



MBA em Inteligência Artificial e Big Data
– Curso 3: Administração de Dados Complexos em Larga Escala –
★ **Redução de Dados** ★

Caetano Traina Júnior

Grupo de Bases de Dados e Imagens – GBdl
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos

Apresenta-se técnicas de **Redução de dados**
para agilizar tarefas de mineração em grandes volumes de dados.





Cada tarefa de Mineração de Dados tem seus **requisitos** sobre como devem ser os dados para produzir boas análises de dados:

- Volume de dados;
- Balanceamento entre as classes;
- Resolução dos tipos de dados;
- Escala das dimensões;
- Correlação entre medidas; ...

Aqui apresentamos técnicas para **Redução de dados** úteis para adequar os dados para atender aos requisitos necessários.



Roteiro



- 1 Conceitos básicos em Redução de Dados
- 2 Técnicas de Redução da Cardinalidade
- 3 Técnicas de Redução da Dimensionalidade
- 4 Técnicas de compressão de valores de atributos



Roteiro



- 1 Conceitos básicos em Redução de Dados
- 2 Técnicas de Redução da Cardinalidade
- 3 Técnicas de Redução da Dimensionalidade
- 4 Técnicas de compressão de valores de atributos



Conceitos básicos em Redução de Dados



- Volumes de dados muito grandes levam à:

- 👉 Alto custo de processamento
- 👉 Muita redundância
- 👉 Maldição da dimensionalidade
- 👉 Aumento do “*overfitting*”
- 👉 Muitas tuplas semelhantes
- 👉 Muitas dimensões correlacionadas
- 👉 Valores muito próximos



Conceitos básicos em Redução de Dados



- Portanto, sempre que possível, é interessante **reduzir os dados**.
- É possível:
 - ★ Reduzir a quantidade de tuplas
 - 👉 **Redução de Cardinalidade – N**
 - ★ Reduzir a quantidade de atributos
 - 👉 **Redução de Dimensionalidade – E**
 - ★ Reduzir a complexidade dos valores dos atributos
 - 👉 **Compressão de Domínio – F**



Conceitos básicos em Redução de Dados



Nesta parte do curso, os conceitos apresentados serão ilustrados usando o gerenciador



Roteiro



- 1 Conceitos básicos em Redução de Dados
- 2 Técnicas de Redução da Cardinalidade
- 3 Técnicas de Redução da Dimensionalidade
- 4 Técnicas de compressão de valores de atributos



Técnicas de Redução da Cardinalidade



- Reduzir a cardinalidade corresponde a executar uma amostragem dos dados, escolhendo um subconjunto das tuplas originais.

Isso pode ser feito com:

- **Amostragem Aleatória** (*unbiased*);
Define-se um fator de amostragem e aleatoriamente escolhe-se a quantidade necessária de amostras para compor a base reduzida.
- **Amostragem Dirigida** (*biased*). Adota-se um critério que cria uma tendência para escolher as tuplas que irão compor a base reduzida.
As técnicas mais importantes para Amostragem Dirigida são:
 - Baseadas em Histogramas;
 - Baseadas em Classes;
 - Baseadas em Densidade (proximidade).





Amostragem Aleatória

Dado um **fator de amostragem**, definido por:

☞ uma **taxa de amostragem** ou ☞ uma **quantidade de amostras**,

escolhem-se tuplas **aleatoriamente** até atingir a quantidade necessária.

- São comuns taxas de 10% até 0,01% de amostras (uma a cada dez até uma a cada 10 mil tuplas), ou mais.
- Nem sempre é útil em bases de dados muito grandes.

Vantagens:

- é rápida;
- é independente da cardinalidade N : $O(1)$

Desvantagens:

- Não preserva *outliers*;
- Sensível a desbalanceamento do conjunto (por exemplo, quando uma classe é muito maior do que outras);
- Reduz a sensibilidade dos processos posteriores;



Tratamento de Amostragem Aleatória em SQL



☞ SQL oferece recursos para **amostragem aleatória**:

- Usando a função de geração de números aleatórios `Random()` para selecionar tuplas:

Função para geração de números aleatórios em SQL

`Random()` - Gera valor aleatório no intervalo `[0.0, 1.0]`
Em distribuição uniforme

`SetSeed(REAL)` - Define a semente para as chamadas subsequentes da função `Random()`.
Recebe valor aleatório em `[0.0, 1.0]`

- Pela cláusula `TABLESAMPLE`;



Tratamento de Amostragem Aleatória em SQL



Geração de números aleatórios em SQL –



Exemplo de geração de números aleatórios:

- Um número:

```
SELECT Random();
```

random
0.639387583267645

- Diversos números:

```
SELECT I AS Seq, Random() AS Valor  
FROM GENERATE_SERIES(1, 100) WITH ORDINALITY I;
```


Seq	Valor
1	0.43826179184539527
2	0.8738957373219449
3	0.04871207732159277
4	0.2614364731611758
5	0.15839698835705818
...	
100	0.709404809089428



Tratamento de Amostragem Aleatória em SQL



Geração de números aleatórios em SQL – PostgreSQL

Além da função `Random()`, que é definida pelo padrão ,

PostgreSQL tem uma função para geração de números aleatórios **em distribuição normal**.

- Para isso é necessário usar o módulo `tablefunc`:

```
CREATE EXTENSION tablefunc;
```

Função para geração de números aleatórios em **distribuição normal**

`Normal_Rand(N, Avg, Sd)` – Gera valor aleatório normal

N – Quantidade de tuplas a ser gerada;

Avg – Valor da média da distribuição gerada;

Sd – Desvio padrão da distribuição gerada;

Exemplo: Gerar 100 números com média 5 e desvio padrão 2:

```
SELECT Row_Number() OVER () AS Seq, Valor
FROM Normal_Rand(100, 5, 2) AS Valor
ORDER BY 1;
```

Seq	Valor
1	5.402137213668504
2	3.535495275475742
3	5.367789771245657
4	4.2659448308660055
...	
100	7.209404809089428

Tratamento de Amostragem Aleatória em SQL

Geração de números aleatórios em SQL



👉 É importante avaliar o **tempo** gasto pelos comandos:

```
SELECT Row_Number() OVER () AS Seq, Valor
FROM Normal_Rand(100, 5, 2) AS Valor
ORDER BY 1;
```

👉 Isso pode ser feito usando **EXPLAIN** <comando> ou **EXPLAIN ANALYZE** <comando>

Seq	Valor
1	Sort (cost=72.33..74.83 rows=1000 width=16) (actual time=0.078..0.082 rows=100 loops=1)
2	Sort Key: (row_number() OVER (?))
3	Sort Method: quicksort Memory: 29kB
4	-> WindowAgg (cost=0.00..22.50 rows=1000 width=16) (actual time=0.029..0.058 rows=100 loops=1)
5	-> Function Scan on normal_rand valor (cost=0.00..10.00 rows=1000 width=8) (actual time=0.023..0.028 rows=100 loops=1)
6	Planning Time: 0.065 ms
7	Execution Time: 0.119 ms



Tratamento de Amostragem Aleatória em SQL



Geração de números aleatórios em SQL –



👉 Tempo para gerar 1.000.000 de números e calcular a média e o desvio padrão da população gerada:

```
EXPLAIN ANALYZE SELECT Count(*), Avg(Valor), StdDev(Valor)
FROM Normal_Rand(1000000, 5, 2) AS Valor; -- Um milhao.
```

count	avg	stddev
1000000	4.99863009049008	1.9983642516850448

Seq	Valor
1	Aggregate (cost=17.51..17.52 rows=1 width=24) (actual time=367.091..367.092 rows=1 loops=1)
2	-> Function Scan on normal_rand valor (cost=0.00..10.00 rows=1000 width=8) (actual time=163.987..266.777 rows=1000000 loops=1)
3	Planning Time: 0.077 ms
4	Execution Time: 370.706 ms

Tratamento de Amostragem Aleatória em SQL

Cláusula **TABLESAMPLE** em SQL



Sintaxe geral da cláusula **TABLESAMPLE** em SQL

(Padrão ISO-SQL-2003)

```
SELECT <atributos>
  FROM <tabela>
      TABLESAMPLE <método> (<argumento> [, ...])
                      [REPEATABLE <semente>]
  ...
```



- Onde:
 - <método> pode ser BERNOULLI ou SYSTEM (pelo padrão, pelo menos)
 - <argumento> é dependente do método (porcentagem de 1.0 a 100.0)
 - <semente> é o valor de inicialização da sequência de aleatórios.
- O objetivo da cláusula TABLESAMPLE é a **execução rápida** de uma amostragem, sem grandes compromissos com as **propriedades de amostragem** no conjunto gerado.




Tratamento de Amostragem Aleatória em SQL



Cláusula **TABLESAMPLE** em  PostgreSQL e  ORACLE

- Tanto  PostgreSQL quanto  ORACLE permitem usar os dois tipos padronizados de métodos de amostragem:

`<sampling_method>= BERNOULLI ou SYSTEM`

( ORACLE: `SAMPLE` ou `SAMPLE BLOCK`).

