



MBA em Inteligência Artificial e Big Data
– Curso 3: Administração de Dados Complexos em Larga Escala –
★ Preparação de dados Agregados em SQL ★

Caetano Traina Júnior

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

Usando as opções **CUBE** e **ROLLUP** na cláusula **GROUP BY** do comando **SELECT** em SQL.

Roteiro



- 1 Recordação: A Cláusula GROUP BY
- 2 Extensões da cláusula GROUP BY para análise multidimensional



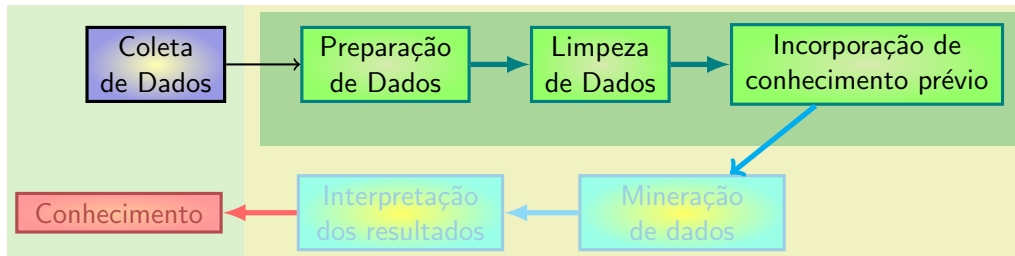
DML – Comando SELECT

Preparação de dados para mineração



Objetivo: Preparar dados para os processos de Mineração de dados.

Ferramenta: SQL – A linguagem universal para **gerenciamento, armazenagem, controle e recuperação** de dados.



DML – Comando SELECT

Realiza as consultas em uma base de dados



Comando SELECT – Realiza as consultas em uma base de dados.

Comando SELECT: sintaxe geral

```
SELECT [ALL | DISTINCT] <lista_atributos>
      FROM <nome_tabelas> | <tabelas_joined>
      [WHERE <condicao>]
      [GROUP BY <lista_atributos>
        [HAVING <condicao>]]
      [UNION · EXCEPT ...]
      [LIMIT <expressão>]
      [ORDER BY <lista_atributos> [ASC|DESC]];
```

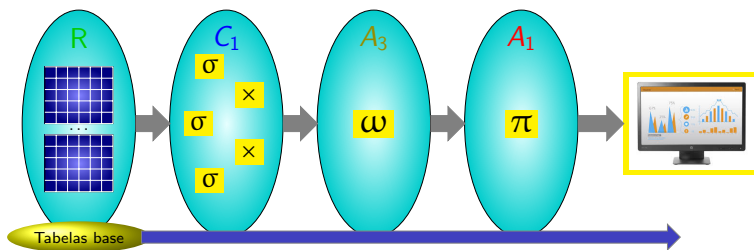


Ordem de execução dos comandos

Execução de consultas sem usar a cláusula GROUP BY




```
SELECT  $A_1$   
FROM  $R$   
[WHERE  $C_1$ ]  
[ORDER BY  $A_3$ ];
```




DML – Comando SELECT



- As cláusulas de **projeção dos atributos** de interesse, **junção das relações** que contêm os dados a serem trabalhados, e a **seleção das tuplas** que devem ser escolhidas,

 todas atuam sobre os dados originais.

- Embora existam muitos recursos disponibilizados por essas cláusulas para o tratamento de dados, nesta parte do curso assumimos que eles já são conhecidos, e não entramos em detalhes.
- Nesta aula, nosso interesse é tratar a preparação de dados por processos de **Agregação de dados**,

 usando os recursos da cláusula **GROUP BY**.



DML – Comando SELECT

Realiza as consultas em uma base de dados



- ★ Quando a cláusula **GROUP BY** é usada, é produzida uma relação com grupos de tuplas.

GROUP BY – agrupa os dados resultantes de uma seleção de acordo com um critério específico.

Terminologia: Agrupamento

Operação de “agrupar” subconjuntos de tuplas que tenham o mesmo valor numa determinada expressão sobre seus atributos.

Terminologia: Funções de Agregação

Funções de agregação recebem como argumento um atributo (singelo ou composto) e retornam um valor que sumariza todos os valores que esse atributo assume em todas as tuplas da relação de entrada.

DML – Comando SELECT

Realiza as consultas em uma base de dados



Comando SELECT: sintaxe geral

```
SELECT [ALL | DISTINCT] <lista_atributos>,  
       <funcao_agreg (atributo)>  
FROM <nome_tabelas> | <tabelas_joined>  
[WHERE <condicao>]  
[GROUP BY <lista_atributos>  
  [HAVING <condicao>]]  
[ORDER BY <lista_atributos> [ASC|DESC]];
```

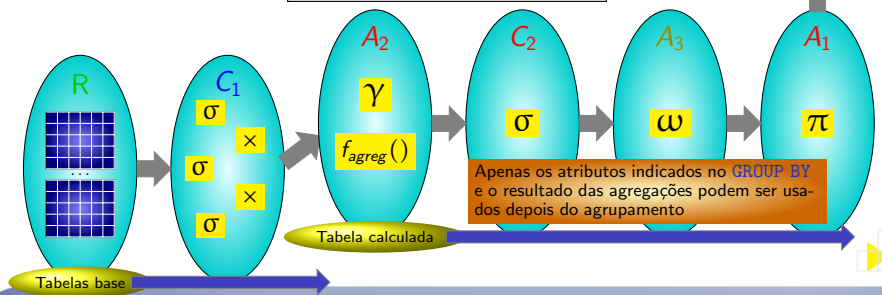


Ordem de execução dos comandos



Execução de consultas usando a cláusula GROUP BY



```
SELECT  $A_1$   
FROM  $R$   
[WHERE  $C_1$ ]  
[GROUP BY  $A_2$   
[HAVING  $C_2$ ]]  
[ORDER BY  $A_3$ ];
```





