APRESENTAÇÃO
DE APOIO

BANCOS DE DADOS NoSQL

Eduardo Henrique Pereira de Arruda – Aula 01

PUCRS online uol edtech

Professores

VINÍCIUS KROTH
Professor Convidado

Desenvolvedor de aplicações SOA nas áreas de contabilidade, financeira e de comércio exterior por mais de 5 anos. Referência em assuntos como projeto e desenvolvimento de SOA e Microservices, Java (EE, Spring framework MVC/WebFlux, jUnit e Gradle), SQL e NoSQL Db's (PostgresSQL, MySQL, MongoDB, Redis e Elasticsearch), AWS Cloud computing, Stress/Chaos testing (Gatlin, Jmeter), ferramentas de Desenvolvimento (Jenkins, Docker e Terraform), Telemetria e Observabilidade (Kibana, Datadog).

EDUARDO HENRIQUE PEREIRA DE ARRUDA
Professor PUCRS

Fundador e CEO da Doc.Space Documentos Digitais. Professor da Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS), onde atua desde 1994 em cursos de graduação, pós-graduação e extensão nas áreas de Ciência da Computação, Engenharia de Software e Sistemas de Informação. Possui graduação (1992) e mestrado (1995) em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (UFRGS) e formações complementares em gestão de TI e Segurança da Informação. Cursa Doutorado no Programa de Pós-Graduação em Ciência da Computação (PPGCC) da PUC-RS. Dedica parte de seu tempo a atividades de incentivo ao empreendedorismo inovador e investe em empresas que adotem modelos de negócio inovadores e escaláveis. Apoia projetos de empreendedorismo social, tendo sido coordenador do projeto Adoções, parceria entre o Poder Judiciário e o Ministério Público do RS com a PUC-RS que, por meio de aplicativo, realiza a aproximação entre candidatos a adoção e crianças e adolescentes em processo de adoção tardia.

Ementa da disciplina

Introdução aos conceitos e características de Big Data como: volume, velocidade, variedade, validade, volatilidade e valência. Introdução aos conceitos de cluster, domínios, agregados, distribuição, tolerância a falhas e sharding. Estudo do Teorema CAP: consistência (Consistency), disponibilidade (Availability), tolerância de partição (Partition). Introdução a Bancos de dados sem esquema prévio, a Banco de dados baseado em documentos, a Banco de dados chave-valor, a Banco de dados colunar e a Banco de dados baseado em grafos.

Bancos de Dados NoSQL

Prof. Eduardo Arruda, MSc.

Pós-graduação em
Desenvolvimento Full Stack

PUCRS | ESCOLA POLITÉCNICA

PROFESSOR
PUCRS

EDUARDO ARRUDA

Coordenador Técnico do Programa Justiça 4.0
(Conselho Nacional de Justiça / PNUD/ONU)

Sócio e CEO da Doc.Space Documentos Digitais

Sócio e CTO da uMov.me

Ex-CIO do Tribunal de Justiça do RS

Professor da Escola Politécnica da PUC-RS

Bacharel e Mestre em Ciência da Computação (UFRGS)

Doutorando em Ciência da Computação (PUC-RS)

Parte I

Conceitos

- Revisão
  - Conceitos de BDs e SGBDs
  - Modelagem conceitual
  - Modelagem lógica relacional
- SGBDs não relacionais

Parte II

Modelagem Não Relacional

- Modelagem conceitual
- Mapeamento para modelo lógico não relacional

Parte III

Implementação

- Utilização prática de SGBDs não relacionais
  - Características
  - Criação de BDs
  - Consultas

Conceitos de BDs e SGBDs

Revisão

Conceitos Básicos

Banco de Dados (BD)

- Coleção armazenada de elementos de dados representando objetos de uma realidade
  - Uma área ou um processo de negócios de uma empresa
  - Exemplos: BD de uma aplicação de vendas on-line, BD acadêmico de uma universidade etc.
- Inúmeros Sistemas de Informação Transacionais (SIT) podem ser desenvolvidos sobre um BD
- Logo, o acesso ao BD é realizado por usuários desses SITs por meio de consultas e atualizações concorrentes

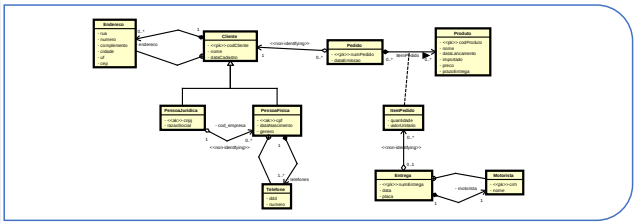
Sistema de Gerência de Bancos de Dados (SGBD)

"*Softwares que incorporam as funções de definição, recuperação e alteração de dados em um banco de dados*"[1]

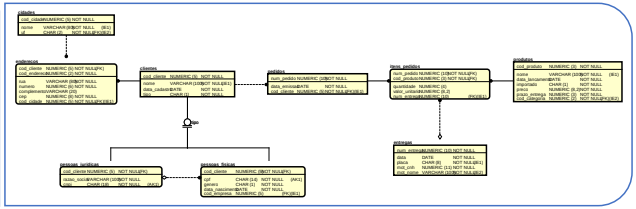
https://db-engines.com/en/ranking

[1]HEUSER, C. A. Projeto de banco de Dados. 6ed. Porto Alegre: Sagra-DC Luzzatto, 2009. 282p.

CONCEITUAL



LÓGICO



FÍSICO

```
create table clientes (
 cod_cliente numeric(5) not null,
 nome varchar(30) not null,
 data_cadastro date not null,
 tipo char(1) not null,
 constraint pk_clientes primary key (cod_cliente),
 constraint chk_cli_tipo check (tipo in ('F','J'))
);

create index idx_cli_nome on clientes (nome);

create table pessoas_juridicas (
 cod_cliente numeric(5) not null,
 cnpj char(14) not null,
 razao_social varchar(100) not null,
 constraint pk_pessoas_juridicas primary key (cod_cliente),
 constraint uk_pj_cnpj unique (cnpj)
);

create table pessoas_fisicas (
 cod_cliente numeric(5) not null,
 cpf char(11) not null,
 data_nascimento date not null,
 genero char(1) not null,
 cod_empresa numeric(5),
 constraint pk_pessoas_fisicas primary key (cod_cliente),
 constraint uk_pf_cpf unique (cpf),
 constraint chk_pf_genero check (genero in ('F','M'))
);

create index ifk_pf_pj on pessoas_fisicas (cod_empresa);

create table estados (
 uf char(2) not null,
 nome varchar(30) not null,
 regiao char(1) not null,
 constraint pk_estados primary key (uf),
 constraint chk_est_regiao check (regiao in ('S','SE','CO','NE','N'))
);

create table cidades (
 cod_cidade numeric(5) not null,
 nome varchar(30) not null,
 uf char(2) not null,
 constraint pk_cidades primary key (cod_cidade)
);

create index ifk_cid_est on cidades (uf);

create table enderecos (
 cod_cliente numeric(5) not null,
 cod_endereco numeric(3) not null,
 rua varchar(30) not null,
 numero numeric(5) not null,
 complemento varchar(10),
 cep numeric(8) not null,
 cod_cidade numeric(5) not null,
 constraint pk_enderecos primary key (cod_cliente,cod_endereco)
);

create index ifk_end_cid on enderecos (cod_cidade);

alter table pessoas_juridicas
add constraint fk_pj_cli foreign key (cod_cliente) references c

alter table pessoas_fisicas
add constraint fk_pf_cli foreign key (cod_cliente) references cli
add constraint fk_pf_pj foreign key (cod_empresa) references pess

alter table cidades
add constraint fk_cid_est foreign key (uf) references estados(

alter table enderecos
add constraint fk_end_cli foreign key (cod_cliente) references cli
add constraint fk_end_cid foreign key (cod_cidade) references cida

alter table telefones
add constraint fk_tel_cli foreign key (cod_cliente) references pess

alter table pedidos
add constraint fk_ped_cli foreign key (cod_cliente) references cli

alter table categorias
add constraint fk_cat_cat foreign key (cod_categoria_pai) referenc

alter table produtos
add constraint fk_prod_cat foreign key (cod_categoria) references

alter table itens_pedidos
add constraint fk_itped_ped foreign key (num_pedido) references pe
alter table itens_pedidos
add constraint fk_itped_prod foreign key (cod_produto) reference
alter table itens_pedidos
add constraint fk_itped_entr foreign key (num_entrega) referenc
```

MODELAGEM DE DADOS

O processo de modelagem de dados de sistemas transacionais tem por objetivos:

1. Modelagem **CONCEITUAL**
   Compreender os requisitos (necessidades) dos futuros usuários do sistema e identificar que dados serão necessários armazenar
2. Modelagem **LÓGICA**
   A partir dos requisitos representados, identificar as melhores estruturas lógicas de dados ideais para implementação
3. Modelagem **FÍSICA**
   Definidas as estruturas lógicas, implementar o banco de dados, criando suas estruturas de dados

MODELOS DE DADOS

Modelos de dados são utilizados para descrever a estrutura de um banco de dados

São empregadas **abstrações** para representar os objetos da realidade que serão modelados, bem como suas características e associações

