第二章 联机分析处理基本概念

- 数据分析模型
- **OLAP的提出**
- 多维数据结构
- 多维数据操作
- 多维数据模型的实现

数据分析模型概述

- 以前的数据分析主要是静态的
  - 不能与数据库中的数据建立动态联系
- 实际需求：更需要复杂、动态的分析
  - 综合多个数据源
  - 从不同角度观察数据
  - 多变的主题与多维数据访问

数据分析模型概述（续）

- **E-R模型：基于数据库的传统数据分析，不适合动态分析**
  - **不支持域（Domain）概念**
  - **只支持主外码表示的联系**
  - **不能表示如：“产品类型”和“客户爱好”通过“销售额” 发生的联系**

Codd提出四种数据分析模型

- 划分依据：
  - 处理数据的范围
  - 用户-分析人员的交互需要
  - 多维分析需求
  - 现有工具的支持
- 四种数据分析模型：
  - 绝对模型
  - 解释模型
  - 思考模型
  - 公式化模型

数据分析模型：绝对模型

- 绝对模型（Categorical Model）
  - 支持工具广泛
  - 静态数据分析
  - 比较历史数据值
  - 综合路径是数据库设计时定义的

数据分析模型：解释模型

- 解释模型（Exegetical Model）
  - 支持工具较多
  - 静态数据分析
  - 利用已有的多层次路径层层细化，找出事实发生的原因

数据分析模型：思考模型

- 思考模型（Contemplative Model）
  - 支持工具较少
  - 动态数据分析（动态性较低）
  - 在一维或多维上引入变量或参数，分析引入后会发生什么
  - 引入变量时，须创建大量综合数据

数据分析模型：公式模型

- 公式模型（Formulatic Model）
  - 至今没有支持工具
  - 动态数据分析（动态性很高）
  - 分析在多维上需引入哪些变量或参数，并分析引入后所产生的结果

数据分析举例

- 目 标：为了扩大商品销售量、分析与销售量相关因素
- 分析模型：
  - 绝对模型：历史数据比较，利用回归分析
    - “某种商品今年的销售情况与以往相比，有何变化？今后趋势？”
  - 解释模型：进一步找出原因
    - “销售量下降与时间、地区、商品、销售渠道中何种因素有关？”
  - 思考模型：
    - 引入年龄（变量），分析销售量与顾客年龄是否有关系？
  - 公式模型：
    - 自动引入各种变量，最终给出与销售量有关的全部因素。

四种分析模型比较

数据分析模型	绝对模型 → 解释模型 → 思考模型 → 公式模型
处理数据范围	历史数据和当前数据 ——→ 预测数据、行为
用户－分析人员交互	少 ——→ 多
多维分析	少 ——→ 多
现有支持工具	多 ——→ 少

第二章 联机分析处理基本概念

- 数据分析模型
- **OLAP的提出**
- 多维数据结构
- 多维数据操作
- 多维数据模型的实现

联机分析处理（OLAP）的提出

- 关系数据库满足了联机事务处理（OLTP）的要求
- 存在着大量的分析型应用 —— RDB无法适应
  - 应用角度：要求对大量的数据从各个角度进行综合分析（多维分析）
  - 技术角度：SQL已经不能很好的适应分析应用需求
    - 查询效率（响应时间）
    - SQL本身的限制，尤其对时间的处理能力
  - 典型分析应用：对一些统计指标 （销售金额）
    从不同角度（维）（时间、地区、商品类型）
    从不同级别（层次）（地区：县、地市、省、大区）
- 在RDBMS上开发前端产品，支持上述应用逻辑

E.F.Codd把这类技术称为“ OLAP ”（1993年）

OLAP应用举例

- 不同时间段的比较（同期比）
  - 各种商品本周（本月、本年）的销售情况与以往相比，有何变化？今后趋势？
- 排序和统计分类 (top N/bottom N)
  - 统计每天销售量、销售额和利润最高的10个商场？
- 客户特定的即席分析 (市场分割、即席分组的情况)
  - 按照季度统计一下东北地区前四个季度的收入情况？

第二章 联机分析处理基本概念

- 数据分析模型
- **OLAP的提出**
- 多维数据结构
- 多维数据操作
- 多维数据模型的实现

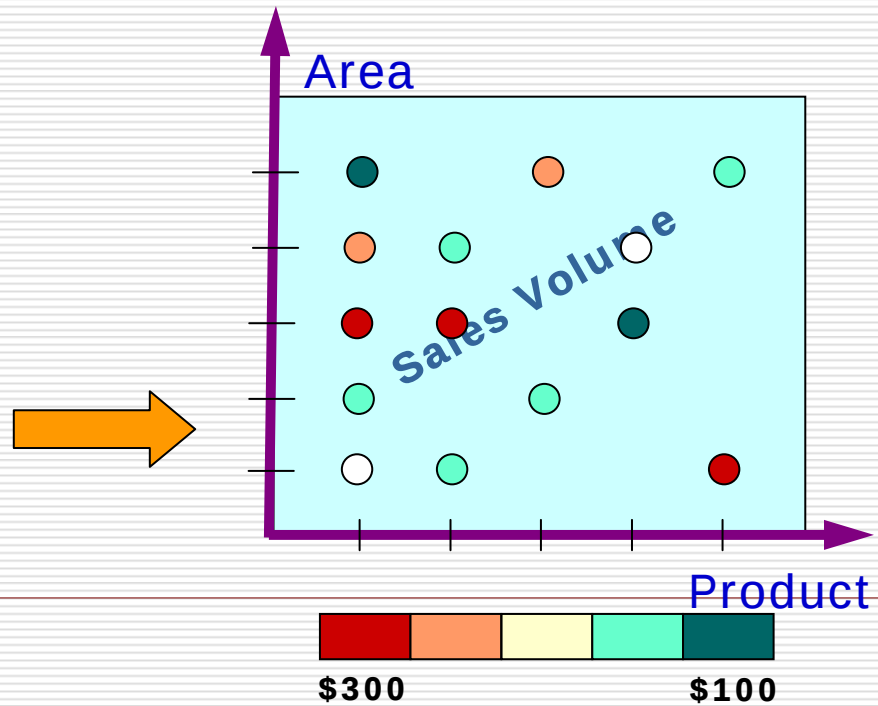
多维数据模型

- 多维数据模型又称多维概念视图，通常用Cube来表示。
- 多维数据模型可以更加直观的表示现实中的复杂关系
- 多维数据模型的基本组成：维、度量（变量、指标）

举例：计算每一个商场、每个产品的销售额

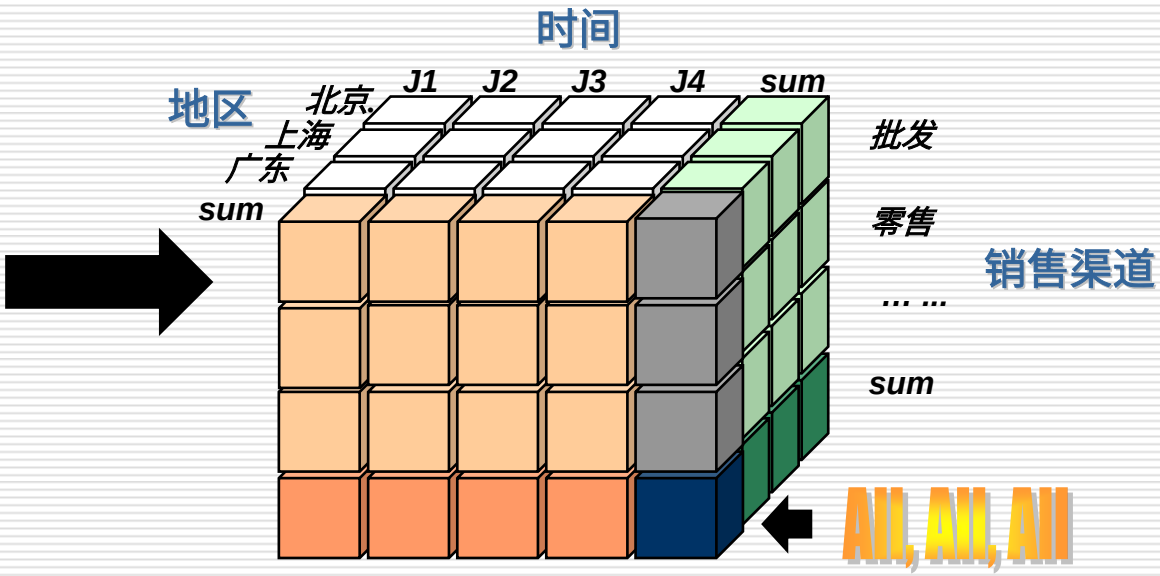
Cross-Tabulation (products/Store)

Sales Volume		Product				
		A	B	C	D	E
Store	I	$200	$150	-	-	$300
	II	$150	-	$150	-	-
	III	$300	$300	-	$100	-
	IV	$250	$150	-	$200	-
	V	$100	-	$250	-	$100



多维数据模型举例

时间	地区	销售渠道	销售额
J1	北京	批发	1200
J1	北京	零售	2300
J1	上海	批发	1233
J1	上海	零售	2122
…	…	…	
J2	北京	批发	3312
J2	上海	批发	3423
…	…	…	



关系表与多维Cube

多维数据模型的组成

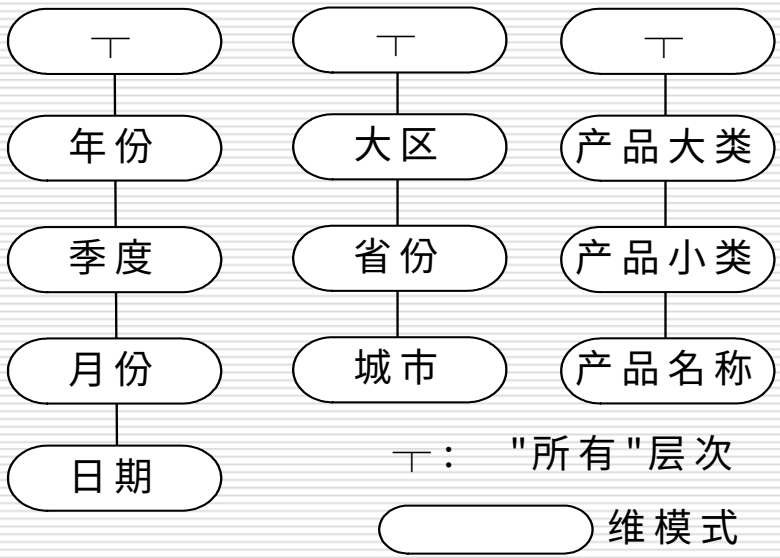
- 维（Dimension）
  - 维层次路径、维层次、维成员 （维实例） 、维层次属性
- 事实（Fact）
  - 度量（Measure）
- 数据立方体（Cube）