- A sumarização dos dados (de toda a tabela ou de cada grupo) é feita por **Funções de agregação**.
 - Funções de agregação do padrão SQL:
 - SUM(expr)
 - MIN(expr)
 - MAX(expr)
 - AVG(expr)
 - COUNT([DISTINCT] expr)
 - COUNT(*)
 - São comuns também, entre outras:
 - Desvio padrão: STDDEV_POP(expr)
 - Variância: VAR_POP(expr)
 - Mediana: MEDIAN(expr) 
 - = Percentile_Disc(.5) WITHIN GROUP (ORDER BY expr) 
 - Coeficiente de correlação do par: CORR(exprY, exprX)
 - ...



Seleção - GROUP BY - Exemplos

Agrupar os dados de uma tabela

- Listar a **média das idades** dos **Alunos** de cada cidade

```
SELECT Cidade, Count(*), Trunc(Avg(Idade),2)
FROM Aluno
GROUP BY Cidade
ORDER BY Cidade;
```

Disciplinas	Count(*)	Avg(Idade)
Araraquara	3	21.33
Campinas	2	19.00
Ibate	1	35.00
Ibitinga	1	21.00
Rio Claro	2	22.50
Sao Carlos	4	23.25
(null)	2	24.00



Seleção - GROUP BY - Exemplos

Agrupar os dados de várias tabelas

- Selecionar, para cada aluno, seu nome e a média das notas das disciplinas **em que ele foi aprovado (nota ≥ 5)**.

```
SELECT A.Nome, AVG(M.Nota)
      FROM Aluno A JOIN Matricula M
                        ON A.NUSP = M.NUSP
     WHERE M.Nota BETWEEN 5.0 AND 10.0
     GROUP BY A.Nome;
```

Nome	Avg(M.Nota)
Celso	7,5
Corina	9,75
Celina	6,5
Carlitos	5
Cibele	7
Cicero	7



Seleção - GROUP BY - Exemplos

Agrupar os dados de uma tabela, selecionando alguns grupos

- Selecionar, o nome e a média das notas das disciplinas em que foi aprovado (nota ≥ 5), dos alunos aprovados **em pelo menos duas disciplinas**.

```
SELECT A.Nome, AVG(M.Nota)
FROM Aluno A JOIN Matricula M
      ON A.NUSP = M.NUSP
WHERE M.Nota BETWEEN 5.0 AND 10.0
GROUP BY A.Nome
      HAVING COUNT(*)>=2;
```

Nome	Avg(M.Nota)
Celso	7,5
Corina	9,75
Cibele	7
Cicero	7





Agrupar os dados de uma tabela, selecionando alguns grupos

- Selecionar os nomes dos alunos que fizeram uma mesma disciplina mais de uma vez. Listar também o nome da disciplina, o número de vezes que ele cursou e a nota máxima que obteve (considerando todas as vezes que cursou).

```
SELECT A.Nome, D.Nome,  
       Count(*), Max(M.Nota)  
FROM Aluno A JOIN Matricula M  
      ON A.NUSP = M.NUSP  
JOIN Turma T  
      ON T.Codigo = M.CodigoTurma  
JOIN Disciplina D  
      ON D.Sigla = T.Sigla  
GROUP BY A.Nome, D.Nome  
HAVING Count(*)>1;
```

A.Nome	D.Nome	Count(*)	Max(M.Nota)
Celso	Algebra	2	7
Celina	Algebra	3	6,2
Cicero	Algebra	2	6.6
Carlitos	Algebra	2	5.1





- 1 Recordação: A Cláusula GROUP BY
- 2 Extensões da cláusula GROUP BY para análise multidimensional
 - O Modelo de Dados Multidimensional
 - Comandos ROLLUP e CUBE
 - Comparando ROLLUP e CUBE
 - Dependências entre atributos
 - Funções para conjuntos de agrupamento



O Modelo de Dados Multidimensional



- Uma aplicação voltada para a **análise dos dados** pode ser modelada considerando diversas **dimensões** ;
- Nesse caso, pode-se representar os dados usando um **Modelo Multidimensional** de dados;
- Cada **dimensão** representa um maneira de identificar (indexar) um aspecto de interesse para a análise dos dados;
 - Dimensões frequentemente utilizadas para a análise são: tempo, aspectos de distribuição Geográfica (localidade dos institutos), *Produtos* (Disciplinas), etc.;
- O conceito é que os dados formam um **“hiper-cubo”**, e a análise é feita observando as **projeções** das agregações de suas células nas faces e arestas do cubo;
- As projeções **“agregam”** os dados de cada *array* de células projetadas, usando as funções de agregação disponíveis.



