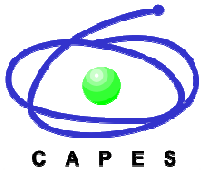
Índice Bitmap e Indexação de Ambientes de *Data Warehousing*

Jaqueline Joice Brito
jjbrito@icmc.usp.br

13 de Junho de 2013



Roteiro

- Índice Bitmap
- Técnicas de otimização
  - Adaptação da apresentação de Sérgio L. Díscola Jr., Thiago L. L. Siqueira e Vinícius Pereira sobre o **FastBit**
- *Data Warehouses* Convencionais
- Índice Bitmap de Junção
- Índice para *Data Warehouses* Geográficos: SB-index

Índice Bitmap

- Conceitualmente, um índice *bitmap* é um vetor de bits para cada chave do índice, em que cada bit substitui um *rowid*; se o bit é 1, então o correspondente *rowid* contém o valor da chave.

- Uma função de mapeamento converte a posição de um bit para o *rowid* real.

RowId	X
1	3
2	2
3	1
4	5
5	4
6	2
7	3

Índice Bitmap

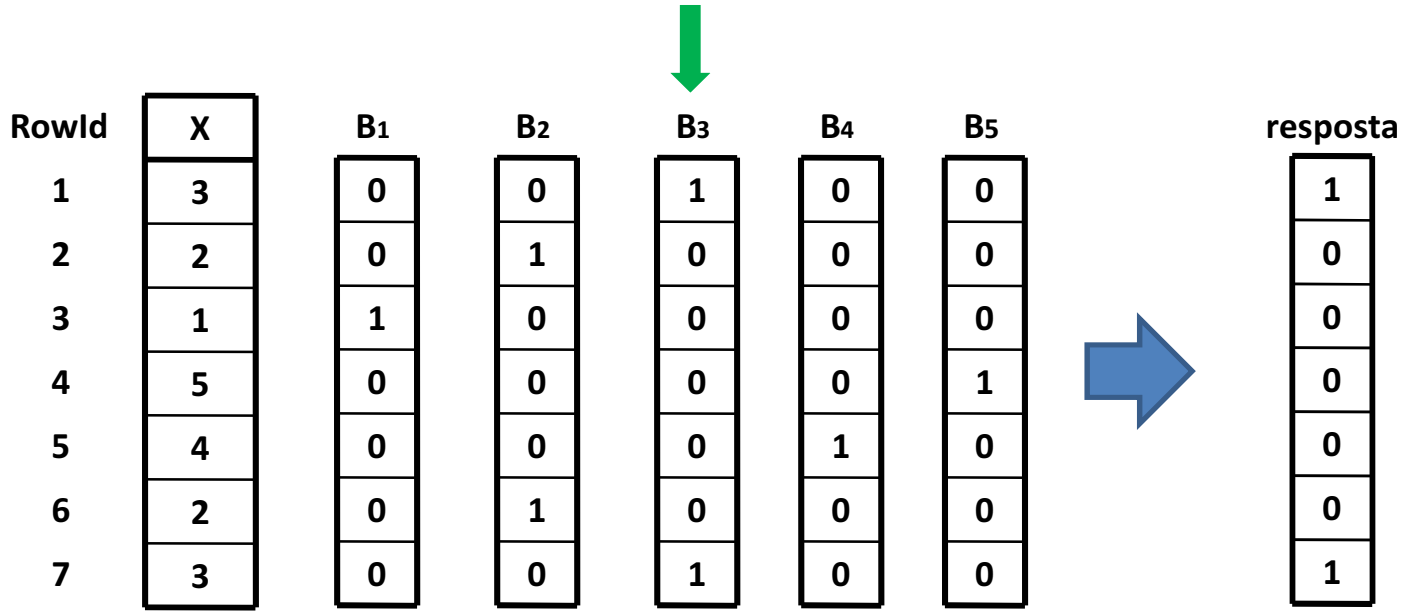
- Conceitualmente, um índice *bitmap* é um vetor de bits para cada chave do índice, em que cada bit substitui um *rowid*; se o bit é 1, então o correspondente *rowid* contém o valor da chave.

- Uma função de mapeamento converte a posição de um bit para o *rowid* real.

RowId	X	B_1	B_2	B_3	B_4	B_5
1	3	0	0	1	0	0
2	2	0	1	0	0	0
3	1	1	0	0	0	0
4	5	0	0	0	0	1
5	4	0	0	0	1	0
6	2	0	1	0	0	0
7	3	0	0	1	0	0

