第 3 章　信息系统审计方法

3.1　证据收集方法

3.1.1　证据收集方法概述

审计人员在审计过程中的一项重要工作是收集、整理、分析审计证据，并据此作为发表审计意见的基础。这是审计工作的核心，也是考核评定审计质量的关键。一个审计事项的完成，必须依据审计的目的、种类和范围来确定其收集和查阅的相关资料，获取相关的审计证据。不同的审计项目，其审计的依据和查看的审计资料虽然不尽相同，但收集证据的步骤和方法却大体是相同的。因此，审计人员必须坚持实事求是的科学态度，以高度的责任感，并按照一定的原则来做好审计证据的收集工作。一般来说，审计证据的收集分为事前收集、事中获取、证据鉴定三个步骤。

事前收集：主要是在审计前收集与审计项目相关的审计证据，也就是评价标准，它是衡量和判断审计对象的正确性、真实性、合法性、合规性、有效性的尺度，主要包括被审计单位自己制定的依据、上级单位起草制定下发的各类文件资料以及国家颁布实施的法律法规等。

事中获取：是审计人员在现场审计中收集的证据，这是审计证据的主要来源。现场审计证据收集的方法很多，信息系统审计与控制协会标准管理委员会提出了信息系统审计师获取审计证据的方法，包括检验、观察、询问和确认、重复操作、验算、计算、分析步骤，以及其他可接受的方法等。

证据鉴定：审计并不是证据越多越好，应以能说明问题为限。那么，审计人员通过各种途径收集到的审计证据，尽管具有证明力，但其证明力还是潜在的，还不能直接用来证明审计项目，必须对证据的相关性、重要性、真实性进行鉴定。对于证据与被审计事项没有内在联系的应果断剔除；只有在证据与证据之间存在联系，且能够相互证实时，其证据才能够被利用。在证据与证据之间内容不一致或存在矛盾的情况下，还应收集更多的相关证据进行判断。同时，对经过鉴定的证据还必须加以综合，也就是说对相关证据从总体上加以归纳、分析、整理，使其条理化。通过综合，选出最适宜的、充分的、有说服力的证据，以此作为发表审计意见和做出审计结论的重要依据。

3.1.2　收集证据的方法

本节介绍的方法与传统的审计方法基本一致，如观察法、问询法、函证法、查阅法、复核

法等，但在内容上有较大的差别。后面几节介绍的方法，如数字取证、软件测试、数据库查询等则完全超出了传统财务审计方法的范畴。

1. 观察法

观察法是指审计人员到被审单位的经营场所、信息系统使用部门以及计算机机房等有关场所进行实地察看，来证实审计事项的一种方法。通过直接地观看视察，了解信息系统操作流程的规范程度以及内部制度的执行情况等，注意其是否符合审计标准和书面资料的记载，从中发现薄弱环节和存在的问题，借以收集书面资料以外的证据。一个经验丰富的审计人员通过观察可以获得被审单位执行情况和信息系统使用情况的直观感受。

但观察法获得的信息是有限的，如观察提供的审计证据仅限于观察发生的时点，并且在相关人员已知被观察时才是有效的。信息系统操作人员、维护人员等相关人员可能采取与日常执行不同的做法，这会影响审计人员对真实情况的了解，甚至给审计人员造成错觉。

2. 问询法

问询法是指审计人员通过调查和询问被审单位内外有关人员证实审计事项的一种方法。问询法可以发现书面资料未能详尽提供的信息以及书面资料本身存在的问题，从而弄清事实真相，取得审计证据。

问询法又分为面询法和问卷法。面询法是由审计人员向被审单位有关人员当面征询意见、核实情况的一种方法。面询法有集体询问、小范围询问、个别询问等多种方式。集体询问指将众人集中于一处，采取会议式的群体询问的方法，适于范围广泛。小范围询问指对二至三人进行查询取证。小范围询问既有别于集体询问，也不同于个别询问，它是分头对少数几人实施询问。其谈话的话题可以相对自由一些，气氛可以宽松一些；当然对询问的对象，审计人员要认真研究和选择，小范围询问适用于两三位被询问者之间无利害冲突，对某些问题有共同的认识，容易达成一致，或者共同参与和介入某些业务。个别询问指审计人员对被查单位内外某一个人单独进行询问。个别询问给人的感觉有些不自在，被询问人总担心说错了什么，心理压力和防备是明显的，这种压力既来自审计人员，也来自被审单位的环境。