BERNOULLI – Equivalente a:

`SELECT * FROM <tabela> WHERE 100*Random() < Argumento;`

- Volta uma quantidade mais correta das tuplas pedidas,
- mas é mais lento (apesar de usar *bitmap* para gerar os RowId).
- **É útil para tabelas não muito grandes.**

SYSTEM – Lê a porcentagem especificada de páginas da tabela e retorna todas as suas tuplas.

- É bem mais rápido, mas:
- volta uma quantidade aproximada de tuplas;
- pode ser tendencioso se houver tendência na armazenagem das tuplas.
- **Deve ser usado apenas para tabelas grandes.**



Tratamento de Amostragem Aleatória em SQL

Cláusula **TABLESAMPLE** em SQL



Exemplo para particionar uma tabela T em dois subconjuntos de tuplas:
 T_R para treino e outro T_E para teste:

```
ALTER TABLE Alunos
  ADD COLUMN Separa CHAR -- 'R' → Treino, 'E' → Teste
  NOT NULL DEFAULT 'E';

UPDATE Alunos TABLESAMPLE BERNOULLI (25.)
  SET Separa='R'
  REPEATABLE(1234);
```



Tratamento de Amostragem Aleatória em SQL

Cláusula TABLESAMPLE em SQL



- Nem sempre a cláusula TABLESAMPLE é uma solução **adequada**, porque:
 - Não existe alternativa para ter um controle mais fino das propriedades das amostragens geradas.
 - Não existe uma cláusula NOT IN TABLESAMPLE
 - 👉 Por exemplo, para dividir a tabela em **um** conjunto de treino e **vários** de teste.
- **mas é possível explorar outras alternativas com comandos simples usando as funções de geração de aleatórios, para gerar amostragens mais “controladas”.**
- **Todas elas tendem a ser mais lentas do que a cláusula TABLESAMPLE.**
- **Mas elas ainda são, na maioria das vezes, mais rápidas do que usar ferramentas externas.**



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Exemplo: Retornar 10% das tuplas.

- Alternativa 1:

```
SELECT *  
  FROM Aluno  
 WHERE Random() < .10;
```

- O conjunto não tem repetição,
- mas pode não ter exatamente 10% das tuplas.
- Requer um *table scan* sobre toda a tabela.



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Exemplo: Retornar uma quantidade predefinida de tuplas (p.ex. $k=1000$).

- Alternativa 2:

```
SELECT *  
  FROM Aluno  
 ORDER BY Random()  
  LIMIT 1000;
```

- O conjunto não tem repetição, e tem exatamente a quantidade de tuplas pedida.
 - Requer um *table scan* de toda a tabela, mais a ordenação dos atributos aleatórios! \Rightarrow Complexidade $O(N + N \cdot \log N)$
- ★ Mas é possível adotar medidas para melhorar substancialmente o custo:



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Exemplo: Retornar uma quantidade predefinida de tuplas (p.ex. $k=1000$).

- Obter uma sobre-amostragem com p' pouco maior do que a taxa p desejada: por exemplo recuperar 20% a mais do que a taxa de amostragem. Se $p = 0,01\%$, pode-se recuperar a fração $p' = p + 20\% = 0,01 \cdot 1,20\% = 0,012$ da tabela:
- Alternativa 3:

```
SELECT *  
  FROM Aluno  
 WHERE Random() < 0.012  
 LIMIT 1000;
```

- Quanto maior o valor de sobre-amostragem, menor a chance da cláusula WHERE produzir menos de k tuplas,
- mas também prejudica mais a aleatoriedade do resultado
- e mais lento o comando fica.



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas

Vamos avaliar as três alternativas.

- Primeiro, criar uma tabela de teste

```
CREATE TABLE Teste(  
    Id      INT,  
    Dados  NUMERIC DEFAULT random()*1000  
);  
  
INSERT INTO Teste  
    SELECT * FROM Generate_Series(1, 1000000);
```

- Qual a quantidade de páginas ocupadas em disco por essa tabela?
(páginas de 8 KBytes)

```
VACUUM ANALYZE Teste;  
  
SELECT RelName, RelTuples, RelPages, Pg_Relation_Size(0Id)  
    FROM pg_Class WHERE RelName='teste';
```

RelName	RelTuples	RelPages	RelSize
teste	1.000.000	5.489	44.965.888



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Avaliando as três alternativas.

```
EXPLAIN ANALYZE SELECT * FROM Teste
      WHERE 100*Random() < .10;
```

```
Seq Scan on teste (cost=0.00..22989.00 rows=333333 width=15)
      (actual time=0.023..62.210 rows=1046 loops=1)
    Filter: (('100'::double precision * random()) < '0.1')
    Rows Removed by Filter: 998954
Planning Time:  0.045 ms
Execution Time: 62.233 ms
```



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Avaliando as três alternativas.

```
EXPLAIN ANALYZE SELECT * FROM Teste  
ORDER BY Random() LIMIT 1000;
```

```
Limit (cost=72817.92..72820.42 rows=1000 width=23)  
  (actual time=202.708..202.851 rows=1000 loops=1)  
    -> Sort (cost=72817.92..75317.92 redrows=1000000 width=23)  
        (actual time=202.707..202.753 rows=1000 loops=1)  
        Sort Key: (random())  
        Sort Method: top-N heapsort Memory: 179kB  
        -> Seq Scan on teste (cost=0.00..17989.00 rows=1000000 width=23)  
            (actual time=0.011..85.980 rows=1000000 loops=1)  
Planning Time: 0.084 ms  
Execution Time: 202.967 ms
```



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



```
EXPLAIN ANALYZE SELECT * FROM Teste  
WHERE Random() < 0.012 LIMIT 1000;
```

```
Limit (cost=0.00..61.47 rows=1000 width=15)  
  (actual time=0.010..4.124 rows=1000 loops=1)  
    -> Seq Scan on teste (cost=0.00..20489.00 rows=333333 width=15)  
      (actual time=0.008..4.103 rows=1000 loops=1)  
        Filter: (random() < '0.012'::double precision)  
        Rows Removed by Filter: 79856  
Planning Time: 0.033 ms  
Execution Time: 4.142 ms
```



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Exemplo: Particionar a tabela a ser processada em um **conjunto de treino** mais **10 conjuntos de teste**.

- A tabela será particionada em 11 subconjuntos.
- Uma alternativa é associar um novo atributo, com o valor de 0 a 10, sendo um deles (**digamos '0'**) indicando o **conjunto de treino**.
 - Esse atributo pode ser **aleatório**
☞ mas não é repetitivo, especialmente se a tabela sofrer atualizações.
 - Outra alternativa é usar uma **função de hash** sobre a chave ou qualquer combinação única de atributos da tabela
☞ e portanto pode ser repetitivo).



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas



Função *Hash* para atributos de tipo TEXT em SQL – PostgreSQL

HashText(Text) - Gera um número aleatório de tipo INT4.

MD5(Text) - Calcula o valor *hash* em MD5 do argumento (tipo TEXT) e retorna um TEXT como um valor com 32 dígitos hexadecimais.

```
SELECT HashText('José da Silva'), MD5('José da Silva');
```

HashText	MD5
-1014468627	3edc9937173b6412c33fa988bb7b7ff6



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas

Exemplo: Particionar a tabela a ser analisada (p.ex `Alunos`) em: **um** conjunto de treino mais **dez** de teste.

- O valor do *hash* constitui um novo atributos da tabela).

👉 Ele pode ser acrescentado à tabela:

```
ALTER TABLE Alunos ADD COLUMN SubConj DOUBLE PRECISION;  
  
UPDATE Alunos *  
    SET SubConj=Abs(HashText(Nome) % 11);
```

👉 Mas nesse caso, como o valor desse atributo é imutável, ele nem precisa ser “materializado”: pode ser obtido numa VIEW.

```
CREATE VIEW PreparaAluno AS  
SELECT *, Abs(HashText(Nome) % 11) AS SubConj  
FROM Alunos;
```



Tratamento de Amostragem Aleatória em SQL

Exemplos de Alternativas

Exemplo: Particionar a tabela em um conjunto de treino mais 10 de teste.

- É rápido – acessa as tuplas apenas quando elas são necessárias.
- Não garante a quantidade exata de tuplas por partição

```
CREATE VIEW PreparaAluno AS  
SELECT *, Abs(HashText(NUSP::TEXT) % 11) AS SubConj  
FROM Alunos;  
  
SELECT Subconj, Count(*) FROM PreparaAluno  
GROUP BY Subconj ORDER BY Subconj;
```

SubConj	Count
0	90950
1	91203
2	91121
3	90684
4	91029
5	91021
6	90937
7	90194
8	90926
9	91263
10	90672

Count(*): 80.000 alunos

140ms





Amostragem Dirigida

A Amostragem Dirigida é usada quando se quer ressaltar alguma característica dos dados.