维

- 维：对数据进行分类的一种结构，以用于从特定的角度观察数据。（例如：时间、地区、产品）
- 维的两个用途
  - 选择针对期望详细程度的层次的数据
  - 分组对细节数据综合（聚集）到相应的详细程度的数据层次

维

- 维的组织方式：维层次路径（HIERARCHY）
- 维层次路径由代表不同详细程度的维层次（Level）组成。
- 维的层次：特定角度的不同细节程度



维

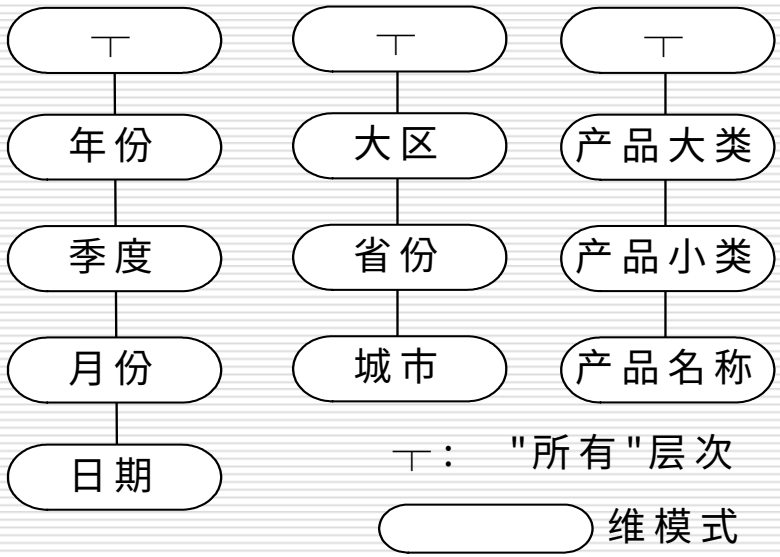
- 维层次中包含
  - 维成员（DIMENSION VALUES）,维成员树
    - 维的一个取值（称为该维的一个成员）,每一个维成员属于某一个特定的维层次。
      - 例如：时间维：三个层次，日、月、年， 维成员： 1999年5月20日、1999年5月；1999年
      - 维成员是数据在该维上的位置描述

      例如：1999年5月20日销售额表示销售额数据在时间维上的位置

      （相当于时间轴上的某一点或某一区间）
    - 不同维层次的取值的组合（对多层次情况）， 例如：5月20日
  - 维层次属性（ATTRIBUTES）:维层次上的描述属性，例如产品的“规格”、“颜色”、“销地”、“产地”…

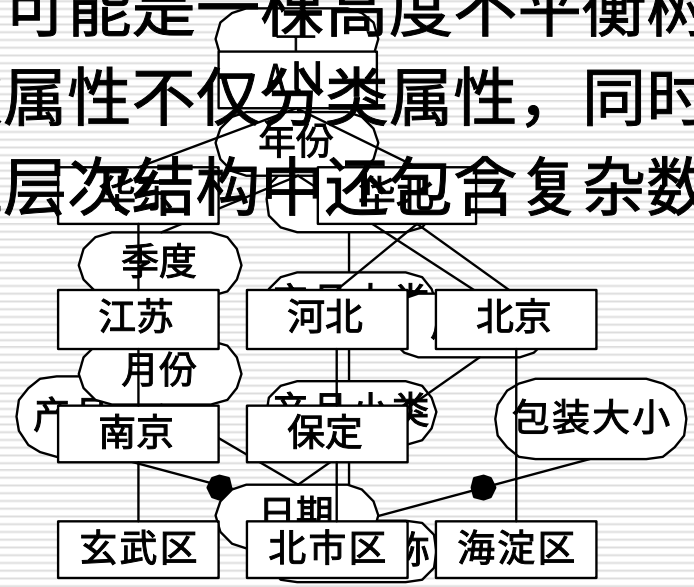
维层次关系

- 定义维层次的聚集和钻取关系
  - 简单维层次关系



复杂的维层次关系

- 较为复杂的维层次关系
  - 一个维包含拥有同一底层数据的多条维层次路径
  - 维成员树可能是一棵高度不平衡树。
  - 在维层次属性不仅分类属性，同时还拥有描述属性
  - 在某些维层次结构中还包含复杂数据类型的维成员



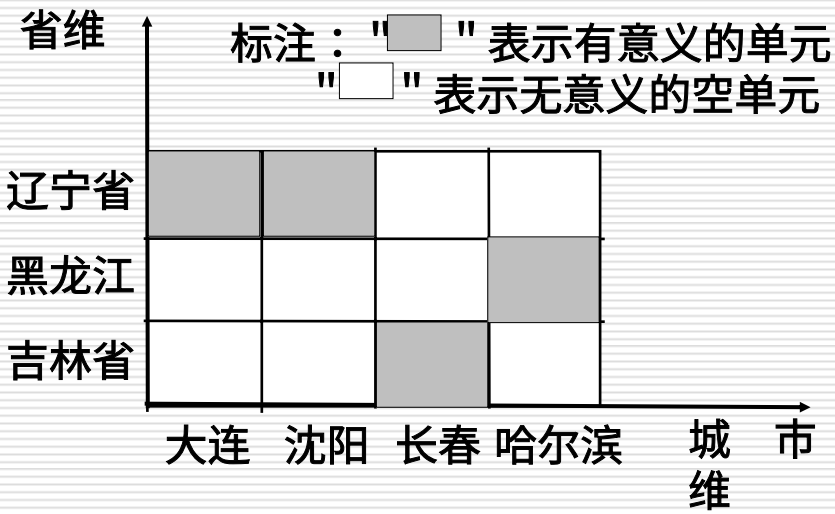
(a)时间维维层次结构
(b)地区维维成员层次结构

(c)产品维维层次结构

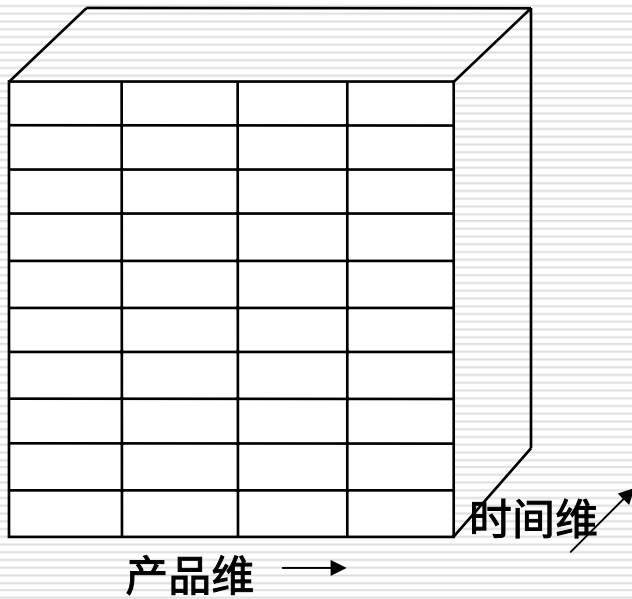
为什么需要维层次关系

- 不支持层次关系带来的问题

  增加维的数目，变成非常“稀疏”的状况



不支持维层次关系



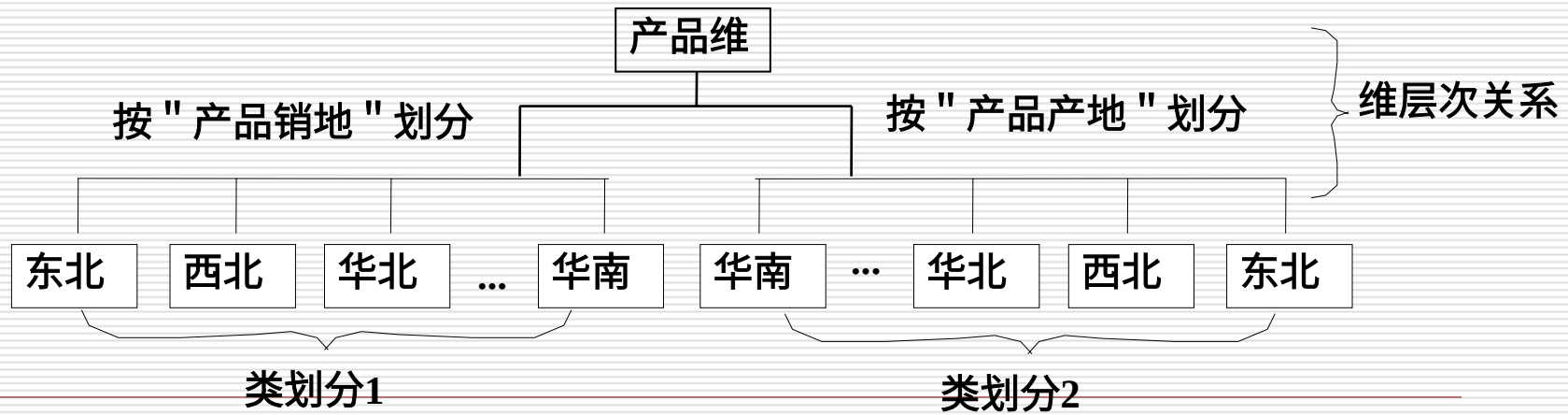
支持维层次关系

维成员属性（维成员的“类”）

- 维成员属性，维成员的描述属性，维成员的“类”
- 按一定的划分标准对维成员全集的一个（分类）划分
  - 划分：即把全集分成了若干子集
  - 各子集的和（并）等于全集
  - 子集间的交为空

维成员属性

- 划分标准一般是实体（维成员）的属性（特征），称为类属性
  - 例如（产品的）“规格”、“颜色”、“销地”、“产地”…
  - 一个类属性，对应一个划分；不同类属性，得到不同类划分



