Normalização de Dados

É um conjunto de limitações impostas a uma estrutura de dados para que a mesma fique mais concisa e sem falhas estruturais.

* Codd -> 1970
* DATE -> 3FN (aperfeiçoada)
* FAGIN -> 4FN e 5FN -> 1977

DEPENDÊNCIA FUNCIONAL

Sejam E uma entidade, X e Y dois atributos quaisquer de E. Diz-se que Y é funcionalmente dependente de X se e somente se cada valor de X estiver associado a exatamente um valor de Y.

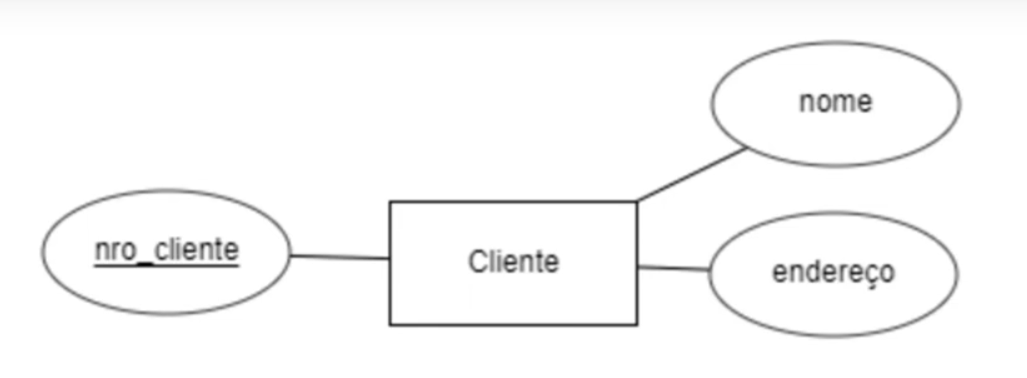
Notação:

X -> Y

Onde, X determina funcionalmente Y, ou Y depende funcionalmente de X.

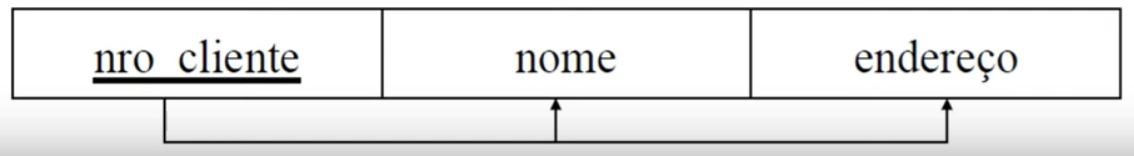
Exemplo -> numero\_pedido -> prazo\_entrega

Anotação gráfica desta relação:



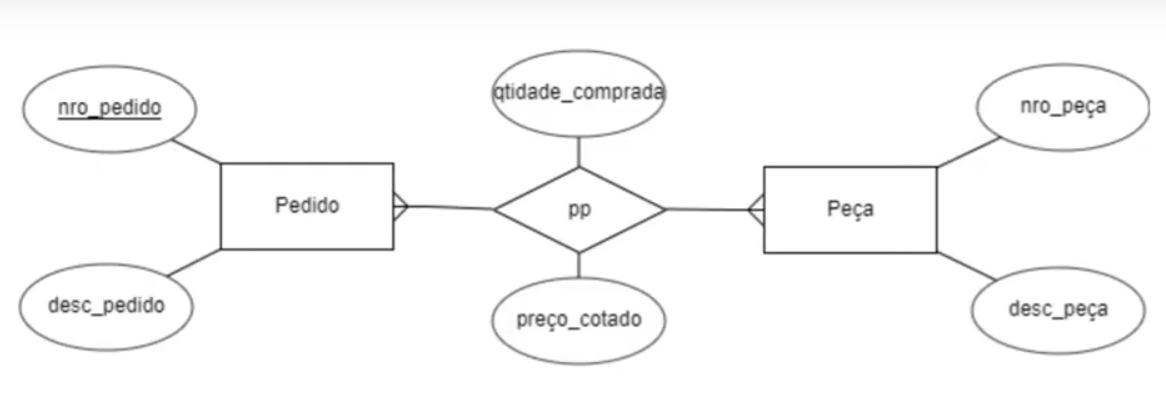
cliente(nro\_cliente, nome, endereço)