每种方式各有利弊，应当根据问询内容的重要性和敏感性，以及被询问人员的职级等因素慎重选择询问方式。如果选择不当，不仅影响查询的效果，有时会使询问陷入僵局。而且调查询问的方式也可以因工作需要而改变，同一被询问者不宜出现在集体询问，又出现于小范围询问和个别询问；多次询问后调查的效果会呈现出递减之势。

问卷法是指审计人员根据审计的目标和要求，根据事先设计的问题表格请被审单位有关人员进行回答来对某些审计事项予以证实以获得审计线索和证据的一种方法。问卷法在信息系统审计中是最重要和最有效地获取审计线索和证据的方法之一，本书提供了许多问卷调查表供读者在今后的信息系统审计工作实践中参考。

“问卷”译自法文 questionnaire 一词，其原意是“一种为统计或调查用的问题单”。问卷设计的好坏将直接影响到审计的质量和工作效率。一般而言，设计问卷有 6 种形式。

(1) 自由叙述式：不给被调查者提供任何答案让其按自己的思想用文字自由地回答。

(2) 多重选择式：让被调查者从提供的互不矛盾的答案中选择出一个或几个答案来。

(3) 判断式：让被调查者以“是”或“否”二择一的方法回答提供的答案。

(4) 评定量表法：让被调查者按规定的一个标准尺度对提供的答案进行评价。

(5) 确定顺序式：让被调查者对提供的几种答案按一定的标准(好恶或赞同与否等)作

出顺序排列。

(6) 对偶比较式：把调查项目组成两个一组让被调查者按一定的标准进行比较。

这六种问卷类型各有其优点和缺点，审计人员要根据审计目标和任务等，综合运用这几种形式，精心设计调查问卷。因此，设计调查问卷必须具备扎实的信息系统审计理论和丰富的信息系统审计实践经验。需要注意的是，调查问卷也可以包括审计人员已掌握的线索，通过回答情况以判断问卷的质量和采信的程度，审计人员可以据此考虑是否修改审计程序或实施追加的审计程序。

3. 函证法

函证法是指审计人员向有关单位或个人发函以证明某一审计事项的一种方法。它通过直接来自第三方对有关信息和现存状况的声明，以获取和评价审计证据。如：为了证实信息系统提供的应收款项和应付款项的准确性，以及电子交易本身的真实性等，常常运用这种方法进行核实。

函证法又分为肯定式函证和否定式函证。所谓肯定式函证就是向有关单位或个人发出询证函，要求其证实所函证的审计事项是否正确，无论对错都要求复函。而否定式函证是向有关单位或个人发出询证函，如果所函证的审计事项相符时不必复函，只有不符时才要求复函。

这两种函证方式各有优缺点，肯定式函证所获取的审计证据较为可靠，但审计成本较高；否定式函证因不可知因素的存在，所获取的审计证据相较而言不可靠，但成本相对较低。必须指出的是，对于重要的应收账项和电子交易不应以审计成本的高低作为减少审计程序的理由。具体采用哪种方式应根据不同情况做出选择，如对电子交易的真实性存在争议，或者数额较大时采用肯定式函证。当符合以下所有条件时，可采用否定式函证：相关的内部控制是有效的；预计差错率较低；欠款余额小的债务人数量很多；有理由确信大多数被函证者能认真对待询证函，并对不正确的情况予以反馈。这里应注意，上述函证方式的选择并不是绝对的，有时候两种方式结合起来使用，将其优缺点互补，可能会更适宜。

询函的设计要突出审计人员提出的问题，且易于理解，便于回答，不至于导致误解和异议。询函的形式可以有多种，常见的是调查表格，但也可以是图表式和文字式。回答问题的方式可以是说明式、是否式、填充式、打钩式等，应当留出空间给被调查人发表自己意见。调查的内容应包括以下几个方面。

(1) 审计部门或单位的名称，即说明函询者主体的身份；如是审计机关，应写“××审计机关函件”字样；如是会计师事务所，应写明“××会计师事务所函件”字样。

(2) 被函询单位和个人名称，单位应写明法定全称，个人应写明其所在单位或通讯地址。

(3) 函询的目的，要求证实的内容和提供资料数据，此项内容应说明清楚，指向明确，写清应证实的业务名称、内容、经手人、涉及的人与事、发生的时间、数量规模、金额凭证及其号码等，并留有被函询人回答问题的空当。