维层次和类的区别

- 表达的含义不同
  - 维层次表达变量在该维的综合的级别

    例：销售额在时间维上按三个级别（日、月、年）进行综合称为三个维层次

    父层次的值由其子层次的值综合得到
  - 维成员的类表达某一子集维成员的共同特征

    即：对应的类属性取相同值

    例如：颜色为红色的产品，不同颜色的产品为不同的类

    同一层次的维成员可划分为类：例如产品大类中的“家电”、“服装”、“文具”等

    不同层次的维成员之间不存在类的关系

维层次和类的区别（续 1）

- 分析动作不同
  - 按维层次进行分析

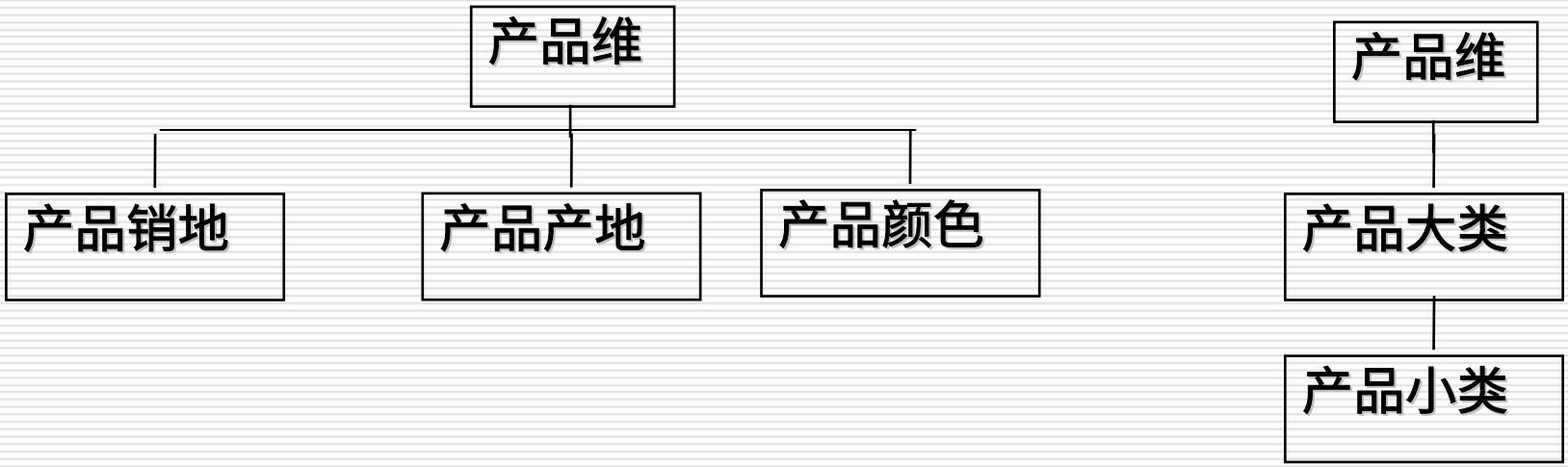
    逐层向上综合数据；逐层向下细化数据；
  - 按维成员的类进行分析

    选择类属性对维成员全集进行分类

    对同类维成员归纳出共同的特性

    按类进行分析不能跨维层次，只在同层次（兄弟结点）进行
  - 将维层次与类交叉组合进行分析（见下图）

维层次和类的区别（续 2）



事实（度量）

- 度量（指标）：数据的实际意义，一般是一个数值度量指标

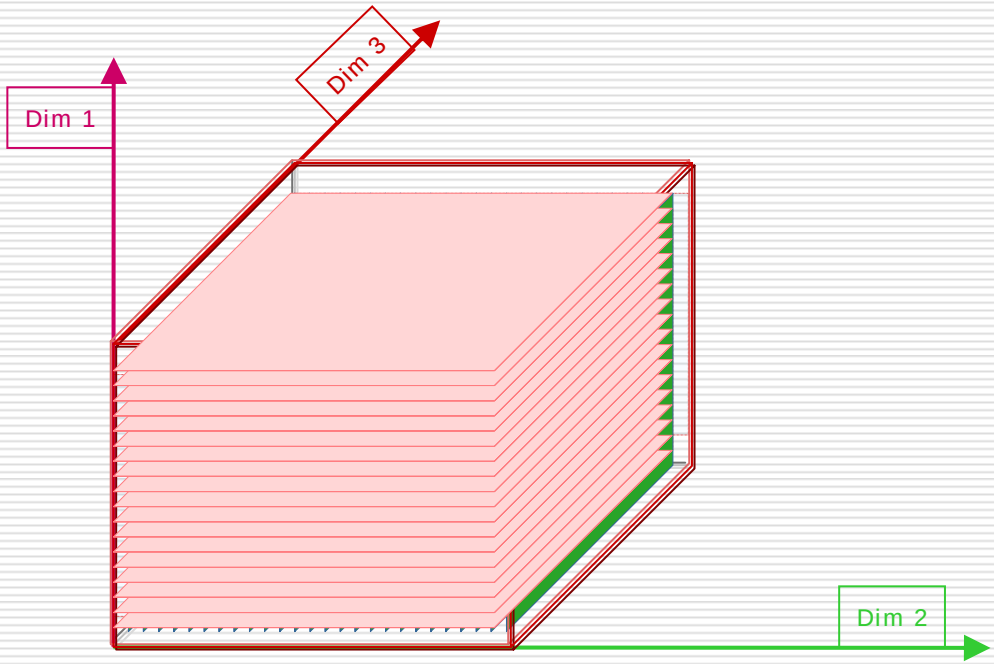
  例如：销售量、销售额，……

  而具体数据（如“10000”）是变量的一个值

- 事实: 存储一个多维数据
  - 表达期望分析的主题（目的、感兴趣的事情、事件或者指标等）
  - 具有一定的粒度，粒度的大小与维层次相关
  - 一个事实中通常包含一个或者多个度量
- 一个度量的两个组件
  - 数字型指标
  - 聚集函数

Cubes

- 按照一定维层次结构和度量（事实）的逻辑上的组织
- 其逻辑上相当于一个多维数组



多维数组

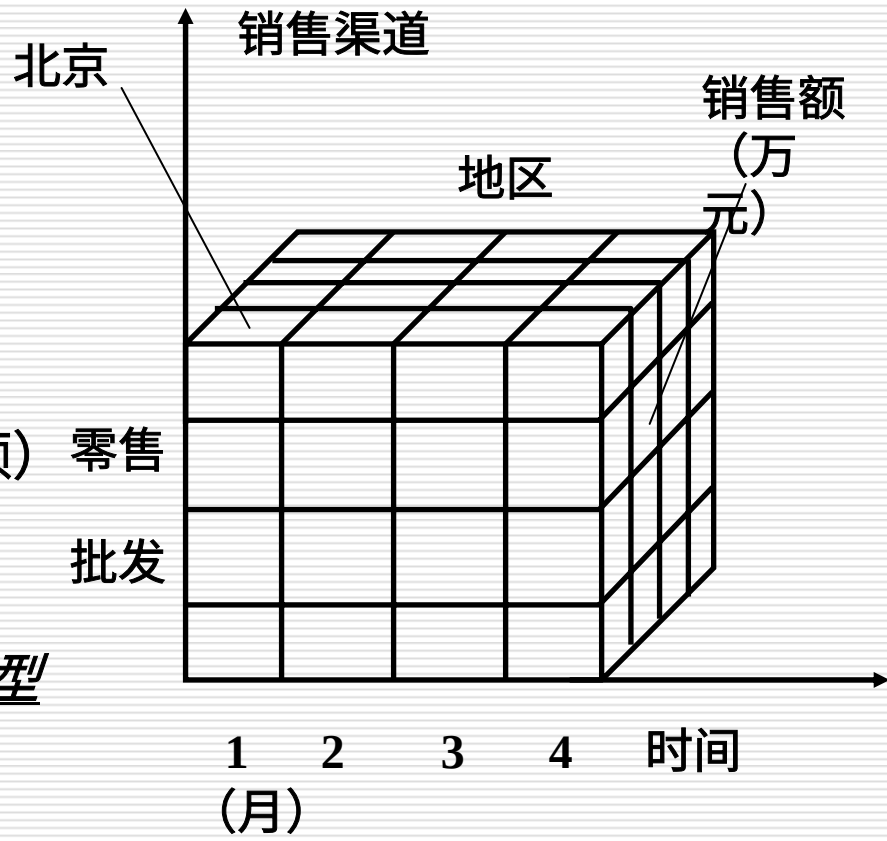
- 多维数组：
  - 一个多维数组表示为：

    （维1，维2，...，维n，变量）
  - 例如：

    （时间、地区、销售渠道、销售额）
  - 可扩展维数：如

    （时间、地区、销售渠道、*商品类型*

    销售额）



数据单元（Cell）

- 数据单元（单元格）：多维数组的取值
  - 可表示为：

    （维1维成员，维2维成员，...，维n维成员，变量的值）

  - 例如：

    （1997年1月，北京，批发，10000）

第二章 联机分析处理基本概念

- 数据分析模型
- **OLAP的提出**
- 多维数据结构
- 多维数据操作
- 多维数据模型的实现

多维分析的基本分析动作

- 切片（Slice）

  从多维数组选定一个二维子集，切出一个“平面”

- 切块（Dice）

  从多维数组选定一个三维子集，切出一个“立方体”

- 旋转

  改变一个报告（或页面）显示的维方向

- 钻取

  根据维层次，改变数据的粒度

切片的定义（1）

- 定义 1：

  在多维数组的某一维上选定一个维成员，

  即从 n 维数组选取n - 1维子集，

  设多维数组（维1，维2，…，维n，变量），

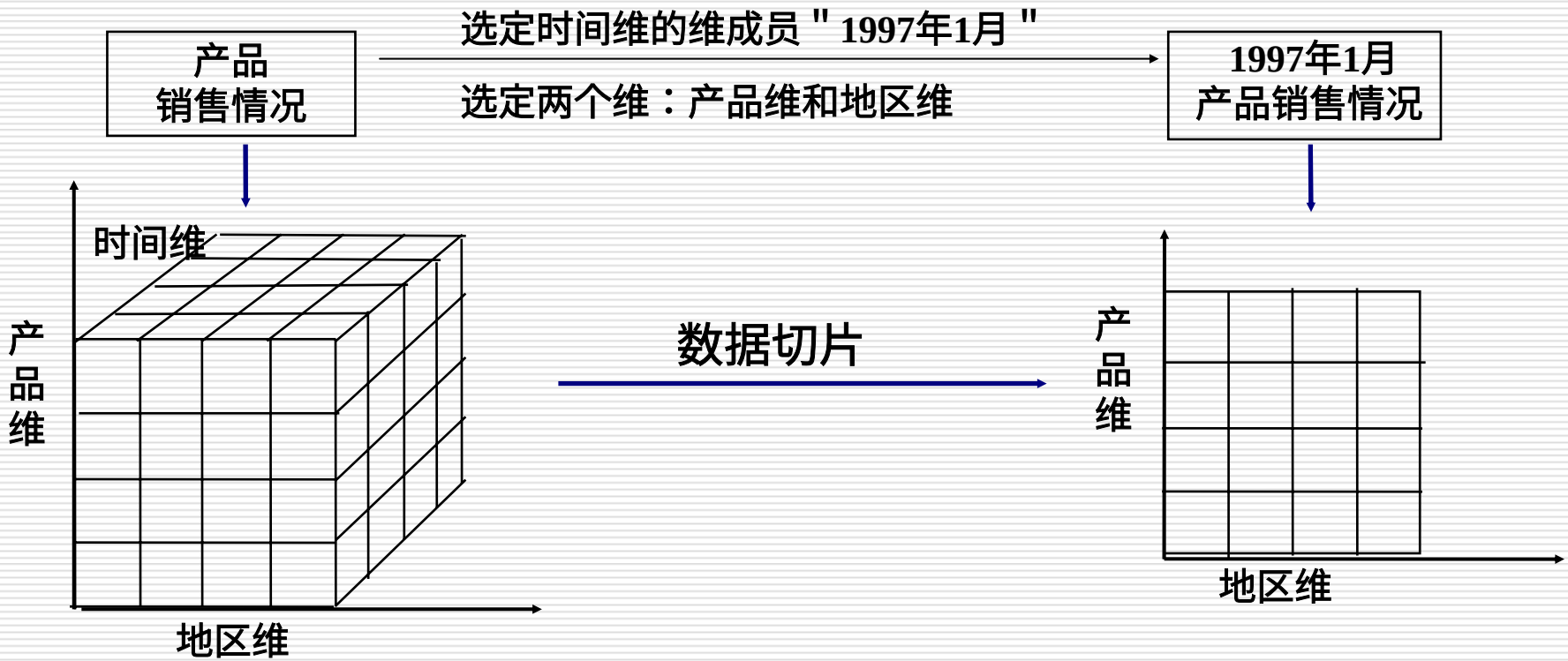
  在维 i 上，选定维成员Vi

  则：多维数组的 n - 1维子集

  （维1，…，维i-1，维成员Vi，维i+1，…,维n，变量）

  为在维 i 上的一个切片

切片举例（1）



注：多维数组（地区、时间、产品、销售额）

若在时间维上选定维成员“1997年11月”