O Modelo de Dados Multidimensional



Os comandos ROLLUP e CUBE:

- Indicam projeções “do lado do cubo visível”, o qual pode ser “rolado” (ROLLUP);
- ou indicam projeções de todo o cubo (CUBE).

```
Matricula=(Aluno, Disciplina, Semestre, Nota, Frequencia)
```

- A análise de uma tabela considera que ela tem diversos atributos que definem as **Dimensões**,

```
Matricula={Aluno, Disciplina, Semestre,...}
```

- e os atributos, usualmente numéricos, que definem os dados a serem agregados, chamados **Fatos**:

```
Matricula={..., Nota, Frequencia}
```

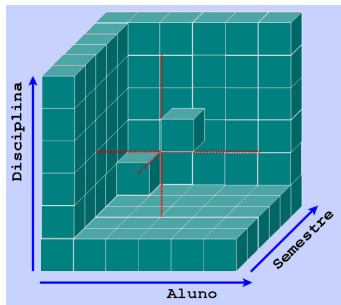


O Modelo de Dados Multidimensional



- O Hipercubo do modelo multidimensional considera cada tupla como um elemento num espaço com a dimensionalidade necessária;

`Matricula=(Aluno, Disciplina, Semestre, Nota, Frequencia)`



Nesse caso, **Aluno**, **Disciplina** e **Semestre** são parte da chave, e definem as dimensões do espaço, e **Nota** e **Frequencia** são os dados de cada ponto no espaço.

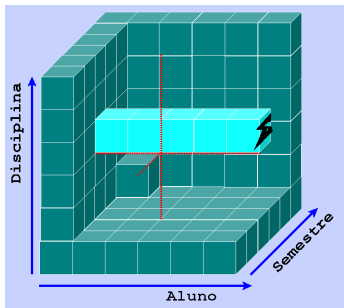


O Modelo de Dados Multidimensional



- O Hipercubo do modelo multidimensional considera cada tupla como um elemento num espaço com a dimensionalidade necessária;

`Matricula=(Aluno, Disciplina, Semestre, Nota, Frequencia)`



- Cada face do cubo mostra a agregação das tuplas que se projetam nessa face.
- Neste exemplo, a projeção agrega as tuplas de todos os **Alunos** para aquela **Disciplina** e **Semestre**.
- Nota: todas as faces paralelas têm a mesma informação.

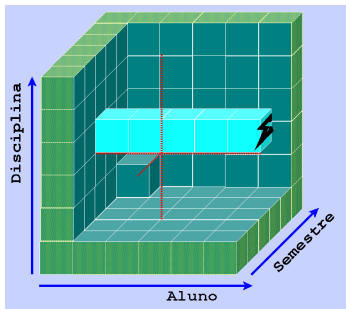


O Modelo de Dados Multidimensional



- O Hipercubo do modelo multidimensional considera cada tupla como um elemento num espaço com a dimensionalidade necessária;

`Matricula=(Aluno, Disciplina, Semestre, Nota, Frequencia)`



- As arestas projetam os dados de cada face. Nesse caso, existe uma aresta que agrega para cada **Aluno**, todos os dados de suas **Disciplina** e **Semestre**, e da mesma maneira as outras arestas mostram os dados de todas as **Disciplinas** e todos os **Semestres**.
- Nota: todas as arestas paralelas têm a mesma informação.

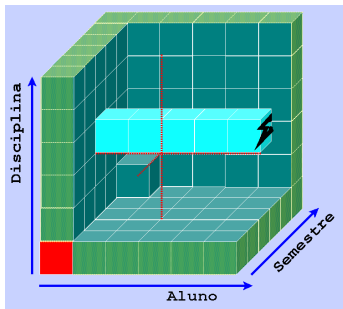


O Modelo de Dados Multidimensional



- O Hipercubo do modelo multidimensional considera cada tupla como um elemento num espaço com a dimensionalidade necessária;

`Matricula=(Aluno, Disciplina, Semestre, Nota, Frequencia)`



- Finalmente, o vértice do cubo, tem a agregação de todas as tuplas de todo o cubo.
- Nota: todos os vértices têm a mesma informação.



O Modelo de Dados Multidimensional

Comando GROUP BY - ROLLUP



Para considerar o modelo multidimensional em SQL:

- ROLLUP**
- produz um conjunto de resultados com as linhas agrupadas normais e os linhas de subtotais.
 - produz agregações cumulativas como subtotais em cada projeção

Comando SQL

```
SELECT <lista_atributos>
FROM . . .
[GROUP BY [<lista_atributos>], [ROLLUP](<lista_atributos>)]
[HAVING <condicao>]]
[ORDER BY <lista_atributos>];
```



O Modelo de Dados Multidimensional

Comando GROUP BY - ROLLUP



- Obter a lista de notas obtidas por cada aluno em cada disciplina, com as respectivas médias gerais de cada disciplina.

Aluno=(Nome, NUSP, Cidade, Idade)

Matricula=(CodigoTurma, NUSP, Nota)

Turma=(Sigla, Numero, Codigo, NNAlunos)

Disciplina=(Sigla, Nome, SiglaPreReq, NNCred)

Temp=(Disciplina.Sigla, Aluno.NUSP, Matricula.Nota)



O Modelo de Dados Multidimensional

Comando GROUP BY - ROLLUP



- Obter a lista de notas obtidas por cada aluno em cada disciplina, com as respectivas médias gerais de cada disciplina.

```
SELECT D.Sigla, A.Nome,  
       AVG(M.Nota) NotaF  
FROM Aluno A  
     JOIN Matricula M  
       ON A.NUSP = M.NUSP  
     JOIN Turma T  
       ON T.Codigo = M.CodigoTurma  
     JOIN Disciplina D  
       ON D.Sigla = T.Sigla  
GROUP BY ROLLUP (D.Sigla, A.Nome);
```

Lembre-se que cada aluno tem uma única nota por disciplina.

