



数据仓库与大数据工程

Data Warehouse and Big Data Engineering

第4部分 数据组织与环境

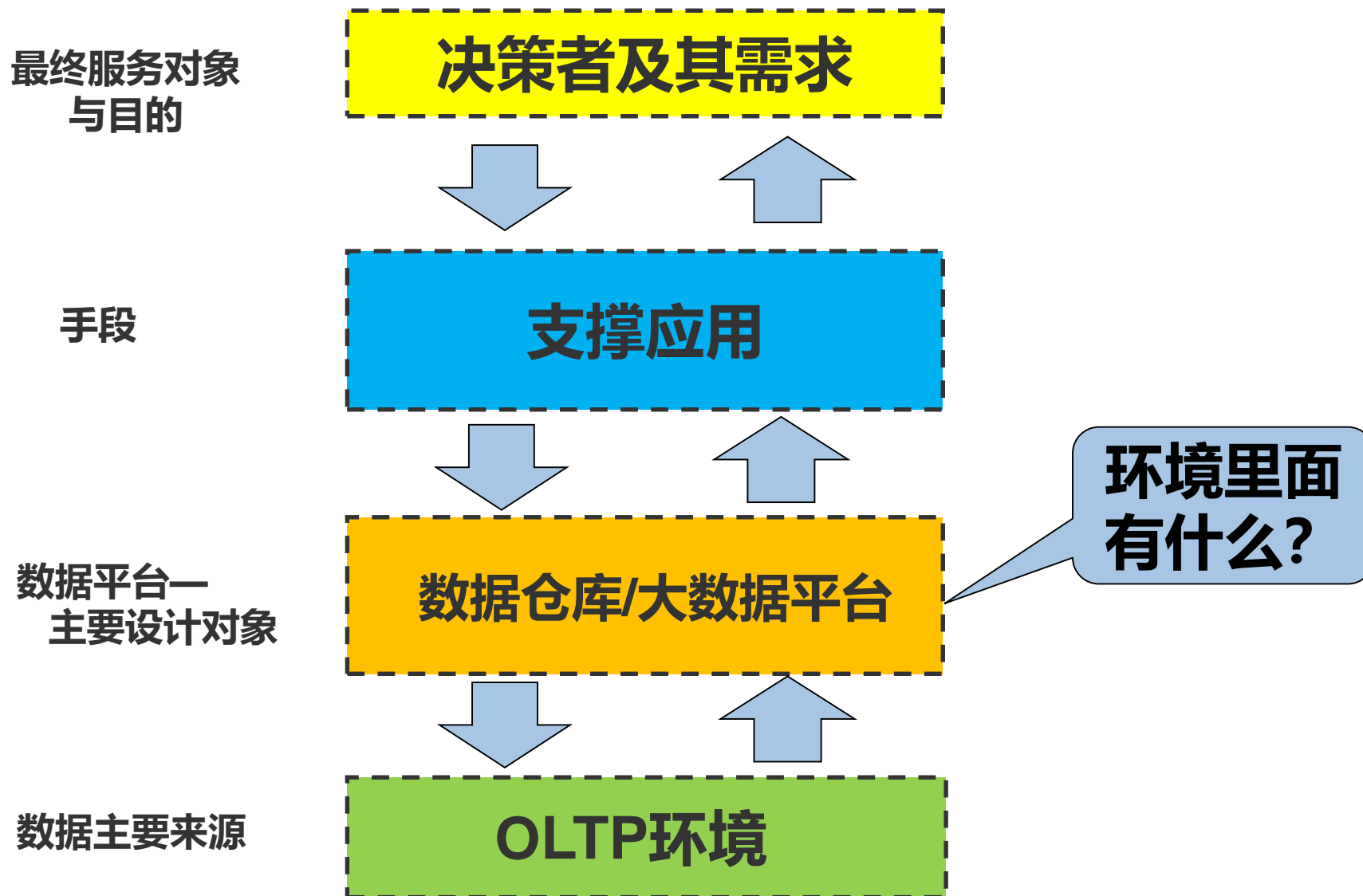
版权所有：

北京交通大学计算机与信息技术学院





为了谁？需要什么样的架构





内容提纲

数据仓库与大数据主要特征

数据仓库与大数据平台总体数据架构

数据仓库与大数据平台数据组织结构

数据环境组成与应用支撑

软硬件技术环境



1. 数据仓库及其特征

- ▶ A Data warehouse is a
 - Subject-oriented –面向主题
 - Integrated—集成
 - Nonvolatile—不可更新、非易失、永久
 - Time-variant—随时间变化
- ▶ collection of data in support of **management's** decision.
- ▶ 新实践：早已不仅仅服务于管理层的决策
- ▶ 数据仓库是一个面向主题的、集成的、非易失的、随时间变化的用于决策支持的数据集合



(1) 主题与面向主题

► Subject—主题，一个高层抽象概念

- 对企业数据进行分析应用的功能集的抽象。
- 在较高层次上将企业信息系统中的数据进行综合、归类并进行分析利用的抽象。

► 客户分析主题

- 分析客户的消费习惯
- 客户群划分
- 客户发展动向
- 通过围绕客户相关的一系列信息集合来支持



数据仓库的面向主题特性

业务系统面向应用

操作型环境



汽车



人寿



健康



意外伤亡

应用

数据仓库面向主题

数据仓库



顾客



保险单



保险费



索赔

主题



通信公司的业务系统与分析主题

► 业务系统

- 语音业务
- 数据业务
- 计费业务
- 网管业务

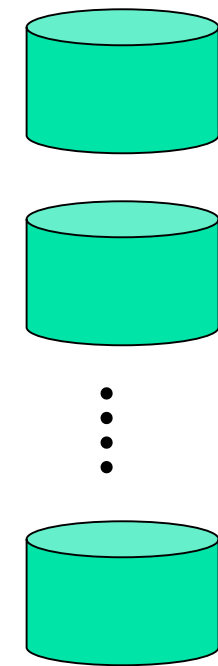
► 分析主题

- 客户分析主题
- 资源分析主题
- 营销渠道分析主题
- 资费模式分析主题

分析主题中的数据来自业务系统，但围绕核心分析问题进行重新组织。



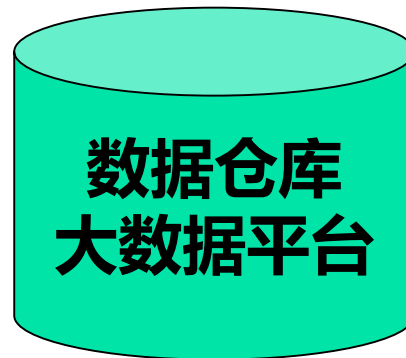
(2) 集成—最重要的特征



数据源

转换
重新格式化
重新序列化
摘要
数据一致化
...

集成要解决
的主要问题





为什么要集成

- ▶ 利用多个方面的数据，进行碰撞、对比，才更有可能得到准确、全面、有价值的信息。
- ▶ 例如：Customer Profiling
 - 生理和自然属性：性别、身高、肤色、...
 - 社会属性：地位、角色、民族、职位、...
 - 内容或行为偏好：喜欢上网、喜欢看小说、喜欢某项运动、...
 - ...
- ▶ 需要有集成的数据，才能做好。

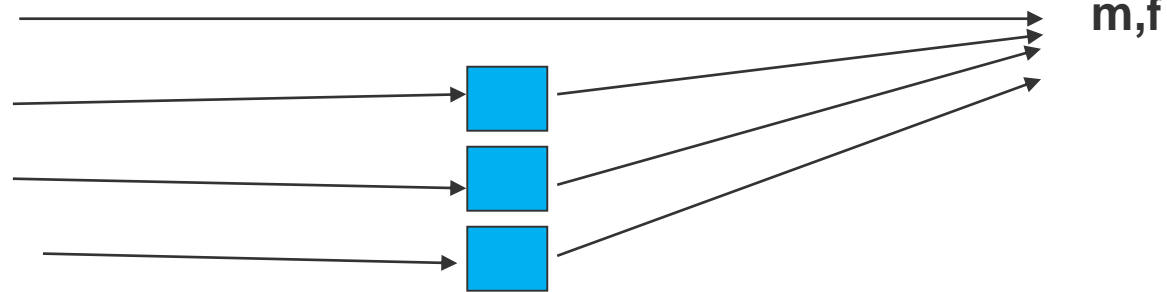


集成问题举例

集成过程的编码一致性转换

应用A m, f
应用B 1, 0
应用C x, y
应用D 男, 女

操作型环境



数据仓库

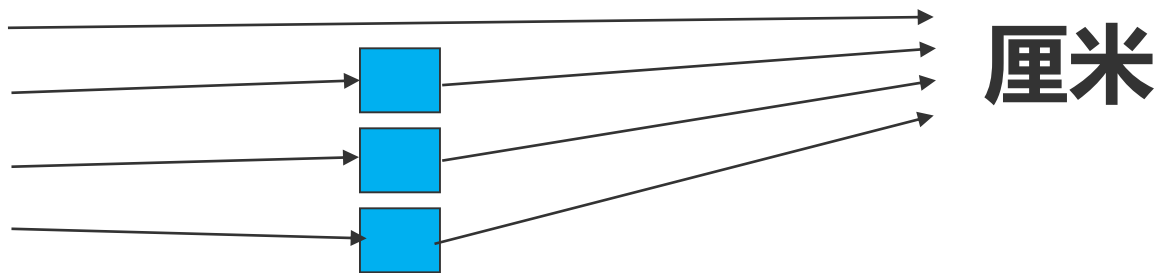
解决编码的不一致问题



集成问题举例

集成过程中的度量单位转换

应用A 管道,厘米
应用B 管道,英寸
应用C 管道,千立方英尺
应用D 管道, 码



操作型环境

数据仓库

度量单位不一致的问题



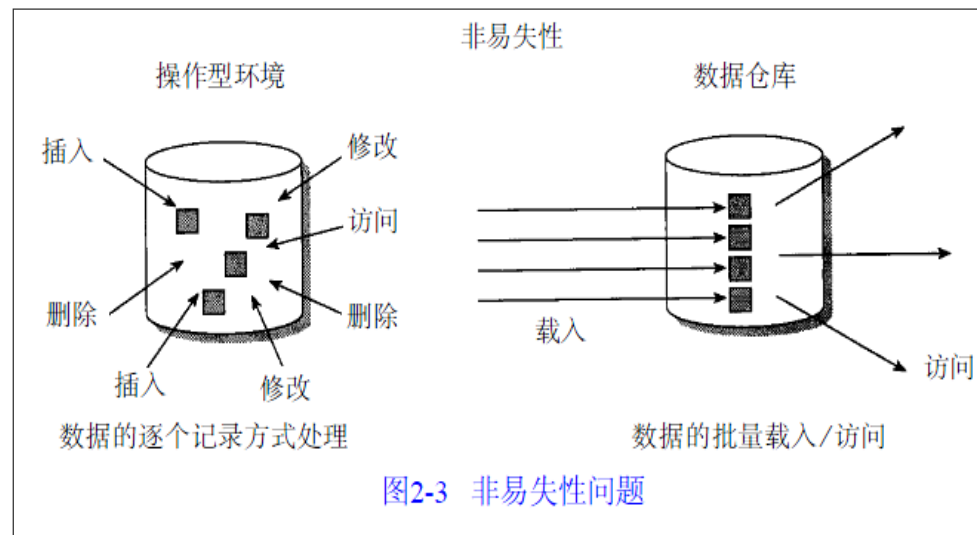
集成需要解决的其他问题

- ▶ 不一致的描述
- ▶ 不一致的关键字
- ▶ 同一属性用不同的名称
- ▶ 同一名称代表不同的属性
- ▶



(3) 非易失性

- ▶ 非易失性(Nonvolatile)
 - 不进行一般意义上的更新
 - 一般以批量方式装载
 - 构成相对永久性的历史数据集





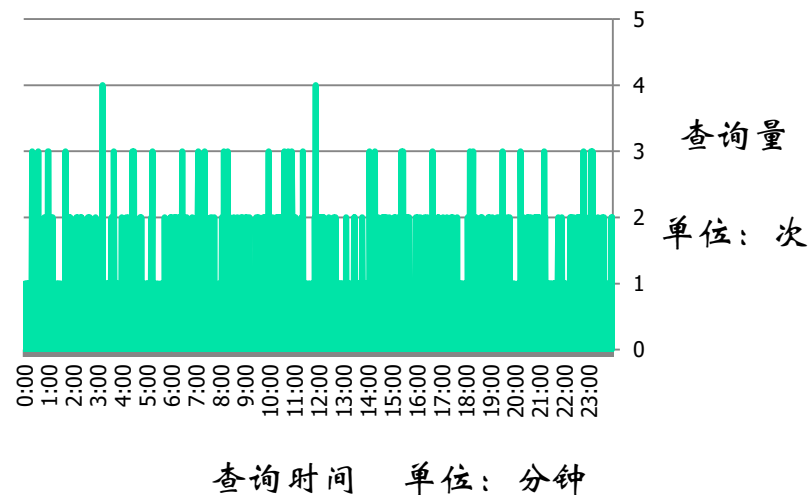
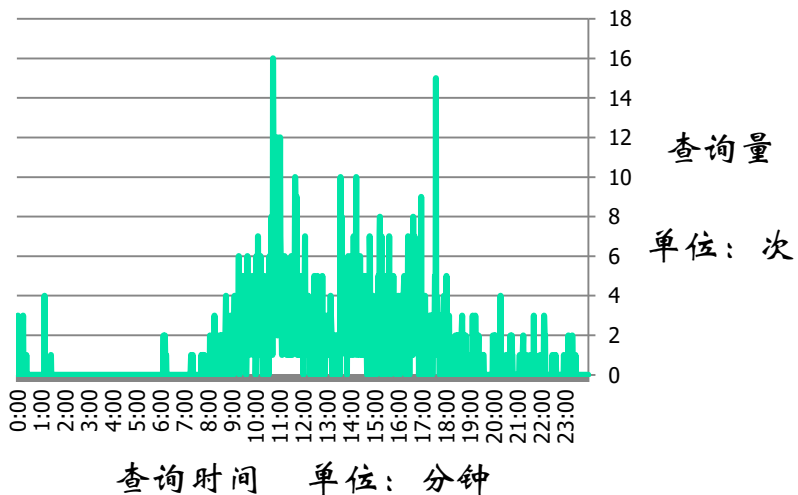
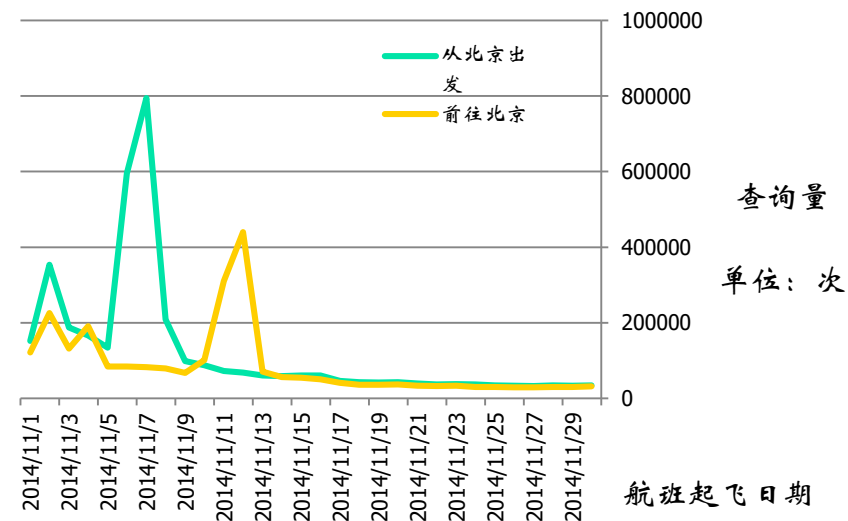
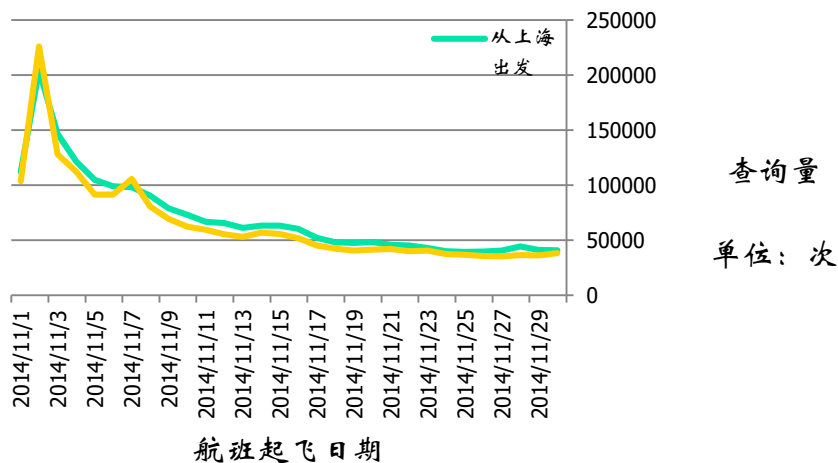
(4) 时变性(time-variant)

► 时变性

- 数据仓库中的数据一般都有时间属性，时间作为关键字结构的一部分。
- 操作型系统的数据都是当前值数据，而数据仓库中则保存着历史数据。
- 数据仓库数据与操作型系统的数据的时间跨度存在很大差异
 - 操作型系统：一般60-90天
 - 数据仓库：5-10年



时间序列数据案例





2. 大数据的主要特征

- ▶ 数据量巨大
- ▶ 数据类型多
- ▶ 数据的产生速度快
- ▶ 可度量性
- ▶ 真实性
- ▶ 复杂性



内容提纲

数据仓库与大数据主要特征

数据仓库与大数据平台总体数据架构

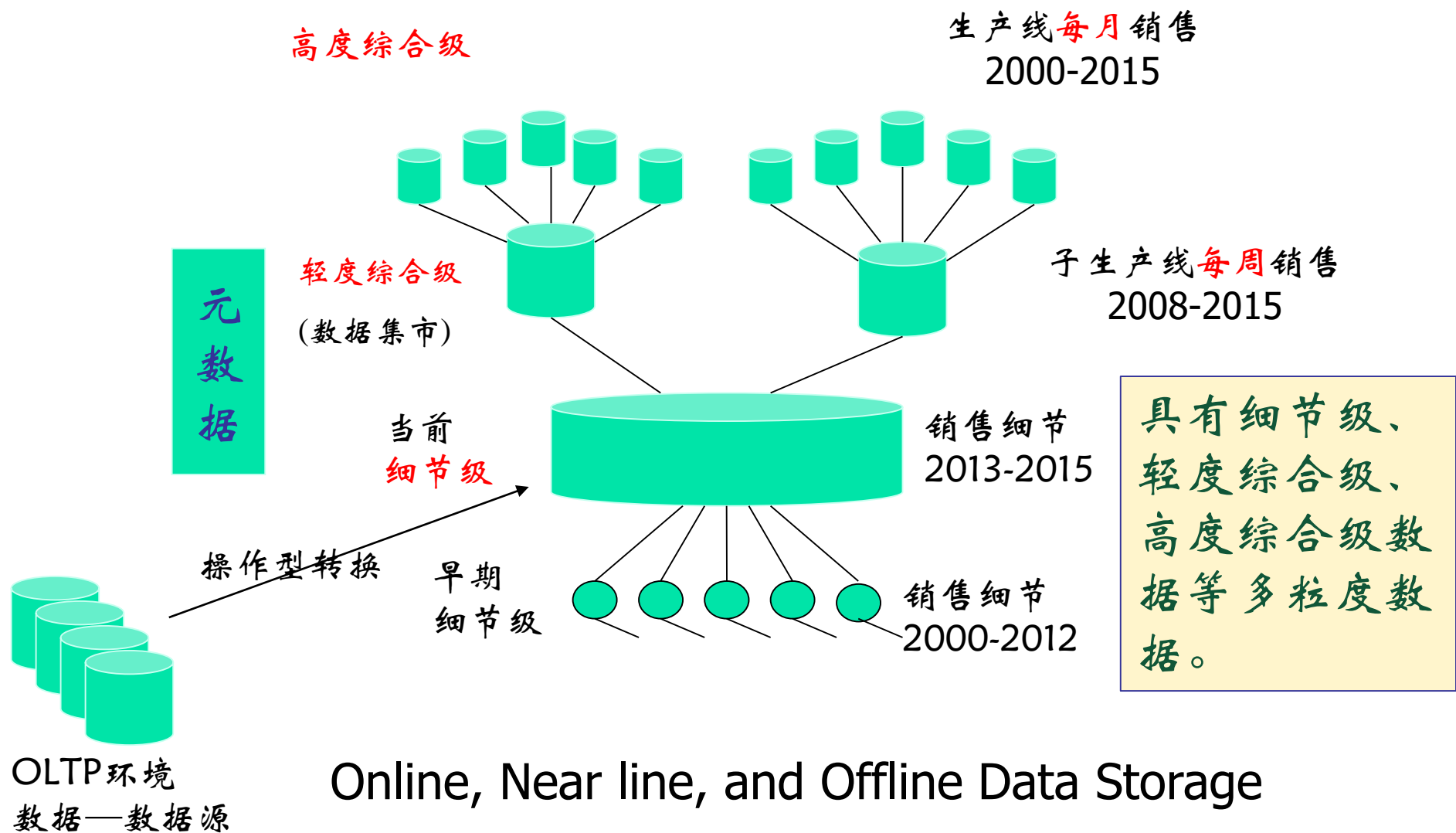
数据仓库与大数据平台数据组织结构

数据环境组成与应用支撑

软硬件技术环境

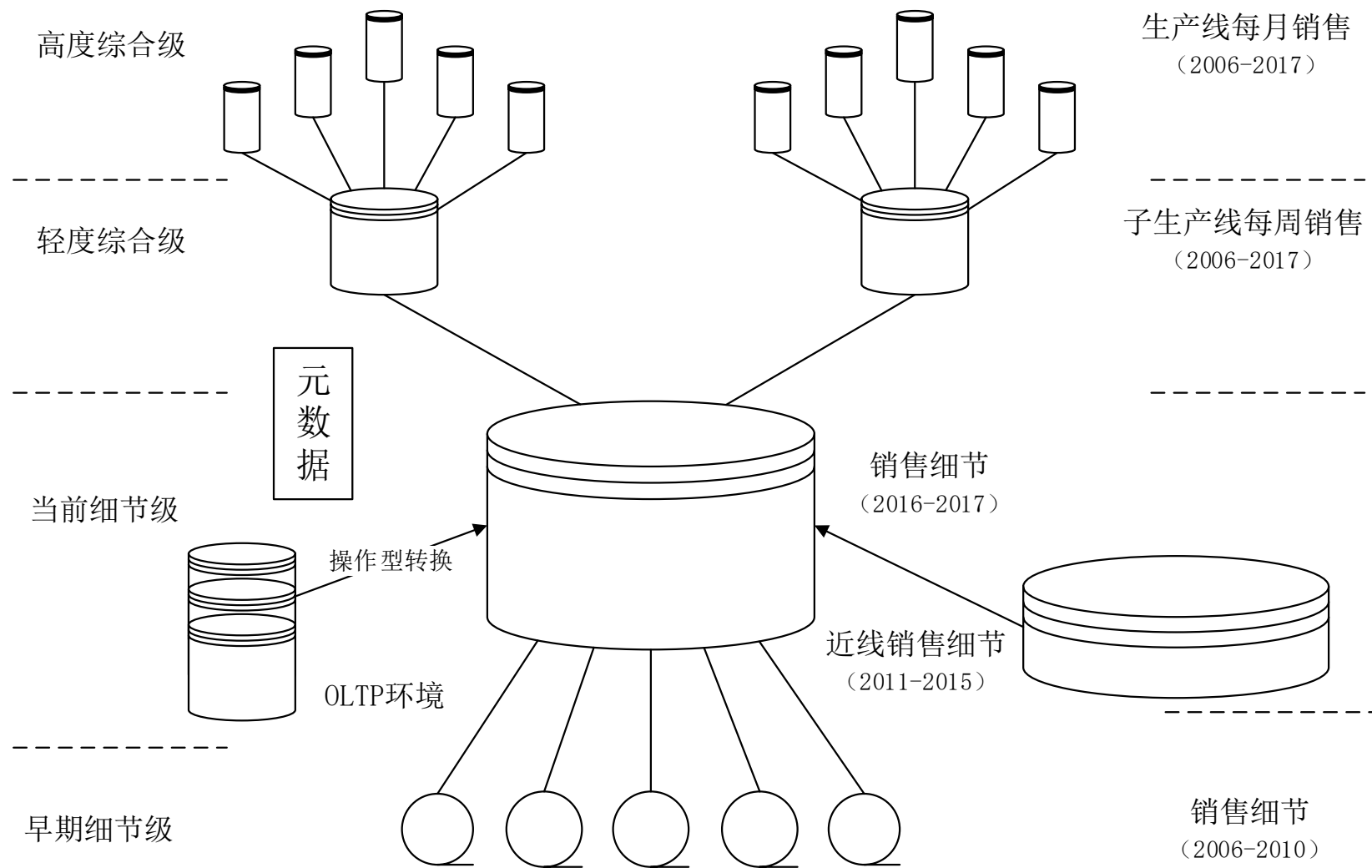


1. 数据组成总体层次结构





相似数据层次架构图





2. 细节数据

- ▶ 细节数据是平台数据中规模最大的数据，来自OLTP环境
- ▶ 细节数据常常又分成
 - 当前细节级
 - 早期细节数据
- ▶ 数据的**产生时间**与**应用情况**在通常情况下是紧密相关的，越新鲜的数据被访问的可能性更高，越陈旧的数据访问频率越低。
- ▶ **不同热度**的细节数据可能会放入**不同的存储层级**中