得到时间维上的切片（地区、“1997年11月”、产品、销售额）

切片的定义（II）

- 定义 2

  选定多维数组的一个二维子集

  设多维数组（维1，维2，…，维n，变量）

  除维 i ，维 j 外，每个维都取定一个维成员Vk（1≤k≤n且k≠i，k≠ j）

  则：多维数组的二维子集

  （V1 , … , Vi -1 , 维i, Vi+1 , … , Vj -1 , 维j, Vj+1 , … , Vn,变量）

  简单表示为:（维i , 维j , 变量）为维i 和维 j 上的一个切片

- 按定义2进行切片，所得切片是一个二维“平面”（其它维的维成员都已确定）。

  二维“平面”易想象，易观察。

切片举例（II）

- 多维数组：

  （地区、时间、产品、销售渠道、销售额）

  选取地区维与产品维，其它维选定维成员

  时间：1997年1月　　销售渠道：零售

  得：

  （地区、“1997年1月”，产品，“零售”，销售额）

  即为：1997年1月零售的产品销售情况（各地区各种产品的销售额）

切块的定义(Ⅰ)

- 定义 1

  在多维数组的某一维上选定某一区间的维成员，

  即限制某一维的取值区间

- 切片是切块的特例，即限制的取值区间只取一个维成员
- 切块可看作由多个邻接的切片迭合而成
- 例如：多维数组（地区，时间，产品，销售额）

  在时间维上选定一区间：“1997年1月至1997年10月”

  得：（地区，“1997年1月至1997年10月”,产品，销售额） 为一切块

切块的定义(II)

- 定义 2

  选定多维数组的一个三维子集

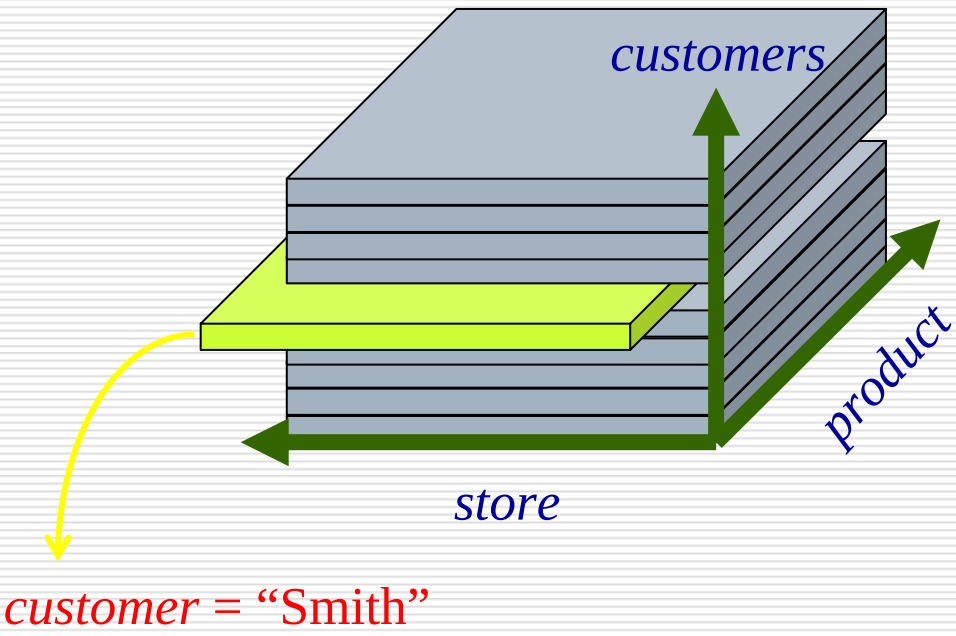
  设多维数组（维1，维2，…，维n，变量）

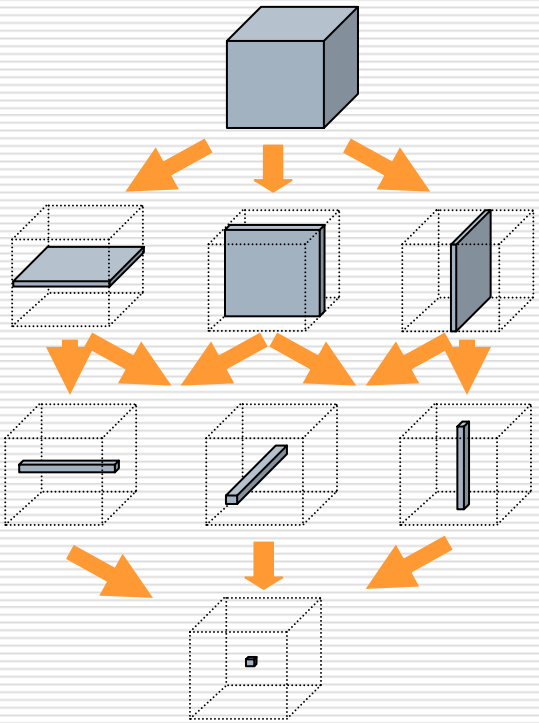
  选定三个维：维 i，维 j，维 k，（该三个维上可取任意维成员或某一空间），其他维上都取定一个维成员

  得：（维 i，维 j，维 k，变量 ）为一个切块。

切片和切块

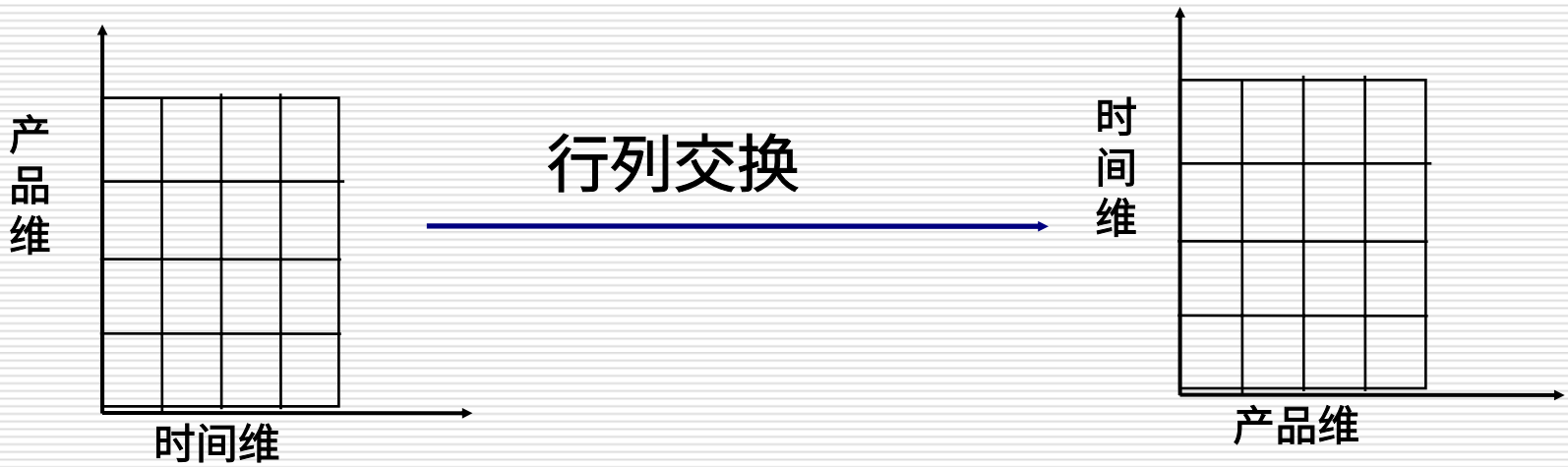
- 在一个或多个维度上的投影操作





旋转的含义

旋转：改变一个报告或页面显示的内容



(a) 把一个横向为时间，纵向为产品的报表
旋转成为
横向为产品和纵向为时间的报表

旋转的含义（续 I ）

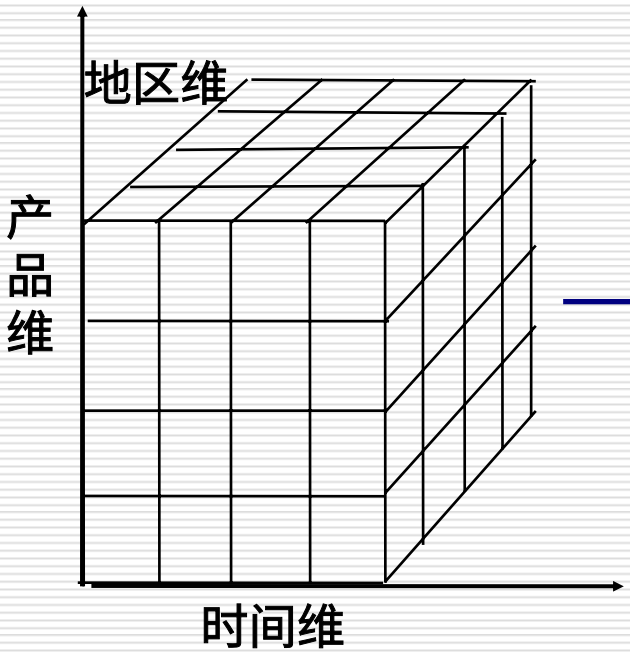
地区→		北京	上海
时间 ↓	产品 ↓	销售量	销售量
第一季度	:		
第二季度	:		
:			