Modelos **CONCEITUAIS**

- Modelo Entidade-Relacionamento (ER)
- Modelo Orientado a Objetos (OO)

Modelos **LÓGICOS**

- Modelo Relacional
- Modelos Pós ou Não-Relacionais, NoSQL ou *Schemaless*

Modelos **FÍSICOS**

- Modelos de implementação suportados pelos SGBDs

FERRAMENTAS CASE

Ferramentas CASE (*Computer Aided Software Engineering*) são utilizadas para apoiar o desenvolvimento de sistemas

Nas atividades prática desta disciplina utilizaremos a ferramenta de modelagem UML (*Unified Modelling Language*) chamada Astah

Mas podem ser utilizadas outras ferramentas que também suportem a UML

Também é possível utilizar a ferramenta BrModelo, que é gratuita e voltada ao ensino do Modelo Conceitual Entidade-Relacionamento

https://astah.net/

Modelagem conceitual

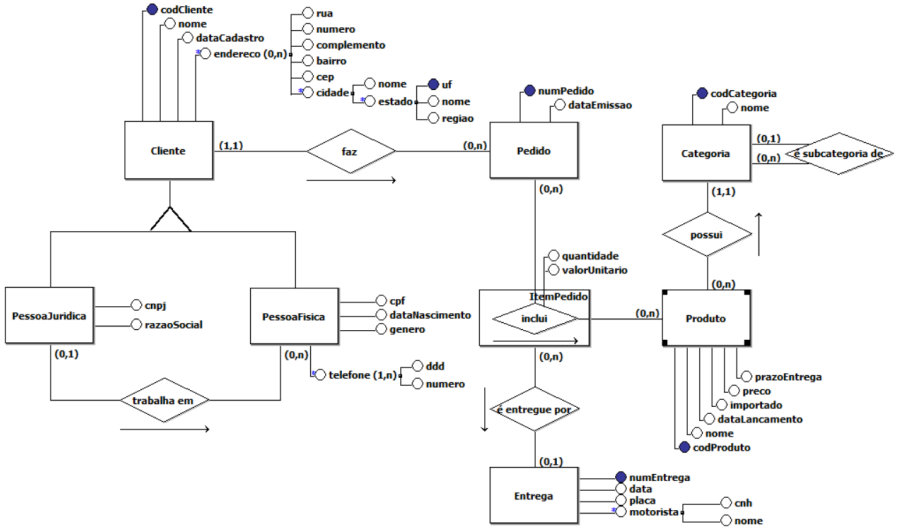
Revisão

MODELO CONCEITUAL

REPRESENTAÇÃO DOS REQUISITOS

O objetivos do **MODELO CONCEITUAL** é permitir a representação dos requisitos a serem atendidos pelo banco de dados de um sistema.

Portanto, não deve haver preocupação excessiva com aspectos de implementação, que poderão ser melhor avaliados na etapa seguinte, de modelagem lógica da estrutura do banco de dados, empregando um MODELO LÓGICO de dados.

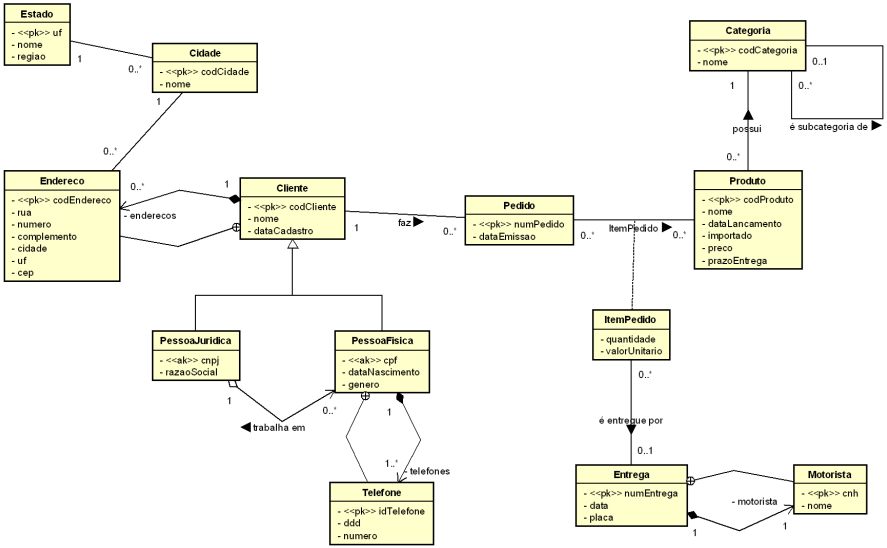


MODELO CONCEITUAL

REPRESENTAÇÃO DOS REQUISITOS

O objetivos do **MODELO CONCEITUAL** é permitir a representação dos requisitos a serem atendidos pelo banco de dados de um sistema.

Portanto, não deve haver preocupação excessiva com aspectos de implementação, que poderão ser melhor avaliados na etapa seguinte, de modelagem lógica da estrutura do banco de dados, empregando um MODELO LÓGICO de dados.



MODELO CONCEITUAL

ENTIDADES/CLASSES E ATRIBUTOS

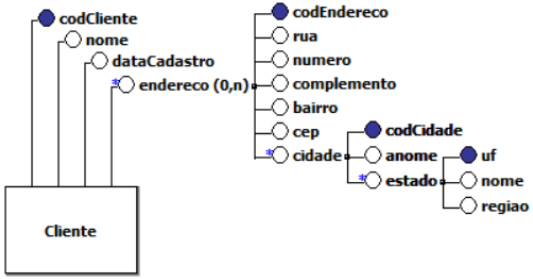
ENTIDADES/CLASSES representam objetos da realidade:
- Pessoas
- Objetos inanimados
- Documentos
- Locais
- Eventos

ATRIBUTOS representam as características das entidades/classes. Eles podem ser:
- Identificadores/não identificadores
- Obrigatórios/opcionais
- Simples/compostos
- Simples/multivalorados

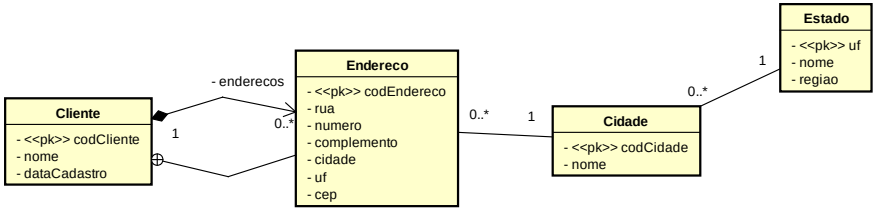
INSTÂNCIA/OBJETO é como se denomina cada ocorrência de uma entidade/classe que representa um único objeto da realidade

Toda entidade/classe deve:
- possuir pelo menos um atributo identificador e um atributo não identificador
- representar pelo menos duas instâncias

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos

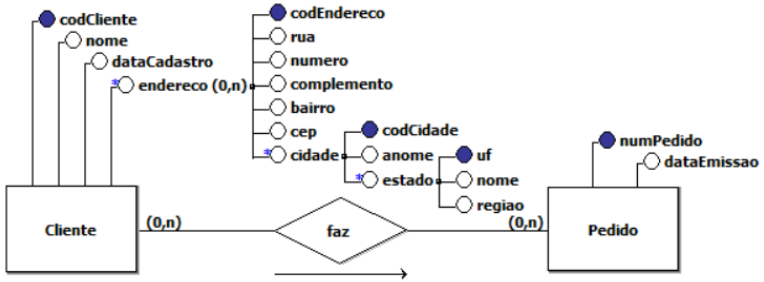


MODELO CONCEITUAL

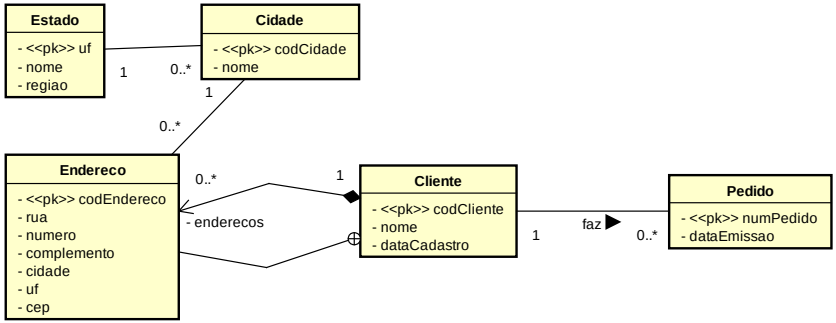
RELACIONAMENTOS OU ASSOCIAÇÕES

RELACIONAMENTOS OU ASSOCIAÇÕES representam os relacionamentos ou associações possíveis entre as instâncias/objetos da realidade

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos



MODELO CONCEITUAL

RELACIONAMENTOS/ ASSOCIAÇÕES 1 PARA N

RELACIONAMENTOS/ASSOCIAÇÕES

1 para N podem ser de dois tipos:

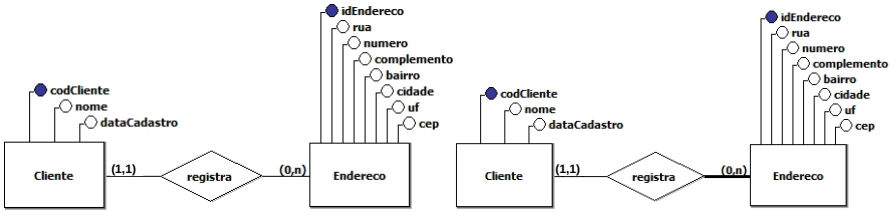
NÃO-IDENTIFICADORES

O identificador da entidade/classe do lado N (filha) é composto somente pelo seu atributo identificador

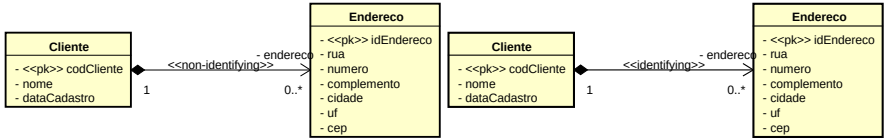
IDENTIFICADORES

O identificador da entidade/classe do lado N (filha) é composto pela combinação do identificador da entidade/classe do lado 1 (pai) e pelo seu atributo identificador

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos



MODELO CONCEITUAL

RELACIONAMENTOS/ ASSOCIAÇÕES 1 PARA N

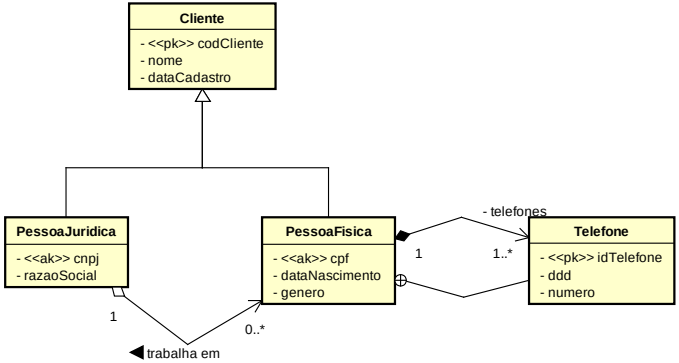
AGREGAÇÃO é uma associação em que as instâncias/objetos da entidade/classe filha (o lado N) podem existir independentemente da instância/objeto pai

COMPOSIÇÃO é uma associação em que as instâncias/objetos da entidade/classe filha (o lado N) não podem existir sem a instância/objeto pai, porque a compõem

Modelo Entidade-Relacionamento

Não trivial

Modelo Orientado a Objetos



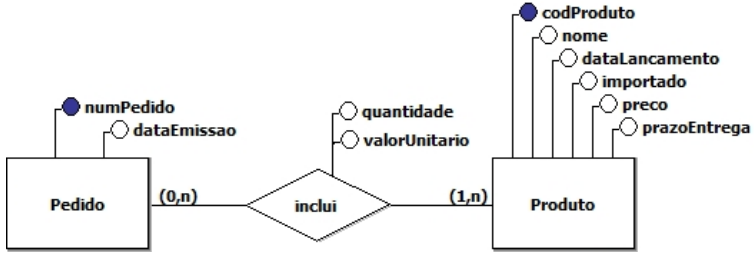
MODELO CONCEITUAL

RELACIONAMENTOS N PARA N

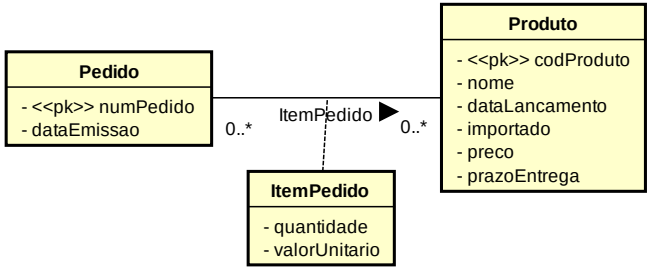
RELACIONAMENTOS N para N indicam que uma instância de uma entidade pode estar relacionada a várias instância da outra entidade vice e versa

É frequente que relacionamentos N para N possuam **ATRIBUTOS DE RELACIONAMENTOS**, para caracterizar a associação entre os objetos

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos

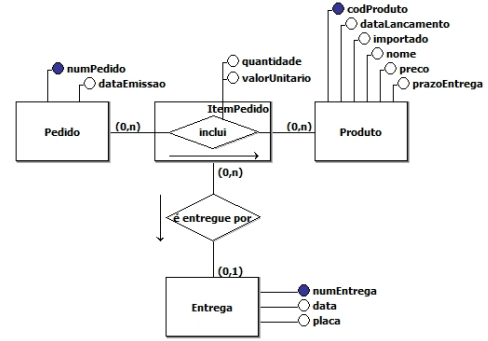


MODELO CONCEITUAL

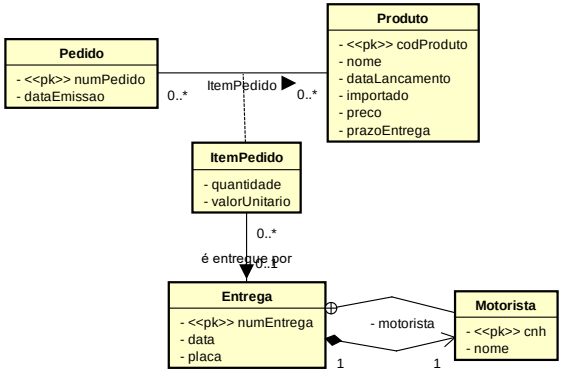
ENTIDADES/CLASSES ASSOCIATIVAS

ENTIDADES/CLASSES ASSOCIATIVAS representam os relacionamentos/associações entre instâncias de um relacionamento, normalmente N para N, com outras entidades/classes

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos

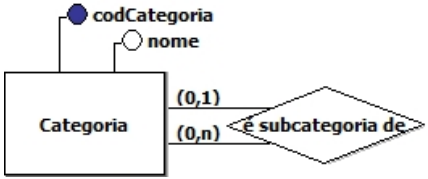


MODELO CONCEITUAL

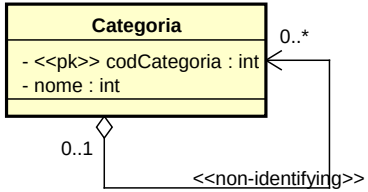
RELACIONAMENTOS UNÁRIOS

RELACIONAMENTOS UNÁRIOS são aqueles que envolvem somente uma entidade/classe

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos

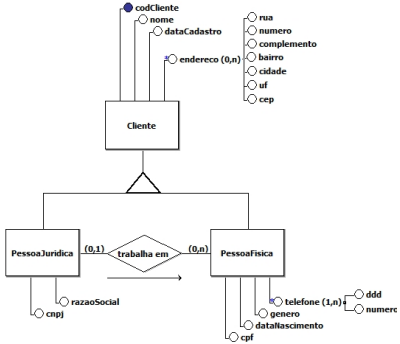


MODELO CONCEITUAL

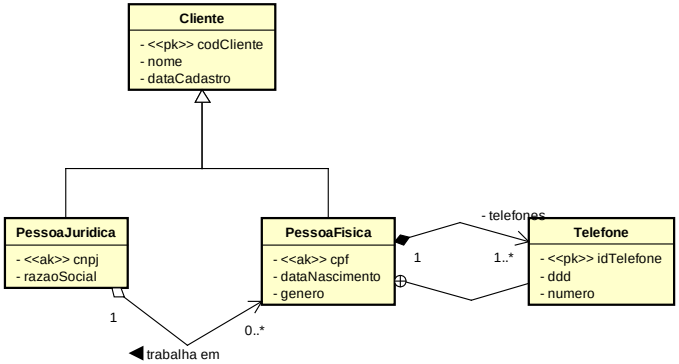
HIERARQUIAS DE ESPECIALIZAÇÃO / GENERALIZAÇÃO

HIERARQUIAS DE ESPECIALIZAÇÃO / GENERALIZAÇÃO permitem reunir em entidades/classes mais genéricas os atributos e os relacionamentos comuns a várias entidades/classes

Modelo Entidade-Relacionamento



Modelo Orientado a Objetos



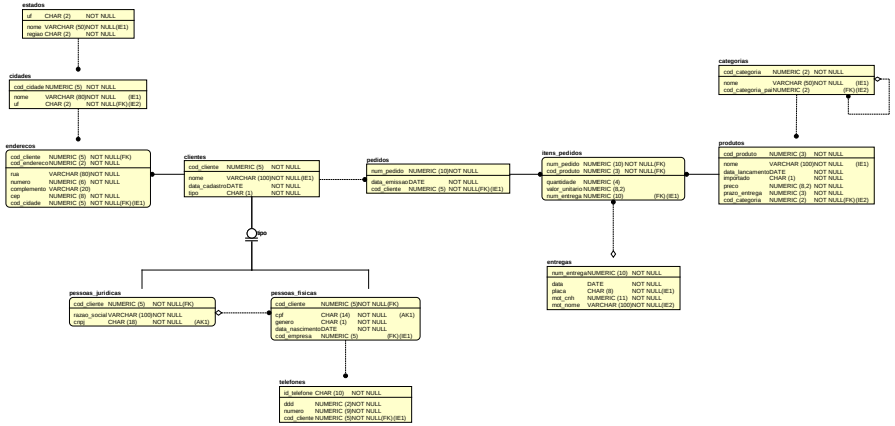
Modelagem lógica relacional

Revisão

MODELO LÓGICO RELACIONAL

REPRESENTAÇÃO DE ESTRUTURAS DE DADOS

Os MODELOS LÓGICOS DE DADOS, como o **MODELO LÓGICO RELACIONAL**, têm por objetivo definir as estruturas de dados que serão utilizadas para implementação dos requisitos representados no ESQUEMA CONCEITUAL, criado utilizando um MODELO CONCEITUAL DE DADOS, como o Modelo Entidade-Relacionamento ou o Modelo Orientado a Objetos.



