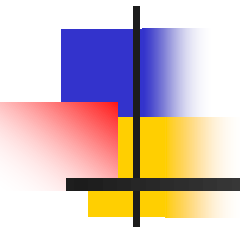
数据仓库与大数据工程

Data Warehouse and Big Data Engineering

第5部分 数据模型及设计方法

5-2 多维数据模型及其设计

北京交通大学计算机与信息技术学院

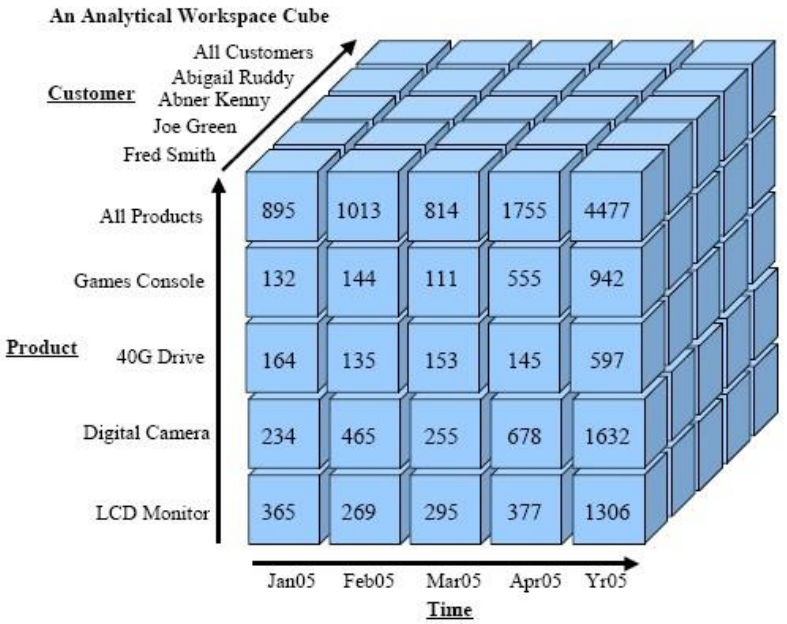


第五部分-2 多维数据模型

Multidimensional Data Model

内容概要

- 1 引言
- 2 多维数据模型简介
- 3 一种多维数据模型的形式化描述
- 4 物化视图概述
- 6 小结



1 引言

- 数据仓库架构特点
  - 操作型应用和分析型应用分离出来
  - 在数据仓库端存储的数据量一般都是很大
  - 涉及到的数据表很多
  - 数据性质，变化速度都不一样
  - 为了提高性能，引入了高粒度级的统计或综合数据
- 数据仓库面向的分析应用需求特点
  - 决策支持需求不固定，比较灵活
  - 需要支持即席查询
  - 复杂分析涉及数据范围和数据量大

数据仓库中的数据关系和应用特点

- 数据与数据之间可能存在计算关系
  - 通过关系数据库系统中的计算代码体现
  - 如最低粒度级数据、日数据、月数据和年度数据之间存在的计算关系
- 数据存在各种各样的应用目的，不同应用各有各的特点。
- 各种应用逻辑是差异可能很大。
- 各种应用基本上是通过各种定制的代码实现的，是非系统化的(不是由数据库系统自动实现的)。

数据仓库分析应用的复杂性

- 用户的查询需求越来越复杂，表现为
  - 数据分析的多角度性
  - 数据分析的集合性、综合性
  - 数据提取的动态性
  - 涉及数据量大
  - …
- 关系数据库系统中简单数据模型已不能完全满足这些需求。为此，需要研究新的、有效的、表达能力强的数据模型以解决这些问题。

纯关系型数据库不能很好的支持分析应用逻辑

- 分析型应用中可能会出现各种复杂的处理模型，纯粹的关系数据库系统中一般并不提供这些处理的支持。
- 以关系型数据库为基础的数据仓库的数据缺乏对应用逻辑模型(功能模型)的系统性支持。
- 纯关系数据库在支持分析应用方面存在缺点
  - 在关系模型所对应的关系表上所能实现的数据利用或分析有关的功能相当有限；
  - 所能提供的分析功能不够灵活；
  - 不同粒度层次间的数据维护实现较为麻烦；
  - 因缺乏模型化、系统化的工具支持，而需要编写大量的代码实现各类功能。

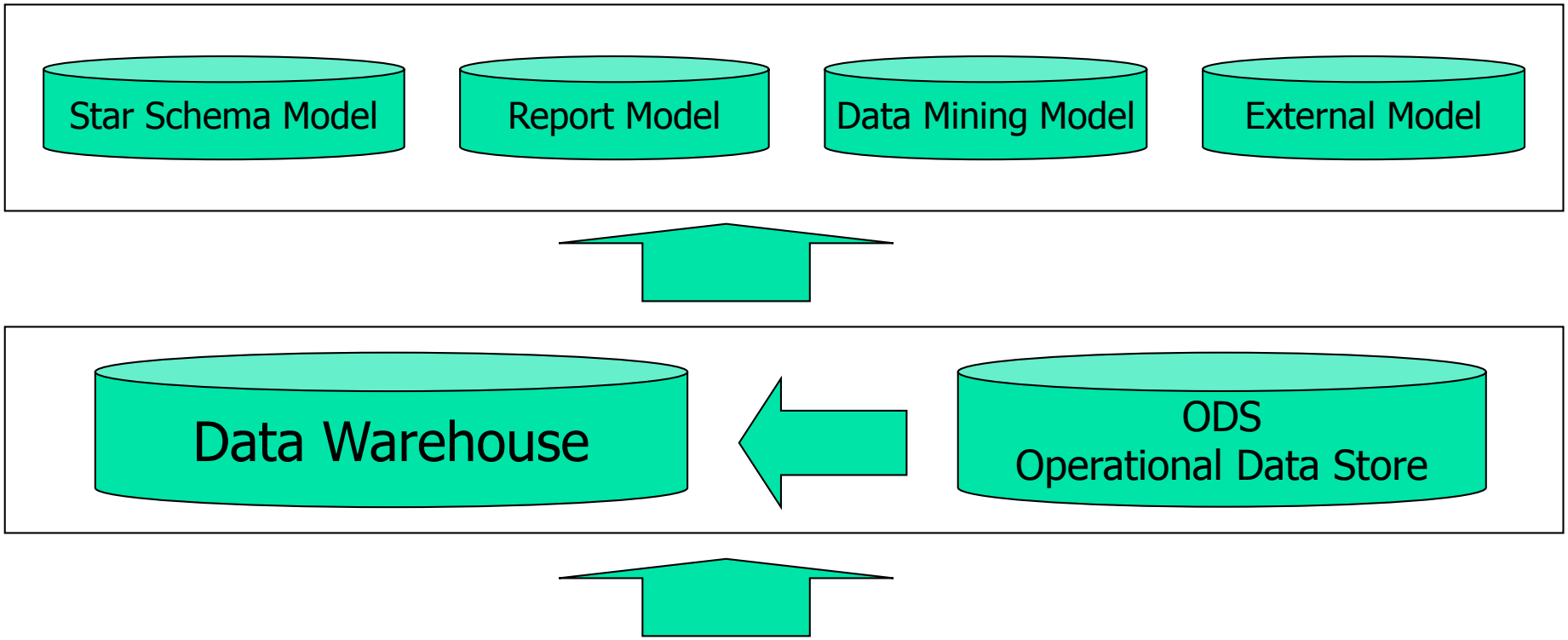
如何解决这些问题？

- 根本办法
  - 对处理的模型进行研究，归结出具有一些代表性的、通用的处理模型。
    - 多维分析模型
    - 数据挖掘处理
    - 数据的自动维护和计算
  - 研究数据模型，提高数据模型的通用性、规范性和功能适应性，用于表示现实世界中的各种数据的结构。
  - 在系统化的、规范的数据模型基础上，就有可能实现某种程度的自动汇总计算、自动维护、自动集成和有效的分析应用支持。

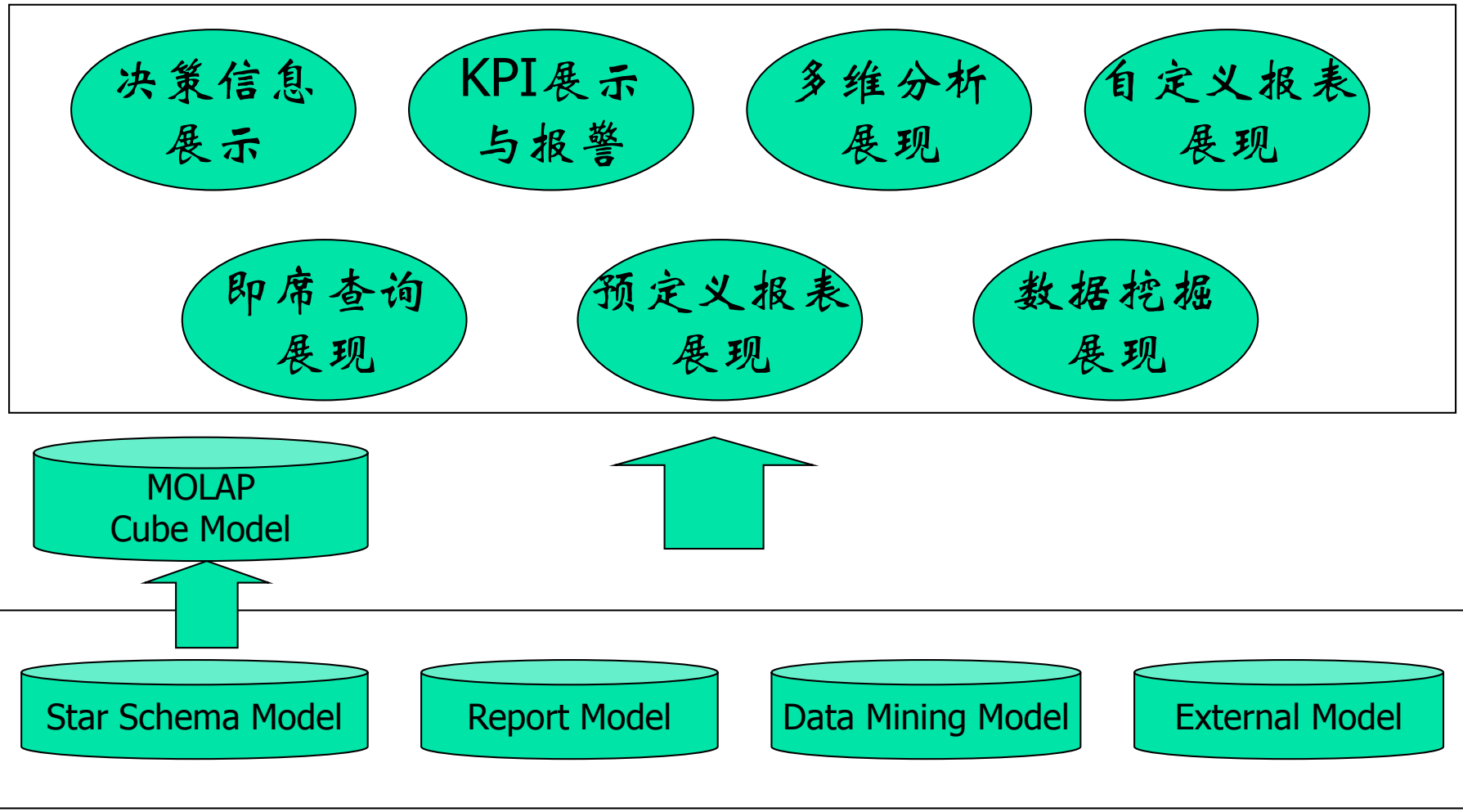
标准化处理模型->数据模型

- 有些处理模型可以标准化，如常见的多维分析动作
- 相应地引出了相应的支持这些标准处理模型的标准数据模型
- 软件厂商相继推出相关的解决方案和软件工具
- 在RDBMS中可以看到许多新的对象
  - dimension, cube, hierarchy, level, …

实例模型第三层模型


Star Schema Model
Report Model
Data Mining Model
External Model
Data Warehouse
ODS
Operational Data Store

实例模型第四层应用与模型



背景案例

1.能否根据T1—T4，计算Q1或Q2的结果？
2. 指定存储平台，设为Oracle或MySQL，你能否写出查询脚本？
3. 脚本效率如何？

- 给定如下模式的数据
  - 销售数据表T1(商店ID，商品ID，销售时间，销售量，单价，金额)
  - 商店表T2(商店ID，商店名称，所在小区ID，所在城区ID，所在城市ID，所在省份ID)
  - 商品表T3(商品ID，商品名称，所属小类ID)
  - 商品类别表T4(小类ID，小类名称，大类ID，大类名称)
- 现有查询模式如下
  - 日销售情况Q1(商店ID，商品ID，日期，总销售量，平均每笔销量，总金额)
  - 各城区商品小类周销售情况Q2(城区ID，商品小类ID，商品小类名称，周，周总销售量，周总金额)

Oracle上的SQL语句实现

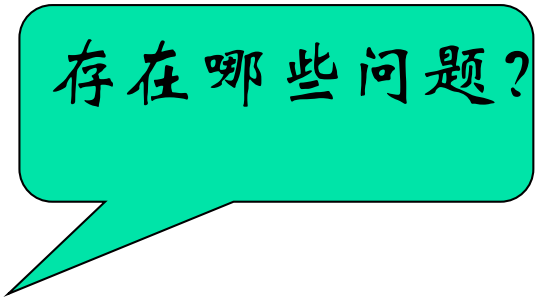
- Q1

```
select T1.商店ID, T1.商品ID, to_char(销售时间,'YYYY-
    MM-DD') 日期, sum(销售量), avg(销售量),sum(金额)
from T1
group by T1.商店ID, T1.商品ID, to_char(销售时间
    ,'YYYY-MM-DD')
order by T1.商店ID, T1.商品ID
```