D.Sigla	A.Nome	NotaF
SCE-179	Celia	6.00
SCE-179	Celso	9.00
SCE-179	Cesar	9.00
SCE-179	Carlos	8.00
SCE-179	Celina	4.00
SCE-179	Cicero	7.00
SCE-179	Carlitos	7.00
SCE-179		7.14
SCE-200	Celia	9.00
SCE-200	Celso	7.00
SCE-200	Cesar	7.00
SCE-200	Carlos	4.00
SCE-200	Celina	8.00
SCE-200	Cibele	5.00
SCE-200	Cicero	10.00
SCE-200	Corina	9.00
SCE-200	Carlitos	4.00
SCE-200	Catarina	8.00
SCE-200		7.10
SMA-179	Celia	9.00
SMA-179	Celso	7.00
SMA-179	Cesar	9.00
SMA-179	Carlos	9.00
SMA-179	Celina	8.00
SMA-179	Cibele	6.00
SMA-179	Cicero	9.00
SMA-179	Corina	10.00
SMA-179	Carlitos	7.50
SMA-179	Catarina	7.00
SMA-179		8.07
		7.55

O Modelo de Dados Multidimensional

Comando GROUP BY - ROLLUP

- Obter as notas e a média geral de cada aluno.

```
SELECT D.Sigla, A.Nome,  
       AVG(M.Nota) NotaF  
FROM Aluno A  
     JOIN Matricula M  
       ON A.NUSP = M.NUSP  
     JOIN Turma T  
       ON T.Codigo = M.CodigoTurma  
     JOIN Disciplina D  
       ON D.Sigla = T.Sigla  
GROUP BY ROLLUP (A.Nome, D.Sigla);
```

D.Sigla	A.Nome	NotaF
SCE-179	Celia	6.00
SCE-200	Celia	9.00
SMA-179	Celia	9.00
	Celia	8.00
SCE-179	Celso	9.00
SCE-200	Celso	7.00
SMA-179	Celso	7.00
	Celso	7.50
SCE-179	Cesar	9.00
SCE-200	Cesar	7.00
SMA-179	Cesar	9.00
	Cesar	8.33
SCE-179	Carlos	8.00
SCE-200	Carlos	4.00
SMA-179	Carlos	9.00
	Carlos	7.00
SCE-179	Celina	4.00
SCE-200	Celina	8.00
SMA-179	Celina	8.00
	Celina	7.00
SCE-200	Cibele	5.00
SMA-179	Cibele	6.00
	Cibele	5.50
SCE-179	Cicero	7.00
SCE-200	Cicero	10.00
SMA-179	Cicero	9.00
	Cicero	8.75
SCE-200	Corina	9.00
SMA-179	Corina	10.00
	Corina	9.50
SCE-179	Carlitos	7.00
SCE-200	Carlitos	4.00
SMA-179	Carlitos	7.50
	Carlitos	6.50
SCE-200	Catarina	8.00
SMA-179	Catarina	7.00
	Catarina	7.50
		7.55



O Modelo de Dados Multidimensional

Comando GROUP BY - CUBE



- CUBE**
- Agrupa as linhas selecionadas com base nos valores de todas as combinações possíveis de expressões na especificação e retorna uma única linha de informação resumida para cada grupo.
 - Produz um conjunto de resultados com todas as linhas do **ROLLUP** mais todas as linhas com as demais combinações possíveis.

Comando SQL

```
SELECT <lista_atributos>
FROM . . .
  [GROUP BY [<lista_atributos>], [CUBE](<lista_atributos>)
    [HAVING <condicao>]]
[ORDER BY <lista_atributos>];
```

Seleção - GROUP BY - CUBE - Exemplo

- Obter a lista de notas obtidas por cada aluno em cada disciplina, com as respectivas médias gerais de cada disciplina e médias de cada aluno.

```
SELECT D.Sigla, A.Nome,  
       AVG(M.Nota) NotaF  
FROM Aluno A  
     JOIN Matricula M  
       ON A.NUSP = M.NUSP  
     JOIN Turma T  
       ON T.Codigo = M.CodigoTurma  
     JOIN Disciplina D  
       ON D.Sigla = T.Sigla  
GROUP BY CUBE (D.Sigla, A.Nome);
```

D.Sigla	A.Nome	NotaF
	Celia	8.00
	Celso	7.50
	Cesar	8.33
	Carlos	7.00
	Celina	7.00
	Cibele	5.50
	Cicero	8.75
	Corina	9.50
	Carlitos	6.50
SCE-179	Celia	6.00
SCE-179	Celso	9.00
SCE-179	Cesar	9.00
SCE-179	Carlos	8.00
SCE-179	Celina	4.00
SCE-179	Cicero	7.00
SCE-179	Carlitos	7.00
SCE-179		7.14
SCE-200	Celia	9.00
SCE-200	Celso	7.00
SCE-200	Cesar	7.00
SCE-200	Carlos	4.00
SCE-200	Celina	8.00
SCE-200	Cibele	5.00
SCE-200	Cicero	10.00
SCE-200	Corina	9.00
SCE-200	Carlitos	4.00
SCE-200	Catarina	8.00
SCE-200		7.10
SMA-179	Celia	9.00
SMA-179	Celso	7.00
SMA-179	Cesar	9.00
SMA-179	Carlos	9.00
SMA-179	Celina	8.00
SMA-179	Cibele	6.00
SMA-179	Cicero	9.00
SMA-179	Corina	10.00
SMA-179	Carlitos	7.50
SMA-179	Catarina	7.00
SMA-179		8.07
SMA-179		7.55



