

数 据 仓 库

(二)

数据仓库

1. 引言
2. 从数据库到数据仓库
3. 数据分析与数据仓库
4. 数据仓库的四大特色
5. **数据仓库的基本结构** ✓
6. 数据仓库的设计
7. 联机分析处理(OLAP)
8. 多维建模 (**Dimensional Modeling**)
9. 数据仓库的应用

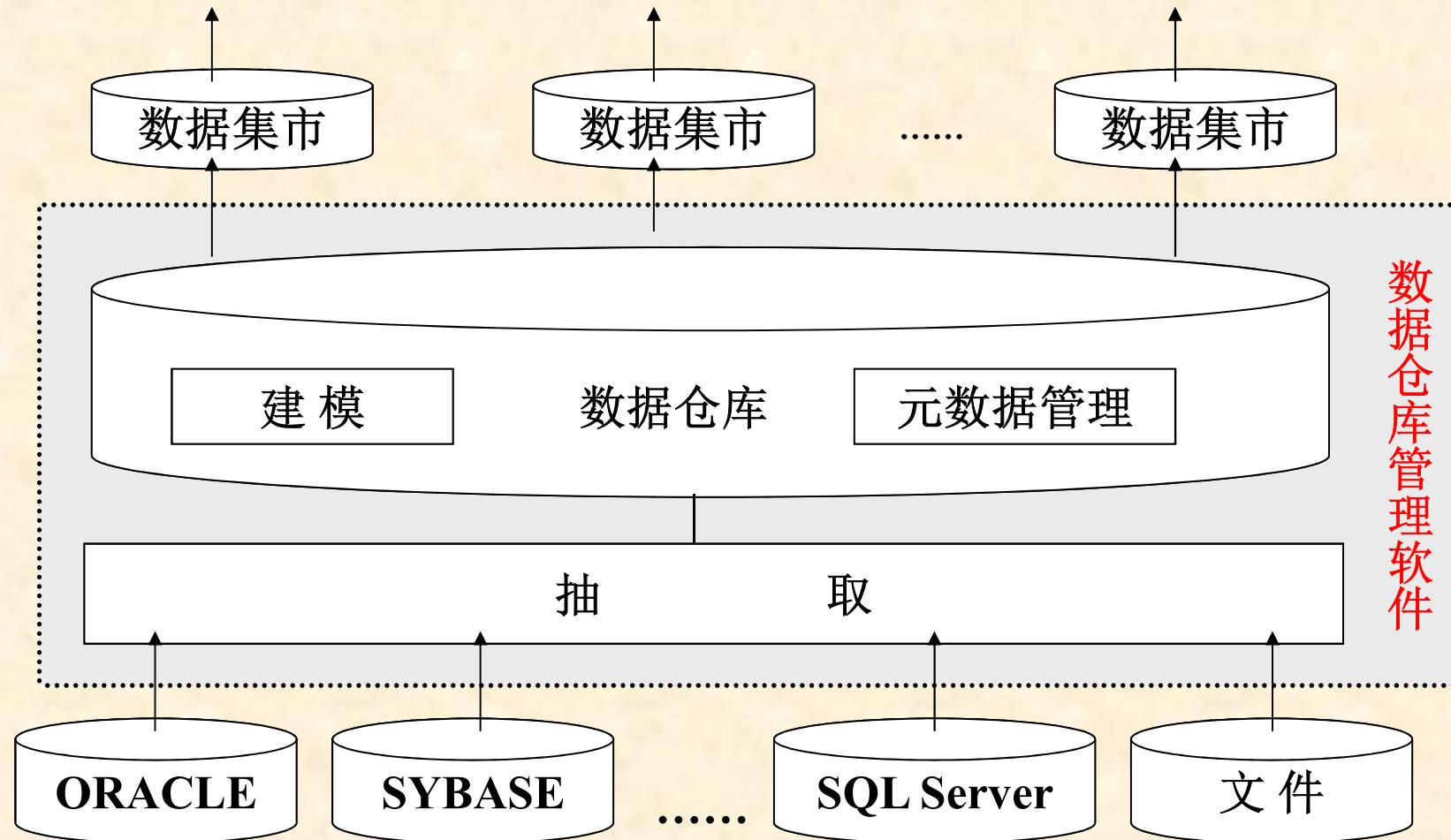
5 数据仓库的基本结构

- 数据的层次划分
- 体系结构
- 关键技术
- 数据仓库与数据集市

5.1 数据的层次划分

- 在一个完整的数据仓库系统中，其数据存在于三个层次，它们是：
 - 1) 数据源
 - 2) 数据仓库
 - 3) 数据集市（Data Mart）
- 三者之间通过数据仓库管理软件联系起来构成一个完整的数据体系。

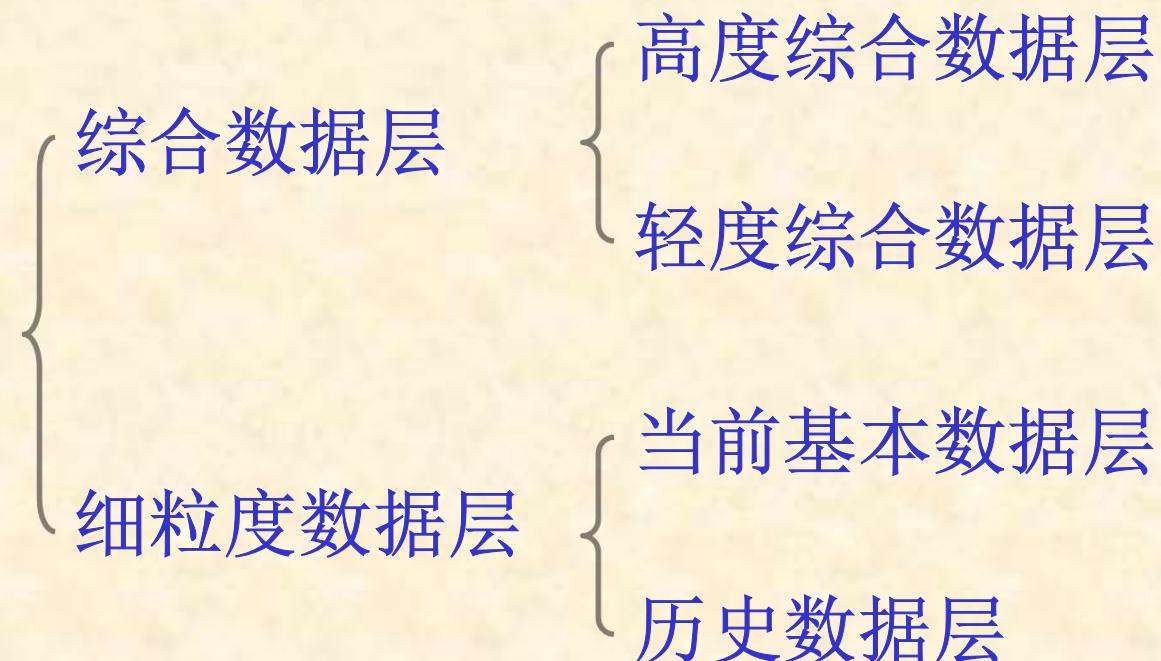
5.1 数据的层次划分

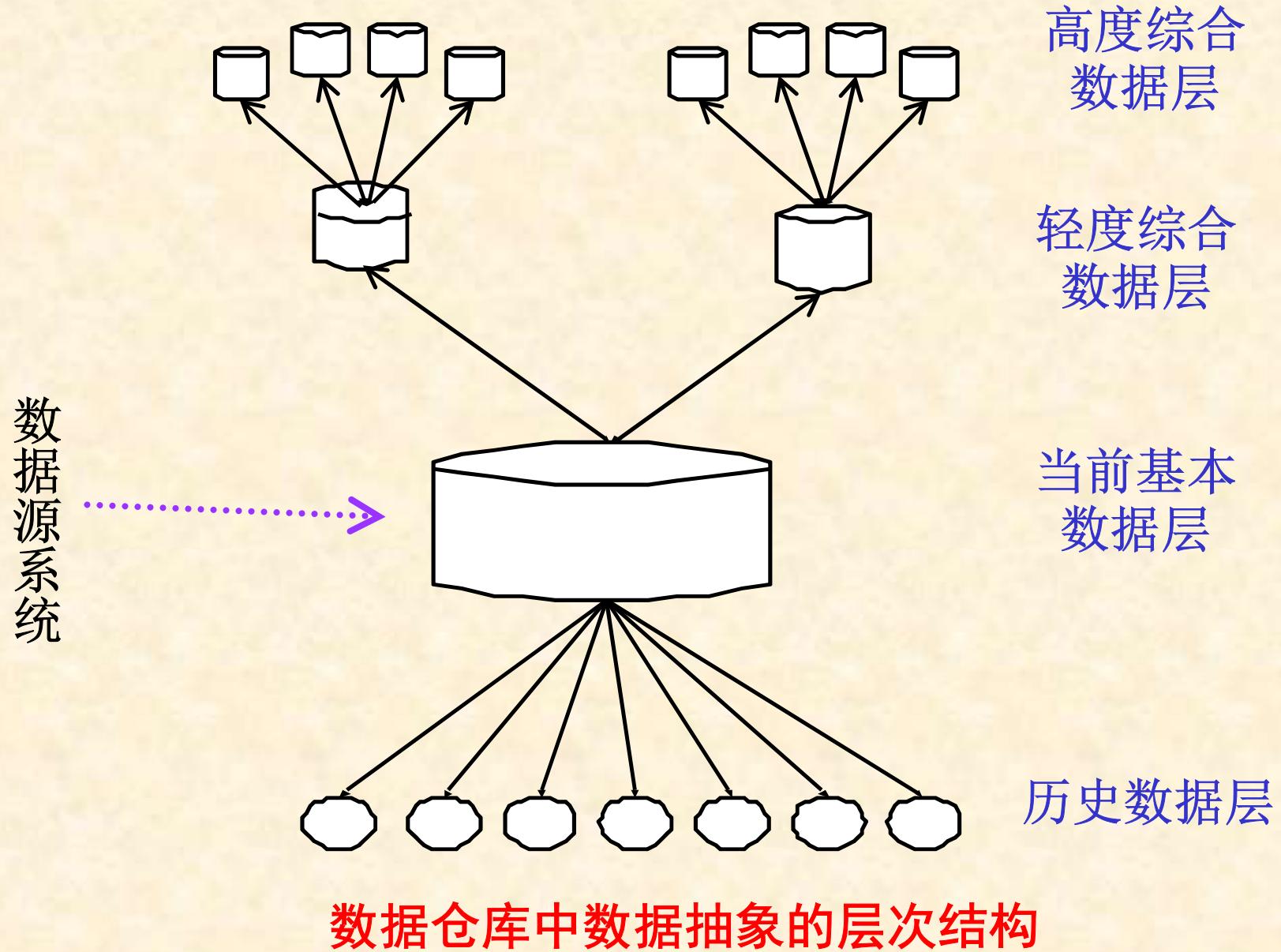


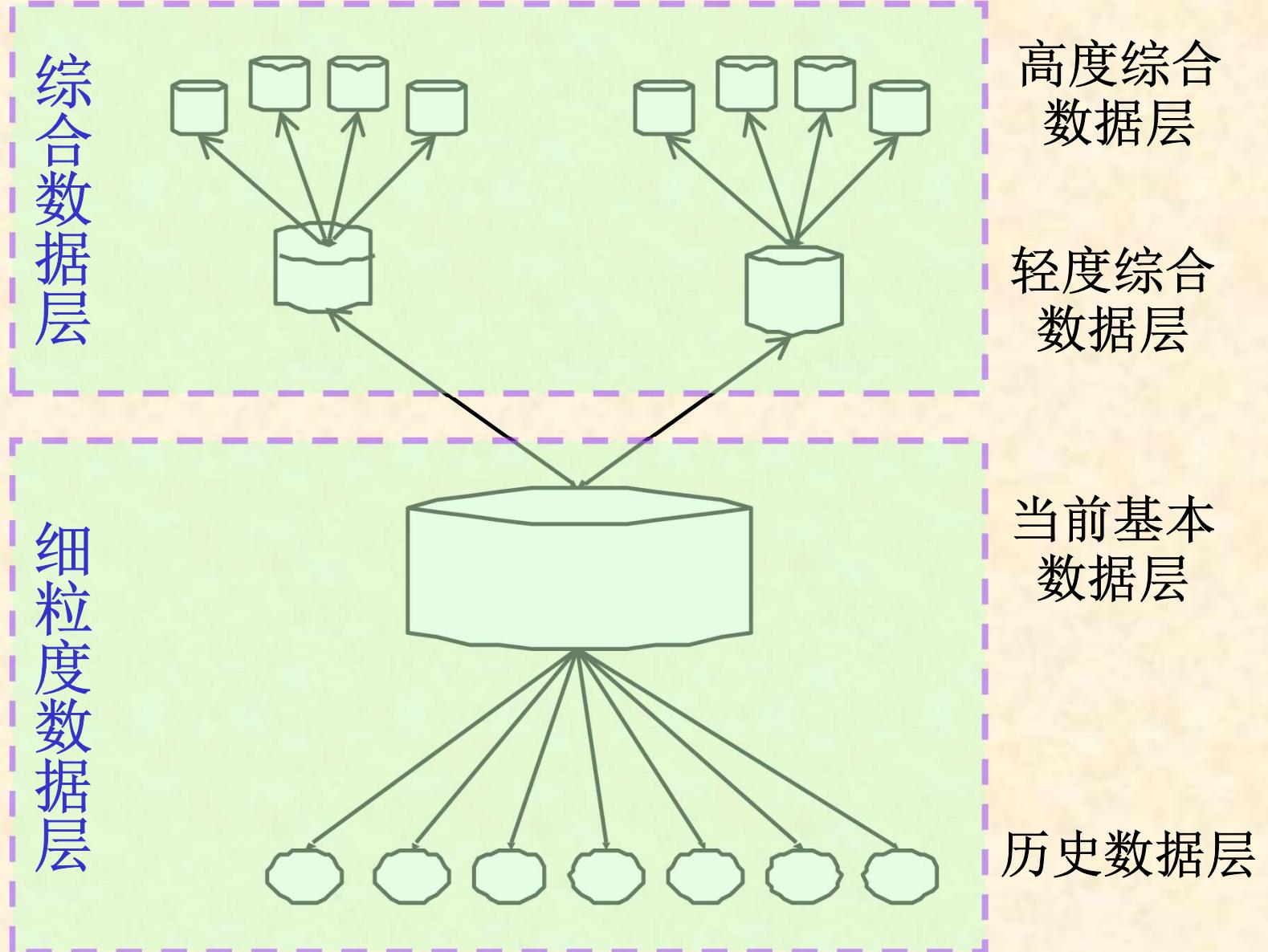
数据仓库中数据层次划分示意图

5.1 数据的层次划分

➤ 数据仓库中的数据，又可以被划分为如下的层次结构：

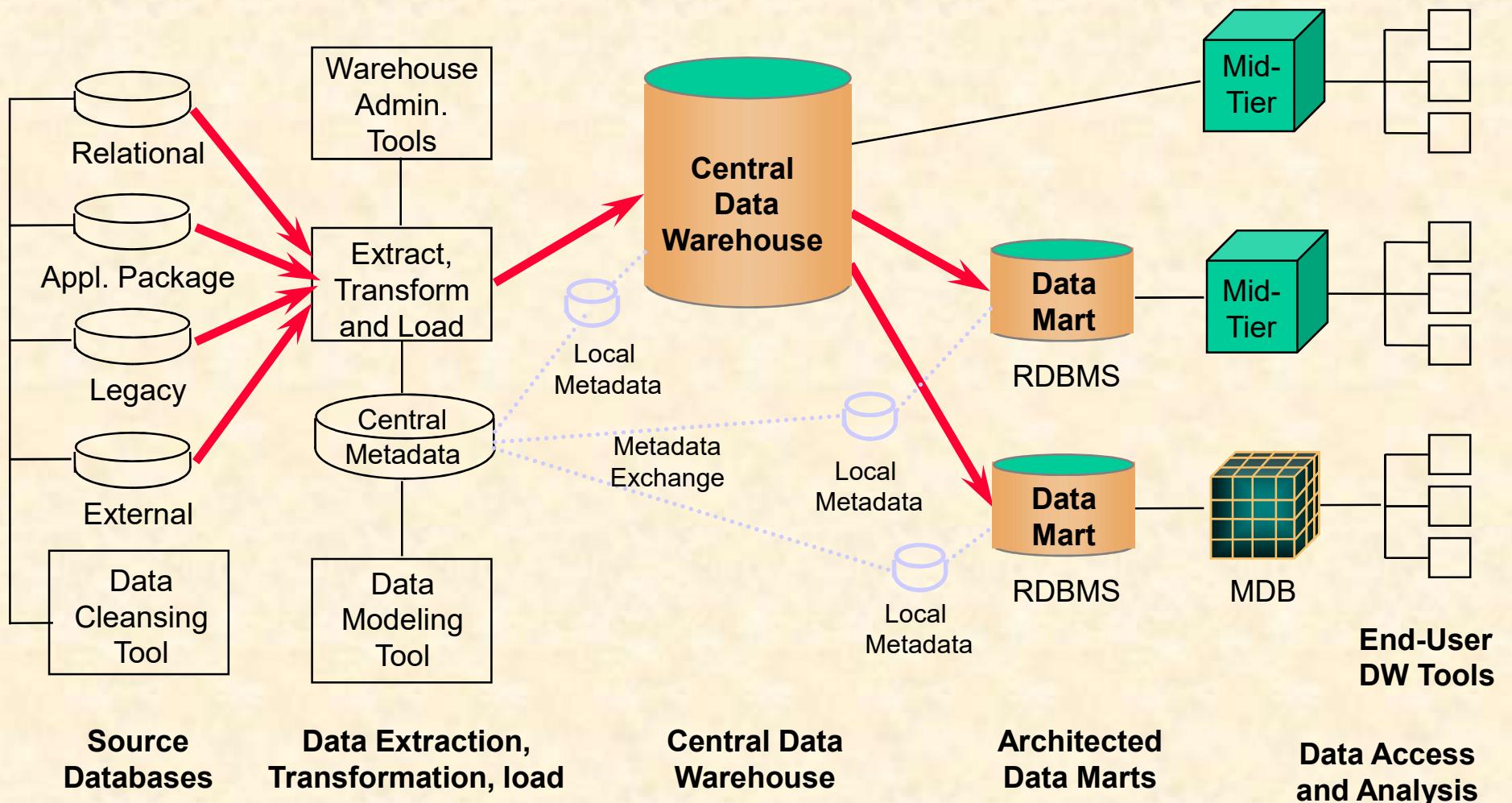




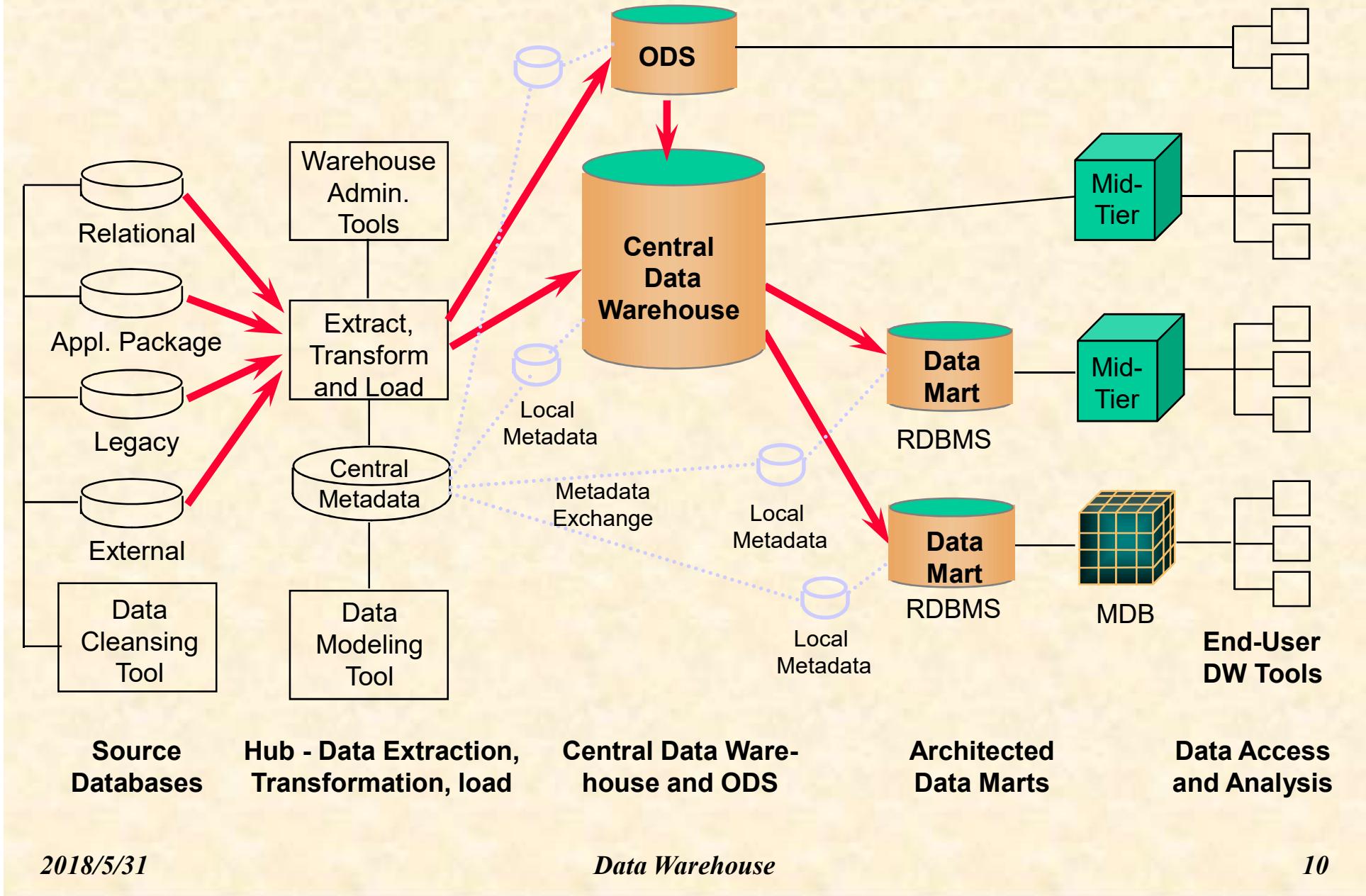


数据仓库中数据抽象的层次结构

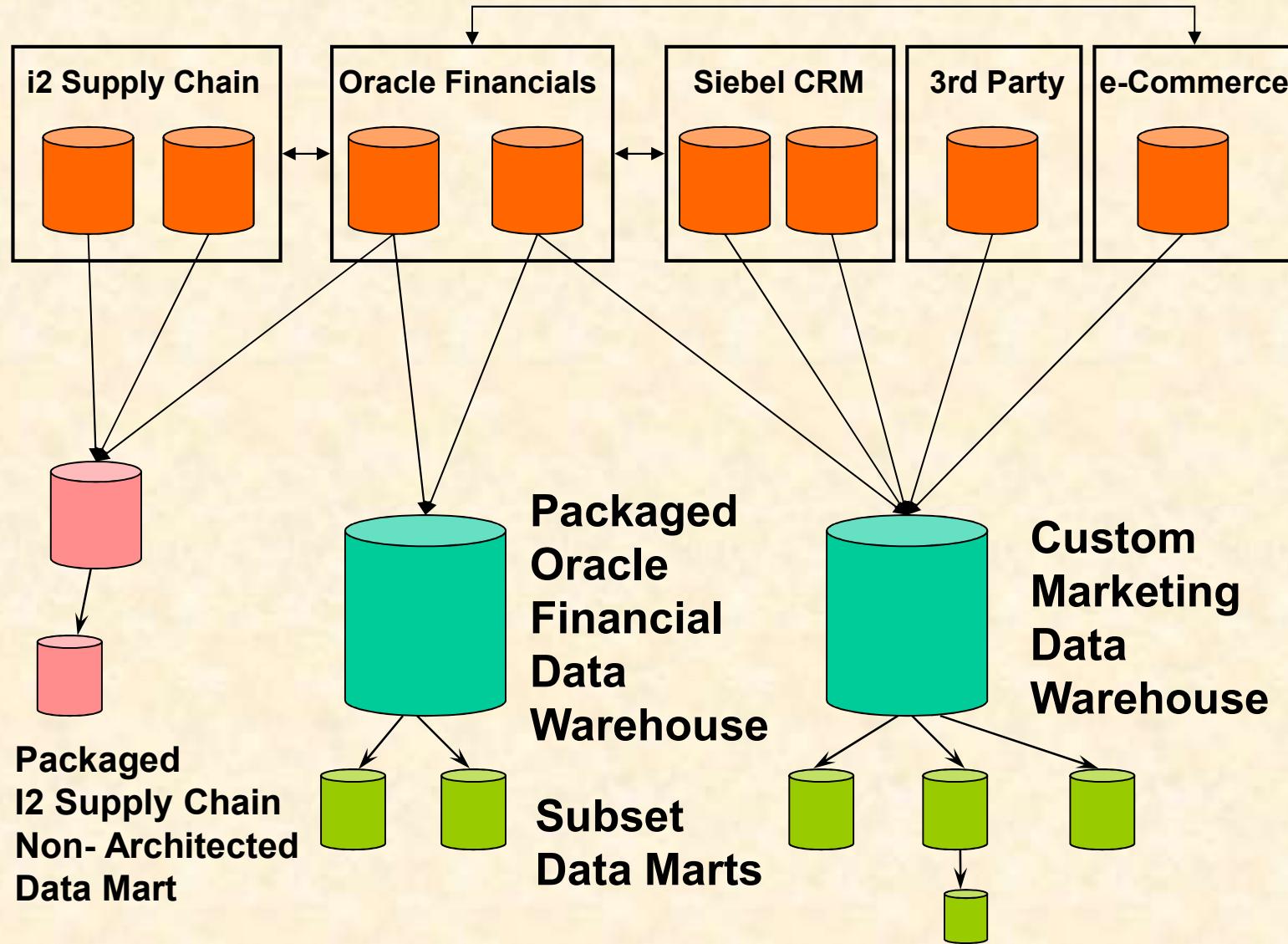
5.2 体系结构 [Pieter ,1998]



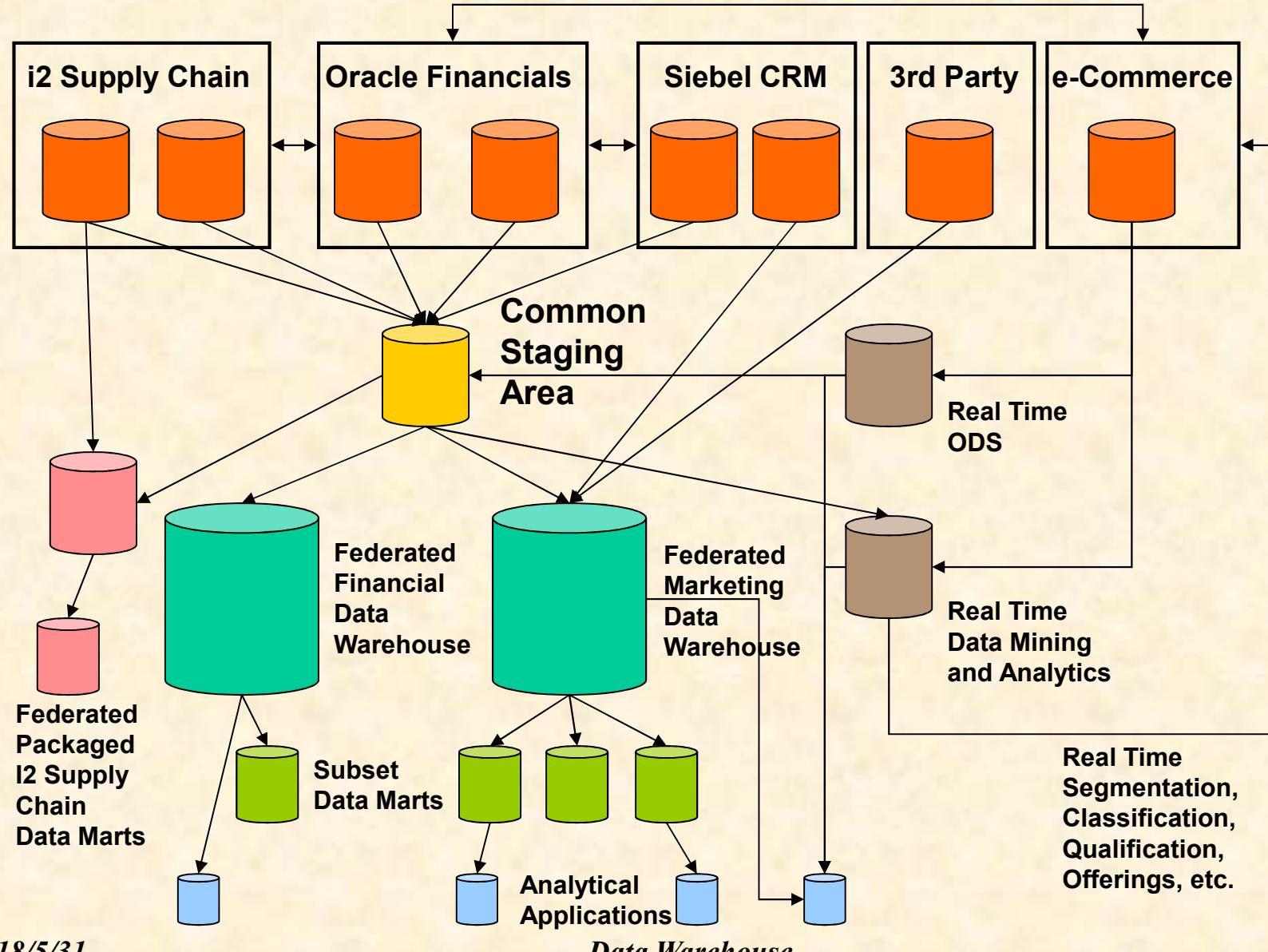
带ODS的体系结构



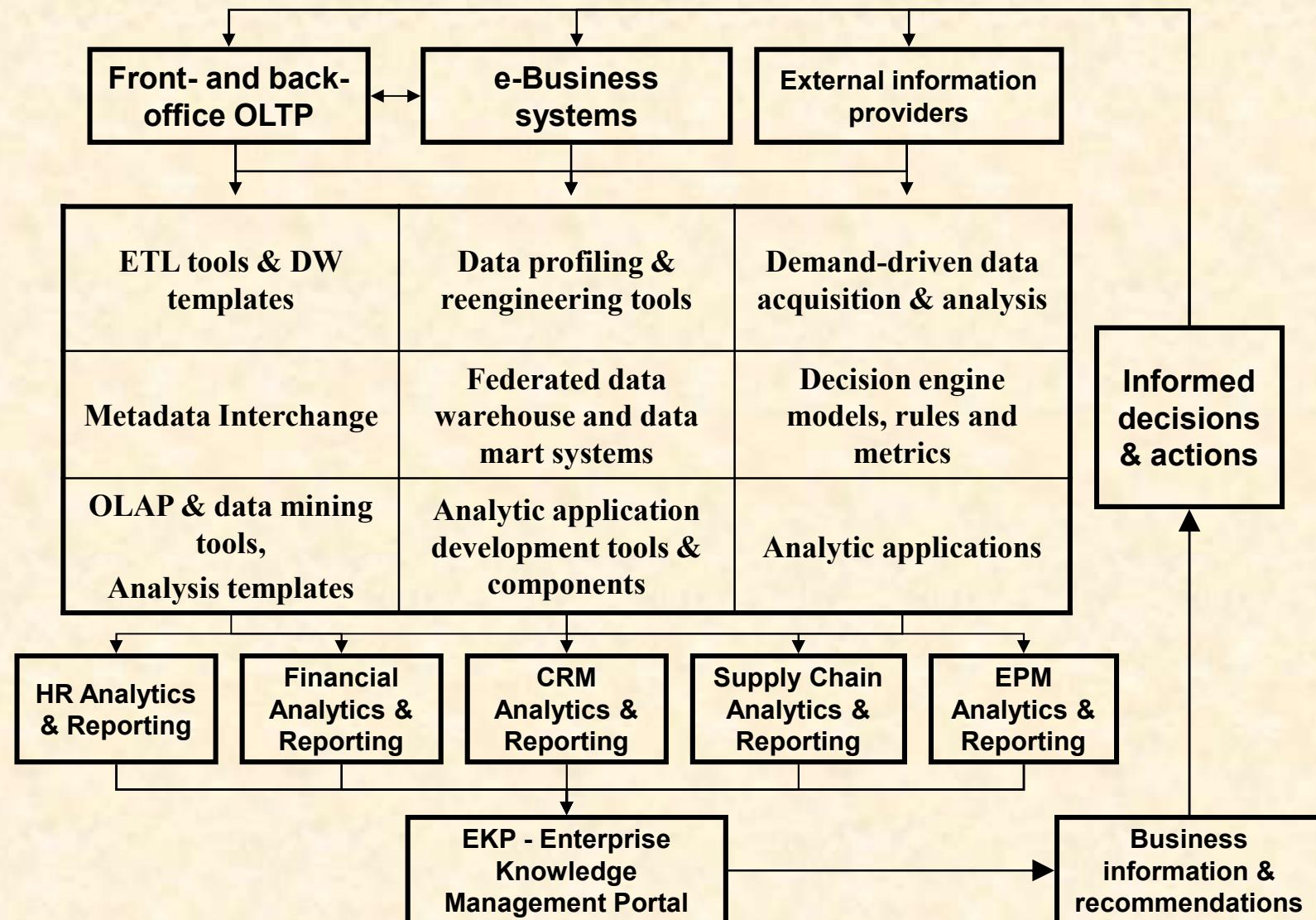
现实环境—异质性 [Douglas Hackney ,2001]



