大数据、云计算、人工智能、信息安全人才培养丛书
“互联网+”新形态一体化精品教材

数据清洗与治理

SHUJU QINGXI YU ZHILI

主编◎张书强　李　巍　田　钧

扫一扫
学习资源库

◆微课视频
◆教学课件
◆电子教案

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

大数据、云计算、人工智能、信息安全人才培养丛书
“互联网+”新形态一体化精品教材

数据清洗与治理

SHUJU QINGXI YU ZHILI

主　编◎张书强　李　巍　田　钧
副主编◎杜　强　刘　洋　唐家运

扫一扫
学习资源库

◆微课视频
◆教学课件
◆电子教案

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

内容提要

本书从初学者角度出发，详细讲解数据清洗与治理的理论和方法。全书共 9 章，内容包括数据清洗概述，数据类型与清洗，数据仓库与 ETL 技术，数据清洗与准备，数据抽取，数据转换与加载，Web 数据采集，RDBMS 数据清洗，数据聚合、合并与重塑。本书所有知识点都结合具体实例和程序讲解，便于读者理解和掌握。本书可作为高等院校计算机应用、大数据技术等相关专业的教材，也可作为数据清洗入门者的自学用书。

图书在版编目（CIP）数据

数据清洗与治理 / 张书强，李巍，田钧主编 . — 上海：上海交通大学出版社，2021.9

ISBN 978-7-313-25319-4

Ⅰ. ①数… Ⅱ. ①张… ②李… ③田… Ⅲ. ①数据处理 Ⅳ. ① TP274

中国版本图书馆 CIP 数据核字（2021）第 174725 号

数据清洗与治理

SHUJU QINGXI YU ZHILI

主　　编：张书强　李巍　田钧

出版发行：上海交通大学出版社

邮政编码：200030

印　　制：北京华创印务有限公司

开　　本：889 mm × 1194 mm　1/16

字　　数：345 千字

版　　次：2021 年 9 月第 1 版

书　　号：ISBN 978-7-313-25319-4

定　　价：56.00 元

地　　址：上海市番禺路 951 号

电　　话：6407 1208

经　　销：全国新华书店

印　　张：14.5

印　　次：2021 年 9 月第 1 次印刷

大数据、云计算、人工智能、信息安全人才培养丛书

大数据系列专家团队		
姓名	职称	研究方向
刘开南	教授	大数据边缘计算
刘明正	教授	数据分析采集
谢泽奇	教授	数据可视化
李志伟	教授	数据仓库
高丽杰	教授	大数据安全
钱振江	高级工程师	数据清洗
李　魏	教授	数据治理
李晓英	教授	数据可视化
王素华	教授	数据标准
张　晨	教授	大数据分析
靳　帅	教授	大数据技术
赵　鹏	高级工程师	数据可视化分析
张　帅	教授	大数据采集与分析
叶远浓	教授	数据可视化
张　洁	教授	数据仓库
梁军学	教授	大数据安全
张　浩	教授	数据清洗
张仁鹏	教授	数据治理
代　飞	教授	数据可视化
杨杉杉	教授	数据仓库技术
赵　噗	高级工程师	大数据安全

序

随着计算机与信息技术的迅猛发展和普及应用，行业应用系统的规模迅速扩大，行业应用所产生的数据呈爆炸性增长，动辄达到数百 TB 甚至数十至数百 PB 规模的行业、企业大数据已远远超出了传统计算技术和信息系统的处理能力。因此，寻求有效的大数据处理技术、方法和手段已成为现实世界的迫切需求。人们将越来越多地意识到数据对企业的重要性。大数据时代对人类的数据驾驭能力提出了新的挑战，也为人们获得更为深刻、全面的洞察能力提供了前所未有的空间与潜力。“大数据开启了一次重大的时代转型”，大数据将带来巨大的变革，改变我们的生活、工作和思维方式，改变我们的商业模式，影响我们的经济、政治、科技和社会等各个层面。

大数据行业应用需求日益增长，未来越来越多的研究和应用领域将需要使用大数据并行计算技术，大数据技术将渗透到每个涉及大规模数据和复杂计算的应用领域。不仅如此，以大数据处理为中心的计算技术将对传统计算技术产生革命性的影响，广泛影响计算机体系结构、操作系统、数据库、编译技术、程序设计技术和方法、软件工程技术、多媒体信息处理技术、人工智能，以及其他计算机应用技术。

大数据的高速发展也代表着新的生产力、新的发展方向，新一轮产业升级和生产力飞跃过程中，人才作为第一位的资源将发挥关键作用。大数据技术的发展将给研究计算机技术的专业人员带来新的挑战和机遇。国内外 IT 企业对大数据技术人才的需求正快速增长，未来 5 ~ 10 年内业界将需要大量的掌握大数据处理技术的人才。这一人才需求与能够提供的人才数量存在一个巨大的差距。目前，由于国内外高校开展大数据技术人才培养的时间不长，技术市场上掌握大数据处理和应用开发技术的人才还十分短缺。

本套大数据、云计算、人工智能、信息安全人才培养丛书中大数据方向的教材明确了适用专业、培养目标、培养规格、课程体系、师资队伍、教学条件、质量保障等各方面要求，是联合多所高校及多家知名企业共同编撰而成的校企合作教材，充分体现了产教深度融合、校企协同育人的理念，是在校企合作机制和人才培养模式的协同创新下打造的精品教材，既适合高等院校相关专业学生使用，也适用于企业相关岗位专业人才的培养。

计算机科学与技术教授 / 工学博士

大数据（高级）分析师

前言

随着国家将大数据战略提升到国家战略高度，将大数据视为经济发展和转型的重要科技依据，越来越多的企业、组织将数据视为重要资产。近年来，数据库钻研团队对数据清洗非常关注，并且常常将数据清洗与数据仓库、数据挖掘及数据关联关系联系在一起。在网络信息系统或数据仓库中，数据清洗的意义变得尤为重要，因为不同类型的数据源通常以不同的形式出现。经剖析发现，“脏”数据出现的原因以及存在的形式就是数据清洗的原理，从“脏”数据产生的源头对数据进行剖析，对数据集进行考查，进而提取数据清洗规则，最终利用所提取的清洗规则发现数据集中的“脏”数据，然后对该类数据进行清洗。

本书从初学者角度出发，采用理论与实践相结合的方式，详细讲解如何识别数据记录中的错误数据并将其去除，再重新检查和校验数据，去除重复记录，消除异常数据，修正错误数据，以确保数据一致性来提高数据质量。全书共 9 章，主要内容包括数据清洗概述，数据类型与清洗，数据仓库与 ETL 技术，数据清洗与准备，数据抽取，数据转换与加载，Web 数据采集，RDBMS 数据清洗，数据聚合、合并与重塑。本书内容由浅入深，案例丰富，能有效帮助初学者快速入门。

本书在编写上具有以下特色：

（1）通过章前的“学习目标”“知识导图”“本章导入”明确本章要学习的内容，使学生做好学习的准备；通过文中的“说明”“提示”“注意”等模块，丰富知识内容，提高学生的学习兴趣。

（2）在精练语言的基础上，充分利用图、表等尽量形象地描述知识点，帮助学生更好地理解所学内容。内容由浅入深，循序渐进，符合学生的认知规律。

（3）计算机技术发展很快，本书着重讲解当前的最新知识和主流技术，使学生学到的知识和技术都与行业密切联系，做到学以致用。

此外，本书作者还为广大一线教师提供了服务于本书的教学资源库，有需要者可致电 13810412048 或发邮件至 2393867076@qq.com。

本书可作为高等院校计算机相关专业的教材，也可作为相关技术人员培训或工作的参考用书。由于编写时间仓促，加之大数据技术发展迅猛，书中存在的不足和疏漏之处，敬请广大读者批评指正，在此表示衷心的感谢。

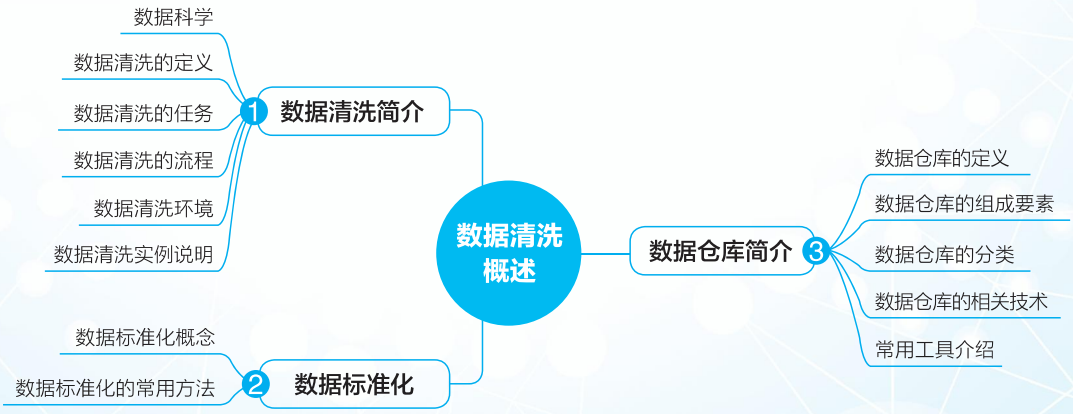
目录

第 1 章 数据清洗概述

学习目标

1. 了解数据清洗的概念。
2. 理解数据标准化概念。
3. 掌握数据仓库的分类。

知识导图



笔记

本章导读

在当今信息技术时代，大数据堪称为一项伟大的技术，它改变了传统的数据收集、处理和应用模式，为众多领域的跨越式发展带来了新的机遇和挑战。大数据的战略价值不是追求掌握庞大的数据量，而在于对这些富有内涵的数据进行专业化处理，获取具有更强决策力、洞察力和流程优化能力的信息资产，进而指导科学决策和生产实践。人类在努力将数据转化为信息和知识的同时，也面临着海量数据中夹杂着“脏”数据的挑战。因此，对原始数据进行有效清洗并将其转化为易理解和易利用的目标数据，已成为人类进行大数据分析和应用过程中的关键一环。数据清洗（data cleaning）用来对数据进行审查和校验，进而删除重复信息，纠正存在的错误，并保持数据的一致性、精确性、完整性和有效性。由此可见，数据清洗在整个大数据分析过程中扮演着重要的角色。本章主要阐述数据清洗的基本概念和相关技术。

1.1 数据清洗简介

1.1.1 数据科学

现代社会的各个角落无不充斥着种类繁多、数量庞大的数据，这些数据不仅包括传统的结构型数据，还包括如网页、文本、图像、视频、语音之类的非结构型数据。大数据的兴起和研究热潮将数据科学推到风口浪尖。大数据不仅是一门技术，更代表了一种潮流和一个时代，而数据科学则是一门新兴的以数据为研究中心的学科。作为一门学科，数据科学以数据的广泛性和多样性为基础，探寻数据研究的共性。例如，自然语言处理和生物大分子模型里都用到隐马尔可夫过程和动态规划方法，其根本原因是它们处理的都是一维的随机信号。再如，图像处理和统计学习中都用到正则化方法，因此用于图像处理的算法和用于压缩感知的算法有许多共同之处。

数据科学是一门关于数据的工程，它需要同时具备理论基础和工程经验，需要掌握各种工具的用法。数据科学主要包括两个方面，即用数据的方法研究科学和用科学的方法研究数据。前者包括生物信息学、天体信息学、数字地球等领域；后者包括统计学、机器学习、数据挖掘、数据库等领域。这些学科都是数据科学的重要组成部分，但只有把它们有机地放在一起，才能形成整个数据科学的全貌。数据科学的综合性也对数据科学家提出了较高的技能要求，他们需要掌握的知识包括计算机、统计学、数据处理和数据可视化等。

数据清洗是数据科学家完成数据分析和处理任务过程中必须面对的一个环节。具体来说，数据科学的处理过程一般包括如下几个步骤。

（1）问题陈述：明确需要解决的问题和任务。

（2）数据收集与存储：通过多种手段采集和存放来自众多数据源的数据。

（3）数据清洗：对数据进行针对性的整理和规范，以便于后面的分析和处理。

（4）数据分析和挖掘：运用特定模型和算法寻求数据中隐含的知识和规律。

（5）数据呈现和可视化：以恰当的方式呈现数据分析和挖掘的结果。

（6）科学决策：根据数据分析和处理结果决定问题的解决方案。

需要指出的是，上述数据科学过程的 6 个步骤并非全部需要，而且上述步骤的执行是一个反复迭代的过程。例如，在一个数据分析项目中可能需要不止一次地执行数据清洗和数据呈现操作。此外，数据分析和挖掘方法会影响数据清洗的手段和方式。

笔记

1.1.2 数据清洗的定义

来自多样化数据源的数据内容并不完美，存在许多“脏”数据，即数据不完整，存在错误和重复的数据、数据不一致和冲突等缺陷。统计资料表明，“脏”数据大约占总数据量的 5%，但会对建立的数据处理和应用系统造成不良影响，扭曲从数据中获得的信息，影响数据应用系统的运行效果，从而进一步影响数据挖掘效能，最终影响决策管理。为了减少这些“脏”数据对数据分析和挖掘结果的影响，必须采取各种有效的措施对采集的原始数据进行有效的预处理，这一预处理过程称为“数据清洗”，即在数据集中发现不准确、不完整或不合理的数据，并对这些数据进行修补或移除，以提高数据质量的过程。

目前，对于数据清洗并没有统一的定义，其定义依赖于具体的应用领域。广义上讲，数据清洗是将原始数据进行精简，以去除冗余和消除不一致，并使剩余的数据转换成可接收的标准格式的过程；而狭义上的数据清洗特指在构建数据仓库和实现数据挖掘前对数据源进行处理，使数据实现准确性、完整性、一致性、唯一性和有效性，以适应后续操作的过程。一般而言，凡是有助于提高信息系统数据质量的处理过程，都可认为是数据清洗。简单地说，就是从数据源中清除错误数值和重复记录，即利用特定技术和手段，基于预定义的清洗规则从数据源中检测和消除错误数据、不完整数据和重复数据，从而提高信息系统的数据质量。

1.1.3 数据清洗的任务

数据清洗就是对原始数据重新进行审查和校验的过程，目的在于删除重复信息、纠正存在的错误，并使得数据保持精确性、完整性、一致性、有效性及唯一性，还可能涉及数据的分解和重组，最终将原始数据转换为满足数据质量或应用要求的数据。对于任何大数据项目而言，数据清洗过程都是必不可少的步骤。此外，格式检查、完整性检查、合理性检查和极限检查也在数据清洗过程中完成。数据清洗对保持数据的一致和更新起着重要的作用，因此被用于如银行、保险、零售、电信和交通等多个行业。

当前，数据清洗主要有 3 个应用领域：数据仓库（data warehouse，DW）、数据库中知识的发现（knowledge discovery in database，KDD）和数据质量管理（data quality management，DQM）。

在数据仓库领域，当对多个数据库进行合并时或对多个数据源进行集成时，需要使用数据清洗。例如，当同一个实体的记录在不同数据源中以不同格式表示或被错误表示的情况下，合并后的数据库中就会出现重复的记录。数据清洗就需要识别出重复的记录并消除它们，也就是所谓的数据合并 / 清除（merge/purge）问题。在数据仓库环境中，需要考虑数据仓库的集成性与面向主题的需要，包括数据的清洗及结构转换。

在数据库中的知识发现领域，数据清洗为 KDD 过程的首个步骤，即对数据进行预处理。KDD 应用中数据清洗的主要任务是提高数据的可用性，如去除噪声、无关数据、空

笔记

值并考虑数据的动态变化等。例如，在字符分类问题中可以使用机器学习技术进行数据清洗，包括使用特定算法检查数据库，以及检测遗失和错误的数据并予以纠正。

数据质量管理用于解决信息系统中的数据质量及集成问题。在数据质量管理领域中，数据清洗从数据质量的角度出发，把数据清洗过程和数据生命周期集成在一起，对数据正确性进行检查，并改善数据质量。

数据清洗对随后的数据分析非常重要，因为它能提高数据分析的准确性。但是，数据清洗依赖复杂的关系模型，会带来额外的计算开销和处理延迟，所以要在数据清洗模型的复杂性和分析结果的准确性之间进行权衡。

1.1.4 数据清洗的流程

数据清洗的基本原理是通过分析“脏”数据的产生原因和存在形式，用数据溯源的思想，从“脏”数据产生的源头开始分析数据，对数据流经的每一环节进行考查，从中提取数据清洗的规则和策略，基于已有的业务知识对原始数据集应用数据清洗规则和策略去发现“脏”数据，并通过特定的清洗算法清洗“脏”数据，从而得到满足预期要求的数据。

1. 分析数据并定义清洗规则

首先定义错误类型，通过全面详尽的数据分析检测数据中的错误或不一致情况，包括手工检查数据样本和通过计算机自动分析程序发现数据集中存在的缺陷。然后，根据数据分析的结果定义数据清洗规则，并选择合适的数据清洗算法。

2. 搜寻并标识错误实例

手工检测数据集中的属性错误需要花费大量时间和精力，成本高昂且此过程本身容易出错。因此，一般倾向于利用高效的检测方法自动搜寻数据集中存在的各类错误包括数据值是否符合字段域、业务规则，或是否存在重复记录等。常用的检测方法主要有基于统计的方法、聚类方法和关联规则方法。消除重复记录首先要检测出标识同一个实体的重复记录，即匹配与合并过程。检测重复记录的算法主要有字段匹配算法、Smith-Waterman 算法和 Cosine 相似度函数。

3. 纠正发现的错误

在原始数据集上执行预定义操作并以得到验证的数据清洗转换规则修正检测到的错误数据，或处理冗余和不一致的数据。需要注意，当在源数据上进行数据清洗时应备份源数据，以防需要撤销清洗操作。根据“脏”数据存在的形式，执行一系列的数据清除和数据格式转换步骤，解决模式层和实例层的数据质量问题。为了使数据匹配和合并变得方便，应该将数据属性值转换成统一的格式。

4. “干净”数据回流

当完成数据清洗后，应用文档记录错误实例和错误类型，并修改数据录入程序，以减少可能的错误。同时，用“干净”的数据替换原始数据集中的“脏”数据，以便提高信息系统的数据质量，还可避免再次抽取数据后进行重复的清洗工作。

5. 数据清洗的评判

数据清洗执行完毕后，有必要对数据清洗的效果进行评价。数据清洗的评价标准主要包括两个方面：数据可信性和数据可用性。

数据可信性包括数据精确性、完整性、一致性、有效性和唯一性等指标。精确性描述

数据是否与其对应的客观实体的特征相一致；完整性描述数据是否存在缺失记录或缺失字段；一致性描述同一实体的同一属性的值在不同的系统中是否一致；有效性描述数据是否满足用户定义的条件或是否在一定的阈值范围内；唯一性描述数据是否存在重复记录。

数据可用性的考查指标主要包括时间性和稳定性。时间性描述数据是当前数据还是历史数据；稳定性描述数据是否是稳定的，是否在其有效期内。

需要指出的是，数据清洗是一项十分繁重的工作。数据清洗在提高数据质量的同时要付出一定的代价，包括投入的时间、人力和物力成本。通常情况下，大数据集的数据清洗是一个系统性的工作，需要多方配合以及大量人员的参与，还需要多种资源的支持。

笔记

1.1.5 数据清洗环境

数据清洗环境是指为进行数据清洗所提供的基本硬件设备和软件系统，特别是已得到广泛应用的开源软件和工具。下面简要列出数据清洗操作常用的一些软件和工具。

（1）终端窗口和命令行界面，如 Mac OS X 上的 Terminal 程序或 Linux 上的 bash 程序。在 Windows 上，有些命令可以通过 Windows 的命令提示符运行，而另外一些命令则要通过功能更强的命令行程序运行，如 Cygwin。

（2）适合程序员使用的编辑器，如 Mac 上的 TextWrangler，Linux 上的 vi 或 emacs，或是 Windows 上的 Notepad++、Sublime Text 等。

（3）Python 客户端程序，如 Enthought Canopy。另外，还需要足够的权限安装一些程序包文件。

（4）电子表格程序，如 Microsoft Excel 和 Google Sheets。

（5）数据库软件，如 MySQL 和 Microsoft Access。

1.1.6 数据清洗实例说明

本节给出一个简单的数据清洗实例，主要是清除隐藏在原始数据集中的噪声数据。在实际情况下，数据处理平台系统经常会遇到各种各样的关于指标均值计算的问题，遵循数理统计的规律，此时噪声对数据均值计算的负面影响是显著的。以网站文件下载为例，假定一组记录文件下载时间长度的原始数据集见表 1-1。直接计算网站文件平均下载时长，结果约为 23000s，即约 6h，与实际情况严重不符，说明这一数据集受到了显著的噪声的影响而导致部分数据值出现严重偏差。为此，必须对原始数据集做异常值识别，并尽可能剔除错误数据。

表 1-1　各个文件的下载时间

序号	下载时长 /s	序号	下载时长 /s
1	30	7	446
2	1	…	…
3	476	2401	956449
4	1034	2402	3844
5	1	2403	2065553
6	59		

笔记

具体来说，可以基于数据的分布特征，利用分箱法或聚类法识别数据集中的噪声数据。一般情况下，对于离散程度适中的数据源来说，数据自身分布将会集中在某一区域内，所以利用数据自身分布特征识别噪声数据，再根据分箱或聚类方法在数据集中域中识别离群值及异常值。分箱法需要考虑某一数据近邻的数据，并使用平滑数据值替换当前的有序数据。与分箱法相比，聚类提供了识别多维数据集中噪声数据的方法。在很多情况下，把整个记录空间聚类，能发现在字段级检查中未发现的孤立点。聚类就是将数据集分组为多个类或簇，同一个簇中的数据相互之间有较高的相似度，而不同簇中的数据差别较大。将散落在外，不能归并到任一类中的数据称为“孤立点”或“离群点”，并作为噪声数据（异常值）进行剔除，如图 1-1 所示。