- Compensar o desbalanceamento de classes,
- Enfatizar alguma tendência, ...



Técnicas de Redução da Cardinalidade

Amostragem Dirigida: Histogramas



Amostragem Dirigida Baseada em Histogramas

Nas técnicas de amostragem de casos baseada em histogramas:

- 1 O analista escolhe os atributos mais importantes (ou do ponto de vista do conhecimento que ele tem da aplicação, ou baseado em resultados de processos de DM anteriores),
- 2 A seguir, (se necessário) cria um **processo de discretização** dos atributos contínuos.

Dai:

- 3 Cria-se um **histograma multidimensional** com os atributos escolhidos,
- 4 **Escolhem-se tuplas do conjunto original** de tal maneira que cada *bin* do histograma atenda a determinada propriedade (todos tenham o mesmo número de tuplas, ou tenha um valor mínimo e máximo para a quantidade de tuplas, etc.), **o mais aleatoriamente possível**.

As técnicas mais importantes para os processos envolvidos na **Amostragem Baseada em Histogramas** são discutidas a seguir.



O Processo de Discretização

O processo de discretização deve transformar os valores de um domínio contínuo em um contra-domínio discreto, ou reduzir o conjunto de valores do domínio discreto, de maneira que a quantidade de valores do contra-domínio seja pequeno o suficiente para atender às necessidades da amostragem.

As técnicas mais comuns de Discretização Baseada em Histogramas são:


- Equi-largura (*Equi-width*)
- Equi-altura (*Equi-height*)
- Equi-entropia (*Equi-entropic*)

Elas formam a base para todos os demais processos de amostragem dirigida.





Histogramas em Equi-largura (para um único atributo)

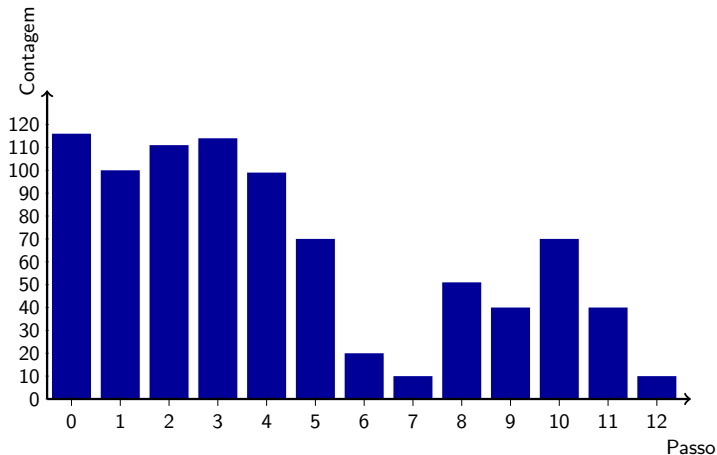
- 1 Encontra-se o menor e o maior valor do domínio do atributo  **Domínio ativo**;
- 2 Divide-se o domínio ativo no número de faixas definido pelo analista, criando-se **faixas de valores de igual largura para o domínio** daquele atributo:
 $faixa(x) \rightarrow \mathbb{N}$ é o número do bin do valor x ;
(opcionalmente pode-se eliminar os k -menores e os k -maiores valores para reduzir distorções);
- 3 Atribui-se a cada tupla o valor do contra-domínio correspondente;
- 4 Gera-se o histograma contando-se o número de tuplas em cada *bin*;
- 5 (opcionalmente pode-se mostrar o histograma ao analista, para controle).

Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura



Histogramas em Equi-largura têm passo constante:



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura



Exemplo (1.1) em SQL: Histograma de **Idades** da tabela **Alunos**

```
SELECT Idade, Count(*) AS Conta
FROM Aluno
GROUP BY Idade
ORDER BY Idade;
```

Resultado:

Idade	Conta
19	1
20	1
21	3
22	2
23	1
24	1
25	1
27	1
35	1
	2

31ms para 15 alunos
(105ms para 80.000 alunos).

Problema: somente existem *bins* para dados que existem!



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.2) em SQL: Histograma de **Idades** da tabela **Alunos**

Solução para incluir todos os bins:

```
SELECT Bins.B AS Idade,  
       CASE WHEN Tab.Conta IS NULL THEN 0  
            ELSE Tab.Conta END Contagem  
FROM  
  (WITH Lim AS (  
    SELECT Min(Idade) Mi, Max(Idade) Ma  
    FROM Aluno)  
    SELECT Generate_Series(Lim.Mi+1, Lim.Ma-1)  
           AS B FROM Lim) AS Bins  
  LEFT OUTER JOIN  
  (SELECT Idade, Count(*) Conta  
   FROM Aluno  
   GROUP BY Idade) AS Tab  
  ON Bins.B=Tab.Idade;
```

1: Encontrar o **menor** e o **maior** valor do domínio ativo do **atributo**



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.2) em SQL: Histograma de **Idades** da tabela **Alunos**

Solução para incluir todos os bins:

```
SELECT Bins.B AS Idade,  
       CASE WHEN Tab.Conta IS NULL THEN 0  
            ELSE Tab.Conta END Contagem  
FROM  
  (WITH Lim AS (  
    SELECT Min(Idade) Mi, Max(Idade) Ma  
    FROM Aluno)  
    SELECT Generate_Series(Lim.Mi+1, Lim.Ma-1)  
           AS B FROM Lim) AS Bins  
  LEFT OUTER JOIN  
  (SELECT Idade, Count(*) Conta  
   FROM Aluno  
   GROUP BY Idade) AS Tab  
  ON Bins.B=Tab.Idade;
```

2: Criar todos os valores na **faixa do domínio ativo** (remove k -menores e k -maiores)



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.2) em SQL: Histograma de **Idades** da tabela **Alunos**

Solução para incluir todos os bins:

```
SELECT Bins.B AS Idade,  
       CASE WHEN Tab.Conta IS NULL THEN 0  
            ELSE Tab.Conta END Contagem  
FROM  
  (WITH Lim AS (  
    SELECT Min(Idade) Mi, Max(Idade) Ma  
    FROM Aluno)  
    SELECT Generate_Series(Lim.Mi+1, Lim.Ma-1)  
           AS B FROM Lim) AS Bins  
  LEFT OUTER JOIN  
  (SELECT Idade, Count(*) Conta  
   FROM Aluno  
   GROUP BY Idade) AS Tab  
  ON Bins.B=Tab.Idade;
```

3: Gera-se o histograma **contando o número de tuplas** em cada bin



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.2) em SQL: Histograma de **Idades** da tabela **Alunos**

Solução para incluir todos os bins:

```
SELECT Bins.B AS Idade,  
       CASE WHEN Tab.Conta IS NULL THEN 0  
            ELSE Tab.Conta END Contagem  
FROM  
  (WITH Lim AS (  
    SELECT Min(Idade) Mi, Max(Idade) Ma  
    FROM Aluno)  
    SELECT Generate_Series(Lim.Mi+1, Lim.Ma-1)  
           AS B FROM Lim) AS Bins  
  LEFT OUTER JOIN  
  (SELECT Idade, Count(*) Conta  
   FROM Aluno  
   GROUP BY Idade) AS Tab  
  ON Bins.B=Tab.Idade;
```

4: Bins sem tuplas ficam com valor zero (não nulo)



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.2) em SQL: Histograma de **Idades** da tabela **Alunos**

Idade	Contagem
19	1
20	1
21	3
22	2
23	1
24	1
25	1
26	0
27	1
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	1

33ms para 15 alunos

62ms para 80.000 alunos

Tuplas com idade **null** não são contadas

(Pode usar **FULL** OUTER JOIN, mas daí não se deve eliminar os extremos)



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.3) em SQL: Histograma de **Idades** da tabela **Alunos**
Agrupar de cinco em cinco:

```
SELECT Floor(Idade/5.00)*5 as Ini, Count(*) AS Conta
FROM Aluno
GROUP BY Ini
ORDER BY Ini;
```

Resultado:

Idade	Conta
15	1
20	8
25	2
35	1
	2

32ms para 15 alunos
170ms para 80.000 alunos



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.4) em SQL: Histograma de **Idades** da tabela **Alunos**

Agrupar em **cinco** bins:

```
WITH MinMax AS (SELECT Min(Idade) Mi, Max(Idade) Ma
                  FROM Aluno)
SELECT Idade,
       Width_bucket(Idade, (SELECT Mi FROM MinMax),
                     (SELECT Ma FROM MinMax), 4) as Bin,
       Count(*) Conta
FROM Aluno
GROUP BY Idade      ORDER BY Idade;
```

Resultado:

Idade	Bin	Conta
19	1	1
20	1	1
21	1	3
22	1	2
23	2	1

24	2	1
25	2	1
27	3	1
35	5	1
		2

32ms para 15 alunos

170ms para 80.000 alunos



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura



Função para dividir números em várias faixas –



`Width_bucket(Valor Real, Ini Real, Fim Real, Count INT) –`

Retorna em qual faixa (bucket) o `Valor` dado está dentro dos números entre `Ini` e `Fim`, dividindo por `Count` pontos de corte (quer dizer, divide em `Count+1` faixas).