联合型数据仓库/数据集市体系结构

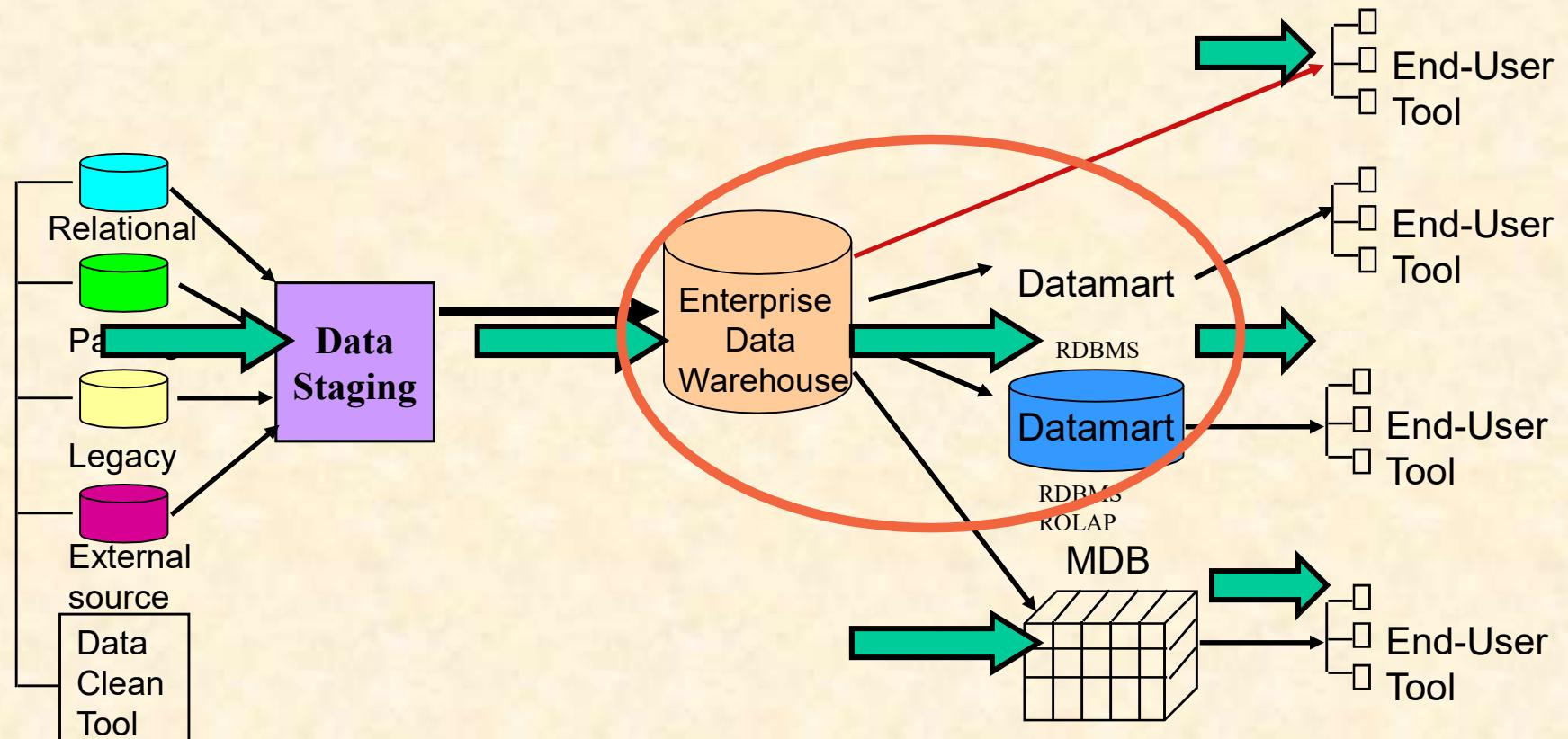


闭环的联合型BI体系结构



数据仓库的焦点问题-数据的获得、存储和使用

- ❖ 数据仓库和集市的加载能力至关重要
- ❖ 数据仓库和集市的查询输出能力至关重要



5.3 关键技术

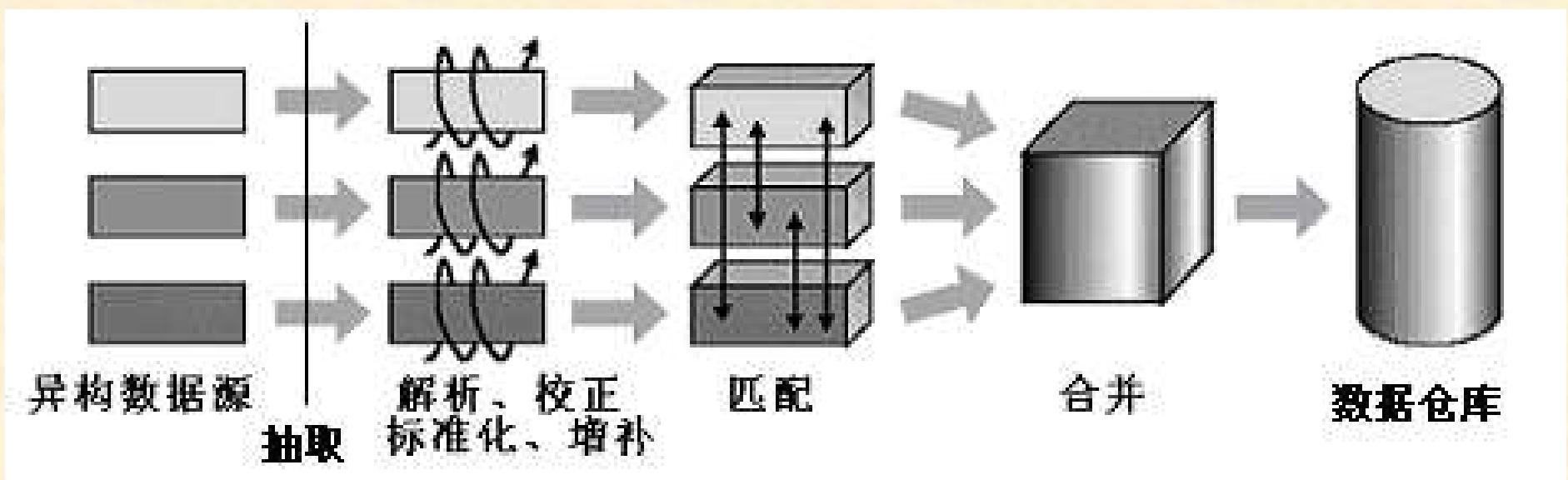
➤ 数据仓库的关键技术

- 1) 数据的抽取
- 2) 数据仓库管理
- 3) 结果的展现

5.3 关键技术

1) 数据的抽取

- 数据的抽取是数据进入仓库的入口。由于数据仓库是一个独立的数据环境，它需要通过抽取过程将数据从联机事务处理系统、外部数据源、脱机的数据存储介质中导入数据仓库。



5.3 关键技术

- 多数据源
 - 数据仓库的数据来源于多个数据源。
 - 不同格式的数据: 由于企业在长期事务处理过程中随数据库管理系统本身发展，形成了企业内从简单到复杂、从小型到大型的各种，其中有大型关系数据库、对象数据库、桌面数据库、各种非格式化的数据文件等。
 - 不同的数据操作平台
 - 不同的物理位置
- 数据源可以是递归的
 - 数据仓库的数据源可以是另外一个数据仓库(或数据集市)或OLAP服务器。

5.3 关键技术

➤ 常见的数据源有：

- ✓ 流行的关系数据库系统: **Oracle,Sybase,SQL Server,DB2**等
- ✓ 面向对象数据库系统: Objectstore等
- ✓ 传统的桌面数据库系统: **foxbase,foxpro**等
- ✓ 文件系统中的数据文件: **UNIX,WINDOWS**等
- ✓ 其它数据源: **word,excel**等

➤ 数据的抽取: 数据抽取软件

5.3 关键技术

2) 数据仓库管理

- 数据仓库的组织管理方式决定了它有别于传统数据库的特性，同时也决定了其对外的数据表现形式。
 - 数据量很大
 - 并行处理
 - 针对决策支持查询的优化
 - 支持多维分析的查询模式

3) 结果的展现

- 报表，OLAP，数据挖掘
- 多通过第三方的工具软件来完成

5.3 关键技术

2. 数据仓库管理

- 数据仓库中的数据
 - 企业内部各个部门当前及其历史上的细节性业务数据
 - 为了进行分析决策操作而生成的分析型数据
- 对数据仓库中数据的管理
 - 需要借助成熟的数据库技术对其进行存储管理
 - 利用改造过的关系数据库系统来组织和管理数据仓库中的数据。

5.3 关键技术

- 数据仓库管理层一般由如下几部分组成：
 - 数据仓库管理系统
 - 数据仓库建模
 - 数据抽取与刷新
 - 元数据管理

5.3 关键技术

2. 数据仓库管理 - 数据仓库管理系统

- 一般采用传统的关系数据库管理系统或其变种来实现数据仓库管理系统。
 - 如可用Oracle, Sybase, SQL Server等作适当改进即可成为数据仓库管理系统。
- 数据仓库数据的**安全、归档、备份、维护、恢复**等工作，也需要利用数据库管理系统(DBMS)的现有功能来实现。

5.3 关键技术

2. 数据仓库管理 - 数据仓库建模

➤ 数据仓库建模

- 建立数据仓库的模式。

➤ 数据仓库的模式结构

- 如同数据库的模式设计一样，我们也要设计建立数据仓库的数据模式。
- 如果采用关系数据库系统作为数据仓库管理的工具，则数据仓库的模式结构在形式上与关系模式一样。

➤ 数据仓库的建模过程

- 数据仓库的建模方式有别于传统的关系数据库建模，需要有独立的数据仓库建模工具作为数据仓库管理工具的一部分。

5.3 关键技术

2. 数据仓库管理 - 数据的抽取与刷新

➤ 数据抽取

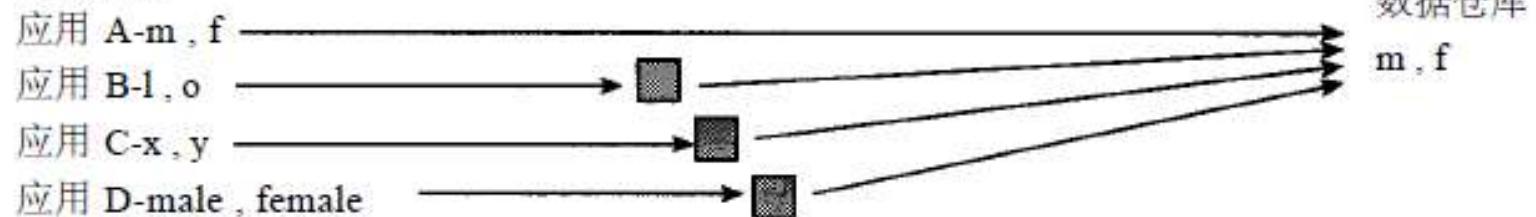
- 对数据源中数据通过网络进行抽取，并经加工、转换、综合后形成数据仓库中的数据。

➤ 数据刷新

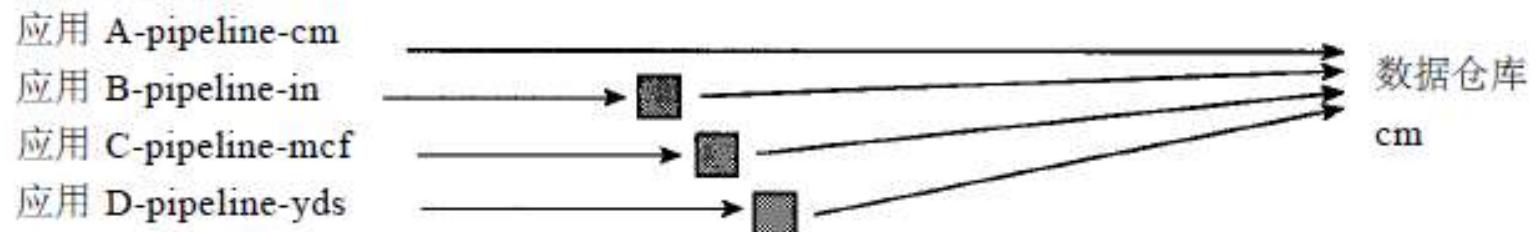
- 对数据仓库中数据的修改、删除和增加
- 数据刷新的过程与抽取类似，但刷新的数据量往往小于抽取的数据量。
- 由于仅需要对修改过的数据进行刷新，因而其实现难度与复杂性要大于数据抽取。

5.3 关键技术

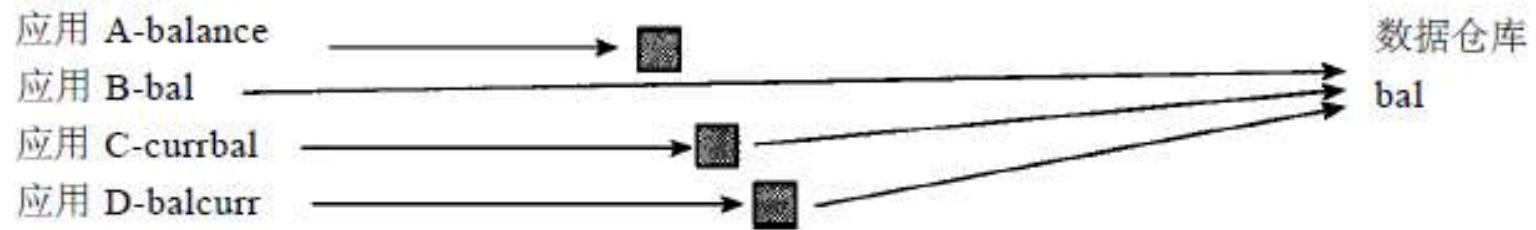
编码转换



度量单位转换



字段转换



5.3 关键技术

2. 数据仓库管理 - **数据抽取**

➤ 数据抽取的重要性

- 必须屏蔽底层数据的结构复杂性和物理位置的复杂性
- 能够实现对数据仓库中数据的自动刷新
- 对数据仓库的元数据和数据进行维护

➤ 数据抽取的实现方法

- 通过通用的数据库接口程序或协议从中抽取数据
- 编制特殊的数据抽取函数进行数据抽取

5.3 关键技术

2. 数据仓库管理 - 数据刷新

➤ 数据仓库系统必须能够感知到在OLTP数据库中数据的变化情况，并及时有效地把这些变化反映到数据仓库中去，以使得数据仓库中的数据能真实地反映实际情况，因此必须对数据仓库进行数据刷新。一般数据刷新的方法包括：

- ✓ 时间戳
- ✓ DELTA文件
- ✓ 建立映象文件
- ✓ 日志文件

5.3 关键技术

2. 数据仓库管理 - 数据刷新方法 - 时间戳

➤ 适用情况

- 若数据库中的记录有时间属性，则可根据OLTP数据库中的数据有无更新，以及在执行更新操作时数据的修改时间标志来实现数据仓库中数据的动态刷新。

➤ 缺点

- 大多数数据库系统中的数据并不含有时间属性。

5.3 关键技术

2. 数据仓库管理 - 数据刷新方法 - DELTA文件（增量文件）

➤ 适用情况

- 有些OLTP数据库的应用程序在工作过程中会形成一些 DELTA文件以记录该应用所作的数据修改操作，可根据该 DELTA文件进行数据刷新。

➤ 优点

- 采用此方法可避免对整个数据库的对比扫描，具有较高的 刷新效率。

➤ 缺点

- 这样的应用程序并不普遍，修改现有的应用程序的工作量 又太大。

5.3 关键技术

2. 数据仓库管理 - 数据刷新方法 - 建立映象文件

➤ 实现方法

- 在上一次数据刷新后对数据库作一次快照
- 在本次刷新之前再对数据库作一次快照
- 比较两个快照的不同，从而确定数据仓库的数据刷新操作。

➤ 缺点

- 需要占用大量的系统资源
- 可能较大地影响原有数据库系统的性能

5.3 关键技术

2. 数据仓库管理 - 数据刷新方法 - 日志文件

➤ 实现方法

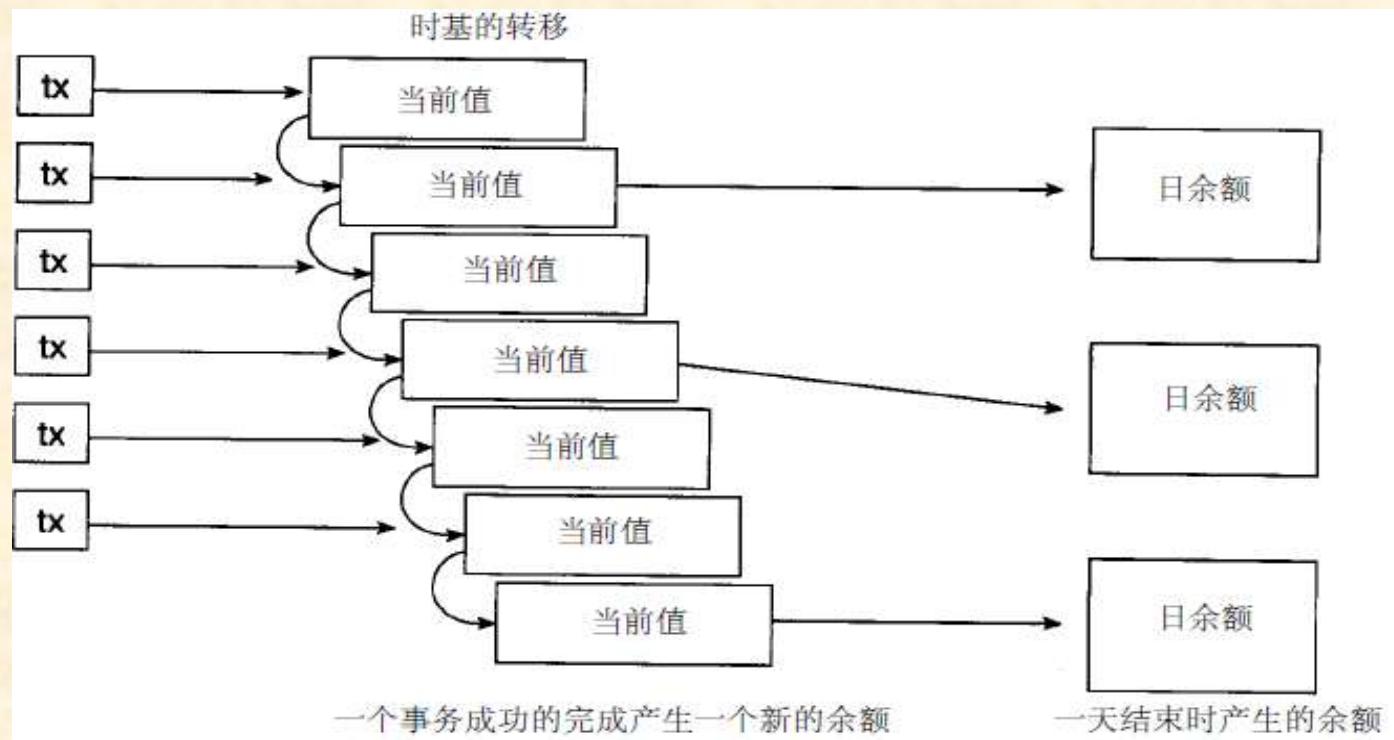
- 一般OLTP数据库都有日志文件，可根据OLTP数据库的日志信息来实现数据仓库的数据刷新。

➤ 优点

- 日志是OLTP数据库的固有机制
 - 不会影响原有OLTP数据库的性能
 - 具有比DELTA文件和建立映象文件更高的刷新效率
- 在一个数据仓库系统中，可以同时采用上述的四种数据刷新方式，以满足不同数据源的数据刷新需要。

5.3 关键技术

2. 数据仓库管理 – 时基变化



操作型数据通常是当前值数据，在被访问的时刻其精度是有效的，而且是可更新的。数据仓库中的数据是不能更新的，必须附有时间元素。

元数据库及元数据管理

- 元数据：是用来描述数据的数据。
 - 要有效的管理数据仓库，必须设计一个描述能力强、内容完善的元数据。
 - 描述关于源数据的说明，包括源数据的来源、源数据的名称、源数据的定义、源数据的创建时间等对源数据进行管理所需要的信息。
- 描述和定位数据组件、它们的起源及它们在数据仓库进程中的活动；关于数据和操作的相关描述(输入、计算和输出)。
 - 从哪个特定的数据源而来，经过哪些转换、集成过程。
- 可用文件存在元数据库中。

元数据库及元数据管理

- 元数据分类：技术元数据；商业元数据；数据仓库操作型信息。*-[Alex Berson etc, 1999]*
- 技术元数据
 - 包括为数据仓库设计人员和管理员使用的数据仓库数据信息，用于执行数据仓库开发和管理任务。包括：
 - 数据源信息
 - 转换描述（从操作数据库到数据仓库的映射方法，以及转换数据的算法）
 - 目标数据的仓库对象和数据结构定义
 - 数据清洗和数据增加的规则
 - 数据映射操作
 - 访问权限，备份历史，存档历史，信息传输历史，数据获取历史，数据访问，等等

元数据库及元数据管理

➤ 商业元数据

– 给用户易于理解的信息，包括：

- 主题区和信息对象类型，包括查询、报表、图像、音频、视频等
- Internet主页
- 支持数据仓库的其它信息，例如对于信息传输系统包括预约信息、调度信息、传送目标的详细描述、商业查询对象等

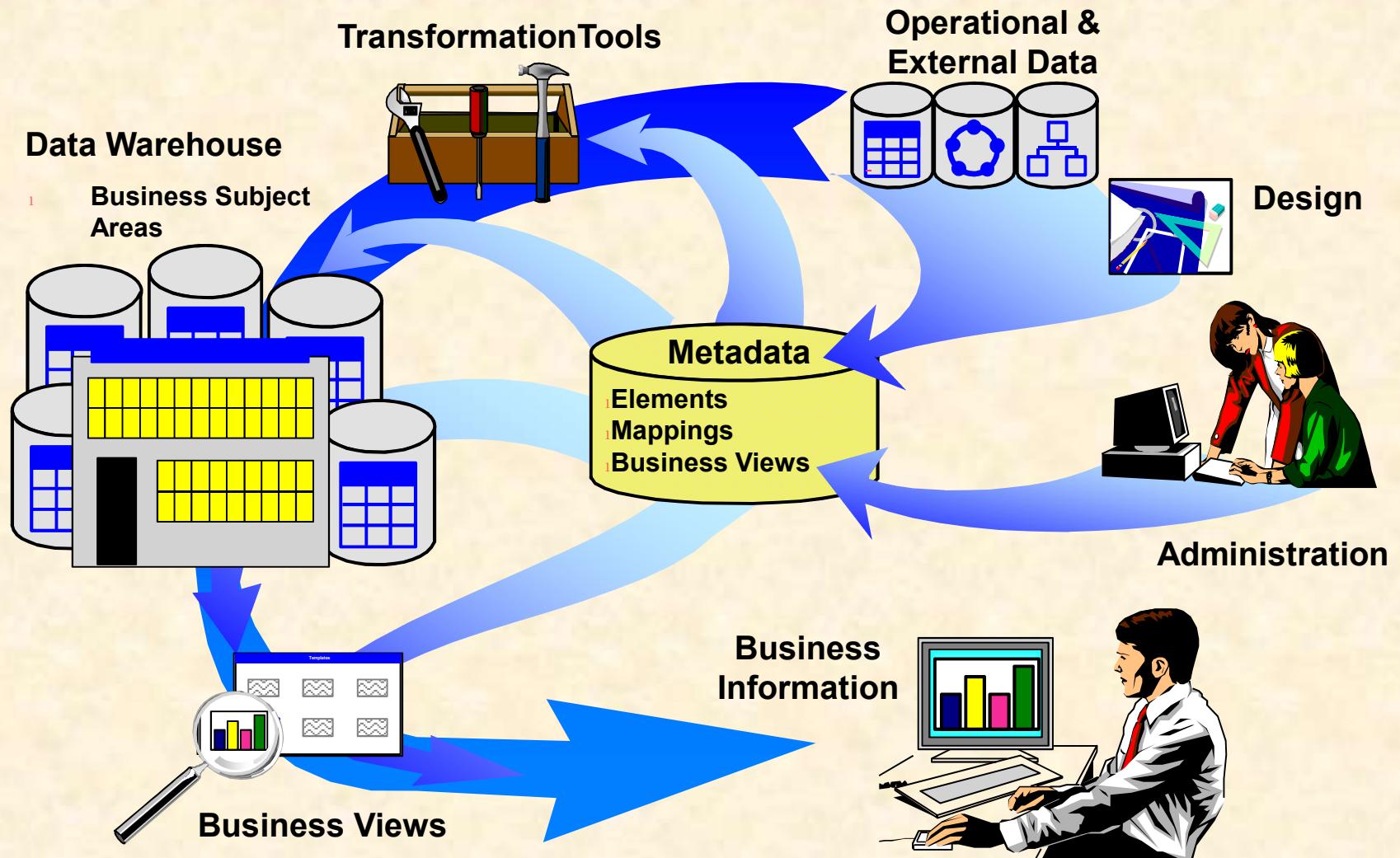
➤ 数据仓库操作型信息

– 例如，数据历史（快照，版本），拥有权，抽取的审计轨迹，数据用法

Metadata Repository

- **Meta data is the data defining warehouse objects**
 - 数据仓库的结构信息
 - schema, view, dimensions, hierarchies, derived data definition, data mart locations and contents
 - 操作型元数据
 - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
 - 数据综合（summarization）算法的描述
 - 从操作型事务环境到数据仓库结构间的映射关系
 - 系统性能相关元数据
 - 商业元数据
 - business terms and definitions, ownership of data, charging policies

5.3 关键技术



元数据的作用

5.4 数据仓库与数据集市

数据集市（Data Mart）

- 早期的数据仓库概念仅仅提供一个多个数据源的数据集成功能，为最终用户访问多个数据源提供统一的数据视图和访问接口，数据仓库的作用仅仅表现为：
 - ✓ 统一的数据模式
 - ✓ 统一的数据表示
 - ✓ 统一的数据属性
- 否则，在网络环境中，即使存在多个可用的数据源，但最终用户可能仍然得不到什么可用的信息。

5.4 数据仓库与数据集市

➤ 建立数据集市的原因

- 数据仓库是一种反映主题的全局性数据组织。但是,全局性数据仓库往往太大, 在实际应用中将它们按部门或个人分别建立反映各个子主题的局部性数据组织, 它们即是**数据集市**。因此, 有时我们也称它为**部门数据仓库**。
- 例: 在有关商品销售的数据仓库中可以建立多个不同主题的数据集市:
 - ✓ 商品采购数据集市
 - ✓ 库房使用数据集市
 - ✓ 商品销售数据集市

5.4 数据仓库与数据集市

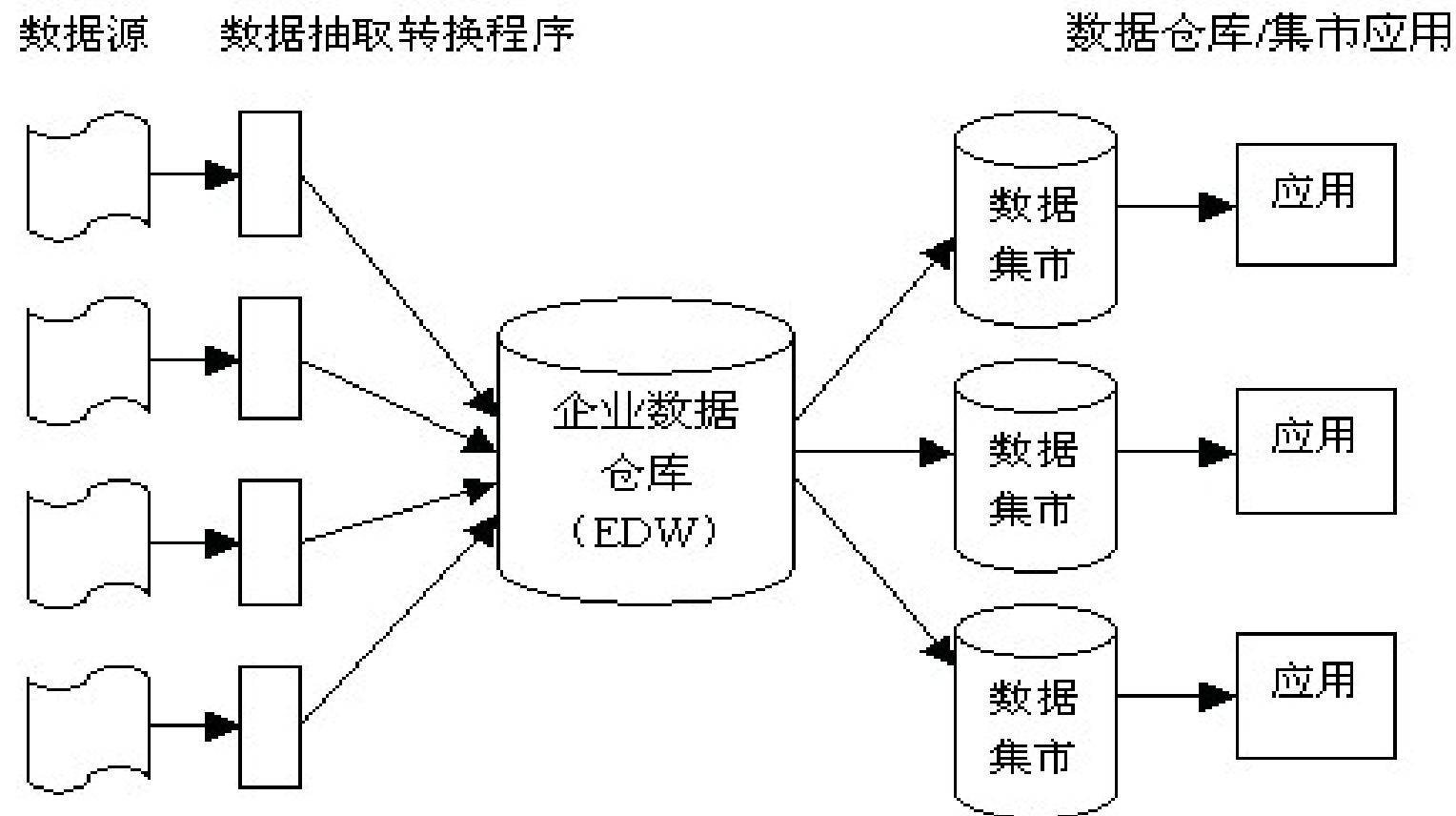
- 数据仓库与数据集市的关系类似于传统关系数据库系统中的 表 与 视图 的关系。数据集市的数据来自数据仓库，它是数据仓库中数据的一个部分与局部，是一个数据的再抽取与组织的过程。
- 建立数据仓库与数据集市的过程可以有两条途径，这实际上是反映了一个完整的企业级数据仓库的建立过程：
 - ① 从 全局数据仓库 到 数据集市
 - ② 从 数据集市 到 全局数据仓库