(4) 函询的具体要求，包括回答的方式、内容、格式和复函的时间、回答人的签名、单位的盖章等。

(5) 其他礼貌之语。

(6) 发函单位的名称(盖章)、发函时间等。

询函的格式设计各审计单位有所不同，没有统一的要求。

4. 查阅法

查阅法是指审计人员通过查阅被审单位的有关资料和技术文档等获得审计线索和证据的一种方法。查阅法要求对被审单位的业务资料、财务资料和信息系统技术资料等从形式到内容进行认真阅读，阅读中不能断章取义、片面理解，要在全面分析、客观公正的基础上，寻求相关的审计证据。对审计发现的问题，在做好笔录的同时，必须获得相关材料的影印件，这样收集的证据才是有价值的证据。

查阅软件文档是了解被审计组织的信息系统的最重要手段之一。软件文档(document)是用来记录、描述、展示软件项目开发过程中一系列信息的处理过程，通过书面或图示的形式对软件项目整体活动过程或结果进行描述、定义、规定、报告及认证。它描述和规定了软件项目开发的每一个细节，使用软件的操作命令，及软件产品投产以后，对产品使用过程中意见及产品缺陷、质量等方面的说明。它和计算机程序共同构成了能完成特定功能的计算机软件(有人把源程序也当做文档的一部分)。软件文档的编制，可以用自然语言、特别设计的形式语言、介于两者之间的半形式语言(结构化语言)、各类图形和表格。文档可以书写，也可以在计算机支持系统中产生，但它必须是可阅读的。按照文档产生和使用的范围，软件文档大致可分为三类。

(1) 开发文档：这类文档是在软件开发过程中，作为软件开发人员前一阶段工作成果的体现和后一阶段工作依据的文档。包括软件需求说明书、数据要求说明书、概要设计说明书、详细设计说明书、可行性研究报告、项目开发计划。

(2) 管理文档：这类文档是在软件开发过程中，由软件开发人员制定的须提交给管理人员的一些工作计划或工作报告，使管理人员能够通过这些文档了解软件开发项目安排、进度、资源使用和成果等。包括项目开发计划、测试计划、测试报告、开发进度月报及项目开发总结。

(3) 用户文档：这类文档是软件开发人员为用户准备的有关该软件使用、操作、维护的资料。包括用户手册、操作手册、维护修改建议、软件需求说明书。

根据查阅目的不同，查阅法又可以细分为审阅法、核对法、分析法和比较法等几种形式。

审阅法是对被审计组织的信息系统的文档资料以及被审单位的会计资料和其他资料进行详细的阅读和审查的一种审计方法。审阅法侧重于包括软件文档在内的书面资料的真实性、合法性。审阅法是最基本、最重要的方法。

核对法是指核对被审计组织的信息系统处理流程与相关软件文档内容等的一致性、系统处理流程与业务处理流程的一致性、系统操作执行与内部控制规定的一致性等，以获取审计证据的方法。可以采用软件测试手段核对信息系统实际处理流程与技术文档之间的一致性。

分析法是对被审计组织的信息系统流程的分析，目前业务流程的分析，采用数据流图、控制流程图等技术进行分析，通过分析了解被审单位的管理情况、内部控制情况、信息系统运行情况等。

比较法通过信息系统输入与输出的比较、信息系统应用与实际结果的比较，以及与同行业其他企业信息系统投入与功能等方面的比较，以证实某个审计事项的真实性和可靠性，获得审计证据的一种方法。在实际工作中，应当灵活运用这些方法，才能收到比较好的效果。

5. 复核法

复核法是指审计人员对被审计组织的信息系统的输出结果进行一次重复性的验算，以证明信息系统提供的输出结果是否正确的一种方法。复核法的具体内容包括：审计人员通过财务软件系统把被审单位提供的财务数据重新生成财务报表，以证明被审单位是否存在两套账或财务报表生成方式是否存在错弊等；通过软件黑盒法，在财务软件系统输入一个数字，根据在财务总账、明细账、报表中出现的位置来验算被审单位的财务归结方面是否存在错弊问题。这里讨论的复核法虽然与传统审计中的名称一样，但包含的内容完全不同。

3.2 数字取证方法

3.2.1 数字取证的概念

数字取证(digital forensics)是指为了揭示与数字产品相关的犯罪行为，利用一切科学、合法、正确的计算机技术与工具，对计算机系统中的数据进行检查、识别、收集、分析、提取、保存的活动。计算机系统既是取证的工具也是取证的对象。

基于单机的数字取证是针对一台可能含有证据的非在线计算机进行证据获取的技术，包括存储设备的恢复技术、解密技术、隐藏数据的显现技术、磁盘映像拷贝技术和信息搜索与过滤技术等。