MODELO LÓGICO RELACIONAL

TABELAS

TABELAS são os elementos básicos para prover persistência a dados em SGBDs relacionais

As TABELAS são organizadas em **COLUNAS**, que correspondem aos ATRIBUTOS de entidades/classes no modelo conceitual.

As TABELAS armazenam os dados em **LINHAS ou REGISTROS**, sendo que cada um deles provê persistência para uma INSTÂNCIA de uma entidade/classe do modelo conceitual.

As tabelas, por serem repositórios de dados, recebem NOMES NO PLURAL, no padrão *snake_case*.

Esquema Lógico Relacional

clientes

cod_cliente	NUMERIC (5)	NOT NULL
nome	VARCHAR (100)	NOT NULL (IE1)
data_cadastro	DATE	NOT NULL
tipo	CHAR (1)	NOT NULL

Criação do Esquema Físico em Linguagem SQL

```
create table clientes (
  cod_cliente numeric(5) not null,
  nome varchar(100) not null,
  data_cadastro date not null,
  tipo char(1) not null,
  constraint pk_clientes primary key (cod_cliente),
  constraint chk_cli_tipo check (tipo in ('F','J'))
);
```

MODELO LÓGICO RELACIONAL

TIPOS DE DADOS

As COLUNAS das TABELAS possuem um TIPO DE DADO, que define o tipo de valor que irão armazenar.

Esquema Lógico Relacional

clientes

cod_cliente	NUMERIC (5)	NOT NULL
nome	VARCHAR (100)	NOT NULL (IE1)
data_cadastro	DATE	NOT NULL
tipo	CHAR (1)	NOT NULL

Criação do Esquema Físico em Linguagem SQL

```
create table clientes (
  cod_cliente numeric(5) not null,
  nome varchar(100) not null,
  data_cadastro date not null,
  tipo char(1) not null,
  constraint pk_clientes primary key (cod_cliente),
  constraint chk_cli_tipo check (tipo in ('F','J'))
);
```

Exemplos de Tipos de Dados Básicos

- **CHAR(tamanho)**: sequências de caracteres de tamanho fixo
- **VARCHAR(tamanho)**: sequências de caracteres de tamanho variável
- **NUMERIC(total de dígitos[, casas decimais])**: números inteiros ou decimais
- **DATE** ou **DATETIME**: informações relativas a datas e horários ou ambos

MODELO LÓGICO RELACIONAL

VALORES PADRÃO

É possível definir um **VALOR PADRÃO** (**DEFAULT**) para uma determinada coluna.

Este valor será utilizado caso não seja fornecido nenhum valor para aquela coluna por um comando de inserção de dados.

Esquema Lógico Relacional

pessoas_fisicas

cod_cliente	NUMERIC (5)	NOT NULL	(FK)
cpf	CHAR (14)	NOT NULL	(AK1)
genero	CHAR (1)	NOT NULL	
data_nascimento	DATE	NOT NULL	
cod_empresa	NUMERIC (5)		(FK) (IE1)

Criação do Esquema Físico em Linguagem SQL

```
create table pessoas_fisicas (
  cod_cliente numeric(5) not null,
  cpf char(14) not null,
  data_nascimento date not null,
  genero char(1) default 'F' not null,
  cod_empresa numeric(5),
  constraint pk_pessoas_fisicas primary key (cod_cliente),
  constraint ak_pf_cpf unique (cpf),
  constraint chk_pf_genero check (genero in ('F','M'))
);
```

MODELO LÓGICO RELACIONAL

VALIDAÇÃO DE VALORES

É possível validar os valores possíveis para uma determinada coluna criando uma restrição de **VALIDAÇÃO DE VALORES** (**CHECK**).

Esquema Lógico Relacional

pessoas_fisicas

cod_cliente	NUMERIC (5)	NOT NULL	(FK)
cpf	CHAR (14)	NOT NULL	(AK1)
genero	CHAR (1)	NOT NULL	
data_nascimento	DATE	NOT NULL	
cod_empresa	NUMERIC (5)		(FK) (IE1)

Criação do Esquema Físico em Linguagem SQL

```
create table pessoas_fisicas (
 cod_cliente numeric(5) not null,
 cpf char(14) not null,
 data_nascimento date not null,
 genero char(1) default 'F' not null,
 cod_empresa numeric(5),
 constraint pk_pessoas_fisicas primary key (cod_cliente),
 constraint ak_pf_cpf unique (cpf),
 constraint chk_pf_genero check (genero in ('F','M'))
);
```

OU

```
create table pessoas_fisicas (
 cod_cliente numeric(5) not null,
 cpf char(14) not null,
 data_nascimento date not null,
 genero char(1) default 'F' not null,
 cod_empresa numeric(5),
 constraint pk_pessoas_fisicas primary key (cod_cliente),
 constraint ak_pf_cpf unique (cpf)
);

alter table pessoas_fisicas
add constraint chk_pf_genero check (genero in ('F','M'));
```

MODELO LÓGICO RELACIONAL

RESTRIÇÕES DE INTEGRIDADE
RESTRIÇÃO DE NULIDADE

As restrições de nulidade (**NOT NULL**) estabelecem a obrigatoriedade de que os comandos de atualização de dados forneçam valores para determinadas colunas.

Esquema Lógico Relacional

clientes

cod_cliente	NUMERIC (5)	NOT NULL	
nome	VARCHAR (100)	NOT NULL	(IE1)
data_cadastro	DATE	NOT NULL	
tipo	CHAR (1)	NOT NULL	

Criação do Esquema Físico em Linguagem SQL

```
create table clientes (
  cod_cliente numeric(5) not null,
  nome varchar(100) not null,
  data_cadastro date not null,
  tipo char(1) not null,
  constraint pk_clientes primary key (cod_cliente),
  constraint chk_cli_tipo check (tipo in ('F','J'))
);
```

MODELO LÓGICO RELACIONAL

RESTRIÇÕES DE INTEGRIDADE
CHAVES PRIMÁRIAS

O MODELO LÓGICO RELACIONAL suporta a definição de **RESTRIÇÕES DE INTEGRIDADE** (**CONSTRAINTS**) sobre os dados das tabelas.

As restrições de entidade ou de **CHAVE PRIMÁRIA** (**PRIMARY KEY**) validam a unicidade de uma ou mais colunas, que serão utilizadas como os identificadores das LINHAS/REGISTROS da TABELA.

Esquema Lógico Relacional

clientes

cod_cliente	NUMERIC (5)	NOT NULL
nome	VARCHAR (100)	NOT NULL (IE1)
data_cadastro	DATE	NOT NULL
tipo	CHAR (1)	NOT NULL

Criação do Esquema Físico em Linguagem SQL

```
create table clientes (
  cod_cliente numeric(5) not null,
  nome varchar(100) not null,
  data_cadastro date not null,
  tipo char(1) not null,
  constraint pk_clientes primary key (cod_cliente),
  constraint chk_cli_tipo check (tipo in ('F','J'))
);
```

OU

```
create table clientes (
  cod_cliente numeric(5) not null,
  nome varchar(100) not null,
  data_cadastro date not null,
  tipo char(1) not null,
  constraint chk_cli_tipo check (tipo in ('F','J'))
);

alter table clientes
add constraint pk_clientes primary key(cod_cliente);
```

MODELO LÓGICO RELACIONAL

RESTRIÇÕES DE INTEGRIDADE
CHAVES ALTERNATIVAS

As restrições de **CHAVES ALTERNATIVAS** (**UNIQUE**) estabelecem a obrigatoriedade de que uma ou mais colunas de uma tabela sejam ÚNICAS em conjunto.

São identificadores alternativos além da chave primária da tabela.