5.4 数据仓库与数据集市

- 根据建设企业级数据仓库与部门级数据集市的顺序与过程，可以将数据仓库与数据集市之间的关系划分为以下的四种结构：
 - (1) 自顶向下的结构
 - (2) 自底向上的结构
 - (3) 总线结构的数据集市
 - (4) 企业级数据集市结构

5.4 数据仓库与数据集市

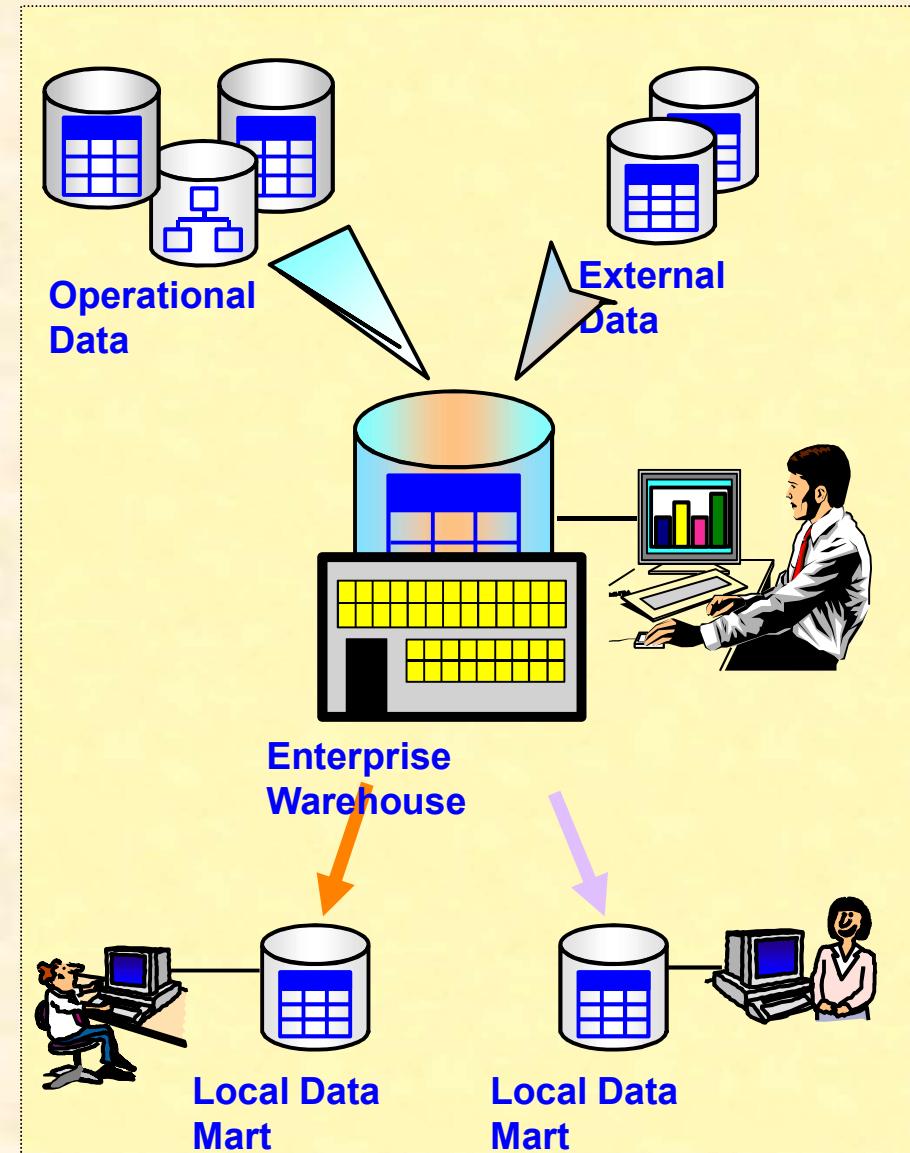
(1) 自顶向下的结构 (Bill Inmon, DWDM)



5.4 数据仓库与数据集市

(1) 自顶向下的结构

- 构建企业数据仓库
 - ① 公共中央数据模型
 - ② 数据再加工
 - ③ 减少冗余和不一致性
 - ④ 搜集历史的、细节的、全局的数据
- 基于企业数据仓库构建数据集市
 - ① 选定企业模型下的部门主题
 - ② 聚集数据
 - ③ 建立集市数据对企业数据仓库的依赖关系



5.4 数据仓库与数据集市

(1) 自顶向下的结构

— 优点

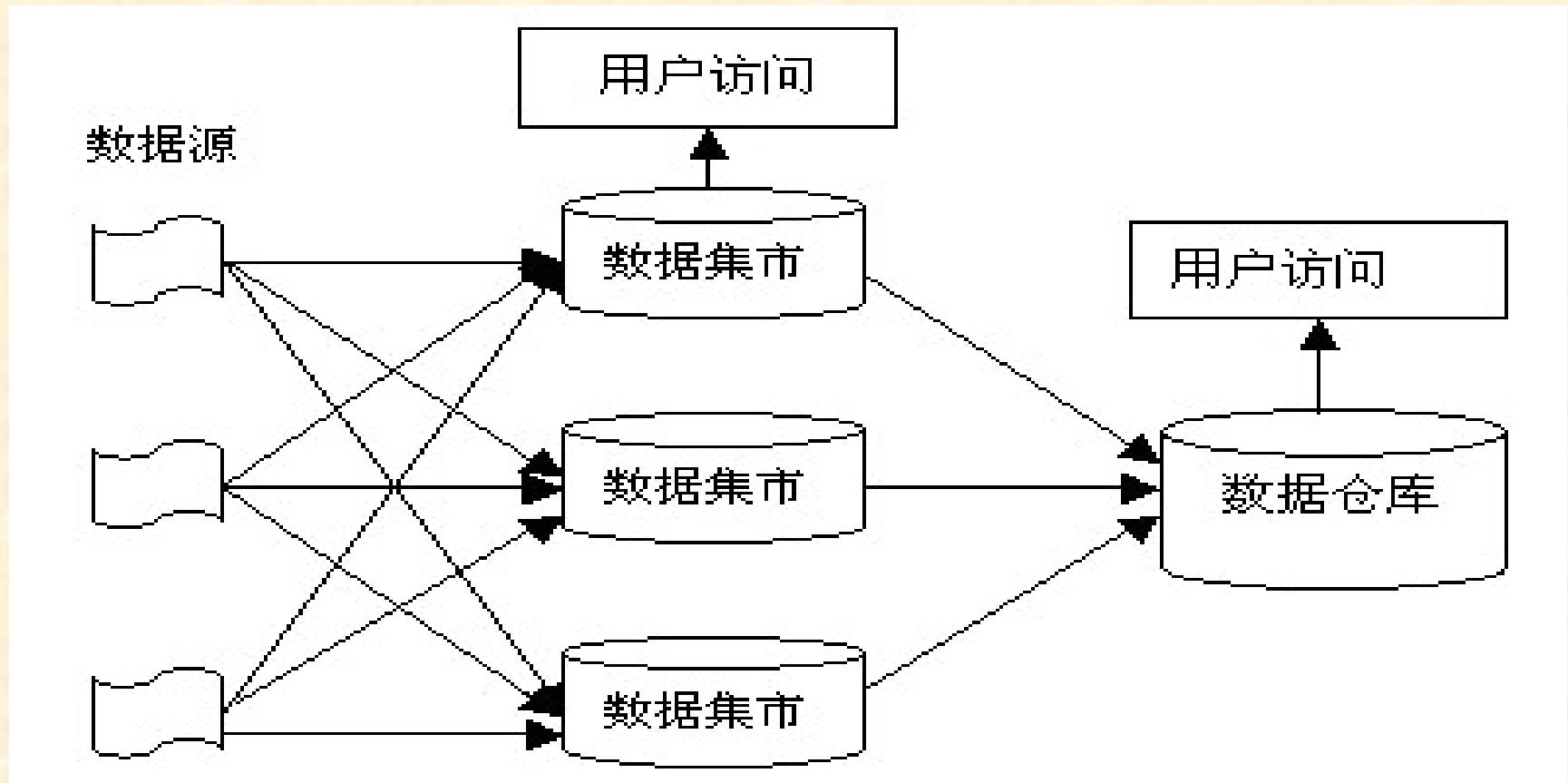
- 建立数据集市能够减轻DW访问负载
- 各部门可以任意处理数据
- 数据转换和整合在 DW 阶段统一完成
- 数据缓冲功能

— 缺点

- 成本高、见效慢、数据集市间不共享资源

5.4 数据仓库与数据集市

(2) 自底向上的结构 (Kimball, DMDW)



5.4 数据仓库与数据集市

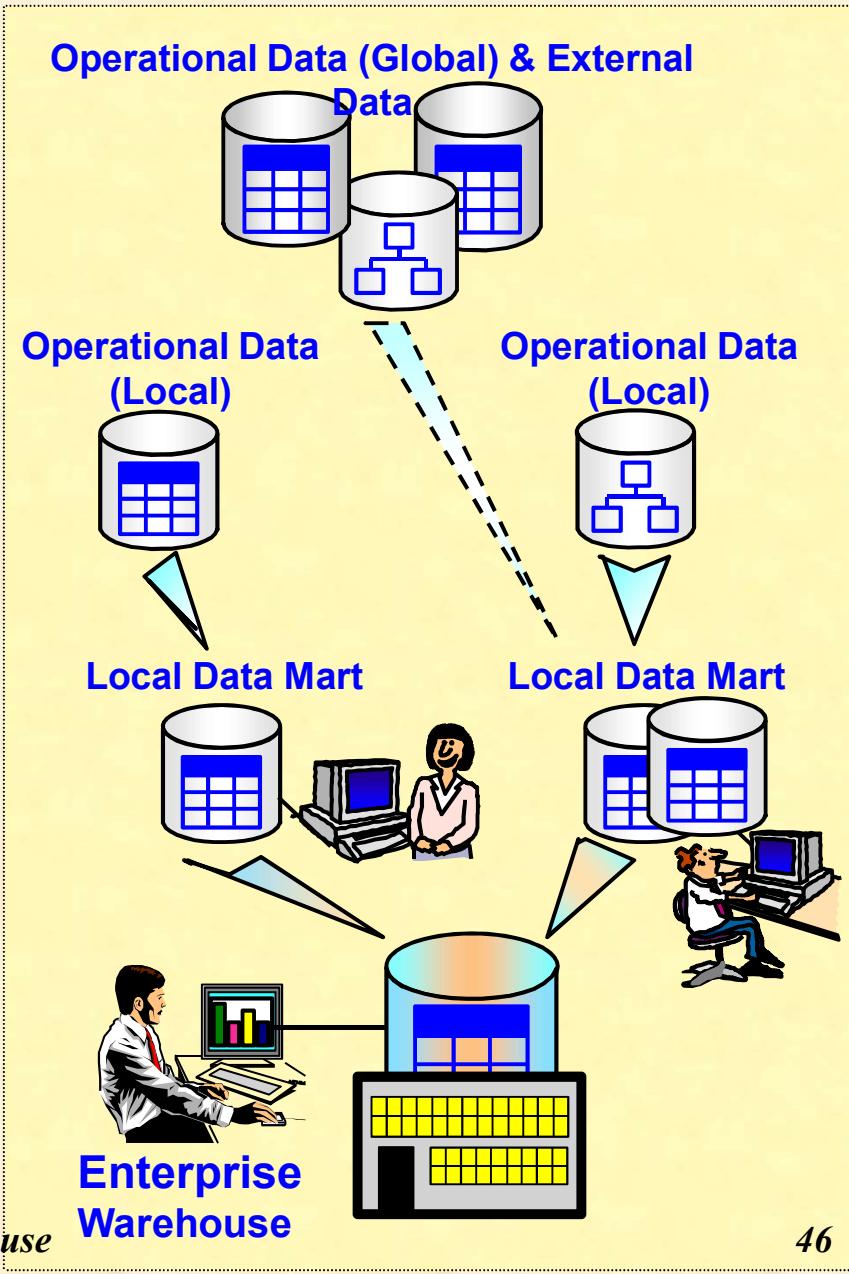
(2) 自底向上的结构

□ 构建数据集市

- ① 划定主题区
- ② 快速实施，本地自治
- ③ 易于复制
- ④ 数据再加工
- ⑤ 允许一定的冗余和不一致

□ 基于数据集市构建企业数据仓库

- ① 确定各数据集市的可用性
- ② 模型的合并
- ③ 消除不同数据集市之间的数据不一致性



5.4 数据仓库与数据集市

(2) 自底向上的结构

— 优点

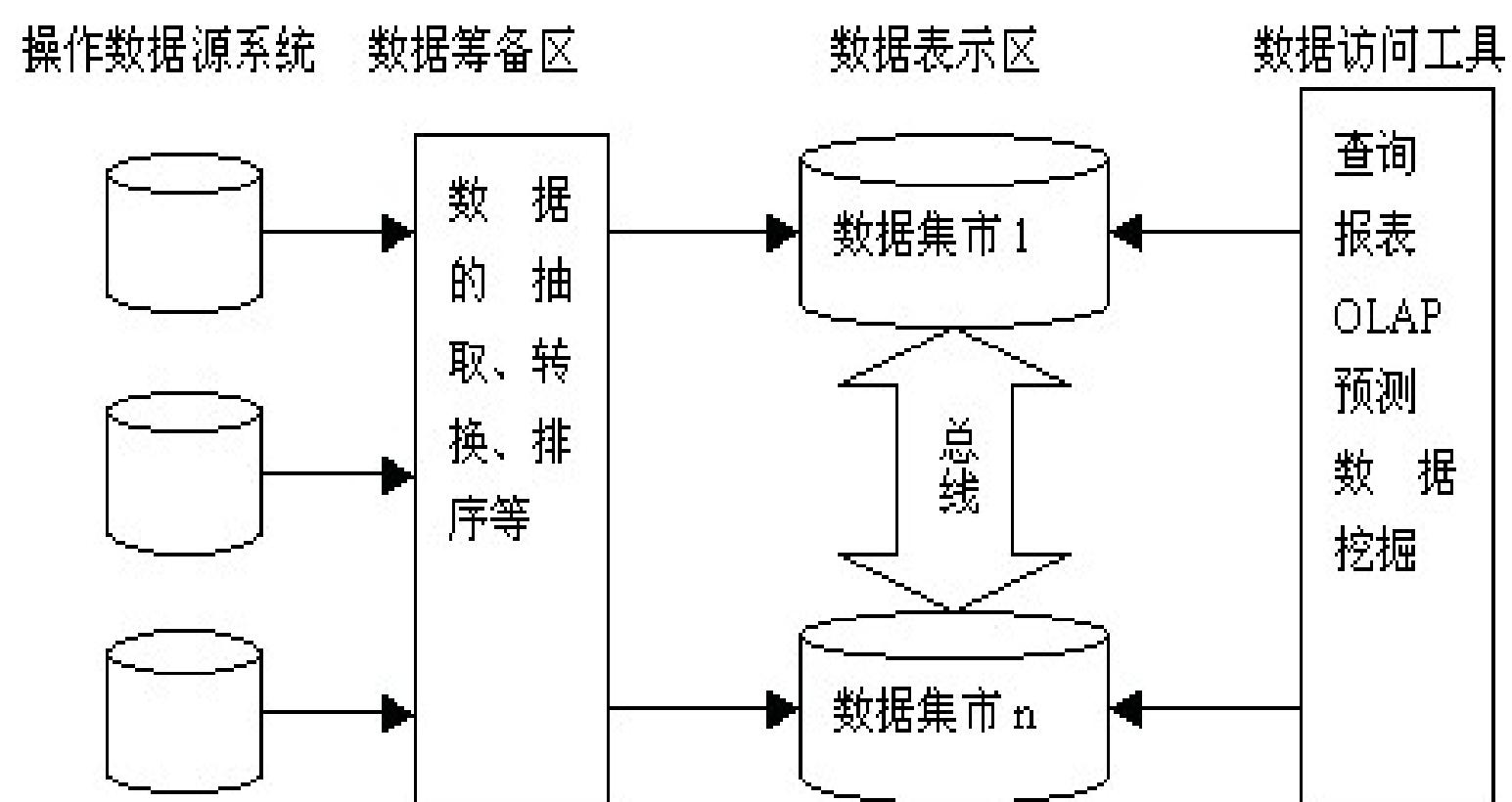
- 见效快、启动资金少

— 缺点

- 各个部门都要进行数据清理整合
- 可能造成“蜘蛛网”、数据不一致等问题
- 总体上没有节约资金

5.4 数据仓库与数据集市

(3) 总线结构的数据集市



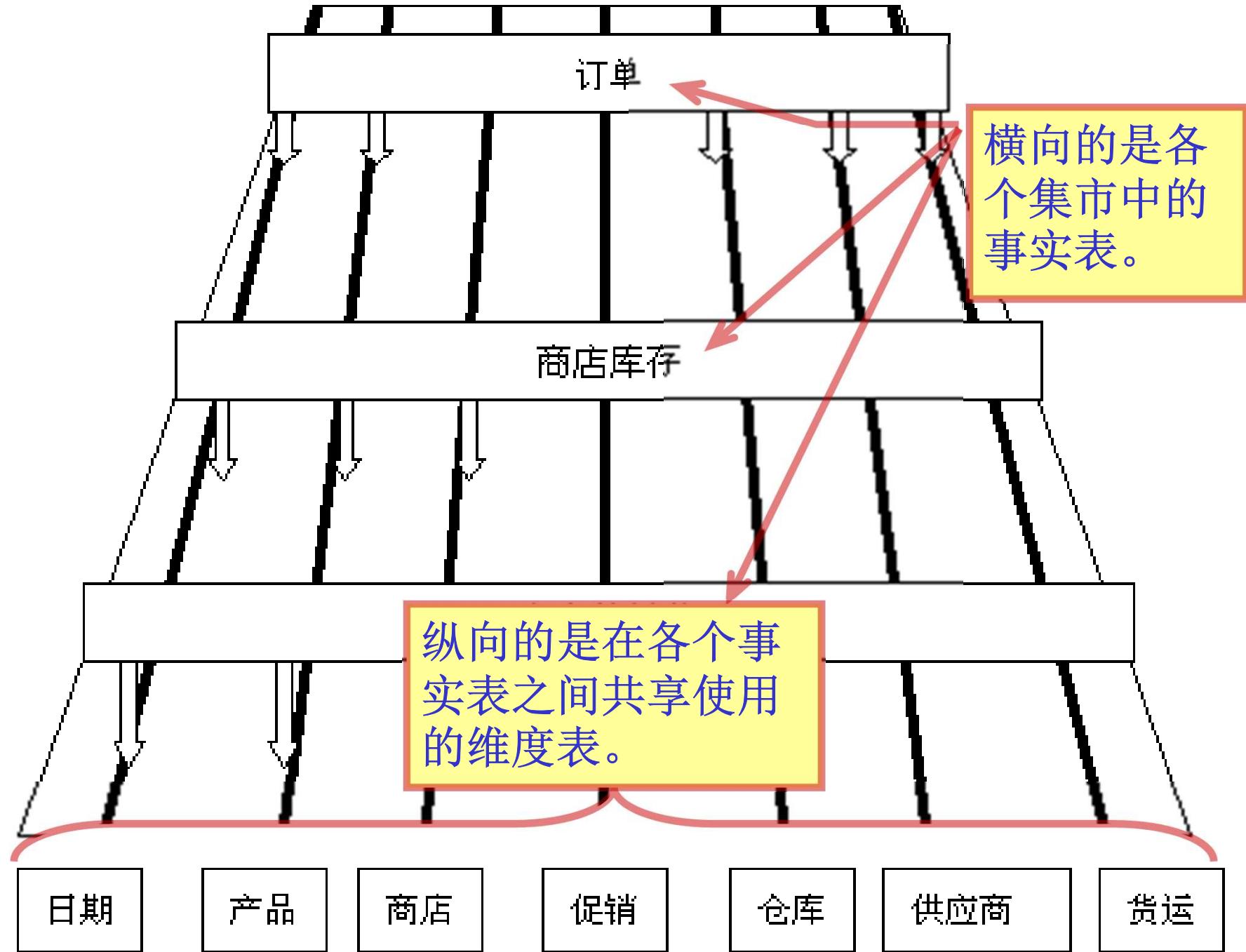
5.4 数据仓库与数据集市

(3) 总线结构的数据集市

- 特点

- 不必建立全局的企业数据仓库，而是直接建立各个数据集市
- 各个数据集市不是孤立的，相互之间通过一种共享维表和事实表的“总线结构”紧密联系在一起。

(如下图)



5.4 数据仓库与数据集市

(3) 总线结构的数据集市

– 优点

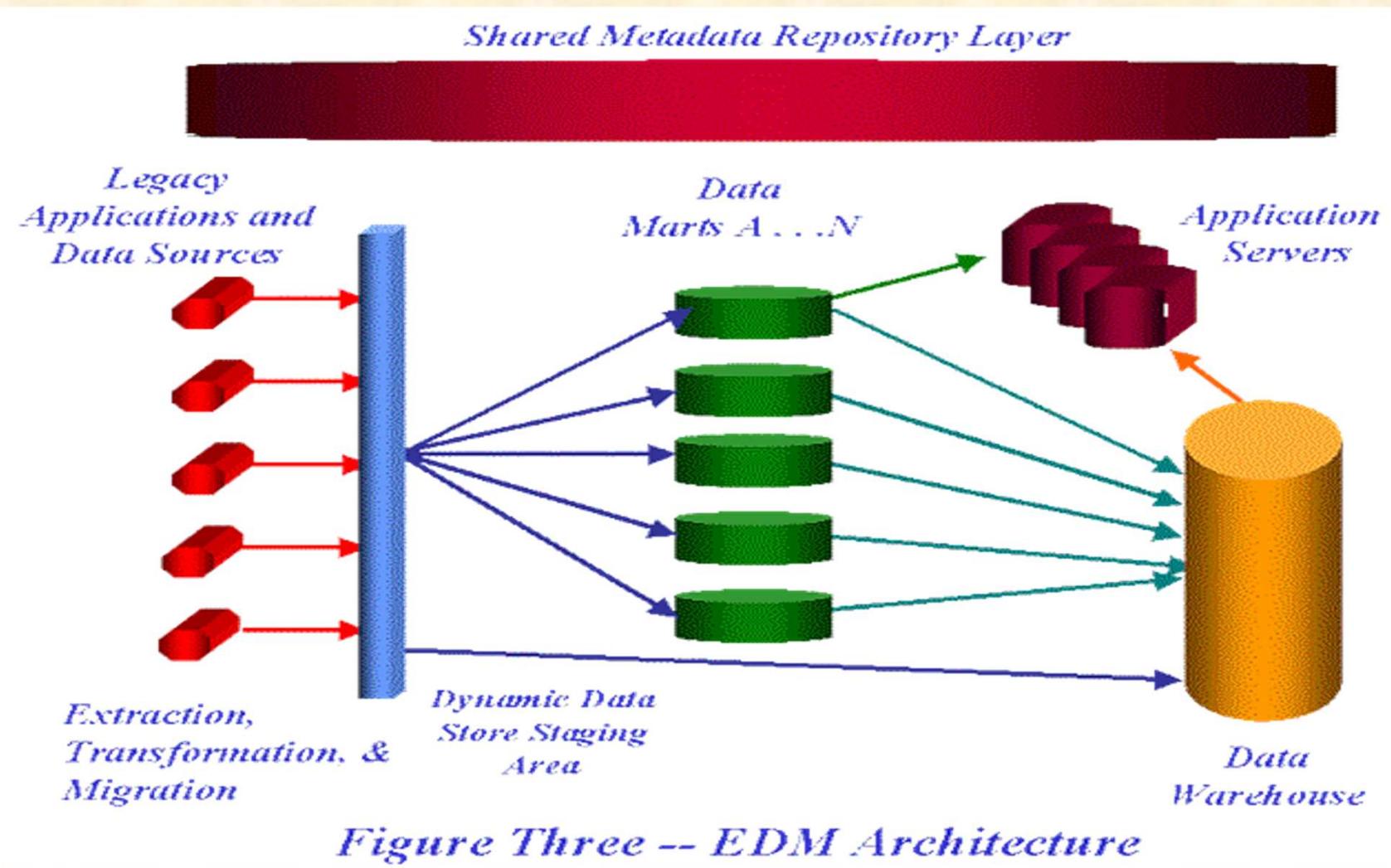
- 共享维表和事实表，解决了建立数据集市的许多问题

– 缺点

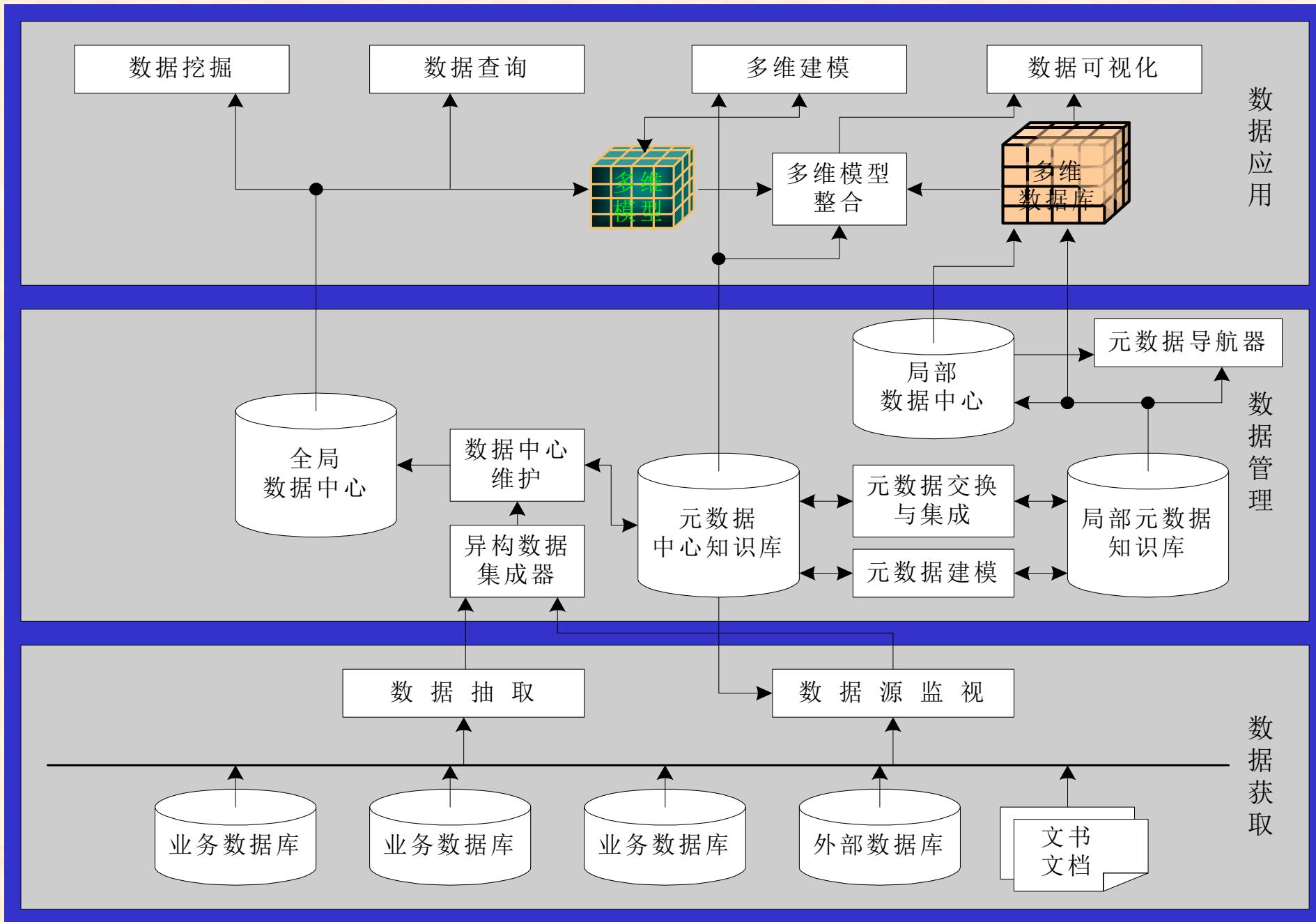
- 这种结构基于多维模型，应用限制于 OLAP
- 多个数据源直接影响多个集市，造成结构不十分稳定

5.4 数据仓库与数据集市

(4) 企业级数据集市结构



数据仓库系统的三层结构



数据仓库

1. 引言
2. 从数据库到数据仓库
3. 数据分析与数据仓库
4. 数据仓库的四大特色
5. 数据仓库的基本结构
6. 数据仓库的设计 ✓
7. 联机分析处理(OLAP)
8. 多维建模 (**Dimensional Modeling**)
9. 数据仓库的应用