- Q2？
  - 先需要对时间进行按周进行处理。
  - 然后作多个表的关联
  - 再进行分组处理

提高效率的办法

- 预计算，生成表T5和T6
  - 日销售情况表T5(商店ID，商品ID，日期，总销售量，平均每笔销量量，总金额)
  - 各城区商品小类周销售情况表T6(城区ID，商品小类ID，商品小类名称，周，周总销售量，周总金额)
- 预计算需要解决问题
  - 建表
  - 编写初次计算代码
  - 编写基于新增数据的内容维护代码，加入到计划任务中



新的需求

- 1.希望能让用户随时变换数据分析的角度，让用户能随时得到多种类型的数据。
- 2. 希望少写代码，只需做简单的设计，不再写结构生成、数据生成和维护的代码。
- 3. 希望有一个系统能感知用户的需求变化，自动选择对象预计算以提高查询响应速度。
- 4. 希望能有工具自动设计数据模型，展示各种格式的数据报表。
- 5. …

数据模型与多维模型

- 数据模型有关问题是企业信息化领域包括数据仓库领域内的重要而且是核心问题；
- 在这些问题中，数据的多角度性也就是多维性，是重要而基本的概念。
- 多维数据模型是用于高效实现多维分析的一种数据模型。
- 多维数据模型是支持OLAP分析的数据模型基础。

多维数据模型的功能目标

- 多维数据模型的功能是描述多维数据，有效支持多维分析操作。
- 多维数据模型应该达到以下目标
  - 逻辑上表示主题中的实体和实体间的关系；
  - 逻辑上表示主题各属性的定义域，即维的定义域；
  - 逻辑上应该能支持一定程度上的自动聚集逻辑，实现数据的动态计算，视图间的动态计算；
  - 逻辑上应该能实现有效的维护机制；
  - 能支持多维分析；
  - 支持分布式逻辑。

各种多维模型

- 为了有效地表示现实世界中的各种数据模型，人们提出了多种多维模型
  - 超级立方体格, hyper cube
  - 数据立方体格, data cube
  - 视图DAG图
  - …

多维数据模型相关的关键词

- 多维，模型，格，视图，基事实表
- Dimension, Data cube , Hierarchy, Materialized View, Base fact table, Hyper cube, Lattice.

内容概要

- 1 引言
- 2 多维数据模型简介
- 3 一种多维数据模型的形式化描述
- 4 物化视图概述
- 6 小结

2 多维模型简介

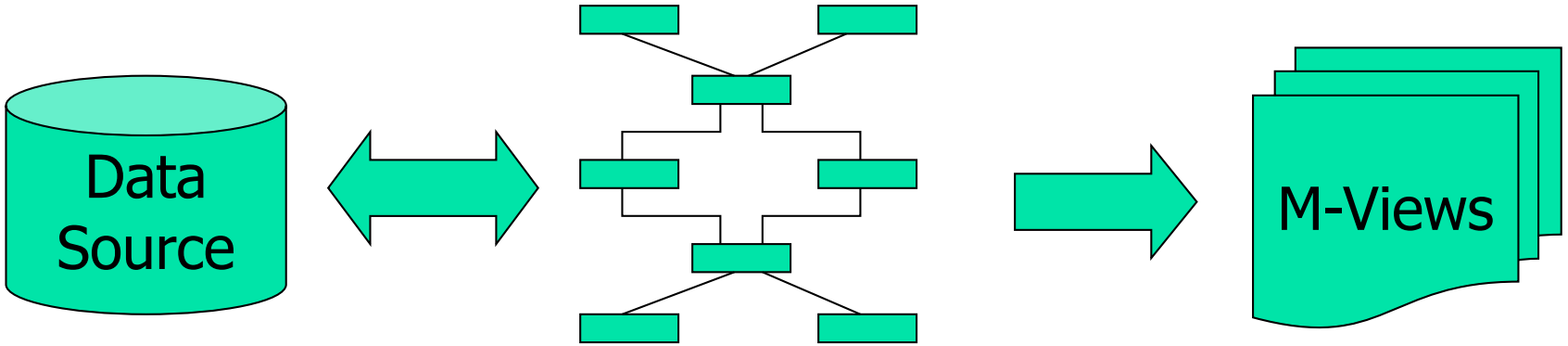
- 2.1 多维视图与维的逻辑结构
- 2.2 维层次结构的物理表示
- 2.3 多维空间与数据立方体模型
- 2.4 分布式模型

2.1 多维视图与维的逻辑结构

什么是多维数据模型？

- 多维数据模型(Multidimensional Data Model)
  - 数据模型(Data Model):
    - a model used to describe data
    - Through which user can get data
  - 多维性(Multidimensional):
    - Providing data multi-dimensionally
    - Multidimensional views
- 用于支持多维分析(在线分析处理)

数据源、多维模型、多维视图



多维视图

- 在常见的概念中，多维模型的数据视图或数据集为多维空间中的点集，这样的数据视图被称为多维视图(multidimensional view)
- 多维视图中的属性分为
  - 维属性 dimension attribute
  - 度量属性measure attribute。
- 多维视图的简单模式描述
  - MDV=(d_1, d_2, …, d_n, m_1, m_2, …, m_m)

多维空间举例

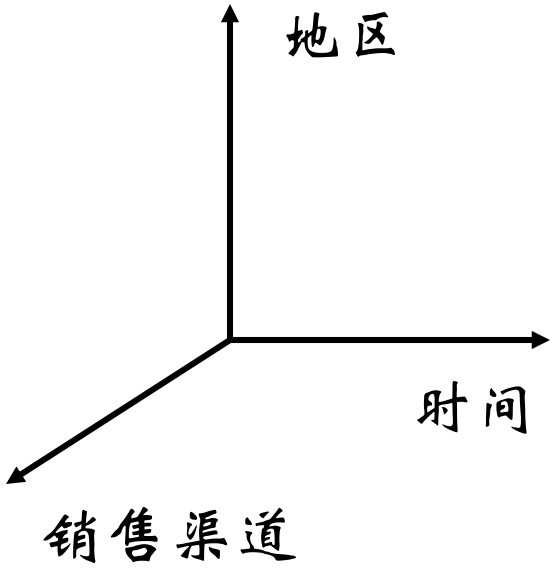
- 维
  - 销售渠道：零售，批发
  - 时间：各季度
  - 地区：北京，上海，广州，西安
- 度量指标
  - 销售额
- 该多维视图的模式为
  - MDV=(销售渠道，时间，地区，销售额)

同基度量指标

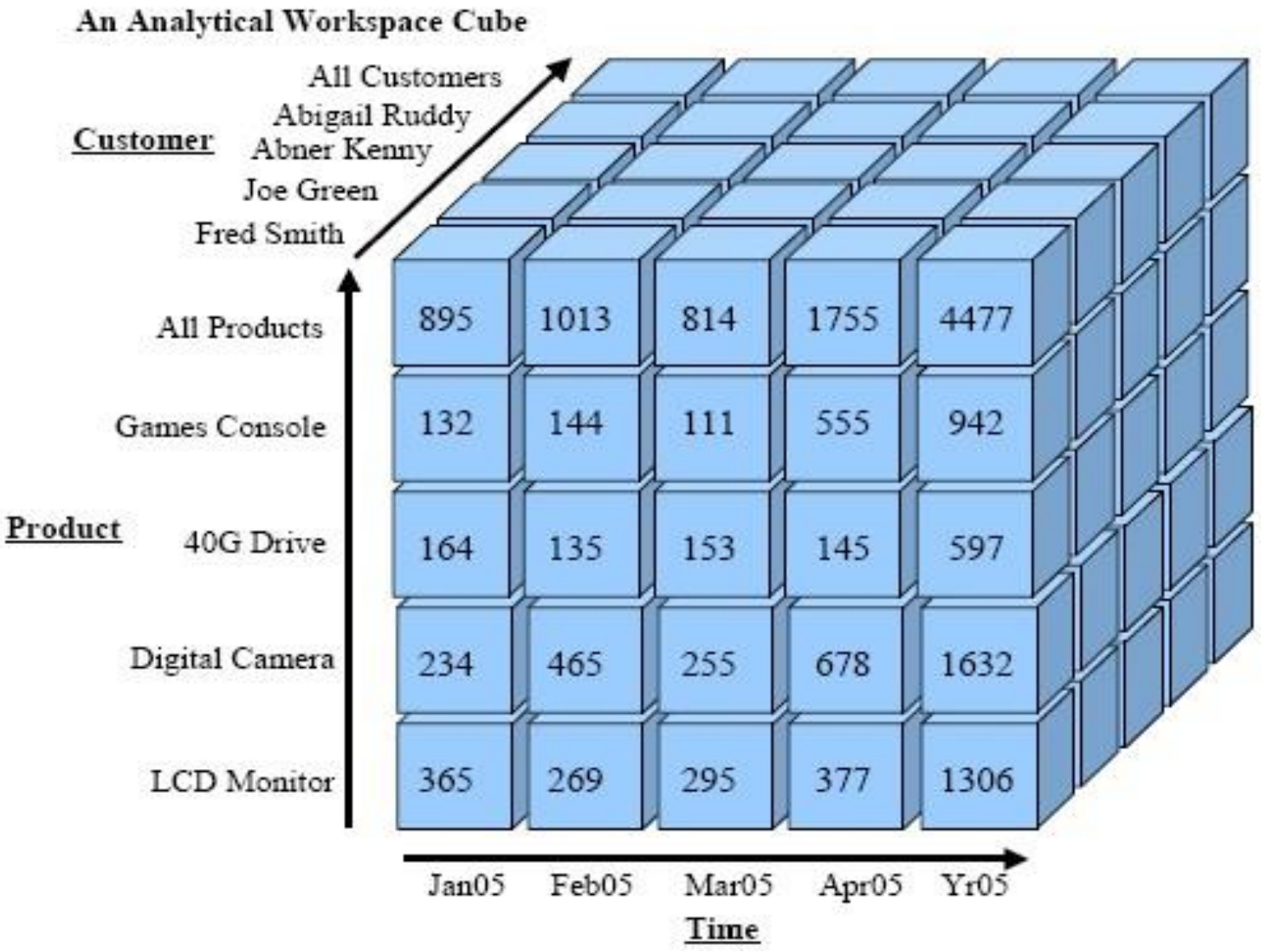
- 同基度量指标的概念
  - 同基度量指标是指定义在同一多维空间下的两个不同指标。
  - 如
    - 最大销售额，最小销售额，平均销售额，盈亏。

例 三个维度的销售额

时间	销售渠道	地区	
		北京	上海
一季度	零售	1000	2000
	批发	50000	40000
二季度	零售	1200	1800
	批发	52000	41000
三季度	。。。	。。。	。。。



多维数据的立方体例子


An Analytical Workspace Cube
All Customers
Abigail Ruddy
Customer
Abner Kenny
Joe Green
Fred Smith
All Products
Games Console
Product
40G Drive
Digital Camera
LCD Monitor
895
1013
814
1755
4477
132
144
111
555
942
164
135
153
145
597
234
465
255
678
1632
365
269
295
377
1306
Jan05
Feb05
Mar05
Apr05
Yr05
Time

多维空间

- 不同维的维属性的组合构成了视图的多维空间，用于描述度量属性。
- 度量属性的值是在多维空间中的某一个点上的取值。
- 定义域和取值空间
  - 多维空间的各维属性决定了度量属性的定义域
  - 度量属性也有自己的取值空间。

多维数据模型需要解决的核心问题

- 1. 表达多维空间与多维视图
- 2. 描述与实现多维空间之间的关系
- 3. 实现多维空间的数据计算
- 4. 实现多维空间的数据存储
- 5. 实现数据的维护
- 6. 解决性能问题
- 7. 可扩展性问题

多维数据模型的基本构成

- 由事实表和维表构成
  - 事实表(Fact Table)
    - 包含度量指标和维度外键
  - 维表(Dimension Table)
    - 包含维度、层次

事实表及其属性

- 对于数据模型中的核心数据表，一般称为事实表(fact table)，事实表的属性(字段)可以分成三类：维属性，度量属性，其它属性。
- 一个维一般都包含有多个层次属性(Levels)。
- 例如
  - 日期维:日，月，季，周，年
  - 货物类别维:货物ID，货物小类，货物大类

维属性的层次性

- 维的这些属性可以具有层次（Hierarchy）特点，每一层属性都是对上一层属性在概念上的泛化（generalization）或者是归类化。
- 泛化
  - 时间→日期→月→年
  - 产品→产品小类→产品大类

{20070101—20070131}→200701
Date→Month

各种维逻辑结构模型

- 多维模型的维的不同逻辑结构模型主要包括
  - 简单的不考虑维层次结构的模型
  - 单链层次结构模型
  - 代数格层次结构模型
  - 用于特殊应用模型的一些层次模型。

(1)不考虑维层次结构的模型

- 最简单的一种维逻辑结构
- 不考虑维的层次结构
- 仅从核心事实表出发，以维的最细节层属性，即从与事实表对应维属性的相同细节层出发，对事实表进行多维分析。

例

Fact table

OrderNo
SalesPersonID
CustomerNO
ProdNo
DateKey
CityName
Quantity

以这种多维模型为基础的多维系统能自动展示如下多维视图：
(OrderNo, CustomerNo, ProdNo, DateKey, Quantity)
(CustomerNo, ProdNo, DateKey, Quantity)
(CustomerNo, DateKey, Quantity)