Esquema Lógico Relacional

pessoas_fisicas

cod_cliente	NUMERIC (5)	NOT NULL (FK)	
cpf	CHAR (14)	NOT NULL	(AK1)
genero	CHAR (1)	NOT NULL	
data_nascimento	DATE	NOT NULL	
cod_empresa	NUMERIC (5)		(FK) (IE1)

Criação do Esquema Físico em Linguagem SQL

```
create table pessoas_fisicas (
 cod_cliente numeric(5) not null,
 cpf char(14) not null,
 data_nascimento date not null,
 genero char(1) not null,
 cod_empresa numeric(5),
 constraint pk_pessoas_fisicas primary key (cod_cliente),
 constraint ak_pf_cpf unique (cpf),
 constraint chk_pf_genero check (genero in ('F','M'))
);
```

OU

```
create table pessoas_fisicas (
 cod_cliente numeric(5) not null,
 cpf char(14) not null,
 data_nascimento date not null,
 genero char(1) not null,
 cod_empresa numeric(5),
 constraint pk_pessoas_fisicas primary key (cod_cliente),
 constraint chk_pf_genero check (genero in ('F','M'))
);

alter table pessoas_fisicas
add constraint constraint ak_pf_cpf unique (cpf);
```

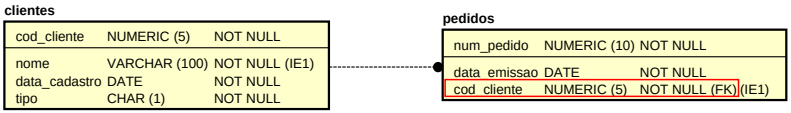
MODELO LÓGICO RELACIONAL

RESTRIÇÕES DE INTEGRIDADE
CHAVES ESTRANGEIRAS

No modelo lógico relacional, as colunas das tabelas podem referenciar umas às outras, para garantir a correta implementação dos **RELACIONAMENTOS/ASSOCIAÇÕES** representados no ESQUEMA CONCEITUAL.

As chamadas RESTRIÇÕES DE INTEGRIDADE REFERENCIAL garantem que os dados armazenados estão corretos em relação

Esquema Lógico Relacional



Criação do Esquema Físico em Linguagem SQL

```
create table pedidos (
   num_pedido numeric(10) not null,
   data_emissao date not null,
   cod_cliente numeric(5) not null,
   constraint pk_pedidos primary key (num_pedido),
   constraint fk_ped_cli foreign key (cod_cliente) references clientes
);
```

OU

```
create table pedidos (
   num_pedido numeric(10) not null,
   data_emissao date not null,
   cod_cliente numeric(5) not null,
   constraint pk_pedidos primary key (num_pedido)
);

alter table pedidos
add constraint fk_ped_cli foreign key (cod_cliente) references clientes;
```

SGBD Oracle

```
alter table pedidos
add constraint fk_ped_cli foreign key (cod_cliente) references clientes
[ on delete { cascade | set null } ];
```

SGBD Microsoft SQL Server

```
alter table pedidos
add constraint fk_ped_cli foreign key (cod_cliente) references clientes
[ on delete { no action | cascade | set null | set default } ];
[ on update { no action | cascade | set null | set default } ];
```

Criação de *surrogate keys*

Oracle (sequence)

```
create sequence seq_itens_pedidos
  start with 1
  increment by 1;

alter table itens_pedidos
add id_item_pedido numeric(10)
  default seq_itens_pedidos.nextval
  not null;

insert into itens_pedidos(num_pedido, ...)
values(...);
```

Oracle (UUID)

```
create or replace and compile
   java source named "RandomUUID"
as
public class RandomUUID {
  public static String create() {
    return
      java.util.UUID.randomUUID().toString();
  }
}
/

create or replace function randomuuid
   return varchar2
as language java
name 'RandomUUID.create() return
java.lang.String';
/

alter table itens_pedidos
add id_item_pedido char(36);

insert into itens_pedidos(id_item_pedido,
num_pedido, ...)
values(randomUUID(), ...);
```

Microsoft SQL Server

```
alter table itens_pedidos
add id_item_pedido int identity(1,1);

insert into itens_pedidos(num_pedido, ...)
values(...);
```

NORMALIZAÇÃO DE DADOS

Nem sempre a um esquema lógico relacional de dados na forma normalizada é a melhor alternativa considerando-se aspectos de desempenho, especialmente em soluções de BI.

Nessas situações pode-se optar por utilizar **esquemas não-normalizados**, desde que a redundância seja controlada.

- Processo pelo qual cria-se um modelo relacional de dados pela aplicação de Formas Normais a dados brutos, ditos não normalizados
- Visa obter um banco de dados livre de anomalias de atualização
- Baseado no conceito de Forma Normal:

Uma forma normal é uma regra que deve ser obedecida por uma tabela para que esta seja considerada "bem projetada"[1]

[1]HEUSER, C. A. Projeto de banco de Dados. 6ed. Porto Alegre: Sagra-DC Luzzatto, 2009. 282p.

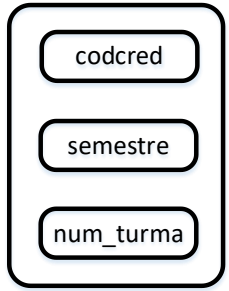
Normalização de Dados: Exemplo

CODCRED	NOME	COD_DEPTO	NOME_DEPTO	CURRICULO	CREDITOS	MATR_PROF	NOME_PROF	FORMACAO	TURMA	HORARIO	MATR_ALUNO	NOME_ALUNO	SEMESTRE	G1	G2
4634C-04	Sistemas de Gerência de Banco de Dados	463	Fundamentos da Computação	2201	4	44981	Eduardo Arruda	Doutor	268	2NP4NP	131082644	Ben-Hur Silva	20151	4,1	
4634A-02	Laboratório de Banco de Dados I	463	Fundamentos da Computação	2201	2	44981	Eduardo Arruda	Doutor	138	6LM	131082644	Ben-Hur Silva	20151	6,8	
4634C-04	Sistemas de Gerência de Banco de Dados	463	Fundamentos da Computação	2201	4	44981	Eduardo Arruda	Doutor	268	2NP4NP	334455667	Diel Silva	20151	1,7	4,0
4634A-02	Laboratório de Banco de Dados I	463	Fundamentos da Computação	2201	2	24	Cristiano Galina	Mestre	268	6NP	334455667	Diel Silva	20151	6,5	
4634C-04	Sistemas de Gerência de Banco de Dados	463	Fundamentos da Computação	2201	4	24	Cristiano Galina	Mestre	138	2LM4LM	546465737	Alexandre Silva	20151	5,3	
4634A-02	Laboratório de Banco de Dados I	463	Fundamentos da Computação	2201	2	44981	Eduardo Arruda	Doutor	138	6LM	546465737	Alexandre Silva	20151	9,0	
4634C-04	Sistemas de Gerência de Banco de Dados	463	Fundamentos da Computação	2201	4	24	Cristiano Galina	Mestre	138	2LM4LM	657465843	Eu Silva	20151	2,7	7,0

1 DEFINIR IDENTIFICADOR

DETERMINAR QUE ATRIBUTOS IDENTIFICAM O FATO

No exemplo, podemos supor que o fato são as turmas de uma disciplina em dado semestre.

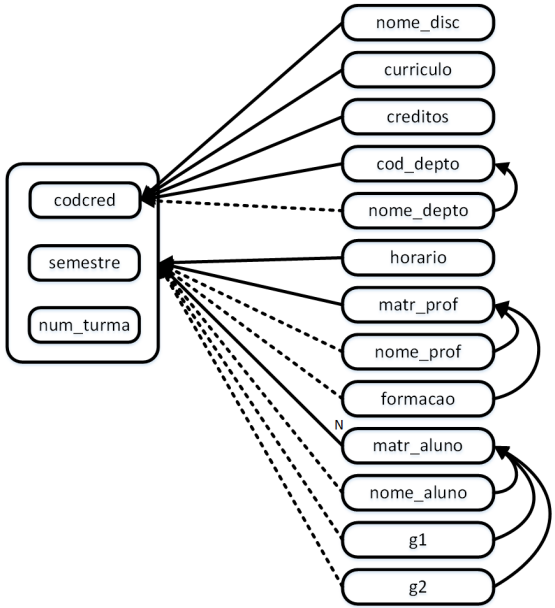


2 DEPENDÊNCIAS FUNCIONAIS

IDENTIFICAR DEPENDÊNCIAS FUNCIONAIS ENTRE OS ATRIBUTOS

Definimos as dependências funcionais dos atributos em relação ao identificador e as dependências entre os atributos.

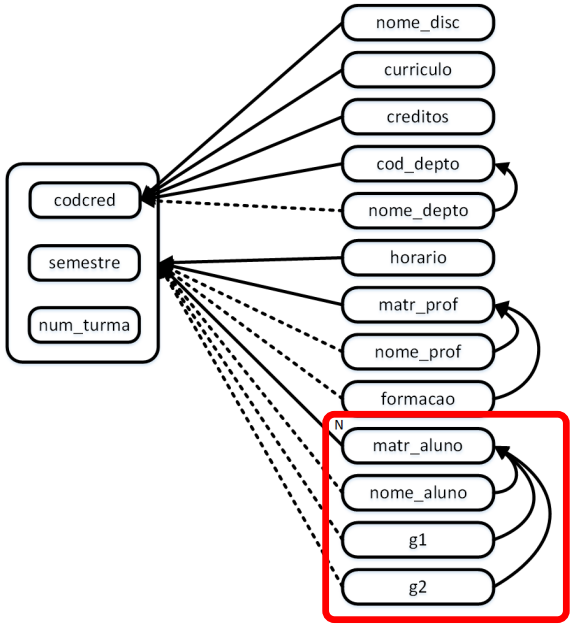
Também devemos identificar os atributos multivalorados.