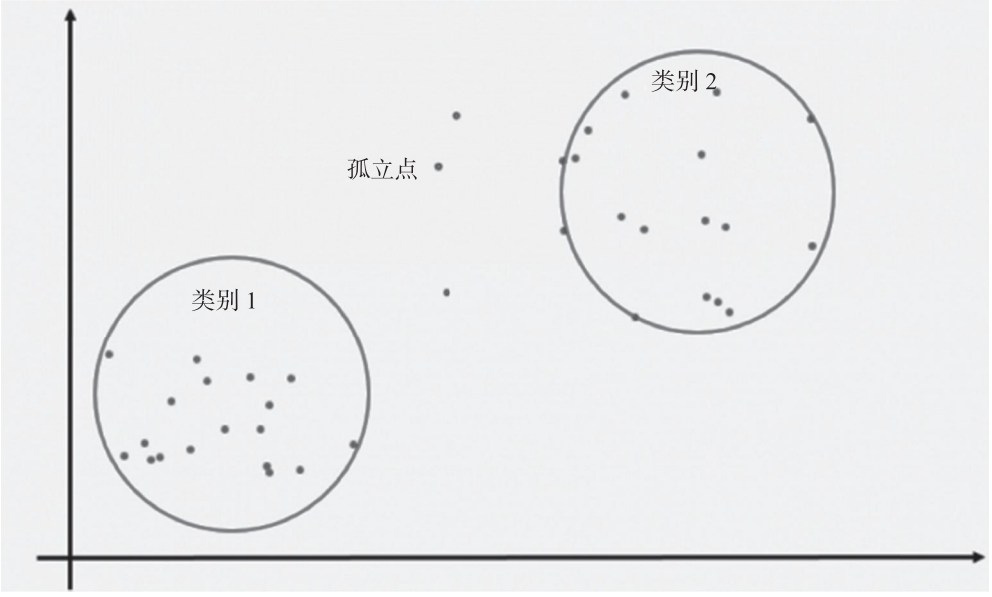


图 1-1　基于聚类的孤立点识别

对于表 1-1 中的数据，清洗数据时首先将数据集等分为 2403 个区间，找到数据的集中域 [0，3266]。然后利用分箱法对取值在 [0，3266] 的数据做进一步分析，对新数据组剔除离群值，得到清洗后的离群数据组。最后统计计算清洗后的目标数据源的平均下载时长为 192.93s，约 3.22min，符合网站文件下载的实际情况。从这个简单的例子可看出，基于数据的分布特征，数据清洗可以采用分箱法或聚类方法较快捷地识别和剔除数据集中的噪声数据，从而获得良好的清洗效果。

对于普遍的数据传输和存储，数据去重（data deduplication）技术是一种专用的数据压缩技术，用于消除重复的数据。在数据去重过程中，一个唯一的数据块或数据段将分配一个标识并存储，该标识会加入一个标识列表。当去重过程继续时，一个标识已存在于标识列表中的新数据块将被认为是冗余的块。该数据块将被一个指向已存储数据块指针的引用替代。通过这种方式，任何给定的数据块只有一个实例存在。去重技术能够显著地减少存储空间，对大数据存储系统具有非常重要的作用。

1.2 数据标准化

笔记

1.2.1 数据标准化概念

数据标准化 / 规范化（data standardization/normalization）是机构或组织对数据的定义、组织、分类、记录、编码、监督和保护进行标准化的过程，有利于数据的共享和管理，可以节省费用，提高数据的使用效率和可用性。在进行数据分析之前，通常需要先将数据标准化，利用标准化后的数据进行数据分析和处理。数据标准化处理主要包括数据同趋化处理和无量纲化处理两个方面。数据同趋化处理主要解决不同性质的数据问题，对不同性质的指标直接求和不能正确反映不同作用力的综合结果，必须先考虑改变逆指标数据性质，使所有指标对测评方案的作用力同趋化，然后再求和才能得出正确结果。数据无量纲化处理主要用于消除变量间的量纲关系，解决数据评价分析中数据的可比性。

例如，多指标综合评价方法需要把评价对象不同方面的多个描述信息综合起来，得到一个综合指标，由此对评价对象做整体评判，并进行横向或纵向比较。而在多指标评价体系中，由于各评价指标的性质不同，因此通常具有不同的量纲和数量级。当各指标间的水平相差很大时，如果直接用原始指标值进行分析，就会突出数值水平较高的指标在综合分析中的作用，相对削弱数值水平较低指标的作用。因此，为了保证结果的可靠性，需要对原始指标数据进行标准化处理。

1.2.2 数据标准化的常用方法

1. max-min 标准化

max-min（最大 - 最小）标准化方法也称为离差标准化，是对原始数据进行线性变换的方法。设 minA 和 maxA 分别为属性 A 的最小值和最大值，将 A 的一个原始值 x 通过 max-min 标准化映射成在区间 [0，1] 中的值 x'，其公式为 $x'=(x-\text{minA})/(\text{maxA}-\text{minA})$。

2. z-score 标准化

z-score（标准分数）标准化方法基于原始数据的均值（mean）和标准差（standard deviation）进行数据的标准化，将 A 的原始值 x 标准化到 x'，其公式为 $x'=(x-\text{mean})/\text{standard deviation}$。z-score 方法适用于属性 A 的最大值和最小值未知的情况，或有超出取值范围的离群数据的情况。

3. decimal scaling 标准化

decimal scaling（小数定标）标准化方法通过移动数据的小数点位置进行标准化。小数点移动多少位取决于属性 A 的取值中的最大绝对值。使用 decimal scaling 标准化将属性 A 的原始值 x 标准化到 x' 的公式为 $x'=x/10^j$。其中，j 是满足条件的最小整数。

举一个例子，假定 A 的取值范围是 –986~917，则 A 的最大绝对值为 986。使用小数定标标准化，即用每个值除以 1000（即 j=3），这样，–986 就被标准化为 –0.986。

4. 其他标准化方法

还有一些标准化方法的做法是将原始数据除以某一值，如将原始数据除以行或列的

笔记

和，称为总和标准化；如将原始数据除以每行或每列中的最大值，则称为最大值标准化；如将原始数据除以行或列的和的平方根，则称为模标准化（norm standardization）。

需要指出的是，在选择标准化方法之前必须充分了解数据特征，然后再决定使用恰当的标准化方法。如果需要保证性能且数据量很大时，就不适合采用 min-max 或 z-score，因为它们都需要先遍历所有数据，找出极值或均值后才能进行标准化。

1.3 数据仓库简介

1.3.1 数据仓库的定义

人类正处在信息“爆炸”时代，面对浩如烟海的数据，怎么组织和存储数据，才能使人们从各种各样巨量的数据集中快速高效地获取所需的信息，成为亟待解决的问题。数据仓库与数据挖掘的出现为人们解决这些问题带来新的有效途径。数据仓库（data warehouse，DW）是因信息系统业务发展需要，基于传统数据库系统技术发展形成并逐步独立出的一系列新的应用技术，目标是通过提供全面、大量的数据存储有效支持高层决策分析。当前，数据仓库已成为一种能提供重要战略信息的新范例，在金融、零售、公共事务、航空和制造业等多个行业发挥着重要作用。

1988 年，为解决企业集成问题，IBM 公司的研究员 Barry Devlin 和 Paul Murphy 创造性地提出了一个新的术语——数据仓库。之后，IT 厂商开始构建实验性的数据仓库。1991 年，W. H. Inmon 出版了著作《建立数据仓库》，使得数据仓库真正开始应用。W. H. Inmon 在书中对数据仓库的定义是：数据仓库是决策支持系统和联机分析应用数据源的结构化数据环境，是一个面向主题的（subject oriented）、集成的（integrated）、相对稳定的（non-volatile）、反映历史变化（time variant）的数据集合，用于支持经营管理中的决策制定过程。对于数据仓库而言，主题是一个在较高层次上将数据归类的标准，每个主题对应一个宏观的分析领域。数据仓库的集成特性是指在数据进入数据仓库之前，必须经过数据加工和集成，这是建立数据仓库的关键步骤。数据仓库的稳定性是指数据仓库反映的是历史数据，而不是日常事务处理产生的数据，数据经加工和集成进入数据仓库后是极少或根本不修改的。此外，数据仓库是不同时间的数据集合，它要求数据仓库中的数据保存时限能满足进行决策分析的需要，而且数据仓库中的数据都要标识该数据的历史时期。

相比来说，数据库是面向事务设计的，而数据仓库是面向主题设计的。数据库设计是尽量避免冗余，一般需要符合范式的规则；数据仓库设计是引入冗余，采用反范式的方式。数据库是为捕获数据而设计，数据仓库是为分析数据而设计。数据库一般存储在线交易数据，数据仓库一般存储的是历史数据。

1.3.2 数据仓库的组成要素

数据仓库不是一种提供战略信息的软件或硬件产品，而是一个便于用户找到战略信息和做出更好决策的计算环境，是一个以用户为中心的环境。数据仓库需要提供数据抽取、

数据转换、数据装载和数据存储功能，并为用户提供交互接口。典型数据仓库的基本组成要素包括源数据单元、数据准备单元、数据存储单元、信息传递单元、元数据单元和管理控制单元。

笔记

1. 源数据单元

数据仓库的源数据单元主要包括 4 种类别的数据，分别是来源于企业各种操作系统的生产数据、企业的内部数据、定期保存的存档数据和企业的外部数据。

2. 数据准备单元

在从不同数据来源得到数据之后，需要对数据进行必要的检查、修正和转换，以便于数据仓库存储、查询和分析数据。通常，数据准备单元需要经历 3 个数据处理阶段，即数据抽取、数据转换和数据装载，也就是后面要讲的 ETL 过程。

3. 数据存储单元

数据仓库的数据存储单元是一个相对独立的部分。传统的数据库系统通常为在线的交易处理应用程序提供数据支持，并且数据以适合快速查询和处理的数据格式加以存储。而在数据仓库中，需要存储大量的历史数据供数据分析使用。在传统数据库中，当发生交易时就要进行数据的更新，这使得数据库中的数据随时可能改变。与此不同，数据分析人员对数据仓库中的数据进行分析处理时，需要稳定可靠的数据，并且数据应能反映过去某个特定时段的情况。此外，数据仓库中的数据库必须是开放的。根据不同的需要，用户可以使用多个商家提供的工具。当前，大多数据仓库都采用关系数据库管理系统。

4. 信息传递单元

为了向数据仓库的各类用户提供适当的信息，信息传递单元包含了多种信息传递方式，包括在线实时传递、内部网传输、外部网传递和电子邮件等。例如，信息传递单元可以为新手和临时用户提供定制的数据报表和查询，为商业专业分析人员和高级用户提供复杂查询、多维分析和统计等功能。此外，数据仓库还可为数据挖掘应用程序提供所需的数据。

5. 元数据单元

数据仓库的元数据与数据库管理系统中的数据字典类似。在数据字典中，保存了逻辑数据结构、文件地址以及索引等信息，包含的是关于数据库中数据本身信息的数据。同样，元数据是描述数据仓库中数据本身信息的数据，其作用相当于电话黄页，可以看成描述数据仓库内容的一本字典。

6. 管理控制单元

管理控制单元负责管理和协调数据仓库中的各项服务和行动，如可以控制数据抽取、转换和装载的行动，协调向用户传递的信息。管理控制单元可以与数据库管理系统协同工作，确保数据得到正确存储，并可以监视数据的整个流通过程。

1.3.3 数据仓库的分类

根据企业构建数据仓库的主要应用场景，可以将数据仓库分为以下 4 种类型，每种类型的数据仓库系统都有不同的技术指标与要求。

笔记

1. 传统数据仓库

企业把数据分成内部数据和外部数据。内部数据包括 OLTP 交易系统和 OLAP 分析系统的数据。企业首先需要将这些数据集中起来，经过转换放到这类数据库中，如 Teradata、Oracle 和 DB2 等，然后在数据库上对数据进行加工，建立各种主题模型，再提供报表分析业务。一般来说，数据的处理和加工是通过离线的批处理完成的，通过各种应用模型实现具体的报表加工。

2. 实时处理数据仓库

随着业务的发展，企业客户需要对实时的数据做一些商业分析。例如，零售行业需要根据实时的销售数据调整库存和生产计划，电力企业需要处理实时的传感器数据排查故障，以保障电力的生产等。此类行业用户对数据的实时性要求很高，传统的离线批处理的方式不能满足需求，因此需要构建实时处理的数据仓库。数据可以通过各种方式完成采集，然后数据仓库可以在指定的时间内对数据进行处理和统计分析等，再将数据存入数据仓库，以满足其他业务的需求。

3. 关联发现数据仓库

在一些场景下，企业可能不知道数据的内联规则，需要通过数据挖掘的方式找出数据之间的关联关系、隐藏的联系和模式等，从而挖掘出数据的价值。很多行业的新业务都有这方面的需求，如金融行业的风险控制、反欺诈等业务。上下文无关联的数据仓库一般需要在架构设计上支持数据挖掘，并提供通用的算法接口操作数据。

4. 数据集市

数据集市一般是用于某一类功能需求的数据仓库的简单模式，往往由一些业务部门构建，也可以构建在企业数据仓库上。一般来说，数据集市的数据源较少，但对数据分析的延时有很高的要求，并需要与各种报表工具很好地对接。

1.3.4 数据仓库的相关技术

1. 数据清洗

数据仓库需要从种类各异的多个数据源中导入大量数据，这些数据存在字段的含义不同、量纲不统一、字段长度不一致、同一对象在不同数据源中的表示各异等数据质量问题，从而影响决策分析结果的正确性。因此，设计数据仓库的一个重要任务就是通过数据清洗保证数据的一致性与正确性。

2. 数据粒度

数据粒度指数据仓库中保存数据的细化或综合程度。数据仓库中存储的数据粒度将直接影响到数据仓库中数据的存储质量及查询质量，并进一步影响数据仓库能否满足最终用户的分析需求。数据仓库中的数据粒度大致可划分为详细数据、轻度抽象和高度抽象 3 级。显然，数据粒度越小，其细化程度越高。设计数据仓库时，首先要合理确定数据粒度。为了合理确定数据粒度，首先需要对数据的记录数和数据仓库的磁盘空间进行估算，然后根据历史经验选择合适的粒度水平，并根据用户的需求变化动态调整。

笔记

3. 索引优化

不论是数据库，还是数据仓库，索引查找都是优化查询响应时间的重要方法，索引建立的好坏直接影响数据访问效率。因此，为了提高数据仓库的处理能力，需要合理使用索引技术。在数据仓库环境下，位图索引优于 B 树索引，但随着基数的增加，位图索引存在不可克服的缺点。如何高效地建立数据仓库的索引，提高查询性能，从整体上使系统得到优化，是需要进一步研究和探讨的问题。

4. 物化视图选择和维护

数据仓库中存储了大量来自多个异质数据源中的数据，这些数据在数据仓库中以物化视图（materialized view）的形式存在。物化视图的选择和维护策略是数据仓库研究的重要问题之一。数据仓库中采用物化视图进行快速查询和分析，能有效提高查询速度和响应时间。此外，当数据源因元素的插入、删除和更新发生变化时，物化视图必须做相应更新，以保证查询结果的正确性。

5. 数据仓库的管理与维护

大型数据库中存储着海量数据，一般可达 TB 级，并且存储的数据生命周期也较长，对数据的更新和维护提出了更高要求。为了减少数据更新量，数据仓库一般采用增量式更新策略。数据仓库维护的关键是如何从局部抽取数据并将抽取的数据转换为全局物化视图。此外，数据仓库的安全性涉及企业的绝大部分数据，必须建立有效的安全策略和授权访问控制机制。最后，数据仓库必须提供稳定可靠的数据备份和恢复策略，以避免造成无法挽回的损失。

1.3.5 常用工具介绍

数据仓库以关系数据库为依托，以数据仓库理论为指导，以在线分析处理（OLAP）为主要应用，以 ETL 工具进行数据集成、整合、清洗、加载转换，以前端工具进行前端报表展现浏览，目标是为达到整合企业信息，把数据转换成信息和知识，提供科学决策支持。

数据仓库不只是一门纯粹的技术，更是一种架构和理念，核心在于对数据的整合集成，把企业原始数据进行集成、归类、分析，从而提供企业决策分析需要的目标数据。数据库和数据仓库从物理设计角度应该是一致的，都是基于传统的关系数据库理论，而且两者有融合的趋势。SQL Server、Sybase、DB2 和 Oracle 都是传统的关系数据库，但是，只要经过合理的数据模型设计或参数设置，也可将其转变为很好的数据仓库实体。与此同时，数据仓库也在不断地发展演变之中。例如，SybaseIQ 和 Teradata 是典型的数据仓库，但它们不适于设计 OLTP 系统。

目前，OLAP 已逐渐融合到数据仓库中，例如微软的 Analysis Service 和 DB2 的 OLAP Server，通过自身提供的专用接口可以加快多维数据的转换处理。当然，也有如 Essbase 这样纯粹的 OLAP 产品，实际上许多大型 OLAP 都采用 Essbase。

对于 ETL 而言，市场上广泛使用的 ETL 工具主要包括 Informatica PowerCenter、IBM 的 DataStage、SQL Server 搭配的 SSIS、Oracle 的 OWB 和 ODI，以及开源的 Kettle 等。

数据仓库可用的报表工具很多，专业性的报表工具有 Hyperion、BO、Congos 和

笔记

Brio，这些产品的价格相对昂贵。便宜的报表工具有微软的 ReportServices。

来自多个异构数据源的数据首先经过提取、检验、整理、加工和组织，然后存放到数据仓库的数据库中。接下来，为了方便用户（业务决策人员、各级管理人员和业务分析人员）灵活使用数据仓库中的数据，数据仓库还应为用户提供一套前端数据访问和分析工具。目前市场上能获得的数据访问和分析工具种类繁多，主要有关系型查询工具、数据多维视图工具和 DSS（决策支持系统）工具等。

第 2 章 数据类型与清洗

学习目标

1. 了解数据的基本格式。
2. 了解数据的基本类型。
3. 掌握数据转换原理。
4. 掌握数据清洗过程。

知识导图

笔记

本章导读

为了便于使用，容易记忆，常常要对计算机加工处理的对象进行编码，用一个编码符号代表一条信息或一串数据，这就是数据编码。常用的编码方案有单极性码、极性码、双极性码、归零码、双相码、不归零码、曼彻斯特编码、差分曼彻斯特编码、多电平编码、4B/5B 编码等。但是，由于调查、编码和录入误差，数据中可能存在一些无效值或缺失值，需要给予适当的处理。常用的处理方法有估算、整例删除、变量删除和成对删除，采用不同的处理方法可能会对分析结果产生影响，尤其是当缺失值的出现并非随机且变量之间明显相关时。因此，在调查中应当尽量避免出现无效值和缺失值，保证数据的完整性，数据清洗从名字上看就是把“脏”的“洗掉”，指发现并纠正数据文件中可识别的错误的最后一道程序，包括检查数据一致性，处理无效值和缺失值等。因为数据仓库中的数据是面向某一主题的数据的集合，这些数据从多个业务系统中抽取而来而且包含历史数据，这样就避免不了有的数据是错误数据、有的数据相互之间有冲突，这些错误的或有冲突的数据显然是我们不想要的，因此称为“脏”数据，简单说，不干净的数据会导致分析过程中的错误以及结果的错误，所以数据格式、类型、编码就显得尤为重要。

2.1 数据类型

在计算机中广义存在的两种文件类型是文本文件和二进制文件；简单来说，平时我们能看懂的记事本、表格等都是文本文件，计算机能读懂但由非人类可读字符组成的文件是二进制文件。

2.1.1 文件格式

文件格式（或文件类型）是指计算机为了存储信息而使用的对信息的特殊编码方式，用于识别内部存储的资料。比如有的存储图片，有的存储程序，有的存储文字信息。每类信息都可以一种或多种文件格式保存在计算机存储中。每种文件格式通常会有一种或多种扩展名可以用来识别，但也可能没有扩展名。扩展名可以帮助应用程序识别文件的格式。

对于硬盘机或任何计算机存储来说，有效的信息只有 0 和 1 两种。所以计算机必须设计相应的方式进行信息—位元的转换。对于不同的信息，有不同的存储格式。

在网上收集数据的时候可能会遇到以下几种情况：

（1）数据可以以文件的形式下载。

（2）数据可以通过交互界面访问，比如利用查询接口访问数据库系统。

（3）数据通过持续不断的流的形式进行访问。

（4）通过应用程序接口（API）访问。

1. 文本文件格式

文本文件格式是一种由若干行字符构成的计算机文件。文本文件存在于计算机文件系统中。通常，通过在文本文件最后一行放置文件结束标志指明文件结束。文本文件是指一

种容器，而纯文本是指一种内容。文本文件可以包含纯文本。一般来说，计算机文件可以分为文本文件和二进制文件两类。

笔记

最常见的文本文件类型主要有三种：分隔格式（结构化数据）、JSON 格式（半结构化数据）、HTML 格式（非结构化数据）。

1）分隔格式

分隔文件就是文本格式文件，这种文件的行和列由统一的符号分隔；分隔的字符叫作分隔符。最常见的分隔符就是制表符和逗号，这两种方案分别出现在制表符分隔值（TSV）和逗号分隔值（CSV）中。

（1）查看不可见的分隔字符。分隔文件中的换行符、回车符等都是不可见的，如何让它们可见呢？比如用 notepad++ 打开的文件，要显示分隔符可以这么操作：视图 --> 显示符号 --> 在显示所有字符处打勾；其他的软件文本行都可以找到各自的方式查看。

（2）封闭错误数据。举一个例子：在一个以“ ,”为分隔符的分隔文件中，如果有一个工资数据的格式为“76，888”，这个逗号是工资中的千位分隔符，但在此分隔文件中，逗号又是分隔符，因此引起错误；处理方案主要有以下两种：

①创建分隔文件时检查确保没有引起歧义的逗号（也就是说，删掉上面工资中的逗号，不使用那种格式的工资）。

②使用额外的符号对数据进行封闭处理（比如上面的工资数据用引号包含，将里面的逗号封闭起来）。

（3）字符转义。如果字符本身就含有引号，那么此时再用引号进行封闭显然不合适；也许你会说可以用单引号、双引号进行区别，但是单引号在英文中又与名词所有格冲突，因此也不算最好的办法，遇到这种问题最好的解决办法就是使用转义字符：比如“ light \"red blue\" ”，这样，通过反斜杠 \ 对双引号进行转义就不会与外面的双引号有冲突了。

2）JSON 格式

JSON 格式的数据是当前较为流行的数据格式，它是一个半结构化的数据，使用键值对的字典类型格式，数据的顺序无关紧要，还可以缺失某些值，它还支持多层级结构和多值属性。

3）HTML 格式

HTML 文件是一种网页文件，是非格式化的；里面包含很多冗余的数据，针对这种数据目前也有很多提取办法，网络信息爬虫就是专门针对这种网页数据爬取的，各式各样的提取规则以后会涉及。

2. 归档文件

归档文件就是内部包含文本文件或二进制文件等许多文件的独立文件。磁带归档（TAR）文件通常以 .tar 为后缀名，这种文件一般只归档并不会压缩；通常在 Windows 系统下直接使用相应的软件就可以归档了，而在 Linux 系统下则需要相应命令：tar cvf name.tar name1.csv name2.csv（创建归档文件）、tar xvf name.tar（打开归档文件）

3. 压缩、解压缩文件

1）压缩文件的概念

压缩文件其实与归档文件类似，只是后缀名不同。一般地，ZIP、RAR 文件既归

笔记

档，又压缩。在 Windows 下压缩、解压缩都很方便，主要利用程序进行压缩、解压缩，简单说，就是经过压缩软件压缩的文件叫压缩文件。压缩的原理是把文件的二进制代码压缩，减少相邻的 0，1 代码，比如 000000，可以把它变成 6 个 0 的写法 60，以减少该文件的空间。压缩文件的基本原理是查找文件内的重复字节，建立一个相同字节的“词典”文件，并用一个代码表示。比如，文件里多次出现一个相同的词“中华人民共和国”，这时若用一个代码表示这个词并将其写入“词典”文件，就可以达到缩小文件的目的。

