



# Banco de Dados

Diego Silveira Costa Nascimento

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte  
diego.nascimento@ifrn.edu.br

6 de março de 2019

- 1 Introdução
- 2 Abordagem Entidade-relacionamento
- 3 Abordagem Relacional
- 4 Normalização
- 5 Structured Query Language (SQL)
- 6 Consulta SQL



- 1 Introdução
- 2 Abordagem Entidade-relacionamento
- 3 Abordagem Relacional
- 4 Normalização
- 5 Structured Query Language (SQL)
- 6 Consulta SQL



## Definição

São conjuntos de registros dispostos em estrutura regular que possibilita a organização dos dados e produção de informação.



# Sistema Gerenciador de Banco de Dados (SGBD)

## Definição

É uma coleção de programas de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos entre vários usuários e aplicações.

## Exemplos

- Oracle
- Microsoft SQL Server
- Postgre SQL
- MySQL
- MariaDB
- MongoDB
- IBM DB2

- 1 Introdução
- 2 Abordagem Entidade-relacionamento**
- 3 Abordagem Relacional
- 4 Normalização
- 5 Structured Query Language (SQL)
- 6 Consulta SQL



## Definição

É uma descrição dos tipos de informações que estão armazenadas em um banco de dados.

- Para construir um modelo de dados usa-se uma linguagem de modelagem de dados;
- A linguagem de modelagem pode ser textual ou gráfica;
- Existem linguagens de modelagem para descrever modelos de dados em diferentes níveis de abstração e objetivos; e
- Cada representação de um modelo de dados recebe a denominação de esquema de banco de dados.



## Definição

Conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações no banco de dados.

## Ilustração

**Pessoa**

**Departamento**

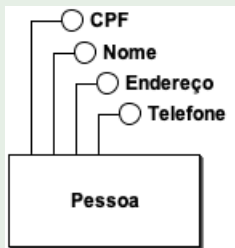




## Definição

Dado que é associado a cada ocorrência de uma entidade.

## Ilustração



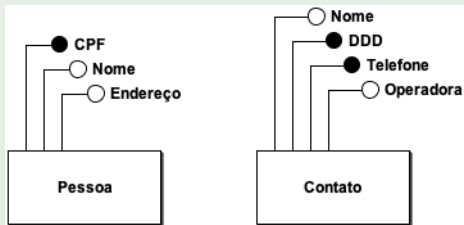
# Identificador da Entidade

## Definição

Um identificador é um conjunto de um ou mais atributos (e possivelmente relacionamentos, como visto abaixo) cujos valores servem para distinguir uma ocorrência da entidade das demais ocorrências da mesma entidade.

- Simples; ou
- Composta.

## Ilustração



## Definição

Conjunto de associações entre ocorrência de entidades.

## Ilustração



## Definição

É o número (mínimo, máximo) de ocorrências de entidade associadas a uma ocorrência da entidade em questão através do relacionamento.

Classificação de relacionamentos binários:

- 1:1 (um-para-um);
- 1:N (um-para-muitos); e
- N:N (muitos-para-muitos)



- Entidade associativa; e
- Generalização.



- 1 Introdução
- 2 Abordagem Entidade-relacionamento
- 3 Abordagem Relacional**
- 4 Normalização
- 5 Structured Query Language (SQL)
- 6 Consulta SQL



- É um conjunto não ordenado de linhas (tuplas);
- Cada linha é composta por uma série de campos (valor do atributo);
- Cada campo é identificado por nome de campo (nome de atributo); e
- O conjunto de campos das linhas de uma tabela que possuem o mesmo nome formam uma coluna.



## Definição

E a forma na qual se estabelece relações entre linhas de tabelas de um banco de dados relacional.

Tipos de chave:

- Primária; e
- Estrangeira.





## Definição

É uma coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela.

## Departamento

CODIGO	DESCRICAO
D1	Compras
D2	Engenharia
D3	Vendas



## Definição

É uma coluna cujos valores aparecem necessariamente na chave primária de uma tabela.