3 1ª FORMA NORMAL

ELIMINAR MULTIVALORAÇÕES

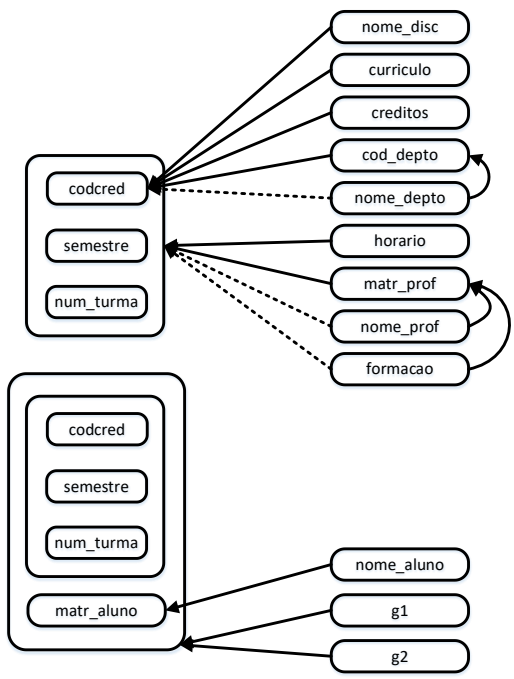
Cada turma de uma disciplina em dado semestre reúne N alunos.



3 1ª FORMA NORMAL

ELIMINAR MULTIVALORAÇÕES

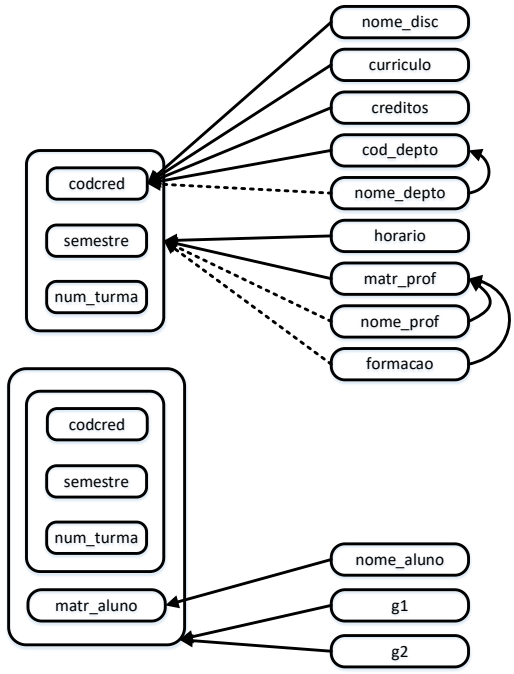
Devemos "remover" a multivaloração, mantendo o vínculo com o identificador original.



4 2ª FORMA NORMAL

ELIMINAR DEPENDÊNCIAS FUNCIONAIS PARCIAIS

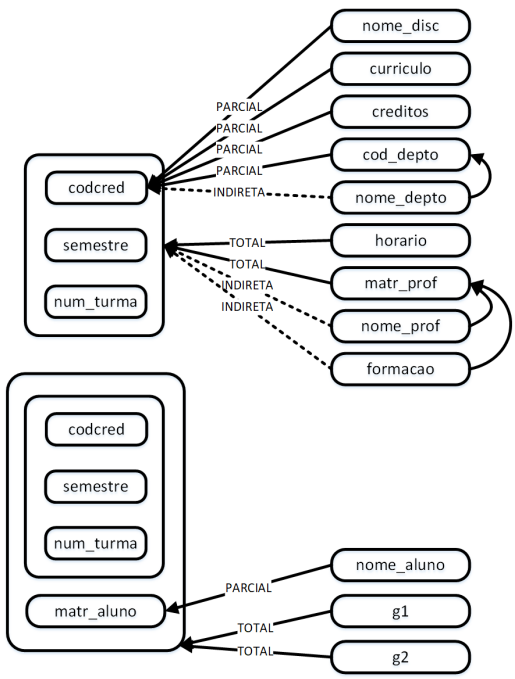
Uma dependência funcional é PARCIAL em relação ao identificador quando o atributo não depende de todo o identificador.



4 2ª FORMA NORMAL

ELIMINAR DEPENDÊNCIAS FUNCIONAIS PARCIAIS

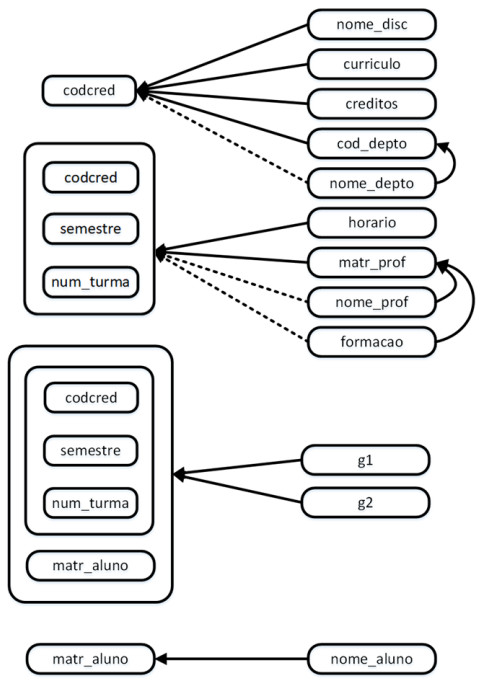
Uma dependência funcional é PARCIAL em relação ao identificador quando o atributo não depende de todo o identificador.



4 2ª FORMA NORMAL

ELIMINAR DEPENDÊNCIAS FUNCIONAIS PARCIAIS

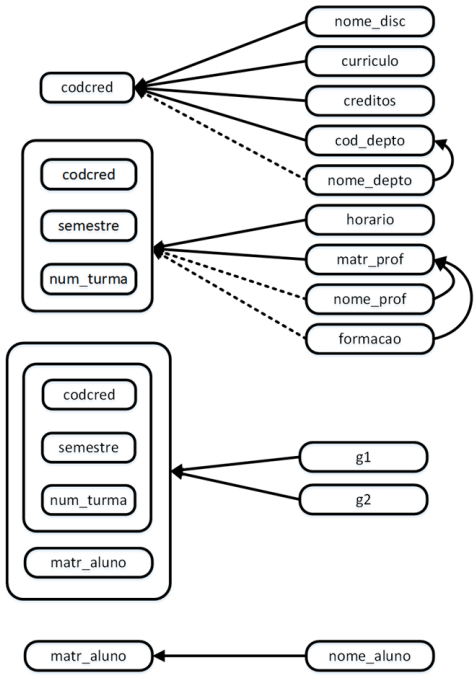
Elimina-se as dependências funcionais PARCIAIS removendo as cadeias de dependência mas mantendo-se o vínculo com o identificador.



5 3ª FORMA NORMAL

ELIMINAR DEPENDÊNCIAS FUNCIONAIS INDIRETAS

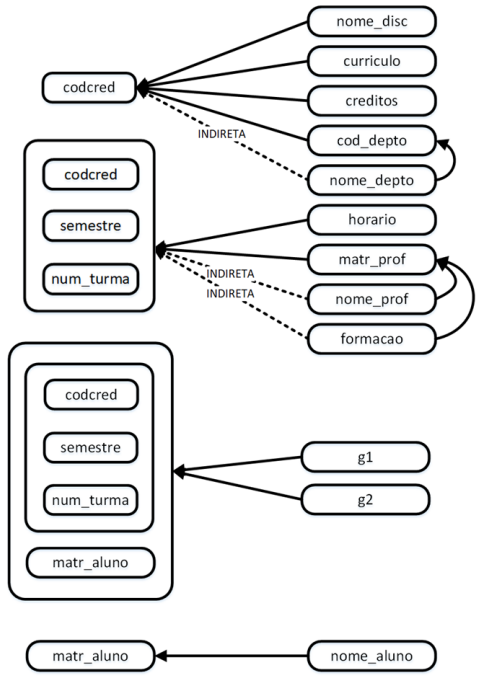
Uma dependência funcional é INDIRETA em relação ao identificador quando o atributo depende de outro atributo que, este sim, depende do identificador.



5 3ª FORMA NORMAL

ELIMINAR DEPENDÊNCIAS FUNCIONAIS INDIRETAS

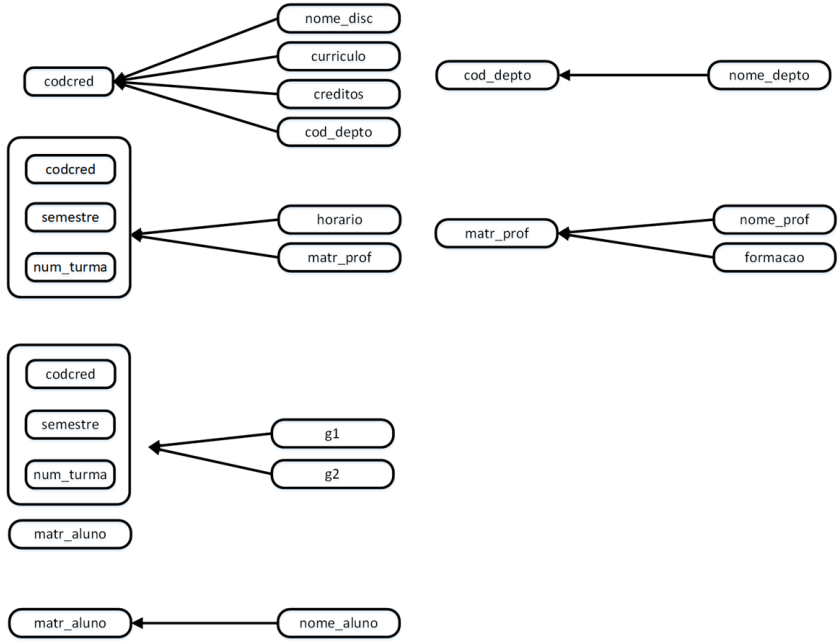
Uma dependência funcional é INDIRETA em relação ao identificador quando o atributo depende de outro atributo que, este sim, depende do identificador.