Comparando ROLLUP e CUBE



O efeito de um comando

```
SELECT a, b, c, f(x)
GROUP BY ROLLUP(a, b, c);
```

é equivalente ao comando

```
SELECT a, b, c, f(x)
GROUP BY a, b, c
UNION ALL
SELECT a, b, null, f(x)
GROUP BY a, b
UNION ALL
SELECT a, null, null, f(x)
GROUP BY a
UNION ALL
SELECT null, null, null, f(x);
```

Ou seja, $\text{ROLLUP}(a, b, c) \iff \text{fx}(a, b, c), \text{fx}(a, b), \text{fx}(a), \text{fx}()$



Comparando ROLLUP e CUBE



Efeito de um agrupamento por ROLLUP

Um `ROLLUP(a, b, c, d, ...)` gera o agrupamento de:

Todo o (hiper)cubo:

`fx()`

Uma aresta do (hiper)cubo:

`fx(a)`

Uma face bidimensional do (hiper)cubo:

`fx(a, b)`

Uma face tridimensional do (hiper)cubo:

`fx(a, b, c)`

Uma face quadridimensional do (hiper)cubo:

`fx(a, b, c, d)`

...

👉 Portanto um `ROLLUP` de n atributos é equivalente à união de $n+1$ agrupamentos.



Comparando ROLLUP e CUBE



Efeito de um agrupamento por CUBE

Um $CUBE(a, b, c)$ gera o agrupamento de:

Todo o cubo (ou hipercubo):	$fx()$
Todas as aresta do cubo:	$fx(a), fx(b), fx(c)$
Todas as faces do cubo:	$fx(a, b), fx(a, c),$ $fx(b, c)$
Todas as células do cubo:	$fx(a, b, c)$
...	

Portanto um $CUBE$ de n atributos é equivalente à união de 2^n agrupamentos.



Porque usar ROLLUP e CUBE



- Se ROLLUP e CUBE podem ser obtidos pela união de `GROUP BY`, porque usar esses comandos?

- 1 - Facilita e torna mais claro o comando, auxiliando também a manutenção;
- 2 - Agiliza a resposta.

Custo de agrupamentos

Gerar um agrupamento é uma operação cara, demorada. Gerar n agrupamentos demora n vezes o custo de um `GROUP BY`.

Todos os agrupamentos induzidos por ROLLUP e CUBE são gerados de uma vez só, portanto é pouco mais caro do que o custo de um `GROUP BY` só.



Generalizando os comandos ROLLUP e CUBE



Comando Conjuntos de Agrumento – Grouping sets

Cada agrupamento gerado por um comando ROLLUP ou CUBE é chamado um **Conjunto de Agrupamento**.

- Por exemplo, ROLLUP(a,b) gera três Conjuntos de agrupamentos, e CUBE(a,b) gera quatro Conjuntos de agrupamentos.
- Um terceiro comando da cláusula GROUP BY permite generalizar e especificar um por um os conjuntos de agrupamento desejados:

Comando SQL

```
SELECT [ALL | DISTINCT] <lista_atributos>
FROM <nome_tabelas> | <tabelas_joined>
[WHERE <condicao>]
[GROUP BY [<lista_atributos>], [GROUPING SETS](<lista_atributos>)]
[HAVING <condicao>]]
[ORDER BY <lista_atributos> [ASC|DESC]];
```


Generalizando os comandos ROLLUP e CUBE

Por exemplo, o comando:

```
SELECT Cidade, Idade, Count(*)  
FROM Aluno A  
GROUP BY GROUPING SETS (Cidade, Idade, ());
```

produz os agrupamentos: `Cidade`, `Idade` e `()`, gerando:

Cidade	Idade	COUNT(*)
(null)	22	2
(null)	25	1
(null)	21	4
(null)	20	1
(null)	23	2
(null)	35	1
(null)	19	1
Araraquara	(null)	3
Campinas	(null)	1
Ibate	(null)	1
Ibitinga	(null)	1
Rio Claro	(null)	2
Sao Carlos	(null)	4
(null)	(null)	12

Não produz, p.ex., o agrupamento (`cidade, idade`), que o `CUBE` produziria.

Agrupamentos `GROUPING SETS` também são processados numa operação só.



Sintaxe geral para ROLLUP, CUBE e GROUPING SETS

Um docinho sintático

- Os comandos **ROLLUP**, **CUBE** e **GROUPING SETS** seguem a sintaxe geral da cláusula **GROUP BY**

- Dado um comando **SELECT**:

```
SELECT  $A_1$ 
FROM  $R$ 
[WHERE  $C_1$ ]
[GROUP BY  $A_2$ 
  [HAVING  $C_2$ ]]
[ORDER BY  $A_3$ ];
```

- Embora a execução do comando comece processando a lista de relações R da cláusula **FROM**,

☞ a análise sintática começa processando a lista de atributos A_1 .

- Assim, cada elemento da lista A_1 fica numerada,

☞ **e as listas de atributos seguintes – A_2 e A_3 – podem se referenciar a eles pelo seu número ordinal.**



Sintaxe geral para ROLLUP, CUBE e GROUPING SETS

Um docinho sintático



Consulta:

Liste quantos alunos são de cada cidade e idade, ordenando pela idade e pelo nome da cidade entre os que têm mesma idade.

