava em uma tabela de histórico de alteração, o usuário e data/hora da alteração. Em SQL, para se criar um trigger utiliza-se do CREATE TRIGGER, e para removê-lo deve-se usar DROP TRIGGER. Um gatilho típico é composto de três componentes, que seguem o Modelo: **evento - condição - ação**^[1].


```
drop trigger if exists gat_delecao_alunoNovo;

delimiter $$
create trigger gat_delecao_alunoNovo
before delete on alunoNovo
for each row
begin
insert into alunoExcluido values
(old.idAluno, old.nome, old.email, old.cpf, old.ativo);
end;
$$
delimiter ;
#Volta ao Normal no Final da Programação
# delimiter ;
```

```
#toda vez que eu for programar no Mysql, sou Obrigado
#Sempre, a mudar o fim, o fim é ponto vírgula
#mas programar devo mudar de ; para qualquer outro sinal
# o que eu utilizo muito $$
```

```
# Irá Dar Erro
```

```
#create trigger gat_ERRO
# before delete on alunoNovo
# for each row
# begin
#
# insert into alunoExcluido values
# (old.idAluno, old.nome, old.email, old.cpf, old.ativo);
#
# end;
```

```
#Mostra A TRIGGER
```

```
show create trigger gat_delecao_alunoNovo;
```

```
+-----+-----+
| Trigger | sql_mode |
| SQL Original Statement | |
```

```
+-----+  
-----+  
-----  
-----  
-----
```

```
|          gat_delecao_alunoNovo          |  
STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |  
CREATE DEFINER=`root`@`localhost` trigger gat_delecao_alunoNovo  
  before delete on alunoNovo  
  for each row  
  begin  
  
    insert into alunoExcluido values  
      (old.idAluno, old.nome, old.email, old.cpf, old.ativo);  
  
  end | latin1          | latin1_swedish_ci  |  
latin1_swedish_ci |
```

```
+-----+  
-----+  
-----  
-----  
-----
```

```
select * from alunoNovo;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1
4	garra	garra@gmail.com	5555	0

```
select * from alunoEXcluido;
```

#Testar o Gatiho

```
delete from alunoNovo where idAluno=4;
```

```
select * from alunoNovo;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1

```
select * from ALunoExcluido;
```

idAluno	nome	email	cpf	ativo
4	garra	garra@gmail.com	5555	0

#AlunoExcluido é um Backup da AlunoNovo para Delecao
#isso Aconteceu devido A TRIGGER (AO GATILHO)

```
{Aluno _ idAlunoPK, nome, email, cpf, ativo}
{Disciplina_ idDisciplinaPK, nome, nota1, nota2, media,
id_ALunoFK}
{Telefone _ idTelefone, numero, operadora, tipo,Id_AlunoFK }
```

```
create table disciplina(
    idDisciplina int primary key Auto_increment,
    nomeDisciplina varchar (35),
    nota1 double,
    nota2 double,
    id_aluno int,
    foreign key(id_Aluno) references AlunoNovo(idAluno)
on delete cascade
);
```

```
create table disciplinaExcluida as select * from disciplina
where idDisciplina <0;
drop trigger if exists gat_delecao_disciplina;
```

```
delimiter $$
create trigger gat_delecao_disciplina
before delete on disciplina
for each row
begin
```

```

insert into disciplinaExcluida values (old.iddisciplina,
old.nomedisciplina, old.nota1, old.nota2, old.id_aluno);
end;
$$
delimiter ;

```

```
desc disciplina;
```

Field	Type	Null	Key	Default	Extra
idDisciplina auto_increment	int(11)	NO		PRI	NULL
nomeDisciplina	varchar(35)	YES		NULL	
nota1	double	YES		NULL	
nota2	double	YES		NULL	
id_aluno	int(11)	YES	MUL	NULL	

```
select * from alunoNovo;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1

```

insert into disciplina values (null,'mysql',9,9, 1);
insert into disciplina values (null,'java',7,8, 2);
insert into disciplina values (null,'java',9,8, 3);

```

```
#Atualiza ...
```

```
commit;
```

```
select * from alunonovo;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
2	belem	belem@gmail.com	3333	1
3	lu	lu@gmail.com	4444	1

```
select * from disciplina;
```

idDisciplina	nomeDisciplina	nota1	nota2	id_aluno
1	mysql	9	9	1
2	java	7	8	2
3	java	9	8	3

```
delete from alunoNovo where idAluno=2;
```

#irá apagar o ALunoNovo o idAluno da tabela ALunoNovo e a Disciplina

```
select * from alunoNovo;
```

idAluno	nome	email	cpf	ativo
1	porfirio	porfirio@gmail.com	2222	1
3	lu	lu@gmail.com	4444	1

```
select * from disciplina;
```

idDisciplina	nomeDisciplina	nota1	nota2	id_aluno
1	mysql	9	9	1
3	java	9	8	3

```
select * from alunoexcluido;
```

idAluno	nome	email	cpf	ativo
4	garra	garra@gmail.com	5555	0
2	belem	belem@gmail.com	3333	1

```
select * from disciplinaexcluida;
```



```
select a.nome, d.nomedisciplina,(d.nota1 + d.nota2)/2 as media  
from alunonovo a, disciplina d where a.idAluno = d.id_aluno;
```

nome	nomedisciplina	media
porfirio	mysql	9
lu	java	8.5



Escola de Nerds

www.cotiinformatica.com.br