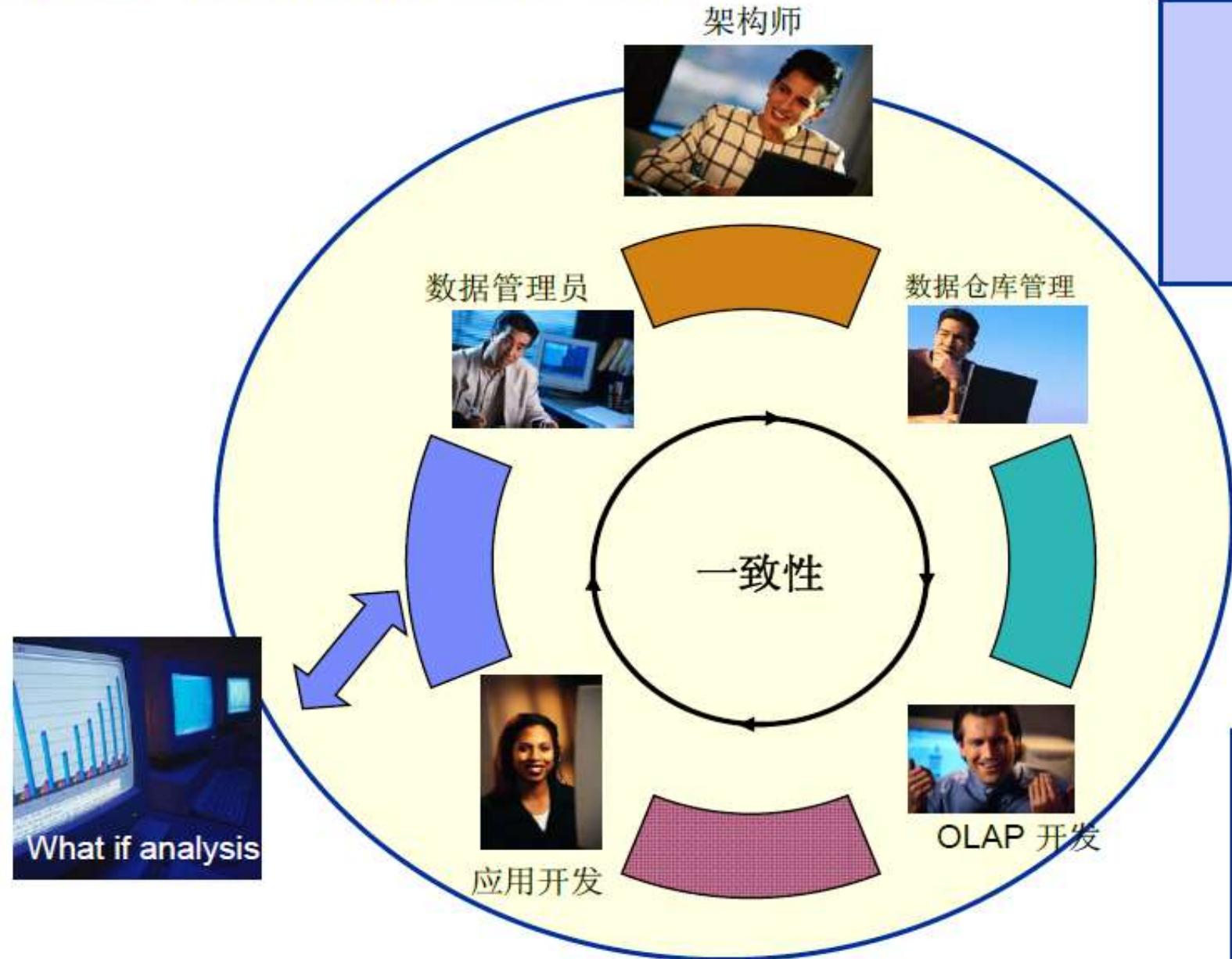
6 数据仓库的设计

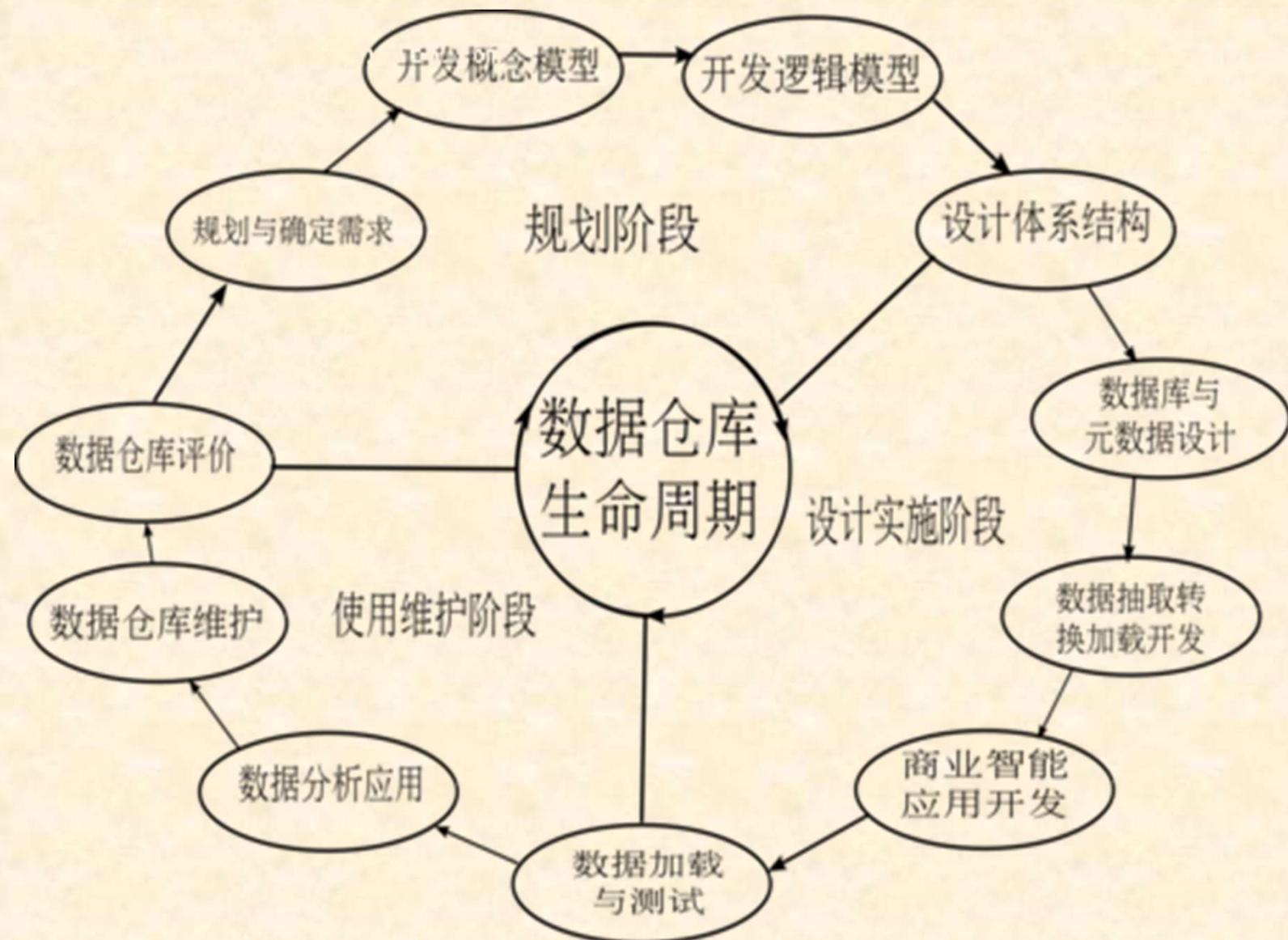
- 数据仓库设计与数据库设计：
 - 原理一致，方法不同
- 在事务型数据处理中需要作数据库设计，而在分析型数据处理中则需作数据仓库设计，这两者在目标上是一致的 – **数据建模**
- 因此，数据库设计中的很多设计思想与方法都可在数据仓库中得到应用，但是由于事务型与分析型的数据处理的不一致，因此两者在设计中的很多方面也存在着差别。

6 数据仓库的设计

- 面向OLTP的数据库设计有着明确的应用需求，严格遵循系统生命周期的阶段划分，每个阶段都规定有明确的任务，上一阶段确定的任务完成后，产生一定格式的文档交给下一阶段。
- 数据仓库是面向主题的、集成的、不可更新的、随着时间的变化而不断变化的，这些特点决定了数据仓库的系统设计不能采用同开发传统的OLTP数据库一样的设计方法。

数据仓库/分析的生命周期





6 数据仓库的设计

□ 数据仓库的设计过程必须遵循三条原则：

- 面向主题原则
- 数据驱动原则
- 原型法设计原则

6 数据仓库的设计

□设计原则 1 – 面向主题原则

➤ 建立数据仓库的目的

- 构建数据仓库的目的是面向企业的管理人员，为经营管理提供决策支持信息。因此数据仓库的组织设计必须以用户决策的需要来确定，即从用户决策的主观需求（主题）开始。

➤ 数据仓库中数据的组织方法

- 为了进行数据分析首先要有分析的主题，**以主题为起始点，进行相关数据的设计**，最终建立起一个面向主题的分析型环境。

- 在数据库设计中则是以客体（Object）为起始点，即以客观操作需求为设计依据。

6 数据仓库的设计

➤ 例如：‘商品销售’主题

- 建立目的？

- 管理人员能够在适当的时候，订购适当的商品，并把它们分发到适当的商店中去销售，以提高商场的销售总金额。

- 需要执行的分析操作？

- 分析什么样的商品，在什么样的时间和商店内畅销
 - 即分析商品的销售额与商品类型、销售时间及商店位置之间的变化关系
- 管理人员将据此决定他们的经营策略

6 数据仓库的设计

□设计原则 2 – 数据驱动原则

➤ 数据的来源？

- 数据仓库是在现存数据库系统基础上进行开发的，它着眼于有效地提取、综合、集成和挖掘已有数据库中的数据资源，服务于企业高层领导管理决策分析的需要。
- 数据仓库中的数据必须是从已有的数据源中抽取出 来，是已经存在的数据，或是对已经存在的数据进 行加工处理而获得。

6 数据仓库的设计

➤ 数据驱动方法（原则）

- 在数据仓库设计中，由于其所有数据均建立在已有的数据库基础上，即从已经存在于操作型环境中的数据出发进行数据仓库的设计，这种设计方法被称为“数据驱动”方法。

6 数据仓库的设计

□设计原则 3 – 原型法设计原则

➤ 不断变化的分析需求

- 数据仓库系统的原始需求不明确，且不断变化与增加，开发者最初并不能确切了解到用户的明确而详细的需求，用户所能提供的无非是需求的大方向或部分需求，更不能较准确地预见到以后的需求。
- 采用原型法来进行数据仓库的开发是比较合适的，即：从构建系统的基本框架着手，不断丰富与完善整个系统。

6 数据仓库的设计

➤ 逐步求精的设计过程

- 数据仓库用户的需求是在设计过程中不断细化明确的。
- 数据仓库系统的开发也是一个经过不断循环、反馈而使系统不断增长与完善的过程。
- 在数据仓库开发的整个过程中，自始至终要求决策人员和开发者的共同参与和密切合作，不做或尽量少做无效工作与重复工作。

6 数据仓库的设计

➤ 数据仓库的设计步骤和结构

- 数据仓库设计: 一个商务分析框架
 - 数据仓库为商务分析提供了什么?
 - 提供竞争优势, 加强生产能力, 促进了与顾客的联系.
 - 自顶向下的视图
 - 使得我们可以选择数据仓库所需的相关信息
 - 数据源视图
 - 揭示被操作数据库系统捕获, 存储和管理的信息
 - 数据仓库视图
 - 包括事实表和维表.
 - 商务查询视图
 - 是从最终用户的角度透视数据仓库中的数据

6 数据仓库的设计

➤ 数据仓库设计大致有如下几个步骤：

1) 系统规划

- 明确主题（如订单，发票，库存，记账管理，销售和一般分类账）
- 技术准备

2) 概念设计

3) 逻辑设计

4) 物理设计

5) 数据仓库生成

6) 数据仓库的运行与维护

6 数据仓库的设计

□ 系统规划 – 明确主题

- 在数据仓库设计的开始，首先要做的事是有关分析人员需要确定具体领域的分析对象，这个对象就是主题。
- 主题是一种较高层次的抽象，对它的认识与表示是一个逐步完善的过程。
- 在开始时不妨先确定一个初步的主题概念以利于设计工作的开始，此后随着设计工作的进一步开展，再逐步扩充与完善。（原型设计法）

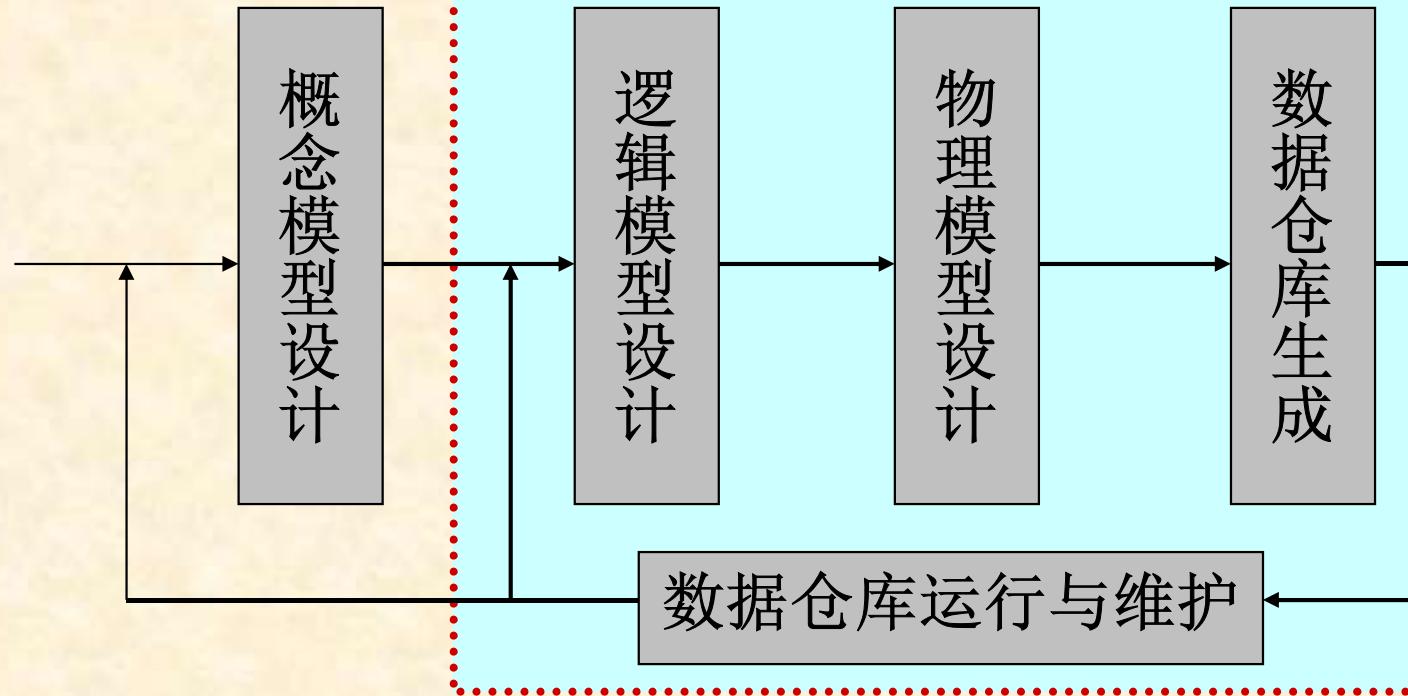
6 数据仓库的设计

□ 系统规划 – 技术准备工作

- 准备具体的技术要求和物理实现环境，包括：
 - 技术评估
 - 其内容包括数据仓库的性能指标，如：
 - ✓ 数据存取能力（包括管理大数据量数据的能力）
 - ✓ 模型重组能力
 - ✓ 数据装载能力
 - 技术环境准备
 - 在评估基础上提出数据仓库的软硬件平台要求，包括计算机、网络结构、操作系统、数据库及数据仓库软件的选购要求等。

6 数据仓库的设计

针对每一个选定的当前实施的主题



数据仓库设计步骤示意图

什么是模型？

- 模型是现实世界中的事物（包括客观规律）的一种客观的抽象表示或模拟。
- 模型既反映事物的原型，又不等于该原型，或者说它是原型的一种近似。
- 数据模型是指使用**实体、属性及其关系**对**企业运营和逻辑规则**进行统一的定义、编码和命名。
- 数据模型是业务人员、it人员和开发商的之间进行沟通的一套语言.

模型的分类

1. **业务模型**是分别从业务过程和客户对应的业务状况和业务参与者的角度来描述系统的业务过程
2. **概念模型**是按用户的观点对数据建模，它是对现实世界的第一层抽象，是用户和数据库设计人员之间交流的工具。

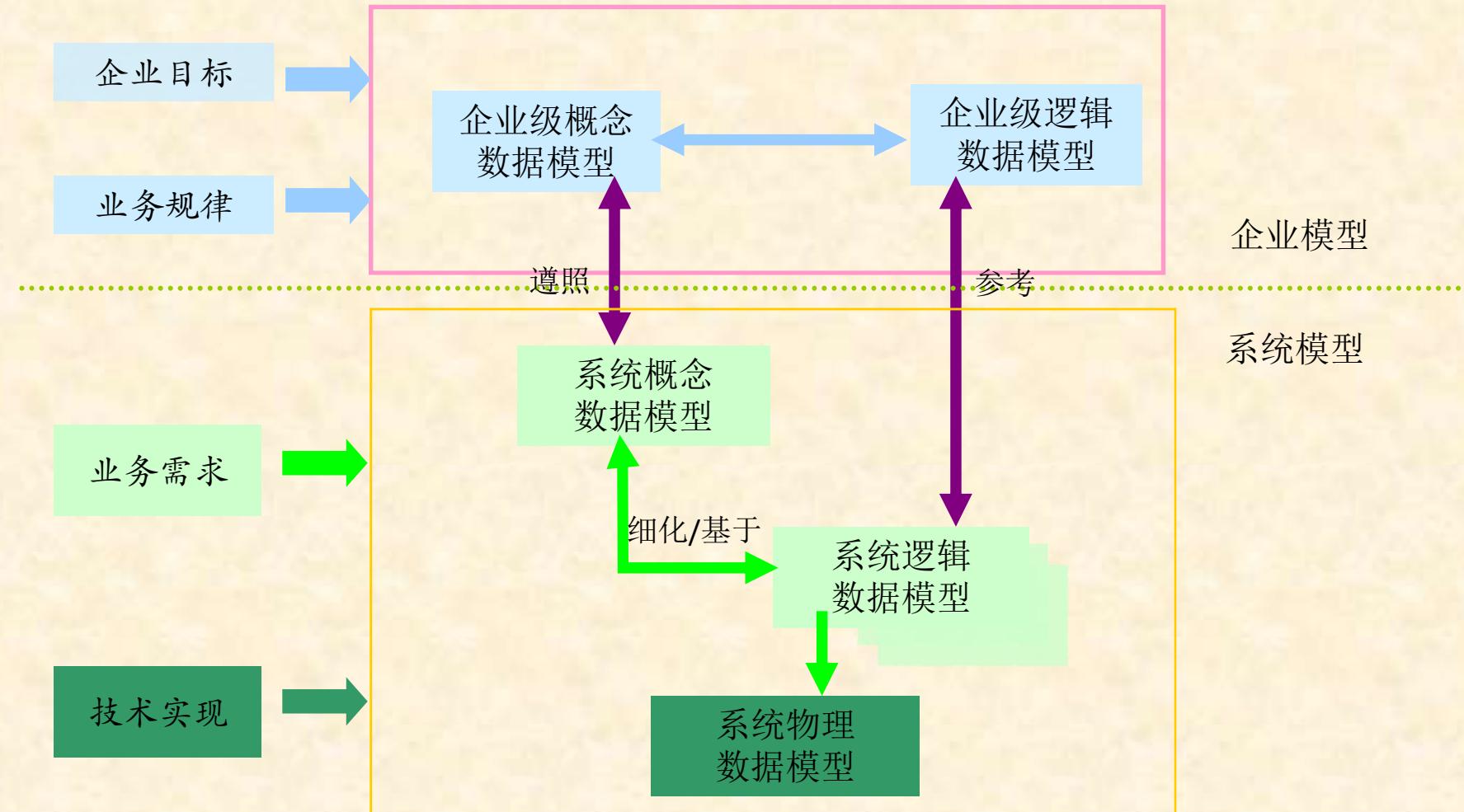
概念模型是一种高层次的模型，定义了重要的业务事务及其联系，它的主要核心是由**数据主题**或及其集合，以及主体间的业务关系组成。

模型的分类

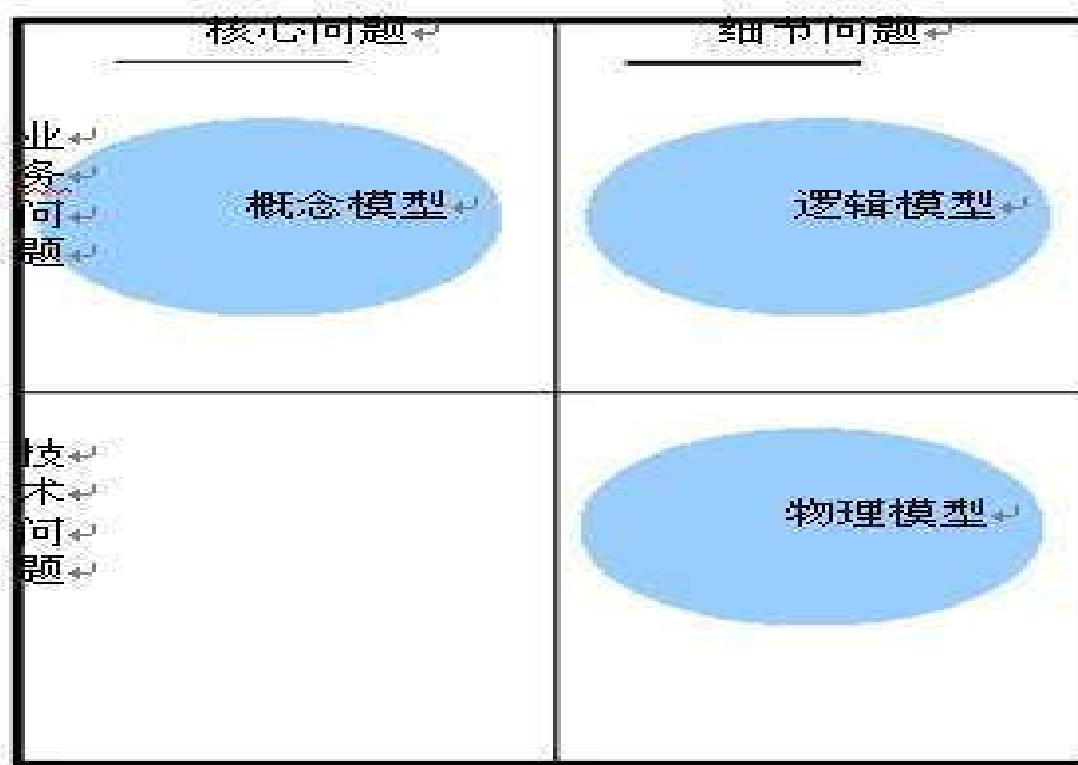
3. **逻辑模型**是对概念模型的进一步分解和细化，描述了实体、属性和实体之间的关系，使用通用的字符长度和类型来描述。
4. **物理模型** 描述了模型实体的细节并对数据冗余和性能进行平衡。

在进行物理模型设计时需要考虑该模型最终落地到所选数据库的特有特性、数据类型、长度、索引、数据存储。在进行物理模型设计时需要先确定**DB**架构和应用架构。

企业数据模型定位



模型所关注的几个不同领域



- 概念模型更关注业务相关的核心问题
- 逻辑模型主要描述业务问题的各个细节
- 物理模型则侧重表达数据的技术细节

数据模型的特点

1. 数据模型是整个系统建设过程的**导航图**。通过数据模型可以清楚地表达企业内部各种业务主体之间的相关性，使不同部门的业务人员、应用开发人员和系统管理人员获得关于系统的统一完整的视图。
2. 有利于数据的**整合**。数据模型是整合各种数据源的重要手段，通过数据模型，可以建立起各个业务系统与数据仓库之间的映射关系，实现源数据的有效采集。
3. 通过数据模型的建立，**可以排除数据描述的不一致性**。如：同名异义、同物异名等等，使系统的各方参与人员基于相同的事实在进行有效沟通。

数据模型的特点

4. 由于数据模型对现有的信息以及信息之间的关系从逻辑层进行了全面的描述，当未来业务发生变化或系统需求发生变化时，可以很容易地实现系统的扩展。数据结构的变化不会偏离原有的设计思想。

5. 可以消除数据仓库中的冗余数据。数据模型的建立可以使开发人员清楚地了解数据之间的关系，以及数据的作用。在数据仓库中根据需求采集那些用于分析的数据，而不需要那些纯粹用于操作的数据。

模型与需求及业务的关系

□ 客户开展与市场、合作伙伴等相关的服务活动，是逻辑模型设计的出发点和依据

- 建立逻辑数据模型的第一步：**业务分析**，参与人员主要是业务人员（或熟悉业务的技术人员）。
- 划定需求业务范围之后，业务分析首先要了解企业开展哪些活动，每项活动涉及哪些对象和资源，需要哪些相关活动等等。
- **识别并列出业务活动中涉及的每一个实体及其主要属性，并完善其属性。**设计出逻辑数据模型的大致框架后，可以在概念模型主题域划分的基础上**对实体再次进行归类**，以便将来理解和改善。

□ 需求分析从“深度”上明确设计时数据**最细粒度**是多少

- 由于数据仓库的底层细节数据决定了将来数据仓库可以分析到的最细程度，因此其逻辑模型设计阶段要根据当前及近期可能的最细分析程度决定其实体或属性的最细粒度。

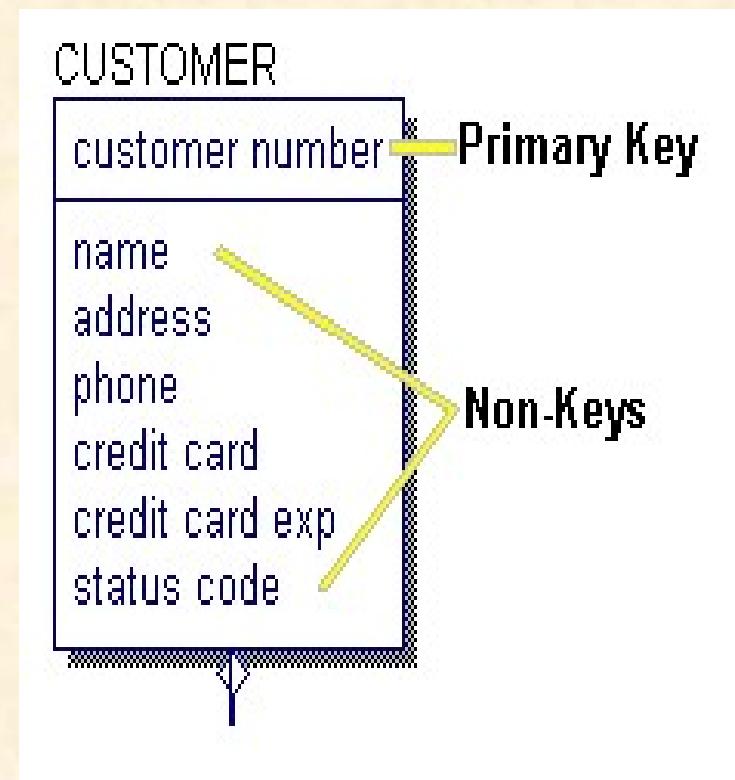
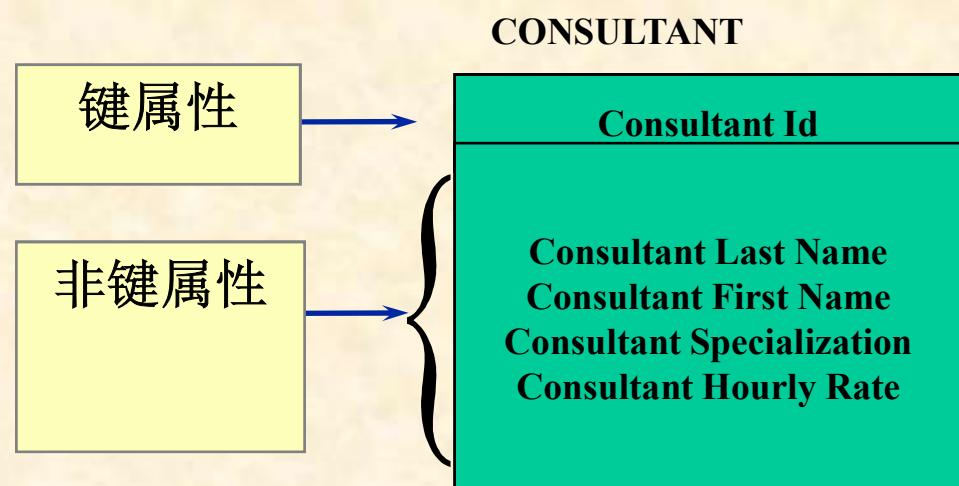
模型设计分类——不同角度

➤ 第三范式 (3NF, 即 Third Normal Form)

- 范式是数据库逻辑模型设计的基本理论
- 符合第三范式的关系必须具有以下三个条件：
 - 每个属性的值唯一,不具有多义性;
 - 每个非主键属性必须完全依赖于整个主键,而非主键的一部分;
 - 每个非主键属性不能依赖于其他关系中的属性
(否则这种属性应该归到其他关系中去)。

模型设计分类——不同角度

- 两种属性类别
 - 键属性(Key)
 - 非键属性(Non Key)



模型设计分类——不同角度

◆关系数据库的三范式建模：通常将三范式建模方法用于建立各种操作型数据库模型设计。

◆Inmon提倡的**三范式数据仓库建模**：和操作型数据库系统的三范式建模在侧重点上有些不同。

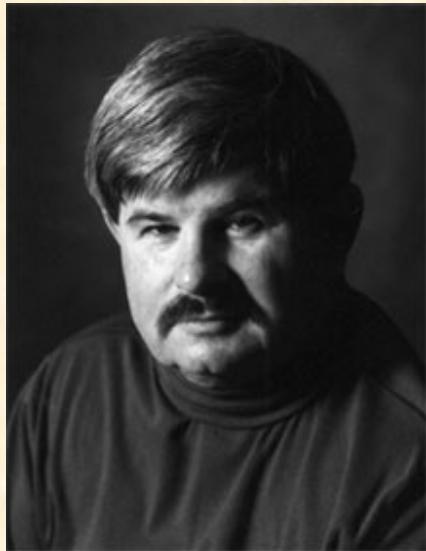
- 第一层是实体关系层，也即企业的业务数据模型层，在这一层上和企业的操作型数据库系统建模方法是相同的；
- 第二层是数据项集层，在这一层的建模方法根据**数据的产生频率及访问频率**等因素，与企业的操作型数据库系统的建模方法产生了不同；
- 第三层物理层是第二层的具体实现。

◆第三类是**Kimball**提倡的数据仓库的**维度建模**

- 星型模型
- 雪花模型
- 星座模型

多个事实表共享维表. 可以看作是星型模式集

- 维度建模是一种面向用户需求的、容易理解的、访问效率高的建模方法。



Are you an Inmonite or a Kimbalite ?

Data Driven Data Warehouse

or

Metric driven data warehouse

Inmon vs. Kimball

数据仓库分层比较表

分层方法	具体分层	优点	缺点
Inmon	数据源接口—ODS—EDW—DM	按照第三范式建模，可得到统一口径	衍生出更多分层，意见不统一
Kimball	数据源接口—DM	实践证明非常有效，体现了数据仓库“面向主题”的特点	按照事实表、维表来构建数据库、数据集市，和第三范式有冲突

Inmon：在你设计好所有的组件之前，不做任何工作

优势：易于维护，高度集成

NCR, IBM...

劣势：结构死板，部署周期较长

Microsoft...