基于网络的数字取证是在网上跟踪犯罪分子或通过网络通信的数据信息资料获取证据的技术，包括 IP 地址获取技术、针对电子邮件和新闻组的取证技术、网络入侵追踪技术、数据挖掘技术、网络数据包截获技术等。

在实际应用中往往将基于单机和设备的数字取证技术与基于网络的数字取证技术结合使用，以利于充足可靠的数字证据的发掘和收集。

3.2.2 数字取证的作用

《审计准则第 1301 号：审计证据》中明确将电子介质的记录形式列为审计证据，同时认为它比口头形式的审计证据更可靠。其中的第三十四条指出："某些会计数据和其他信息只能以电子形式存在，或只能在某一时点或某一期间得到，审计人员应当考虑这些特点对审计程序的性质和时间的影响。当信息以电子形式存在时，审计人员可以通过使用计算机审计技术实施某些审计程序。"

2010 年颁布的《审计准则》第七十六条指出："审计人员在检查被审计单位相关信息系统时，可以利用被审计单位信息系统的现有功能或者采用其他计算机技术和工具，检查中应当避免对被审计单位相关信息系统及其电子数据造成不良影响。"第八十七条提出："采集被审计单位电子数据作为审计证据的，审计人员应当记录电子数据的采集和处理过程。"此外，2012 年发布的《信息系统审计指南——计算机审计实务公告第 34 号》第十五条要求审计人员"在对电子数据的真实性产生疑问时，有权要求被审计单位按照审计机关提供的方案实施信息系统的系统测试和数据测试；对被审计单位信息系统不符合法律、法规和政府有关主管部门有关规定的，有权责令限期整改；对故意开发或者使用舞弊功能的单位和个人，有权依法追究其责任。"

由此可见，由于被审计组织的高度信息化，审计人员在收集审计证据时数字取证方法已

经成为一种必不可少的证据收集方法。

3.2.3 数字取证的方法

数字取证的方法分为六大类。

1. 识别类方法

识别类方法是用于判定可能与断言或与突发事件时间相关的项目、成分和数据的一种数字取证方法。识别类方法中使用到的典型技术有事件检测、签名处理、配置检测、误用检测、系统监视以及审计分析技术等。

进行证据识别的数据主要来源于计算机主机系统、计算机外部设备和网络方面。从计算机主机系统可以获取的数据信息包括系统类型、主机配置、主机运行环境变量、系统存在的漏洞信息、硬盘中存储的其他数据信息；计算机的外部设备如打印机、复印机、扫描仪等设备中很可能存在案件的关键线索和重要信息，因此对外部设备的取证也是取证不容忽视的环节；通过网络或者其他接口对犯罪主机或被侵害主机进行勘查取证或者在网络中对涉案数据进行截获，可以获取关于操作系统类型、网络的拓扑结构、端口服务、用户的地理位置和身份、攻击行为、服务器日志记录等数据信息。证据识别中可能用到的具体技术包括数据复制技术、数据恢(修)复技术、数据解密技术、端口及漏洞扫描技术、对比搜索技术、数据挖掘技术、日志分析技术等。

2. 保全类方法

保全类方法是用于保证证据状态的完整性的一种数字取证方法。保全类方法中使用到的典型技术有镜像技术、证据链监督技术、时间同步技术等。

证据保全是指采取有效措施保护电子证据的完整性、原始性及真实性。具体手段有：运用镜像拷贝或不可擦写光盘对数据进行备份；对于服务器上记载的电子证据可以采用加密、数字签名、物理隔离、建立安全监控系统监控的方法进行保全；采用关联数据保全的方法对可能包含有涉案证据但无法与涉案数据分离的数据进行整体备份和保存。证据保全可以划分为三个阶段：证据分析前的保全、证据分析过程中的保全和分析后对证据的保全。证据分析前的保全是分析前的必要工作，目的在于避免分析过程中因操作失误造成数据的丢失；证据分析过程中的保全是一个动态的过程，在分析的过程中适时进行保全可以为解释操作行为的合理性提供有力保障，为法庭质证提供审计依据；证据分析后的保全比较简单，这里不再介绍。证据保全中可能使用到的具体技术包括数据复制技术、数据加密技术、数据隐藏技术、数字摘要技术、数字签名和数字时间戳技术、数字审计技术等。