nro\_cliente -> {nome, endereço}



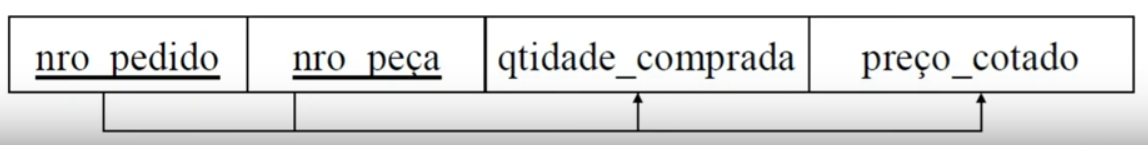
Dependência funcional total

Em uma relação com uma PK composta, um atributo não-chave que depende dessa PK como um todo, e não somente como parte dela, possui dependência funcional total.



pp (nro\_pedido, nro\_preça, qtidade\_comprada, preço\_cotado}

{nro\_pedido, nro\_peça} -> {qtidade\_comprada, preço\_cotado}



Dependência funcional parcial

Há dependência funcional parcial quando um atributo não-chave depende apenas de parte de uma chave primária composta.

(Há dependência funcional, mas ela é de parte parcial da chave, apenas)

Normalmente, ocorre quando há dados no lugar errado ou quando há redundância de dados

Dependência funcional transitiva

Ocorre quando um atributo não depende funcionalmente da chave primária, mas depende de outro atributo não-chave.

Normalmente, ocorre quando há má modelagem dos dados.

X -> Y e Y -> Z

Dependência multivalorada

Ocorre quando, para cada valor de um atributo A, existe um conjunto de valores de outros atributos B e C que estão associados a ele, mas que são independentes entre si.

Formalização: um atributo B de uma relação R é multidependente de um atributo A de R se um valor de A é associado a uma coleção de valores de B, independentemente de qualquer valor de outro atributo C de R.

Notação:

A ->-> B ou A ->> B

NORMALIZAÇÃO

Processo de análise que visa assegurr que uma relação está bem formada.

Ideia: decompor relações co manomalias para produzir relações menores e mais bem estruturadas.

Relações normalizadas -> inserção, remoção e atualização isentas de anomalias

Anomalias

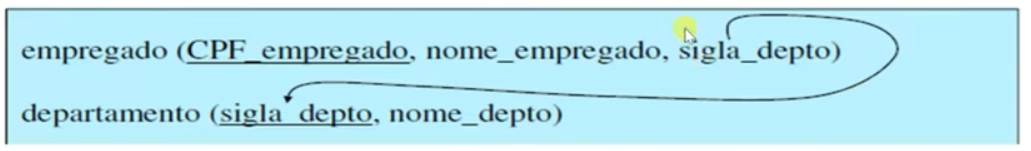
São problemas que ocorrem em BDs.

Decorrem de mau planejamento

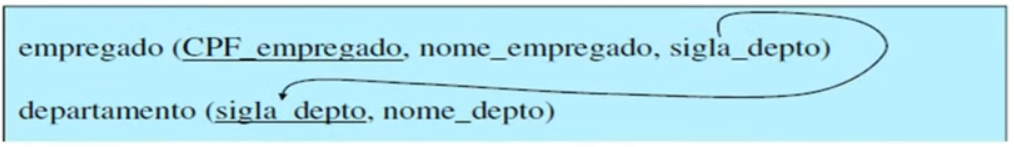
* Excesso de dados em uma mesma relação
* Dados em locais errados

Três tipos

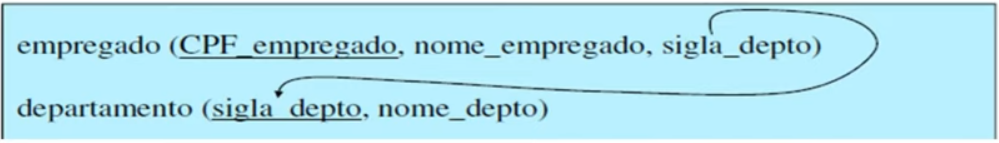
1. Anomalia de inserção



1. Anomalia de remoção



1. Anomalia de atualização



Normalização

Conceito introduzido em 1972 por Codd

São testes para certificar que uma relação satisfaz formas normais (FN).

Codd propôs que boas relações atendem a pelo menos 3 formas normais.

