



Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
MBA em Inteligência Artificial e Big Data
– Curso 3: Administração de Dados Complexos em Larga Escala –

Questões da Quinzena 2

Prof. Dr. Caetano Traina Júnior

1 Compressão de dados

Os exercícios seguintes são baseados nas tabelas do conjunto de dados disponibilizados pelo *Repositório COVID-19 DataSharing/BR*.

Exercício 1)

Escreva um comando em SQL que crie um histograma equi-largura de distribuição das idades dos pacientes, de maneira que a largura de cada **bin** do histograma seja de '**duas idades**'.

Atente para que todas as '**idades possíveis**', desde 0 até a maior idade registrada nos dados esteja representada no histograma.

Resposta:Dificuldade: 

Este exercício consiste essencialmente em aplicar a “técnica de geração de histogramas que inclui todos os *bins*” mostrada em aula para a tabela [Alunos](#), agora aplicada à [Pacientes](#). Note que aqui não se pede para excluir os extremos, mas o mesmo ajuste que é usado para contabilizar as idades da tabela-alvo deve ser aplicado para a geração dos índices dos *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(2021-AA_Nascimento) Mi, Max(2021-AA_Nascimento) Ma
       FROM Pacientes)
      SELECT Generate_Series(2*(Lim.Mi/2), Lim.Ma,2) AS B FROM Lim) AS Bins
LEFT OUTER JOIN
      (SELECT 2*((2021-AA_Nascimento)/2) Idade, Count(*) Conta
      FROM Pacientes
      GROUP BY 1 ORDER BY 1) AS Tab
ON Bins.B=Tab.Idade;
```

Exercício 2)

Modifique esse comando para gerar um histograma equi-largura com 10 *bins*.

Resposta:

Dificuldade:

NOTAS:

1. Este exercício consiste essencialmente em aplicar a “técnica de geração de histogramas com quantidade de *bins* definida” mostrada em aula para a tabela *Alunos*, agora aplicada à *Pacientes*.
2. Lembre-se que a função *Width.bucket* recebe como parâmetro o número de divisões para separar os elementos, portanto é um menos do que o número de *bins* desejado ($15=16-1$).
3. Essa função não contabiliza as tuplas onde o atributo de separação (*2021-AA_Nascimento*) é nulo, portanto o agrupamento final irá acrescentar um histograma separado para contabilizar essas tuplas. Existem duas possibilidades para obter então um histograma com 16 bins:

- 1 reservar um *bin* para contabilizar os nulos, e portanto dividir o atributo de separação em $14=((16-1)-1)$; ou

- 2 remover a contagem de nulos do histograma final.

Como a quantidade de nulos é bem maior do que as contagens de idades conhecidas, na solução abaixo foi adotada essa opção, uma vez que evita a distorção que esse desbalanceamento causaria numa eventual visualização do histograma resultante.

```
SELECT Bin, Sum(Conta) FROM
(WITH MinMax AS (
    SELECT Min(2021-AA_Nascimento) Mi, Max(2021-AA_Nascimento) Ma
    FROM Pacientes)
SELECT 2021-AA_Nascimento, -- Atributo desnecessário aqui
    Width_bucket(2021-AA_Nascimento,
        (SELECT Mi FROM MinMax),
        (SELECT Ma FROM MinMax), 15) as Bin,
    Count(*) Conta
FROM Pacientes
GROUP BY 1
ORDER BY 1) Temp
WHERE BIN IS NOT NULL
GROUP BY Bin;
```

Exercício 3)

Escreva um comando em SQL que crie um histograma equi-altura com 10 *bins* da distribuição por idade, corrigindo a atribuição dos *bins* para que o histograma inclua todos os pacientes com a mesma idade no mesmo *bin*.

Resposta:

Dificuldade: 

NOTAS:

1. Este exercício pode aplicar a “técnica de geração de histogramas de equi-altura de *bins* definida” mostrada em aula para a tabela [Alunos](#), mas deve-se lembrar que a solução mostrada em aula usando a função [NTile](#) não respeita a divisão de valores entre os bins. Portanto, é necessário corrigir essa atribuição, o que pode ser feito particionando a tabela original pelas [Idades](#), independente de qual seja o *bin* atribuído a cada tupla, e re-atribuir sempre um mesmo valor para toda a partição (por exemplo o primeiro valor da partição).
2. O exemplo mostrado em aula executa a atribuição das tuplas a cada *bin*. A geração do histograma é feita pelo comando [SELECT](#) [externo](#).