Kimball：只要用户有需求，就可以构建他们想要的组件，然后在进行集成。

优势：构建迅速，最快的看到投资回报率，敏捷灵活

劣势：作为企业资源不太好维护，结构复杂，数据集市集成困难

6 数据仓库的设计

- 数据仓库的设计一般应遵循以下流程：
 - 在进行系统设计与开发之前，各个业务部门人员应该就核心的业务概念及其关系（即概念模型）达成一致
 - 系统设计时，技术人员与业务人员一起，直接进行逻辑模型的设计，而不再单独设计概念模型
 - 逻辑模型设计完成之后，再根据所选的数据库产品及其他因素，进行物理模型的设计。

6 数据仓库的设计

□ 概念模型设计

1. 确定系统边界：决策类型、需要的信息、原始信息
2. 确定主要的主题及其内容
3. OLAP设计（维度及其类别；事实及其数值和指标）



□ 确定系统边界

- 要做的决策类型有哪些？
- 决策者感兴趣的是什么问题？
- 这些问题需要什么样的信息？
- 要得到这些信息需要包含哪些数据源？

6 数据仓库的设计

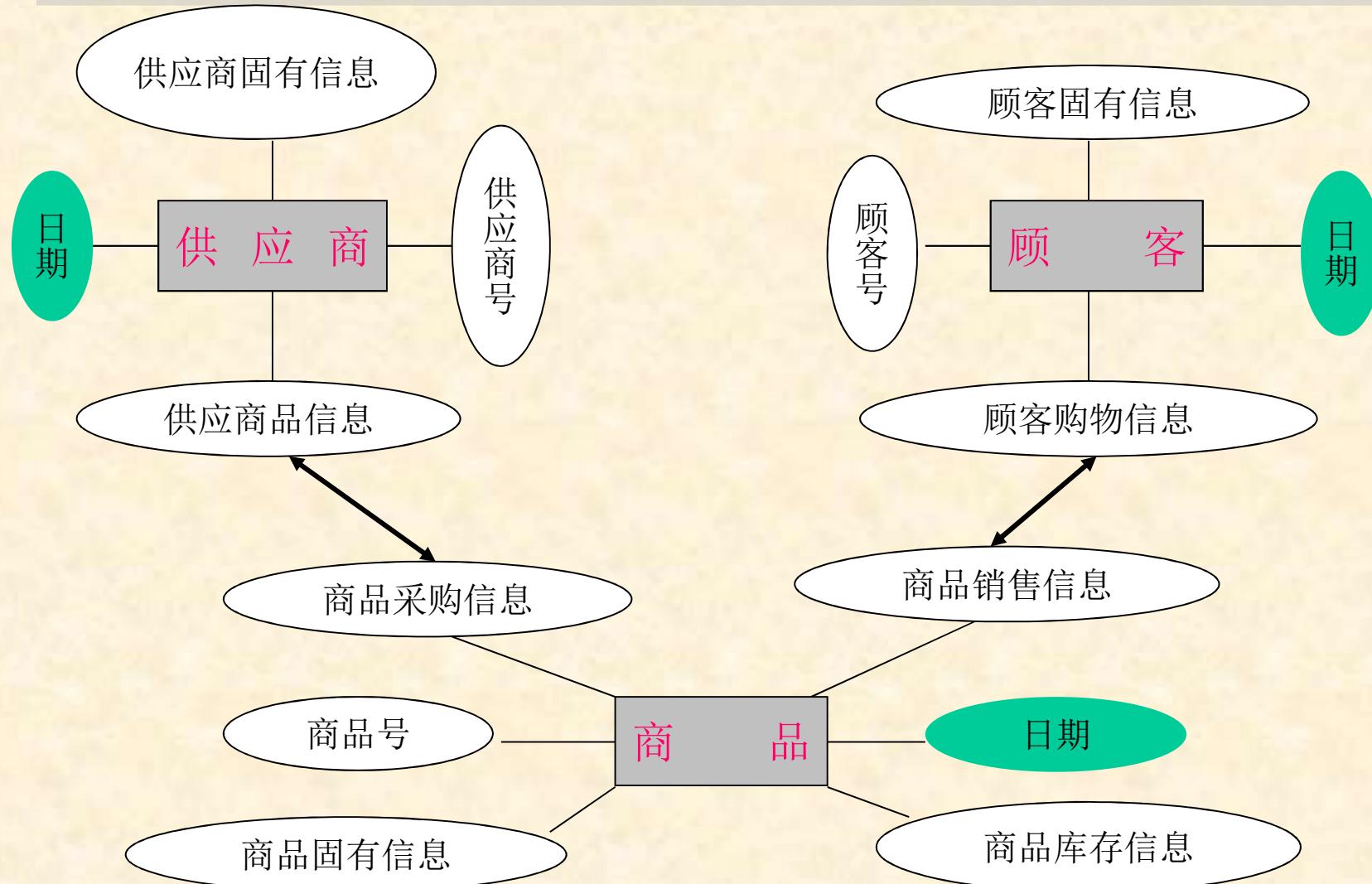
➤ 确定主要的主题

- 即明确数据仓库的分析对象，然后对每个主题的内容进行较详细的描述，包括：
 - 确定主题及其属性信息
 - 描述每个属性的取值情况
 - 固定不变的
 - 半固定的
 - 经常变化的
 - 确定主题的公共关键字
 - 明确主题之间的关系：主题间联系及其属性
- 在确定上述内容后，就可以用传统的实体联系模型（E-R模型）来表示数据仓库的概念数据模型。
 - 例如：用于商场管理者的数据仓库

6 数据仓库的设计

主题名	公共码键	属性信息
商品	商品号	<ul style="list-style-type: none">➤ 固有信息：商品号，商品名，类别，颜色等➤ 采购信息：商品号，供应商号，供应价，供应日期，供应量等➤ 销售信息：商品号，顾客号，售价，销售日期，销售量等➤ 库存信息：商品号，库房号，库存量，日期等
供应商	供应商号	<ul style="list-style-type: none">➤ 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	<ul style="list-style-type: none">➤ 固有信息：顾客号，姓名，性别，年龄，文化程度，住址，电话等➤ 购物信息：顾客号，商品号，售价，购买日期，购买量等

6 数据仓库的设计



商品、顾客和供应商之间的E-R图

6 数据仓库的设计

❖ 逻辑模型设计

1. 将E-R图转换成关系数据库的二维表
2. 定义数据源和数据抽取规则

□ 在逻辑模型的设计过程中，需要考虑以下一些问题：

- ① 适当的粒度划分
- ② 合理的数据分割策略
- ③ 定义合适的数据来源等

6 数据仓库的设计

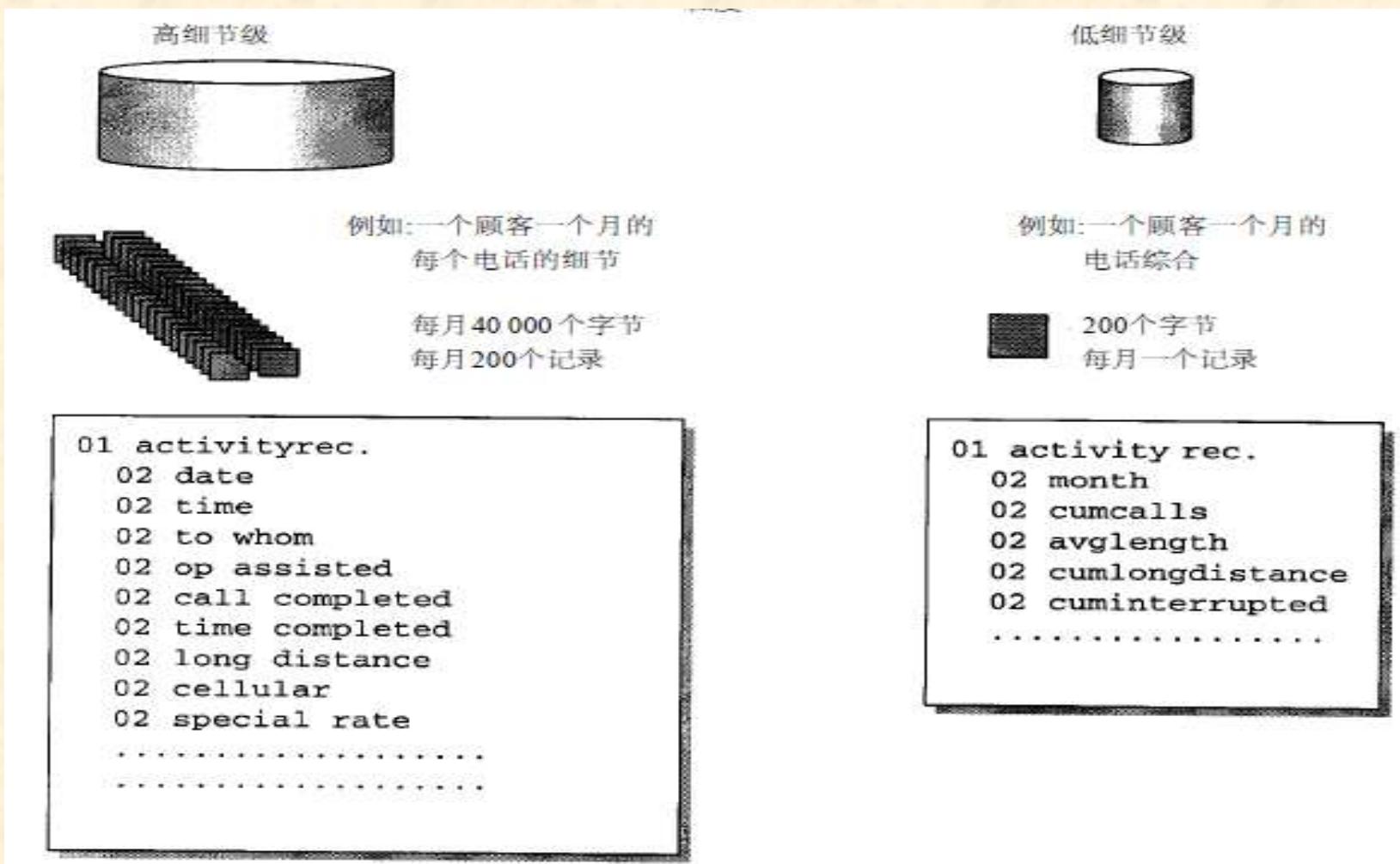
❖ 逻辑模型设计 – 粒度划分

- ✓ 在设计过程中需要考虑数据仓库中数据粒度的划分原则，即数据单元的详细程度和级别。
 - 数据越详细，粒度越小，级别就越低
 - 数据综合度越高，粒度越大，级别就越高
- ✓ 一般将数据划分为：详细数据、轻度总结、高度总结三种粒度，或者采用更多级的粒度划分方法。例如：
 - 根据时间跨度进行的统计有：天，周，月，季度，年
 - 对于不适合进行统计的属性值，可以采样获取数据
- ✓ 粒度的划分将直接影响到数据仓库中的数据量以及所适合的查询类型，粒度划分是否适当是影响数据仓库性能的一个重要方面。

6 数据仓库的设计

- 商品固有信息:
 - 商品表（商品号，商品名，类型，颜色，…）/* 细节数据 */
- 商品采购信息:
 - 采购表（商品号，供应商号，供应日期，供应价，…）/* 细节数据 */
 - 采购表1（商品号，时间段1，采购总量，…）/* 综合数据 */
 -
 - 采购表n（商品号，时间段n，采购总量，…）
- 商品销售信息:
 - 销售表（商品号，顾客号，销售日期，售价，销售量，…）/* 细节数据 */
 - 销售表1（商品号，时间段1，销售总量，…）/* 综合数据 */
 -
 - 销售表n（商品号，时间段n，销售总量，…）
- 商品库存信息:
 - 库存表（商品号，库房号，库存量，日期，…）/* 细节数据 */
 - 库存表1（商品号，库房号，库存量，星期，…）/* 样本数据 */
 -
 - 库存表n（商品号，库房号，库存量，年份，…）
- 其它导出数据:

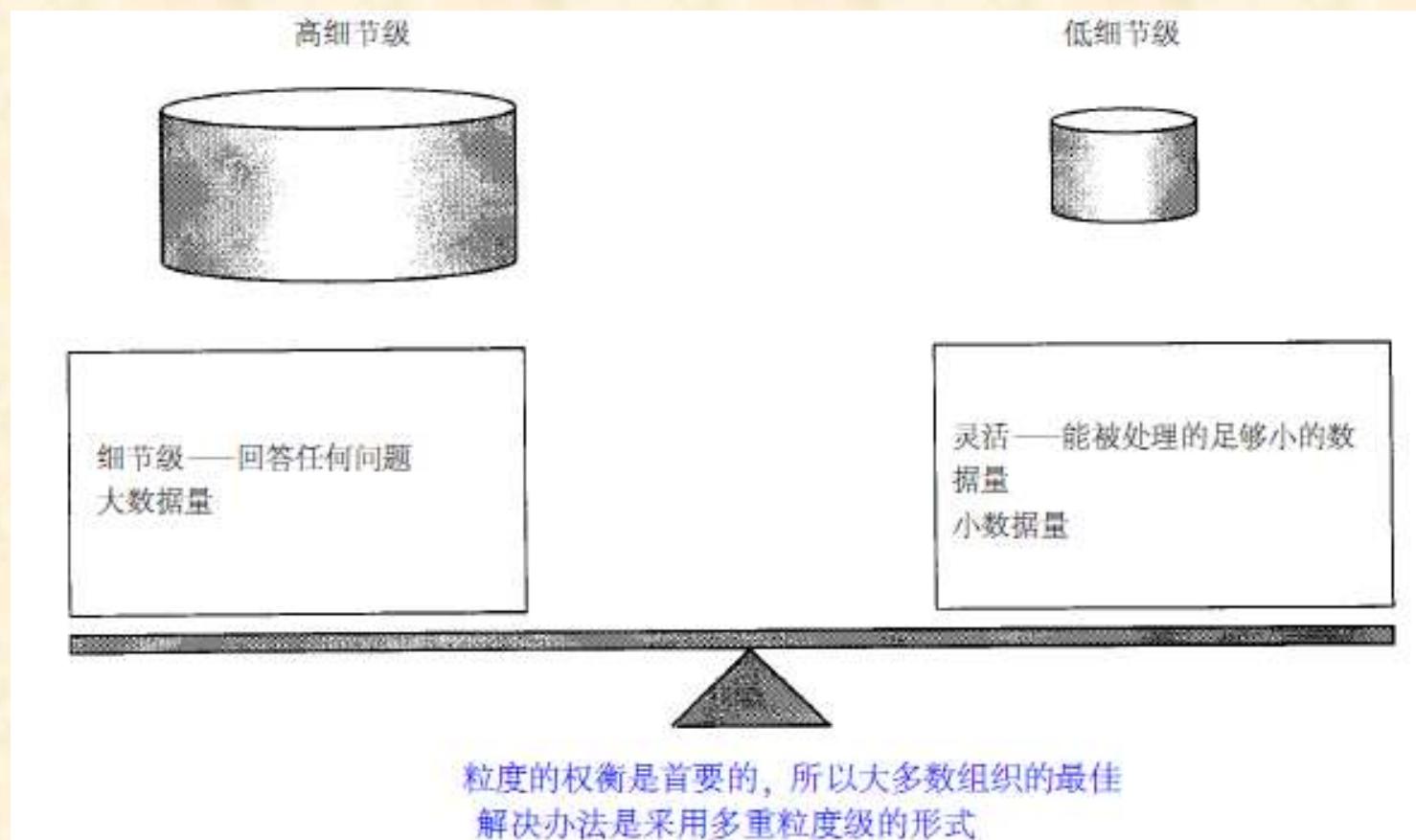
6 数据仓库的设计



- 低粒度，能回答：“张三上周是否给在上海的女友打了电话？”
- 高粒度，无法明确地回答这个问题

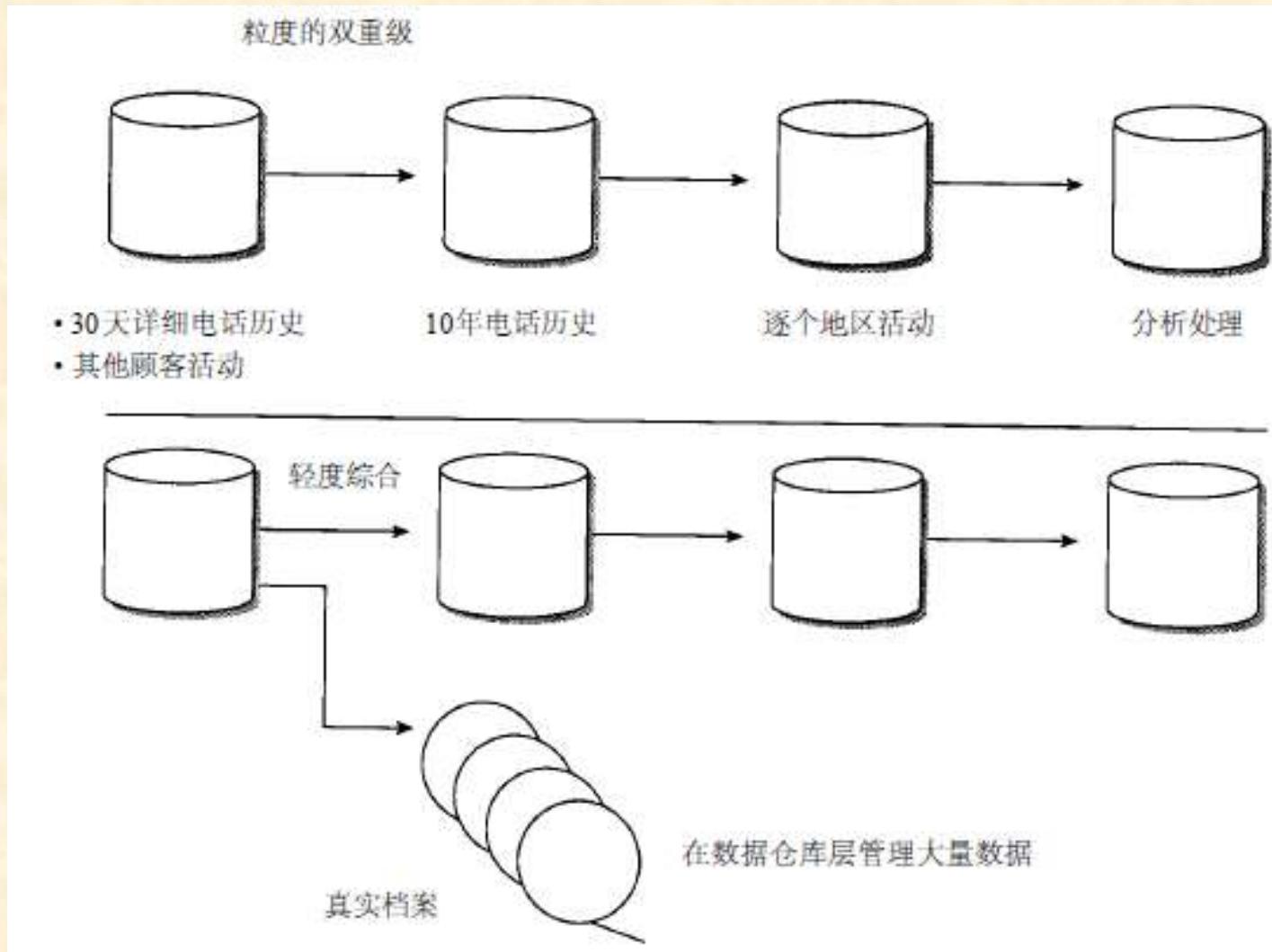
6 数据仓库的设计

- 高粒度，善于回答：“上个月人们从南京打出的长途电话平均多少个？”
- 低粒度，需要大量的资源来回答这个问题（查询每一条记录）

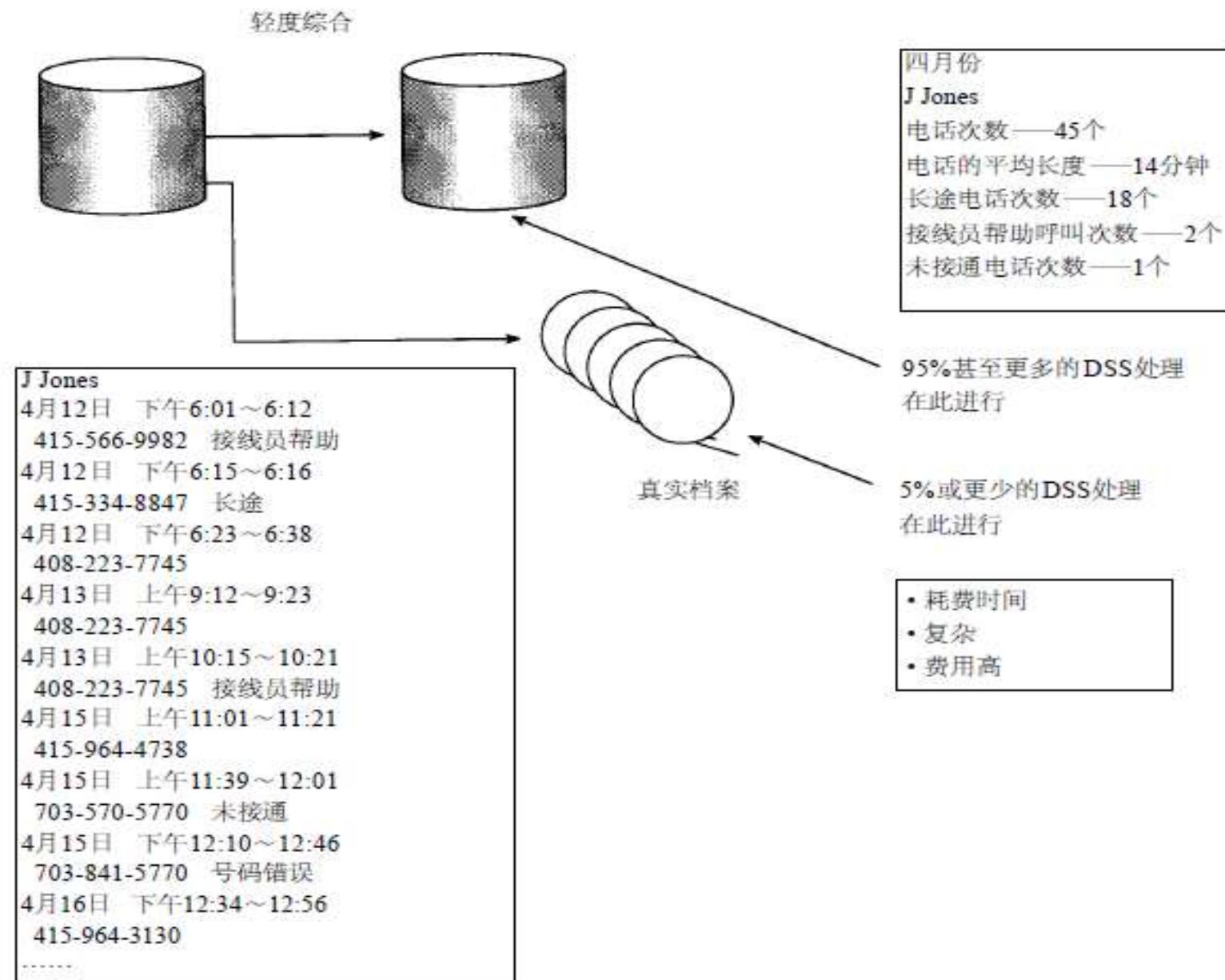


6 数据仓库的设计

➤ 轻度综合数据 + “真实档案” 细节数据

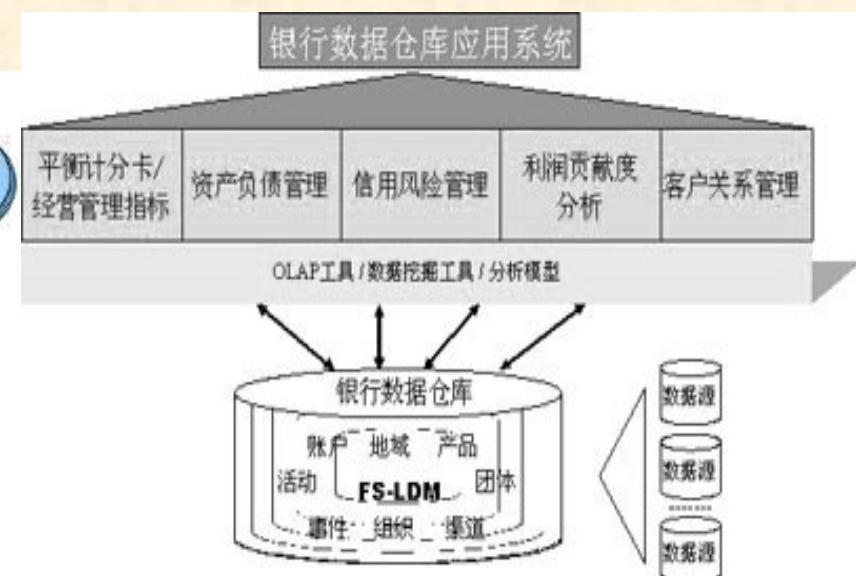


6 数据仓库的设计

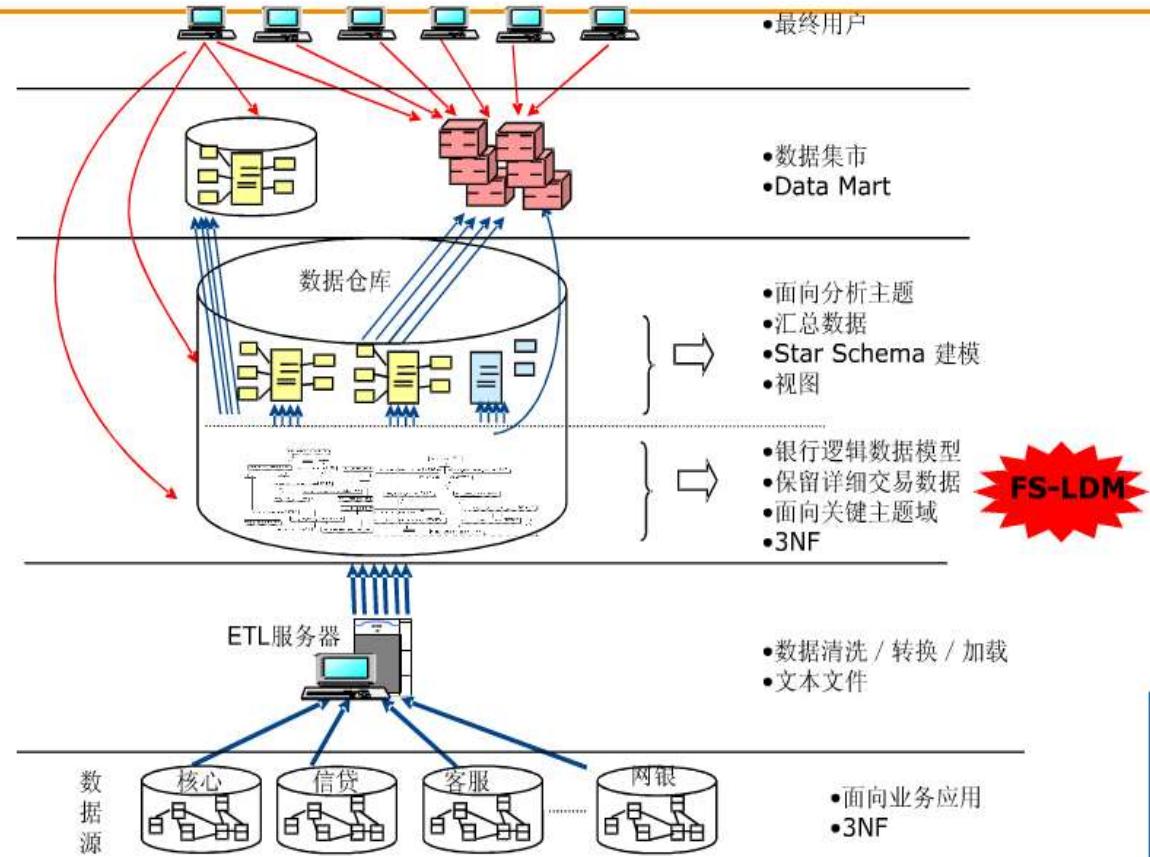


行业应用中的数据仓库设计模型示例

- 从业务角度看，相应的数据模型千差万别
- IBM、NCR对不同行业具有不同的数据仓库设计模型
 - 银行业
 - IBM : BDWM(Banking data warehouse model)
 - NCR : FS-LDM
 - 电信业
 - IBM : TDWM(Telecom Data warehouse model)
 - NCR : TS-LDM



LDM在数据仓库系统中的地位



Finance Subject Area



重要主题域在各业务领域的实例



Banking

- 个人客户
- 企业客户
- 同业客户
- 商户
-

- 储蓄帐户
- 定期帐户
- 信用卡帐户
- 贷款合同
-

- 查询 / 改密
- 存款 / 取款
- 转帐
- 投诉 / 咨询
-

Insurance

- 投保人
- 受益人
- 担保人
- 营销主管
-

- 家财险保单
- 健康险保单
- 货运险保单
- 人身险保险
-

- 查询 / 投诉
- 投保 / 续保
- 保费缴纳
- 理赔 / 赔付
-

Brokerage

- 投资者
- 证券公司
- 经济人
- 结算机构
-

- 股东帐户
- 结算备付金帐户
- 保证金帐户
- 服务协议
-

- 查询 / 咨询
- 买入 / 卖出
- 托管 / 配售
- 指定交易
-

行业应用中的数据仓库设计模型示例

➤ 共性



- 进行全面的业务梳理，改进业务流程
- 建立全方位数据视角，消灭信息孤岛和数据差异
- 实现上层业务变动和数据仓库之间的灵活性
 - 分离出底层技术的实现和上层业务的展现
 - 当上层业务发生变化时，通过数据模型，底层的技术实现可以非常轻松地适应业务的变动
- 帮助数据仓库系统本身的设计
 - 开发人员和业务人员达成系统建设范围的界定，以及长期目标规划，从而使整个项目组明确当前任务，加快整个系统建设的速度

行业应用中的数据仓库设计模型示例

➤ 数据模型架构

- 系统记录域：业务数据存储区
- 内部管理域：用于内部管理的元数据
- 汇总域：系统记录域的汇总，用于保证分析域的主题分析性能，满足部分的报表查询需求。
- 分析域：具体主题业务分析数据
- 反馈域：可选

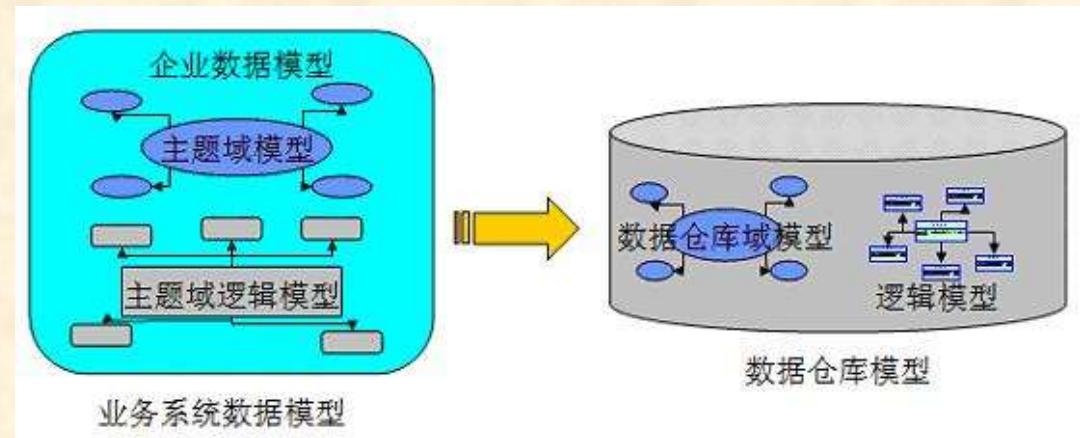


数据仓库数据模型架构

行业应用中的数据仓库设计模型示例

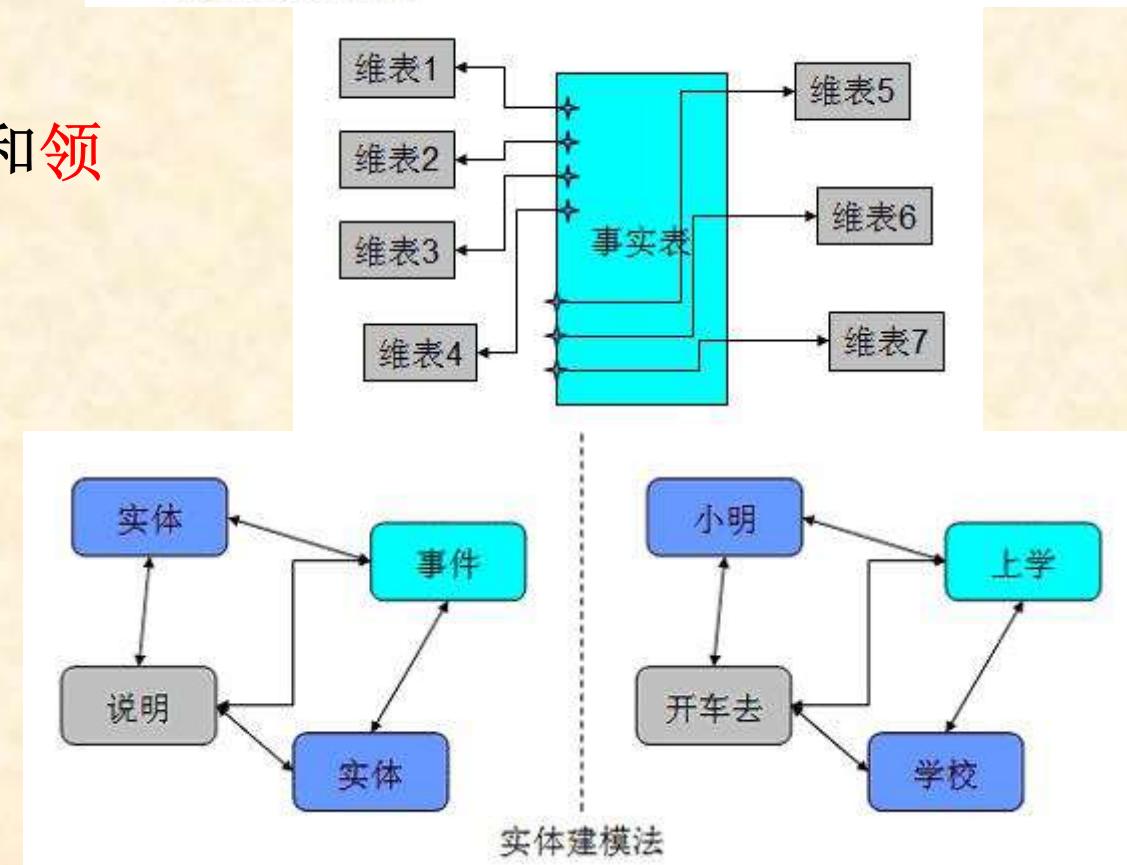
➤ 数据仓库建模方法

- 1. ? ? ?
- 2. ? ? ?
- 3. ? ? ?