Existem outras formas:

- Forma normal de Boyce-Codd (FNBC)

- 4 FN

- 5 FN

1 FN

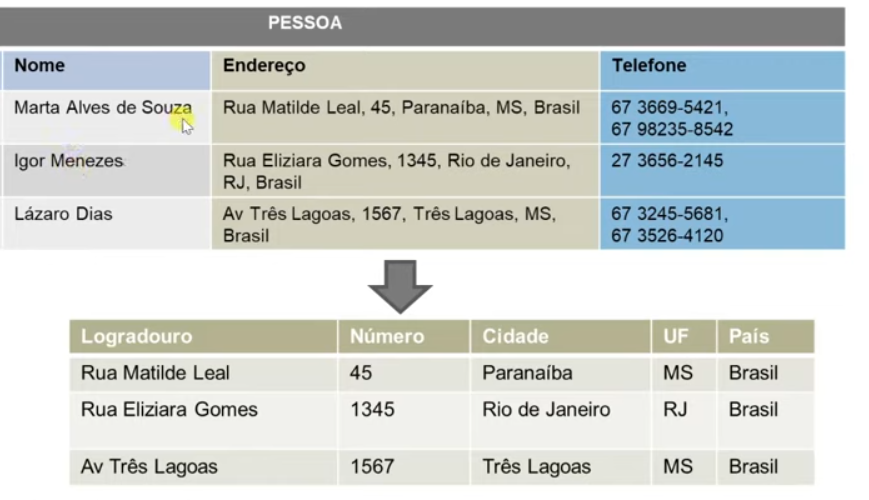
Reprova atributos multivalorados, compostos e suas combinações



Endereço: é um atributo composto, pois podemos separar os valores dele

Telefone: é multivalorado por ter mais de um valor no mesmo campo

1 Passo:



2 Passo:



Uma relação está na 1FN se:

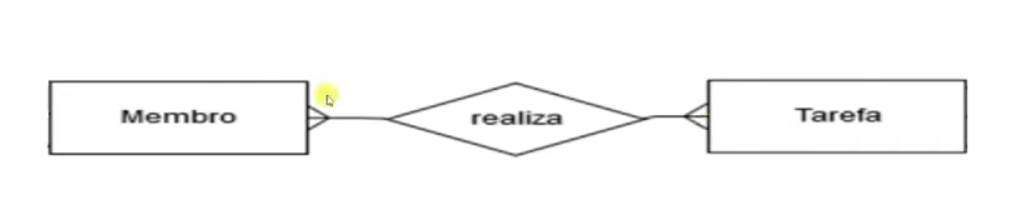
* Todos os valores são atômicos (indivisíveis)
* Há apenas um dado por coluna
* Existe uma chave primária (pelo menos)
* Não há relações aninhadas (tabelas dentro de tabelas)

2 FN

Uma relação está na 2FN se:

* Está na 1FN
* Não contém dependências parciais (Todas com dependência total)





N para N



Uma relação está em 2FN se:

- Se está em 1FN

- Se a chave primária for simples (apenas um atributo)

- Se nenhum atributo não-chave existe na relação (todos os atributos são parte da chave primária)

- Se todo atributo não-chave é dependente funcionalmente de todo o conjunto de atributos da chave primária

3 FN

Uma relação está na 3FN se está na 2FN e não há atributo não-chave determinado funcionalmente por outro atributo não-chave.