Os dois comandos geram exatamente o mesmo resultado:

```
SELECT Cidade, Idade, Count(*)  
FROM Aluno  
GROUP BY 1, 2  
ORDER BY 1, 2;
```



```
SELECT Cidade, Idade, Count(*)  
FROM Aluno  
GROUP BY Cidade, Estado  
ORDER BY Cidade, Estado;
```



Sintaxe geral para ROLLUP, CUBE e GROUPING SETS

Um docinho sintático

- podem ser usados tanto atributos da relação quanto expressões:

```
SELECT CASE
    WHEN Grau < 'MS-3' THEN 'Sem Doutorado'
    WHEN Grau = 'MS-3' THEN 'Doutor'
    WHEN Grau > 'MS-3' THEN 'Pós-Doutor'
    ELSE 'Desconhecido' END AS 'Oque',
    Count(*)
FROM professor
GROUP BY ROLLUP (1);
```

Oque	COUNT(*)
(null)	10
Desconhecido	1
Pós-Doutor	2
Doutor	4
Sem Doutorado	3



Dependências entre atributos



- Nos exemplos mostrados até agora, cada dimensão é constituída de apenas um atributo, e cada uma é independente das demais.
- É comum que ocorram duas variações:
 - Uma dimensão: • é expressa por mais de um atributo;
 - pode ter diversos níveis de granularidade (hierarquia).



Dependências entre atributos

Vários atributos por dimensão



- Pode ocorrer de uma dimensão corresponder a diversos atributos.
- Por exemplo, suponha que professores podem ter nomes repetidos, e que usa-se sua data de nascimento e departamento para desempatar ((Nome, DataNasc, Depto) é **UNIQUE**), e existe a relação das disciplinas que professores ministram:

```
Professor=(NFunc, Nome, DataNasc, Depto, Idade, Nivel)
```

```
Ministra=(NFunc, Sigla, Curso, Depto, NNCred)
```

- Suponha que se queira obter o total de disciplinas ministradas por curso e o total delas ministradas por professor e por curso. Pode-se emitir o comando:

```
SELECT M.Curso, P.NFunc, Count(M.Curso, P.NFunc) Total  
FROM Professor P JOIN Ministra M  
ON P.NFunc=M.NFunc  
GROUP BY GROUPING SETS (M.Curso, P.NFunc);
```

o que gera:

Dependências entre atributos

Vários atributos por dimensão



```
SELECT M.Curso, P.NFunc, Count(M.Curso, P.NFunc) Total
FROM Professor P JOIN Ministra M
ON P.NFunc=M.NFunc
GROUP BY GROUPING SETS (M.Curso, P.NFunc);
```

M.Curso	P.NFunc	Total
Computação	(null)	12
E. Elétrica	(null)	15
Matemática	(null)	9
(null)	17398298	4
(null)	49174021	3
(null)	70298334	1
(null)	90233208	5



Dependências entre atributos

Vários atributos por dimensão

- Se for necessário que os professores sejam identificados por seu (Nome, DataNasc, Depto), pode-se escrever:

```
SELECT M.Curso,  
       P.Nome, P.DataNasc, P.Depto, Count(M.Curso, P.NFunc) Total  
FROM Professor P JOIN Ministra M  
  ON P.NFunc=M.NFunc  
GROUP BY GROUPING SETS (M.Curso, (P.Nome, P.DataNasc, P.Depto));
```

(Note o parêntesis juntando os atributos que formam um único grupo.)

M.Curso	P.Nome	P.DataNasc	P.Depto	Total
Computação	(null)	(null)	(null)	12
E. Elétrica	(null)	(null)	(null)	15
Matemática	(null)	(null)	(null)	9
(null)	Pedro	12-04-1982	SCC	4
(null)	Paulo	06-11-1985	SCC	3
(null)	Pitágoras	14-01-1959	SMA	1
(null)	Péricles	22-11-1961	SEL	5



Dependências entre atributos

Dimensões Hierárquicas



- O modelo de dados multidimensional pode ser generalizado no modelo **Floco de neve**.
- Isso significa que uma determinada dimensão pode ser composta por mais de um atributo, de tal maneira que ela pode ser refinada em diversos níveis hierárquicos.
- Por exemplo, pode-se querer analisar as disciplinas pelos institutos e pelos departamentos a que elas estão vinculadas, e os professores podem ser por sua vez agrupados por seu nível.



Dependências entre atributos

Dimensões Hierárquicas



Professor=(NFunc, Nome, DataNasc, Depto, Idade, Nivel)

Ministra=(NFunc, Sigla, Ano, Semestre)

Discip=(Sigla, Nome, Curso, Depto, Nivel, NumCred)

- Por exemplo, suponha que se quer analisar quantas disciplinas foram ministradas pelos institutos e departamentos:

ROLLUP(Curso, Depto, Sigla)

e também, pelos professores e seus níveis:

ROLLUP(Nível, NFunc)

```
SELECT D.Curso, D.Depto, D.Sigla,  
       P.Nível, P.NFunc, Count(D.Sigla, P.NFunc) Total  
FROM Professor P JOIN Ministra M ON P.NFunc=M.NFunc  
                JOIN Disciplina D ON M.Sigla=D.Sigla  
GROUP BY ROLLUP(D.Curso, D.Depto, D.Sigla), ROLLUP(P.Nível, P.NFunc);
```

