数据库系统概论新技术篇

数据仓库与联机分析处理技术(3)

陈红

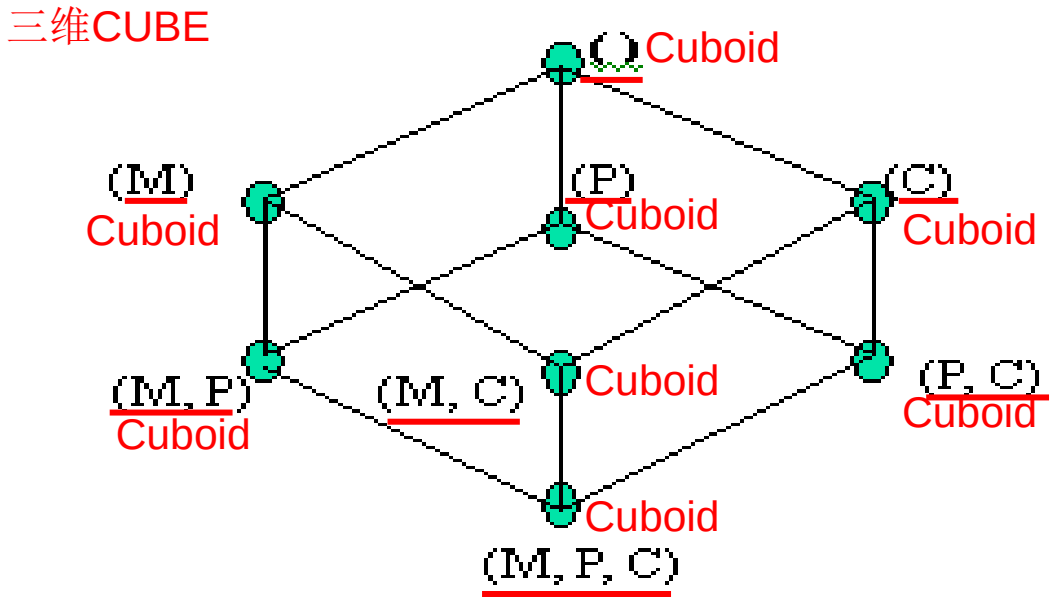
中国人民大学信息学院

数据仓库与OLAP的关键技术

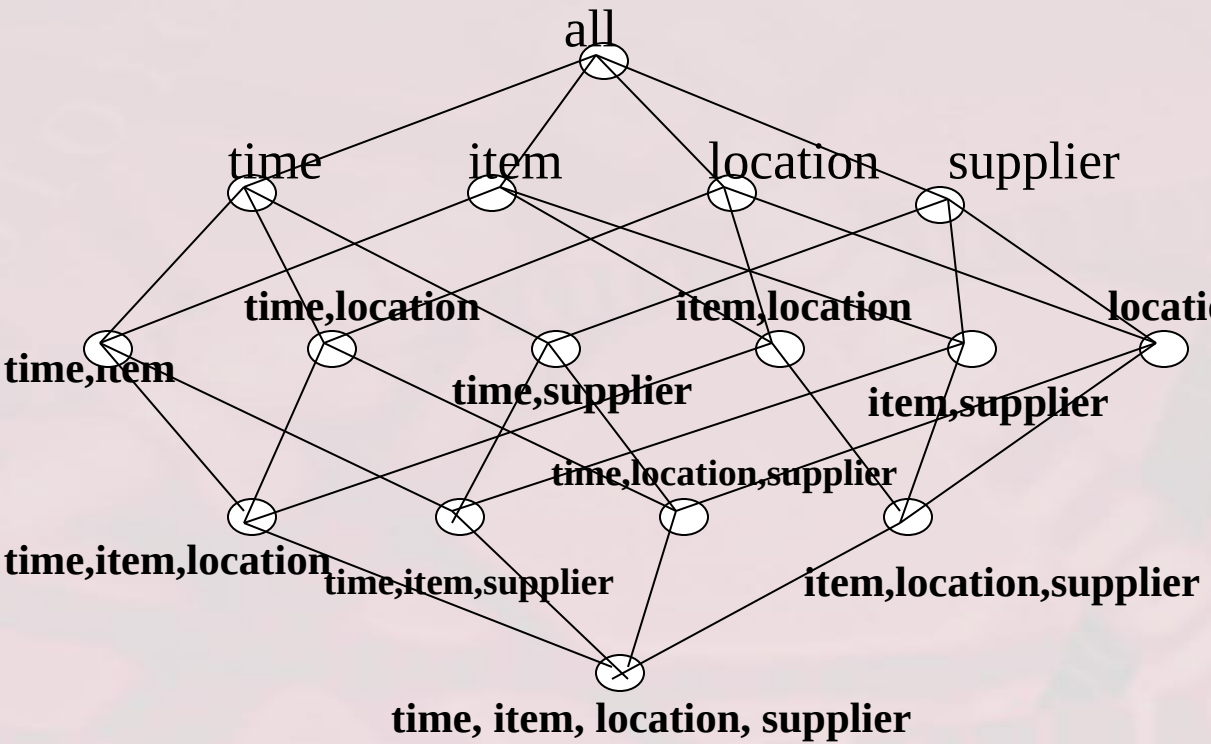
- 多维数据模型
- **CUBE计算技术**
- 实体化视图技术
- 精简数据方体技术
- 索引技术

CUBE计算技术

❖数据方体的格结构



Cube: A Lattice of Cuboids



0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D cuboids

4-D(base) cuboid

CUBE计算算法

- Naïve算法
- 2^N-Algorithm
- PipeSort算法
- PipeHash算法
- OverLap算法
- Partitioned-Cube算法
- ArrayCube算法
- BUC算法

减小Data cube体积的方法

- 选择部分数据方体实体化：**Greedy**
- 近似压缩：小波变换，多项式
- 精简数据立方体的方法：**Condensed Cube，Dwarf, Quotient Cube, QC-tree**等

数据仓库与OLAP的关键技术

- 多维数据模型
- CUBE计算技术
- 实体化视图技术
- 精简数据方体技术
- 索引技术

实体化视图技术

- 数据仓库包含了海量数据，OLAP服务器需要在秒级内回答决策支持查询，加速OLAP查询的主要方法之一是预计算一部分数据立方体，并对其进行实体化存储。称之为实体化视图。

- 实体化视图技术涉及的范围
  - 选择需要实体化的数据方体
  - 实体化视图维护
  - 利用实体化视图回答查询

实体化视图的选择

- 选择视图进行实体化需要考虑的因素

（1）选择的视图实体化后要有利于以后的查询操作

（2）选择的视图实体化时需要的空间开销要小

（3）对实体化的视图进行维护的开销要小

实体化视图选择方法

- 视图选择问题是一个NP-完全问题
- 实体化视图选择方法分类
  - 静态视图选择方法
  - 动态视图选择方法
  - 混合视图选择方法

静态视图选择方法(1)

- 基本思路
  - 在维护时间窗口内，根据查询的统计数据，把那些比较频繁发生的查询进行实体化，从而提高以后到达的查询的响应速度
  - 在下一个维护时间窗口到来之前，这些实视图不发生变化

静态视图选择方法(2)

- 静态视图选择的经典算法
  - 贪心算法
  - PBS算法
  - 基于AND-OR图的算法
  - Invert-tree 算法
  - 基因算法
  - 随机算法

静态视图选择方法的缺点

- 决策支持分析具有典型的动态特性，用户的分析型查询通常是难以预测的。
- 数据仓库中的数据和查询的特征都是随着时间而变化的,静态选择方法得到的结果可能很快就过期了。
- 系统没有办法改变一个错误的选择结果，更无法利用那些不能被实视图集合回答的查询的中间结果。

动态视图选择方法

- 动态方法可以看成是查询负载驱动的视图选择方法,它会对查询负载进行语法分析,从而枚举相关的候选视图
- 典型算法
  - 基于块文件的方法
  - **DynaMat**
  - 基于谓词的方法
  - 基于缓存预测的方法
  - 基于查询分类的方法

