

ARMAZENAMENTO E PROCESSAMENTO DE BANCOS DE DADOS RELACIONAIS

EDUARDO C. DE ALMEIDA ¹

EDUARDO@INF.UFPR.BR



VISÃO GERAL DA APRESENTAÇÃO

- ▶ Entender como um banco de dados aparenta no nível lógico de armazenamento e como consultar os dados deste armazenamento

(PRÉ 1958) ... NO INÍCIO ERAM TREVAS ...

- ▶ Sem sistema operacional
- ▶ Sem sistema de arquivos
- ▶ Sem sistema de bancos de dados
- ▶ Sem sistema de comunicação
- ▶ Sem modelo de multi-programação
- ▶ Sem ...
- ▶ Somente máquina crua e a necessidade de organizar os dados!!!



The Origin of the Integrated Data Store (IDS):
The First Direct-Access DBMS
C. Bachman
IEEE, 2009

1958/61 – INTEGRATED DATA STORE (IDS)

- ▶ Mensagens dos processos -> OLTP
- ▶ Diagrama de estruturas de dados
- ▶ Processamento "tupla-por-vez"



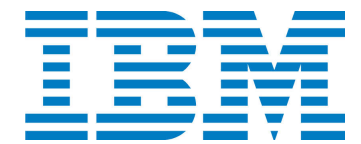
The Origin of the Integrated Data Store (IDS):
The First Direct-Access DBMS
C. Bachman
IEEE, 2009

1960 – IBM INFORMATION MANAGEMENT SYSTEM (IMS)

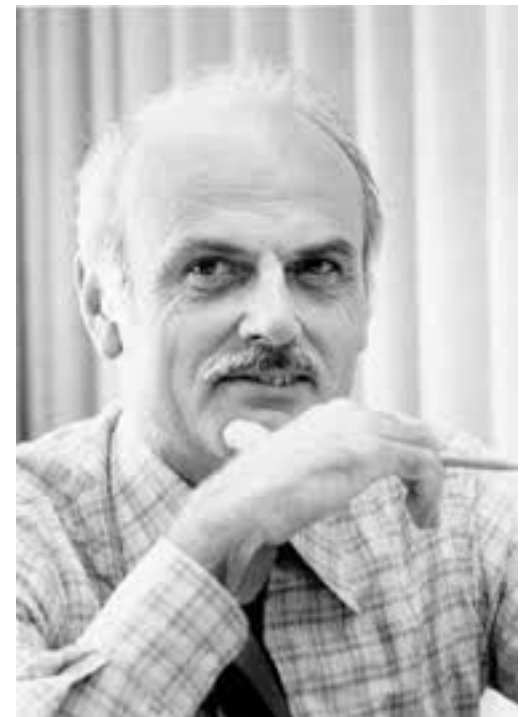
- ▶ Gestão dos componentes do foguete da missão Apollo da NASA
- ▶ Modelo Hierárquico de dados
- ▶ Implementação complexa
- ▶ Sem padrões de manipulação de dados
- ▶ Sem independência estrutural

1970 – MODELO RELACIONAL

Objetivo: Evitar a manutenção dos programas observada no IMS/CODASYL (alterar programas após mudança no esquema)



- ▶ Armazenamento em estruturas relacionais
- ▶ Formas normais (evitar anomalias)
- ▶ Acesso via linguagem de alto nível



A relational model of data for large shared data banks

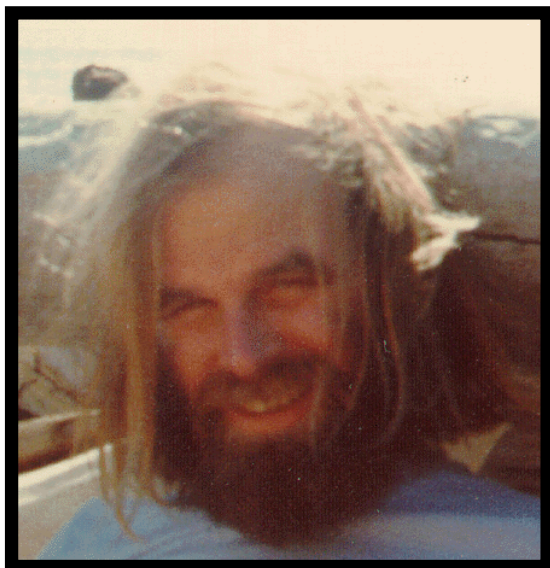
E. Codd

Comm. of the ACM, 1970

1970 – MODELO RELACIONAL

Primeiras implementações:

- ▶ IBM: System-R
- ▶ UC Berkeley: Ingres
- ▶ Oracle

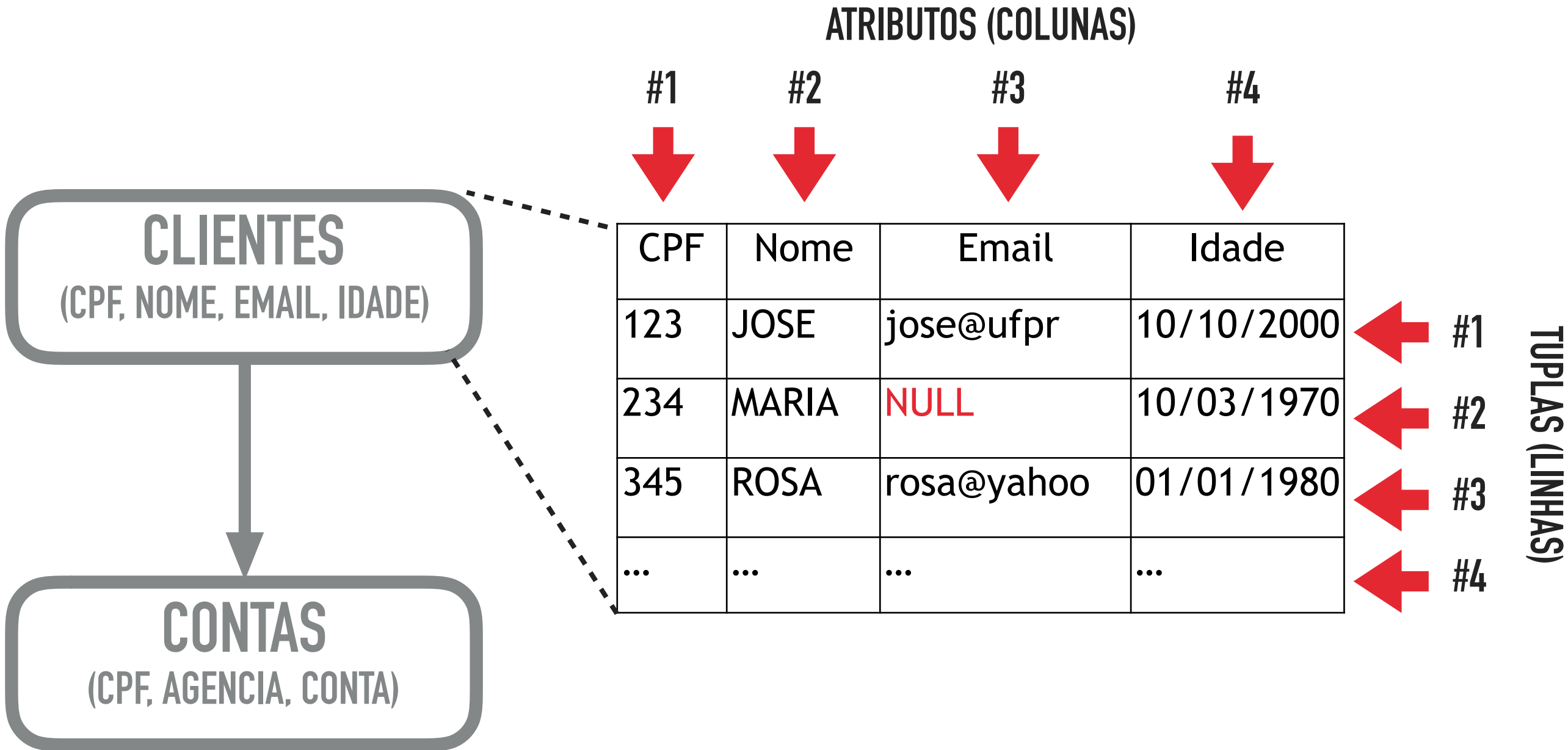


1980/90 – IMPLEMENTAÇÕES DO MODELO RELACIONAL

- ▶ System-R -> DB2
- ▶ SQL como padrão de linguagem
- ▶ Oracle ganha mercado
- ▶ Várias empresas implementam o modelo
- ▶ “Fork” do Sybase pela Microsoft

