本文主要内容来自 SpriCoder的博客 换了更清晰的图片并对原文的疏漏做了补充和修正。

本文提供了 pdf 版，以供打印： 商务智能-04-数据仓库的设计 | EagleBear2002 的博客 。

1. 数据仓库设计的由来

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，这两者在原理上是一致的。

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，两者在设计中的很多方面也存在着差别

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

创建数据仓库的重要有两部分：

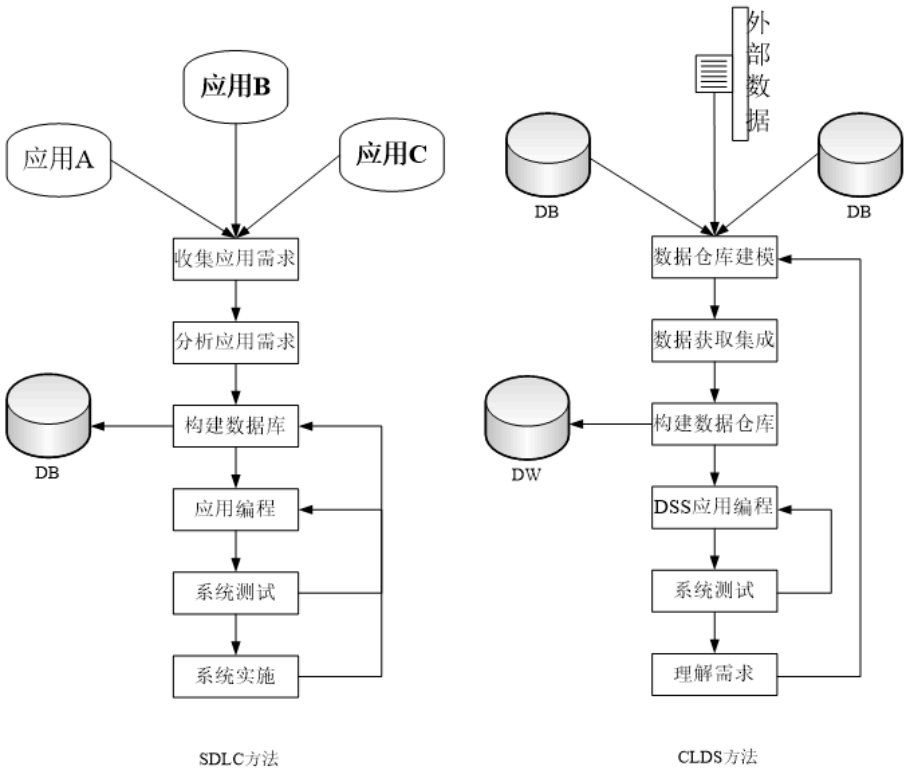
1. 与操作型系统接口的设计（ETL）
2. 数据仓库本身的设计

数据仓库的需求只有在已装载部分数据并已开始使用时才能完全确定清晰，因此往往无法使用传统的需求驱动的方法进行。

2. 操作型数据库 vs. 数据仓库

方面	操作型数据库	数据仓库
面向的处理类型	面向应用	面向分析
面向的需求	确定的应用需求	不确定的分析需求
系统设计目标	事务处理性能	全局一致的数据环境
数据来源或系统的输入	事务相关数据	多种多样
系统设计的方法和步骤	SDLC（系统生命周期）	CLDS

SDLC 和 CLDS



1. SDLC（Software Development Lifecycle Model）是在最开始做需求分析，CDLS 可以在理解需求的部分进行需求分析。
2. 变化来源：数据仓库由于外部环境在变更则一定需要变更
    1. 客观变化：外部环境等客观变化
    2. 主观变化：每天的想法不一样
3. 上一个分析人员留下的部分不可能是很客观的

3. 数据仓库设计的原则

数据仓库是面向主题的、集成的、非易失、时变的，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，其设计过程必须遵循下述三条原则：

1. 面向主题原则
2. 数据驱动原则
3. 原型法设计原则

3.1 设计原则 1：面向主题原则

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：

1. 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
2. 在数据库设计中则是以客体为起始点，即以客观操作需求为设计依据

例如：“商品销售”主题：

1. 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，以提高商品的销售总金额
2. 需要执行的分析操作：
    1. 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
    2. 管理人员将据此决定他们的经营策略

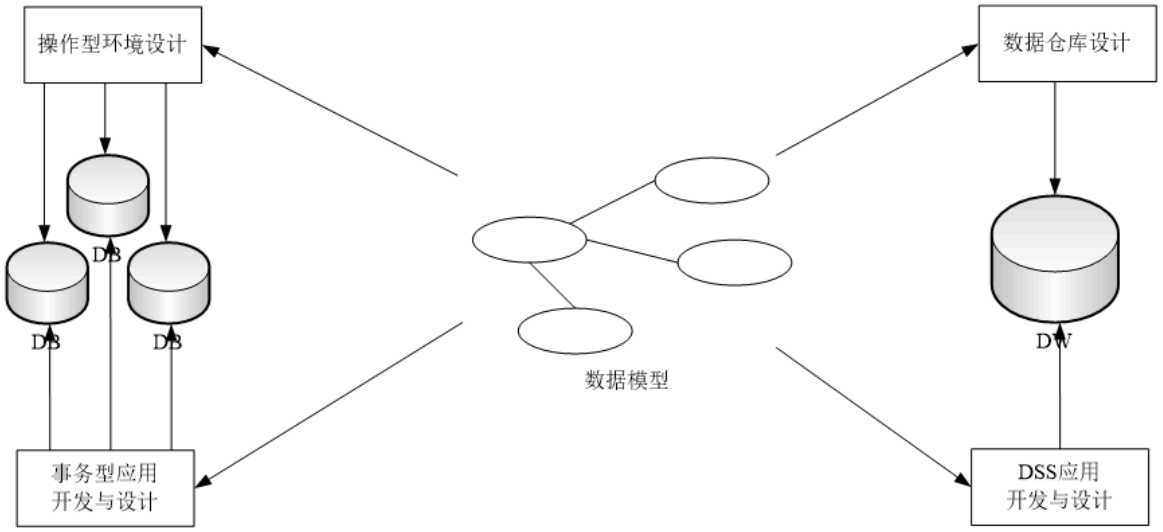
3.2 设计原则 2：数据驱动原则

数据的来源：

1. 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
2. 因此，数据仓库中的数据必须是从已有的数据源中抽取而来，是已经存在的数据或对已经存在的数据进行加工处理而获得”