2）压缩的原理

由于计算机处理的信息是以二进制数的形式表示的，因此压缩软件就是把二进制信息中相同的字符串以特殊字符标记来达到压缩的目的。为了有助于理解文件压缩，请在脑海里想象一幅有蓝天、白云的图片。对于成千上万单调重复的蓝色像点而言，与其一个一个定义“蓝、蓝、蓝……”长长的一串颜色，还不如告诉计算机“从这个位置开始存储 1117 个蓝色像点”简洁，而且还能大大节约存储空间。这是一个非常简单的图像压缩的例子。其实，所有的计算机文件都是以“1”和“0”的形式存储的，与蓝色像点一样，只要通过合理的数学计算公式，文件的体积都能够被大大压缩，以达到“数据无损稠密”的效果。总的来说，压缩可以分为有损和无损两种。如果丢失个别的数据不会造成太大的影响，这时忽略它们是一个好主意，这就是有损压缩。有损压缩广泛应用于动画、声音和图像文件中，典型的代表就是影碟文件格式 mpeg、音乐文件格式 mp3 和图像文件格式 jpg。但是，更多情况下压缩数据必须准确无误，人们便设计出无损压缩格式，比如常见的 zip、rar 等。压缩软件（compression software）自然就是利用压缩原理压缩数据的工具，压缩后生成的文件称为压缩包（archive），体积只有原来的几分之一，甚至更小。当然，压缩包已经是另一种文件格式了，如果想使用其中的数据，首先得用压缩软件把数据还原，这个过程称为解压缩。常见的压缩软件有 winzip、winrar 等。

有两种形式的重复存在于计算机数据中，zip 就对这两种重复进行了压缩。

（1）短语形式的重复，即三个字节以上的重复，对于这种重复，zip 用两个数字：1. 重复位置距当前压缩位置的距离；2. 重复的长度，用来表示这个重复，假设这两个数字各占一字节，于是数据便得到了压缩，这很容易理解。[一字节有 0~255 共 256 种可能的取值，三字节有 256 × 256 × 256 共 1600 多万种可能的情况，更长的短语取值的可能情况以指数方式增长，出现重复的概率似乎极低，实则不然，各种类型的数据都有出现重复的倾向，一篇论文中，为数不多的术语倾向于重复出现；一篇小说，人名和地名会重复出现；一张上下渐变的背景图片，水平方向上的像素会重复出现；程序的源文件中，语法关键字会重复出现（写程序时，多次复制、粘贴），以几十 K 为单位的非压缩格式的数据中，倾向于大量出现短语式的重复。经过上面提到的方式进行压缩后，短语式重复的倾向被完全破坏，所以在压缩的结果上进行第二次短语式压缩一般是没有效果的。

（2）单字节的重复，一个字节只有 256 种可能的取值，所以这种重复是必然的。其中，某些字节出现次数可能较多，另一些则较少，在统计上有分布不均匀的倾向，这是

容易理解的，比如一个 ASCII 文本文件中，某些符号可能很少用到，而字母和数字则使用较多，各字母的使用频率也是不一样的，据说字母 e 的使用概率最高；许多图片呈现深色调或浅色调，深色（或浅色）的像素使用较多（这里顺便提一下：png 图片格式是一种无损压缩，其核心算法就是 zip 算法，它和 zip 格式的文件的主要区别在于：作为一种图片格式，它在文件头处存放了图片的大小、使用的颜色数等信息）；上面提到的短语式压缩的结果也有这种倾向：重复倾向于出现在离当前压缩位置较近的地方，重复长度倾向于比较短（20 字节以内）。这样就有了压缩的可能：给 256 种字节取值重新编码，使出现较多的字节使用较短的编码，出现较少的字节使用较长的编码，这样，变短的字节相对于变长的字节更多，文件的总长度就会减少，并且字节使用比例越不均匀，压缩比例越大。

压缩、解压的格式有非常多的选择，影响选择压缩、解压格式主要有以下三个关键因素：

①压缩和解压缩的速度。

②压缩比率（文件到底能缩减多少）。

③压缩方案的互操作性。

3）无损压缩

如果从互联网上下载了许多程序和文件，可能会遇到很多 ZIP 文件。这种压缩机制是一种很方便的发明，尤其是对网络用户，因为它可以减小文件中的比特和字节总数，使文件能够通过较慢的互联网连接，实现更快的传输，此外还可以减少文件的磁盘占用空间。下载文件后，计算机可使用 WinZip 或 Stuffit 这样的程序展开文件，将其复原到原始大小。如果一切正常，展开的文件与压缩前的原始文件将完全相同。

大多数计算机文件类型都包含相当多的冗余内容——它们会反复列出一些相同的信息。文件压缩程序就是要消除这种冗余现象。与反复列出某一块信息不同，文件压缩程序只列出该信息一次，然后当它在原始程序中出现时再重新引用它。

4）有损压缩

我们将压缩类型称为无损压缩，因为重新创建的文件与原始文件完全相同。所有无损压缩都基于这样一种理念：将文件变为“较小”的形式，以利于传输或存储，并在另一方收到它后复原，以便重新使用它。

有损压缩则与此大不相同。这些程序直接去除“不必要”的信息，对文件进行剪裁，以使它变得更小。这种类型的压缩大量应用于减小位图图像的文件大小，因为位图图像的体积通常非常庞大。为了了解有损压缩的工作原理，让我们看看计算机是如何对一张扫描的照片进行压缩的。

对于此类文件，无损压缩程序的压缩率通常不高。尽管图片的大部分看起来都是相同的——例如，整个天空都是蓝色的——但是大部分像素之间都存在微小的差异。为了使图片变得更小，同时不降低其分辨率，必须更改某些像素的颜色值。如果图片中包含大量的蓝色天空，程序会挑选一种能够用于所有像素的蓝色。然后，程序重写该文件，所有天空像素的值都使用此信息。如果压缩方案选择得当，就不会注意到任何变化，但是文件大小

笔记

会显著减小。

当然，对于有损压缩，在文件压缩后您无法将其复原成原始文件的样子。必须接受压缩程序对原始文件的重新解释。因此，如果需要完全重现原来的内容（如软件应用程序、数据库等），则不应该使用这种压缩形式。

2.1.2 数据损耗

由于数据类型其他地方介绍得非常多而且较详细，这里就不做介绍了。在数据处理上涉及最多的数据类型为数字、日期、字符串。

数据转换在数据清洗过程中必不可少，但是在介绍具体的转换之前，我们先介绍一下转换过程中数据损耗的问题。

数据损耗有时会发生，有时不会发生，有时的数据损耗是允许的，有时却不允许，这就要视具体情况具体对待。一般损耗的风险因素有以下两个：

（1）同种类型间不同范围的转换。比如 200 长的字符串转换到 100 长的字符串上，超过长度的都会被截掉；最容易忽略的就是数字类型，比如长整型转换为整型，等等。

（2）不同精度间的转换。比如原本四位精度的数字转为两位精度，多余的精度信息将会被舍弃，造成数据的丢失。

2.1.3 数据类型间相互转换的策略

1. SQL 级别的类型转换

1）将 MySQL 数据格式化为字符串

比如从数据库中查到一条日期数据 2020-01-21 04:51:00，我们希望得到 4:51am，Friday，January 21，2020，代码如下：

```
select concat(
   concat(hour(date),':',minute(date)),
   if (hour(date) < 12, 'am', 'pm'),
   concat(', ', dayname(date), ', ', monthname(date), ' ', day(date), ', ', year(date))
)
from table_name where mid=21;
或者使用 date_format() 函数：
select date_format(date, '%1:%i%p, %W, %M %e, %Y') from tbl_name where mid=21;
```

用这个语句查出的唯一不同是 AM 是大写的，只将 am 那一部分拆开转化为小写再用 concat( ) 函数链接即可。

2）从字符串类型转换到 MySQL 日期类型

比如有一段字符串“ ....>....>.....>sent：Thursday，August 17，2020 6:29 PM>....”，我们要提取里面的日期并转化为 MySQL 的 datetime 类型，代码如下：

```
select str_to_date(
   substring_index(
      substring_index(reference, '>', 3),
      'sent:',
      -1),
   '%W, %M，%e, %Y, %h:%i %p')
from referenceinfo where mid =22;
```

substring_index( ) 函数按指定字符对字符串进行分隔，并得到指定位置的字符串；

str_to_date( ) 函数将字符串格式的日期转化为 datetime 类型。

3 ）将字符串类型转换数据转化为小数

比如有某个字符串“…$18.47/bbl…”，现在需要提取字符串里的数字并转化为小数，代码如下：

```
select convert(substring_index(
    substring(body,locate('$', body) + 1),
   '/bbl',
   1),
decimal(4,2)
)
as price from tbl_name where body like '%$%' and body like '%/bbl%' ;
```

substring( ) 函数用于定位，substring_index( ) 函数用于分隔，convert( ) 函数用于将字符转换为数字。

2. 文件级别的类型转换

下面以 Excel 中的类型转换为例说明。

Excel 中的类型转换一般可以直接设置单元格格式，修改数据的类型；也可以使用函数判断，例如 "=istext(A2)" 就可以判断 A2 中的内容是不是文本，若是，返回 TRUE；若不是，则返回 FALSE。同样，isnumber( ) 函数也是如此。

另外，还有这样的转换方式 =TEXT（A4，"yyyy-mm-dd"），这样就可以直接将数字类型的日期转换为字符串类型。

2.1.4 0 值、空值与 null 值的处理

在数据处理时有些是我们不需要的数据，比如 0 值、空值和 null 值，这 3 种值不进行处理往往会导致一些错误，并且这 3 种值是不同的，在具体的情景下处理方式不一。

1. 0 值

0 值是最容易处理的空值，因为它具有一定的意义；可以拿它进行排序、比较、数学运算（不能当除数）；有了这些基础，0 值处理起来就很容易了。

2. 空值

与 0 值相比，空值的处理要难一些，因为它在不同的情况下产生的含义不同，比如“两个引号紧挨着，这时可以说它为空值”“两个引号之间有一个空格，这时可以说它是空格”这些问题。

笔记

笔记

3. null

null 不等于任何值，甚至它本身。只有在不希望出现任何数据的情况下才使用 null。一般地，对 null 值的处理，通常是用指定的值（一般用 0）替换，不过，对这种替换应做好规划，使它有利于接下来的数据处理，并做好记录，以免以后遗忘。

知识拓展

字符编码

不管是数据库里的数据，还是文本文件中的数据，存储时都采用了一定的编码、解码格式，当使用不同的编码格式对文件或数据库内容进行读取时，就会产生编码冲突，因此要通过调整编码使之一致，或使用向下兼容的编码类型。

2.2 数据转换

2.2.1 将电子表格转换为 CSV 类型

将电子表格转换为 CSV 类型相对比较简单，基本上用软件打开电子表格后选择“另存为”就可以定义另存文件的格式和编码了，这是比较简单快捷的。不过，也有一些地方需要注意：

在另存为 CSV 文件时，只有当前工作表中的内容会被保存，这是因为 CSV 文件只能描述一组数据集，如果你的电子表格里有多个工作表，就需要分别单独存为 CSV 文件。

2.2.2 将电子表格转换为 JSON

电子表格转换为 JSON 数据稍微麻烦一点，但也有很多方法可解决。比如 Excel 中可以下载 Office 中的应用工具 excel-to-json ，这个工具可以将 Excel 中的数据转换为 JSON 格式；还可以使用在线转换的方式，通过 http://www.bejson.com/json/col2json/，只要将电子表格中的数据复制到在线转换的框中就可以得到 JSON 数据；另外，有种专门用于转换表格为 JSON 的小工具，使用起来也很方便。

2.2.3 将数据库中的数据转换为 CSV 或 JSON

（1）使用 MySQL 的命令行输出 CSV 文件，但这种方式无法输出为 JSON 格式的数据，代码如下：

```
select concat(firstname, " ", lastname) as name, email_id
into outfile 'filename.csv'
fields terminated by ',' optionally enclosed by '"'
lines terminated by '\n'
from tbl_name;
```

（2）使用工具 phpMyAdmin。phpMyAdmin 是基于 Web 的 MySQL 数据库客户端程序，它可以将一整张表的数据或查询出的结果数据直接输出为 CSV 或 JSON 格式的文件。具体就不演示了，安装好后稍微研究一下就会使用了。

笔记

2.2.4 使用 Python 实现数据转换

（1）使用 Python 实现 CSV 到 JSON 的转换。用程序方式的转换方式有多种，最简单的、能想象到的是使用内置的 CSV 和 JSON 库，代码如下：

```
import json
import csv

# 读取 CSV 文件
with open('filename.csv') as file:
    file_csv = csv.DictReader(file)
    output = '['
    # 处理每个目录
    for row in file_csv:
        output += json.dumps(row) + ','
    output = output.rstrip(',') + ']'

# 把文件写入磁盘
f = open('filename.json', 'w')
f.write(output)
f.close()
```

提示

还可以使用 Python 工具包的 csvkit 库实现。

（2）使用 Python 实现 JSON 到 CSV 的转换。读取 JSON 文件并转换为 CSV 的代码如下：

```
import json
import csv

with open('filename.json', 'r') as f:
    dicts = json.load(f)

out = open('filename.csv', 'w')
writer = csv.DictWriter(out, dicts[0].keys())
writer.writeheader()
writer.writerrows(dicts)
out.close()
```

2.3 清洗 PDF 文件中的数据

可移植文档格式（PDF）存储的文件相对较复杂，因为它是以二进制的形式存储的，格式固定，不可修改，使用起来很方便，但是里面的信息相对较难提取。下面介绍提取 PDF 中的信息的方式。

笔记

2.3.1 直接复制

有时 PDF 文件里的内容可以复制出来再整理，这对于需要从 PDF 中提取少量信息的人来说十分便利；但是，数据比较多时这样做效率低下，而且有的 PDF 根本无法复制，因此这种方式局限性很大。

2.3.2 pdfMiner3k

这个 Python 程序包使我们可以从 PDF 中提取出需要的信息。它自带的 pdf2txt 和 dumpPDF 工具可以提取并输出相应的信息。

示例代码如下：

```
import sys
import importlib
importlib.reload(sys)

from pdfminer.pdfparser import PDFParser,PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import PDFPageAggregator
from pdfminer.layout import LTTextBoxHorizontal,LAParams
from pdfminer.pdfinterp import PDFTextExtractionNotAllowed

'''
 解析 PDF 文本，保存到 TXT 文件中
'''
path = r'../../data/pdf/ 阿里巴巴 Java 开发规范手册 .pdf'
def parse():
    fp = open(path, 'rb') # 以二进制读模式打开
    # 用文件对象创建一个 PDF 文档分析器
    praser = PDFParser(fp)
    # 创建一个 PDF 文档
    doc = PDFDocument()
    # 连接分析器与文档对象
    praser.set_document(doc)
    doc.set_parser(praser)

    # 提供初始化密码
    # 如果没有密码，就创建一个空的字符串
    doc.initialize()

    # 检测文档是否提供 TXT 转换，若不提供，就忽略
    if not doc.is_extractable:
        raise PDFTextExtractionNotAllowed
    else:
```

笔记

```
        # 创建 PDF 资源管理器来管理共享资源
        rsrcmgr = PDFResourceManager()
        # 创建一个 PDF 设备对象
        laparams = LAParams()
        device = PDFPageAggregator(rsrcmgr, laparams=laparams)
        # 创建一个 PDF 解释器对象
        interpreter = PDFPageInterpreter(rsrcmgr, device)

        # 循环遍历列表，每次处理一个 page 的内容
        for page in doc.get_pages(): # doc.get_pages() 用于获取 page 列表
            interpreter.process_page(page)
            # 接受该页面的 LTPage 对象
            layout = device.get_result()
                # 这里，layout 是一个 LTPage 对象，里面存放着这个 page 解析出的各种对象，一般包括
LTTextBox, LTFigure, LTImage, LTTextBoxHorizontal 等，想获取文本，就先获得对象的 text 属性
            for x in layout:
                if (isinstance(x, LTTextBoxHorizontal)):
                    with open(r'../../data/pdf/1.txt', 'a') as f:
                        results = x.get_text()
                        print(results)
                        f.write(results + '\n')

if __name__ == '__main__':
    parse()
```

pdfMiner3k 在提取行信息的 PDF 时比较规整，但在提取表格数据上表现得较糟糕。

2.3.3 Tabula

Tabula 工具包是专门用来提取 PDF 表格数据的，同时支持 PDF 导出为 CSV、Excel 格式。它是一个基于 Java 的程序，因此，如果要在 Python 上使用，则需要这么安装：pip install tabula-py；tabula-py 依赖库包括 java、pandas、numpy，所以需保证运行环境中安装了这些库。

示例代码如下：

```
import tabula

df = tabula.read_pdf("E:\\test\\cb7b5.pdf", encoding='gbk', pages='all')
print(df)
for indexs in df.index:
    # 遍历打印企业名称
    print(df.loc[indexs].values[1].strip())
```

笔记

2.3.4 pdf2htmlEX

pdf2htmlEX 包的作用是将 PDF 格式转换成 HTML 格式。示例代码如下：

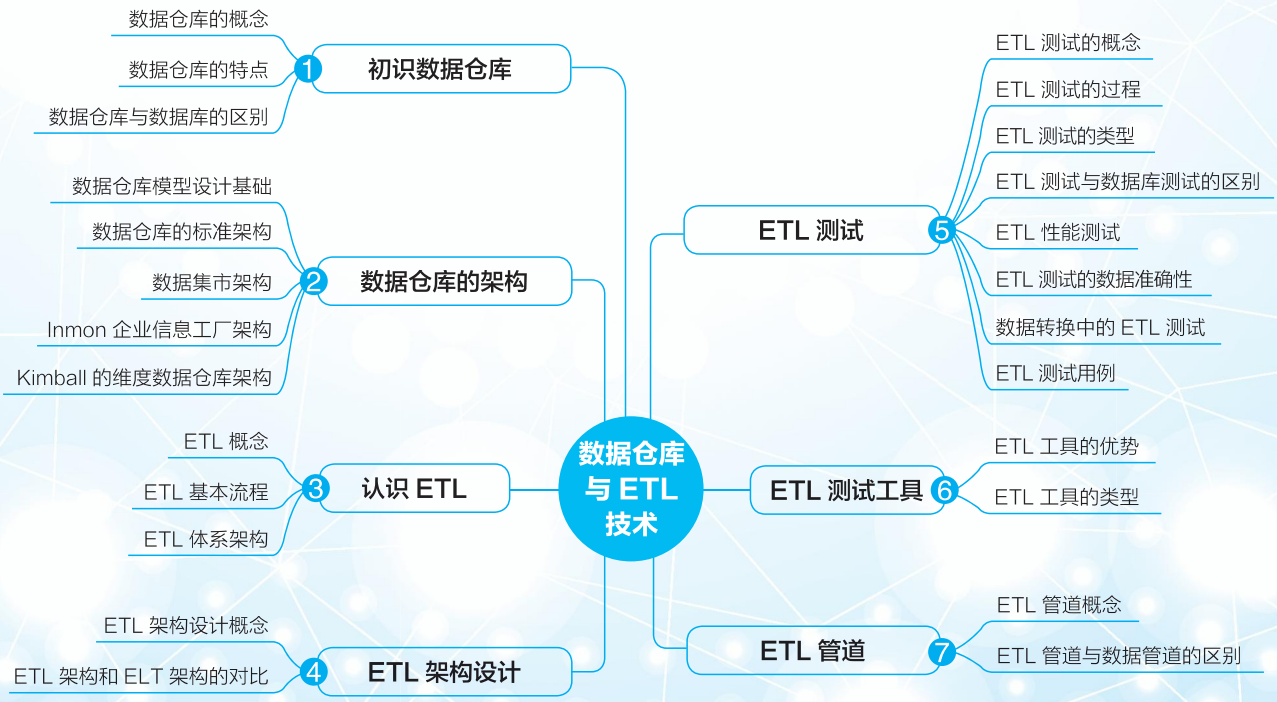
```
import subprocess

subprocess.call('"D:\Program Files (x86)\pdf2htmlEX-win32-0.14.6-upx-with-poppler-data\pdf2htmlEX.exe"
--dest-dir E:\\test\extract\\2017gq\\out  E:\\test\extractb7b5.pdf', shell=True)
```

第 3 章 数据仓库与 ETL 技术

学习目标

1. 了解数据清洗的概念。
2. 理解数据标准化的概念。
3. 掌握数据仓库的分类。

知识导图



笔记

本章导读

如何在数据仓库建设规划的前期做好业务数据准备和系统建设规划是企业在数据中心建设过程中最关心的问题。在数据仓库构建中，ETL 贯穿项目始终，它是整个数据仓库的生命线，包括从数据清洗、整合，到转换、加载等各个过程，如果说数据仓库是一座大厦，那么 ETL 就是大厦的根基，ETL 抽取整合数据的好坏直接影响最终的结果展现。所以，ETL 在整个数据仓库项目中起着十分关键的作用，必须将其摆到十分重要的位置。本章以目前数据中心系统运维、数据仓库项目建设和运维方面的一些实际工作为例，并结合一定数据仓库建设的理论知识，详细介绍关于数据建设环节中 ETL 部分的基本知识和实际应用。

3.1 初识数据仓库

3.1.1 数据仓库的概念

数据仓库（data warehouse），可简写为 DW 或 DWH，是为企业所有级别的决策制订计划过程，提供所有数据类型的战略集合。它出于分析性报告和决策支持的目的而创建，为需要业务智能的企业提供指导业务流程改进、监视时间、成本、质量以及控制。

数据仓库是一个面向主题的、集成的、随时间变化的、信息本身相对稳定的数据集合，用于支持管理决策过程。数据仓库本身并不“生产”任何数据，同时自身也不需要“消费”任何数据，数据来源于外部，并且开放给外部应用使用。

3.1.2 数据仓库的特点

1. 数据仓库是面向主题的

与传统的数据库不一样，数据仓库是面向主题的，那什么是主题呢？首页主题是一个较高层次的概念，是较高层次上企业信息系统中的数据综合、归类并进行分析的对象。在逻辑意义上，它是企业中某一个宏观分析领域所涉及的分析对象。换句说话，用户用数据仓库进行决策。一个主题通常与多个操作型系统有关，而操作型数据库的数据组织面向事务处理任务，各个任务之间是相互隔离的。