混合视图选择方法

- 静态选择方法和动态选择方法的有效结合，既能充分利用静态选择方法改善查询响应时间的作用，又能发挥动态方法自动视图调整的功能。

混合视图选择方法

- 主要思想
  - 把视图集合划分为动态视图集合和静态视图集合。
  - 从静态视图集合中选出的实视图可以保留多个视图维护窗口。
  - 从动态视图集合中选出的实视图可以即时生成或被替换。
  - 静态视图集合中的视图主要用来更好地维护动态视图集合中的视图，并回答一些粒度较细的查询。

实体化视图的维护

- 在数据装载和刷新时，应该对实体化视图进行有效的更新。实体化视图的更新主要有两种策略：

1) 重计算策略：更新数据时对相关的视图重新计算。这种方法的效率太低，开销太大，在实际中并不常用。

2) 增量维护策略：利用自维护等技术，捕获发生的变化，利用这些变化更新相应的视图。

捕获数据变化的方法

- 四种方法
  - 时标方法：通过原始数据上的时间标签进行识别
  - DELTA文件：记录自上次维护之后的所有数据变化
  - 前后映象文件：每次维护时给数据库设立一个快照
  - 日志文件：利用数据库日志文件捕获数据的变化

利用实体化视图进行查询处理

- 典型算法
  - 基于one-level规则的算法
  - Bucket算法
  - Inverse-Rules算法
  - MiniCon算法
  - CoreCover算法

数据仓库与OLAP的关键技术

- 多维数据模型
- CUBE计算技术
- 实体化视图技术
- 精简数据方体技术
- 索引技术

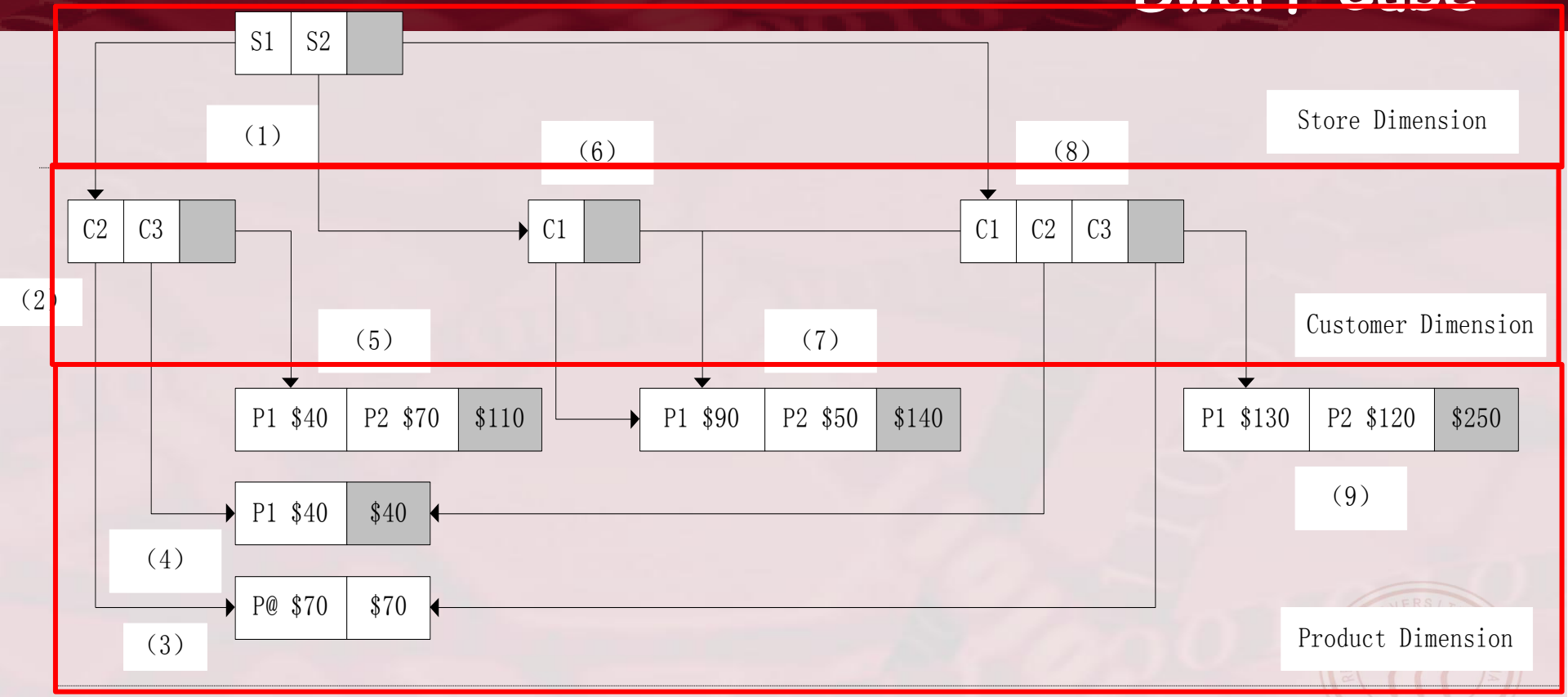
精简数据方体技术的特点

- 完全实体化，查询时不需要进一步的聚集计算。有别于只实体化一部分视图来减少CBUE体积的方法。
- 通过去除CUBE的内在冗余来缩减CUBE体积。
- 回答查询时不需要解压缩，且可以精确地回答查询。有别于通过数据压缩来减少CBUE体积的方法。
- 适用于所有的OLAP应用。有别于那些压缩后的数据只能用来回答某种类型的特殊查询的方法。

精简数据方体的方法

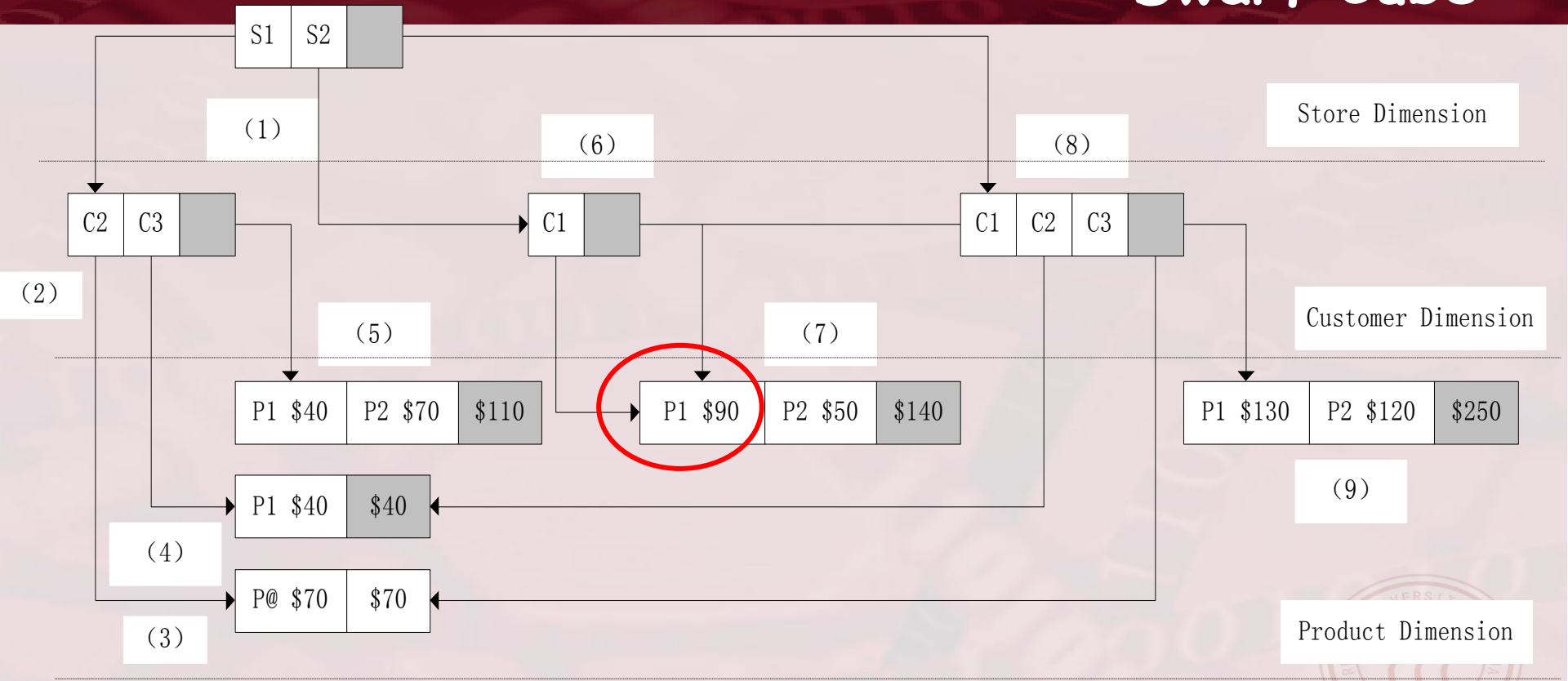
- Dwarf Cube
- Condensed Cube
- Quotient Cube
- QC-Tree