在数据仓库设计中，由于其所有数据均应建立在已有的数据库基础上，即是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：

1. 从已有的数据库系统出发
2. 按照分析领域对数据及数据间的联系重新考察
3. 利用数据模型有效识别原有数据库中的数据和数据仓库中主题的数据的“共同性”



1. 数据仓库不能凭空产生
2. 首先构造操作型数据环境，然后构造分析型数据环境
3. 实际情况下使用这样的数据驱动模型是比较困难的
4. 考虑是否要先创建数据模型

3.3 设计原则 3：原型法设计原则

不断变化的分析需求：

1. 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
2. 因此，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

逐步求精的设计过程：

1. 数据仓库用户的需求是在设计过程中不断细化明确的。同时，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
2. 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

启发方式：

1. 一个迭代的开发依赖于在上一个迭代所获得的结果
2. 首先载入一部分数据供 DSS 分析员使用和查看
3. 然后根据最终用户的反馈，修改数据或添加其他数据。这种反馈过程贯穿于数据仓库的整个开发生命周期之中

需求预测仍然是十分重要的。实际情况通常介于启发和预测两者之间。

4. 从操作型数据开始

1. 将数据从操作型环境移入到数据仓库环境不是简单的抽取

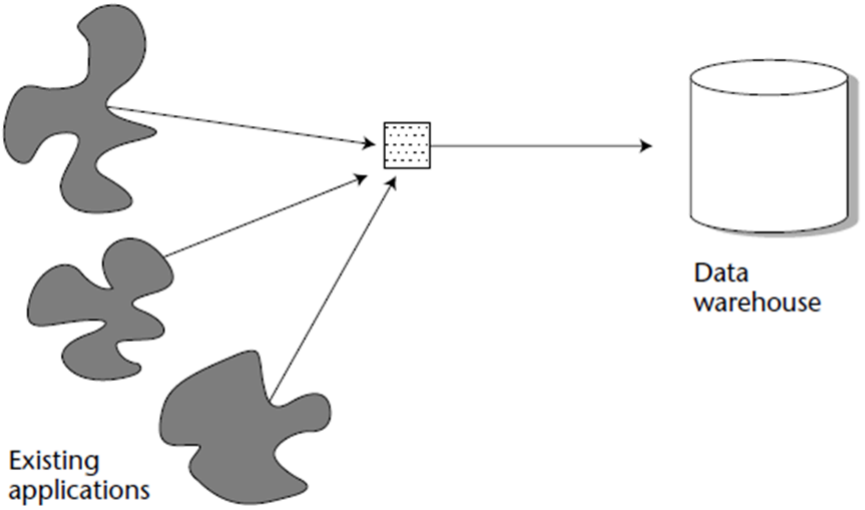


Figure 3-1 Moving from the operational to the data warehouse environment is not as simple as mere extraction.

通常现有的操作型系统中的数据缺乏集成，建立现有的操作型应用时，根本没有考虑过以后可能存在的集成问题。每一个应用都拥有自己特殊的需求，可能会出现下列情况：

1. 不同名同物：一些相同的数据以不同的名字出现在各个地方
2. 同名不同物：一些不同的数据在不同的地方以相同的方式标注
3. 数据不完整：这里的数据不在其他地方出现
4. 度量单位不同：一些数据用相同的名字存在不同的地方，却使用不同的度量单位

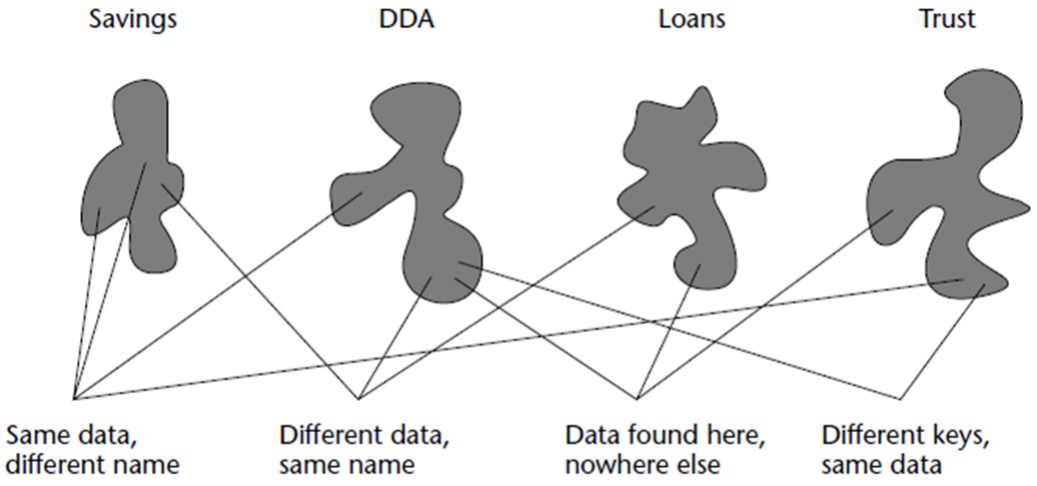


Figure 3-2 Data across the different applications is severely unintegrated.