2. 数据仓库是集成的

数据仓库中的数据是从原来分散的数据库数据（MySQL 等关系型数据库）中抽取出来的。操作型数据库与 DSS 分析型数据库差别甚大。第一，数据仓库的每一个主题对应的源数据在所有各个分散的数据库中，有许多重复和不一样的地方，且来源于不同的联机系统的数据都和不同的应用逻辑捆绑在一起；第二，数据仓库中的综合数据不能从原来有的数据库系统直接得到。因此，在数据进入数据仓库之前，必然要经过统一与综合，这一步是数据仓库建设中最关键、最复杂的一步，所要完成的工作有：

（1）统计源数据中的所有矛盾，如字段的同名异议、异名同义、单位不统一、字长不统一等。

（2）进行数据的综合和计算。数据仓库中的数据综合工作可以在原有数据库抽取数据时生成，但许多是在数据仓库内部生成的，即在进入数据仓库之后进行综合而生成。

3. 数据仓库的数据是随着时间的变化而变化的

笔记

数据仓库中的数据不可更新是针对应用来说的，也就是说，数据仓库的用户进行分析处理是不进行数据更新操作的。但并不是说在从数据集成输入数据仓库开始到最后被删除的整个生存周期中，所有的数据仓库数据都是永远不变的。

数据仓库的数据是随着时间变化而变化的，这是数据仓库的特征之一。这一特征主要有以下三个表现：

（1）数据仓库随着时间变化不断增加新的数据内容。数据仓库系统必须不断捕捉 OLTP 数据库中变化的数据，追加到数据仓库中，也就是要不断生成 OLTP 数据库的快照，经统一集成增加到数据仓库中；但对于确实不再变化的数据库快照，如果捕捉到新的变化数据，则只生成一个新的数据库快照增加进去，而不会对原有的数据库快照进行修改。

（2）数据库随着时间变化不断删去旧的数据内容。数据仓库内的数据也有存储期限，一旦过了这一期限，过期数据就要被删除。只是数据库内的数据时限要远远长于操作型环境中的数据时限。在操作型环境中一般只保存 60~90 天的数据，而在数据仓库中则需要保存 5~10 年的数据，以适应 DSS 趋势分析的要求。

（3）数据仓库中包含大量的综合数据，这些综合数据中很多与时间有关，如数据经常按照时间段进行综合，或隔一定的时间进行抽样等。这些数据要随着时间的变化不断重新综合。因此，数据仓库的数据特征都包含时间项，以标明数据的历史时期。

4. 数据仓库的数据是不可修改的

数据仓库的数据主要用于企业决策分析，涉及的数据操作主要是数据查询，一般不进行修改操作。数据仓库的数据反映的是一段相当长的时间内历史数据的内容，是不同时点的数据库快照的集合，以及基于这些快照进行统计、综合和重组的导出数据，而不是联机处理的数据。数据库中进行联机处理的数据经过集成输入到数据仓库中，一旦数据仓库中存放的数据已经超过数据仓库的数据存储期限，这些数据将从当前的数据仓库中删去。因为数据仓库只进行数据查询操作，所以数据仓库中的系统要比数据库中的系统简单得多。数据库管理系统中许多技术难点，如完整性保护、并发控制等，在数据仓库的管理中几乎可以省去。但是，由于数据仓库的查询数据量往往很大，所以就对数据查询提出了更高的要求，要求采用各种复杂的索引技术；同时，数据仓库面向的是商业企业的高层管理层，他们会对数据查询的界面友好性和数据表示提出更高的要求。

总结

数据仓库就是整合多个数据源的历史数据进行细粒度的、多维的分析，可以有效地帮助高层管理者或者业务分析人员做出商业战略决策或商业报表。

3.1.3 数据仓库与数据库的区别

在 IT 的架构体系中，数据库是必须存在的，必须有地方存储数据。比如现在的网购等电商，物品的存货，货品的价格，用户账户的余额等，数据都存放在后台数据库中。最简单的理解，我们现在使用的微信、微博和 QQ 等账户和密码在后台数据库中是一个 user 表，字段至少有两个码，即用户名和密码，然后数据就一行一行地存在表中。登录时，填写用户名和密码后，这些数据会回传到后台，与表上面的数据匹配，一旦匹配成功，就能登录。若匹配不成功，就会报错，这就是数据库。数据库在生产环境中是用来干活的。凡是与业务有关应用挂钩的，都使用数据库。

数据仓库是 BI 下的一种技术。由于数据库与业务应用挂钩，所以一个数据库不可能

笔记

装下一家公司的所有数据。数据库的表设计往往是针对某一个应用进行的。比如刚刚的登录功能，这张 user 表上就只有这两个字段可以满足登录功能的需要，但是可能不满足分析的需要。比如想知道哪个时间段用户最多？哪个用户一年购物最多？诸如此类的指标，就要重新设计数据库的表结构。对于数据分析和数据挖掘，我们引入了数据仓库的概念。数据仓库的表结构是依照分析需求、分析维度、分析指标进行设计的。数据库和数据仓库的区别见表 3-1。

表 3-1　数据库和数据仓库的区别

差异项	数据库	数据仓库
特征	操作处理	信息处理
面向	事务	分析
用户	DBA、开发	经理、主管、分析人员
功能	日常操作	长期信息需求、决策支持
DB 设计	基于 ER 模型，面向应用	星形 / 雪花模型，面向主题
数据	当前的、最新的	历史的、跨时间维护
汇总	原始的、高度详细	汇总的、统一的
视图	详细、一般关系	汇总的、多维的
工作单元	短的、简单事务	复杂查询
访问	读 / 写	大多为读
关注	数据进入	信息输出
操作	主键索引操作	大量的磁盘扫描
用户数	数百到数亿	数百
DB 规模	GB 到 TB	≥ TB
优先	高性能、高可用性	高灵活性
度量	事务吞吐量	查询吞吐量、响应时间

数据库软件：是一种软件（并不是链接数据库的图形化客户端），用来实现数据库逻辑过程，属于物理层。

数据库：是一种逻辑概念，用来存放数据的仓库，通过数据库软件实现。数据库由很多表组成，表是二维的，一张表里有很多字段。字段一字排开，对数据就一行一行地写入表中。数据库中的表能够用二维表现多维的关系，如 Oracle、DB2、MySQL、Sybase、MS SQLServer 等。

数据仓库：是数据库概念的升级。从逻辑上理解，数据库和数据仓库没有区别，都是通过数据库软件存放数据的地方，只不过从数据量来说，数据仓库要比数据库庞大得多。数据仓库主要用于数据挖掘和数据分析，辅助领导做决策。

数据库与数据仓库的区别实际是 OLTP 与 OLAP 的区别，见表 3-2。

知识拓展

OLTP 与 OLAP

操作型处理也称为联机事务处理（on-line transaction processing，OLTP），也可以称面向交易的处理，它是针对具体业务在数据库联机的日常操作，通常对少数记录进行查询、修改。用户较为关心操作的响应时间、数据的安全性、完整性和并发的支持用户数等问题。传统的数据库系统作为数据管理的主要手段，主要用于操作型处理。

分析型处理也称为联机分析处理（on-line analytical processing，OLAP），一般针对某些主题历史数据进行分析，支持管理决策。

表 3-2　OLTP 与 OLAP 的区别

操作型处理	分析型处理
细节的	综合或者提炼的
实体 - 关系（E-R）模型	星形模型或雪花模型
存储瞬间数据	存储历史数据，不包含最近的数据
可更新的	只读、只追加
一次操作一个单元	一次操作一个集合
性能要求高，响应时间短	性能要求宽松
面向事务	面向分析
一次操作数据量小	支持决策需求
数据量小	数据量大
客户订单、库存水平和银行账户查询	客户收益分析、市场细分

3.2 数据仓库的架构

3.2.1 数据仓库模型设计基础

1. 关系数据模型

关系数据模型中，表设计规范化是通过应用范式规则实现的。最常用的范式有第一范式（1NF）、第二范式（2NF）、第三范式（3NF）。非规范化的员工表见表 3-3。

表 3-3　非规范化的员工表

id	name	mobile	zip	province	city	district	deptNo	deptName
101	张三	13910000001 13910000002	100001	北京	北京	海淀区	D1	部门 1
101	张三	13910000001 13910000002	100001	北京	北京	海淀区	D2	部门 2
102	李四	13910000003	200001	上海	上海	静安区	D3	部门 3
103	王五	13910000004	510001	广东省	广州	白云区	D4	部门 4

（1）第一范式表中的列只能含有原子性（不可再分）的值。表 3-3 中，张三有两个手机号存储在 mobile 列中，违反了 1NF 规则。为了使表满足 1NF，数据应该修改为表 3-4

笔记

所示的结构。

表 3-4　满足 1NF 的员工表

id	name	mobile	zip	province	city	district	deptNo	deptName
101	张三	13910000001	100001	北京	北京	海淀区	D1	部门 1
101	张三	13910000002	100001	北京	北京	海淀区	D1	部门 1
101	张三	13910000001	100001	北京	北京	海淀区	D2	部门 2
101	张三	13910000002	100001	北京	北京	海淀区	D2	部门 2
102	李四	13910000003	200001	上海	上海	静安区	D3	部门 3
103	王五	13910000004	510001	广东省	广州	白云区	D4	部门 4

（2）第二范式要同时满足下面两个条件：满足第一范式；没有部分依赖。例如，员工表的一个候选键是 {id，mobile，deptNo}，而 deptName 依赖于 {deptNo}，同样，name 仅依赖于 {id}，因此不是 2NF 的。为了满足第二范式的条件，需要将这个表拆分成 employee、dept、employee_dept、employee_mobile 四个表，见表 3-5~ 表 3-8。

表 3-5　满足 2NF 的 employee 表

id	name	zip	province	city	district
101	张三	100001	北京	北京	海淀区
102	李四	200001	上海	上海	静安区
103	王五	510001	广东省	广州	白云区

表 3-6　满足 2NF 的 dept 表

deptNo	deptName
D1	部门 1
D2	部门 2
D3	部门 3
D4	部门 4

表 3-7　满足 2NF 的 employee_dept 表

id	deptNo
101	D1
101	D2
102	D3
103	D4

表 3-8　满足 2NF 的 employee_mobile 表

id	mobile
101	13910000001
101	13910000002
102	13910000003
103	13910000004

（3）第三范式要同时满足下面两个条件：满足第二范式；没有传递依赖。例如，员工表的 province、city、district 依赖于 zip，而 zip 依赖于 {id}，换句话说，province、city、district 传递依赖于 {id}，违反了 3NF 规则。为了满足第三范式的条件，可以将这个表拆分成 employee 和 zip 两个表，见表 3-9 和表 3-10。

笔记

表 3-9　满足 3NF 的 employee 表

id	name	zip
101	张三	100001
102	李四	200001
103	王五	510001

表 3-10　满足 3NF 的 zip 表

zip	province	City	District
100001	北京	北京	海淀区
200001	上海	上海	静安区

在关系数据模型设计中，一般需要满足第三范式的要求。如果一个表有良好的主外键设计，就应该是满足 3NF 的表。规范化带来的好处是通过减少数据冗余提高更新数据的效率，同时保证数据完整性。然而，在实际应用中也要防止过度规范化的问题。规范化程度越高，划分的表越多，在查询数据时越有可能使用表连接操作。

如果连接的表过多，就会影响查询的性能。关键问题是要依据业务需求，仔细权衡数据查询和数据更新的关系，制定最适合的规范化程度。还有一点需要注意的是，不要为了遵循严格的规范化规则而修改业务需求。

2. 维度数据模型

维度数据模型简称维度模型（dimensional modeling，DM），是一套技术和概念的集合，用于数据仓库设计。不同于关系数据模型，维度模型不一定要引入关系数据库。

逻辑上相同的维度模型，可用于多种物理形式，比如维度数据库或简单的平面文件。根据数据仓库大师 Kimball 的观点，维度模型是一种趋向于支持最终用户对数据仓库进行查询的设计技术，是围绕性能和易理解性构建的。尽管关系模型对于事务处理系统表现非常出色，但它并不是面向最终用户的。

事实和维度是两个维度模型中的核心概念。事实表示对业务数据的度量，而维度是观察数据的角度。事实通常是数字类型的，可以进行聚合和计算，而维度通常是一组层次关系或描述信息，用来定义事实。例如，销售金额是一个事实，而销售时间、销售的产品、购买的顾客、商店等都是销售事实的维度。维度模型按照业务流程领域（即主题域）建立，例如进货、销售、库存、配送等。不同的主题域可能共享某些维度，为了提高数据操作的性能和数据一致性，需要使用一致性维度，例如几个主题域间共享维度的复制。术语“一致性维度”源自 Kimball，指的是具有相同属性和内容的维度。

1）维度数据模型建模过程

维度数据模型通常以一种被称为星形模式的方式构建。所谓星形模式，就是以一个事实表为中心，周围环绕着多个维度表。还有一种模式叫作雪花模式，是对维度做进一步规

笔记

范化后形成的。本节后面会讨论这两种模式。一般使用下面的过程构建维度模型：

第一步：选择业务流程。

第二步：声明粒度。

第三步：确认维度。

第四步：确认事实。

这种使用四步设计法建立维度模型的过程，有助于保证维度模型和数据仓库的可用性。

（1）选择业务流程。确认哪些业务处理流程是数据仓库应该覆盖的，是维度方法的基础。因此，建模的第一个步骤是描述需要建模的业务流程。例如，需要了解和分析一个零售店的销售情况，那么与该零售店销售相关的所有业务流程都是需要关注的。为了描述业务流程，可以简单地使用纯文本将相关内容记录下来，或者使用“业务流程建模标注”（BPMN）方法，也可以使用统一建模语言（UML）或其他类似的方法。

（2）声明粒度。确定业务流程后，下一步是声明维度模型的粒度。这里的粒度用于确定事实中表示的是什么，例如，一个零售店的顾客在购物小票上的一个购买条目。在选择维度和事实前必须声明粒度，因为每个候选维度或事实必须与定义的粒度保持一致。在一个事实所对应的所有维度设计中强制实行粒度一致性是保证数据仓库应用性能和易用性的关键。从给定的业务流程获取数据时，原始粒度是最低级别的粒度。建议从原始粒度数据开始设计，因为原始记录能够满足无法预期的用户查询。汇总后的数据粒度对优化查询性能很重要，但这样的粒度往往不能满足对细节数据的查询需求。不同的事实可以有不同的粒度，但同一事实中不要混用多种不同的粒度。维度模型建立完成之后，还有可能因为获取了新的信息，而回到这一步修改粒度级别。

（3）确认维度。设计过程的第三步是确认模型的维度。维度的粒度必须与第二步声明的粒度一致。维度表是事实表的基础，也说明了事实表的数据是从哪里采集来的。典型的维度都是名词，如日期、商店、库存等。维度表存储了某一维度的所有相关数据，例如，日期维度应该包括年、季度、月、周、日等数据。

（4）确认事实。确认维度后，下一步是维度模型四步设计法的最后一步，就是确认事实。这一步识别数字化的度量，构成事实表的记录。它是和系统的业务用户密切相关的，因为用户正是通过对事实表的访问获取数据仓库存储的数据。大部分事实表的度量都是数字类型的，可累加，可计算，如成本、数量、金额等。

2）维度规范化

与关系模型类似，维度也可以进行规范化。对维度的规范化（又叫雪花化），可以去除冗余属性，是对非规范化维度做的规范化处理，在下面介绍雪花模型时，会看到维度规范化的例子。一个非规范化维度对应一个维度表，规范化后，一个维度会对应多个维度表，维度被严格地以子维度的形式连接在一起。实际上，在很多情况下，维度规范化后的结构等同于一个低范式级别的关系型结构。

查询性能原因。分析型查询需要聚合计算或检索很多维度值，此时第三范式的数据库会遭遇性能问题。如果需要的仅仅是操作型报表，可以使用第三范式，因为操作型系统的用户需要看到更细节的数据。

正如在前面关系模型中提到的，对于是否应该规范化的问题存在一些争论。总体来说，当多个维度共用某些通用的属性时，做规范化是会有益的。例如，客户和供应商都有

省、市、区县、街道等地理位置的属性，此时分离出一个地区属性就比较合适。

设计维度数据模型时，会因为如下原因而不对维度做规范化处理：

（1）规范化会增加表的数量，使结构更复杂。

（2）不可避免的多表连接，使查询更复杂。

（3）不适合使用位图索引。

3）维度数据模型的特点

（1）易理解。相对于规范化的关系模型，维度模型容易理解且更直观。在维度模型中，信息按业务种类或维度进行分组，这会提高信息的可读性，也方便了对数据含义的解释。简化的模型也让系统以更为高效的方式访问数据库。关系模型中，数据被分布到多个离散的实体中，对于一个简单的业务流程，可能需要很多表联合在一起才能表示。

（2）高性能。维度模型更倾向于非规范化，因为这样可以优化查询的性能。介绍关系模型时多次提到，规范化的实质是减少数据冗余，以优化事务处理或数据更新的性能。

（3）可扩展。维度模型是可扩展的。由于维度模型允许数据冗余，因此当向一个维度表或事实表中添加字段时，不会像关系模型那样产生巨大的影响，带来的结果是更容易容纳不可预料的新增数据。这种新增可以是单纯地向表中增加新的数据行而不改变表结构，也可以是在现有表上增加新的属性。基于数据仓库的查询和应用不需要过多改变就能适应表结构的变化，老的查询和应用会继续工作而不会产生错误的结果。但是，对于规范化的关系模型，由于表之间存在复杂的依赖关系，改变表结构前一定要仔细考虑。

4）星形模式

星形模式是维度模型最简单的形式，也是数据仓库以及数据集市开发中使用最广泛的形式。星形模式由事实表和维度表组成，一个星形模式中可以有一个或多个事实表，每个事实表引用任意数量的维度表。星形模式的物理模型像一颗星星的形状，中心是一个事实表，围绕在事实表周围的维度表表示星星的放射状分支，这就是星形模式这个名字的由来。

星形模式将业务流程分为事实和维度。事实包含业务的度量，是定量的数据，如销售价格、销售数量、距离、速度、重量等是事实。维度是对事实数据属性的描述，如日期、产品、客户、地理位置等是维度。一个含有很多维度表的星形模式有时被称为蜈蚣模式，显然，这个名字也是因其形状而得来的。蜈蚣模式的维度表往往只有很少几个属性，这样可以简化对维度表的维护，但查询数据时会有更多的表连接，严重时会使模型难于使用，因此在设计中应尽量避免使用蜈蚣模式。

（1）事实表。事实表记录了特定事件的数字化的考量，一般由数字值和指向维度表的外键组成。通常会把事实表的粒度级别设计得比较低，使得事实表可以记录很原始的操作型事件，但这样做的负面影响是累加大量记录可能会更耗时。事实表有以下三种类型：

①事务事实表。记录特定事件的事实，如销售。

②快照事实表。记录给定时间点的事实，如月底账户余额。

③累积事实表。记录给定时间点的聚合事实，如当月总的销售金额。一般需要给事实表设计一个代理键作为每行记录的唯一标识。代理键是由系统生成的主键，它不是应用数据，没有业务含义，对用户来说是透明的。

（2）维度表。维度表的记录数通常比事实表少，但每条记录都包含有大量用于描述事实数据的属性字段。维度表可以定义各种各样的特性，以下是几种最常用的维度表：

笔记

①时间维度表。描述星形模式中记录的事件所发生的时间，具有所需的最低级别的时间粒度。数据仓库是随时间变化的数据集合，需要记录数据的历史，因此每个数据仓库都需要一个时间维度表。

②地理维度表。描述位置信息的数据，如国家、省份、城市、区县、邮编等。

③产品维度表。描述产品及其属性。

④人员维度表。描述与人员相关的信息，如销售人员、市场人员、开发人员等。

⑤范围维度表。描述分段数据的信息，如高级、中级、低级等。

通常给维度表设计一个单列、整型数字类型的代理键，映射业务数据中的主键。业务系统中的主键本身可能是自然键，也可能是代理键。自然键指的是由现实世界中已经存在的属性组成的键，如身份证号就是典型的自然键。

（3）优点。星形模式是非规范化的，在星形模式的设计开发过程中，不受应用于事务型关系数据库的范式规则的约束。星形模式的优点如下：

①简化查询。查询数据时，星形模式的连接逻辑比较简单，而从高度规范化的事务模型查询数据时，往往需要更多的表连接。

②简化业务报表逻辑。与高度规范化的模式相比，由于星形模式的查询更简单，因此简化了普通的业务报表（如每月报表）逻辑。

③获得查询性能。星形模式可以提升只读报表类应用的性能。

④快速聚合。基于星形模式的简单查询能够提高聚合操作的性能。

⑤便于向立方体提供数据。星形模式广泛用于高效地建立 OLAP 立方体，几乎所有的 OLAP 系统都提供 ROLAP 模型（关系型 OLAP），它可以直接将星形模式中的数据当作数据源，而不用单独建立立方体结构。

（4）缺点。星形模式的主要缺点是不能保证数据完整性。一次性插入或更新操作可能会造成数据异常，而这种情况在规范化模型中是可以避免的。星形模式的数据装载一般都是以高度受控的方式，用批处理或准实时过程执行的，以此抵消数据保护方面的不足。

星形模式的另一个缺点是对于分析需求不够灵活。它更偏重为特定目的建造数据视图，因此实际上很难进行全面的数据分析。星形模式不能自然地支持业务实体的多对多关系，需要在维度表和事实表之间建立额外的桥接表。

5）雪花模式

雪花模式是一种多维模型中表的逻辑布局，其实体关系图有类似于雪花的形状，因此得名。与星形模式相同，雪花模式也由事实表和维度表组成。所谓的“雪花化”，就是将星形模式中的维度表进行规范化处理。当所有的维度表完成规范化后，就形成了以事实表为中心的雪花型结构，即雪花模式。将维度表进行规范化的具体做法是把低基数的属性从维度表中移除并形成单独的表。基数指的是一个字段中不同值的个数，如主键列具有唯一值，所以有最高的基数，而像性别这样的列基数就很低。在雪花模式中，一个维度被规范化成多个关联的表，而在星形模式中，每个维度由一个单一的维度表来表示。一个规范化的维度对应一组具有层次关系的维度表，而事实表作为雪花模式里的子表，存在具有层次关系的多个父表。星形模式和雪花模式都是建立维度数据仓库或数据集市的常用方式，适用于加快查询速度比高效维护数据的重要性更高的场景。这些模式中的表没有特别的规范

化，一般都被设计成一个低于第三范式的级别。

规范化的过程就是将维度表中重复的组分离成一个新表，以减少数据冗余的过程。正因为如此，规范化不可避免地增加了表的数量。在执行查询的时候，不得不连接更多的表。但是，规范化减少了存储数据的空间需求，而且提高了数据更新的效率。这一点在前面介绍关系模型时已经进行了详细讨论。