```
SELECT BinCorreto, Min(2021-AA_Nascimento), Max(2021-AA_Nascimento), Count(*)
FROM ( WITH Temp AS (
        SELECT *,
            NTILE(10) OVER(ORDER By 2021-AA_Nascimento) AS Bin
        FROM Pacientes
        WHERE AA_Nascimento IS NOT NULL)
    SELECT *,
        CASE WHEN AA_Nascimento=LAG(AA_Nascimento) OVER(
            ORDER BY AA_Nascimento)
            THEN First_Value(Bin) OVER(
                Partition BY AA_Nascimento ORDER BY AA_Nascimento)
            ELSE Bin END AS BinCorreto
    FROM Temp) AS Partes
GROUP BY 1
ORDER BY 1 ;
```

Exercício 4)

Escreva um comando em SQL que acrescente três novos atributos: **Numerico**, **Origem** e **Exame** na tabela **ExamLabs** que discretizem respectivamente os atributos:

1. CD.ValorReferencia, indicando se o atributo tem um valor Numérico (Numero) (possivelmente com outros textos) ou é puramente textual (Texto),
2. DE.Origem, contabilizando as origens em (Hosp)ital, (Lab)oratório, (inter)nação ou (pronto) socorro para caracterizar: (Hosp)ital, (Lab)oratório, (Atend)imento e os demais como (Outros) – use as letras entre parênteses para buscar esse padrão no atributo,
3. DE.Exame, contabilizando os exames como sendo de (Hemogr)ama, (colest)erol, e (covid) ou (pcr) ou (igm) ou (igg) para caracterizar (Hemograma), (Colesterol), (Covid) e (outros).

Use o texto entre parênteses tanto para buscar esse padrão no atributo quanto para atribuir como valor da discretização, conforme o caso.

Resposta:

Dificuldade: 

NOTAS:

1. Este exercício segue o modelo usual de discretização de atributos, usando as estruturas de controle **CASE**.
2. A dificuldade maior é a identificação de *subcadeias* que representem número, o que foi feito na solução apresentada usando expressões regulares simples. Padrões mais elaborados podem ser usados.

```
SELECT *,
    CASE WHEN CD_ValorReferencia ~ '.*[\d,+-.]'
        THEN 'Numerico'
        ELSE 'Texto' END Numerico,
    CASE WHEN de_origem ~* 'hosp' THEN 'Hosp'
        WHEN de_origem ~* 'lab' THEN 'Lab'
        WHEN de_origem ~* 'intern' OR de_origem ~* 'pronto' THEN 'Atend'
        ELSE 'Outro' END AS Origem,
    CASE WHEN De_Exame ~* 'hemograma' THEN 'Hemograma'
        WHEN De_Exame ~* 'colest' OR De_Exame ~* 'tri.*glic' THEN 'Colesterol'
        WHEN De_Exame ~* '(covid)|(pc.*r)|(igm)|(igg)' THEN 'Covid'
        ELSE 'Outros' END AS Exame
FROM examlabs;
```

Exercício 5)

Escreva um comando em SQL que crie o histograma tri-dimensional equi-largura de distribuição de exames (da tabela ExamLabs), tendo por dimensões:

1. DE.Hospital,
2. DE.Origem, contabilizando as origens em (Hosp)ital, (Lab)oratório, (inter)nação ou (pronto) socorro para caracterizar: (Hosp)ital, (Lab)oratório, (Atend)imento e os demais como (Outros)
3. DE.Exame, contabilizando os exames como sendo de (Hemogr)ama, (colest)erol, e (covid) ou (pcr) ou (igm) ou (igg) para caracterizar (Hemograma), (Cholesterol), (Covid) e (outros).

Resposta:

Dificuldade:

NOTAS:


1. Este exercício consiste essencialmente em aplicar a “técnica de geração de histogramas equi-largura” mostrada em aula para a tabela [Alunos](#), mas para as três dimensões solicitadas. A técnica consiste em fazer a separação de cada tupla que deve ser associada a cada dimensão discretizando os valores, segundo a descrição solicitada, gerando os atributos discretizados [Hospital](#), [Origem](#) e [Exame](#) respectivamente, de maneira equivalente à adotada no exercício anterior.
2. O agrupamento final pode ser feito por um comando [GROUP BY](#), mas na solução apresentada foi gerado um [CUBE](#), de maneira que todas as projeções do histograma tridimensional estão igualmente disponíveis.