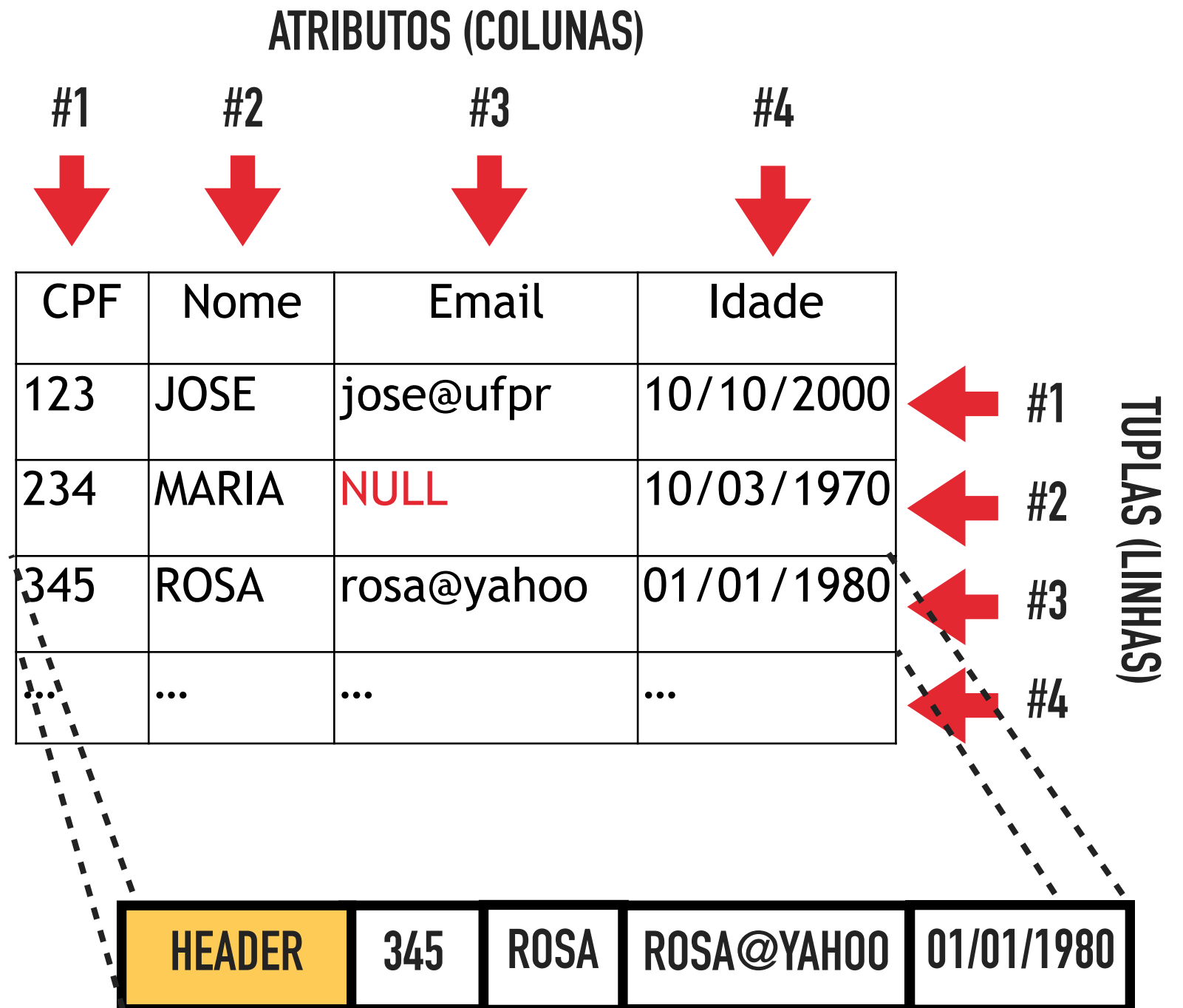


MODELO RELACIONAL



LAYOUT DE UMA TUPLA

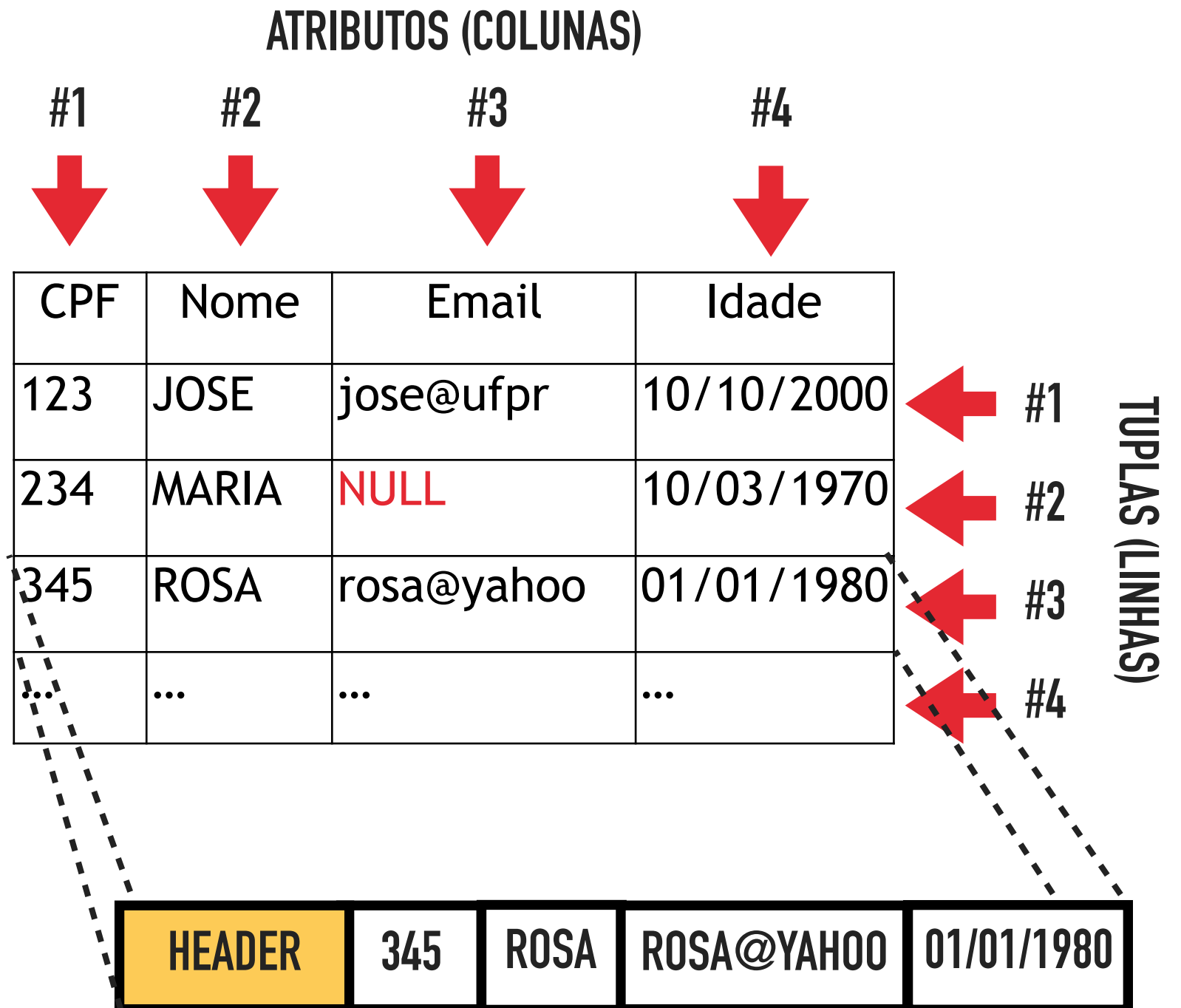
- ▶ Tupla é uma sequencia de bits interpretados pelo BD
- ▶ Header mantém visibilidade e bitmap de NULLs



LAYOUT DE UMA TUPLA

- ▶ Tupla é uma sequencia de bits interpretados pelo BD
- ▶ Header mantém visibilidade e bitmap de NULLs
- ▶ Atributos armazenados na ordem da definição

```
CREATE TABLE CLIENTES(  
  CPF      INT;  
  NOME     VARCHAR(50);  
  EMAIL    VARCHAR(50);  
  IDADE    DATE;  
);
```



LAYOUT DE UMA TUPLA

ATRIBUTOS (COLUNAS)

```
CREATE TABLE CLIENTES(
```

COMO OS SISTEMAS DE BANCO DE DADOS ARMAZENAM AS TUPLAS ?