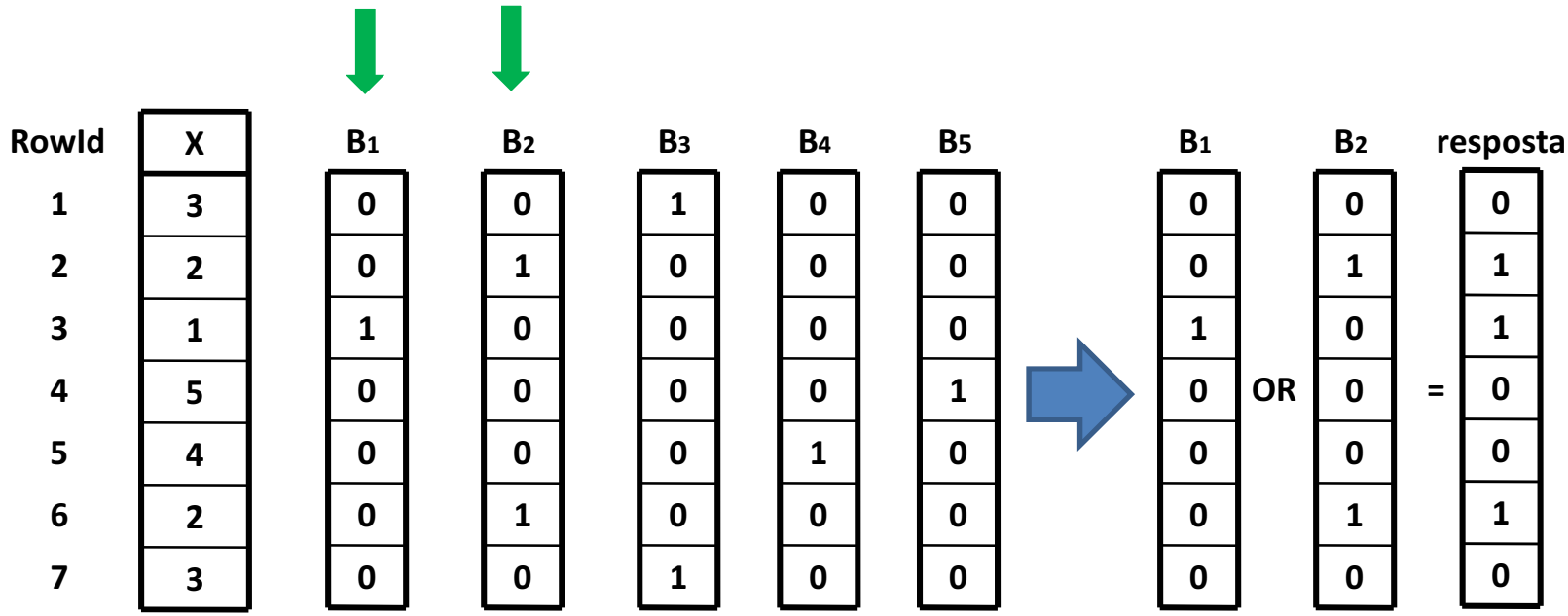
Índice Bitmap

- Exemplo: RowIds que possuem o valor X = 3?



Índice Bitmap

- Exemplo: RowIds que possuem o valor X <= 2?



Índice Bitmap

Vantagens:

- Tempos de respostas reduzidos para uma larga gama de consultas *ad hoc.*

- Espaço de armazenamento pequeno, quando comparado com outros tipos de índice.

Desvantagens:

- A atualização de índices *bitmap* não é tão eficiente quanto a de índices árvores-B, o que faz com que índices *bitmap* sejam indicados para BDs *read-only*, o que é o caso de DW.

- Para que haja ganho de espaço, é preciso que o número de valores da chave do índice seja pequeno.

Índice Bitmap

Bitmaps Esparsos

- Atributo A do tipo inteiro: 4 bytes por valor
- Cardinalidade de A: 1.000
- 5.000 tuplas na relação

➢ **Custo de armazenamento**

– Valores armazenados na relação

- 5.000x4 bytes = 20.000 bytes ≈ 20KB

Índice Bitmap

Bitmaps Esparsos

- Atributo A do tipo inteiro: 4 bytes por valor
- Cardinalidade de A: 1.000
- 5.000 tuplas na relação

➢ **Custo de armazenamento**

– Valores armazenados na relação

- 5.000x4 bytes = 20.000 bytes ≈ 20KB

– Índice Bitmap:

- 1.000x(5.000 +4x8) bits = 5.032.000/8 bytes ≈ 629KB

≈ **30 vezes mais espaço**

Codificação (*Encoding*) por igualdade e por faixa

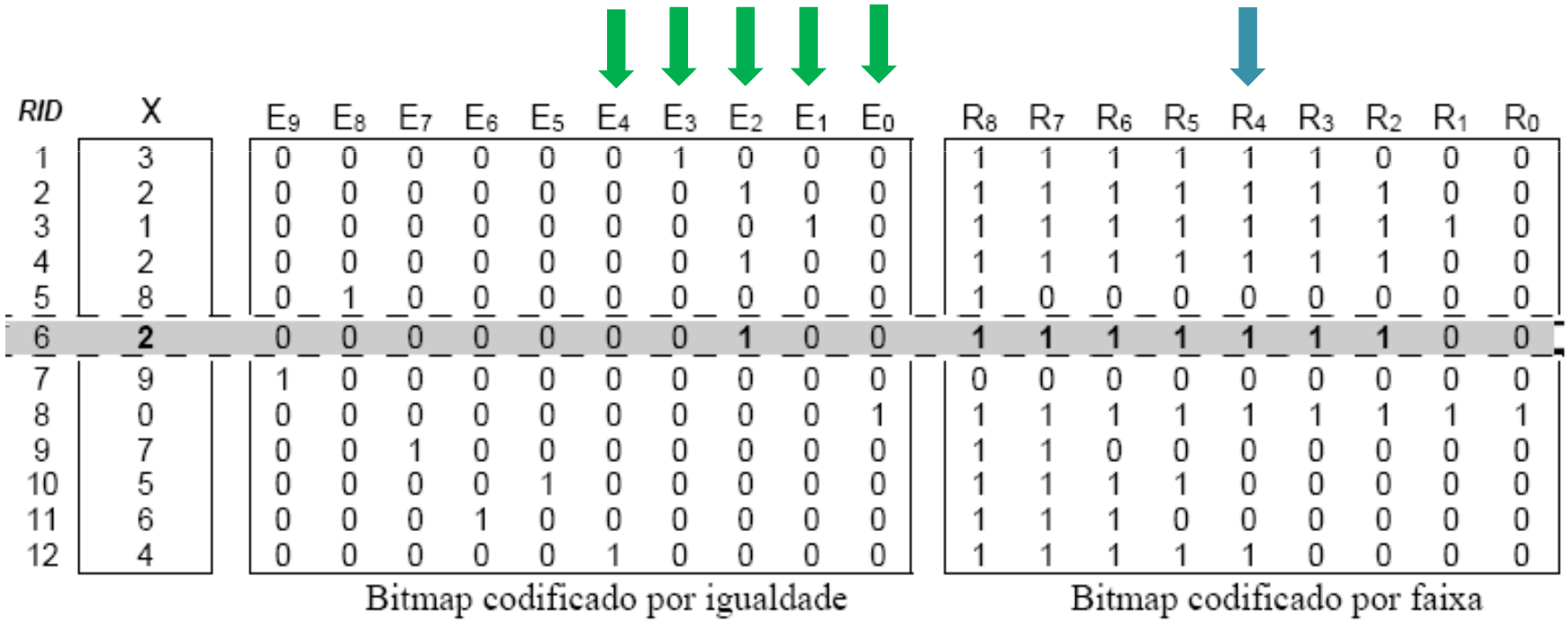
- Bitmap codificado por faixa de valores:
  - Exemplo: X $\geq$ 2
  - Nas linhas em que X$\geq$2, R_i (i$\geq$2) são fixados em 1 e R_j (j<2) em 0

RID	X	E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
1	3	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0
2	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
3	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0
4	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
5	8	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
6	**2**	0	0	0	0	0	0	0	**1**	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0
7	9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	7	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
10	5	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
11	6	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
12	4	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	0	0

Bitmap codificado por igualdade | Bitmap codificado por faixa

Codificação (*Encoding*) por igualdade e por faixa

- Consulta: X ≤4.
- Codificação por igualdade: acesso a 5 bitmaps.
- Codificação por faixa: acesso a 1 bitmap: R4.



RID	X	E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
1	3	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0
2	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
3	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0
4	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
5	8	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
6	**2**	0	0	0	0	0	0	0	**1**	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0
7	9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	7	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
10	5	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
11	6	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
12	4	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	0	0

Bitmap codificado por igualdade

Bitmap codificado por faixa

Codificação binária

- Índice bitmap codificado:
  - Codificação binária
  - Mantém no máximo $\log_2|X|$ arrays
  - Componentes: Tabela de mapeamento, Índice bitmap codificado e Funções de recuperação

- Exemplo:

X	B_1	B_0
a	0	0
b	0	1
c	1	0
b	0	1
a	0	0

a	00
b	01
c	10

Tabela de Mapeamento

Funções de recuperação

$a = B_1'B_0'$

$b = B_1'B_0$

$c = B_1B_0'$

Codificação binária

- **Processamento de consultas**

  Exemplo: tuplas onde X = a or X =b

  $B_1'B_0'$ **OR** $B_1'B_0 = B_1'$, ou seja, usar apenas a negação de B_1.

  B_1' = 11011, logo apenas a terceira linha não faz parte da resposta.

➢ Exemplo:

X	B_1	B_0
a	0	0
b	0	1
c	1	0
b	0	1
a	0	0

a	00
b	01
c	10

Tabela de Mapeamento

Funções de recuperação

a = $B_1'B_0'$

b = $B_1'B_0$

c = B_1B_0'

Binning

- Construção de um bitmap para uma caixa (*bin*), ao invés de para cada valor de atributo.
- Exemplo: criação de um índice para um atributo X, cujo domínio são os números reais.

X	[0,20) B_0	[20,40) B_1	[40,60) B_2	[60,80) B_3	[80,100) B_4
32,5	0	1	0	0	0
95,0	0	0	0	0	1
26,9	0	1	0	0	0
18,5	1	0	0	0	0
62,8	0	0	0	1	0
68,2	0	0	0	1	0
59,3	0	0	1	0	0

Binning

- Construção de um bitmap para uma caixa (*bin*), ao invés de para cada valor de atributo.
- Exemplo: criação de um índice para um atributo X, cujo domínio são os números reais.

X	[0,20) B_0	[20,40) B_1	[40,60) B_2	[60,80) B_3	[80,100) B_4	B_{cand}
32,5	0	1	0	0	0	1
95,0	0	0	0	0	1	0
26,9	0	1	0	0	0	1
18,5	1	0	0	0	0	0
62,8	0	0	0	1	0	1
68,2	0	0	0	1	0	1
59,3	0	0	1	0	0	1

Consulta: conte o número de linhas em que $35 \leq X \leq 65$.

- **Respostas podem possuir falsos candidatos: Refinamento!**

Binning

- Construção de um bitmap para uma caixa (*bin*), ao invés de para cada valor de atributo.
- Exemplo: criação de um índice para um atributo X, cujo domínio são os números reais.