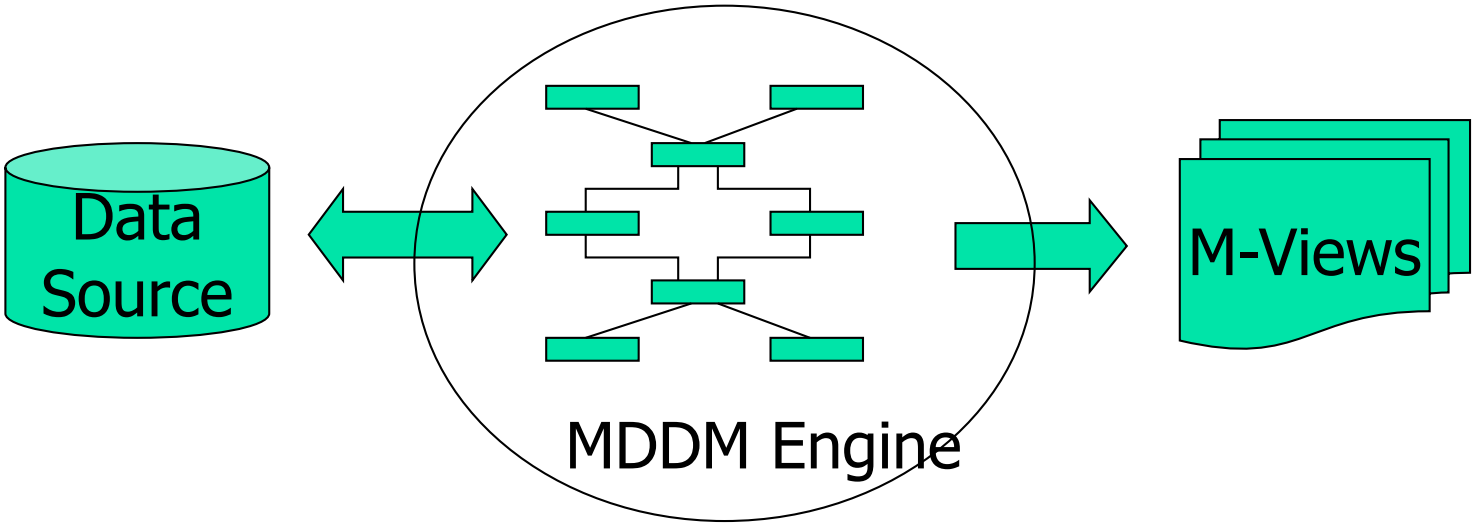
无法自动展示高粒度级的视图
(大客户，月，平均数量)
(产品大类，季，总数量)
(地区，月，总数量)
要提供这些视图还需要额外编码

应用和缺点

- 应用
  - 许多早期有关数据仓库各种应用的讨论或算法。
- 缺点
  - 这种维逻辑结构模型显然过于简单，采用这种多维模型的系统无法有效地支持一些基本的OLAP操作，如钻取(drill down)和卷取(roll up)等。

原因分析之例

- 例如
  - 设有数据处理引擎E，设有基事实表，如果引擎E采用不考虑维层次结构的多维模型。
  - 假设时间维的最细节层为日，即基本事实表中保存的每天的操作型数据。



续

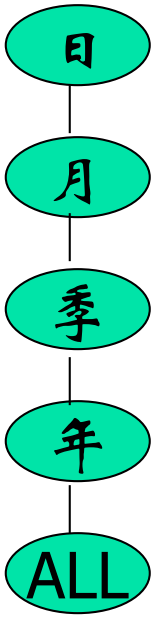
- 如果要实现每月的数据汇总，即roll up操作：
  - 因为引擎不知道日与月之间存在的关系(原因：数据模型没有相应的关系表示)，则为了实现这项功能，用户需要单独编写代码，引擎无法实现系统的支持。
- 如果有了月的数据(用户自己的编写代码显示出来的月数据)，如果要想看每天的细目
  - 也无法从月这个起点转换到天(drill down)，也需要额外编写代码。

非层次模型小结

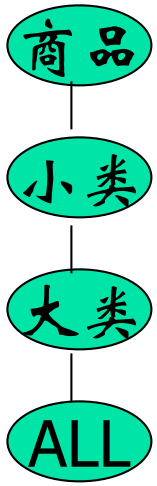
- 各种系统多数采用考虑维层次结构的逻辑模型。
- 所有支持层次结构的模型都隐含支持不考虑层次结构的模型。

(2)单链模型

- 一种最简单的考虑维层次结构的模型
  - 链状层次模型
  - 能表示最常见的维层次结构。



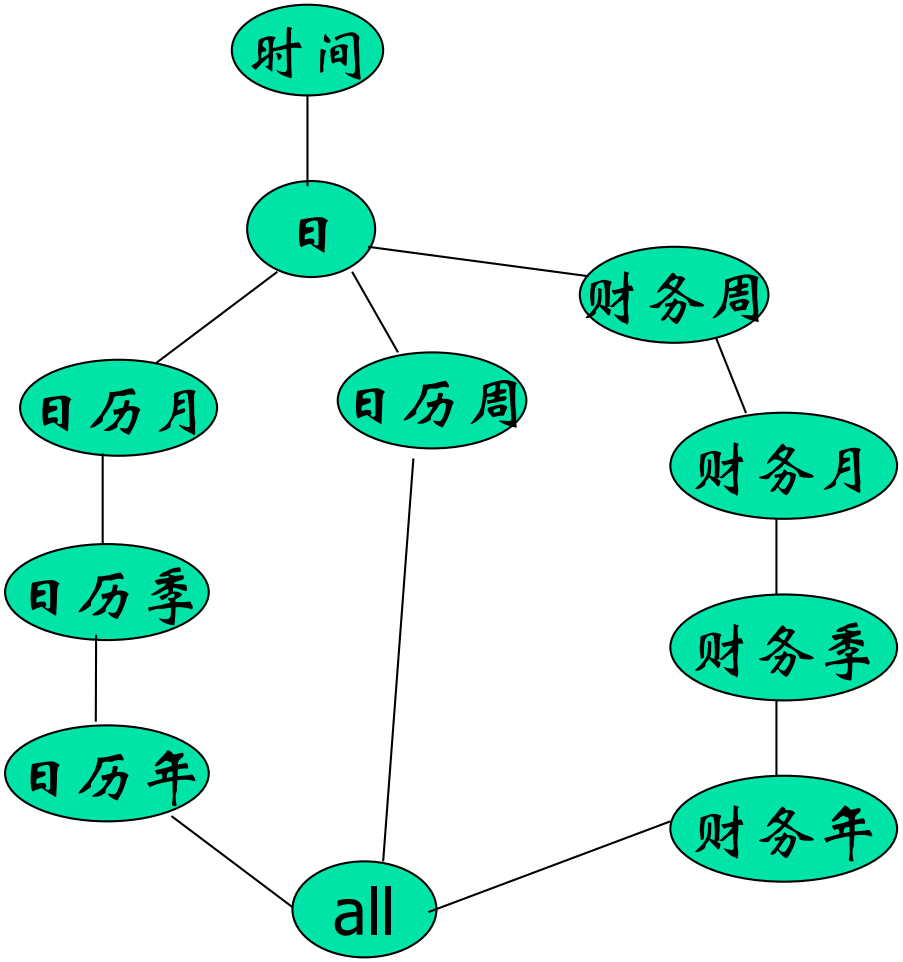
- 关系元组　　映射关系，函数
  - <日，月>　{ … }→{ … }
  - <月，季>　日集合→月集合
  - <季，年>　{20070101—20070131}→200701
  - <年，all>　Partial mapping

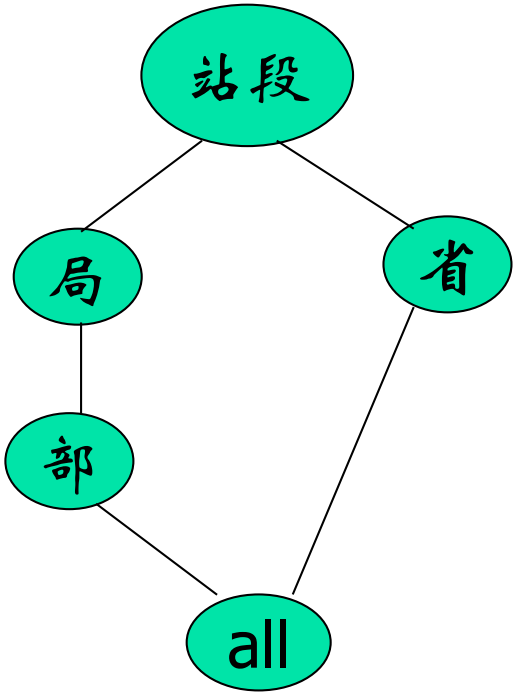


(3)代数格模型

- 为了能支持偏序结构，有许多文献和系统提出采用代数格模型。这是一种较为完备的模型。
- 这种模型的维层次结构为一个偏序，表达能力较强，也是最为常见的和应用得最多的模型。
- 已有数据仓库工具采用这种层次结构模型。各种不同的多层次链模型(如oracle系统中的多个hierarchies)在多数情况下也可以转换成这种代数格模型。

示例





(4) 特殊模型

- 针对一些特殊的应用模型，不同的研究者在一些研究论文中还提出了一些不同于以上结构的特殊模型。这些特殊模型的缺点是无法解决通用性问题，缺乏系统工具支持，每个主题的设计和实现都需要采用特例化的方式实现。
- 例如：某些行业的财务科目数据，就属于一种特殊的数据模型。这种模型将不同级别的科目数据做在一个维表中。
- 为此，只有进行特殊编码，或者是进行规范化处理

(5)一种更为复杂的模型

- 例如：
  - 在铁路客货票数据应用模型中，铁路组织结构中铁路局划分在地理上与省份的划分是交叉重叠的，实际应用需要将地域按(局，省)划分成更细的单位。
  - 再如在有些应用中，可能存在(月份，周)的时间划分方法。在这种情况下，以上数据模型都无法内在地支持这种应用模型。
- 这些实例模型本质上是一种单维多层次组合关键字模型。
- 为了系统化地支持这种模型，可以在常见的维格的基础上，对维层次结构模型进行扩展，得到一种维组合格结构，能有效地支持单维多层次组合关键字模型

2.2 维层次结构的物理表示

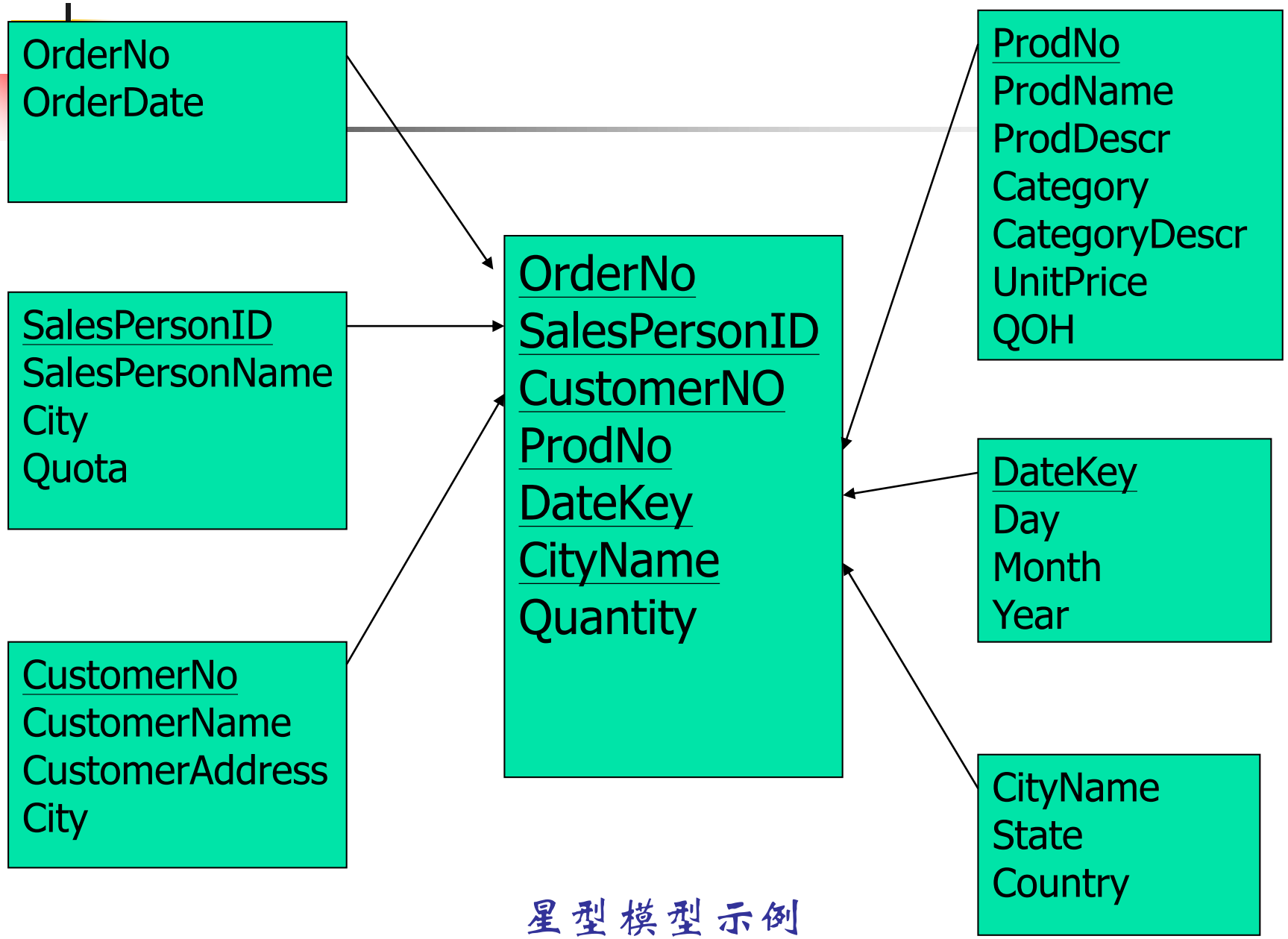
- 实现多维模型时首要的问题
  - 根据维的逻辑模型，给出各维的定义域。
- 在关系型系统中，一般以维表的形式给出各个维的维层次的定义域。例如
  - 商品表，站表，日期表等
- 在多维存储系统中，同样需要用适当的维表对维或维层次进行定义。例如：在sql server的hierarchy定义中，需要明确各个层次的定义。

星型与和雪花模型

- 我们可以不区分数据存储方式，将用于描述维逻辑结构所需的属性集及给定的值域称为维表。
- 根据维表的物理组织方式的不同，维层次的表示方法一般可分为
  - 星型模型，star schema
  - 雪花型模型，snowflake schema

星型模型

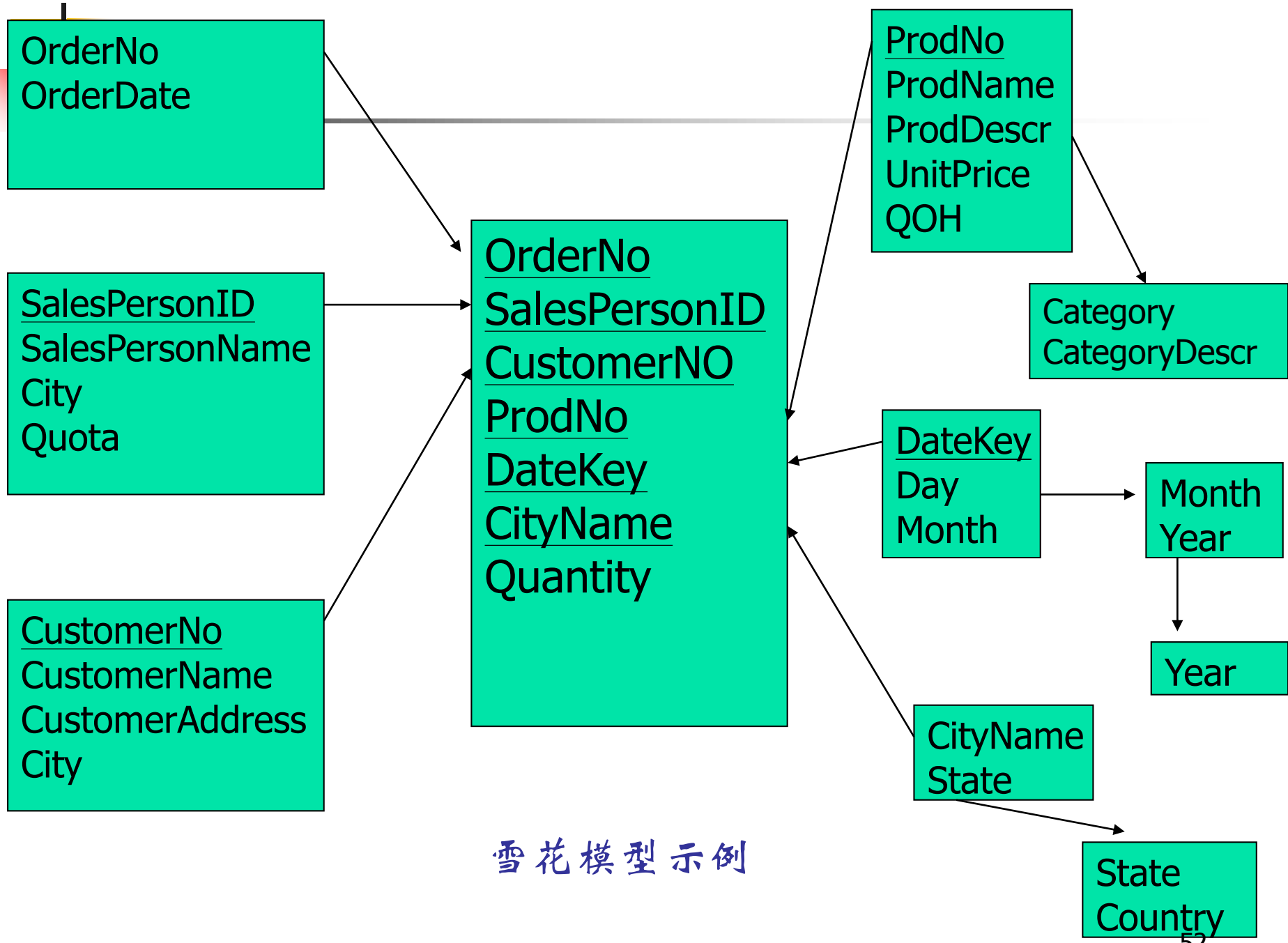
- 典型维层次模型
  - 用单张维表来表示维的层次结构；
  - 应用广泛，最为常见；
- 缺点
  - 在层次结构复杂时，很容易造成冗余。
  - 可能因维表设计不满足关系范式的要求，难以为高层维节点提供足够的描述信息。


OrderNo
OrderDate
SalesPersonID
SalesPersonName
City
Quota
CustomerNo
CustomerName
CustomerAddress
City
OrderNo
SalesPersonID
CustomerNO
ProdNo
DateKey
CityName
Quantity
ProdNo
ProdName
ProdDescr
Category
CategoryDescr
UnitPrice
QOH
DateKey
Day
Month
Year
CityName
State
Country

星型模型示例

雪花模型

- 另一种常见的变通方法
  - 采用满足关系范式的多张维表来表示一个维的层次结构；
  - 避免单个维表所带来的冗余


OrderNo
OrderDate
SalesPersonID
SalesPersonName
City
Quota
CustomerNo
CustomerName
CustomerAddress
City
OrderNo
SalesPersonID
CustomerNO
ProdNo
DateKey
CityName
Quantity
ProdNo
ProdName
ProdDescr
UnitPrice
QOH
Category
CategoryDescr
DateKey
Day
Month
Month
Year
Year
CityName
State
State
Country

雪花模型示例

星型模型和雪花模型比较

- 星型模式没有显式地支持属性层次结构，存在冗余，维层次信息查询效率高(不需要连接操作)。
- 通过对多维表进行规范化，雪花模式显式地表示出多维层次结构。实现维表维护方便。
- 星型模式中的多维表的非规范化的结构，对维的浏览而言，可能更为合适。
- 采用雪花模式对维数据的浏览略为麻烦一些。
- 雪花模型没有数据冗余问题。

2.3 多维空间及数据立方体模型

- 多维模型系统中最基本的操作
  - 从不同角度对不同的指标进行观察（分析）。
- 将观察角度构成的空间
  - 多维空间
  - 空间的每个维都有一个定义域。
- 多维模型的表示能力（表示与提供数据的能力）
  - 多维空间的数目，多维空间的性质
  - 取决于模型所采用的维结构模型和多维空间定义模型。

多维视图与数据立方体

- 每个多维空间及定义于该空间中的各点的指标值集构成一个多维视图。
- 多维视图的属性可以划分为两类
  - 维属性(多维空间中的点坐标分量)
  - 度量属性(指标属性)。
- 所有的多维视图构成一个数据立方体
  - data cube

5.2.3.1 Hypercube格模型

- Hypercube格模型
  - 维结构模型采用简单的不考虑维层次结构的模型。
- 例如
  - 有企业销售数据
    - Sales(ProductId, StoreId, TimeId, Sales)
  - 假设不考虑维的层次结构，则这个数据模型可以看成是由三个维属性
    - ProductId，SotreId，TimeId
  - 一个度量属性Sales构成。

示例续

- 如果需要从不同的角度对销售量数据进行分析。例如
  - 需要知道各种产品在各个店的销售情况，
  - 则在关系型系统中，一般需要执行如下的查询语句
  - SELECT ProductId, StoreId, SUM(Sales)
  - From Sales
  - GROUP BY ProductId, StoreId

示例续

- 考虑所有可能的组合情况，则总共有$2^3=8$种查询方式。
- 可用语句中的分组属性来表示不同的查询。如
  - (P, S, T)表示各产品在各店各时期的销售情况。
  - (P，S)则表示前面例子中的查询。
- 查询之间存在一定的计算关系。如
  - 假设上面的各产品在各店的销售情况已经计算并保存在数据库中，则关于各种产品的总销售情况的查询结果就可以根据(P, S)的内容计算得到。

视图

- 对于分组规则相同的查询
  - 如果条件不同，则得到结果也可能会不同。
  - 但这些结果具有相同的结构。
- 为了表示这些结构相同的查询，一般用抽象的视图来代表这些查询，视图的值为不带查询条件的查询结果。
- 例子
  - 有8个不同的视图，构成一个视图集，设用VS表示。

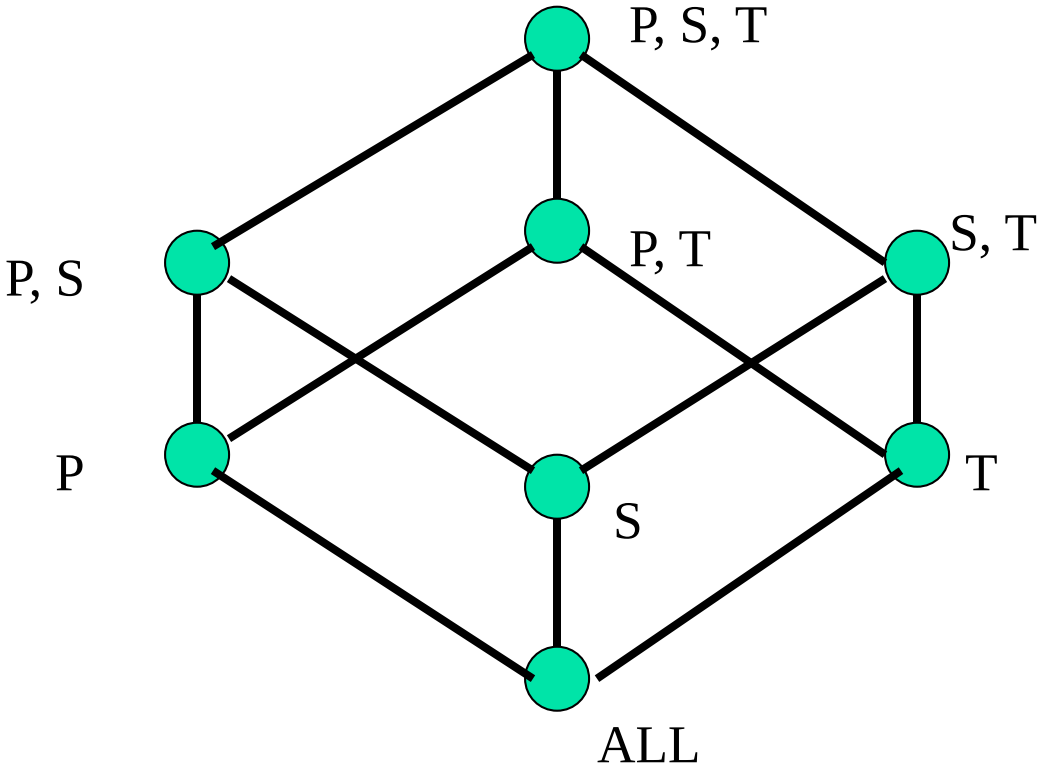
视图间关系

- 这些视图间存在一定的计算关系。
- 例如，如果已知视图(P, S, T)，
  - 则显然，可用(P, S, T)的结果计算(P, S)和(S, T)，
  - 也可以说，可以根据（P, S, T），生成(P, S)和(S, T)。
- 即视图之间存在计算关系，这种关系，如果从表面上看，是一种子集关系。
- 但实际上，不仅仅是子集关系，还存在一层聚集关系。

关系的表示

- 可以用≤表示这种关系。VS和≤构成一个代数系统〈VS, ≤〉。
- 而且可以证明〈VS, ≤〉是格。
- 这样的格被称为Hypercube Lattice，可称为超级立方体格。
- 例子中的Hypercube格可以用Hasse图表示如后图。

关系表示之例



2.3.2考虑维层次结构的多维模型

- Hypercube缺点
  - 过于简单，表达能力不够。
- 如果多维模型的维层次结构模型采用包含维层次结构的模型，则可以提高数据立方体的表达能力。
- 多数这样的立方体模型都可以表示为格的结构。

2.3.3组合复合格模型

- 一种基于维组合格的数据立方体格模型，称为组合复合格数据立方体模型。
- 该模型能兼容复合格和Hypercube格结构，同时通过数据立方体内在地、系统化地支持了维内多层次组合关键字多维视图。

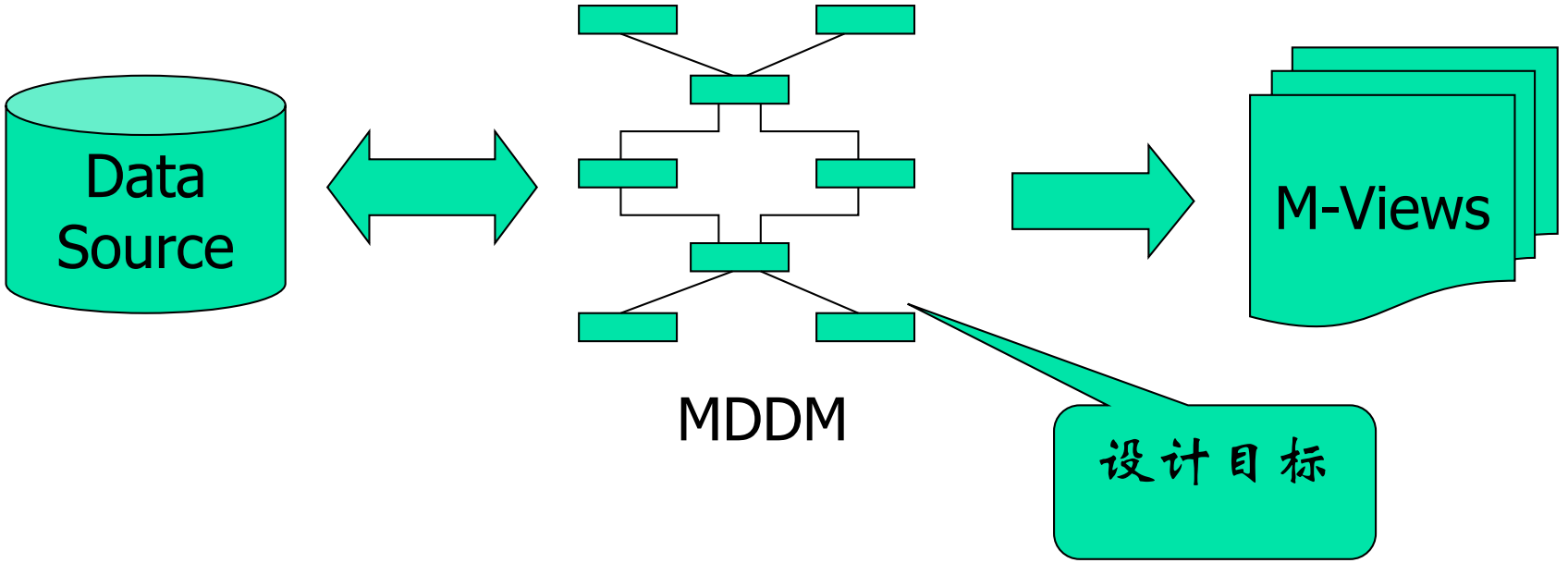
内容概要

- 5.1 引言
- 5.2 多维数据模型简介
- 5.3 一种多维数据模型的形式化描述
- 5.4 物化视图概述
- 5.6 小结

3 一种多维数据模型的形式化描述

- 基于维格的维层次模型是一种较为完备的数据模型。
- 本节将系统地给出维格及基于维格的数据立方体模型的一种形式化描述。

回忆:数据源、多维模型、多维视图



目标

- 设计一种数据表示模型
- 这样的模型应具有能力
  - 能够描述基本数据模型
  - 能够表示维的层次结构及其实例
  - 能够描述多维空间
  - 能够表示多维视图及其集合
  - 能够表示多维视图之间的逻辑关系
  - 能为多维视图的自动计算提供支持
  - 能为多维视图内容的自动维护提供支持
  - …

3.1 基本概念描述

- 基事实表——分析用基本数据表
  - Base fact table
    - 星型模型或雪花型模型中的事实表
    - 多维模型的核心数据表
- 基事实表的属性分为三类
  - 维属性
  - 度量属性
  - 其它属性

基事实表定义

- 基事实表定义为元组，(FN, DAS, MAS, OAS)，其中
  - FN为基事实表表名；
  - DAS为非空基事实表属性集，称为维属性集；
  - MAS为非空基事实表属性集，称为度量属性集；
  - OAS是基事实表其它属性集。
- 基事实表模型
  - Sales(ProductId, StoreId, TimeId, Sales)
  - DAS={ProductId, StoreId, TimeId}
  - MAS={Sales}
  - OAS={}

维格定义

假定允许维结构包括层次关系，且允许维层次结构为一个偏序

- 维层次结构模型是一个偏序集⟨DN, ≤⟩，其中
  - DN是非空有限维节点集。是定义于DN上的一个偏序，且DN中任意两个元素都有最小上界和最大下界。
  - 每个维节点d与一个属性集相关，标识为AS(d)，其中包含一个关键属性k(d)，关键属性的值域记为dom(d)，称为维节点d的定义域。
- 在偏序集⟨DN, ≤⟩中
  - 存在一个最小元素d_{min}=ALL，dom(ALL) = all
  - 存在一个最大元素d_{max}
  - 任取元素$d_i \in D$，有$d_i \le d_{max}$且$ALL \le d_i$。
- 维层次结构的定义符合代数格的定义，称其为维格，用DL表示。

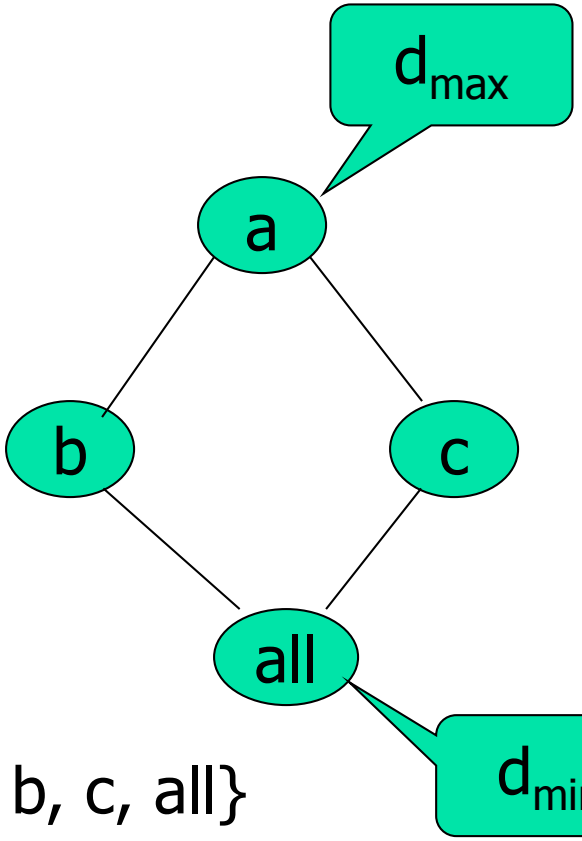
维元素关系表示

- 维格中层次节点之间的关系≤确定了两个维节点间相对的层次高低，如果在DN中，有d_j和d_i，而且$d_j \leq d_i$，则称d_j的层次高于d_i的层次。
- 因为维格是一个图结构，因此节点间的层次高低只是相对的概念。
  - d_{max}是最低层节点，d_{min}是最高层节点。另外，如果有$d_j \leq d_i$，则称d_i是d_j的祖先，反之，d_j是d_i的子孙。

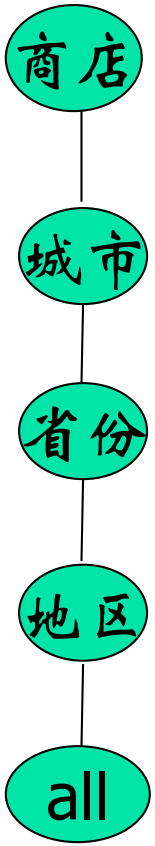
直接后继与维格表示

- 直接后继
  - 设有维格DL=⟨DN, ≤⟩，若有
    - di, dj∈DN，
    - 且 (dj≤di)∧(dj≠di)∧¬∃dk((dj≤dk)∧(dk≤di))，
  - 则记dj与di间的关系dj<di。
  - 且称di为dj的父节点，dj为di的直接后继或儿子。
- 为了用图的结构表示维格，可用一条指向直接后继的弧来表示维节点与其直接后继间的关系，这样就构成了维格的Hasse图表示。

示例



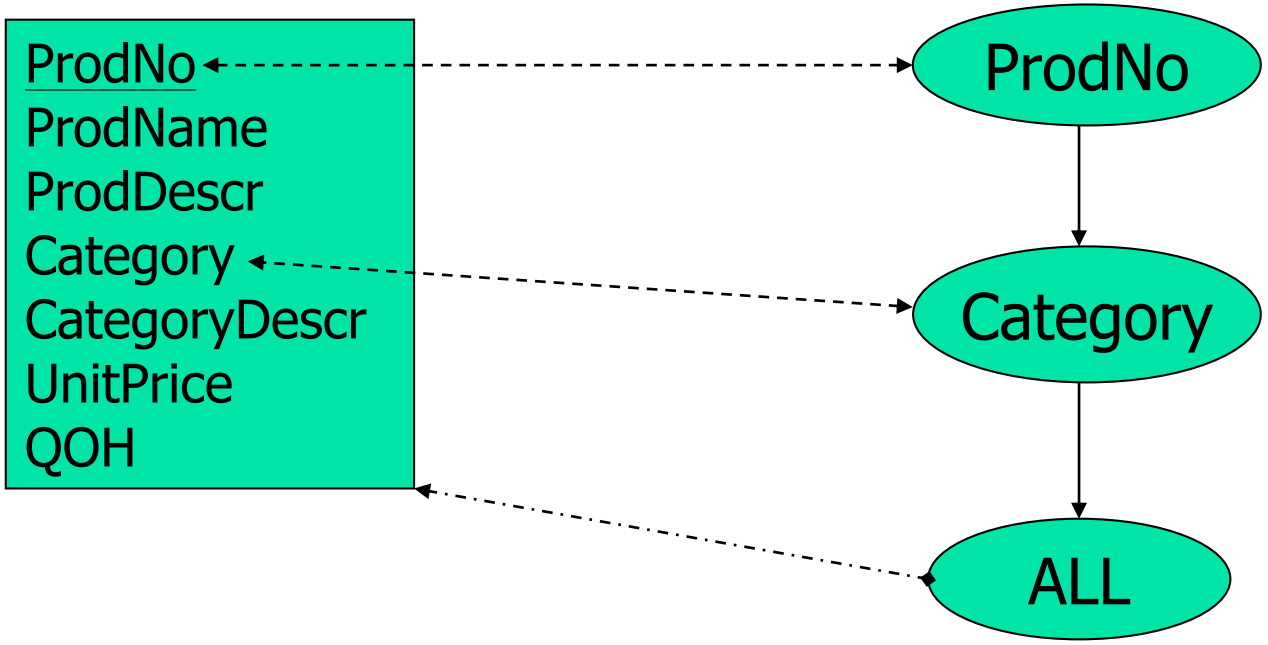
DN={a, b, c, all}



DN={商店, 城市, 省份, 地区, all}

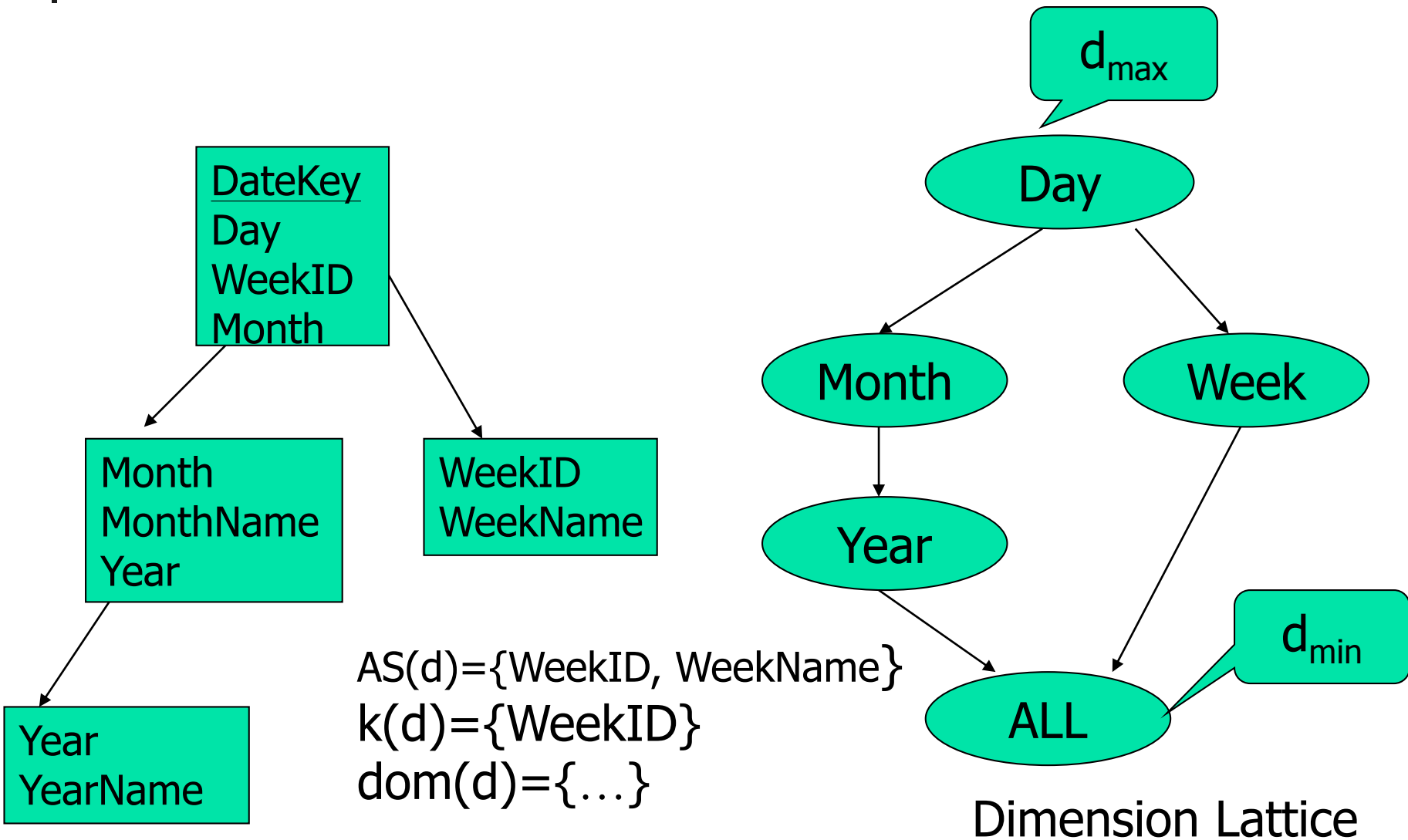
Dimension Lattice

示例-产品维格



Dimension Lattice

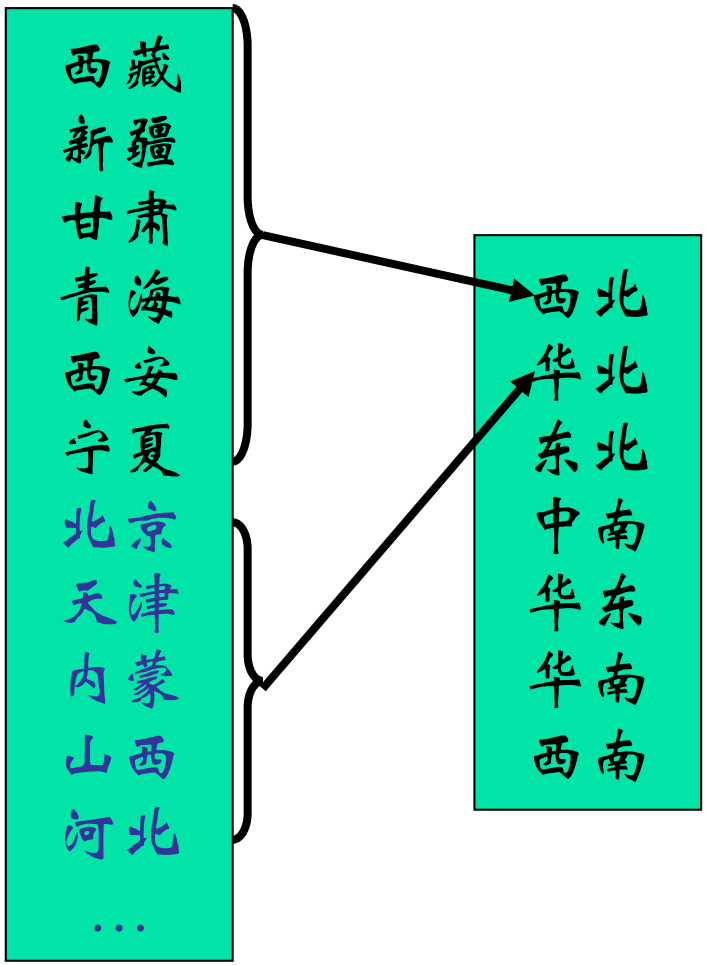
示例-日期维

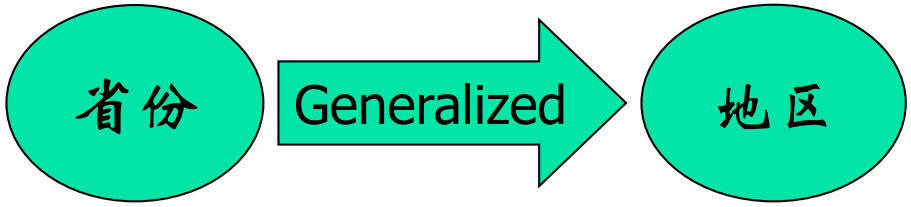

DateKey
Day
WeekID
Month
Month
MonthName
Year
WeekID
WeekName
Year
YearName
AS(d)={WeekID, WeekName}
k(d)={WeekID}
dom(d)={…}
dmax
Day
Month
Week
Year
ALL
dmin
Dimension Lattice

节点间关系的体现?

- 前面两个定义
  - 是维层次结构的逻辑定义
  - 对于结构中的任意两个存在关系的元素，它们之间存在抽象的关系≤
- 节点间的关系是如何体现的?
- 在维层次结构中，相对于两个存在关系≤的维层次节点而言，从低层次节点到高层次节点，实际上是一个概念泛化的过程。

泛化的例子


西藏
新疆
甘肃
青海
西安
宁夏
北京
天津
内蒙
山西
河北
...
西北
华北
东北
中南
华东
华南
西南


省份
Generalized
地区

再如

- 货物小类与货物大类
  - 若干个货物小类属于同一个货物大类
  - 即货物大类中包含几个货物小类。
  - 这相当于是货物小类集合到货物大类集合间存在一个多到一的部分映射。

卷起函数 Roll up function

- 这样的一个部分映射是卷起(Roll up)操作的基础
- 称这种部分映射为维节点间的卷起函数(Roll up Function)。
- 前例中
  - RF:{中国省份}→{中国地区}
  - 其中，若干省份总称为某个地区

Roll up Function

- 卷起函数的定义
  - 设有维格$DL=\langle DN, \leq \rangle$，任取DN中节点$d_i$和$d_j$，若有$d_j \leq d_i$，
    - 则规定两个节点的定义域间必须至少存在一个部分映射$RF_{ij}: dom(d_i) \rightarrow dom(d_j)$，且$ran(RF_{ij}) \subseteq dom(d_j)$，
      - 其中ran表示函数的值域
  - 则称RF_{ij}是d_i到d_j的卷起函数。

例如

省份ID	省份简称	地区ID
北京	京	华北
天津	津	华北
河北	冀	华北
山西	晋	华北
内蒙	蒙	华北
…	…	…

再如

商店ID	商店地址	城市ID	城市名称	省份ID	省份简称	地区	国家/地区
A	A地址	昌平	昌平区	北京市	京	华北	中国
…							
C	C地址	武汉	武汉市	湖北省	鄂	中南	中国
…							
E	E地址	厦门	厦门市	福建省	闽	华东	中国
…							
G	G地址	香港	香港市	香港	HK	华南	香港
H							
…							

相关定理

- 定理
  - 在同一维格中，任意两个存在祖孙关系的节点间都存在至少一个卷起函数。
- 例如
  - 城市→地区之间必然存在至少一个卷起函数
  - 所有西北各省的城市都属于西北地区
- 原因
  - 偏序是自反、传递、反对称的二元关系

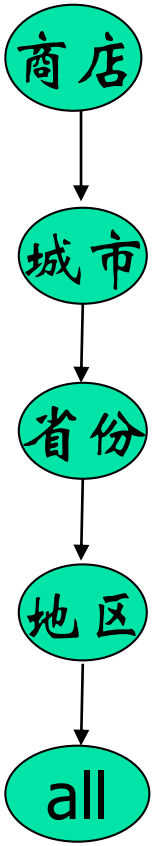
卷起函数集

- 卷起函数集
  - 设有维格DL=⟨DN, ≤⟩，对于所有的维节点d，求出其所有的直接后继$\{d_1, \ldots, d_m\}$，将d与所有直接后继间的卷起函数组织成一个集合RFS
  - 称RFS为维格DL的卷起函数集。
- 问题：
  - 维格中卷起函数集表示方法？
  - 一个表与多个表：
  - 星型与雪花型

卷起路径

- 为了表示节点间的域变换路径，可以定义如下的卷起路径。
- 卷起路径
  - 在维格的Hasse图中，任取节点d_i和d_j，若$d_j \leq d_i$，则至少存在一条从d_i到d_j的简单路径RP_{ij}，称其为卷起路径。

例如



$RP_{商店→省份}$：（商店，城市，省份）
$RP_{城市→all}$：（城市，省份，地区，all）
$RP_{地区→地区}$：（地区，地区）

卷起路径有什么用？
假设有商店销售数据，如何自动计算各地区销售情况？

维格实例

- 维格层次结构：
  - 确定了结构中各节点的逻辑关系
  - 给定各层次节点的定义域后，再给定RFS则明确了节点间的映射关系(卷起函数—关系的具体化)
  - 从而以实例化的形式得到与维格的逻辑结构相对应的实例(instance)。
- 可以称这样的一个实例为维格实例。

维格实例的定义

- 维格实例：
  - 给定维格DL，并确定维格的卷起函数集RFS，就确定了一个维格实例，可以表示为元组(DL, RFS)。
- Instance of a dimension lattice
- 逻辑层次结构相同的维格，给定不同性质的维结点定义，给定不同卷起函数，则将成为另外一个维格实例

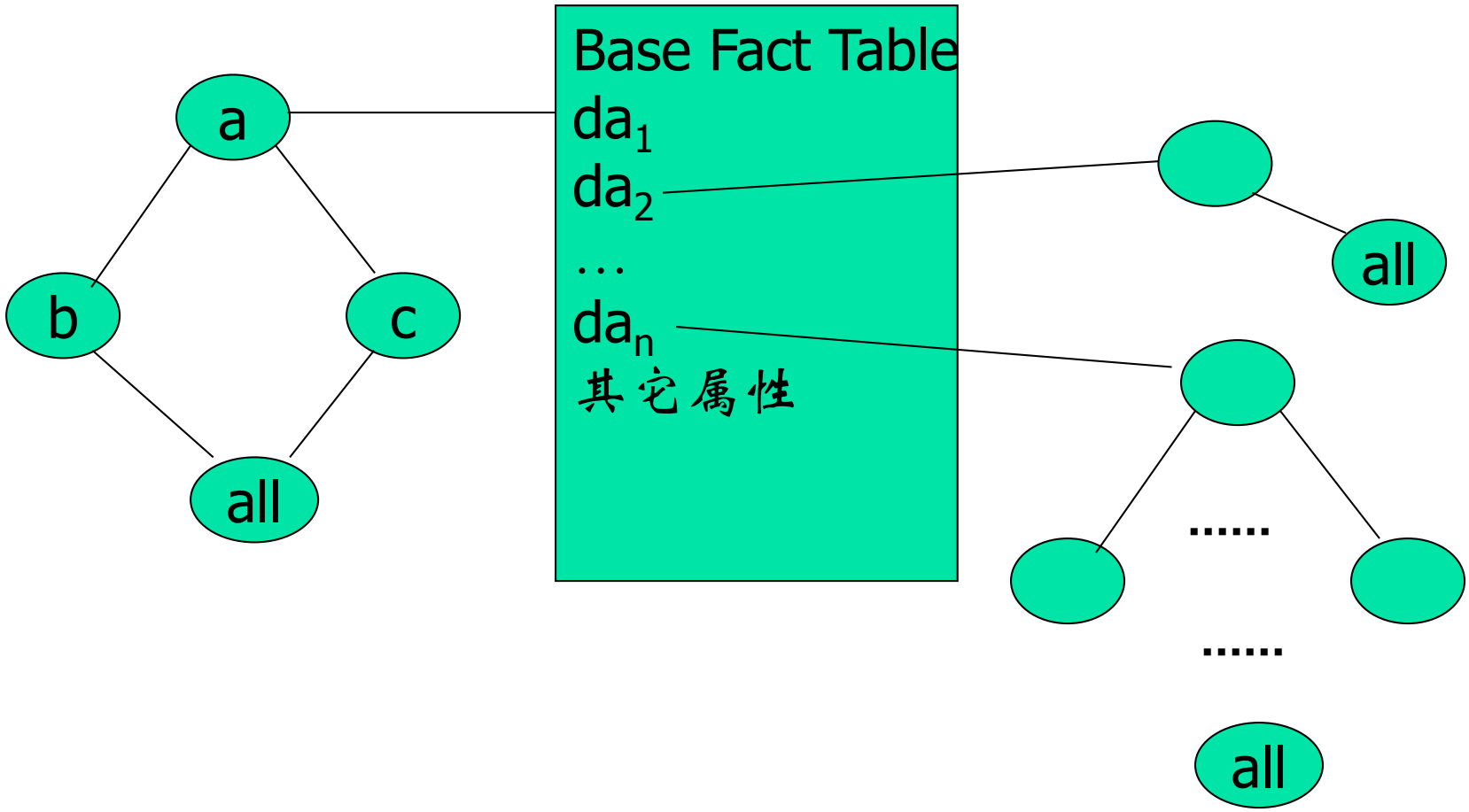
维的定义

- 维
  - 设有基事实表F=(FN, DAS, MAS, OAS)
  - 基事实表维定义为元组(DLI, da)，其中
    - da∈DAS，是F的一个维属性
    - DLI为定义于da上的维格。
  - 即基事实表的一个维属性和定义于其上的维层次结构相关联，构成基事实表的一个维。

维格与事实表的关联方式

- 给定一个基事实表维(DLI, da)，一般情况下，DL和da的关联方式为
  - 令DL的最大元素d_{max}的关键属性为da的外键，即dom(da)$\subseteq$ dom(d_{max})。
  - 例如，前面的例中表示的维格与Sales中的TimeId相关联，可以得到日期维。
  - 地区维格实例与商店ID相关联即得到地区维，其中，任取sales中的商店ID$\in$dom(商店)，sales中的商店ID和d_{max}的关键字商店ID存在外键关系

事实表、维



如何形式表示观察角度和层次？

- 多维分析的一个最基本的操作：
  - 从任意角度和层次观察基事实表数据。
- 在多维模型中，需要采用适当的模型来表示给定的观察所涉及的角度和所处的层次。
- 如何表示呢？

维节点向量

- 可以给出如下的维节点向量的概念。
- 维节点向量或维结点组合
  - 设在基事实表F上定义了n个维，并用DS={D_1, D_2, …, D_n}表示这n维的集合。
  - 从DS中的每个维中各取一个维层次节点，构成一个维层次节点的有序元组DG=(d_{x1}, d_{x2}, …, d_{xn})，
    - d_{xi}是D_i维的维格中某一层次节点，称这样的元组为DS的维节点向量。
- 维节点向量用于表示...?

维节点向量表示观察

- 维节点向量规定了观察所涉及的维
  - 当d_{xi}为ALL时，可以认为观察不涉及维D_i
- 向量中的每一个分量规定了对应维的观察层次。
- 因此
  - 维节点向量可用于标识一个针对基事实表的观察。

多维空间

- 因为向量中的每一个分量属于不同的维，并且与一个域相关；
- 因此，每一个维节点向量定义了一个多维空间。
- 空间的点集
  - 维节点向量的各分量的域的笛卡尔积。
- 表示：
  - 用V(DG)表示由DG所决定的多维空间。

多维空间之例

- 例如，对于某销售模型，设DS={产品维，销售主体维，日期维}
  - 设有DG=(产品小类，销售地区，月份)，DG即为DS的一个维节点向量。
  - V(DG)表示了由所有产品小类、销售地区、月份构成的一个三维空间。

问题

- 多维空间中能看到什么呢？有什么指标？
  - 总销售量
  - 总销售额
  - 平均销售额
  - …

度量指标

- 在多维模型中，还需要明确观察的指标，即多维空间中的度量值measure
- 度量值的来源：
  - 基事实表中度量属性
- 在对应的多维空间中，需要针对基事实表的度量属性值，采用适当的函数进行运算，得到的多维空间中的量。

多维空间与度量

- 基事实表的各个维属性及取值范围，确定了一个多维空间，表示为V(F)。
- 对于一次观察而言，同样也有对应的多维空间V(DG)。
- 因此，要得到观察的结果，必须确定：
  - V(F)→V(DG)的空间变换函数。
  - 有哪些空间变换函数？

分组与聚集函数

- 实际上，从各维维格最大节点到V(DG)对应维的节点间的卷起函数正好实现这种空间变换功能。这种变换即为常见的分组聚集操作中的分组操作。
- 对分组结果再进行聚集操作的普通的运算函数包括：
  - 求和、求最大值、求最小值、求平均值和元组计数。
- 这种函数称为聚集函数。

多维模型中的聚集项

- 在多维模型中，可以将观察中涉及的指标量称为如下定义的聚集项。
- 聚集项定义
  - 给定聚集函数f，基事实表F的一个度量属性m，给定一个多维空间V(DG)；
  - 将F的事实空间中的不同点投影到空间V(DG)中，
  - 再对映射到V(DG)中同一点的事实表中各元组的m列值用f进行聚集
  - 得到V(DG)中的一个点的度量值。
- 将这样的度量值称为从基表空间到空间V(DG)的一个聚集项，表示为f(m)。

例

- 例如，给定DG=(产品小类，销售地区，月份)：
  - 根据DG对基事实表元组进行分组
  - 再对同组元组的销售量属性用聚集函数SUM进行聚集操作
  - 得到的聚集项为每个月各产品小类在每个销售地区的总销售额。
- 一个空间中只能有一个聚集项吗？

聚集项集

- 同时，为了保持不同空间之间度量的一致性，以便于系统性的数据维护和访问，可以要求为不同空间定义相同数目的结构相同的聚集项。
- 我们将这些定义于不同空间上的结构和数目一致的聚集项统称为聚集项集AIS。

聚集项的值域

- 在不同的多维空间上定义的相同结构的聚集项的类型是相同的
- 但可能具有不同的值域
- 例如：不同空间中销售额

3.2 多维模型

- 有了基事实表以及定义于其上的维集合和聚集项集合，就构成了一个多维数据模型。
- 多维数据模型
  - 多维数据模型MDDM是一个三元组(F, DS, AIS)
  - 其中
    - F为基事实表
    - DS为定义于基事实表上的维集合
    - AIS是定义于基事实表和DS上的聚集项集合。

说明

- 模型中
  - F为给定的基事实表
  - DS确定了定义在F上的各个多维分析的维以及各维层次节点间域空间变换规则
  - AIS确定了各个多维空间中的分析指标。
- 多维模型是用于表示和描述数据用的，多维模型数据表示机制是什么？

多维视图

- 访问数据：对多维模型的观察
  - 可以将对多维模型的观察称为查询。
- 给定多维模型的一个多维空间，可以在这个空间上进行包含不同条件、不同指标的查询。
- 如前所述，为了统一标识这些查询，也为了能为回答这些多维空间中的相同查询建立有效的支持机制，在多维模型中，一般采用多维视图的概念。
- 如何描述视图？

多维视图定义

- 设有多维模型MDDM，多维视图定义为：
  - MDV=(MDDM，DG)
  - 其中DG是DS的一个维节点向量。
  - 若DG=(d_{x1}, d_{x2},…,d_{xn})， AIS={a_1, a_2, …, a_m}，规定MDV的模式为($k(d_{x1})$, $k(d_{x2})$, …, $k(d_{xn})$, m_1, m_2, …, m_m)，$k(d_{xi})(i=1..n)$是D_i维维节点d_{xi}的关键属性。
  - 称{ $k(d_{x1})$, $k(d_{x2})$, …, $k(d_{xn})$}为视图的维属性集，$m_i(i=1..m)$是AIS中的对应聚集项a_i在空间V(DG)下的对应属性列。

定义解释

- MDDM给定了视图结果值的计算环境：
  - 基事实表F
  - 定义于F上的维集DS
  - 定义于所有多维空间上的聚集指标
- DG确定了视图所对应的多维空间。
- DS中各维顶端节点到d_{xi}的卷起路径的集合及相应的卷起函数集确定了：
  - 从基事实表F的事实空间到视图MDV的数据空间V(DG)的空间变换规则

定义解释续

- 而聚集项集AIS中的各个聚集项的聚集函数则指明了：
  - 每个F空间的度量属性值到V(DG)中的对应聚集项值的计算规则。
- 因此，该定义所描述的多维视图能为所有基于该空间的多维查询提供聚集数据。

多维视图的数目

- 给定多维模型MDDM=(F, DS, AIS)，设DS={D1, D 2, …, Dn}，则可能存在的不同的维节点向量的数目为：
  - $Num=\Pi_{i=1..n}|DN_i|$
  - 即
    - MDDM定义了Num个多维空间，也就是Num个多维视图。
    - 除了基事实表以外，多维模型可以向用户提供的数据还包括所有的这些多维视图。
- 问题：用什么表示这些数据的集合?

Data Cube及其定义

- 数据立方体(Data Cube)就是用来表示多维模型可以为应用提供的由不同角度、不同层次数据组成的数据集合。
- 数据立方体
  - 设有多维模型MDDM，数据立方体定义为二元组
    - C=(MDDM, MDVS)，其中MDVS是定义于MDDM上的多维视图的集合。
- 即数据立方体是定于某一主题的多维数据模型上的多维视图的集合体。

解释

- 定义适用于所有的数据模型。
  - 如果MDDM中的DS中的各维不考虑维的层次结构，则MDVS中所具有的多维视图的数目为$2^{|DS|}$。即表明该立方体是一个Hypercube。
  - 如果各维是其它的结构模型，如单链模型，维格模型或多层次链模型，所对应的数据立方体则具有不同结构的多维视图集。
  - 简单的多维模型、单链模型可以看成是维格模型的特例，而多层次链模型则在多数情况下可以转化为维格模型。
  - 因此，只要支持维格模型，则可以支持多数数据模型。

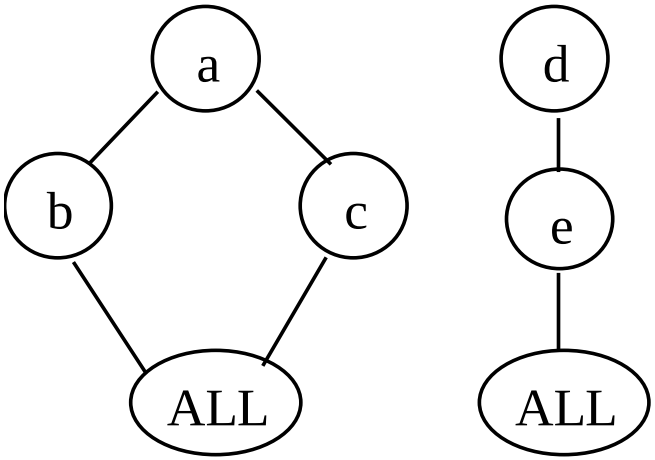
视图间关系与立方体格

- 在数据立方体的多维视图间，可以定义如下关系$\leq_1$：
- 视图间祖孙关系
  - 设有数据立方体C=(MDDM, MDVS)，设有
    - 视图$v_1=(d_{x1}, d_{x2},\ldots,d_{xn})\in$MDVS
    - $v_2=(d_{y1}, d_{y2},\ldots,d_{yn})\in$MDVS
  - 若对所有的i(i=1…n)，有$d_{yi}\leq d_{xi}$，
  - 则称v1是v2的祖先，v2是v1的子孙。记为v2≤1v1。
- 定理：代数系统$\langle$MDVS, $\leq_1\rangle$是格。

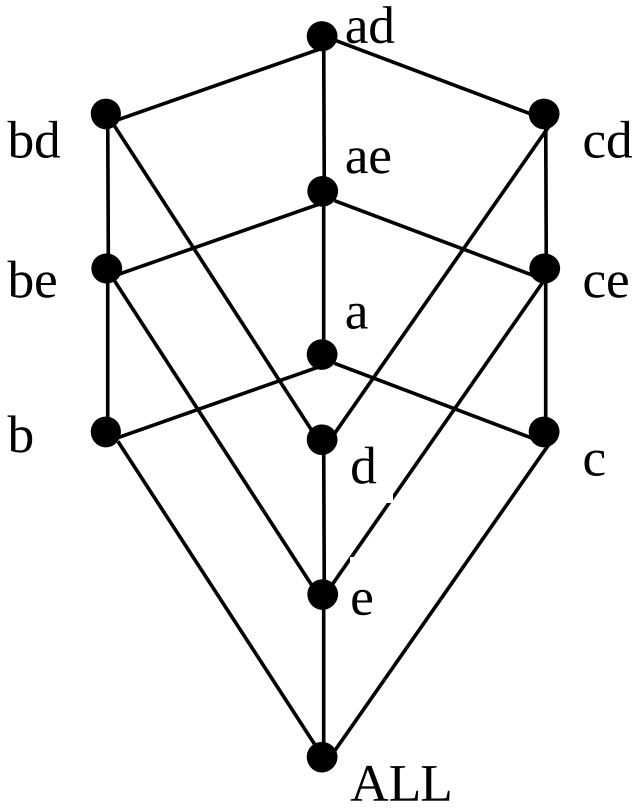
关系

- 视图间父子关系
  - 设有数据立方体C=(MDDM, MDVS)，在多维视图集MDVS 中，若有视图vj，vi，满足
    - $(v_j \leq_1 v_i) \wedge (v_j \neq v_i) \wedge \neg \exists v_k((v_j \leq_1 v_k) \wedge (v_k \leq_1 v_i))$
  - 则记v_j与v_i间的关系为$v_j <_1 v_i$。
  - 且称v_i为v_j的父节点，v_j为v_i的直接后继或儿子。
- 我们称立方体格中的最大元素为顶端视图，最小元素为底端视图。显然，顶端视图是最细节层的视图。底端视图是最抽象层的视图。

例



维层次结构



立方体格

例子说明

- 图的左边是两个维格例子，右边的图是由这两个维格生成的立方体格的Hasse图，
- 图中标记为ae的节点表示由DG=(a, e)所决定的视图，图中省略了ALL的表示，即节点b表示DG={b, ALL}的视图，其它类推。ae是顶端视图。ALL 表示DG={ALL, ALL}的视图，是立方体格中的底端视图。

计算关系及定理

- 在数据立方体中，对于一个视图v=(d_{x1}, d_{x2},…,d_{xn})，它的属性包括各个d_{xi}所对应关键属性列，以及由AIS所对应的指标集。对于v而言，它值集是从基事实表中经过多维聚集而得来的，也就是立方体中的所有视图的数据都可以从基事实表计算得到。
- 定理：
  - 设有数据立方体C=(MDDM, MDVS)，如果有v_1, $v_2 \in$MDVS，且$v_2 \leq_1 v_1$，则可以根据v_1的值计算得到v_2的值。

数据关系

- 从定理可知，如果数据立方体中的视图间存在祖孙关系，则它们之间存在数据导出关系。
- 也是就是说子孙的值集可以依赖于祖先的值集得到。
- 所以立方体中多维视图间的关系$\leq_1$也可以看成是数据导出关系或数据生成依赖关系。

其它更复杂的模型

- 组合复合格模型
- 分布式多维模型
- 其它模型

多维模型的典型案例

- 用户指定一个空间，指定指标，给定条件，多维模型能为用户自动提供所需的数据
- 例如：
  - 今年前三季度华北地区某型号交换机销售总量如何？
  - 为什么今年华北地区销售增长迅速？各省增长情况如何？
  - 河北省卖得不错，看看河北各市各月销售情况？

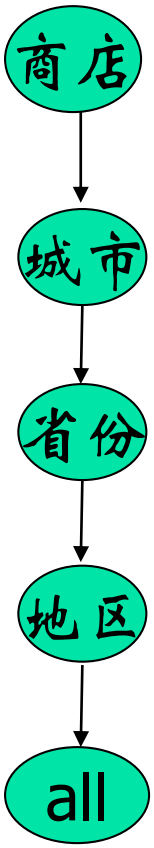
如何实现这些计算呢?