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.5) em SQL: Histograma de **Idades** da tabela **Alunos**

Agrupar em **cinco** bins, indicando as faixas explicitamente:

```
WITH MinMax AS
  (SELECT 4 AS NB, Min(Idade) AS Mi, Max(Idade) AS Ma FROM Aluno)
SELECT Trunc((SELECT Mi FROM MinMax)+
  ((Bin-1)*((SELECT Ma FROM MinMax)-(SELECT Mi FROM MinMax))/
    (SELECT NB FROM MinMax)),2) AS Ini,
  Trunc(((SELECT Mi FROM MinMax) +
    (Bin)*((SELECT Ma FROM MinMax)-(SELECT Mi FROM MinMax))/
    (SELECT NB FROM MinMax)),2) AS Fim,    Conta
FROM (
  SELECT Width_Bucket(Idade, (SELECT Mi FROM MinMax),
    (SELECT Ma FROM MinMax), (SELECT NB FROM MinMax)) AS Bin,
    Count(*) as Conta
  FROM Aluno
  GROUP BY Bin          ORDER BY Bin) Histo;
```



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura

Exemplo (1.5) em SQL: Histograma de **Idades** da tabela **Alunos**

Agrupar em **cinco** bins, indicando as faixas explicitamente:

```
WITH MinMax AS
  (SELECT 4 AS NB, Min(Idade) AS Mi, Max(Idade) AS Ma FROM Aluno)
SELECT Trunc((SELECT Mi FROM MinMax)+
  ((Bin-1)*((SELECT Ma FROM MinMax)-(SELECT Mi FROM MinMax))/
    (SELECT NB FROM MinMax)),2) AS Ini,
  Trunc(((SELECT Mi FROM MinMax) +
    (Bin)*((SELECT Ma FROM MinMax)-(SELECT Mi FROM MinMax))/
    (SELECT NB FROM MinMax)),2) AS Fim,    Conta
FROM (
  SELECT Width_Bucket(Idade, (SELECT Mi FROM MinMax),
    (SELECT Ma FROM MinMax), (SELECT NB FROM MinMax)) AS Bin,
    Count(*)
  FROM Aluno
  GROUP BY Bin
```

Resultado:

Ini	Fim	Conta
19	23	7
23	27	3
27	31	1
35	39	1
		2

32ms para 15 alunos



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura



Histogramas em Equi-largura (para mais de um atributo)

Quando o histograma envolve mais do que um atributo:

- O histograma é multidimensional, onde cada dimensão corresponde a um dos atributos originais;
- O analista precisa especificar o número de *bins* por atributo individualmente;
- A contagem de tuplas é feita tanto para cada *bin* multidimensional quanto para todas as projeções em cada atributo.

Em SQL: Os diversos atributos são especificados na cláusula **GROUP BY**, e existe um comando **Generate_Series()** para cada atributo, unidos por um produto cartesiano (**CROSS JOIN**).



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-largura



Vantagens:

- É rápida;
- É linear na cardinalidade N : $O(N)$

Desvantagens:

- Tende a não preservar **outliers** e agrupamentos/classes pequenos;
- Sensível a desbalanceamento do conjunto;
- Reduz a sensibilidade dos processos posteriores;
- Não escala com o aumento da cardinalidade.





Histogramas em Equi-altura (para um único atributo)

- 1 Ordenam-se as tuplas do conjunto de dados usando-se o atributo como chave;
- 2 (opcionalmente pode-se eliminar os k -menores e os k -maiores valores para reduzir distorções);
- 3 Divide-se a sequência de tuplas ordenadas no número de faixas definido pelo analista, de maneira que cada faixa tenha (aproximadamente) o mesmo número de tuplas. Portanto **cada faixa de valores tende a ter a mesma quantidade de tuplas**;
- 4 Note-se que todas as tuplas de mesmo valor ficam na mesma faixa, portanto cada faixa é um valor do contra-domínio;
- 5 Atribui-se a cada tupla o valor do contra-domínio correspondente;
- 5 Gera-se o histograma contando-se o número de tuplas em cada *bin*.

Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-altura

A função de janelamento **NTILE** pode ser usada para dividir as tuplas em faixas segundo a ordenação dada por um (ou mais) atributo de *tiling*.

```
NTILE(NBins) OVER (  
    [PARTITION BY <atrib particao>, ... ]  
    [ORDER BY <atrib para 'tiling'> [ASC | DESC], ...]  
);
```

onde:

- **NBins** é o número de bins onde as tuplas serão distribuídas;
- **PARTITION BY <atribs particao>** pode indicar possíveis classificações para gerar histogramas distintos;
(em geral não é usado para esta aplicação)
- **ORDER BY <atrib para 'tiling'> [ASC | DESC], ...** deve indicar qual(is) atributos compõe cada dimensão do histograma.



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-altura



Exemplo (2.1) em SQL: Histograma de Equi-altura **Idades** com 10 *bins* da tabela **Alunos**

```
SELECT Bin, Min(idade), Max(Idade), Count(*)  
  FROM (SELECT *,  
            NTILE(10) OVER(ORDER By Idade) AS Bin  
        FROM Alunos) AS Partes  
 GROUP BY Bin  
 ORDER BY Bin;
```

Solucao aproximada: NTILE nao respeita a divisao de valores.



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-altura



Exemplo (2.1) em SQL: Histograma de Equi-altura **Idades** com **10 bins** da tabela **Alunos**

Bin	Min	Max	Count
1	15	21	8000
2	21	21	8000
3	21	22	8000
4	22	23	8000
5	23	23	8000
6	23	24	8000
7	24	26	8000
8	26	28	8000
9	28	32	8000
10	32	81	8000

34ms para 15 alunos
122ms para 80.000 alunos





Histogramas em Equi-altura (para mais de um atributo)

Quando o histograma envolve mais do que um atributo:

- O histograma é multidimensional, onde cada dimensão corresponde a um dos atributos originais;
- O analista especifica o número de *bins* individualmente por atributo;
- Em geral procura-se dividir todos os atributos nas mesmas faixas, independentemente dos valores dos demais atributos, para não haver priorização entre os atributos;
- Nesse caso, a contagem de tuplas é feita tanto para cada *bin* multidimensional quanto para todas as projeções em cada atributo;



Porém, é possível dividir um atributo pelos valores dos anteriores (priorizando os atributos) — isso pode ser necessário quando algum atributo tem muito poucos valores;

Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-altura



Exemplo em SQL: Histograma de Equi-altura em **Idades** com **3 bins** e **Cidades** com **4 bins** da tabela **Alunos**

```
SELECT BinI, BinC, Min(Idade), Min(Cidade),  
       Max(Idade), Max(Cidade), Count(*)  
  FROM (SELECT *, NTILE(3) OVER(ORDER By idade) AS BinI,  
              NTILE(4) OVER(ORDER By Cidade) AS BinC  
        FROM Alunos) AS Partes  
 GROUP BY CUBE(Bini, Binc)  
 ORDER BY Bini, Binc;
```

CUBE gera todas as contagens.



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-altura

Exemplo (2.2) em SQL: Histograma de Equi-altura em **Idades** com 3 bins e **Cidades** com 4 bins da tabela **Alunos**



BinI	BinC	MinI	MinC		MaxC	Count
1	1	17	Abaetetuba-PA	22	Itaquaquecetuba-SP	7593
1	2	17	Itaquaquecetuba-SP	22	São Gonçalo-RJ	7345
1	3	17	São Gonçalo-RJ	22	São Paulo-SP	5791
1	4	15	São Paulo-SP	22	Votuporanga-SP	5938
1		15	Abaetetuba-PA	22	Votuporanga-SP	26667
2	1	23	Abaetetuba-PA	25	Itaquaquecetuba-SP	5833
2	2	23	Itaquaquecetuba-SP	25	São Gonçalo-RJ	5998
2	3	22	São Gonçalo-RJ	25	São Paulo-SP	7460
2	4	22	São Paulo-SP	25	Votuporanga-SP	7376
2		22	Abaetetuba-PA	25	Votuporanga-SP	26667
3	1	26	Abaetetuba-PA	69	Itaquaquecetuba-SP	6574
3	2	26	Itaquaquecetuba-SP	81	São Gonçalo-RJ	6657
3	3	25	São Gonçalo-RJ	68	São Paulo-SP	6749
3	4	25	São Paulo-SP	67	Votuporanga-SP	6686
3		25	Abaetetuba-PA	81	Votuporanga-SP	26666
	1	17	Abaetetuba-PA	69	Itaquaquecetuba-SP	20000
	2	17	Itaquaquecetuba-SP	81	São Gonçalo-RJ	20000
	3	17	São Gonçalo-RJ	68	São Paulo-SP	20000
	4	15	São Paulo-SP	67	Votuporanga-SP	20000
		15	Abaetetuba-PA	81	Votuporanga-SP	80000

637ms para
80.000 alunos



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-altura



Vantagens:

- Preserva melhor agrupamentos/classes pequenos;
- É menos sensível a desbalanceamento do conjunto;

Desvantagens:

- Mais lenta que Histogramas em Equi-largura;
- Não linear: $O(N \cdot \log N)$
- Tende a não preservar **outliers**;
- Não escala bem com o aumento do número de atributos no histograma .



Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-entropia



Histogramas em Equi-entropia:

- Neste caso, é necessário que se conheça alguma característica do conjunto completo, como por exemplo a existência de um atributo de classificação;
- O objetivo aqui é dividir os atributos em faixas de tal maneira que cada faixa priorize aquela característica, por exemplo, maximize uma das classes.
- Caso seja usado mais de um atributo, considera-se sua concatenação para a criação das faixas (em ordenação lexicográfica).

Um exemplo é a técnica de *Run-length encoding* para conjuntos que tenham um atributo indicando uma classe conhecida.




Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-entropia



Por exemplo: *Run-length encoding* (para um único atributo)

- 1 Ordenam-se as tuplas do conjunto de dados usando-se o atributo concatenado à classe como chave;
 - 2 Encontram-se as sequências contínuas de tuplas da mesma classe;
 - 3 Mede-se a entropia das classes de cada sequência;
 - 4 Enquanto o número de sequências é maior do que o número de faixas solicitadas para o histograma, fundem-se as duas sequências adjacentes que levem à menor variação de entropia possível e recalcula-se a entropia da faixa resultante;
-  Note-se que todas as tuplas de mesmo valor ficam na mesma faixa, portanto cada faixa resultante é um valor do contra-domínio;
- 5 Atribui-se a cada tupla o valor do contra-domínio correspondente;
 - 6 Gera-se o histograma contando-se o número de tuplas em cada *bin*.

Técnicas de Redução da Cardinalidade

Discretização por Histogramas em Equi-entropia



Run-length encoding

Vantagens:

- Facilita a identificação de agrupamentos/classes pequenos;
- Bem menos sensível a desbalanceamento do conjunto.

Desvantagens:

- Lenta;
- Não linear: $O(N^2 \cdot \log N)$;
- Não escala bem com o aumento da dimensionalidade;
- Tratar mais de um atributo depende da semântica dos atributos.



Técnicas de Redução da Cardinalidade

Amostragem Baseada em Histogramas



Recordando:

Nas técnicas de amostragem de casos baseada em histogramas,

- 1 Cria-se o histograma multidimensional com os atributos mais importantes, ✓
 - 2 executa-se o processo de discretização dos atributos escolhidos, ✓
 - 3 e finalmente escolhem-se tuplas do conjunto original o mais aleatoriamente possível, mas de tal maneira que cada *bin* do histograma gerado atenda a determinada propriedade (todos tenham o mesmo número de tuplas, ou tenha um valor mínimo e máximo para a quantidade de tuplas, etc.).
- ★ Esse último passo é relativamente independente do processo de geração do histograma, e pode ser executado em complexidade linear sobre a cardinalidade N do conjunto (e independente da complexidade F dos atributos e da dimensionalidade E): $O(N)$.



Técnicas de Redução da Cardinalidade

Amostragem Baseada em Histogramas



Exemplo (3.1) em SQL: Amostragem usando Histograma de **Idades** da tabela **Alunos** de cinco em cinco anos:

```
WITH Histo AS (  
    SELECT Floor(Idade/5.00)*5 as Ini, Count(*) AS Conta  
    FROM Alunos  
    GROUP BY Ini),  
MaxBin AS (  
    SELECT (Max(Conta)/Min(Conta))::Double Precision AS Mx FROM Histo),  
Sample AS (SELECT * FROM Alunos A, Histo H  
    WHERE H.Ini<=A.Idade AND H.Ini+5>A.Idade AND  
    Random()*H.Conta/(SELECT Mx FROM Maxbin) <0.01)  
SELECT Ini, Count(*)  
FROM Sample  
GROUP BY Ini ORDER BY Ini
```

1: Aqui a tabela de resultado inclui os atributos da relação **Alunos concatenados** aos atributos **Ini** (Idade base de 5 em 5 onde a tupla é contabilizada) e **Conta** (total de tuplas que existem nesse bin).

Técnicas de Redução da Cardinalidade

Amostragem Baseada em Histogramas



Exemplo (3.1) em SQL: Amostragem usando Histograma de **Idades** da tabela **Alunos** de cinco em cinco anos:

```
WITH Histo AS (  
    SELECT Floor(Idade/5.00)*5 as Ini, Count(*) AS Conta  
    FROM Alunos  
    GROUP BY Ini),  
MaxBin AS (  
    SELECT (Max(Conta)/Min(Conta))::Double Precision AS Mx FROM Histo),  
Sample AS (SELECT * FROM Alunos A, Histo H  
    WHERE H.Ini<=A.Idade AND H.Ini+5>A.Idade AND  
    Random()*H.Conta/(SELECT Mx FROM Maxbin) <0.01)  
SELECT Ini, Count(*)  
FROM Sample  
GROUP BY Ini ORDER BY Ini
```

2: Aqui a tabela resultado é o próprio histograma.



Técnicas de Redução da Cardinalidade

Amostragem Baseada em Histogramas

Exemplo (3.1) em SQL: Amostragem usando Histograma de **Idades** da tabela **Alunos** de cinco em cinco anos:

Resultado:

Idade	Conta
15	495
20	481
25	474
30	505
35	469
40	329
45	170
50	151
55	106
60	67
65	26
80	1

289ms para
80.000 alunos

Idade	Conta
15	1
17	64
18	467
19	1001
20	1829
21	14724
22	11724
23	13522
24	5567
25	4679
26	3999
27	3569
28	3144
29	2786
30	2406
31	2002
32	1736
33	1407
...	...

Comparando com
o total de tuplas:

66ms para
80.000 alunos



Técnicas de Redução da Cardinalidade

Amostragem Baseada em Histogramas



- As técnicas de amostragem baseadas em histogramas são as mais básicas, e são universalmente usadas para auxiliar as demais formas de amostragem.
- Elas frequentemente são consideradas parte do processo de preparação de dados, e por isso é mais indicado, e mais frequente, executá-las direto nos SGBDs.



Técnicas de Redução da Cardinalidade

Amostragem Dirigida — Baseada em classes



Amostragem Dirigida — Baseada em classes

Para amostragem baseada em classes também é necessário que exista um atributo de classificação, porém neste caso o efeito das classes é mais marcante, pois qualquer um dos métodos de amostragem por histograma pode ser usado em separado para cada classe.

Existem duas grandes divisões de algoritmos para a amostragem baseada em classes:

- Para classes definidas no mesmo espaço do conjunto global;
- Para classes definidas em sub-espacos distintos do conjunto global;



Técnicas de Redução da Cardinalidade

Amostragem Dirigida — Baseada em classes



Amostragem Baseada em classes definidas no espaço todo

É necessário escolher um atributo de classificação

Amostragem Baseada em classes definidas em projeções do espaço

É necessário escolher um atributo de classificação e quais são os atributos relevantes para cada classe.

- O processo de amostragem pode ser qualquer daqueles já estudados (aleatório ou baseados em qualquer forma de histograma)
- Aplica-se o processo de amostragem para cada classe conhecida.



Técnicas de Redução da Cardinalidade

Amostragem Dirigida — Baseada em classes



Vantagens:

- Sensível a desbalanceamento do conjunto;

Desvantagens:

- Requer o conhecimento prévio de classes;

No geral, mantém as mesmas vantagens/desvantagens do método de amostragem/discretização adotado.





Amostragem Baseada em classes definidas no espaço todo

É necessário conhecer um atributo de classificação

Variação: Agrupamentos podem ser identificados durante o processo de amostragem.

- Nesse caso, aplica-se um processo de análise em multi-resolução do espaço (**semelhante à análise fractal**), onde
 - 1 O espaço vai sendo sucessivamente dividido em híper-retângulos,
 - 2 sendo que apenas os híper-retângulos com contagem de elementos acima de um limiar são re-divididos.
 - 3 No final, cada híper-retângulo tem os elementos amostrados (independente da resolução), segundo a propriedade de amostragem necessária.
- Note-se que os agrupamentos são naturalmente localizados pela densidade dos híper-retângulos.



Técnicas de Redução da Cardinalidade

Amostragem Dirigida — Baseada em classes



Amostragem Dirigida Baseada em Classes

Vantagens:

- Pode identificar agrupamentos pequenos;
- Pouco sensível a desbalanceamento do conjunto;
- Relativamente rápida $O(N \cdot E \cdot F)$.

Desvantagens:

- Requer um volume maior de memória.