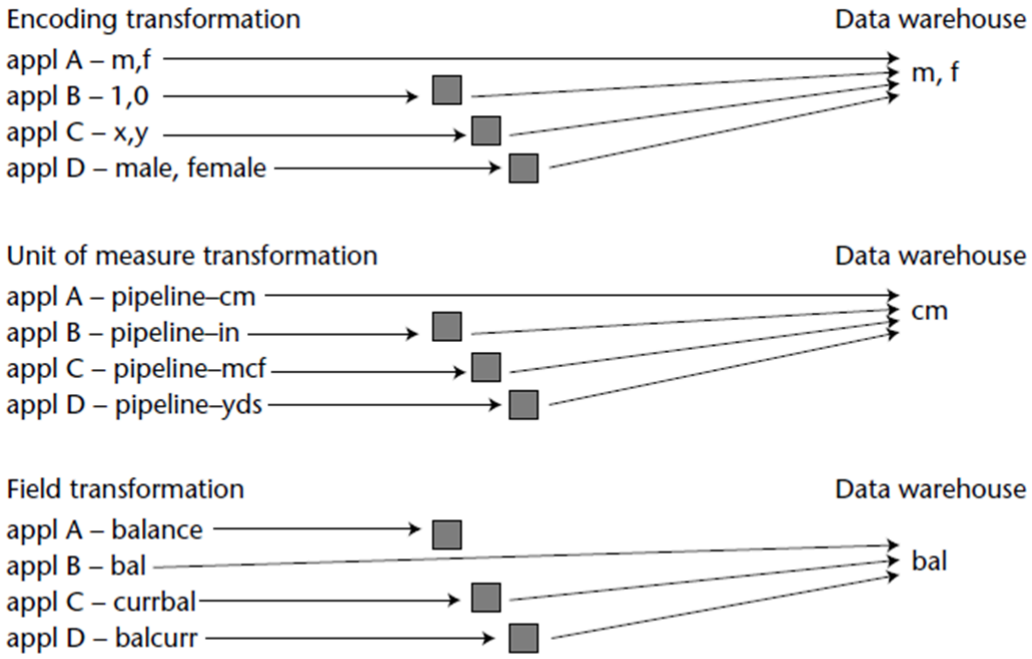


Figure 3-3 To properly move data from the existing systems environment to the data warehouse environment, it must be integrated.

从操作型环境到数据仓库有三种装载工作要做：

1. 装载档案数据
2. 装载在操作型系统中目前已有的数据
3. 将自数据库上次刷新以来在操作型环境中不断发生的变化（更新）从操作型环境中装载到数据仓库中，可以通过以下多种方式来实现：
    1. 时间戳
    2. DELTA 文件
    3. 建立映象文件
    4. 日志文件

需要考虑存取操作型数据的效率

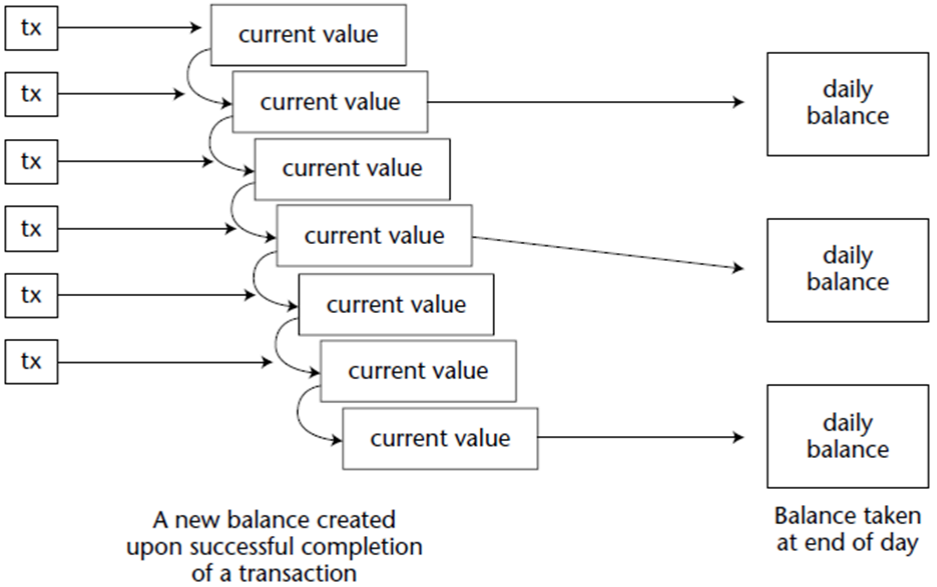


Figure 3-5 A shift in the time-basis is required as data is moved over from the operational to the data warehouse environment.

数据量的管理：

1. 数据集成

2. 多粒度
3. 数据转储

如果没有仔细管理和压缩数据量，那么在数据仓库中聚集的庞大数据量将无法实现仓库的目标。

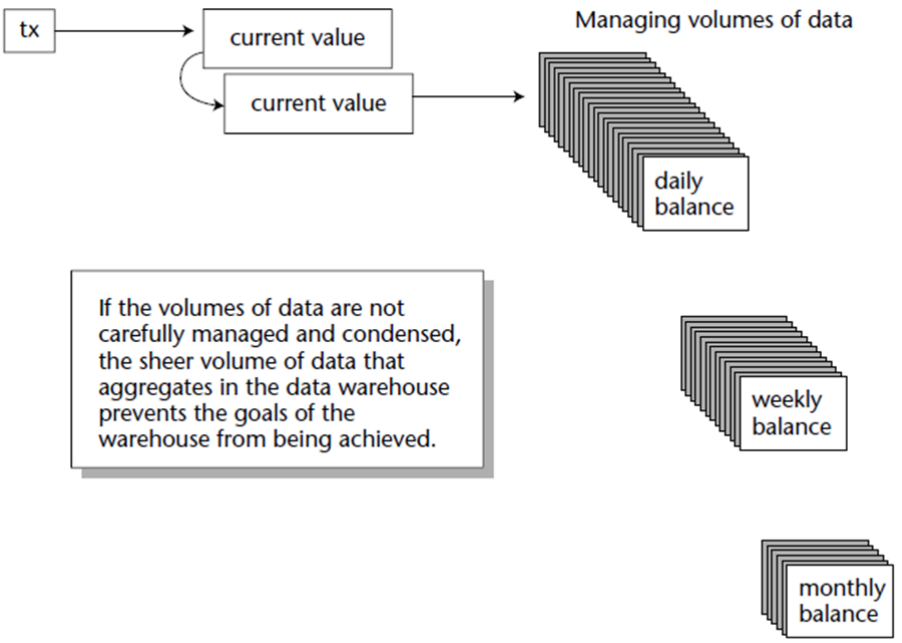


Figure 3-6 Condensation of data is a vital factor in the managing of warehouse data.