从存储空间的角度看，典型的情况是维度表比事实表小很多，这就使得雪花化的维度表相对于星形模式来说，在存储空间上的优势没那么明显。举例来说，假设在 220 个区县的 200 个商场共有 100 万条销售记录。星形模式的设计会产生 1 000 200 条记录，其中事实表有 1 000 000 条记录，商场维度表有 200 条记录，每个区县信息作为商场的一个属性，显式地出现在商场维度表中。在规范化的雪花模式中，会建立一个区县维度表，该表有 220 条记录，商场表引用区县表的主键，有 200 条记录，事实表没有变化，还是 1 000 000 条记录，总的记录数是 1 000 420（1 000 000+200+220）。在这种特殊情况（作为子表的商场记录数少于作为父表的区县记录数）下，星形模式所需的空间反而比雪花模式少。如果商场有 10 000 个，情况就不一样了，星形模式的记录数是 1 010 000，雪花模式的记录数是 1 010 220，从记录数看，还是雪花模型多。但是，星形模式的商场表中会有 10 000 个冗余的区县属性信息，而在雪花模式中，商场表中只有 10 000 个区县的主键，而需要存储的区县属性信息只有 220 个，当区县的属性很多时，会大大减少数据存储占用的空间。

有些数据库开发者采取一种折中的方式，底层使用雪花模型，上层用表连接建立视图模拟星形模式。这种方法既通过对维度的规范化节省了存储空间，同时又对用户屏蔽了查询的复杂性。但是，当外部的查询条件不需要连接整个维度表时，这种方法会带来性能损失。

雪花模式是和星形模式类似的逻辑模型。实际上，星形模式是雪花模式的一个特例（维度没有多个层级）。某些条件下，雪花模式更具以下优势：

（1）一些 OLAP 多维数据库建模工具专对雪花模型进行了优化。

（2）规范化的维度属性节省了存储空间。

雪花模型的主要缺点是维度属性规范化增加了查询的连接操作和复杂度。相对于平面化的单表维度，多表连接的查询性能有所下降，但雪花模型的查询性能问题近年来随着数据浏览工具的不断优化而得到缓解。与具有更高规范化级别的事务型模式相比，雪花模式并不确保数据完整性。向雪花模式的表中装载数据时，一定要有严格的控制和管理，避免数据的异常插入或更新。

3.2.2 数据仓库的标准架构

随时信息化时代的高速发展，海量数据可用 Hadoop、Spark 等大数据工具进行分析查询，但是对于数据仓库来说还远远不够，从海量数据中分析一个报表或者多个报表，大数据工具足矣；但是，对于在有限的资源动态的数据情况下，向前可历史追溯，向后对不断增加的报表实现兼容，就需要一套科学的数据管理方法。数据仓库是一门数据管理的科学，数据仓库的核心就是计算、存储和维护之间的博弈。

1. 标准数据仓库分层

标准数据仓库可以分为四层：数据采集层（operational data store，ODS，又称临时存

笔记

笔记

储层）、数据仓库层（data warehouse detail，DWD）、数据共享层（data wareHouse middle，DWM）、数据应用层（accelerated parallel processing，APP）。如图 3-1 所示。

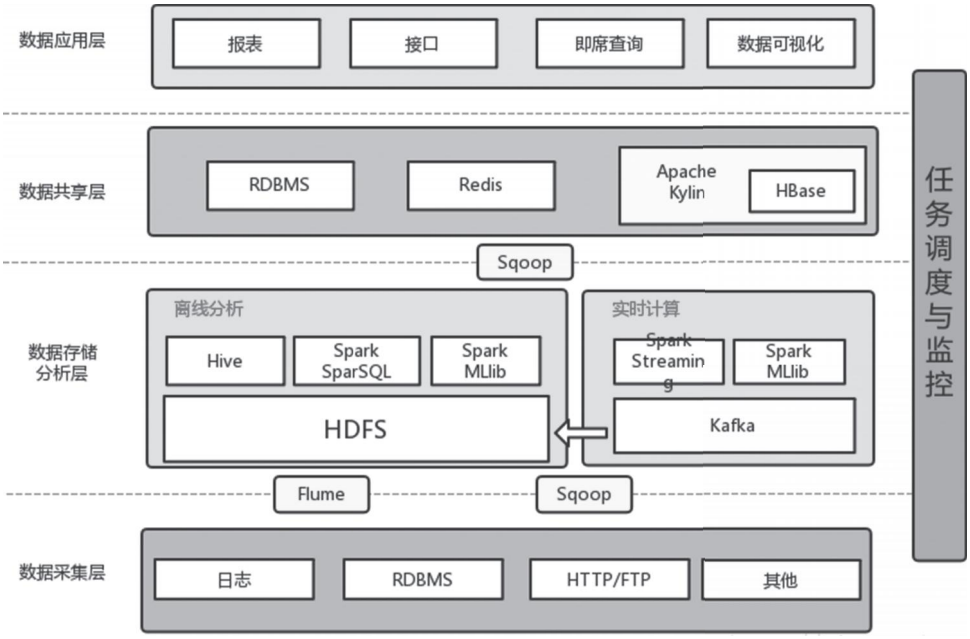


图 3-1　数据仓库架构图

1）ODS 层

ODS 层为临时存储层，是接口数据的临时存储区域，为后一步的数据处理做准备。一般来说，ODS 层的数据和源系统的数据是同构的，主要目的是简化后续数据加工处理的工作。从数据粒度上来说，ODS 层的数据粒度是最细的。ODS 层的表通常包括两类：一类用于存储当前需要加载的数据；一类用于存储处理完后的历史数据。历史数据一般保存 3~6 个月后需要清除，以节省空间。但不同的项目要区别对待，如果源系统的数据量不大，可以保留更长的时间，甚至全量保存。

ODS 层的任务是把数据从各种数据源中采集和存储到数据库上，期间有可能做一些 ETL（抽取 extra，转化 transfer，装载 load）操作。数据源种类可以有多种：

（1）日志：所占份额最大，存储在备份服务器上。业务数据库：如 MySQL、Oracle。来自 HTTP/FTP 的数据：合作伙伴提供的接口。其他数据源：如 Excel 等需要手工录入的数据。数据存储与分析 HDFS 是大数据环境下数据仓库 / 数据平台最完美的数据存储解决方案。

（2）业务数据：业务数据库的种类也是多种多样，有 MySQL、Oracle、SQL Server 等，这时，我们迫切需要一种能从各种数据库中将数据同步到 HDFS 上的工具，Sqoop 是一种，但是 Sqoop 太过繁重，而且不管数据量大小，都需要启动 MapReduce 执行，而且需要 Hadoop 集群的每台机器都能访问业务数据库，DataHub 也可以。Flume 可以通过配置与开发，也可以实时地从数据库中同步数据到 HDFS。

（3）Ftp/Http 的数据源：有一些合作伙伴提供的数据，需要通过 Ftp/Http 等定时获取。

2）DWD 层

DWD 层为数据仓库层，DWD 层中的数据应该是一致的、准确的、干净的，即对源系统数据进行清洗（去除了杂质）后的数据。这一层的数据一般遵循数据库第三范式，其

数据粒度通常和 ODS 的粒度相同。在 DWD 层会保存 BI 系统中所有的历史数据，例如保存 10 年的数据。

3）DWM 层

DWM 层为数据共享层，这层数据是面向主题组织数据的，通常是星形或雪花结构的数据。从数据粒度来说，这层数据是轻度汇总级的数据，已经不存在明细数据了。从数据的时间跨度来说，通常是 DWD 层的一部分，主要目的是满足用户分析的需求，而从分析的角度来说，用户通常只分析近几年（如近三年）的数据即可。从数据的广度来说，仍然覆盖了所有业务数据。

4）APP 层

APP 层为数据应用层，这层数据完全是为了满足具体的分析需求而构建的数据，也是星形或雪花结构的数据。从数据粒度来说，是高度汇总的数据。从数据的广度来说，则并不一定覆盖所有业务数据，而是 DWM 层数据的一个真子集，从某种意义上来说是 DWM 层数据的一个重复。从极端情况来说，可以为每一张报表在 APP 层构建一个模型支持，达到以空间换时间的目的。数据仓库的标准分层只是一个建议性质的标准，实施时需要根据实际情况确定数据仓库的分层，不同类型的数据也可能采取不同的分层方法。

2. 数据仓库分层的优势

（1）用空间换时间，通过大量的预处理提升应用系统的用户体验（效率），因此数据仓库会存在大量冗余的数据。

（2）如果不分层，源业务系统的业务规则发生变化将会影响整个数据清洗过程，工作量巨大。

（3）通过数据分层管理可以简化数据清洗的过程，因为把原来一步完成的工作分成多个步骤完成，相当于把一个复杂的工作拆成多个简单的工作，把一个大的黑盒变成一个白盒，每一层的处理逻辑都相对简单，容易理解，这样比较容易保证每个步骤的正确性，当数据发生错误的时候，只局部调整某个步骤即可。

3.2.3 数据集市架构

数据集市是按主题域组织的数据集合，用于支持部门级的决策。有两种类型的数据集市：独立数据集市和从属数据集市。

（1）独立数据集市。它来自市场的实施过程，被广泛使用。图 3-2 是独立型数据集市的架构，它的特点是简单，容易实现，而且实施时间短，但是其最大问题是，快速的实施，廉价的过程，导致须长期提供费用和效率低下。

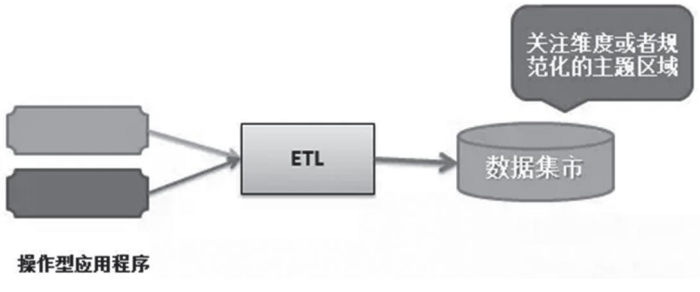


图 3-2　独立数据集市的架构

笔记

说明

数据层次的划分不是固定不变的，可以根据工作中实际项目的需要进行适当裁剪或者添加。如果业务相对简单和独立，可以将核心数据层与分析应用层进行合并。另外，分析应用的数据可以是多维模型数据，也可以是关系模型数据，甚至是原始数据。

笔记

开发一个独立的数据集市是获得可见结果的最有效的方法，因为不需要做跨部门、跨功能的分析，并且数据集市可以很快投入生产，因此能够迅速而廉价地获得结果，所以很多机构都应用这种方法。而且很多 ERP 集成商的系统中也自带了类似的功能作为一个卖点吸引客户。虽然它有很多优点，但是最致命的缺点是，短期的成功却带来长期的棘手问题，特别是独立型数据集市支持多主题区域时，会导致多个部门数据不一致，就是数据“打架”的现象，并且使得各个数据集市成为信息孤岛，缺乏兼容性。因此，这种方案很多时候是不可接受的。

独立数据集市集中于部门所关心的单一主题域，数据以部门为基础部署，无须考虑企业级别的信息共享与集成。例如，制造部门、人力资源部门和其他部门都有他们自己的数据集市。独立数据集市从一个主题域或一个部门的多个事务系统获取数据，用以支持特定部门的业务分析需要。一个独立数据集市的设计既可以使用实体关系模型，也可以使用多维模型。数据分析或商业智能工具直接从数据集市查询数据，并将查询结果显示给用户。一个典型的独立数据集市架构处理过程如图 3-3 所示。

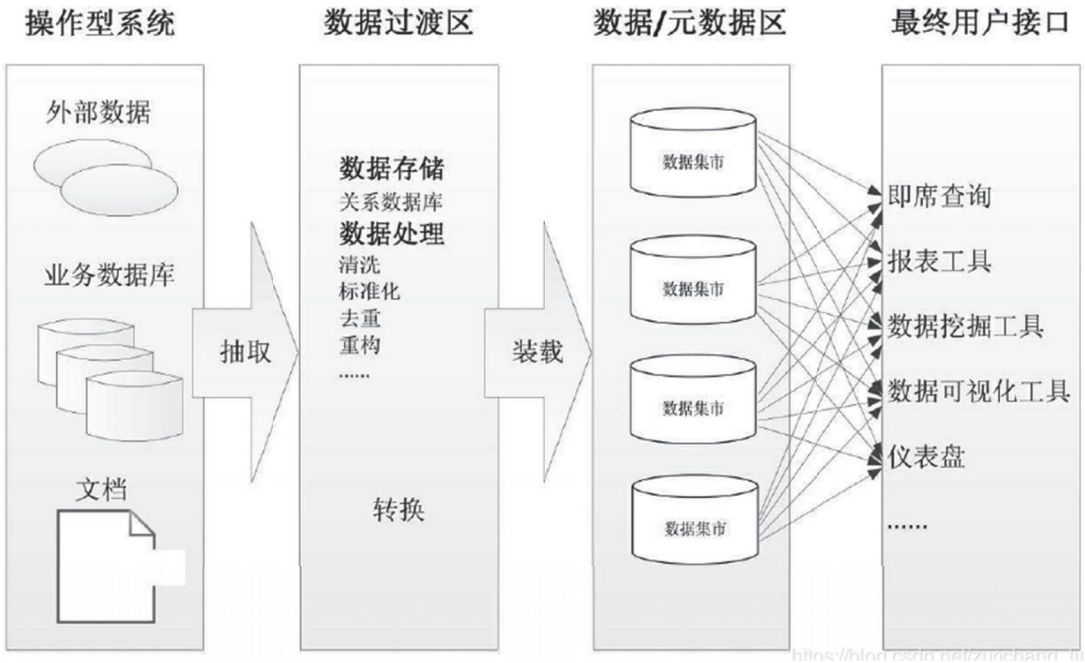


图 3-3　一个典型的独立数据集市架构处理过程

因为一个部门的业务相对于整个企业要简单，数据量也小得多，所以部门的独立数据集市具有周期短、见效快的特点。如果从企业整体的视角观察这些数据集市，你会看到每个部门使用不同的技术，建立不同的 ETL 过程，处理不同的事务系统，而在多个独立的数据集市之间还会存在数据的交叉与重叠，甚至会有数据不一致的情况。从业务角度看，当部门的分析需求扩展，或者需要分析跨部门或跨主题域的数据时，独立数据市场会显得力不从心。而当数据存在歧义，比如同一个产品在 A 部门和 B 部门的定义不同时，将无法在部门间进行信息比较。

（2）从属数据集市。如 Bill Inmon 所说，从属数据集市的数据来源于数据仓库。数据仓库里的数据经过整合、重构、汇总后传递给从属数据集市。从属数据集市的架构如图 3-4 所示。

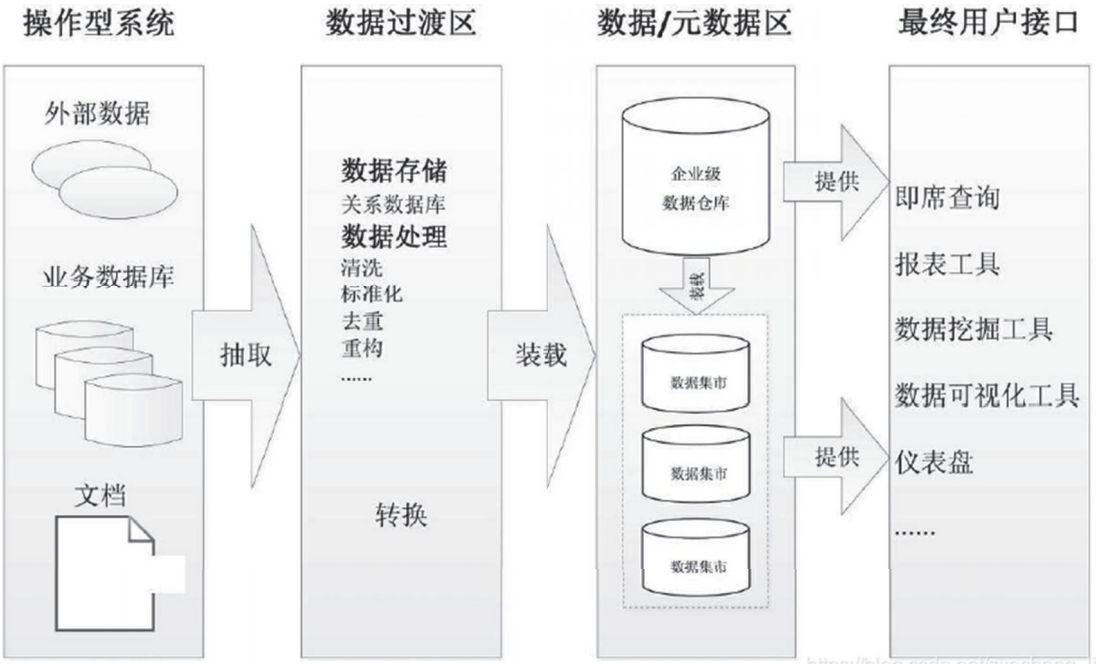


图 3-4　从属数据集市的架构

3.2.4 Inmon 企业信息工厂架构

Inmon 企业信息工厂架构体系架构如图 3-5 所示，左边是操作型系统或者事务系统，里面包括很多种系统，有数据库在线系统、文本文件系统等。这些系统的数据经过 ETL 的过程，加载数据到企业数据仓库中。ETL 的过程是整合不同系统的数据，经过整合，清洗和统一，因此我们可以称之为数据集成。

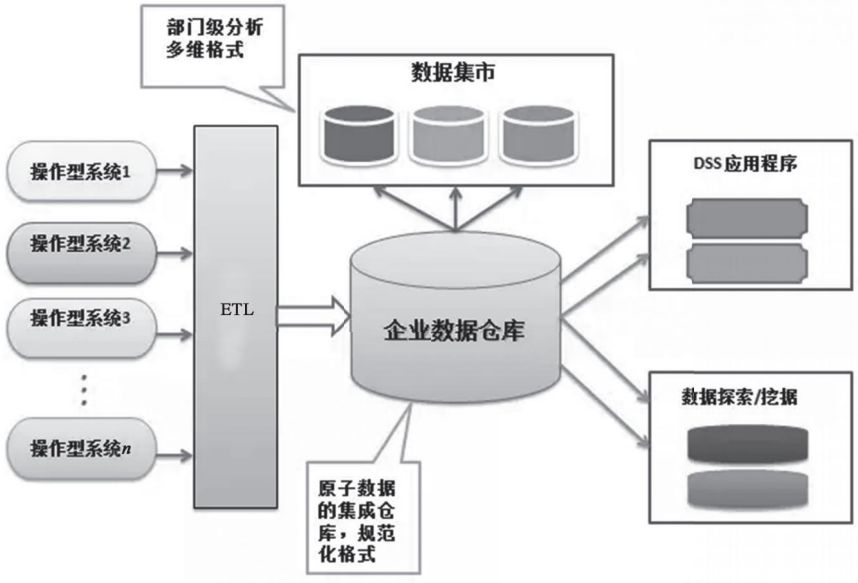


图 3-5　Inmon 企业信息工厂架构体系架构

企业数据仓库是企业信息化工厂的枢纽，是原子数据的集成仓库，但是由于企业数据仓库不是多维格式，因此不适合分析型应用程序，BI 工具直接查询。它的目的是将附加

笔记

的数据存储用于各种分析型系统。

数据集市是针对不同的主题区域，从企业数据仓库中获取信息，转换成多维格式，然后通过不同手段的聚集、计算，最后提供给最终用户分析使用，因此 Inmon 把信息从企业数据仓库移动到数据集市的过程描述为“数据交付”。

应用系统：这些应用是组织中的操作型系统，用来支撑业务。它们收集业务处理过程中产生的销售、市场、材料、物流等数据，并将数据以多种形式进行存储。操作型系统也称为源系统，为数据仓库提供数据。

ETL 过程：ETL 过程从操作型系统抽取数据，然后将数据转换成一种标准形式，最终将转换后的数据装载到企业级数据仓库中。ETL 是周期性运行的批处理过程。

企业级数据仓库：是该架构中的核心组件。正如 Inmon 数据仓库定义的，企业级数据仓库是一个细节数据的集成资源库。其中的数据以最低粒度级别被捕获，存储在满足三范式设计的关系数据库中。

部门级数据集市：是面向主题数据的部门级视图，数据从企业级数据仓库获取。数据在进入部门数据集市时可能进行聚合。数据集市使用多维模型设计，用于数据分析。重要的一点是，所有的报表工具、BI 工具或其他数据分析应用都从数据集市查询数据，而不是直接查询企业级数据仓库。如图 3-6 所示，我们来看图中的组件是如何协同工作的。

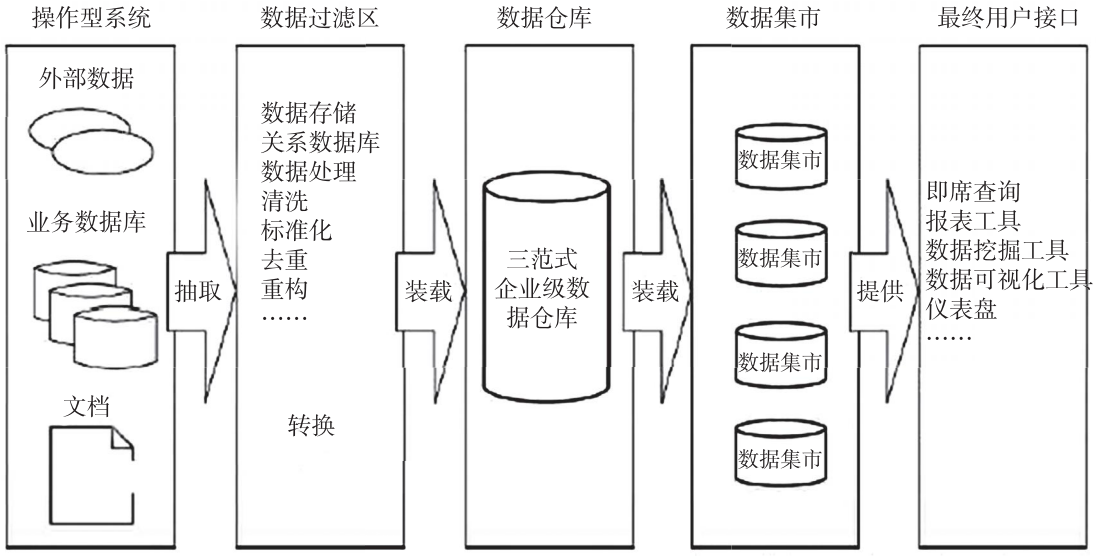


图 3-6　组件协同工作

3.2.5 Kimball 的维度数据仓库架构

Kimball 的维度数据仓库是基于维度模型建立的企业级数据仓库，它的架构有的时候可以称为“总线体系结构”，与 Inmon 提出的企业信息化工厂有很多相似之处，都是考虑原子数据的集成仓库。Kimball 的维度数据仓库架构如图 3-7 所示。