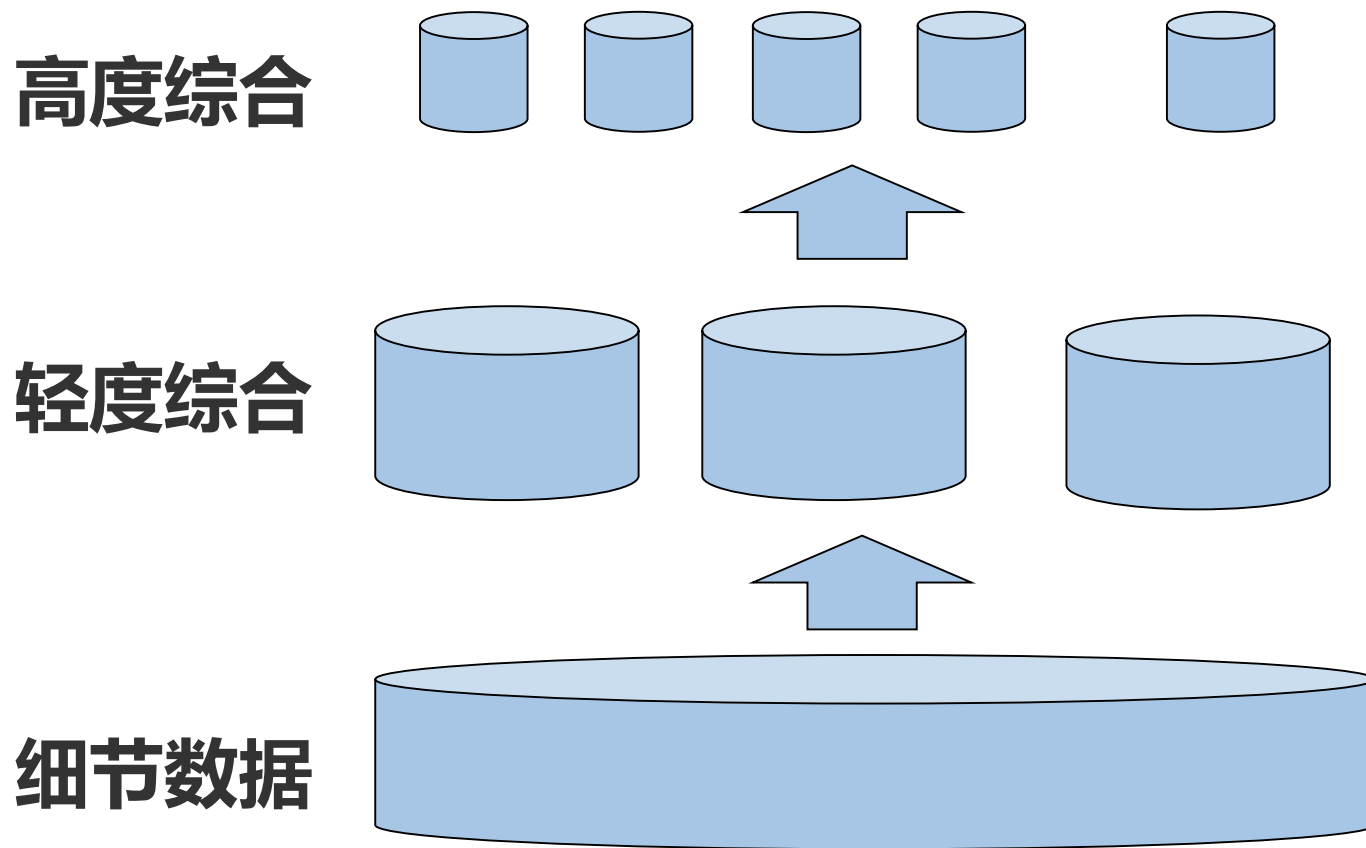


3. 环境中的多级存储体系

- ▶ 根据向用户提供服务的模式，细节一般可以分成三个层级
 - Online—热数据，提供在线服务，使用率高
 - Near line—近线数据，温数据，使用率不高
 - Offline—离线数据，归档数据，冷数据，使用率不高
- ▶ 三个层级的划分标准
 - 时间标准
 - 使用热度标准
 - 被使用与否
- ▶ 注意：需要建立三层数据体系间的**流动解决方案**



4. 多层级综合或汇总数据



思考：为什么要有不同层级的综合数据？

粒度与多粒度级数据



5. 元数据

- ▶ **元数据(Metadata)是数据架构中关于数据的数据，即描述一个数据环境中数据的组成、结构、定义、关系、处理流程等的数据**
- ▶ **Metadata is "data / information that provides information about other data"**
- ▶ **元数据是数据管理与图书情报领域的一个非常重要的概念。**
- ▶ **在传统的OLTP系统和数据仓库环境中都应该具有元数据。**
- ▶ **元数据本身也是数据仓库数据架构中重要组成部分。**



元数据的分类

- ▶ **Descriptive metadata** describes a resource for purposes such as **discovery and identification**. It can include elements such as title, abstract, author, and keywords.
- ▶ **Structural metadata** is metadata about containers of data and indicates how compound objects are put together, for example, how pages are ordered to form chapters. It describes the types, versions, relationships and other characteristics of digital materials.
- ▶ **Administrative metadata** provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it.
- ▶ 元数据另一种分类角度：业务元数据和技术元数据



元数据相关概念

- ▶ 数据库中的**数据字典**就是元数据的一种。
- ▶ 元数据对于数据仓库就好比数据字典对于数据库一样重要
- ▶ 用于对各种数据进行描述，说明它们之间的关系，是数据仓库的应用灵魂，是不可或缺的组成部分。
- ▶ 元数据模型
 - 用于表达元数据的数据模型
- ▶ 元元模型
 - 用于表达元数据的数据结构的模型
 - 元数据管理工具层的数据模型



内容提纲

数据仓库与大数据主要特征

数据仓库与大数据平台总体数据架构

数据仓库与大数据平台数据组织结构

数据环境组成与应用支撑

软硬件技术环境



数据组织结构

- ▶ 面向主题的数据组织方法
- ▶ 粒度与多粒度级数据组织
- ▶ 数据分区
- ▶ 活样本数据集组织方法
- ▶ 三种常见细节数据与汇总数据组织形式
- ▶ 整个架构中的操作型数据窗口
- ▶ 数据清除问题



1. 面向主题的数据组织方法

- ▶ 面向主题的概念
- ▶ 面向主题与面向业务应用组织数据的区别
- ▶ 主题划分方法
- ▶ 主题区域的概念
- ▶ 主题区域的重叠
- ▶ 主题区域数据集特性
- ▶ 主题划分案例



(1) 什么是面向主题？

► 什么是面向主题？

- 数据仓库应该针对企业的主要分析主题进行构造。
- 主题：数据分析的关注范围；分析需求集；数据利用需求集；

► 面向主题是分析型应用的需求特点。



(2) 面向主题与面向业务应用组织数据的区别

► 面向业务应用的数据组织方法

- 紧密围绕具体**业务场景的业务流程**相关数据需求，去设计数据模型，组织业务应用，记录相应业务数据
- 具有大量与具体业务过程与环节相关的细节数据

► 面向决策支持主题的数据组织方法

- 面向决策支持主题的需求，去企业信息系统环境中找到相应数据，进行**集成与合理的数据组织**，加工出决策支持需要的信息或知识
- 从现有数据出发，服务决策支持需求主题，组织数据，加工信息，总结规律



分析型应用与业务型应用的不同

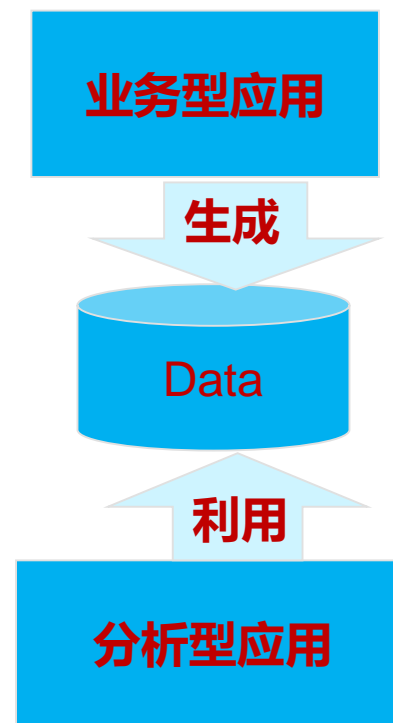
► 业务型应用(操作型应用)

- 如何快捷地实现业务，**生成并记录数据**
- 基础数据的制造者。

► 分析型（决策支持型）应用

- 根据事实数据，对企业运营情况进行分析，用于决策支持；
- 利用数据，在数据中寻找信息；
- 利用信息服务于业务应用
- 数据的消费者。

数据生成与消费关系





分析型应用举例

► 商品采购

- 业务应用系统：怎样更方便、更快地实现商品采购业务
- 分析处理系统：同一商品选择哪个采购渠道？如何给某个产品选择合适营销客户？

► 电信公司

- 业务应用系统：如何保证为客户服务的业务能正常开展？如何确保信号质量和稳定性？
- 分析系统：客户转网分析；客户分析；产品推荐；



(3) 主题划分方法

1. 收集决策支持需求
 2. 将决策支持分析需求进行归类
 3. 用一个主题名称对将**归类出的不同需求**集进行命名
 4. 根据主题内的每项分析需求，组织主题所需要的数据范围，确定出**主题区域**。
- 例如，收集需求后，发现有许多需求与供应商有关，则把这些需求归成一类，称为“供应商分析”主题。



主题划分案例1

► 电信公司数据仓库主题划分

► 北京移动组织架构

- 综合部、战略与法律事务部、计划建设部、财务部、HR、**市场经营部、品质保障部、集团客户部、数据业务部、网络部、信息系统部、采购部、审计部、客户服务中心、网络优化中心、网络运行支撑中心、工程建设中心、传输中心、培训中心、行政中心、党群工作部、纪检监察、工会**



电信公司的客户主题

- ▶ 客户主题主要完成经营分析中对客户的**各种属性**，客户的**消费行为**及客户的**发展动向**等进行分析。
- ▶ 主题细分
 - 个人客户
 - 大客户
 - 集团客户



客户主题分析与处理

- ▶ 个人客户数据处理
- ▶ 大客户异动分析
- ▶ 个人大客户分析
- ▶ 大客户新业务分析
- ▶ 集团客户分析



电信公司的用户主题

- ▶ **用户资料是整个经营分析系统的核心部分。是其它分析主题的基础。**
- ▶ **个人用户的基本资料信息数据刷新以及资料中关键维度的分析**
- ▶ **个人用户的业务变更数据的获取及分析数据的生成**
- ▶ **个人用户的积分数据的获取及积分的聚合数据获取**



用户主题处理与分析功能

- ▶ 用户基本信息处理
- ▶ 用户转网分析
- ▶ 用户扩展信息处理
- ▶ 用户积分信息处理
- ▶ 用户品牌套餐变更



电信公司的资源主题

- ▶ 从时间、地域、和状态的角度对业务资源包括号码和卡资源的使用情况进行分析，以提高资源的利用率。
- ▶ 号码资源
 - 对号码资源的使用情况分析，统计各个品牌的使用率、吉祥度等
- ▶ 卡资源
 - 对卡资源进行分析处理



其它主题

- ▶ 服务使用主题
- ▶ 账务收益主题
- ▶ 客户服务主题
- ▶ 结算主题
- ▶ ...



(5) 主题区域及其特性

- ▶ **主题域或主题区域—Subject Area**
 - ▶ **解释1：主题所涉及的决策支持需求领域或范围**
 - ▶ **解释2：与该主题决策支持需求相关的数据构成的数据区域或数据范围—主题数据区域**
- ▶ **显然，范围明确的决策支持需求需要相应的一定范围内的数据集集合来支撑**



主题区域应该具有的总体特性

► 主题区域应该具有：

- **独立性**

- 具有独立的内涵，明确的界限，可以有交叉。

- **完备性**

- 对任何一个该领域的决策支持需求，都应能在该领域中找到所需的数据。
- 有一个逐步完善的过程。



主题与关系型数据表集

- ▶ **每一个主题是通过一系列的相应的数据表物理地实现的。**
- ▶ **数据表集在逻辑上所表示的、所能支持的分析范围即称为主题域。**
- ▶ **在关系型的物理数据平台上**
 - **一个主题域 \leftrightarrow 一套物理数据表**



传统数据仓库中的主题数据集

► 在企业级数据仓库中

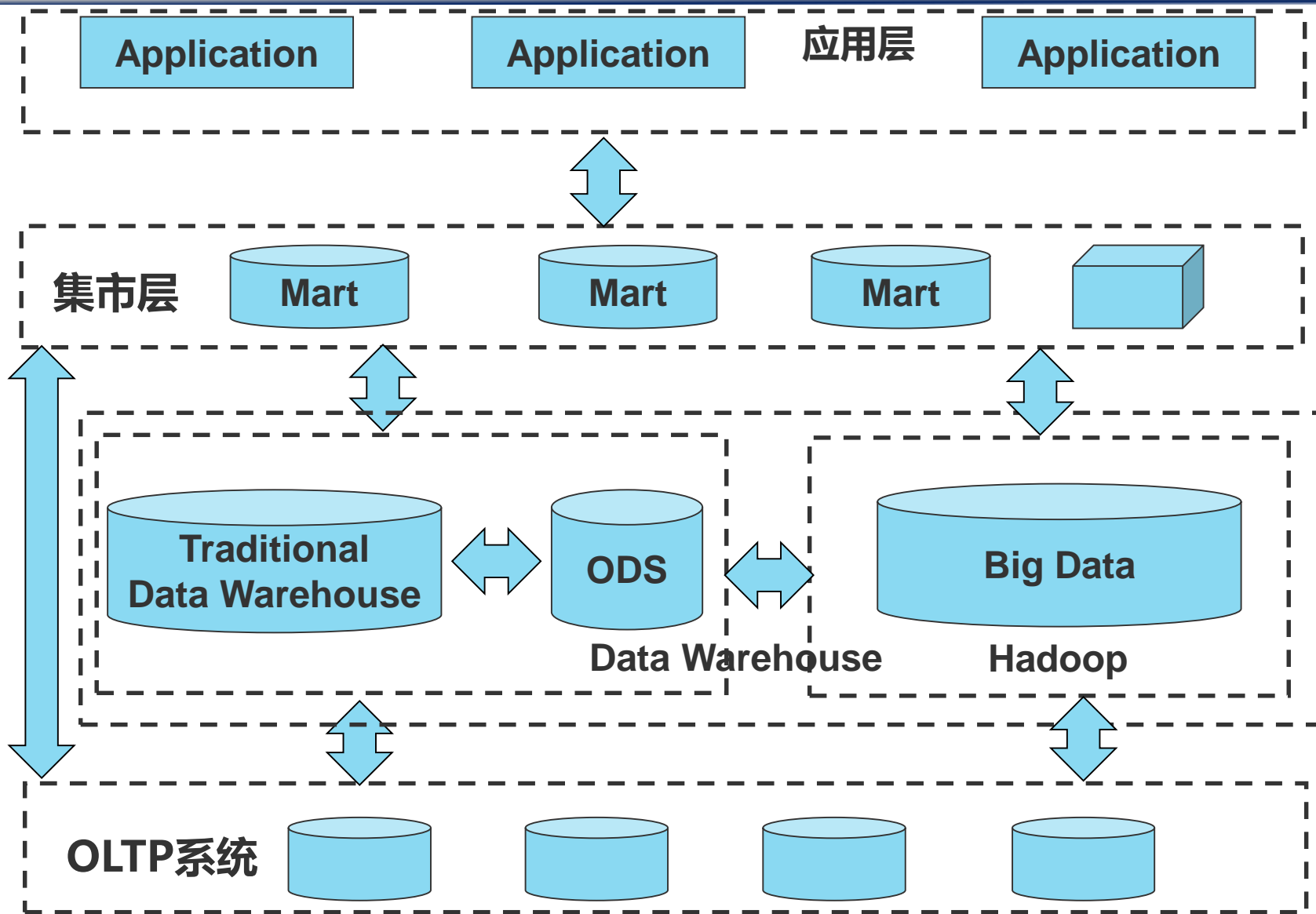
- 用一组界定的物理表表示相应主题

► 在数据集市层的OLAP工具中

- 常见用数据立方体(Data Cube)来表示针对主题的更小范围的分析.
- 数据立方体的数据基础，也是一组数据表集



数据仓库+大数据架构：混合架构





在混合型架构中主题数据集

- ▶ 混合型架构中主题区域组成
 - 大数据平台中数据集
 - 关系型平台中的表集
- ▶ 两类架构中的数据集之间存在**语义关系与计算逻辑**



案例2—电子商务相关的数据分析主题

- ▶ 商品分析主题
- ▶ 供应商分析主题
- ▶ 客户分析主题
- ▶ 店铺分析主题



主题相关数据(主题域)举例

► 商品分析主题

- 商品固有数据
- 商品类别数据
- 商品采购数据
- 商品销售数据
- 商品库存数据
- 商品被搜索记录
- 商品被点击记录
- ...



主题相关数据(主题域)举例

► 供应商分析主题

- 供应商固有数据
- 供应商商品数据
- 供应商退货数据
- ...



主题相关数据(主题域)举例

► 商铺

- 商铺基本信息
- 商铺销售数据
- 商铺采购数据
- 商铺被投诉记录
- 商铺供应商退货数据
- 商铺客户退货数据
- ...



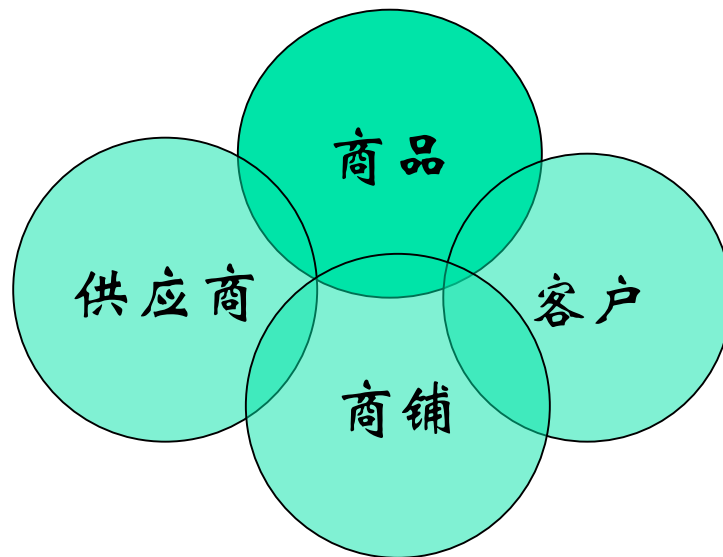
主题相关数据(主题域)举例

► 客户分析主题

- 客户固有数据
- 客户购物记录
- 客户搜索记录
- 客户点击记录
- 客户投诉记录
- 客户退货记录
- ...



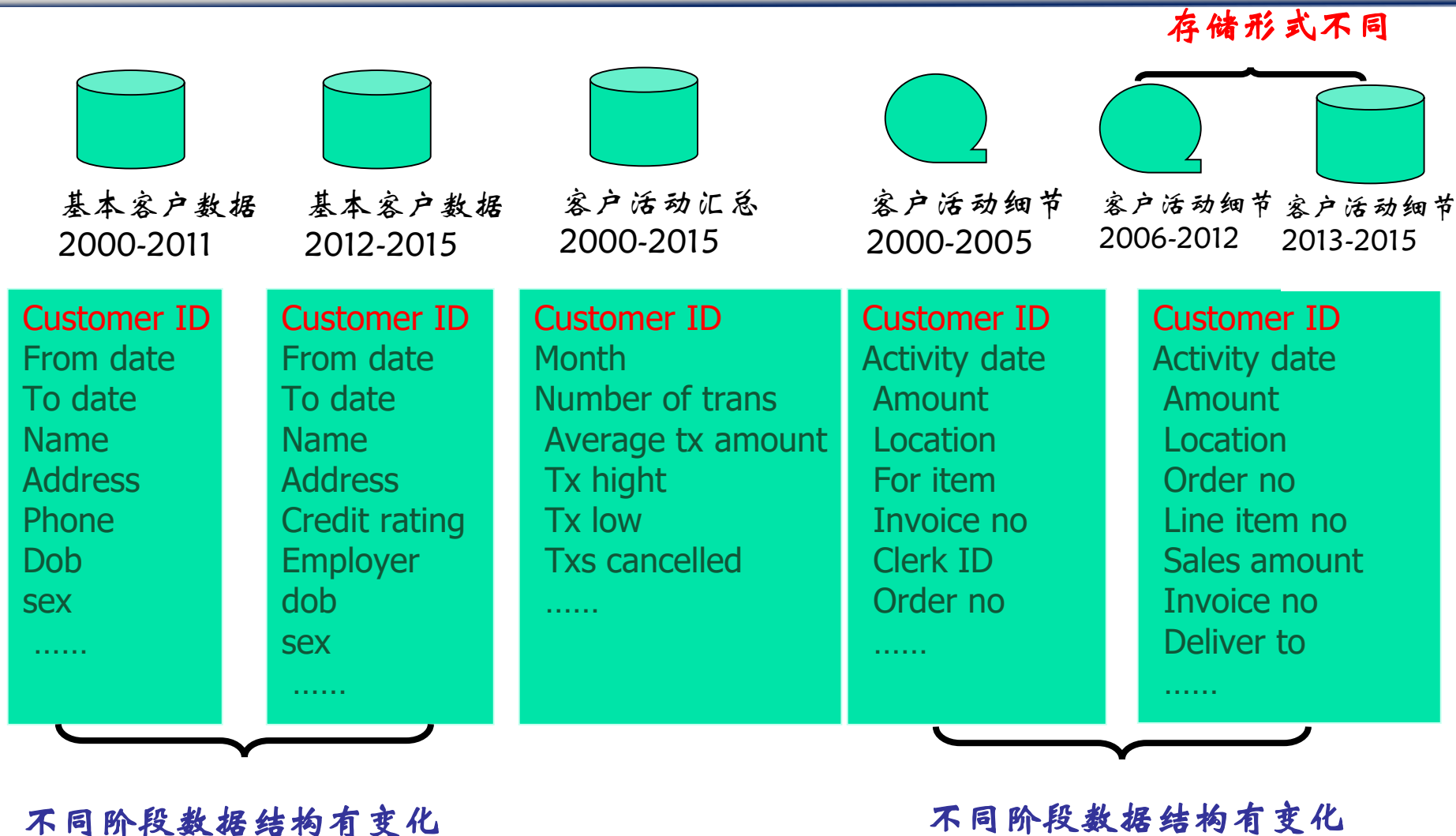
主题区域之间存在数据重叠特点



- ▶ 表现了两个主题之间的联系
 - 商品销售信息与客户购物信息
- ▶ 仅仅是逻辑上的重叠，而不应是物理上的重叠
- ▶ 仅仅是在细节级上的重叠
- ▶ 并不是两两重叠



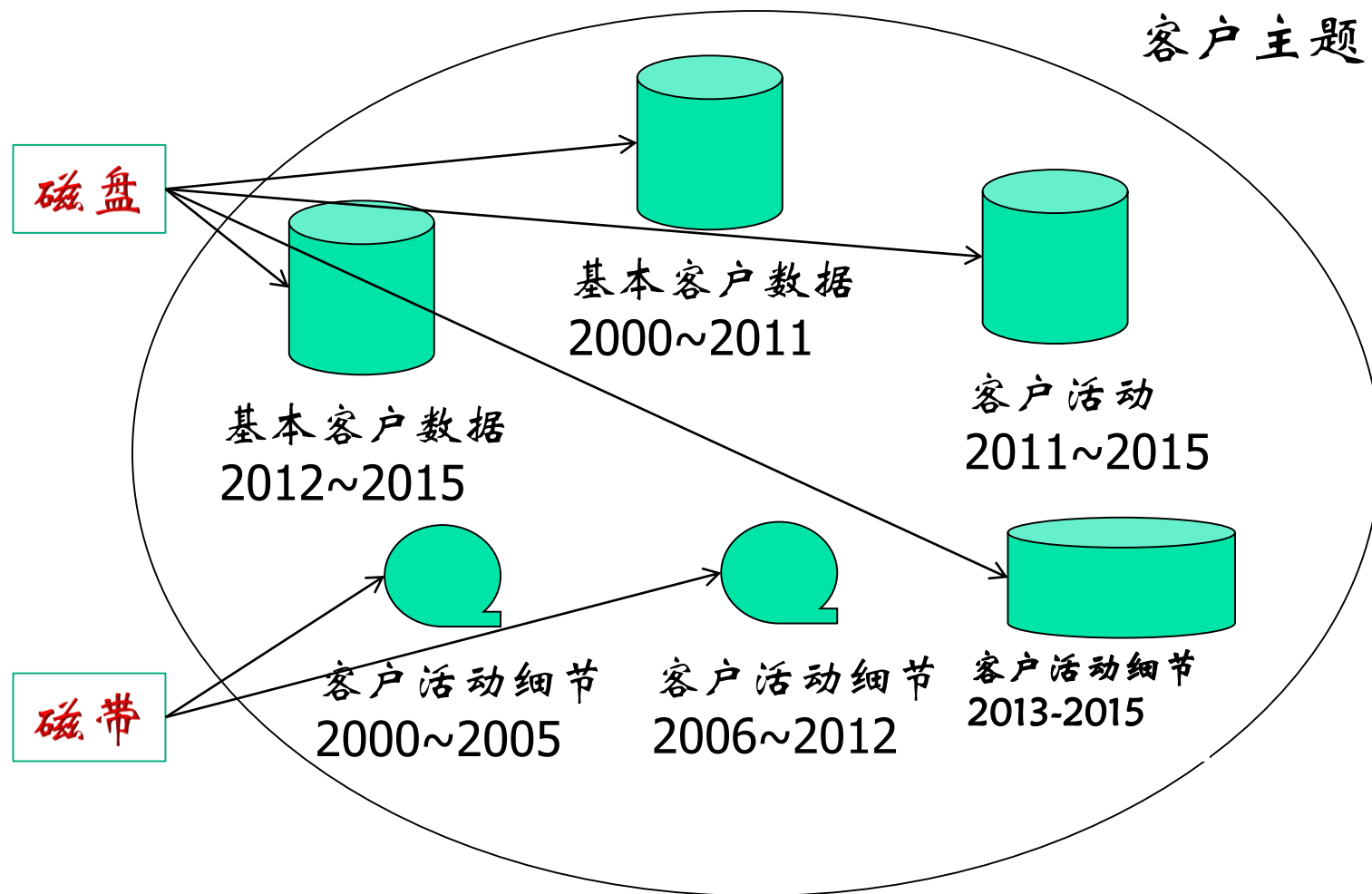
主题域数据结构及存储形态特点



数据间存在计算关系，粗细不同



存储介质的不同(磁盘和磁带)



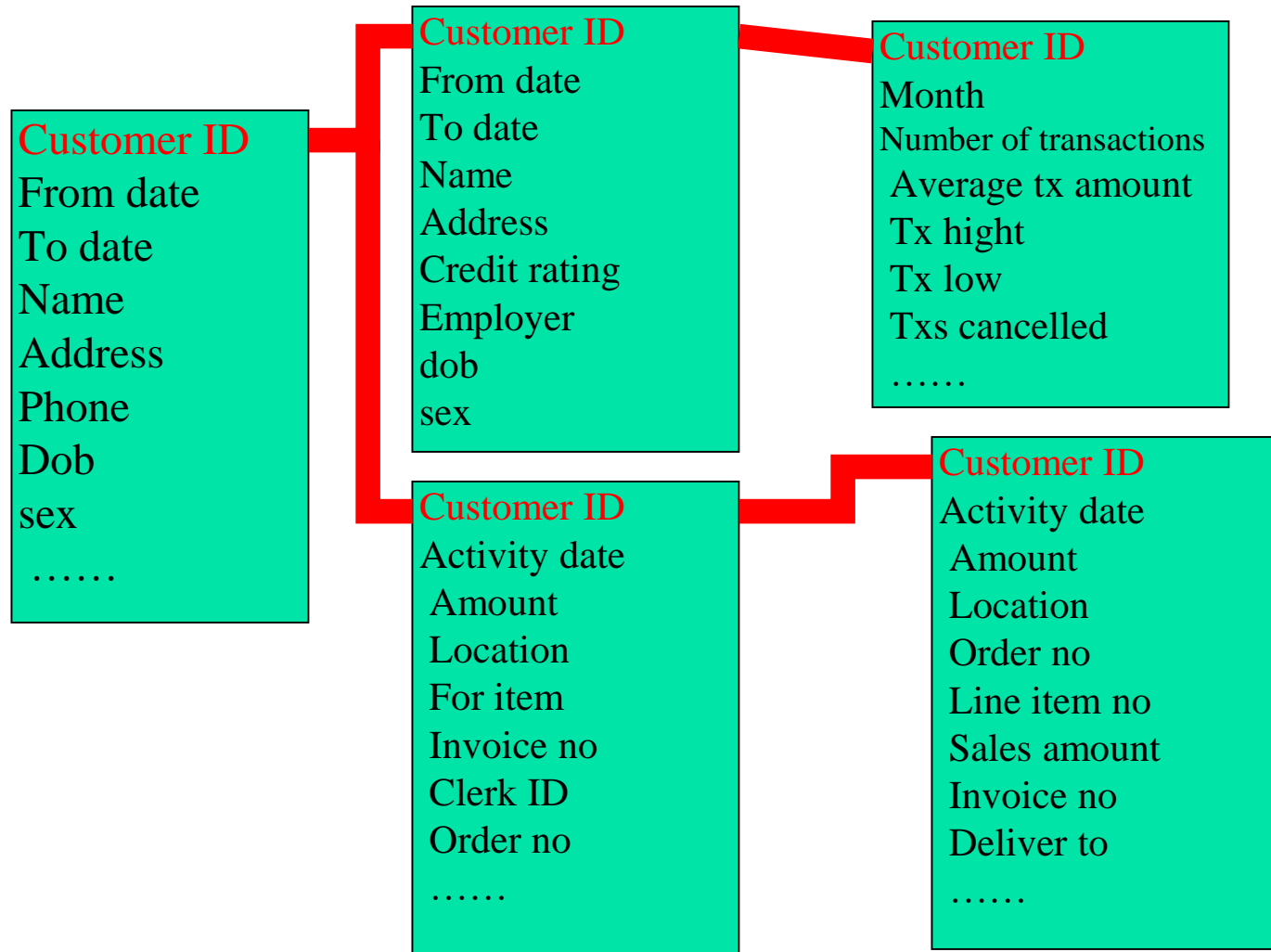


存储介质的不同意味着...