Dependências entre atributos

Dimensões Hierárquicas



```
SELECT DCurso, D.Depto, D.Sigla,  
       P.Nível, P.NFunc, Count(D.Sigla, P.NFunc) Total  
FROM Professor P JOIN Ministra M ON P.NFunc=M.NFunc  
       JOIN Disciplina D ON M.Sigla=D.Sigla  
GROUP BY ROLLUP(DCurso, D.Depto, D.Sigla), ROLLUP(P.Nível, P.NFunc);
```

DCurso	D.Depto	D.Sigla	P.Nível	P.NFunc	Total
Computação	SCC	SCC-224	MS-3	17398298	2
Computação	SCC	SCC-179	MS-3	49174021	1
Elétrica	SEL	SCC-179	MS-3	17398298	1
Matemática	SMA	SMA-179	MS-3	70298334	1
Elétrica	SEL	SMA-179	MS-3	17398298	1
...					
Computação	SCC	SCC-224	MS-3	(null)	2
Computação	SCC	SCC-224	(null)	(null)	3
Elétrica	SEL	(null)	MS-3	17398298	1
Computação	SCC	(null)	MS-3	(null)	4
Computação	SCC	(null)	(null)	(null)	6
Elétrica	(null)	(null)	MS-3	17398298	0
Computação	(null)	(null)	MS-3	(null)	6
Computação	(null)	(null)	(null)	(null)	8
(null)	(null)	(null)	MS-3	17398298	1
(null)	(null)	(null)	MS-3	(null)	9
(null)	(null)	(null)	(null)	(null)	12

Dependências entre atributos

Dimensões Hierárquicas



- Veja que o comando

```
GROUP BY ROLLUP(a, b, c), ROLLUP(x, y);
```

gera o produto cartesiano dos agrupamentos gerados por cada operador de agrupamento:

a	b	c	x	y	Σ
α	β	γ	χ	ϵ	*
α	β	γ	χ	(null)	*
α	β	γ	(null)	(null)	*
α	β	(null)	χ	ϵ	*
α	β	(null)	χ	(null)	*
α	β	(null)	(null)	(null)	*
α	(null)	(null)	χ	ϵ	*
α	(null)	(null)	χ	(null)	*
α	(null)	(null)	(null)	(null)	*
(null)	(null)	(null)	χ	ϵ	*
(null)	(null)	(null)	χ	(null)	*
(null)	(null)	(null)	(null)	(null)	*



Dependências entre atributos

Dimensões Hierárquicas



- O mesmo vale para qualquer operador de conjunto de agrupamentos (**ROLLUP**, **CUBE**, **GROUPING SETS**), e qualquer quantidade de comandos de agrupamento.
- Atributos colocados fora de comandos de agrupamento na cláusula **GROUP BY** formam um agrupamento. Por exemplo:

```
GROUP BY x, CUBE(a, b), y;
```

gera:

x	a	b	y	Σ
χ	α	β	ϵ	*
χ	(null)	β	ϵ	*
χ	α	(null)	ϵ	*
χ	(null)	(null)	ϵ	*



Dependências entre atributos

Dimensões Hierárquicas

- Um Hipercubo em que as dimensões são hierárquicas pode ser expressa numa cláusula **GROUP BY** em que cada dimensão é expressa por um comando de **ROLLUP** com os atributos da hierarquia na ordem do atributo com granularidade mais ampla para a mais refinada.
- Por exemplo, numa aplicação que tem as dimensões tempo, geografia e produto expressa pelos atributos:
 - Tempo: {Ano, Semestre, Mes, Semana}
 - Geografia: {Estado, Cidade}
 - Produto: {Instituto, Departamento, Grupo}

pode ter um hipercubo gerada pelo comando:

```
GROUP BY ROLLUP(Ano, Semestre, Mes, Semana)  $n_1 = 4$ ,  
          ROLLUP(Estado, Cidade)  $n_2 = 2$ ,  
          ROLLUP(Instituto, Departamento, Grupo)  $n_3 = 3$ 
```

gerando $(n_1 + 1) * (n_2 + 1) * (n_3 + 1) = 5 * 3 * 4 = 60$ agrupamentos distintos.



Dependências entre atributos

Dimensões Hierárquicas



- Um Hipercubo em que as dimensões são hierárquicas pode ser expressa numa cláusula **GROUP BY** em que cada dimensão é expressa por um comando de **ROLLUP** com os atributos da hierarquia na ordem do atributo com granularidade mais ampla para a mais refinada.
- Por exemplo, numa aplicação que tem as dimensões tempo, geografia e produto expressa pelos atributos:
 - Tempo: {Ano, Semestre, Mes, Semana}
 - Geografia: {Estado, Cidade}
 - Produto: {Instituto, Departamento, Grupo}

pode ter um hipercubo gerada pelo comando:

```
GROUP BY CUBE(Ano, Semestre, Mes, Semana)  $n_1 = 4$ ,  
         CUBE(Estado, Cidade)  $n_2 = 2$ ,  
         CUBE(Instituto, Departamento, Grupo)  $n_3 = 3$ 
```

gerando $2^{n_1+1} * 2^{n_2+1} * 2^{n_3+1} = 2^5 * 2^3 * 2^4 = 2^{12} = 4096$ agrupamentos distintos.

Funções para conjuntos de agrupamento



- Comandos para gerar conjuntos de agrupamento trazem dois problemas:
 - 1 Como saber se **nulls** correspondem aos nulos criados pelos comandos ou se correspondem a nulos que estão armazenados na base de dados?
 - 2 Como identificar precisamente as tuplas do resultado que correspondem a cada nível de agrupamento?