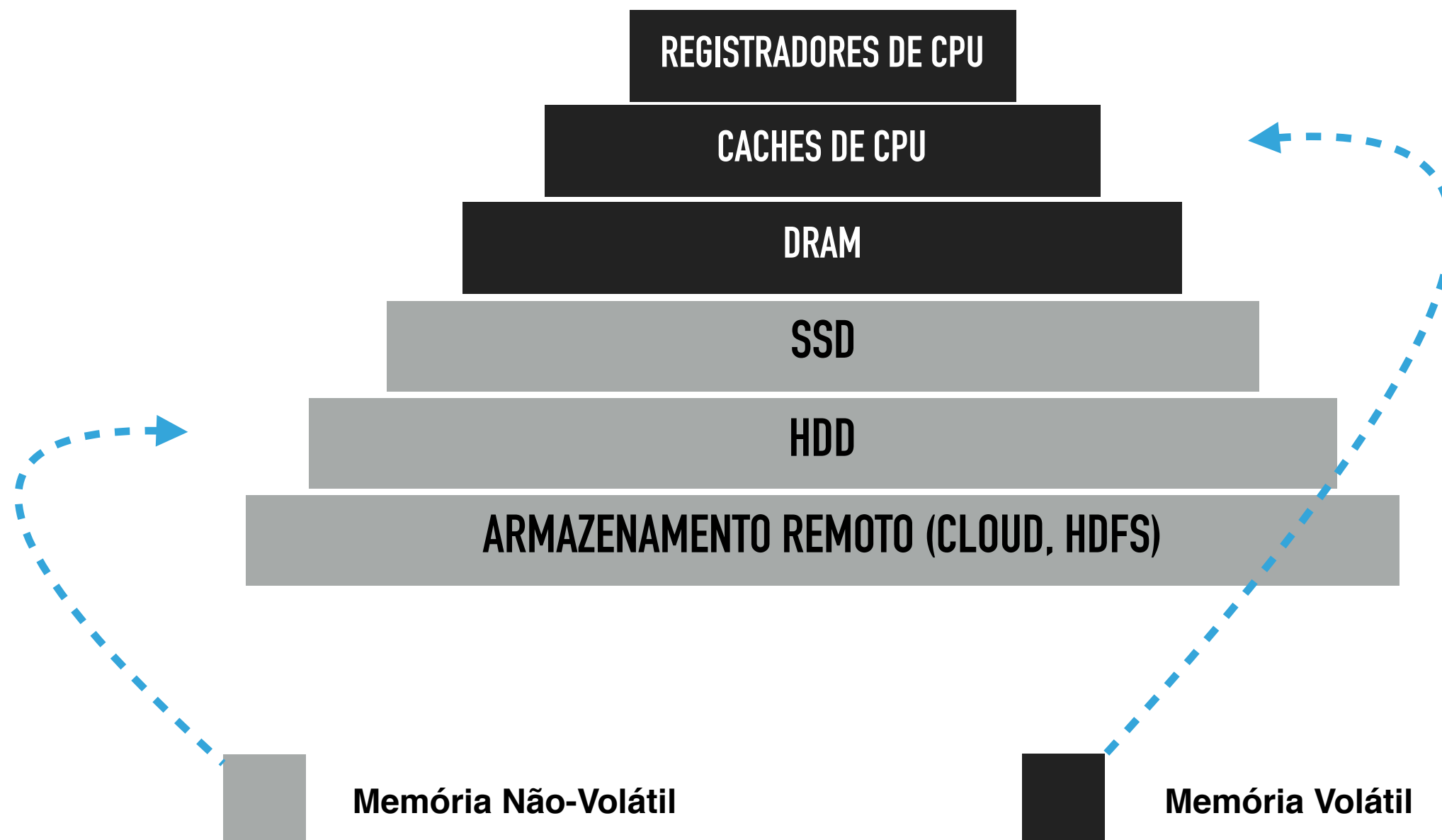
► T
b

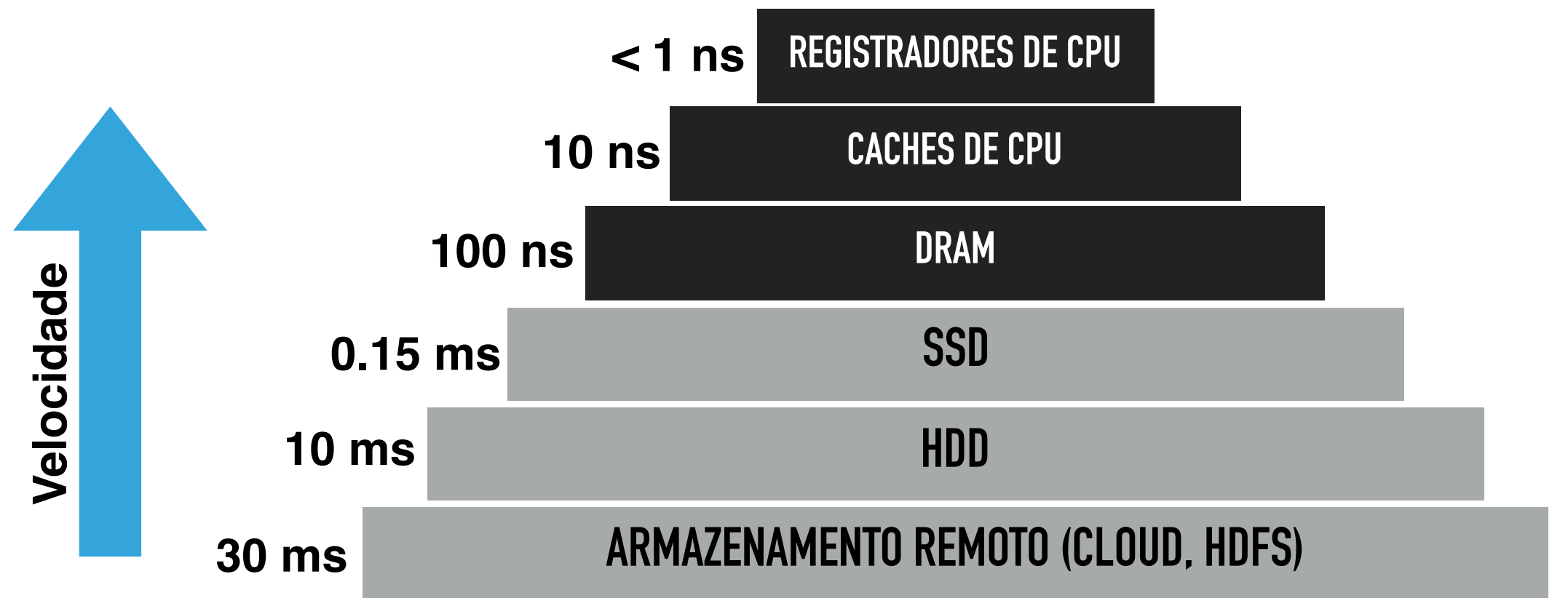
► Header mantém visibilidade e bitmap de NULLs