Amostragem Baseada em Densidade

Para amostragem baseada em densidade, é necessário que o analista defina um limiar (*threshold*) de distância (ou similaridade) dentro da qual considera-se que apenas um elemento (ou um número pequeno deles, pré-definido pelo analista) seja suficiente para **representar** todos eles.

Essas técnicas são também conhecidas como baseadas em **Vizinhança** ou em **k-vizinhos mais próximos** (*k-Nearest Neighbors* — *kNN*).

Vantagens:

- Preserva bem agrupamentos/classes pequenos;
- Praticamente insensível a desbalanceamento do conjunto;

Desvantagens:

- Muito lenta;
- Requer muita memória;
- Complexidade elevada: $O(N^3 \cdot E^2)$ ou maior;





- 1 Conceitos básicos em Redução de Dados
- 2 Técnicas de Redução da Cardinalidade
 - Amostragem Aleatória
 - Amostragem Dirigida — Baseada em Histogramas
 - Amostragem Dirigida — Baseada em classes
 - Amostragem Dirigida — Baseada em densidade
- 3 Técnicas de Redução da Dimensionalidade
- 4 Técnicas de compressão de valores de atributos





Redução da Dimensionalidade — Objetivo:

Eliminar dimensões irrelevantes ou fracamente relevantes e dimensões redundantes (correlacionadas), de maneira a causar o mínimo de impacto nos processos subsequentes (pode até melhorar o desempenho).

- O resultado é um conjunto de dados com dimensão final $D < E$.
- Existem basicamente duas maneiras de reduzir a dimensionalidade:
 - **Selecionar** as D dimensões mais importantes;
 - **Transformar** (extrair) as dimensões aumentando seu poder de discriminação e reduzindo sua quantidade.



Técnicas de Redução da Dimensionalidade

Seleção de atributos



Seleção de atributos

As técnicas de seleção de atributos escolhem dentre aqueles existentes, quais são os D mais importantes para caracterizar os dados, desprezando-se os demais.

As técnicas mais importantes para a seleção de atributos são:

- Maximizar o ganho de informação: máxima entropia, PCA (linear), *Random forests*;
- Busca aleatória usando algoritmos genéticos;
- Identificação de correlações;
- Manual: o analista adiciona/remove atributos baseado em funções sobre ou entre os atributos (entropia ou variância mínima, valores nulos, correlações par a par, etc.).





Seleção individual de atributos

- A seleção é baseada apenas em propriedades dos atributos, tais como
 - Proporção de tuplas com valores nulos
 - Variância pequena
- É necessário definir um valor de corte para eliminar atributos

 Usualmente isso é feito de maneira empírica:

Avaliação exaustiva da precisão obtida por um processo de mineração que usa os atributos selecionados.

- É necessário **medir a propriedade** a ser usada, para **cada atributo**.



Técnicas de Redução da Dimensionalidade

Seleção individual de atributos

- Na maioria das vezes, é mais vantajoso executar a medida da propriedade a ser avaliada em SQL.

Exemplo: Em quantas tuplas um atributo é nulo?

Seja o atributo **nota da primeira prova: NotaP1** da relação de Matriculas do aluno:

 Proporção de tuplas com valores nulos

```
SELECT Count(*) FILTER(WHERE NotaP1 IS NULL) AS Nulos,  
       Count(*) AS Total  
FROM Matricula;
```

Nulos	Total
70.111	643.586

220ms

 Variância pequena:

```
SELECT Variance(NotaP1) FROM Matricula
```

Variance
3.173756

332ms



Técnicas de Redução da Dimensionalidade

Seleção individual de atributos



- Mas é necessário calcular **cada propriedade** sobre **todos os atributos**.
- Como o Modelo Relacional é um **meta-modelo**, as informações sobre todos os objetos de uma base de dados, incluindo **todas as tabelas** da base e **todos os atributos** de cada tabela estão no **Catálogo da base de dados**.





Terminologia: **Meta-Modelo**

Um modelo capaz de modelar a si mesmo é chamado um Meta-Modelo.


- O Modelo Relacional é um Meta Modelo
- Um SGBD implementa o meta-modelo relacional com Tabelas de Tabelas, Tabelas de Atributos, etc.
- Essas tabelas são ditas **“do sistema”** e são mantidas em um esquema separado, chamado **catálogo**
- O **Catálogo da base de dados** inclui todas as tabelas gerenciadas, em qualquer esquema da base.



Técnicas de Redução da Dimensionalidade

O catálogo do



- No gerenciador , o **catálogo** é mantido em dois esquemas separados, chamados `Information_Schema` e `Pg_Catalog`.
- Os nomes dos objetos do sistema começam com `'sql_'` ou `'pg_'`, respectivamente.
- Os “objetos” do sistema (incluindo tabelas, visões, funções, etc.) têm suas propriedades comuns (incluindo nome) armazenadas na relação `Pg_Class`.



Técnicas de Redução da Dimensionalidade

O catálogo do  PostgreSQL

Por exemplo

Listar todas as tabelas do usuário com suas quantidades de atributos e a quantidade de páginas em disco e de tuplas em cada tabela.

```
SELECT RelName, RelNAtts, RelKind, RelPages, RelTuples
FROM pg_class Cl JOIN Pg_NameSpace NS
      ON NS.OID=Cl.RelNameSpace
WHERE (Cl.RelKind='r' or Cl.RelKind='v') AND
      NS.NSpName='public';
```

RelName	RelNAtts	RelKind	RelPages	RelTuples
professor	4	r	1	6159
aluno	4	r	1	80000
turma	4	r	1	23560
discip	4	r	1	11890
matricula	3	r	1	643534
ministra	3	r	1	15232
niveis	2	v	0	0



Técnicas de Redução da Dimensionalidade

O catálogo do



Exemplo

Listar todas as colunas de todas as tabelas do usuário:

```
SELECT SELECT Cl.RelName, A.AttName, A.AttNum
FROM pg_class Cl JOIN Pg_NameSpace NS
                        ON NS.OID=Cl.RelNameSpace
      JOIN Pg_Attribute A ON A.AttRelId=Cl.OID
WHERE (Cl.RelKind='r' or Cl.RelKind='v') AND
      NS.NSpName='public' AND
      A.AttNum>0
ORDER BY 1, 3;
```

RelName	AttName	AttNum
aluno	nome	1
aluno	nusp	2
aluno	idade	3
aluno	cidade	4
discip	sigla	1
discip	nome	2

Técnicas de Redução da Dimensionalidade

O catálogo do  PostgreSQL

Todos os objetos da base de dados estão no **catálogo**:

Exemplo

Listar todos os esquemas definidos nesta base de dados:

```
SELECT NspName, NspOwner
FROM pg_catalog.pg_namespace
WHERE nspname NOT LIKE 'pg_%'
ORDER BY nspname;
```

NspName	NspOwner
anoetivo	10
historico	10
secretaria	38225
public	10
...	



Técnicas de Redução da Dimensionalidade

Dados sobre os atributos de uma tabela

Exemplo

Listar todos os atributos de uma dada tabela, com seu respectivo **tipo de dados**:

```
SELECT C.RelName, A.AttName, A.AttNum, T.TypName
FROM pg_Class C, pg_attribute A, pg_type T
WHERE C.RelName NOT LIKE 'pg_%' AND C.RelName NOT LIKE 'sql_%' AND
      C.RelKind='r' AND
      A.AttRelId=C.OID AND
      A.AttTypId=T.OID AND
      A.AttNum>0 AND -- AttNum<0 ==> Atributo de sistema
      C.RelName='alunos'
ORDER BY A.AttNum;
```

RelName	Name	AttNum	TypName
alunos	nusp	1	numeric
alunos	nome	2	varchar
alunos	idade	3	int4
alunos	cidade	4	varchar


16ms



Técnicas de Redução da Dimensionalidade

Dados sobre os atributos de uma tabela



- Em  PostgreSQL, os diversos tipos de dados disponíveis para o usuário (mais de 44 na versão V.14), são representados por poucos tipos fundamentais.
- Os tipos numéricos são representados apenas pelos tipos:
 - Inteiros – `'int2'`, `'int4'`, `'int8'`
 - Ponto flutuante – `'float4'`, `'float8'`
 - Decimais – `'numeric'`
- Os tipos textuais são internamente representados como `'character varying'`
- Isso facilita testar os tipos dos dados armazenados.




Técnicas de Redução da Dimensionalidade


Dados sobre os atributos de uma tabela



- Para medir as propriedades de um atributo, é necessário criar uma consulta específica para aquele atributo.
- Para gerar consultas sobre **todos os atributos**, pode-se usar **SQL dinâmico**.
- Por exemplo, pode-se criar uma função, em **PLPgSQL** para obter as **estatísticas de interesse** sobre todos os atributos.

Exemplo

Obter:  de todos os atributos de uma tabela:
a **quantidade de nulos** e a **Cardinalidade do domínio**, e

 dos atributos de tipos numéricos:
a **Variância** e o **Desvio Padrão**.

