FIEP SESI SENAI IEL

SENAI

Banco de Dados - PostgreSQL

Julio Antunes

Formação Acadêmica

- Graduação - Tecnologia na Informação - VIZIVALI;
- Pós graduação - MBA em Gerenciamento de Projeto – FGV;

Histórico instrutor

- Lógica, Programação e DB – 2007/2008 – Águia Informática;
- Lógica, Programação e DB – 2010 – Águia Informática;
- Programação Powerbuilder – Aprender e Crescer – 2010 – Sudotec;
- Lógica, Programação e DB – 2011 – Visão do Futuro;

Julio Antunes

Histórico profissional - TI

- Técnico de informática – 2003 a 2005 – Mega informática;
- Programador de computadores – 2005 a 2008 – CISS;
- Analista de sistemas – 2008 a 2009 – HSBC;
- Arquiteto de banco de dados – 2010 a 2013 – CISS;
- Arquiteto de solução – 2014 a 2016 – CISS;
- Fundador e CEO – 2014 – SOS Serviço;
- Sócio e Diretor Técnico – 2015 – ServoFiel tecnologia;

Metodologia de trabalho

Aulas

- Teóricas;
- Práticas;
- Trabalhos em sala de aula;
- Exercícios;
- Pesquisas extra sala de aulas;

Avaliações

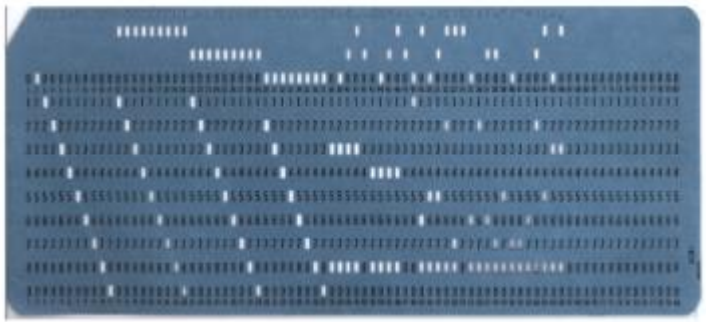
- 2 Avaliações;
- 30/08 – Descritiva com temas abordados até a aula anterior;
- 24/10 – Descritiva e prática como todos os temas e a entrega de um modelo de banco de dados;

Vamos combinar?

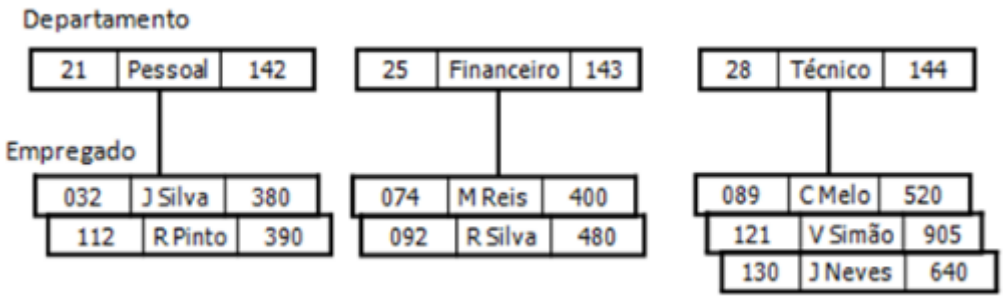
Introdução

História do banco de dados

- 1950 – Armazenamento de dados em cartões perfurados;



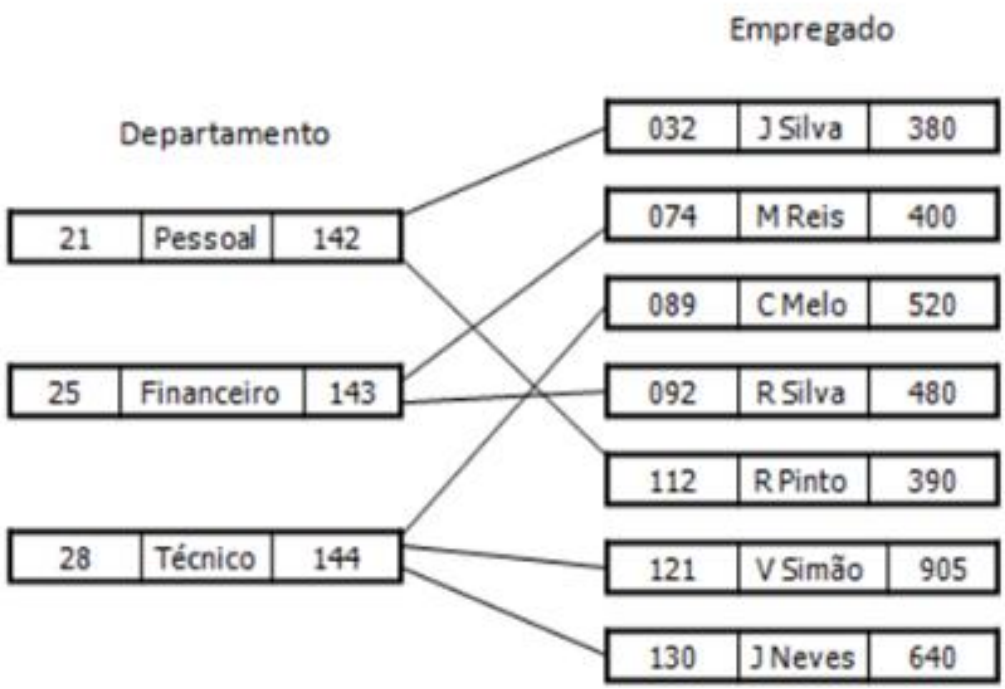
- 1960 – Armazenamento de dados computacionais estruturado (hierarquia);



Introdução

História do banco de dados

- 1970 – Banco de dados em rede (coleção de registros);



Introdução

História do banco de dados

- 1970 – Banco de dados relacional (tabelas interligadas);

Empregado

NumEmp	NomeEmp	Salário	Dept
032	J Silva	380	21
074	M Reis	400	25
089	C Melo	520	28
092	R Silva	480	25
112	R Pinto	390	21
121	V Simão	905	28
130	J Neves	640	28

Departamento

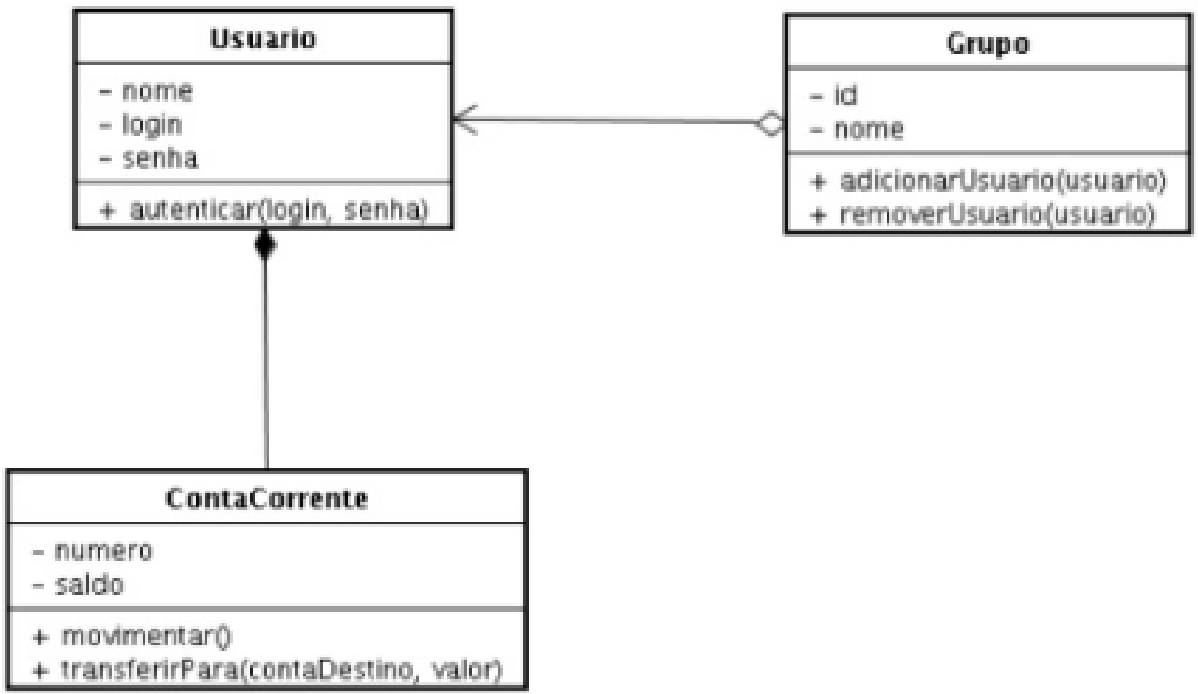
NumDept	NomeDept	Ramal
21	Pessoal	142
25	Financeiro	143
28	Técnico	144