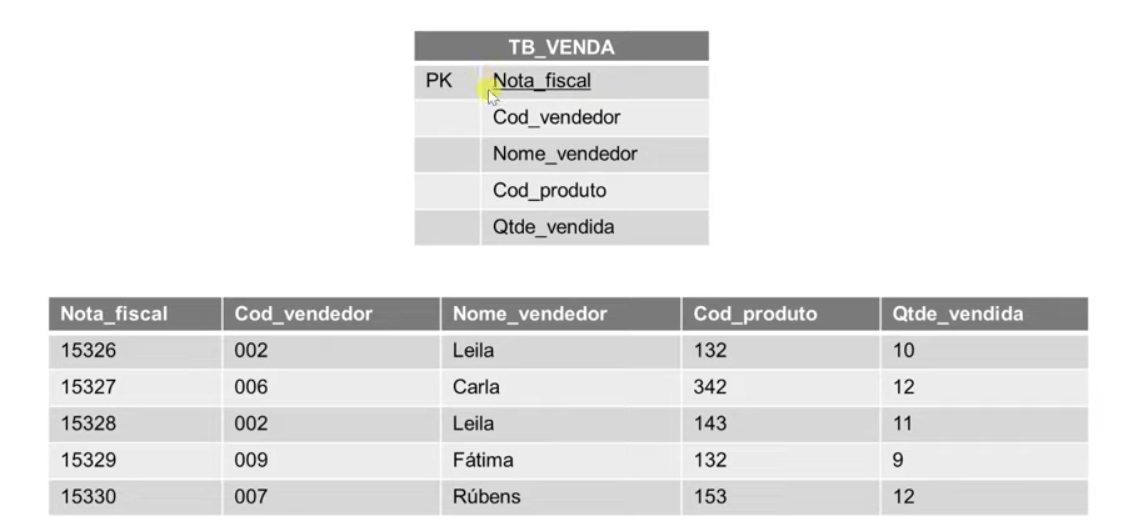
- Não há dependência transitiva

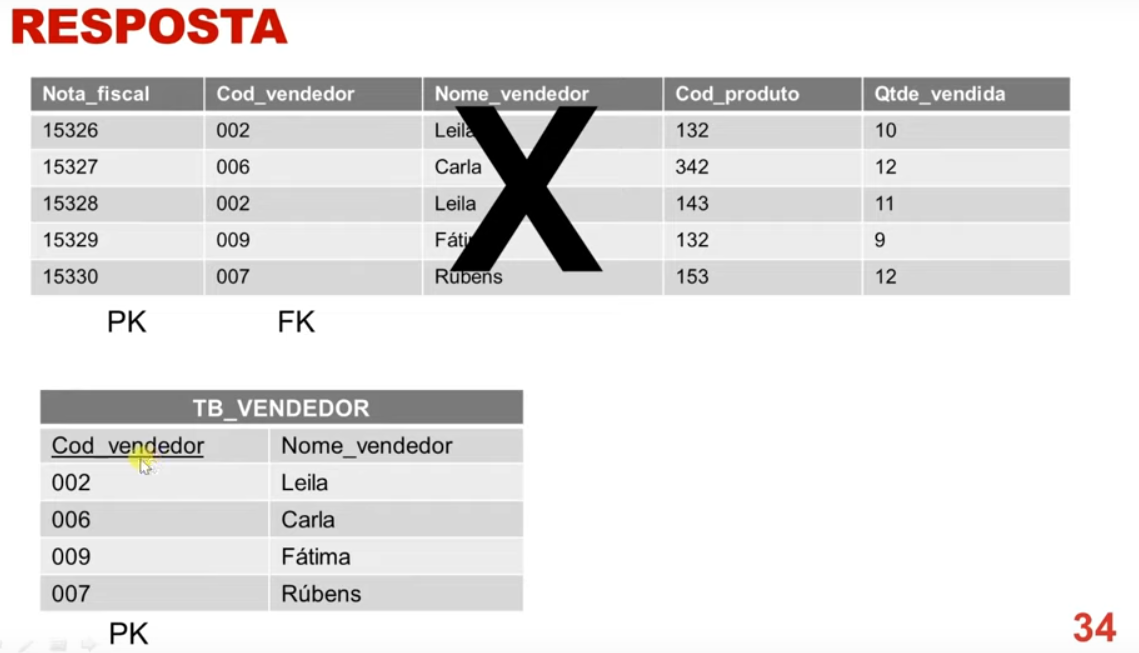
Solução para relações fora da 3FN

- Para cada atributo não-chave que determina funcionalmente outro atributo, crie uma nova tabela.

Este atributo não-chave será a PK desta nova tabela.

Mova todos os atributos dependentes funcionalmente da nova PK para a nova tabela.





