第三章　数据仓库设计

- 数据仓库设计方法概述
  - DW设计与DB设计
  - DW设计的三级数据模型
  - 性能问题
  - 数据仓库中的元数据
- 数据仓库设计步骤
  - 概念模型设计
  - 逻辑模型设计
  - 物理模型设计
  - 数据仓库生成
  - 数据仓库的使用和维护

DW设计与DB设计方法比较

- 处理类型不同
  - DB：操作型数据环境，面向业务
  - DW：面向主题的分析型数据环境，面向分析，从基本主题开始，不断发展新主题
- 面向需求不同
  - DB： 一组较确定的应用(业务处理)需求;
    较确定的数据流
  - DW： 需求不确切(定);分析处理需求灵活;
    没有固定模式；用户对分析处理需求不甚明了；
    其设计很难以需求为基础

DW设计与DB设计方法比较（续Ⅰ）

- 设计目标不同
  - DB：事务处理的性能（OLTP），支持多用户并发访问，高效的增、删、改操作
  - DW：建立DSS的数据环境， 全局的分析环境，支持用户快速的分析和查询
- 数据来源不同
  - DB： 企业的业务流程中产生的数据
  - DW： 系统内部，主要从OLTP系统中获取，经过转换、重组、综合; 同时包括部分外部信息

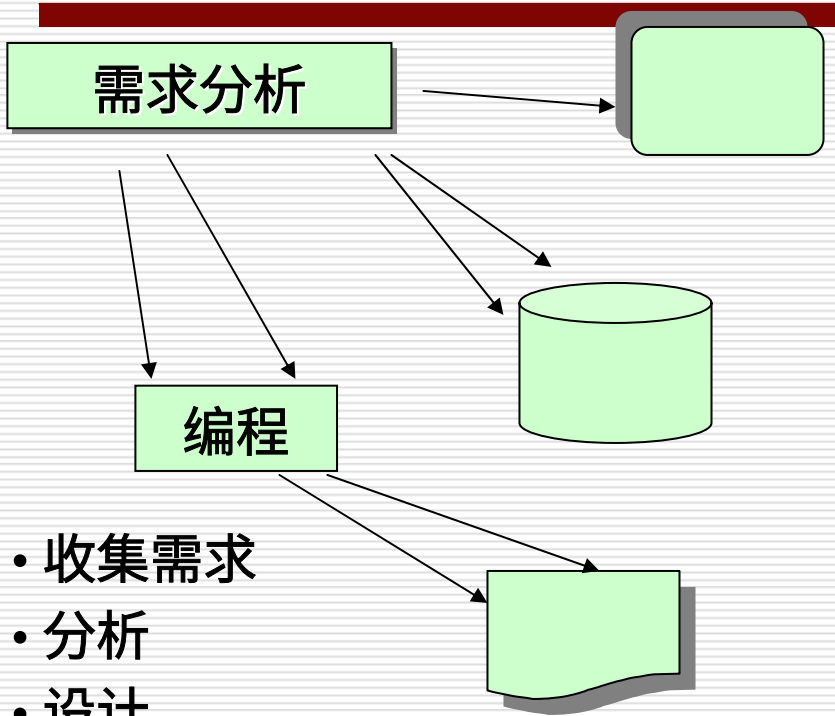
DW设计与DB设计方法比较（续II）

- 设计方法不同
  - **DB : SDLC(System Development Life Cycle)**

    — 应用需求驱动
  - **DW: CLDS**

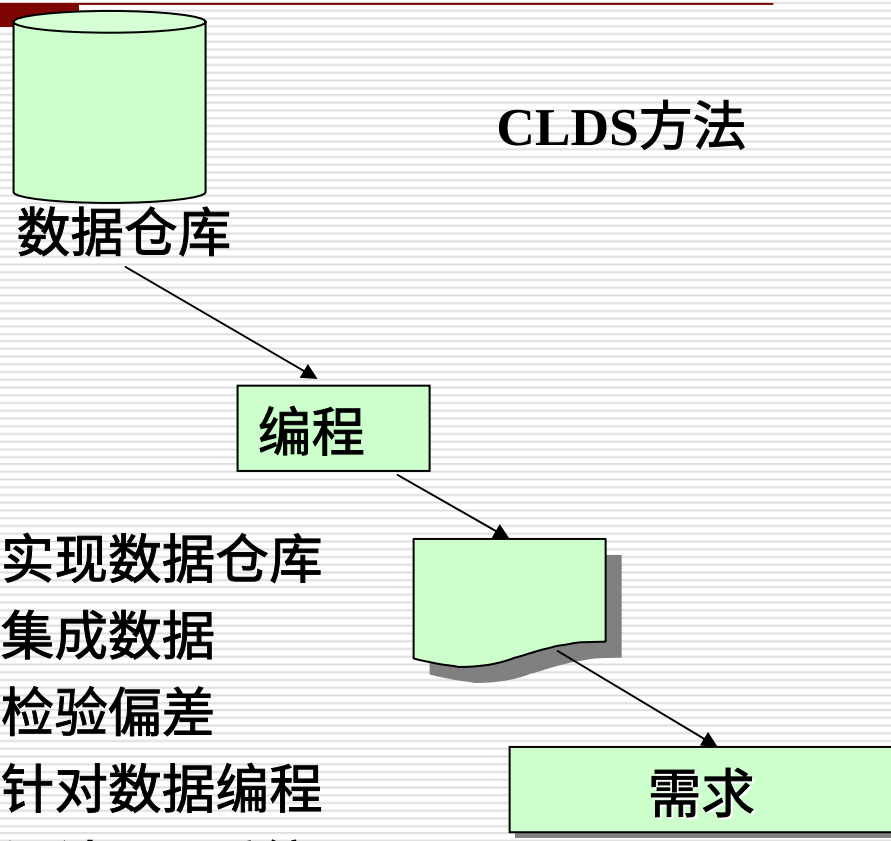
    — 数据驱动＋需求驱动

Inmon: SDLC与CLDS方法比较



- 收集需求
- 分析
- 设计
- 编程
- 测试
- 集成
- 实现

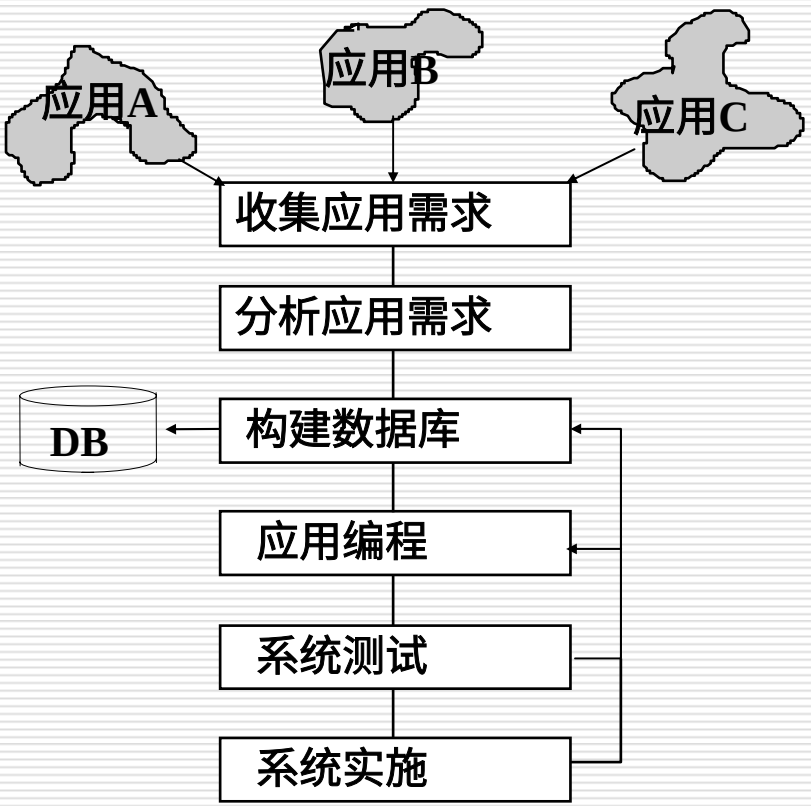
SDLC方法



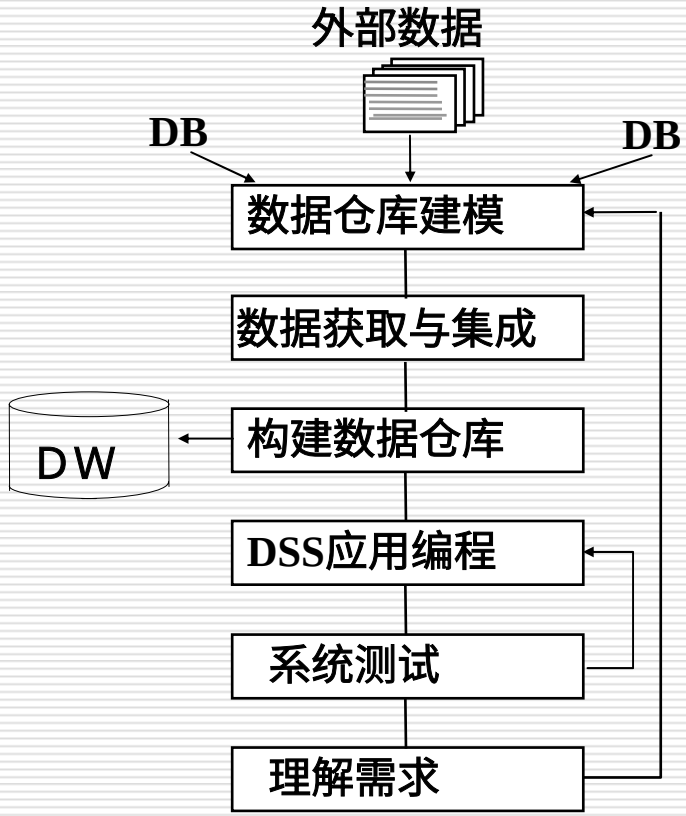
CLDS方法

- 实现数据仓库
- 集成数据
- 检验偏差
- 针对数据编程
- 设计DSS系统
- 分析结果
- 理解需求

SDLC与CLDS方法比较

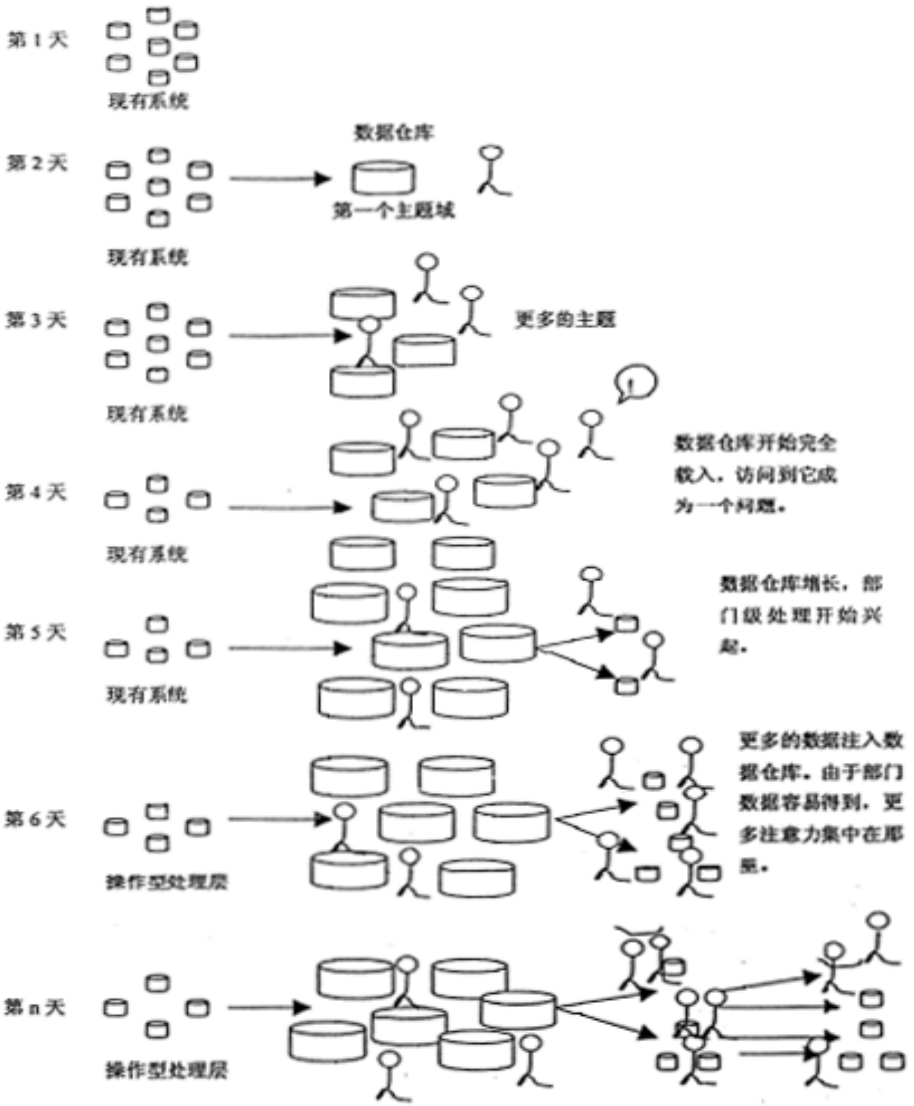


SDLC方法



CLDS方法

第1天到第n天现象


第1天
现有系统
数据仓库
第2天
第一个主题域
现有系统
第3天
更多的主题
现有系统
第4天
数据仓库开始完全
载入，访问到它成
为一个问题。
现有系统
第5天
数据仓库增长，部
门级处理开始兴
起。
现有系统
第6天
更多的数据注入数
据仓库，由于部门
数据容易得到，更
多注意力集中在那
里。
操作型处理层
第n天
操作型处理层

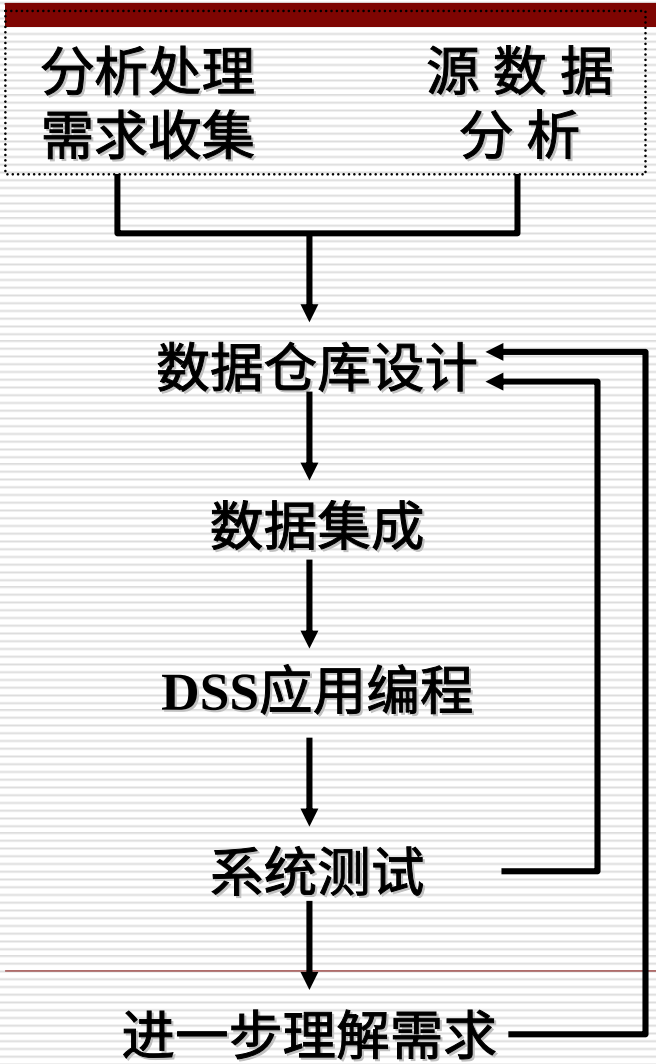
在数据仓库建设过程中明确需求

- 数据仓库建造过程中，如果开发者等完全明确需求之后开始工作，那么这个仓库永远建不起来
- 开发人员与DSS分析员的反馈循环十分重要

数据仓库设计的原则

- 坚持“以数据驱动为中心，数据驱动和需求驱动相结合”的原则。
- 数据驱动是指根据当前数据基础和质量等情况，进行数据源分析。
- 需求驱动是指根据业务方向性需求、业务问题等，确定系统范围和需求框架。

在实际工程中的设计方法



- 数据仓库的设计和实现是一项工程，是不断建立、发展和完善、循环求精的过程，并不是一个可以简单购买的产品。

在实际工程中的设计方法（续I）

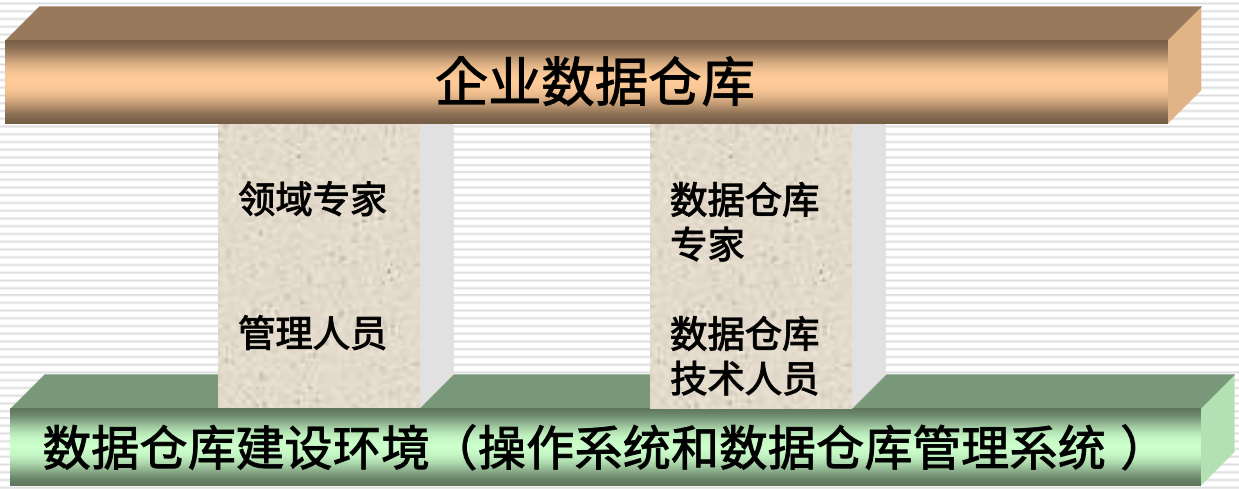
- 确定范围与项目定义
  - 主要任务
    - 系统边界的界定
    - 定义并描述项目
  - 步骤
    - 了解用户方向性需求，发现业务问题，确定范围；
    - 对业务问题进行排序，选择高优先级业务问题，界定系统边界；
    - 定义和识别项目的目的、范围、前景、价值、约束、风险、障碍等，制定质量管理、配置管理等计划，形成项目定义文档；
    - 确定主题域，建立概念模型。

在实际工程中的设计方法（续II）

- 应用系统及其数据的调研与分析
  - 目标
    - 为数据仓库系统发现运行稳定、数据可靠的源系统，并考察其数据状况
  - 主要任务
    - 对系统边界内的应用系统进行调研和分析，制定高层应用系统流程图，识别所有主要的应用系统及其主要内容
    - 分析主要应用系统的数据，形成应用系统数据分析文档

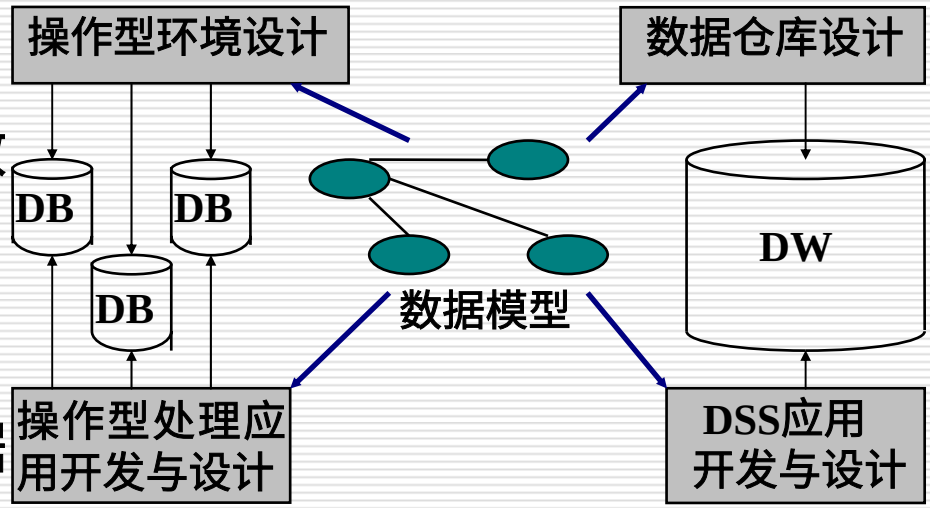
在实际工程中的设计方法（续III）

- 坚持“统一规划、分步实施、步步见效、逐步完善”的原则
- 开发模型：演化模型（快速原型法）
- 企业数据仓库的建设需要领域专家和数据仓库技术专家之间的相互协作；



数据驱动系统设计方法的基本思路

- 从源数据出发，分析数据，为新应用（分析处理）所用
- 根据分析处理的特点重新考察数据间联系，重组数据
- 数据（结构）具有相对的稳定性，而处理变化很快，强调数据模型的作用，支持识别 DB 与 DW 中的数据的“ 共同性 ”

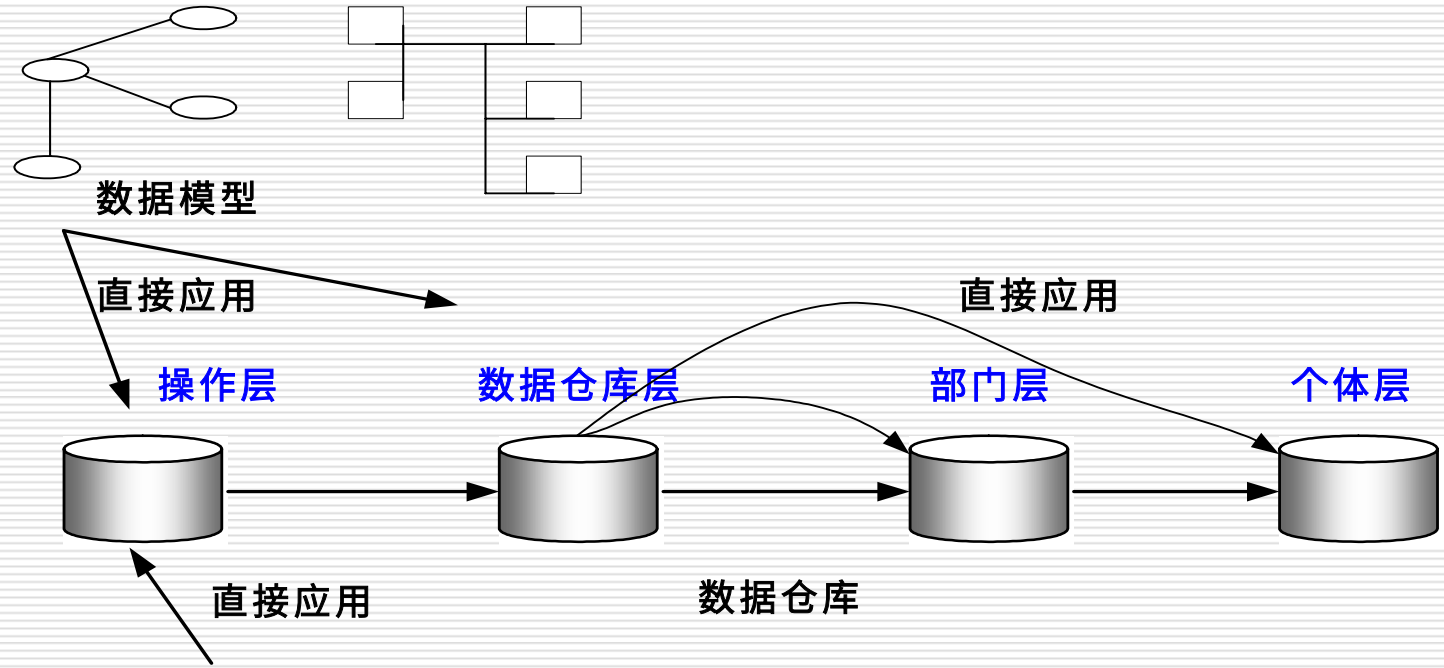


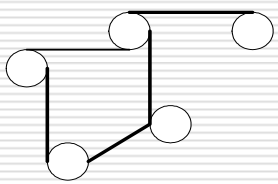
数据驱动系统设计方法的中心 — 数据模型

DW设计的三级数据模型

- 数据模型：
  - 对现实世界的抽象
  - 不同的抽象程度对应不同级别的数据模型
- **DW与DB的三级数据模型的区别：**
  - **DW的数据模型中扩充了码结构，包含时间元素**
  - **DW的数据模型中不包含纯操作型数据；但包含一些导出数据**

过程模型与数据模型

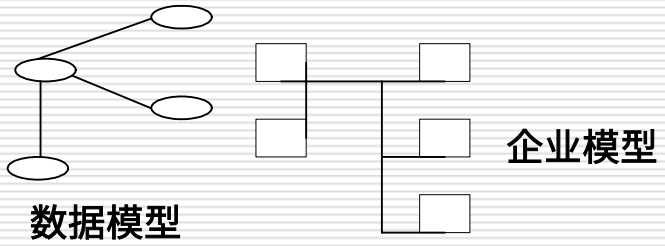


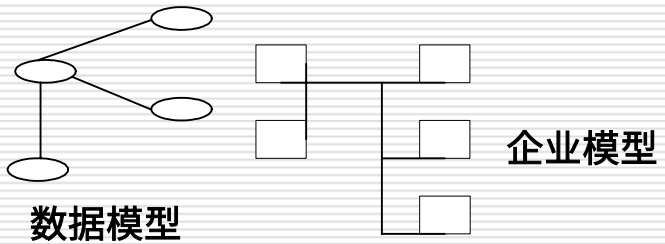


过程模型

- 过程模型仅使用于操作型环境
- 数据模型既可用于操作型环境，有可用于数据仓库环境

建模的不同层次间的关系

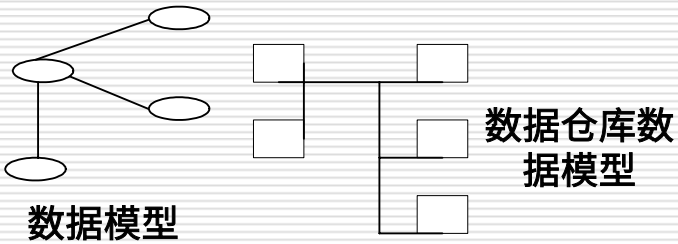




操作型环境

1、操作型数据模型等价于企业数据模型

2、数据库设计之前要加入性能因素



数据仓库

1、去掉纯操作型数据

2、关键字中加入时间元素

3、合适之处增加导出数据

4、创建人工关系

稳定性分析

零件表　　很少更改　　不时更改　　经常更改

零件ID
描述信息
主要替换件
库存量
订单单位
最低应达库存
主要供应商
订货到交货的时间
可以接受的废品率
加急
上次订单日期
上次订货量
上次发往地
发货清单
订货量
……