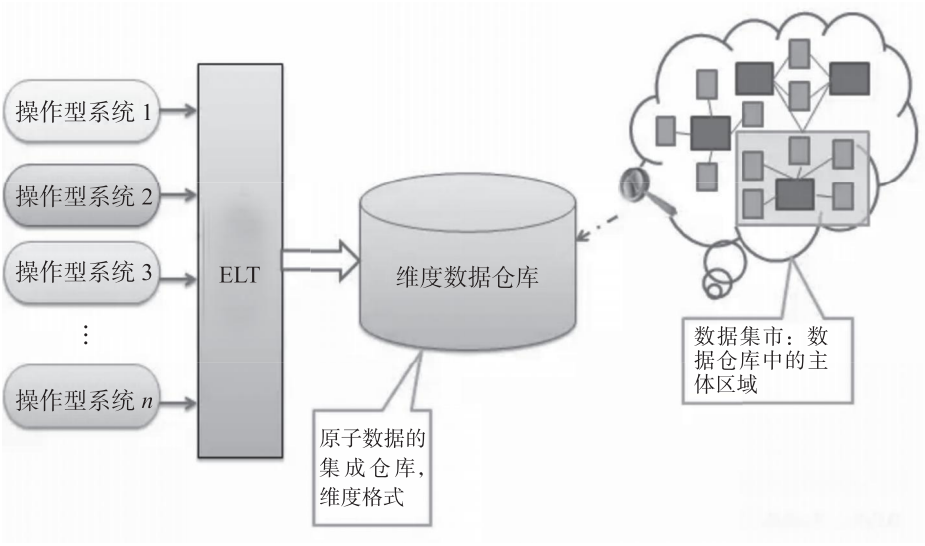


图 3-7　Kimball 的维度数据仓库架构

Inmon 企业信息工厂架构体系架构与 Kimball 的维度数据仓库架构这两种结构有很多相似之处：

（1）都假设操作型系统和分析型系统是分离的。

（2）数据源（操作型系统）都众多。

（3）ETL 整合了多种操作型系统的信息，集中到一个企业数据仓库。

当然，如果区别它们的不同，首先最大的不同就是企业数据仓库的模式不同，Inmon 采用的是第三范式的格式，而 Kimball 则采用了多维模型——星形模型，并且还是最低粒度的数据存储。其次维度数据仓库可以被分析系统直接访问，当然这种访问方式在分析过程中很少使用。最后就是数据集市的概念有逻辑上的区别，在 Kimball 的架构中，数据集市由维度数据仓库的高亮显示的表的子集表示。

Kimball 与 Inmon 架构还有核心数据仓库的设计和建立的区别。Kimball 的数据仓库包含高粒度的企业数据，使用多维模型设计，这也意味着数据仓库由星形模式的维度表和事实表构成。分析系统或报表工具可以直接访问多维数据仓库里的数据。在此架构中的数据集市也与 Inmon 中的不同。这里的数据集市是一个逻辑概念，只是多维数据仓库中的主题域划分，并没有自己的物理存储，也可以说是虚拟的数据集市。

当然，有时在 Kimball 的架构中有一个可变通的设计，就是在 ETL 的过程中加入 ODS 层，使得 ODS 层中能保留第三范式的一组表作为 ETL 过程的过渡，如图 3-8 所示。但是，这个思想在 Kimball 看来只是 ETL 的过程辅助而已。

笔记

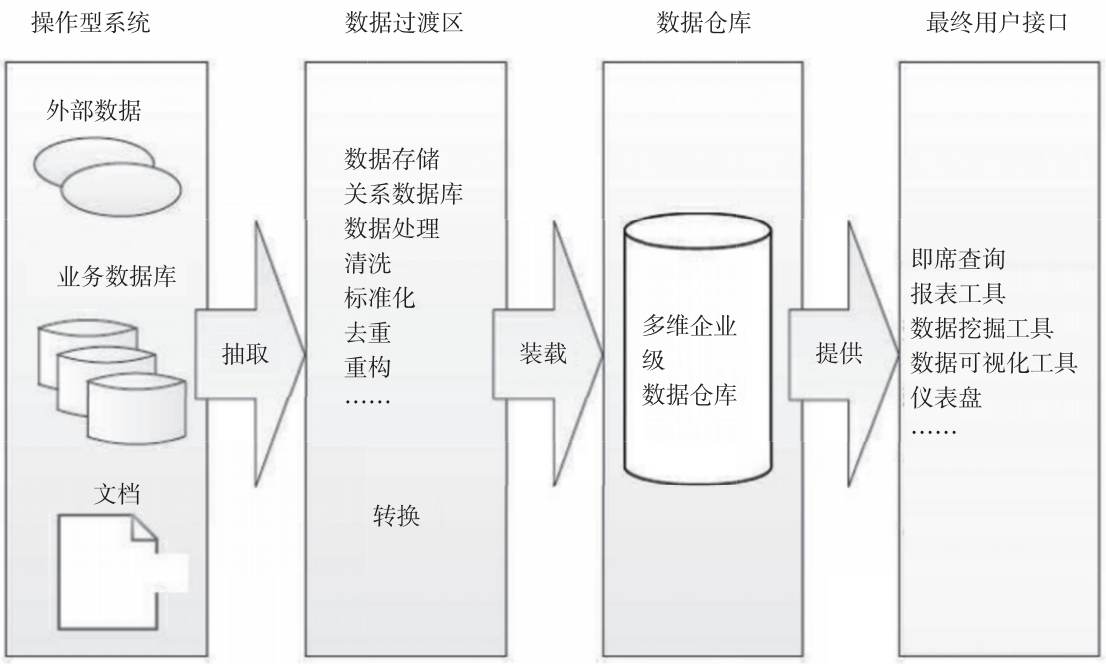


图 3-8　组件协调工作

另外，还可以把数据集市和企业维度数据仓库分离开，这样就多一层所谓的展现层（presentation layer），这些变通的设计只要符合企业本身分析的需求，都是可以接受的。

3.3 认识 ETL

3.3.1 ETL 概念

ETL（extract-transform-load）用来描述将数据从来源端经过抽取（extract）、转换（transform）、加载（load）至目的端的过程，能够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。

它将 OLTP 系统中的数据经过抽取，并将不同数据源的数据进行转换、整合，得出一致性的数据，然后加载到数据仓库中。简而言之，ETL 是完成从 OLTP 系统到 OLAP 系统的过程。如果说数据仓库的模型设计是一座大厦的设计蓝图，数据是砖瓦，那么 ETL 就是建设大厦的过程。在整个项目中最难的部分是用户需求分析和模型设计，而 ETL 规则设计和实施的工作量最大，约占整个项目的 60% ~ 80%，这是国内外从众多实践中得到的普遍共识。

3.3.2 ETL 基本流程

ETL 是构建数据仓库的重要一环，用户从数据源抽取出所需的数据，经过数据清洗，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，企业通过各种技术手段把数据转换为信息、知识，已经成了提高其核心竞争力的主要瓶颈。而 ETL 则是主要的一个技术手段。

ETL 技术是目前采用较多的数据交换技术之一，ETL 实际上是数据抽取、数据转换

和数据加载三种操作的总称。通常，ETL 操作发生在数据仓库中，通过 ETL 工具从数据源中抽取需要的数据，根据需要的格式对数据进行格式转换、清洗，去除冗余数据，将不同格式存储的数据进行格式统一，然后将加工处理后的数据加载到新的数据库中进行存储。

笔记

ETL 技术也就是数据抽取、数据转换和数据加载三种操作。通常，ETL 操作发生在数据仓库中，通过 ETL 工具从数据源中抽取需要的数据，根据实际需要的数据格式，对原有数据进行数据格式转换、数据清洗，剔除冗余数据等操作，以使得不同格式存储的数据能够实现格式统一。这样处理之后的数据会被加载到另外的新数据库中进行存储，如图 3-9 所示。

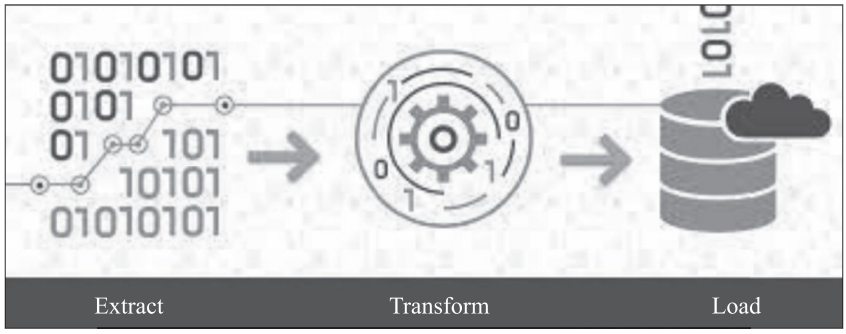


图 3-9　ETL 基本流程

ETL 系统的设计思想是以工作流为控制调度中心，利用元数据信息保存和记录 ETL 过程信息，定义 ETL 过程中各种转换规则的工作以组件的形式存在，并提供可视化界面供用户定义、修改 ETL 的流程，系统通过数据访问模块访问数据源和数据目标。

将工作流技术和元数据技术应用到 ETL 工具中已经是目前 ETL 工具发展的一种趋势。利用这两种技术，ETL 流程的灵活性和可控制性将大大提高，用户可以根据需要自由选择数据清洗和数据转换操作，并按照数据处理的流程灵活编排所选择的操作；同时，ETL 工具的可扩展性也得到增强，在工作流技术的支持下，所有的数据清洗和转换操作都成为工作流活动的资源，因此工具可以根据需要灵活添加清洗和转换操作，从而扩展相应的处理能力。

1. 数据抽取

数据抽取部分的主要功能是负责从数据源中抽取待加工的数据。数据抽取过程根据抽取方式的不同，可以分为全量抽取和增量抽取两种方式。全量抽取是指从源数据库中一次性将数据抽取出来，集成到目标数据库中。这种抽取方式可以满足对数据传输过程中一致性的要求，保证源数据库中数据与目标数据库中存储数据的一致性。增量抽取方式操作的数据对象都是数据规模较小的数据，这样的数据对服务器需求不高，因此采用增量抽取方式可以减轻服务器和网络带宽的压力。增量抽取方式一般使用于在数据仓库搭建之后的日常维护操作中。

这部分需要在调研阶段做大量的工作，首先要清楚数据是从几个业务系统中来，各个业务系统的数据库服务器运行什么 DBMS，是否存在手工数据，手工数据量有多大，是否存在非结构化的数据等，当收集完这些信息之后，才可以进行数据抽取的设计。

（1）对于与存放数据仓库的数据库系统相同的数据源处理方法，这类数据源在设计上比较容易。一般情况下，DBMS（SQL Server、Oracle）都会提供数据库链接功能，在数据仓库数据库服务器和原业务系统之间建立直接的链接关系，就可以写 Select 语句直接访问。

笔记

（2）对于与数据仓库数据库系统不同的数据源的处理方法，这类数据源一般情况下也可以通过 ODBC 的方式建立数据库链接——如 SQL Server 和 Oracle 之间。如果不能建立数据库链接，则有两种方式可以完成：一种方式是通过工具将源数据导出为 .txt 或者 .xls 文件，然后再将这些源系统文件导入到 ODS 中；另外一种方法是通过程序接口完成。

（3）对于文件类型数据源（.txt，.xls），可以培训业务人员利用数据库工具将这些数据导入到指定的数据库，然后从指定的数据库中抽取。或者还可以借助工具实现。

（4）增量更新的问题。对于数据量大的系统，必须考虑增量抽取。一般情况下，业务系统会记录业务发生的时间，我们可以用它作为增量的标志，每次抽取之前首先判断 ODS 中记录的是否为最大的时间，然后根据这个时间去业务系统取大于这个时间的所有记录，并利用业务系统的时间戳（一般情况下，业务系统没有或者部分有时间戳）。

2. 数据的清洗与转换

数据的清洗（cleaning）、转换（transform）过程是将上一步骤中抽取的数据进行清洗，并按照预先设定的规则转换成目标数据库能够接受的同一标准格式。由于数据抽取步骤中得到的数据从不同的数据源产生，这些数据根据采用的数据库不同，会以不同的格式存储。此外，不同来源的数据进行整合的过程中可能产生冲突数据和冗余数据等问题，这就需要在载入目标数据库之前对数据进行清洗处理，使待载入的数据从格式和内容上实现正确和统一。

一般情况下，数据仓库分为 ODS、DW 两部分，通常的做法是先从业务系统到 ODS 做清洗，将“脏”数据和不完整数据过滤掉，再从 ODS 到 DW 的过程中转换，进行一些业务规则的计算和聚合。

1）数据清洗

数据清洗的任务是过滤不符合要求的数据，将过滤的结果交给业务主管部门，确认是否过滤掉，还是由业务单位修正之后再进行抽取。

不符合要求的数据主要有不完整的数据、错误的数据、重复的数据三大类。

（1）不完整的数据：这类数据主要是一些应该有的信息缺失，如供应商的名称、分公司的名称、客户的区域信息缺失、业务系统中主表与明细表不能匹配等。将这类数据过滤出来，按缺失的内容分别写入不同 Excel 文件向客户提交，要求在规定的时间内补全。补全后才写入数据仓库。

（2）错误的数据：这类错误产生的原因是业务系统不够健全，在接收输入后没有进行判断而直接写入后台数据库造成的，比如数值数据输成全角数字字符、字符串数据后面有一个回车操作、日期格式不正确、日期越界等。这类数据也要分类，对于类似于全角字符、数据前后有不可见字符的问题，只能通过写 SQL 语句的方式找出来，然后要求客户在业务系统修正之后抽取。日期格式不正确的或者是日期越界的这类错误会导致 ETL 运行失败，这类错误需要去业务系统数据库用 SQL 的方式挑出来，交给业务主管部门要求限期修正，修正之后再抽取。

（3）重复的数据：对于这类数据——特别是维度表中会出现这种情况——将重复数据记录的所有字段导出来，让客户确认并整理。

数据清洗是一个反复的过程，不可能在几天内完成，只有不断地发现问题，解决问题。对于是否过滤、是否修正的问题，一般要求客户确认，对于过滤掉的数据，写入

Excel 文件或者将过滤数据写入数据表，在 ETL 开发的初期可以每天向业务单位发送过滤数据的邮件，促使他们尽快地修正错误，同时也可以作为将来验证数据的依据。数据清洗需要注意的是不要将有用的数据过滤掉，应对每个过滤规则认真进行验证，并要用户确认。

笔记

2）数据转换

数据转换的任务主要是进行不一致数据的转换、数据粒度的转换，以及一些商务规则的计算。

（1）不一致数据的转换：这个过程是一个整合的过程，将不同业务系统的相同类型的数据统一，比如同一个供应商在结算系统中的编码是 XX0001，而在 CRM 中的编码是 YY0001，这样，在抽取过来之后统一转换成一个编码。

（2）数据粒度的转换：业务系统一般存储非常明细的数据，而数据仓库中的数据是用来分析的，不需要非常明细的数据。一般情况下，会将业务系统数据按照数据仓库粒度进行聚合。

（3）商务规则的计算：不同的企业有不同的业务规则、不同的数据指标，这些指标有的时候不是简单地加加减减就能完成，这个时候需要在 ETL 中将这些数据指标计算好后存储在数据仓库中，以供分析使用。

3. 数据加载

数据加载的过程主要是将已经清洗过的数据装载到目标数据仓库的指定位置上。数据规模的大小不同，处理方式也不同。对于少量的数据库更新操作，直接使用 SQL 语句实现；对于批量数据载入的请求，则需要使用专门的加载工具实现，通过使用加载工具加载实现对数据的分隔和异步载入。

3.3.3 ETL 体系架构

数据的抽取、转换和加载是构建数据仓库过程中最复杂，也是至关重要的一个步骤，我们通常用两种办法处理 ETL 流程：一种是异步（asynchronous）ETL 方式，也称为文本文件（flat file）方式；另外一种是同步（synchronous）ETL 方式，也称为直接传输（direct transfer）方式。根据项目各自的特点，合理选择恰当的数据抽取流程，确定抽取过程中的监督核查机制，对数据仓库项目的成功可以起到事半功倍的作用。在第一种模式中，数据被从源数据库抽取到文本文件中，通过网络传输被传送到目标服务器，然后再装载到目标数据库中。对于后一种模式，顾名思义，数据抽取过程是直接从源到目标的，没有通过中间路径或者手段。在数据仓库工程的开发中，这两种体系架构被普遍采用。

1. 异步 ETL 架构

异步 ETL 系统体系架构如图 3-10 所示。图中，源和目标数据库分别位于独立的数据中心。负责 ETL 抽取功能的应用程序可以安装在源数据库，也可以安装在目标数据库，当然更可以安装在独立（第三方）的服务器上。ETL 的顺序是：源数据库→文本文件→目标数据库。

笔记

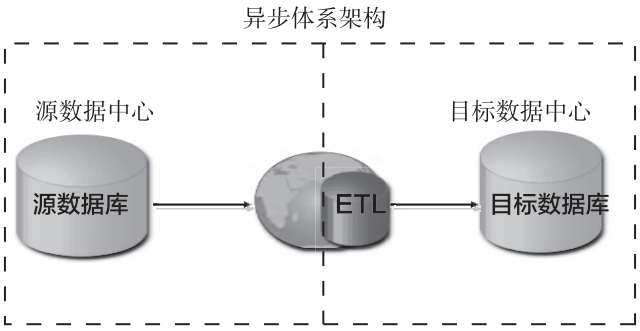


图 3-10　异步 ETL 系统体系架构

2. 同步 ETL 架构

同步 ETL 系统体系架构如图 3-11 所示，抽取过程没有任何中间步骤过渡。其顺序是：

源数据库 —ETL 程序→ 目标数据库

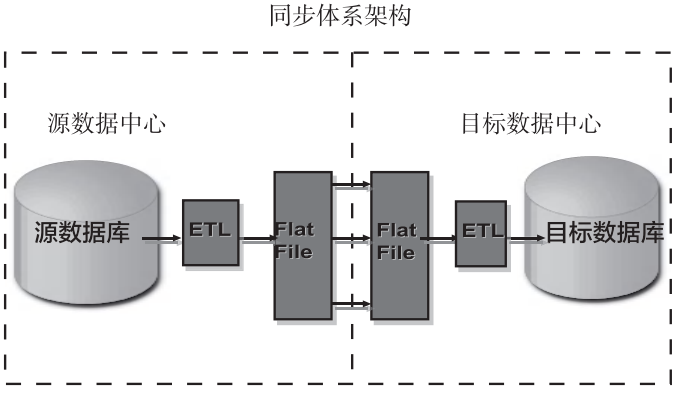


图 3-11　同步 ETL 系统体系架构

依循数据仓库的工作方式，原始资料由源数据库被抽取出来后，将在中间过程被写入到 operational data store（ODS），ODS 是用来存储中间数据和核查校验数据的。通过 ODS，数据将被萃取、预先被计算及整理，而后被导入数据仓库做进一步的报表生成与分析。所以，通常意义的 ETL 过程涵盖了两方面的内容：

（1）从源数据库到中间步骤的 ODS。

（2）从 ODS 到最终的数据仓库。

3. 两种体系架构的比较

ETL 负责将分布的、异构数据源中的数据（如关系数据、平面数据文件等）抽取到临时中间层后进行清洗、转换、集成，最后加载到数据仓库或数据集市中，成为联机分析处理、数据挖掘的基础。

ETL 是数据仓库中非常重要的一环，是承前启后的必要一步。相对于关系数据库，数据仓库技术没有严格的数学理论基础，它更面向实际工程应用。所以，从工程应用的角度考虑，按物理数据模型的要求加载数据并对数据进行一些系列处理，处理过程与经验直接相关，同时这部分工作直接关系数据仓库中数据的质量，从而影响联机分析处理和数据挖掘结果的质量。

数据仓库是一个独立的数据环境，需要通过抽取过程将数据从联机事务处理环境、外

部数据源和脱机的数据存储介质导入数据仓库中；在技术上，ETL 主要涉及关联、转换、增量、调度和监控等几个方面；数据仓库系统中的数据不要求与联机事务处理系统中的数据实时同步，所以 ETL 可以定时进行。但多个 ETL 的操作时间、顺序和成败对数据仓库中信息的有效性至关重要。两种不同体系结构的特点见表 3-11。

笔记

表 3-11　两种不同体系结构的特点

异步（asynchronous ）	同步（synchronous）
比同步模式提供了更好的数据处理性能，需要的处理时间更少，因为通过网络传输文件的速度（FTP）比直接通过数据库存取数据快很多	避免性能瓶颈问题的解决办法是缩小每次抽取的时间粒度，例如将抽取周期定为每日抽取，这样可以保证每次抽取的增量数据的数目很小
在数据抽取过程中，应尽量避免本次抽取定义的时间区间内的源数据在抽取过程中同时产生变动的情况，换句话说，抽取的理想状况是抽取的同时希望前端系统的数据是静止的，没有增、删、改的情况发生。对于 ODS 系统来说，这通常不是一个问题，因为数据不会频繁地发生变动，而对于 OLTP 系统来说，应该选择源数据变化较少的时段完成抽取工作	与异步方式类似，应该避免抽取时间区间和前端系统的生产时段重合。当然，如果前端系统是一个 7 × 24 小时都有新数据插入的系统，一种解决办法是设置一个时间区间，定义每次抽取的开始和结束时间值。本次抽取的开始时间为上次抽取的结束时间，本次抽取的结束时间为机器系统时间（sysdate）或者是目前数据库系统中已有记录的最大时间戳值。实际上就是定义某个时间区间内的源数据的快照（snapshot），这样就可以避免重复装载的情况发生。除此之外，还应该充分考虑源和目标两套数据库系统的死机的时间因素
需要两套 ETL 包，一套用来抽取，一套用来装载，两套包都需要由专门的系统管理人员监视装载过程是否会发生错误	只需要一个 ETL 软件包。系统管理人员也只需要监视一套系统
当 ETL 发生错误时，可以采用简单的处理办法修复数据：当抽取失败时，修正问题并重新从源中抽取；当装载过程发生问题，回滚（rollback），返回上一次装载的状态并再次运行装载的程序。如果需要，甚至可以将数据仓库系统恢复到某一个时点的状态，并批量地装载文本文件	与异步模式类似，当 ETL 发生错误时，也可以采用相应方式修复数据。例如，倘若源数据库中保留了所有历史记录，ETL 程序可以根据设定的时间区间随心所欲修复上一次装载失败的数据。当然，这一切的前提是必须先删除上一次装载失败的数据，从而在目标库中产生垃圾数据，回滚（rollback），返回到上一次加载数据前的状态。此外，如果 ETL 程序设计得合理，我们可以根据目标表的主键（primary key）确定装载过程中插入或者更新记录的策略，如果源记录的主键是新的，那么就插入该记录；如果源记录的主键在目标数据库中已经存在，就用源记录更新目标记录，这样可以避免多次重复装载的情况发生。同时，适当调大装载的时间区段也不会担心发生数据重复加载的情况
需要有专门的核查（audit）程序监控数据传输或者装载过程是否有失败或者记录缺失的情况发生	因为是直接复制（direct copy），所以可以避免传送过程中的许多错误。可以在源和目标库中运行 Sum( ) 和 Count( ) 等聚合函数对数据质量进行校验
能够简化查错（trouble shooting）的过程，由于在中间步骤生成了一个文本文件，事实上这就是我们需要抽取源数据的快照（snapshot），当有问题发生时，可以很容易地确定究竟是哪个环节发生了问题，是源、目标，还是网络传输	与异步模式相比，查错过程有些复杂，很难判断上一次究竟抽取的是哪些记录，因为源数据在不断发生变化；很难判断问题究竟是由于抽取程序，还是由于源数据的变化引起的
因为增加了中间过渡的文本文件，源和目标之间没有直接的联系，所以只要文本文件的结构不发生变化，源和目标的结构即使改变，也不会对 ETL 流程产生很大的影响	在 ETL 工具中，源和目标的关系是被绑定在具体的映射（mapping）中的，当源或者目标的结构发生变化，相应的映射也要做修改