X	[0,20) B_0	[20,40) B_1	[40,60) B_2	[60,80) B_3	[80,100) B_4	Resposta
32,5	0	~~1~~	0	0	0	~~1~~
95,0	0	0	0	0	1	0
26,9	0	~~1~~	0	0	0	~~1~~
18,5	1	0	0	0	0	0
62,8	0	0	0	1	0	1
68,2	0	0	0	~~1~~	0	~~1~~
59,3	0	0	1	0	0	1

Consulta: conte o número de linhas em que $35 \leq X \leq 65$.

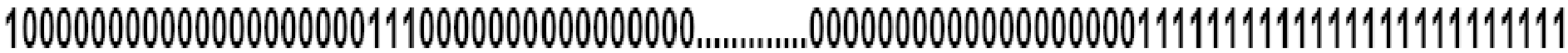
- **Respostas podem possuir falsos candidatos: Refinamento!**

Compressão WAH

- **Baseada em *run-lenght encoding* (RLE)**

- **Corrida:**
  - Preenchimento (*fill*): representa bits idênticos consecutivos usando RLE
  - Cauda (*tail*): 0's e 1's misturados, sem compressão
  - Uma corrida é composta por um preenchimento, ou por uma cauda, ou por um preenchimento seguido de uma cauda.

- **Determina preenchimentos e caudas para armazenar palavras (unidades operacionais do *hardware*)**



Compressão WAH

1000000000000000000001110000000 000000000.....0000000000000 00000011111111111111111111111111

- **Divisão**
  - 176 grupos de 31 bits: 176 * 31 = 5456 bits

- **Fusão de grupos vizinhos com bits idênticos**

31 bits: **1000000000000000000001110000000**

174*31 bits: **000000000.....0000000000000**

31 bits: **00000011111111111111111111111111**

- **Corridas:**

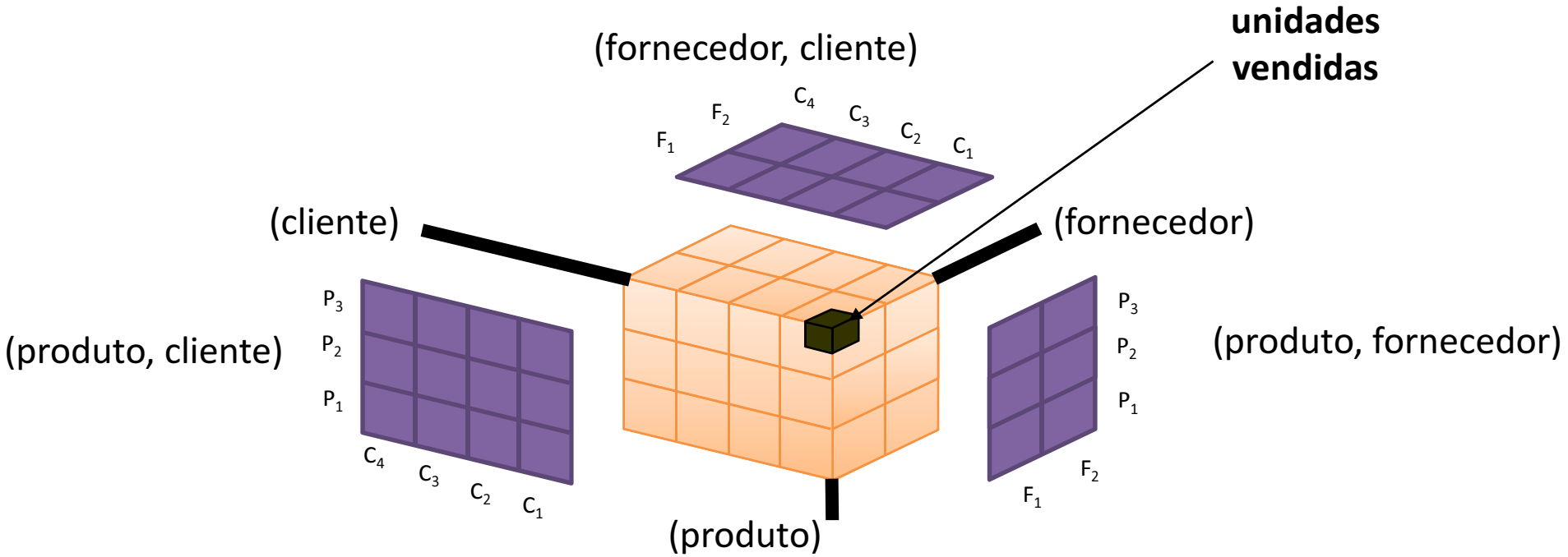
1ª corrida: **01000000000000000000001110000000**