Dwarf Cube



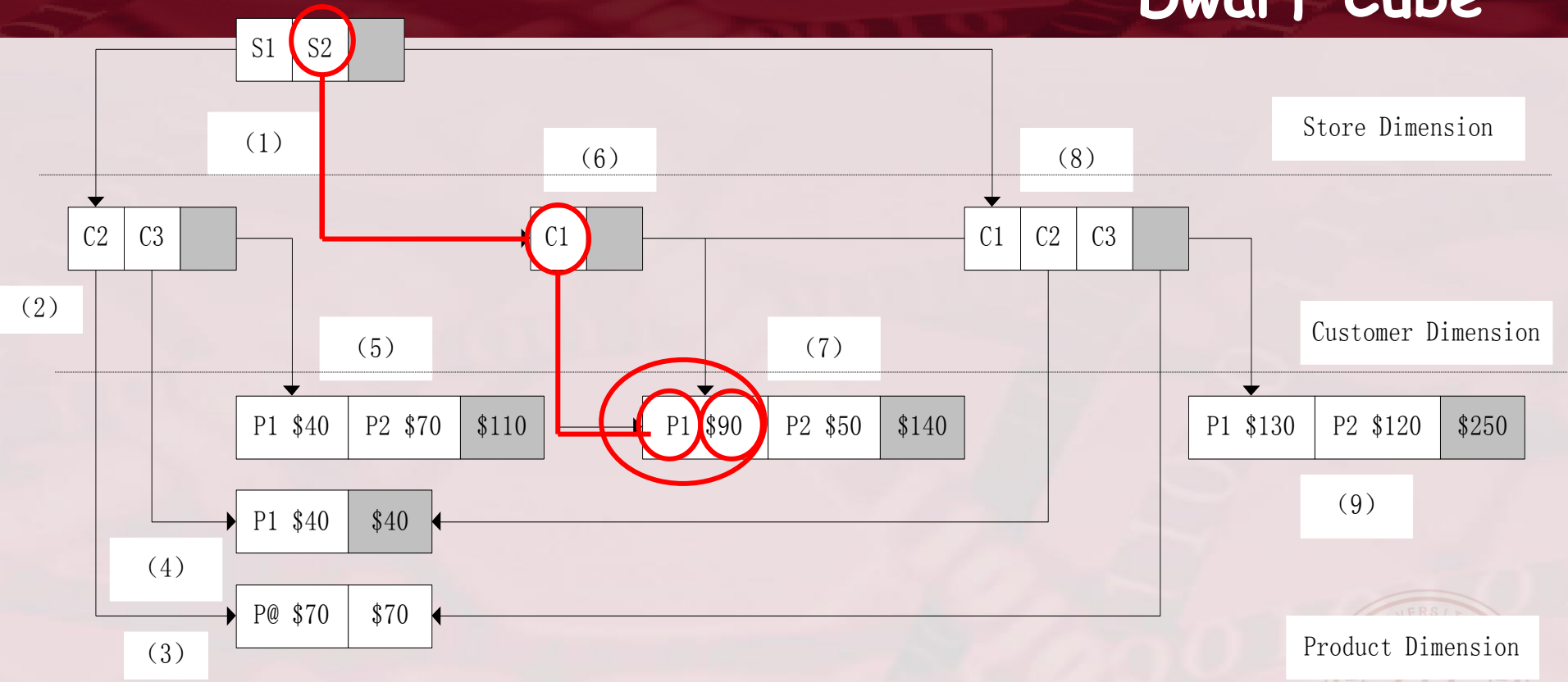
一个Dwarf Cube的例子

Dwarf Cube



一个Dwarf Cube的例子

Dwarf Cube



一个Dwarf Cube的例子

TID	Cuboid	A	B	C	M
1	ALL	*	*	*	380
2	ABC	0	1	1	50
3	A	0	*	*	50
4	AB	0	1	*	50
5	AC	0	*	1	50
6	ABC	1	1	1	100
7	A	1	*	*	100
8	AB	1	1	*	100
9	AC	1	*	1	100
10	ABC	2	3	1	80
11	A	2	*	*	80
12	AB	2	3	*	80
13	AC	2	*	1	80
14	B	*	3	*	80
15	BC	*	3	1	80
16	ABC	4	5	1	70
17	A	4	*	*	70
18	AB	4	5	*	70
19	AC	4	*	1	70
20	ABC	6	5	2	80
21	A	6	*	*	80
22	AB	6	5	*	80
23	AC	6	*	2	80
24	C	*	*	2	80
25	BC	*	5	2	80
26	B	*	1	*	150
27	B	*	5	*	150
28	C	*	*	1	280
29	BC	*	1	1	150
30	BC	*	5	1	70

(a)

TID	A	B	C	M	SDSET
1	0	1	1	50	{{A},{AB},{AC},{ABC}}
2	1	1	1	100	{{A},{AB},{AC},{ABC}}
3	2	3	1	80	{{A},{AB},{AC},{ABC}}
4	4	5	1	70	{{A},{AB},{AC},{ABC}}
5	6	5	2	80	{{A},{AB},{AC},{ABC}}
6	*	3	*	80	{}
7	*	3	1	80	{}
8	*	*	2	80	{}
9	*	5	2	80	{}
10	*	1	*	150	{}
11	*	5	*	150	{}
12	*	*	1	280	{}
13	*	1	1	150	{}
14	*	5	1	70	{}
15	*	*	*	380	{}

(b)

TID	A	B	C	M	SDSET
1	0	1	1	50	{{A},{AB},{AC},{ABC}}
2	1	1	1	100	{{A},{AB},{AC},{ABC}}
3	2	3	1	80	{{A},{B},{AB},{AC},{BC},{ABC}}
4	4	5	1	70	{{A},{AB},{AC},{BC},{ABC}}
5	6	5	2	80	{{A},{C},{AB},{AC},{BC},{ABC}}
6	*	1	*	150	{}
7	*	5	*	150	{}
8	*	*	1	280	{}
9	*	1	1	150	{}
10	*	*	*	380	{}

(c)