零件ID
描述信息
订单单位
可以接受的废品率
发货清单
……

零件ID
主要替换件
最低应达库存
主要供应商
加急
……

零件ID
库存量
上次订单日期
上次订货量
上次发往地
订货量
……

- 根据各个数据属性是否经常变化的特性将这些属性分组
- 按这些属性分组进行表的划分

DW设计的三级数据模型

- 概念模型：“信息世界”中的信息结构
  用E—R方法，以主题替代实体

- 逻辑模型：一般采用关系模型

- 物理模型：物理存储结构、存储方法
  如：建立数据分片、合并表，建立包括广义索引在内的各种索引机制，引入冗余，生成导出数据等

Inmon的三级数据模型

- 高级数据模型：

  采用E—R方法

- 中级数据模型：

  称为dis（Data Item Set）

  一个dis与E—R中的一个主题域（实体）对应

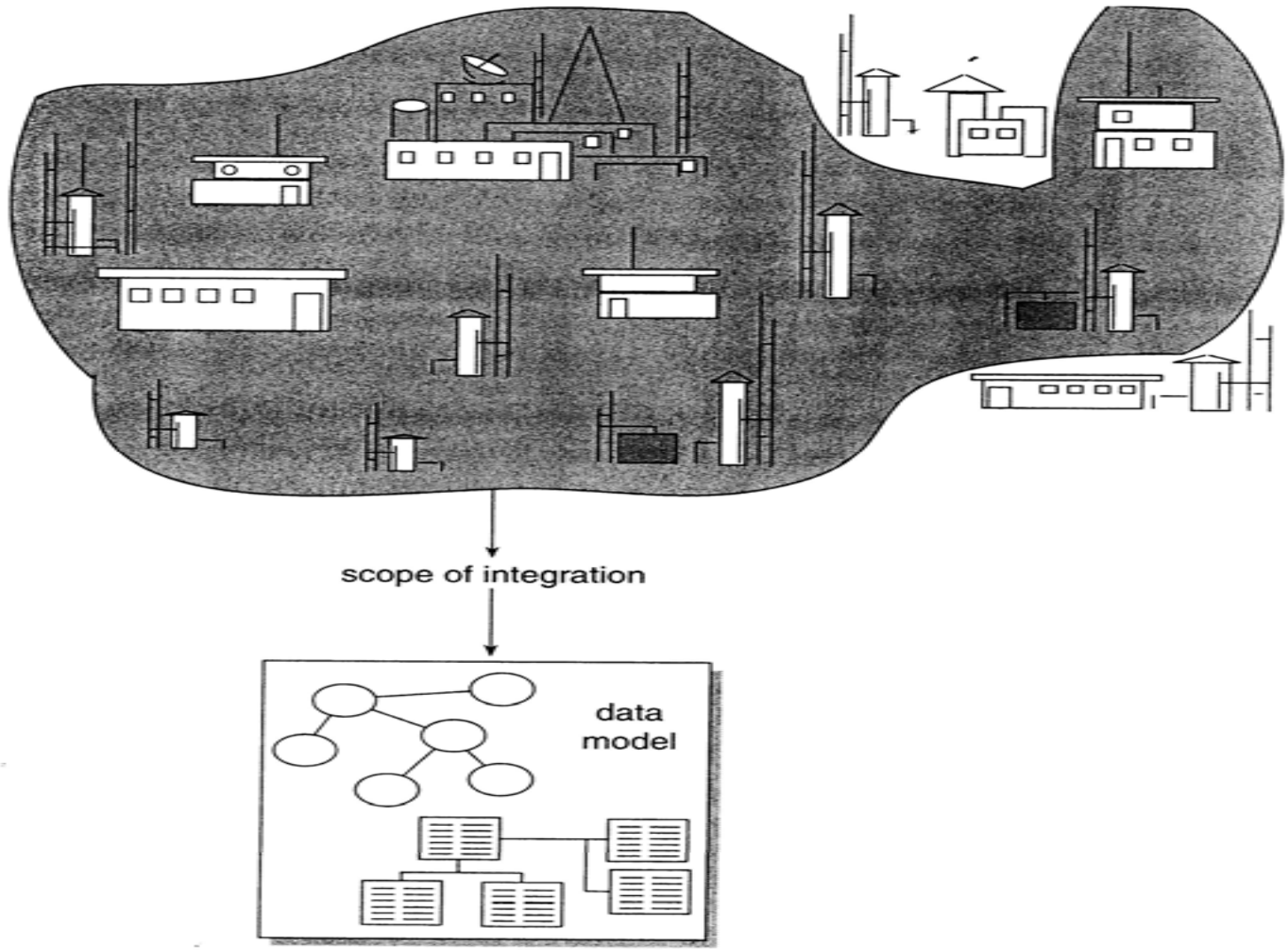
- 低级数据模型：

  物理模型

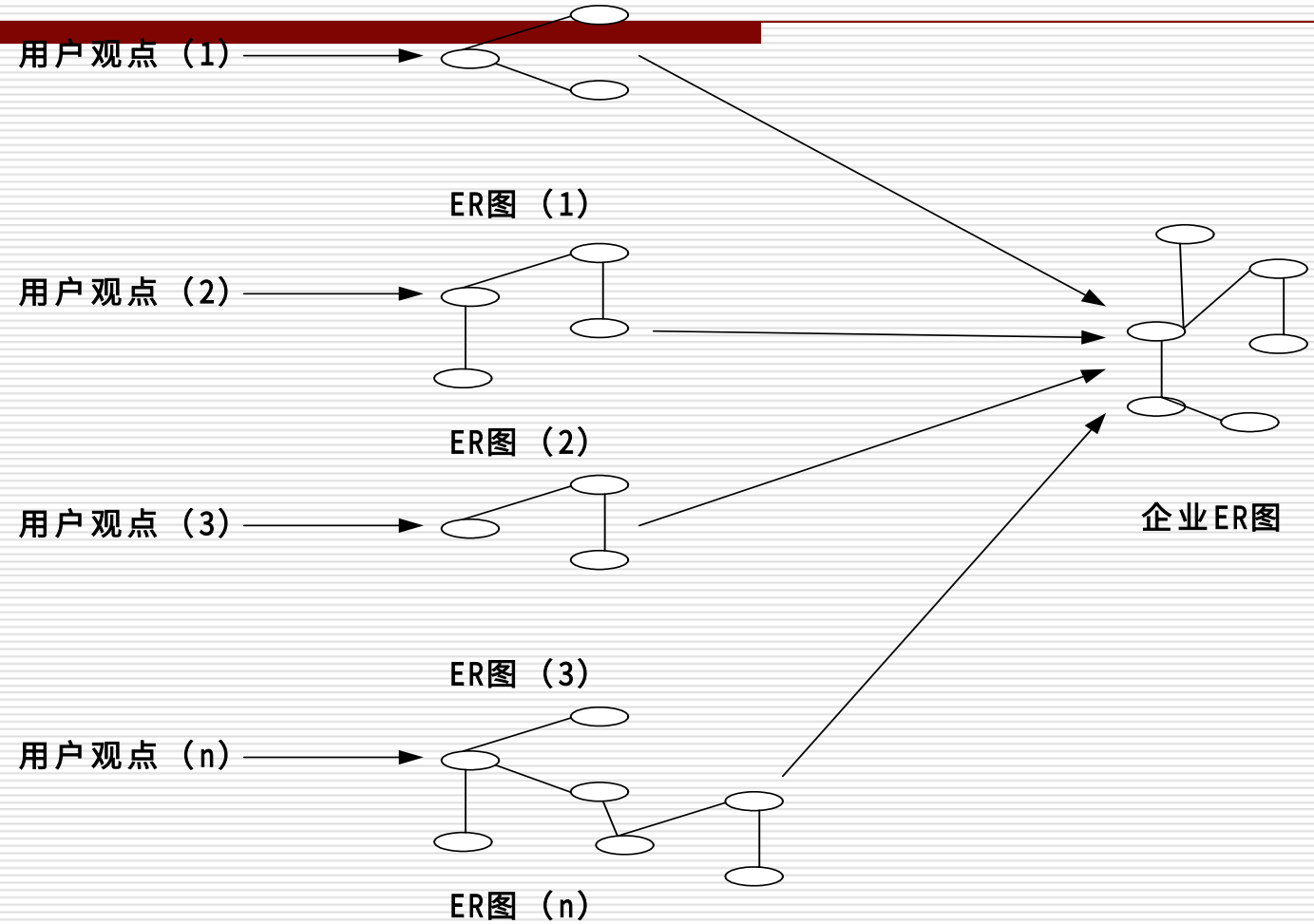
确定集成范围

- 集成范围：定义数据模型的边界，确定实体属于或不属于模型范围
- 集成范围需要在建模之前进行定义
- 集成范围由系统的建模者、管理人员和最终用户共同确定

确定集成范围（续）

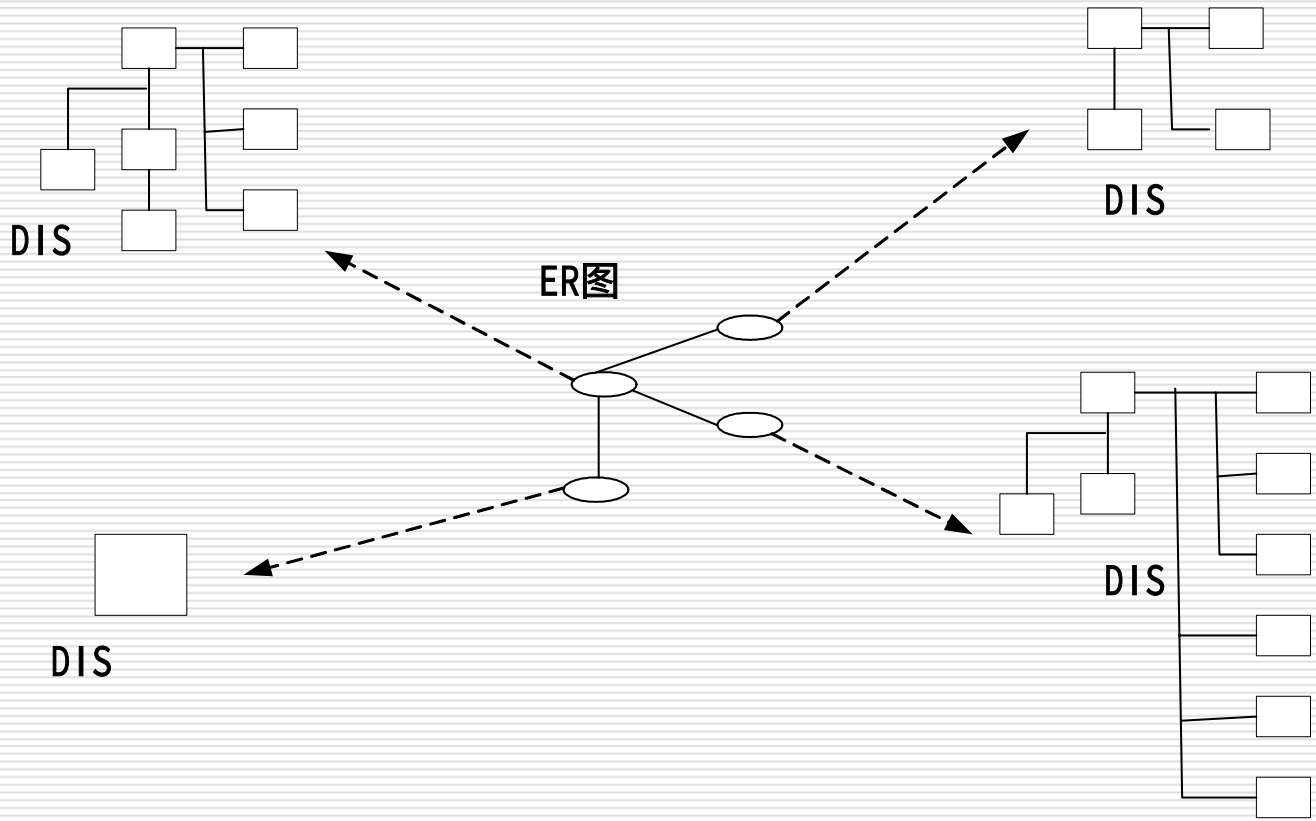


企业ER图



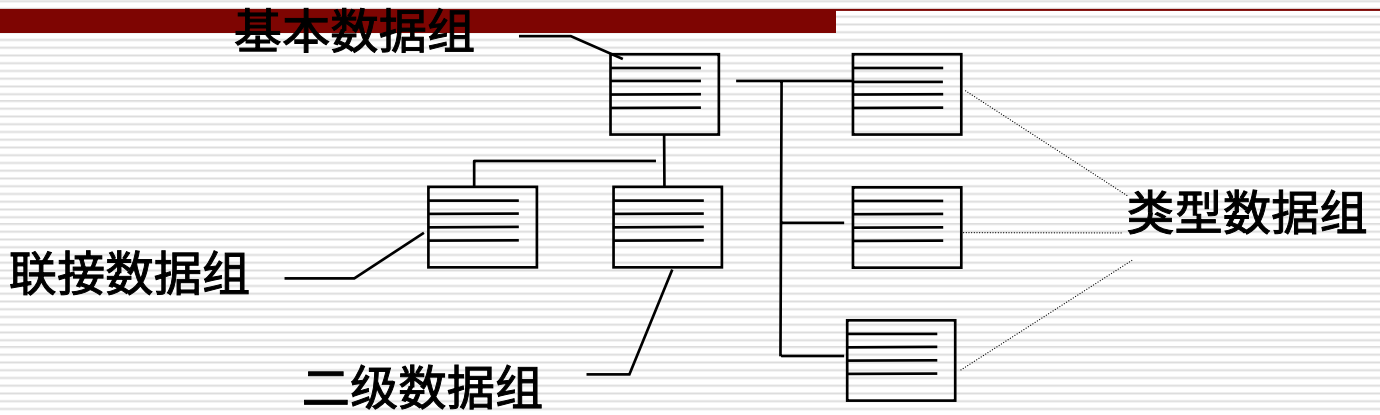
企业ER图是由反映不同用户观点的ER图构造而成的

ER图与DIS



ER图中的每个实体都有与其对应的DIS进一步定义

DIS的基本结构

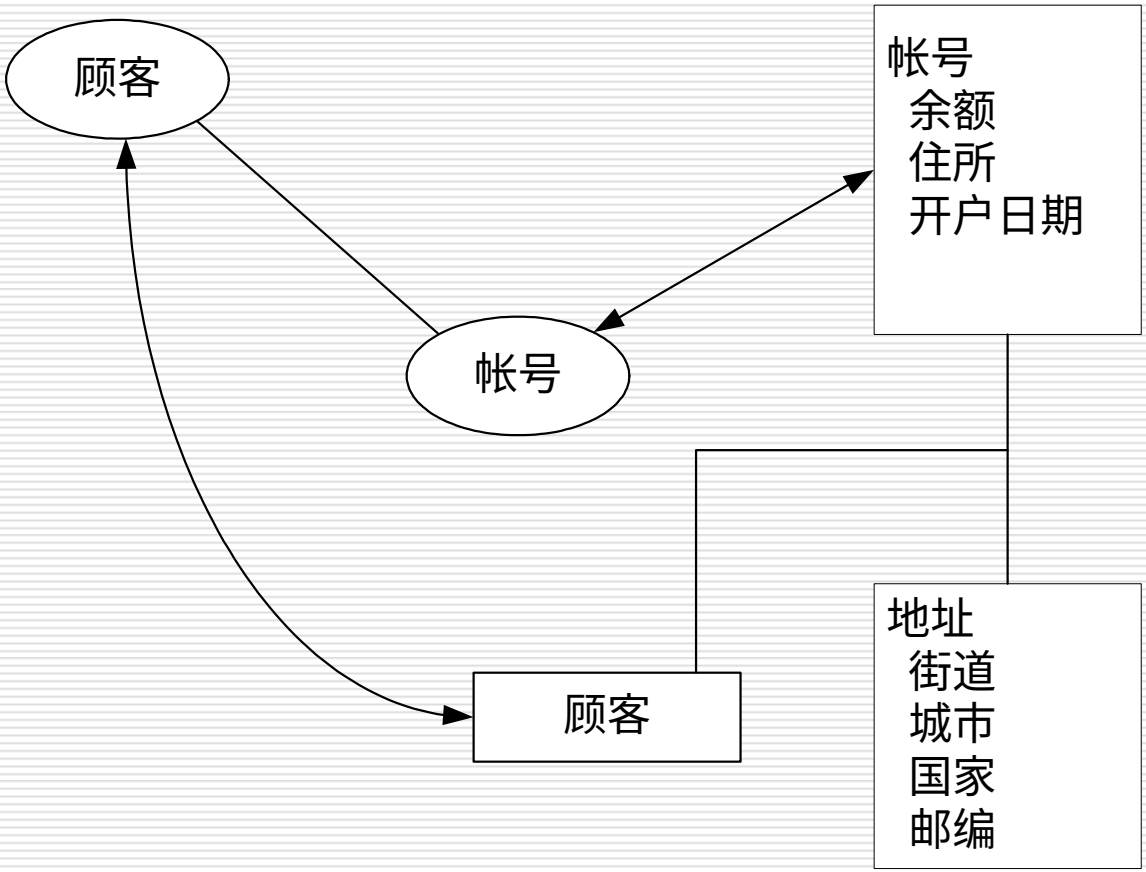


其中：

- 基本数据组：应包括主题的码和属性，一个主题只存在一个基本数据组。
  如：“ 顾客 ”主题中的顾客号、顾客名、性别等。
- 联接数据组：反映主题之间的联系，往往是一个主题的公共码键。
- 二级数据组：相对稳定的数据组。如：顾客的地址、电话、文化程度等。
- 类型数据组：频繁变动的数据组。如：顾客的购物记录。

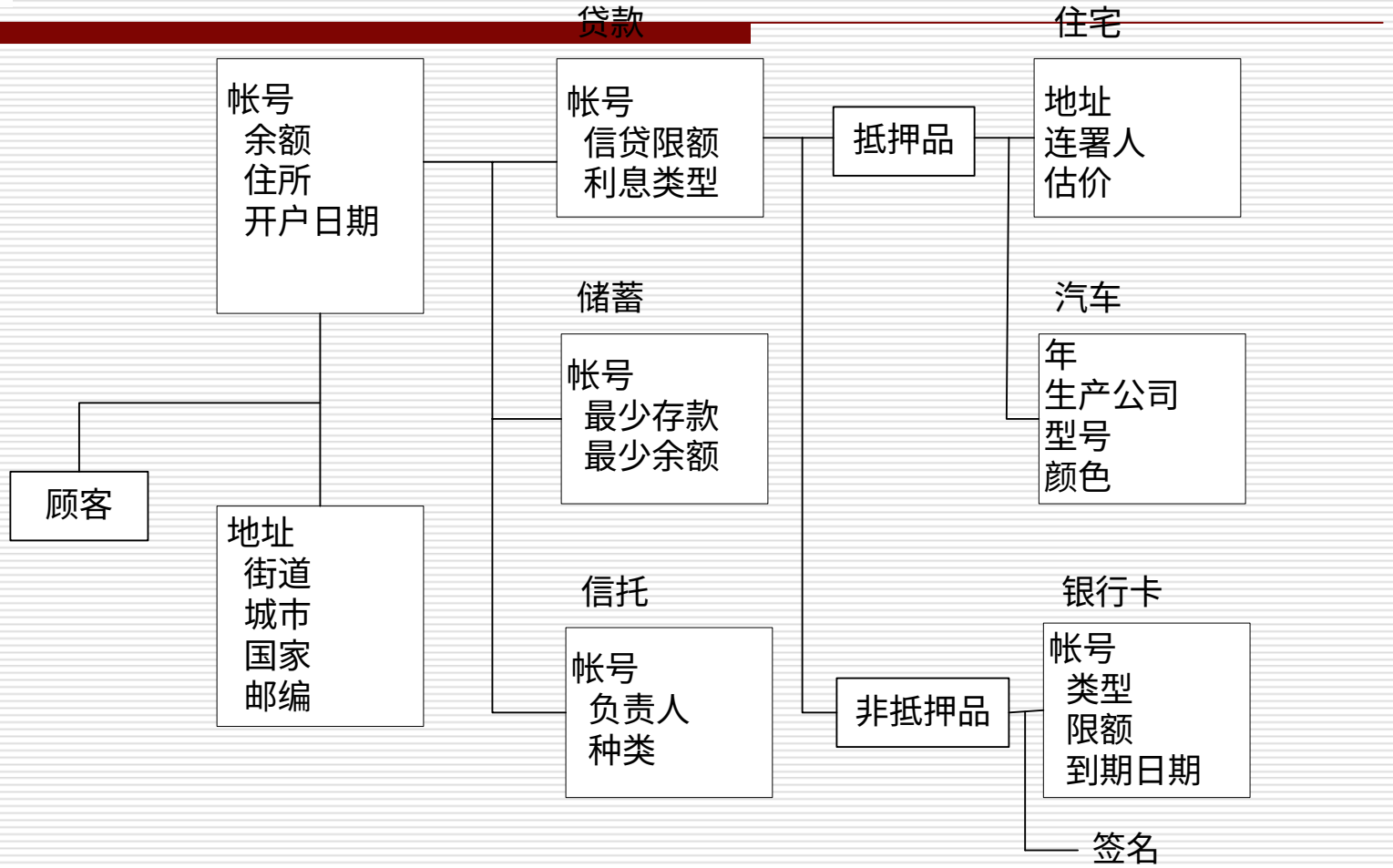
稳定性：基本数据组 > 二级数据组 > 类型数据组

联接数据组



在ER图中标明的关系在DIS中由联接数据组体现

扩展的DIS



一个扩展的DIS，表明银行可提供的不同贷款类型

显示不同子分类标准的DIS

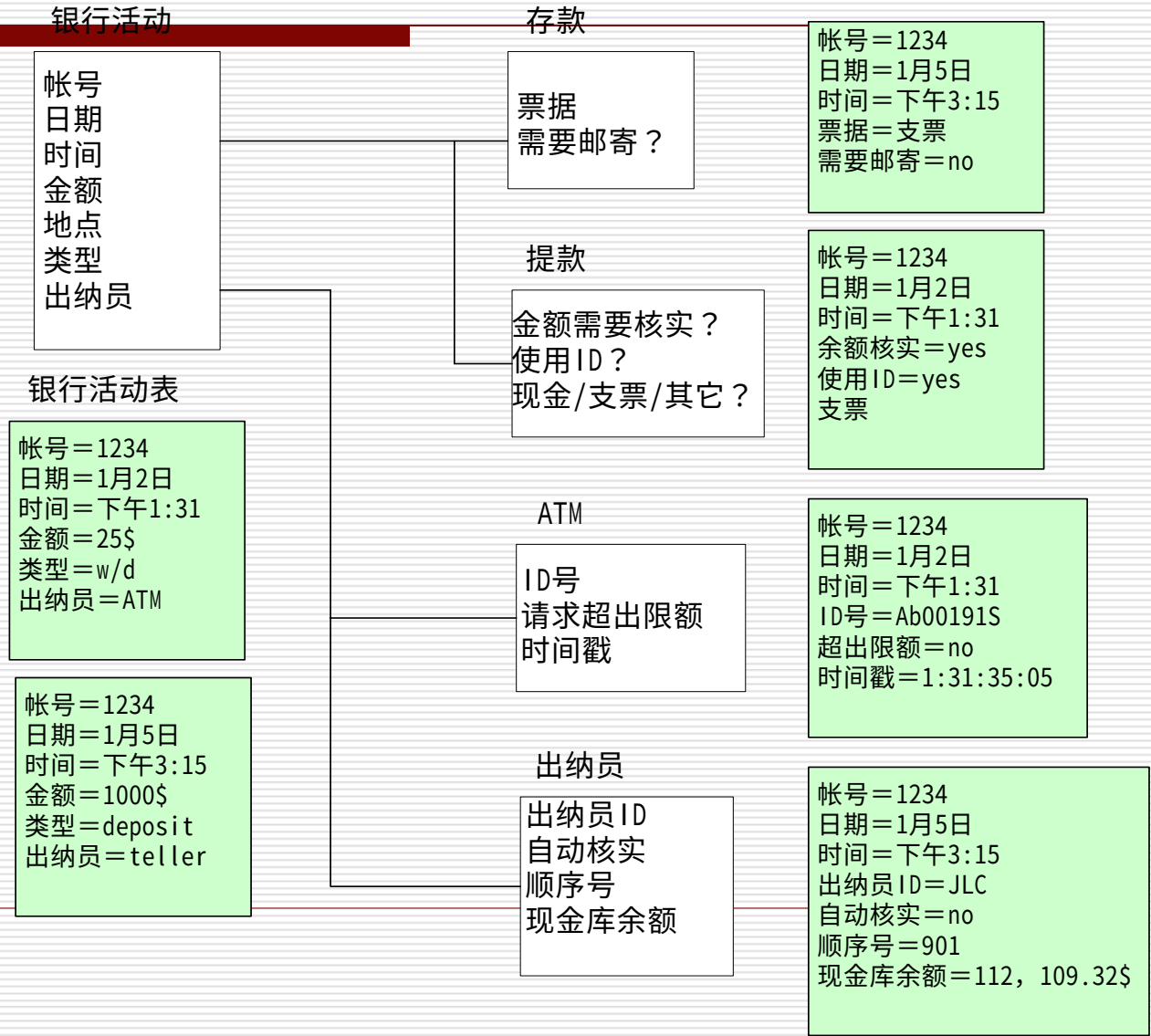
该DIS所反映的业务活动类型

- ATM存款
- ATM提款
- 出纳存款
- 出纳提款

数据模型产生的物理表与数据模型的关系

两个交易

- 1月2日下午1:31，ATM提款
- 1月5日下午3:15，出纳存款



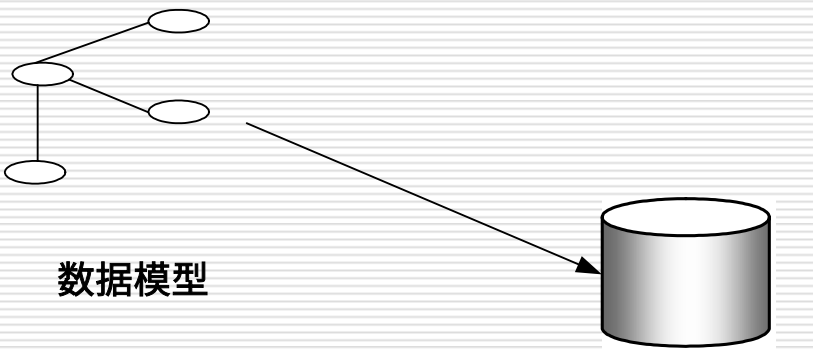
性能问题

提高系统性能，主要是要提高系统的物理I/O性能。

在数据仓库的设计中，应尽量减少每次查询处理要求的I/O次数，而使每次I/O又能返回尽量多的记录。

提高数据仓库性能的主要途径：

- 粒度划分
- 数据分片
- 合并表
- 选择冗余
- 进一步分离数据
- 导出数据
- 建立广义索引



粒度划分

- 粒度：数据的综合程度。

  例如：细节— 轻度综合— 高度综合

  - 一张表的数据量很大时，就需要两个级别的粒度。
  - 粒度的划分，主要考虑行数。因为按行组织索引，索引依赖于行数，索引大小直接影响I/O次数。
  - 有关专家认为，如果数据量只有10000行时，不考虑粒度，如果有一千万行时，需要一个低的粒度级。

空间/行数计算

估算数据仓库环境中的行数/空间大小

1. 对每一个已知的表：

   计算一行所占字节数的

   —最大估计值

   —最小估计值

   对一年内：

   最大行数可能是多少？

   最小行数可能是多少？

   对五年内：

   最大行数可能是多少？

   最小行数可能是多少？

   对表的每个关键字：

   该关键字的大小（按字节）是多少？

   一年总的最大空间＝最大行大小×一年内最大行数

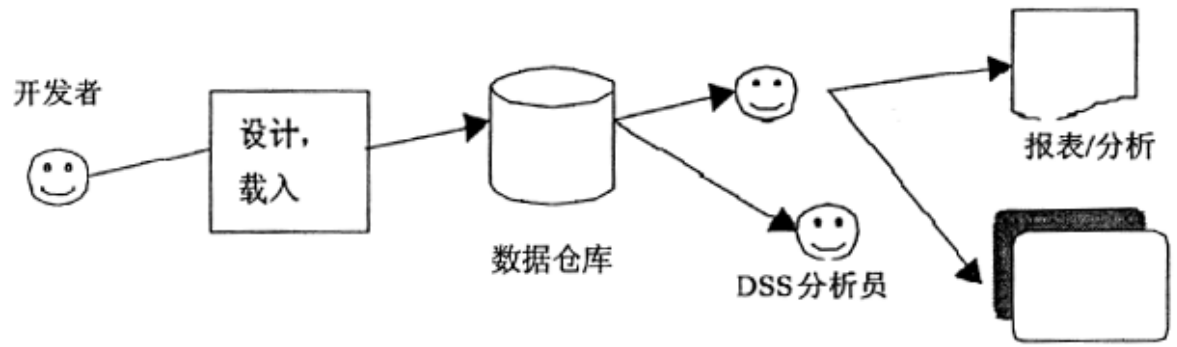
   一年总的最小空间＝最小行大小×一年内最小行数

2. 对所有已知的表重复第 1 步。

粒度设计过程中的一个参考

一年期		五年期	
100,000,000	数据同时存在于磁盘与溢出存储器上，但大部分是在溢出存储器上，需要认真设计粒度	1,000,000,000	数据同时存在于磁盘与溢出存储器上，但大部分是在溢出存储器上，需要认真设计粒度
10,000,000	可能有一些数据存储于溢出存储器，但大部分仍处于磁盘中，需要考虑粒度问题	100,000,000	可能有一些数据存储于溢出存储器，但大部分仍处于磁盘中，需要考虑粒度问题
1,000,000	数据存储在磁盘上，几乎可以采用任何数据库设计	10,000,000	数据存储在磁盘上，几乎可以采用所有数据库设计
100,000	数据存储在磁盘上，可以采用任何数据库设计	1,000,000	数据存储在磁盘上，可以采用任何数据库设计

粒度的确定



经验规则：

在第一次的设计过程中，如果有50%是正确的，那么整个设计就是成功的。

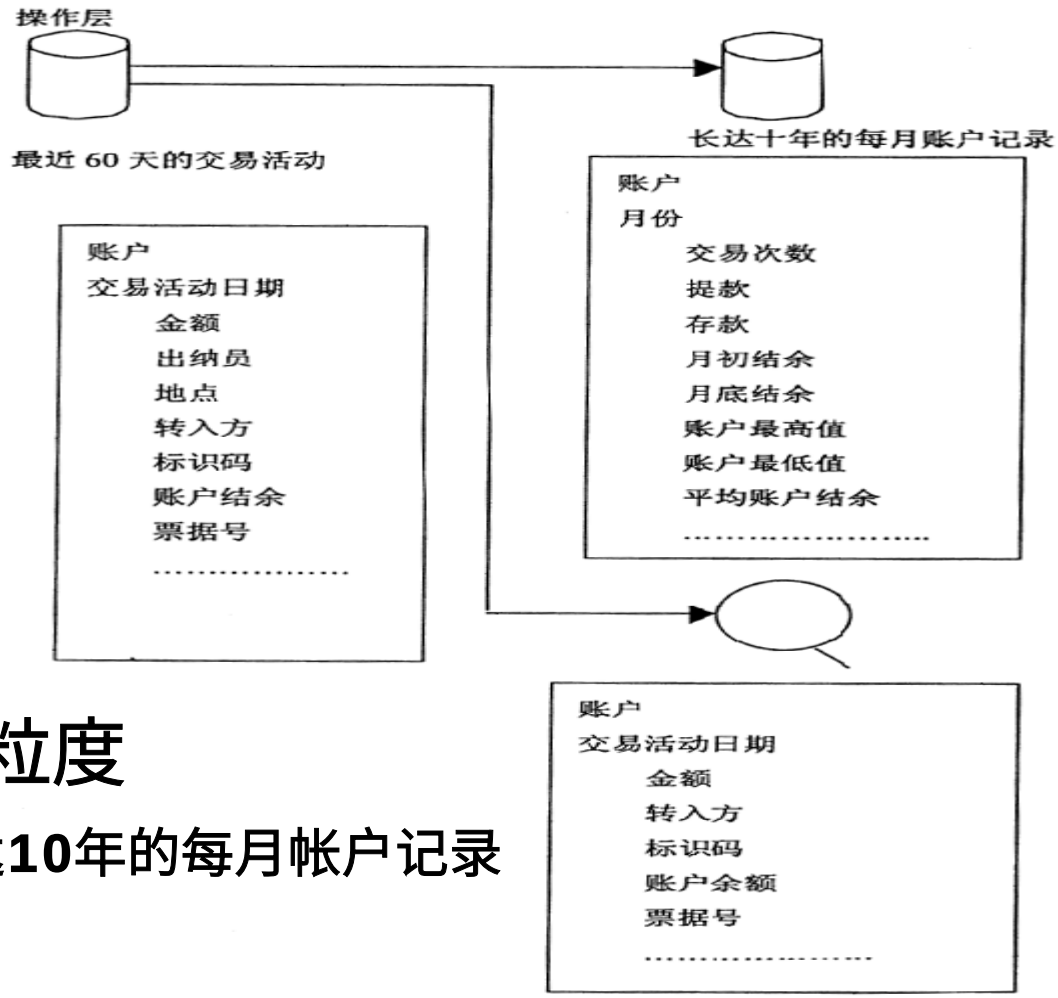
- 快速建立数据仓库的很小的子集并认真听取用户的反馈意见。
- 使用原型法。
- 参考别人的经验。
- 找一个有经验的用户协同你工作。
- 以企业中已有的功能需求作为参考。
- 用模拟的输出进行 JAD（联合应用程序设计）会议。

由开发人员与最终用户共同决定

提高数据粒度的方法

- 当源数据放入数据仓库时，对它进行汇总
- 当源数据放入数据仓库时，对它求平均或进行计算
- 把最大/最小的一组值放入数据仓库
- 只把显然需要的数据放入数据仓库
- 用条件逻辑选取记录的一个子集放入数据仓库

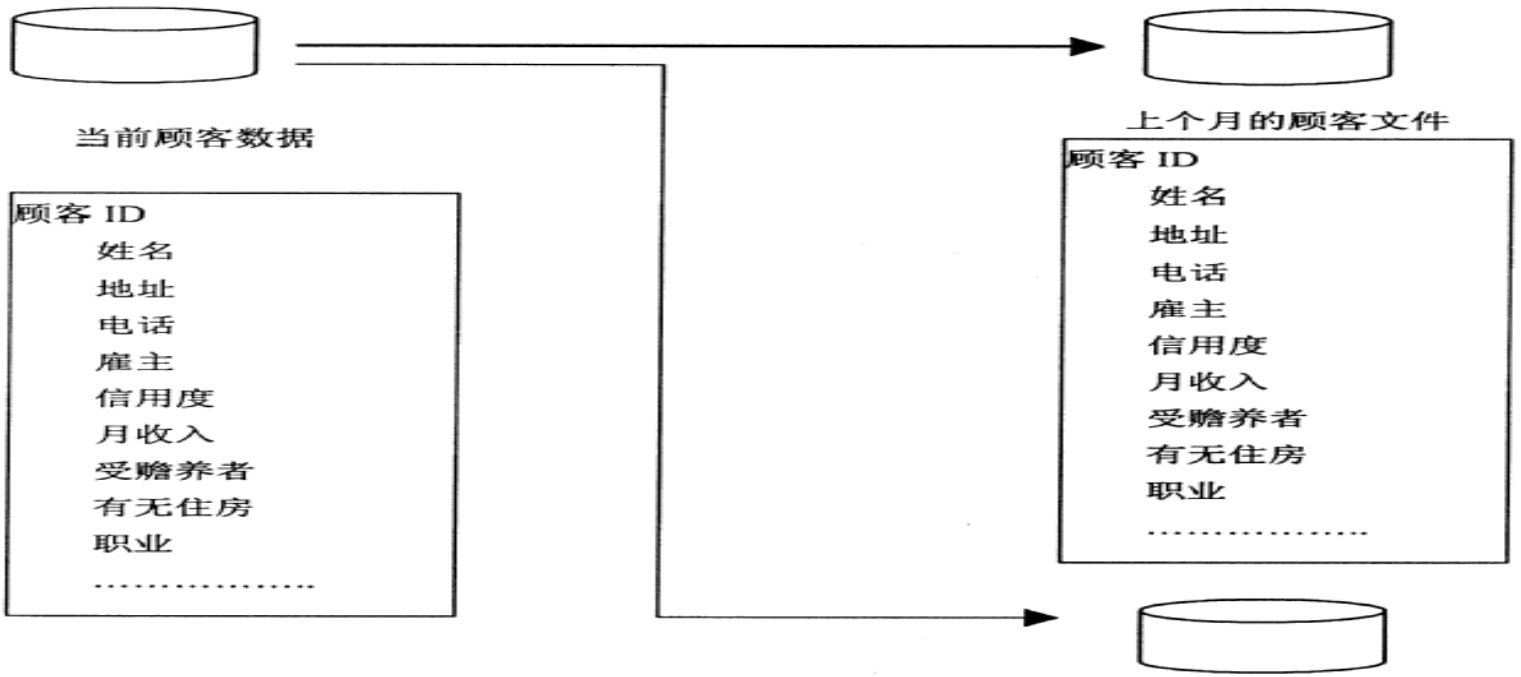
粒度划分举例：银行业（I）



银行环境中的双重粒度

- 轻度综合粒度——长达10年的每月帐户记录
- 档案级

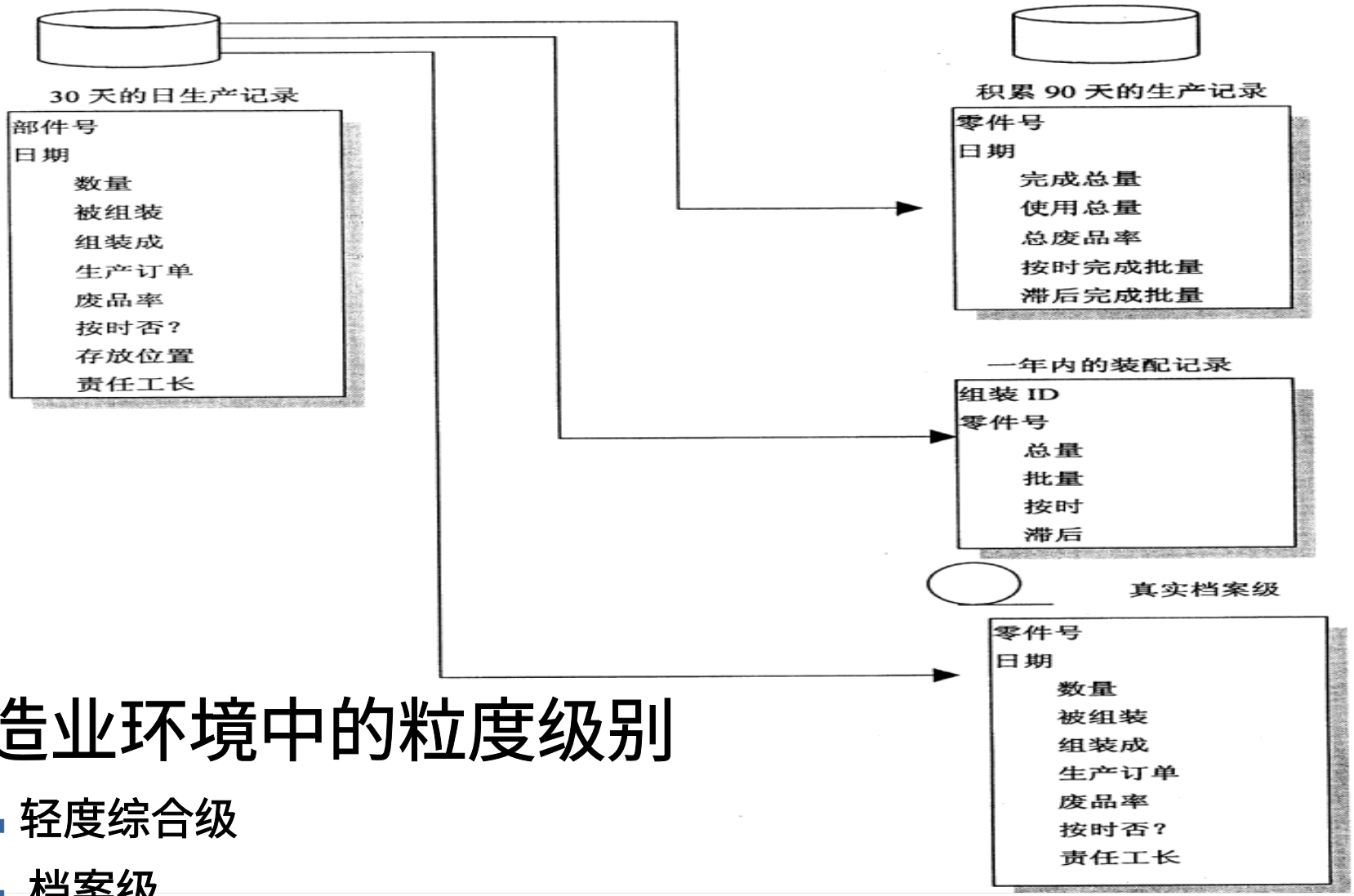
粒度划分举例：银行业（II）



银行环境中双重粒度的另一种形

- 上个月的顾客文件
- 过去十年的连续顾客记录

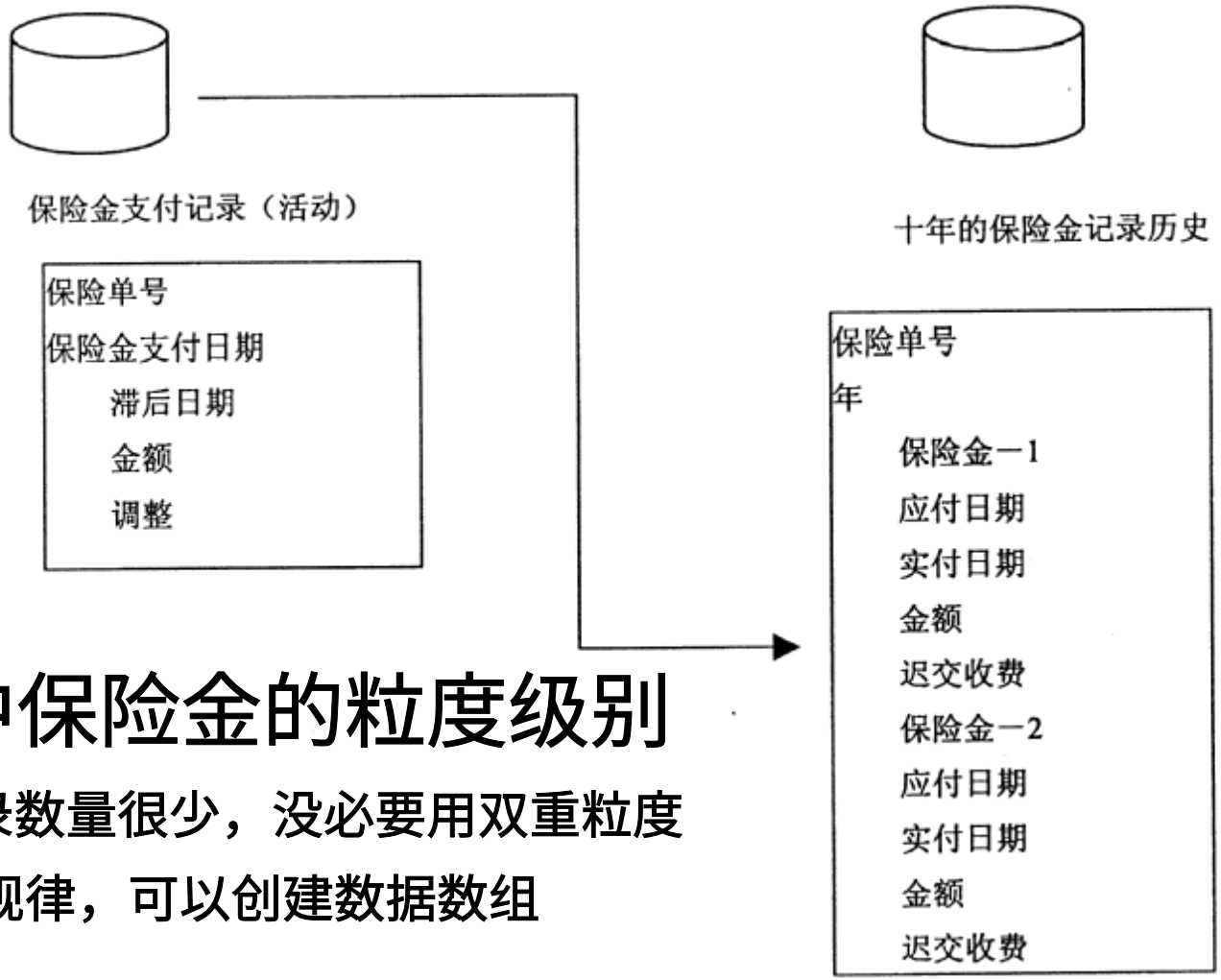
粒度划分举例：制造业



制造业环境中的粒度级别

- 轻度综合级
- 档案级

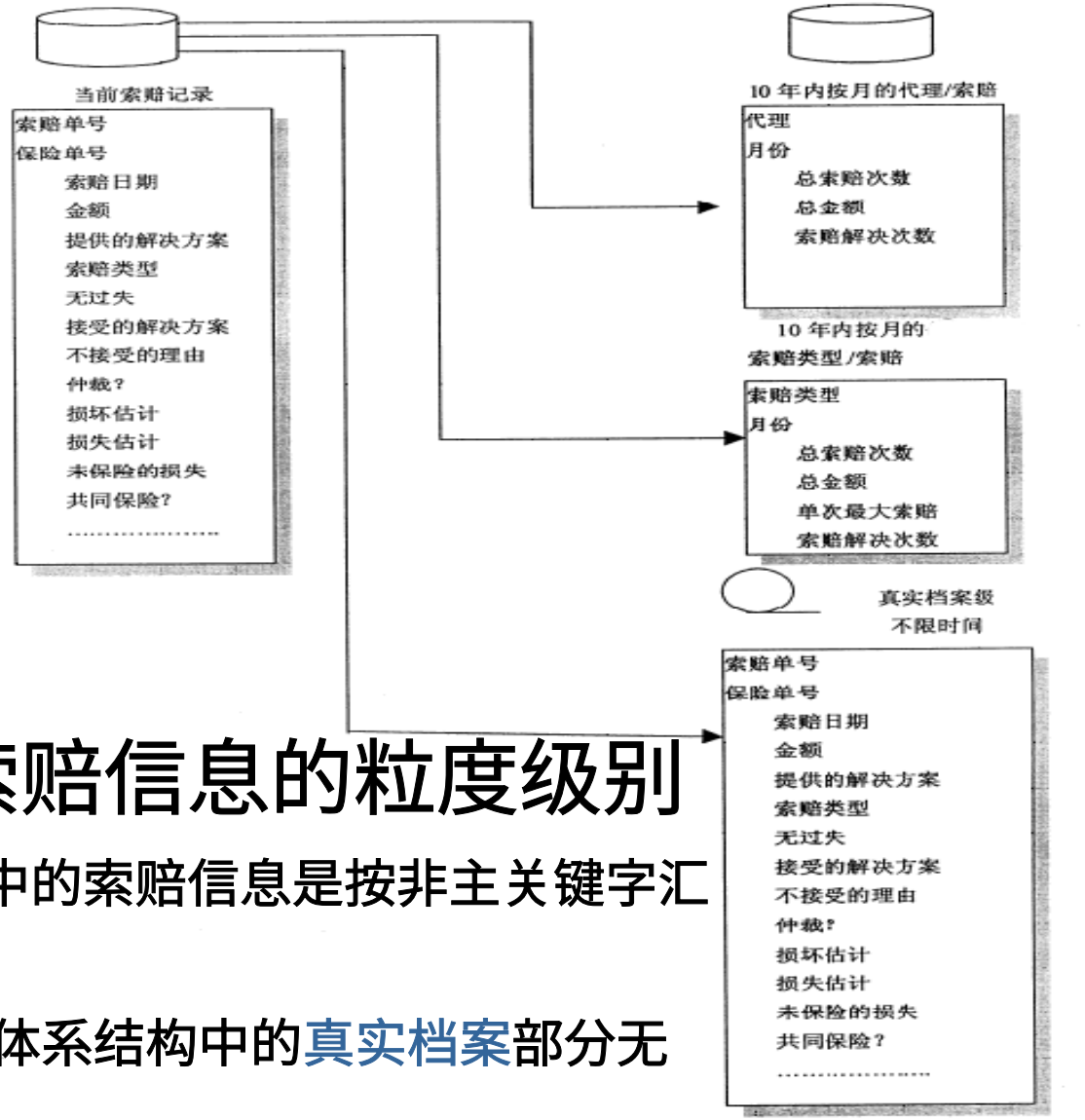
粒度划分举例：保险业（I）



保险业环境中保险金的粒度级别

- 保险金支付记录数量很少，没必要用双重粒度
- 保险金记帐有规律，可以创建数据数组

粒度划分举例：保险业（II）



保险业环境中保险索赔信息的粒度级别

- 数据仓库的轻度汇总部分中的索赔信息是按非主关键字汇总
- 索赔信息必须在数据仓库体系结构中的真实档案部分无期限存放

数据分片

- 分片：把逻辑上统一的数据分割成较小的、可以独立管理的物理单元（分片）进行存储。
  - 可按时间、按地区、按业务类型进行数据分片

1996	家电类	日用化工类	针织服装类	副食类
第一季度	分片１	分片２	分片３	分片４
第二季度	分片５	分片６	分片７	分片８
第三季度	分片９	分片１０	分片１１	分片１２
第四季度	分片１３	分片１４	分片１５	分片１６

商品销售数据的分割

为什么需要数据分片

- 在OLTP环境中，修改操作较为频繁，采用数据分片技术容易导致记录在分片之间的转移，转移代价较高；在OLAP环境中，数据极少更新，利用数据分片技术可以大大提高性能。
  - 减少内存的使用空间：系统仅需要将用户查询所涉及的数据调入内存
  - Join操作代价较低：将一个大的Join操作分解为若干个小Join操作之和
  - 数据分片后，可以利用并行操作提升查询的效率
  - 增加灵活性。例如：在不同的时间段，可以有不同的模式定义

    1989年的顾客数据处理其数据项比1988年多

数据分片：选择分片的标准

- 选择分片的标准
  - 数据量的大小（而非记录行数）
  - 数据分片处理的对象的特征以及属性之间的相关性

例如：商品按类和时间作为分片的标准

供应商按地区和时间作为分片的标准

  - 易于实现（实施）

例如：按时间、业务类型

  - 与粒度划分策略统一起来

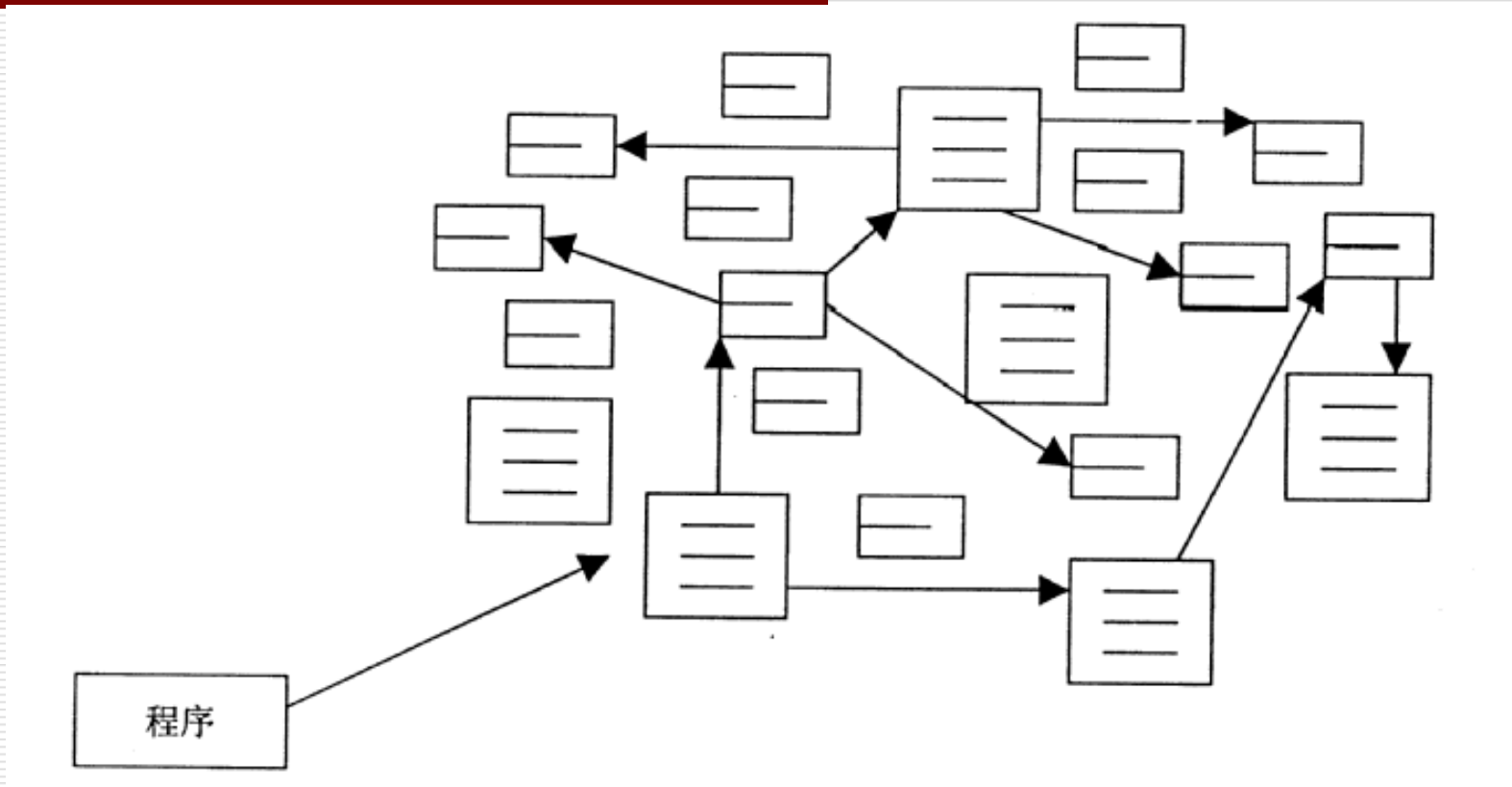
例如：按时间于商品类对销售数据进行综合（粒度划分）

每一粒度再按时间与商品类进行分片，分片后仍便于做高度综合。

数据分片：数据分片的策略

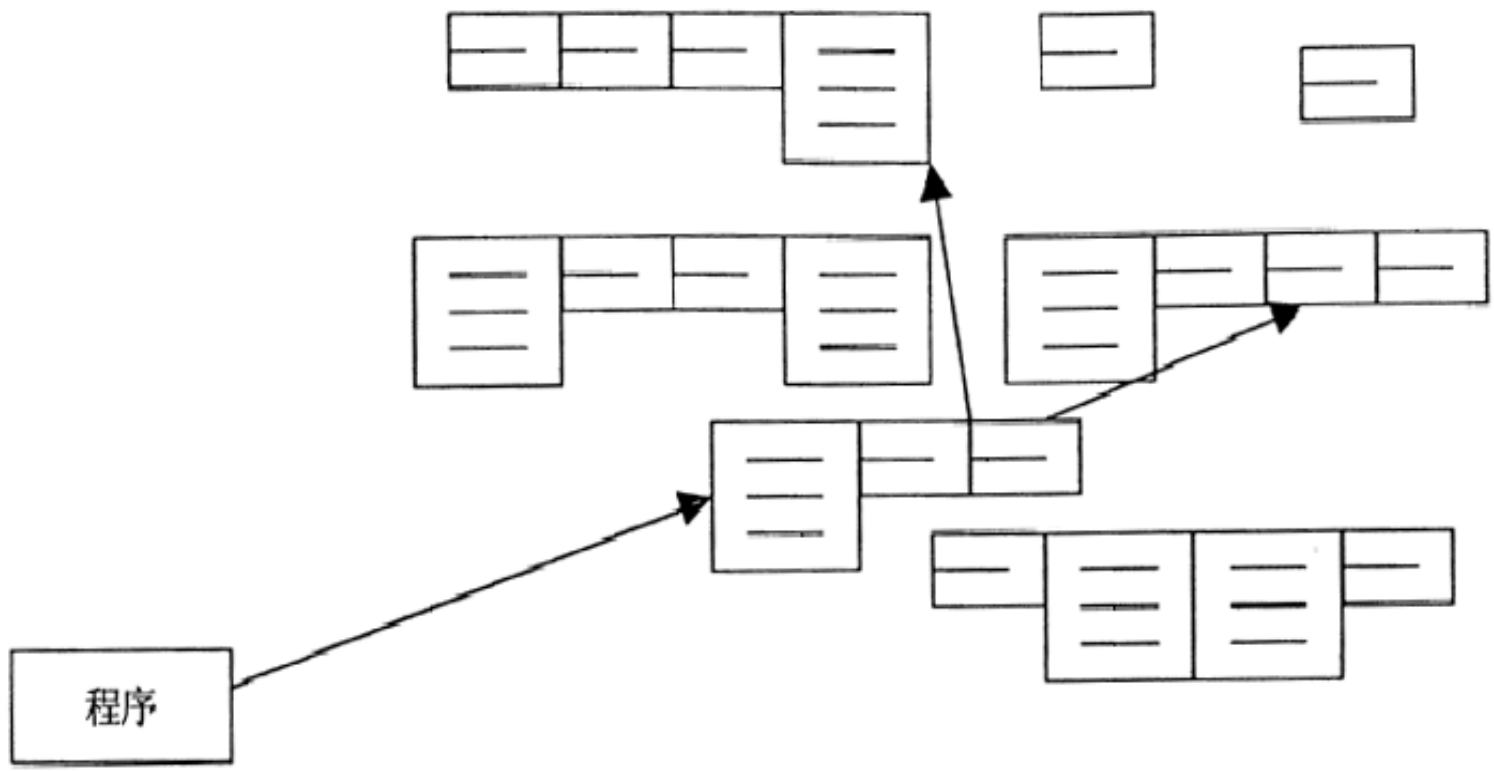
- 对于给定的磁盘个数，进行数据分片的方法
  - 范围分片：利用属性值的范围进行数据分片
    - 优点：数据逻辑比较清楚
    - 缺点：导致数据分配的不平衡
  - 循环轮转法：按照一定顺序，依次存放各个数据。数据分配均匀。
  - **Hashing方法：利用Hash函数。数据分配均匀。**

合并表



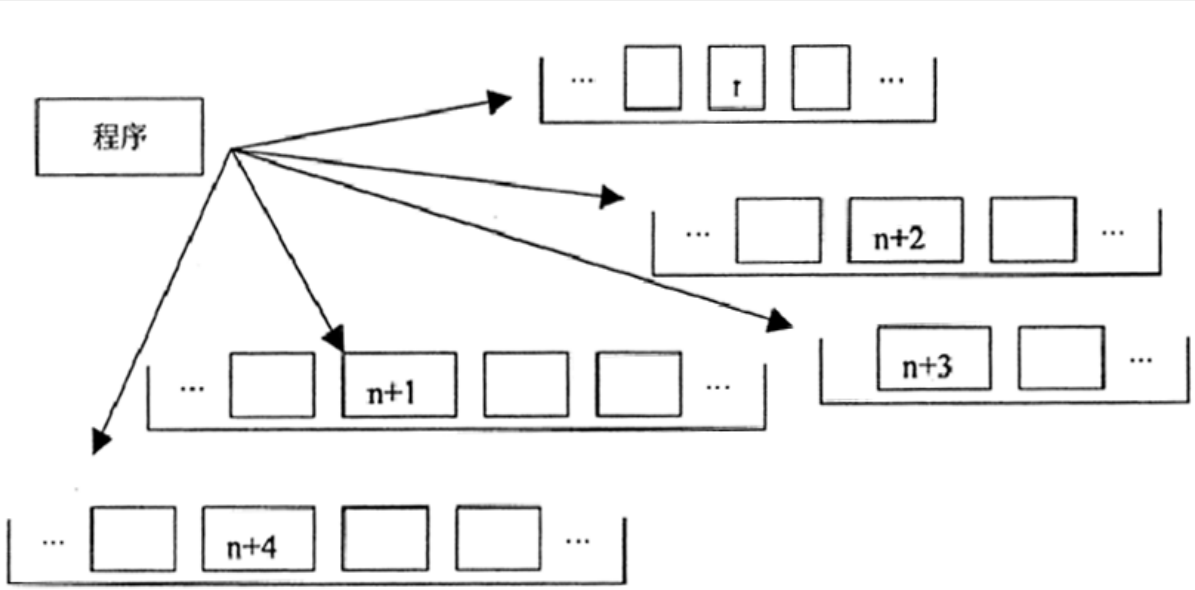
- 当有许多表时，动态连接需要进行大量的I/O

合并表（续）

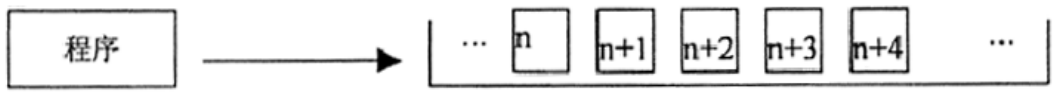


- 表在物理上合并之后，只需要较少的I/O

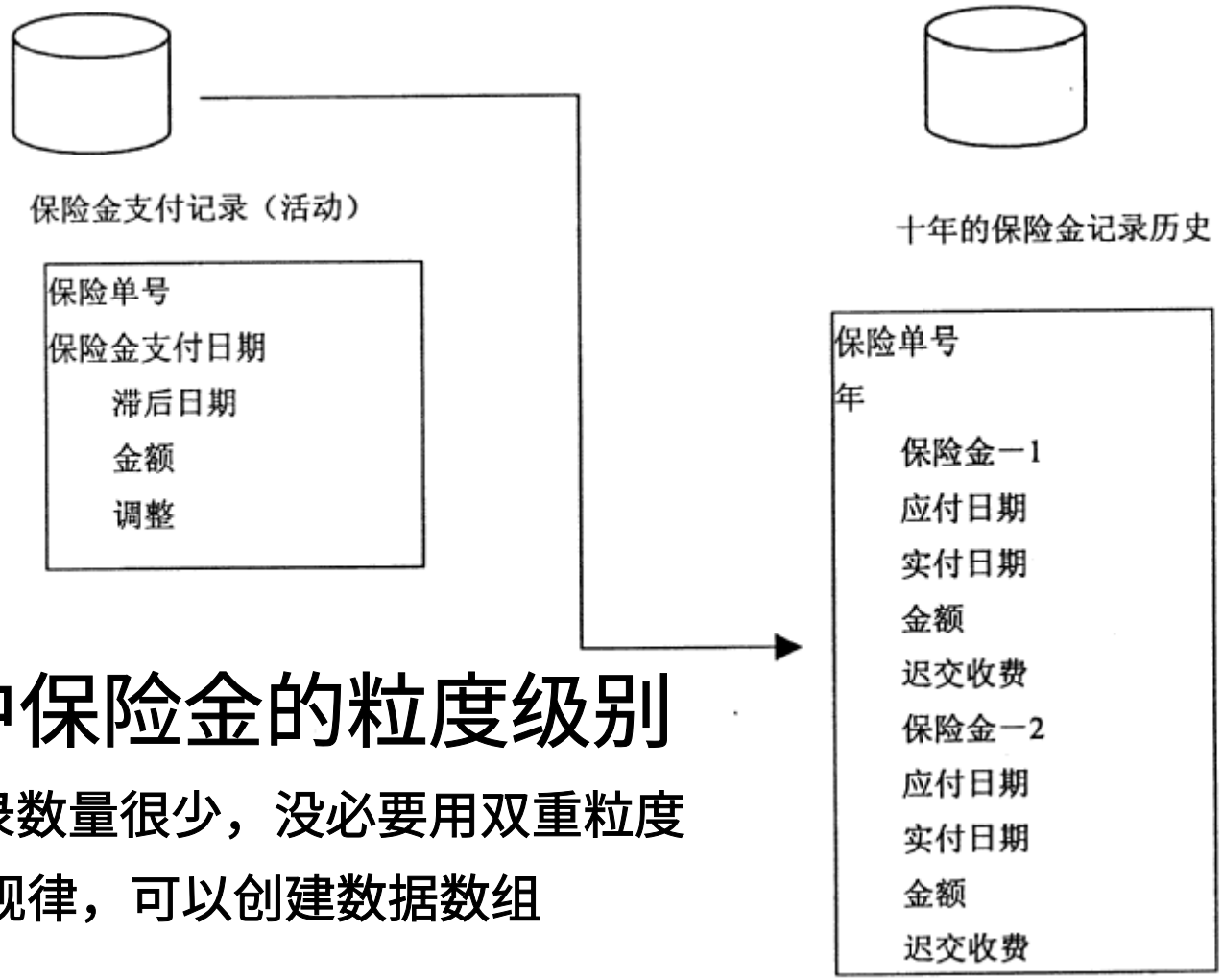
数据数组



- 在适合的情况下，创建数据数组可以提高性能，节省资源
  - 数列中值的数量稳定
  - 数值按顺序访问
  - 创建与修改有规律



数据数组举例：保险业



保险业环境中保险金的粒度级别

- 保险金支付记录数量很少，没必要用双重粒度
- 保险金记帐有规律，可以创建数据数组

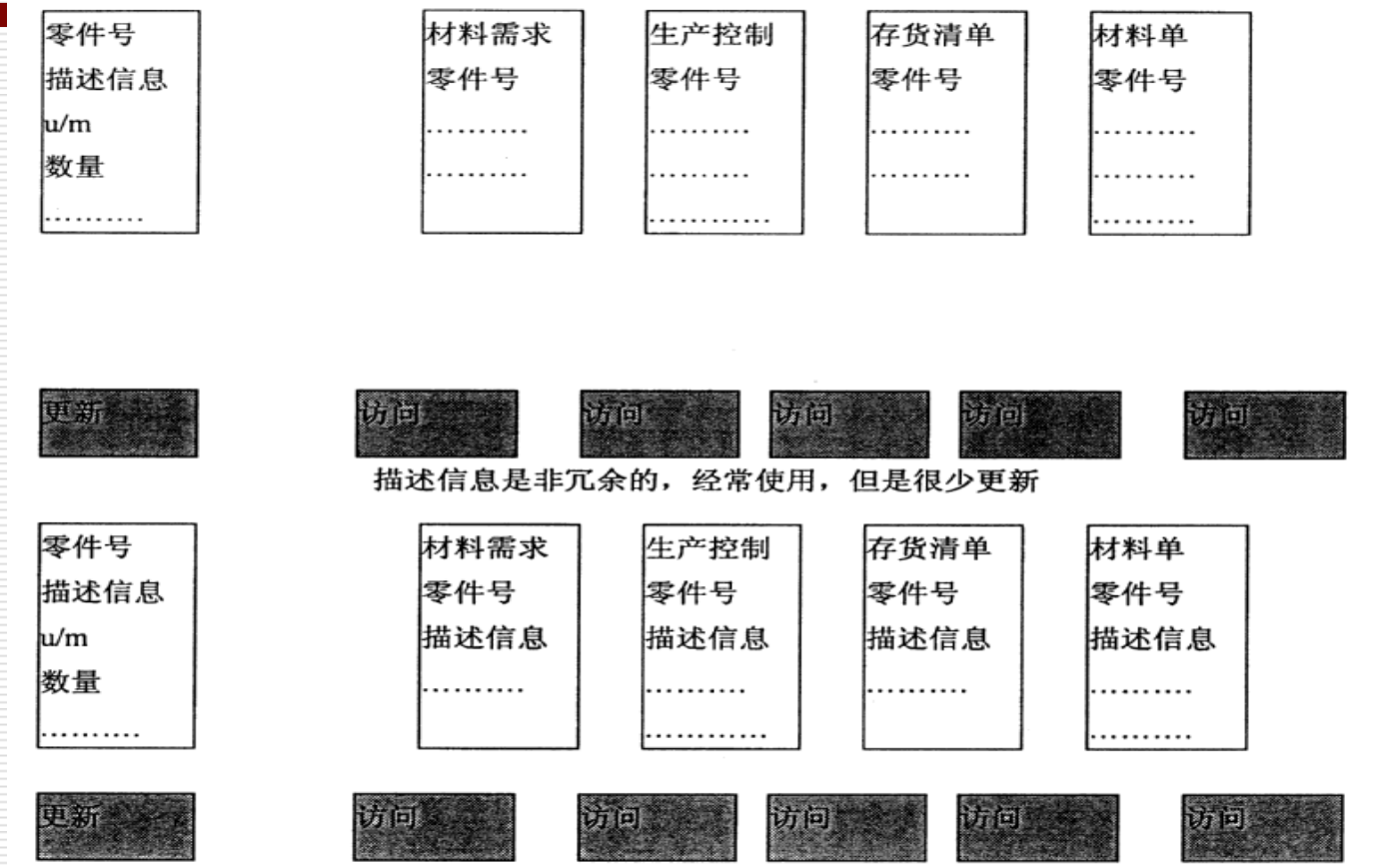
引入冗余

- 一项数据属性（主外码不算此类）存在于多个关系模式中

  例如：在采购表/销售表中增加商品名称、商品类型等

- 提高了性能，省去了Join操作

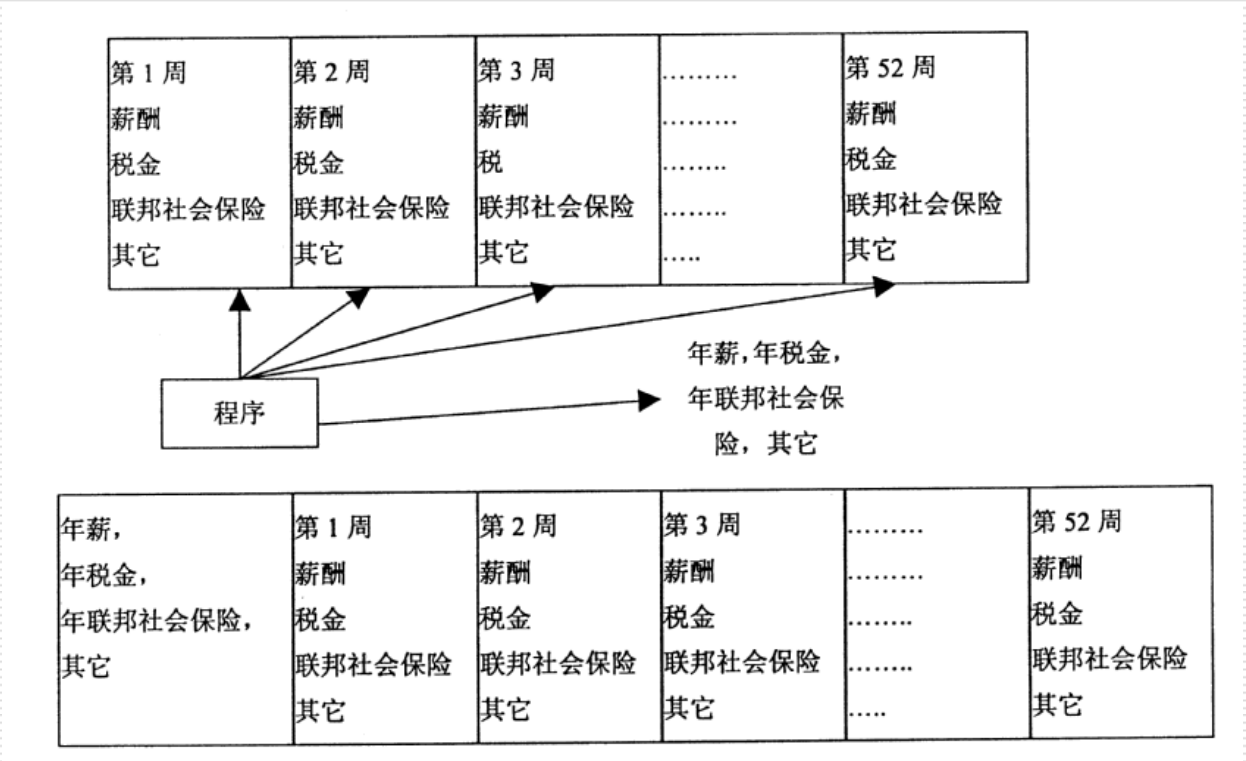
引入冗余：例



- 尽管描述信息冗余，但很少更新，提高了查询性能

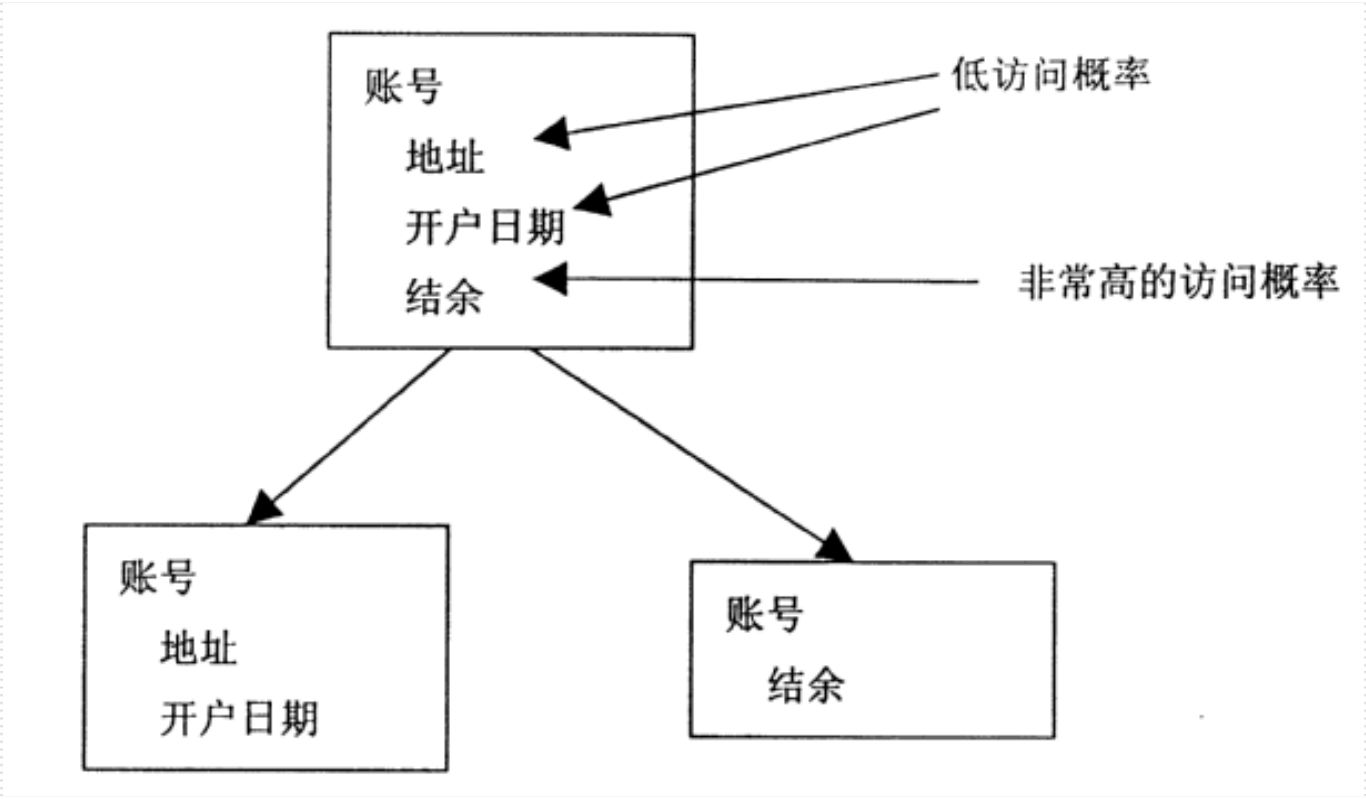
导出数据

- 事先在源数据基础上，进行汇总或计算，生成导出数据



导出数据，只计算一次就可以永久使用了

分离数据



- 根据访问概率的巨大差异进一步分离数据

数据仓库中的索引技术

- 位图索引（Bitmap Index）
  - 针对一些特殊的列建立索引
  - 列中的每一个值对应一个向量中的一位
  - 向量的长度对应与记录的条数
  - 不适合列中值的个数太多的情况

基本表

客户号	地区	类型
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

类型索引

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

地区索引

RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

数据仓库中的索引技术（续）

- 连接索引（Join Index）
  - 一个表对另一个表中包含本表中相关列内容的行进行索引。

产品

产品编码	名称	单价	索引项
P1	电视	5000	R1, R2
p2	冰箱	4000	R3, R4

销售情况

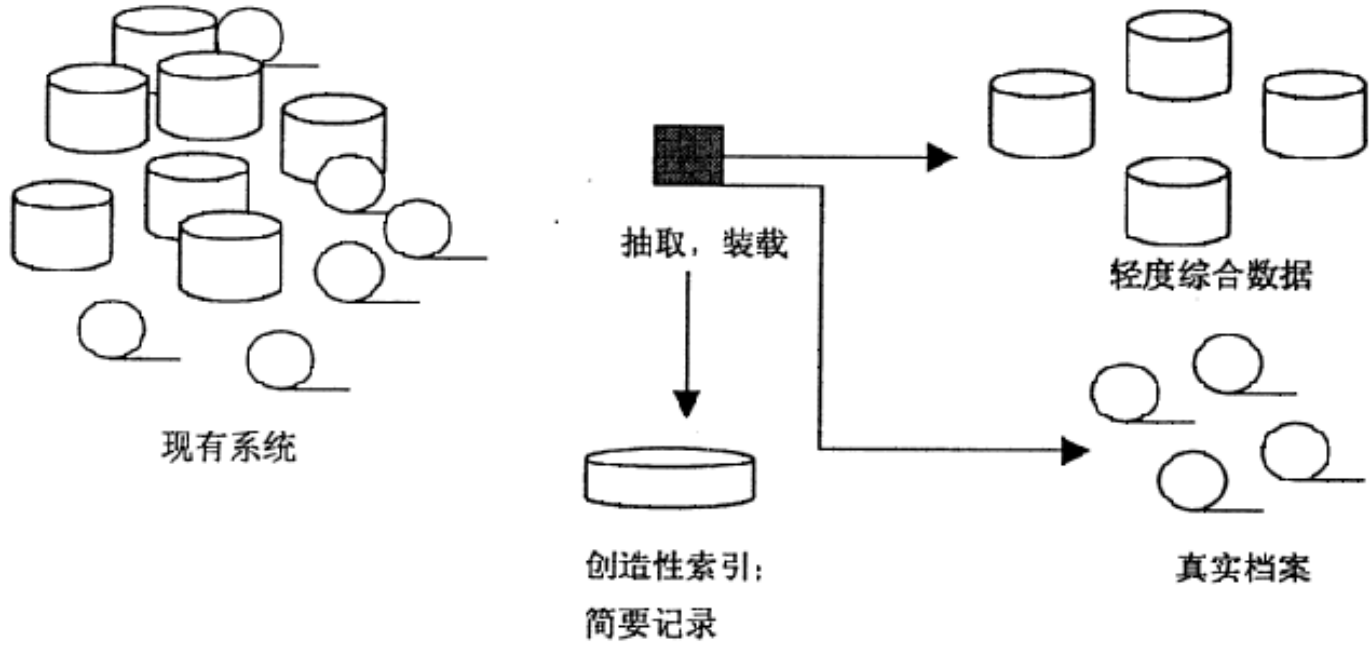
记录号	产品编码	月份	数量
R1	P1	2000/1	5128
R2	P1	2000/2	3246
R4	p2	2000/1	3457
R3	p2	2000/2	4030

建立广义（创造性）索引

- 建立广义索引
  - 用于处理最大（小）值问题

    例如：每月销售最好的前5种商品？
  - 当数据装入到DW时，生成“广义索引”内容
  - 广义索引随着数据仓库的发展，数目会增加，但每个索引的规模小，需要在元数据中定义“广义索引”

建立广义（创造性）索引（续）



- 卷中的前十名顾客是____。
- 这个抽取中的平均交易额是 *nnn.nn*$。
- 最高的交易额是 *nnn.nn*$。
- 仍然活动但没有购买的顾客数为 *nn*。

数据仓库中的元数据

元数据：关于数据的数据；
描述数据结构、内容、码、索引等信息。

- 元数据的重要性
- 元数据的内容

元数据的重要性

- 管理人员做分析时，往往先从元数据入手。

例如：从元数据中查广义索引，再进一步搜索

- 支持数据转换：DB环境的数据 DW环境的数据

元数据描述“转换”；元数据本身具有良好的灵活性，适应变化。

例如：不同时期，数据结构是变化的

- 支持对数据仓库中数据的理解

例如：结构、粒度层次、分片策略、索引等

数据仓库元数据的内容

- 源数据的描述：数据源名，存储地点，存储内容简述

  记录系统定义： 主题名，属性名，源表名，源属性名

  …… ……
- 数据仓库中数据的描述

  主题描述：主题名，主题的公共码键, 有关描述信息等

  逻辑模型的定义； 关系名，属性1，属性2，。。。，属性n

  粒度的定义

  数据分片的定义

  广义索引： 广义索引名，属性1，属性2，。。。，属性n

  …… ……
- 数据转换的描述

  数据进入数据仓库的转换规则

第三章　数据仓库设计

- 数据仓库设计方法概述
  - DW设计与DB设计
  - DW设计的三级数据模型
  - 性能问题
  - 数据仓库中的元数据
- 数据仓库设计步骤
  - 概念模型设计
  - 逻辑模型设计
  - 物理模型设计
  - 数据仓库生成
  - 数据仓库的使用和维护

数据仓库的设计步骤

基 本 思 路：

- 数据驱动
  - 从现存数据库系统基础上进行开发：抽取、综合、集成
  - 服务于管理决策分析
- 原型法
  - 不断反馈、循环、理解需求，使系统增长、完善
  - 步骤是大体上的，不是绝对的顺序
  - 决策人员的参与极其重要
- 大体步骤

  概念模型设计➱逻辑模型设计➱物理模型设计➱

  数据仓库生成➱数据仓库运行与维护

数据仓库设计的基本步骤

数据仓库运行与维护

概念模型设计

1.界定系统边界

2.确定主题域

技术准备工作

1.技术评估

2.技术环境准备

分析主题域，针对每一个选定的当前实施的主题域

逻辑模型设计

1.粒度层次划分

2.数据分片策略

3.记录系统定义

4.关系模式定义

物理模型设计

1.确定存储结构

2.确定索引结构

3.确定存放位置

4.确定存储分配

数据仓库生成

1.设计接口

2.数据装入

概念模型设计

- 任务
  - 确定系统边界
  - 确定主题域及其内容
- 确定系统边界
  - 深入了解目前拥有的操作型数据
  - 了解方向性需求。 如：决策类型；决策者感兴趣的问题。
  - 确定信息需求，确定数据覆盖范围。
  - 了解源数据（DB系统）的状况 例如“涉及的信息”包含DB中的哪些部分
- 确定主题域
  - 确定系统所包含的主题域
  - 确定主题域的内容：公共码键，代表主题的属性组
  - 主题域之间的联系

商场DW的概念模型设计：确定系统边界

★ 确定系统边界

- 了解方向性需求：把握商场的商品采购情况和销售情况

  分析：

  顾客的购买趋势；商品供应市场的变化趋势；供应商信用等级

  所涉及的信息：

  商品销售数据、采购数据、库存数据、顾客信息、供应商信息

- 了解源数据（DB系统）的状况：

  采购、库存、销售、人事等子系统，

  前三个子系统的数据集合为系统边界

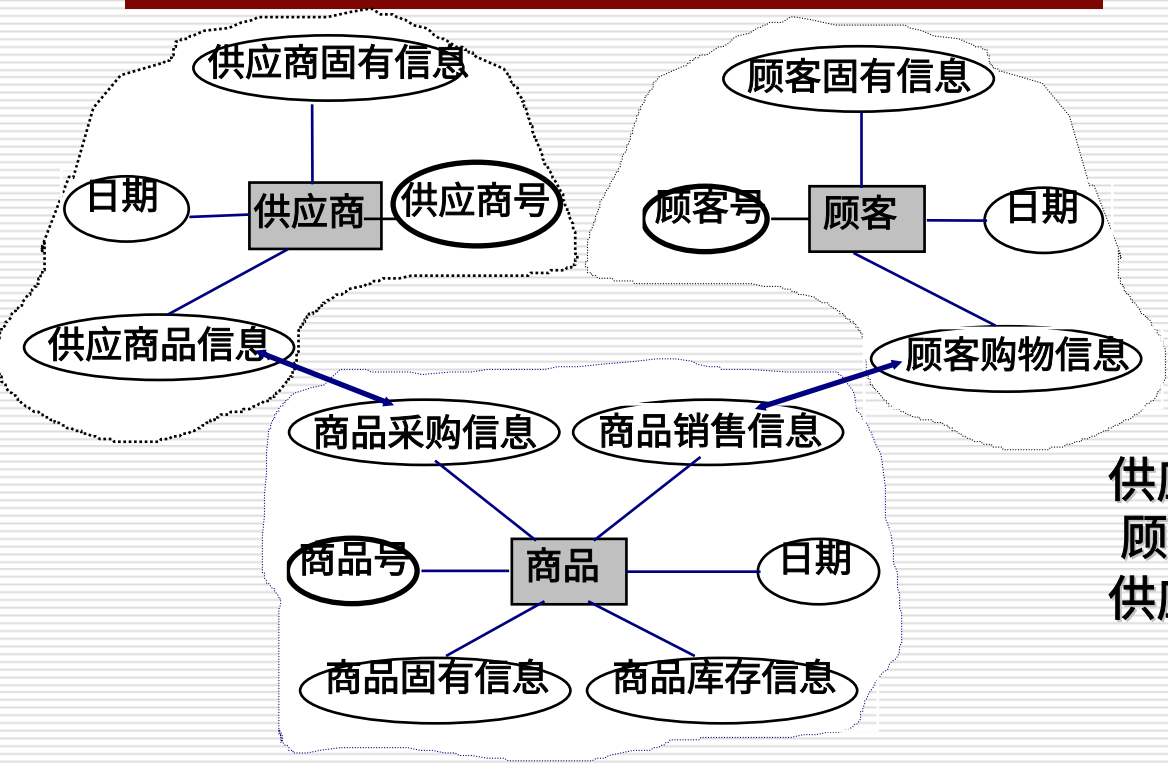
商场DW的概念模型设计：确定主题域

- 确定主题域
  - 确定三个主题：商品、顾客、供应商
  - 每个主题域的内容
  - 主题域之间的联系

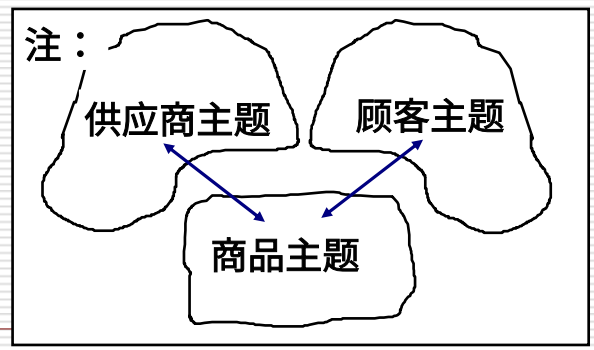
商场DW的概念模型设计：每个主题的内容

主题名	公共码键	属性组
商品	商品号	商品固有信息：商品号，商品名，类别，颜色等; 商品采购信息：商品号，供应商号，供应价，供应日期，供应量等; 商品销售信息：商品号，顾客号，售价，销售日期，销售量等; 商品库存信息：商品号，库房号，库存量，日期等;
供应商	供应商号	供应商固有信息：供应商号，供应商名，地址，电话，供应商类型等; 供应商品信息：供应商号，商品号，供应价，供应日期，供应量等;
顾客	顾客号	顾客固有信息：顾客号，顾客名，性别，年龄，文化程度，住址，电话等; 顾客购物信息：顾客号，商品号，售价，购买日期，购买量等；

商场DW的概念模型设计：主题域之间的联系



供应商与商品之间有供应联系—n：m
顾　客与商品之间有购买联系—n：m
供应商与顾客之间没有直接联系



逻辑模型设计任务

- 对主题域中所包含的内容进行进一步的细化，每一个主题域包含若干个数据组（表）；
- 消除纯粹是操作型的数据：如操作人员、校对人员、客户电话号码等
- 对主题域之间的关系进一步细化为表与表之间的关系，对多对多的关系进行有效分解；
- 增加时间属性：数据仓库中的数据反映历史变化的过程，它是一定时间的数据快照，因此必须包含时间主键；

逻辑模型设计：分析主题域

- 分析主题域
  - 对（E—R）概念模型中的主题进行选取，选取当前实施的主题域。

  由于开发的过程是逐步完成的，DW中的主题可逐步增加
  - 所选主题域大小合适

    大：足以建立一个可应用系统

    小：实施快、方便

    例如：实施“ 商品 ”主题

    大：可以满足经营决策者的初始要求

    小：只有一个主题

逻辑模型设计：粒度层次划分

- 粒度层次的划分（以商场DW为例）
  - 数据量很大，宜采用多重粒度

    商品上千种；商品来源也许多，每月销售数据更多
  - 对商品销售记录：进行销售分析及销售趋势分析

    销售数据：主要关心销售额、销售量

    粒度层次：商品（商品号、商品子类、商品大类）

    时间段（周、月、季、年）

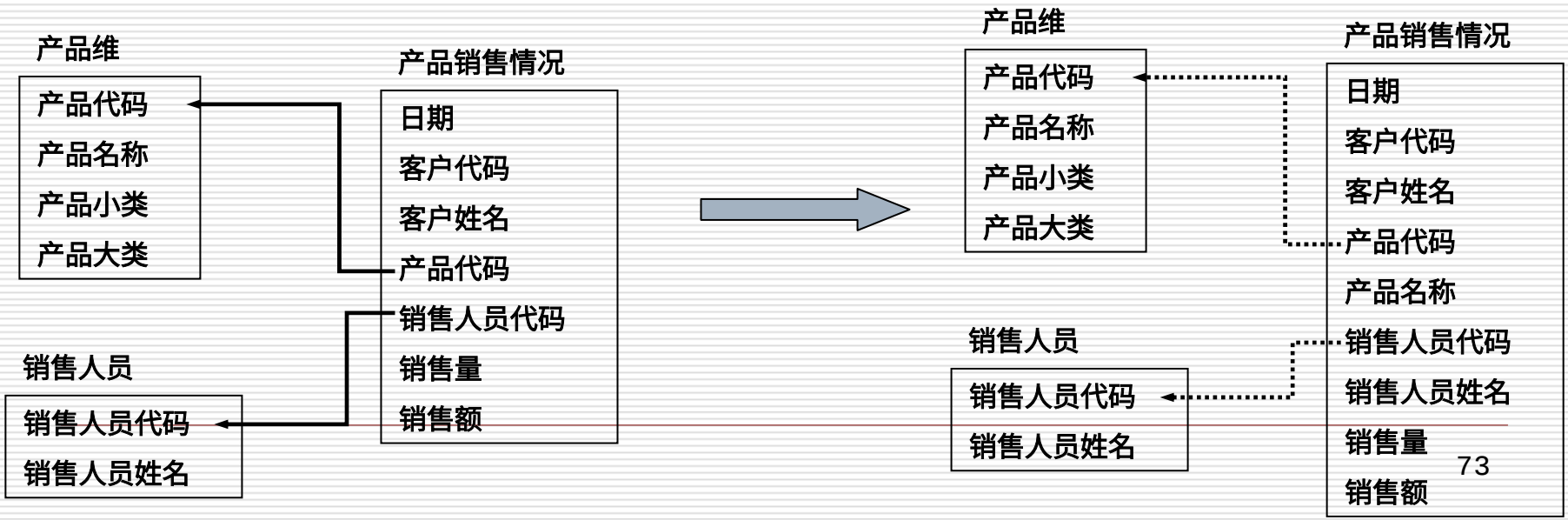
逻辑模型设计：数据分片策略

- 数据分片策略
  - 数据量（不是记录行数）大小
  - 数据分析处理要求：与分析处理的对象有关
  - 分片标准：尽量自然、易实施

    例如：按时间、按地区、按业务类型
  - 与粒度层次划分相适应

逻辑模型设计：引入冗余

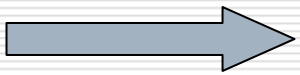
- 将包含在多个表中的有关数据进行合理合并
  - 数据通常一起被查询
  - 提高查询效率
  - 减少Join操作



逻辑模型设计：增加导出数据

- 增加派生数据：对于用户经常需要分析的数据，或者为了提供系统访问的效率，可以适当增加派生数据；（以商场数据为例）

时间
产品名称
单位成本
销售量
销售额

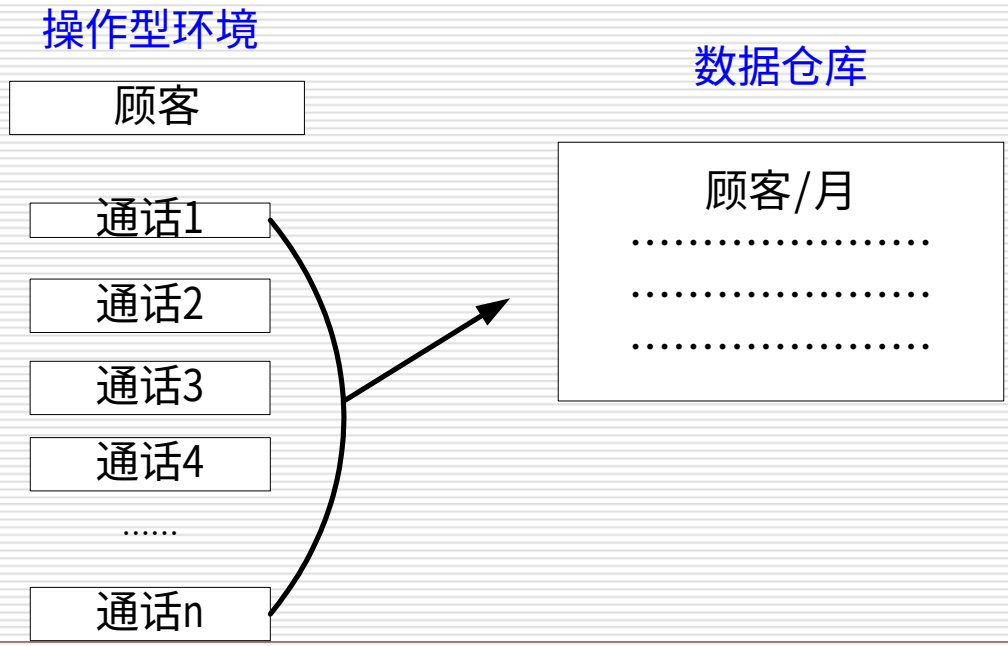


日期
产品名称
单位成本
销售量
销售额
利　润

  - 加入不同级别粒度的汇总数据

简要记录

- 把操作型数据中许多不同的、详细记录组合在一起
- 以聚集形式代表许多条操作型记录



聚集每月的通话记录以提供一个复合的代表性记录

简要记录（续）

- 优点
  - 为最终用户的访问和分析提供了一种紧凑方便的数据组织形式
  - 使数据量降低2～3个数量级
- 缺点
  - 信息的细节程度降低

逻辑模型设计：关系模式定义

- 关系模式定义

  - 由多个表来实现主题（组织主题域的数据）
    各表之间依靠公共码键相联系

  - 表的划分，各个表的关系模式
    细节数据用表来组织；综合数据也用表来组织

例：“商品”主题各个表的关系模式

公共码键：商品号

(1).商品固有信息： 商品表(商品号, 商品名, 类型, 颜色 …) /* 细节级 */

(2).商品采购信息: 采购表1(商品号, 供应商号, 供应日期, 供应价, 供应数量, …) /*细节级 */

采购表2(商品号, 时间段1, 采购总量 ,…)

…

采购表n(商品号, 时间段n, 采购总量,…) /* 时间段不等的综合表 */

(3).商品销售信息: 销售表1(商品号, 顾客号, 销售日期, 售价, 销售量, …) /* 细节级 */

销售表2(商品号, 时间段1, 销售总量, …)

…

销售表n(商品号, 时间段n, 销售总量, …) /* 时间段不等的综合表 */

(4).商品库存信息: 库存表1(商品号, 库房号, 库存量, 日期, …) /* 细节级 */

库存表2(商品号, 库房号, 库存量, 星期, …)

库存表3(商品号, 库房号, 库存量, 月份, …)

…

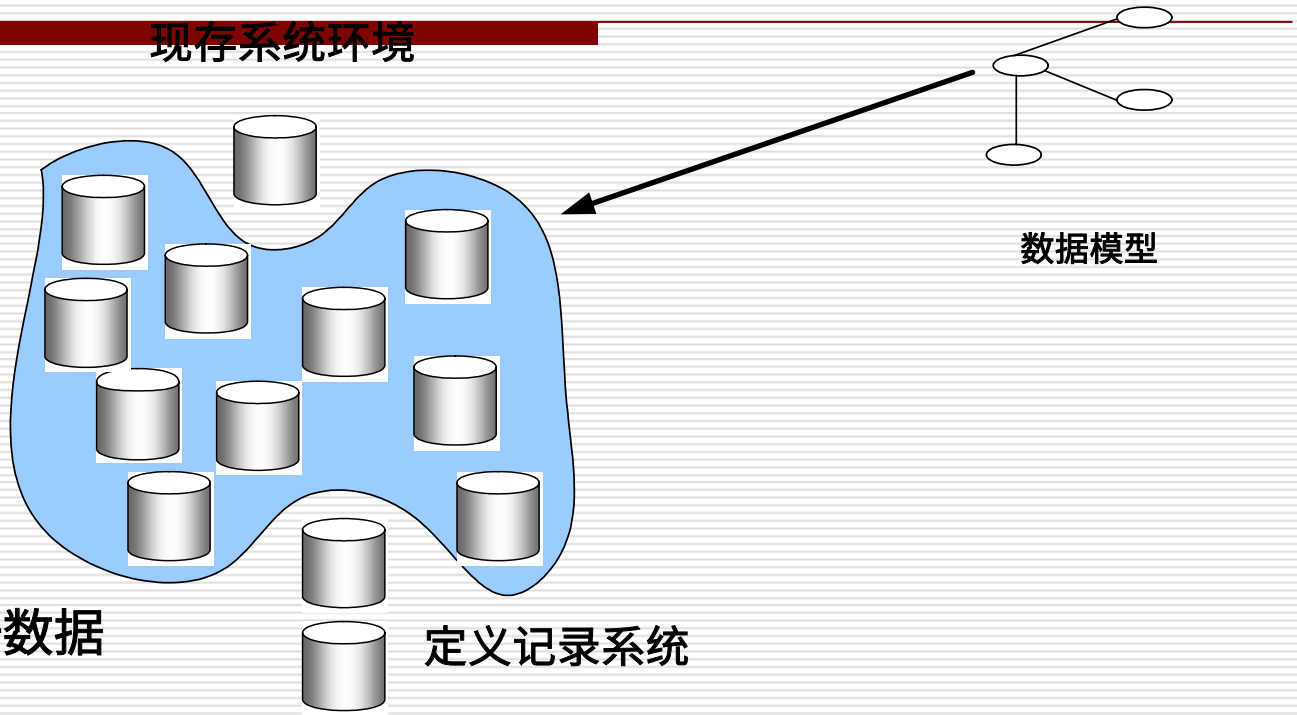
库存表n(商品号, 库房号, 库存量, 年份, …) /* 样本数据粒度形式 */

(5). 其它导出数据: … 其它 …

逻辑模型设计：定义记录系统（I）

- 定义记录系统
  - 根据DW中多个表的关系模式，从源数据中选择最合适的数据作为记录系统
  - 所选数据所在表的关系模式最接近DW中多个表的关系模式
  - 记录系统定义，记入DW的元数据中

定义记录系统（II）



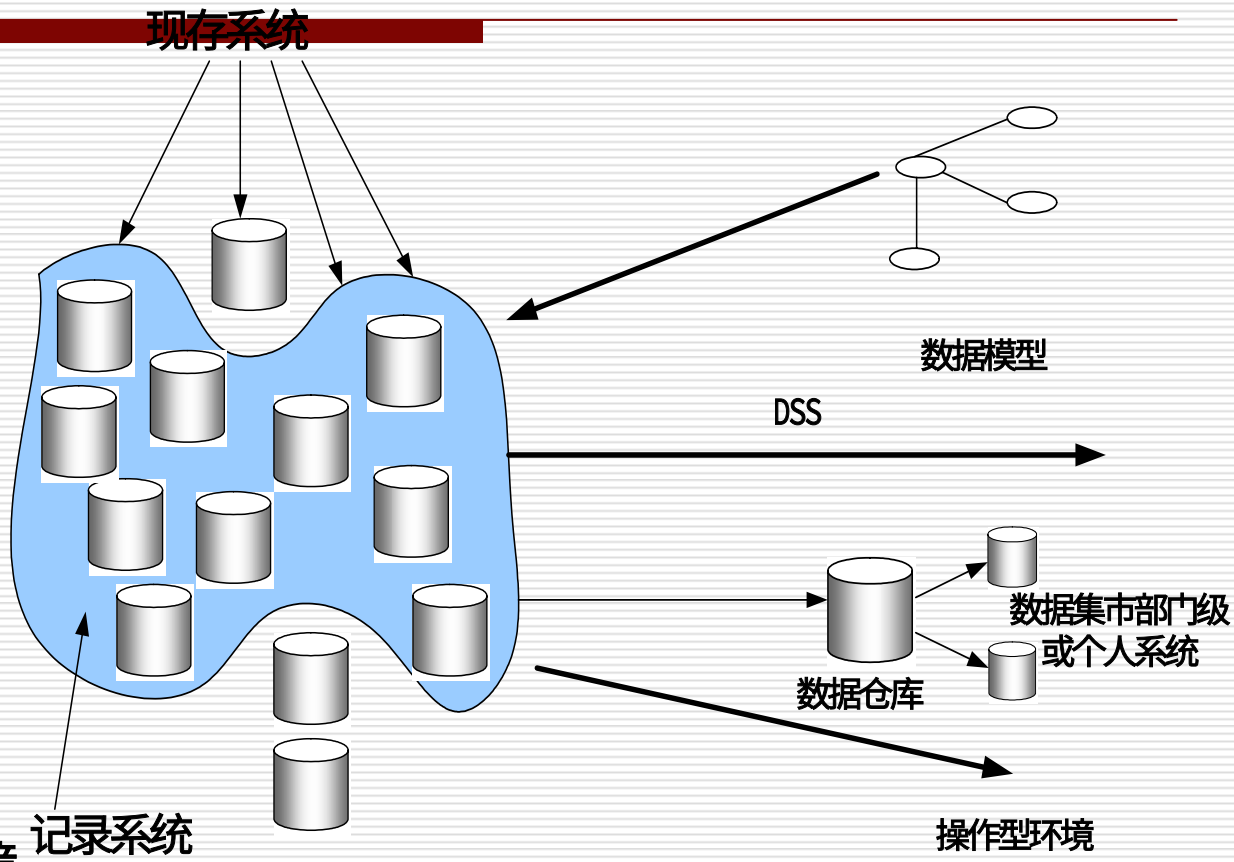
表述数据模型的最好数据

- 最实时
- 最准确
- 最完备
- 与输入现存系统环境的数据源最近
- 最具有结构兼容性

定义记录系统策略方面的考虑

由于以下“变化动因”，
操作型环境处于一种“混乱”状态

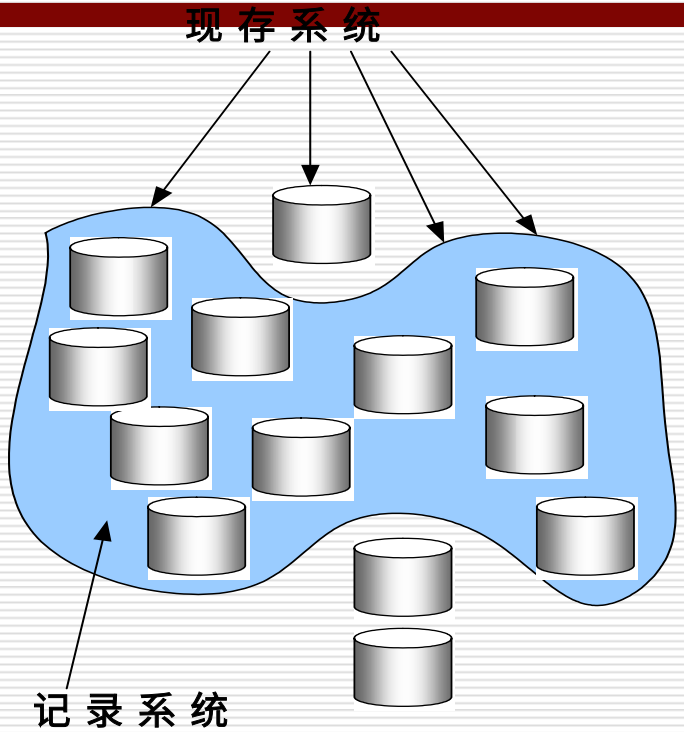
- 系统老化
- 技术老化
- 组织上的巨变
- 大幅改变了需求

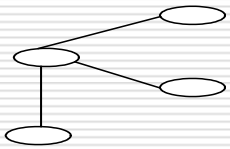


需要重建操作型处理环境

- 数据体系结构设计人员得将变化动因与体系结构的概念结合起来，实现操作型处理环境的重建

重建操作型处理环境





以数据模型为指导 数 据 模 型

- 差异列表
  该数据模型与现存系统不同之处
- 影响分析
  每一个差异项目是如何表明差别的
- 资源估计
  修复差异项目需要多少费用
- 给管理层的报告
  1、要修复什么
  2、估计需要的资源
  3、工序
  4、损失分析

例：“商品”主题定义记录系统

从原有的采购子系统、库存子系统、销售子系统中选择合适的数据：

- 采购子系统：商品号、类别、供应商号、供价、进货日期、进货数量;
- 库存子系统：商品号、商品名、类别、库存量、时间、库房号等;
- 销售子系统：商品号、类别、顾客号、销售价、销售量、销售日期等

记录系统定义在元数据中的描述

主题名	属性名	数据源系统	源表名	源属性名
商品	商品号	库存子系统	商品	商品号
商品	商品名	库存子系统	商品	商品名
商品	类别	库存子系统	商品	类别
商品	供应商号	采购子系统	订单	供应商号
商品	供应日期	采购子系统	订单	日期
商品	供应价	采购子系统	订单细则	单价
商品	顾客号	销售子系统	顾客	顾客号
商品	销售日期	销售子系统	销售	日期
商品	售价	销售子系统	销售	单价
商品	销售量	销售子系统	销售	数量
商品	库存量	库存子系统	库存	库存量
商品	日期	库存子系统	库存	日期
商品	库房号	库存子系统	库房	库房号

物理模型设计

- 任务
  - 确定数据的存储结构
  - 确定索引策略
  - 确定数据存放位置
  - 确定存储分配

物理模型设计：存储结构、索引策略

- 确定数据的存储结构

  选择合适的存储结构时应该权衡的主要因素：

  - 存取时间
  - 存储空间利用率
  - 维护代价

- 确定索引策略

  根据DW中数据不可更新的特点，可以设计多种索引。

  如：广义索引等

物理模型设计：确定存放位置

- 确定数据的存放位置
  - 根据重要程度、使用频率和响应时间对数据进行分类，不同类的数据可以存放在不同的存储设备中
  - 考虑冗余存储、数据序列等方法。

物理模型设计：确定存储分配

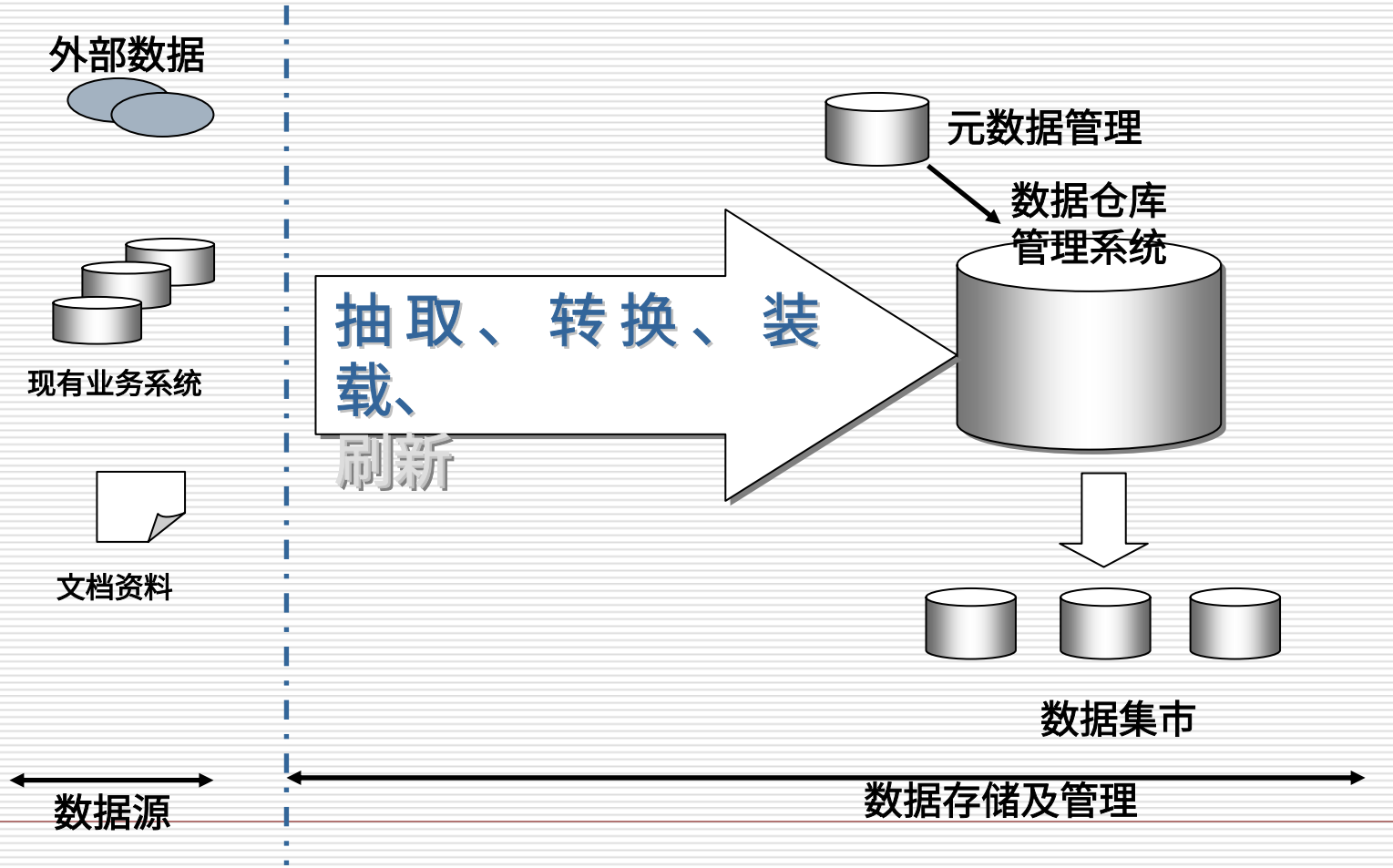
- 确定存储分配：选择存储分配的参数（设定），进行优化

  如：存储块的大小、缓冲区的大小和个数

数据仓库的生成

- 任务：
  - 设计接口
  - 装入数据
- 设计接口
  - 抽取所需的完整的数据
  - 数据基于时间的转换
  - 为以后的数据追加作好准备（如：加时间标签、建立前后映象文件）
- 装入数据：运行接口程序，将数据装入到DW中