SQL {DDL, DML, DQL, DCL, DTL}

Desenvolvida nos anos 70 pela IBM

DDL – Data Definition Language

DML – Data Manipulation Language

DQL – Data Query Language

DCL – Data Control Language

DTL – Data Transaction Language

DDL

- CREATE

- DROP

- ALTER

- TRUNCATE

- COMMENT

- RENAME

DML

- INSERT

- DELETE

- UPDATE

- CALL

- EXPLAINPLAN

- LOCK TABLE

DQL

- SELECT

DCL

- GRANT

- REVOQUE

- DENY

DTL

- COMMIT

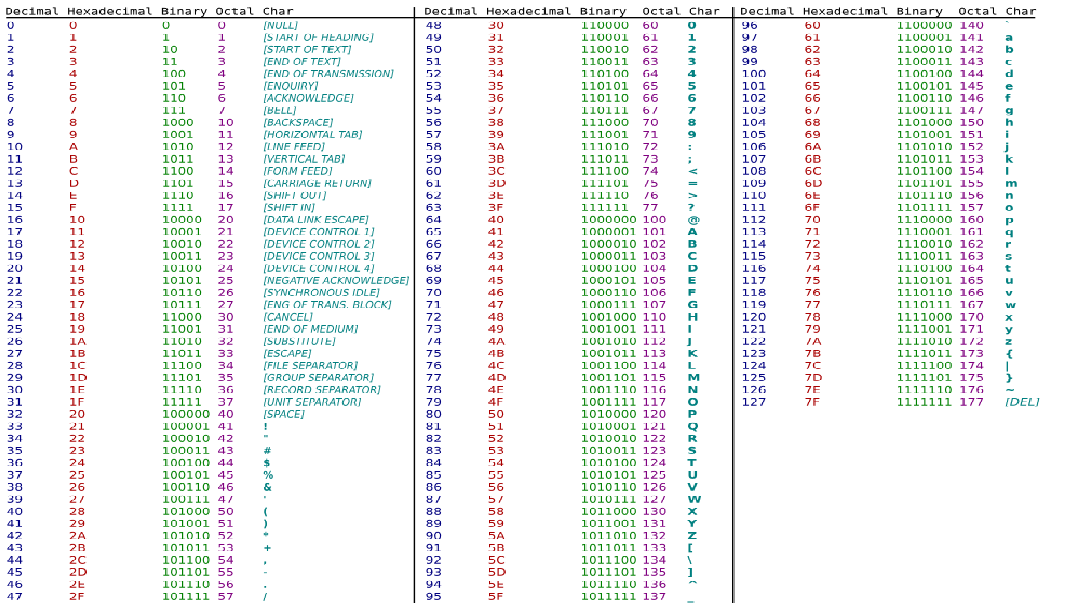
- SAVEPOINT

- ROLLBACK

TIPOS DE DADOS

Obs: todos os dados são armazenados no banco de dados em formato de bytes.

Tabela ASCII



Caracter

- char

- varchar

- nchar

- nvarchar

- text

- tinytext

- mediumtext

- longtext

- char

- varchar

Números exatos

- bit

- decimal ou numeric

Numerais

- tinyint

- smallint

- mediuint

- Int

- bigint

- real

- float

- double

- money

- decimal

Data e hora

- date

- datetime

- smalldatetime

- time

- timestamp

- blob

Linhas de comentário

-- Comentario

# Comentario

/\*

Múltiplas linhas de comentário

\*/

(DDL) CREATE DATABASE

CREATE DATABASE IF NOT EXISTS <NOME\_BANCO>;

USE <NOME\_BANCO>;

(DDL) CREATE TABLE

CREATE TABLE IF NOT EXISTS <NOME\_TABELA>

(“coluna 1” “tipo\_dados”,

“coluna 2” “tipo dados”,

…);

SHOW

SHOW DATABASES;

SHOW TABLES;

DESCRIBE

DESC <NOME\_TABELA>;

DESCRIBE <NOME\_TABELA>;

NOT NULL: garante que uma coluna não pode ter o valor NULL.

DEFAULT: fornece um valor padrão para uma coluna quando nenhum é especificado.

UNIQUE: garante que todos os valores numa coluna são diferentes.

CHECK: garante que todos os valores numa coluna satisfazem um determinado critério.

PRIMARY KEY: utilizado para identificar de forma única uma linha na tabela.

FOREIGN KEY: utilizado para garantir a integridade referencial dos dados.

CREATE TABLE ESTADO (

SIGLA\_EST VARCHAR(02) NOT NULL,

NOME\_EST VARCHAR(100) NOT NULL,

PRIMARY KEY (SIGLA\_EST)

);

CREATE TABLE MUNICIPIO (

COD\_MUN INT NOT NULL,

NOME\_MUN VARCHAR(100) NOT NULL,

SIGLA\_EST VARCHAR(02),

PRIMARY KEY (COD\_MUN),

CONSTRAINT FK\_EST FOREIGN KEY (SIGLA\_EST) REFERENCES

ESTADO (SIGLA\_EST);

CONSTRAINTS

- Not Null

- Unique

- Check

- Primary Key

- Foreign Key

FOREIGN KEY

- On delete: significa que uma ação referencial será executada quando um registro for excluído da tabela pai.