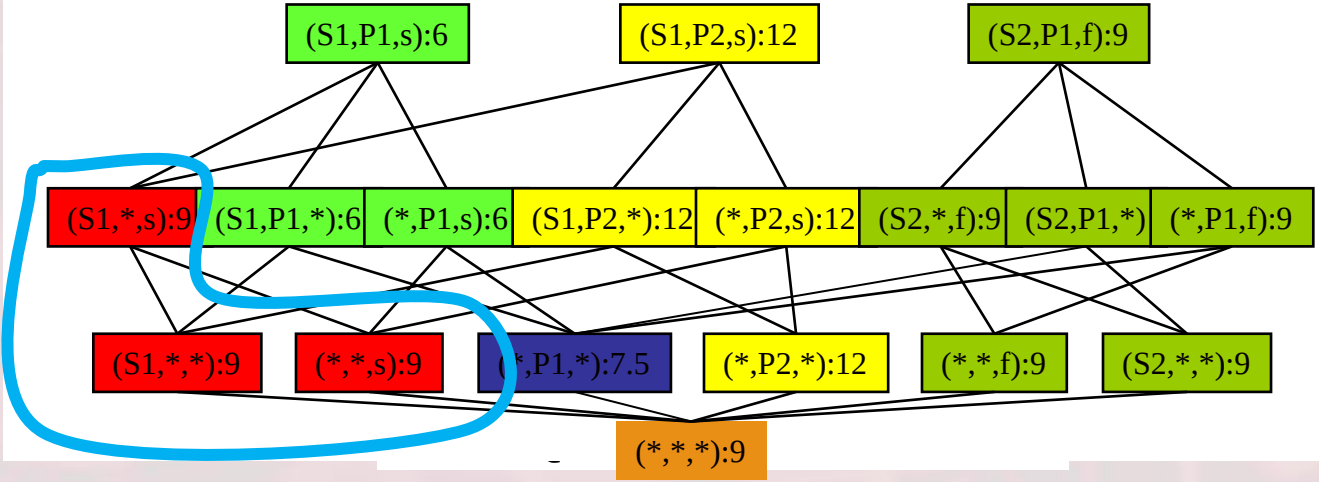
(a)完整的数据方体

(b) Condensed数据方体

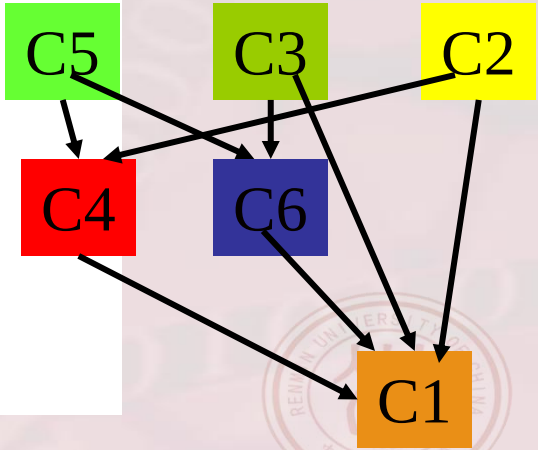
(c) 最小Condensed数据方体

Store	Product	Season	Sale
S1	P1	s	6
S1	P2	s	12
S2	P1	f	9

(a) 基本表 sales

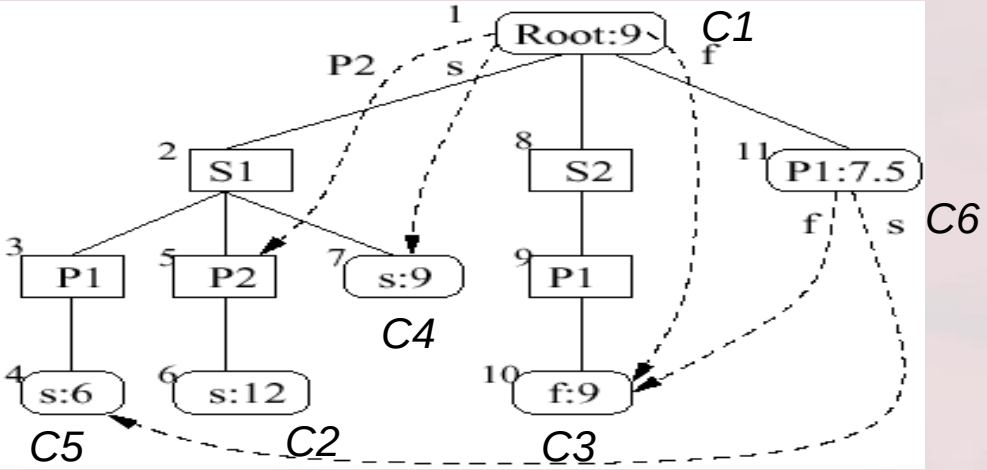


(b) 划分后的等价类



(c) quotient cube格结构

Class	Upper Bound
C_1	$(*, *, *)$
C_2	$(S1, P2, s)$
C_3	$(S2, P1, f)$
C_4	$(S1, *, s)$
C_5	$(S1, P1, s)$
C_6	$(*, P1, *)$



等价类及与其对应的QC-Tree

数据仓库与OLAP的关键技术

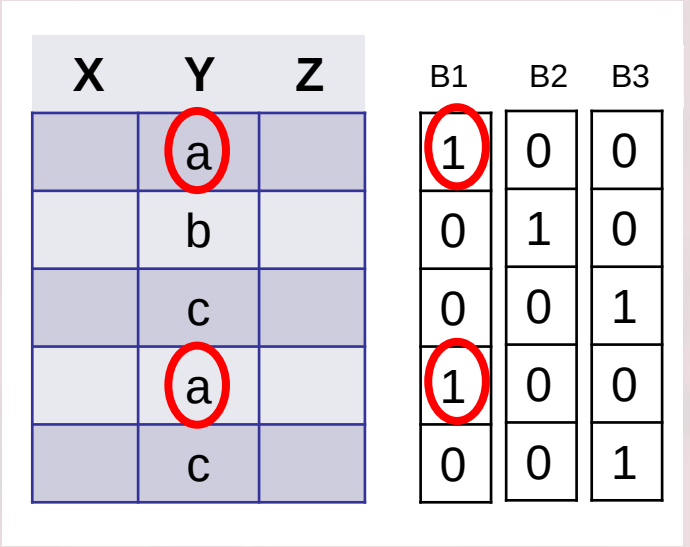
- 多维数据模型
- CUBE计算技术
- 实体化视图技术
- 精简数据方体技术
- 索引技术

索引技术

- 标准BitMap索引
- Encoded Bitmap索引
- Bitmap Join索引
- Bit-Sliced索引

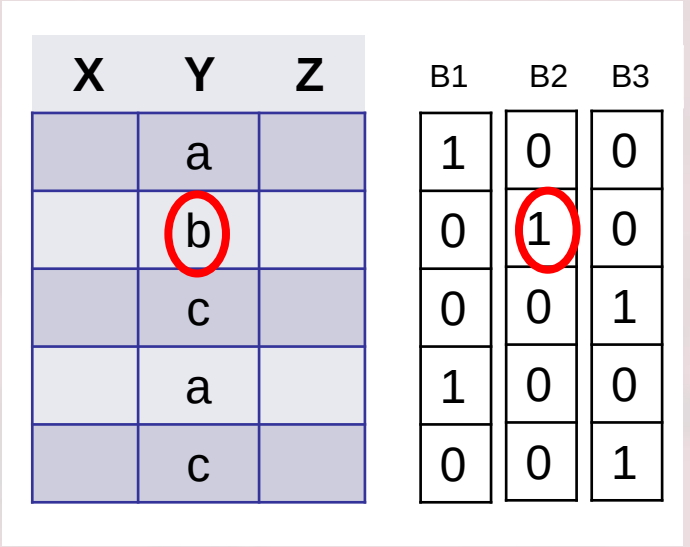
标准bitmap索引

- 基本思想是对每一个列值，给它创建一个相应的由0和1组成的序列



标准bitmap索引

- 基本思想是对每一个列值，给它创建一个相应的由0和1组成的序列

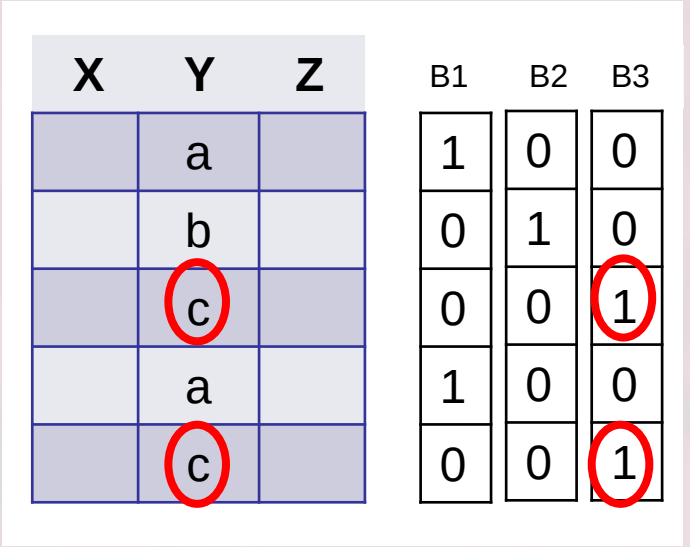


X	Y	Z
	a	
	b	
	c	
	a	
	c	

B1	B2	B3
1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

标准bitmap索引

- 基本思想是对每一个列值，给它创建一个相应的由0和1组成的序列



X	Y	Z
	a	
	b	
	c	
	a	
	c	

B1	B2	B3
1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

标准bitmap索引

- 基本思想是对每一个列值，给它创建一个相应的由0和1组成的序列

- bitmap索引的特点
  - 索引的大小与列的不同值的个数成正比，适用于不同值的个数较少的属性列
  - 可以使用有效的位操作来快速回答查询

X	Y	Z
	a	
	b	
	c	
	a	
	c	

B1	B2	B3
1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

Encoded Bitmap索引

- **Encoded bitmap索引采用编码技术来大大减小bitmap索引位向量的个数。**
- **如果某个属性列的基数为K，标准bitmap索引所需要的位向量的个数为k个，而Encode bitmap索引所需要的位向量的个数为$\lceil \log_2 K \rceil$个。**
- **Encoded bitmap索引需要一个编码映射表。**