2ª corrida: **10000000000000000000000010101110**

3ª corrida: **00000011111111111111111111111111**

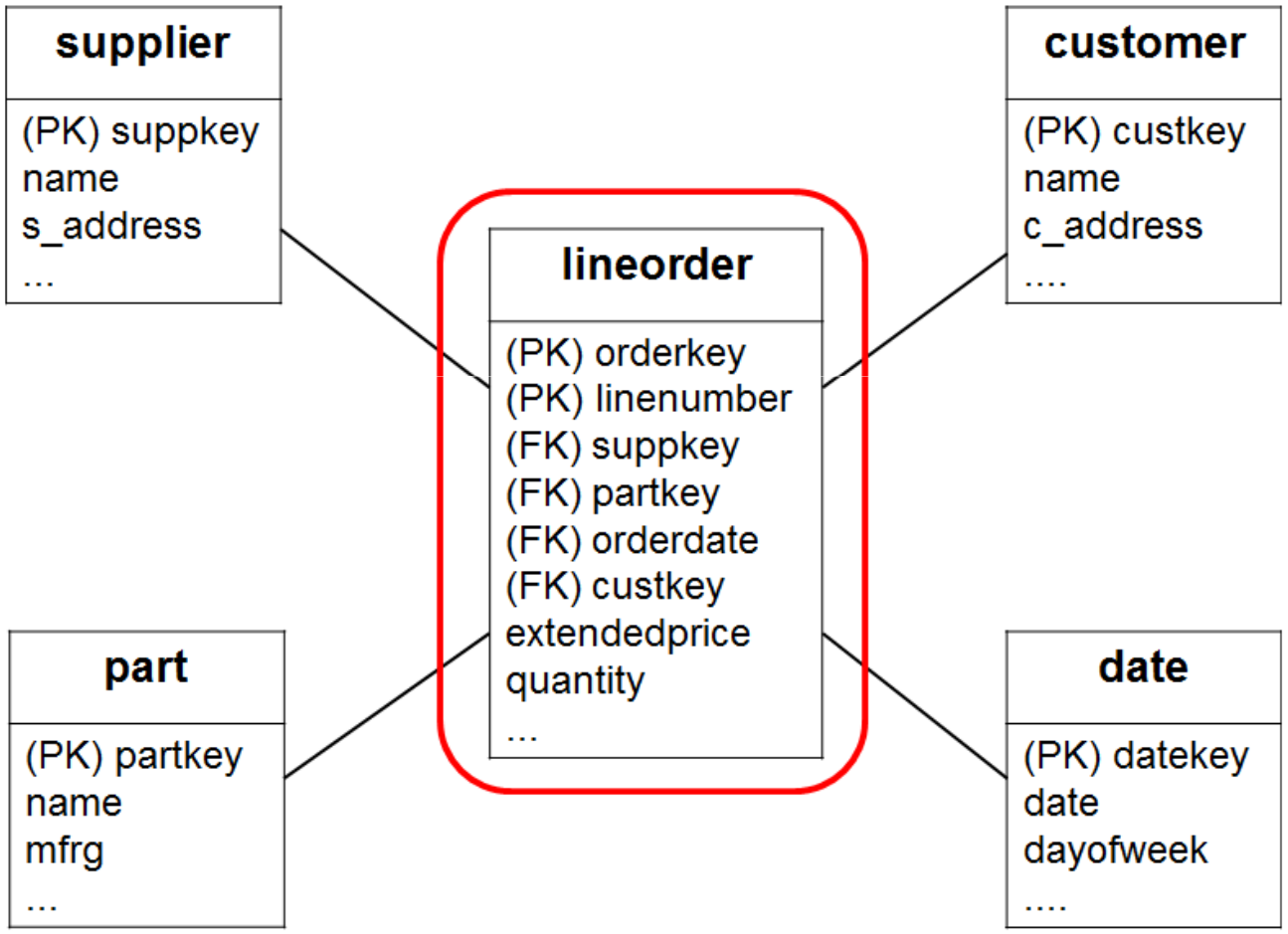
5456 bits foram reduzidos para 3 corridas de 32 bits = 96 bits

- Banco de dados voltado ao processamento analítico para a tomada de decisão
- Modelagem multidimensional
  - Medidas numéricas: objetos de análise
  - Dimensões: perspectiva/contexto para as análises
    - Hierarquia de atributos: categoria $\preceq$ marca $\preceq$ produto



Esquema Estrela

- B*enchmark* SSB



Processamentos de consultas OLAP

- Alto custo de processamento
  - **Grande volume** de dados
  - Junção estrela: várias **operações de junção**

- Otimização de consultas
  - Visões materializadas
    - Armazenamento físico de resultados de junções entre tabelas
  - Índices
    - Índice Bitmap de junção

Índice Bitmap de junção

Tabela de dimensão ***Supplier***

suppkey	*name*	*city*
1	Supp1	A
2	Supp2	B
3	Supp3	C
4	Supp4	A

Tabela de fatos ***Quantity***

suppkey	*partkey*	*quant*
1	1	2
2	1	3
3	2	7
4	2	12
1	3	5
2	3	2
4	3	9
1	5	10
2	5	4

Índice Bitmap de junção do atributo *city*

A	**B**	**C**
1	0	0
0	1	0
0	0	1
1	0	0
1	0	0
0	1	0
1	0	0
1	0	0
0	1	0

Índice Bitmap de junção

Tabela de dimensão ***Supplier***

suppkey	*name*	*city*
1	Supp1	A
2	Supp2	B
3	Supp3	C
4	Supp4	A

Qual é a quantidade total de unidades vendidas por fornecedores das cidades A e C?