将某行维换向列维

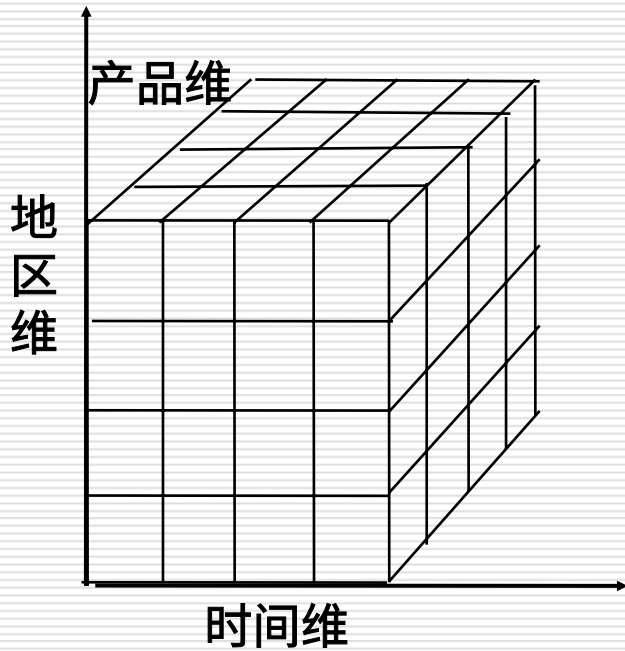
地区→	北京		上海
时间→	第一季度	第二季度	.....
产品 ↓			
: :			

(b) 把一个纵向为时间和产品，横向为地区的报表变成一个纵向为产品和横向为地区和时间的报表。

旋转的含义（续II）

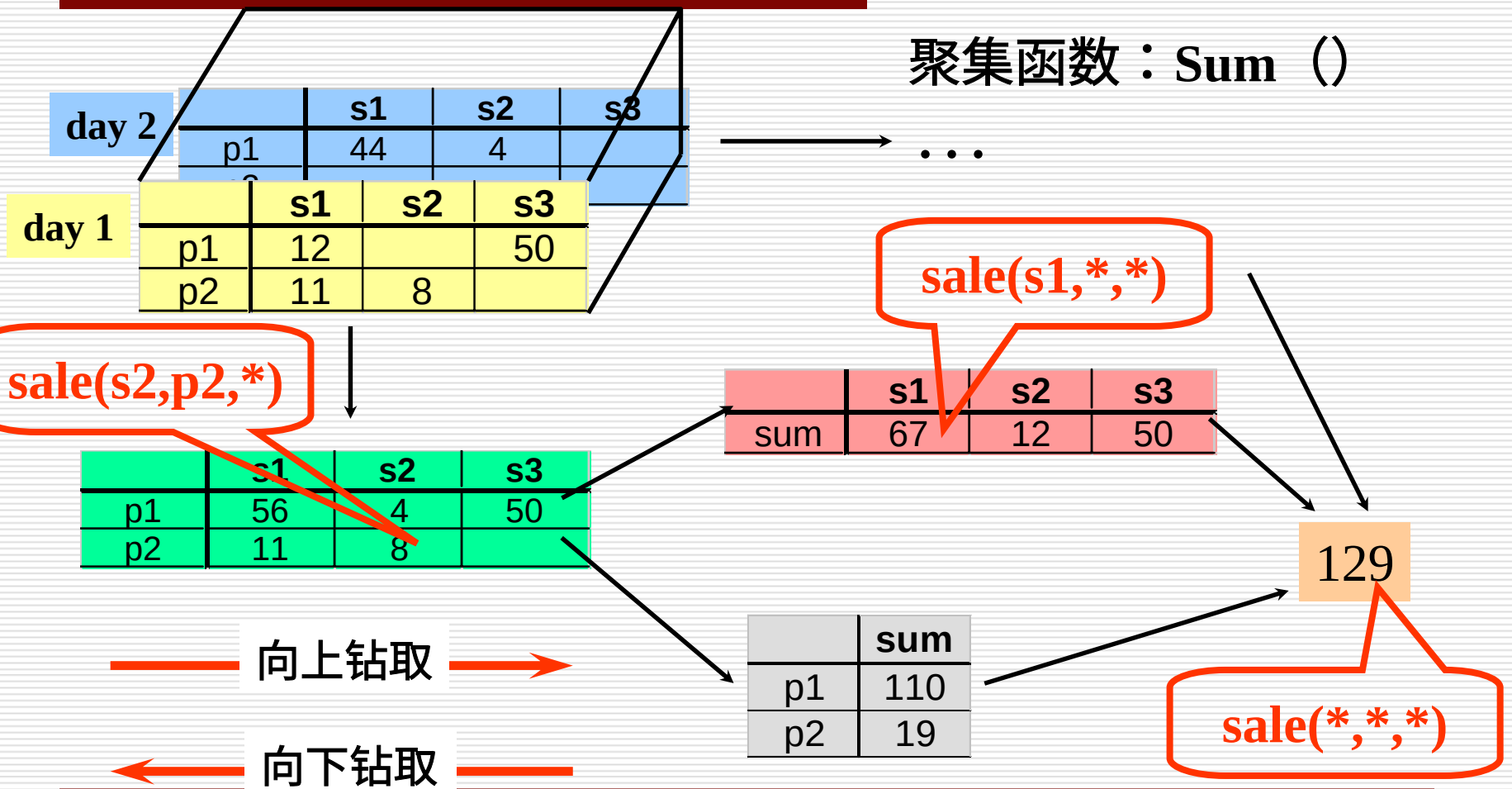


旋转以改变
页面显示



(c) 把一个横向为时间，纵向为产品的报表变成一个横向仍为时间和纵向旋转为地区的报表。

钻取操作


聚集函数：Sum（）
day 2
day 1
s1 s2 s3
p1 44 4
p1 12 50
p2 11 8
sale(s2,p2,*)
sale(s1,*,*)
sum 67 12 50
p1 56 4 50
p2 11 8
129
sum
p1 110
p2 19
sale(*,*,*)
向上钻取
向下钻取

OLAP的其它操作

- **Drill through（穿透）**
  - 钻取操作的进一步衍生，尤其对于ROLAP模式，直接得到最为底层的细节数据，数据通常以关系表的形式给出
- **Ranking（排序）**
  - 对数据单元的度量取值进行排序，获得top/bottom的若干数据取值。

OLTP与OLAP的比较

OLTP 数据	OLAP 数据
原始数据	导出数据
细节性数据	综合性或提炼性数据
当前值数据	历史数据
可更新	不可更新，但周期性刷新
一次处理的数据量小	一次处理的数据量大
面向应用，事务驱动	面向分析，分析驱动
面向操作人员，支持日常操作	面向决策人员，支持管理需要

- OLAP所用数据来自OLTP数据库
- 进行了预综合和多维化处理
- OLAP更强调界面的可视化和灵活性
- 可视化：多维报表，各种统计图形，…
- 灵活性：切片、切块、旋转；逐层细化，…

OLTP vs. OLAP 的比较

- **OLTP: On-Line Transaction Processing**
  - **Many short transactions (queries + updates)**
  - **Examples:**
    - **Update account balance**
    - **Enroll in course**
    - **Add book to shopping cart**
  - **Queries touch small amounts of data (one record or a few records)**
  - **Updates are frequent**
  - **Concurrency is biggest performance concern**
- **OLAP: On-Line Analytical Processing**
  - **Long transactions, complex queries**
  - **Examples:**
    - **Report total sales for each department in each month**
    - **Identify top-selling books**
    - **Count classes with fewer than 10 students**
  - **Queries touch large amounts of data**
  - **Updates are infrequent**
  - **Individual queries can require lots of resources**

OLAP & OLTP的主要区别（1）

不同的性能需求

- 联机事务处理 (OLTP):
  - 快速的相应时间非常重要 (< 1 second)
  - 在任何时候，数据随时更新，必须保持数据的一致性和完整性
- 联机分析处理 (OLAP):
  - 查询可能耗费大量的资源
  - 可能使得CPUs 和磁盘处于紧张的工作状态
  - 操作通常基于某一个时间点的静态的数据“快照”
- OLAP与OLTP必须实现环境分离
  - OLAP可能导致OLTP系统性能的降低，甚至崩溃
- 例如:
  - 分析查询需要计算所有的销售量
  - 为保证数据的一致性，防止脏数据的读出，对销售表进行“加锁”
  - 新的销售事务无法提交

OLAP & OLTP的主要区别（2）

不同的数据建模需求

- 联机事务处理 (OLTP):
  - 为保证数据的一致性，需要设计规范化的模式
  - 复杂的数据模型，包含大量的数据表
  - 查询和修改操作相对比较受限
- 联机分析处理 (OLAP):
  - 简单的数据模型非常重要
    - 允许业务人员执行各类即席查询
  - 通常采用非规范化的模型
    - 更少的连接操作 → 提高查询性能
    - 更少的数据表 → 易于理解数据模式

OLAP & OLTP的主要区别（3）

分析需要综合多个不同的数据源

- OLTP系统主要服务于某一个特定的应用系统
  - 例如：在线商场的订单管理系统
- OLAP需要集成多个不同的数据源
  - 包含销售、订单、采购等
- OLAP包含历史数据
  - 确定长时间范围内的一些模式
  - 发现一段时间内的变化情况
- 数据集成是OLAP系统的重点之一

第二章 联机分析处理基本概念

- 数据分析模型
- **OLAP的提出**
- 多维数据结构
- 多维数据操作
- 多维数据模型的实现

多维数据模型的实现技术

- **Relational OLAP (ROLAP)**
  - 利用关系数据库来存储和管理基本数据和聚合数据，并利用一些中间件来支持缺失数据的处理
  - 具有良好的可扩展性
- **Multidimensional OLAP (MOLAP)**
  - 利用多维数据库来存放和管理基本数据和聚合数据，其中需要对稀疏矩阵处理技术
  - 对预综合的数据进行快速索引
- **Hybrid OLAP (HOLAP)**
  - 利用关系数据库来存储和管理基本数据，利用多维数据库来存储和管理聚合数据。

多维数据的组织存放（细节数据）

- 例　维1：产品；维成员：冰箱、彩电、空调；
  维2：地区；维成员：东北、西北、华北；
  变量：销售量
- RDB中的数据组织

产品名称	地区	销售量
冰箱	东北	50
冰箱	西北	60
冰箱	华北	100
彩电	东北	40
彩电	西北	70
彩电	华北	80
空调	东北	90
空调	西北	120
空调	华北	140

MDB中的数据组织

	东北	西北	华北
冰箱	50	60	100
彩电	40	70	80
空调	90	120	140

MDB方法的优点（细节数据）

- 清晰简明，占用存储少
- 性能好，尤其像“冰箱销售总量是多少？”的查询

  RDB方法：找出有关“冰箱”的记录，再对销售量求和

  MDB方法：找到有关“冰箱”的行，按行求和

多维数据的组织存放（综合数据）

- RDB中的数据组织

产品名称	地区	销售量
冰箱	东北	50
冰箱	西北	60
冰箱	华北	100
冰箱	总和	210
彩电	东北	40
彩电	西北	70
彩电	华北	80
彩电	总和	190
空调	东北	90
空调	西北	120
空调	华北	140
空调	总和	350
总和	东北	180
总和	西北	250
总和	华北	320
总和	总和	750