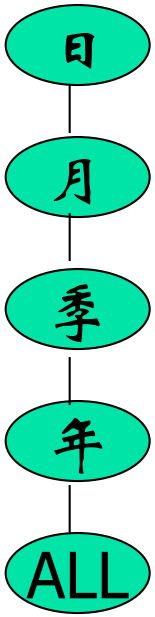
- 设有SalesCube=(MDDM, MDVS)
- MDDM=(Sales, DS, AIS)
- Sales={Sales, DAS, MAS, OAS}
  - DAS={商店ID, 产品ID, 日期ID}
  - MAS={销售量}
  - OAS={}

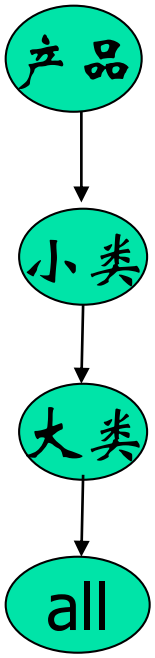
DS及AIS

- DS={产品维，地区维，日期维}
  - 产品维=(产品维格实例, Sales.产品ID)
  - 地区维=(地区维实例, Sales.商店ID)
  - 日期维=(日期维实例, Sales.日期ID)
- AIS=(总销售量，平均销售量)
  - 总销售量=sum(销售量)
  - 平均销售量=avg(销售量)

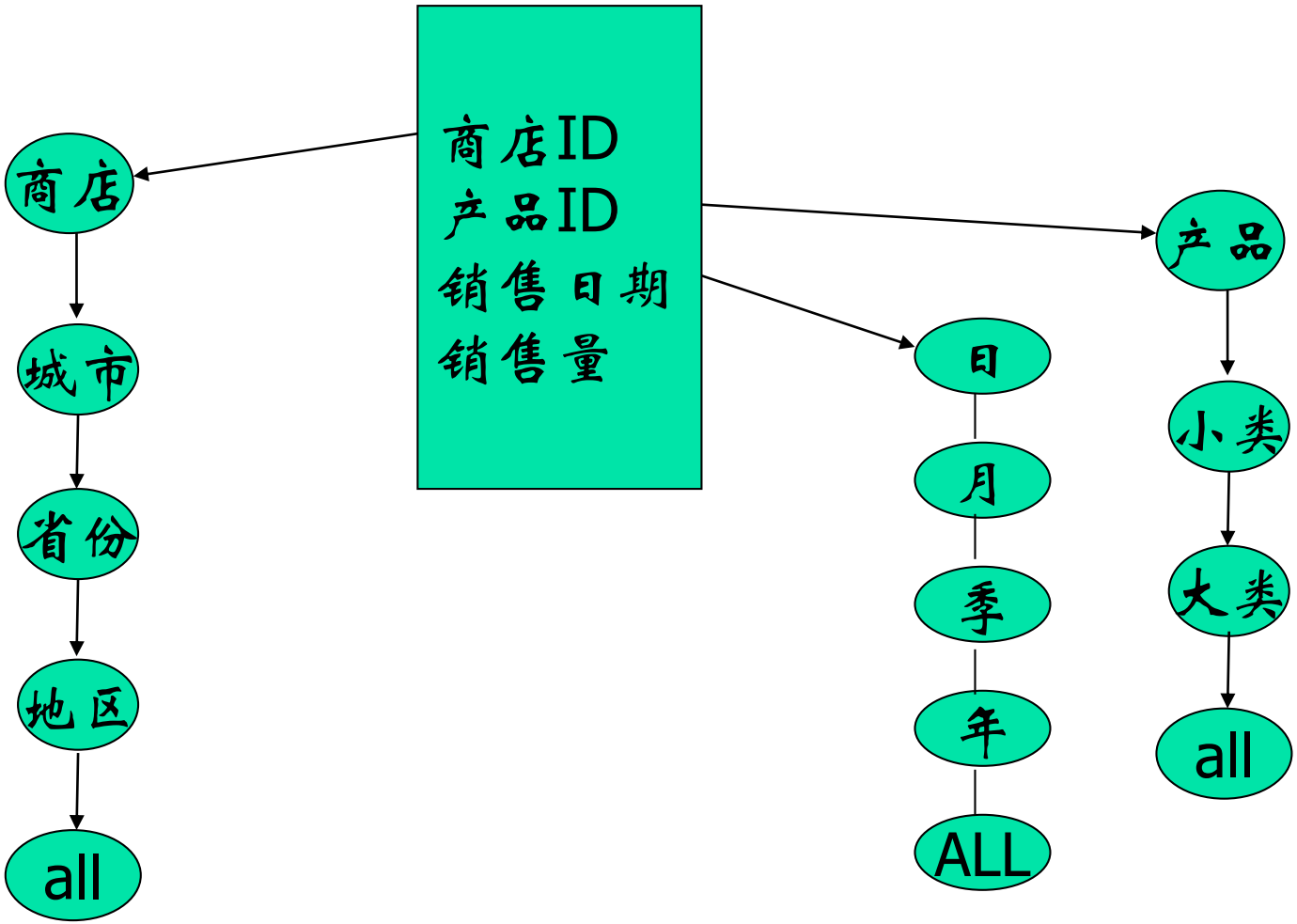
各维格实例







维格实例与基事实表的关联



MDVS

- MDVS=按照给定维结点向量定义所得到所有多维视图的集合
- 视图数：5*5*4=100个视图

问题与多维空间

- 今年前三季度华北地区某型号交换机销售总量如何？
  - DG=(地区，产品，季度)
- 为什么今年华北地区销售增长迅速？各省增长情况如何？
  - DG=(省份，产品，季度)
- 河北省卖得不错，看看河北各市各月销售情况？
  - DG=(城市，产品，月份)

第一问题的解答

- DG=(地区，产品，季度)
- 数据源：基本事实表=(商店ID，产品ID，日期ID)
- 计算方法：
  - 确定卷起路径
    - $RP_{商店,地区}$，$RP_{产品,产品}$，$RP_{日,季度}$
  - 确定卷起路径上的各个卷起函数
  - 得到空间变换规则

空间变换

- 通过卷起函数的复合，得到如下
  - $RF_{商店,地区}$
  - $RF_{产品,产品}$
  - $RF_{日,季度}$
- 利用这三个卷起函数实现如下空间变换
  - (商店，产品，日)→(地区，产品，季度)
- 利用这个变换实现分组操作

变换过程中限制条件

- 如下变换中
  - $RF_{商店,地区}$
  - $RF_{产品,产品}$
  - $RF_{日,季度}$
- 限制条件
  - 地区确定为华北地区
  - 产品确定某交换机
  - 季度为今年前三个季度

实际变换工作

- 将基事实表空间中符合限制条件的元组，按空间变换规则进行分组
- 属同一地区，同一交换机，同一季度的销售记录为同一组

聚集操作

- 将同一组中元组，针对销售量属性的值
  - 采用SUM函数计算销售总量
  - 采用AVG函数计算平均销售量
- 最终得到查询结果
- 其它问题的解答方法相同

存在的问题

- 效率问题
  - 如何提供效率？
  - 是否有必要都从基本事实表算起？
  - 如何实现这些机制？

解决办法

- 假设某个空间的所有可能值数据已经计算好了
- 则针对它的子空间查询可以利用该空间中的值进行，以提高效率

内容概要

- 1 引言
- 2 多维数据模型研究现状
- 3 一种多维数据模型的形式化描述
- 4 物化视图概述
- 5 小结

4 物化视图概述

- 为了提高数据查询效率，人们提出了一种重要的概念
  - materialized view
  - 物化视图或实体化视图。

4.1 概述

- 传统的数据库视图都是虚视图，即视图只有代码，但并没有对应的数据，数据都是在查询涉及到该视图时动态的生成的。
- 但是，对于数据查询来说，视图可以作为一个表的看待，在逻辑上与普通数据表并没有太大的区别。
- 虚视图存在效率问题

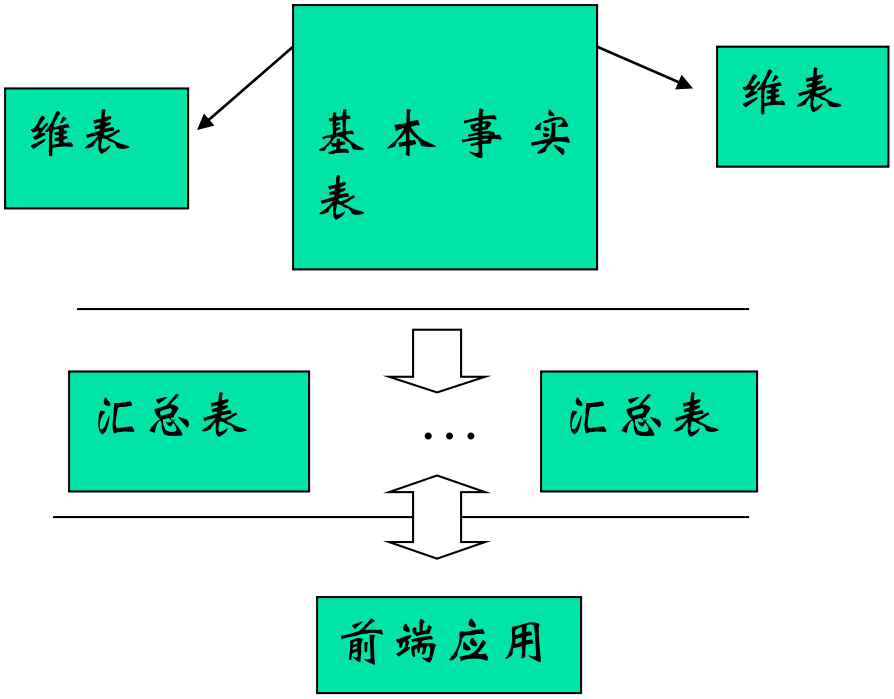
效率问题

- 虚视图数据生成的动态性
  - 即每次查询的时候都需要重新计算
  - 会导致效率的降低
- 特别是在数据仓库或大型数据库这种背景下，视图的数据源具有数据量大、数据源复杂、计算时间长的特点。
- 特别是如果所有查询都以基本事实表作为数据源的话，数据计算时间可能会过长。

解决办法

- 为了解决效率低的问题，人们采用各种方法来解决的这些问题
  - 预计算一些汇总数据表，作为查询的数据源，就是一种常见的办法。

示意图



引入MV

- 有些数据库系统也引入Materialized View的对象
- Materialized View在外观上像View
- 在实质上，它是普通数据表，但是具有附加的逻辑，可能可以实现一些有效的计算功能，并提供不同刷新机制。

举例

- Oracle 8i 的实体化视图(snapshot)
- 一般信息：name, schema, tablespace, 是否启用查询重写
- 刷新信息
  - 何时进行刷新：根据需要，每次提交时，自动
  - 刷新方法：PRIMARY KEY, ROWID
  - 刷新类型：FORCE, COMPLETE, FAST
  - 上次刷新日期：
  - 刷新组：

其它信息

- 存储信息：
  - 区信息，空间利用率信息，事务处理数量
- 选项：
  - 并行程度
  - 事件记录：是否生成事件日志
  - 是否将经常访问的数据放到cache
- 主信息：
  - 所有者，表，等

其它

- oracle materialized view 是在snapshot的基础上发展起来，很大程度上与以前snapshot并没有太大的区别。
- 对物化视图而言，最重要的问题就是如何刷新的问题。

4.2 物化视图设计问题

- 确定物化视图数据源，物化视图的数据刷新
- 将数据立方体的部分视图物化，以提高查询效率，并力图在查询效率、维护效率与空间开销上得到一个合适折衷。
- 因此，如何在多维模型的多维视图集中确定需要物化的视图，成为数据立方体设计的主要工作。

研究问题

- 目前，针对物化视图选择问题的研究从数据模型上大致可以分为两类
  - 一类是基于多维数据模型的数据立方体物化视图选择问题；
  - 另一类是各种非数据立方体模型(非模型化)的物化视图选择问题。
- 对于视图选择算法，首要的问题就是为数据模型中的多维视图设计适当的代价模型作为选择依据。因此，设计合理的代价模型，并为代价模型提供基本的参数，成为数据立方体设计关键。

内容概要

- 1 引言
- 2 多维数据模型研究现状
- 3 一种多维数据模型的形式化描述
- 4 物化视图概述
- 5 小结

小结

- 1 引言
- 2 多维数据模型研究现状
- 3 一种多维数据模型的形式化描述
- 4 物化视图概述

课后思考

- 1.多维数据模型的主要构成和功能.
- 2. 星型模型的结构与特点，举例说明具体的星型模型结构。
- 3. 雪花型模型的结构与特点，举例说明具体的雪花型模型结构。
- 4. 物理视图的功能特点。