Tabela de fatos ***Partsupp***

suppkey	*partkey*	*quant*
1	1	2
2	1	3
3	2	7
4	2	12
1	3	5
2	3	2
4	3	9
1	5	10
2	5	4

A		**C**			*quant*
1		0		1	2
0		0		0	~~3~~
0		1		1	7
1		0		1	12
1	OR	0	=	1	5
0		0		0	~~2~~
1		0		1	9
1		0		1	10
0		0		0	~~4~~

Resposta: 45

Data Warehouse Geográfico

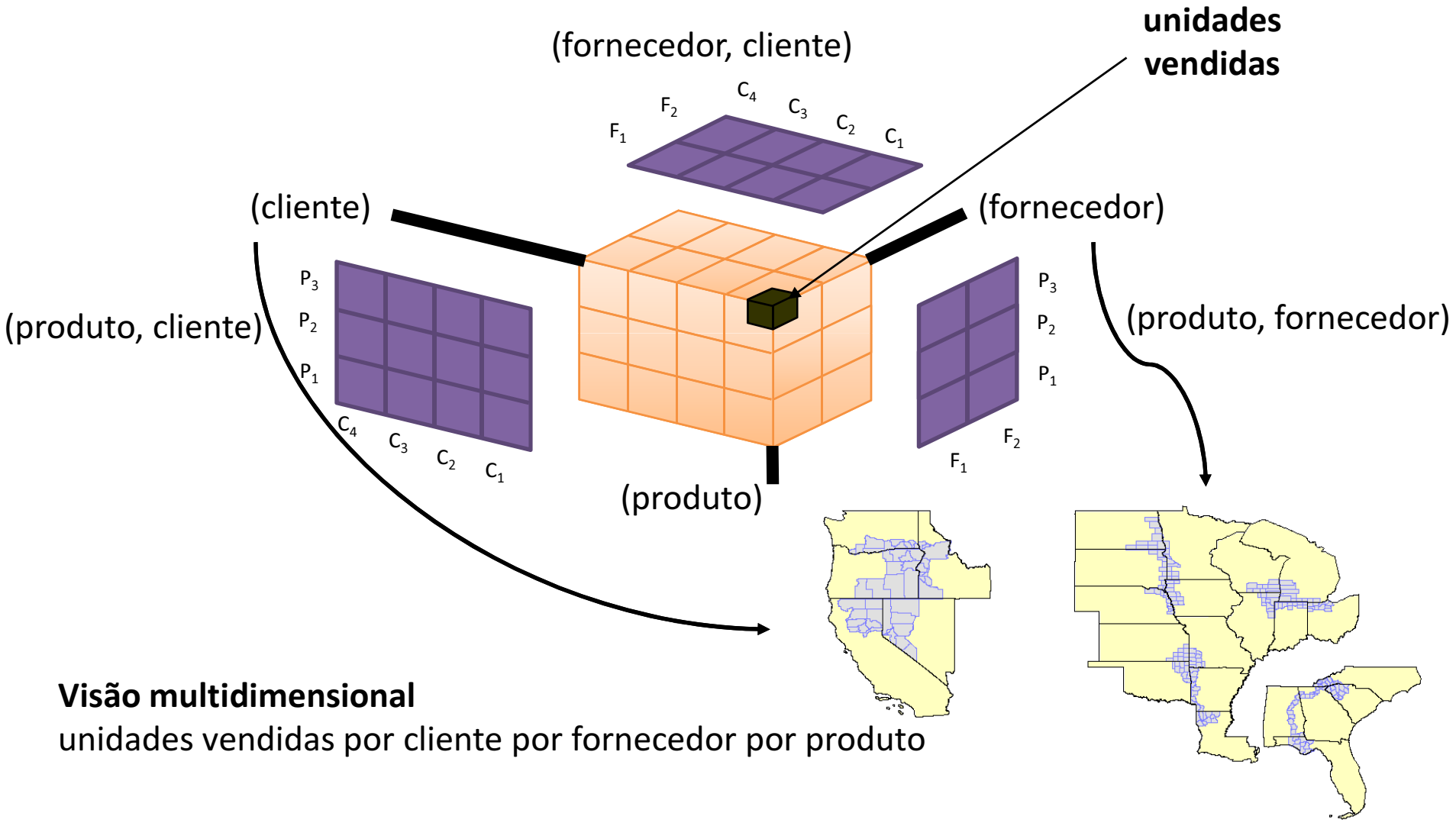
- Adição de dados espaciais

  - Armazenados como:
    - **Atributos** em dimensões
    - **Medidas** em tabela de fatos

  - Abstrações de **fenômenos do mundo real**
    - Cidades, endereços, regiões de plantações, etc
    - Representações básicas: **pontos, linhas e polígonos**

  - Análises georreferenciadas
    - Operações OLAP com **predicados espaciais**

Data Warehouse Geográfico

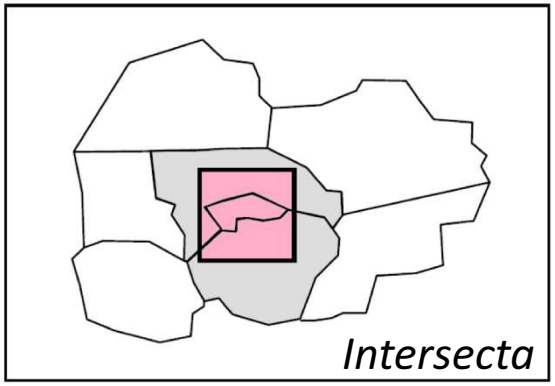


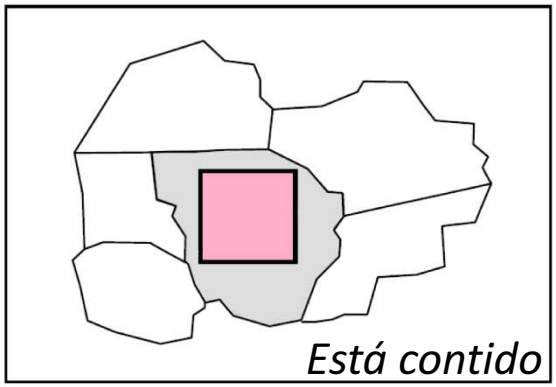
Visão multidimensional
unidades vendidas por cliente por fornecedor por produto