X	Y	Z
	a	
	b	
	c	
	a	
	c	

B1	B2
0	0
0	1
1	0
0	0
1	0

编码映射表

a	00
b	01
c	11

Bitmap Join索引

- **Bitmap Join索引是标准Bitmap索引的一个简单变种。**
- **它通过主外码之间的关系，将星型模型中事实表的属性值与维表中匹配的行关联起来，实质上就是预计算了一个二元连接。**
- **多值的B**
  Bitmap
  的数据量

Customer

Customer_id	Gender	Country
1	F	Mexico
2	M	Canada
3	F	USA
4	M	Canada
5	F	USA

Bitmap Join index

	Gender		Country		
sales_id	M	F	Mexico	Canada	USA
1	1	0	0	1	0
2	0	1	0	0	1
3	0	1	1	0	0
4	0	1	0	0	1
5	1	0	0	1	0
6	1	0	0	1	0
7	0	1	0	0	1
8	1	0	0	1	0
9	0	1	0	0	1
10	0	1	1	0	0
11	0	1	0	0	1

Sales

time_id	customer_id	Revenue
20120301	4	3452
20120301	3	4432
20120302	1	5356
20120303	5	2352
20120303	2	5536
20120304	4	6737
20120305	3	5648
20120306	2	9345
20120306	5	5547
20120307	1	7578
20120308	3	5533

Bitmap Join索引

- **Bitmap Join索引是标准Bitmap索引的一个简单变种。**
- **它通过主外码之间的关系，将星型模型中事实表的属性值与维表中匹配的行关联起来，实质上就是预计算了一个二元连接。**
- **多值的B**
  Bitmap J
  的数据量

Customer

Customer_id	Gender	Country
1	F	Mexico
2	M	Canada
3	F	USA
4	M	Canada
5	F	USA

Bitmap Join index

	Gender		Country		
sales_id	M	F	Mexico	Canada	USA
1	1	0	0	1	0
2	0	1	0	0	1
3	0	1	1	0	0
4	0	1	0	0	1
5	1	0	0	1	0
6	1	0	0	1	0
7	0	1	0	0	1
8	1	0	0	1	0
9	0	1	0	0	1
10	0	1	1	0	0
11	0	1	0	0	1

Sales

time_id	customer_id	Revenue
20120301	4	3452
20120301	3	4432
20120302	1	5356
20120303	5	2352
20120303	2	5536
20120304	4	6737
20120305	3	5648
20120306	2	9345
20120306	5	5547
20120307	1	7578
20120308	3	5533

Bitmap Join索引

- **Bitmap Join索引是标准Bitmap索引的一个简单变种。**
- **它通过主外码之间的关系，将星型模型中事实表的属性值与维表中匹配的行关联起来，实质上就是预计算了一个二元连接。**
- **多值的B**
  Bitmap J
  的数据量

Customer

Customer_id	Gender	Country
1	F	Mexico
2	M	Canada
3	F	USA
4	M	Canada
5	F	USA

Bitmap Join index

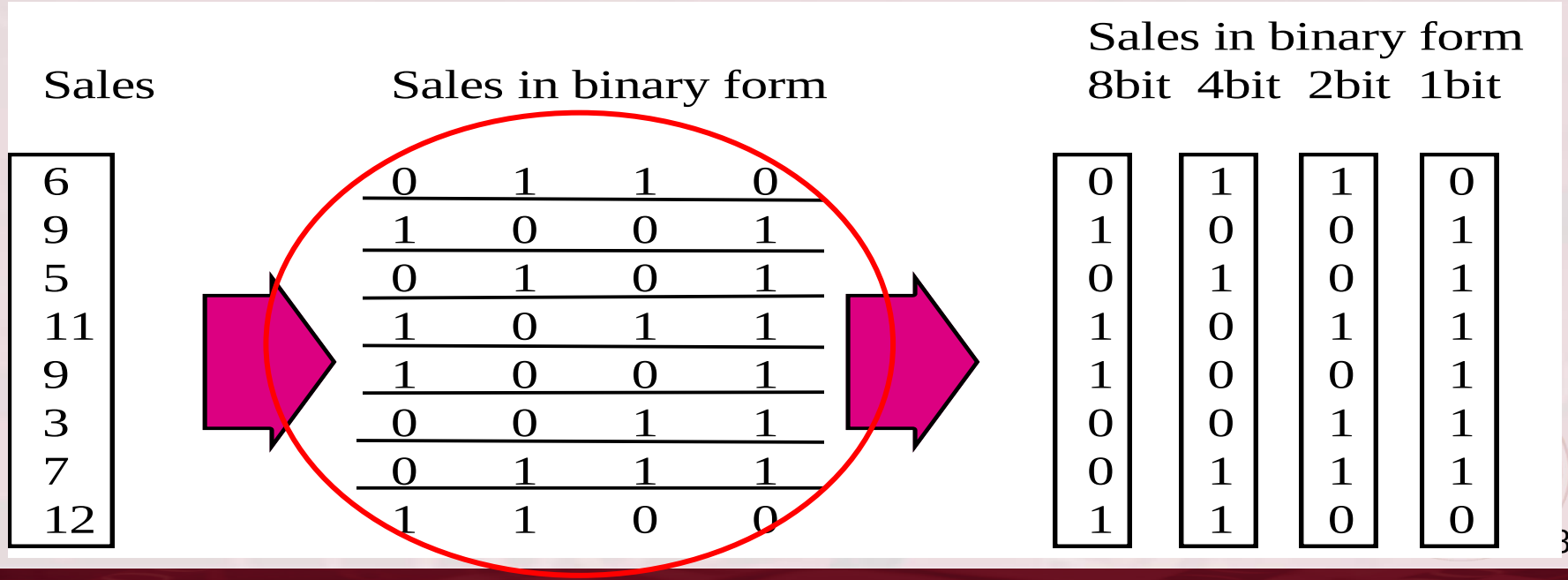
	Gender		Country		
sales_id	M	F	Mexico	Canada	USA
1	1	0	0	1	0
2	0	1	0	0	1
3	0	1	1	0	0
4	0	1	0	0	1
5	1	0	0	1	0
6	1	0	0	1	0
7	0	1	0	0	1
8	1	0	0	1	0
9	0	1	0	0	1
10	0	1	1	0	0
11	0	1	0	0	1

Sales

time_id	customer_id	Revenue
20120301	4	3452
20120301	3	4432
20120302	1	5356
20120303	5	2352
20120303	2	5536
20120304	4	6737
20120305	3	5648
20120306	2	9345
20120306	5	5547
20120307	1	7578
20120308	3	5533