- DB2;
- Oracle;
- SQL Server;
- MySQL;
- PostgreSQL;
- Firebird;
- ASA;

Introdução

História do banco de dados

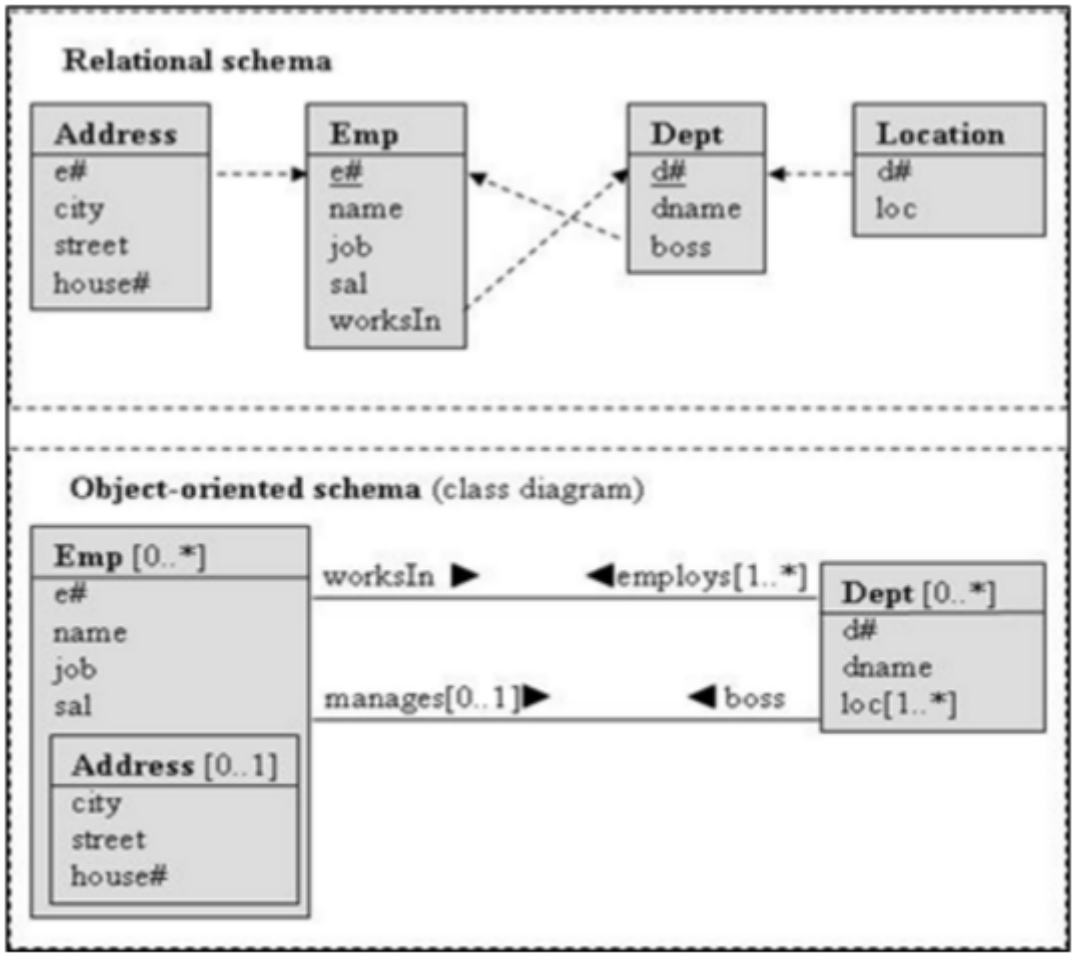
- 1980 – Banco de dados orientado a objeto (interação melhor com linguagens de programação);



Introdução

História do banco de dados

- 1990 – Banco de dados objeto relacional (explosão da web 24X7);
- PostgreSQL;



Introdução

História do banco de dados

- 2000 – Banco de dados avançados (mobilidade, troca instantânea de mensagens, armazenamento de arquivos, grandes volume de dados);
- SQLite;
- MongoDB;
- NOSQL;
- Rediz;
- Cassandra;

Banco de Dados

O que é dado?

- “Dado é a menor informação fornecida ou processada por um computador”;
- Dados é um conjunto de informações;

Exemplo

- Idade = 20 anos;
- Nome = Pedro;

Dados + análise

- Pedro é maior de idade;

Banco de Dados

O que é banco de dados?

- “É uma coleção de dados relacionados, organizados e armazenados visando facilitar a manipulação, permitindo realizar alterações, inserções, remoções e consultas”;
- Coleção ou base de dados = qualquer substantivo concreto ou abstrato do mundo real;

Pedro Silva
20 anos
Dois Vizinhos

Maria das Flores
15 anos
Salto do Lontra

Matemática
Português
História

Banco de Dados

Qual sua importância?

- Os dados são fundamentais para nortear os planos de ação de uma empresa (uma política de promoções personalizada, uma analise de crédito, uma analise de compras, etc...);
- Todos os dados geram informações e conhecimento (perfil de clientes, quanto de estoque tenho do produto, quantas dias do ano choveu granizo);
- *Manter os dados íntegros e com qualidades a qualquer preços!*

Banco de Dados

Apresentação dos dados

- A apresentação dos dados em modelo relacional é semelhante a uma planilha eletrônica;
- Gerenciável;
- Integridade;
- Segurança;
- Atomicidade;
- Compartilhada;

	A	B	C	D	E	F
1	Planilha de funcionários					
2						
3	**Código**	**Nome**	**Cargo**	**Setor**	**Idade**	**Salário**
4	1	José Maria da Silva	Faxineiro	Limpeza	41	R$ 1.200,00
5	2	Maria José da Silva	Auxiliar	Cobrança	34	R$ 850,00
6	3	Pedro de Sá	Vigilante	Segurança	48	R$ 1.500,00
7	4	Alexandre Borges	Consultor	RH	35	R$ 1.600,00
8	5	Cintia Oliveira	Auxiliar	Contabilidade	25	R$ 700,00
9	6	Ana Carolina Souza	Auxiliar	Contabilidade	26	R$ 700,00
10	7	Marcos Smith	Programador	Informática	29	R$ 1.800,00
11	8	Adolfo Pinheiro	Consultor	RH	45	R$ 1.600,00
12	9	José do Rego	Gerente	Administração	54	R$ 2.000,00

SGBD

Sistema Gerenciador de Banco de Dados

- É um o conjunto de programas de computador responsáveis pelo gerenciamento de uma ou mais base de dados;
- Gerenciar acesso;
- Manipular os dados;
- Organizar os dados;
- Fornecer os dados;

Projetar um banco de dados

Como elaborar um projeto de banco de dados?

- Analise dos requisitos (minimundo);
- Projeto conceitual (visão macro do banco de dados);
- Projeto lógico (modelos internos, nomes de tabelas, regras, tipos, nomes, tamanhos, etc..);
- Projeto físico (os scripts para a criação dos objetos no banco de dados) - SGDB;

Minimundo

Analise dos requisitos - Entidade

Identificar as caracteristicas e comportamentos das entidades.

- Caracteristicas (atributos);
  - Nome, Idade, Cor,
  - Peso, Sexo, Autor,
  - Editora, Folhas, Endereço;
- Comportamento (ações);
  - Um pessoa pode ter um ou mais cachorros;
  - Uma pessoa pode ler um ou mais livros;