► Atributos armazenados na ordem da definição



HIERARQUIA DE MEMÓRIA

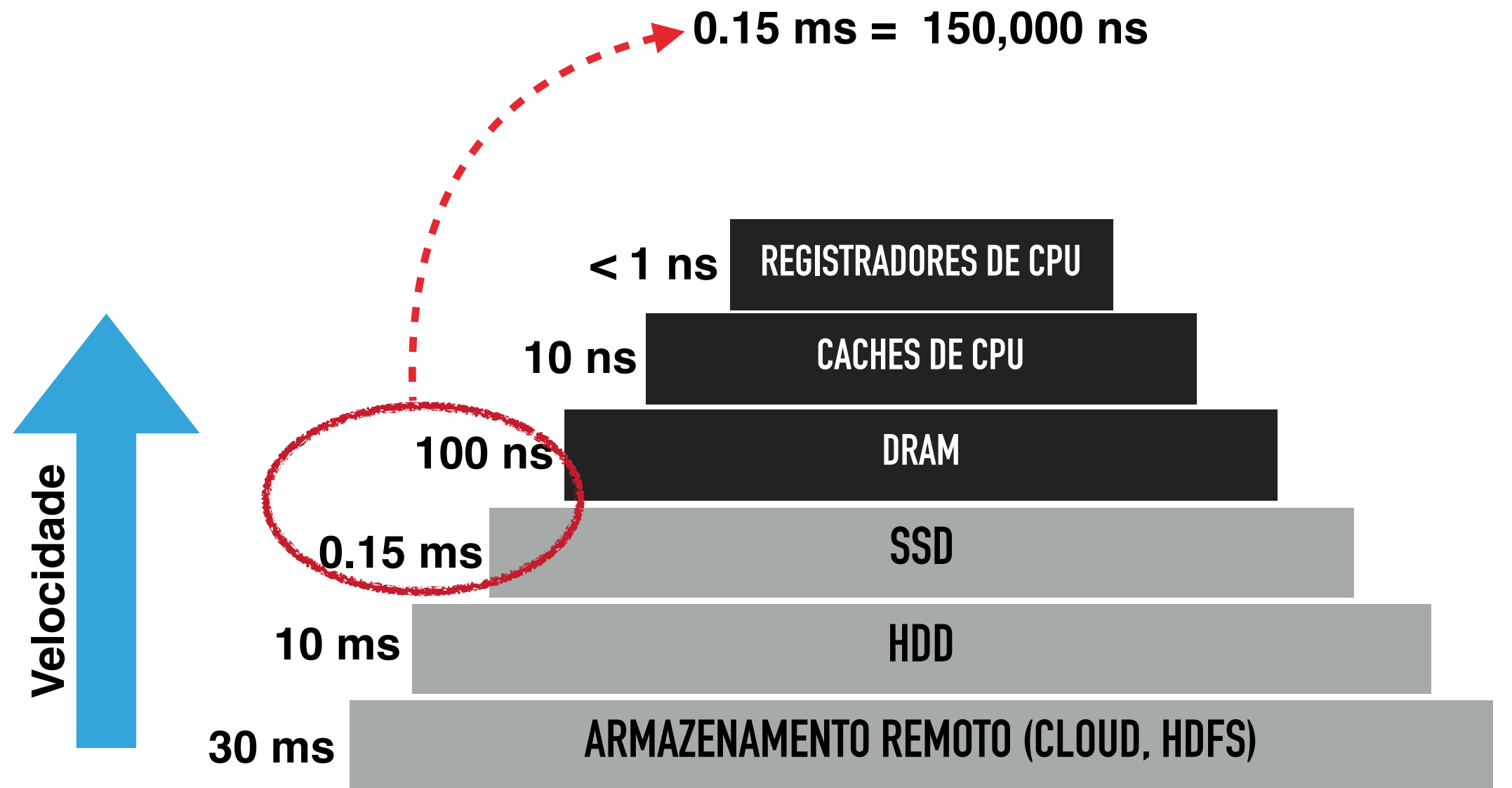




Memória Não-Volátil



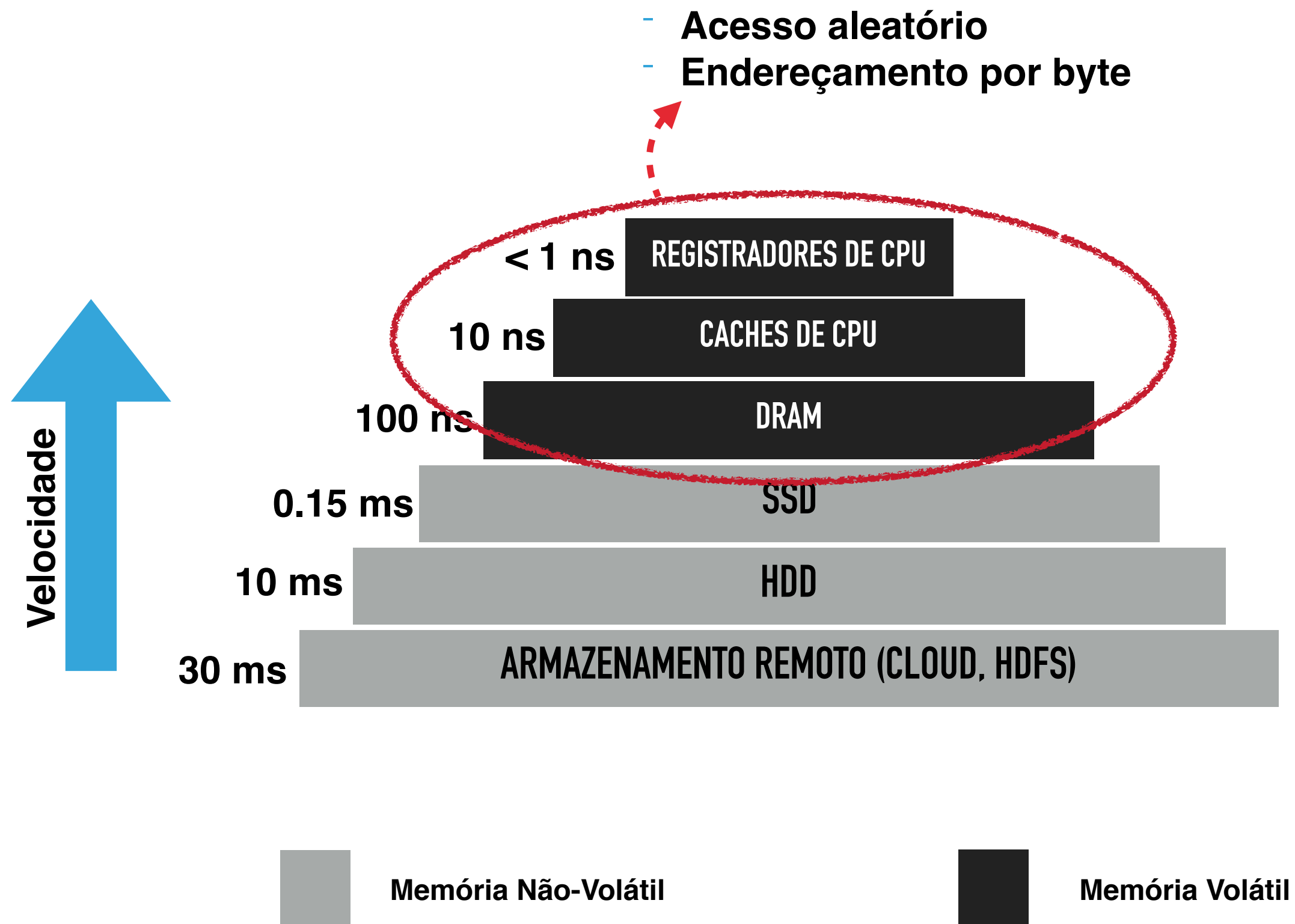
Memória Volátil

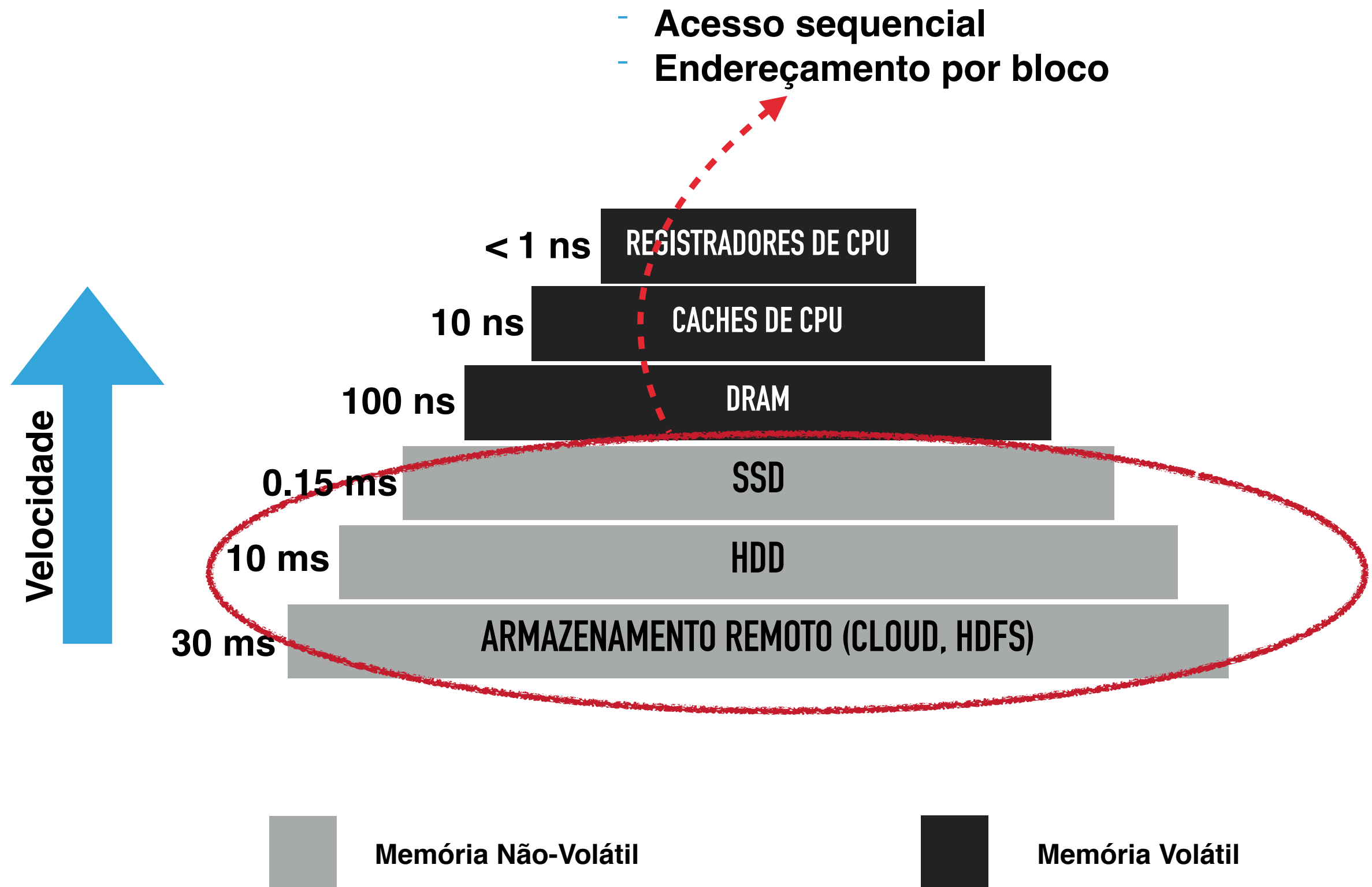


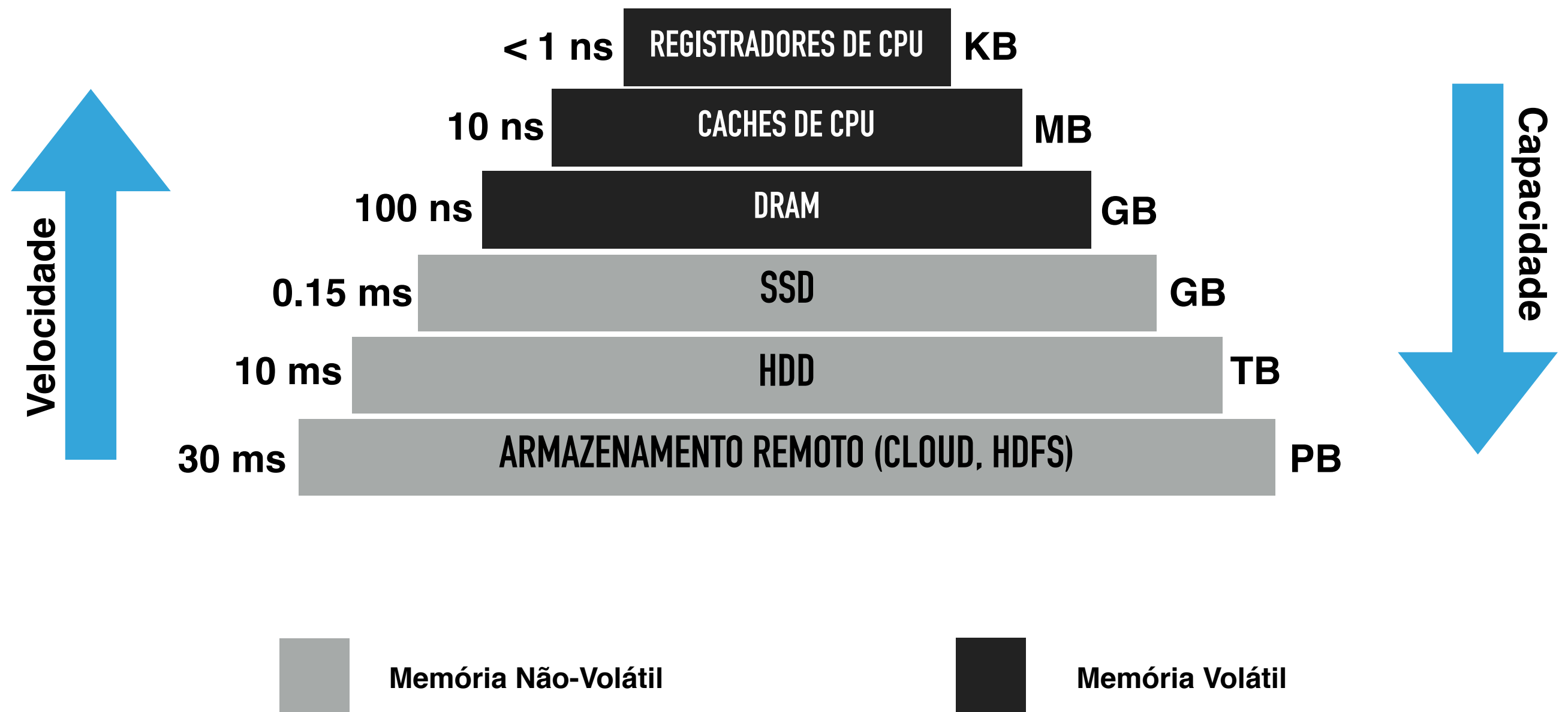
Memória Não-Volátil



Memória Volátil







ROW-STORES

PAGINAS DE BANCO DE DADOS

- ▶ Bloco de dados de tamanho fixo (unidade minima de armazenamento)
- ▶ Contém: tuplas, metadados, indices, logs.
- ▶ Tamanhos de Página:
 - ▶ de hardware: 4KB
 - ▶ de SO: 4KB
 - ▶ de BD: até 16KB



PAGINAS DE BANCO DE DADOS