- ▶ 可能具有**多个DBMS**对不同数据分别进行管理
- ▶ 某些数据没有被DBMS所管理
 - 但也算是数据仓库和大数据平台的一部分
- ▶ 存放原则—或数据迁移的原则
 - **访问频繁**且占用存储空间小的数据存放在**快速且相对昂贵**的存储介质上。
 - **访问较少**且占用存储空间大的数据存放在**廉价、慢速**的介质上。

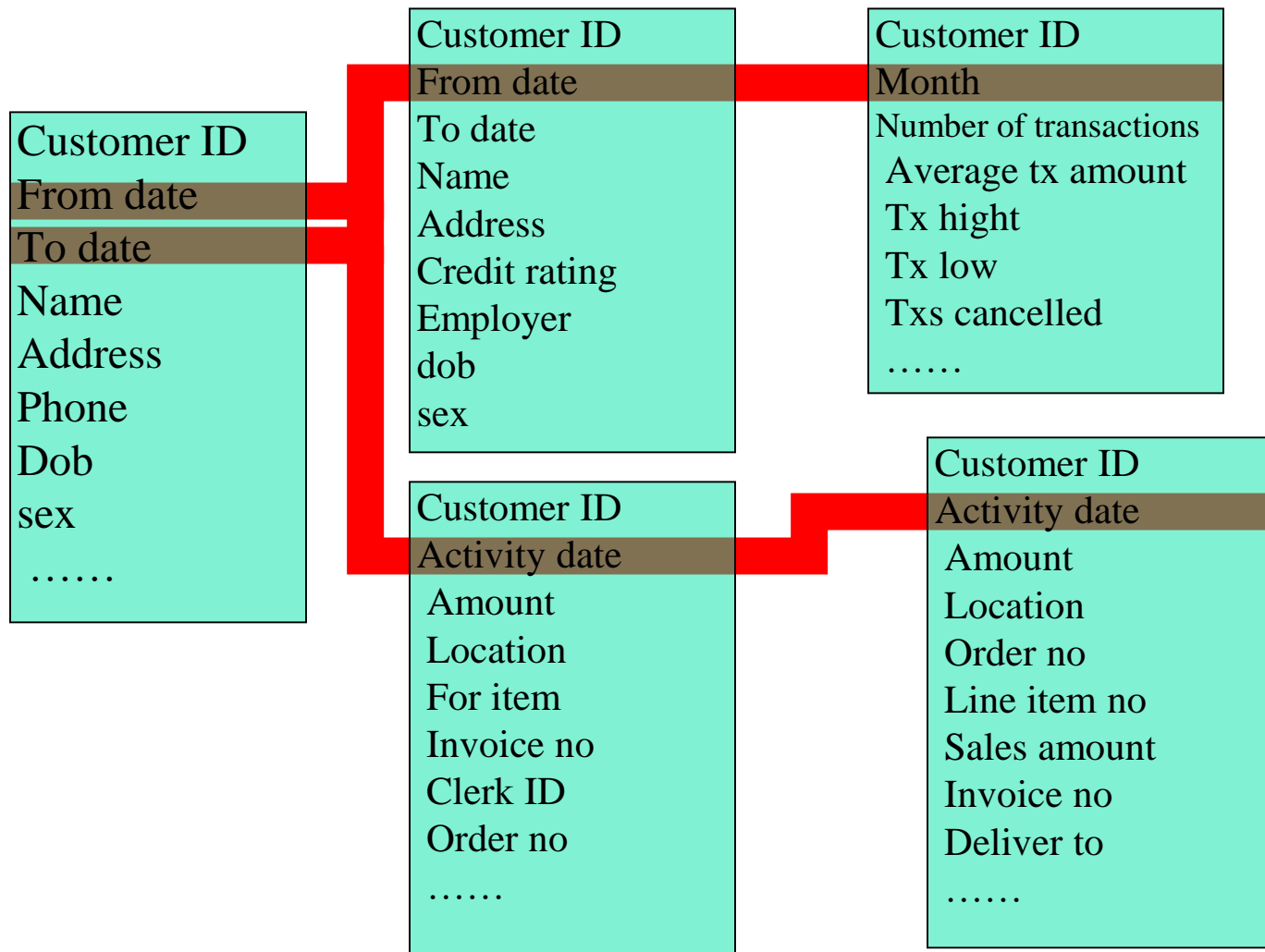


同一主题数据集一般具有共同主键





主题数据集中的数据中都具有时间





2. 粒度与多粒度级数据组织

- ▶ 粒度的概念与理解
- ▶ 平台中多粒度级数据关系
- ▶ 平台中粒度级设计因素

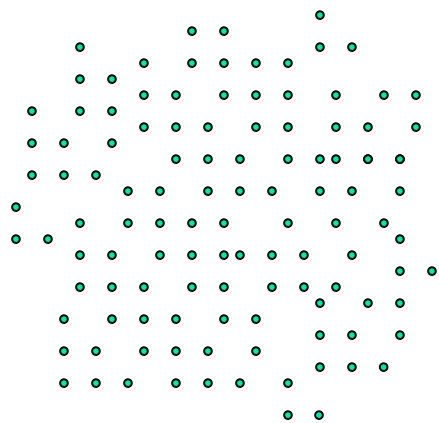


(1) 粒度(Granularity)

- ▶ 在数据仓库环境中，**分析应用需求**决定了环境中**必须具备不同粗细程度或层次的数据**。引出了**粒度**的概念：
 - **数据粒度**是描述数据环境中各种数据的**细节程度或综合程度**的高低的指标
- ▶ 数据细节程度越高，数据集的粒度级就越低，数据粒度越小
- ▶ 数据细节程度越低，数据集粒度级就越高，数据粒度越大
- ▶ 粒度级设计是数据仓库中一个重要的设计问题，影响整个平台的架构。

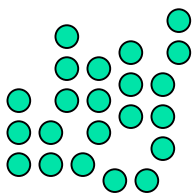


(2) 粒度的比拟

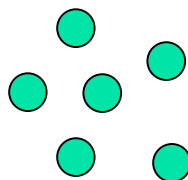


细沙粒

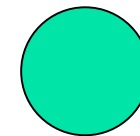
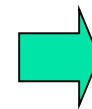
粒度级低



小沙粒



泥团

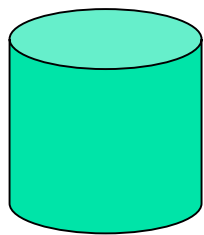


大石头

粒度级高

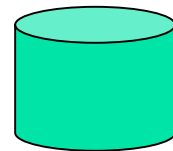


两级粒度案例



客户一个月内的所有
通话细节

低粒度级—高细节级

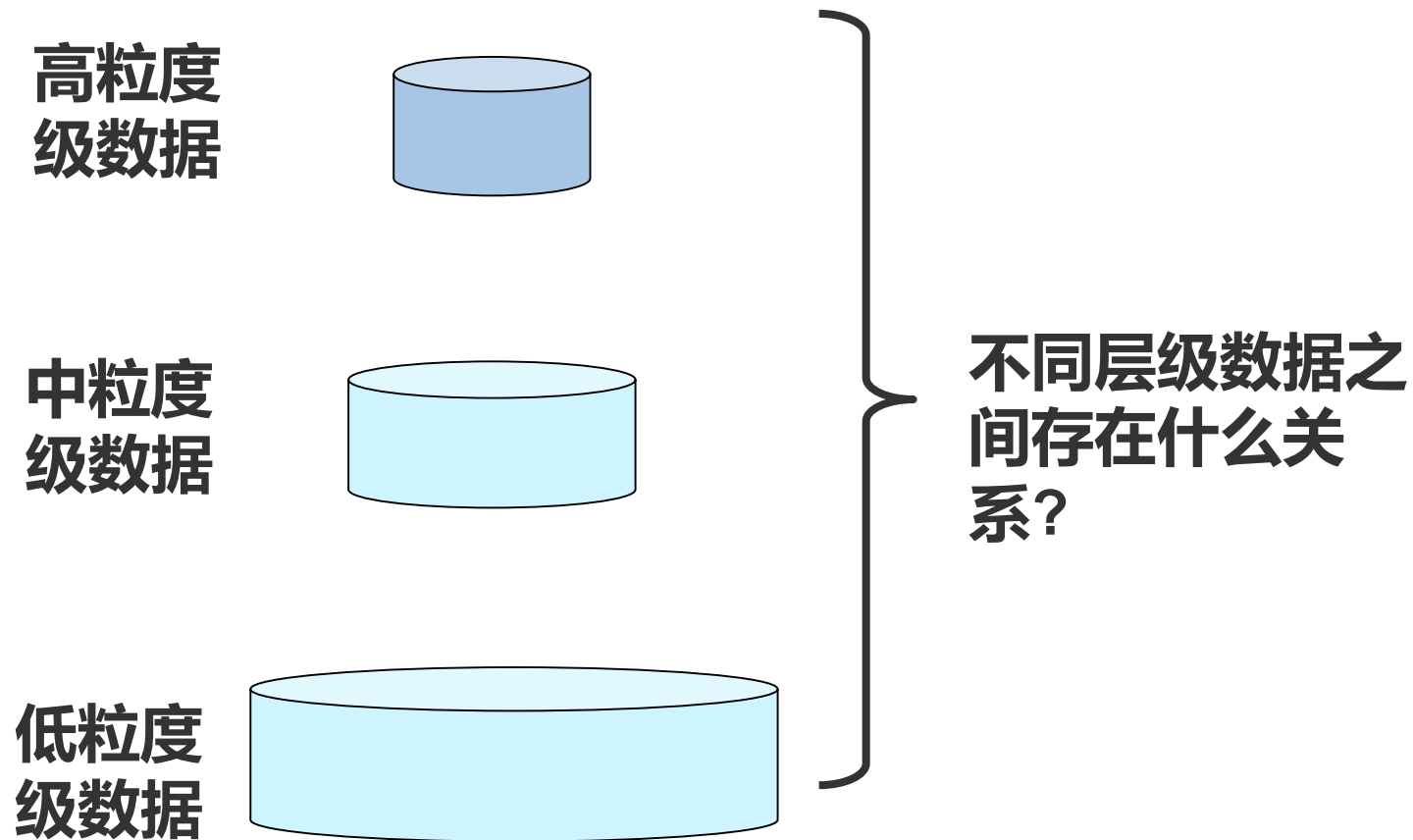


客户一个月内的电话
通话汇总

高粒度级—低细节级



(3) 主题数据区域内多粒度级数据间的关系



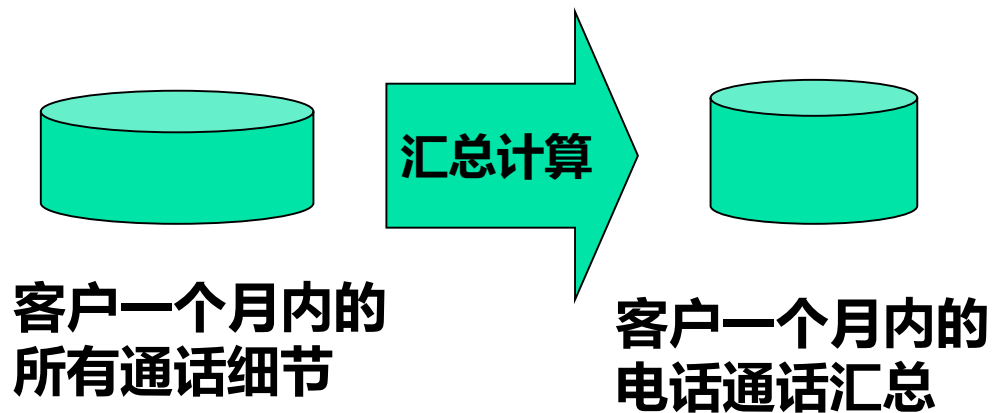
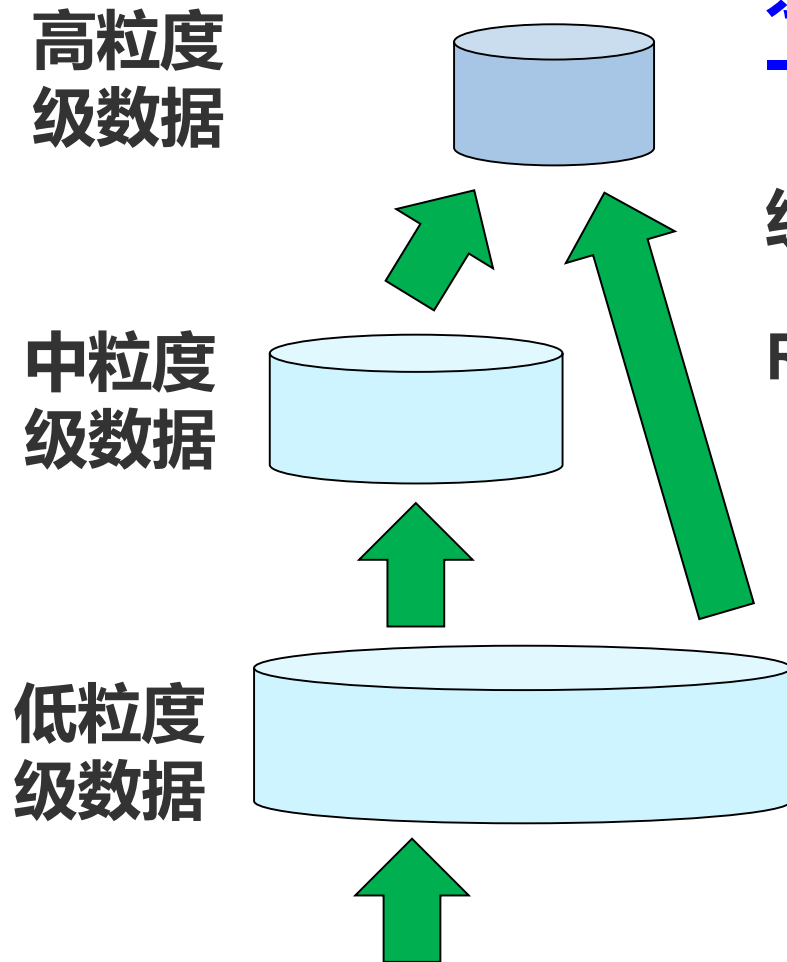


关系1—数据计算关系

关系1：计算关系，或数据依赖关系，从细节数据到高粒度级数据的多对一汇总关系。

具有细节数据，总能汇总得到高粒度级的数据。

对应于BI工具中的常见操作名称：
ROLL UP，卷起，上卷，汇总





回忆数据库中的分组聚集操作

► Grouping By & Aggregation

► Group By a_1, a_2, \dots, a_k

- 分组操作：将元组根据分组标准将 (a_1, a_2, \dots, a_k) 的属性值完全相同的元组归入同一组

► Aggregation Functions

- 根据多个元组的值，采用某个聚集函数，计算出一个指标
- 常见标准聚集函数：Sum, Avg, Count, Min, Max, ...
- 自定义任意多对一映射函数

► select $a_1, a_2, \text{sum}(\dots), \text{avg}(\dots), \text{count}(\dots)$ from ...

► where ...

► group by a_1, a_2



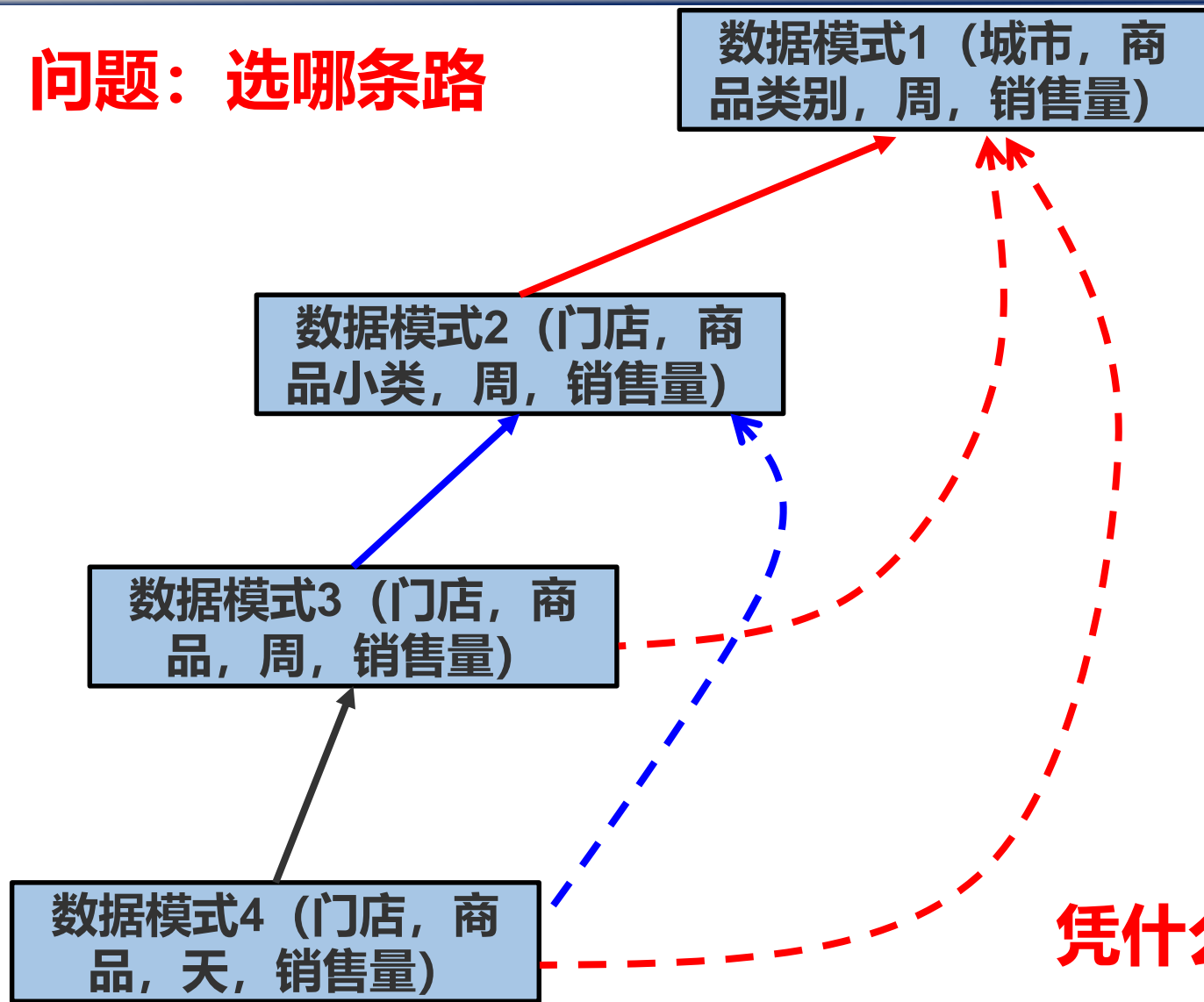
汇总计算关系案例

- ▶ 设有商品分析主题内如下各粒度级数据
 - 数据模式1 (城市, 商品类别, 周, 销售量)
 - 数据模式2 (门店, 商品小类, 周, 销售量)
 - 数据模式3 (门店, 商品, 周, 销售量)
 - 数据模式4 (门店, 商品, 天, 销售量)
- ▶ 计算任务1: 批量汇总计算—Batch maintenance
- ▶ 计算任务2: 增量汇总计算—Incremental maintenance
- ▶ 问题: 如何实现这些计算需求



批量或增量维护关系分析

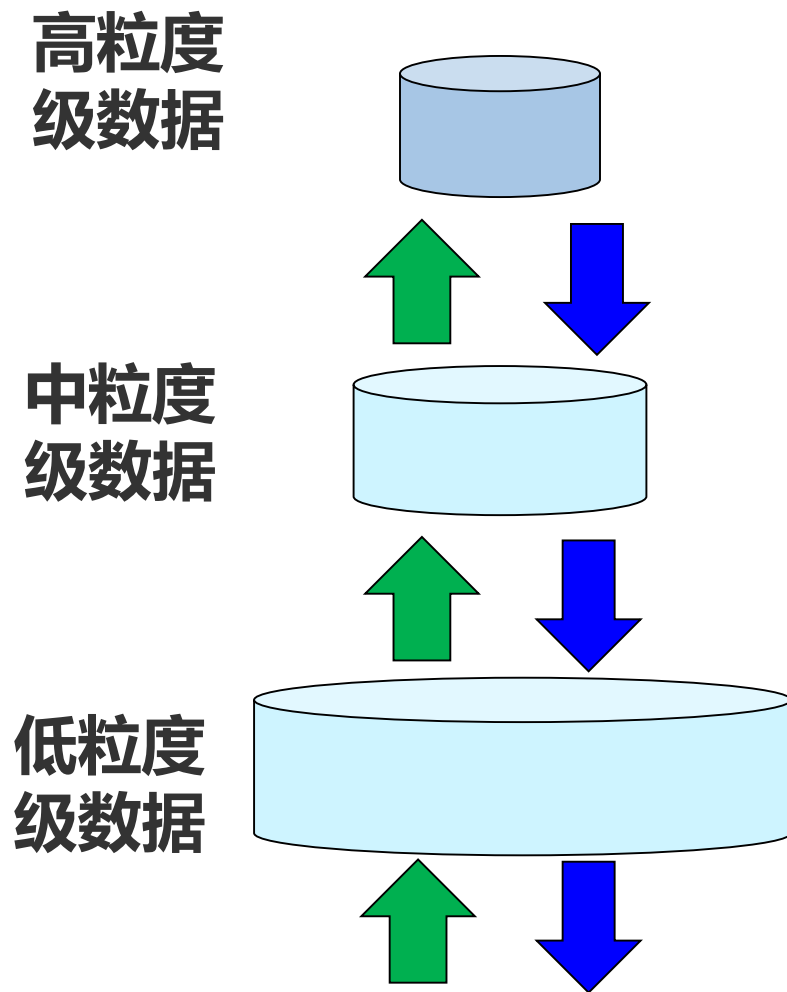
问题：选哪条路



问题：
凭什么去选择计算用数据源？



关系2——一对多细化分析关系



关系2：一对多细化分析关系，
高粒度级中的一条数据对应于低粒度级数据中的多条数据。

对应于BI工具中的常见操作，
Drill down：钻取，下钻，即
从某高粒度的数据项出发，进
而了解对应的细节数据



Drill Down相关问题案例与思考

► 设有如下各粒度级数据

- 数据模式1 (城市, 商品类别, 周, 销售量)
- 数据模式2 (门店, 商品小类, 周, 销售量)
- 数据模式3 (门店, 商品, 周, 销售量)
- 数据模式4 (门店, 商品, 天, 销售量)

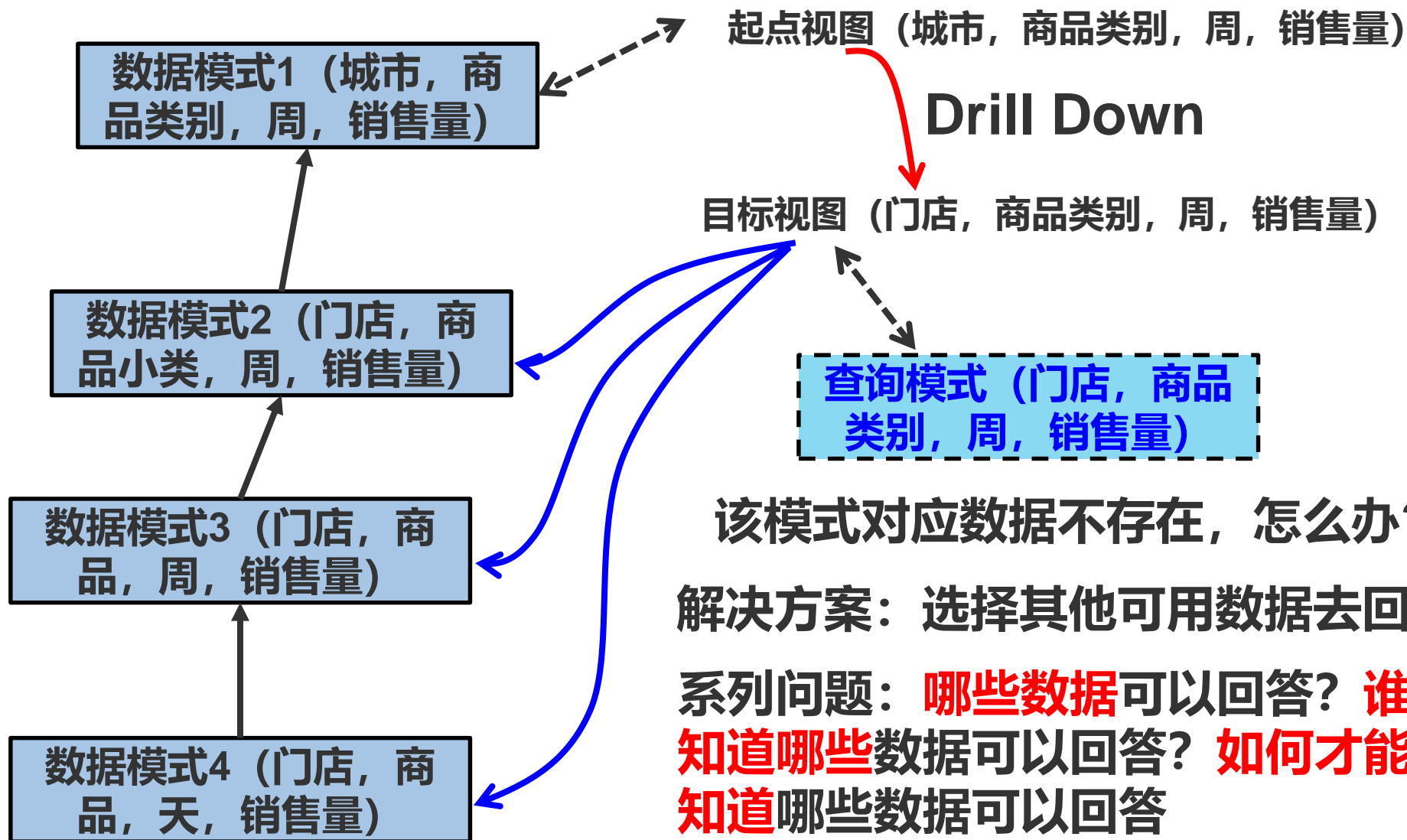
► 任务场景1:

- 用户界面数据视图模式与数据模式1一致, 管理者发现**上周城市A的商品类别B销量大增 (发现异常, 产生疑惑)**, 想看看该城市各门店商品类别B的上周销售情况 (**原因分析, 需要进一步的信息**), (Give me what I say I want, and I can tell you what I really want)
- 查询需求结果模式为 (**门店, 商品类别, 周, 销售量**), 显然, 系统中没有与本模式对应的数据集

► 问题: 如果响应这个Drill Down需求?