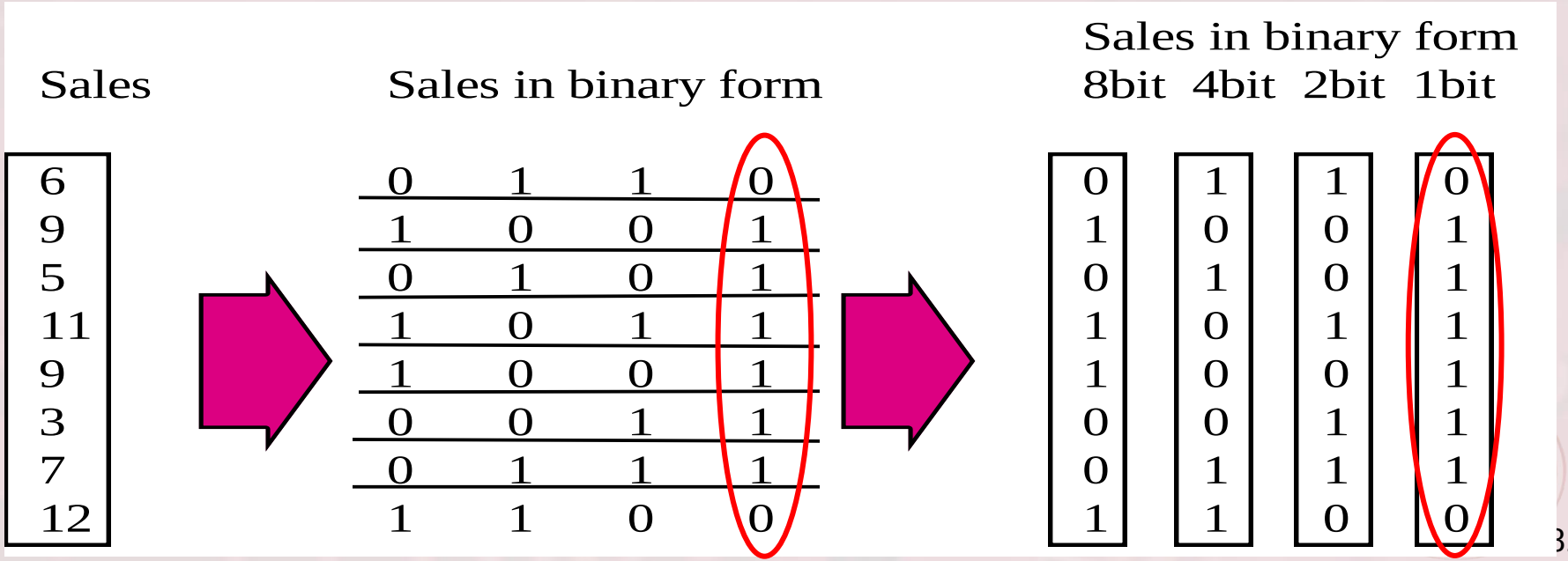
Bit-Sliced索引

- **Bit-Sliced索引是将属性列的域值按照某种方式进行垂直分割，然后以二进制位图的形式存储。**



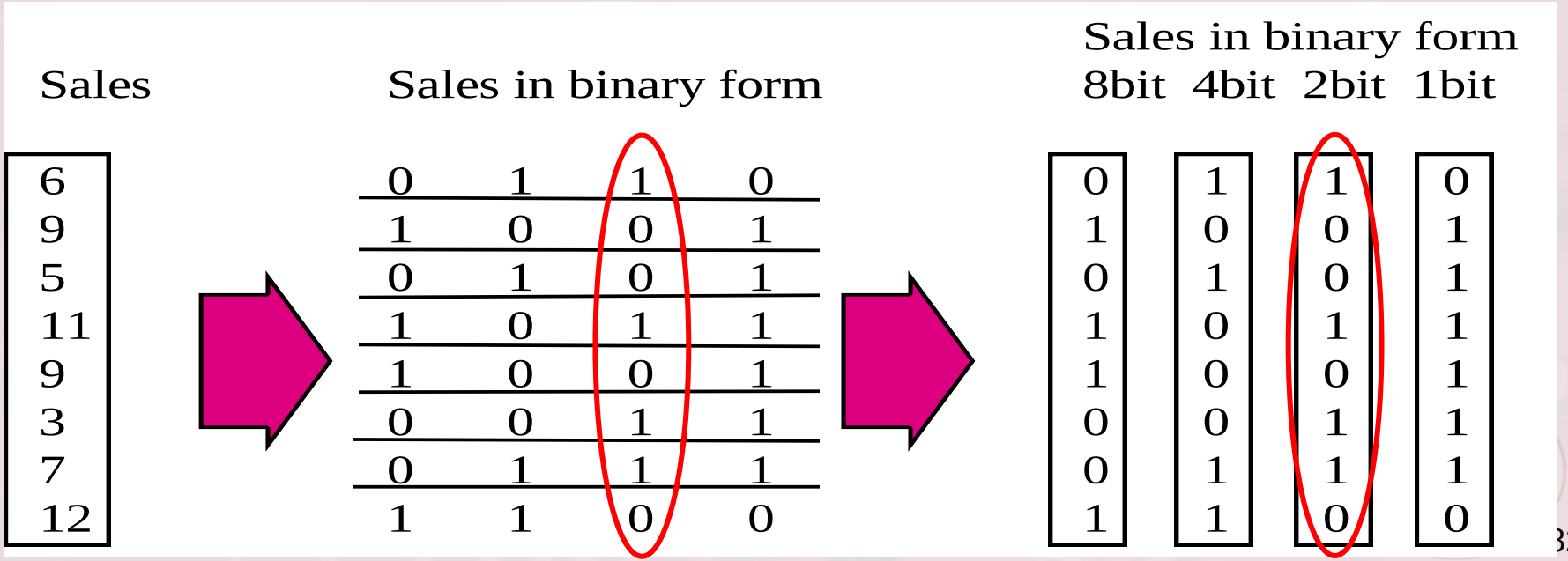
Bit-Sliced索引

- **Bit-Sliced索引是将属性列的域值按照某种方式进行垂直分割，然后以二进制位图的形式存储。**