数据仓库的生成



转换和集成的复杂性

- 在整个数据仓库建设中，设计接口要花去大约80%精力
- 复杂性体现：
  - 迁移时的技术难点：
    - DBMS的变化，即记录系统是在一个DBMS中，而数据仓库在另一个DBMS中
    - 操作系统的变化，记录系统在一个操作系统中，而数据仓库在另一个操作系统中
    - 记录系统涉及多个DBMS和/或操作系统时，需要将源自不同DBMS和操作系统的数据合并起来
    - 在Web记录中获取基于Web的数据，一旦捕获到数据以后如何才能将数据放入数据仓库中使用
    - 基本数据格式的变化，如某个环境中的数据是用ASCII码存储的，而数据仓库中的数据是用EBCDIC存储的

转换和集成的复杂性（续I）

- 复杂性体现（续）：
  - 选择数据十分复杂

    为判定一个记录是否需要抽取，需要对其他文件记录进行协调查询
  - 输入关键字需要重建并进行转换

    简单情况：关键字加入时间成分；

    复杂情况：需要重新散列或重新构造

转换和集成的复杂性（续II）

- 复杂性体现(续）：
  - 对输入数据进行清理

    取值范围检查、交叉记录检验等
  - 数据文件进行合并
    - 存在多个数据源时，加载到数据仓库要进行文件合并
    - 不同输入文件使用不同的关键字结构，合并程序必须进行关键字解析
    - 因为多个输入文件的顺序可能不相同甚至互不相容，输入文件需要重新排序
  - 会产生多个输出结果

    创建数据仓库时，会产生不同综合层次的结果

转换和集成的复杂性（续III）

- 复杂性体现(续）：
  - 需要提供缺省值
    输出值没有对应的输入源
  - 刷新时的效率
    区分需要与不需要抽取的操作型数据
  - 经常需要进行数据的汇总
    多个操作型输入记录合并成单个“简要”数据仓库记录
  - 重命名操作的跟踪
    数据元素从操作型环境到数据仓库转移过程中，可能被改名字，因而必须生成记录这些变化的文档
  - 进行数据格式转换
    输入记录具有不常见的或非标准的格式

转换和集成的复杂性（续IV）

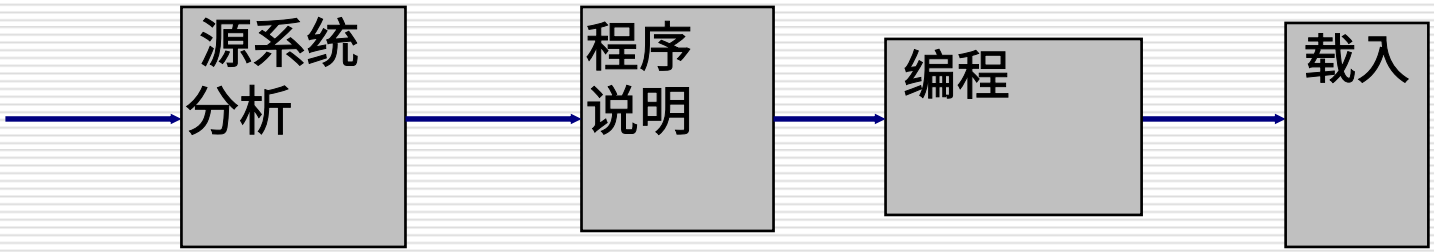
- 复杂性体现(续）：
  - 大容量输入问题

    并行装载、并行读出，必须引入特殊的设计方法
  - 加入时间元素

    数据仓库反映对信息的历史需求，当操作型数据载入到数据仓库时应加入时间元素
  - 必须符合企业数据模型

    数据源的应用程序往往是很久以前设计的经过多次维护，但没有相关文档，并且未考虑与其他应用的集成，因而建数据仓库时必须考虑企业数据模型所体现的有关规则与限制

转换与集成的步骤



- 考虑数据从操作型环境到DSS环境中的映射问题
- 以程序说明的形式将接口形式化，用于把数据从操作型环境引入数据仓库中
- 所有编程的标准活动
- 执行前面开发的程序，把数据载入数据仓库

源系统分析

- 目的：实现数据仓库需要的源数据及时、完备、准确、接近来源和易于访问，并且与数据仓库需要的数据一致
- 考虑问题
  - 当数据从操作型环境转移到数据仓库环境时的键码结构/键码处理
  - 属性
    - 如果有多个来源可供选择应如何处理
    - 如果没有来源可供选择应如何处理
    - 当数据被选择传输给DSS环境时必须做何种变化—编码/解码、转换等
  - 从数据的当前值如何创建时间变量
  - 结构—如何在操作型数据结构基础上创建DSS数据结构
  - 关系—操作型的关系如何在DSS环境中体现

程序规范说明

- 目的：使数据的抽取和集成尽可能高效和简单
- 考虑问题
  - 判定什么样的操作型数据需要抽取
    - 操作型数据是否有时间戳
    - 是否是增量文件
    - 是否有系统日志或审计日志可以使用
    - 现有的源代码和数据结构是否可以改变以产生一个增量文件
    - 前映像和后映像文件是否需要清除
  - 抽取后如何存储
    - DSS数据是否预分配、预格式化
    - 是否增加数据
    - 是否替换数据
    - 是否在DSS环境进行更新

编程

- 目的：生成的代码将是高效、有文档说明、灵活、准确和完备的
- 考虑问题
  - 开发伪代码
  - 编码
  - 编译
  - 检查代码
  - 各种形式的测试—单元测试、重点测试

载入

- 目的：生成一个可访问的、可理解的、能够为DSS群体的需要服务的数据仓库
- 考虑问题
  - 载入的频率
  - 移出过期数据
  - 管理多层次粒度
  - 刷新活样本数据（如果活样本数据表已经建立）

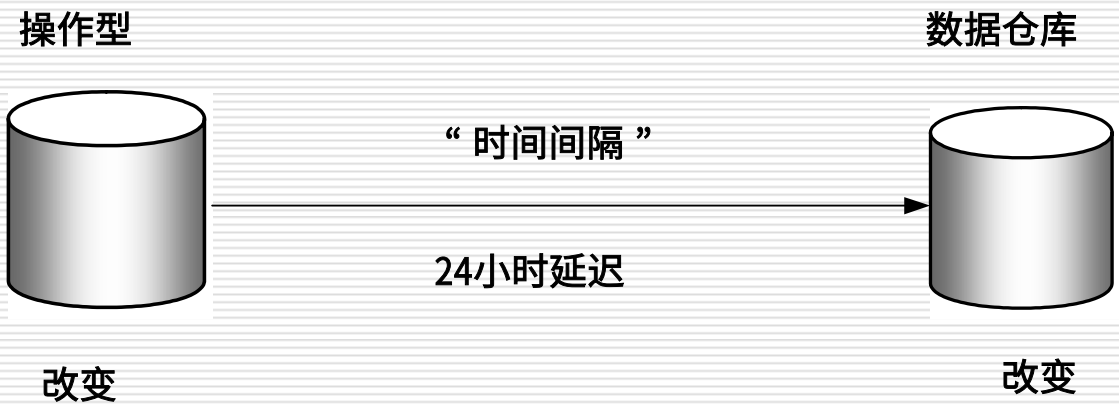
数据仓库记录的触发机制

- “事件/快照”交互
  - 重要活动
  - 时间推移

数据仓库中的每个快照都是由某一事件触发的

数据周期

- 定义：从操作型环境数据发生变化起到该变化反映到数据仓库中所用的时间。

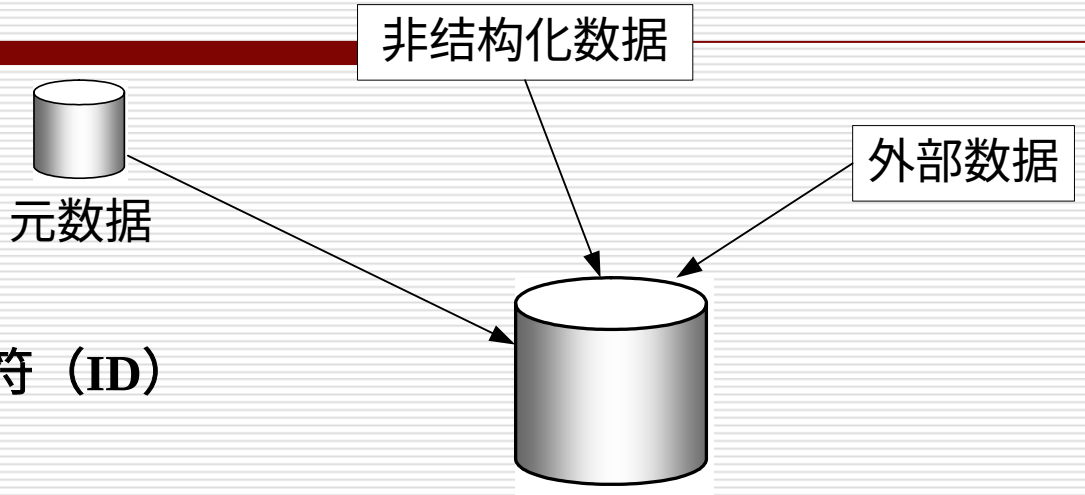


从发现操作型环境中的改变到这个改变在数据仓库得以体现需要经过至少24小时延迟——“ 时间间隔 ”

数据仓库接口实现的两种方法

- 程序员自己动手设计处理集成的接口程序
- 购买ETL（抽取/转换/装载）软件
  - 产生源代码
  - 产生参数化运行模块

对外部数据/非结构化数据，元数据起着新的作用



- 文件标识符（ID）
- 进入日期
- 描述
- 来源
- 分类
- 索引字
- 移出日期
- 长度
- 相关引用

数据仓库的设计复查

- 复查的时间
  - 在数据仓库环境中，一个主要主题域设计好了，并准备加入到数据仓库环境中时，就应进行复查
- 负责设计复查的人员
  - 数据管理员（DA）
  - 数据库管理员（DBA）
  - 程序员
  - DSS分析人员（最重要的参与者）
  - 除DSS分析人员外的最终用户（最重要的参与者）
  - 计算机管理部门
  - 系统支持人员
  - 管理人员

数据仓库的设计复查（续）

- 复查的主题
  - 任何可能会导致失败的的设计、开发、项目管理或者应用问题；即任何有碍成功的障碍
- 复查的结果
  - 对各种问题管理的评价和对进一步行动的建议
  - 有关系统在设计中的位置以及复查时间的文档
  - 一个“行动要目”表，阐明详细的目标和行动步骤，作为复查过程结果的特定目标和行为
- 复查管理—复查过程的领导，不能是项目的管理者或开发者
  - 有利于用新的眼光，从外部角度观察系统
  - 有利于提出建设性的批评

数据仓库设计复查典型问题的举例1

- 从数据模型中找出了多少主要主题？有多少是正在实现的？有多少是已全面实现的？有多少是由正在被复查的项目来实现的？有多少是能在不远的将来得到实现的？

解答：通常，数据仓库环境一次只实现一个主题，最初的少数几个主题应该被当作实验一样，到了后来，早期开发工作中的所学到的经验，就能在主题的实现中应用。

数据仓库设计复查典型问题的举例2

- 已经标识出的主要主题是否都已经划分到较低的细节级？
  - 是否标明了各个关键字？
  - 各个属性是否已经明确？
  - 关键字和属性是否已组合起来？
  - 不同数据分组之间的关系找出来没有？
  - 每一组的时间变化特性找出来没有？

解答：对于数据仓库环境来说，需要有一个数据模型作为它的中心。在正常情况下，这种数据模型有三个层次：一个用于标识实体和关系的高层模型；一个用于标识关键字、属性和关系的中层模型；以及一个可以在此进行数据库设计的低层模型。然而，在开始建立DSS环境时，并不是所有的数据都需要先模型化到最低层，但至少高层模型应该建好。

数据仓库设计复查典型问题的举例3

- 是否已确定操作型环境的记录系统？
  - 每一个属性的数据源确定没有？
  - 是否已经找到某一个或另外一个属性会成为数据源的条件？
  - 如果某一个属性没有数据源，是否确定了它的默认值？
  - 数据仓库环境中的那些数据属性的属性值的公用度量确定没有？
  - 数据仓库环境中的那些数据属性的共同编码结构定义好没有？
  - 数据仓库环境中的那些数据属性的公共关键字结构确定没有？记录系统的关键字在哪些地方不符合DSS的关键字结构？找到转换途径没有？

解答：数据模型建立好以后，记录系统就定好了。记录系统通常存在于操作型环境中，记录系统代表了支持数据模型的现存数据中最好的数据源。集成问题在定义记录系统时是一个非常重要的因素。

数据仓库设计复查典型问题的举例4

- 从操作型记录系统中抽取数据到数据仓库环境的过程的频率确定没有？当前抽取过程如何从上次的抽取过程中识别出操作型数据的变化？
- 通过查看时戳数据？
- 通过改变操作型应用代码？
- 通过查看日志文件？或是某个动作记录文件？
- 通过查看某个变化文件？
- 通过比较“前”和“后”映像？

解答：抽取过程的频率之所为成为一个问题，是由于刷新中所需要的资源、刷新过程的复杂性、以及数据及时刷新的需要等原因造成的。数据仓库的可用性常常与数据仓库的刷新频率有关。

从技术角度而言，在抽取过程中判定应该扫描哪些数据是最复杂的问题之一。在有些情况下，需要从一个环境中传到下一个环境中的操作型数据是相当明确的。在另外一些情况下，根本就无法知道应该对哪些数据进行扫描，并将其作为载入数据仓库环境的候选数据。

数据仓库设计复查典型问题的举例5

- DSS环境中通常包含多少数据量？如果数据量很大话，那么
  - 是否应指定多重粒度级？
  - 是否应该对数据进行压缩？
  - 是否应进行定期数据清除？
  - 是否需要将数据移到准在线存储器？以什么频率转移？

解答：除了抽取过程所处理的大量数据外，设计者自己需要考虑数据仓库环境中实际的数据量。通过对数据仓库环境中数据量的分析，直接地产生了数据仓库环境中数据粒度问题，并因此可能会出现多重粒度级。

数据仓库设计复查典型问题的举例6

- 数据仓库环境中的数据应有怎么样的粒度级？
  - 高级别？
  - 低级别？
  - 多重粒度级？
  - 要不要进行轮转综合？
  - 是否有一个真实档案数据层？
  - 是否有一个活样本数据层？

解答：显然，在数据仓库环境中，最重要的设计问题是数据的粒度和采用多重粒度级的可能性。简言之，如果数据仓库环境的粒度级已经正确地设计好了，那么所有其它问题就变得简单明了了；如果数据仓库环境的粒度级没有正确地设计好，那么所有其它问题将会变得复杂而繁重。

数据仓库设计复查典型问题的举例7

- 在数据仓库环境中，将会识别出各个主题域间的哪些关系？这些关系的实现：
  - 能不能用外部关键字？
  - 能不能利用人工关系？
  - 数据仓库环境中，建立和维护关系时需要哪些额外工作？

解答：数据仓库设计者要做的最重要的设计决策之一，就是该如何实现数据仓库环境中的数据之间的关系。在数据仓库中，数据关系的实现方式几乎不可能套用操作型环境中的数据关系的实现方式。

数据仓库设计复查典型问题的举例8

- 主要主题该如何划分？(按年？按地域？按功能单元？按生产线？)对数据进行的分区的精细程度如何？

解答：考虑到数据仓库环境所固有的数据量以及数据用途的不可预测性，必须要求把数据仓库数据在物理上划分为小单元，以便能对它们进行独立地管理。我们要面对的设计问题不是是否应该进行分区的问题，而是该如何进行分区的问题。一般地，分区是在应用层而不是系统层进行的。

对分区策略进行复查的时候，应注意以下问题：

- 当前的数据量
- 未来的数据量
- 数据的当前用途
- 数据的未来用途
- 仓库中其它数据的分区问题
- 其他数据的用途
- 数据结构变动性

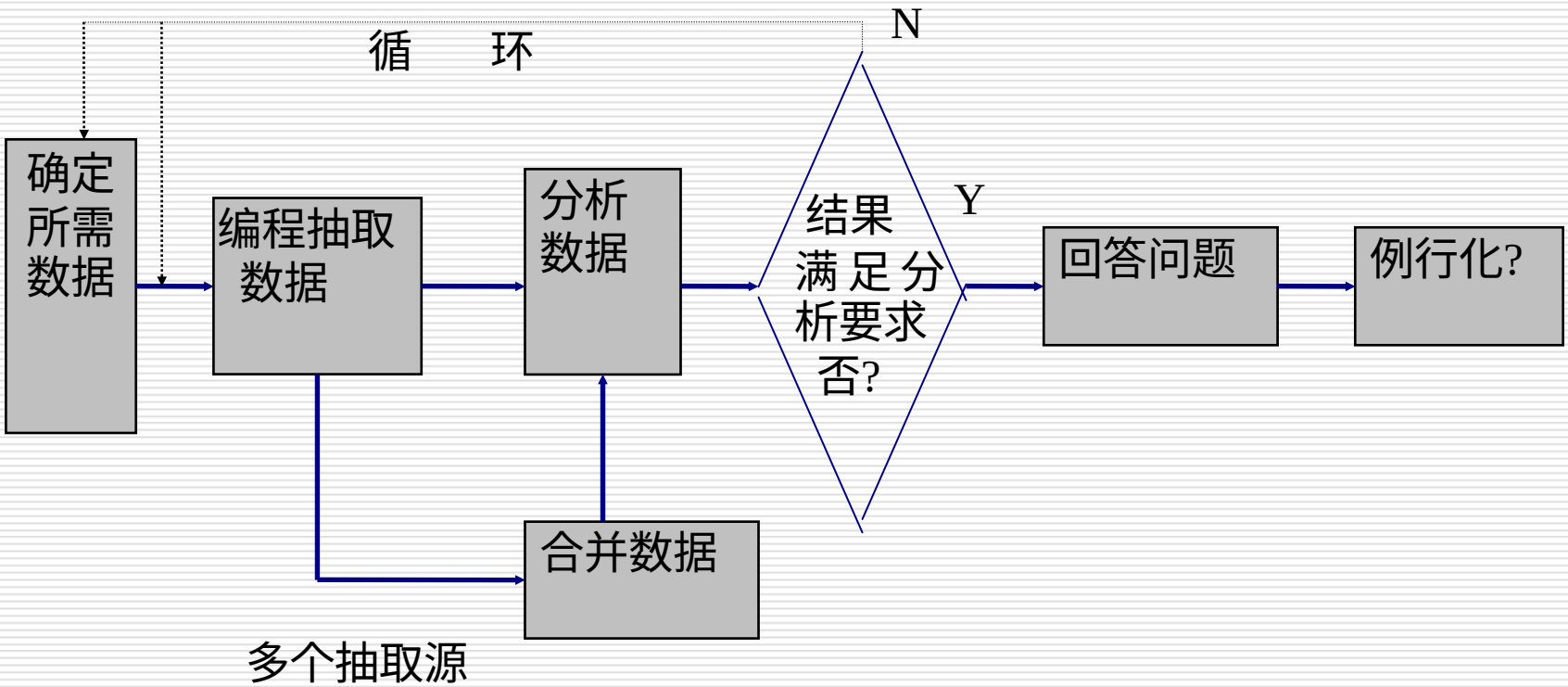
数据仓库的使用和维护

- 任务：
  - 建立DSS应用
  - 理解需求、完善系统

- 建立DSS应用
  - DSS应用开发的特点
    - 从数据出发
    - 不断循环过程（启发式开发）
  - DSS应用分类
    - 例行分析处理 — 部门级
    - 启发式分析处理 — 个人级（即席分析处理）

数据仓库的使用和维护（续I）

- DSS应用开发的步骤。



数据仓库的使用和维护（续II）

- 理解需求、完善系统及DW维护
  - 理解需求、完善系统
    - 增加主题（如：在商场DW中增加“ 顾客 ”主题）
    - 调整粒度层次
    - 增加属性（如对“ 商品 ”主题增加“ 商品档次 ”属性）
    - ……
  - DW维护
    - 数据装入（刷新当前详细数据，将过时数据转化为历史数据）
    - 清除不再使用的数据
    - 追加数据（确定刷新频率）
    - 管理元数据

Inmon提出的设计步骤