Drill Down关系分析





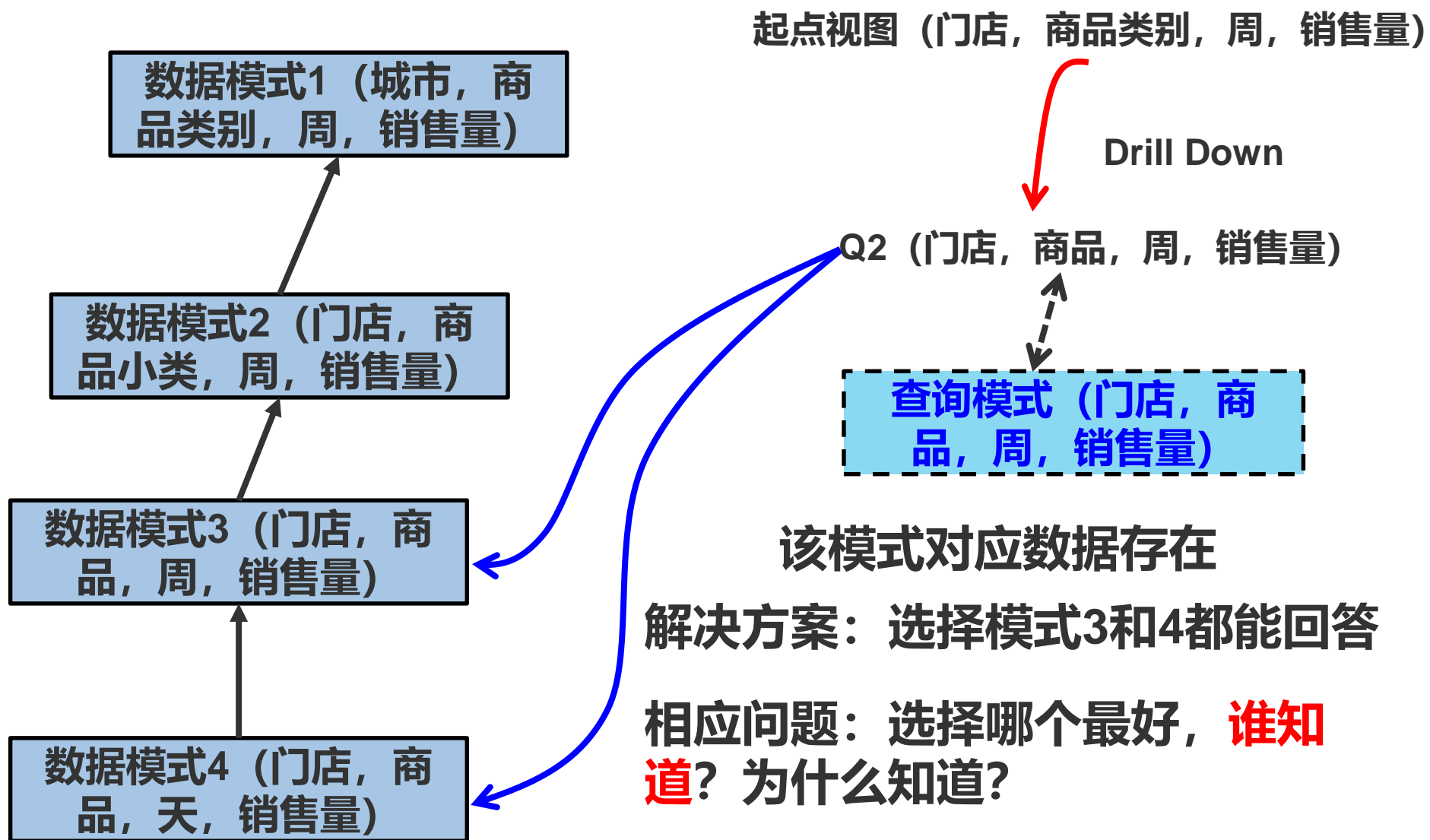
Drill Down相关问题案例续

► 任务场景2:

- 若已经切换到城市A各门店商品类别B的上周销售视图，视图2（门店，商品类别，周，销售量），发现**门店1**的**商品类别B**上周销售量特别大（**阶段性信息，管理者脑中疑惑得到部分解答**）。
- 此时想看看到底是**门店1**哪个**商品**上周销量特别大（**还存在疑惑，还需要进一步的信息**）。
- 进一步的查询需求结果模式为（门店，商品，周，销售量）
- 问题：如何响应这个Drill Down需求？

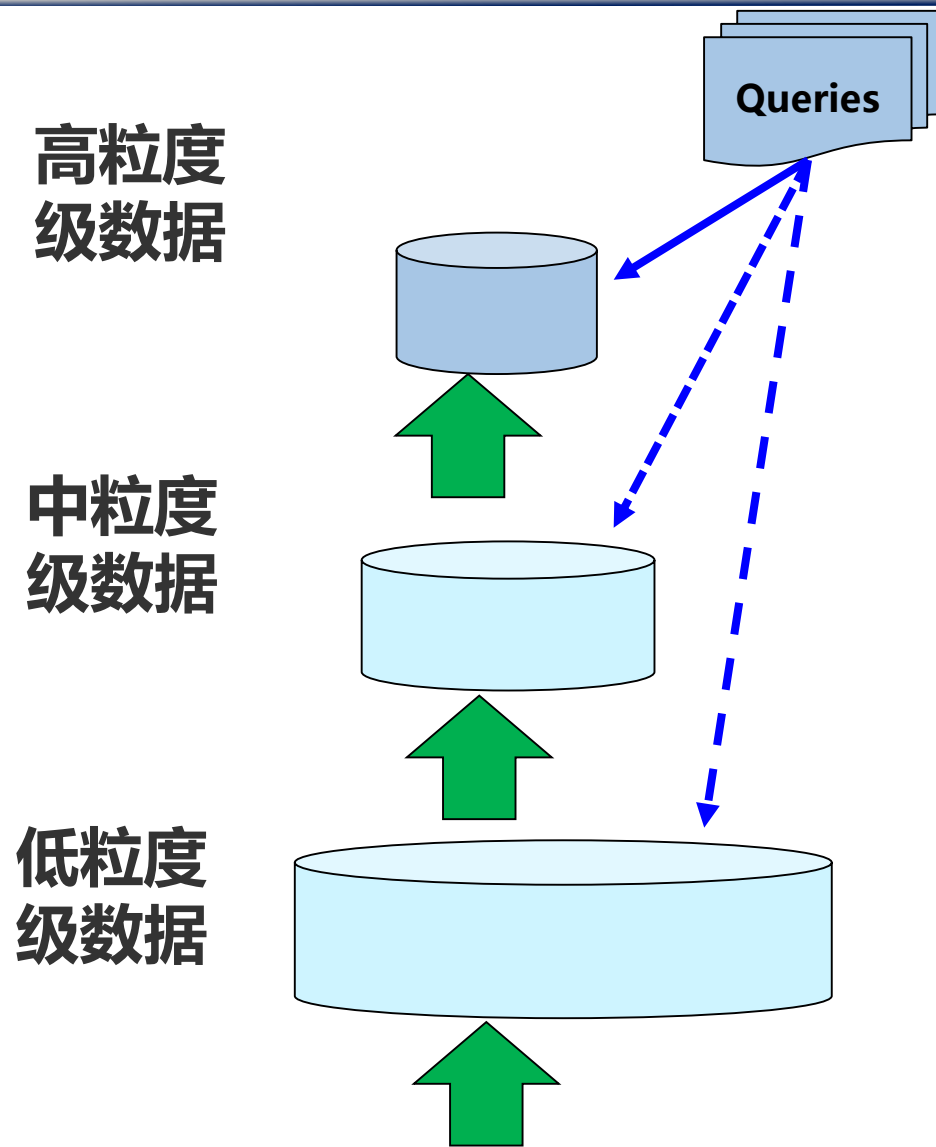


Drill Down关系分析





关系3—查询负载分担关系



关系3：查询负载分担关系

通过高粒度级数据来支撑粒度级在其之上的查询，降低低粒度级数据体系的查询支撑压力。

建立多粒度级数据的根本原因，空间换时间

在数据一致的情况下，在可回答的前提下，优先选择高粒度级数据回答查询。

如何保证多层数据间的一致性是数据维护的问题



集合性查询更为普通

- ▶ 在数据仓库的分析应用中，集合性查询更普遍，例如：
 - 北京移动用户上月平均打了多少个长途电话？
- ▶ 因此，综合型数据更能有效回答分析查询问题
- ▶ 普通分析型处理访问模式对数据的需求
 - 95%的信息处理发生在轻度综合级以上
 - 只有5 %的信息处理发生在细节层上。
- ▶ 但是，世界在变化...



多粒度级系统中的关键关系

- ▶ 必须保存各类模式对应的数据之间的关系
- ▶ 这种关系数据也是典型的元数据
- ▶ 关系的用途
 - 让**程序员知道**，根据数据间关系，编写定制程序，满足这种动态查询需求，这是多数工程实践中采用的方法
 - 用于**某些OLAP分析工具中间件**，中间件系统掌握数据间关系以后，自动生成计算代码，满足查询需求

相应知识：

OLAP分析工具或BI工具的常见功能、实现机制、内部结构
多维数据模型，元数据

数据模型设计部分将介绍相关概念



(4) 粒度级设计需要考虑的因素

▶ 粒度级设计

- 设计需要保存哪些粒度级数据，需要保存多长时间

▶ 常见因素

- 空间约束
- 时间约束
- 业务要求
- 其他要求（法律法规等）

▶ 粒度级设计方法，见后续数据模型设计



3. 环境中的数据分区或分割

- ▶ 分区定义
- ▶ 分区的必要性与普通性
- ▶ 分区的常见原则
- ▶ 分区透明性



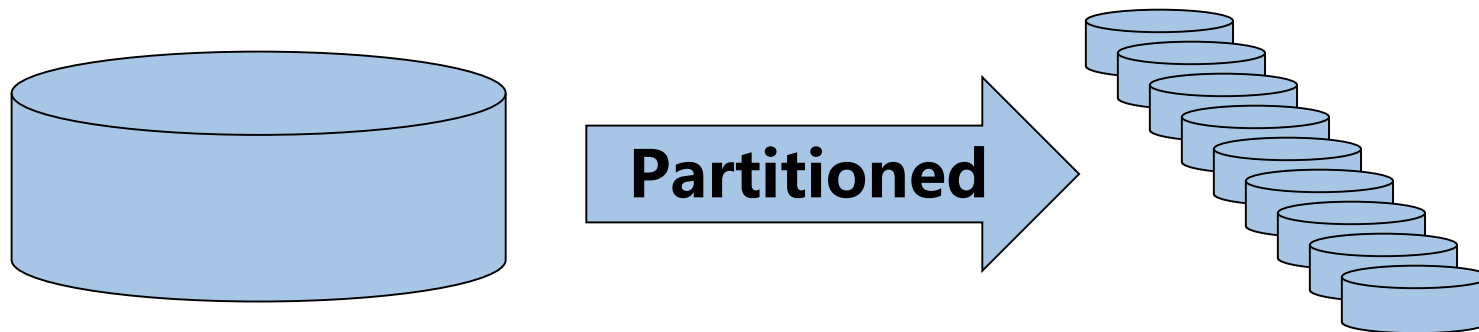
(1) 数据分区的定义

► Data Partitioning, Data Partition

- A partition is a division of a logical database or its constituent elements into distinct independent parts. Database partitioning is normally done for **manageability**, **performance** or **availability** reasons, or for **load balancing**

► 分区的目的

- 使得数据能够形成易于管理和操作的较小数据单元.





(2) 分区的必要性与普遍性

- ▶ 在数据仓库与大数据平台构建中，问题**不是要不要分区**的问题，而是**怎么分区**的问题。
- ▶ 如果数据单元过大，则下面的操作不好实现：
 - 重新调整结构，Restructuring
 - 索引，Indexing
 - 顺序扫描，Sequential scanning
 - 数据重组，Reorganization
 - 恢复，Recovery
 - 监控，Monitoring
- ▶ 因此，**几乎所有的细节数据都需要分区**



思考并回答问题

- ▶ 为什么将大数据集进行分割管理后有利于**数据的插入效率**?
- ▶ **数据分割**对**索引**有什么影响?
- ▶ 对大数据集进行数据分割为什么有利于**数据清除**?
 - 相似问题，什么样的房子好拆?
 - 相类似问题，为什么动车组容易装配
- ▶ 对大数据集分割与并行与分布式计算的关系?



(3) 两种基本数据分割方法

- ▶ 设有数据集 $S(a_1, a_2, \dots, a_n)$, 且有一个对 S 的分割方案 (S_1, S_2, \dots, S_k)
- ▶ **横向分割**——行分割
 - $\forall S_{i=1..k}$, S_i 的模式与 S 完全相同, 且 $\bigcup_{i=1..k} S_i = S$
- ▶ **纵向分割**——列分割
 - $\forall S_{i=1..k}$, 设其模式为 $(a_{i1}, a_{i2}, \dots, a_{im_i})$,
 $\bigcup_{i=1..k} \{a_{i1}, a_{i2}, \dots, a_{im_i}\} = \{a_1, a_2, \dots, a_n\}$, 且每个子模式的主键与 S 相同。



(4) 分区标准—划分属性

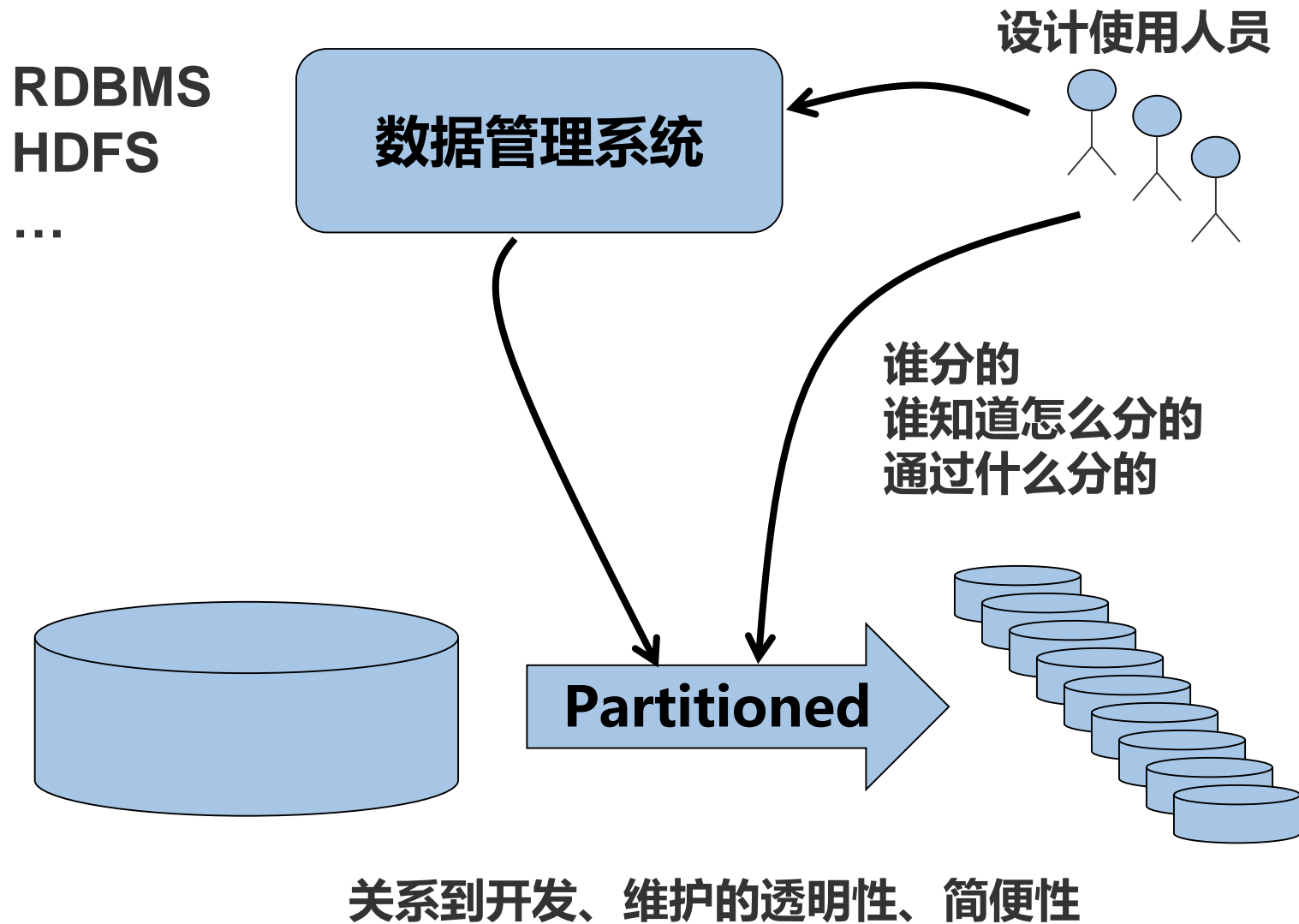
► 标准

- 根据时间范围
- 根据业务类别
- 根据地理位置
- 根据组织结构单元
- 根据记录主键序号范围
- 以上几种方式的结合

► 分区的选择由开发者决定，但是一般必须按日期进行划分。



(5) 数据分区与数据管理系统





分区与数据管理系统间的关系

► **系统层分区**—由数据管理系统层自动分

- 系统层的分区是系统提供的一个功能，各个分区在逻辑上一个表，物理上属于不同的分区，系统知道分区间的逻辑关系。
- HDFS自动切分后多备份存储，Oracle等RDBMS分区

► **应用层分区**

- 应用层的分区完成由应用代码实现，由开发者和程序员控制，系统并不知道分区间存在什么关系。
- 管理、设计都很麻烦，透明性不好
- 在一些关系型数据库，因为前后数据版本有小区别，不得已需要这样，HDFS能部分解决这样的问题



两种分区类别：应用层和系统层分区的优缺点

- ▶ 系统层分区：由DBMS实现数据分区
- ▶ 应用层分区：由应用平台实现数据的分区管理。
- ▶ 一般在系统层分区的便于访问，**少写访问代码**，在应用层的访问如果有涉及多个表，则需要**手工编写代码加以实现**。
- ▶ 但是，因为不同分区可能在结构上会有些不同，所以，也常常需要在应用层进行分区。
- ▶ 在关系型平台中，如果在系统层进行分区，则**分区间的数据的结构**必须是一样的，如果数据的时间跨度很长，则数据定义很有可能会发生变化。这样的话，这种方法就不可行。
- ▶ 多数情况，有可能是两种方法相结合。



Oracle的分区方法

► 范围分区

- 按照属性顺序范围

► 散列分区

- 按照主键序号散列

► 复合分区

- 以上两种方法的复合使用



分区举例

► 一个寿险公司的例子

- 2000 health claims
- 2001 health claims
- 2002 health claims
- 1999 life claims
- 2000 life claims
- 2002 life claims
- 2000 casualty claims
- 2001 casualty claims
- 2002 casualty claims

► 例子中的两个分区标准: year, type of claim



Hadoop distributed file system

- ▶ The Hadoop distributed file system (HDFS) is a **distributed, scalable, and portable file-system** written in Java for the Hadoop framework. A Hadoop cluster has nominally a single namenode plus a cluster of datanodes, although redundancy options are available for the namenode due to its criticality.
- ▶ Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses TCP/IP sockets for communication. Clients use remote procedure call (RPC) to communicate between each other.
- ▶ Because the namenode is the single point for storage and management of metadata, it can become a bottleneck for supporting a huge number of files, especially a large number of small files.



HDFS

- ▶ HDFS stores **large files** (typically in the range of gigabytes to terabytes) **across multiple machines**. It **achieves reliability** by replicating the data across multiple hosts, and hence theoretically **does not require RAID storage** on hosts (but to increase I/O performance some RAID configurations are still useful).
- ▶ With the default replication value, 3, data is stored on three nodes: **two on the same rack, and one on a different rack**.
- ▶ Data nodes can **talk to each other to rebalance data**, to move copies around, and to keep the replication of data high.



- ▶ HDFS was designed for mostly **immutable files** and may not be suitable for systems requiring **concurrent write-operations**



4. 活样本数据集组织方法

- ▶ **Living Sample Database or Dataset**
 - ▶ **一种特殊的数据设计**
 - **原因：数据量太大无法完整保存全量**
 - **活样本：需要周期性的刷新的数据仓库数据的一个子集。**
 - ▶ **在有些情况下，这种设计**
 - **非常有用**
 - **可以节省大量的资源**
 - **访问起来很方便**
- 全社会行业统计样本数据
语料库**



缺点、适用场合和采样方法

► 严重的缺点

- 并不是一个通用目的的数据库，仅仅是样本
- 不适合于需要处理单条记录的数据

► 适用场合

- 统计分析
- 趋势分析
- 综合性、集合性视图
- 机器学习

► 采样方法

- 随机



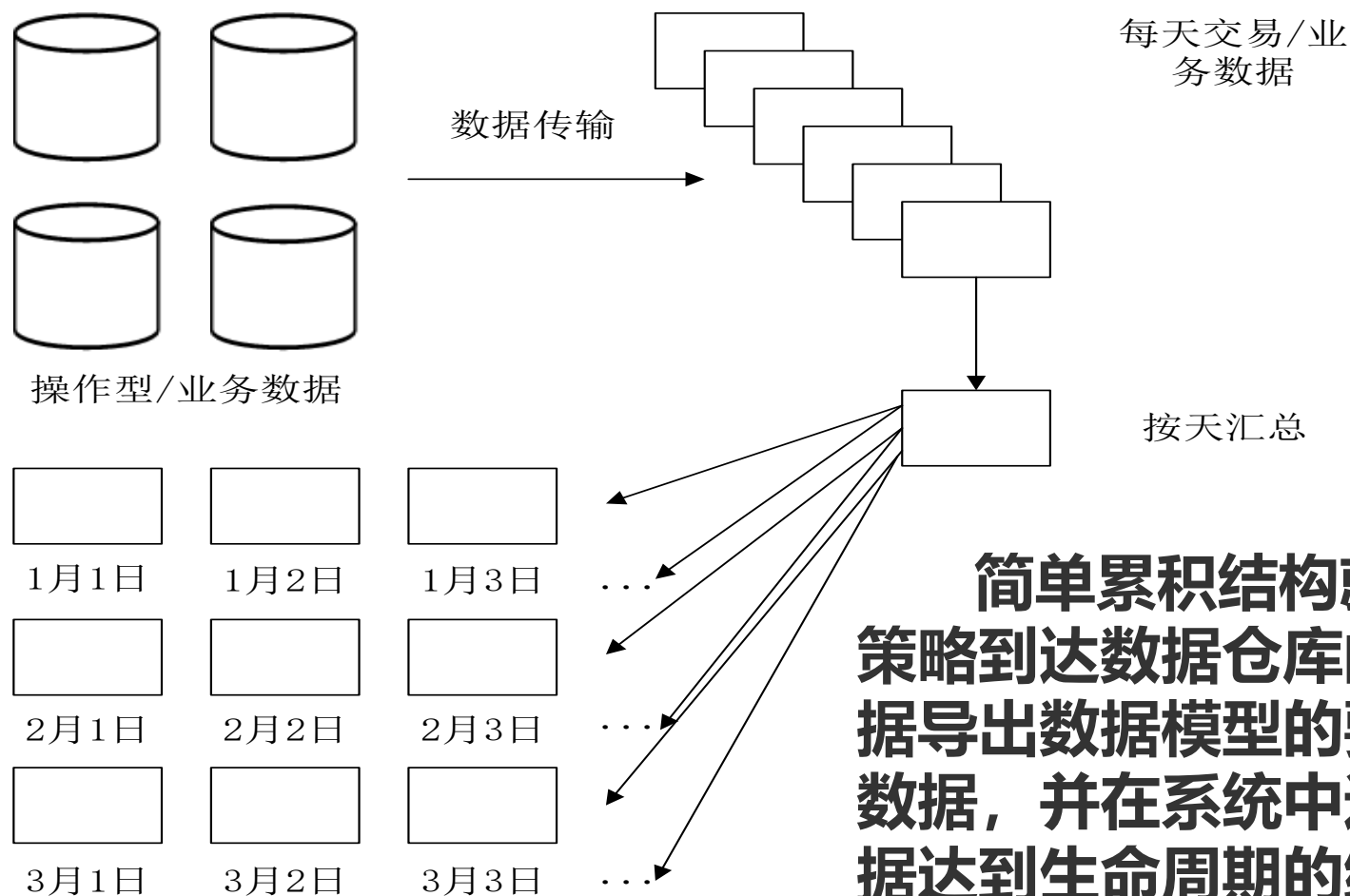
5. 三种常见导出数据组织形式

► 三种常见的导出数据组织形式

- Simple cumulative structure, 简单累积
- Rolling summary, 轮转综合
- Simple Direct, 简单直接文件