Funções para conjuntos de agrupamento



```
SELECT Cidade, Idade, Count(*)  
FROM Aluno A  
GROUP BY GROUPING SETS (Cidade, Idade, ());
```

Seja o comando:

Se todos os professores têm idade e cidade conhecidas:

Cidade	Idade	COUNT(*)
(null)	22	2
(null)	25	1
(null)	21	4
(null)	20	1
(null)	23	2
(null)	35	1
(null)	19	1
Araraquara	(null)	3
Campinas	(null)	1
Ibate	(null)	1
Ibitinga	(null)	1
Rio Claro	(null)	2
Sao Carlos	(null)	4
(null)	(null)	12

Se é inserido um professor sem idade nem cidade conhecidas:

Cidade	Idade	COUNT(*)
(null)	22	2
(null)	25	1
(null)	(null)	1
(null)	21	4
(null)	20	1
(null)	23	2
(null)	35	1
(null)	19	1
Araraquara	(null)	3
Campinas	(null)	1
Ibate	(null)	1
Ibitinga	(null)	1
Rio Claro	(null)	2
Sao Carlos	(null)	4
(null)	(null)	1
(null)	(null)	13

Funções para conjuntos de agrupamento



- A função **GROUPING** pode ser aplicada a um atributo (resultante de uma operação de agrupamento):

```
SELECT ... [GROUPING(<Atrib>)] ...  
      GROUP BY ... CUBE | ROLLUP | GROUPING SETS(<Atrib>)
```

- ela retorna o inteiro (1) quando **<Atr>** é um **null** criado por um operador de conjunto de agrupamento, e retorna zero para qualquer outro valor, incluindo um **null** armazenado.



Funções para conjuntos de agrupamento



```
SELECT CASE WHEN GROUPING(Cidade)=1 THEN  
        '-' ELSE Cidade END AgCidade,  
        CASE WHEN GROUPING(Idade)=1 THEN  
        '-' ELSE Idade::CHAR END AgIdade,  
        Count(*)  
FROM Aluno A  
GROUP BY GROUPING SETS (Cidade, Idade, ());
```

AgCidade	AgIdade	COUNT(*)
-	22	2
-	25	1
-	(null)	1
-	21	4
-	20	1
-	23	2
-	35	1
-	19	1
Araraquara	-	3
Campinas	-	1
Ibate	-	1
Ibitinga	-	1
Rio Claro	-	2
Sao Carlos	-	4
(null)	-	1
-	-	13



Funções para conjuntos de agrupamento



```
SELECT CASE WHEN GROUPING(Cidade)=1 THEN  
        '-' ELSE Cidade END AgCidade,  
        CASE WHEN GROUPING(Idade)=1 THEN  
        '-' ELSE Idade::CHAR END AgIdade,  
        Count(*)  
FROM Aluno A  
GROUP BY GROUPING SETS (Cidade, Idade, ())  
HAVING NOT((Cidade IS Null AND GROUPING(Cidade)=0) OR  
            (idade IS Null AND GROUPING(Idade)=0));
```

AgCidade	AgIdade	COUNT(*)
-	22	2
-	25	1
-	21	4
-	20	1
-	23	2
-	35	1
-	19	1
Araraquara	-	3
Campinas	-	1
Ibate	-	1
Ibitinga	-	1
Rio Claro	-	2
Sao Carlos	-	4
-	-	13

A função **GROUPING** pode ser usada também na cláusula **HAVING**.



Funções para conjuntos de agrupamento



- A função **GROUPING** não está no padrão ISO.
- Em **ORACLE** ela pode receber somente um atributo como argumento, e retorna 1 ou 0.
- Em **Postgres** ela pode receber qualquer número n de atributos como argumento, e retorna um inteiro entre 0 e $2^n - 1$.
Seu retorno corresponde ao número que em binário teria 1 para cada atributo que tenha (**null**) gerado por um operador de conjunto de agrupamento.
- Esse mesmo efeito é obtido em **ORACLE** pela função **GROUPING_ID**.



Funções para conjuntos de agrupamento



```
SELECT Cidade, Idade, Count(*),  
        GROUPING(Cidade) AgCid, GROUPING(Idade) AgId,  
        GROUPING_ID(Cidade,Idade) GrupoId  
FROM Aluno A  
GROUP BY GROUPING SETS (Cidade, Idade, ());
```

Cidade	Idade	COUNT(*)	AgCid	AgId	Grupold
(null)	22	2	1	0	2
(null)	25	1	1	0	2
(null)	(null)	1	1	0	2
(null)	21	4	1	0	2
(null)	20	1	1	0	2
(null)	23	2	1	0	2
(null)	35	1	1	0	2
(null)	19	1	1	0	2
Araraquara	(null)	3	0	1	1
Campinas	(null)	1	0	1	1
Ibate	(null)	1	0	1	1
Ibitinga	(null)	1	0	1	1
Rio Claro	(null)	2	0	1	1
Sao Carlos	(null)	4	0	1	1
(null)	(null)	1	0	1	1
(null)	(null)	13	1	1	3





MBA em Inteligência Artificial e Big Data
– Curso 3: Administração de Dados Complexos em Larga Escala –
★ **Preparação de dados Agregados em SQL** ★

Caetano Traina Júnior

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



Cube & Rollup
FIM