Projeto conceitual

Entidade

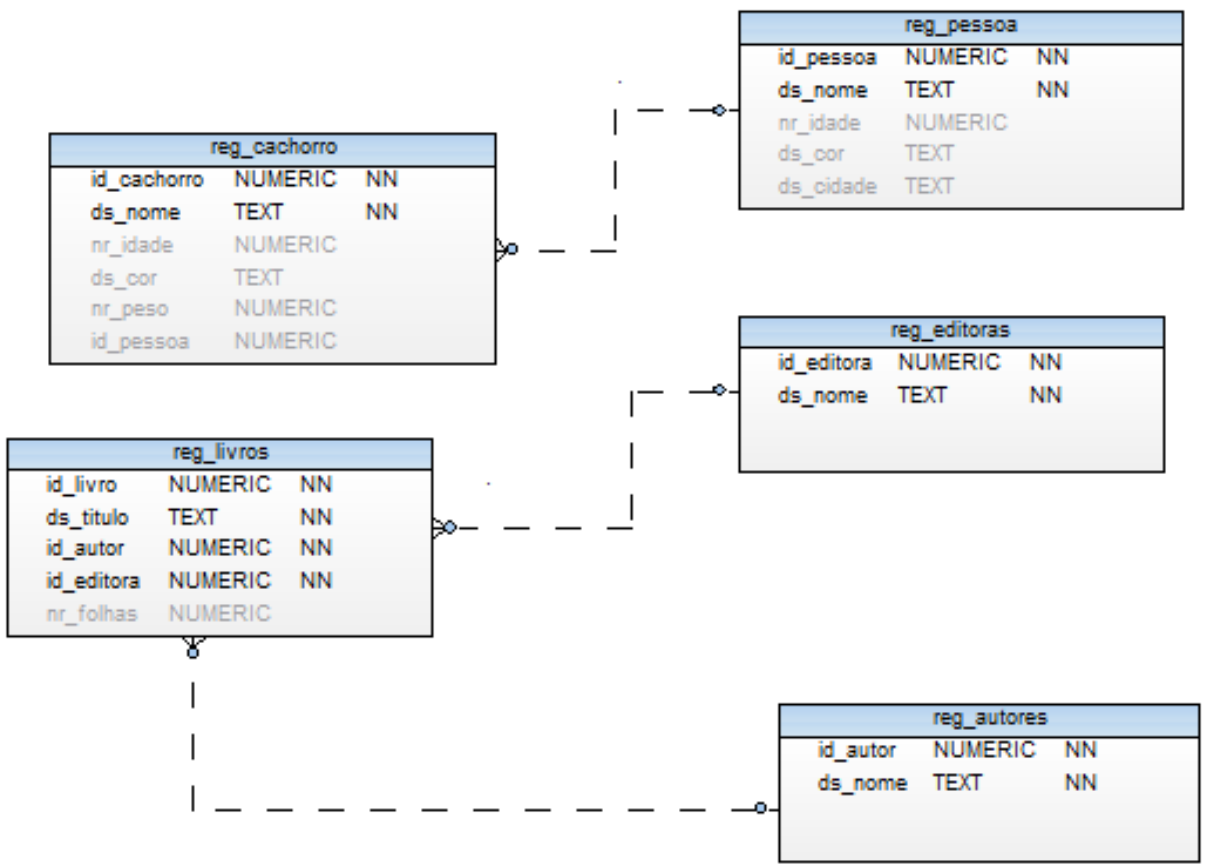
- Caracteristicas (atributos);

CACHORRO	PESSOA	LIVROS
Scoob	Pedro da Silva	Usando a Cabeça
2 Anos	43 anos	Luiz Gonzaga
Marrom	Branco	Abril
5 KG	85 KG	40 Folhas
Macho	Masculino	
	Dois Vizinhos	

Projeto lógico

Tabelas

- Tabelas, colunas, tipo, tamanho, relacionamento, abstração, normalização;



Projeto físico

SQL

Structured Query Language, ou Linguagem de Consulta Estruturada

```
CREATE TABLE reg_cachorro
(
  id_cachorro integer NOT NULL,
  ds_nome character varying(40) NOT NULL,
  nr_idade integer,
  ds_cor character varying(40),
  nr_peso numeric(7,3)
)
```

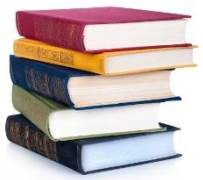
```
CREATE TABLE reg_cachorro (
  id_cachorro INT NOT NULL,
  ds_nome VARCHAR(40) NOT NULL,
  nr_idade INT NULL,
  ds_cor VARCHAR(40) NULL,
  nr_peso DECIMAL(7,3) NULL )
```

Exercício

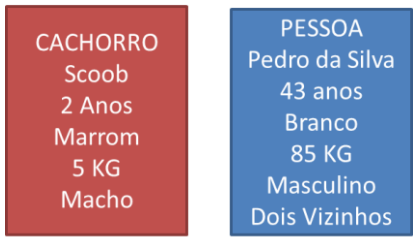
Projetar banco de dados

Projeto para uma locadora de filmes.

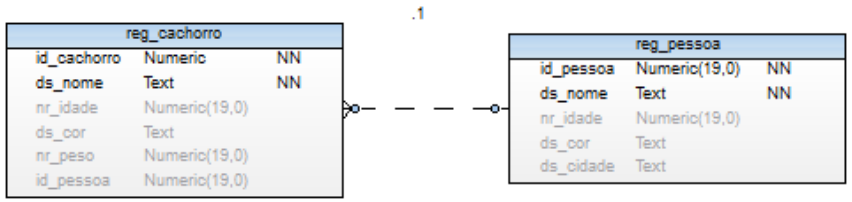
- Analise de requisitos (minimundo);



- Projeto conceitual;



- Projeto lógico;



Tabelas (Relação ou Entidade)

Uma tabela é uma simples estrutura de linhas e colunas.

- **Colunas ou atributos**
  - São as características que se deseja conhecer sobre os objetos que compõem a realidade (entidades);
  - Uma coluna é composta por um tipo de dados;
- **Linha ou tuplas**
  - Uma lista ordenada de colunas representa um registro;
  - Os registros não precisam conter informações em todas as colunas;
  - Um registro é uma instância de uma tabela, ou entidade.

Nome	CPF	Telefone	Email	Dt. Nascimento
Pedro da Silva	12345678912	4699110099	pedro@gmail.com	18/02/2001
Maria		4588229987	maria@hotmail.com	31/12/1995

Integridade da informação

- Duplicação de dados é reduzida;
- Políticas de atualização podem ser padronizadas;
- Consistência de dados;
- Melhor qualidade da informação;

Tipos de integridade

- Integridade de Domínio (Colunas);
- Integridade de Entidade (Tabelas);
- Integridade Referencial (Relacionamento);

Integridade da informação

- **Integridade de Domínio => Colunas**
  - Tipo de dado;
  - Permite nulo;
  - Valor default;
  - Intervalo de valores;
- **Integridade de Entidade => Tabelas**
  - Um nível acima;
  - Chave primária (Primary Key);
  - Índice único;
- **Integridade Referencial => Relacionamento**
  - Ligação de uma tabela com outra através de colunas (Foreign Key);
  - A coluna a ser ligada deve ser PK na outra tabela;
  - Garantir que um registro filho sempre tenha um pai;

Integridade da informação

Relacionamentos

É uma associação entre entidades distintas.

Ligação de um campo da tabela X com um campo da tabela Y.

- **Um para um (1 para 1)**
  - Relações unívoca entre si;
  - Pessoa -> Funcionário;
- **Um para muitos (1 para N)**
  - Chave primária da tabela 1 vai como FK para a tabela N;
  - Pessoa -> Contatos;
- **Muitos para muitos (N para N)**
  - Necessário criar uma nova tabela associativa;
  - Leva a PK das tabelas principais para a tabela associativa;

Tipos de dados

- **Inteiros**
  - SMALLINT (-32768 a 32767)
  - INTEGER ou INT (-2147483648 a -2147483647)
  - BIGINT (-9223372036854775808 a 9223372036854775807)
- **Decimais**
  - DEC, DECIMAL ou NUMERIC (65 dígitos);
  - DECIMAL (6,2) -> 999.99
- **Data e Hora**
  - DATE (0000-00-00) -> yyyy-mm-dd
  - TIME (00:00:00) -> hh:mm:ss
  - DATETIME ou TIMESTAMP (0000-00-00 00:00:00) -> yyyy-mm-dd hh:mm:ss

Tipos de dados

- **Caracteres, string, texto**
  - CHAR (tamanho) -> CHAR(10) -> `'AB' -> 'AB        '` (10 bytes)
  - VARCHAR (tamanho) -> VARCHAR(10) -> `'AB' -> 'AB'` (2 bytes)
- **Binário ou arquivos**
  - BLOB

Constraints

- **DEFAULT**

  Atribuir um valor padrão para uma coluna sempre que uma nova linha for inserida na tabela.

- **NOT NUL**

  Indica que o conteúdo de uma coluna não pode ser nulo.

- **UNIQUE**

  Indica que não pode haver repetição do valor da coluna na tabela.

- **AUTO_INCREMENT**

  Gera um valor incremental automaticamente e único;

  O tipo de dados deve ser numérico, não pode ser nulo e deve ser único;

  Utilizado na inserção de uma nova linha da tabela;

Constraints

- **PRIMARY KEY**

  Chave primária é a coluna ou grupo de colunas que permite identificar como único registro dentro de uma tabela, ela é UNIQUE e NOT NULL.