多维数据的组织存放（综合数据）

- MDB中的数据组织

输入成员　　导出成员

	东北	西北	华北	总和
冰箱	50	60	100	210
彩电	40	70	80	190
空调	90	120	140	350
总和	180	250	320	750

输入成员

导出成员

- 表中交点处的数字称为数据单元
- MDB中的维对应RDB中的数据域（列）
- MDB中的数据单元（包括相关的维成员）对应RDB中的记录（行）
- 本表中有16个数据单元，上一表中有16条记录

MDB方法的优点（综合数据）

- 多维概念表达清晰，占用存储少
- 对数据进行综合的速度快（只需按行/列累加）
- 在RDB中，“ 总和 ”作为某个域上的取值（属性值）与列定义语义不符

多维数据库存储

- 由许多（经压缩的）类似于数组的对象构成
  - 每个对象由聚集成组的单元块组成
- 每个单元块按类似于多维数据的结构存储
  - 通过直接偏移计算进行存取
  - 每个对象带有（压缩的）索引和指针结构
- 分析时常需维间的组合
  - 从而需“旋转”（数据立方体）及“切片”
  - 须高效的稀疏数据处理能力

多维数据库存取

- 多维查询语言----MDSQL 类似于SQL

例：列出1996年1月到6月彩电在北京地区的销售量和成本

多维查询语言表达语法1：

```
Select Dimension PRODUCT 彩电
Select Dimension REGION 北京
Select SALES.units,SALES.cost
Across TIME DOWN REGION,PRODUCT,SALES
List Period JAN96-JUN96
```

多维查询语言表达语法2：

```
Get SALES.units,SALES.cost
By PRODUCT=“彩电”
By REGION=“北京”
By TIME=JAN96-JUN96
```