3. 收集类方法

收集类方法是用于提取或捕获突发事件的项及其属性的一种数字取证方法。该类方法与调查人员为在数字环境下获取证据而使用的特殊手段和产品相关。收集类方法中使用到的典型技术有复制软件、无损压缩以及数据恢复技术等。

证据收集的具体任务包括：原始数据的备份或打印；系统软硬件配置信息、日志记录和其他存储在计算机系统或网络中的原始数据信息以及其他显在数据信息的收集；隐藏数据的显现、被删除数据的恢复、毁坏数据的修复；时间、日期信息及具体操作步骤的详细记录等。该过程中可能用到的技术包括数据复制技术、扫描技术、数据恢(修)复技术、数据截取技术、“陷阱”取证技术等。

4. 检查类方法

检查类方法是用于对突发事件的项及其属性或特征进行仔细的检查的一种数字取证方法。检查类方法涉及从收集来的数据中进行检查并识别和提取潜在的证据。检查类方法中典型的技术有追踪、过滤技术、模式匹配、隐藏数据发现以及隐藏数据提取等。

证据检查是以证据收集为基础的，它是对收集来的数据进行检查并从中识别和提取可以作为证据的数据的过程。另外，在取证过程结束时，审计人员通过回顾检查的方式检查取证过程可能存在的漏洞时往往涉及证据检查技术的运用。证据检查中可能用到的具体技术有数据挖掘技术、对比搜索技术、扫描技术、数据解密技术等。

5. 分析类方法

分析类方法是用于为了获取结论而对数字证据进行融合、关联和同化的一种数字取证方法。分析类方法中典型的技术有追踪技术、统计分析技术、协议分析技术、数据挖掘技术、时间链分析技术等。

利用证据收集技术所获取的涉案数据是最原始的形式，为揭示这些数据与案件的联系就必须对这些数据进行检查和分析，证明这些数据就是攻击或者犯罪的证据，从而为证明案件真相提供证据支持。证据分析类技术包含检查类技术和分析技术。用于证据分析的技术包括对比分析技术、日志分析技术、数据挖掘技术、数据解密技术、攻击源追踪技术等。

6. 呈堂类技术

呈堂类技术是用于客观、清晰、准确地报告舞弊事项的一种数字取证方法。证据呈堂过程中可能用到的主要技术有证据链监督技术、数字摘要技术、数字签名技术、数字时间戳技术等。

数据呈堂的主要任务如下：

① 通过计算机作案的日期和时间、计算机运行环境变量、操作系统版本、计算机硬盘的状况以及其他相关情况记录的归档处理。

② 从取证工作的准备阶段到证据呈堂整个过程中证据的完整性情况证明。

③ 病毒评估分析报告、文件种类、取证工具许可证书、专家对电子证据的分析结果的归档处理和呈交。

④ 其他需要说明和解释事项的处理等。

3.2.4 数字取证的工具

数字取证工具即数字取证中所使用的软件和硬件工具。数字取证工具是由取证过程中所使用的软件、操作系统中已存在的一些命令工具、专门开发的工具软件、取证工具包以及某些工具性的硬件设施所组成的。它能够通过网络嗅探、网络追踪与定位、审计线索的搜索与分析，对审计证据进行保全、恢复和分析，最终生成审计报告。数字取证工具为数字取证带来了专业化和智能化的取证手段，方便了审计证据的搜集、获取、保全乃至数字取证的整个环节。因此，取证工作的成功与否在很大程度上取决于审计人员所使用的取证工具。

1. 非专用取证工具

非专用取证工具包括证据识别类的密码破译工具、数据恢复工具、文件浏览工具和网络监控工具等；证据保全类的磁盘镜像工具、反删除工具、加密工具、磁盘擦除工具等；证据收集类的磁盘镜像工具、数据截取、数据欺骗工具等；证据检查类的图片检查工具、文本搜索工具、磁盘分区检测工具等；证据分析类的磁盘分析工具、数据挖掘工具、日志分析工具、

对比搜索工具、密码破译工具等；证据呈堂类的归档工具等。这些软件虽然不是专门为取证开发的工具，但是完全可以为数字取证所用。

2. 专用取证工具

Guidance Software 公司开发的 Encase 是目前使用最广的一款专用的多功能取证软件工具，被美国知名信息安全杂志《SC Magazine》评鉴为五颗星，在法庭上具有相当高的接受度，已累积不少个案，美国 FBI 取证实验室采用 Encase 作为其作数字取证的工具。