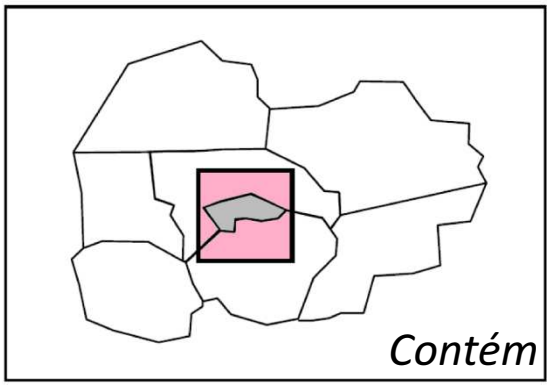
Range Queries

- Predicado espacial que considera o relacionamento topológico de uma **janela de consulta** e um conjunto de objetos espaciais

- Principais tipos de relacionamentos
  - *Intersecta*
  - *Está contido*
  - *Contém*

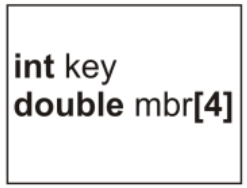




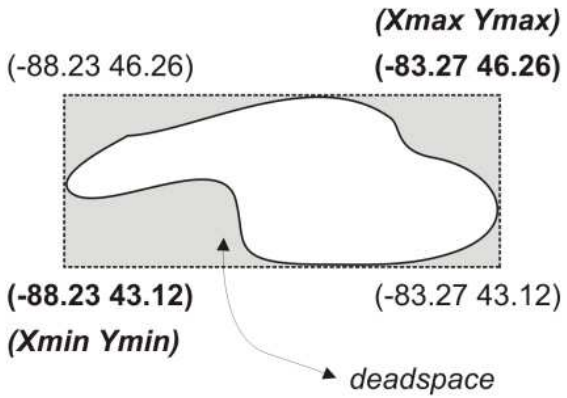


SB-index (Siqueira et al, SAC, 2009)

- É uma estrutura de indexação para DWG, implementado como um vetor unidimensional do tipo *sbitvector*.



tipo sbitvector



Objeto espacial e seu MBR

Vetor SB-index

1	2	3
-88.23	-85.16	-84.64
43.12	46.58	42.58
-83.27	-80.24	-79.45
46.26	49.71	49.27

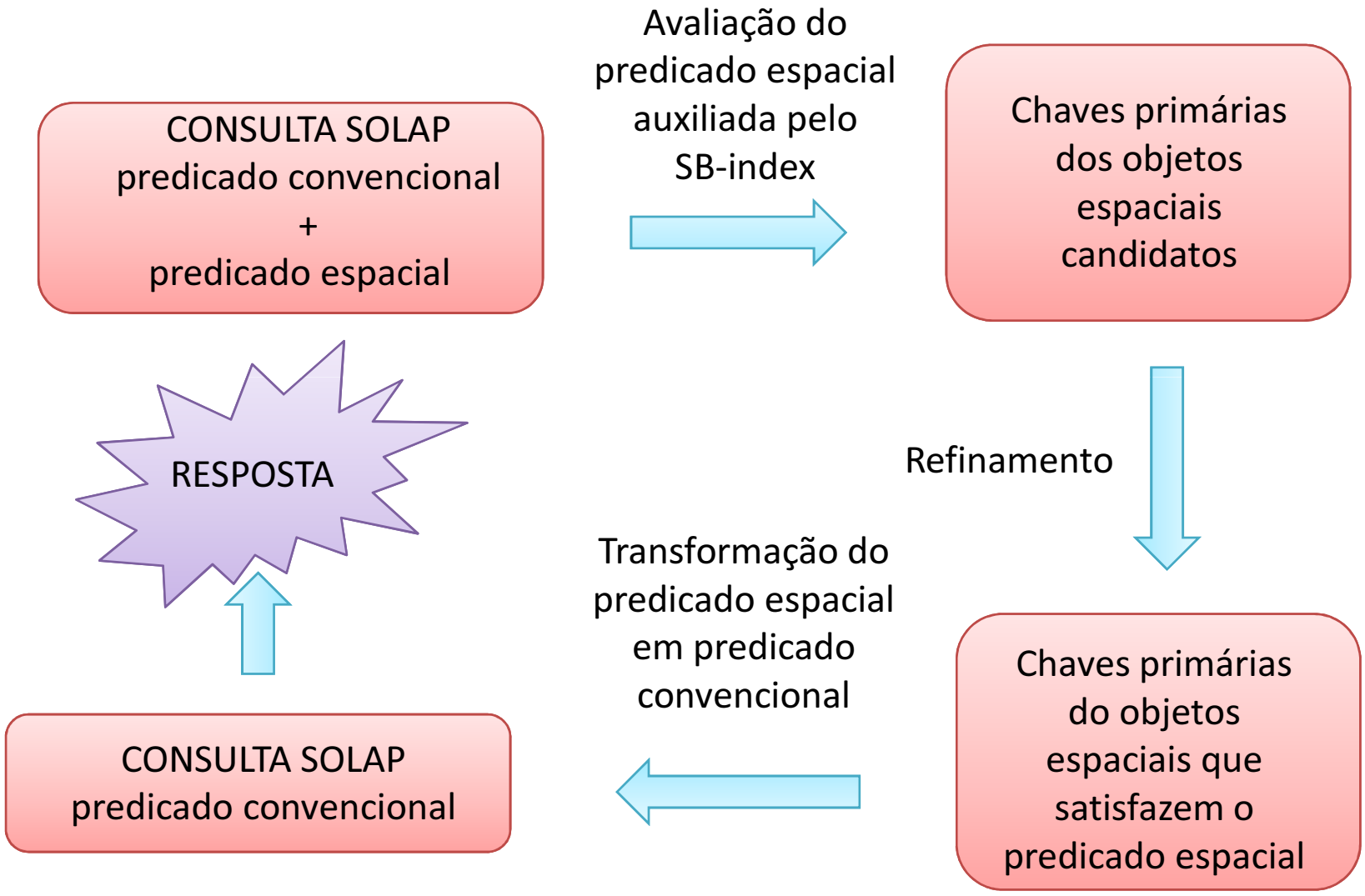
...