## Empregado

CODIGO	NOME	COD_DEPART
E1	Souza	D1
E2	Santos	D2
E3	Silva	D2
E4	Soares	D3



- 1 Introdução
- 2 Abordagem Entidade-relacionamento
- 3 Abordagem Relacional
- 4 Normalização**
- 5 Structured Query Language (SQL)
- 6 Consulta SQL



## Definição

É uma regra que deve ser obedecida por uma tabela para que esta seja considerada “bem projetada”.

Tipos de normalização:

- Primeira forma normal (1FN);
- Segunda forma normal (2FN);
- Terceira forma normal (3FN); e
- Quarta forma normal (4FN).



# Tabela Não Normalizada

## Projeto\_Empregado

COD_PROJ	TIPO DESC	COD_EMP	NOME	CAT	SAL	INI	DURAC
LSC001	Desenvolvimento Sistema de Estoque	2146	João	A1	4	01/11/91	24
LSC001	Desenvolvimento Sistema de Estoque	3145	Sílvio	A2	4	02/10/91	24
LSC001	Desenvolvimento Sistema de Estoque	6126	José	B1	9	03/10/92	18
LSC001	Desenvolvimento Sistema de Estoque	1214	Carlos	A2	4	04/10/92	18
LSC001	Desenvolvimento Sistema de Estoque	8191	Mário	A1	4	01/11/92	12
PAG02	Manutenção Sistema de RH	8191	Mário	A1	4	01/05/93	12
PAG02	Manutenção Sistema de RH	4112	João	A2	4	04/01/91	24
PAG02	Manutenção Sistema de RH	6126	José	B1	9	01/11/92	12



# Primeira Forma Normal (1FN)

## Definição

Diz-se que uma tabela está na primeira forma normal, quando ela não contém tabelas aninhadas.



# Tabela na 1FN

## Projeto

COD	TIPO	DESC
LSC001	Desenvolvimento	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

## Empregado

COD_PROJ	COD_EMP	NOME	CAT	SAL	INI	DURAC
LSC001	2146	João	A1	4	01/11/91	24
LSC001	3145	Sílvio	A2	4	02/10/91	24
LSC001	6126	José	B1	9	03/10/92	18
LSC001	1214	Carlos	A2	4	04/10/92	18
LSC001	8191	Mário	A1	4	01/11/92	12
PAG02	8191	Mário	A1	4	01/05/93	12
PAG02	4112	João	A2	4	04/01/91	24
PAG02	6126	José	B1	9	01/11/92	12

# Dependência Funcional

## Definição

Diz-se que uma coluna C2 depende funcionalmente de uma coluna C1 (ou que a coluna C1 determina a coluna C2) quando, em todas linhas da tabela, para cada valor de C1 que aparece na tabela, aparece o mesmo valor de C2.

## Exemplo

CODIGO	SALARIO
E1	10
E3	10
E1	10
E2	5
E3	10
E2	5
E1	10



# Segunda Forma Normal (2FN)

## Definição

Uma tabela encontra-se na segunda forma normal, quando, além de estar na 1FN, não contém dependências parciais.



# Tabela na 2FN

## Projeto\_Empregado

COD_PROJ	COD_EMP	INI	DURAC
LSC001	2146	01/11/91	24
LSC001	3145	02/10/91	24
LSC001	6126	03/10/92	18
LSC001	1214	04/10/92	18
LSC001	8191	01/11/92	12
PAG02	8191	01/05/93	12
PAG02	4112	04/01/91	24
PAG02	6126	01/11/92	12



## Empregado

COD_EMP	NOME	CAT	SAL
2146	João	A1	4
3145	Sílvio	A2	4
6126	José	B1	9
1214	Carlos	A2	4
8191	Mário	A1	4
8191	Mário	A1	4
4112	João	A2	4
6126	José	B1	9



# Terceira Forma Normal (3FN)

## Definição

Uma tabela encontra-se na terceira forma normal, quando, além de estar na 2FN, não contém dependências transitivas.