Técnicas de Redução da Dimensionalidade



```
CREATE OR REPLACE FUNCTION MinhasEstatisticas(Tab TEXT) RETURNS
TABLE(NomeAtrib TEXT, Tipo TEXT, Nulls INTEGER,
      Cardinality INTEGER, Variance DOUBLE PRECISION, StdDev DOUBLE PRECISION) AS $$
DECLARE
    Var_r Record;      Var_Cmd TEXT;      Var_Cmd2 TEXT;
BEGIN
    Var_Cmd='SELECT A.AttName::TEXT AN, T.TypName::TEXT ATy
FROM pg_Class C, pg_attribute A, pg_type T
WHERE C.RelName NOT LIKE 'pg_%' AND C.RelName NOT LIKE 'sql_%' AND
      C.RelKind='r' AND
      A.AttRelId=C.OID AND
      A.AttTypId=T.OID AND A.AttNum>0 AND
      C.RelName = '||Tab||''';

    FOR Var_r IN EXECUTE Var_Cmd
    LOOP
        Var_Cmd2:='SELECT Count(*) from '||Tab||' WHERE '||Var_r.AN||' IS NULL;';
        EXECUTE Var_Cmd2 INTO Nulls;
        Var_Cmd2:='SELECT Count(DISTINCT '||Var_r.AN||'), ';
        IF Var_r.ATy IN('int2', 'int4', 'int8', 'float4', 'float8', 'numeric')
        Var_Cmd2:=Var_Cmd2||'Var_Pop('||Var_r.AN||'), stddev_pop('||Var_r.AN||')'; ELSE
        Var_Cmd2:=Var_Cmd2||'NULL, NULL'; END IF;
        Var_Cmd2:=Var_Cmd2||' FROM '||Tab||''';
        EXECUTE Var_Cmd2 INTO Cardinality, Variance, StdDev;
        NomeAtrib:=Var_r.AN;
        Tipo:=Var_r.ATy;
        RETURN NEXT;
    END LOOP;
END; $$ LANGUAGE plpgsql;
```

Técnicas de Redução da Dimensionalidade

Dados sobre os atributos de uma tabela



```
SELECT * FROM MinhasEstatisticas('alunos');
```

Resultado:

NomeAtrib	Tipo	Nulls	Cardinality	Variance	StdDev
nusp	numeric	0	80000	674394187368065	25969100.6269
nome	varchar	32	78706		
idade	int4	0	55	24.02323	4.90135
cidade	varchar	0	700		

930ms para 80.000 alunos



Técnicas de Redução da Dimensionalidade

Dados sobre os atributos de uma tabela



```
SELECT * FROM MinhasEstatisticas('matricula');
```

Resultado:

NomeAtrib	Tipo	Nulls	Cardinality	Variance	StdDev
codigoturma	int8	0	24278	105356414.5422	10264.32728
nusp	numeric	0	79984	672637761957588	25935260.9772
notap1	numeric	70111	100	3.17375	1.78150
notap2	numeric	105131	59	2.72037	1.64935
notasub	numeric	235156	99	3.59636	1.89640
mediap	numeric	0	84	2.64423	1.62611
mediat	numeric	35134	101	5.23371	2.28773
nf	numeric	0	93	2.79450	1.67167
frequencia	numeric	1	1	0	0

9.5s para 701701 matrículas



Técnicas de Redução da Dimensionalidade

Seleção de atributos



As técnicas de **seleção de atributos baseadas em identificação de correlações** procuram eliminar atributos altamente correlacionados com outros.

- Identificação de correlações bi-lineares – Por exemplo:
 - **Coeficiente de correlação de Pearson (PCC)** – Para variáveis contínuas (numéricas);
 - **Medida χ^2 (qui-quadrado) de Pearson's** – Para variáveis discretas;
- Podem ser usadas técnicas:
 - Gulosas (*greedy Backward/Forward*)
 - ☞ As técnicas exaustivas/regenerativas são muito demoradas;
 - Baseadas em Teoria dos Fractais.
- Identificação de correlações baseadas em técnicas de **ganho de informação**:



Técnicas de Redução da Dimensionalidade

Seleção de atributos



As técnicas baseadas em **ganho de informação** em geral dependem de conhecimento sobre o conjunto, como por exemplo a existência de um **atributo de classificação**:

- Busca-se obter:
 - O máximo de correlação com o atributo de classificação;
 - O mínimo de correlação com os demais atributos.
- Exemplos:
 - Atributos mais eficientes para poda em *Random Forests*
 - Algoritmo *Relief*





Seleção de atributos com *Random Forests*

- 1 Constroi um grande número de árvores de decisão (1.000 a 5.000) para o atributo de classificação, com 2 ou 3 níveis e poucos atributos de decisão (5 a 8) escolhidos aleatoriamente;
- 2 ordenam-se os atributos medindo a frequência com que são usados em cada nível:

$$(\text{Score}_{\text{Atr}_i} = \# \text{Splits}(\text{raiz}) / \# \text{Candidatos}(\text{Raiz}) + \# \text{Splits}(\text{Nivel1}) / \# \text{Candidatos}(\text{Nivel1}) + \dots)$$

- 3 No final, escolhe as k dimensões i com maior valor de $\text{Score}_{\text{Atr}_i}$





O Algoritmo *Refief* para Seleção de Atributos

O algoritmo usa um atributo de classificação binário como objetivo para avaliar a importância de cada atributo e ranquear as E dimensões por essa importância.

- ❶ Associa um peso, inicialmente igual para todas as dimensões ($W_i = 0, i = 1 \dots E$);
- ❷ Escolhe aleatoriamente elementos t_s do conjunto de dados e procura:
 - *nearest hit* = t_h : o elemento mais próximo a t_s da mesma classe; e
 - *nearest miss* = t_m : o elemento mais próximo da outra classe;
- ❸ Atualiza o peso de cada dimensão de acordo com quão bem cada um distingue a instância de seu *nearest hit* e seu *nearest miss*:
$$W_i = W_i + (t_s[i] - t_h[i])^2 - (t_s[i] - t_m[i])^2, i = 1 \dots E$$
- ❹ No final, escolhe as k dimensões i com maior valor de W_i .



Transformação de atributos

As técnicas de transformação de atributos aplicam uma transformação no espaço de dados, reorientando os eixos que definem o espaço de maneira a maximizar a distribuição dos dados, com um número menor de dimensões suficientes para caracterizar adequadamente os dados.

As técnicas mais importantes para a transformação de atributos são:

- Análise das componentes principais (*Principal Component Analysis* — PCA)
- Mapeamentos em Escala Multidimensional (*Multidimensional Scaling Mapping* — MDS mapping)
- Identificação de variáveis latentes.



Técnicas de Redução da Dimensionalidade

Análise das componentes principais

- As transformações do espaço de dados podem ser lineares ou não.
- O principal método de **transformação linear** é a análise das componentes principais.

Análise das componentes principais – PCA

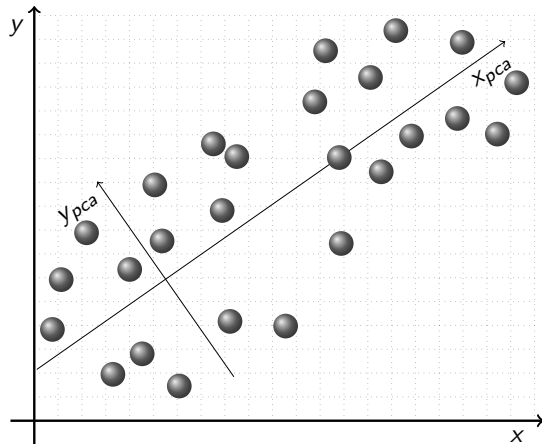
As técnicas de análise das componentes principais consideram que **o conjunto de elementos é um objeto** que se estende pelo espaço original, mas que não o preenche totalmente. Dessa maneira, é possível:

- 1 Rotacionar o objeto, de maneira que sua maior extensão esteja paralela a um eixo (identificando-se a direção onde o objeto tem maior variância nos valores de seus atributos);
- 2 Rotacionar novamente o objeto mantendo o eixo já escolhido, e assim sucessivamente para todas as dimensões do objeto original;
- 3 Escolher apenas os $D \leq E$ eixos com maior extensão — **as componentes principais** — como os novos atributos mais importantes.



Técnicas de Redução da Dimensionalidade

Análise das componentes principais



Técnicas de Redução da Dimensionalidade

Análise das componentes principais



- As dimensões são ordenadas na sequência de extensões cada vez menores;
- Pode-se estabelecer:
 - um limiar mínimo para o corte das dimensões (*threshold*);
 - uma quantidade definida D de dimensões.
- O resultado é uma coleção de D dimensões, $D \leq E$, tal que o valor de cada dimensão $d_i \in \mathbb{D}$ é uma função dos atributos originais, e que representa as transformações de rotação do espaço induzidas pelo método PCA.