- On update: indica que uma ação referencial será executada quando um registro for modificado na tabela pai.

- Cascade: permite excluir ou atualizar os registros relacionados presentes na tabela filha automaticamente, quando um registro na tabela pai for atualizado (on update) ou excluido (on delete). É a opção mais comum aplicada.

- Restrict: Impede que ocorra a exclusão ou a atualização de um registro da tabela pai, caso ainda hajam registros na tabela filha. Uma exceção deviolação de chave estrangeira é retornada.

- No action: essa opção equivale à opção RESTRICT, porém a verificação de integridade referencial é executada depois da tentativa de alterar a tabela.

- Set null: esta opção é usada para definir com o valor NULL o campo da tabela filha, aplicado quando um registro da tabela pai for atualizado ou excluído.

CHECK:

A avaliação do critério de pesquisa deve usar uma expressão Booleana (true/false) como base e não pode fazer referência a outra tabela.

A restrição CHECK no nível de coluna pode fazer referência somente à coluna restrita.

Exemplo: CONSTRAINT ckIdade CHECK (Idade <= 100)

CONSTRAINT CK\_carreira\_id CHECK (carreira\_id LIKE ‘C[0-9][0-9Ç’] (Aceita C12 e não B23)

CONSTRAINT CK\_carreira\_id CHECK (carreira\_id IN (‘C01’, ‘C11’, ‘C12’, ‘C15’, ‘C17’) OR carreira\_id LIKE ‘9[0-9][0-9]’)

Usando com o Like:

O operador LIKE é usado em uma condição para procurar um padrão especificado.

Exemplo: PAIS LIKE ‘Espanha’.

O operador IN testa vários valores de uma lista:

Exemplo: PAIS IN (‘Alemanha’, ‘França’, ‘Reino Unido’);

PAIS NOT IN (‘Alemanha’, ‘França’, ‘Reino Unido’);

DLL – DROP TABLE

Exclui a estrutura de uma tabela do banco de dados

Obs: Se uma tabela for referenciada (tiver filhos), não será possível excluí-las sem antes apagar as filhas.

DDL – ALTER TABLE

ALTER TABLE <nome\_tabela> ADD <nome\_coluna> <tipo\_de\_dados>

ALTER TABLE <nome\_tabela> CHANGE COLUMN <nome\_atual> <novo\_nome> <novo\_tipo>

ALTER TABLE <nome\_tabela> DROP COLUMN <nome\_coluna>

ALTER TABLE <nome\_tabela> ADD CONSTRAINT PRIMARY KEY <nome\_campo>

O Change é utilizado para renomear uma coluna, alterar o tipo, valor padrão etc.

O Modify Column fica limitado apenas a alterações de tipo, valor padrão etc.

Rename

Exemplo: ALTER TABLE <nome\_tabela> RENAME <novo\_nome>

ALTER TABLE <nome\_tabela> ADD CONSTRAINT <nome\_ck> <tipo\_ck> (<campo>)

ALTER TABLE <nome\_tabela> DROP CONSTRAINT <nome\_campo>;

ALTER TABLE <nome\_tabela> DROP PRIMARY KEY;

DML – INSERÇÃO

Insere uma linha em uma tabela.

- INSERT VALUES

- INSERT SELECT

- INSERT EXEC

- SELECT INTO

TRUNCATE:

Apaga todos os dados e mantém a estrutura da tabela.

INSERT SELECT:

Insere o conjunto de resultado devolvido por uma consulta em uma tabela especificada.

INSERT INTO <tabela 2> (column1, column2, column3, …)

SELECT column1, column2, colum3, … FROM <tabela 1>

WHERE condition;

INSERT INTO autores2

SELECT id\_autor, nome\_autor, nacionalidade FROM autores

WHERE nacionalidade IN ("americano", "portugues");

INSERT INTO autores2

SELECT id\_autor, nome\_autor, nacionalidade FROM autores

WHERE nacionalidade LIKE "%A%";

Ordem de escrita:

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY

Ordem de execução:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

DISTINCT:  
é usada em consultas para retornar apenas valores distintos (únicos) em uma determinada coluna ou conjunto de colunas.

SELECT DISTINCT column1, column2, …

FROM table

WHERE condition;

CONCAT