## Empregado

COD_EMP	NOME	CAT
2146	João	A1
3145	Sílvio	A2
6126	José	B1
1214	Carlos	A2
8191	Mário	A1
8191	Mário	A1
4112	João	A2
6126	José	B1



# Tabela na 3FN

## Categoria

CAT	SAL
A1	4
A2	4
B1	9



# Quarta Forma Normal (4FN)

## Definição

Uma tabela encontra-se na quarta forma normal, quando, além de estar na 3FN, não contém dependências multi-valoradas.



## Equipamento

COD	DESC
EQ1	Projektor
EQ2	Notebook
EQ3	Roteador





## Projeto\_Empregado\_Equipamento

COD_PROJ	COD_EMP	COD_EQUI
PAG02	8191	EQ1
PAG02	4112	EQ1
PAG02	6126	EQ1
PAG02	8191	EQ2
PAG02	4112	EQ2
PAG02	6126	EQ2
PAG02	8191	EQ3
PAG02	4112	EQ3
PAG02	6126	EQ3



# Tabela na 4FN

## Projeto\_Equipamento

COD_PROJ	COD_EQUI
PAG02	E1
PAG02	E2
PAG02	E3



- 1 Introdução
- 2 Abordagem Entidade-relacionamento
- 3 Abordagem Relacional
- 4 Normalização
- 5 Structured Query Language (SQL)**
- 6 Consulta SQL



- É uma linguagem de pesquisa declarativa padrão para banco de dados relacional ;
- Implementada no projeto de pesquisa do System R da IBM em meados dos anos 70;
- Padronizada em 1986 e melhorada em 1989;
- SQL-2 ou SQL-92: Padrão hoje em vigor; e
- SQL-3: Em fase de desenvolvimento. Vai estender o padrão atual com conceitos de orientação a objeto e outros novos conceitos de BDs.



- Definição de dados; e
- Manipulação dos dados.



- Criação e exclusão de banco de dados;
- Criação, alteração e exclusão de tabela;
  - Especificação de restrições;
- Criação de visão;
- Criação de procedimento armazenado;
- Criação de função; e
- Criação de gatilho.



# Criando Banco de Dados

## Estrutura

```
CREATE DATABASE <nome do banco>;
```

## Exemplo

```
CREATE DATABASE academico;
```

## Cuidado

```
DROP DATABASE <nome do banco>;
```



# Criando Tabela

## Estrutura

```
CREATE TABLE <nome da tabela>(  
    <nome do atributo 1> <tipo de dado>,  
    <nome do atributo 2> <tipo de dado>  
)
```

## Exemplo

```
CREATE TABLE tb_aluno(  
    matricula INT,  
    nome VARCHAR(50),  
    sexo CHAR  
)
```

## Cuidado

```
DROP TABLE <nome da tabela>;
```



## Exemplo

```
CREATE TABLE tb_disciplina(  
    identificador INT,  
    descricao VARCHAR(50),  
    credito INT  
)
```

```
CREATE TABLE tb_disciplina_cursada(  
    matricula_aluno INT,  
    identificador_disciplina INT,  
    semestre INT,  
    ano INT,  
    nota FLOAT  
)
```



# Alterando Tabela

## Estrutura

```
ALTER TABLE <nome da tabela>  
ADD <nome do atributo> <tipo de dado>
```

## Exemplo

```
ALTER TABLE tb_aluno  
ADD cpf VARCHAR(11)
```



# Comandos de Manipulação de Dados

- Inserir;
- Selecionar;
- Atualizar; e
- Excluir.



## Estrutura

```
INSERT INTO <nome da tabela>  
(<nome do atributo 1>,<nome do atributo 2>)  
VALUES  
('valor 1','valor 2')
```

## Exemplo

```
INSERT INTO tb_aluno  
(matricula,nome,sexo,cpf)  
VALUES  
('1','João da Silva','M','12345678')
```



# Selecionando Registros

## Estrutura

```
SELECT <nome do atributo 1>, <nome do atributo 2>  
FROM <nome da tabela>
```

## Exemplo

```
SELECT matricula, nome  
FROM tb_aluno
```

Caso necessite selecionar todos os atributos de uma única vez, deve-se apenas utilizar o símbolo de asterisco (\*) em substituição do(s) nome(s) da(s) coluna(s).