用关系结构表示多维数据

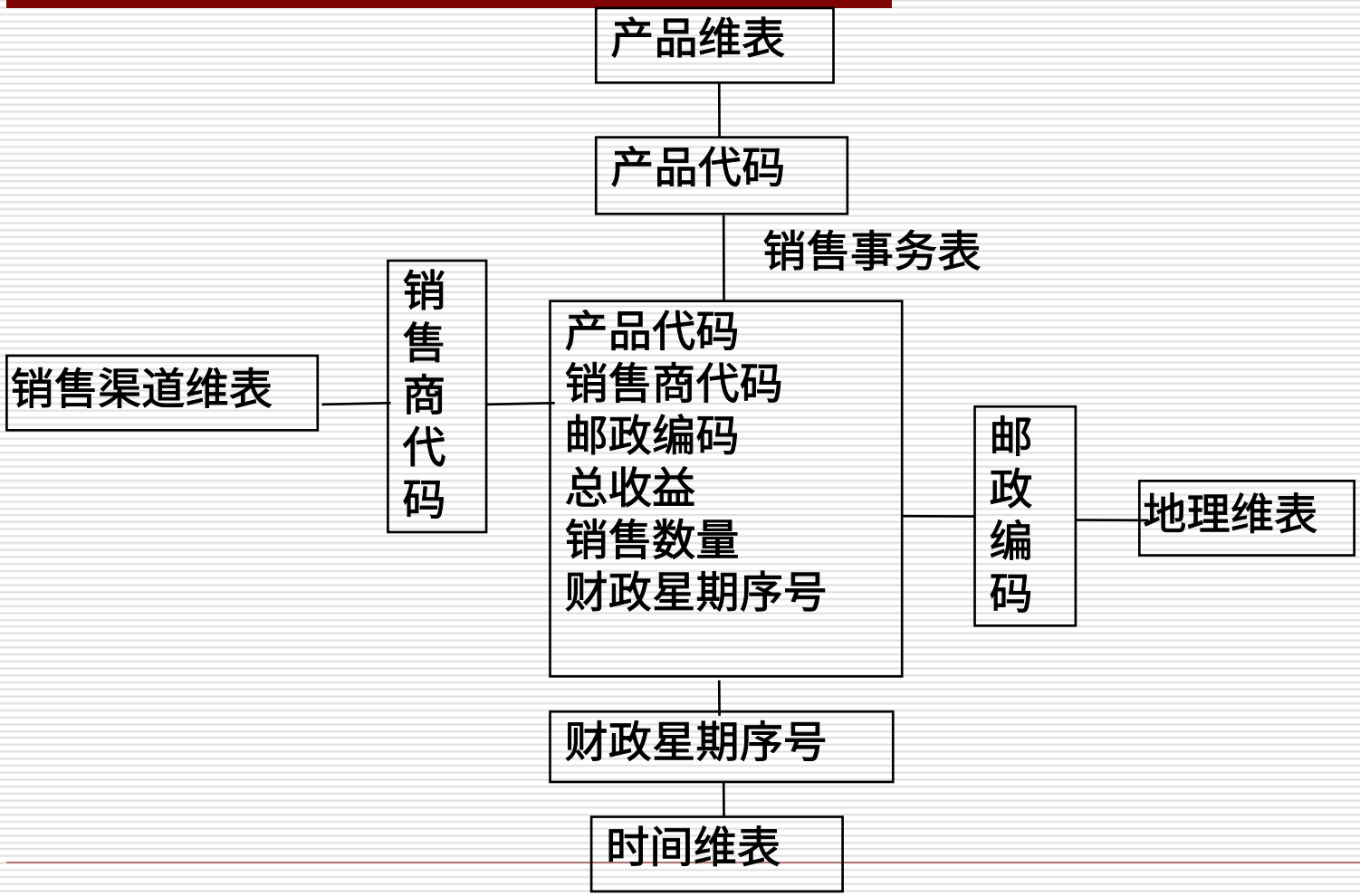
- 关系数据库使用广泛，相当成熟
- 用二维表 表达多维概念

  用两类表来表示多维结构: 事实表，维表
- 事实(fact)表

  用来存储变量值和各维的码值
- 维表

  用来存储维的描述信息(元数据)，包括层次和类等

星型模式（Star Schema）



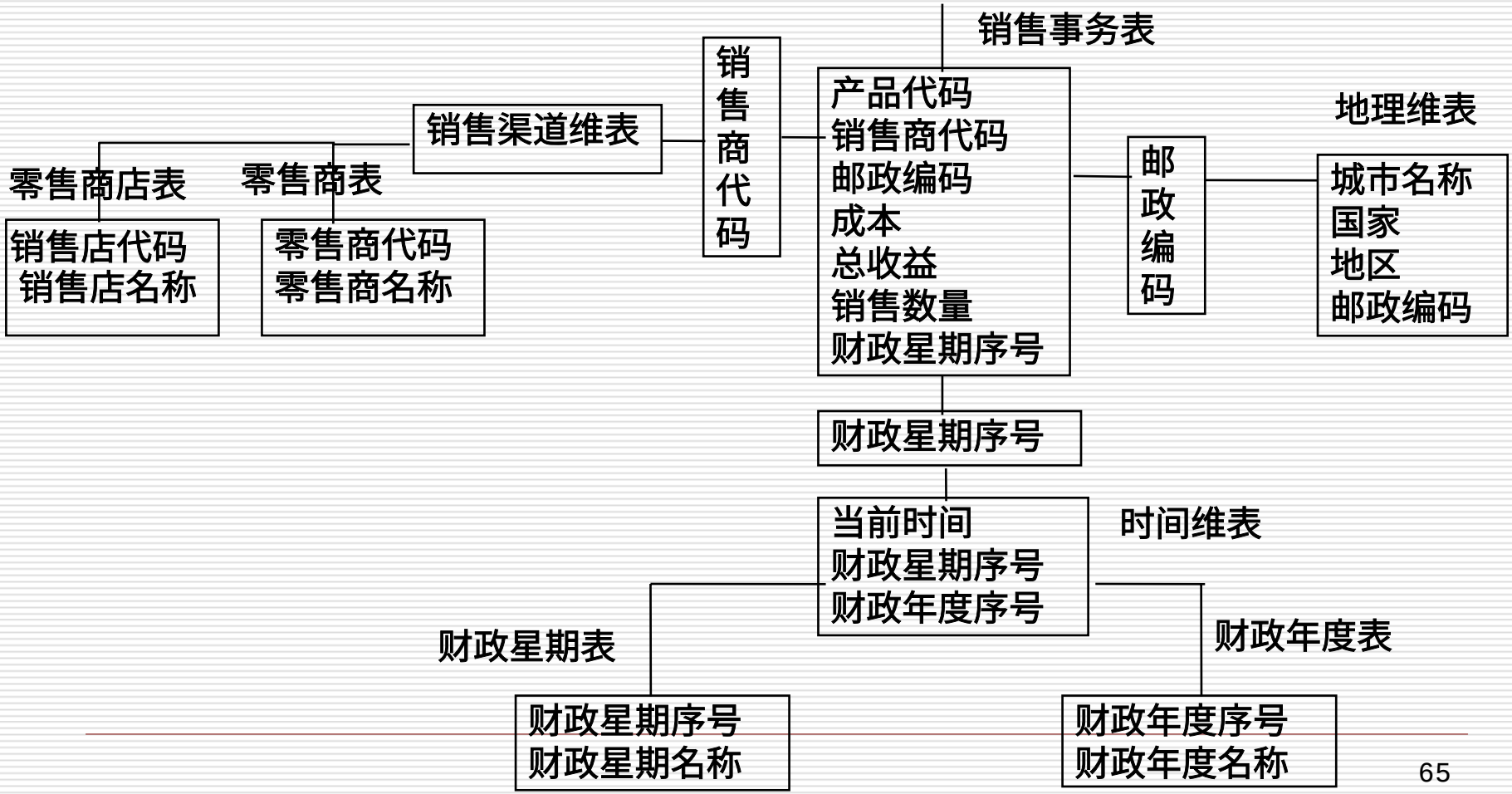
星型模式（Star Schema）

- 销售事务表为事实表
- 四个维表：产品、销售渠道、地理、时间
- 事实表中的变量值为总收益、销售量

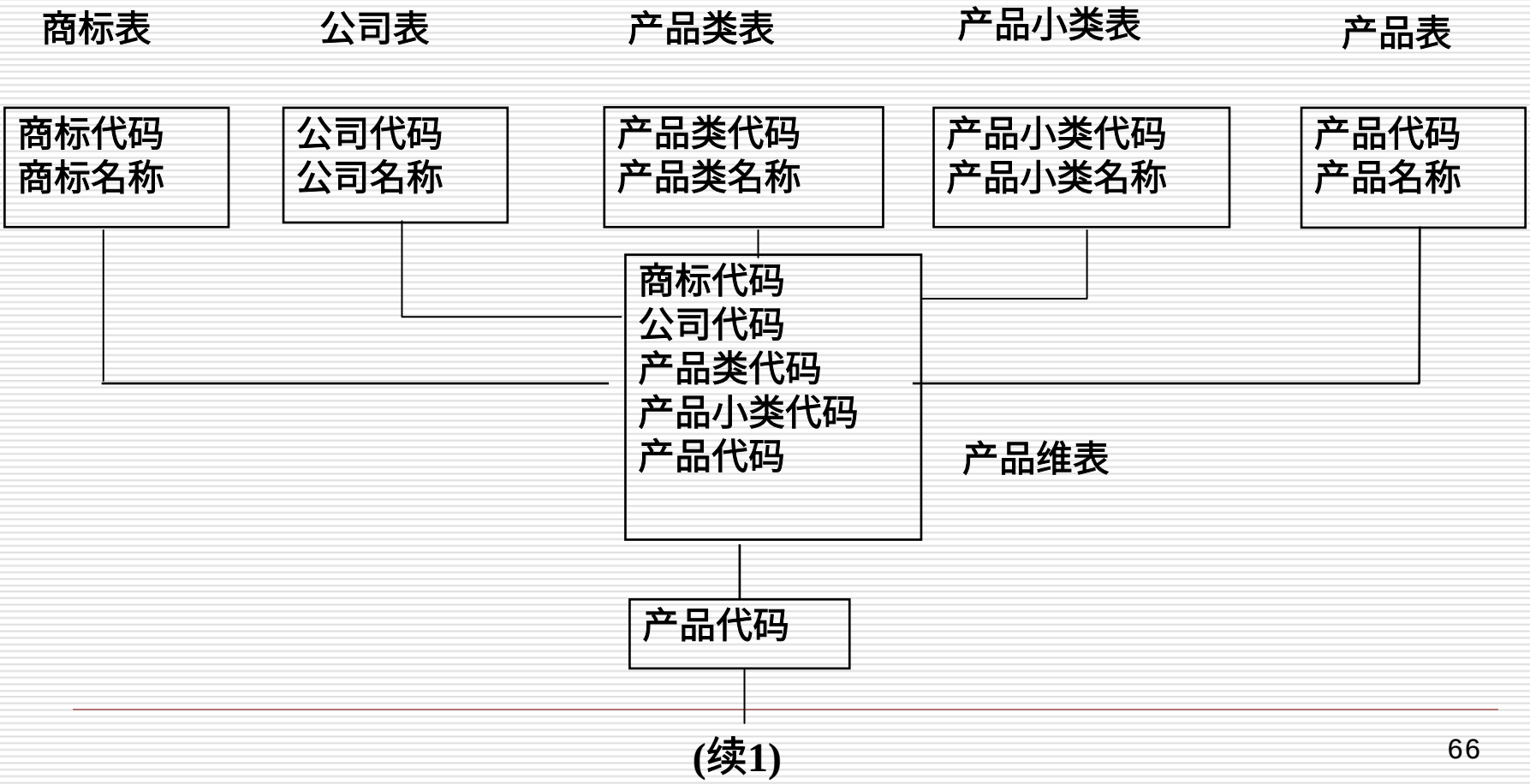
  维表码值为：产品代码、销售商代码、邮政编码、财政星期序号
- 事实表与维表的连接操作实现数据的多维查询，查询变量的值及对应的维成员

雪花模式(Snow Flake Schema)

(续1)



雪花模式(续)


商标表
公司表
产品类表
产品小类表
产品表
商标代码
商标名称
公司代码
公司名称
产品类代码
产品类名称
产品小类代码
产品小类名称
产品代码
产品名称
商标代码
公司代码
产品类代码
产品小类代码
产品代码
产品维表
产品代码
(续1)

雪花模式(Snow Flake Schema)

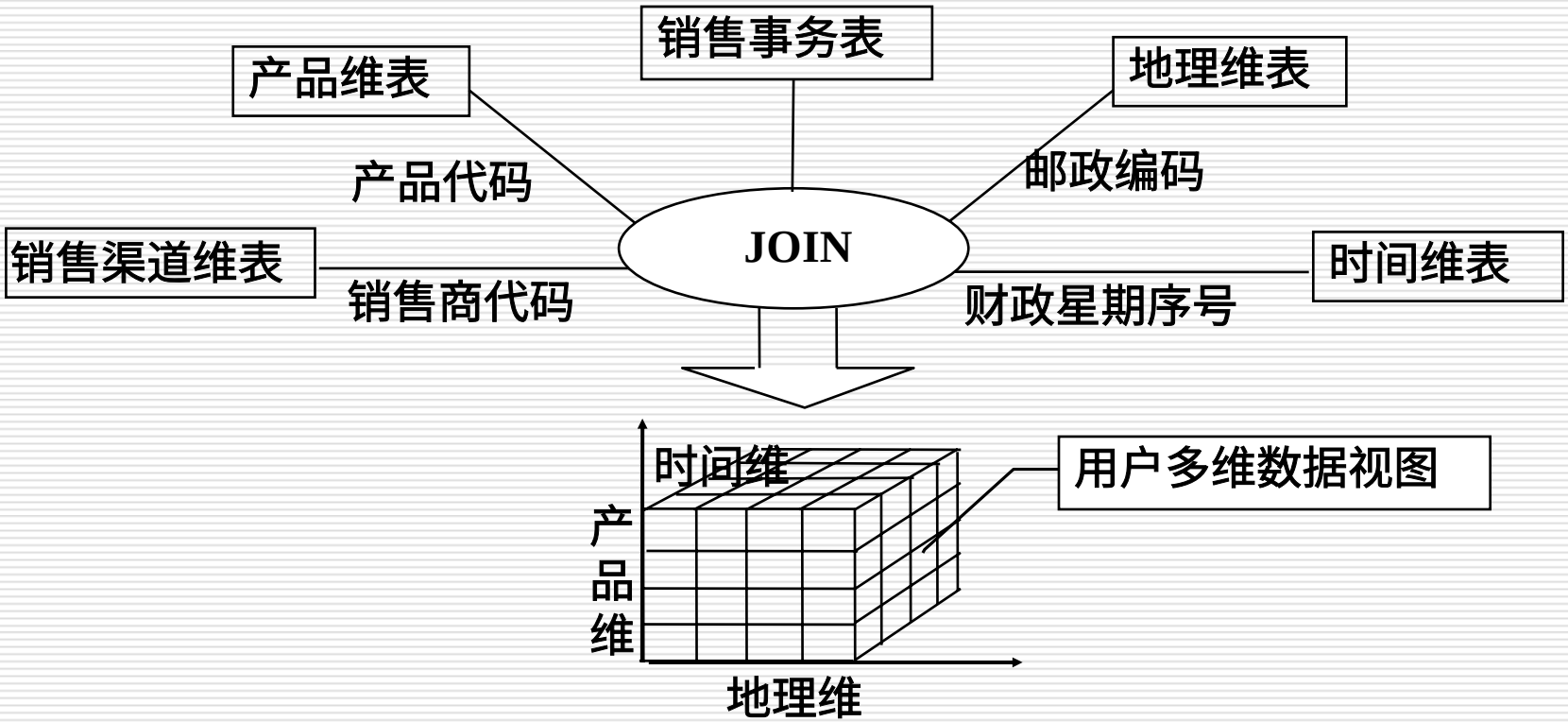
- 维的层次和类组合构成复杂的维
  - 无法用一张维表表示复杂的维
  - 用多张表来描述复杂的维

    每个维层次对应一张表；每个类属性对应一张表；

    维表用来存放上述各表（对应层次的表、对应类的表）的码值

多维数据的存取(事实的提取)

- 通过事实表和维表的连接操作来完成



多维数据的存取(事实的提取)