➤ 实体建模法：E-R图

- 局限于**业务建模和领域概念建模阶段**



实例

【例】试画出销售分析的概念模型

解：首先根据销售分析的实际需求，确定信息包的维度、类别和指标与事实：

- (1) **维度**：包括日期维、销售地点维、销售产品维、年龄组别维、性别维等。
- (2) **类别**：确定各维的详细类别，如：日期维包括年(10)、季度(40)、月(120)等类别，括号中的数字分别指出各类别的数量；销售地点维包括国家(15)、区域(45)、城市(280)、区(880)、商店(2000)等类别，括号中的数字同样分别指出各类别的数量；类似地，可以确定销售产品、年龄组别维、性别维等的详细类别。
- (3) **度量和事实**：确定用于进行分析的数值化信息，包括预测销售量、实际销售量和预测偏差等。

销售分析的概念模型

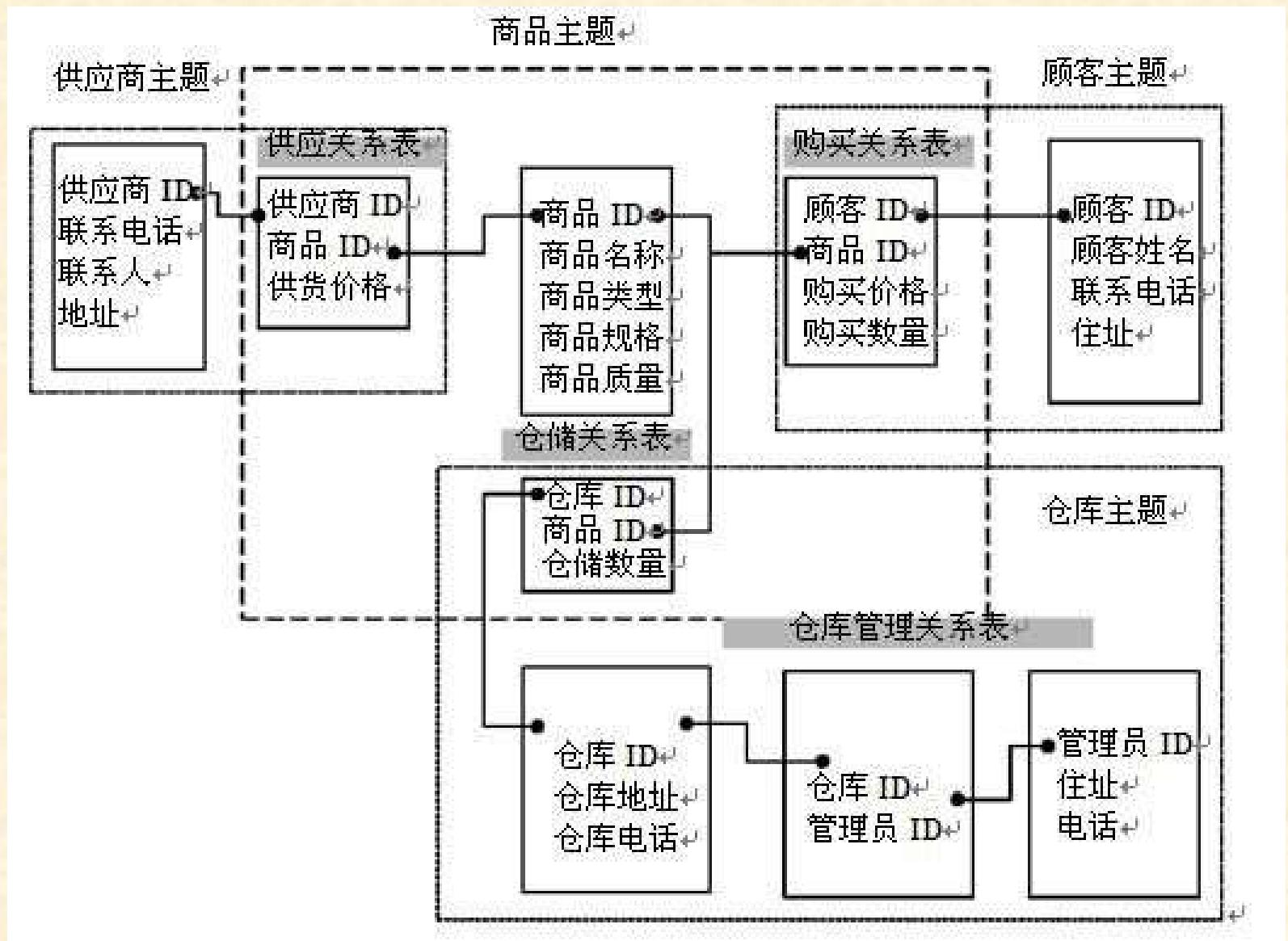
信息包： 销售分析

维度 

类别

日期	销售地点	销售产品	年龄组别	性别		
年(10)	国家(15)	产品类(6)	年龄组(8)	性别组(2)		
季度(40)	区域(45)	产品组(48)				
月(120)	城市(280)	产品(240)				
	区(880)					
	商店(2000)					
度量和事实：						
预测销售量、实际销售量、预测偏差						

概念模型图实例

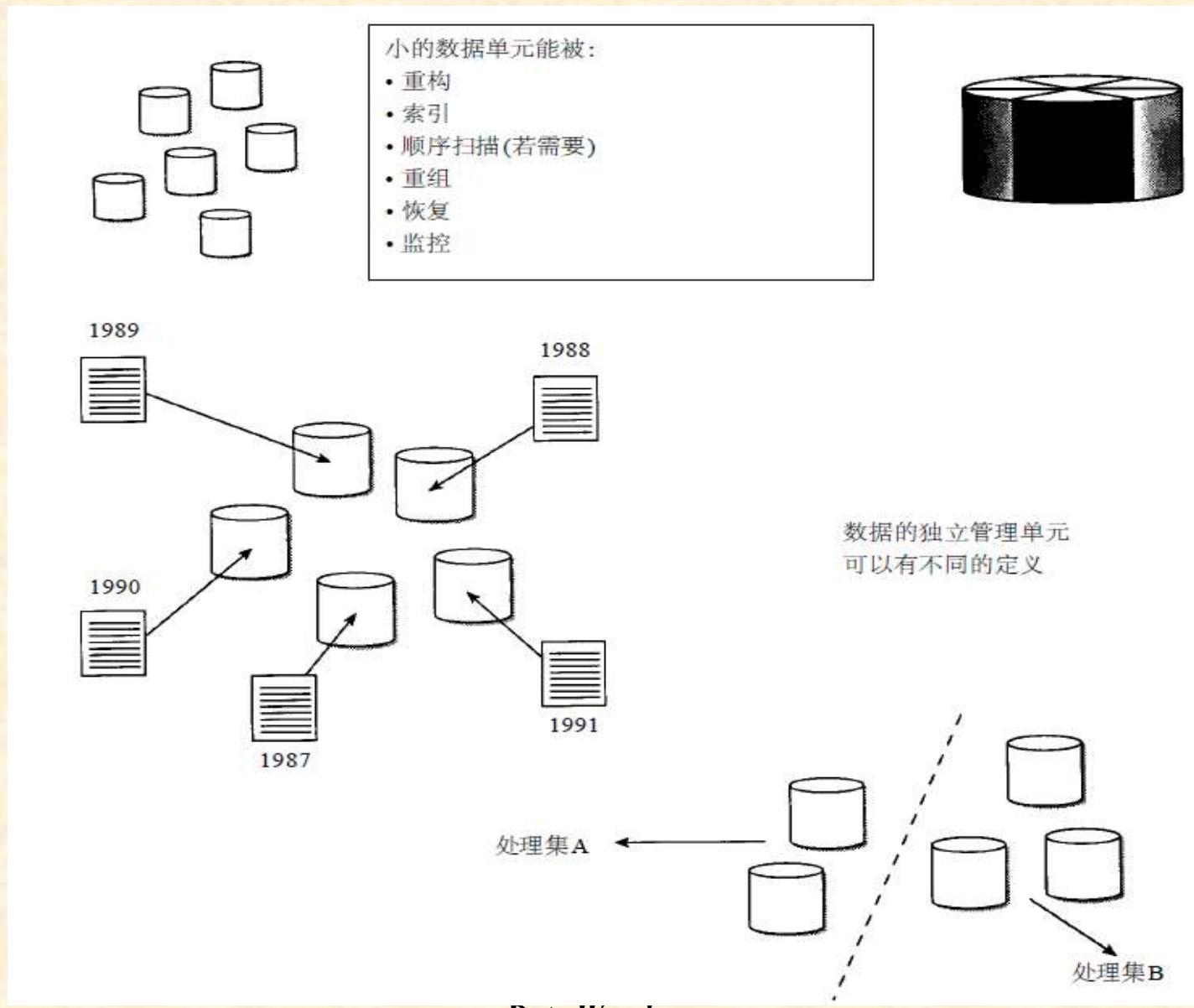


6 数据仓库的设计

❖ 逻辑模型设计 – 数据分割

- ✓ 数据分割是指把逻辑上是统一整体的数据分割成较小的、可以独立管理的物理数据单元进行存储，以便于重构、重组和恢复，以提高创建索引和顺序扫描的效率。
- ✓ 选择数据分割的因素有：
 - 数据量的大小
 - 数据分析处理的对象（主题）
 - 简单易行的数据分割标准
 - 数据粒度的划分策略
- ✓ 通常采用‘时间’属性作为数据分割的依据
- ✓ 类似于数据库中的数据分片技术，其目的是为了提高数据仓库的性能（数据仓库的本质之一：灵活地访问数据）

6 数据仓库的设计



6 数据仓库的设计

❖ 例子：人寿保险公司数据仓库的分割

- 1988年健康索赔。
- 1989年健康索赔。
- 1990年健康索赔。
- 1987年人寿保险索赔。
- 1988年人寿保险索赔。
- 1989年人寿保险索赔。
- 1990年人寿保险索赔。
- 1988年意外伤亡索赔。
- 1989年意外伤亡索赔。
- 1990年意外伤亡索赔。

“日期+索赔类型”

6 数据仓库的设计

❖ 逻辑模型设计 – 定义数据来源及其抽取规则

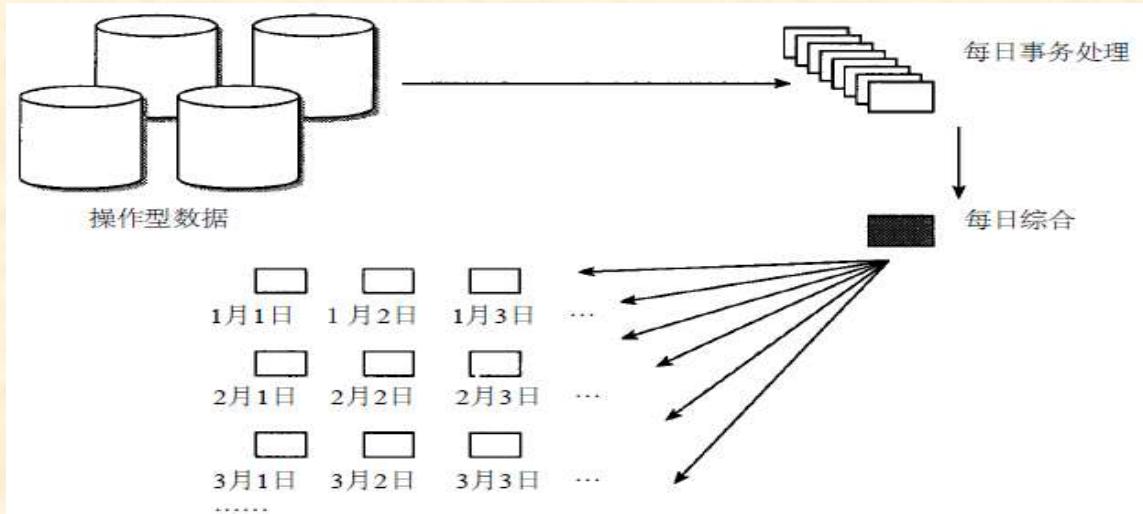
- ✓ 定义数据仓库中数据的来源，以及数据的抽取规则，例如：

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	采购子系统	商品	类别
.....

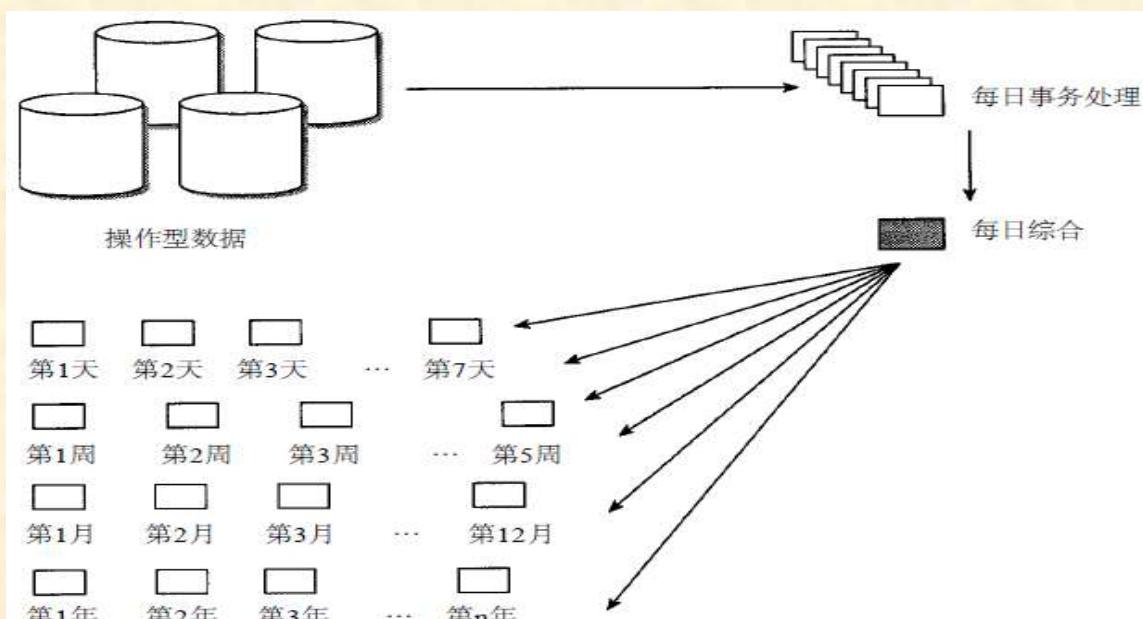
6 数据仓库的设计

❖ 逻辑模型设计 – 数据组织形式

- ✓ 简单堆积结构
 - 需要许多存储空间
 - 无细节丢失
 - 许多处理与数据有关



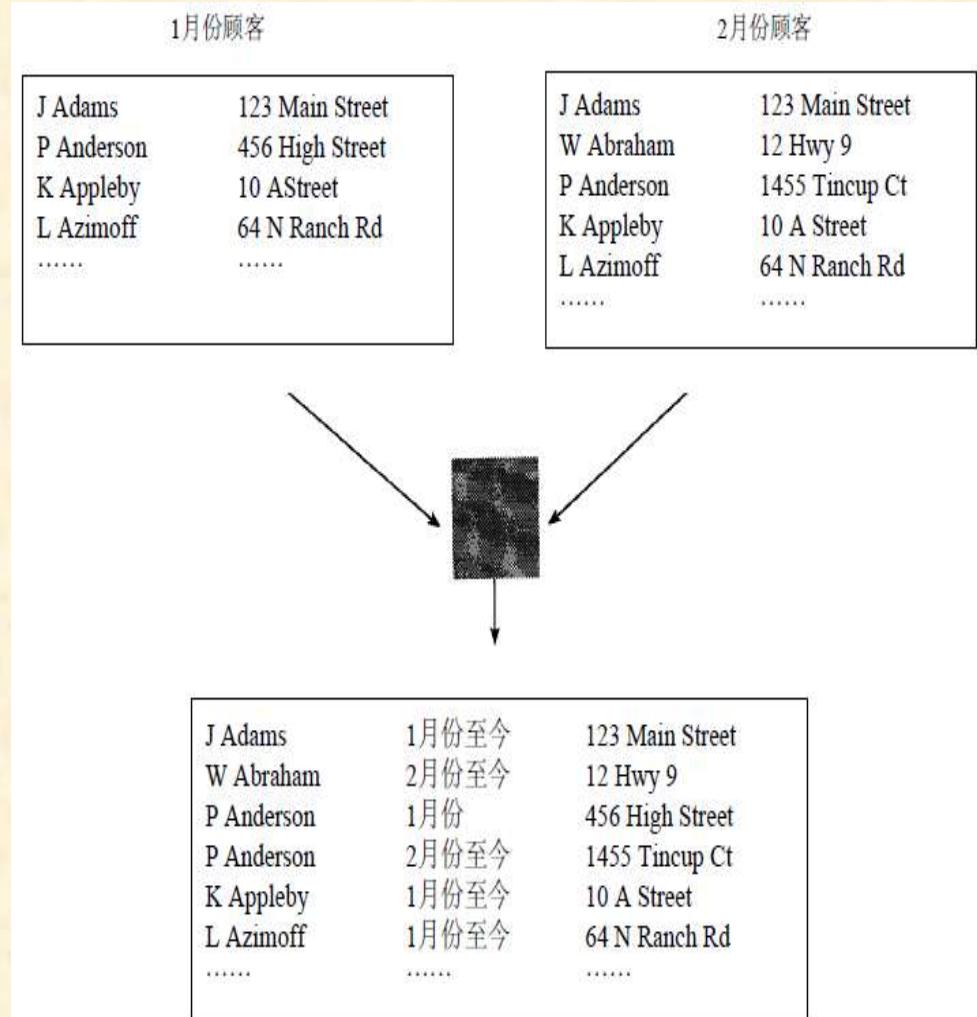
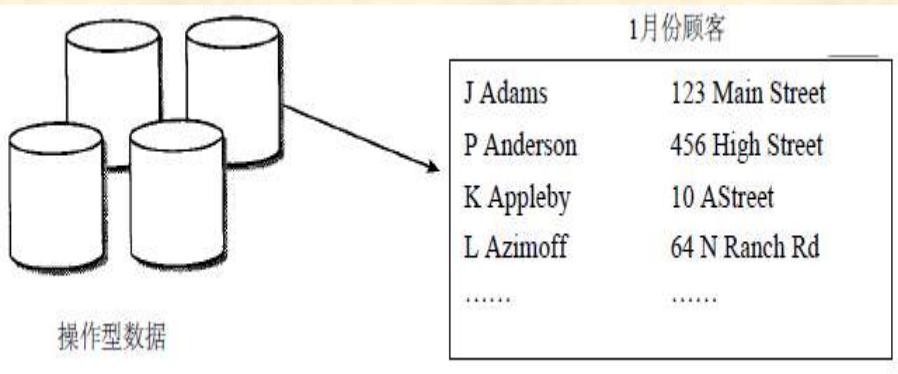
- ✓ 轮转综合结构
 - 非常紧凑
 - 一些细节丢失
 - 提取越久的数据，越不详细



6 数据仓库的设计

✓ 简单直接文件结构

- 直接拖入，无累积
- 以较长时间为单位的快照



6 数据仓库的设计

❖ 物理模型设计

在逻辑模型设计基础上确定数据的存储结构、确定索引策略、确定存储分配及数据存放位置等与物理有关的内容，物理模型设计的具体方法与数据库设计中的大致相似。其目的是为了提高数据仓库系统的访问性能。常用的一些技术有：

- 合并表
- 建立数据序列
- 引入冗余
- 表的物理分割
- 生成导出数据
- 建立广义索引

6 数据仓库的设计

❖ 物理模型设计 – 合并表

- 在常见的一些分析处理操作中，可能需要执行多表连接操作。为了节省I/O开销，可以把这些表中的记录混合存放在一起，以降低表的连接操作的代价。这样的技术被称为 合并表。
- 合并表技术与传统关系数据库中的集簇(Clustering)技术类似。

6 数据仓库的设计

❖ 物理模型设计 – 建立数据序列

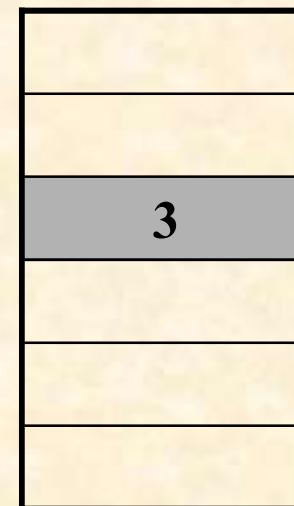
- 按照数据的处理顺序调整数据的物理存放位置，以减少系统的磁盘I/O的开销。



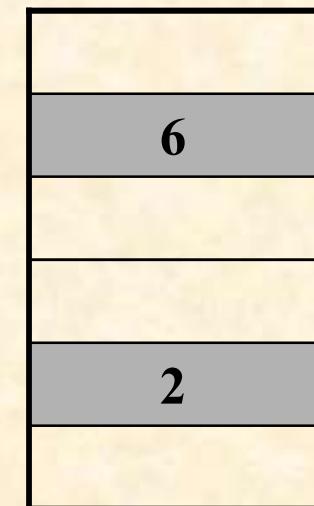
page1



page2



page3



page4

初始存储组织的数据分布图
(按编号顺序访问数据块)

6 数据仓库的设计

❖ 物理模型设计 – 建立数据序列

1
2
3
4
5
6

page1

7
8

page2

page3

page4

调整后的存储组织数据分布图

6 数据仓库的设计

❖ 物理模型设计 – 引入冗余

- 在面向某个主题的分析过程中，通常需要访问不同表中的多个属性，而每个属性又可能参与多个不同主题的分析过程。因此可以通过修改关系模式把某些属性复制到多个不同的主题表中去，从而减少一次分析过程需要访问的表的数量。

- 采用该种数据组织方法会带来大量的数据冗余存储，数据仓库系统必须保证这些冗余数据的一致性。由于数据仓库中的数据是稳定的，很少执行更新操作，不会因此带来过高的数据更新的代价，但可以有效地提高数据仓库系统的性能。

6 数据仓库的设计

❖ 物理模型设计 – 表的物理分割

- 类似于在逻辑设计阶段的数据分割
- 可以根据表中每个属性数据的访问频率和稳定性程度对表的存储结构进行分割
 - 对于访问频率较高的属性，可以单独考虑其物理存储组织，以便选择合适的索引策略和特定的物理组织方式。
 - 对于需要频繁更新的属性，也可以单独组织其物理存储，以免因数据更新而带来的空间重组、重构等工作。

6 数据仓库的设计

❖ 物理模型设计 – 生成导出数据

- 在原始的细节数据的基础上进行一些统计和计算，生成导出数据，并保存在数据仓库中。
- 采用该方法既可以：
 - 避免在分析过程中执行过多的统计或计算操作，减少输入/出的次数；
 - 避免了不同用户进行的重复统计操作可能产生的偏差。

6 数据仓库的设计

❖ 物理模型设计 – 建立广义索引

- 用于记录数据仓库中数据与‘最’有关的统计结果的索引被称为‘广义索引’。如：
 - 当月销售额最高的商店？
 - 当月销售情况最差的商品？
 -
- 这样的广义索引的数据量是非常小的，可以在每次进行数据仓库数据加载工作时生成或刷新这样的广义索引。用户可以从已经建立的广义索引里直接获取这些统计信息，而不必对整个数据仓库进行扫描。

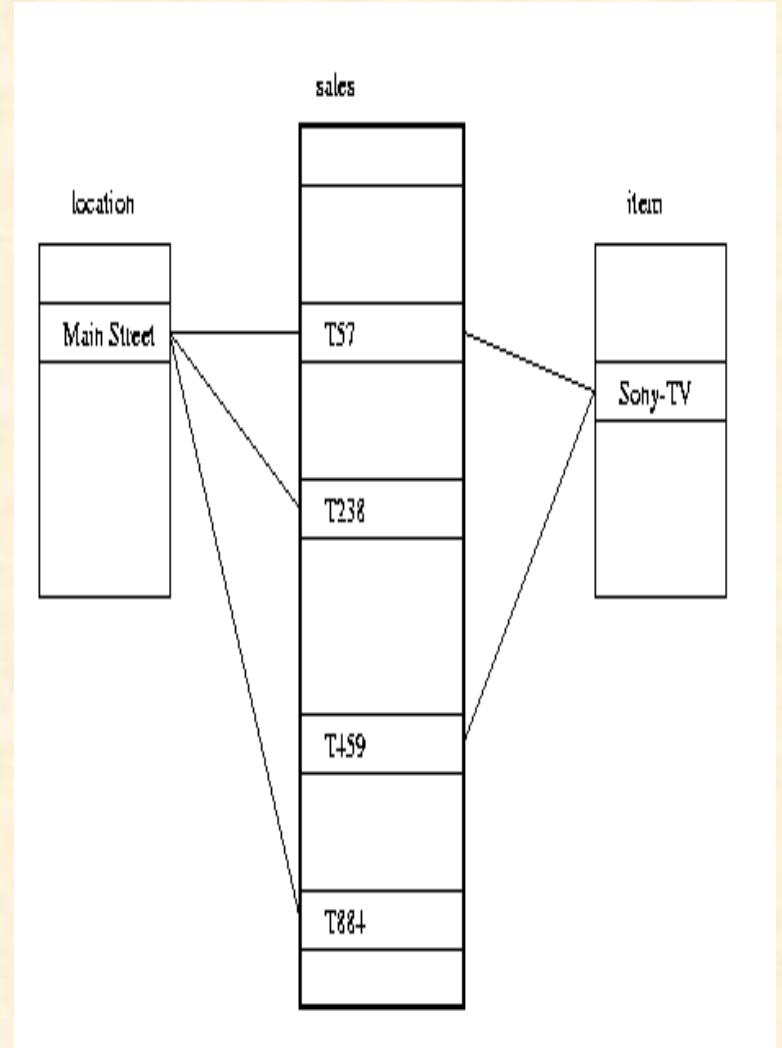
Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The i -th bit is set if the i -th row of the base table has the value for the indexed column
- not suitable for high cardinality domains

Base table			Index on Region				Index on Type		
Cust	Region	Type	RecID	Asia	Europe	America	RecID	Retail	Dealer
C1	Asia	Retail	1	1	0	0	1	1	0
C2	Europe	Dealer	2	0	1	0	2	0	1
C3	Asia	Dealer	3	1	0	0	3	0	1
C4	America	Retail	4	0	0	1	4	1	0
C5	Europe	Dealer	5	0	1	0	5	0	1

Indexing OLAP Data: Join Indices

- Traditional indices:
map the values to a list of record ids
- Join index for data warehouse
relates the values of **the dimensions** to **rows** in the fact table.
 - Join indices can span multiple dimensions

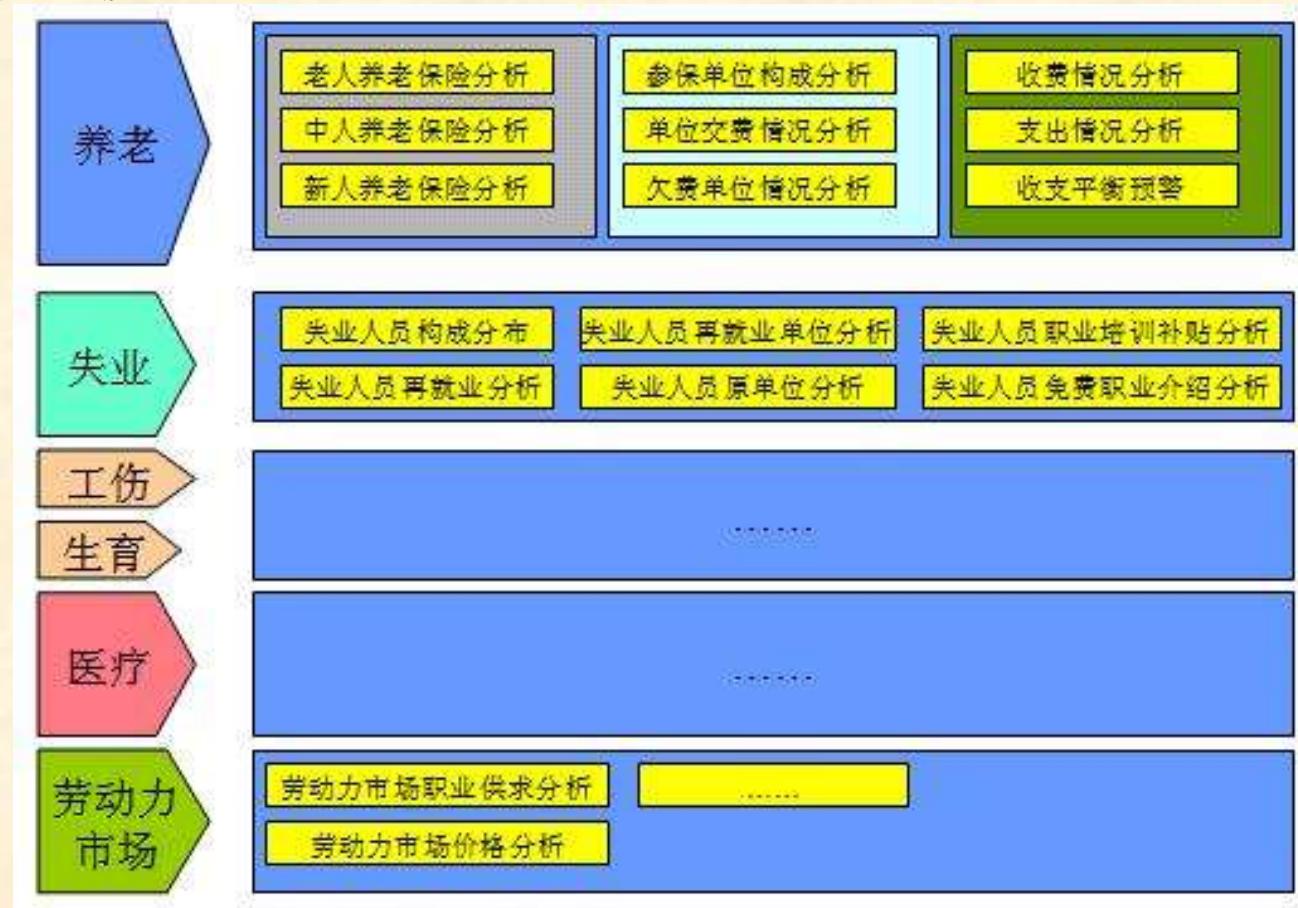


数据仓库建模实例2

➤ 社保行业

- 业务领域：养老、失业、工伤、生育、医疗保险、劳动力市场

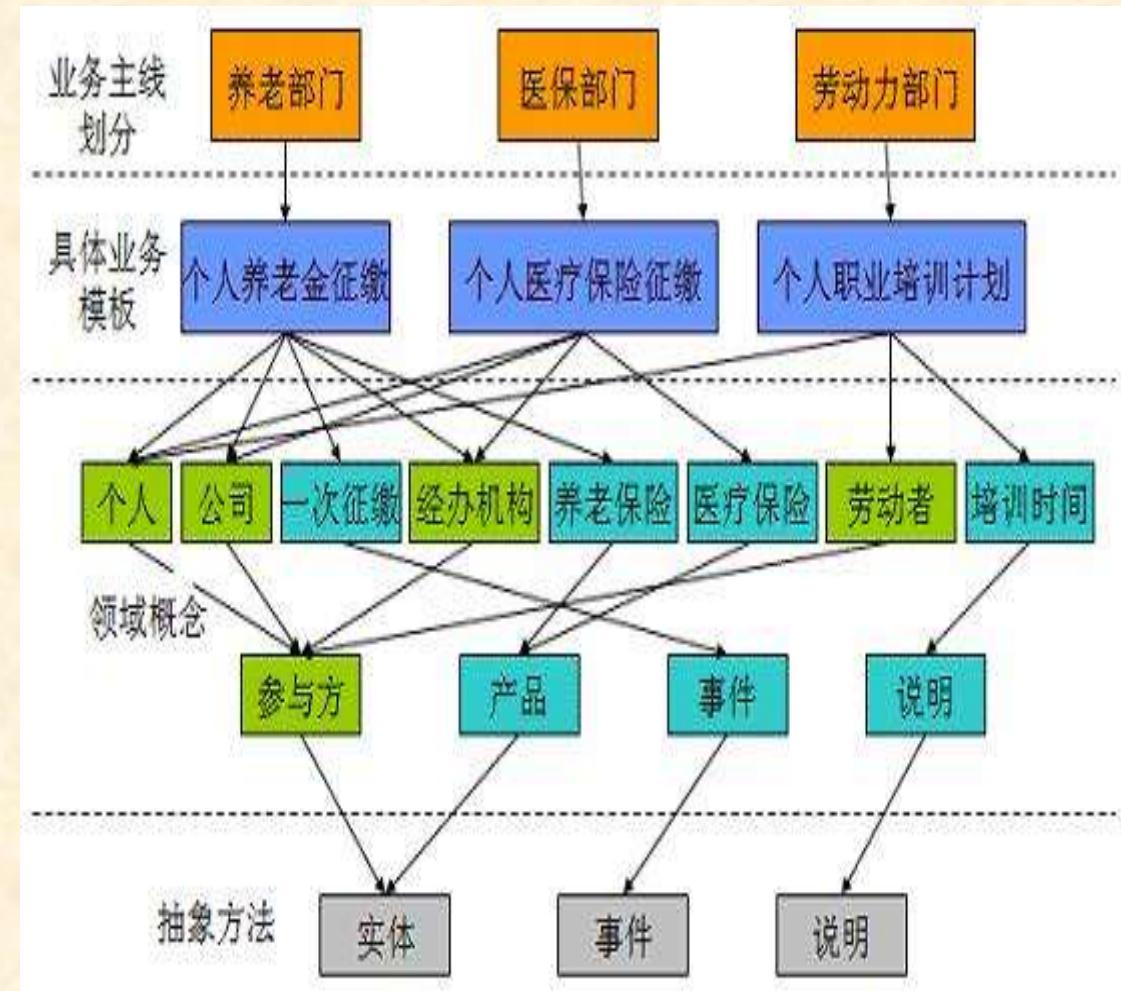
➤ 1. 业务建模



数据仓库建模实例2

➤ 2.领域概念建模：实体建模法

- 抽象方法层
- 领域概念层
 - 选用抽象概念较低的实体，方便业务人员和技术人员之间的交流和沟通
- 具体业务层
 - 实体的不同组合
- 业务主线层



数据仓库建模实例2

➤ 3. 逻辑建模

- 实例化每一个抽象的实体
 - “人”、“公司”等
 - 考虑其属性有哪些：年龄、性别、受教育程度等
- 找出抽象实体间的联系，并将其实例化
- 找出抽象事件的关系，并对其进行说明
 - “事件”中的地域、事件等因素的考量等
- 3NF 建模法
 - 采用 ERWIN 等建模工具绘制符合 3NF 的关系型数据模型