Encase 是一款基于图形界面的取证应用程序，支持在不同类型操作系统、文件系统、储存媒体中，甚至是在运行的系统中进行取证。Encase 能够在 Windows、Macintosh、Linux、UNIX 等一些主流平台上运行，主要功能有数据浏览、搜索、磁盘浏览、数据预览、建立案例、创立证据文件、保存案例等。

AccessData 公司的 Forensic Tool Kit(FTK)是 UTK 中最主要的工具，已经逐渐为执法人员与民营企业所接受，其处理电子邮件证据功能相当优秀，尤其是电子邮件已经逐渐成为计算机犯罪判决的关键证物，FTK 可以迅速地从磁盘影像中过滤出所有的电子邮件，撷取出邮件中所包含图像文件以及可供检索的信息，这是取证工具中首屈一指的分析功能。同时，FTK 也可以产生一份图文并茂的完整取证报告。

e-fense 公司主要从事数字取证、计算机安全以及相关的培训，其产品 Helix 是基于 Knoppix 软件对数字取证和事件回报功能整合的取证工具。e-fense 公司在网络上提供了开放源码的 Live-Linux 光盘套件，并定期更新维护。Helix 软件支持 Windows 以及 UNIX 系统的相关取证，通过 Live-Linux 光盘可以直接启动计算机并执行，可针对运行中或已关机的系统进行分析。由于该系统是基于 Linux 平台开发的，故支持 Ext2/Ext3 的文件系统，以及支持一些比较少见的 ReiserFS、JFS 与 XFS 文件系统。Helix 最大的特点在于：通过它的 Live-Linux 光盘开机，Helix 不会挂载任何硬盘上面的分割区，而且是以只读模式开启硬盘上的信息，Helix 不会涉及 swap space，也就是说可以确保不会改变证据现场。

计算机取证勘查箱被认为是适应范围广、复制功能强、携带方便、使用灵活的移动取证平台。硬盘拷贝机能够在不借助计算机和操作系统的情况下 100%地复制硬盘中的数据，这是任何拷贝软件所无法做到的，并且硬盘拷贝机通过计算校验和的方式确保数据复制的准确性。

3.2.5 数字取证的规范

1. 证据识别规范

证据识别是对被取证计算机中的数据及与之相关设备上的数据进行认知和判断，从计算机所储存的数据中找出与审计事项相关的数据证据。因此证据识别规范应包括人员配置规范、现场保护规范、全面检查规范、记录规范，以及工具选用和操作方法规范等。

2. 证据保全规范

在证据保全时应要求取证人员做到：合理选用和操作保全工具；制作并保管备份；确保证据保存环境的绝对安全；严格监督管理等。对于证据的移交、保管、开封、拆卸等要详细记录，保障证据的完整性和真实性。

3. 数字证据的收集、检查和分析规范

数字证据的收集、检查和分析这三个过程紧密联系，在检查的同时进行证据的分析和收

集，在收集和分析的基础上进行检查，以查缺补漏。

4. 数字证据呈堂规范

数字证据呈堂规范也称证据的移交规范，是数字取证的最后一个环节，负责将获取的数字证据移交有关部门。要采取相应的措施确保数字证据和取证结论内容从取证结束到法庭证据公示前的完整性。

3.3 数据库查询方法

3.3.1 数据库查询工具

SQL(structured query language，结构化查询语言)属于高级的非过程化编程语言，是沟通数据库服务器和客户端的重要工具，允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以，具有完全不同底层结构的不同数据库系统，可以使用相同的 SQL 作为数据输入与管理的接口。

SQL 是 IBM 的圣约瑟研究实验室为其关系数据库管理系统 SYSTEM R 开发的一种查询语言，它是以 Codd 的理论为基础开发的。SQL 结构简洁，功能强大，简单易学，所以自从 IBM 公司 1981 年推出以来，SQL 得到了广泛的应用。如今无论是像 Oracle、Sybase、DB2、Informix、SQL Server 这些大型的数据库管理系统，还是像 Visual Foxpro、PowerBuilder 这些 PC 上常用的数据库开发系统，都支持将 SQL 作为查询语言。

1992 年，ISO(国际标准化组织)和 IEC(国际电子委员会)发布了 SQL 国际标准，称为 SQL-92。ANSI(美国国家标准局)随之发布的相应标准是 ANSI SQL-92，有时称为 ANSI SQL。尽管不同的关系数据库使用的 SQL 版本有一些差异，但大多数都遵循 ANSI SQL 标准。SQL Server 使用 ANSI SQL-92 的扩展集，称为 T-SQL，其遵循 ANSI 制定的 SQL-92 标准。SQL 语言包含 4 个部分：