Técnicas de Redução da Dimensionalidade

Análise das componentes principais



Análise das componentes principais

Vantagens:

- Bastante adequada quando existem muitas correlações (lineares) entre as dimensões originais;
- Método razoavelmente rápido: $O(N \cdot E)$

Desvantagens:

- Distorce o espaço original (perde a semântica dos atributos);
- Distorce dados com correlações não lineares;
- Somente opera em espaços euclidianos.





Mapeamentos em Escala Multidimensional

As técnicas de mapeamentos em escala multidimensional estendem a técnica de PCA para preservar as correlações não lineares.

- 1 Elas são baseadas na escolha de *pivots* no espaço original que definem os eixos de maior variância (como a PCA);
- 2 Mas ao invés de rotacionar o espaço, as técnicas criam um **espaço mapeado** baseado na distância (usualmente a distancia euclidiana) entre cada elemento de dados e os pares de *pivots*;
- 3 Por exemplo, a técnica *FastMap* representa cada coordenada pela projeção ortogonal de cada elemento sobre o eixo que passa por cada par de *pivots*;
- 4 O número de pares de *pivots* define a dimensionalidade final D do espaço mapeado.

Técnicas de Redução da Dimensionalidade

Mapeamentos em Escala Multidimensional



- Note-se que as projeções de cada dimensão mapeada têm variância cada vez menor;
- Pode-se estabelecer:
 - um limiar mínimo para corte das dimensões (*threshold*);
 - ou uma quantidade definida D de dimensões.
- O resultado é uma coleção de D dimensões, $D \leq E$ tal que o valor de cada dimensão $d_i \in \mathbb{D}$ é uma função dos atributos originais.
 - No caso do método FastMap, essa função representa as projeções de cada elemento sobre os eixos (possivelmente não ortogonais) do espaço induzidos pelo método.





Mapeamentos em Escala Multidimensional

Vantagens:

- Adequada mesmo quando existem correlações não lineares entre as dimensões originais;
- Método razoavelmente rápido: $O(N \cdot E)$
- Pode operar sobre dados em espaços métricos, inclusive com métricas não-euclidianas;

Desvantagens:

- Distorce o espaço original (perde a semântica dos atributos);
- Para algumas métricas não euclidianas, pode distorcer o espaço original



Roteiro



- 1 Conceitos básicos em Redução de Dados
- 2 Técnicas de Redução da Cardinalidade
- 3 Técnicas de Redução da Dimensionalidade
 - Seleção de atributos
 - Transformação (ou extração) de atributos
- 4 Técnicas de compressão de valores de atributos



Técnicas de compressão de valores de atributos

Introdução



Compressão de valores de atributos — Objetivo:

Reduzir a quantidade, a complexidade e a redundância dos valores associados a cada atributo, visando diminuir a complexidade computacional e o volume de dados manipulados pelos algoritmos de análise.

- As técnicas de compressão de valores são usadas com dois objetivos
 - Reduzir a **redundância** nos dados, por exemplo eliminando ruídos nas medidas;
 - reduzir **complexidade** dos dados, por exemplo em dados multimídia.



Técnicas de compressão de valores de atributos



- Uma técnica trivial é reduzir a cardinalidade do domínio do atributo:

Discretização

- **Exemplo (4.1) em SQL:**

Analisar as notas dos alunos
desprezando os dígitos fracionários:

```
SELECT Floor(NotaP1) as ContaNota,  
       Count(*) as Contagem  
FROM Matricula  
GROUP BY ContaNota  
ORDER BY ContaNota;
```

ContaNota	Contagem
0	2924
1	22792
2	60303
3	99600
4	122049
5	123574
6	103535
7	65796
8	26836
9	4179
10	2
	70111

429ms para
701701 matrículas

Analisar por níveis:

```
SELECT CASE Floor(NotaP1/2)  
        WHEN 1 THEN 'I'  
        WHEN 2 THEN 'C' WHEN 3 THEN 'B'  
        WHEN 4 THEN 'A' WHEN 5 THEN 'A'  
        ELSE 'R'  
        END AS Conceito,  
       Count(*) as Contagem  
FROM Matricula  
GROUP BY Conceito  
ORDER BY Conceito;
```

Conceito	Contagem
A	31017
B	169331
C	245623
I	159903
R	95827

557ms para
701701 matrículas



Técnicas de compressão de valores de atributos

Redução da redundância nos dados



Redundância nos dados

A Teoria Matemática da Comunicação (ou Teoria da Informação) visa medir quanto de **Informação** e quanto de **Redundância** existe em determinado volume de dados.

- **Informação** é a porção de dados que deve ser preservada para manter a capacidade de interpretação e/ou identificação dos dados;
- **Redundância** é a porção de dados que pode ser removido (e posteriormente re-inserido) sem que isso altere a capacidade de interpretação e/ou identificação dos dados;
- **Compressão de dados** é uma técnica para reduzir a redundância sem afetar a informação contida nos dados. Ela opera em duas fases:
 - **Modelagem**: identifica-se e descreve-se a redundância que existe nos dados;
 - **Codificação**: identifica-se e descreve-se quanto os dados originais diferem do modelo.

Técnicas de compressão de valores de atributos

Processos de medida de dados



Redundância nos dados

- Qualquer processo real de medida de dados pode ser descrito como uma fonte que gera uma sequência de símbolos de um domínio finito, chamado alfabeto.
- A redução da redundância nos dados medidos corresponde a eliminar a redundância das sequências de símbolos gerados, isto é, visa codificar de maneira mais compacta o modelo de medida dos dados.
- A principal técnica de redução da redundância nos dados é o Modelo das medidas independentes e a Entropia de Shannon.



Técnicas de compressão de valores de atributos

Modelo de medidas independente e a Entropia de Shannon



- O Modelo de medidas independente assume que as medidas obtidas são estatisticamente independentes uma das outras:

☞ Dado um alfabeto $A = \{a_1, a_2, \dots, a_S\}$ e as probabilidades de ocorrência de cada símbolo $P = \{p(a_1), p(a_2), \dots, p(a_S)\}$ em A :

assume-se que cada $p(a_i)$ é independente de $p(a_j)$, $\forall i, j \in A$.

- Como a ocorrência de cada símbolo é inesperada, então o conteúdo de informação $I(a_i)$ de cada símbolo a_i em termos da sua probabilidade de ocorrência associada $p(a_i)$ é dado por:

$$I(a_i) = \log_2 \left(\frac{1}{p(a_i)} \right) = -\log_2(p(a_i)).$$



Técnicas de compressão de valores de atributos

Modelo de medidas independente e a Entropia de Shannon



- A base 2 do logaritmo indica que a informação é expressa em **bits** (unidades binárias);
- Portanto:
 - cada símbolo a_i pode ser representado em $-\log_2 p(a_i)$ bits!
 - um símbolo com maior probabilidade de ocorrer numa medida deveria ser codificado usando menos bits!
- Em média, a quantidade de informação de um determinado volume de dados que segue o modelo de medidas independentes é dada pela **Entropia de Shannon**, expressa como:

$$E_{Shannon} = \sum_{i=1}^S p(a_i) I(a_i) = - \sum_{i=1}^S p(a_i) \log_2 p(a_i)$$

- Portanto, a entropia é o comprimento do código binário mais compacto possível para um dado volume de dados.



Técnicas de compressão de valores de atributos

Modelo de medidas independente e a Entropia de Shannon



Por exemplo:

- Seja um processo de medida independente,
 - com alfabeto $A = \{\alpha, \beta, \gamma, \delta\}$ e
 - com as probabilidades associadas $p(\alpha) = 0.65$, $p(\beta) = 0.20$, $p(\gamma) = 0.10$ e $p(\delta) = 0.05$, respectivamente.
- A entropia desse processo é $E = -(0.65 \log_2 0.65 + 0.20 \log_2 0.20 + 0.10 \log_2 0.10 + 0.05 \log_2 0.05) \approx 1,42$ bits/símbolo.
- Portanto, cada sequência de mil símbolos gerados por esse processo de medida pode ser codificado em 1.420 bits.
- As técnicas de discretização baseadas em *Run-length encoding* e a técnica de codificação de **Huffman** são baseadas nesse conceito.



Técnicas de compressão de valores de atributos

Redução da complexidade nos dados



Complexidade nos dados

- Dados multimídia e os dados complexos em geral são difíceis de serem comparados:
- Extraem-se características de cada objeto, de maneira que cada elemento passa a ser representado por um ponto num espaço das características.

A partir daí, a busca passa a ser realizada comparando os elementos num espaço de similaridade

👉 Buscas por similaridade.





MBA em Inteligência Artificial e Big Data

– Curso 3: Administração de Dados Complexos em Larga Escala –

★ **Redução de Dados** ★

Caetano Traina Júnior

Grupo de Bases de Dados e Imagens – GBdI
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos



Redução de Dados
FIM