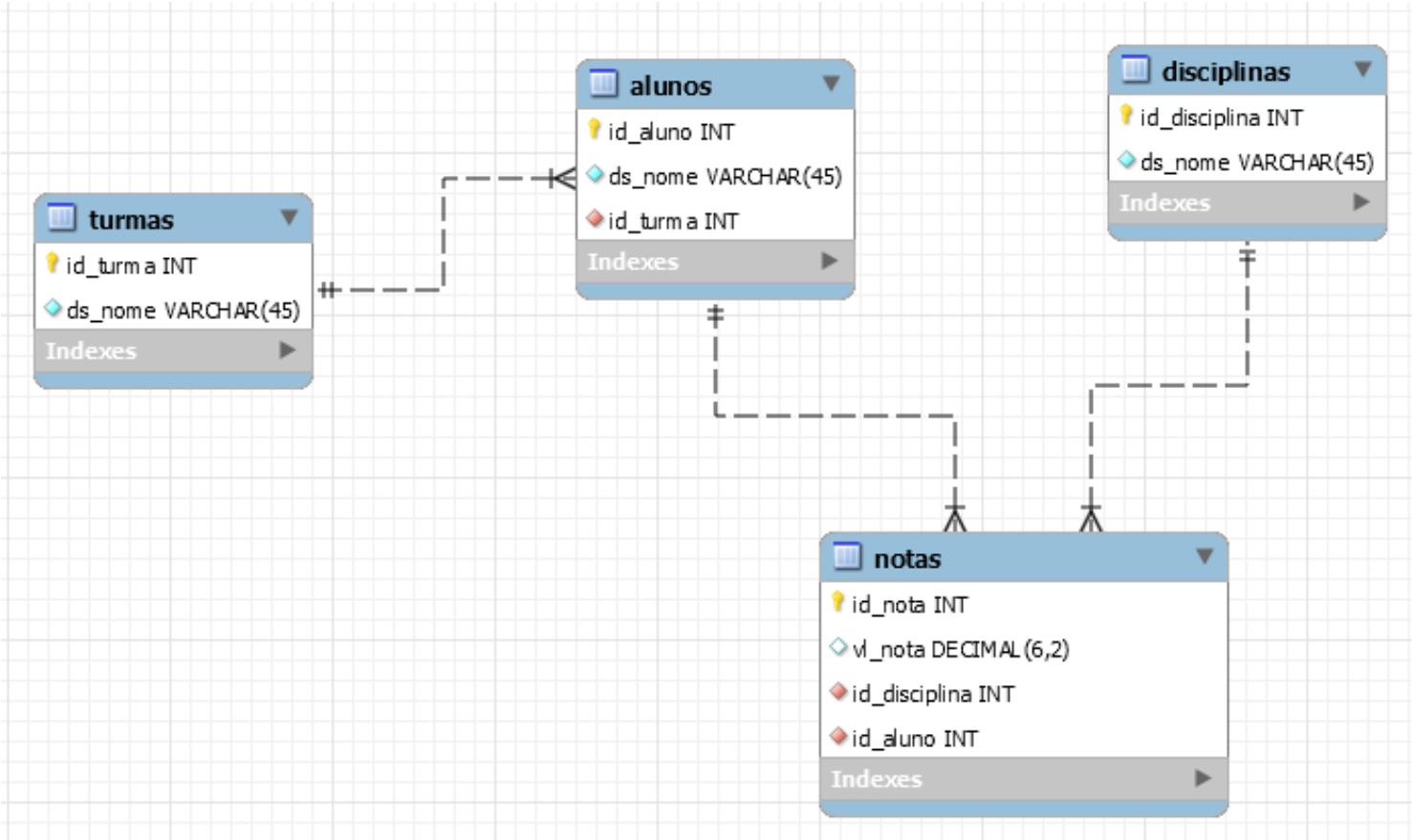
- **FOREIGN KEY**

  Chave estrangeira é a coluna que estabelece o relacionamento entre duas tabelas. Assim a integridade da informação é garantida.

DER

- **Diagrama de Entidade e Relacionamentos**

  Descreve a estrutura lógica de um banco de dados de forma gráfica com alto nível de abstração.



Padrões

- Criar e estabelecer a utilização de padrões de nomenclaturas;
- Todos os DBA envolvidos pensarem e agirem semelhantes;
- Torna a integração de novos membros à equipe com mais facilidade;
  - Nomes de tabelas;
  - Nomes de colunas;
  - Nomes de chave primária;
  - Nomes de chaves estrangerias;
  - Nomes de índices;
  - Nomes de views;
  - Nomes de procedures;
  - Nome de funções;
  - Padrões de SQL;
  - etc...

Ex. Padrões

- Nome de objetos sempre em minúsculos (tabelas, colunas, constraints);
- Usar somente letras;
- Não usar caracteres especiais, acentos e números nos nomes de objetos;
- Separar com underline (_) entre nomes comuns (filmes_generos);
  - **Correto**
  - fimes_generos;
  - dt_nascimento;
  - **Errado**
  - fimes_gêneros;
  - Dt_nascimento;
  - FILMES;
  - 123_tabela;

Prática DER

- **Diagrama de Entidade e Relacionamentos**
  - Instalação da ferramenta Workbench;
  - Overview da ferramenta;
  - Criação do modelo de notas de alunos por disciplina;

DER - Trabalho

- **Diagrama Entidade Relacionamento – Locação de Filmes**
  - Tabelas:
    - produtores , filmes , generos, filmes_generos, tipos_midias, filmes_tipos_midias;
    - estados, cidades, pessoas, clientes, funcionarios;
  - Integridades (de domínio, de entidade e referencial);
  - Tipos de dados (colunas);
  - Relacionamento (1 para 1, 1 para N e N para N);
  - Constraints (PRIMARY KEY e FOREIGN KEY)

  Salvar e enviar para julioantunes.ti@gmail.com

Cachorro	
Nome:	Pluto
Idade:	2 anos
Comprimento dos pêlos:	Curtos
Cor dos pêlos:	Bege
Cor dos olhos	Castanhos
Peso:	5 Kg

Outro objeto “cachorro” apresentaria valores diferentes para os mesmos atributos, por exemplo:

Cachorro	
Nome:	Snoopy
Idade:	4 anos
Comprimento dos pêlos:	Compridos
Cor dos pêlos:	Cinza
Cor dos olhos	Pretos
Peso:	8 Kg

SQL - Linguagem de Consulta Estruturada

- **SQL**
  - É uma linguagem declarativa para banco de dados relacional;
  - É um padrão de acesso e manipulação de um banco de dados;
  - Se divide básicamente em:
    - DDL (Linguagem de definição de dados);
    - DML (Linguagem de manipulação de dados);
    - DCL (Linguagem de controle de dados);
    - DTL (Linguagem de Transação de dados);
    - DQL (Linguagem de consulta de dados);

SQL - DDL

- **Linguagem de Definição de Dados**
  - Uma linguagem para definir a estrutura de um banco de dados;
  - Permite ao utilizador definir tabelas novas e elementos associados seguindo um padrão SQL;
  - Se divide básicamente em:
    - CREATE – Criar um objeto novo no banco de dados;
    - DROP – Remove um objeto do banco de dados;
    - ALTER – Altera a estrutura de um objeto do banco de dados;

SQL - DDL

- **Linguagem de Definição de Dados**
  - CREATE

    Criar tabela no banco de dados

```
CREATE TABLE alunos (
        id_aluno INTEGER NOT NULL,
        ds_nome VARCHAR(80) NOT NULL
);
```

  - DROP

    Remover tabela do banco de dados

```
DROP TABLE alunos;
```

  - ALTER

    Alterar uma tabela no banco de dados

```
ALTER TABLE alunos ADD CONSTRAINT PRIMARY KEY
pk_alunos (id_aluno);
```

SQL - DDL

Distribuições overview

- XAMPP (Apache, MySQL, PHP e Perl)

  https://www.apachefriends.org/pt_br/index.html;
- WAMP (Apache, MySQL e PHP) www.wampserver.com;
- MySQL Comunnity https://www.mysql.com/products/community;
- MySQL Enterprise https://www.mysql.com/products/enterprise;
- MySQL download http://dev.mysql.com/downloads;

SQL - DDL

Começando o banco de dados

- Instalação do PostgreSQL;

  http://www.enterprisedb.com/products-services-training/pgdownload#windows

- Documentação PostgreSQL

  https://www.postgresql.org/docs/9.4/static/index.html)

- Prompt de comando PostgreSQL;
  - Windows + R -> cmd -> cd\ ->

  cd C:\Program Files\PostgreSQL\9.4\bin>

SQL - DDL

Começando o banco de dados

- Conectar ao serviço do PostreSQL;

`psql -U postgres` (`-U` parametro de usuário)

Informar senha para o usuário postgres se solicitar;

- Criar banco de dados dbnotas (ver banco explorer);

```
create database dbnotas;
```