- ▶ Bloco de dados de tamanho fixo (unidade minima de armazenamento)
- ▶ Contém: tuplas, metadados, índices, logs.
- ▶ Tamanhos de Página:
 - ▶ de hardware: 4KB
 - ▶ de SO: 4KB
 - ▶ de BD: até 16KB

Tamanho padrão



4 KB

ORACLE®

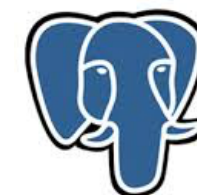
IBM

DB2

8 KB



Microsoft®
SQL Server®



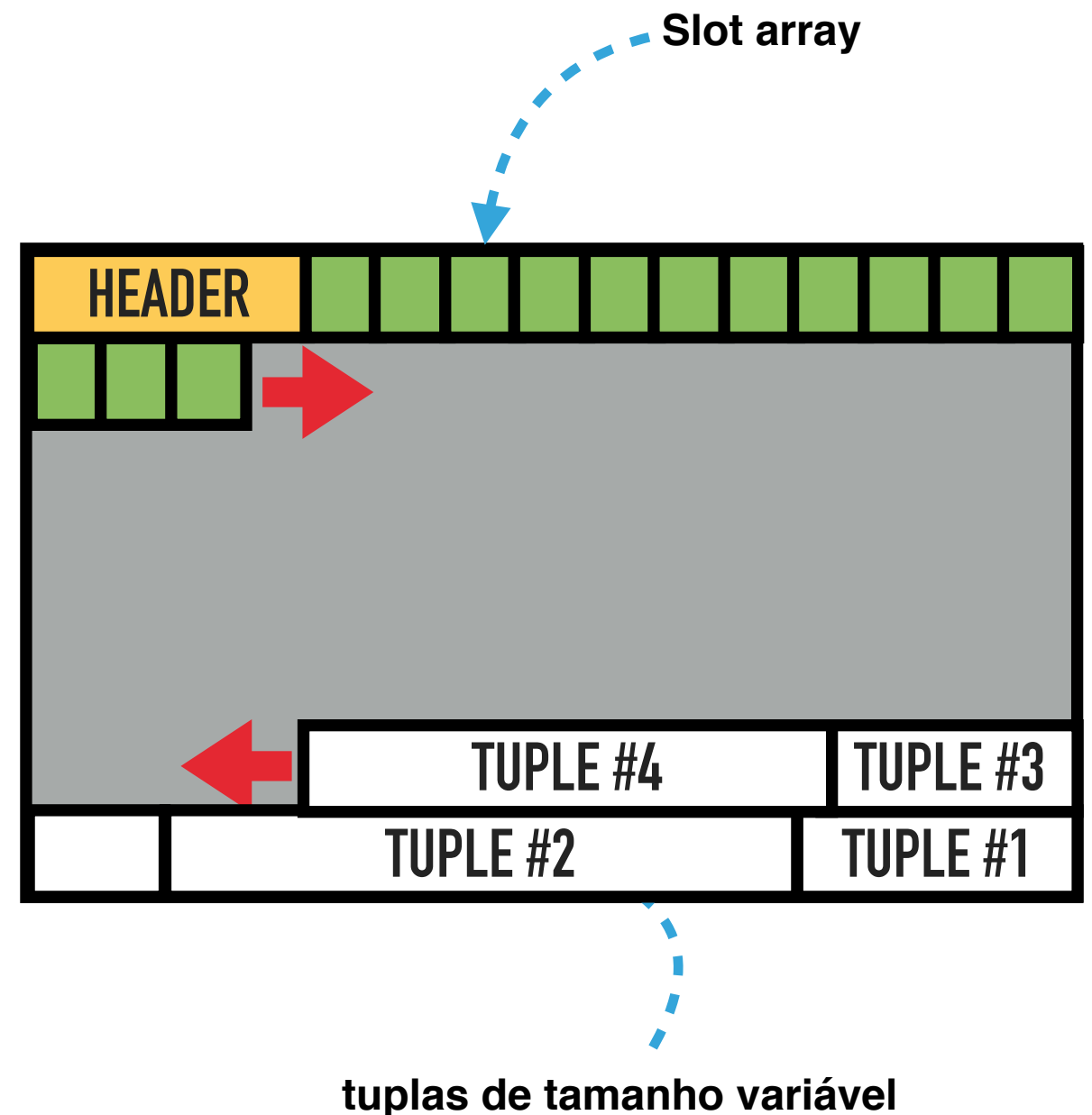
PostgreSQL

16 KB



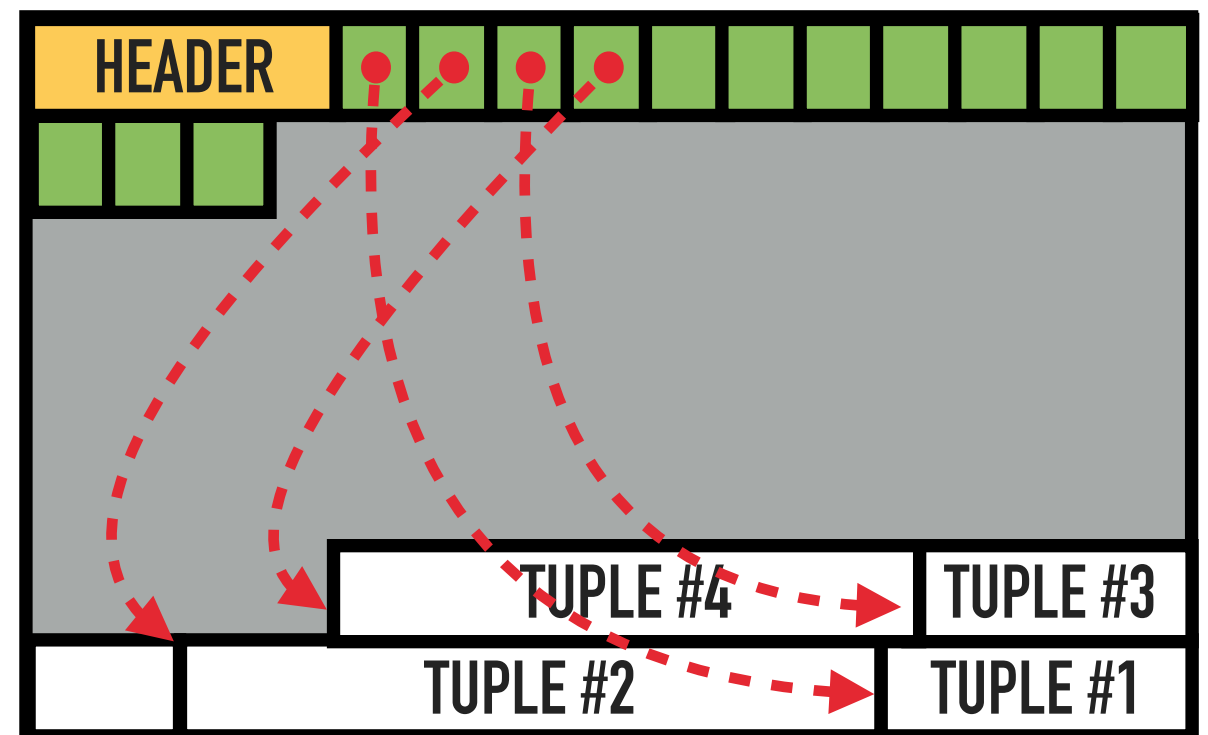
LAYOUT DE UMA PAGINA: N-ARY STORAGE MODEL (NSM)

- ▶ Unidade mínima de armazenamento
- ▶ Padrão: slotted page
- ▶ Header mantém numero de slots e offset dos slots