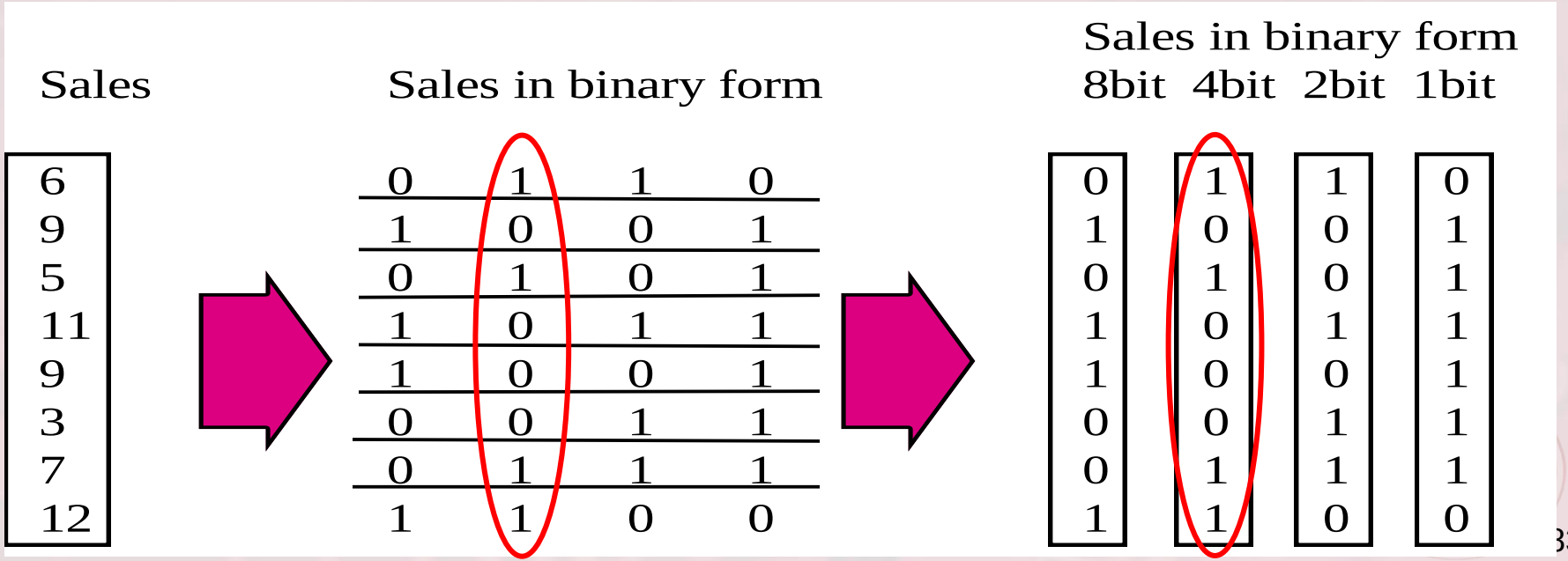
Bit-Sliced索引

- **Bit-Sliced索引是将属性列的域值按照某种方式进行垂直分割，然后以二进制位图的形式存储。**



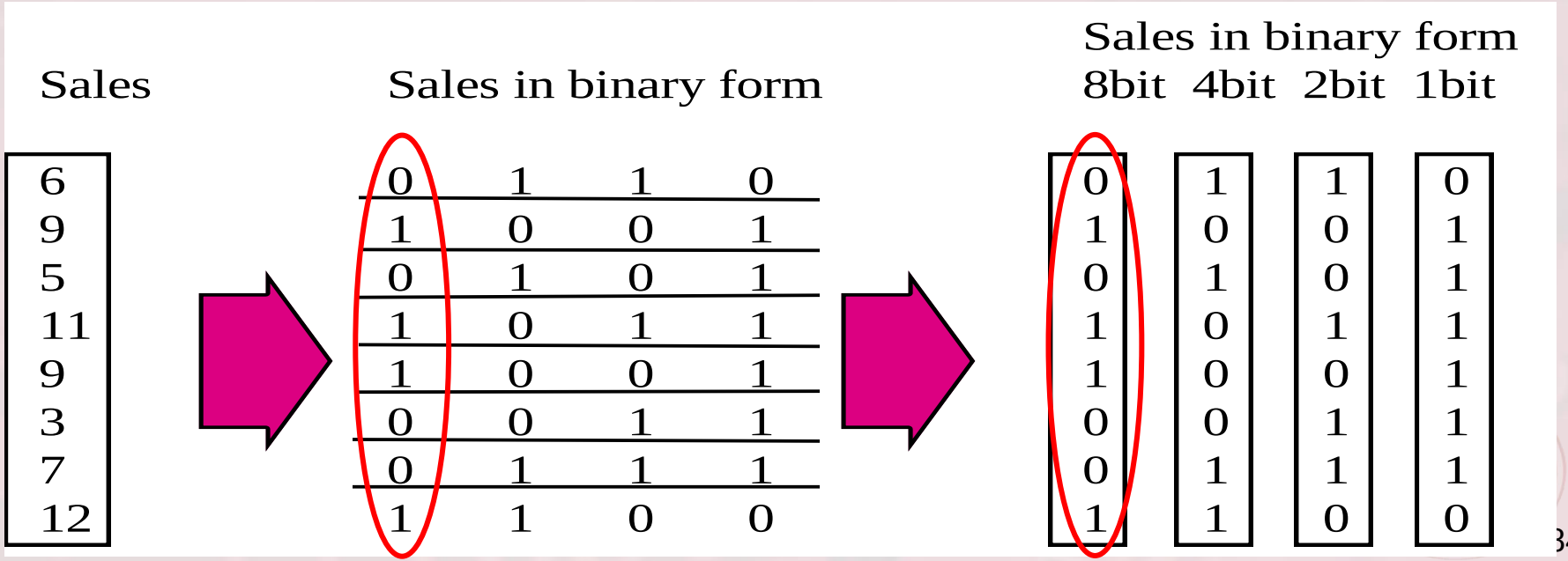
Bit-Sliced索引

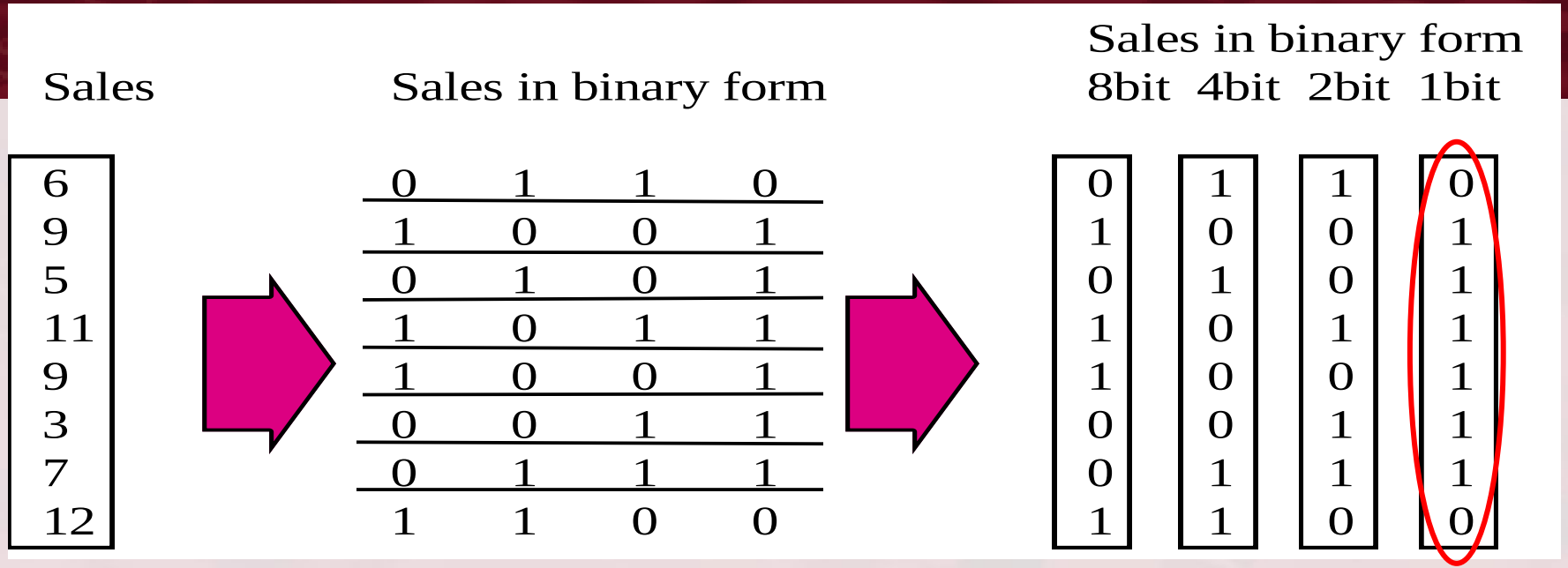
- **Bit-Sliced索引是将属性列的域值按照某种方式进行垂直分割，然后以二进制位图的形式存储。**