(1) 简单累积



简单累积结构就是系统以按一定时间策略到达数据仓库的细节数据为基础，根据导出数据模型的要求，不断地生成导出数据，并在系统中进行累积，直至导出数据达到生命周期的终点才清理出系统。

需要以强大的存储能力作为保障



案例

- ▶ 例如，假设铁路客票系统产生的新销售记录每天按一定的时间间隔分批次进入客运大数据平台。
- ▶ 平台中除将新销售记录按要求进行累积存储到存储平台以外，还需要根据销售记录进行每日售票统计，例如，假设需要按模式1：（车次，发站，到站，发车日期，座位级别，销售日期，销量）生成各车次分OD分类别座位销售情况数据。
- ▶ 在这个例子中，简单累积策略根据原始客票销售数据与模式1之间的计算关系，采用一定的计算策略，每生成模式1所对应的数据时，在平台不断累积并保存数据。直到数据达到较长的年限，如5年以后，才可能会清理出平台。



例：简单累积结构的使用

► 下列数据表哪些适合于简单累积结构

- 商场销售数据表
- 商场顾客表
- 商场销售人员表
- 银行存取款交易数据表
- 银行网点名单数据表
- 电信通话记录表
- 铁路或航空售票数据表
- 航班查询日志

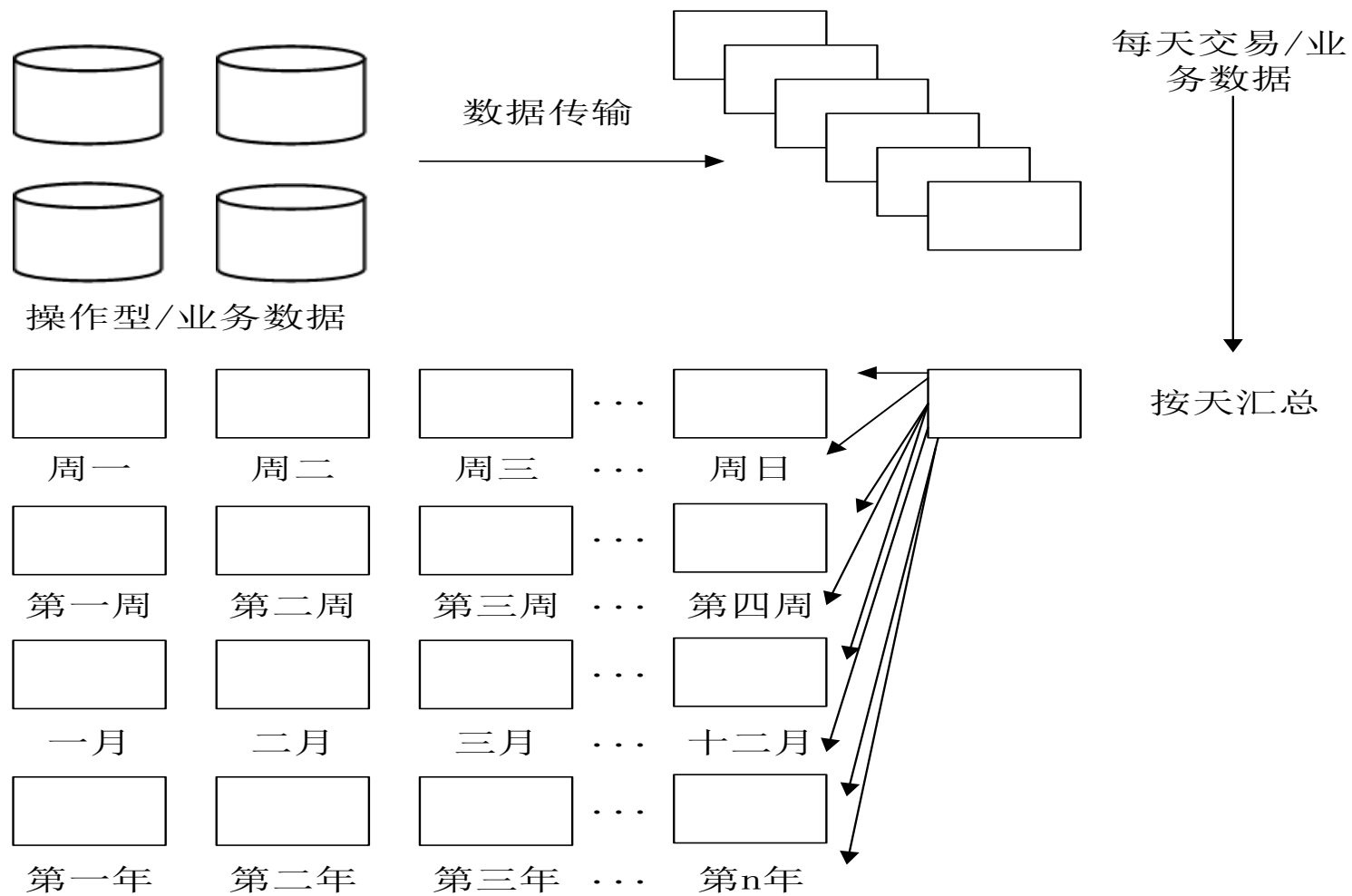


(2) 轮转综合方式

- ▶ 在有些应用背景中，特别是存储空间有限情况下，也可能会采用轮转综合策略。
- ▶ 轮转综合策略在**存储空间有限且应用需求允许**的情况下，将有限的存储空间划分给彼此相关的一组不同层次的导出数据使用，**每个层次的导出数据只保留长度固定的时间跨度的数据**。
- ▶ 例如，将系统中某个主题紧密相关的导出数据划分日、周、月、年四个级别。
 - 天级别数据最多循环保存最近的15天
 - 周级别数据最多保留最近的10周
 - 月级别数据最多保存最近的24个月
 - 年级别数据最多保存最近5年
- ▶ 并根据这个原则以轮转的方式将指定时间级别的最老数据清除出平台。



轮转综合示意图





轮转综合

- ▶ 在现实环境中，轮转综合是一种很有用的策略，特别是细节数据规模巨大，存储空间又非常有限，且各层级数据时间窗口的设计能满足应用要求的情况下，是非常有益的策略。
- ▶ 但是，其缺点也很明显，即在设置的时间周期进行的轮转的过程中，**各层级数据都会丢失数据**。每层的时间窗口设得越小，丢失的相应层级的数据就越多。
- ▶ 当然，在累积策略中，因为导出数据达到一定的年限以后也会被清理出平台。因此，在轮转综合策略中，极端情况下如果将相应层的导出数据的时间窗口设置成与累积策略相同时长，则该策略等价于累积式的策略。



轮转综合与简单累积的比较

► 轮转综合

- 紧凑
- 丢失一些细节
- 越旧的数据，丢失的细节越多

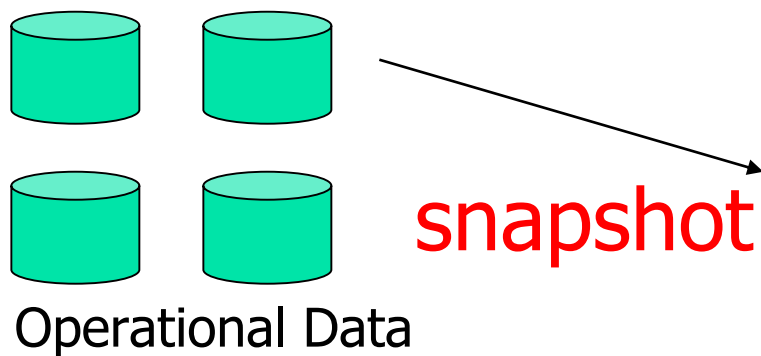
► 简单累积

- 需要大量的存储
- 不丢失细节
- 需要有更多的数据处理功能



(3)快照拼接策略

- 快照拼接策略是指根据同一个数据集不同时间点上的快照数据集或新增变化数据，生成反映事物状态变化的连续数据条目，形成连续数据文件的策略。



J Adams	123 Main Street
P Anderson	456 High Street
K Appleby	10 A Street
L Azimoff	64 N Ranch Rd.

January Customers



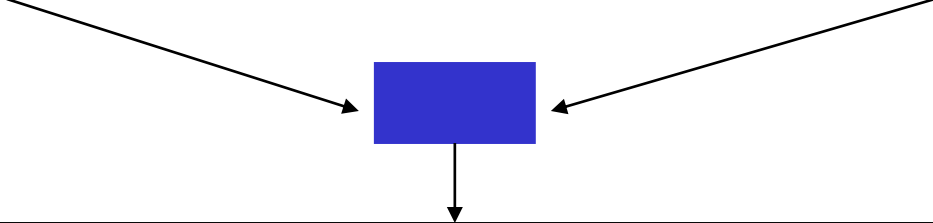
根据快照拼接生成连续文件

January Customers

J Adams	123 Main Street
P Anderson	456 High Street
K Appleby	10 A Street
L Azimoff	64 N Ranch Rd.
.....	

February Customers

J Adams	123 Main Street
W Abraham	12 Hwy 9
P Anderson	1455 Tincup Ct
K Appleby	10 A Street
L Azimoff	64 N Ranch Rd.
.....	



J Adams	Jan-present	123 Main Street
W Abraham	Feb-present	12 Hwy 9
P Anderson	Jan-Jan	456 High Street
P Anderson	Feb-present	1455 Tincup Ct
K Appleby	Jan-present	10 A Street
L Azimoff	Jan-present	64 N Ranch Rd.
.....		



案例示意

- ▶ 再如，假设铁路客票系统会根据旅客的消费历史及旅行记录进行部分旅客信用评级，每次评级以后将评级有变化的用户数据发送到数据仓库平台，此时就可以根据原有旅客信用历史数据，根据新的用户信用变化数据，形成新的旅客连续信用记录

旅客ID	信用级别	开始时间	结束时间
ID1	C	2014.6.1	2015.7.31
ID1	B	2015.8.1	2017.1.10
ID1	A	2017.1.11	
ID2	B	2015.7.1	2017.3.1
ID2	A	2017.3.2	
ID3	B	2014.5.31	
...			



快照拼接及数据维护策略

- ▶ 显然，在以快照的形式从数据源获取变化数据时，实施拼接策略后，将不断地根据连续文件和新的快照进行拼接。
- ▶ 完成拼接以后，若新快照的变化内容都已经反映到相应的连续数据中，则拼接过的快照数据原则上就可以抛弃了，因此，拼接策略实际上也是一个**压缩操作**。
- ▶ 当然，如果业务规则要求或空间允许，也可以将这些快照以原始备份数据的形式备存。



6. 整个系统架构中的操作型数据窗口问题

- ▶ What's the "Window of Opportunity"?
- ▶ 机会之窗，时间窗口，时间段
- ▶ 例
 - We have a window of opportunity here to deal with this disease.
 - News analysis: "window of opportunity" for state banks to reduce NPLs(non-performing loan)
 - Window of opportunity to back up data



问题

- ▶ 数据仓库/大数据平台中的数据是一种历史数据，档案数据
 - 给定一种细节数据集 D ，按固定时间间隔（秒、分钟、小时或天） g 进行更新，设平台中 D 的最老数据的时间为 T_e ，最新数据的时间 T_n ，当前时间为 T_c ，下一次更新时间为 T_{n+1} 。
 - 则平台中的细节数据集 D 的时间窗口如图：





数据仓库或大数据平台中的历史数据

► 为什么要有这么多的数据？

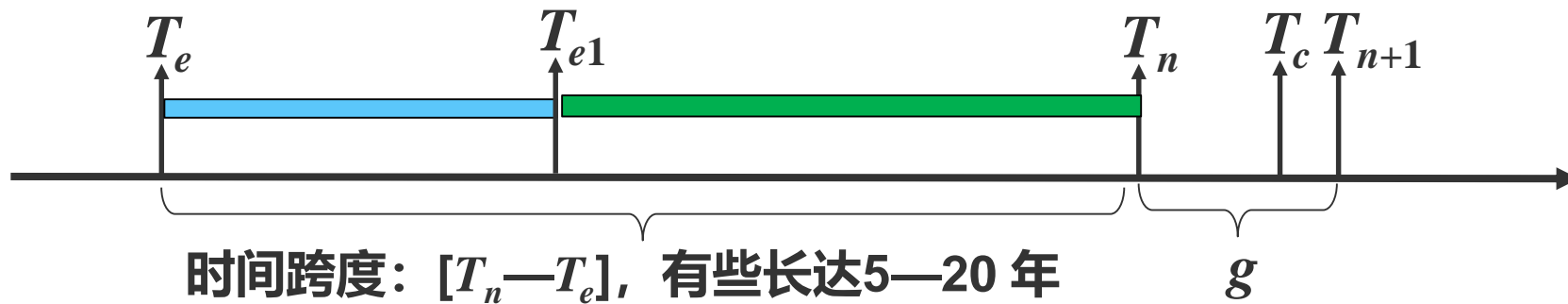
- 历史业务查询
- 分析型应用的要求，长期规律性分析
- 比如，做时间序列预测，做拟合，都需要有一定数目的历史时间序列数据作为输入。

在整个架构环境中，还有任何其它的细节数据吗？在何处？
有，在操作型环境中

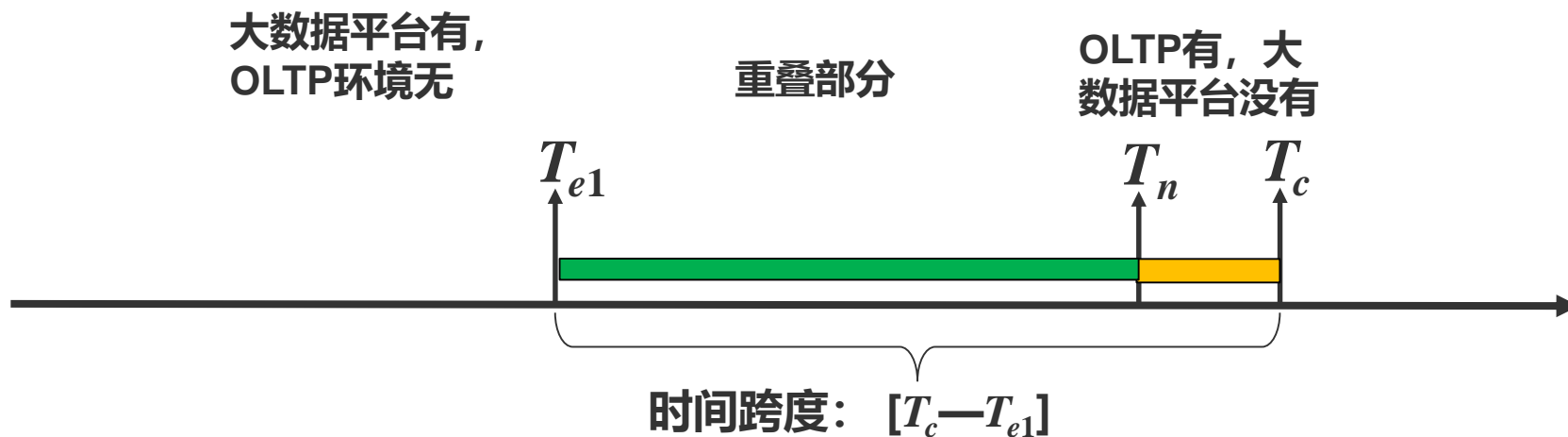


大数据平台与OLTP环境的细节数据时间窗口差异

DW/BigData
细节数据



OLTP
环境的细节数据



操作型环境的细节型历史数据：新、量小
有些数据并不进入数据仓库或大数据平台



银行案例—潜在可用查询数据源分析



查询1: 当天的交易记录

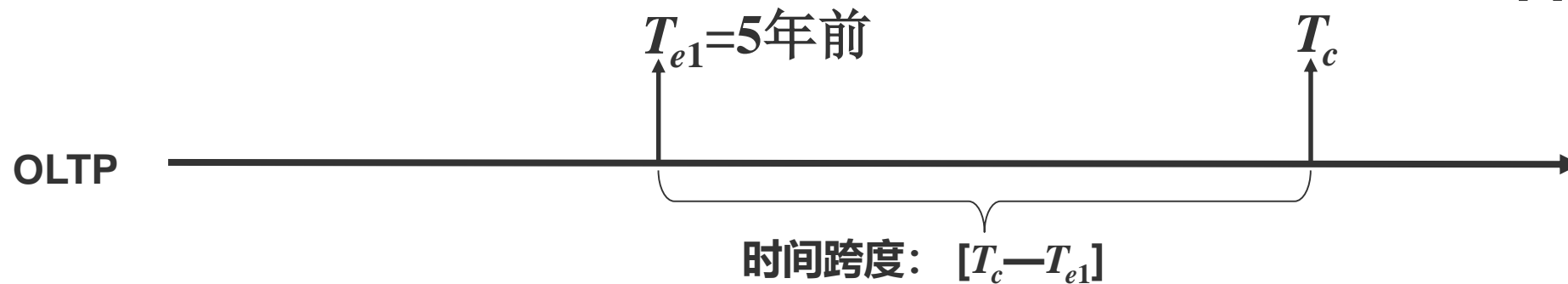
查询4: 8年前某月的是否给某个商店付了一笔款

查询2: 本月的交易记录

查询5: 12年前的某笔业务是否有问题

查询3: 上个月的交易记录

若 $g=1$ 秒, 则...





Operational windows of industries

- ▶ 行业间的OLTP系统操作型窗口的大小各不相同。
 - 保险业，面向用户直接交易少，数据量比较小
 - 操作型窗口10年以上，10-30年
 - 有些银行提供交易明细查询有很大的变化
 - 中行：5年
 - 工商：10年
 - 中信：6年
 - 建行：2年
 - 一些大型企业，业务多，不同操作型系统下的操作型窗口大小可能各不相同。



► 操作型环境的硬件处理能力

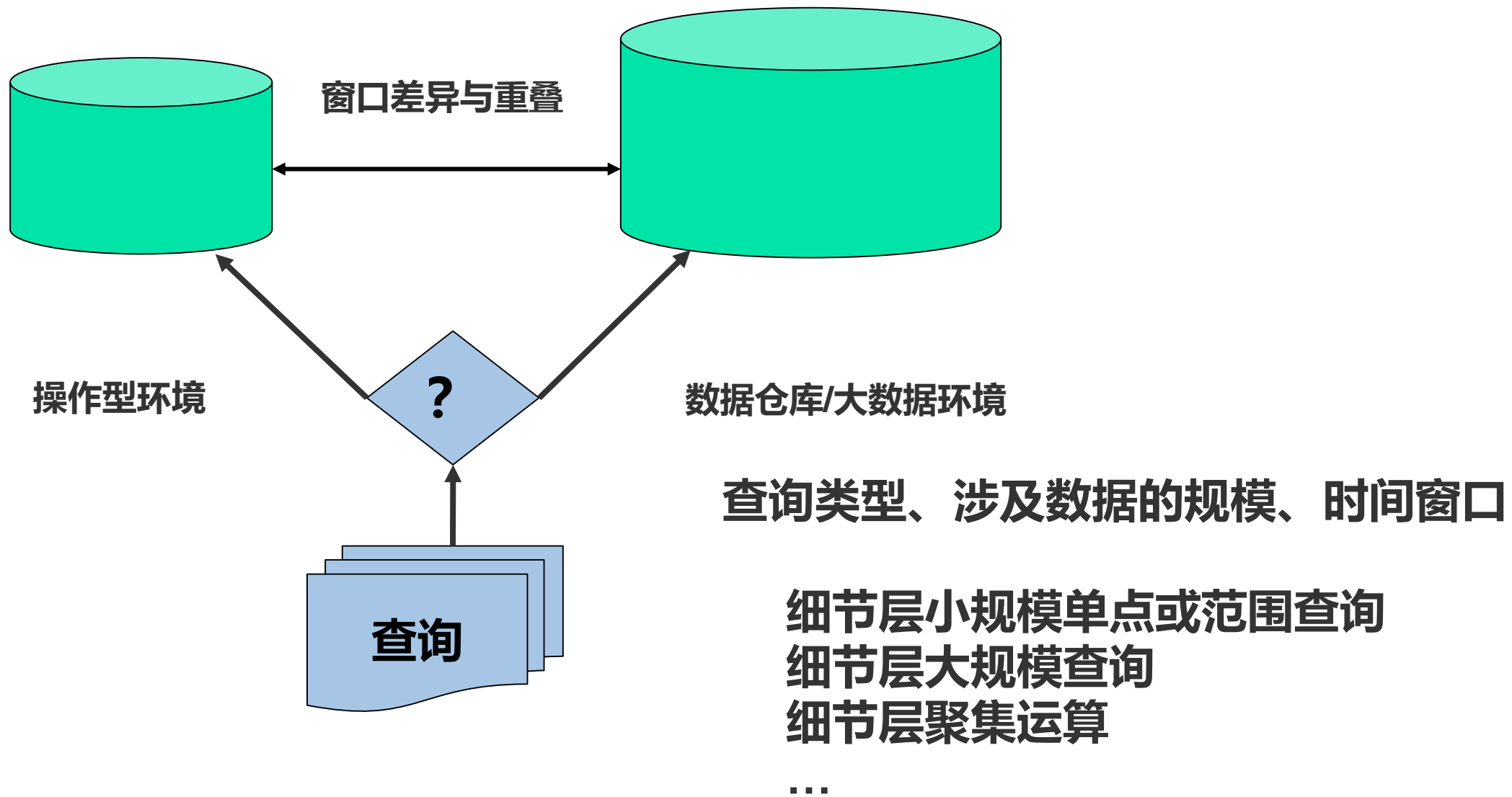
- 可用的存储空间
- 并发处理能力

► 业务发展需求

- 行业竞争需求，为用户提供更好的细节数据查询服务



整个架构中细节数据查询服务分布决策





银行案例续—查询数据源决策与应用架构

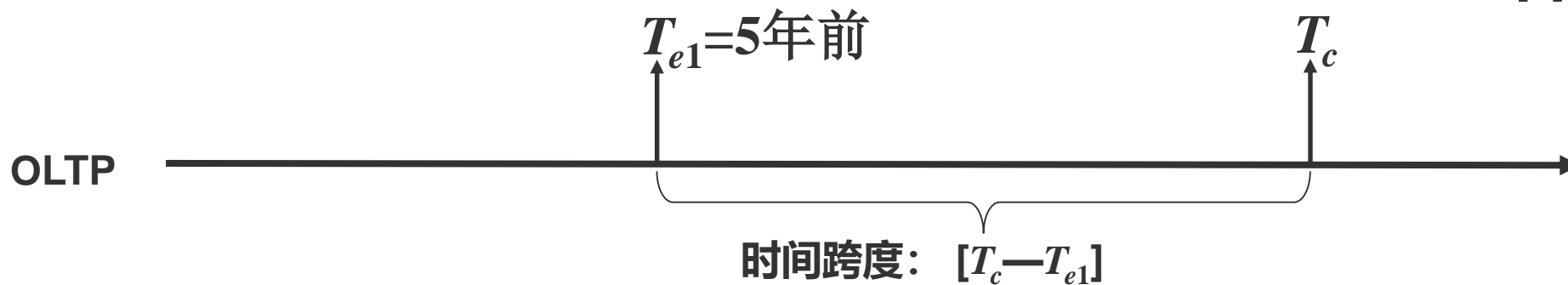


场景1：反贪污反洗钱业务，调查多账户间历史潜在异常交易

场景2：反诈骗业务，

- (1) 寻找潜在诈骗用银行账号；
- (2) 实时发现、拦截或押后潜在受骗交易。

若 $g=2$ 秒，则...