笔记

续表

异步（asynchronous ）	同步（synchronous）
在异步模式中，因为源和目标的数据接口是分开的，所以只要定义好中间的文本文件数据接口，就可以有两个不同的开发团队完成独立的源和目标的开发工作。当各自模块完成后，再将其装配，这样会大大提高开发效率	要求 ETL 的开发人员不仅需要熟悉源的结构，对目标的结构也要有所了解。当然，只有一个团队进行开发，也消除了不同团队间进行沟通而存在的障碍
需要将数据导出成字节流（byte stream，ASCII 或 Unicode）并写入文本文件中。如果源包含图形数据，要将其导出成文本，这样实现起来就有一定的难度	从源到目标的直接映射（direct mapping），不需要从 ASCII 或者 Unicode 作为中间过渡
维护过程中要求不能误删文件或者删除文件中的某些记录。需要特别小心，以免破坏文本文件之间的关联关系	与异步模式相比，由于主表和子表之间主外键相关联，基于数据一致性的原则和数据库相应预防机制，所以不容易发生误删主表记录的情况，数据库中的各表记录能够保持相互间的匹配关系
数据转换过程包括两个步骤：①将数据库中的表导出成中间过渡的文本；②装载文件。导出的处理过程比较灵活，可以从源表导出，也可以从相关的视图导出，甚至可以将源表内容先导出到临时表，再导出到文本文件。处理导出过程的存储过程的位置也很灵活，既可以在源上，也可以在目标上	数据转换过程只有一个步骤，一次性完成导出和装载的工作，这简化了设计和测试的过程，但是另一方面也降低了灵活性
要求具备两套安全控制机制，对于源数据库有读权限，对于目标数据库有写权限。同时还需要在源和目标服务器上有写文件的权限（用于存放中间文本文件和上传文件到目标服务器）	与异步模式类似，也需要对于源数据库有读权限，对于目标数据库有写权限。但是抽取过程可以不需要源和目标服务器上操作系统级的文件管理权限

3.4 ETL 架构设计

3.4.1 ETL 架构设计概念

企业信息化建设过程中，业务系统各自为政、相互独立造成的“数据孤岛”现象尤为普遍，业务不集成、流程不互通、数据不共享，给企业进行数据的分析利用、报表开发等带来了巨大困难。在此情况下，数据仓库的建设就显得必不可少了，将相互分离的业务系统的数据源整合在一起，建立一个统一的数据采集、处理、存储、分发、共享中心，实现企业全局数据的系统化运作管理，为 DSS（决策支持系统）、BI（商务智能）、经营分析系统等深度开发应用奠定基础，挖掘数据价值。

在企业搭建数据仓库的过程中，有一个核心环节——ETL。如果说数据仓库是一座大楼，那么 ETL 就是大楼的地基。ETL 负责将分布的、异构数据源中的数据（如关系数据、平面数据文件等）抽取到临时中间层后进行清洗、转换、集成，最后加载到数据仓库或数据集市中，成为联机分析处理、数据挖掘的基础。ETL 设计和实施的工作量一般要占数据仓库总工作量的 60% 以上，数据仓库建成后的日常运维的好坏也严重依赖 ETL 的设计使用，所以说 ETL 是整个数据仓库的生命线，ETL 工具的选择对于整个数据仓库项目的成功非常重要。

笔记

1. 集结区

准备数据，通常也叫作数据管理，是指获取数据并将数据转化成信息，最终将这些信息提交到前端的查询界面。后台不提供查询服务，数据仓库方法论假设在后台数据访问是被严格禁止的，这是前台的唯一目的。数据仓库的后台部分经常被称为集结区（staging area）。数据集结主要是指写入磁盘，连同 ETL 的四个主要步骤都要有数据集结。

2. 集结区的意义

将数据存储在物理集结区，还是存储在内存中直接处理？这个问题是 ETL 架构中最根本的选择之一。开发的 ETL 处理的效率很大程度上取决于能否很好地均衡物理 I/O 与内存处理。能够在将数据写入集结表和保持在内存两种方法间取得理想的均衡是一个很大的挑战，也是优化处理过程中必须考虑的问题。最终的决定取决于下面两个彼此矛盾的目标：

将数据以最快的速度从数据源获取到最终目标在处理过程发生错误时，能够进行恢复而无须从头开始，根据环境和业务需求的不同，数据集结的策略会有很大的不同。如果计划在内存中处理所有的 ETL 数据，不要忘记任何一种数据仓库，无论其架构和运行环境如何，都包含了一个某种形式的集结区。之所以在加载到数据仓库之前集结数据，主要基于如下的考虑：

（1）可恢复。在大多数的企业环境中，数据从源系统中抽取出来后，会进行一系列重要的转换，假设对于某张表，其转换的工作量很大，那么根据我们的最佳实践，应该在数据一抽取完马上就进行集结。这些集结表（在数据库或者文件系统中）可以作为恢复点。一旦转换过程发生错误，利用这些表，处理过程就无须再次访问源系统。同样的道理，如果加载过程失败，也无须重新进行转换。如果集结数据单纯是为了达到恢复的目的，那么数据应该存储在文件系统中的顺序文件中，而不是数据库。以恢复为目的的数据集结对于从业务系统中抽取数据尤其重要，因为业务系统中的数据会被覆盖和修改。

（2）备份。通常，巨大的数据量使得在数据库级别上进行可靠的数据仓库备份变得不可行。只要加载文件已经进行了保存、压缩和归档，那么我们就可以避免数据库故障所带来的灾难。如果集结表存储在文件系统，就可以被压缩成非常小的文件并存储在网络上。当需要重新向数据仓库中加载数据时，仅需要对加载文件解压缩并重新加载。

（3）审计。很多时候，源系统和目标系统之间的数据沿袭在 ETL 代码中丢失，当审计 ETL 流程时，数据集结区的存在使得对 ETL 流程中的不同阶段的直接比较成为可能，因为这时候审计人员（或者程序员）可以简单地比较原始的输入文件和输出文件检查逻辑转换规则。当源系统覆盖了历史数据时，集结数据特别有用。当一个事件发生后，你可能对数据仓库中几天甚至几周的数据信息的完整性产生疑问，这时候对相应时段的集结区的抽取数据进行检查将能够帮助你恢复对数据仓库的数据准确性的信心。

最终目标：将数据以最快的速度从数据源获取到最终目标；在处理过程中发生错误时，能够进行恢复，而无须从头开始。

3.4.2 ETL 架构和 ELT 架构的对比

ETL 工具目前有两种技术架构——ETL 架构和 ELT 架构。

笔记

1. ETL 架构

ETL 架构按其字面含义理解就是按照 E—T—L 这个顺序流程进行处理的架构：先抽取、然后转换、完成后加载到目标数据库中。在 ETL 架构中，数据的流向是从源数据流到 ETL 工具，ETL 工具是一个单独的数据处理引擎，一般会在单独的硬件服务器上，实现所有数据转化的工作，然后将数据加载到目标数据仓库中。如果要提高整个 ETL 过程的效率，则只能增强 ETL 工具服务器的配置，优化系统处理流程（一般可调的东西非常少），如图 3-12 所示。

图 3-12　ETL 架构设计

ETL 在转化的过程中，主要体现在以下几方面。

（1）空值处理：可捕获字段空值，进行加载或替换为其他含义数据，并可根据字段空值实现分流加载到不同目标库。

（2）规范化数据格式：可实现字段格式约束定义，对于数据源中时间、数值、字符等数据，可自定义加载格式。

（3）拆分数据：依据业务需求对字段可进行分解。例如，主叫号 861082585313-8148，可进行区域码和电话号码分解。

（4）验证数据正确性：可利用 Lookup 及拆分功能进行数据验证。例如，主叫号 861082585313-8148，进行区域码和电话号码分解后，可利用 Lookup 返回主叫网关或交换机记载的主叫地区，进行数据验证。

（5）数据替换：对于业务因素，可实现无效数据、缺失数据的替换。

（6）Lookup：查获丢失数据 Lookup 实现子查询，并返回用其他手段获取的缺失字段，保证字段完整性。

（7）建立 ETL 过程的主外键约束：对无依赖性的非法数据，可替换或导出到错误数据文件中，保证主键唯一记录的加载。

ETL 架构的优势：

（1）ETL 可以分担数据库系统的负载（采用单独的硬件服务器）。

（2）ETL 相对于 EL-T 架构，可以实现更复杂的数据转化逻辑。

（3）ETL 采用单独的硬件服务器。

（4）ETL 与底层的数据库数据存储无关。

2. ELT 架构

在 ELT 架构中，ELT 只负责提供图形化的界面设计业务规则，数据的整个加工过程都在目标和源的数据库之间流动，ELT 协调相关的数据库系统执行相关的应用，数据加工

过程既可以在源数据库端执行，也可以在目标数据仓库端执行（主要取决于系统的架构设计和数据属性）。若 ELT 过程需要提高效率，通过对相关数据库进行调优，或者改变执行加工的服务器就可以达到。一般数据库厂商会力推该种架构，像 Oracle 和 Teradata 都极力宣传 ELT 架构，如图 3-13 所示。

笔记

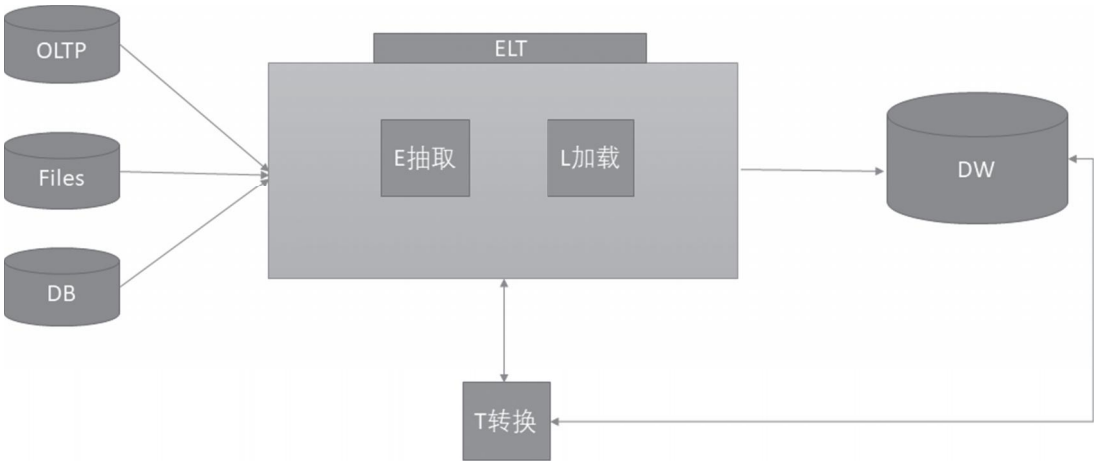


图 3-13　ELT 架构

ELT 架构的优势：

（1）ELT 主要通过数据库引擎实现系统的可扩展性（尤其是当数据加工过程在晚上时，可以充分利用数据库引擎的资源）。

（2）ELT 可以保持所有的数据始终在数据库中，避免数据的加载和导出，从而保证效率，提高系统的可监控性。

（3）ELT 可以根据数据的分布情况进行并行处理优化，并可以利用数据库的固有功能优化磁盘 I/O。

（4）ELT 的可扩展性取决于数据库引擎和其硬件服务器的可扩展性。

（5）通过对相关数据库进行性能调优，ETL 过程获得 3~4 倍的效率提升一般不是特别困难。

如何选择两种架构主要取决于公司现有的网络架构、预算，以及它已经使用云和大数据技术的程度。但是，当 3 个焦点区域中的任何一个或全部都很关键时，可以考虑使用 ELT。

当摄取速度是优先考虑问题时，必须使用 ELT。因为 ELT 不必等待数据被处理掉，然后加载（此处加载数据和转换可以并行发生）。这里，摄取过程更快，并提供比 ETL 更快的原始信息。

将数据转化为商业智能的优势在于能够将隐藏模式表现为可操作的信息。通过保存所有历史数据，组织可以随时间线、销售模式、季节性趋势或任何新兴指标进行挖掘，这对组织而言至关重要。在这种情况下，可以访问原始数据，因为数据在加载之前未被转换。大部分云数据中，原始数据先被存储，然后被提炼，或者存储处理过的信息。例如，数据科学家更喜欢使用原始数据的访问，而业务用户则更喜欢将规范化的数据用于商业智能。

当使用高端数据处理引擎（如云数据仓库或 Hadoop）时，ELT 可以利用本机处理能力实现更高的可扩展性。ETL 和 ELT 都是节省时间的方法，用于从原始数据生成商业智能。

笔记

ETL 架构和 ELT 架构的比较见表 3-12。

表 3-12　ETL 架构和 ELT 架构的比较

参数 / 项目	ETL	ELT
处理	数据在暂存服务器中传输，然后移动到数据仓库数据库	数据保留在数据仓库的 DB 中
转型 / 转换	转换在 ETL 服务器和暂存区域中完成	转换在暂存区域中执行
代码用法	ETL 用于少量数据，计算密集型转型	ELT 用于大数据
加载时间	首先，数据分阶段加载，然后加载到目标系统中，这是一个耗时的过程	在 ELT 中，数据仅在目标系统中加载一次。在这个过程中花费的时间较少
转换时间	ETL 过程需要时间完成转换。随着数据量的增长，转换时间也会增加	在 ELT 过程中，速度绝不取决于数据的大小
维护时间	当选择要加载和转换的数据时，需要高度维护	由于数据始终可用，因此 ELT 需要低维护
实施复杂性	在 ELT 中，更容易在早期实施它	要实施 ELT 流程，组织应该对专家技能和工具有深入的了解
数据湖支持	ETL 不支持数据湖	ELT 允许将数据湖与非结构化数据一起使用
支持数据仓库	ETL 模型用于关系数据和结构化数据	ELT 用于可扩展的云基础架构，支持结构化和非结构化数据
复杂性	ETL 过程仅加载在设计时识别的基本数据	ELT 仅涉及从后向输出开发，并仅加载相关数据
成本	在 ETL 过程中，中小型企业的成本很高	ELT 包括使用在线软件作为服务平台的低入门成本
查找	在 ETL 过程中，需要在临时区域中提供维度和事实	在 ELT 中，所有数据都可用，因为提取和加载只在一个动作中发生
计算	在 ETL 中，现有列被覆盖或需要附加数据集并推送到目标平台	在 ELT 中，很容易将列添加到现有表中
硬件	在 ETL 中，这些工具具有独特的硬件要求，这是昂贵的	ELT 是一个新概念，实施起来很复杂
支持非结构化数据	ETL 支持关系数据	ELT 有助于提供非结构化的现成数据

3.5 ETL 测试

3.5.1 ETL 测试的概念

ETL 测试是在将数据移动到生产数据仓库系统之前完成 ETL 测试，也称为表平衡或产品协调。ETL 测试与数据库测试的范围和测试期间遵循的步骤不同。ETL 测试是为了确保转换后从源加载到目标的数据是准确的。它涉及在源和目的地之间使用的各个阶段的数据验证，如图 3-14 所示。

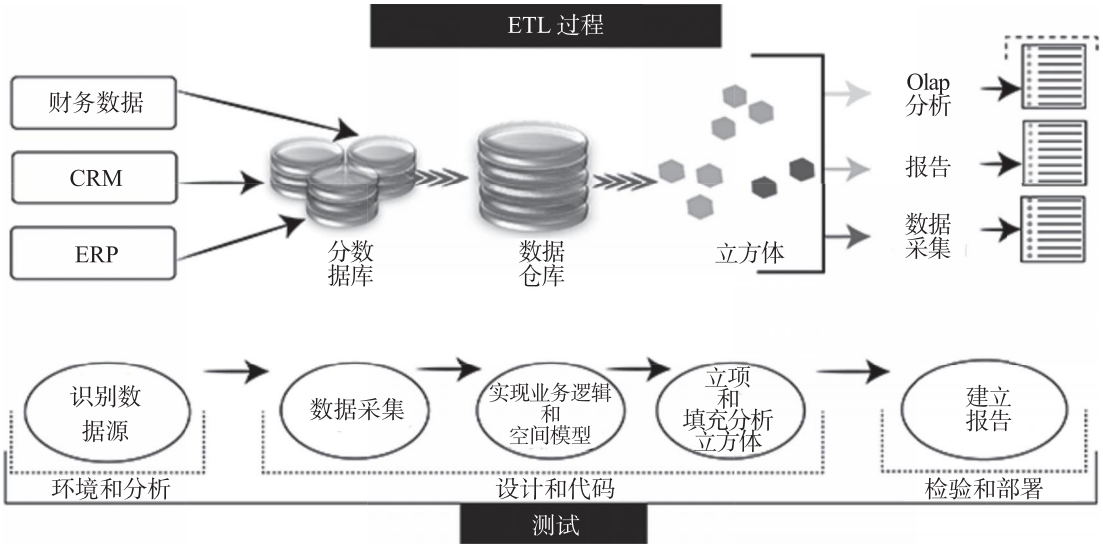


图 3-14　ETL 测试

3.5.2 ETL 测试的过程

传统的测试逻辑以设计输入为主，即一个测试用例主要考虑：该用例的输入数据对程序逻辑的覆盖程度，而且结果验证是相对简单的。这里称之为单维度导向的测试 。

而 ETL 测试则多加了一个方面，对输出数据的验证逻辑也同样需要合理的设计 ，否则就无法准确判断输出结果是否正确，也就谈不上正确测试了。既需要考虑输入数据的逻辑覆盖情况，又需要合理设计输出数据的验证逻辑，这里称之为双维度导向的测试。

ETL 测试分五个阶段进行：

（1）ETL 测试可识别数据源和要求。

（2）数据恢复。

（3）实现维度建模和业务逻辑。

（4）构建和填充数据。

（5）构建报告。

3.5.3 ETL 测试的类型

（1）新数据仓库测试：它是从核心构建和验证的。在此测试中，输入取自客户的要求和不同的数据源。但是，新的数据仓库是在 ETL 工具的帮助下构建和验证的。以下是不同用户群体的责任。

业务分析师：业务分析师收集并记录需求。

基础设施人员：这些人建立了测试环境。

QA 测试人员：QA 测试人员开发测试计划和测试脚本，然后执行这些测试计划和测试脚本。

开发人员：开发人员为每个模块执行单元测试。

数据库管理员：数据库管理员测试性能和压力。

用户：用户进行功能测试，包括 UAT（用户验收测试）。

笔记

（2）生产验证测试：当数据移动到生产系统时，对数据进行此测试。Informatica 数据验证选项提供 ETL 测试和管理功能的自动化，以确保数据不会危及生产系统。

（3）目标测试源（验证）：完成此类测试，以验证转换了预期数据值的数据值。

（4）应用程序升级：自动生成此类 ETL 测试，从而节省测试时间。此类测试检查从较旧的应用程序中提取的数据是否与新应用程序中的数据完全相同。

（5）元数据测试：元数据测试包括测量数据类型、数据长度和检查索引 / 约束。

（6）数据准确性测试：完成此测试是为了确保数据按预期准确加载和转换。

（7）数据转换测试：在许多情况下完成数据转换测试。需要为每一行运行多个 SQL 查询，以验证转换规则。

（8）数据质量测试：数据质量测试包括语法和参考测试。为避免在业务流程期间由于日期或订单号而导致的任何错误，数据质量已完成。

（9）语法测试：它将根据无效字符、字符模式、不正确的大小写顺序等报告“脏”数据。

（10）参考测试：它将根据数据模型检查数据。例如，客户 ID 数据质量测试包括数字检查、日期检查、精确检查等。

（11）更改请求：在这种情况下，数据已添加到现有数据仓库。可能存在客户需要更改当前业务规则的情况，或者他们可以集成新规则。

（12）报告测试：数据仓库的最终结果，报告测试。应通过验证报告中的数据和布局测试。报告是创建重要业务决策的重要资源。

3.5.4 ETL 测试与数据库测试的区别

ETL 测试和数据库测试都涉及数据验证，但两者不相同。ETL 测试通常对数据仓库中的数据执行，而数据库测试则在事务系统上执行。数据从不同的应用程序进入事务数据库。ETL 测试和数据库测试的区别见表 3-13。

1）ETL 测试涉及的操作

（1）验证从源到目标系统的数据移动。

（2）源系统和目标系统中的数据计数验证。

（3）ETL 测试根据要求和期望验证转换，提取。

（4）ETL 测试验证表关系是否在转换期间连接并且密钥是保留者。

（5）在数据库测试中执行的操作。

（6）数据库测试侧重于数据准确性、数据的正确性和有效值。

2）数据库测试涉及的操作

（1）数据库测试侧重于验证具有有效数据值的表中的列。

（2）要验证是否维护主键或外键，请使用数据库测试。

（3）数据库测试用于验证列中是否缺少数据。在这里是检查列中是否有任何空值应具有有效值。

（4）验证列中数据的准确性。例如，月份数列的值不应大于 12。

表 3-13　ETL 测试和数据库测试的区别

功能 / 项目	ETL 测试	数据库测试
首要目标	执行 ETL 测试以进行 BI 报告的数据提取、转换和加载	执行数据库测试，以验证和集成数据
业务需求	ETL 测试用于预测信息和分析报告	数据库测试用于集成来自多个应用程序和服务器影响的数据
适用系统	ETL 测试包含无法在业务流环境中使用的历史数据	数据库测试包含业务流程所在的事务系统
建模	使用多维方法	使用 ER 方法
数据库类型	ETL 测试适用于 OLAP 系统	数据库测试用于 OLTP 系统
数据类型	ETL 使用具有较少连接、更多索引和聚合的反规范化数据	数据库使用带连接的规范化数据
常用工具	使用 QuerySurge、Informatica 等工具	QTP、Selenium 工具用于数据库测试