5 3ª FORMA NORMAL

ELIMINAR DEPENDÊNCIAS FUNCIONAIS INDIRETAS

Elimina-se as dependências funcionais INDIRETAS removendo as cadeias de dependência mas mantendo-se o vínculo com o identificador.



Exercício 1

Modelagem conceitual e mapeamento para modelo lógico relacional

SGBDs não relacionais

Revisão

Requisitos de BDs

C*onsistency* (Consistência)

Cada leitura recebe a gravação mais recente ou um erro.

A*vailability* (Disponibilidade)

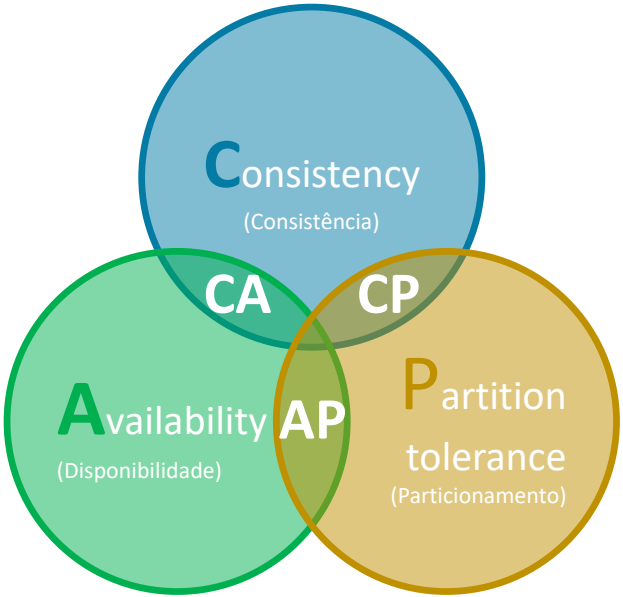
Cada solicitação recebe uma resposta (sem erro), sem a garantia de que contém a gravação mais recente.

P*artition tolerancy* (Tolerância de partição ou particionamento)

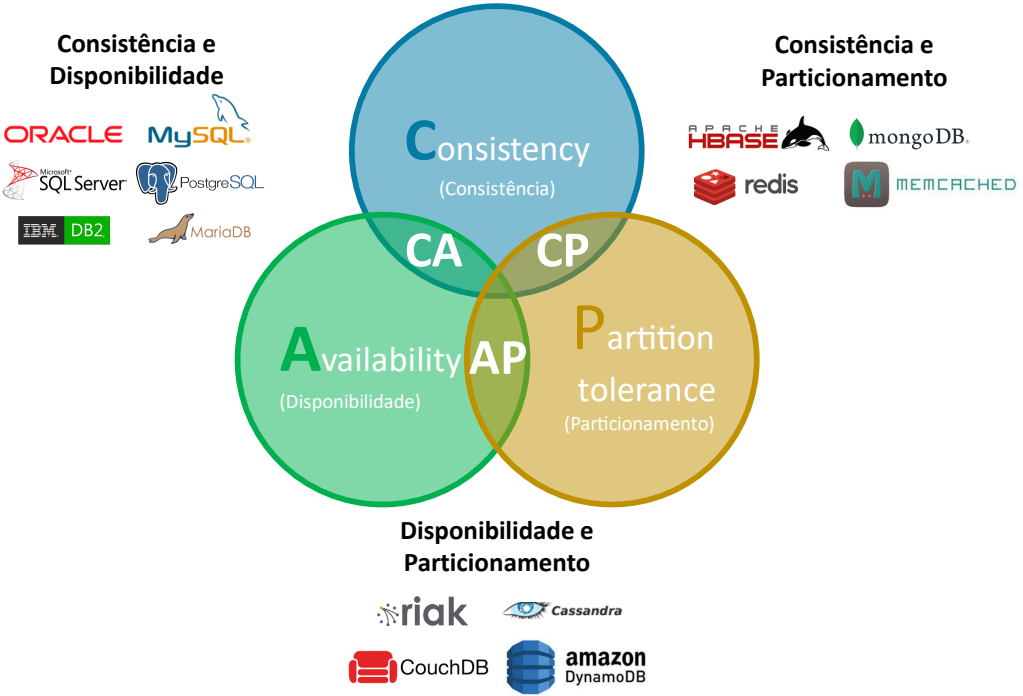
O sistema continua a operar apesar de um número arbitrário de mensagens serem descartadas (ou atrasadas) pela rede entre os nós

Teorema CAP

- Proposto por Eric Brewer em 1998
- Quando ocorre uma falha de partição (P), deve-se decidir se a operação será cancelada e, portanto, diminuirá a disponibilidade (A), mas garantirá a consistência (C), ou prosseguirá com a operação e, assim, fornecerá disponibilidade (A), mas arriscará a inconsistência (~C).
- Assim, se houver um particionamento de dados, deve-se escolher entre consistência (C) ou disponibilidade (A).



Eric Brewer argumenta que o conceito "dois de três" pode ser um pouco enganador, porque os projetistas só precisam sacrificar a consistência (C) ou disponibilidade (A) na presença de partições (P), mas naquela época o particionamento era pouco frequente.


Consistência e Disponibilidade
ORACLE
MySQL
Microsoft SQL Server
PostgreSQL
IBM DB2
MariaDB
Consistency
(Consistência)
CA
CP
Availability
(Disponibilidade)
AP
Partition tolerance
(Particionamento)
Consistência e Particionamento
APACHE HBASE
mongoDB.
redis
MEMCACHED
Disponibilidade e Particionamento
riak
Cassandra
CouchDB
amazon DynamoDB

Modelos de Transações de SGBDs

Atomic
Consistent
Isolated
Durable

Basically **A**vailable
Soft state
Eventually consistent

Modelos de Transações de SGBDs

Atomic
Consistent
Isolated
Durable

Basically **A**vailable
Soft state
Eventually consistent

Requisitos ACID

Atomicidade

- Cada transação é executada corretamente ou o processo é interrompido e o banco de dados volta ao estado anterior ao início da transação.
- Isso garante que todos os dados no banco de dados sejam válidos.

Consistência

- Uma transação processada nunca colocará em risco a integridade estrutural do banco de dados.

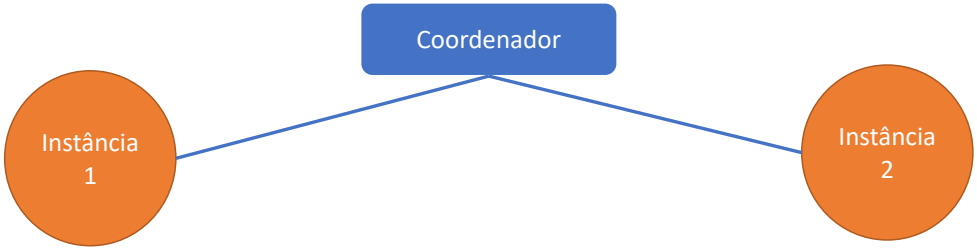
Isolamento

- As transações não podem comprometer a integridade de outras transações interagindo com elas enquanto ainda estão em andamento.

Durabilidade

- Os dados relacionados à transação concluída persistirão mesmo nos casos de queda de rede ou energia. Se uma transação falhar, ela não afetará os dados manipulados.