## Estrutura

```
UPDATE <nome da tabela>  
SET <nome do atributo> = '<valor>'  
WHERE <condição>
```

## Exemplo

```
UPDATE tb_aluno  
SET nome = 'João da Silva Filho'  
WHERE matricula = '1'
```



## Estrutura

```
DELETE FROM <nome da tabela>  
WHERE <condição>
```

## Exemplo

```
DELETE FROM tb_aluno  
WHERE matricula = '1'
```



- Declarativas:
  - Domínio;
  - Vazio;
  - Padrão;
  - Checagem;
  - Unicidade;
  - Chave primária; e
  - Referencial.
- Procedimental:
  - Procedimento armazenado
  - Função; e
  - Gatilho.





## Exemplo

```
ALTER TABLE tb_aluno  
MODIFY COLUMN nome VARCHAR(30) NOT NULL;
```



## Exemplo

```
ALTER TABLE tb_aluno  
ALTER sexo SET DEFAULT 'M';
```

## Cuidado

```
ALTER TABLE <nome da tabela>  
ALTER <nome da coluna> DROP DEFAULT;
```



## Exemplo

```
ALTER TABLE tb_disciplina_cursada  
ADD CONSTRAINT ck_nota  
CHECK (nota>=0 AND nota<=10);
```

## Cuidado

```
ALTER TABLE <nome da tabela>  
DROP CHECK <nome da restrição>;
```



## Exemplo

```
ALTER TABLE tb_aluno  
ADD CONSTRAINT uc_cpf  
UNIQUE (cpf);
```

## Cuidado

```
ALTER TABLE <nome da tabela>  
DROP INDEX <nome da restrição>;
```



## Exemplos

```
ALTER TABLE tb_aluno  
ADD CONSTRAINT pk_aluno  
PRIMARY KEY (matricula);
```

```
ALTER TABLE tb_disciplina  
ADD CONSTRAINT pk_disciplina  
PRIMARY KEY (identificador);
```

```
ALTER TABLE tb_disciplina_cursada  
ADD CONSTRAINT pk_tb_disciplina_cursada  
PRIMARY KEY (matricula_aluno, identificador_disciplina, semestre, ano)
```

## Cuidado

```
ALTER TABLE <nome da tabela>  
DROP PRIMARY KEY;
```



## Exemplos

```
ALTER TABLE tb_disciplina_cursada  
ADD CONSTRAINT fk_aluno  
FOREIGN KEY (matricula_aluno)  
REFERENCES tb_aluno(matricula);
```

```
ALTER TABLE tb_disciplina_cursada  
ADD CONSTRAINT fk_disciplina  
FOREIGN KEY (identificador_disciplina)  
REFERENCES tb_disciplina(identificador);
```

## Cuidado

```
ALTER TABLE <nome da tabela>  
DROP FOREIGN KEY <nome da restrição>;
```



## Estrutura

```
CREATE VIEW <nome> AS  
<consulta sql>;
```

## Exemplo

```
CREATE VIEW vw_aluno AS  
SELECT nome, cpf  
FROM vw_aluno;
```

## Cuidado

```
DROP VIEW <nome da visão>;
```



## Estrutura

```
DELIMITER $  
  
CREATE PROCEDURE <nome>(<parâmetro> <tipo do parâmetro>)  
BEGIN  
    <comando sql>;  
END  
$
```

## Exemplo

```
DELIMITER $  
  
CREATE PROCEDURE inserir_aluno(a VARCHAR(30), b CHAR(1), c VARCHAR(11))  
BEGIN  
    INSERT INTO tb_aluno (nome, sexo, cpf) VALUES (a,b,c);  
END  
$
```