(1) 数据定义语言(DDL)，如 CREATE、DROP、ALTER 等语句。

(2) 数据操作语言(DML)，如 INSERT(插入)、UPDATE(修改)、DELETE(删除)语句。

(3) 数据查询语言(DQL)，如 SELECT 语句。

(4) 数据控制语言(DCL)，如 GRANT、REVOKE、COMMIT、ROLLBACK 等语句。

SQL 语句可以嵌入宿主语言(如 C、Java 语言等)中使用，也可在终端上以联机交互方式使用，在信息系统审计时，审计人员利用 SQL 工具查询信息系统中的数据。

SQL 提供了 SELECT 语句进行数据库查询。SQL 的数据查询语句的语法格式为：

```
SELECT [DISTINCT]<属性表> FROM R1[<别名>], …, Rn[<别名>]
WHERE condition
[GROUP BY<分组属性表>[HAVING<分组选择条件>]]
[ORDER BY<列名>[< ORDER >], …,<列名>[< ORDER >]]
```

其中，属性表是作为查询结果的新关系所包含的属性，可选项 DISTINCT 表示查询结果中不出现重复元组。$R_1 \cdots R_n$ 是 n 个不同的关系，“别名”是各个关系的简化名称，condition 是查询条件表达式，由逻辑运算符、算术比较符、常值和属性名等构成。可选从句 GROUP BY…HAVING 表示依照＜分组属性表＞中的属性值和＜分组选择条件＞把结果关系分组，ORDER BY 从句实现查询结果排序。

整个 SELECT 语句操作过程如下：将 FROM 子句所指出的关系进行连接，从中选取满足 WHERE 子句中条件 condition 的行(元组)，然后按 GROUP 子句给定列的值进行分组，并按 ORDER 子句的要求排序，最后根据 SELECT 子句给出的列名或列表达式将查询结果表输出。

下面介绍如何采用 SQL 查询工具查询、统计、分析信息系统中的数据。

3.3.2 对单个表的查询

SQL 语句的语法是“SELECT * FROM 表名 WHERE 条件”。用 WHERE 指定查询条件，可以指定单个条件或多个条件，并可配合函数或关键字使用，也可以不用 WHERE，只用函数查询。是否能准确找到所需的数据，条件设置是关键，同时审计人员也要了解被审计单位的业务和信息系统流程等。表 3-1 罗列了一些常用的查询条件。

表 3-1 常用的查询条件

查 询 条 件	谓 词
比较	=,>,<,<=,>=,<>,! >,! <,! =
范围	BETWEEN AND,NOT BETWEEN AND
集合	IN,NOT IN
匹配	LIKE,NOT LIKE
空值	IS NULL,IS NOT NULL
逻辑	AND,OR,NOT

(1) 单个条件的查询，设定的条件放在 WHERE 后面。如在存款表中查询所有存款金额 10 万元以上(含 10 万元)的记录：

```
SELECT * FROM 存款表
        WHERE 金额>=100000;
```

这是最基本的查询语句。

(2) 多个条件的查询，各条件间可能是 and 或 or 的关系。如在存款表中查询存款金额 10 万元以上(含 10 万元)，并且存款时间 2 年以上(含 2 年)的全部记录：

```
SELECT * FROM 存款表
        WHERE 金额>=100000 and 存款年限>=2;
```

对多个条件的查询，根据关键字 and 或 or 来决定各种条件之间的关系。

(3) 对关键字的内容进行查询。如把所有摘要中含有“招待费”的凭证找出来：

```
SELECT * FROM 凭证表
        WHERE 摘要 like '%招待费%';
```

查询包含某些关键字的条件也可以是多个，采用关键字 and 或 or 进行组合。

(4) 查询内容包含在一定的范围内。如在科目代码表中查询科目代码分别是 101、102 的记录：

```
SELECT * FROM 科目代码表
        WHERE 科目代码 IN ('101','102');
```

常用在已基本确定查询范围之后。当所关心内容类别较多时可使用下面将提到的两张表的嵌套查询。

(5) 不用写条件,用聚集函数进行查询。如要查找最大的一笔存款额:

```
SELECT max(发生金额) FROM 存款表;
```

函数在审计实际工作当中非常有用,我们在下一部分详细介绍。

3.3.3 对单个表的统计

审计人员还可以利用查询语句提供的强大功能进行分组、排序、计算等统计工作,综合运用聚集函数、GROUP BY、ORDER BY 等可以进行各种统计分析,形成一系列的审计中间表。

```
SELECT count( * ) FROM 凭证表
        WHERE 金额>=100000;
SELECT avg(发生金额) FROM 存款表;
```

