# 数据仓库系统综合测试策略的研究①

## Study on Synthetically Testing Strategy in Data Warehouse

谷 岩 （广州大学 510091）

**摘要**：数据仓库系统的软件测试是一个综合测试过程，本文给出了数据仓库系统的测试策略及测试方法。由于采用的测试方法不可能彻底发现系统的所有错误，因此本文给出了测试终止的标准。

**关键词**：数据仓库 单元测试 组装集成测试 加载测试 交付测试 测试策略

## 1 引言

测试是保证系统可靠性的重要手段，数据仓库系统测试与一般软件测试不同的是，数据仓库的测试不仅包括对软件系统的测试，同时包括对数据仓库的测试。在测试过程中必须保证测试的充分性，同时注意测试数据的覆盖。由于数据仓库系统错误的复杂性，在软件工程范围内需要综合应用测试技术，根据定义域中的取值，通过执行和观察，将预期的行为和实际的行为作比较，以确认测试的结果，因此数据仓库系统测试是一个综合测试的过程。综合测试分为 4 个步骤，即单元测试、组装集成测试、加载测试和交付测试。图 1 描述了整个测试过程。

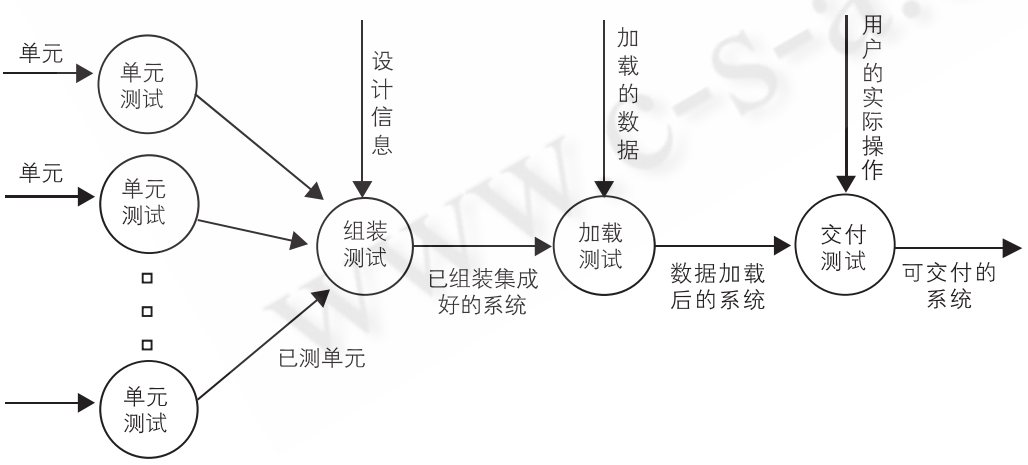


图 1 数据仓库系统的软件测试步骤

## 2 单元测试

单元测试是对系统中的每个单元进行独立测试。该测试从系统的内部结构出发，以详细设计的说明为指导，测试单元内的重要控制路径，力求在单元范围内发现错误。由于单元测试的目的在于发现各单元内部可能存在的各种错误，因此单元测试往往采用白盒测试法。而且多个单元可以平行地独立进行单元测试。

单元测试除了对单元进行功能测试，以检查其功能是否满足系统的规格说明中的需求之外，还要重点对单元的下列基本性能进行评价：

（1）单元接口。对通过被测单元的数据流进行测试，以检查数据能否正确地输入和输出数据仓库。测试要点是：重点检查调用和被调用单元之间的参数个数、属性、次序等是否一致；单元内部函数的参数个数、属性、次序是否正确；数据的输入与输出是否正确；文件属性及其读写是否正确；全局变量的定义在各单元中是否一致。

（2）局部数据结构。单元的局部

① 基金项目：广东省科技攻关项目（2004B10101044） 广州市科技攻关项目（2004Z3－D0391）

数据结构是常见的错误来源,应设计测试用例以发现下列错误:

① 不正确的或不一致的数据类型说明;

② 错误的初始化或缺省值;

③ 使用尚未赋值或尚未初始化的变量;

④ 不正确的变量名;

⑤ 不一致的数据类型;