Bit-Sliced索引

- **Bit-Sliced索引是将属性列的域值按照某种方式进行垂直分割，然后以二进制位图的形式存储。**



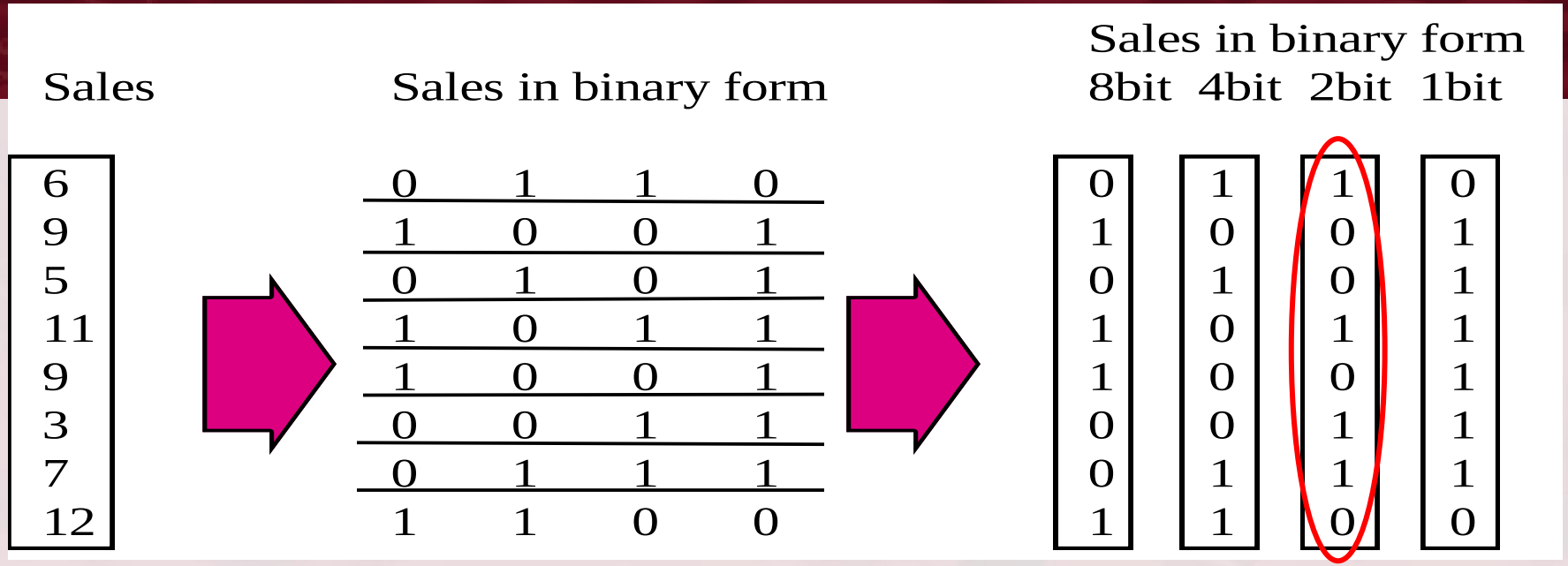


select sum(sales) from customers

计算方法：

#1bits on * 1 + #2bits on *2 +#4bits on *4 +#8bits on * 8 ....

6*1 + 4*2 +4*4+4*8 = 62

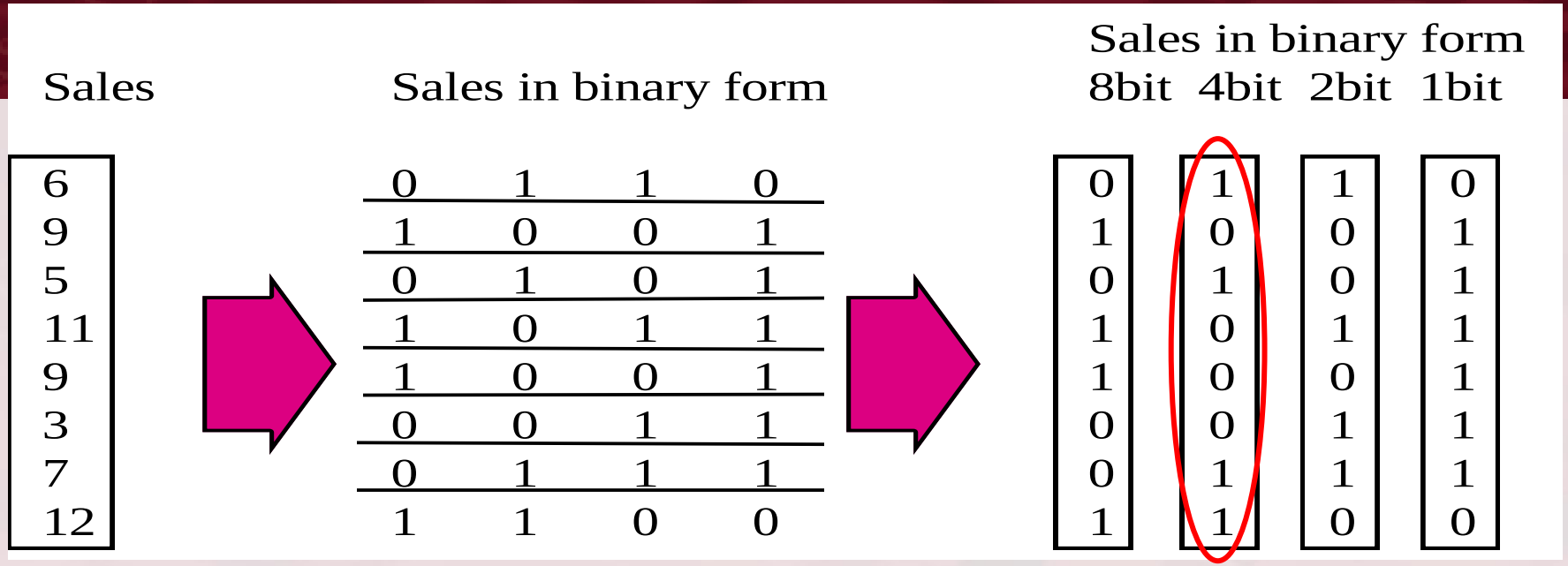


select sum(sales) from customers

计算方法：

#1bits on * 1 + #2bits on *2 +#4bits on *4 +#8bits on * 8 ....

6*1 + 4*2 +4*4+4*8 = 62

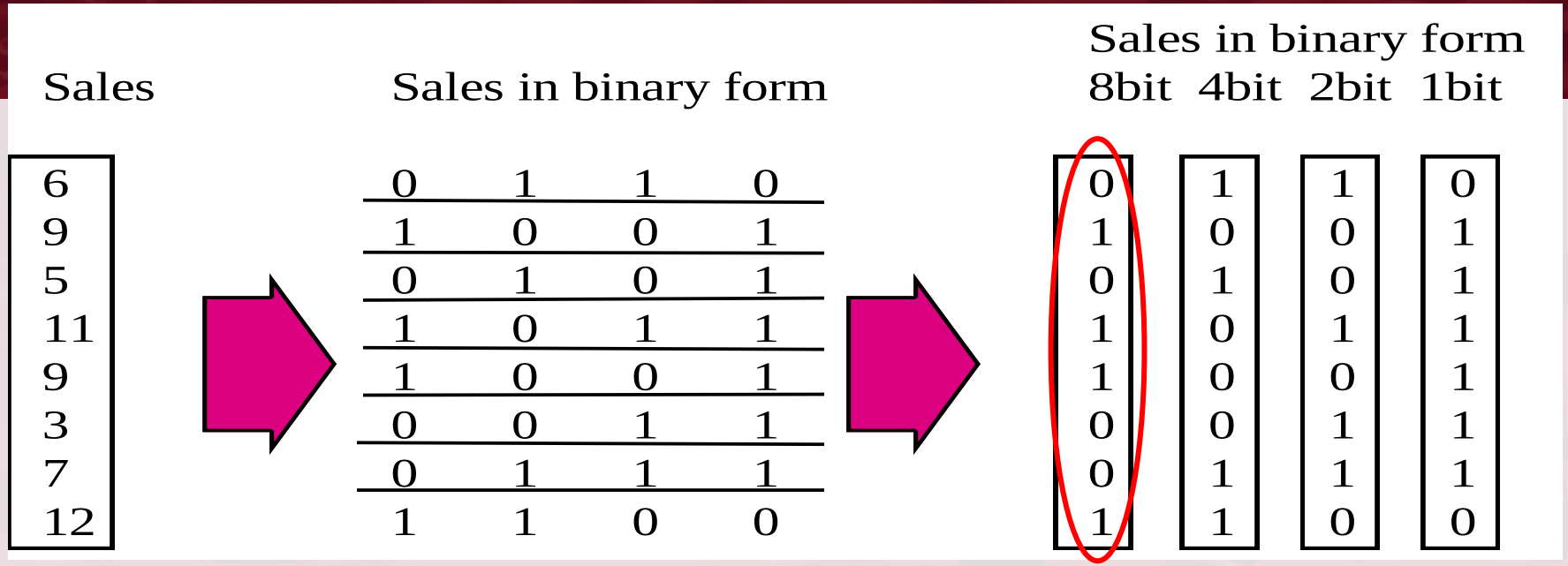


select sum(sales) from customers

计算方法：

#1bits on * 1 + #2bits on *2 +#4bits on *4 +#8bits on * 8 ....

6*1 + 4*2 +4*4+4*8 = 62



select sum(sales) from customers

计算方法：

#1bits on * 1 + #2bits on *2 +#4bits on *4 +#8bits on * 8 ....

6*1 + 4*2 +4*4+4*8 = 62

小结

- CUBE计算技术
- 实体化视图技术
  - 实体化视图选择
  - 实体化视图维护
  - 利用实体化视图回答查询
- 精简数据方体技术
  - Dwarf Cube
  - Condensed Cube
  - Quotient Cube
  - QC-Tree
- 索引技术
  - 标准BitMap索引
  - Encoded Bitmap索引
  - Bitmap Join索引
  - Bit-Sliced索引