数据仓库建模实例2

- 关系
- 元组
- 属性
- 主码/主键
- 域
 - 属性的取值范围;
- 分量
 - 元组中的一个属性组;
- 关系模式
 - 对关系的描述，用关系名（属性名1， 属性名2，， 属性名n）

数据仓库建模实例2

➤ 4.物理建模

- 因数据仓库平台而异
- 1. 生成创建表的脚本
- 2. 针对不同的数据仓库平台，进行相应的优化
 - DB2 数据仓库：创建 MQT 表加速报表生成等
 - Oracle、SQL Server 等不同优化
- 3. 针对数据集市的需要，按照维度建模的方法，生成一些事实表、维表等
- 4. 针对数据仓库的 ETL 过程和元数据管理的需要，生成一些数据仓库维护的表，例如：日志表

数据仓库建模实例2

➤ 4. 物理建模

- 物理存取方式、数据存储结构、数据存放位置（高速/低速设备）、存储分配（块尺寸、缓冲区大小和个数）等
- 物理模型设计实现时，考虑的主要因素
 - I/O存取时间、存储空间利用率和维护代价
- 物理模型设计时的性能优化手段
 - 考虑：数据仓库的数据量大，但操作单一
 - 合并表
 - 建立数据序列
 - 引入冗余
 - 进一步细分数据
 - 生成导出数据
 - 建立专用负责的索引，如广义索引（一旦建立不需维护）

6 数据仓库的设计

❖ 数据仓库生成

➤ 建立数据模式

- 根据逻辑设计与物理设计的设计结果建立数据仓库的数据模式。

➤ 编制数据抽取程序

- 根据数据仓库元数据中的定义信息，编制抽取程序，将数据源中的数据作加工以形成数据仓库中的数据。

➤ 数据加载

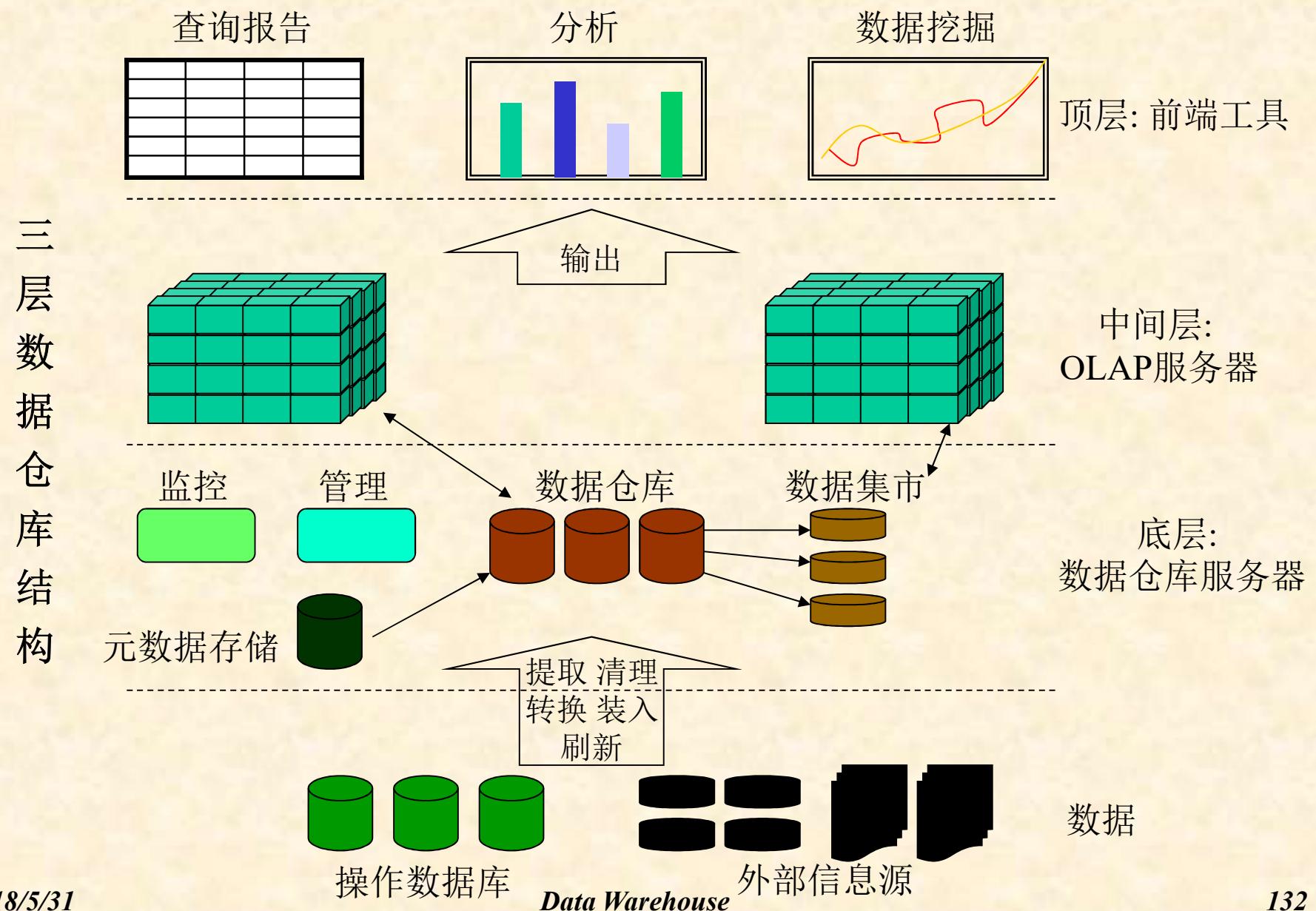
- 将数据源中的数据，通过数据抽取程序加载到数据仓库的模式中去。

6 数据仓库的设计

❖ 三层数据仓库结构

- 底层是数据仓库服务器（一般为一个关系数据库系统）
- 中间层是**OLAP**服务器, 它典型的实现是:
 - 关系**OLAP**模型, 即扩充的关系**DBMS**, 它将多维数据上的操作映射为标准的关系操作;
 - 或者是多维**OLAP**模型, 一种特殊的服务器, 它直接实现多维数据和操作
- 顶层是客户, 它包括查询和报告工具, 分析工具和/或数据挖掘工具(如趋势分析, 预测等).

6 数据仓库的设计



6 数据仓库的设计

❖ 数据仓库的使用与维护

- 数据仓库建立后，即可建立分析、决策型的应用系统。
- 在应用系统的使用过程中不断加深理解，改进主题，依照原型法的思想使系统更趋完善。
- 在系统的运行过程中，随着数据源中数据的不断变化，需要通过**数据刷新**操作来维护数据仓库中数据的一致性。
- 通常，数据仓库使用的时间越长，进化地越好。
- 从产生报告和回答预先定义的查询—分析数据，提供报表和图表—决策，多维分析和复杂的切片和切块—知识发现，并使用数据挖掘工具进行决策。

6 数据仓库的设计

❖ 管理存储多介质的大量数据

- 高效索引
- 数据的并行存储/管理
- 元数据管理
- 数据的高效装载
- 数据压缩

在数据仓库环境中I / O资源比C P U资源少得多，
解压缩影响不大。

6 数据仓库的设计

- 数据仓库的关键是数据的存储和管理
- 数据仓库一般遇到的几个问题都与此有关：
 - 大数据量的存储和管理
 - 并行处理
 - 针对决策支持查询的优化
 - 支持多维分析的查询方式

6 数据仓库的设计

❖ 数据分层的意义

- **清晰数据结构**: 每一个数据分层都有它的作用域, 这样我们在使用表的时候能更方便地定位和理解。
- **数据血缘追踪**: 业务表的来源众多, 如果一张表出问题了, 能够快速准确地定位到问题, 并清楚它的危害范围。
- **减少重复开发**: 规范数据分层, 开发一些通用的中间层数据, 能够减少极大的重复计算。
- **把复杂问题简单化**: 将复杂的任务分解成多个步骤来完成, 每一层只处理单一的步骤。当数据出现问题之后, 可以不用修复所有的数据, 只需要从有问题的步骤开始修复。
- **屏蔽原始数据的异常**
- **屏蔽业务的影响**: 不必改一次业务就需要重新接入数据。

6 数据仓库的设计

❖ 数据分层的意义

分层前



分层后



6 数据仓库的设计

❖ 如何分层

- 一般数据仓库分为下面三个层，即：数据运营层、数据仓库层和数据产品层。



6 数据仓库的设计

❖ 操作数据存储（ODS）

- 在许多情况下，DB-DW的两层体系结构并不适合企业的数据处理要求。因为，虽然可以粗略地把数据处理分成操作型和分析型，但这两种处理并不是泾渭分明的。
- ODS（Operational Data Store）作为一个中间层次，一方面，它包含企业全局一致的、细节的、当前的或接近当前的数据，另一方面，它又是一个面向主题、集成的数据环境，适合完成日常决策的分析处理。

6 数据仓库的设计

❖ 操作数据存储（ODS）

- 为了弥补业务系统和数据仓库之间的数据同步差距而提出
- 是数据仓库体系结构中的一个可选部分，具备数据仓库的部分特征和OLTP 系统的部分特征
- “面向主题的、集成的、当前或接近当前的、不断变化的” 数据
- 最大特点是数据是可更新的，甚至由业务系统通过触发器直接更新

6 数据仓库的设计

❖ 数据仓库层 (DW)

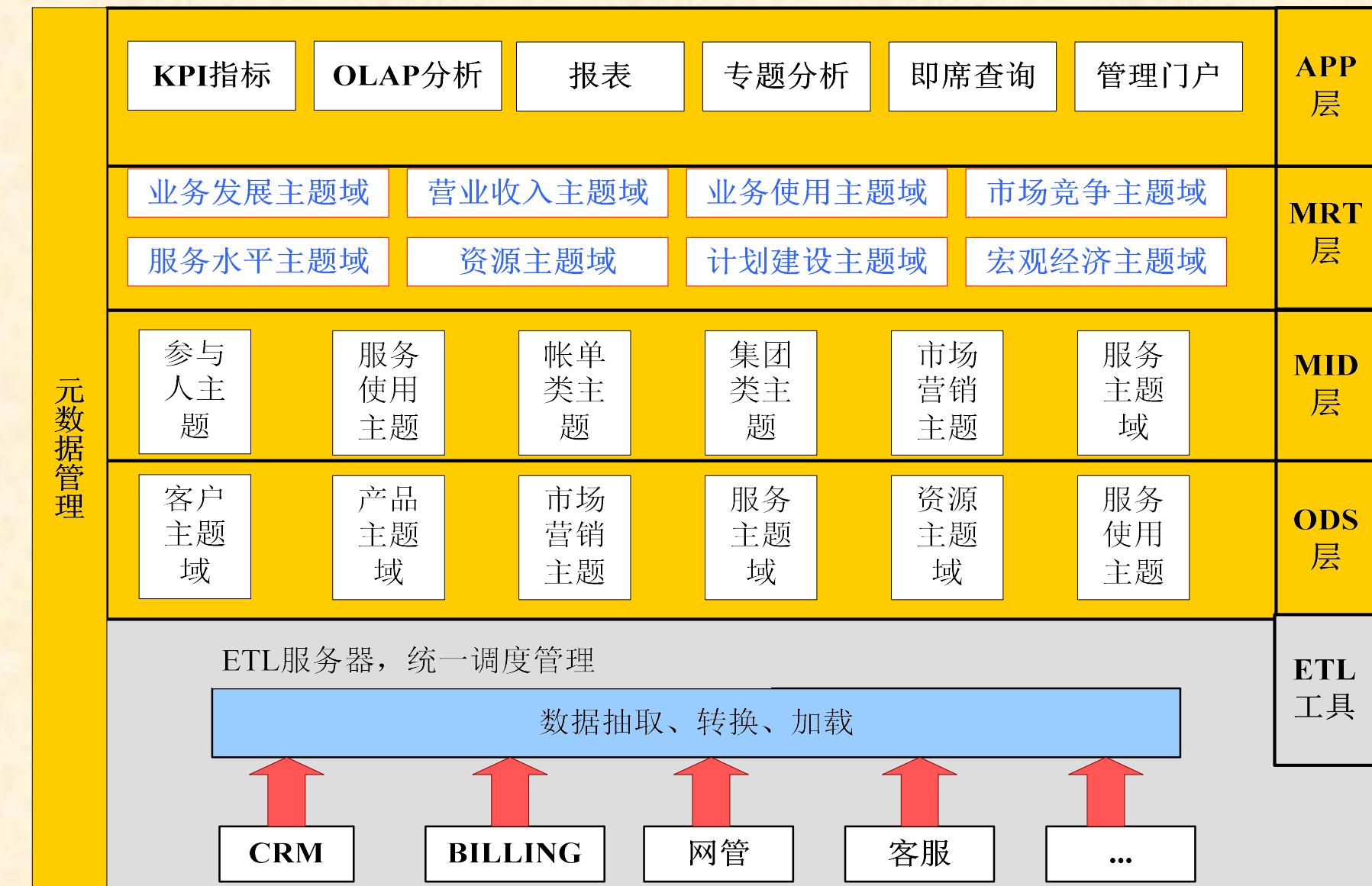
- 是数据仓库的主体。从 ODS 层中获得的数据按照主题建立各种数据模型。例如以研究人的旅游消费为主题的数据集中，便可以结合航空公司的登机出行信息，以及银联系统的刷卡记录，进行结合分析，产生数据集。在这里，我们需要了解四个概念：维（dimension）、事实（Fact）、指标（Index）和粒度（Granularity）

6 数据仓库的设计

❖ 数据产品层（APP）

- 这一层是提供为数据产品使用的结果数据
- 在这里，主要是提供给数据产品和数据分析使用的数据，一般会存放在 Es、Mysql 等系统中供线上系统使用，也可能会存在 Hive 或者 Druid 中供数据分析和数据挖掘使用。
- 常说的报表数据，一般就放在这里。

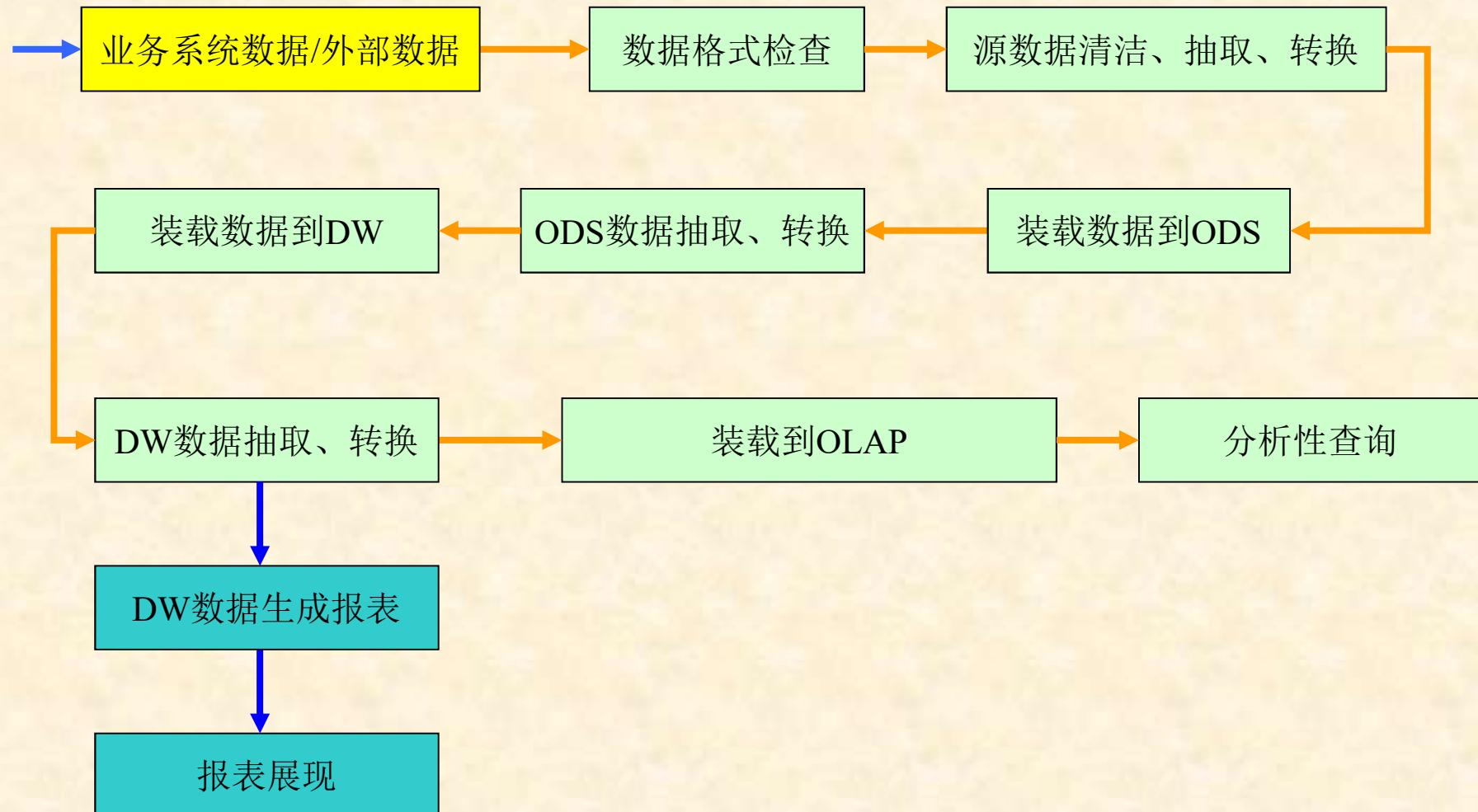
实例：数据仓库体系结构



实例：数据仓库层次描述

	STAGE层 (数据缓冲层)	ODS层 (明细数据层)	MID层 (数据挖掘层)	MRT层 (分析应用层)
作用	提供业务系统数据文件的临时存储。	提供业务系统细节数据的长期沉淀；为未来分析类需求的扩展提供历史数据支撑；支撑中间汇总层数据生成	支撑DM层数据生成；方便应用需求处理，提高性能；支撑专题分析和数据挖掘	面向分析类应用所构建的数据存储；为报表、KPI、OLAP和指标体系等应用提供数据支撑
数据模型	与业务系统一致	3NF,与企业级数据模型一致	介于DM与DW之间，反范式设计，增加数据冗余	多维模型
数据存储粒度	存储业务系统数据的原始粒度	存储详单、客户资料等细节数据的原始粒度；经过转换处理后的数据	对用户等数据的轻度加工	中度、高度汇总数据
数据周期	临时性	长期保留，详单类可考虑6个月左右	长期保留	原则上保留所有数据

实例：数据仓库数据处理流程

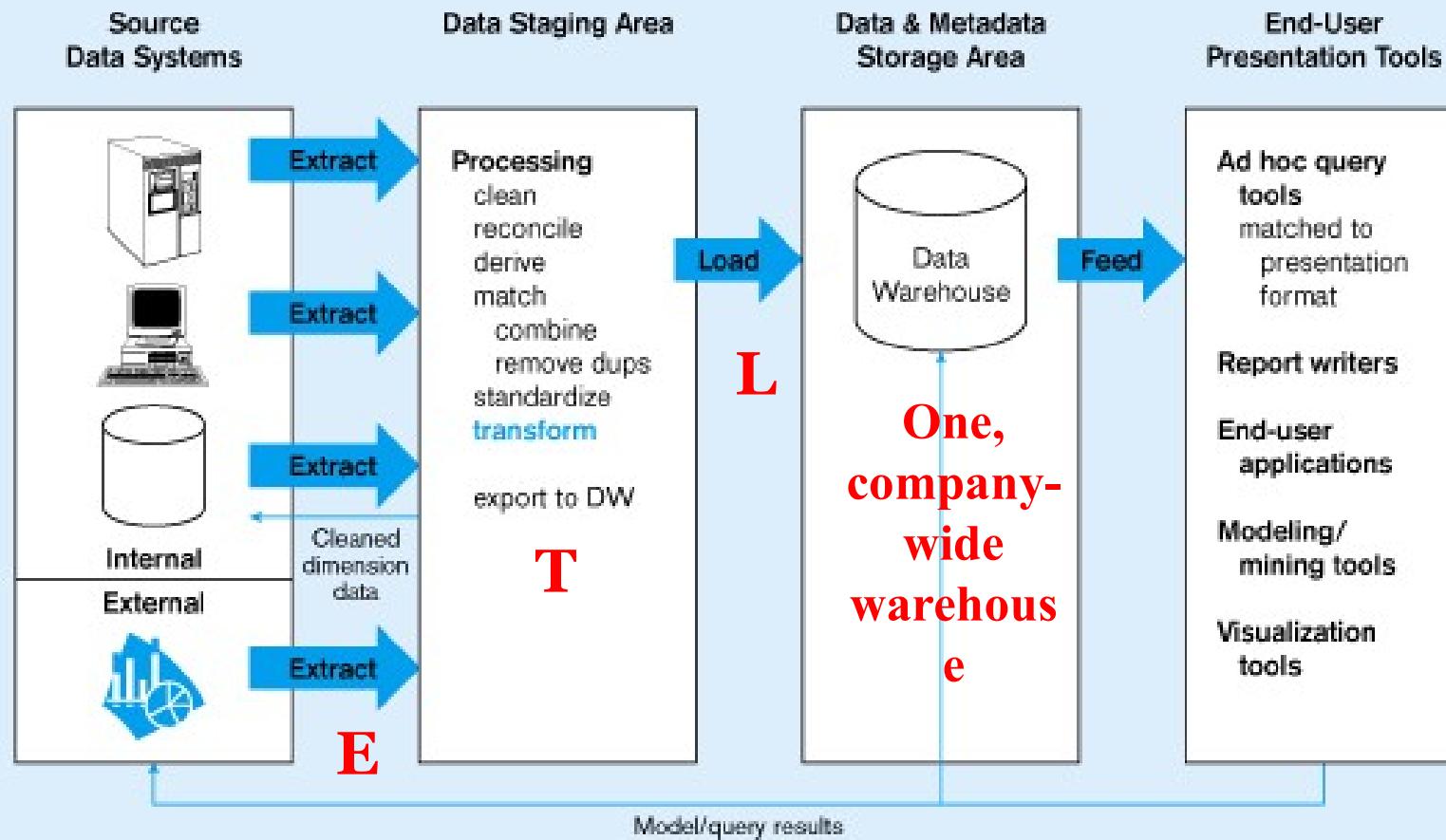


7 ETL

❖ ETL定义

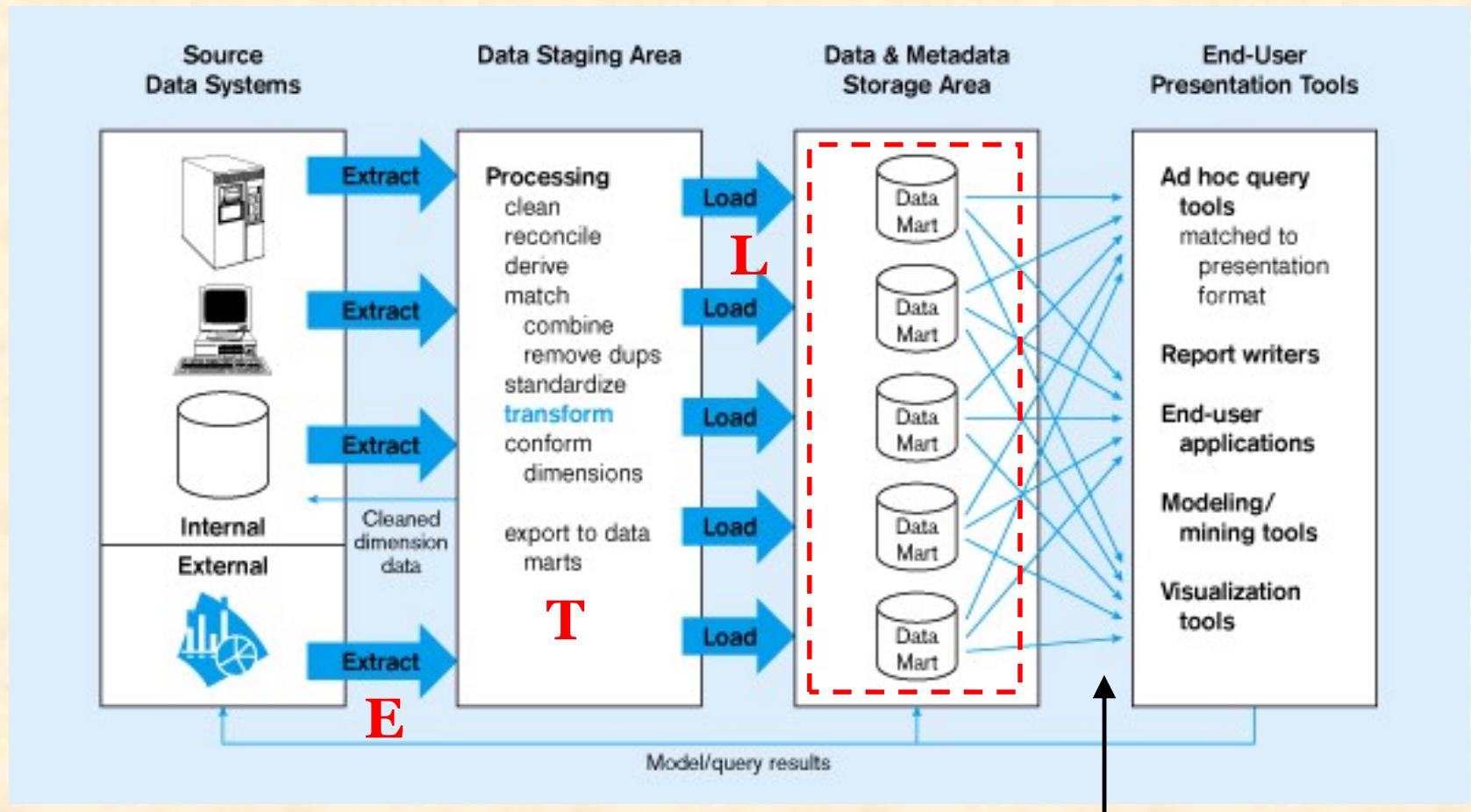
- ❖ Extract-Transform-Load
- ❖ 数据抽取（Extract）、转换（Transform）、装载（Load）的过程。
- ❖ 用户从数据源抽取出所需的数据，经过数据清洗、转换，最终按照预先定义好的数据仓库模型，将数据加载到数据仓库中去
- ❖ ETL是BI/DW的核心和灵魂，按照统一的规则集成并提高数据的价值，是负责完成数据从数据源向目标数据仓库转化的过程，是实施数据仓库的重要步骤
- ❖ ETL是数据仓库系统中最重要的概念之一，ETL在一个数据仓库系统项目中要花一半以上的时间

ETL for Data Warehouse



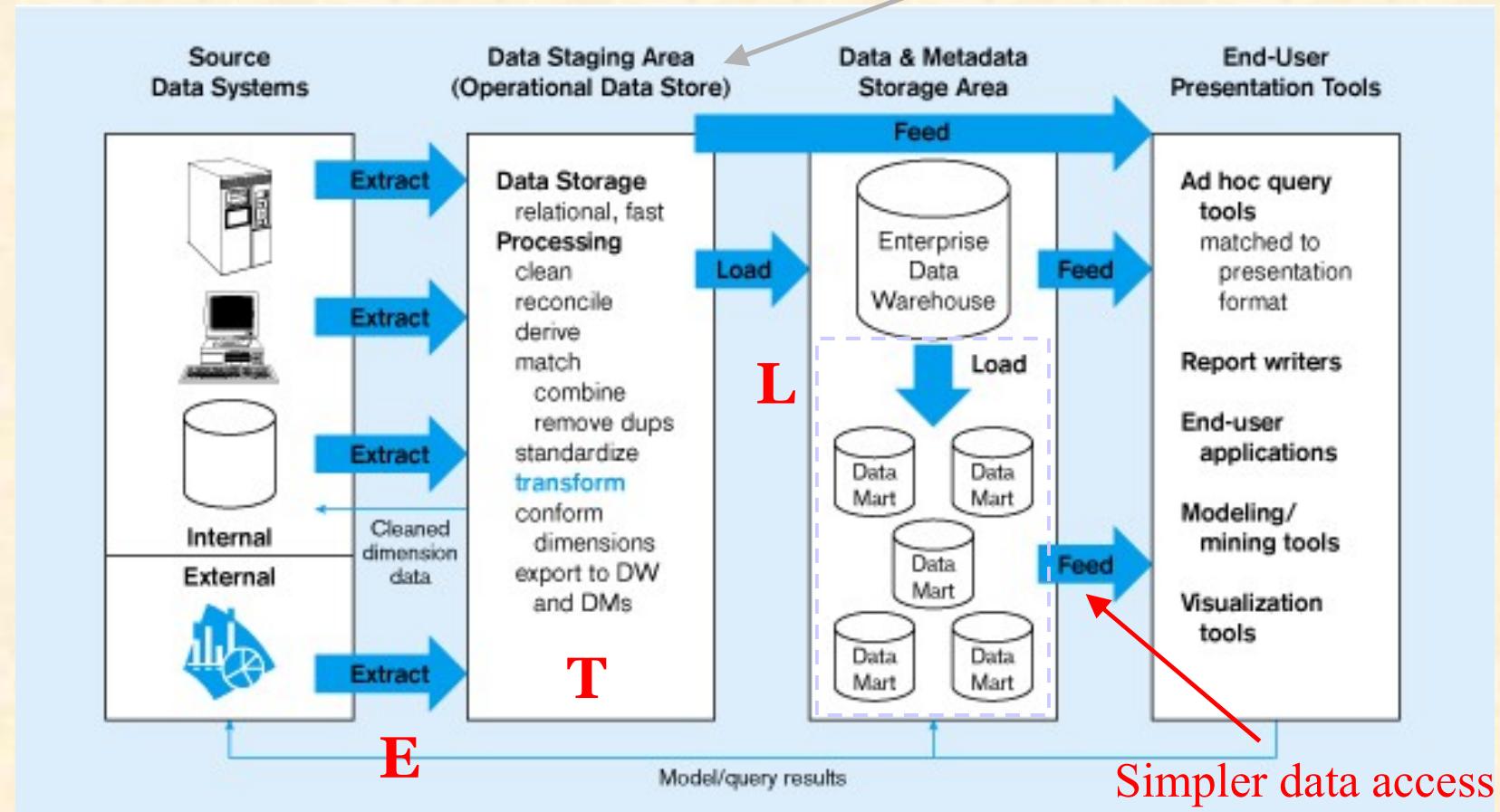
ETL for Independent Data marts:

Mini-warehouses, limited in scope



ETL for *Dependent data mart with operational data store*

ODS provides option for obtaining *current* data

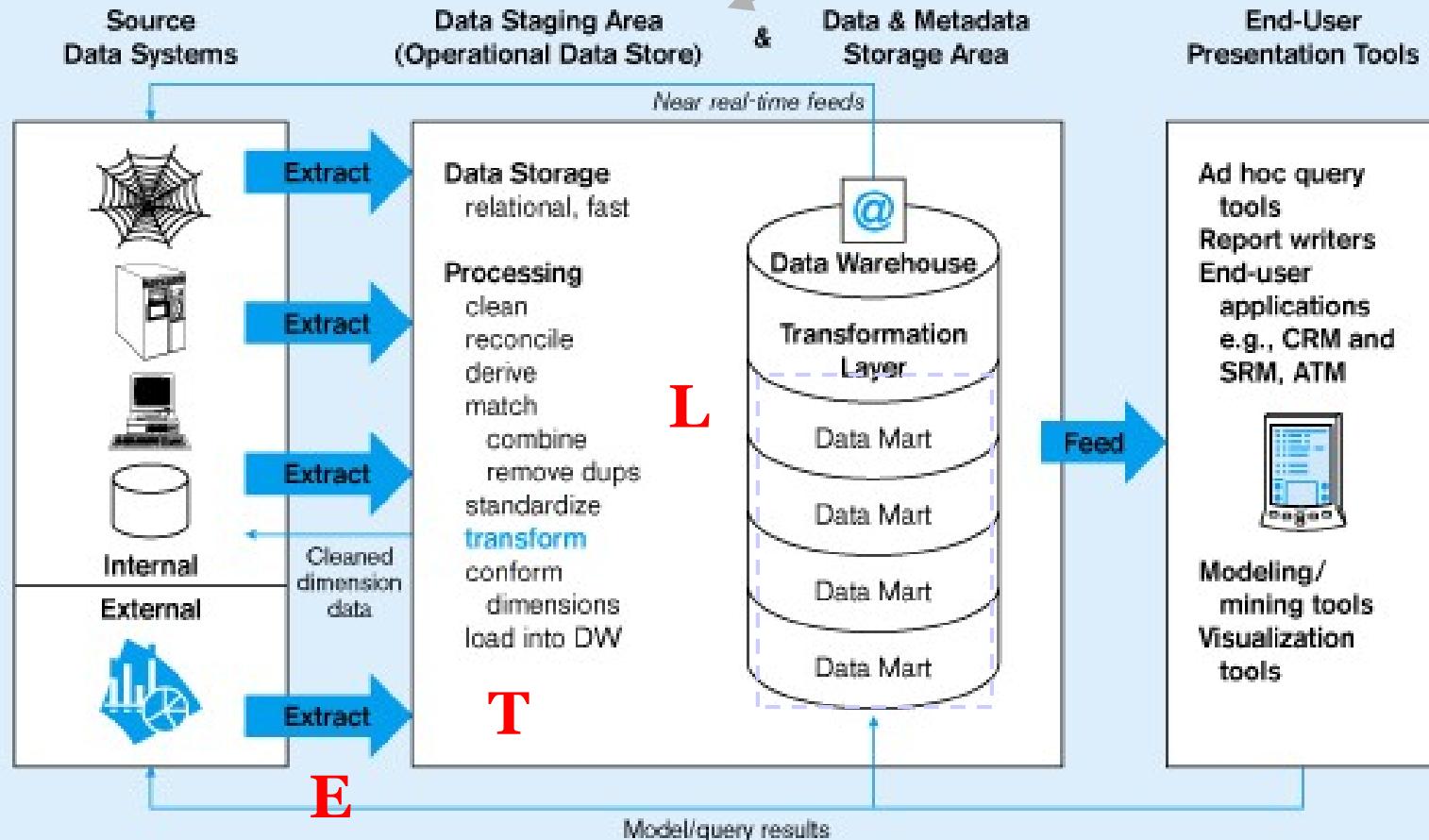


Single ETL for
dependent data mart with operational data store
(EDW)

Dependent data marts
loaded from EDW

ETL for Logical data mart and Active/RT data warehouse

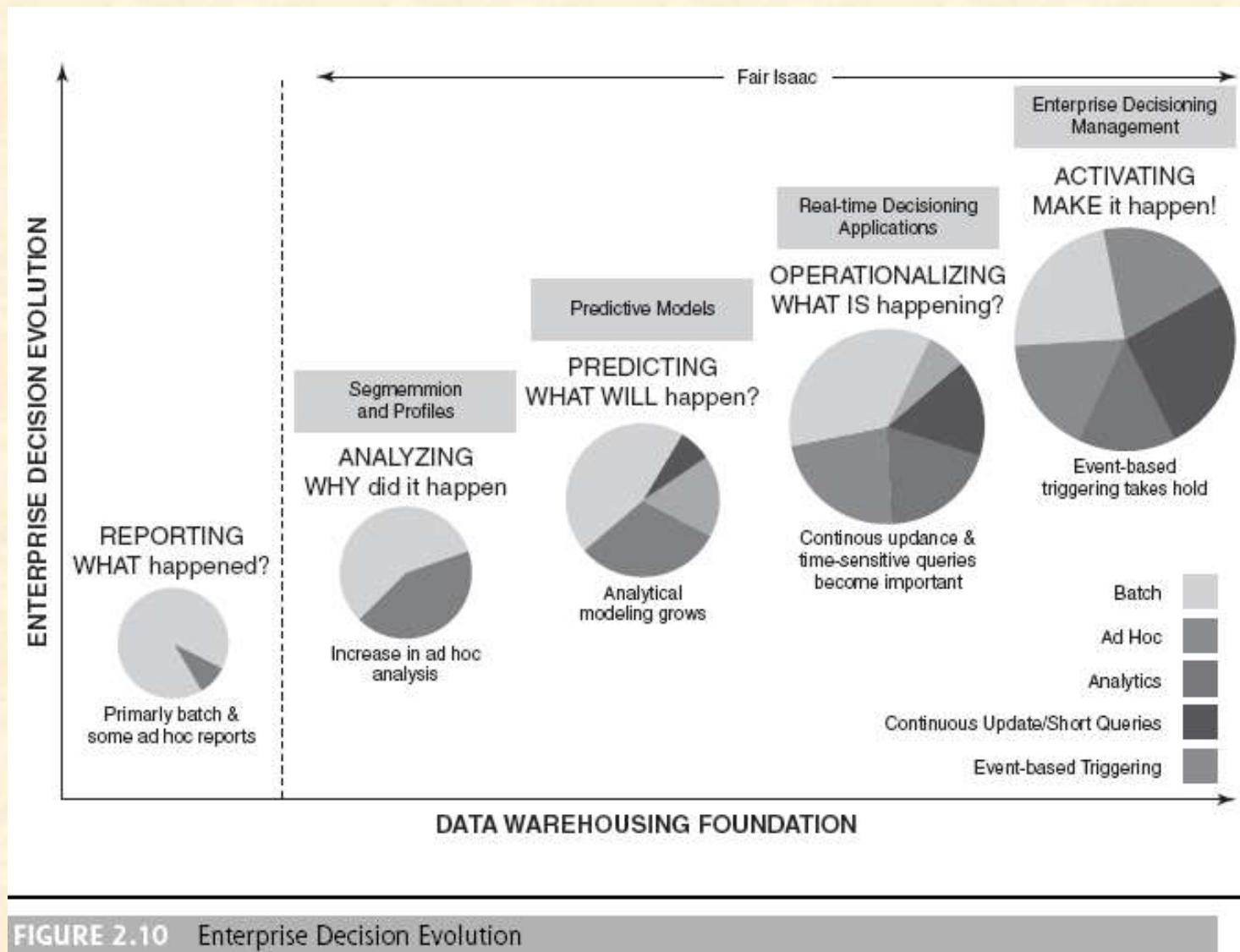
ODS and data warehouse
are one and the same



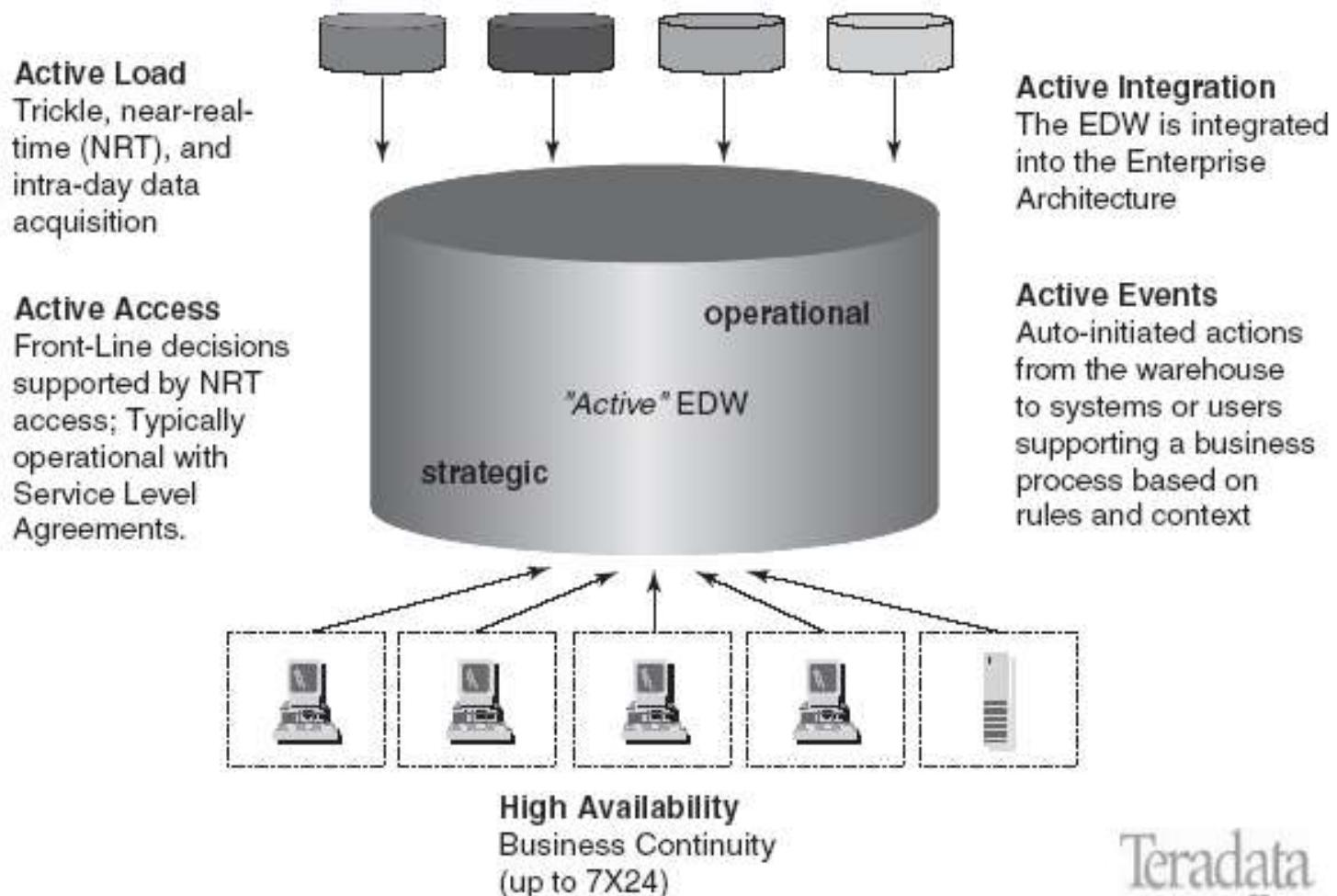
Near real-time ETL for
Active/RT Data Warehouse

Data marts are NOT separate databases,
but logical *views* of the data warehouse
→ Easier to create new data marts

Real-Time Data Warehousing



"Active" is Enterprise Data Warehousing plus any of these active elements:

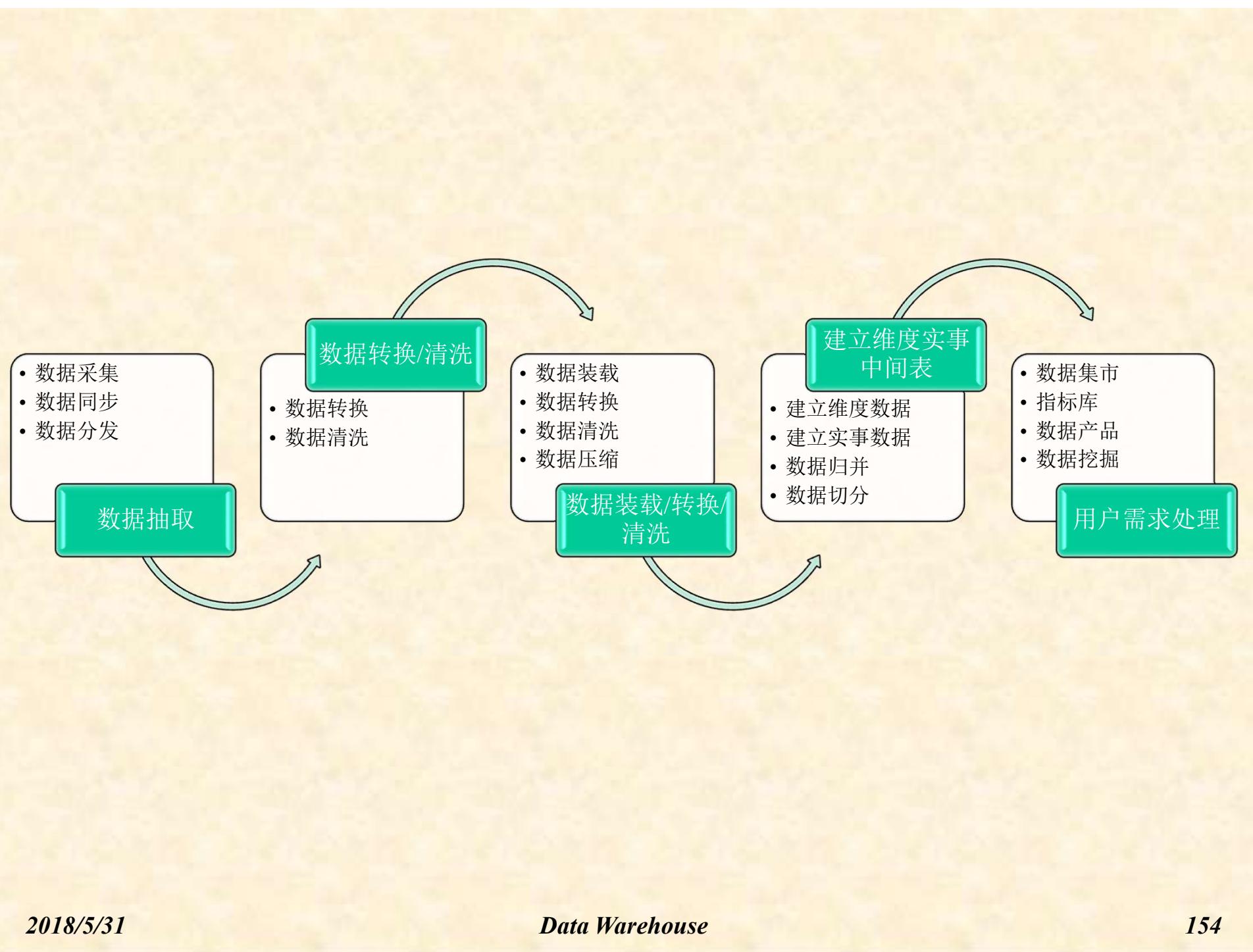


Teradata
a division of HCR

FIGURE 2.11 The Teradata Active EDW

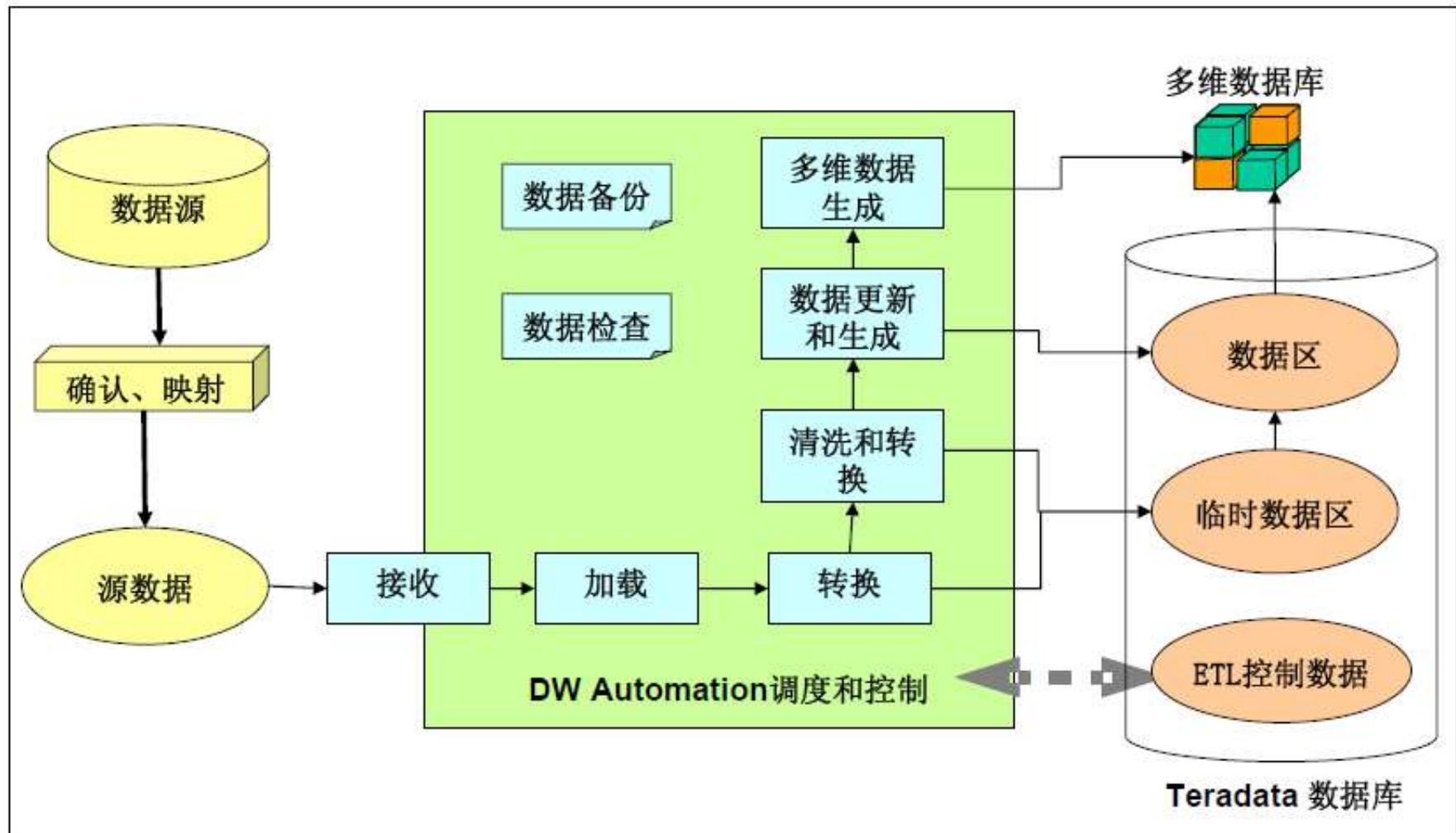
Real-Time Data Warehousing

- **The need for real-time data**
 - A business often cannot afford to wait a whole day for its operational data to load into the data warehouse for analysis
 - Provides **incremental** real-time data showing every state change and almost analogous patterns over time
 - Maintaining metadata in sync is possible
 - Less costly to develop, maintain, and secure one huge data warehouse so that data are centralized for BI tools
 - An EAI (Enterprise Application Integration) with real-time data collection can reduce or eliminate the nightly batch processes



7 ETL

❖ ETL典型流程



❖ ETL应用过程

➤ 数据抽取

抽取主要是针对各个业务系统及不同网点的分散数据，充分理解数据定义后，规划需要的数据源及数据定义，制定可操作的数据源，制定增量抽取的定义。（数据源和文件等多种形式）

Static vs. Incremental

➤ 数据传输

数据传输是通过网络负责把远程的数据到本地目录下。

7 ETL

❖ ETL应用过程

➤ 数据的清洗和转换

1. 清洗主要是针对系统的各个环节可能出现的数据二义性、重复、不完整、违反业务 规则等问题，允许通过试抽取，将有问题的纪录先剔除出来，根据实际情况调整相应的清洗操作。

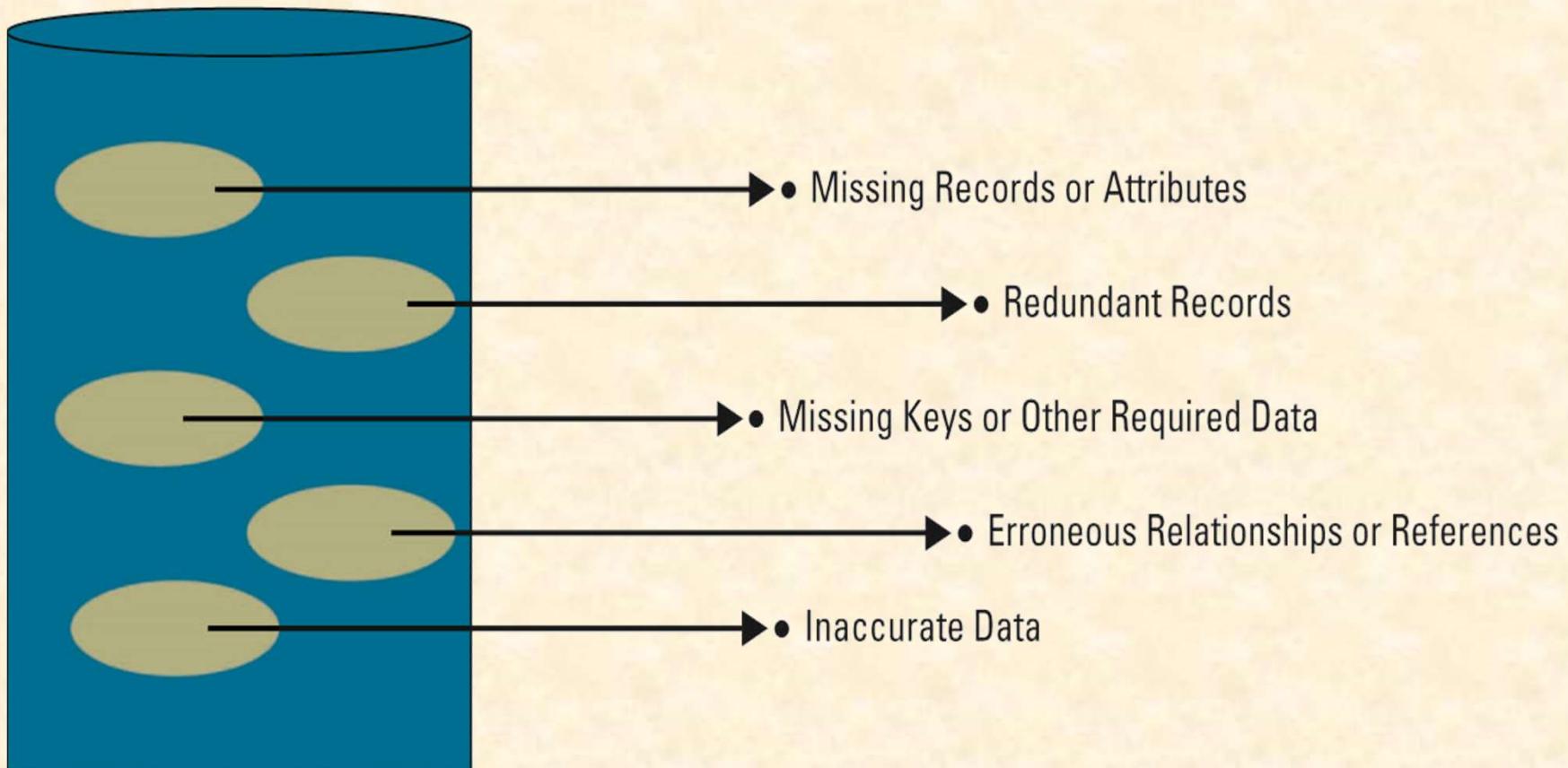
Fixing errors:

misspellings,
erroneous dates,
incorrect field usage,
mismatched addresses,
missing data,
duplicate data,
inconsistencies

Also:

decoding,
reformatting,
time stamping,
conversion,
key generation,
merging,
error detection/logging,
locating missing data

Cleansing



7 ETL

❖ ETL应用过程

➤ 数据的清洗和转换

2. 转换主要是针对数据仓库建立的模型，通过一系列的转换来实现将数据从业务模型 到分析模型，通过内建的库函数、自定义脚本或其他的扩展方式，实现了各种复杂的转换，并且支持调试环境，清楚的监控数据转换的状态。数据转换是真正将源数据变为目标数据的关键环节，它包括数据格式转换，数据类型转换、数据汇总计算、数据拼接等等。

Record-level:

Selection – data partitioning

Joining – data combining

Aggregation – data summarization

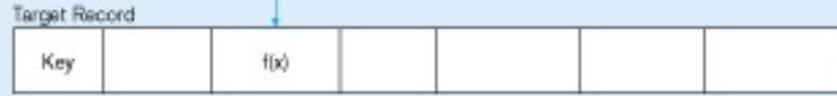
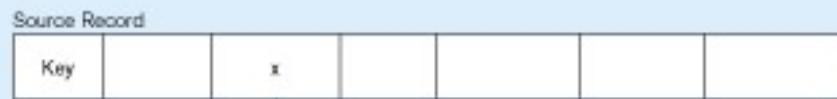
Normalization

Field-level:

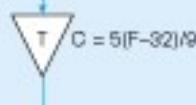
single-field – from one field to one field

multi-field – from many fields to one, or one field to many

Single-field transformation



In general – some transformation function translates data from old form to new form



Algorithmic transformation uses a formula or logical expression

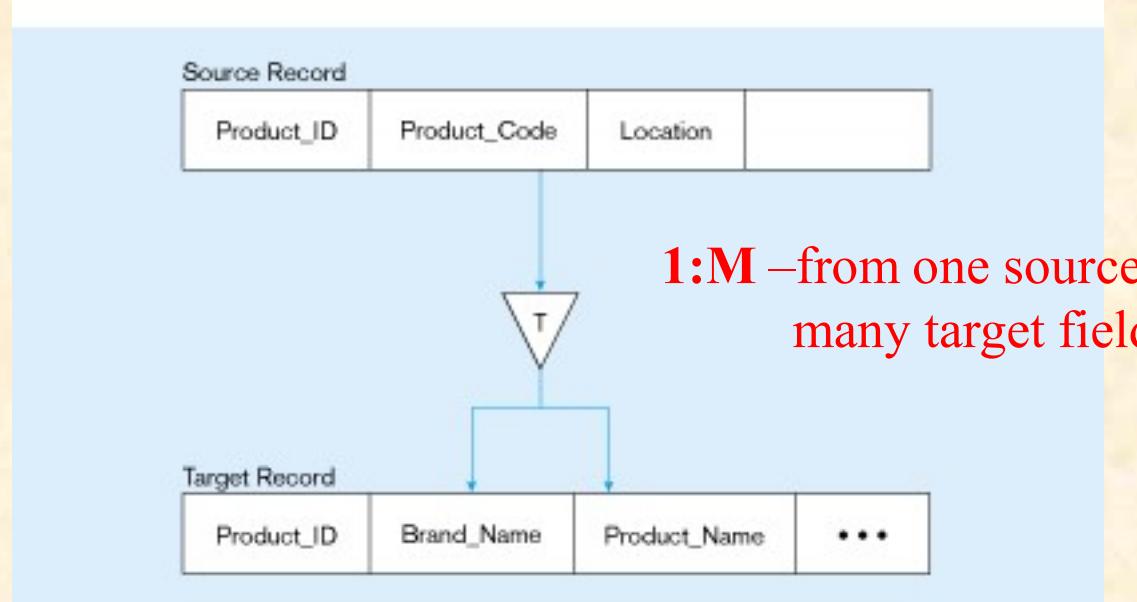
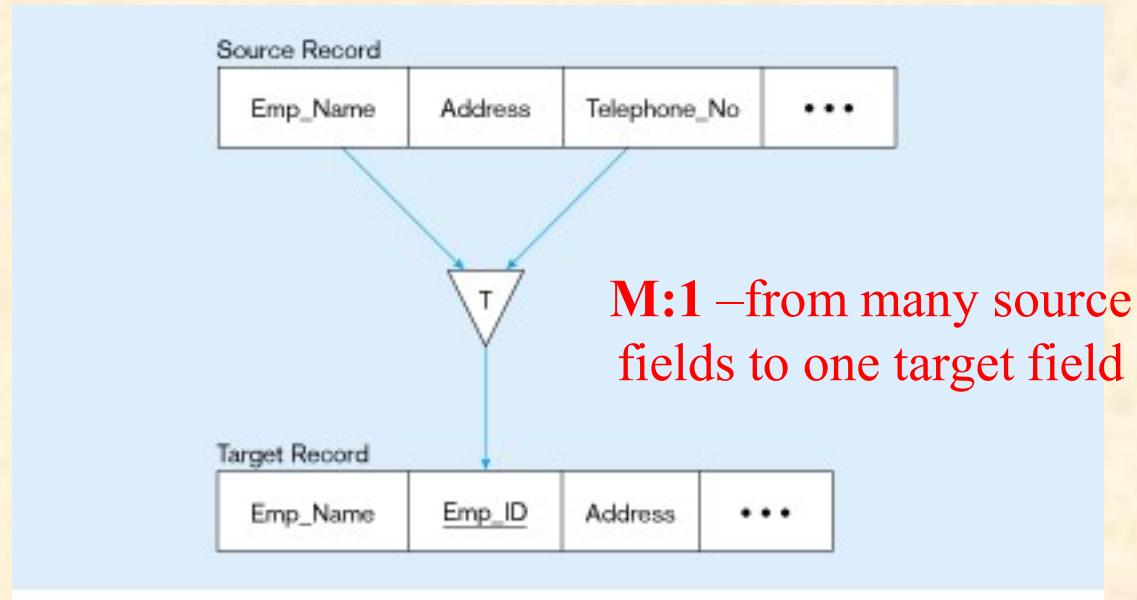


Code	Name
AL	Alabama
AK	Alaska
AZ	Arizona
...	

Table lookup – another approach



Multifield transformation



❖ ETL应用过程

➤ 数据加载入库

数据加载主要是将经过转换和清洗的数据加载到数据仓库里面，即入库，可以通过数据文件直接装载或直连数据库的方式来进行数据装载，可以充分体现高效性

Refresh mode:

bulk rewriting of target data at
periodic intervals

Update mode:

only changes in source data are
written to data warehouse

❖ ETL应用过程

➤ ETL调度

ETL的调度控制方式有两种：

1. 自动方式

由系统每天定时或准实时启动后台程序，自动完成数据仓库ETL处理流程。

2. 手动方式

用户可以通过前台监控平台，对单个目标或批量目标进行手工调度。

❖ ETL应用过程

➤ 监控

主要是监控ETL的整个过程，通过扫描ETL各模块的日志中的关键值，如记录时间等信息与当前的状态作比较，如果超过某一个值，则认为该模块运行可能出现问题，应告警。

7 ETL

❖ ETL在技术上涉及到：增量、全量、定时、调度、监控等方面技术

- ❖ 增量数据：流水类数据、话单类数据的抽取方式；
- ❖ 全量数据：用户信息类数据，状态会更新发生变化的数据；
- ❖ 定时抽取：数据抽取一般在生产系统比较闲暇的时候进行，凌晨时候比较多，而且按照要分析数据的周期，还分为按日、按月数据；
- ❖ 作业调度：由于涉及到的业务系统的数据量庞大，需要分批进行抽取，以及抽取数据后面的一系列处理过程；
- ❖ 作业监控：对所有作业执行的监控