LAYOUT DE UMA PAGINA: N-ARY STORAGE MODEL (NSM)

- ▶ Unidade mínima de armazenamento
- ▶ Padrão: slotted page
- ▶ Header mantém numero de slots e offset dos slots
- ▶ Página direciona slots para offsets



N-ARY STORAGE MODEL (NSM) – ROW STORE

Vantagens:

- ▶ Inclusões, exclusões e atualizações rápidas
- ▶ Bom para consultas que trazem toda a tupla

Desvantagens:

- ▶ Grandes varreduras de dados
- ▶ Consulta de subconjuntos de atributos

OBJECT-STORES

1990 – BD ORIENTADO A OBJETO

Benefícios

- ▶ Suportam acesso direto aos objetos das linguagens de programação
- ▶ Converte automaticamente os tipos de objetos (evita mapeamento O-R)
- ▶ Ainda existem em outras formas (XML, JSON)



MODELO ORIENTADO A OBJETO -> RELACIONAL

RELACIONAL



APLICAÇÃO

```
class Clientes{
    int        cpf;
    String     nome;
    String     email;
    Date       idade;
    Contas     cnt[ ];
};
```

MODELO ORIENTADO A OBJETO -> RELACIONAL

RELACIONAL

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000

CPF	Agencia	Conta
123	3200	123-45
123	3200	333-90

APLICAÇÃO

```
class Clientes{  
    int        cpf;  
    String     nome;  
    String     email;  
    Date       idade;  
    Contas     cnt[ ];  
};
```

MODELO ORIENTADO A OBJETO -> RELACIONAL

RELACIONAL

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000

CPF	Agencia	Conta
123	3200	123-45
123	3200	333-90

APLICAÇÃO

```
class Clientes{  
    int        cpf;  
    String     nome;  
    String     email;  
    Date       idade;  
    Contas     cnt[ ];  
};
```

MODELO ORIENTADO A OBJETO

JSON

```
{
  "cpf": 123,
  "nome": "JOSE",
  "email": "jose@ufpr",
  "idade": "10/10/2000",
  "contas": [
    {
      "agencia": 3200,
      "conta": 123-45,
    },
    {
      "agencia": 3200,
      "conta": 333-90,
    }
  ]
}
```

APLICAÇÃO

```
class Clientes{
    int        cpf;
    String     nome;
    String     email;
    Date       idade;
    Contas     cnt[ ];
};
```

MODELO ORIENTADO A OBJETO

JSON

```
{
  "cpf": 123,
  "nome": "JOSE",
  "email": "jose@ufpr",
  "idade": "10/10/2000",
  "contas": [
    {
      "agencia": 3200,
      "conta": 123-45,
    },
    {
      "agencia": 3200,
      "conta": 333-90,
    }
  ]
}
```

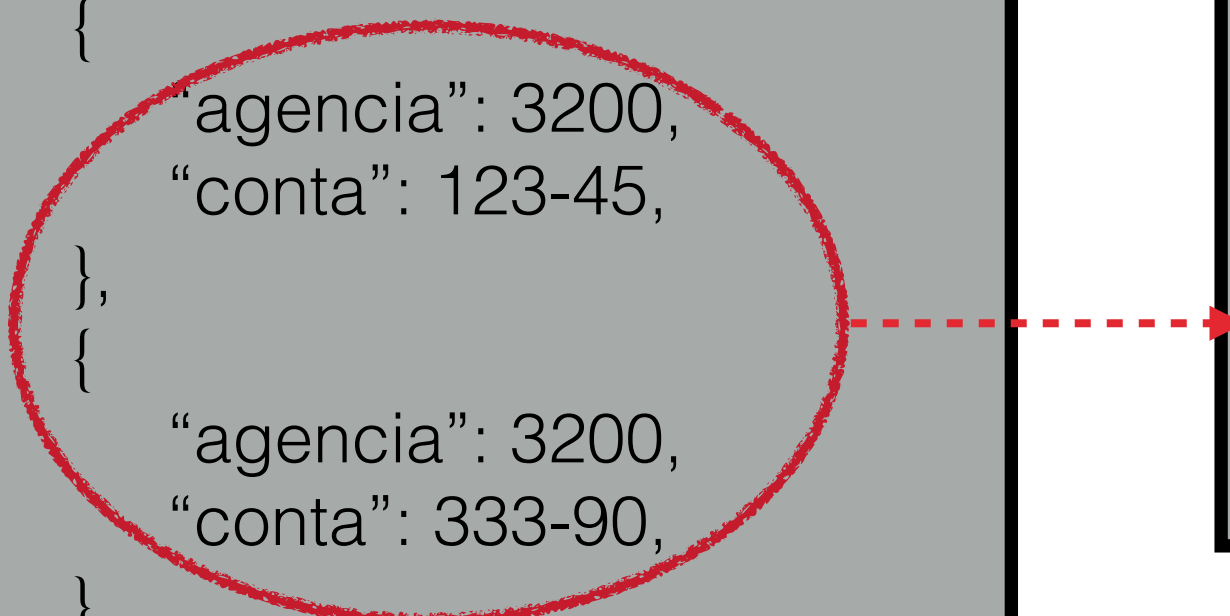
APLICAÇÃO

```
class Clientes{
  int      cpf;
  String   nome;
  String   email;
  Date     idade;
  Contas[] contas;
};
```

MODELO ORIENTADO A OBJETO

JSON

```
{
  "cpf": 123,
  "nome": "JOSE",
  "email": "jose@ufpr",
  "idade": 10/10/2000,
  "contas": [
    {
      "agencia": 3200,
      "conta": 123-45,
    },
    {
      "agencia": 3200,
      "conta": 333-90,
    }
  ]
}
```



APLICAÇÃO

```
class Clientes{
  int      cpf;
  String   nome;
  String   email;
  Date     idade;
  Contas   cnt[ ];
};
```



CODIGO OBJETO EM ARMAZENAMENTO RELACIONAL

RELACIONAL

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000

CPF	Agencia	Conta
123	3200	123-45
123	3200	333-90

JSON

```
{
  "cpf": 123,
  "nome": "JOSE",
  "email": "jose@ufpr",
  "idade": "10/10/2000",
  "contas": [
    {
      "agencia": 3200,
      "conta": "123-45",
    },
    {
      "agencia": 3200,
      "conta": "333-90",
    }
  ]
}
```

RELACIONAL

```
SELECT AVG(IDADE)  
FROM CLIENTES;
```

JSON

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000

CPF	Agencia	Conta
123	3200	123-45
123	3200	333-90

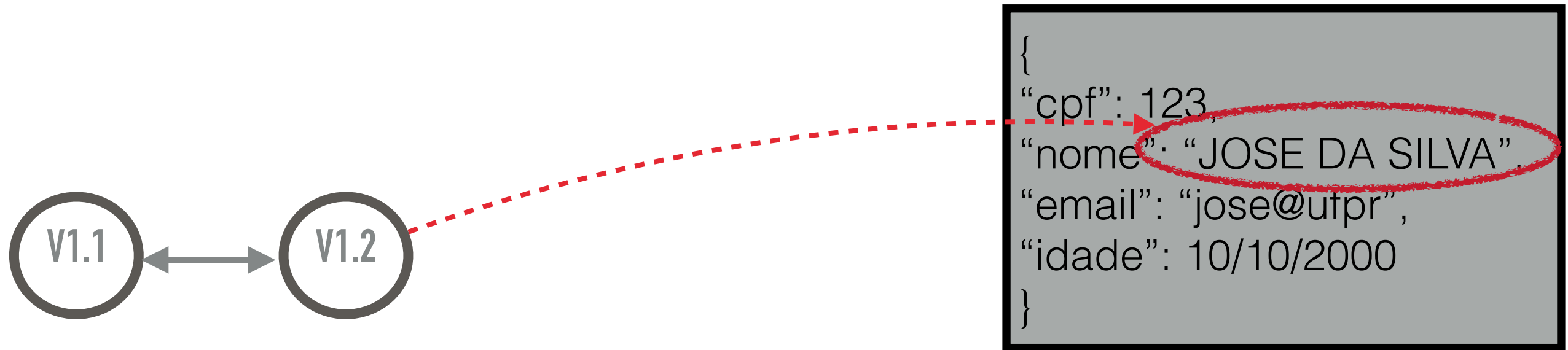
```
{  
  "cpf": 123,  
  "nome": "JOSE",  
  "email": "jose@ufpr",  
  "idade": 10/10/2000,  
  "contas": [  
    {  
      "agencia": 3200,  
      "conta": 123-45,  
    },  
    {  
      "agencia": 3200,  
      "conta": 333-90,  
    }  
  ]  
}
```