⑥ 数据的上下溢出或引用错。此外,也应对全局变量数据结构进行相应的测试。

(3) 路径测试。选择适当的测试用例对单元中重要的执行路径进行测试,以发现单元中不正确的计算、错误的比较或不正常的控制流而导致的错误。

(4) 出错处理。当单元在运行过程中出错,系统应当具有较强的出错处理能力,以保证其逻辑上的正确性。当评价出错处理性能时,所应进行测试的可能错误有:

① 出错的描述难以理解;

② 指出的错误并不是所遇到的错误;

③ 出错的描述不足以确定出错的原因;

④ 对错误条件的处理不正确;

⑤ 出错时还未进行出错处理就先进行系统干预等。

(5) 边界测试。边界测试是单元测试步骤中的最后任务。因为在取值范围的边界上出现错误是软件的常见错误,所以应特别注意测试数据流和控制流中小于、等于或大于确定的比较值时出错的可能性。

单元的功能需求是设计测试用例的依据。每个测试用例都要有确定的预期结果,这样才能判断模块是否包含错误。在对单元进行测试时,每个单元在整个软件系统中不是孤立的,不能独立运行,它需要由其他单元来调用和驱动,单元的执行还依赖于被它调用的下级单元。因此为了模拟单元与它周围单元的关系,需要设置若干辅助测试单元。辅助测试单元分两种:

第一,驱动单元(driver)。用来模拟所测单元的主程序。它接受测试数据,并向被测单元传送测试数据,启动被测单元,回收并输出测试结果。

第二,桩单元(stub)。用来模拟被测单元在执行过程中所要调用的单元。它接受被测单元输出的数据并完成它所指派的任务。桩单元采用原来子单元的接口,可做很少的数据处理。

图 2 表示了一个被测单元进行单元测试的模拟测试环境。其中设置了一个驱动单元和三个桩单元。驱动单元在测试中接受测试数据,把相关的数据传给被测单元,启动被测单元,并打印出相应的结果。桩单元由被测单元调用,它们仅做很少的数据处理,例如打印入口和返回,以便于检查被测单元与其下级单元的接口,并最后将控制返回到它的上级单元。驱动单元和桩单元的使命在单元测试结束后终止。

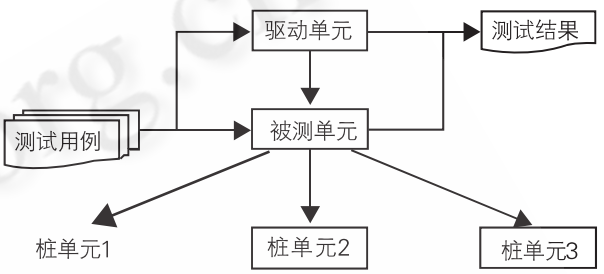


图 2　单元测试的测试环境

如果设计出来的单元内集程度高,且一个单元只有一个功能,那么单元测试就简单易行。但实际上许多单元很难进行充分的单元测试,特别是单元间接口的全面测试,因此必须进行下一步的组装集成测试。

## 3　组装集成测试

通过了单元测试的单元,往往需要按照系统开发模型和设计框架的要求组装成一个完整的系统,在组装过程中进行的测试称为组装集成测试。而在组装过程中,程序还会出现许多错误,其原因是:

(1) 由于单元与单元之间存在着许多较复杂的接口,稍有疏忽就会出错。比如数据在通过接口时可能丢失,全局变量的数据结构在单元引用时出错等;

(2) 由于在单元测试中使用了驱动单元和桩单元,它们只是对真实单元的一个简化,与真实单元并不完全等效,因此真实单元组装后,就会出现一些错误;

(3) 单元测试中的单元在测试时仅局限在单元内部,当组装在一起后,一个单元很可能会对另一个单元产生不利的影响;

(4) 在单元测试中单个单元的允许误差,在单元被组装起来后,其误差往往会被放大,甚至会达到无法容忍的地步。

组装集成测试的目标是发现与单元接口有关的错

误,逐步将经过单元测试的单元构成一个满足设计要求的软件结构。组装集成测试应完成的任务是:

① 制定综合测试的策略。根据程序结构选择非增量式测试法或增量式测试法;

② 确定综合测试的步骤,设计测试用例;