```
SELECT De_Hospital AS Hospital,
       CASE WHEN de_origem *'hosp' THEN 'Hosp'
            WHEN de_origem *'lab' THEN 'Lab'
            WHEN de_origem *'intern' OR de_origem *'pronto' THEN 'Atend'
            ELSE 'Outro' END AS Origem,
       CASE WHEN De_Exame *'hemograma' THEN 'Hemograma'
            WHEN De_Exame *'colest' OR De_Exame *'tri.*glic' THEN 'Cholesterol'
            WHEN De_Exame *'(covid)|(pc.*r)|(igm)|(igg)' THEN 'Covid'
            ELSE 'Outros' END AS Exame,
       Count(*)
FROM examlabs
GROUP BY CUBE (1,2,3)
ORDER BY 1,2,3 Desc;
```

Exercício 6)

Escreva um comando em SQL para gerar uma amostragem de aproximadamente 1% das tuplas de tal maneira que a quantidade de tuplas seja (aproximadamente) a mesma para todas as classes (<Exame, Hospital> distintos) contabilizadas no exercício anterior (use um histograma bi-dimensional, sem considerar a [Origem](#)).

Resposta:

Dificuldade: 

NOTAS:

1. Neste exercício, dada a natureza essencialmente completa de cada operação intermediária (completa no sentido de que toda a relação envolvida é inteiramente acessada), o processamento segue uma sequência de passos. Eles estão indicados na sequência de criação de tabelas no comando [WITH](#):

- 1 [ADComp](#): Discretiza o atributo [DE.Exame](#) nos valores necessários;
- 2 [Histo](#): Gera o histograma bidimensional solicitado;
- 3 [Sequencia](#): Gera uma sequencia de identificação de cada célula do histograma e calcula o tamanho do menor e do maior *bin*;
- 4 [Sample](#): Executa a amostragem das tuplas usando o valor das faixas calculado no passo anterior;
- 5 [RESULTADO](#): Conta a quantidade de tuplas selecionadas de cada *bin* do histograma original.

```
WITH ADComp AS (SELECT *,
    CASE WHEN De.Exame *'hemograma' THEN 'Hemograma'
    WHEN De.Exame *'colest' OR De.Exame *'tri.*glic' THEN 'Colesterol'
    WHEN De.Exame *'(covid)|(pc.*r)|(igm)|(igg)' THEN 'Covid'
    ELSE 'Outros' END AS Exame
FROM examlabs),
Histo AS (SELECT De.Hospital, Exame,
    Count(*) Tot
FROM ADComp
GROUP BY 1,2),
Sequencia AS (SELECT Row_Number() OVER (ORDER BY De.Hospital, Exame),
    MAX(Tot) OVER() Ma,
    MIN(Tot) OVER() Mi,
    *
FROM Histo),
Sample AS (SELECT P.*
FROM ADComp P JOIN Sequencia S
    ON (P.De.Hospital, P.Exame)=(S.De.Hospital, S.Exame)
    WHERE 100.0*Random()::FLOAT8<1.-Tot::FLOAT8/(Ma-Mi) )
SELECT De.Hospital, Exame, Count(*)
FROM Sample
GROUP BY 1,2
```

Exercício 7)


Considere que a relação

<code>Hemograma={ID_Paciente, Basofilos, Bastonetes, Blastos, CHCM, Eosinofilos, ... VolPlaq}</code>
--

foi criada na base de dados, tal como definida no **Exercício 8** da **Primeira Lista de Exercícios** da matéria. Para cada atributo, obtenha:

- seu tipo de dado,
- a quantidade de nulos,
- a cardinalidade,
- o valor mínimo,
- o valor máximo,
- o valor médio,
- a variância,
- e o desvio padrão

Resposta:

Dificuldade: 

NOTAS:

1. A solução corresponde a modificar a função `MinhasEstatisticas()` mostrada em aula para que ela retorne as informações pedidas, e usá-la sobre a tabela criada.
2. A função, já modificada, está mostrada no final desta lista de exercícios.
3. As informações pedidas podem ser listadas com:

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