Índice Bitmap de junção

1	2	3
1	0	0
0	1	0
0	0	1
1	0	0
1	0	0
0	1	0
1	0	0
1	0	0
0	1	0

SB-index (Siqueira et al, SAC, 2009)

- Processamento de consultas



Referências

- O'Neil, P. E.; Graefe, G. Multi-table joins through bitmapped join indices. ACM SIGMOD Record, v. 24, n. 3, p. 8–11, 1995. Antoshenkov, G. Byte-aligned bitmap compression. In: CONFERENCE ON DATA COMPRESSION, 1995, Snowbird, UT, USA. Washington, DC, USA: IEEE Computer Society, 1995. p. 476.

- Chan, C. Y.; Ioannidis, Y. E. An efficient bitmap encoding scheme for selection queries. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1999, Philadelphia, Pennsylvania. New York, NY, USA: ACM, 1999. p. 215–226.

- Goyal, N.; Zaveri, S. K.; Sharma, Y. Improved bitmap indexing strategy for data warehouses. In: INTERNATIONAL CONFERENCE ON INFORMATION ECHNOLOGY, 9., 2006, Bhubaneswar, Orissa, India. Proceedings... Washington, DC, USA: IEEE Computer Society, 2006. p. 213–216.

- Brito, J. J.; Siqueira, T. L. L.; Times, V. C.; Ciferri, R. R.; Ciferri, C. D. A. Efficient processing of drill-across queries over geographic data warehouses. In: DaWaK, 2011. p. 152-166.

- Rotem, D.; Stockinger, K.; Wu, K. Optimizing candidate check costs for bitmap indices. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 2005, Bremen, Germany. New York, NY, USA: ACM, 2005. p. 648–655.

- Siqueira, T. L. L. SB-Index: um índice espacial baseado em Bitmap para data warehouse geográfico. Dissertação (Mestrado em Ciências da Computação), UFSCAR, São Carlos, Brasil, 2009.

Referências

- Siqueira, T. L. L.; Ciferri, R. R.; Times, V. C.; Ciferri, C. D. A. A spatial bitmap-based index for geographical data warehouses. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 24., 2009a, Honolulu, Hawaii, USA. New York, NY, USA: ACM, 2009a. p. 1336–1342.

- Stockinger, K.; Wu, K.; Shoshani, A. Evaluation strategies for bitmap indices with binning. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, 15., 2004, Zaragoza, Spain. Berlin/Heidelberg: Spring, 2004. p. 120–129.

- Wu, K.; Otoo, E. J.; Shoshani, A. Optimizing bitmap indices with efficient compression. ACM Transactions on Database Systems, v. 31, n. 1, p. 1–38, 2006.

- Wu, K.; Stockinger, K.; Shoshani, A. Breaking the curse of cardinality on bitmap indexes. In: INTERNATIONAL CONFERENCE ON SCIENTIFIC AND STATISTICAL DATABASE MANAGEMENT, 20., 2008, Hong Kong, China. Berlin/Heidelberg: Springer-Verlag, 2008. p. 348–365.

- Wu, K. FastBit: an efficient indexing technology for accelerating data-intensive science. Journal of Physics: Conference Series, v. 16, n. 1, p. 556–560, 2005.

- Wu, M.-C.; Buchmann, A. P. Encoded bitmap indexing for data warehouses. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 14., 1998, Orlando, Florida, USA. Washington, DC, USA: IEEE Computer Society, 1998. p. 220–230.

Índice Bitmap
e Indexação de Ambientes de *Data Warehousing*

Dúvidas?

ICMC USP SÃO CARLOS
Grupo de Bases de Dados e Imagens USP - São Carlos
CAPES
FAPESP
CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico