0Tan0101900re10674G7810r0:0r540oia 力010.01079t

010400:

GDc1 rQ7010L

0001Q100i0101G

06700101070r04010010m00M05010 101001010913L0 m200

I0ft0 准 0 1 7orjr0tu010u101010u 2YCp⁰04

0101 G1 GLG r 0r0410 0Q

**第** **4** **章** **数据清洗与数据集成**

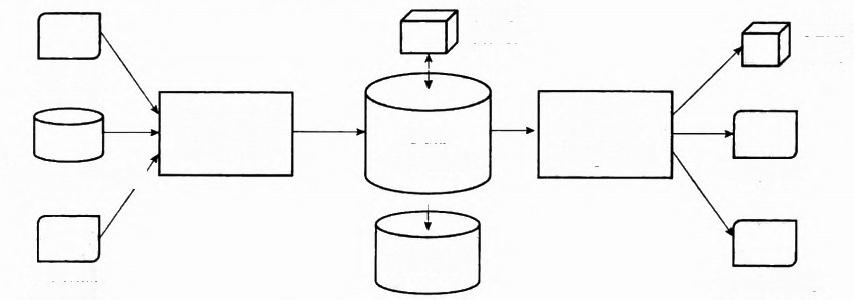
**4.1** **数据抽取、转换与装载**

面向数据服务(或者OLTP) 应用的数据库， 一般不运行特别复杂的数据分析任务。 为了对数据进行分析，我们从这些数据库里抽取、转换和装载 (Extract,Transform and Load,ETL) 数据到数据仓库中，进而在它之上运行复杂的分析负载，包括 OLAP 分 析 和数据挖掘，从数据里挖掘模式和知识，如图4- 1所示。

Meta Data

External Sources

Data Mining



OLAP Query

Internal Sources

Query&

Analysis

Component

Reports

A

Data Integration Component

Data

Warehouse

Operational DBs

OLAP Server

**图** **4** **-** **1** **典型的数据仓库架构**

在这个过程中，如果从多个异构的数据源 ETL 数据到数据仓库中，而且这些数据源 存在各种异构性及不一致性，那么就需要对数据进行集成。可见，数据集成是从多个数据 源建立统一的数据视图的一种技术。如果以数据仓库的方式实现数据集成，需要从各个数 据源对数据进行 ETL 操作。建立数据仓库是数据集成的一种模式，另外两种模式是联邦 数据库模式和中介者模式。如果仅仅从一个数据源 ETL 数据到数据仓库中，那么无须进 行数据集成。在这种情况下， ETL 和数据集成没有关系。

在进行ETL 操作时，如果数据源的数据质量较差，在进行数据转换时，需要利用数



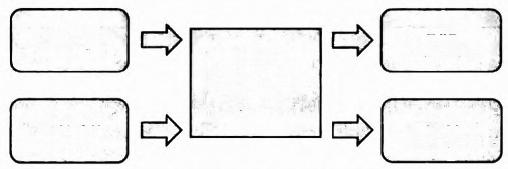
据清洗技术，解决数据质量问题。如果数据源的数据质量得到保证，则无须数据清洗，数 据的转换操作就是比较简单的，比如进行简单的规范化处理。

数据清洗是一种消除数据里面的错误、去掉重复数据的技术。它可以集成在ETL 过 程中，在从数据源建立数据仓库的过程中发挥作用，也可以直接运行在某个数据库上，数 据经过清洗以后，最后还是保存到原来的数据库里。

**4.2** **数据清洗**

**4.2.1** **数** **据** **清** **洗** **的** **意** **义**

基于准确的数据(高质量)进行分析，才有可能获得可信的分析结果，基于这些分析 结果，才有可能做出正确的决策，否则，在不准确的数据(包含很多错误)上进行分析， 有可能导致错误的认识和决策，即垃圾进，垃圾出 (Garbage in,then Garbage out)。由 此可见，数据的质量是一个重要的问题。据估计，数据中的异常 (Anomaly) 和 杂 质 (Impurity), 一般占到数据总量的5%左右。数据质量的重要性见图4-2。



**Bad**

Decision

Good Decision

**Garbage**

Accurate Data

Data

Management

System

**图4-2** **数据质量的重要性**

数据清洗可以是对单个数据库里的数据的错误进行修正，比如剔除重复数据 (De-Du- plicate) 。 此外，数据清洗往往和数据集成联系在一起，当从多个数据源进行数据集成时， 通过数据清洗技术，剔除数据中的错误，以便获得高质量的数据。

在以下的讨论中，我们将围绕关系数据模型进行讨论。需要注意的是，我们将要展现 的数据清洗技术不仅仅适用于关系数据的清洗，同样的原理和方法在其他类型数据的清洗 中也可以灵活运用。

数据是对现实世界的实体/事实的符号化表示，表现为一系列符号化的值。在关系模 型中， 一个元组表示客观世界的一个实体 (Entity) 。 实体包含一组属性 (Property), 每 个属性具有属于某个值域的一个值。这些描述信息称为数据模式 (Schema)。 具体的每个 实体则称为实例 (Instance)。 比如，学生实体，具有姓名、性别、出生年月、身高、体重 等属性。每个具体的学生是一个实例，分别具有具体的姓名、出生年月、身高和体重。

正如上文提到的，低质量的数据导致无效的分析结果，不仅不能很好地支持政府管 理、企业的运营决策以及科学研究的分析任务等，反而可能造成误判和损失。比如在商业 方面，如果客户的地址不正确.有可能造成(促销)信件无法寄达。在邮件列表 (Mail- ing List) 里的重复客户，导致商家发送重复的(促销)信件，造成不必要的成本。客户 重复接到一样的促销信件会相当郁闷。对客户消费习惯和偏好进行刻画的数据不准确，导



致定向广告发送到不合适的人群，广告产品和用户需求不匹配。

**4.2.2** **数据异常的不同类型**

数据清洗的目的是剔除数据中的异常。首先，我们必须了解数据中有什么异常情况。

数据的异常可以分为三类，分别是语法类异常 (Syntactical) 、 语义类异常 (Seman- tic) 和覆盖类异常 (Coverage Anomaly)。语法类异常指的是表示实体的具体的数据的值 和格式的错误。语义类异常则指数据不能全面、无重复地表示客观世界的实体。覆盖类异 常指的是数据库中的记录集合不能完整地表示客观世界中的所有实体，数据库中的实体数 量比客观世界中的实体数量要少。

**1.** **语法类异常**

第一种语法类异常是词法错误 (Lexical Error)。它指的是实际数据的结构和指定的结 构(即数据模式)不一致。比如，在一张人员表中，每个实体有四个属性，分别是姓名、 年龄、性别和身高，某些记录只有三个属性，这就是词法错误(见表4-1)。

**表4-** **1** **包含词法错误的数据库表**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名(Name) | 年龄(Age) | 性别(Gender) | 身高(Height)(m) |
| Peter | 23 | M | 1.78 |
| Tom | 24 | M | 1.75 |
| Tom | 22 |  | 1.62 |

第二种语法类异常是值域格式错误 (Domain Format Error)。它指的是实体的某个属 性的取值不符合预期的值域中的某种格式。值域是数据的所有可能取值构成的集合。比如 姓名是字符串类型，在名和姓之间有一个“,”,那么 “John,Smith” 是正确的值，“John Smith”则不是正确的值。

第三种语法类异常是不规则的取值 (Irregularity) 。 它指的是对取值、单位和简称的 使用不统一、不规范。比如在一个数据库表里面，员工的工资字段有的用“元”作单位， 有的用“万元”作单位。

**2.** **语义类异常**

第一种语义类异常是违反完整性约束规则 (Integrity Constraint Violation)。对于关 系数据模型来讲有三类完整性，分别是实体完整性、参照完整性和用户自定义完整性(在 第 2 章“OLTP 与数据服务”中详细描述)。违反完整性约束规则是指一个元组或者几个 元组不符合上述完整性约束规则。比如，我们规定员工表的工资字段必须大于0,如果某 个员工的工资小于0,就违反了完整性约束规则。

第二种语义类异常是数据中出现矛盾 (Contradiction) 。 即一个元组的各个属性取值， 或者不同元组的各个属性的取值，违反了这些取值的依赖关系。比如，我们可以根据员工 的应发工资减去个人所得税计算出实发工资。如果在数据库表里某位员工的实发工资不等 于应发工资减去个人所得税，就出现了矛盾。

第三种语义类异常是数据的重复值 (Duplicate)。它指的是两个或者两个以上的元组 表示同一个实体。需要注意的是，不同元组的各个属性的取值有可能不是完全相同的。

第四种语义类异常是无效元组 (Invalid Tuple)。它指的是某些元组并没有表示客观 世界的有效实体。比如，学生表里有一个学生，名称是“王涛”,但是学校里并没有这

个人。

**3.** **覆盖类异常**

第一种覆盖类异常是值的缺失 (Missing Value)。它指的是在进行数据采集时就没有 采集到相应的数据。比如元组的某个属性，它的值是空值 (NULL), 也就是没有值。如 果我们规定数据库表的某个属性不能为空 (NOT NULL) 的约束条件，并且由数据库管 理系统实施这个约束，也就是随时检查用户输人数据是否符合要求，只有符合要求的数据 才能入库，那么用户就没有可能把空值输入到数据库里。

第二种覆盖类异常是元组的缺失 (Missing Tuple)。它指的是在客观世界中，存在某 些实体，但是并没有在数据库中通过元组表示出来。也就是说，这些实体在数据库里缺失 了，在数据库里根本就没有相应的元组。

**4.2.3** **数据质量**

数据质量是一个相当宽泛的概念，它包含很多方面。针对某个数据集，我们根据若干 评价标准 (Criteria), 对数据集的质量进行评价。然后把这些评价标准的得分进行综合， 可以获得一个关于数据集质量的综合评分。

关于数据集质量的评价标准 (Quality Criteria), 可以组织成一个层次结构 (Hierar-

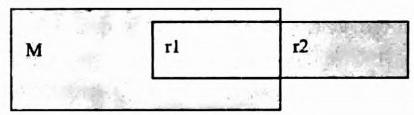
chy) 。上层数据质量标准的得分是其子标准得分的一个综合加权得分。

(1)正确性 (Accuracy) 。 正确性指的是数据集里所有正确的取值相对于所有取值的 比例。这个标准包括三个子标准，分别是完整性 (Integrity) 、 一 致性 (Consistency) 和 密度 (Density)。

①完整性 (Integrity), 继续划分为完备性 (Completeness) 和有效性 (Validity) 。 完 备性指的是在数据集里表示为一个元组的实体，相对于我们所建模的现实世界的所有实体 的集合 M 所占的比例。这个标准考察现实世界的实体是否已经被表示在数据集里了。有 效性则指的是已经在数据集里表示为元组的实体，有多大比例是来自现实世界的。之所以 提出有效性，是因为数据集中的实体，有些可能并未真正代表现实世界中的实体，有可能 是瞎编的。

在图4-3中，数据集中的元组由r1 子集和r2 子集组成， r1 表示来自M 的实体，r2 表示非来自M 的实体，M 表示现实世界的实体，那么完备性=rl/M, 有效性=r1/(rl+

r2)。



**图4-3** **数据集中的实体**

②一致性 (Consistency), 继续划分为模式符合性 (Schema Conformance) 和统一性 (Uniformity)。模式符合性指的是符合数据模式的元组占所有元组的比例。不符合数据模 式主要是指数据的取值不在值域范围之内，比如年龄字段的取值为300岁。统一性指的是 数据集里不包含不规则性 (Irregularity) 的属性占所有属性的比例。不规则的取值 (Ir- regularity) 指的是取值、单位和简称的使用不统一。比如在一个数据库表里，每个员工的



工资字段，有的用“元”作单位，有的用“万元”作单位。

③密度 (Density), 指的是所有元组里.各个属性上的缺失值 (Missing Value) 占 所 有应该存在的所有属性上的取值的比例。这些元组代表了现实世界里的实体。

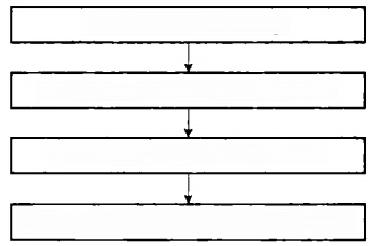
(2)唯 一 性 (Uniqueness), 是指代表相同实体的重复元组占数据集里所有元组的比 例。高质量的数据集、现实世界的实体都在里面得到表示，而且仅仅表示一次。如果有两 个以上元组表示同一个实体，就出现了重复。

**4.2.4** **数据清洗的任务和过程**

数据清洗是剔除数据里的异常，使得数据集成为现实世界的准确、没有重复的 (Cor- rect &.Duplicate Free) 表示的过程。它包含对数据的 一 系列操作，这些操作包括：

(1)对元组及其各个属性值的格式进行调整，使之符合值域要求，使用统一的计量单位、 统一的简称等。(2)完整性约束条件的检查和实施 (Enforcement) 。(3) 从已有的取值导 出缺失的值。(4)解决元组内部和元组之间的矛盾冲突 (Contradiction)。(5) 消除、合 并重复值。(6)检测离群值 (Outlier), 这些离群值极有可能是无效的数据。

数据清洗的过程可以分为4个主要步骤，分别是：(1)对数据进行审计，把数据异常 (Anomaly) 的类型标识出来。(2)选择合适的方法，用于检测和剔除这些异常。(3)在 数据上执行这些方法。第2步和第3步实际上是定义数据清洗的工作流和执行这个工作 流。(4)最后，后续处理和控制阶段将检查清洗结果，把在前面步骤中没有纠正过来的错 误元组进行进一步处理。数据清洗流程见图4-4。



数据审计 (Data auditing)

定义工作流 (Workflow specification)

执行工作流 (Workflow execution)

后续处理和控制 (Post processing/control)

**图4-4** **数据清洗流程**

**1.** **数据审计**

数据清洗的第一步是找出数据中包含的各种异常情况 (Anomaly) 。 一般通过对数据 进行解析以及采用各种统计方法来检测数据的异常。对整个数据集的每个属性进行分析， 可以统计出该属性的最大/最小长度、取值范围、每个取值的频率、方差、唯一性、空值 出现的情况、典型的字符串模式、数据集体现出的函数依赖 (Functional Dependency) 关 系、数据中体现的关联性(关联规则)等。函数依赖是指一个关系表的任意两个元组 rl, r2 在属性集 X 和属性集 Y上具有以下的性质：如果 r1 [x]=r2 [x]. 则 rl [y]=r2 [y], 或者若r1 [y] 不等于r2[y], 则 r1 [x] 不等于r2 [x], 称 x 决 定y, 或 者y 依赖 于 x 。举个例子， 一个学生表里， 一个学生的学号决定学生的姓名。

**2.** **定义数据清洗工作流**

为了把数据中的各种异常情况剔除掉，需要对数据进行一系列的操作。这些操作构成



了一个清洗工作流。一般来讲，在工作流里，首先设法剔除语法类异常，因为语法类异常

往往影响其他类异常的检测和剔除。在剔除其他各类异常方面，没有一个严格的前后 关 系 。

**3.** **执行数据清洗工作流**

定义好的数据清洗工作流，其正确性经过验证以后(可以在小批量数据上实验一下), 在整个数据集上执行。在执行这个工作流期间，判断某些元组是否错误，以及需要从备选 集合中选择一个修改方案时，有时候需要领域专家的介人。然而，专家介入是非常耗时 的，更快的办法是对不能立即纠正的数据先写入一个日志文件，待整个工作流执行完了， 再由领域专家统一检查。

**4.** **后续处理和控制**

数据清洗工作流执行结束后，需要对结果进行检查，以确认各个操作是否正确执行， 数据修正的结果是否正确。控制 (Controlling) 是指对于未能在工作流自动化处理阶段完 成纠错而记录下来的元组，由领域专家进行人工干预，手工完成修正。

**4.2.5** **数据清洗的具体方法**

**1.** **数据解析**

在数据清洗过程中，对数据解析 (Parsing), 目的是检测语法错误 (Syntax Error)。 对于错误的字符串取值，比如应该为 “smith” 而写成了 “snith”, 可以通过字符串解析， 以及使用编辑距离 (Edit Distance), 寻找最相近的正确字符串，给出可能的纠正方案。比 如针对 “Snith”, 根据编辑距离，寻找到的可选的正确字符串是 “smith” 和 “snitch” 。 在数据库表中的姓名字段上，取值为 “smith” 的可能性更大。

如果数据是保存在普通的文件中，那么它可能包含词法错误、值域错误 (Domain Er- ror) 等。在数据库表中，一般实施了严格的完整性约束性检查， 一般不会出现词法错误、 值域错误，但是有可能有值域格式错误 (Domain Format Error)。

**2.** **数据转换**

数据转换 (Transformation) 的目的是把数据从一个格式映射到另外一种格式，以适 应应用程序的需要。在实例层面 (Instance Level),对各个元组的各个字段的取值一般采 用标准化 (Standardization) 和规范化方法 (Normalization), 剔除数据的不规则性 (Ir- regularity) 。 数据的标准化是经过转换函数，把数据的值转换成标准形式.比如把性别字 段的值全部转换成“1”/“0”。规范化则把数据映射到一个最小值、最大值所在的范围， 比如把数据映射到[0,1]之间。

对数据进行转换，有时候需要转换其类型 (Type Conversion)。比如性别字段的 “男”和“女”的取值，有的数据源为 “M”(Male) 和 “F”(Female), 有的数据源为 “1”和“0”,有的数据源则是“M”(Man) 和“W”(Woman), 数据类型不一，取值不 一。我们统一转换成字符类型，用“1”表示“男”,“0”表示“女”,方便后续处理。

在模式层面 (Schema Level), 数据转换一般和数据集成紧密联系在一起。数据集成 是把多个数据源的数据整合在一起，需要从各个数据源映射到一个统一的目标模式 (Common Destination Schema), 在各个数据源和目标数据之间建立模式的映射。模式可 以理解为数据库表结构，数据集成将在本章的后半部分介绍。

行修改，包括新增、删除、修改元组以后，数据集仍然满足一系列的完整性约束条件。可 以使用两种策略实施完整性约束条件： 一个是完整性约束条件检查 (Integrity Constraint Checking); 另一个是完整性约束条件维护 (Integrity Constraint Maintenance)。

完整性约束条件检查是如果某些事务 (Transaction) 执行以后.将使得数据集违反某 些完整性约束条件，那么这样的事务被拒绝执行，这是一种事前控制的方法。完整性约束 条件维护考虑的是如何通过一些附加的修改 (Update) 操作，附加到原有的事务上。保证 经过修改的数据集并未违反任何完整性约束条件，这是一种补救策略。

**4.** **重复数据消除**

重复数据消除 (Duplicate Elimination), 也称为记录连接 (Record Linkage)。在数据 清洗中的重复数据消除和数据集成过程中的重复数据消除，目的都是把数据中的重复元组 给剔除掉，只不过后者处理的是来自多个数据源的数据。

两者使用的技术是类似的，这些技术将在数据集成的实体解析 (Entity Resolution) 部分给予详细介绍。在重复数据消除中，首先我们要把重复数据找出来。需要有一个算法 来确定两个或者两个以上的元组是否实际上代表了现实世界的同一个实体。

**5.** **一些统计方法**

统计方法可以用于对数据进行审计，甚至可以对数据中的异常进行纠正。比如，数据 中的离群值 (Outlier) 检测，可以检测出不符合整个数据集的一般分布特征的少量的元组 或者属性值。通过分析各个属性的取值的平均值 (Mean) 、 标准差 (Standard Deviation)、 取值范围 (Range) 等，以及利用聚类算法，领域专家很容易发现一些意想不到的离群值， 有可能意味这些元组是无效的元组。对于无效的离群值，我们可以把它重置 (Reset) 为 一个平均值。此外，对于缺失值 (Missing Value), 也可以通过统计方法进行类似的处理。

这里需要注意的是，在一些数据集里，离群值不一定意味着错误数据，而是实际情况 如此。比如信用卡诈骗，体现出和正常交易不一样的模式，不能把信用卡诈骗看作错误数 据，应该从这些数据中发现这些模式，防止诈骗的发生。数据中的离群值到底是不是错误 数据而必须给予纠正，需要领域专家介入来判断。

**4.3** **数据集成**

**4.3.1** **数据集成**

在很多应用场合，人们需要整合不同来源的数据。比如在统计系统，上级统计部门需 要把多个下级统计部门的数据源整合在一起，以便获得完整的数据，然后做进一步的统计 分析，才能获得有效的分析结果，否则，不完整的数据将导致分析结果不准确。

数据集成是指把数据从多个数据源整合在一起，提供一个观察这些数据的统一视图

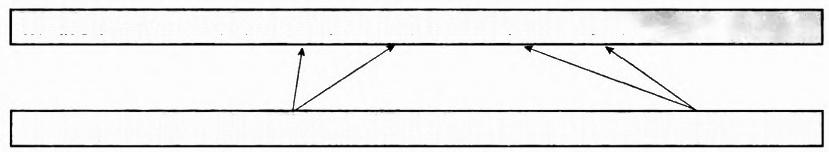


数据集成需要从各个数据源把数据拷贝到目标数据仓库。虚拟式数据集成，数据仍然保留 在各个数据源中，通过某种机制，使得用户可以通过统一的视图对数据进行查询。

**4.3.2** **数据集成需要解决的问题——异构性**

数据集成要解决的首要问题是各个数据源之间的异构性 (Heterogeneity), 异构性就 是差异性。在数据集成中，数据源之间的异构性体现在若干方面。

(1)数据管理系统的异构性。指的是各个数据源采用不同的数据管理系统，包括关系 数据库、面向对象的数据库、XML 数据库等，有些数据源的数据存放在文件系统的文件 中。(2)通信协议异构性。为了从各个数据源获取数据，需要与之进行通信。不同的数据 源，可能采用不同的通信协议。有的数据源有SQL 查询语言接口、有的数据源提供专有 的 API (应用程序编程接口)。(3)数据模式的异构性 (Schema Heterogeneity)。不同数 据源采用不同的数据模式(在关系数据库中，数据模式就是数据库表结构，包括表名、各 个字段名、字段类型、字段的取值范围等的规定)。图4-5展示了两个数据源都采用关系 数据库表保存了客户信息，但它们的表结构是不一样的。(4)数据类型的异构性 (Data Type Heterogeneity)。同样的数据项在不同的原系统中，采用不同的数据类型。比如，有 的数据源把电话号码存为字符串，有的数据源存为数字；有的数据源把姓名存为定长的 (Fixed Length) 字符串，有的数据源存为变长的 (Variable Length) 字符串。(5)取值的 异构性 (Value Heterogeneity)。不同的数据源，有些数据的逻辑取值 (Logical Value)、 物理取值 (Physical Value) 不一样。比如，头衔字段的“教授”的取值，有的数据源为 “Prof”, 有的数据源为 “Prof.” 或 者 “Professor”; 性别字段的“男”或者“女”的取 值，有的数据源为 “M”(Male) 或者 “F”(Female), 有的数据源为“1”或者“0”,有 的数据源则是“M”(Man) 或者“W”(Woman)。(6) 语义异构性 (Semantic Heteroge- neity) 。不同的数据源，某个数据项的取值相同，但是代表不同的含义。比如有的数据源， Title 字段表示职务头衔，比如总裁/副总裁等，同样的字段名称，另外一个数据源可能表 示职称，比如高级工程师/工程师等。



**客户信息表/Customers(ID,firstName,lastName,homePhone,cellPhone,…)**

**客户信息表/Customers(ID,FullName,…),客户电话表/CustomersPhones(ID,Type,PhoneNum)**

**图4-5** **数据模式的异构性**

上述异构性造成了不同数据源之间的数据冲突、不一致。数据集成需要解决这些异构 性问题，也就是解决各个数据源的不一致问题。

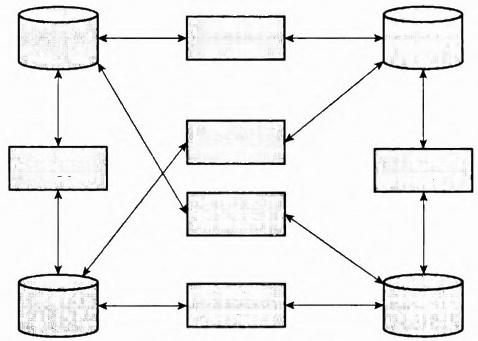
**4.3.3** **数据集成的模式**

我们有三种基本的策略进行数据的集成，分别是联邦数据库 (Federated Database)、 数据仓库 (Data Warehousing)、中介者 (Mediation) (有的文献翻译为仲裁)。



1. 联邦数据库模式

联邦数据库是最简单的数据集成模式，如图4-6所示。它需要在每对数据源之间创 建映射 (Mapping) 和转换 (Transform) 的软件，该软件称为包装器 (Wrapper) 。 当 数 据源X 需要和数据源Y 进行通信和数据集成时，才需要建立X 和 Y 之间的包装器。



Wrapper

Wrapper

Wrapper

Wrapper

Wrapper

Wrapper

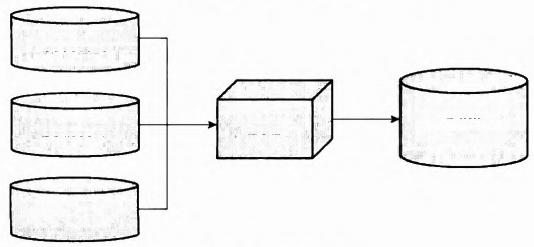
图4-6 联邦数据库

联邦数据库的主要优点是，如果我们有很多的数据源，但是仅仅需要在少数几个数据 源之间进行通信和集成，联邦数据库是最合算的一种模式。其缺点也是很明显的，如果我 们需要在很多的数据源之间进行通信和数据交换，我们需要建立大量的Wrapper 。 在 n 个

数据源情况下，最多需要建立 个 Wrapper, 这将是非常繁重的工作。更为糟糕的 是，如果数据源有变化，需要修改映射和转换机制，对大量的Wrapper 进行更新，变得非 常困难。

2. 数据仓库模式

数据仓库是最通用的一种数据集成模式，如图4-7所示。



Source 1

Data

Source 2

Source n

warehouse

ETL

年 非 =

图4-7 数据仓库

在数据仓库模式中，数据从各个数据源拷贝过来，经过转换，然后存储到一个目标数 据库中。在这种模式下，数据被物化在数据仓库里。数据集成完毕后，我们可以直接对数 据仓库里的数据进行查询。查询处理过程中，无须和数据源打交道。

在图4 - 7中，ETL 是 Extract,Transform,Load 的缩写。ETL 软件负责实现各个 数据源数据的抽取、转换和最后装载到目标数据仓库中。ETL 过程在数据仓库之外完成， 数据仓库负责存储数据，以备查询。



在数据仓库模式下，数据集成的过程实际上是一个ETL 过程，它需要解决各个数据 源之间的异构性、不一致性。如果我们从一个单一的数据源对数据进行抽取、转换和装 载，建立数据仓库，则不是一个数据集成的问题。

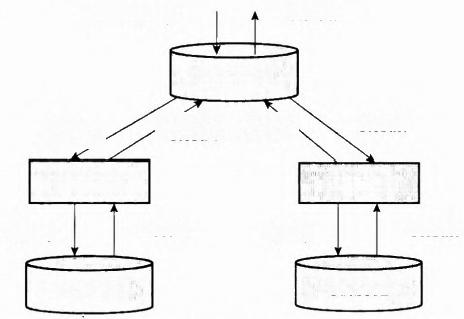
在数据仓库模式下，同样的数据被复制了两份： 一份在数据源里； 一份在数据仓库 里。一个重要的问题是，如何因应数据源里数据的变化，及时更新数据仓库里的数据。我 们可以采用两种方法同步数据源和数据仓库的数据。无论哪种方法，都不能保证数据仓库 的数据是最新的 (Up to Date)。

第一种方法是对数据仓库进行完全重建 (Complete Rebuild)。当数据源发生改变，数 据仓库的数据不是最新的，可以每隔一定时间(比如每天、每周),从各个数据源重新创 建数据仓库。这种方法的主要优点是实现很简单，只需要把 ETL 过程重新运行一遍即可。 缺点也是很明显的，就是每次重建都特别耗时，代价很大，当数据源只发生少许改变时， 这种方法有些不太值当了。

第二种方法是增量式更新 (Incremental Update)。当数据源发生改变，定期根据数据 源的更新，对数据仓库的数据进行适当的更新。这种方法比第一种方法效率更高，代价更 小，但是在实现上更加复杂，需要持续监控和记录数据源的更新，把上次数据仓库更新以 来的数据源的改变都记录下来。另外，需要实现合适的数据仓库更新算法，特别是数据仓 库保存了数据源的汇总数据(在原始数据上经过聚集计算，比如求和、求平均值等)时， 如何根据数据源的更新，对数据仓库的汇总数据进行更新，是一个更为复杂的操作。

**3.** **中介者模式**

数据集成的中介者模式，如图4-8所示。



User Query Result

Mediator

Result Query

Wrapper

Result

Source2

Query

Wrapper

Result Query

Sourcel

Query

Result

**图4-8** **数据集成的中介者模式**

中介者 (Mediator) 扮演的是数据源的虚拟视图 (Virtual View) 的角色，中介者本 身不保存任何数据，数据仍然保存在数据源中。中介者维护一个虚拟的数据模式 (Virtual Schema), 它把各个数据源的数据模式组合起来。

数据的映射(各种转换规则)和传输在查询时刻 (Query Time) 才真正发生。在数据 仓库模式中，有一个从数据源抽取、转换和装载数据的过程，其中的转换操作实现数据源 数据到数据仓库数据的映射，这是两者不一样的地方。

当用户提交查询时，查询被转换成对各个数据源的若干查询。这些查询分别发送到各 个数据源，由各个数据源执行这些查询并且返回结果。各个数据源返回的结果经过合并 (Merge) 后，返回给最终用户。



下面通过一个实例，展示中介者模式的查询转换机制。假设数据源1的数据模式是：

客户信息表/Customers(ID,firstName,lastName,homePhone,cellPhone,…) 数据源2的数据模式是：

|  |
| --- |
| 客户信息表/Customers(ID,FullName,…)  客户电话表/CustomersPhones(ID,Type,PhoneNum) |

基于两个数据源的数据模式，我们定义了中介者的数据模式：

|  |
| --- |
| 客户信息表/Cust(ID,firstName,LastName,…) 客户电话表/CustPhones(ID,Type,PhoneNum,…) |

现在，我们需要查询 customer ID=100的客户的 first name,last name 和 cell phone 信息。我们针对中介者数据模式的查询，用SQL 语句表达为：

|  |
| --- |
| **Select** C.FirstName,C.LastName,P.PhoneNum  **From** Cust C,CustPhones P  **Where** C.ID=P.ID And C.ID=100 And P.Type="cell"; |

这个查询需要转换成两个查询，分别是针对数据源1和数据源2的查询：

|  |
| --- |
| #针对数据源1的查询  **Select** firstName.lastName,cellPhone **From** Customers **Where** C.ID=100; #针对数据源2的查询  #First 和 Last 分别是从FullName 中截取 first name 和 last name 的函数 **Select** First(C.FullName),Last(C.FullName),P.PhoneNum  **From** Customers C,CustomersPhones P  **Where** C.ID=P.ID **And.** ID=100 **And** P.Type="cell": |

两个查询发送给数据源1和数据源2,然后把查询结果进行合并.就可以获得 ID= 100的客户的first name,last name 和 cell phone 等信息。

中介者包括两种类型，分别是GAV(Global as View) 和 LAV(Local as View)。

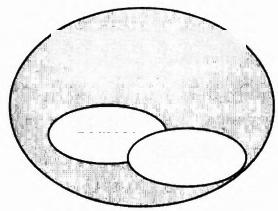
GAV由中介者模式充当所有数据源数据模式的一个视图。中介者通过一些规则，实现 中介者上的查询到针对各个数据源的查询的转换。和单一数据库上的常规视图类似，我们只 能通过视图查找到各个数据源数据的一个子集 (Subset) (参考图4- 9 (a))。

当用户发起针对全局模式的一个查询时，中介者通过一些规则，把查询翻译成针对各 个数据源的查询，然后把这些查询发送给各个数据源进行处理。

LAV 则首先有一个全局数据模式 (Global Schema), 然后基于该全局数据模式定义 各个数据源的模式。每个数据源通过表达式 (Expression) 描述如何从全局模式产生该数 据源的模式。LAV 能够超越各个数据源，涵盖更多的数据(参考图4-9 (b)) 。 下面通过 一个实例，具体了解 LAV。

假设中介者有一个虚拟的关系表Par(c,p), 表示c 和p 的儿子-父亲关系。数据源1



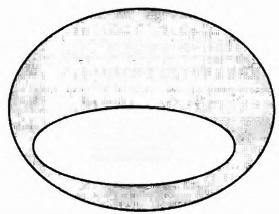


Tuples expressible over

Global view

Sourcel

Source2



Tuples expressible on

**sources**

Tuples by mediators

(a)Global as View (b)Local as View

**图4** **-9** **GAV** **与LAV** **的区别**

提供了一些数据，表达了儿子-父亲关系，即有表达式 “V1(c,p):=Par(c,p)” 。 数 据 源2则提供了一些数据，表达了祖父关系，即有表达式 “V2(c,g):=Par(c,p) 并 且 Par(p,g)”(“:=” 符号表示“定义为”)。在这里我们看到， V1 和 V2 是通过全局模式 的一个表达式来表示出来的。

假设用户现在查询所有人的曾祖父Q(x,w), 那 么Q(x,w) 蕴含着存在y 和 z, 并且具有Par(x,y),Par(y,z),Par(z,w) 等一系列的儿子-父亲关系。如何从数据

源1和数据源2获得这样的曾祖父关系呢，它由三部分组成。

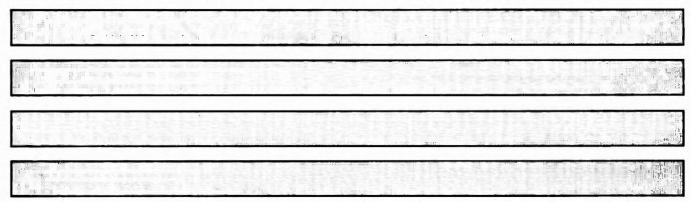
|  |
| --- |
| (1)V1 上符合V1(x,y) 并 且V1(y,z) 并 且V1(z,w) 的所有(x,w)  (2)V2(x,y)And V1(y,w),也是就x 是 y 的孙子，并且Y是 w的儿子  (3)以及V1(x,y)And V2(y,w),也就是x 是 y 的儿子，并且y 是w 的孙子 |