PostgreSQL\9.4\bin>`createdb -U postgres dbnotas`

- Listar banco de dados;

```
\list
```

SQL - DDL

Começando o banco de dados

- Conectar ao banco de dados;

```
\connect dbnotas;
```

- Eliminar bancos de dados

```
drop database dbnotas;
```

PostgreSQL\9.4\bin>`dropdb -U postgres dbnotas`

- Sair e desconectar do banco de dados;

```
\q
```

SQL - DDL

- Criando tabela com colunas NOT NUL com valor DEFAULT e com intervalos de valores;

```
create table alunos (
    id_aluno integer not null,
    ds_nome varchar(80),
    tp_sexo char(1) not null default 'M'
check (tp_sexo = 'M' or tp_sexo = 'F'),
    nr_idade integer
);
```

SQL - DDL

- Ver informações da tabela

```
\d+ alunos;
```

- Alterando tabela removendo coluna;

```
alter table alunos drop column nr_idade;
```

SQL - DDL

- Criando tabela com coluna **auto incremento not null**;

```
create table turmas (
    id_turma serial not null,
    ds_nome varchar(80)
);
```

- Alterar tabela adicionado chave primária para uma coluna;

```
alter table turmas add constraint pk_turmas
primary key (id_turma);
```

SQL - DDL

- Alterar tabela adicionado chave estrangeira;
  - Criar coluna id_turma na tabela de alunos;

```
alter table alunos add column id_turma integer not null;
```

  - Criar FK relacionando as colunas das tabelas

A FK é sempre criada na tabela filha referenciando a tabela pai.

```
alter table alunos add constraint fk_turmas foreign key (id_turma) references turmas (id_turma);
```

DDL - Prática

- Criar banco dbnotas;
- Criar tabela turmas (PK);
- Criar tabela alunos (PK e FK com turmas);
- Criar tabela disciplinas (PK);
- Criar tabela notas (PK, FK com alunos e FK com disciplinas);

SQL - DDL

- Criar backup;

```
pg_dump -U postgres -d dbnotas >
c:\tmp\dbnotas.bkp
```

- Restaurar backup;

```
psql -U postgres -d dbnotas <
c:\tmp\dbnotas.bkp
```

Se o banco de dados não exisitir é necessário criar para depois restaurar o backup

DDL – Locadora - Trabalho

SQL - DDL

- Criar banco dblocadora
- Criar as tabela do modelo de locação de filmes com as integrides de **dominios**, **entidades** e **referencial**.
- Modelo em https://goo.gl/3Sfrli pasta Modelos arquivo locadora.xml;
  - produtores
  - generos
  - filmes
  - filmes_generos,
  - pessoas;
  - locacoes;
  - locacoes_filmes;

Seguir os padrões de nomes:
- Tudo minúsculo;
- Somente letras;
- Não usar espaços;
- Separar palavras com underline;
- Analisar as obrigatoriedades dos campos;

SQL - DML

- **Linguagem de Manipulação de Dados**
  - Uma linguagem usada para inserir, atualizar e eliminar dados;
  - Também segue o padrão SQL;
  - Se divide em:
    - INSERT – Inserir um ou mais registro novo em uma tabela;
    - DELETE – Remove um ou mais registro de uma tabela;
    - UPDATE – Atualizar dados de um ou mais registro de uma tabela;

SQL - DQL

- **Linguagem de Consulta de Dados**
  - Uma linguagem usada para buscar informações em uma ou mais tabelas no banco de dadps;
  - Também segue o padrão SQL;
  - Se divide em:
    - SELECT – Especificar uma consulta com a descrição do resultado resultado;

SQL - DML

- **INSERT**
  - Inserir um ou mais registro novo em uma tabela;
  - Os valores devem ser condizentes ao tipo da coluna;
  - Os valores de colunas do tipo CHAR, VARCHAR, DATE, TIME e DATETIME devem ser inseridos entre aspas simples;

Sintaxe:

```
INSERT INTO nome_da_tabela (nome_coluna_1, nome_coluna_2)
VALUES (valor_coluna_1, valor_coluna_2);
```

Exemplo:

```
INSERT INTO turmas (id_turma, ds_nome) VALUES (1, 'Aprender
e Crescer 2016 Tarde');
INSERT INTO turmas (id_turma, ds_nome) VALUES (2, 'Aprender
e Crescer 2016 Noite');
```

SQL – DML - DQL

- **Clausulas**

  São condições de modificações utilizadas para definir os dados que deseja selecionar, modificar ou eliminar de uma tabela de dados;

  - **FROM** - Utilizada para especificar a tabela que se vai utilizar para atualizar, eliminar ou buscar dados.

```
SELECT FROM nome_da_tabela;
```

Exemplo

```
SELECT FROM turmas;
```

SQL – DML - DQL

- **Clausulas**
  - **WHERE** - Utilizada para especificar as condições que devem reunir os registros que serão selecionados para atualizar, eliminar ou consultar..

```
SELECT FROM nome_da_tabela WHERE nome_coluna =
valor_coluna;
```

Exemplo

```
SELECT FROM turmas WHERE id_turma = 1;
```

SQL - DQL

- **SELECT**
  - Uma linguagem usada para buscar informações em uma ou mais tabelas no banco de dados com a descrição do resultado com consulta simples e complexas;

Sintaxe:

```
SELECT * FROM nome_da_tabela;
```

Exemplo:

```
SELECT * FROM turmas;
SELECT id_aluno, ds_nome FROM alunos WHERE
id_turma = 1;
```

SQL - DML

- **DELETE**
  - Remover um ou mais registro em uma tabela;
  - Remove sempre a linha inteira da tabela;

  Sintaxe:

```
DELETE FROM nome_da_tabela;
```

  Exemplo:

```
DELETE FROM turmas;
DELETE FROM alunos WHERE id_turma = 1;
```

SQL - DML

- **UPDATE**
  - Atualiza colunas de um ou mais registro em uma tabela;
  - Os valores devem ser condizentes ao tipo da coluna;
  - Os valores de colunas do tipo CHAR, VARCHAR, DATE, TIME e DATETIME devem ser inseridos entre aspas simples;

Sintaxe:

```
UPDATE nome_da_tabela SET nome_da_coluna =
valor_da_coluna;
```

Exemplo:

```
UPDATE turmas SET ds_nome = 'Turma 2 B';
UPDATE alunos SET ds_nome = 'Pedro';
UPDATE alunos SET id_turma = 1 WHERE id_aluno = 1;
```

DML - Prática

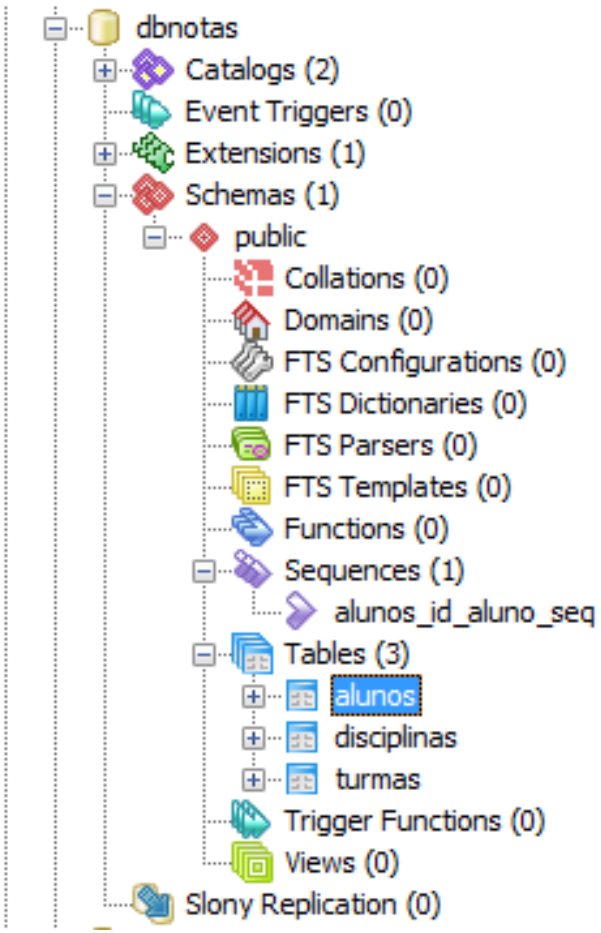
- Inserir as turmas
  - ✓ Aprender e Crescer;
  - ✓ Técnico em Informática
- Inserir alunos
  - ✓ Pedro e Maria para a turma Aprender e Cresecer;
  - ✓ José e Paulo para a turma Técnico em Informática;
- Inserir as disciplinas
  - ✓ Banco de Dados;
  - ✓ Programação Java;
  - ✓ Hardware;

DML - Prática

- Inserir notas
  - ✓ Nota 7.0 para Pedro na disciplina Banco de dados;
  - ✓ Nota 8.5 para Pedro na disciplina Programação Java;
  - ✓ Nota 6.5 para Maria na disciplina Banco de dados;
  - ✓ Nota 9.0 para Maria na disciplina Programação Java;
  - ✓ Nota 4.0 para José na disciplina Hardware;
  - ✓ Nota 8.5 para Paulo na disciplina Hardware;
  - ✓ Nota 10.0 para Paulo na disciplina Programação Java;
  - ✓ Nota 8.4 para José na disciplina Banco de Dados;

SQL - DDL

- PGAdmin – Administração de banco de dados;



SQL - DDL

- Schemas;

  São coleções de objetos dentro de um determinado banco de dados, que organizam vários aspectos e são importantes para segmentação da segurança, facilitando a administração dos objetos e dos dados.

```
CREATE SCHEMA financeiro;
CREATE SCHEMA estoque;
CREATE SCHEMA fiscal;
CREATE SCHEMA contabil;
```

SQL – Operadores Lógicos

- **AND**

  E lógico. Avalia as condições e devolve um valor verdadeiro caso ambos sejam corretos.

  Sintaxe:

```
DELETE FROM nome_da_tabela WHERE coluna1 = valor1 AND
coluna2 = valor2;
```

  Exemplo:

```
UPDATE notas SET vl_nota = 8.0 WHERE id_aluno = 1 AND
id_disciplina = 1;
SELECT vl_nota FROM notas WHERE id_aluno = 1 AND
id_disciplina = 1;
```

SQL – Operadores Lógicos

- **OR**

  OU lógico. Avalia as condições e devolve um valor verdadeiro se um for correto.

  Sintaxe:

```
DELETE FROM nome_da_tabela WHERE coluna1 = valor1 OR
coluna2 = valor2;
```

  Exemplo:

```
DELETE FROM notas WHERE id_aluno = 1 OR id_disciplina =
1;
SELECT * FROM NOTAS WHERE id_aluno = 1 OR id_disciplina
= 1;
```

SQL – Operadores Lógicos

- **NOT**

  Negação lógica. Devolve o valor contrario da expressão.

  Sintaxe:

```
DELETE FROM nome_da_tabela WHERE NOT coluna1 =
valor1;
```

  Exemplo:

```
DELETE FROM alunos WHERE NOT id_aluno = 1;
SELECT * FROM alunos WHERE NOT id_aluno = 1;
```

DML - Prática

- Buscar dados
  - ✓ Buscar nota de Pedro na disciplina Banco de Dados;
  - ✓ Buscar turma de Maria;
  - ✓ Buscar nome das disciplina 1 e 2;
  - ✓ Buscar notas de Paulo e Jose;
  - ✓ Buscar nome e turma de Pedro e Jose;

DML - Prática

- Atualizar turmas
  - ✓ Atualizar Técnico em Informática para Técnologia da Informação;
- Atualizar alunos
  - ✓ Trocar Pedro e Maria para a turma Tecnologia da Informação;
- Atualizar as disciplinas
  - ✓ Alterar de Banco de Dados para Arquitetura de Banco de Dados;
  - ✓ Alterar Hardware para Desenvolvimento de Software;

DML - Prática

- Alterar notas
  - ✓ Alterar nota de Pedro na disciplina Arquitetura de Banco de Dados para 7.5;
  - ✓ Alterar nota de Paulo na disciplina Desenvolvimento de Software para 6.5;
- Eliminar notas
  - ✓ Eliminar notas de Maria;
  - ✓ Eliminar nota de José na disciplina Desenvolvimento de Software;
  - ✓ Eliminar aluno Pedro (erro de FK);
  - ✓ Eliminar disciplano de Arquitetura de Banco de Dados;

SQL – Operadores Relacionais

<	Menor que
>	Maior que
<>	Diferente de
<=	Menor ou igual que
>=	Maior ou igual que
=	Igual a

Exemplo:

```
UPDATE notas SET vl_nota = 8.0 WHERE id_aluno <= 2;

SELECT * FROM alunos WHERE id_aluno <> 1 AND id_turma
<> 1;

DELETE FROM notas WHERE vl_nota >= 8;
```

SQL – Operadores Relacionais

- **BETWEEM**

  Utilizado para especificar um intervalo de valores.

  Exemplo:

```
SELECT vl_nota, id_aluno FROM notas WHERE
id_disciplina BETWEEN 1 AND 2;
SELECT vl_nota, id_aluno FROM notas WHERE
dt_nascimento BETWEEN '1990-01-01' AND '1999-12-31';
```

SQL – Operadores Relacionais

- **LIKE**

  Utilizado na comparação de dados através de uma expressão de %

  para buscar um valor textual;

  Exemplo:

  Buscar apenas nomes que começam com a letra P;

```
SELECT * FROM alunos WHERE ds_nome LIKE 'P%';
```

  Buscar apenas nomes que terminam com a letra o;

```
SELECT * FROM alunos WHERE ds_nome LIKE '%o';
```

  Buscar em qualquer parte pela letra a;

```
SELECT * FROM alunos WHERE ds_nome LIKE '%a%';
```

SQL – Operadores Relacionais

- **IN**

  Utilizado para determinar vários valores a ser comparados a uma coluna.

  Exemplo:

```
SELECT * FROM alunos WHERE id_aluno IN (1,3);
DELETE FROM notas WHERE id_aluno IN (1,3);
SELECT * FROM alunos WHERE ds_nome IN
('Pedro','Maria');
SELECT * FROM notas WHERE vl_nota IN (9,7);
```

SQL – Funções de Agregações

- **AVG**

  Utilizado para calcular a média de um determinado campo de valor;

```
SELECT AVG(vl_nota) FROM notas;
```

- **COUNT**

  Utilizado para devolver a quantidade de registro da seleção;

```
SELECT COUNT(*) FROM notas;
```

```
SELECT COUNT(vl_nota) FROM notas;
```

SQL – Funções de Agregações

- **SUM**

  Utilizado para devolver a soma de todos os valores de uma determinada coluna;

```
SELECT SUM(vl_nota) FROM notas;
```

- **MAX**

  Utilizado para devolver o valor mais alto de uma determinada coluna;

```
SELECT MAX(vl_nota) FROM notas;
```

- **MIN**

  Utilizado para devolver o valor mais baixo de uma determinada coluna;

```
SELECT MIN(vl_nota) FROM notas;
```

DML - Prática

- ✓ Cadastrar mais 1 turma (Tec. Administração), 1 disciplina (Contabilidade) e 3 alunos para a turma cadastrada;
- ✓ Inserir notas 4, 6.5 e 7 para os 3 alunos na disciplina contabildiade no item anterior;
- ✓ Criar a coluna tp_situacao na tabela notas do tipo texto tamanho 20;
- ✓ Atualizar a coluna tp_situacao da tabela notas para ‘REPROVADO’ para as notas menor que 5;
- ✓ Atualizar a coluna tp_situacao da tabela notas para ‘RECUPERAÇÃO’ para as notas entre 5 e 7;
- ✓ Atualizar a coluna tp_situacao da tabela notas para ‘APROVADO’ para as notas maior ou igual a 7;

DML - Prática

- ✓ Identifique a média das notas para a disciplina Banco de Dados;
- ✓ Identifique a maior nota da disciplina Banco de Dados e Programação em Java;
- ✓ Conte a quantidade de notas para a turma;
- ✓ Identifique todos os alunos que tenham a letra ‘o’ no nome;
- ✓ Some as notas para a disciplina Contabilidade;
- ✓ Identifique a menor nota para as disciplinas Contabilidade e Programação Java;
- ✓ Identifique os alunos reprovados;
- ✓ Identifique os alunos em recuperação e a média de suas notas;
- ✓ Identifique a maior nota dos alunos aprovados;

SQL – Clausulas

- **FROM ->** Especificar a tabela a utilizar;
- **WHERE ->** Especificado para indicar uma condição de filtro;
- **GROUP BY**

  Utilizado para separar os registro em grupos distintos.

```
SELECT sum(vl_nota), id_disciplina FROM notas
GROUP BY id_disciplina;
```

- **HAVING**

  Utilizado para expressar a condição que deve satisfazer cada grupo.

```
SELECT sum(vl_nota), id_disciplina FROM notas
GROUP BY id_disciplina HAVING SUM(vl_nota) > 20;
```

SQL – Clausulas

- **ORDER BY**

  Utilizado para ordenar os registros de formas ascendente e descendente.

```
SELECT * FROM notas ORDER BY vl_nota ASC;
SELECT * FROM notas ORDER BY vl_nota DESC;
SELECT * FROM alunos ORDER BY ds_nome ASC;
```

- **DISTINCT**

  Utilizado para selecionar dados sem repetição.

```
SELECT DISTINCT vl_nota, id_disciplina FROM notas;
```

DML - Prática

- ✓ Some as notas agrupadas por disciplinas;
- ✓ Some as notas agrupadas por disciplinas e traga somente as com soma maior que 30;
- ✓ Liste os alunos ordenados pelo nome em ordem alfabética;
- ✓ Liste as notas ordenadas da maior para a menor;
- ✓ Some as notas agrupadas por aluno;
- ✓ Liste as notas ordenadas pelo aluno do menor para o menor e pela nota do maior para o menor;
- ✓ Some as notas agrupadas por disciplina e por aluno e traga somente notas maior que 7 ordendas da maior para a menor;

SQL - DTL

- **DTL – Linguagem de Transação de Dados**
  - Terminar qualquer transação aberta;
  - Uitilizado para confirmar ou desfazer dados em varias tabelas;

    \set AUTOCOMMIT off
  - COMMIT - > Confirmar a transação;
  - ROLLBACK -> Desfazer a transação;

Data View PGAdmin

DTL - Prática

- ✓ Cadastrar turma Medicina;
- ✓ Cadastrar aluno João Pedro;
- ✓ Confirmar com commit;
- ✓ Ver se a turma e Medicina e João Pedro estão cadastrados;
- ✓ Inserir nota 9.5 para João Pedro na disciplina Contabilidade;
- ✓ Executar rollback;
- ✓ Verificar se a nota 9.5 não foi inserida;
- ✓ Atualizar uma nota qualquer;
- ✓ Executar rollback;
- ✓ Verifificar se a nota foi atualizada.

SQL - DCL

- **DCL – Linguagem de Controle de Dados**

  Controla os aspectos de autorização e acesso a dados de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

  - GRANT - > Autoriza o usuário a executar a operação;
  - REVOKE -> Remove ou restringe a capacidade do usuário executar a operação.

SQL - DCL

- **Criar usuário**

```
CREATE USER relatorio WITH PASSWORD '123';
```

- **GRANT**

```
GRANT operação ON objeto nome_obejto TO nome_user;
```

Conceder ao usuário relatorio buscar dados na tabela alunos

```
GRANT SELECT ON TABLE alunos TO relatorio;
```

Conceder ao usuário relatorio buscar e eliminar dados na tabela alunos

```
GRANT SELECT,DELETE ON TABLE alunos TO relatorio;
```

Conceder ao usuário relatorio inserir e atualizar dados na tabela alunos

```
GRANT INSERT, UPDATE ON TABLE alunos TO relatorio;
```

SQL - DCL

- **REVOKE**

```
REVOKE operação ON objeto nome_obejto FROM nome_user;
```

Revogar do usuário relatorio buscar dados na tabela alunos

```
REVOKE SELECT ON TABLE alunos FROM relatorio;
```

Revogar do usuário relatorio buscar e eliminar dados na tabela alunos

```
REVOKE SELECT,DELETE ON TABLE alunos FROM relatorio;
```

Revogar do usuário relatorio inserir e atualizar dados na tabela alunos

```
REVOKE INSERT,UPDATE ON TABLE alunos FROM relatorio;
```

DCL - Prática

- Utilize o usuário postgres para conceder e revogar acesso ao usuário;
- Crie um novo usuário de banco de dados (dba);
- Conceda acesso (GRANT) a select, insert a este usuário para algumas tabelas;
- Conecte ao banco de dados com este novo usuário;
- Tente realizar as operações insert, delete, select e update nas tabelas e analise os resultados das operações;
- Revogue os acesso (REVOKE) concedidos ao usuário dba;
- Conecte ao banco de dados com este novo usuário;
- Tente realizar as operações insert, delete, select e update nas tabelas e analise os resultados das operações;

SQL – DDL - Restrições

- **AUTO_INCREMENT**
- **NOT NUL**
- **DEFAULT**
- **UNIQUE**
- **PRIMARY KEY**
- **FOREIGN KEY**

```
CREATE TABLE alunos
(
  id_aluno serial NOT NULL,
  ds_nome varchar(100) NOT NULL,
  id_turma integer NOT NULL,
  tp_sexo char(1) DEFAULT 'M',
  nr_cpf varchar(11) NOT NULL UNIQUE,
  PRIMARY KEY (id_aluno),
  FOREIGN KEY (id_turma) REFERENCES turmas (id_turma)
)
```

SQL – DDL 2

- **AUTO INCREMENT**

Gerar um sequencial unico cresceste/decrescente;

PostgreSQL é definido como SEQUENCE;

- Ao criar coluna tabela (Tipo serial ou bigserial);

```
CREATE TABLE cad_produto
(
    cod_produto serial NOT NULL,
    nome_produto character varying(100)
)
```

SQL – DDL 2

- **AUTO INCREMENT**
- Criar um objeto SEQUENCE;

```
CREATE SEQUENCE seq_contador_pedido
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 99999999
  START 1;
```

SQL – DDL 2

- **COMO USAR SEQUENCE**

Como valor default em uma coluna

```
ADD COLUMN cod_produto integer;
ALTER COLUMN cod_produto SET NOT NULL;
ALTER COLUMN cod_produto SET DEFAULT nextval('cad_produto_cod_produto_seq'::regclass);
```

Em SELECTs

```
select nextval('seq_contador_pedido')
```

SQL – DDL 2

- **INDICES**

É uma referência associada a uma chave, que é utilizada para fins de otimização, permitindo uma localização mais rápida de um registro quando efetuada uma consulta;

- Indice de otimização;

```
CREATE INDEX ixa_aluno_cpf_nome
  ON alunos (nr_cpf, ds_nome)
```

- Indice único;

```
CREATE UNIQUE INDEX ixu_aluno_cpf
  ON alunos (nr_cpf)
```

SQL – DDL 2

- **CHECK**

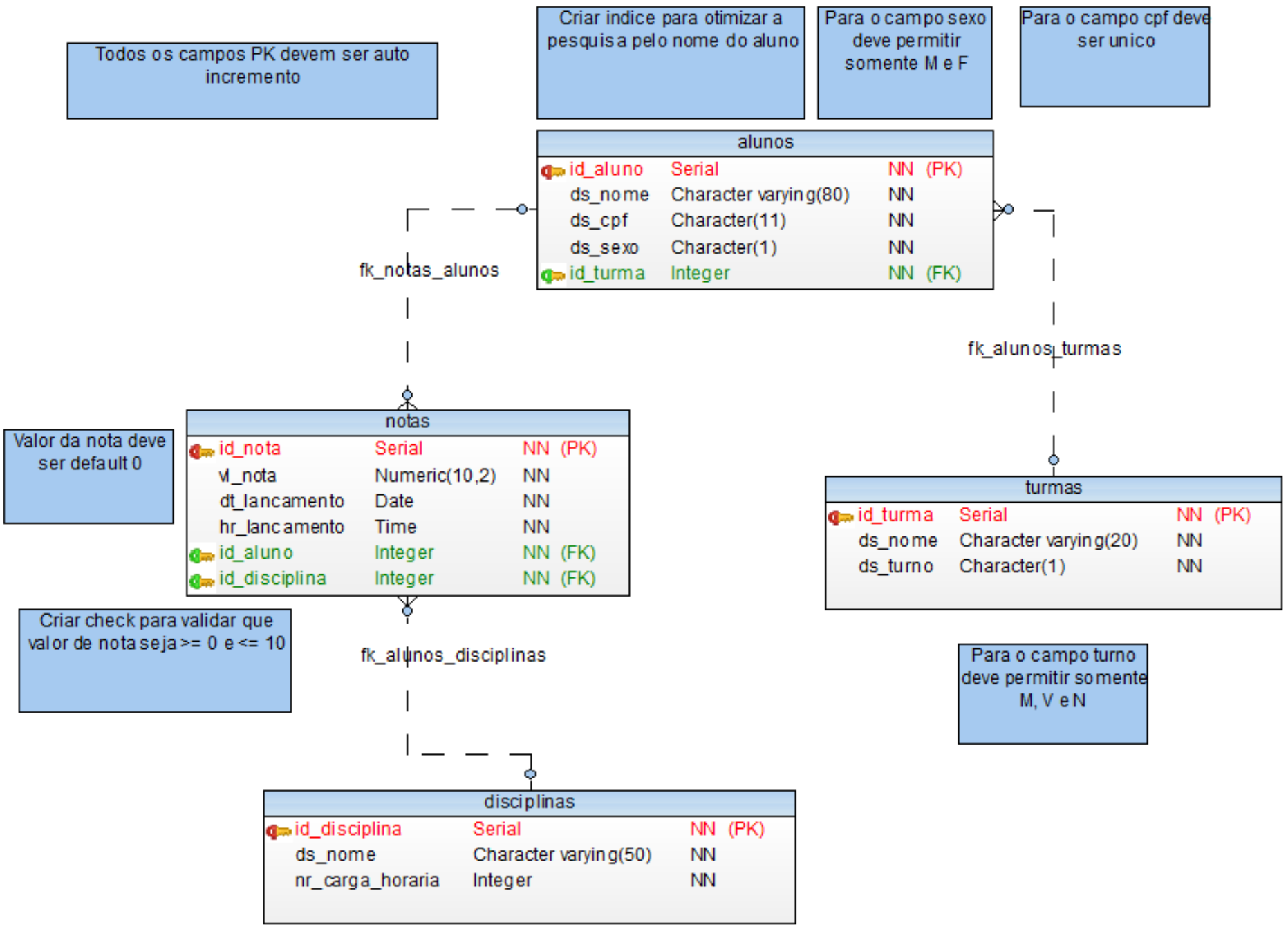
Utilizado para criar uma validação de valores a serem inseridos ou atualizados em uma coluna.

- No check deve ser escrito uma expressão SQL;

```
ALTER TABLE turmas
  ADD CONSTRAINT ckc_turno CHECK (turno = 'N' or turno = 'V')
```

```
ALTER TABLE turmas
  ADD CONSTRAINT ckc_turno CHECK (turno = ANY(ARRAY['V', 'N']))
```

Criar banco de dados **db_notas_alunos**



Popular dados base de dados

- **Cadastrar 3 turmas;**
- **Cadastrar 9 alunos 3 para cada turma;**
- **Cadastrar 3 disciplinas;**
- **Lançar 36 notas, 4 para cada aluno (Usar disciplinas variadas);**
  - Para a data da nota utilizar 2 nota para o dia 14 e 2 para o dia 15

Operadores matemáticos

- **Adição (+)**
```
select 1 + 5
```
- **Subtração (-)**
```
select 10 - 5
```
- **Multiplicação (*)**
```
select 10 * 5
```
- **Divisão (/)**
```
select 10 / 5
```
- **Concatenação (||)**
```
select 'CURSO ' || 123 || ' E ' || ' CRESCER'
```
- **Exponencial (^)**
```
SELECT 7^2
```
- **Fatorial (!)**
```
SELECT 4!
```
- **Mod (Resto da divisão)**
```
SELECT mod(9, 4)
```
- **ROUND (Arredondar valores a partir de casas)**
```
SELECT round(4.567, 2)
```
- **TRUNC (Truncar valores a partir de casas)**
```
SELECT TRUNC(4.567, 2)
```

https://www.postgresql.org/docs/9.5/static/functions-math.html

Funções com string - texto

- **Funções para remover espaços de uma string**
  - TRIM (Remove espaços do inicio e do fim)
  - LTRIM (Remove espaços da esquerda)
  - RTRIM (Remove espaços da direita)
- **Função para localizar por uma palavra em uma string**
  - STRPOS (Retorna a posição que encontra a palavra)
- **Função para substituir uma palavra em uma string**
  - REPLACE (Procura e substitui as palavras pela palavra)
- **Função para obter parte de uma string**
  - SUBSTR (Obtém parte de uma string a partir de uma posição inicial e tamanho)
- **Função para transformar em minúsculo**
  - LOWER(deixa tudo em minúsculo)
- **Função para transformar em maiúsculo**
  - UPPER(deixa tudo em minúsculo)
- **Função para contar o tamanho da string**
  - LENGTH (Retorna inteiro o numero de bytes de uma string)

https://www.postgresql.org/docs/9.5/static/functions-string.html

Funções diversas

- **Obter a data corrente**
```
select current_date
```
- **Obter a hora corrente**
```
select current_time
```
- **Obter a data e hora corrente**
```
select current_timestamp
```
- **Obter data e hora atual**
```
select now()
```
- **CAST (Conversão de tipos)**
```
SELECT CAST('788' AS NUMERIC)
```
- **COALESCE (Trata valor nulo para um valor default)**
```
SELECT COALESCE(null, '0')
```

https://www.postgresql.org/docs/current/static/functions-datetime.html

DML (2)

- **JOIN - WHERE**

Juntar tabelas (obter dados de uma ou mais tabelas através de suas relações).

Tabelas no **FROM** separadas por virgula e relações no **WHERE**, ligando as chaves das tabelas.

```
select   alunos.id_aluno,
         alunos.ds_nome,
         turmas.ds_nome
from     alunos,
         turmas
where alunos.id_turma = turmas.id_turma
```

id_aluno integer	ds_nome character varying(100)	ds_nome character varying(50)
2	Maria	Aprender e Crescer
1	Pedro	Aprender e Crescer
4	Paulo	Técnico em Informática
3	Jose	Técnico em Informática

DML (2)

- **JOIN - ON**

Juntar tabelas (obter dados de uma ou mais tabelas através de suas relações)

Tabelas no **JOIN** ligando as relações no **ON**, ligando as chaves das tabelas.

Traz somente dados que tenham nas 2 tabelas.

```
select   alunos.id_aluno,
         alunos.ds_nome,
         turmas.ds_nome
from     alunos
JOIN turmas ON (alunos.id_turma = turmas.id_turma)
```

id_aluno integer	ds_nome character varying(100)	ds_nome character varying(50)
2	Maria	Aprender e Crescer
1	Pedro	Aprender e Crescer
4	Paulo	Técnico em Informática
3	Jose	Técnico em Informática

DML (2)

- **JOIN – ON - LEFT**

Traz os resultados da tabela da ESQUERDA, tabela do FROM, indiferente de ter linhas na tabela JOIN.

id_aluno integer	ds_nome character varying(100)	ds_nome character varying(50)
2	Maria	Aprender e Crescer
1	Pedro	Aprender e Crescer
4	Paulo	Técnico em Informática
3	Jose	Técnico em Informática
5	Abel	

```
select   alunos.id_aluno,
         alunos.ds_nome,
         turmas.ds_nome
from     alunos
LEFT JOIN turmas ON (alunos.id_turma = turmas.id_turma)
```

DML (2)

- **JOIN – ON - RIGHT**

Traz os resultados da tabela da DIREITA, tabela do JOIN, indiferente de ter linhas na tabela FROM.

```
select   alunos.id_aluno,
         alunos.ds_nome,
         turmas.ds_nome
from     alunos
RIGHT JOIN turmas ON (alunos.id_turma = turmas.id_turma)
```

id_aluno integer	ds_nome character varying(100)	ds_nome character varying(50)
2	Maria	Aprender e Crescer
1	Pedro	Aprender e Crescer
4	Paulo	Técnico em Informática
3	Jose	Técnico em Informática
		Admin

DML (2)

- **Renomear TABELAS e COLUNAS**

Usar a notação AS para renomear coluna ou tabela.

```
select  al.id_aluno,
        al.ds_nome AS nome_aluno,
        al.id_turma,
        tm.ds_nome AS nome_turma
from    alunos AS al
JOIN turmas AS tm ON (tm.id_turma = al.id_turma)
where al.id_turma = 1
```

id_aluno integer	nome_aluno character varying(100)	id_turma integer	nome_turma character varying(50)
1	Pedro	1	Aprender e Crescer
2	Maria	1	Aprender e Crescer

DML (2)

- **SUB SQL**

Sub Consulta no SELECT

```
select   a.id_aluno,
         a.ds_nome,
         (        select sum(vl_nota)
                  from notas n
                  where n.id_aluno = a.id_aluno
         ) as total_nota
from     alunos AS a
```

DML (2)

- **SUB SQL**

Sub Consulta no FROM

```
select   n.vl_nota, n.id_aluno
from     notas AS n
where n.id_aluno in (    select a.id_aluno
                         from alunos a
                         where a.id_turma = 1
                     )
```

DML Pratica

- Criar consulta para trazer o nome do aluno, o nome da disciplina e as suas respectivas notas;
- Uma consulta para somar as notas por disciplinas agrupando por disciplinas, trazer nome da disciplina e nota.
- Criar uma consulta para trazer a soma das notas agrupadas por turma, trazer nome da turma e turno;
- Criar consulta para trazer a média das notas por disciplinas, trazer o nome da disciplina e média da nota;
- Criar consulta para listar as notas dos alunos da turma 1, trazer o nome dos alunos e suas notas;
- Criar consulta para trazer a soma das notas agrupadas por disciplinas arredondadas em 1 casa decimal;

DML Pratica

- Criar consulta para trazer o nome do aluno concatenado a com – e nome da disciplina e as notas de cada aluno;
- Criar consulta para somar as notas das disciplinas agrupadas por disciplinas trazendo nome da disciplina e a soma das notas, na mesma consulta somar a soma das notas da disciplina 1 + a soma das notas da disciplina 2;
- Criar consulta para trazer somente notas de alunos que estão relacionados a uma turma;