笔记

3.5.5 ETL 性能测试

ETL 性能测试用于确保 ETL 系统是否可以处理多个用户和事务的预期负载。性能测试涉及 ETL 系统上的服务器端工作负载。

ETL 性能测试的步骤如下：

（1）找出生产中转化的负荷。

（2）将创建相同负载的新数据或将其从生产数据移动到本地服务器。

（3）禁用 ETL，直到生成所需的代码。

（4）从数据库表中计算所需的数据。

（5）记下 ETL 的最后一次运行并启用 ETL。它将获得足够的压力来转换已创建并运行它的整个负载。

（6）完成 ETL 后，计算创建的数据。

应注意的基本表现如下：

（1）找出转换负载所需的总时间。

（2）找出已改进或删除的性能。

（3）检查是否提取并转移了整个预期负载。

3.5.6 ETL 测试的数据准确性

在 ETL 测试中，我们专注于数据准确性，以确保数据是否按照预期准确加载到目标系统。

以下是执行数据准确性应遵循的步骤：

（1）比较值。在比较值中，将源系统和目标系统中的数据与最小数据或无转换数据进行比较。可以使用各种 ETL 工具进行 ETL 测试，例如 Information 中的源限定符转换。

表达式转换也可以在数据准确性测试中执行。可以在 SQL 语句中使用一组运算符检查源和目标系统中的数据准确性。

（2）检查关键数据列。可以通过比较源系统和目标系统中的不同值检查关键数据列。

```
SELECT cust_name, order_id, city, count(*)  FROM customer GROUP BY cust_name, order_id, city;
```

笔记

3.5.7 数据转换中的 ETL 测试

执行数据转换非常复杂，因为无法通过编写单个 SQL 查询并将其输出与目标进行比较来实现。要对数据转换进行 ETL 测试，必须为每一行编写多个 SQL 查询，以验证转换规则。要为数据转换执行成功的 ETL 测试，需要从源系统中选择足够的样本数据以应用转换规则。

执行 ETL 测试以进行数据转换的重要步骤是：

（1）为输入数据和预期结果创建方案。现在我们将与业务客户验证 ETL 测试。ETL 测试是在设计期间收集需求的最佳方法，可用作测试的一部分。

（2）根据场景创建测试数据。ETL 开发人员将使用场景电子表格自动化填充数据集的整个过程，以便在情况发生变化时提供多功能性和移动性。

（3）利用数据分析结果比较源和目标数据之间每个字段中的值的范围和提交。

（4）验证 ETL 生成字段的准确处理，例如代理键。

（5）验证仓库中与数据模型或设计中指定的数据类型相同的数据类型。

（6）在测试参照完整性的表之间创建数据的场景。

（7）验证数据中的父对子关系。

（8）执行查找转换。查找查询应该是直接的，没有任何数据收集，并且预期根据源表只返回一个值。可以直接在源限定符中加入查找表。如果不是这种情况，我们将编写一个查询，它将查找表与源表中的主表连接，并比较目标中相应列中的数据。

3.5.8 ETL 测试用例

ETL 测试的目的是确保业务转换后从源到目标的加载数据是准确的，适用于信息管理行业中的不同工具和数据库，见表 3-14。

表 3-14　测试用例表

ETL 测试场景	测试用例
映射文档验证	将验证映射文档是否提供了 ETL 信息，日志更改应保留在每个映射文档中
验证	（1）使用相应的映射文档验证目标和源表结构； （2）源表和目标表的数据类型应该相同； （3）源和目标的数据类型的长度应该相同； （4）验证数据字段类型和指定的格式； （5）源数据类型的长度不应小于目标数据类型的长度
约束验证	应根据用户的期望为特定表定义约束
数据一致性问题	（1）数据类型和特定属性的长度可以通过语义定义在文件或表中变化； （2）完整性约束的滥用
完整性问题	（1）在这里，必须确保将所有预期的数据都加载到目标表中； （2）在源和目标之间比较记录计数； （3）检查被拒绝的记录； （4）不应在截断表的列中截断数据； （5）检查边界值分析； （6）比较仓库中加载的数据和源数据之间关键字段的唯一值
正确性问题	（1）此方案用于更正拼写错误或不准确记录的数据； （2）若更正数据，则为 Null，非唯一且超出范围

续表

ETL 测试场景	测试用例
转型 / 转换	此方案用于检查转换
数据质量	（1）此方案用于检查数字并验证它； （2）数据检查：此方案将遵循日期格式，并且对于所有记录应该相同； （3）精确检查； （4）数据检查； （5）Null 检查
Null 验证	此方案将验证 Null 值，其中为特定列指定 not Null 值
重复检查	（1）检查唯一键、主键和任何其他列的验证，根据具有任何重复行的业务需求，该列应该是唯一的； （2）检查从多个列源中提取的任何列中是否存在任何重复值，并将它们组合成一列； （3）根据客户端要求，需要确保在仅具有目标的多个列的组合中没有重复项
日期验证	（1）使用开发中的许多区域了解行创建日期； （2）根据 ETL 开发视角识别现有记录； （3）有时在日期值上会更新和插入
数据清洁度	在加载到临时区域之前，应删除不必要的列

在 ETL 测试性能期间，ETL 测试程序始终使用的两个文档是：

（1）ETL 映射表：ETL 映射表包含源表和目标表的所有信息，包括每个列及其在引用表中的查找。ETL 测试程序需要熟悉 SQL 查询，因为 ETL 测试可能涉及编写具有多个连接的大查询，以在 ETL 的任何阶段验证数据。在编写数据验证查询时，ETL 映射表提供了重要帮助。

（2）源（目标）的 DB 模式：应该可以访问它，以验证映射表中的任何细节。

3.6 ETL 测试工具

ETL 测试工具是用于提取、转换和加载数据的软件。提取、转换和加载有助于组织，使数据在不同的数据系统中可访问、有意义且可用。在当今数据驱动的世界中，无论大小，都会从各种组织、机器和小工具中生成大量数据。

在传统的编程方式中，ETL 都提取并进行一些转换操作，然后将转换后的数据加载到目标数据库文件等。为此，需要用编程语言编写代码，如使用 Java、C#、C++ 等。为了避免使用更多的编码和库，将通过拖放组件减少工作量。

ETL 工具是一组用任何编程语言编写的库，它将简化我们的工作，以便根据需要进行数据集成和转换操作。

例如，在移动设备中，每次浏览网页时，都会生成一定数量的数据。商用飞机每小时可以生成高达 500 GB 的数据。我们现在可以想一想，这些数据有多大。这就是它被称为大数据的原因，但是在对它执行 ETL 操作之前，这些数据是无用的。

3.6.1 ETL 工具的优势

数据仓库工具包含来自不同来源的数据，这些数据在一个地方组合，以分析有意义的模式和洞察力。ETL 处理异构数据并使其同质化，这对数据科学家来说非常顺利。然后，数据分析师分析数据并从中获取商业智能。

笔记

与传统的移动数据方法相比，ETL 更容易使用，这涉及编写传统的计算机程序。ETL 工具包含一个图形界面，可以增加源数据库和目标数据库之间映射表和列的过程。

ETL 工具可以从多个数据结构以及不同平台（如大型机、服务器等）收集、读取和迁移。它还可以在发生变化时识别“增量”变化，使 ETL 工具能够仅复制已更改的数据，而无须执行完整的数据刷新。

ETL 工具包括即用型操作，如过滤、排序、重新格式化、合并和连接。ETL 工具还支持转换调度、监控、版本控制和统一元数据管理，同时一些工具与 BI 工具集成。

使用 ETL 工具比使用将数据从源数据库移动到目标数据存储库的传统方法更有益。

使用 ETL 工具的优点如下。

（1）易用性：ETL 工具的首要优点是易于使用。该工具本身指定数据源以及提取和处理数据的规则，然后实现该过程并加载数据。ETL 消除了编程意义上的编码需求，我们必须编写程序和代码。

（2）运营恢复能力：许多数据仓库都已损坏并产生运营问题。ETL 工具具有内置的错误处理功能，它可以帮助数据工程师构建 ETL 工具的功能，以开发成功且装备精良的系统。

（3）可视流程：ETL 工具基于图形用户界面，提供系统逻辑的可视化流程。图形界面帮助我们使用拖放界面指定规则，以显示流程中的数据流。

（4）适用于复杂数据管理情况：ETL 工具有助于更好地移动大量数据并批量传输。在复杂规则和转换的情况下，ETL 工具简化了任务，这有助于我们进行计算、字符串操作、数据更改，以及多组数据的集成。

（5）增强商业智能：ETL 工具可改善数据访问并简化提取、转换和加载过程。它改善了对直接影响战略和运营决策的信息的访问，这些决策基于数据驱动的事实。ETL 还使业务负责人能够检索基于特定需求的数据，并根据这些需求做出决策。

（6）推进数据分析和清理：与 SQL 中提供的工具相比，ETL 工具具有大量的清理功能。高级功能关注复杂的转换需求，这通常发生在结构复杂的数据仓库中。

（7）增强的商业智能：ETL 工具改进了数据访问，因为它简化了提取、转换和加载的过程。ETL 有助于直接访问信息，从而影响战略和运营决策，这些决策都基于数据驱动的事实。ETL 工具还使业务负责人能够根据其特定需求检索数据，并相应地做出决策。

（8）高投资回报：使用 ETL 工具可以节省成本，使企业获得更高的收益。根据国际数据公司的研究，发现这些实施收集的中位数 5 年投资回报率为 112%，平均回报期为 1.6 年。

（9）性能：ETL 平台的结构简化了构建高质量数据仓库系统的过程。一些 ETL 工具带有性能增强技术，如集群感知和对称多处理。

3.6.2 ETL 工具的类型

随着 ETL 工具的日益普及，数据仓库市场已经看到商用设备的重要性。ETL 工具提供了各种功能，以促进工作流程。

1. 常用工具

ETL（extract-transform-load）即数据抽取、转换、装载的过程，对于企业或行业应用

来说，我们经常会遇到各种数据的处理、转换、迁移，所以，了解并掌握一种 ETL 工具的使用必不可少。最近用 Kettle 做数据处理比较多，所以介绍一下这方面的内容，这里先对比几款主流的 ETL 工具。

笔记

（1）DataPipeline。DataPipeline 是一家为企业用户提供数据基础架构服务的科技公司，DataPipeline 数据质量平台整合了数据质量分析、质量校验、质量监控等多方面特性，以保证数据质量的完整性、一致性、准确性及唯一性，彻底解决数据孤岛和数据定义进化的问题。

（2）Kettle。Kettle 是一款国外开源的 ETL 工具，纯 Java 编写，可以在 Windows、Linux、UNIX 上运行，数据抽取高效、稳定。Kettle 的中文名称为水壶，该项目的主程序员 MATT 希望把各种数据放到一个壶里，然后以一种指定的格式流出。

Kettle 家族目前包括 4 个产品：Spoon、Pan、CHEF、Kitchen。

① Spoon 允许通过图形界面设计 ETL 转换（transformation）过程。

② Pan 允许批量运行由 Spoon 设计的 ETL 转换（例如使用一个时间调度器）。Pan 是一个后台执行的程序，没有图形界面。

③ CHEF 允许创建任务（job）。任务通过允许每个转换、任务、脚本等，更有利于自动化更新数据仓库的复杂工作。任务通过允许每个转换、任务、脚本等，将会被检查，看是否已经正确运行。

④ Kitchen 允许批量使用由 Chef 设计的任务（例如使用一个时间调度器）。Kitchen 也是一个后台运行的程序。

（3）Talend。Talend 是一家专业的开源集成软件公司，为企业提供开源的中间件解决方案，从而让企业能够在他们的应用、系统，以及数据库中赢取更大的价值。Talend 可运行于 Hadoop 集群之间，直接生成 MapReduce 代码供 Hadoop 运行，从而可以降低部署难度和成本，加快分析速度。而且 Talend 还支持可进行并发事务处理的 Hadoop 2.0。

（4）Informatica。Informatica 是全球领先的数据管理软件提供商，在如下的 Gartner 魔力象限位于领导者地位：数据集成工具魔力象限、数据质量工具魔力象限、元数据管理解决方案魔力象限 、主数据管理解决方案魔力象限 、企业级集成平台即服务（EiPaaS）魔力象限。

Informatica Enterprise Data Integration 包括 Informatica PowerCenter 和 Informatica PowerExchange 两大产品，凭借其高性能、可充分扩展的平台，可以解决几乎所有数据集成项目和企业集成方案。

Informatica PowerCenter 用于访问和集成几乎任何业务系统、任何格式的数据，它可以按任意速度在企业内交付数据，具有高性能、高可扩展性、高可用性的特点。Informatica PowerCenter 包括 4 个版本，即标准版、实时版、高级版、云计算版。同时，它还提供了多个可选的组件，以扩展 Informatica PowerCenter 的核心数据集成功能，这些组件包括：数据清洗和匹配、数据屏蔽、数据验证、Teradata 双负载、企业网格、元数据交换、下推优化（pushdown optimization）、团队开发和非结构化数据等。

Informatica PowerExchange 是一系列的数据访问产品，它确保 IT 机构能够根据需要随时随地访问并在整个企业内传递关键数据。凭该能力，IT 机构可以优化有限的资源和数据的业务价值。Informatica PowerExchange 支持多种不同的数据源和各类应用，包括企业

笔记

应用程序、数据库和数据仓库、大型机、中型系统、消息传递系统和技术标准。

（5）DataX。DataX 是阿里巴巴集团内被广泛使用的离线数据同步工具 / 平台，实现包括 MySQL、Oracle、SQL Server、Postgre、HDFS、Hive、ADS、HBase、TableStore（OTS）、MaxCompute（ODPS）、DRDS 等各种异构数据源之间高效的数据同步功能。

2. ETL 工具的功能

基于 ETL 工具的数据仓库使用临时区域、数据集成和访问层执行其功能，由三层结构组成。

（1）暂存层：临时数据库或暂存层用于存储来自不同源数据系统的提取数据。

（2）数据集成层：集成层转换来自暂存层的数据，并将数据移动到数据库。在数据库中，数据被排列成层级组，称为维度、事实和聚合事实。数据仓库系统中维度表和事件的组合称为模式。

（3）访问层：最终用户使用访问层检索分析报告或功能的数据。

任何 ETL 工具都应该有能力连接到类型广泛的数据源和数据格式。对最常用的关系型数据库系统，还要提供本地的连接方式（如对 Oracle 的 OCI），ETL 应该能提供下面的基本功能：

（1）能连接到普通关系型数据库并获取数据，如常见的 Oracle、MS SQL Server、IBM DB/2、Ingres、MySQL 和 PostgreSQL。还有很多，从有分隔符和固定格式的 ASCII 文件中获取数据，从 XML 文件中获取数据，从流行的办公软件中获取数据，如 Access 数据库和 Excel 电子表格使用 FTP、SFTP、SSH 方式获取数据（最好不用脚本），还能从 Web Services 或 RSS 中获取数据。如果还需要一些 ERP 系统里的数据，如 Oracle E-Business Suite、SAP/R3、PeopleSoft 或 JD/Edwards，ETL 工具也应该提供到这些系统的连接。还能提供 Salesforce.com 和 SAP/R3 的输入步骤，但不是套件内，需要额外安装。对于其他 ERP 和财务系统的数据抽取，还需要其他解决方案。当然，最通用的方法是要求这些系统导出文本格式的数据，将文本数据作为数据源。

（2）一个 ETL 工具应该能在任何平台下甚至是能在不同平台的组合上运行。一个 32 位的操作系统在开发的初始阶段可能运行得很好，但是当数据量越来越大时，就需要一个更强大的操作系统。另一种情况，开发一般在 Windows 或 Mac 上运行，而生产环境一般是 Linux 系统或集群，你的 ETL 解决方案应该可以无缝地在这些系统间切换。

（3）一个 ETL 工具应该留给开发人员足够的自由度来使用，而不能通过一种固定的方式限制用户的创造力和设计的需求。ETL 工具可以分为基于过程的工具和基于映射的工具。

基于映射的功能只在源数据和目的数据之间提供一组固定的步骤，严重限制了设计工作的自由度。基于映射的工具一般易于使用，可快速上手，但是对于更复杂的任务，基于过程的工具才是最好的选择。

使用 Kettle 这种基于过程的工具，根据实际数据和需求，可以创建自定义的步骤和转换。

（4）设计完的 ETL 转换应该可以被复用，这是非常重要的。复制和粘贴已经存在的转换步骤是最常见的一种复用，但这不是真正意义上的复用。

Kettle 里有一个映射（子转换）步骤，可以完成转换的复用，该步骤可以将一个转换

作为其他转换的子转换。另外，转换还可以在多个作业里多次使用，同样，作业也可以为其他作业的子作业。

笔记

（5）几乎所有的 ETL 工具都提供了脚本，以编程的方式解决工具本身不能解决的问题。另外，还有少数几款 ETL 工具可以通过 API 或其他方式为工具增加组件。使用脚本语言写函数，函数可以被其他转换或脚本调用。

Kettle 提供了上述的所有功能。Java 脚本步骤可以用来开发 Java 脚本，把这个脚本保存为一个转换，再通过映射（子转换）步骤，又可以变为一个标准的、可以复用的函数。实际上，并不限于脚本，每个转换都可以通过这种映射（子转换）方式复用，如同创建了一个组件。Kettle 在设计上就是可扩展的，它提供了一个插件平台。这种插件架构允许第三方为 Kettle 平台开发插件。

Kettle 里的所有插件，即使是默认提供的组件，实际上也都是插件。内置的第三方插件和 Pentaho 插件的唯一区别是技术支持。假设你买了一个第三方插件（例如一个 SugarCRM 的连接），则技术支持由第三方提供，而不是由 Pentaho 提供。

（6）ETL 项目很大一部分工作都是在做数据转换。在输入和输出之间，数据要经过校验、连接、分隔、合并、转置、排序、合并、克隆、排重、过滤、删除、替换或者其他操作。

3.7 ETL 管道

3.7.1 ETL 管道概念

ETL 管道是指一组过程，这些过程从输入源中提取数据、转换数据，并将其加载到输出目标（如数据集市、数据库和数据仓库）中，以进行分析、报告和数据同步，如图 3-15 所示。

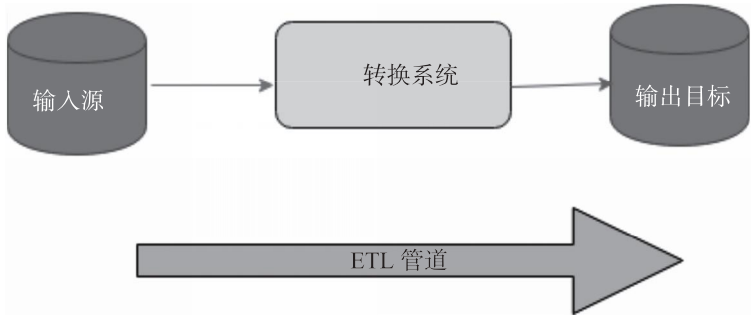


图 3-15　报告和数据同步的过程

当数据加载到数据库或数据仓库时，数据管道不会结束。ETL 目前正在发展，因此它可以支持跨事务系统、运营数据存储、MDM 中心、云和 Hadoop 平台的集成。由于非结构化数据的增长，数据转换过程变得更加复杂。例如，现代数据流程包括实时数据；又如，来自广泛的电子商务网站的网络分析数据。Hadoop 是大数据的代名词，它开发了几种基于 Hadoop 的工具处理 ETL 过程的不同方面。我们可以使用的工具取决于数据的结构、批量或处理的数据流。

笔记

3.7.2 ETL 管道与数据管道的区别

ETL 管道和数据管道几乎都做同样的事情。它们跨平台移动数据并以此方式对其进行转换。ETL 管道与数据管道的主要区别在于构建管道的应用程序。

1. ETL 管道

ETL 管道是为数据仓库应用程序构建的，包括企业数据仓库以及特定于主题的数据集市。当新应用程序替换传统应用程序时，ETL 管道也用于数据迁移解决方案。ETL 管道通常使用精通转换结构化数据的行业标准 ETL 工具构建，如图 3-16 所示。

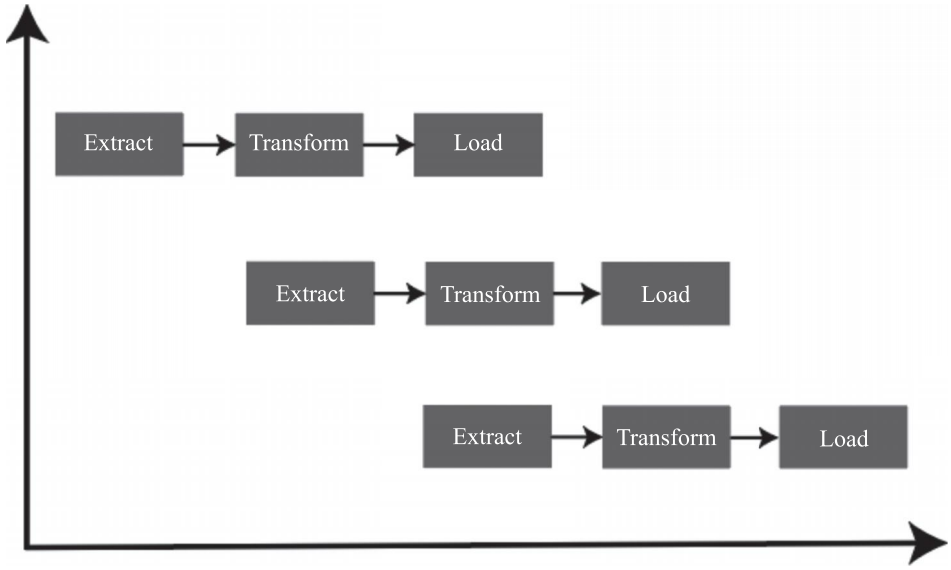


图 3-16　使用行业标准 ETL 工具构建

2. 数据管道

可以为使用数据带来值的任何应用程序构建数据管道。它可用于跨应用程序集成数据，构建数据驱动的 Web 产品，构建预测模型，创建实时数据流应用程序，执行数据挖掘活动，构建数字产品中的数据驱动功能。随着开源大数据技术（用于构建数据管道）的可用性增强，过去十年数据管道的使用有所增加。这些技术能够转换非结构化数据和结构化数据。ETL 管道和数据管道的区别见表 3-15。

表 3-15　ETL 管道和数据管道的区别

ETL 管道	数据管道
ETL 管道定义为从一个系统中提取数据，转换并将其加载到某个数据库或数据仓库的过程	数据管道是指将数据从一个系统移动到另一个系统并沿途转换数据的任何处理元素集
ETL 管道表示管道分批工作。例如，管道每 12 小时运行一次	数据管道也可以作为流评估运行（即每个事件在发生时进行处理）。数据管道类型是 ELT 管道（将整个数据加载到数据仓库并稍后进行转换）