PERMITE VERSIONAMENTO DE OBJETOS

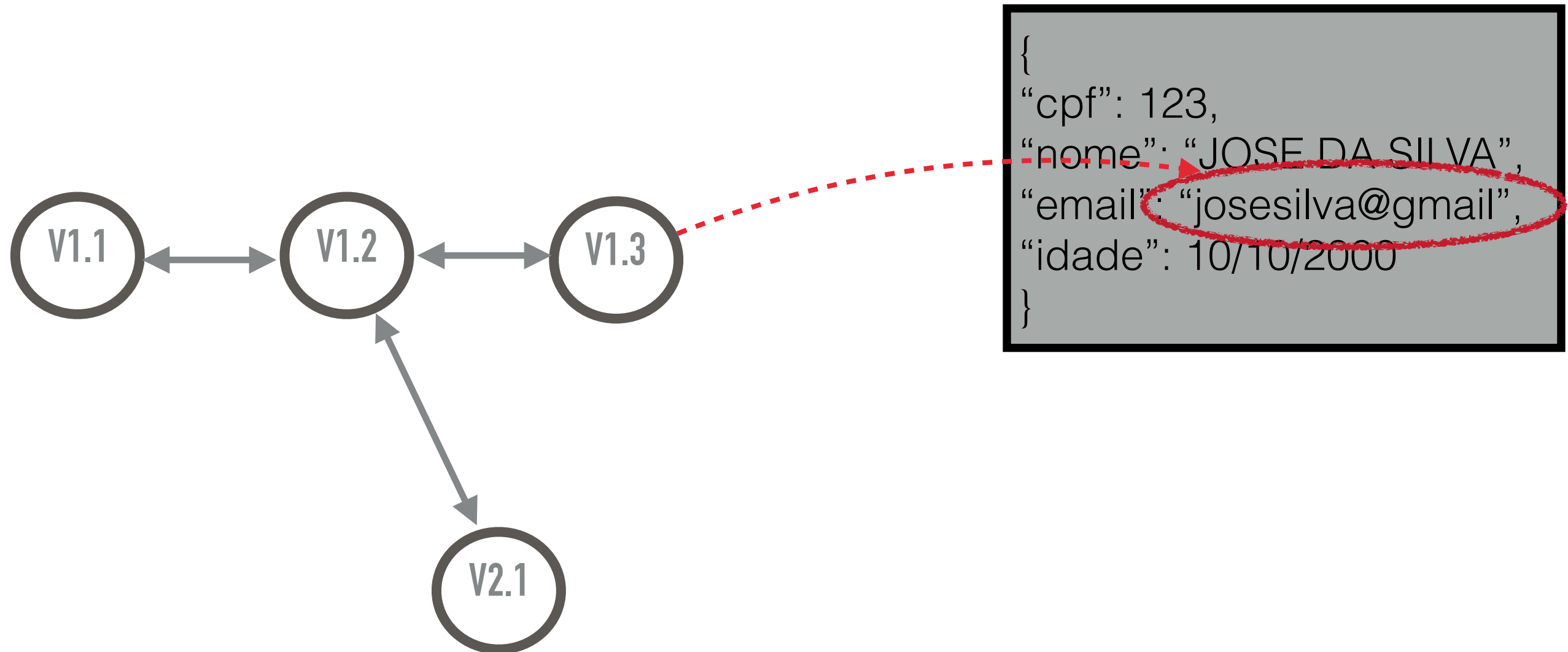
V1.1

```
{  
  "cpf": 123,  
  "nome": "JOSE",  
  "email": "jose@ufpr",  
  "idade": 10/10/2000  
}
```

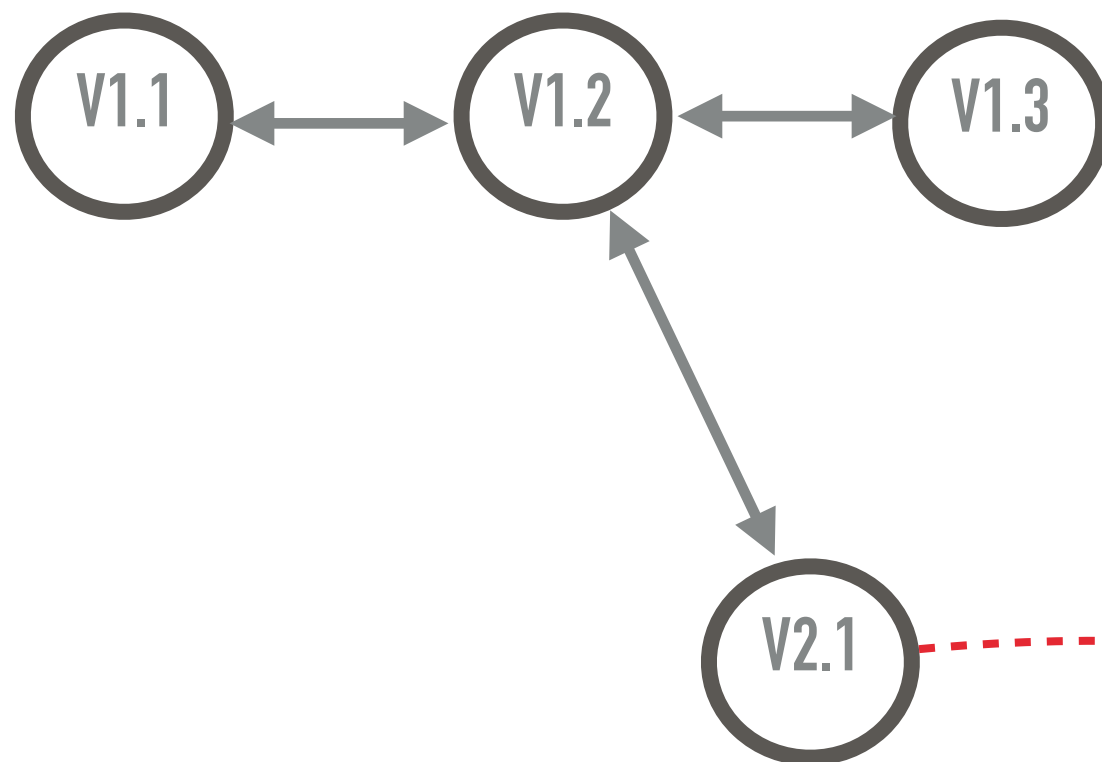
PERMITE VERSIONAMENTO DE OBJETOS



PERMITE VERSIONAMENTO DE OBJETOS



PERMITE VERSIONAMENTO DE OBJETOS



```
{  
  "cpf": 123,  
  "nome": "JOSE DA SILVA",  
  "email": "josesilva@gmail",  
  "idade": 10/10/2000  
}
```

```
{  
  "cpf": 123,  
  "nome": "JOSE DA SILVA",  
  "email": "josesilva@gmail",  
  "idade": 10/10/2000,  
  "fone": [9999,9900,9898]  
}
```

BANCO DE DADOS ORIENTADO A OBJETO

Vantagens:

- ▶ Evita problemas de conversão de tipos
- ▶ Armazena e acessa objetos diretamente

Desvantagens:

- ▶ Desenvolvimento complexo de consultas
- ▶ Sem API padrão de mercado
- ▶ Tipos complexos: video, audio, etc

COLUMN-STORES

2000 – DATA WAREHOUSE

- ▶ Ascensão dos BDs OLAP
- ▶ Armazenamento contíguo de atributos de uma tabela (partic. vertical)



Amazon **Redshift**



DATA WAREHOUSE EM NSM

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000
234	MARIA	NULL	10/03/1970
345	ROSA	rosa@yahoo	01/01/1980
...	...	...	...

```
SELECT AVG(IDADE)  
FROM CLIENTES;
```

DATA WAREHOUSE EM NSM

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000
234	MARIA	NULL	10/03/1970
345	ROSA	rosa@yahoo	01/01/1980
...	...	...	...