- 性能问题

  查询优化、索引技术

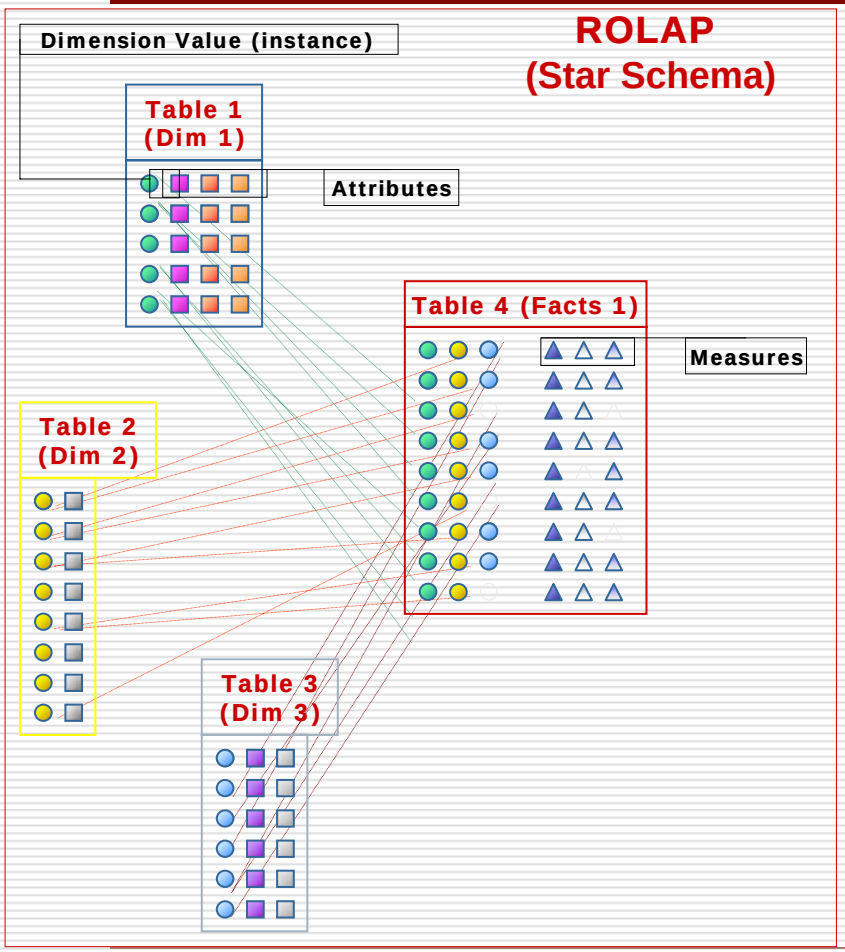
  例如：建立事实表、维表、综合表、系统表

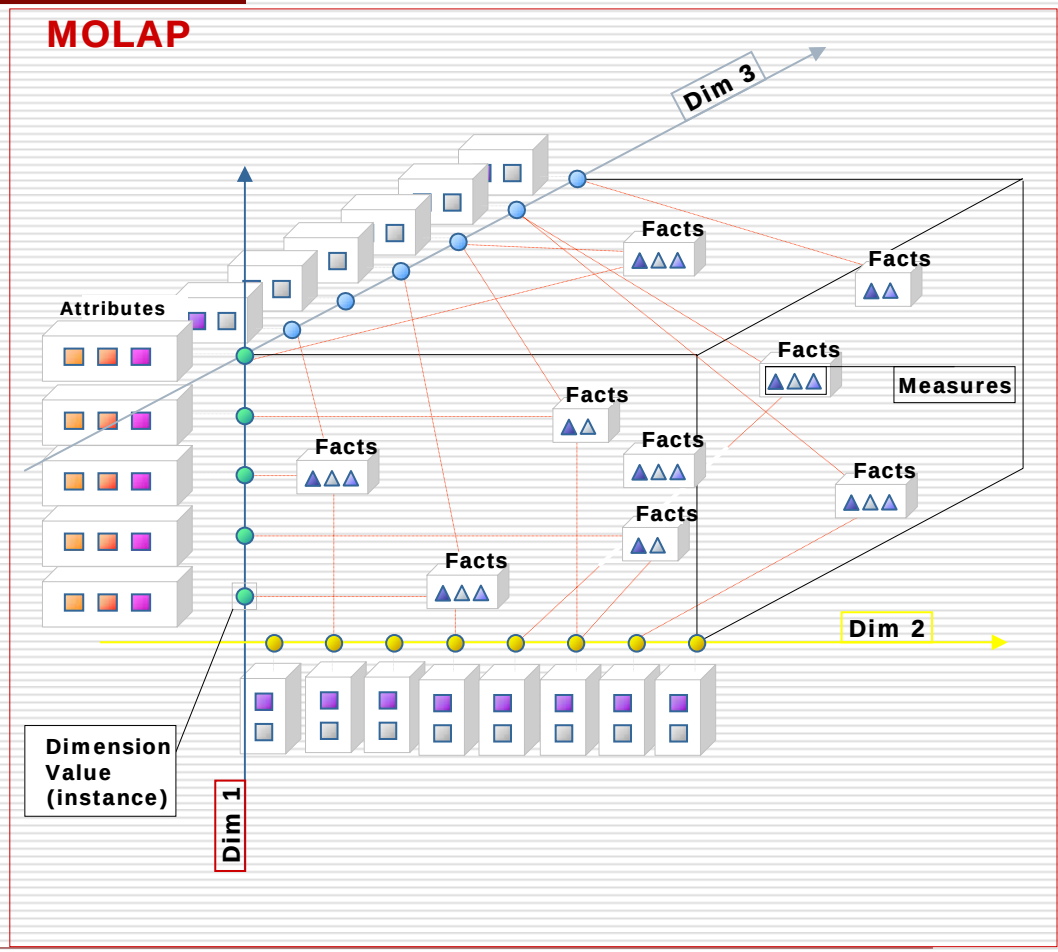
  - 维表：记录维的属性、维层次和类。
  - 综合表：查询常用的维成员和按这些维成员统计的事实数据

    例如：商标代码、地区、零售商代码、总收益、销售数量等组织成一张表

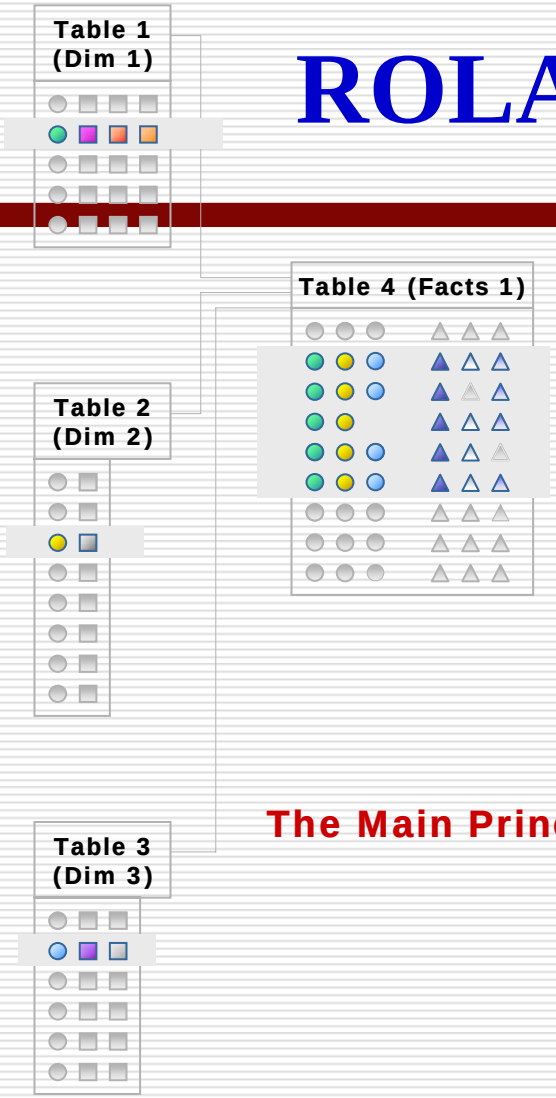
    提高性能；降低系统开销；避免每个维层次上的预综合
  - 系统表：有关事实、维、索引、汇总层次、综合表的字典信息

ROLAP & MOLAP的比较





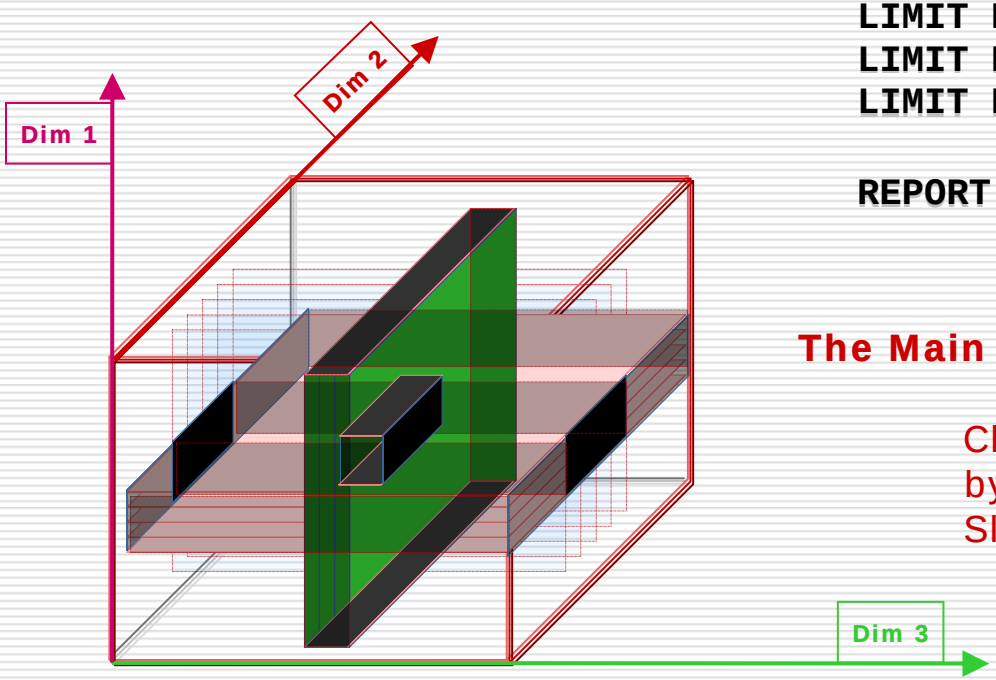
ROLAP中的一个查询




```
SELECT Fact_Column_1
      ,Fact_Column_2
  FROM Table 4    T4   -- Fact
      ,Table 1    T1   -- Dim 1
      ,Table 2    T2   -- Dim 2
      ,Table 3    T3   -- Dim 3
 WHERE T4.Dim_Col_1 = T1.Dim_Col_1
   AND T4.Dim_Col_2 = T2.Dim_Col_1
   AND T4.Dim_Col_3 = T3.Dim_Col_1
   AND T1.Dim_Property_2 = 'Product 1'
   AND T2.Dim_Property_1 = 'City 1'
   AND T3.Dim_Property_1 = 'Salesman 1'
```

The Main Principle: Building (dynamically) Views on the Star Schema Structure by joining Tables (Dim and Facts)

MOLAP中的一个查询




```
LIMIT Dim 1 TO 'Product 1'
LIMIT Dim 2 TO 'City 1'
LIMIT Dim 3 TO 'Salesman 1'

REPORT sales
```

The Main Principle:

Change the context of the desired facts by limiting the Cube(s) to the required Slice/Dice

两种实现技术比较

- **MOLAP结构**
  - 结构简洁、明了
  - 适应数据量相对较少的情况
  - 灵活性稍差
- **ROLAP结构**
  - 结构相对庞大
  - 适应数据量很大的情况
  - 灵活性好

MOLAP的数据存储和管理

- 多维数据库由类似于数组的对象构成；

  对象带有索引和指针结构；

  每个对象由聚集成组的单元块组成；

  单元块通过直接偏移计算进行存取；

- 以维及维成员为主线进行数据管理；

  数据封锁可以达到单元级

MOLAP的数据存储和管理

- 数据存储容量较ROLAP少

  往往利用RDB存储细节数据，MDB存储综合数据

- 元数据以内在方式处理

  元数据描述了层次关系、时间序列信息、报表项、安全存取控制、数据源以及预综合等等。

- 利用多维查询语言直接访问MDB（不借助附加程序）

ROLAP的数据存储和管理

- 以关系数据库系统方法进行数据存储和管理；
  安全控制和存取控制基于表；
  封锁基于表、页面或行；
- 多维概念下的安全及存取控制，RDBMS不支持，需由OLAP Server实现
- 数据存储容量大（因为RDB技术成熟）
  但为了提高性能，须建中间表（预综合），数据冗余大
- 元数据作为应用的一部分，由ROLAP Server管理
- 用户的分析（查询）请求，需SQL和附加的应用程序共同完成
  可以直接在细节数据上提供OLAP 的功能

ROLAP & MOLAP的适应性

ROLAP	MOLAP
• **When** you need a read- and load-optimized Data Store to load and query atomic data (Detail Data) and their linkage into a company-wide context • **When** you do frequently big reloadings (INSERT/UPDATEs) from the OLTP systems into your Data Store	• **When** you need extended analytic, forecasting and planning functions (period comparison, Ranking, Usage Pattern Analysis, etc.) • **When** you calculate and aggregate extensively • **When** you need a seamless ad-hoc analysis performance
Having both needs … Load and Store data from the ROLAP on a higher LEVEL into MOLAP and provide Drill Through Mechanism from MOLAP into ROLAP for lower LEVEL (Details)	

MOLAP适应性

MOLAP的预综合较高

- 适应维数动态变化较差

  增加一维，“超立方”规模迅速增长

- 适应数据变化较差

  数据（或计算）变化时，重计算量相当大

- 适应大数据量较差

ROLAP适应性

ROLAP的预综合度灵活，一般较低

- 适应维数动态变化较好

  增加一维，需增加一些维表、综合表及事实表中的内容

- 适应数据变化的范围大

- 适应大数据量的能力较强，技术成熟