## Cuidado

```
DROP PROCEDURE <nome do procedimento>;
```



## Estrutura

```
DELIMITER $  
  
CREATE FUNCTION <nome>(<parâmetro> <tipo do parâmento>)  
RETURNS <retorno do parâmento>  
  
BEGIN  
    <código>;  
END  
$
```



## Exemplo

```
DELIMITER $

CREATE FUNCTION desc_sexo(sexo char(1))
RETURNS VARCHAR(10)

BEGIN
    IF SEXO = 'M' THEN
        RETURN "MASCULINO";
    ELSEIF SEXO = 'F' THEN
        RETURN "FEMININO";
    ELSE
        RETURN "NAO INFORMADO";
    END IF;
END
$
```

## Cuidado

```
DROP FUNCTION <nome da função>;
```

## Estrutura

```
DELIMITER $

CREATE TRIGGER <nome do gatilho>
AFTER [INSERT,UPDATE,DELETE] ON <nome da tabela>
FOR EACH ROW
BEGIN
    <comando sql>;
END
$
```

## Nova Tabela

```
CREATE TABLE tb_log_nota_disciplina(
    aluno INT,
    disciplina INT,
    semestre INT,
    ano INT,
    nota_anterior FLOAT,
    nota_atual FLOAT
)
```

## Exemplo

```
DELIMITER $

CREATE TRIGGER tg_historico_nota
AFTER UPDATE ON tb_disciplina_cursada
FOR EACH ROW
BEGIN
INSERT INTO tb_historico (aluno, disciplina, semestre, ano, nota_anterior, nota_atual)
VALUES
(NEW.matricula_aluno, NEW.identificador_disciplina, NEW.semestre, NEW.ano, OLD.nota, NEW.nota);
END
$
```

## Cuidado

```
DROP TRIGGER <nome do gatilho>;
```

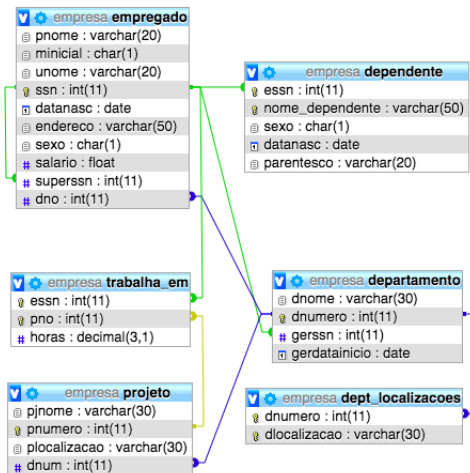


- 1 Introdução
- 2 Abordagem Entidade-relacionamento
- 3 Abordagem Relacional
- 4 Normalização
- 5 Structured Query Language (SQL)
- 6 Consulta SQL**



# Exemplo de um Banco de Dados

## Empresa



## Exemplo

```
SELECT e.pnome AS nome  
FROM empregado AS e
```





# Cláusula WHERE

## Estrutura

```
SELECT <nome do atributo 1>, <nome do atributo 2>  
FROM <nome da tabela>  
WHERE <condição>
```

## Exemplo

```
SELECT e.pnome,  
       e.sexo  
FROM empregado AS e  
WHERE e.ssn = '888665555'
```



- igual: =;
- maior: >;
- menor: <;
- maior igual: >=;
- menor igual: <=;
- diferente: <>;
- entre: between; e
- parte: like (Permitido o uso de máscara com % entre aspas simples).



- e: and;
- ou: or;
- é nulo: is null; e
- não é nulo: is not null.



## Estrutura

```
SELECT <nome do atributo 1>,<nome do atributo 2>  
FROM <nome da tabela>  
WHERE <condição>  
GROUP BY <nome do atributo>
```

## Funções de grupo:

- count();
- min();
- max();
- sum(); ou
- avg().