```

=====
-- Caetano Traina Júnior -- maio de 2021 ----- =====
-- Definir Table_Data_Statistics('tabela') ==
--     - Retorna estatísticas a respeito da tabela dada (compara 'tabela' por igualdade ==
--     - Exemplo: SELECT * FROM Table_Data_Statistics('schema.aluno'); ==
--     - Se 'schema.' não for dado, assume o esquema corrente ==
-- Retorna para cada atributo de 'tabela': ==
--     Nome Atr, Tipo Atr, Numero de nulos, Numero de distincts, ==
--     Valor minimo, Valor máximo, ==
--     e se o atributo é numérico, também a variância e o desvio padrão. ==
-----
DROP FUNCTION IF EXISTS Table_Data_Statistics;
DROP FUNCTION IF EXISTS Table_Data_Statistics;
CREATE FUNCTION Table_Data_Statistics(Tab TEXT) RETURNS
    TABLE(NomeAtrib TEXT, Tipo TEXT, Nulls INTEGER, Cardinality INTEGER,
            MinVal TEXT, MaxVal TEXT, AvgVal DOUBLE PRECISION,
            Variance DOUBLE PRECISION, StdDev DOUBLE PRECISION) AS $$
DECLARE Var_r Record; Var_Cmd TEXT; Var_Cmd2 TEXT; NSchema TEXT;
BEGIN
    SELECT SPLIT_PART(Tab, '.', 1), SPLIT_PART(Tab, '.', 2) INTO NSchema, Tab;
    IF TAB='' THEN TAB=NSchema; SELECT Current_Schema INTO NSchema; END IF;
    Var_Cmd='SELECT A.AttName::TEXT AN, T.TypName::TEXT ATy
    FROM pg_Class C, pg_attribute A, pg_type T , pg_namespace S
    WHERE C.RelName NOT LIKE ''pg_%'' AND C.RelName NOT LIKE ''sql_%'' AND
    C.RelKind=''r'' AND S.Oid=C.RelNameSpace AND S.NspName=''||NSchema||'''' 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 ''||NSchema||''. ''||Tab||
        ' WHERE ''||Var_r.AN||'' IS NULL;';
        EXECUTE Var_Cmd2 INTO Nulls;
        Var_Cmd2:='SELECT Count(DISTINCT ''||Var_r.AN||''), ' ';
        Var_Cmd2:=Var_Cmd2||'Min(''||Var_r.AN||'')::TEXT, Max(''||Var_r.AN||'')::TEXT, ' ';
        IF Var_r.ATy IN('int2', 'int4', 'int8', 'float4', 'float8', 'numeric') THEN
            Var_Cmd2:=Var_Cmd2||'AVG(''||Var_r.AN||''), Var_Pop(''||Var_r.AN||
            ''),stddev_pop(''||Var_r.AN||'')'; ELSE
            Var_Cmd2:=Var_Cmd2||'NULL, NULL, NULL'; END IF;
        Var_Cmd2:=Var_Cmd2||' FROM ''||NSchema||''. ''||Tab||';';
        EXECUTE Var_Cmd2 INTO Cardinality, MinVal, MaxVal, AvgVal, Variance, StdDev;
        NomeAtrib:=Var_r.AN;
        Tipo:=Var_r.ATy;
        RETURN NEXT;
    END LOOP;
END; $$ LANGUAGE plpgsql;

```

2 Indexação

Exercício 8) Considere as restrições de integridade fundamentais do modelo relacional (representadas pelas restrições **PRIMARY KEY**, **UNIQUE** e **FOREIGN KEY** nos SGBDs).

1. Quais restrições usam índices para avaliar a corretude dos comandos de atualização? Porque?
2. Existe uma situação comum de demora acentuada nos comandos **DELETE** que se deve à não criação de um índice para uma das restrições fundamentais.

- Identifique e explique qual é a situação que leva à essa demora;
- Qual índice poderia ser criado para evitar essa demora;
- Dê um exemplo usando a base *Repositório COVID-19 DataSharing/BR*;
- Porque tal índice não é automaticamente criado quando a restrição é declarada.

Resposta 8.1:

Dificuldade:

- As restrições que usam índices são as baseadas em unicidade das chaves: a restrição de chave primária (**PRIMARY KEY**) ou qualquer restrição de chave candidata (**UNIQUE**). Elas garantem que não existe duplicação de valores nos comandos **INSERT** e **UPDATE**, verificando no índice se o novo valor a ser colocado ainda não existe. Sem o índice, é necessário executar uma busca sequencial e ler todas as tuplas da tabela para garantir que o valor ainda não existe.