Replicação de dados em SGBDs ACID



Two-Phase Commit

- Durante a execução da transação, antes do início do protocolo de confirmação de duas fases
  - Quando a aplicação chama tx_begin para iniciar a transação, o coordenador cria um registro de transação em sua memória volátil
  - Cada vez que um gerenciador de recursos chama xa_reg para ingressar na transação como um instância, o coordenador anexa a identidade da instância ao registro da transação

Fase 1: Preparação

- Quando o aplicativo invoca tx_commit
- O coordenador envia mensagem de preparação (coordenação para todas as instâncias):
  - Se a instância quiser abortar a qualquer momento antes ou no recebimento da mensagem, ela aborta e libera bloqueios
  - Se a instância quiser confirmar, ela move todos os registros de atualização para o armazenamento em massa, forçando um registro de preparação para seu log
  - Garante que a instância será capaz de confirmar (apesar de falhas) se o coordenador decidir confirmar (já que os registros de atualização são duráveis)
  - Instância entra no estado preparado
- A instância envia uma mensagem de voto (“pronto” ou “abortando”)
  - não pode mudar de ideia
  - retém todos os bloqueios se o voto estiver "pronto"
  - entra em período incerto (não pode prever o resultado final)

Fase 1: Preparação

- Mensagem de votação (instância para coordenador): instância indica que está “pronto” para se comprometer ou está “abortando”
  - Coordenador registra voto no registro de transação
  - Se algum voto estiver “abortando”, o coordenador decide abortar e apaga o registro da transação
  - Se todos estiverem “prontos”, o coordenador decide o commit, força o registro do commit (contendo o registro da transação) para o seu log (fim da fase 1)
  - Transação confirmada quando o registro de confirmação é durável
  - Como todas as instâncias estão no estado preparado, a transação pode ser confirmada apesar de quaisquer falhas
  - O coordenador envia uma mensagem de confirmação ou aborto para todas as instâncias

Fase 2: Commit

- Envio de mensagem de commit ou abort (coordenador para instância)
  - Se enviar mensagem de commit
    - instância confirma localmente forçando um registro de confirmação em seu log
    - coorte envia mensagem de concluída ao coordenador
  - Se enviar a mensagem de abort, aborta
  - Em ambos os casos, os bloqueios são liberados e o período incerto termina
- Envio de mensagem de conclusão (da instância para coordenador)
  - Quando o coordenador recebe uma mensagem de conclusão de cada instância
    - ele grava um registro completo em seu log e
    - exclui o registro de transação do armazenamento volátil

Modelos de Transações de SGBDs

Atomic
Consistent
Isolated
Durable

Basically Available

Soft state

Eventually consistent

Requisitos BASE

BAsicamente disponível

- Em vez de impor consistência imediata, os bancos de dados NoSQL modelados em BASE garantirão a disponibilidade dos dados, espalhando-os e replicando-os pelos nós do cluster de banco de dados.

Soft state

- Devido à falta de consistência imediata, os valores dos dados podem mudar ao longo do tempo.
- O modelo BASE rompe com o conceito de banco de dados que impõe sua própria consistência, delegando essa responsabilidade aos desenvolvedores.

Eventualmente consistente

- O fato de o BASE não impor consistência imediata não significa que nunca a alcance.
- No entanto, até que isso aconteça, as leituras de dados ainda são possíveis (mesmo que não reflitam a realidade).

Modelagem lógica não relacional

https://www.mongodb.com/

- MongoDB é um SGBD NoSQL gratuito para armazenamento de documentos
- Suporta requisitos ACID em transações distribuídas com múltiplos documentos (a partir da versão 4.2).
- Amplamente utilizado em implementações que obtém benefício de SGBDs *Schema-less* (sem esquema pré-definido)
- Documentos armazenados em formato JSON (BSON), permitindo armazenar documentos com estruturas diferentes ou mesmo variá-las ao longo do tempo
- Suporta indexação e agregação de dados

Organização dos Dados

- Databases: Armazenam registros de dados como documentos (BSON) reunidos em *collections*.
- Collections: Armazenam os documentos, sendo análogas às *tables* em SGBDs relacionais.

Modelagem de Dados

- Suporta integridade referencial
  - Cuidado!!!
- Desnormalização
  - Em implementações não relacionais a desnormalização é um recurso muito usual
  - Muitas vezes bancos de dados relacionais também são desnormalizados
    - Melhorar o desempenho
    - Manter valores históricos

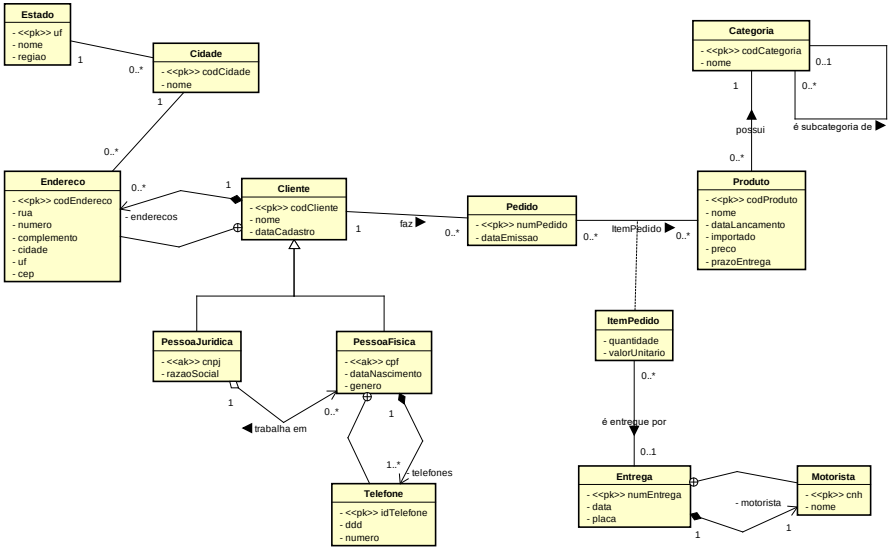
MODELO LÓGICO NÃO RELACIONAL

Baseado em

documentos

mongoDB®

Esquema Conceitual



https://cassandra.apache.org/

- Apache Cassandra é um SGBD NoSQL de código aberto e gratuito
- Baseado em modelo de armazenamento de colunas amplas (*wide columns*) particionadas
- Baseado em requisitos BASE
- Inicialmente projetado no Facebook
  - Arquitetura orientada a eventos (SEDA)
  - Baseado
    - Nas técnicas de armazenamento e replicação do Amazon Dynamo
    - No modelo de mecanismo de armazenamento e dados Google Bigtable do Google

Organização dos Dados nos Nós

- Keyspaces: definem como um conjunto de dados é replicado entre datacenters. Replicação é o número de cópias de um mesmo dado salvas por cluster. Os keyspaces contêm tabelas.
- Tables: definem o esquema tipado para uma coleção de partições. As tabelas contêm partições, que contêm linhas, que contêm colunas. As tabelas do Cassandra podem adicionar novas colunas de forma flexível às tabelas, com tempo de inatividade zero.
- Partitions: Definem a parte obrigatória da chave primária que todas as linhas do Cassandra devem ter para identificar o nó em um cluster onde a linha está armazenada. Todas as consultas de alto desempenho fornecem a chave de partição na consulta.
- Rows: contém uma coleção de colunas identificadas por uma chave primária exclusiva composta pela chave de partição e, opcionalmente, chaves de cluster adicionais.
- Columns: Um único dado com um tipo que pertence a uma linha.

Modelagem de Dados

- Sem *joins*
  - *Joins* devem ser implementados no cliente ou pela desnormalização dos dados
- Sem integridade referencial
  - Cassandra não implementa
- Desnormalização
  - Cassandra porque ele funciona melhor quando o modelo de dados é desnormalizado
  - Novamente: muitas vezes bancos de dados relacionais também são desnormalizados
    - Melhorar o desempenho
    - Manter valores históricos

Modelagem de Dados

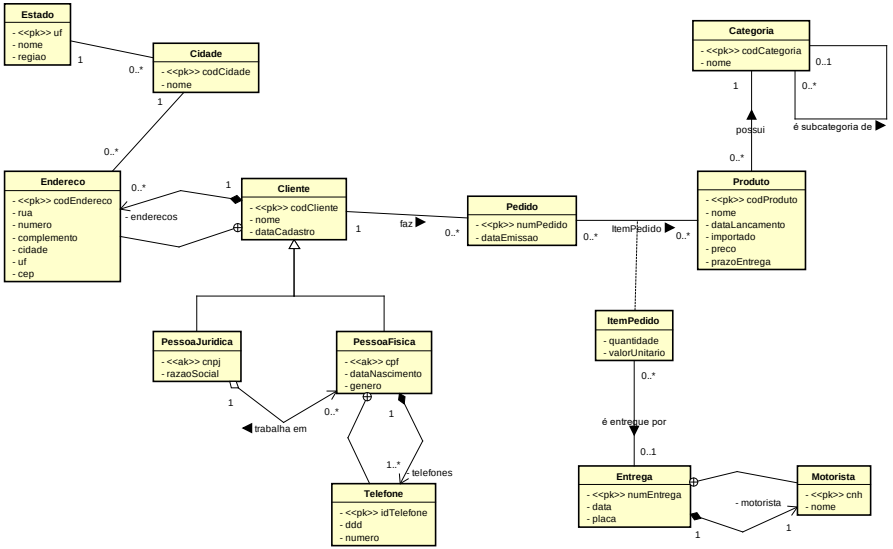
- Projetando para armazenamento ideal
  - Manter as colunas relacionadas definidas juntas na mesma tabela
  - Minimizar o número de partições que devem ser pesquisadas para satisfazer uma determinada consulta
- Classificação é uma decisão de projeto
  - A ordem de classificação nas consultas é fixa e determinada inteiramente pela seleção de colunas de cluster que você fornece no comando CREATE TABLE
  - CQL SELECT oferece suporte a ORDER BY mas somente na ordem especificada pelas colunas de *clustering*

MODELO LÓGICO NÃO RELACIONAL

Baseado em

wide columns

Esquema Conceitual



PUCRS online uol edtech