7. 数据清除问题

- ▶ **数据清除是一个很重要的设计问题；**
- ▶ **在数据仓库中，数据也具有自己的生命周期；**
- ▶ **数据仓库中的一些数据将在某个时间点上清除出数据仓库。**
- ▶ **因此，在数据仓库设计时，应根据数据的组织形式和一致性要求，对数据仓库中数据清理过程进行设计。**



几种数据清除方式

- ▶ 数据增加到轮转综合中，但细节数据丢失。
- ▶ 数据从高效即时访问设计如磁盘转移到大容量离线存储设备上。
- ▶ 数据从系统中进行了实际删除处理。
- ▶ 数据从一种体系结构转移到另一种体系结构中如从操作型系统到数据仓库中。



8. 数据仓库中错误数据处理

- ▶ 作为一个数据仓库设计人员，必须清楚如何处理数据仓库中的错误数据。
- ▶ 数据出错的情况分为两大类
 - 个别例外的情况
 - 大批量的错误

思考问题：

个别例外的错误与大批量错误产生的根源有什么区别？

源头的错误 vs 处理工具的错误



批量型错误

- ▶ **对于批量型的错误，必须找出错误产生的根源**
 - 一般是某个ETL工具出了问题。
 - 一般来说，即使是ETL工具本身没有问题，仍然会些错误数据进入数据仓库环境，
- ▶ **该如何处理这些错误数据**
 - 至少有三种选择方案
 - 各有优劣，没有一种是完全正确或错误。
 - 某一种选择只有某些条件下，比其它别的选择要好。



例如

- ▶ 假设7月1日在操作型系统中账户ABC加入了一条5000元的账目。7月2日在数据仓库中为账户ABC产生了这5000元的一个快照。
- ▶ 接着在8月15日发现了错误。这一账目不应该是5000元，而应是750元。如何纠正数据仓库中的数据呢？



方法1

- ▶ **进入7月2日的数据仓库并找到错误的条目，然后使用更新功能，将5000元替换为750元。如果这可行的话无疑是一种干净彻底的解决方案，但它却引发了新的问题：**
 - **数据集成被破坏。所有在7月2日与8月16日之间生成的报表都将不能一致起来。**
 - **更新必须在数据仓库环境中进行。**
 - **多数时候不是要修正一个条目，而是有很多很多的条目要修正。**



方法2

► 加入修正条目。

- 加入8月16日两个条目，一条是 - 5000元，另一条是 + 750元。这是数据仓库中7月2日与8月16日之间数据仓库中最新数据的最好反映。

► 这样做也有一些缺点：

- 可能要修正很多条目，而非一个。可能进行一项简单的调整也非常困难。
- 有时候由于修正公式非常复杂以至于调整根本是不可能的。



方法3

- ▶ **8月16日重新设置账户的正确数值。8月16日的账目反映了当时账户的余额，而不考虑以前的活动。8月16日加入了一条750元的条目。**
- ▶ **这种方法也有缺点：**
 - **及时简单地将账户重设为当前值需要对应用与过程进行约定。**
 - **这种重设的方法不能对过去的错误进行准确的解释。**



内容提纲

数据仓库与大数据主要特征

数据仓库与大数据平台总体数据架构

数据仓库与大数据平台数据组织结构

数据环境组成与应用支撑

软硬件技术环境



其他数据环境组成与应用支撑

- ▶ ODS环境及其应用支持
- ▶ 常见数据集市环境形态
- ▶ 数据架构间关系与设计



1. ODS环境及其应用支持

- ▶ ODS概述
- ▶ ODS定义
- ▶ ODS功能与实现机制



(1) ODS概述

► 数据仓库的主要贡献

- 明确提出数据处理的两种不同类型:
 - 操作型处理, OLTP
 - 分析型处理, DSS
- 将两种处理在实现中区分开来
- 建立起OLTP-DW/BigDB两层数据存储体系
- 解决数据集成的问题

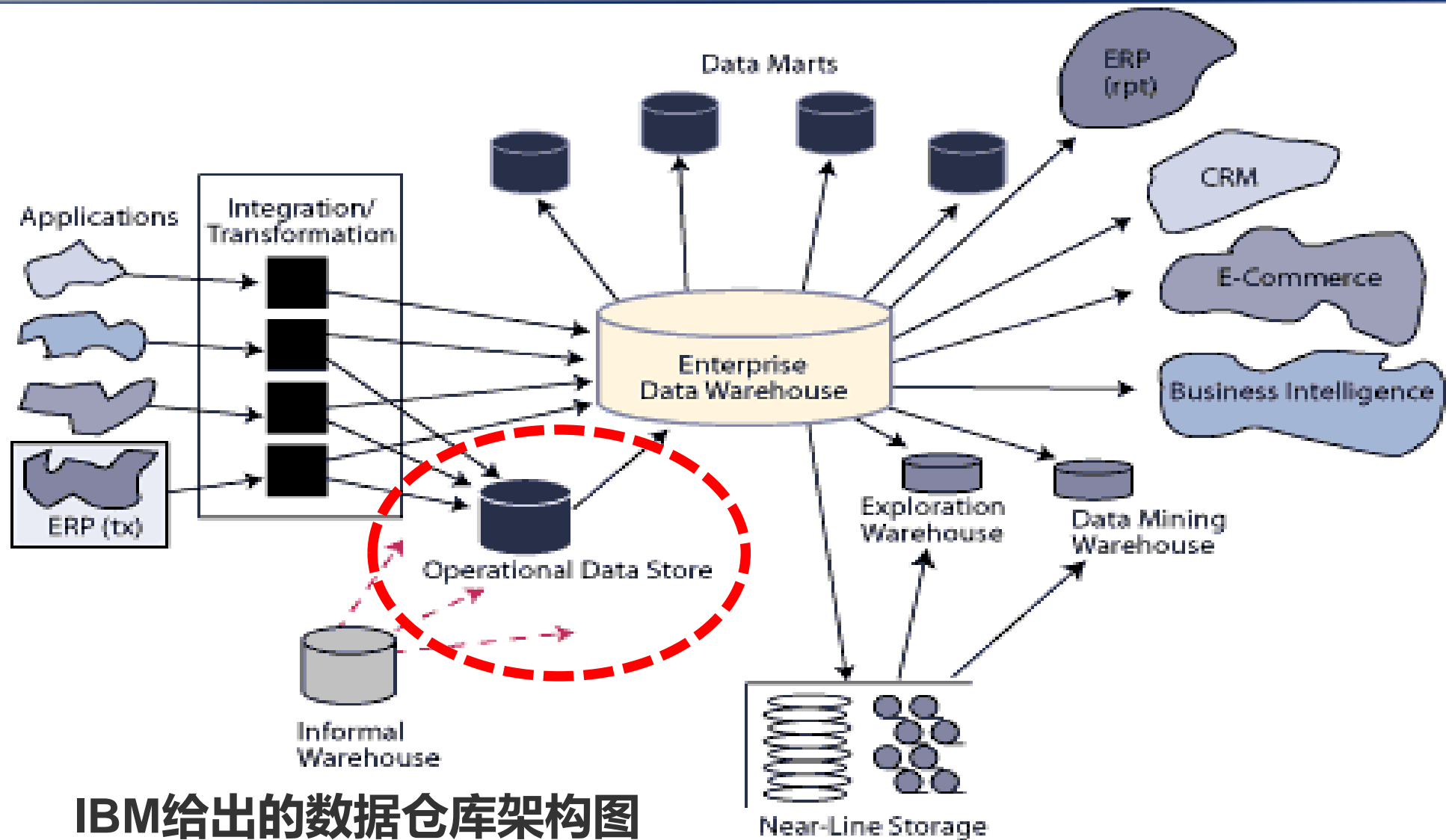


存在问题

- ▶ 在很多情况下，OLTP-DW/BigDB的两层体系结构并不能涵盖企业所有的数据处理要求。
 - 因为企业的数据处理虽然可以较为粗略地划分成操作型和分析型两部分，但有时，这两种处理之间并没非常明晰的界限。
- ▶ 实际的数据处理往往是多层次的。
 - 也就是说，有些处理是操作型的，但不适合在操作型DB中进行，而又存在着一些分析型处理，但不适合在DW中进行。



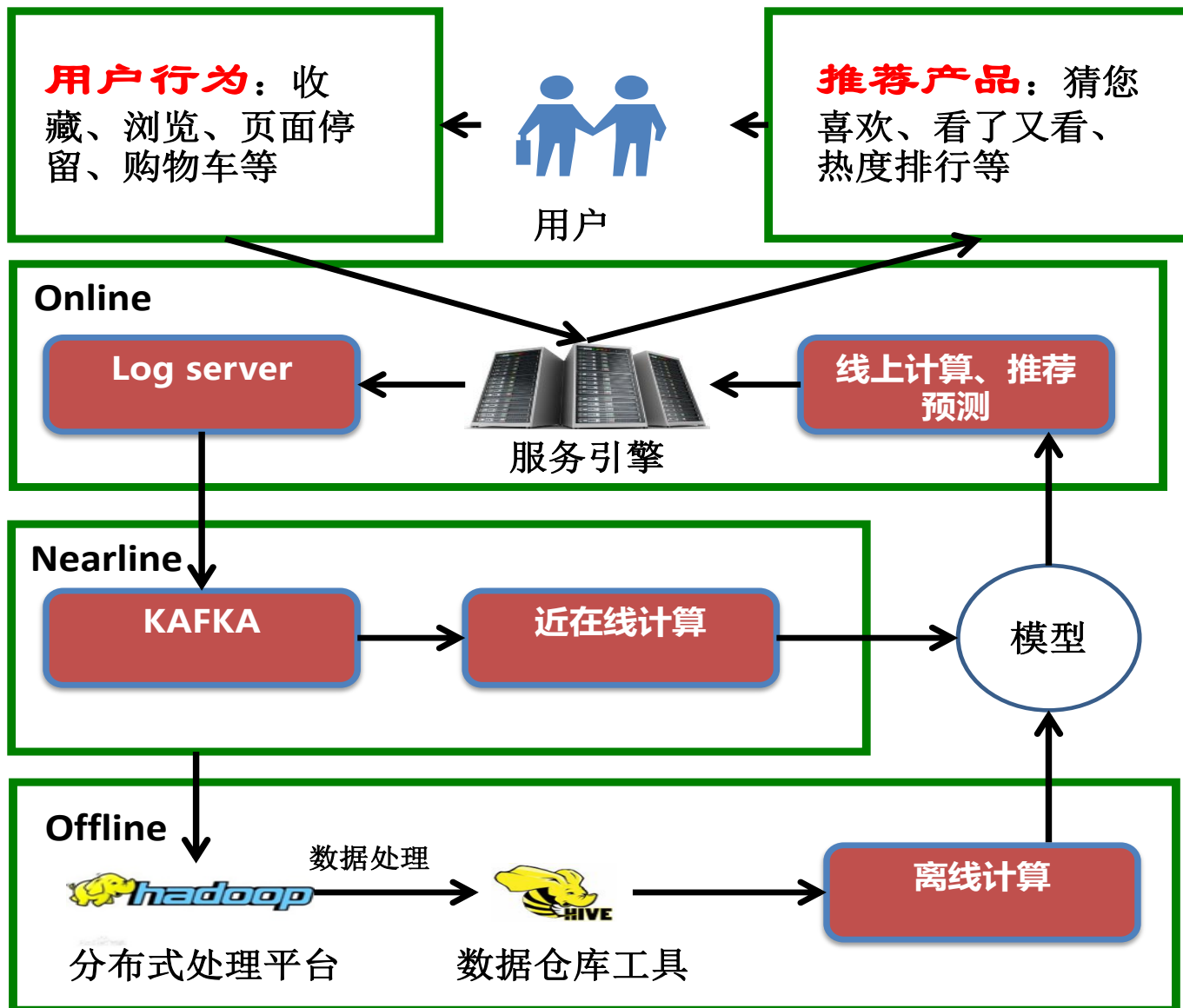
各种架构中图的一—ODS



IBM给出的数据仓库架构图



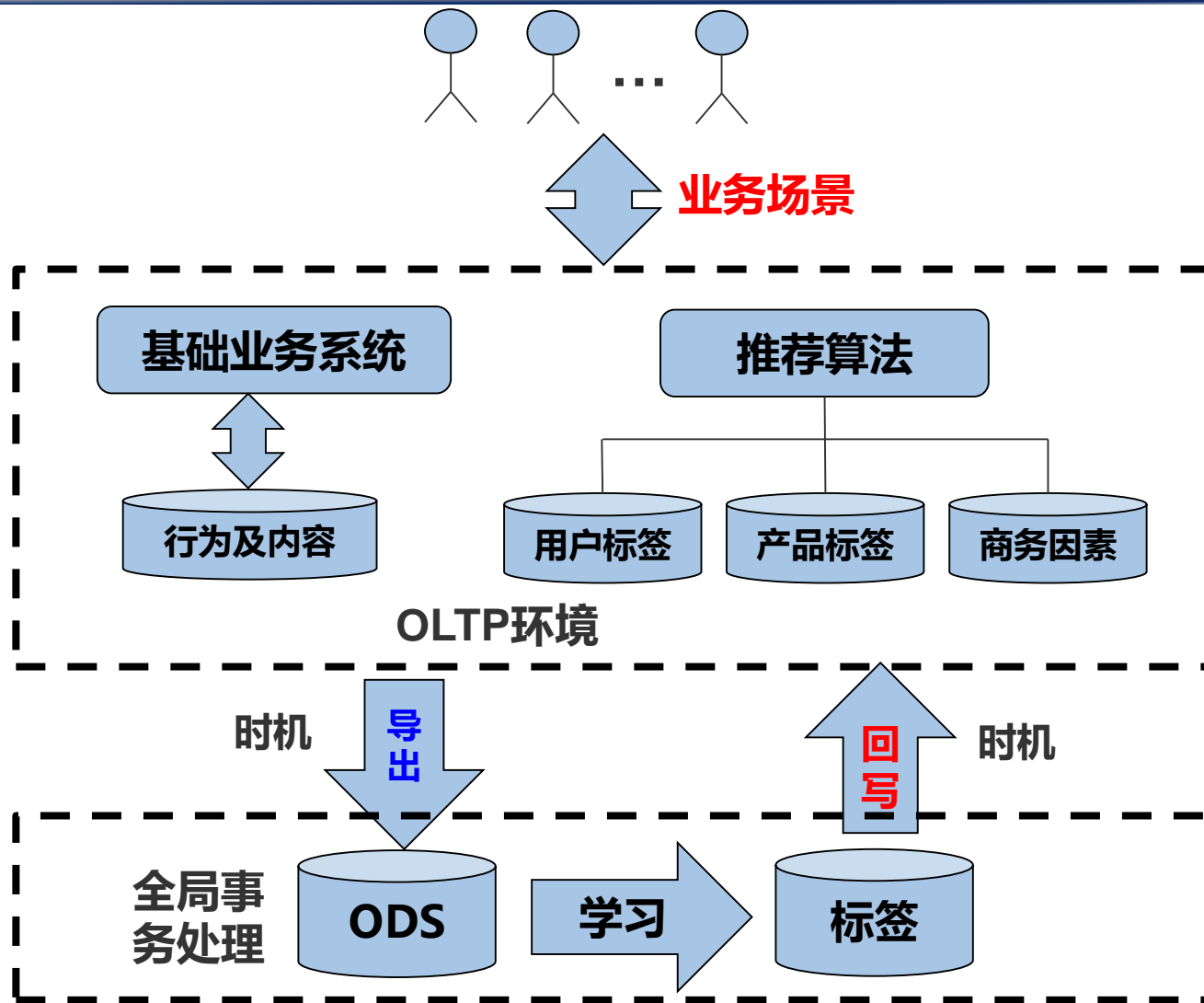
某企业中的三级业务计算或业务环境



近线计算环境的数据存储应该长怎么样？



经典全局事务处理架构





实例

▶ 决策环境

- 商场

▶ 决策人

- 销售部门的中下层管理人员

▶ 决策问题

- 某商品是否要进货？



决策所需掌握的情况

► 需要掌握以下信息

- 储备是否充足?
- 该商品近期销售情况如何?
- 资金情况如何?
- 其他商品的库存情况和销售情况如何?

► 即

- 要综合了解这些信息，才能做较为合理、可行的决策。



在何处实现这些功能？

► 方法1：放到分散的OLTP系统中去做

- 不一定能得到每个部门的准确的一致信息，需要进行部门间的协调配合，工作量会很大。

► 方法2：将其放在数据量巨大的DW/BigDB中去处理

- 显然会较费时，可能涉及许多不必要的数据检索。



这类决策的特点

► 特点

- 不是在线事务处理
 - 也算不上是高层决策分析。
 - 属于日常管理和控制的决策问题
 - 企业中层的管理者经常要解决的、较大量的问题。
- 前述两种解决方法都不太可行，该如何解决？

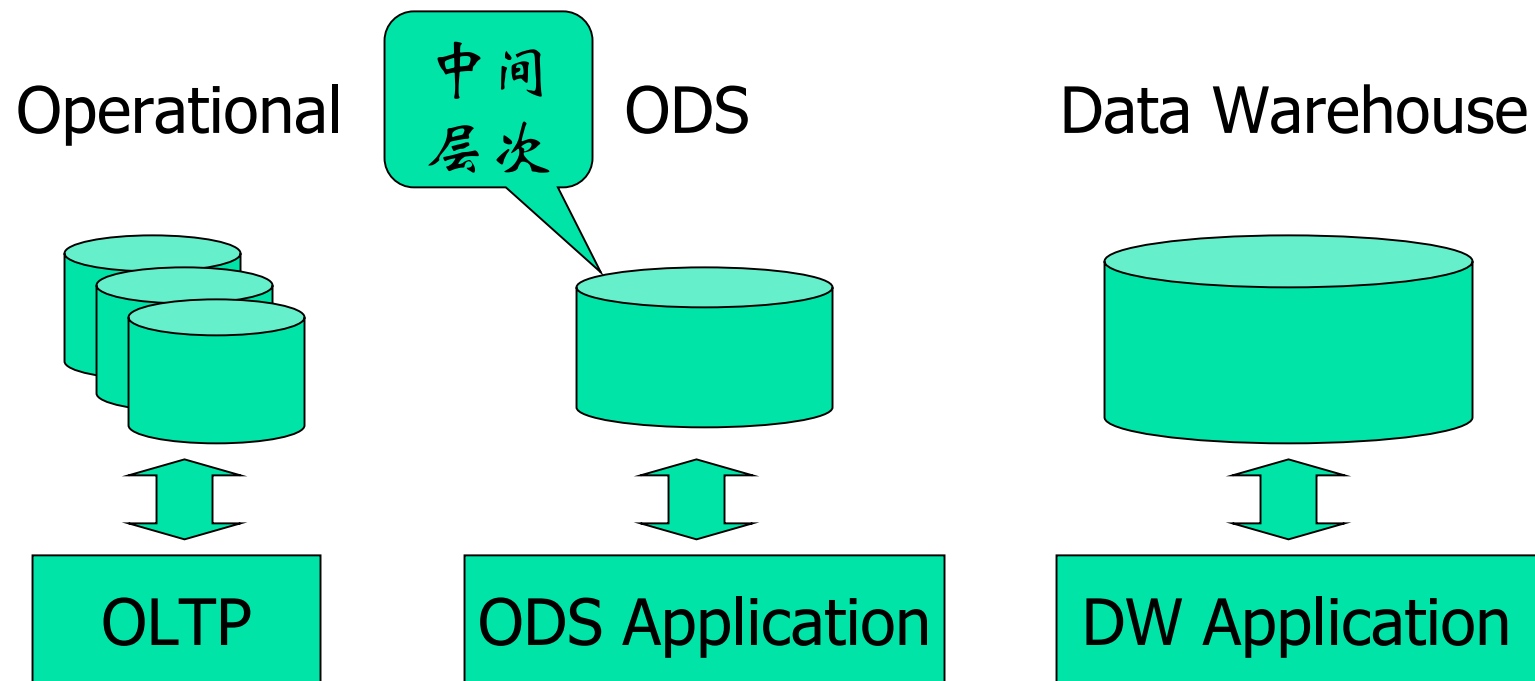


另一种数据环境

- ▶ **这种信息处理的特点引出了一种数据环境**
 - ODS, operational data store
 - 操作型数据存储
- ▶ **它是在OLTP-DW两层体系结构的基础上再增加一个层次ODS, 从而形成OLTP-ODS-DW的三层数据存储体系。**



ODS层次图示





ODS中的数据内容及特点

► 两类数据

- 一方面，它包含企业全局一致的、细节的、当前或接近当前的数据，可以进行全局在线操作型处理；
- 另一方面，它又是一种面向主题、集成的数据环境，且数据量较小，适合于辅助企业完成日常决策的数据分析处理。

► 与数据仓库或大数据平台相比

- 共性：面向主题、集成
- 不同点：时间跨度短、数据规模小、可能允许在线修改

► 与OLTP环境相比

- 不同点：面向主题组织、集成
- 共同点：可能允许在线修改



(2) ODS环境的定义

► 定义

- ODS是用于支持企业日常的全局应用的数据集合

► 保存在ODS中的数据具有四个基本特点

- 面向主题
- 集成的
- 可变的
- 数据是当前或接近当前的



关于ODS

- ▶ **因为数据面向主题，要求ODS中的数据在企业级上应该保持高度的一致性，所以必须对进入ODS数据进行转换和集成。**
- ▶ **区别于与分散在各个OLTP应用的数据：**
 - 面向主题、集成化。
- ▶ **区别与DW中的数据**
 - 存放当前数据或接近当前的数据
 - 可以进行在线修改



Four classes of ODS

- ▶ **From the beginning, it was predicted that there would be different classes of ODSs based on how fast data was loaded into the ODS and the source of the load.**
- ▶ **It was predicted that there would be four classes of ODSs:**



Class I. 实时ODS

- ▶ Transactions were moved to the ODS in an immediate manner from applications - in a range of one to two seconds from the moment the transaction was executed in the operational environment until the transaction arrived at the ODS.
- ▶ In this case, the end user could hardly tell the difference between an activity that had occurred in the operational environment and the same activity as it was transmitted in the ODS environment.

随着近实时集成方案的成熟，分钟级近实时ODS已经很常见



Class II. 小时级ODS

- ▶ **Activities that occurred in the operational environment were stored and forwarded to the ODS every four hours or so.**
- ▶ **In this case, there was a noticeable lag between the original execution of the transaction and the reflection of that transaction in the ODS environment.**
- ▶ **However, this class of ODS was much easier to build and to operate than a Class I ODS.**



Class III. 天级ODS

- ▶ **The time lag between execution in the operational environment and reflection in the ODS is overnight.**
- ▶ **In a Class III ODS there is a noticeable time lag between the execution of the transaction in the operational environment and the reflection of the transaction in the ODS environment. This type of ODS is relatively easy to build.**



Conclusion

- ▶ **As a rule, Class I ODSs are rare. The normal case for an ODS is a type II, III or IV.**
- ▶ **Class I ODSs are difficult to construct and even more difficult to operate. There needs to be a very strong business case for a Class I ODS.**



(3) ODS功能与实现机制

► ODS的产生根源

- 适应企业级的全局应用的需求而产生的

► 企业全局应用可以大致地划分为两类

- 进行企业级在线事务处理
- 即时OLAP数据处理



企业级OLTP与ODS

► 企业级OLTP

- 指在实际数据处理中，一个事务同时涉及多个部门的数据。

► OLTP数据环境的局部性

- 各应用所面对的仅是企业的某个部门，这些部门应用所涉及的仅仅是企业的局部数据。
- 为了获得快速的响应，每个面向应用的DB中不可能包含整个企业的完整数据。

► OLTP数据环境间的可能存在不一致性

- 某个操作型环境的数据组织很少考虑其它操作型环境的特点和需求，因而数据缺乏一致性。

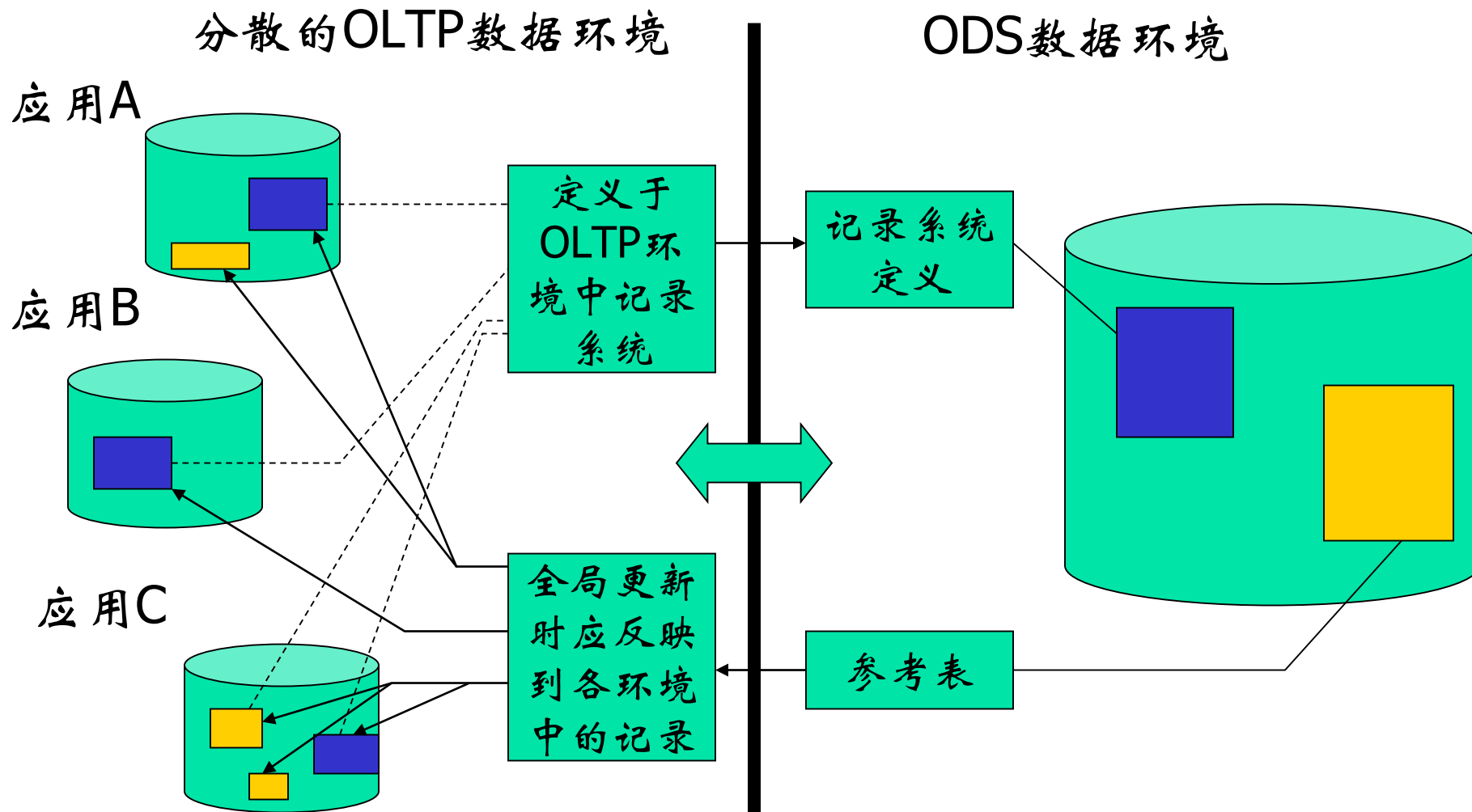


在ODS上建立企业级OLTP

- ▶ 所以，在各分散的操作型环境下要进行企业级事务处理代价会很大。原因是要进行数据集成。
- ▶ 而ODS中的数据已经是面向企业全局集成的，所以建立于ODS之上的OLTP可快速地实现对企业中数据的全局集中管理，因此，ODS的建立克服了原来面向的应用的数据库组织过于分散的特点。
- ▶ 为了实现企业级OLTP，需要建立ODS与OLTP数据环境间的双向映射关系
- ▶ 目的
 - 保持双方数据的一致性



服务于全局OLTP的双向映射关系





双向映射关系所表达的两方面

- ▶ 一方面，ODS记录系统定义表达的是ODS从操作型环境的数据抽取关系。在记录系统上所做的任何修改操作都需要反应在相应的ODS记录中；
- ▶ 另一方面，ODS系统中还存放着一些参考表，它所反应的关系是ODS全局更新时所必须反应到的所有DB中的相关记录的信息。



(2) 在ODS上实现即时OLAP

- ▶ **数据仓库所支持的OLAP的目的**
 - 进行长期的趋势分析。
- ▶ **一般数据仓库的数据量很大，所以OLAP的运行时间都比较长。**
- ▶ **对于企业的日常经营中常常需要进行的一些非战略性的中层决策来说，该如何实现呢？**



(2) Up-to-second OLAP

- ▶ **这类中层决策过程的特点**
 - **不需要参考太多的历史数据**
 - **主要是参考和存取当前的和接近当前的数据**
 - **并且要求有较快的响应速度**
- ▶ **可以把这类对数据的即时分析处理称为“即时OLAP”“up-to-second OLAP”。**



何处实现这种处理？

- ▶ **这种处理不适宜在DW上进行**
- ▶ **建立ODS的一个主要目的**
 - **支持这类处理**
- ▶ **由于ODS中的数据量远较DW小，因此可迅速获取决策信息，甚至达到秒级响应。**



ODS中的两种处理的区别

▶ 以上两种处理可以分别称为

- 企业OLTP：操作型
- 即时OLAP：信息型处理。

▶ 信息型处理

- 指只有查询操作的工作模式。

▶ 操作型处理

- 指含有更新操作的工作模式。

▶ 两种模式在数据处理上的差别导致了所需的技术支持有着很大差异。



技术差异

► 进行企业级OLTP时

- ODS是一个操作型的环境，此时ODS所要求的支持技术包括：事务管理、封锁管理、死锁检查、数据恢复、日志管理以及数据存储管理等复杂技术。

► 进行即时OLAP时

- ODS又是一个分析型环境，此时的数据处理要简单得多，实际所需的支持技术也少得多。



如何兼容这些差异?