③ 测试过程中,在已通过单元测试的基础上,逐一添加模块,每添加一个单元,重点捕捉接口错误,同时重复进行老的测试;

④ 编制综合测试报告。下面介绍组装策略。

### 3.1 非增量式测试

这种测试的基本思路是:首先将各单元独立进行单元测试,然后把所有单元组装在一起进行测试,最终得到一个符合要求的软件系统。

例:被测系统的结构图如图 3 所示,它由 6 个单元构成。下面我们采用非增量式测试法进行系统的测试。设计方法如下:

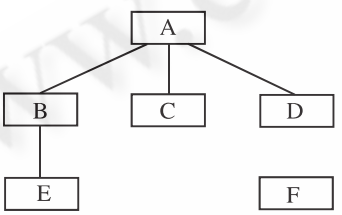


图 3　程序结构图

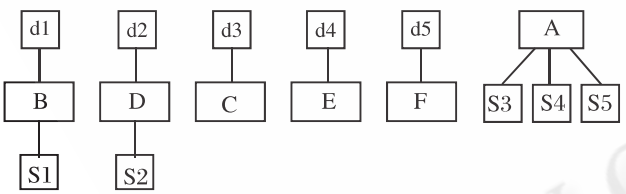


图 4　非增量式测试例

(1)获至宝 对每个独立单元进行单元测试。

根据这 6 个单元在程序结构图中的地位,对 B 和 D 模块分别设计了驱动单元和桩单元,因为它们有上下级单元;对 C、E 和 F 模块只设计驱动单元,因为它们只有上级单元,而没有下级单元;对 A 单元,由于它是主控单元,并调用 3 个下级单元,因此,只为它设计 3 个桩单元。模拟结构图见图 4。

(2) 按程序结构图将 6 个单元组装起来进行组装集成测试。

### 3.2 增量式测试

这种测试的基本思路是:首先将各单元独立地进行单元测试,然后将这些单元组装成较大的系统,在组装过程中边连接边测试,以发现在组装时产生的错误,最终组装成一个符合要求的数据仓库系统。增量式测试可按不同的次序实施,有两种测试策略:

#### 3.2.1 自顶向下增量式测试

该测试方法是按照程序结构图,首先利用桩单元测试主单元,通过测试后,用实际的单元替代桩单元进行测试,重复上述步骤,直至替代了所有桩单元。因此,该测试过程是按结构自顶向下的过程。

在测试过程中,决定单元测试次序的基本原则是:

(1) 尽早测试关键的单元。所谓关键的单元是指比较重要的、比较复杂的、较可能出错或含有新算法的单元;

(2) 尽早测试包含输入、输出功能的单元。

例: 被测系统的结构图如图 3 所示。下面采用自顶向下增量式测试法进行系统的测试。设计方法如下:

① 对顶层的 A 单元,由于它是主控单元,并调用 3 个下级单元,因此为它设计 3 个桩单元 s1、s2、s3,以模拟被 A 单元调用的 B、C、D 三单元,并进行测试。见图 5(a)。

② 将 B、C、D 三单元分别替代桩单元与 A 单元连接,并对 B 和 D 单元配置桩单元 s4 和 s5,以模拟 A 单元对 E 和 F 单元的调用,并进行测试。见图 5(b)。

③ 将 E 和 F 单元分别替代桩单元 s4 和 s5 并与 B 和 D 单元连接,并进行最后的测试。见图 5(c)。

#### 3.2.2 自底向上增量式测试

该测试方法是按照程序结构图,首先利用驱动单元测试最底层单元,通过测试后,用实际的单元替代驱动单元进行测试,重复上述步骤,直至替代了所有驱动单元。因此,该测试过程是按结构自底向上的过程。在测试过程中,决定测试单元的基本原则是:该单元的所有下级单元都已测试过了。

例: 被测程序的结构图如图 3 所示。下面采用自底向上增量式测试法进行系统的测试。设计方法如下:

(1) 将 E、C 和 F 单元分别配以驱动单元 d1、d2 和 d3,以模拟 B、A 和 D 单元对它们的调用,并进行测试。见图 6(a)、(b)和(c)。

(2) 将 B 和 D 单元分别替代驱动单元并与 E 和 F 单元连接,并为 B 和 D 单元配置驱动单元 d4,以模拟 A 单元

对 B 和 D 单元的调用,并进行测试。见图 6(d)和(e)。

(3)最后按图 6(f)的形式完成系统整体的测试。

## 4 加载测试

在完成系统的组装集成测试后,就要进行数据仓库的首次加载,完成数据加载后,需要从数据质量、可用性和性能等方面测试用户对数据仓库的满意程度。在测试过程中,我们可以使用两个专门针对数据仓库平台的测试指标进行辅助测试:

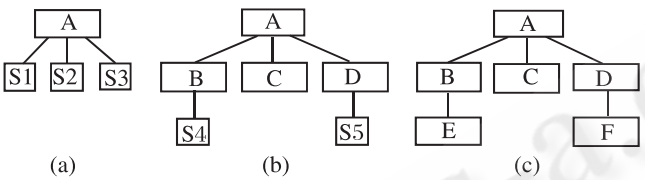


图 5 自顶向下增量式测试例

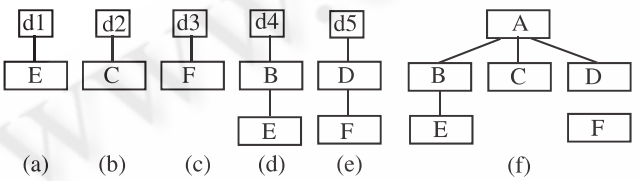


图 6 自底向上增量式测试例

### 4.1 TPC – D 测试

对于数据仓库系统,衡量其数据仓库性能的主要指标是 TPC – D,它主要从 3 方面对数据仓库进行测试:

(1) QppD:描述系统的查询处理能力;

(2) QthD:即流量测试结果,主要描述系统在多个用户同时进行查询时的处理能力,实际上它代表了系统的并行处理能力;

(3) QphD:即价格性能比。显然,前面两个指标的数据越大越好,而最后一个则越小越好。当然,首先要考虑的是系统能否满足业务上的需求。需要说明的是流量测试结果,尽管它描述了系统处理并发查询请求的能力,但这种流量测试往往大多数都是在多用户状态下进行的。实际上 TPC – D 给出了两种测试方式的选择:

① 直接进行多用户状态下的流量测试;

② 或者先在单用户状态下进行测试,然后利用测得的处理能力指标 QppD,并用流量指标的计算公式计算出 QthD。那么在实际的测试过程中如何区分这两种测试结果呢? 只要把 TPC – D 的测试概要下载并打印出来,就可以了解在做流量测试时的 Stream 数目。

### 4.2 Data Challenge 测试

除 TPC – D 以外,还有一个 Data Challenge 的测试标准。与 TPC – D 不同的是,它非常注重考察系统的动态查询能力。在测试数据仓库系统时,应该考虑以下问题:

(1) 目前业务部门有些什么急需解决的问题,这些问题借助传统的生产系统能解决吗?

(2) 目前有多大的数据量,今后的扩展要求如何,能不能做在线的升级?

(3) 系统的并行处理能力怎样? 因为它将直接影响系统处理复杂查询和动态查询的能力;

(4) 对系统的管理是否复杂? 有没有数据库重组的问题? 因为复杂的管理需要很多的数据库管理员,这种人工是非常昂贵的;

(5) 系统的高可用性和可靠性如何? 当系统发生故障时,对业务产生的影响有多大,企业对这种故障的容忍程度如何?

充分考虑了以上各方面的问题后,所投资建立的实际系统一般都能达到预期的效果。

## 5 交付测试

在数据仓库系统交付用户使用之前,需要对数据仓库系统进行交付测试。在测试过程中需要用户真实地使用数据仓库系统,并进行如下测试:

(1) 观察用户在使用数据仓库过程中与数据仓库的交互方式,用户如何使用数据仓库完成决策分析工作?

(2) 在进行管理决策分析时,数据仓库是否能够满足用户的需要?

(3) 数据仓库需要进行哪些变动才能更好地为用户服务?

(4) 通过交付测试不仅要使用户接受数据仓库,更重要的是能够清楚地定义所交付的数据仓库在实际应用中还存在哪些缺点?

(5) 收集数据仓库在可用性、改进内容、数据准确性以及数据仓库需要补充的内容等方面的用户意见。