**SELECT AVG(IDADE)
FROM CLIENTES,**

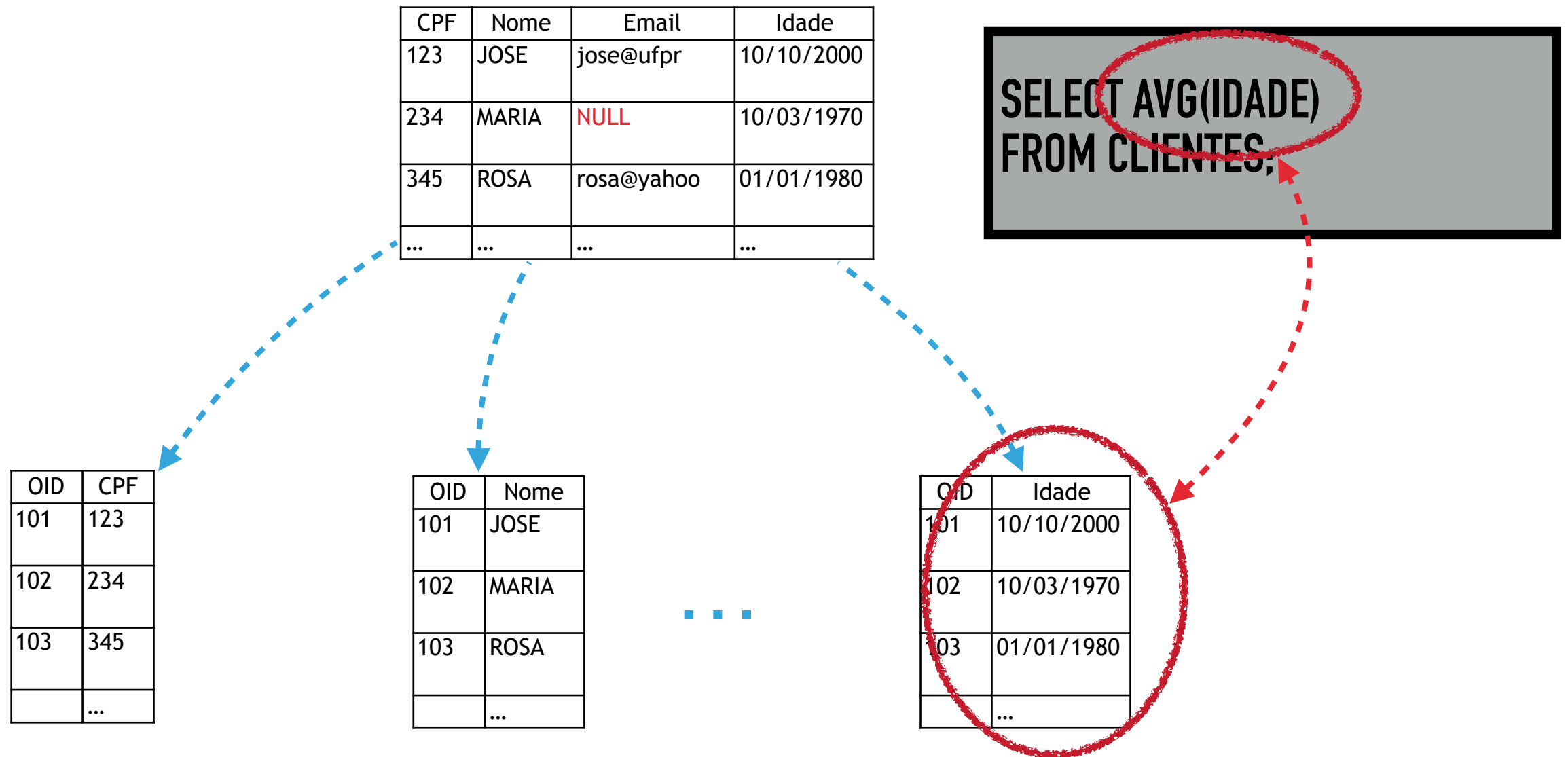
DATA WAREHOUSE EM NSM

CPF	Nome	Email	Idade
123	JOSE	jose@ufpr	10/10/2000
234	MARIA	NULL	10/03/1970
345	ROSA	rosa@yahoo	01/01/1980
...	...	...	...

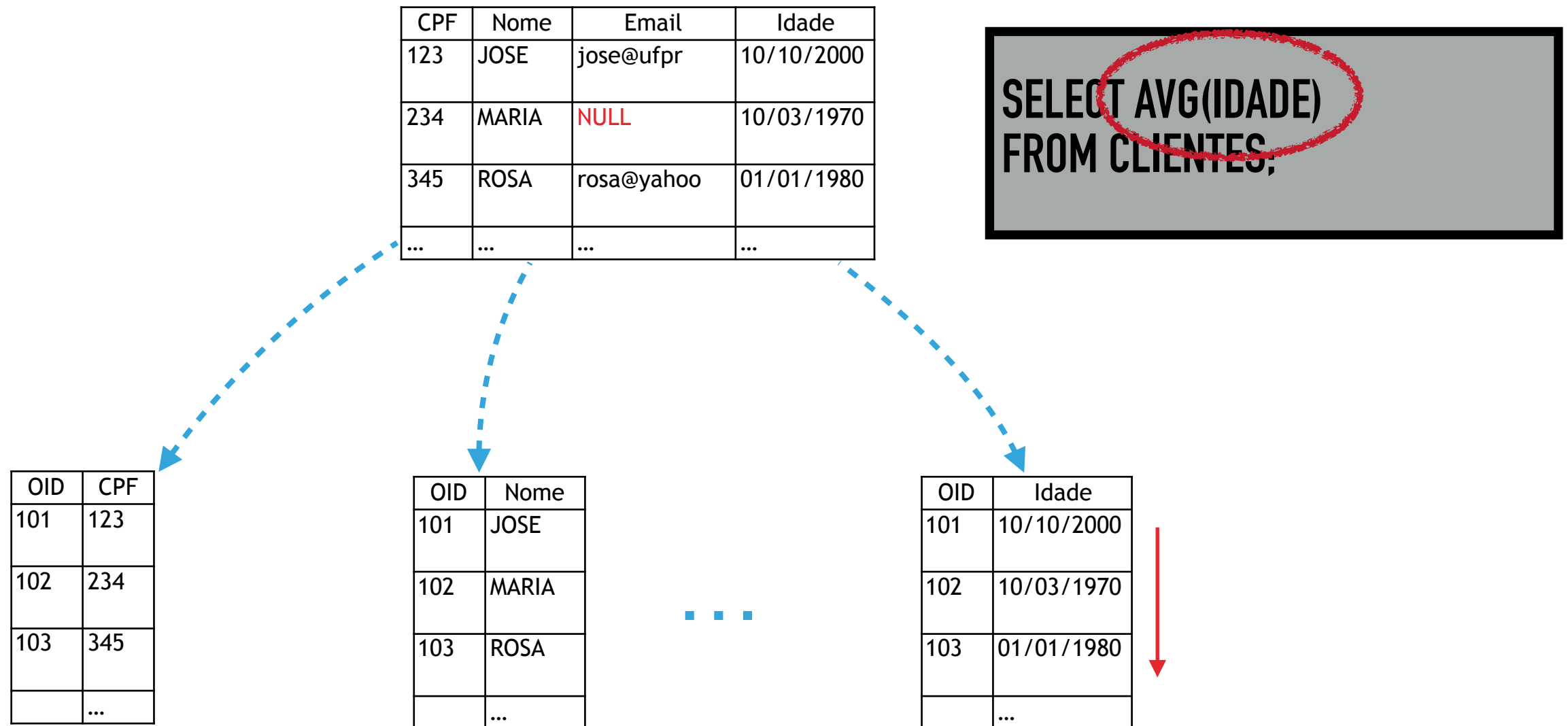
```
SELECT AVG(IDADE)  
FROM CLIENTES,
```

- ▶ Leitura de atributos desnecessários
- ▶ Difícil compressão de dados

DECOMPOSITION STORAGE MODEL (DSM) – COLUMN STORE



DECOMPOSITION STORAGE MODEL (DSM) – COLUMN STORE



COLUMN-STORE VS. ROW-STORE

- ▶ Carregamento de 6.001.215 tuplas
- ▶ PostgreSQL 9.4.18
- ▶ MonetDB
- ▶ Debian Linux 3.16.39



```
curso=# copy lineitem from '/var/lib/postgresql/tpch/appendix/dbgen/lineitem.tbl' with DELIMITER '|';  
COPY 6001215  
Time: 38308.810 ms
```



```
sql>COPY INTO lineitem from '/var/lib/postgresql/tpch/appendix/dbgen/lineitem.tbl' USING DELIMITERS '|','\n';  
6001215 affected rows (17.5s)
```

COLUMN-STORE VS. ROW-STORE

- ▶ TPC-H Q6: Previsão de alteração de receita (ultima de 5 execuções)
- ▶ Tabela lineitem: 16 colunas

```
SELECT
    SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS
    REVENUE
FROM
    LINEITEM
WHERE
    L_SHIPDATE >= '1994-01-01'
    AND L_SHIPDATE < '1995-01-01'
    AND L_DISCOUNT BETWEEN 0.05 AND 0.07
    AND L_QUANTITY < 24;
```



```
revenue
-----
123141078.2283
(1 row)

real      0m1.821s
user      0m0.040s
sys       0m0.004s
```



```
+-----+
| revenue |
+=====+
| 123141078.228300 |
+-----+
1 tuple (140.300ms)
```

DECOMPOSITION STORAGE MODEL (DSM) – COLUMN STORE

Vantagens:

- ▶ Leitura somente dos atributos necessários
- ▶ Compressão (colunas de mesmo tipo)

Desvantagens:

- ▶ Lento para atualizações
- ▶ Lento para consultas pontuais (reconstrução da tupla)

VARIADOS

2010 - NOSQL

- ▶ Formato variado de armazenamento (chave/valor, documento)
- ▶ Ideal para aplicações específicas (blog, email, proc. de texto)
- ▶ Cuidado com transações!!!
- ▶ Sem esquema
- ▶ Sem padrão de linguagem



COLUMN-STORE VS. ROW-STORE VS. NOSQL



```
2018-07-13T20:07:03.638-0300    imported 6001215 documents
2018-07-13T20:07:03.910-0300    connected to: localhost
2018-07-13T20:07:03.917-0300    imported 25 documents

real    2m56.350s
user    3m50.198s
sys     0m14.304s
```



```
curso=# copy lineitem from '/var/lib/postgresql/tpch/appendix/dbgen/lineitem.tbl' with DELIMITER '|';
COPY 6001215
Time: 38308.810 ms
```



```
sql>COPY INTO lineitem from '/var/lib/postgresql/tpch/appendix/dbgen/lineitem.tbl' USING DELIMITERS '|','\n';
6001215 affected rows (17.5s)
```

COLUMN-STORE VS. ROW-STORE VS. NOSQL

```
DB.LINEITEM.FIND(
{
  "SHIPDATE":{
    "$GTE": "1994-01-01",
    "$LT": "1995-01-01",
  }, ...
```



```
real    0m21.114s
user    0m9.056s
sys     0m0.606s
```

```
SELECT
  SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS
  REVENUE
FROM
  LINEITEM
WHERE
  L_SHIPDATE >= '1994-01-01'
  AND L_SHIPDATE < '1995-01-01'
  AND L_DISCOUNT BETWEEN 0.05 AND 0.07
  AND L_QUANTITY < 24;
```



```
revenue
-----
123141078.2283
(1 row)

real    0m1.821s
user    0m0.040s
sys     0m0.004s
```



```
+-----+
| revenue |
+=====+
| 123141078.228300 |
+-----+
1 tuple (140.300ms)
```

2010

- ▶ Sistemas híbridos (OLTP/OLAP)
- ▶ Embarcados
- ▶ Séries Temporais
- ▶ IoT
- ▶ Cloud
- ▶ ...



Cloud Spanner



TIMESCALE