► 方法1:

- 使ODS系统能在不同时间工作于操作型环境和分析型环境两种状态，需要加入切换机制

► 方法2:

- 采用多个服务器，建立多个数据备份，使某些数据库服务器始终处于分析型环境，采用复制技术使用其环境与其他服务器保持一致，同时保证分析处理的性能。



(4) ODS vs DW

- 面向主题和集成性使得ODS 的数据在静态特征上很接近DW中的数据。但是，在ODS与DW之间在如下方面仍然存在基本而重要的区别。
- 内容，数量级，技术支持，面向的需求，使用者

ODS	DW
当前或接近当前的数据 细节数据 可在线更新	历史数据 细节数据和综合数据 不可变快照或不能在线更新

ODS与DW中存放数据内容的差别



数据内容上的区别

► 时间跨度上的区别

- ODS仅存放当前或近期内生成的数据
- DW中大量是长期保存并可重复查询的历史数据。

ODS	DW
当前或接近当前的数据 细节数据 可在线更新	历史数据 细节数据和综合数据 不可变快照或不能在线更新

ODS与DW中存放数据内容的差别



数据粒度及处理

► ODS

- 主要保存的是细节数据
- 也可以获取一定的综合数据，如在进行即时OLAP时，也要经常生成一些统计数据等，但这些综合数据只有在需要时才生成，并且由于ODS数据是随时可更新，这些综合数据只有在生成时是准确，因而在大部分情况下并不长期保存起来。如果所生成的综合数据无需更新，且将被重复访问，那么ODS也可以将这部分综合数据长期保存起来。

► DW

- 不仅保存细节数据而且也保存综合数据。
- 由于数据不常更新，所以可以保存相当数量的各级综合数据以备重复访问，而不必为维护这些综合数据付出多少代价。

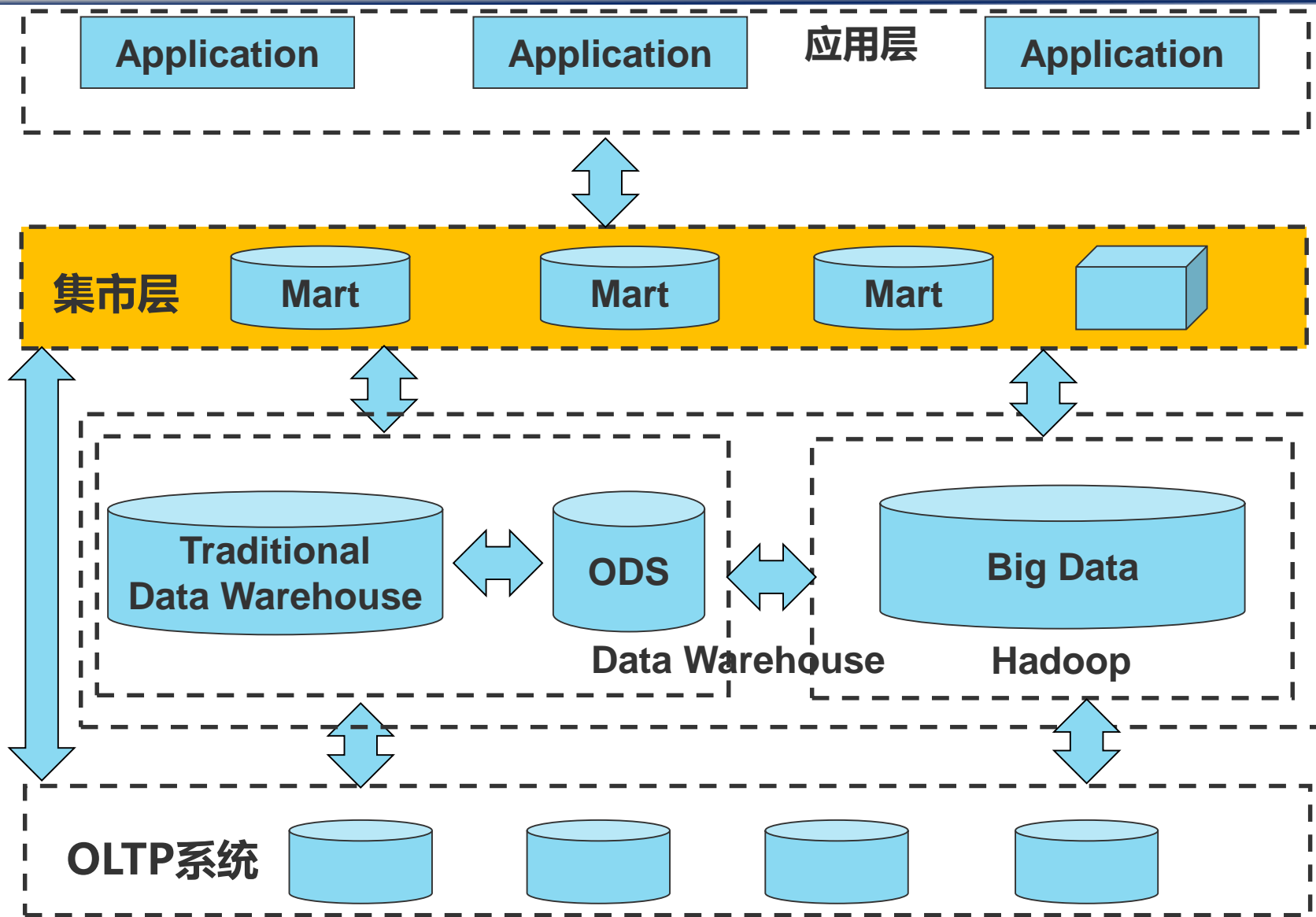


ODS与DW数据量级不同

- ▶ **DW的数据量要远远超出ODS的数据量**
 - **DW保存大量的历史数据**
 - **ODS只存放当前和接近当前的细节数据**
 - **ODS为获得较高的系统性能，一部分固定数据信息不包含在内。**
 - **DW则几乎无所不包，从细节数据到程度各异的综合数据，从当前数据到历史档案数据，有时还涉及外部数据。**



架构中的数据集市





2. 数据集市环境

► Data Mart

- The access layer of the data warehouse environment that is used to get data out to the users
- A subset of the data warehouse and is usually oriented to a specific business line or team.

► 任何一个平台的建设都有其初衷

► 为什么需要数据集市环境？

► 相类似的问题

- 为什么要分家？为什么要成立子公司？为什么要设立独立部门？

► 解耦合需求



为什么要有数据集市

- ▶ 问题其实等价于为什么要有独立的平台?
- ▶ 核心原因就是独立性
 - 计算的独立性需求
 - 相对独立地拥有数据
 - 相对独立的决策支持应用需求
 - 相对独立的环境
 - 硬件环境和技术环境
 - 相对独立的管理



两类数据集市

▶ 无公共大平台依赖的**独立的数据集市**

- 不同的数据集市之间没有公平的平台，相对独立建设与成长，数据集市之间面向的决策支持需求而建设
- 对企业来说，每个这种的数据集市可以看成是一个子数据仓库或大数据平台

▶ 有公共大平台的数据集市

- 数据集市建立在公共的数据仓库或大数据平台之上
- 每个集市服务于不同的决策支持需求集合



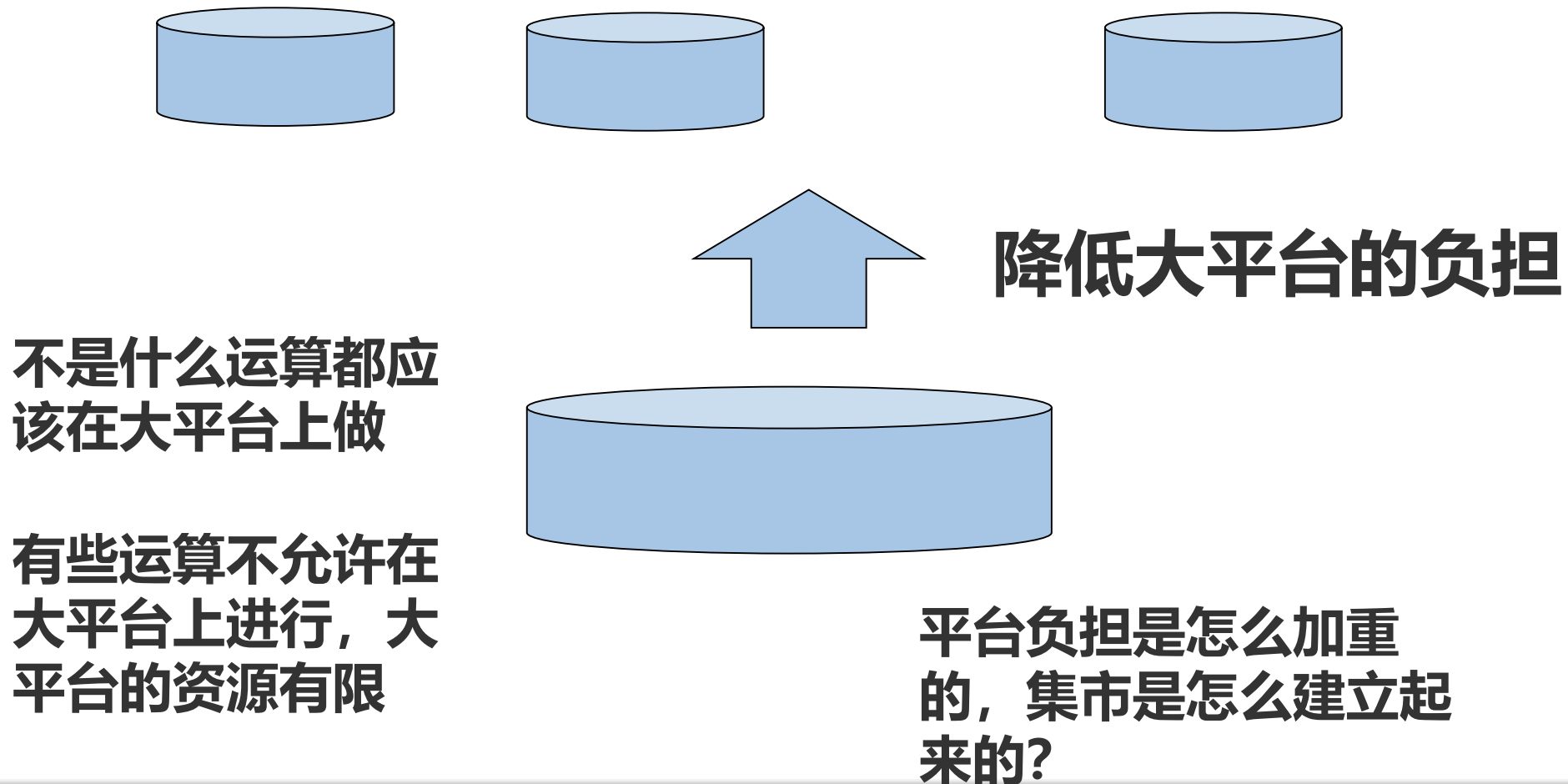
决策支持需求的独立性

- ▶ 数据集市支持的常见需求类别
- ▶ 保存特定部门或主题的中高粒度级数据服务于基础信息处理
 - 对外数据服务、固定统计报表、部分领域或部门的独立统计数据处理平台
- ▶ 特定应用模式的独立支持平台
 - 特定主题的多维分析平台，OLAP分析工具的集市级数据环境
- ▶ 特定离线计算分析环境
 - 大数据量的离线分析计算、科学实验等
 - 大平台上一般不允许做复杂的运算



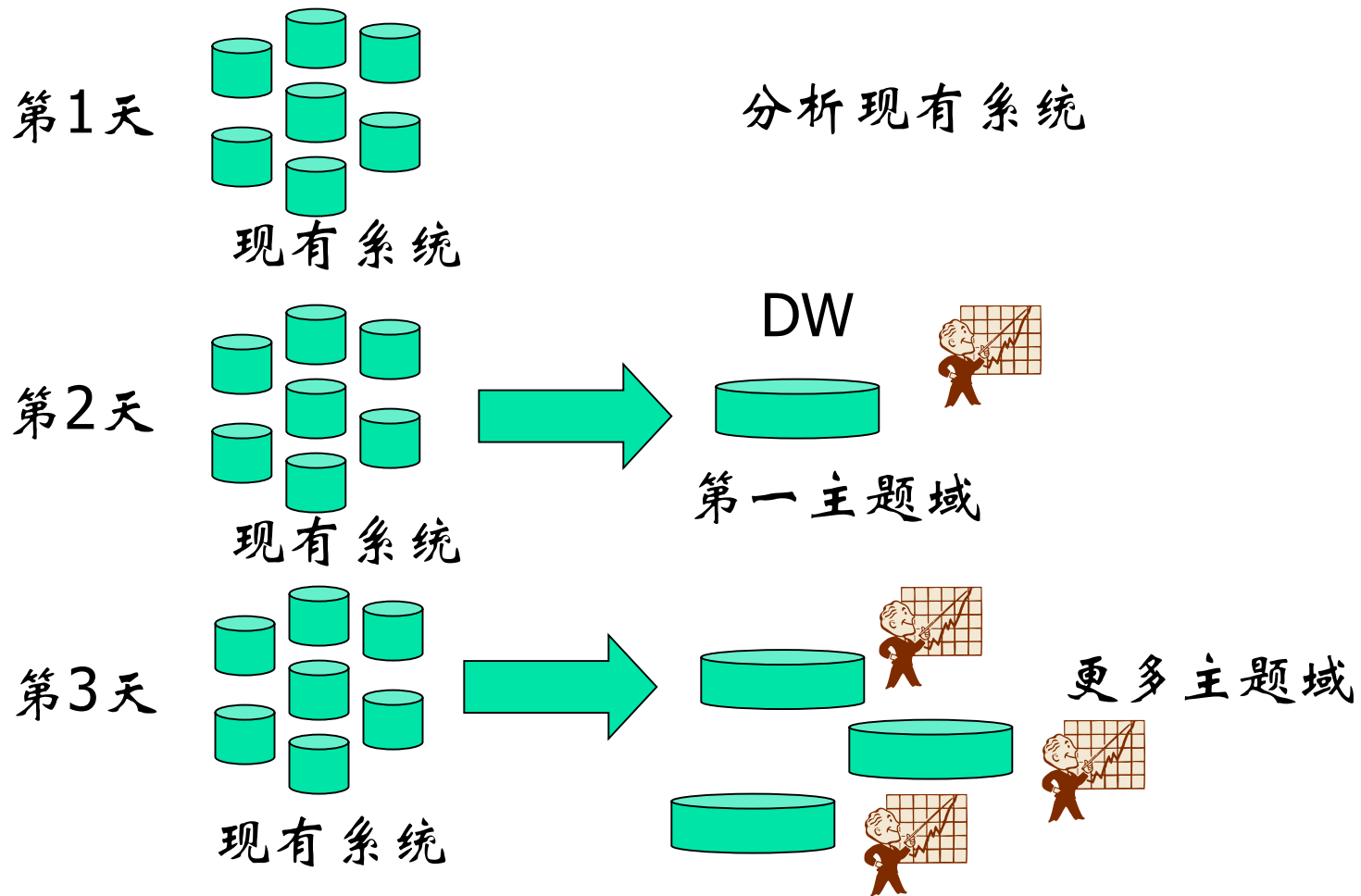
计算独立性所带来的数据集市需求

整个平台架构中计算的分层与分布问题





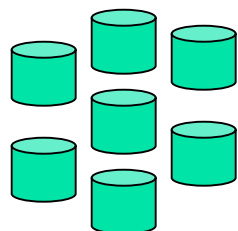
数据仓库的建立过程1—n天



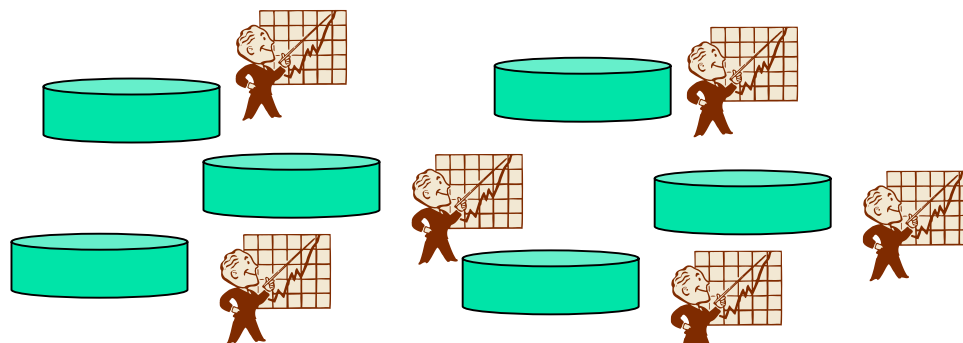
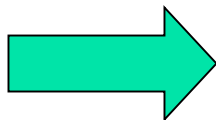


第4,5天，应用在逐步扩展

第4天

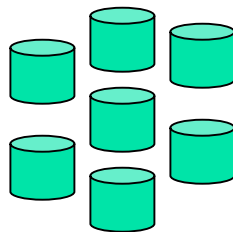


现有系统

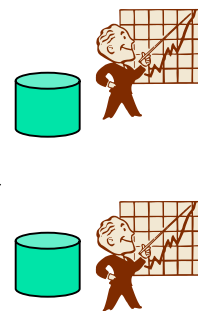
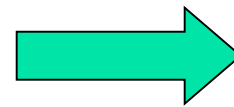
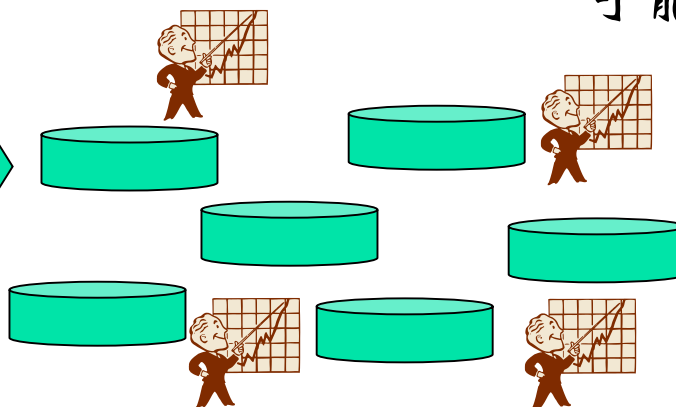
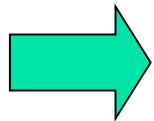


可能会带来访问问题

第5天



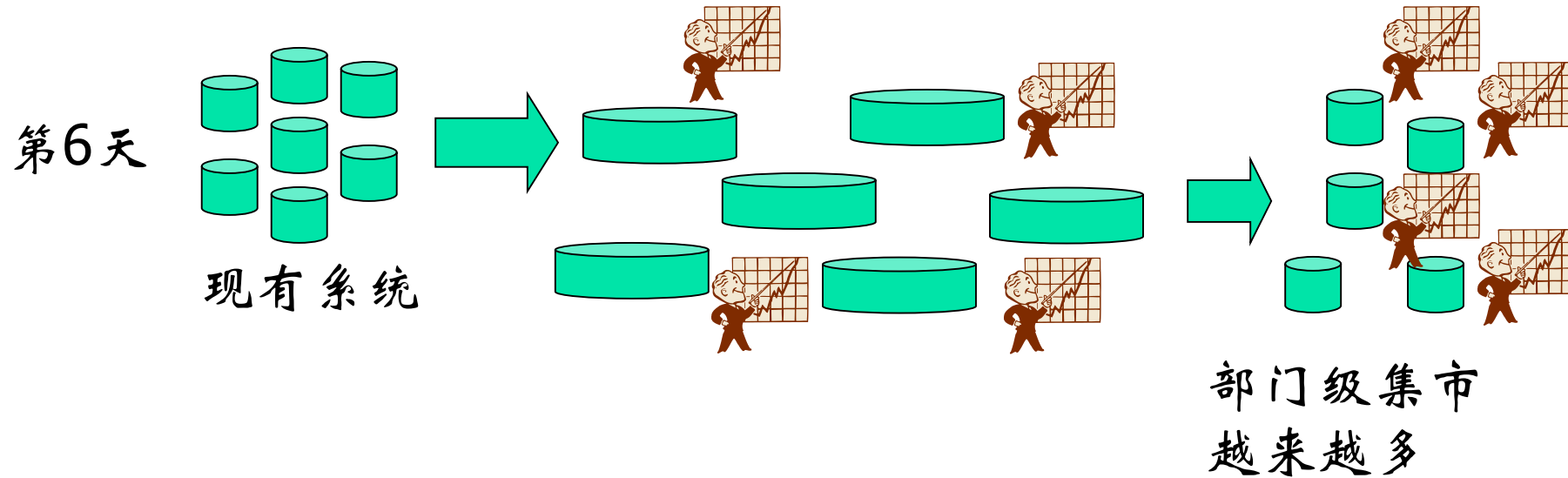
现有系统



部门级集市

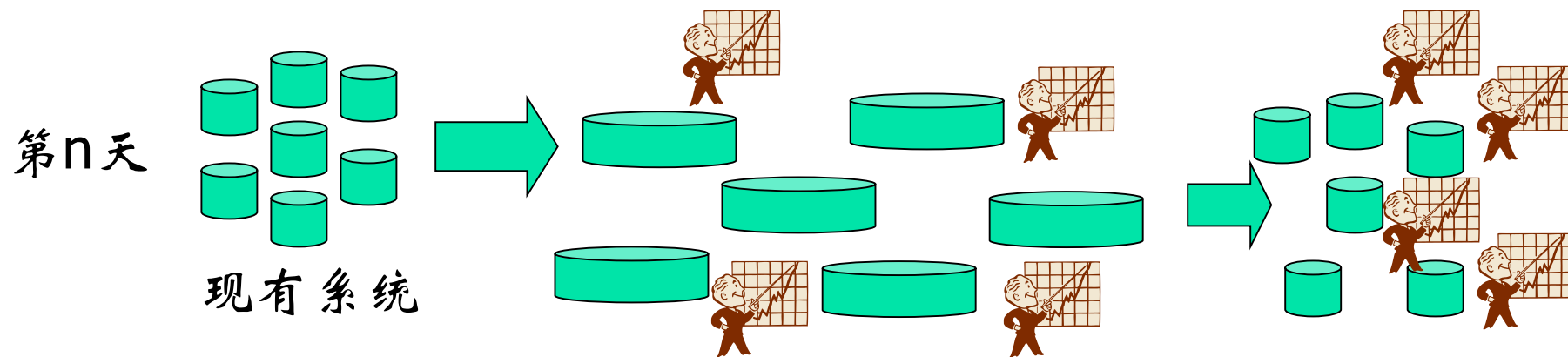


第6天，部门集市越来越多



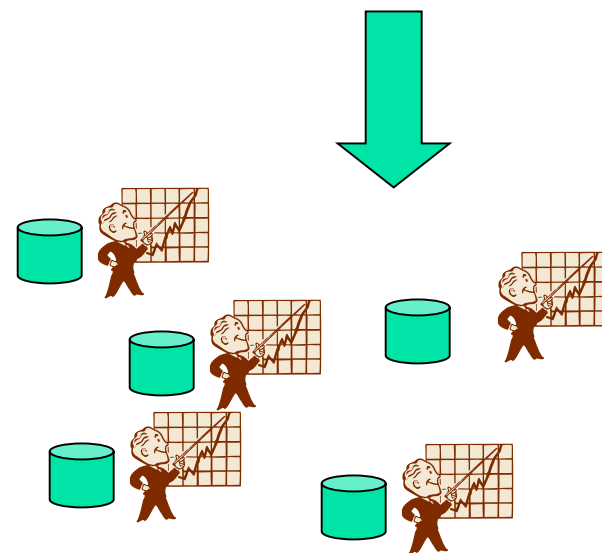


第n天，出现个体层应用



整个过程可能需要几年，需要完成

- (1) 现有系统数据结构分析
- (2) 数据仓库模型设计
- (3) ETL
- (4) 报表设计
- (5) 挖掘分析
- (6) 决策支持系统





内容提纲

数据仓库与大数据主要特征

数据仓库与大数据平台总体数据架构

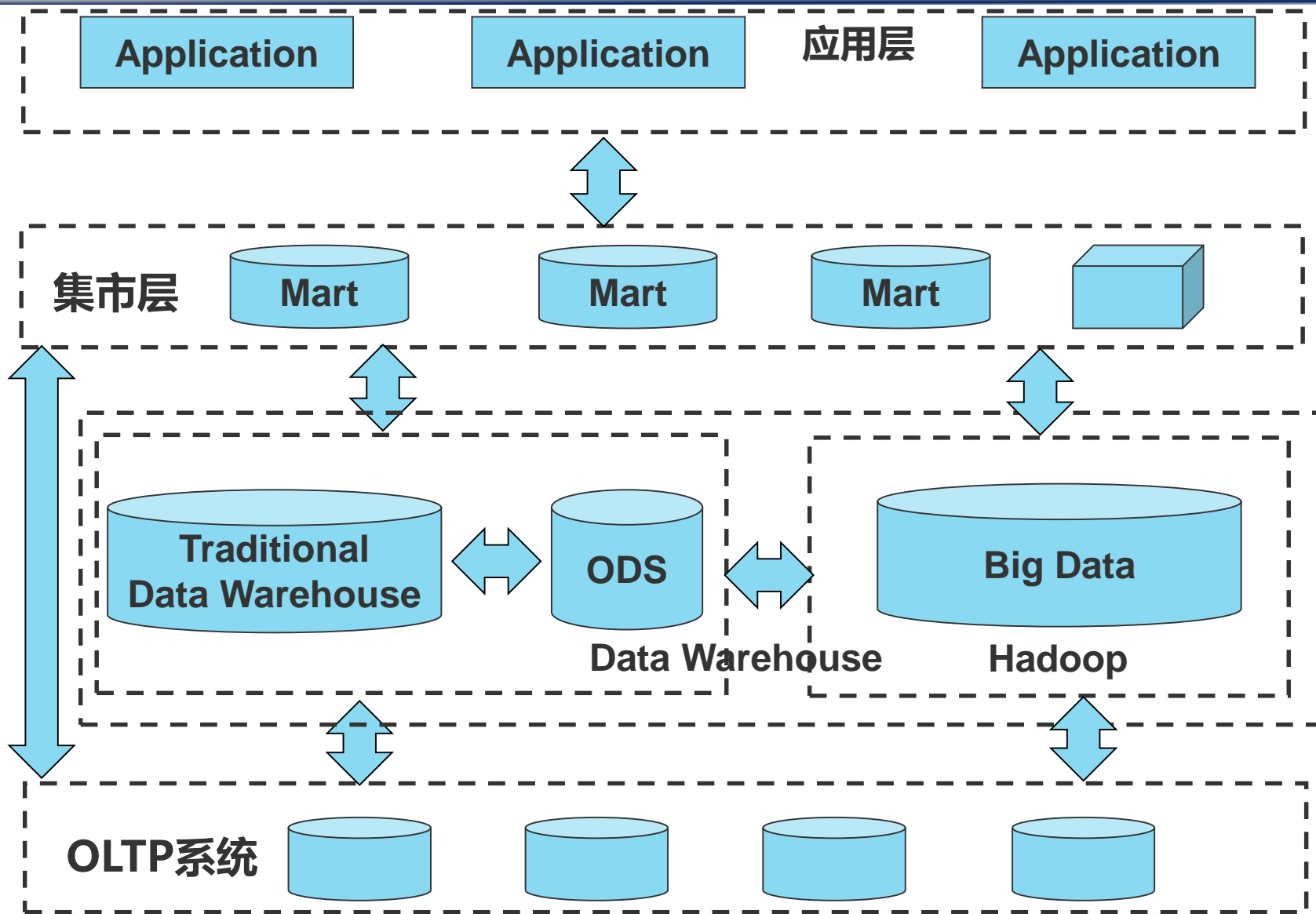
数据仓库与大数据平台数据组织结构

数据环境组成与应用支撑

软硬件技术环境



整个架构需要哪些软硬件





1. 硬件环境

- ▶ 网络环境
- ▶ 数据集成环境
- ▶ 计算环境
 - 高性能计算
 - 集群计算
 - 传统集群、Hadoop、...
- ▶ 存储环境
 - 集中存储、存储网络、分布式存储
- ▶ 数据服务环境
- ▶ 管理、监控与安全环境