- Ao contrario de uma crença bastante popular, as tabelas **não precisam** ter chaves declaradas: elas só são necessárias para garantir automaticamente a unicidade dos valores atualizados. Caso a unicidade possa ser garantida pela própria aplicação, ou a violação da restrição não tenha importância, a declaração de unicidade não precisa ser declarada. Isso agiliza muito as operações de atualização, e pode ser usada quando uma aplicação precisa ingerir rapidamente uma grande quantidade de informação, deixando o processo de validação para outro momento mais “calmo”, quando as restrições podem ser então (re-)ativadas e validadas.

Resposta 8.2:

Dificuldade:

- Qual é a situação que leva à essa demora: remover tuplas em relações que são referenciadas por uma restrição de chave estrangeira (**FOREIGN KEY**).
Seja uma restrição **AtrE FOREIGN KEY RelD(AtrD)** declarada na relação **AtrE**. Ela obriga que **AtrD** seja chave primária em **RelD**. Agora foi solicitado um comando **DELETE FROM AtrD ...**.
A demora se deve à necessidade de procurar na relação **RelE** se não existe tupla cujo valor de **AtrE** seja igual a **AtrD**: A procura ocorre na relação **RelE**, para evitar que **AtrE** fique “órfão”. O atributo **AtrE** não precisa ser chave e pode não estar em índice algum, causando uma busca sequencial.
- O índice que poderia ser criado para evitar essa demora: é sobre o(s) atributo(s) **AtrE** da relação **RelE**, mesmo sem ser **UNIQUE**.
- O Atributo **ID_Atendimento** é a chave primária da relação **Desfechos**, e o atributo com o mesmo nome na relação **ExamLabs** o tem como chave estrangeira. Dessa maneira, se for removido um desfecho, é necessário constatar que não existe nenhum exame que referencie esse desfecho. Se existir, a ação indicada por **ON DELETE** da restrição deve ser executada para cada uma das tuplas que o referencia, sendo necessário procurá-las na relação **ExamLabs**. Para agilizar a procura, deve ser criado o índice:

CREATE INDEX FK_ID_Atend ON ExamLabs(ID_Atendimento)
- Porque esse índice não é automaticamente criado quando a restrição é declarada: Atributos que são chave estrangeira são em geral “importantes” o suficiente para serem frequentemente usados em consultas, mas não necessariamente sozinhos. Portanto, é muito comum que sejam criados índices onde eles estão envolvidos, e frequentemente em chaves de acesso composta. Assim, é melhor delegar o gerenciamento de tais índices ao DBA, que tem uma visão mais ampla das variadas maneiras como o atributo **AtrE** é usado nas diversas consultas.

Exercício 9) Seja o comando em SQL que cria a tabela de Exames de Hemograma, como definida no Exercício 8 da primeira lista de exercícios.

1. Crie os índices necessários para agilizar a execução desse comando.

Exercício 10) Considere um comando para associar a cada exame, a identificação do paciente, com sua idade e com o desfecho do atendimento onde o exame foi executado.

1. Escreva o comando.
2. Crie os índices necessários para agilizar a execução desse comando.
3. Existe algum índice, associado às restrições de integridade indicada pelos meta-dados da definição do *Repositório COVID-19 DataSharing/BR* que já auxiliam essa consulta? Quais e para que parte do comando?