4.1 过程或数据模型与设计环境

1. 过程模型（仅仅适合于操作型环境）
    1. 功能分解
    2. 第 0 层上下文图表
    3. 数据流图
    4. 结构图表
    5. 状态转换图
    6. HIPO 图
    7. 伪代码
2. 数据模型（既适合操作型环境，也适合于分析型环境）

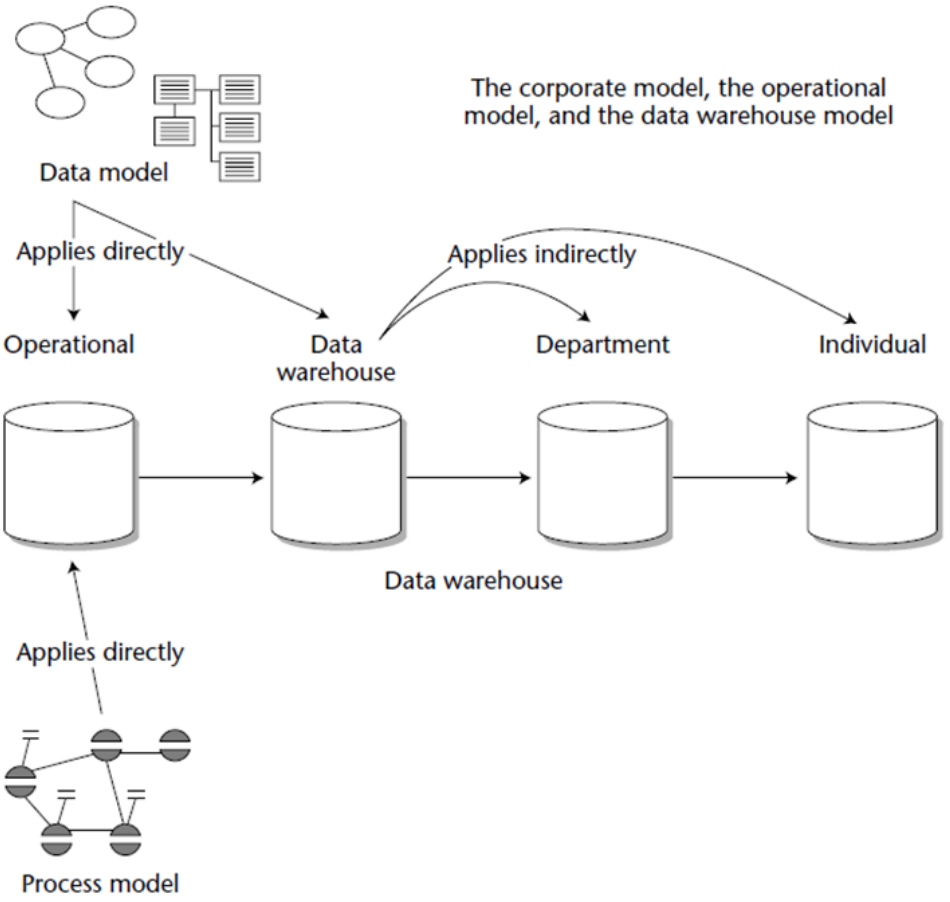
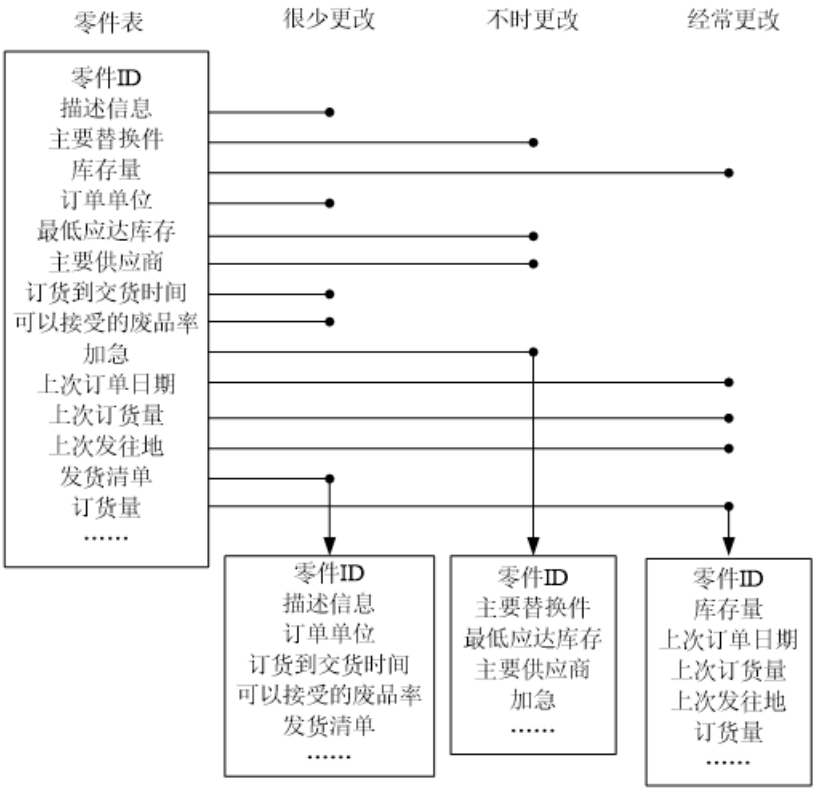


Figure 3-7 How the different types of models apply to the architected environment.

4.2 数据模型的性质

1. 稳定性分析，根据各个数据属性的变化特性将这些属性分组，稳定性和客观世界有关的，比如姓名这个数据的特性决定了其是相对稳定的。
    1. 很少更改：姓名等等
    2. 不时更改：年份等等
    3. 经常更改：库存量等等
2. 根据稳定程度讨论逻辑优化、物理优化及 ETL 设计等；
3. 变化的频率需要通过人工等方式来确定，在优化的过程中是完全按照可变性进行修改的。



4.3 数据模型与迭代开发

迭代开发的重要性：

1. 业界成功的记录强烈地建议这样做
2. 用户在第一遍迭代开发完成以前不能清晰的提出需求
3. 只有实际结果切实而且明确时，管理部门才会作出充分的承诺。因此，必须能很快地见到可见结果

数据模型在每一遍开发中都起着路标的作用，在开发结束时，所有遍次的开发结果融合在一起：

1. 当不同遍次的开发是基于不同的数据模型或没有数据模型时，会产生众多的重复工作，且缺乏一致性
2. 在某一个时刻必须是确定的，并且需要明晰远景和当前的数据模型的关系。
3. 最后一定能拼接出一个方形，不多不少不重（如图 3-24 所示）
4. 重叠可能是某两个开发团队对于某一部分做了两次，难以保证一致性问题
5. 之前的版本则是只有边界是重复的（如图 3-25 所示）
6. 数据模型应该是一个蓝本

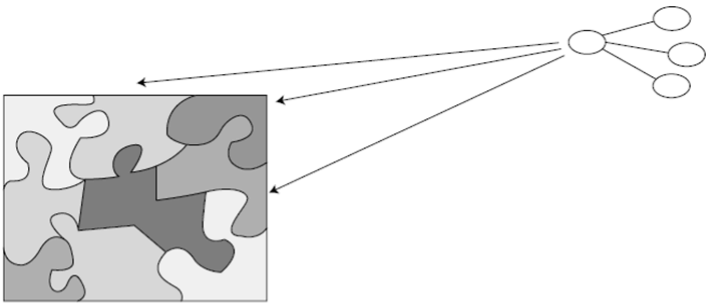


Figure 3-24 At the end of the development effort, all the iterations fit together.

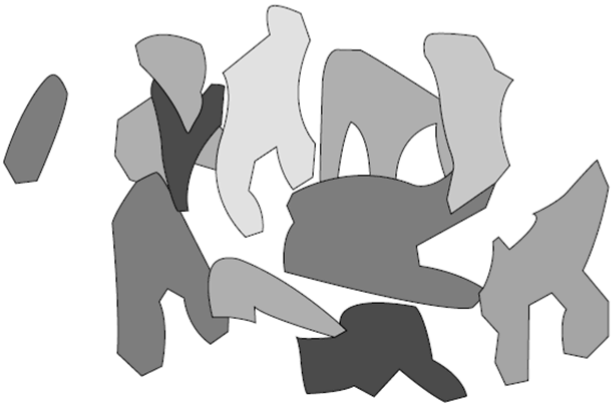


Figure 3-25 When there is no data model, the iterations do not form a cohesive pattern. There is much overlap and lack of uniformity.

4.4 数据仓库设计的三级数据模型

概念模型：

1. 为一定目标设计系统、收集信息而服务的概念型工具，是客观世界到机器世界的一个中间层次，在这之前是将需求部分转化为概念模型。
2. E-R 法

逻辑模型：

1. 描述了数据仓库的主题的逻辑实现
2. 关系模型

物理模型：

1. 逻辑模型在数据仓库中的实现
2. 逻辑模型转换到物理模型不在本课程讨论范围，同样的，从物理模型到逻辑模型也是。

> 基本上是对应的，但是相对比较少

1. 高级模型：
    1. 对数据的抽象程度最大
    2. E-R 图
2. 中级模型：
    1. 数据项（dis-data item set）
    2. dis 是 E-R 图的细分，大体上 E-R 图中的每个实体对应一个 dis
3. 低级模型：物理数据模型

5. 数据仓库的设计步骤

数据仓库设计大致有如下几个步骤：

1. 系统规划
    1. 明确主题

    2. 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

5.1 系统规划

5.1.1 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。

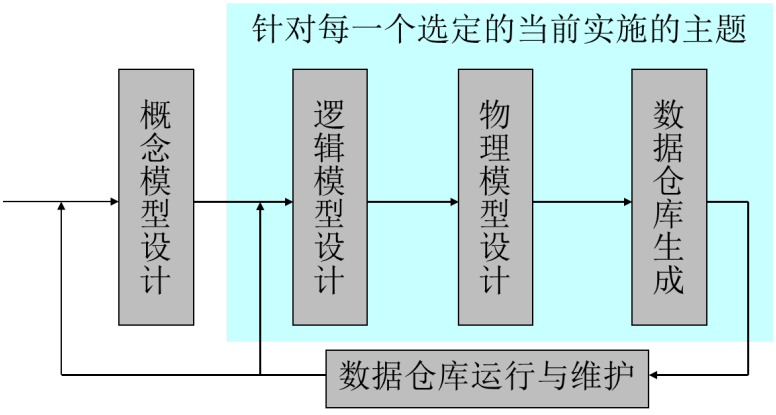
主题是一种较高层次的抽象，对它的认识与表示是一个逐步完善的过程。因此，在开始时不妨先确定一个初步的主题概念以利于设计工作的开始，此后随着设计工作的进一步开展，再逐步扩充与完善（原型设计法）。

5.1.2 技术准备

准备具体的技术要求和物理实现环境，包括：

1. 技术评估，其内容包括数据仓库的性能指标，如：
    1. 数据存取能力（包括管理海量数据的能力）
    2. 模型重组能力
    3. 数据装载能力
2. 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

5.1.3 数据仓库的设计/生成和运维



概念模型设计：星型模型或雪花模型

5.2 概念模型设计

1. 确定系统边界
2. 确定主要的主题及其内容
3. OLAP 等分析应用的设计：在后续分析和二次分析的过程中是非常重要的

5.2.1 确定系统边界

1. 要做的决策类型有哪些?
2. 决策者感兴趣的是什么问题?
3. 这些问题需要什么样的信息?

4. 要得到这些信息需要包含哪些数据源?

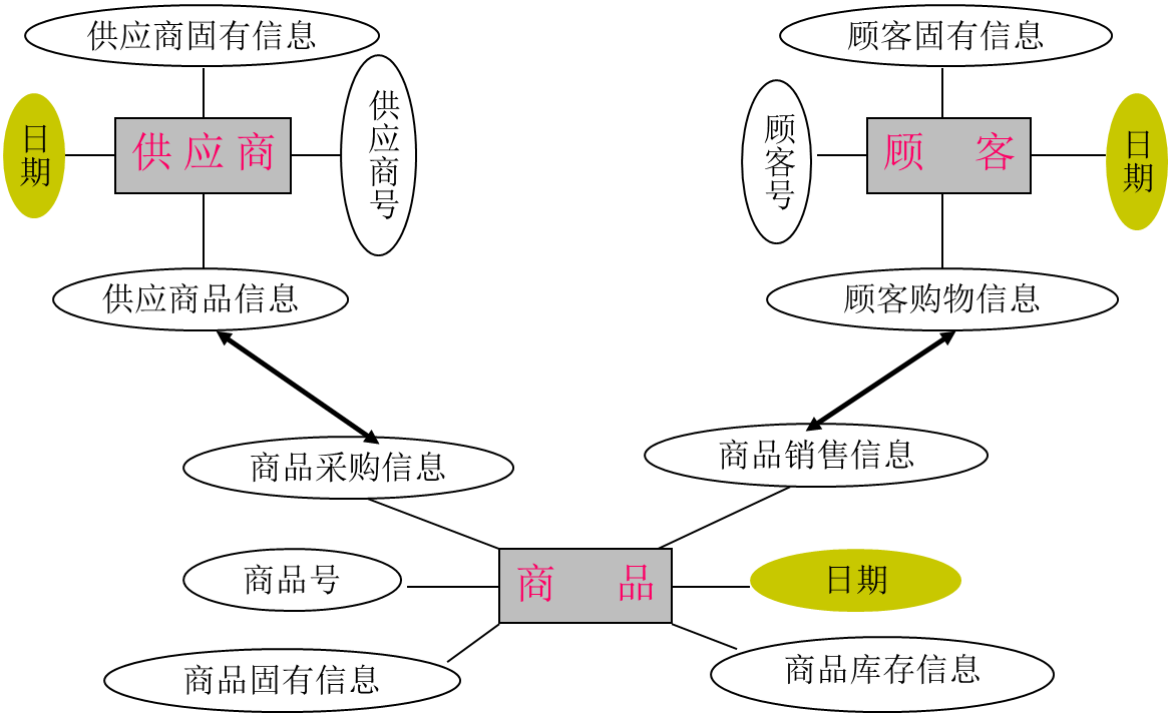
5.2.2 确定主要的主题及其内容

确定主要的主题:

1. 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:
    1. 确定主题及其属性信息
    2. 描述每个属性的取值情况：变化情况
        1. 固定不变的
        2. 半固定的
        3. 经常变化的
    3. 确定主题的公共码键
    4. 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

商品、顾客和供应商之间的 E-R 图



5.3 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
3. 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

5.3.1 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:

1. 数据越详细，粒度越小，级别就越低
2. 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:

1. 根据时间跨度进行的统计有：天，周，月，季度，年

2. 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

5.3.1.1 商品销售：“时间”属性的粒度设计

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

2. 商品销售信息：（细节数据、天、周、月、季度、年）

```
/* 细节数据 */
销售表（商品号，顾客号，销售日期，售价，销售量，...）
/* 以“天”为时间统计单位的综合数据 */
销售表 d1（时间属性_日 ，商品属性，... ，销售总量）
/* 以“周”为时间统计单位的综合数据 */
销售表 d2（时间属性_周 ，商品属性， ... ，销售总量）
/* 以“月”为时间统计单位的综合数据 */
销售表 d3（时间属性_月 ，商品属性， ... ，销售总量）
/* 以“季度”为时间统计单位的综合数据 */
销售表 d4（时间属性_季度，商品属性， ... ，销售总量）
/* 以“年”为时间统计单位的综合数据 */
销售表 d5（时间属性_年，商品属性， ... ，销售总量）
```

5.3.1.2 商品销售：“商品”属性的粒度设计

1. 商品销售信息：商品属性角度的粒度设计补充，细节、商品、小类、大类

```
/* 细节数据 */
销售表（商品号，顾客号，销售日期，售价，销售量，...）
/* 以具体“商品”为商品属性的统计单位的综合数据 */
销售表 p1（时间属性_X ，商品属性_商品号，... ，销售总量）
/* 以“小类”为商品属性的统计单位的综合数据 */
销售表 p2（时间属性_X ，商品属性_小类 ， ... ，销售总量）
/* 以“大类”为商品属性的统计单位的综合数据 */
销售表 p3（时间属性_X ，商品属性_大类 ， ... ，销售总量）
```

2. 将时间、商品等属性综合起来考虑，上述的统计表有很多。

5.3.1.3 商品库存：“时间”属性的粒度设计

商品库存信息：

```
/* 细节数据 */
库存表（商品号，库房号，库存量，日期，...）
/* 样本数据 */
库存表 1（商品号，库房号，库存量，取样时间_周，...）
库存表 2（商品号，库房号，库存量，取样时间_月，...）
库存表 3（商品号，库房号，库存量，取样时间_季度，...）
库存表 4（商品号，库房号，库存量，取样时间_年，...）
```

数据综合的方式与前面的商品销售主题不同

5.3.1.4 用于商场管理者的数据仓库

主题名	公共码键	属性信息
商品	商品号	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

主题名	公共码键	属性信息
供应商	供应商号	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	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

5.3.2 数据分割

数据的分割是指把逻辑上是统一整体的数据分割成较小的、可以独立管理的数据单元进行存储（关系），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：

1. 数据量的大小
2. 数据分析处理的对象（主题）
3. 简单易行的数据分割标准
4. 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

数据分割技术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

5.3.3 定义数据来源及其抽取规则

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
……	……	……	……	……

5.4 物理模型设计（重要）

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。

在任务上，数据仓库和数据库没有太大差异。

常用的一些技术有：

1. 合并表
2. 建立数据序列
3. 引入冗余
4. 表的物理分割
5. 生成导出数据
6. 建立广义索引

规范化/反规范化

5.4.1 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，以减低表的连接操作的代价。

合并表技术与传统关系数据库中的集簇（Clustering）技术类似，如下图所示

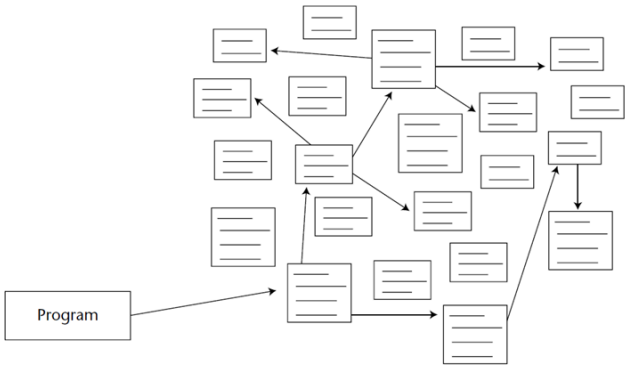


Figure 3-26 When there are many tables, much I/O is required for dynamic interconnectability.

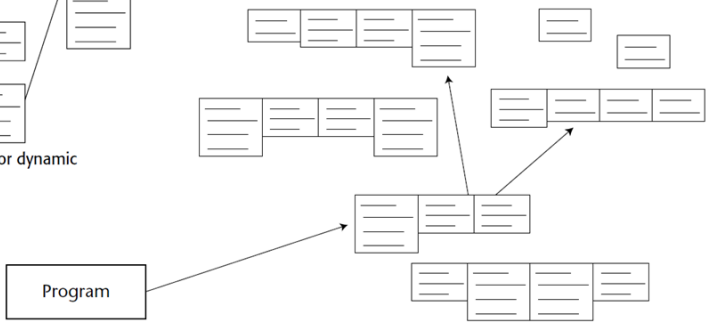


Figure 3-27 When tables are physically merged, much less I/O is required.

1. 代码具有一定的随意性
2. 数据库只能在左上部分和右下部分只能选择一个，否则会导致冗余性
3. 但是数据仓库中，在持有左上数据的基础上，也可以持有右下的数据
4. 右下的数据也可以放置在数据集市（狭义数据仓库）中。

5.4.2 建立数据序列

考虑创建一个数据序列，这样如果数据存放在一行中，那么一次 I/O 就足以检索到了。

通常当数列中值的数量稳定、数据是按顺序访问的、数据的创建与修改在统计上是以非常有规律的方式进行等条件都满足时，创建一个数据序列才是有意义的。

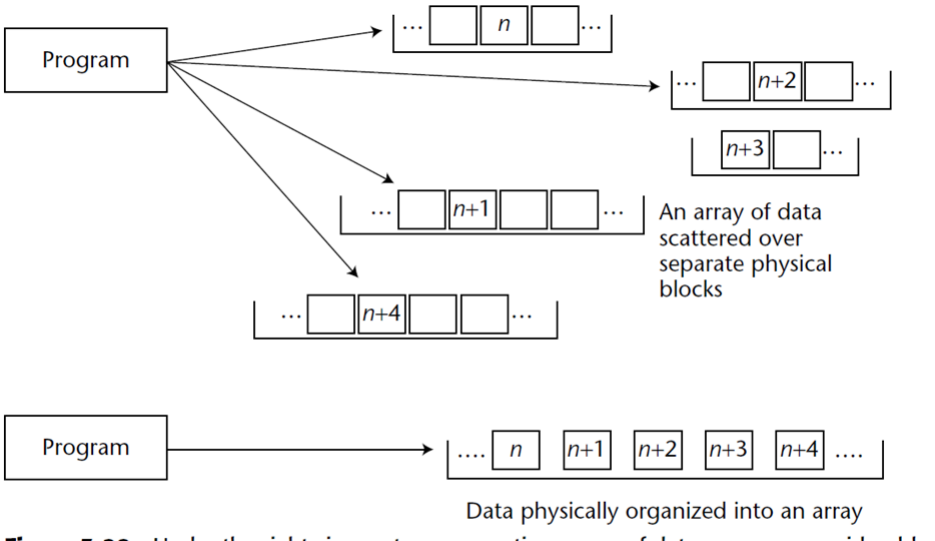
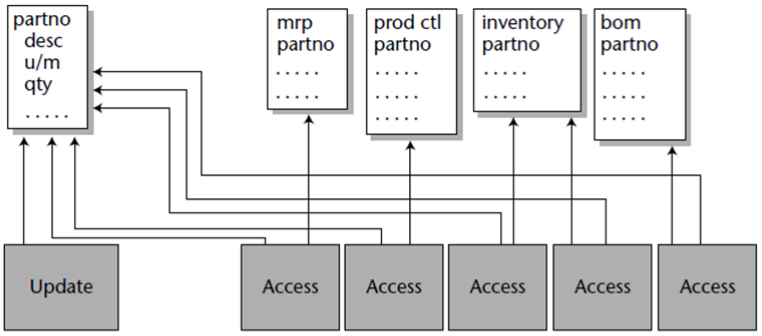


Figure 3-28 Under the right circumstances, creating arrays of data can save considerable resources.

如果目前的存储系统是顺序式的和非顺序式的是有区别的。

5.4.3 引入冗余

1. 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；
2. 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，数据稳定性这一点上就是和数据仓库系统有着本质区别。



Description is nonredundant and is used frequently, but is seldom updated.

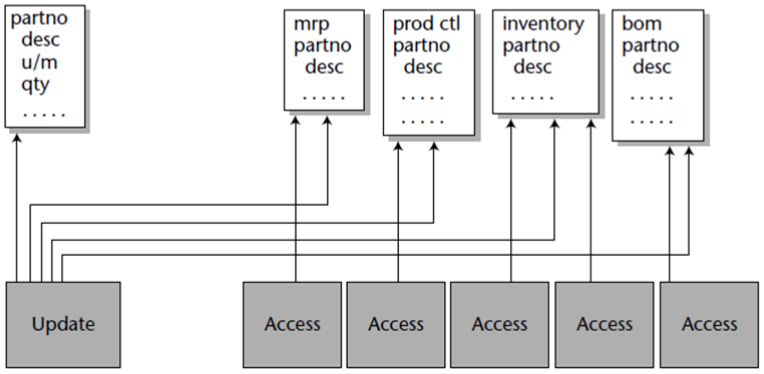


Figure 3-29 Description is redundantly spread over the many places it is used. It must be updated in many places when it changes, but it seldom, if ever, does.

1. 其他表通过外关键字 partno 和原来的表进行关联；
2. 每一条连接线都是跨表的描述访问，由于跨表访问，导致了部分关键字是冗余存储的；
3. 一般的数据库不会有过多冗余，因为无法保证备份是一致的：因为不是所有的冗余表对用户应用都是透明的；
4. 就算执行了冗余存储，和数据库本身也没有关系；
5. 然而在数据仓库中没有这个问题，因为数据仓库是通过 ETL 进行同步更新的。

5.4.4 表的物理分割

1. 类似于在逻辑设计阶段的数据分割
2. 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
    1. 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；
    2. 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。
3. 分拆方式
    1. 横拆：将元组拆出来
    2. 纵拆：将属性拆出来
4. ETL 不是对所有的数据都进行检查有无修改，比如姓名就不必经常查看修改；

5. 表的物理分割对于各种关于更新的属性是比较重要的。

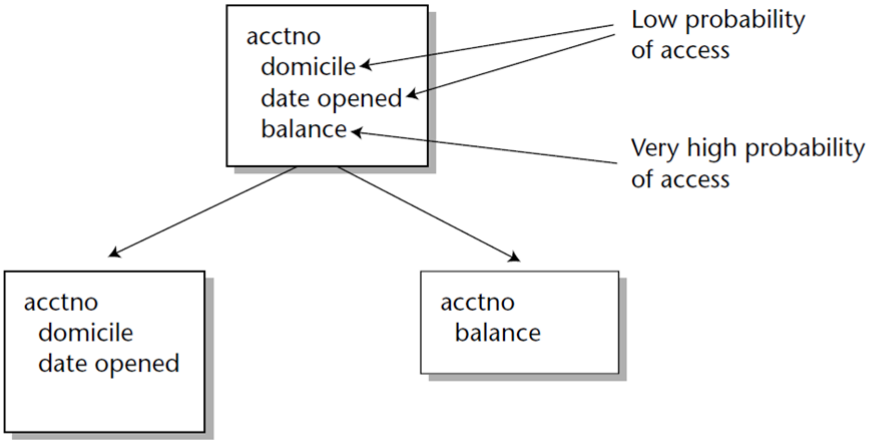


Figure 3-30 Further separation of data based on a wide disparity in the probability of access.

5.4.5 生成导出数据

在原始、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。

优点：

1. 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，提高分析操作的性能；
2. 避免了不同用户进行重复统计操作可能产生的偏差。

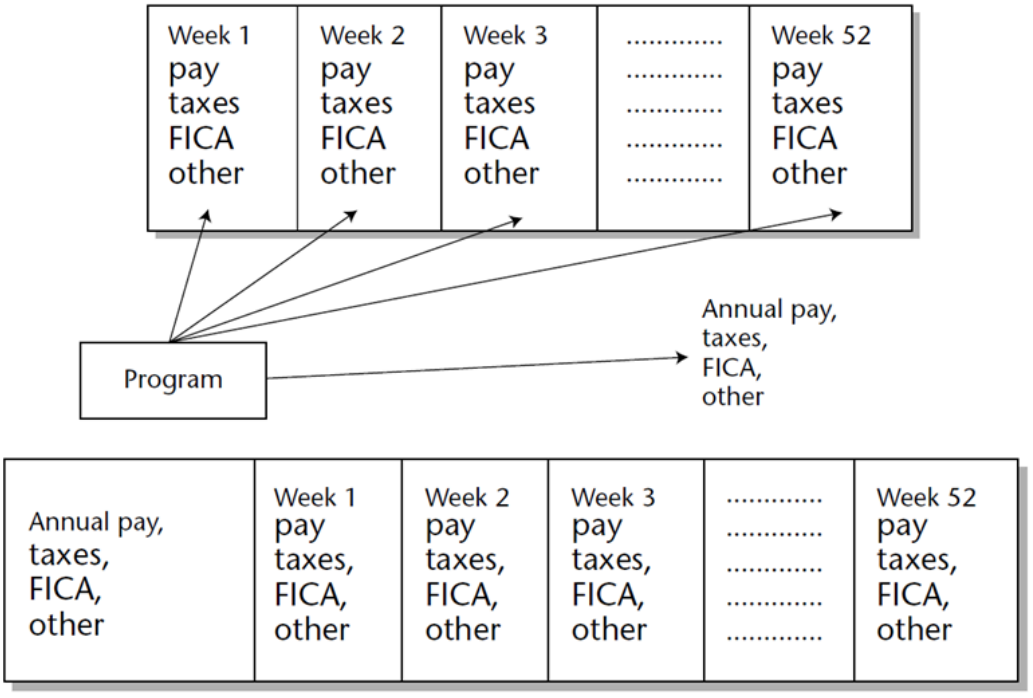


Figure 3-31 Derived data, calculated once, then forever available.

5.4.6 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：

1. 当月销售额最高的商店?
2. 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。

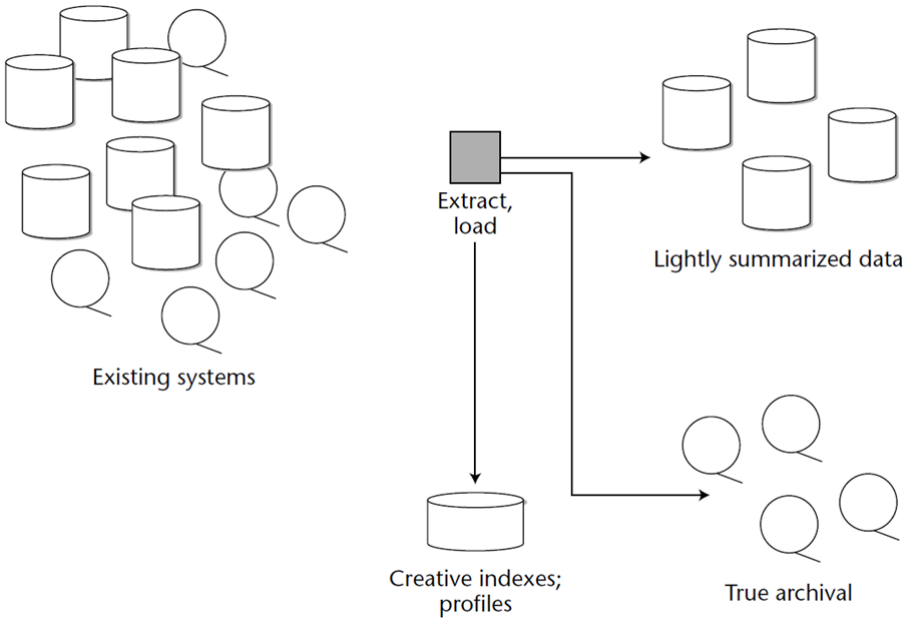


Figure 3-32 Examples of creative indexes.

1. 广义索引能不能运行在数据库上？不能，因为广义索引会影响到操作型数据环境的效率
2. 使用冗余仓库，来避免对数据仓库的大量刷新查询操作的发生。

5.5 数据仓库生成

1. 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
2. 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
3. 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数据模式中去

5.6 数据仓库的使用与维护

1. 在数据仓库建立后，就可以建立分析、决策型的应用系统
2. 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
3. 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

5.7 数据仓库的生命周期