(1) 网络环境

► 网络环境独立性要求

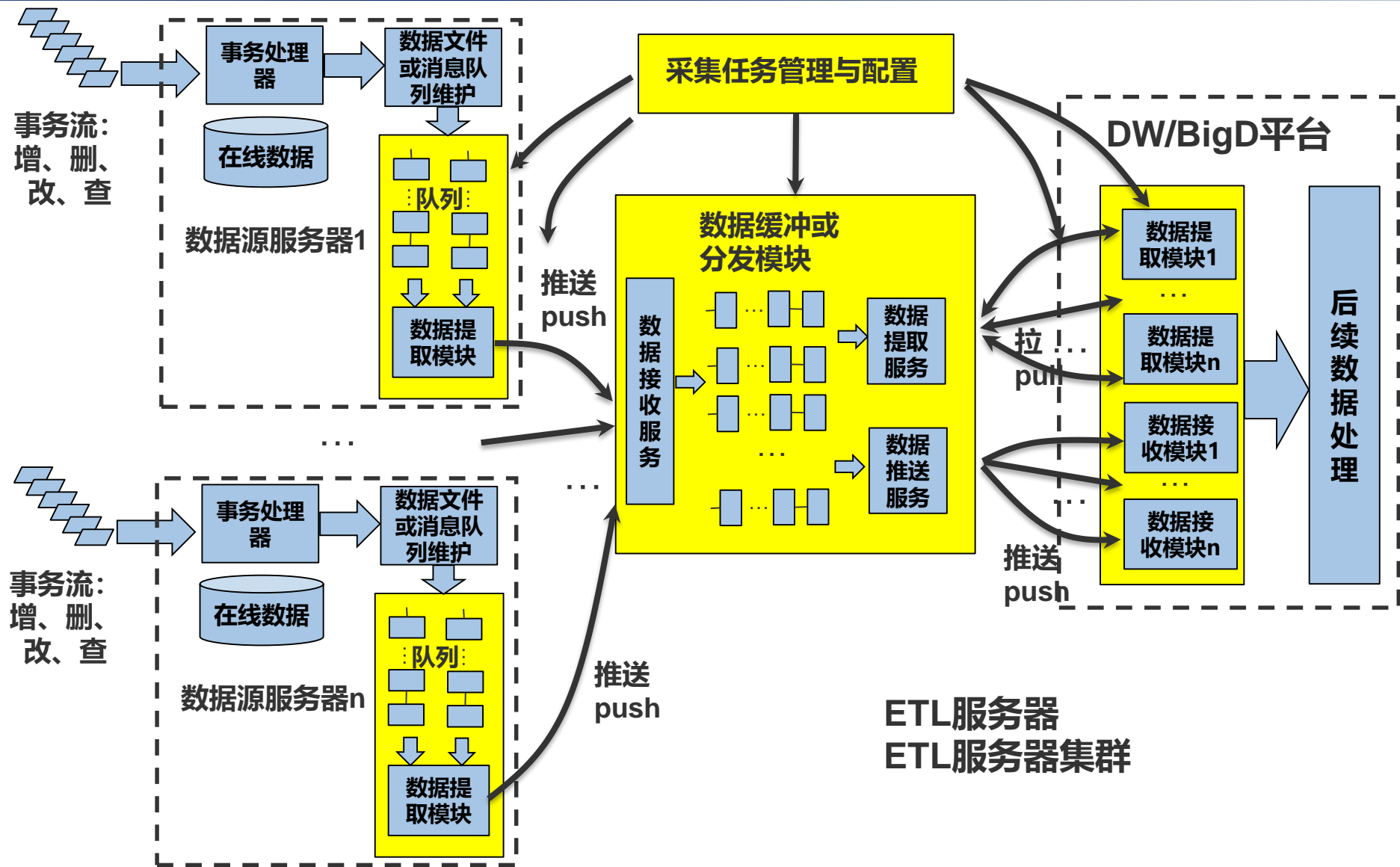
- OLTP环境与数据仓库或大数据平台的网络支撑环境一般应相对独立
- 原因：保证两类环境的性能要求

► 网络环境组成

- 接口网络环境
- 核心处理平台环境
 - 服务器集群子网环境
 - 存储网络环境
- 对外服务网络环境



(2)数据集成硬件环境





(3)计算所需环境

► 分布式存储并行计算集群

- 虚拟机或物理机集群环境，支撑Hadoop、Cassandra等计算与存储集群

► GPU服务集群

► 大内存高性能计算集群

► 常规传统服务器

- RDBMS数据库服务器
- 各类应用服务器
 - 基本计算、多维分析、数据挖掘、机器学习算法服务器

► ODS计算与服务环境



(4) 存储环境构成

- ▶ 服务器直连存储环境
 - 单机存储
 - 单机直连磁盘阵列—硬件阵列、闪盘阵列
 - HADOOP集群
- ▶ 存储区域网—SAN, Storage Area Network
- ▶ 网络存储环境—NAS, Network Attached Storage
- ▶ 冷存储环境
 - 光盘存储体系、磁盘体系
- ▶ 大内存存储环境—针对内存数据库而言
- ▶ 数据备份环境
 - 灾备存储、备份服务器



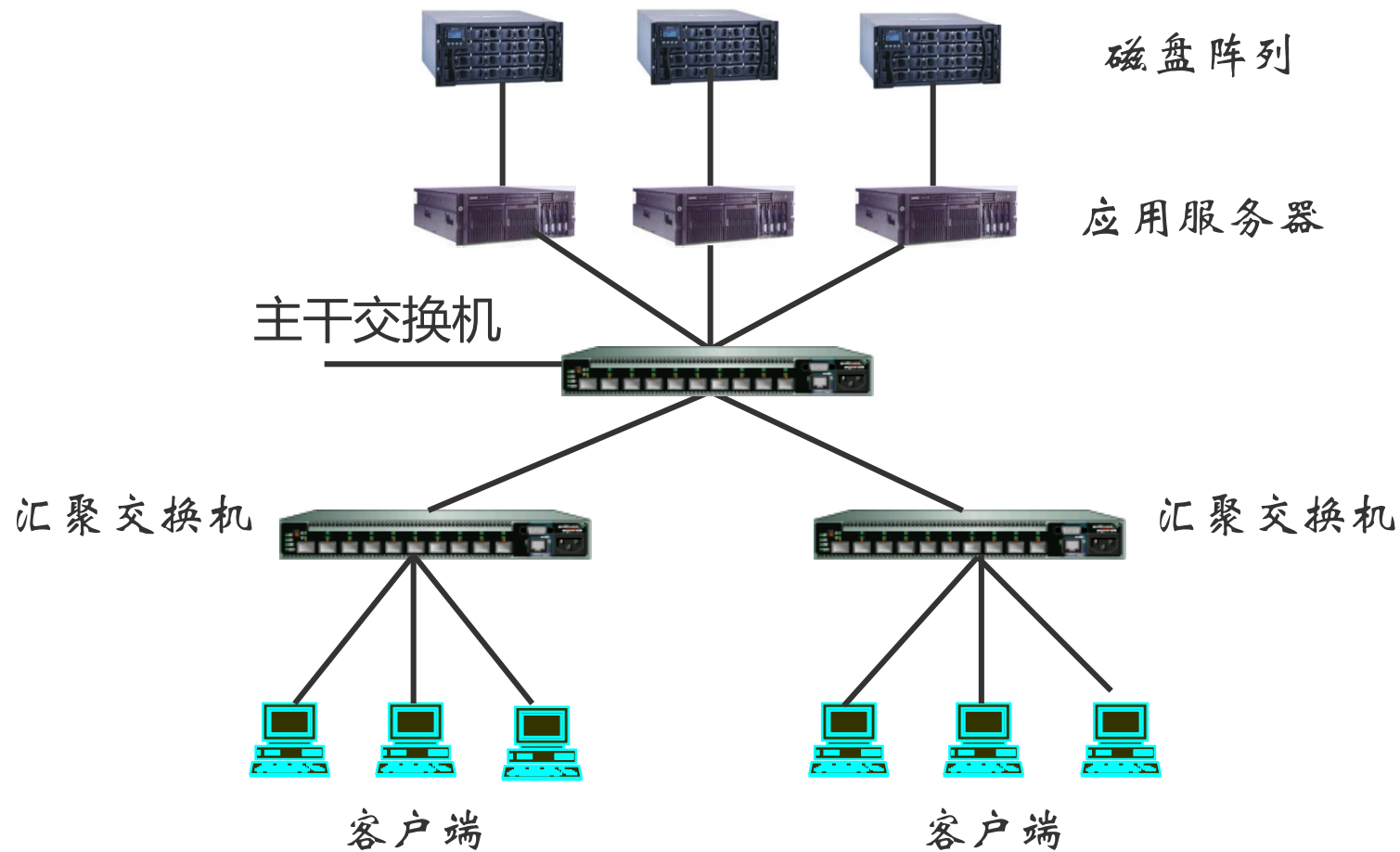
当前常见存储体系简介

► 常见存储体系

- **DAS, Direct Attached Storage**
- **NAS, Network Attached Storage**
- **SAN, Storage Area Networks**

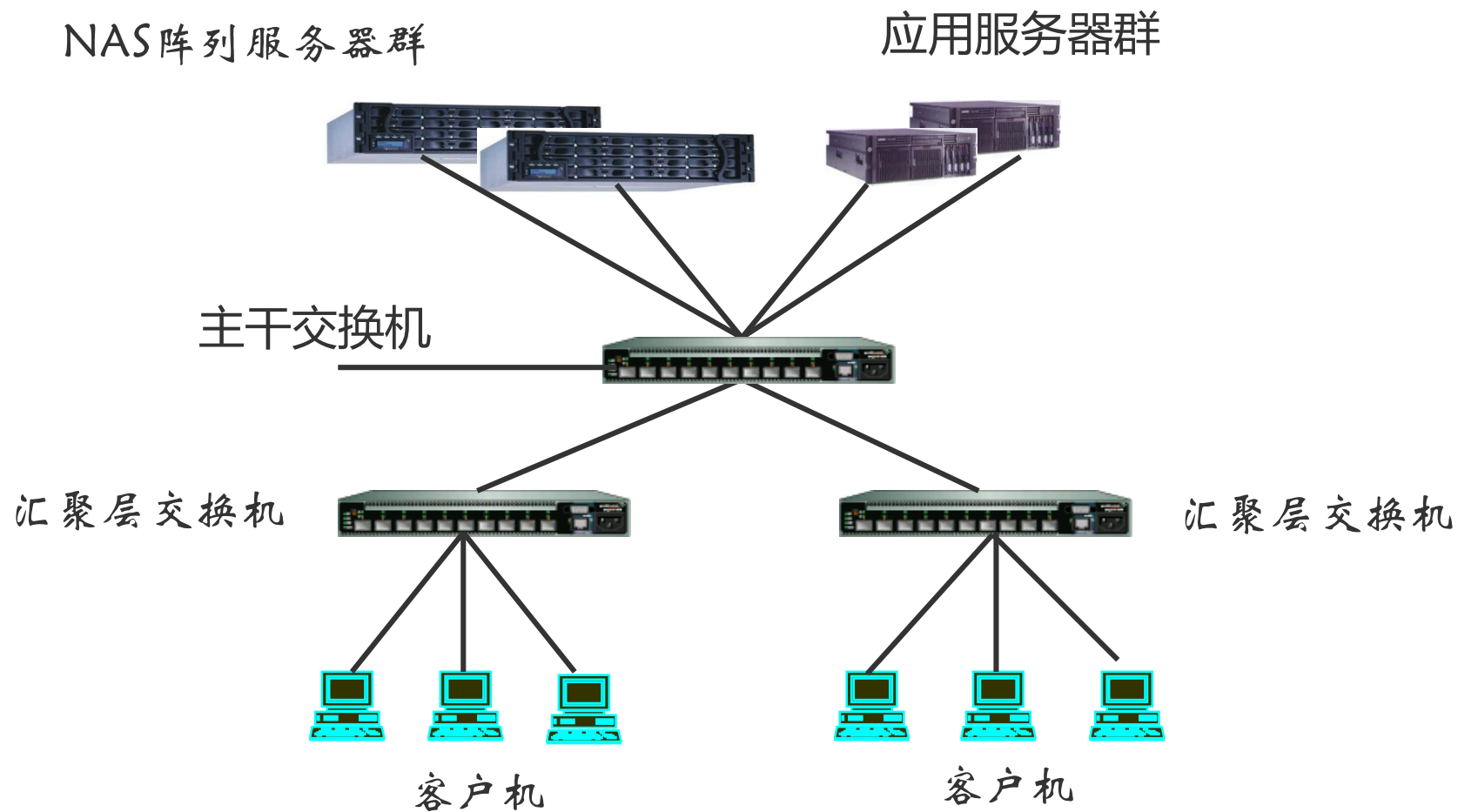


DAS存储体系结构实例图



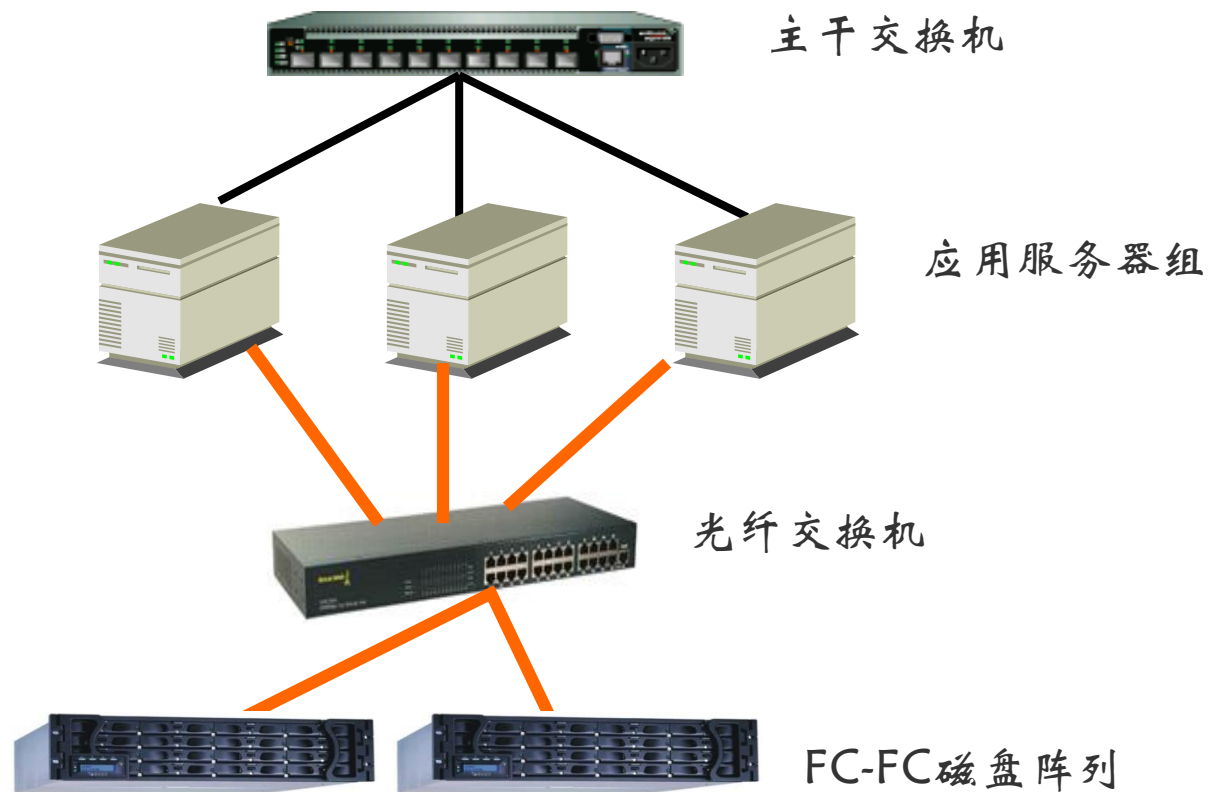


NAS存储体系结构实例图





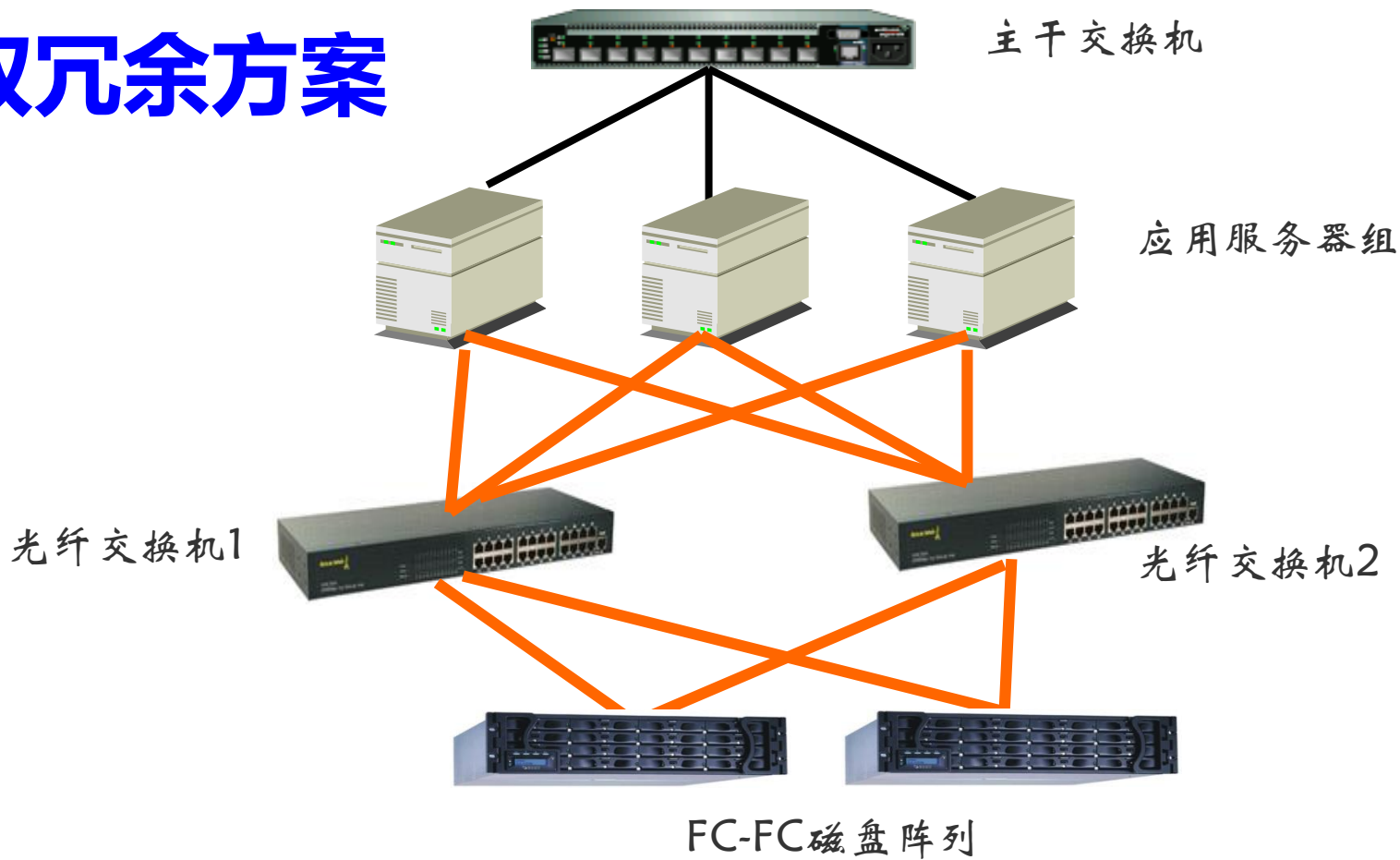
SAN存储体系结构实例图一





SAN存储体系结构实例图二

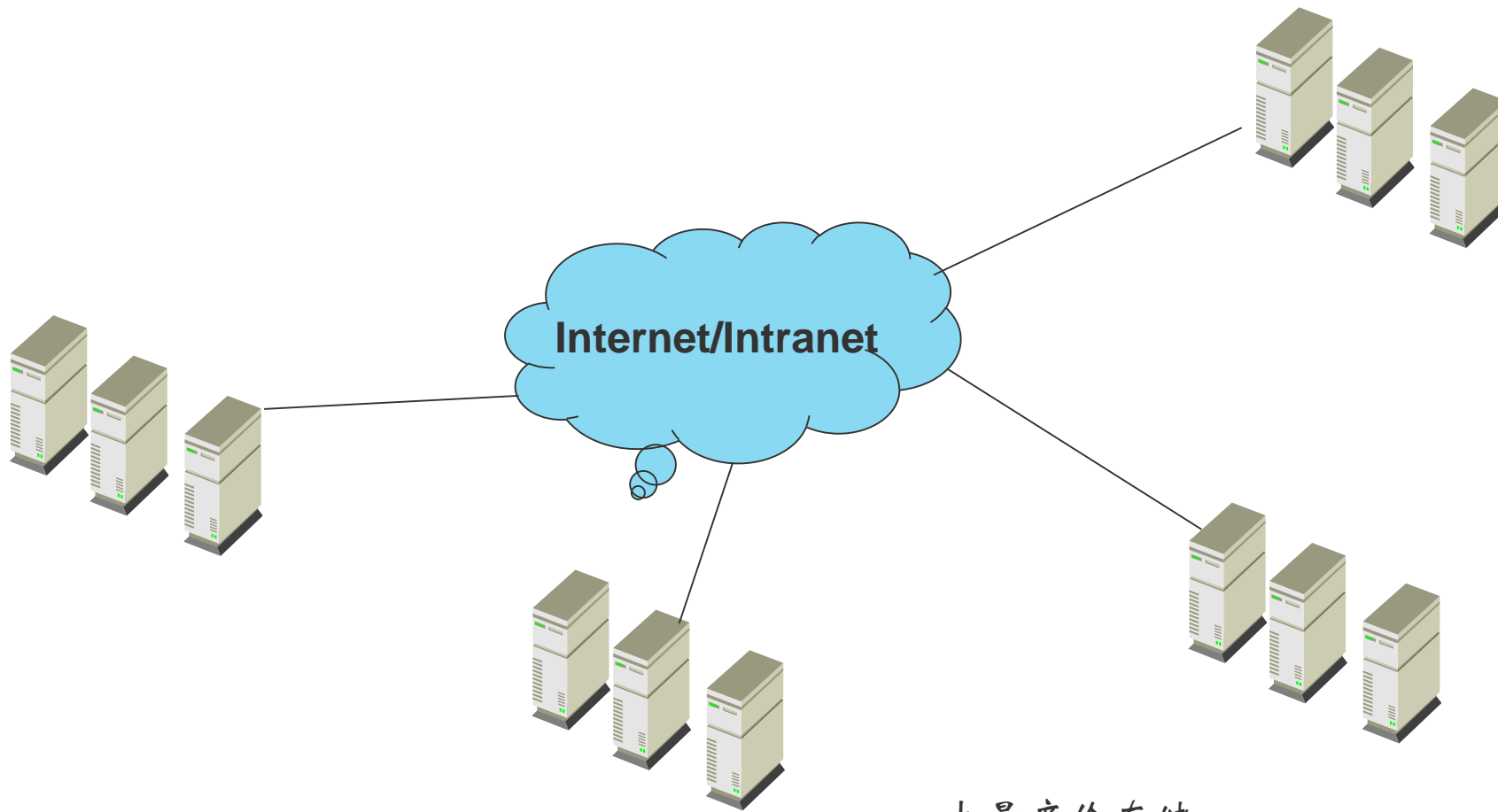
双冗余方案





新一代分布式存储与并行计算架构

► 分布式存储架构



大量廉价存储
与服务设备



(5) 数据服务硬件环境

- ▶ **WEB服务环境**
 - ▶ **WEB数据服务集群**
- ▶ **直连内容服务环境**
 - ▶ **数据打包服务**
 - ▶ **数据转发服务**
 - ▶ **FTP服务**
 - ▶ **即时请示响应式数据服务**



(6) 管理、监控与安全环境

- ▶ **元数据服务器**
- ▶ **管理服务器**
 - 用户认证、安全控制
- ▶ **系统运行监控服务器环境**
- ▶ **安全环境**
 - 防火墙、入侵检测



(7) 开发与测试环境

- ▶ 样本数据环境
- ▶ 仿真计算与模型实验环境
 - 各类服务器
 - 数据挖掘、机器学习算法服务器、GPU服务器
 - 代码服务器
 - 采集服务器
 - 数据源仿真
 - 集群
 - 网络
 - ...



2. 软件环境

► 系统平台支撑软件

- 云平台工具、虚拟化工具、操作系统

► 数据存储平台软件

- RDBMS、大数据平台、内存数据库

► 各类中间件

- 消息中间件、OLAP中间件、WEB服务中间件、元数据管理、计划任务管理、...



2. 软件环境

► 数据流相关软件环境

- 数据采集、数据集成、数据更新
- 中间件、平台组件、定制开发

► 计算架构支撑平台

- 并行与分布式计算架构软件
- 流式计算软件架构
- 深度学习计算架构

► 用户算法与应用系统

► ...



本部分小结

- ▶ **要开展设计，必须理解存在哪些设计对象，本部分主要叙述数据仓库与大数据环境的构成，了解平台中的组成元素，理解组成成分间的关系**

本部分结束!



BEIJING JIAOTONG UNIVERSITY