## Exemplo

```
SELECT e.sexo,  
COUNT (*)  
FROM empregado AS e  
GROUP BY e.sexo
```

# Cláusula HAVING

## Estrutura

```
SELECT <nome do atributo 1>,<nome do atributo 2>  
FROM <nome da tabela>  
WHERE <condição>  
GROUP BY <nome do atributo>  
HAVING < condição>
```

## Exemplo

```
SELECT e.dno,  
AVG (e.salario)  
FROM empregado AS e  
GROUP BY e.dno  
HAVING AVG(e.salario) < 55000
```

## Importante

Predicados da cláusula WHERE são aplicados depois da formação dos grupos.

# Cláusula ORDER BY

## Estrutura

```
SELECT <nome do atributo 1>,<nome do atributo 2>  
FROM <nome da tabela>  
WHERE <condição>  
ORDER BY <nome do atributo><tipo de ordenação>
```

Tipo de ordenação:

- Ascendente: asc; ou
- Descendente: desc.

## Exemplo

```
SELECT *  
FROM empregado AS e  
ORDER BY e.pnome DESC
```



# Junções entre Tabelas

- Junção cruzada;
- Junção interna; e
- Junção externa:
  - Esquerda; e
  - Direita;



## Estrutura

```
SELECT <atributos>  
FROM <primeira tabela> CROSS JOIN <segunda tabela>
```

## Exemplo

```
SELECT e1.pnome,  
       e2.pnome  
FROM empregado AS e1 CROSS JOIN empregado AS e2
```





## Estrutura

```
SELECT <atributo 1>,<atributo 2>  
FROM <primeira tabela> INNER JOIN <segunda tabela> ON (<condição>)
```

## Exemplo

```
SELECT e1.pnome ,  
       e2.pnome  
FROM empregado AS e1 INNER JOIN empregado AS e2 ON (e1.ssn = e2.superssn)
```



## Estrutura

```
SELECT <atributo 1>,<atributo 2>  
FROM <primeira tabela> LEFT OUTER JOIN <segunda tabela> ON (<condição>)
```

## Exemplo

```
SELECT e1.pnome,  
       e2.pnome  
FROM empregado AS e1 LEFT OUTER JOIN empregado AS e2 ON (e1.ssn = e2.superssn)
```



## Estrutura

```
SELECT <atributo 1>,<atributo 2>  
FROM <primeira tabela> RIGHT OUTER JOIN <segunda tabela> ON (<condição>)
```

## Exemplo

```
SELECT e1.pnome,  
       e2.pnome  
FROM empregado AS e1 RIGHT OUTER JOIN empregado AS e2 ON (e1.ssn = e2.superssn)
```



- Na cláusula SELECT;
- Na cláusula FROM; ou
- Na cláusula WHERE.



## Estrutura

```
SELECT <atributo 1>,<atributo 2>,  
(<subconsulta>)  
FROM <tabela>
```

## Exemplo

```
SELECT e.pnome,  
(SELECT COUNT(*) FROM dependente AS d WHERE e.ssn = d. esn) AS qt_dependente  
FROM empregado AS e
```



# Subconsulta na Cláusula From

## Estrutura

```
SELECT <atributo 1>,<atributo 2>  
FROM (<subconsulta>)
```

## Exemplo

```
SELECT e.pnome,  
       d.qt_dependentes  
FROM empregado AS e  
INNER JOIN  
( SELECT d.essn , COUNT(*) AS qt_dependentes  
  FROM dependente AS d  
  GROUP BY d. essn  
) AS d  
ON (e.ssn = d.essn)
```



- in;
- not in;
- exists; e
- not exists.



# Subconsulta na Cláusula Where

## Estrutura

```
SELECT <atributo 1>,<atributo 2>  
FROM <tabela>  
WHERE (subconsulta)
```

## Exemplo

```
SELECT e.ssn,  
       e.pnome  
FROM empregado AS e  
WHERE e.ssn IN (SELECT d.essn  
                FROM dependente AS d  
                GROUP BY d.essn)
```





## Exemplo

```
SELECT e.ssn,  
       e.pnome  
FROM empregado AS e  
WHERE EXISTS (SELECT d.ssn  
                FROM dependente AS d  
                WHERE d.ssn = e.ssn  
                GROUP BY d.ssn)
```