显然，最后找出来的曾祖父关系，比 V1 和 V2 所包含的数据要多，因为第二部分和 第三部分交叉参考了V1 和 V2 的数据，才得出曾祖父关系，这是单独靠V1 和 V2 所不能 找到的。

GAV 的主要优势是易于设计和实现，但是通过GAV 只能看到全部数据的一个子集。 LAV 比 起GAV 来，难以设计和实现，但是它具有更大的扩展性 (Extensible), 新的数据 源可以很容易增加进来，只要从全局模式定义新数据源的模式即可。

**4.3.4 实体解析** **(Entity Resolution)**

来自不同数据源的数据，即便它们表示的是同样的对象(实体),但具体的数据有可能 是不一样的。比如，图4-10中的各个记录表示的是同一个对象(实体),但具体数据是不一 样的。比如，对于名字来讲，有的是用全称John William, 有的是用简称John Will. 等 。



实体信息 (John William,252 Star rd.,MA,01609,508-543-2222)

实体信息 (John Will.,252 Star road,MA,01609,508-543-2222)

实体信息 (John William,252 Star rd.,Massachusetts,01609-3321,508-543-2222)

实体信息 (John William,252 Star rd.,MA,01609,(508)543-2222)

**图4-10** **统一对象的不同表示形式**

造成上述不同表示形式表示同一对象的原因包括：(1)拼写错误，比如把 “Smith” 拼写成 “Smoth”;(2) 采用不同的数据值域，比如对于婚否字段，采用 “YES/NO”“1/0”

或者 “T/F” 。(3) 采用同义词 (Synonym) 、 简称 (Abbreviation) 或者名称的不同写法 (Variant Name), 比如对于街道，采用 “St.”“St” 或 者 “Street” 等。(4)不同地区 (Locale) 的书写习惯不一样，比如对于日期，美国采用“月-日-年”的格式，比如“02- 10-2000”,中国则习惯采用“年-月-日”的格式，比如“2000-02-10”。年月日之间的 分隔符还可以使用“/”“.”等符号。

实体解析 (Entity Resolution) 是找出表示同样实体的记录，并且把这些记录连接 (Record Linking) 在一起的过程。 一般来讲，可以使用如下的方法，对记录进行实体解 析：(1)编辑距离 (Edit Distance), 通过使用编辑距离函数，计算不同字符串之间的编辑 距离，可以计算字符串字段之间的相似度。比如 “John William” 和 “John Will.” 的相 似度，比“John William”和 “Bill Clinton”的相似度要高。(2)对数据进行规范化 (Normalization) 处理和使用领域本体 (Ontology), 我们可以使用字典，把记录中出现的 简称都转换成标准的全称。领域本体包含一个领域的主要概念及其关系，可以帮助我们查 找同义词。(3)对数据进行聚类 (Clustering) 和划分 (Partitioning), 对从各个数据源获 得的所有记录进行聚类分析，相似的记录归入同一类簇 (Cluster), 对于隶属同一类簇的 元组 (Tuple), 我们可以进一步检验它们是否互相匹配，表示同一个实体。

找出了表示同一对象的记录以后，如何合并 (Merge) 这些记录仍然是一个问题。如 果不同的记录，各个字段只有拼写错误或者采用了不同的同义词等情况，合并记录不是困 难的事情，但是如果数据本身存在冲突，要找到哪个值是正确的，不是一件容易的事。在 这种情况下，我们可以把所有的结果都报告出来。比如在下面的实例中，某个客户有两个

冲突的地址，我们可以把这两个地址整合在一起展示 (report) 给用户，让他做决断。

合并前的数据为：

|  |  |  |  |
| --- | --- | --- | --- |
| 标识(ID) | 姓名(Name) | 地址(Address) | 电话(Phone) |
| 100 | Susan Williams | 123 Oak St. | 607-761-7916 |
| 100 | Susan Will. | 456 Maple St. | 607-761-7916 |

合并的结果是：

|  |  |  |  |
| --- | --- | --- | --- |
| 标识(ID) | 姓名(Name) | 地址(Address) | 电话(Phone) |
| 100 | Susan Williams | {123(Oak St., 456 Maple St. | 607-761-7916 |

**4.4** **思考题**

**(1)数据清洗。**

**(2)数据异常的不同类型。**

**(3** **)数据质量及其内涵。**

**(4)数据清洗的具体任务和过程。**



(5)数据清洗的具体技术。

(6)数据集成。

(7) 物理式数据集成、虚拟式数据集成。

(8)数据集成中的异构性问题。

(9)数据集成的模式——联邦数据库、数据仓库、中介者。

(10)中介者的类型： Global as View,Local as View。

(11)实体解析。