一些常见的聚集函数如表 3-2 所示。

表 3-2 常见的聚集函数

函数形式	功能说明
COUNT(*)	统计元组的个数
COUNT(列名)	统计某一列中字段的个数
AVG(列名)	计算某一列中值的平均值
SUM(列名)	计算某一列中值的和
MAX(列名)	求出某一列中的最大值
MIN(列名)	求出某一列中的最小值

(1) 对多个条件的查询,得到符合条件的某字段的合计值:

```
SELECT sum(金额) FROM 存款表
    WHERE 金额>=100000 and 存款时间 >= #2011-1-1#
        and 存款时间<= #2011-12-31#
    ORDER BY 金额 DESC,存款时间 ASC;
```

该情况一般用于统计某种条件下的合计值,常用于对某种情况下的总金额情况进行了解。

(2) 对某个字段进行分组,查询各字段属性分别对应的发生次数,并按发生次数进行排序:

```
SELECT 款项代码,count(款项代码) AS 次数 FROM 大额现金支取表
      GROUP BY 款项代码
  ORDER BY count(款项代码)DESC;
```

该情况常用于对某个所关心字段的分类统计,得出每个类别所发生的次数,并可按发生次数进行排序。

(3) 按某个字段的长度进行分类并进行排序:

```
SELECT len(科目代码) AS 发生次数 FROM 科目代码表
```

```
GROUP BY len(科目代码)
ORDER BY len(科目代码) DESC;
```

该情况用于了解某字段的长度分组情况，可对所取得的原始数据进行验证，以证实原始数据的规范性或找出不规范情况的问题所在。

(4) 分组查询，GROUP BY 与 HAVING 联用：

```
SELECT 客户号,count( * ) AS 发生次数 FROM 存款表
      WHERE 发生金额 >= 100000
      GROUP BY 客户号
      HAVING count( * )>1;
```

常用于要列出所关心某字段的某一个或几个条件限制下发生次数超出某范围的情况。如查询发生多次(即同一关键字发生的记录数大于 1)大额存取的客户号及其存取次数，查询结果按客户号分组。

GROUP BY 子句将表按列的值分组，列值相同的分在一组，GROUP BY 后可以有多个列名，先按第一列名分组，按第二列名在已分得的组中再分组，直到 GROUP BY 子句所指名的列都是具有相同的值基本组。使用 GROUP BY 子句后，SELECT 子句所取的值必须在基本组中是唯一的，即只能是 GROUP BY 子句所指名的列或聚集函数。上例中，按学号对学生选课情况进行分组后，SELECT 子句所取的值 SNAME 和 SUM(GRADE)在每个基本组中都是唯一的。

3.3.4 生成审计中间表

以上介绍了如何对单个表进行查询和统计，但是这些结果并未直接存储成新表，不利于对查询结果进一步分析利用，审计人员常常使用“SELECT * INTO 表名(审计中间表) FROM 表名 WHERE 条件”这种语句格式把查询结果保存在审计中间表中(一张新表)。这些审计中间表非常有用，其作用类似审计工作底稿，审计人员可以直接对这些中间表做进一步统计分析等。

(1) 例如审计人员可以把查出的金额超过 10 万元的存款记录放在一张审计中间表中，这张审计中间表称为“大额存款发生情况表”。具体语句如下：

```
SELECT * INTO 大额存款发生情况表
        FROM 存款表
        WHERE 金额 >= 100000;
```

(2) 两张结构相同的表合并生成一张新表，使用关键字 UNION 进行联合查询：

```
SELECT * INTO 审计中间表 FROM
(SELECT * FROM 原始表 1 UNION SELECT * FROM 原始表 2);
```

该情况常用于对多个原始表的整理以生成中间表，便于后续的查询工作。如合并采集到的原始数据，将原本分月存储的数据表合并成分年度存储的数据表。

(3) 连接两张表，生成新的中间表：

```
SELECT a. * ,b.客户名称 INTO 含客户名称的现金支取表
    FROM 现金支取表 AS a,客户基本信息表 AS b
```

```
WHERE a.款项代码 = b.款项代码;
```

使用该类语句的前提是两张表存在相关联的关键字，常用于一张表根据两表的连接关键字引入另一张表的某些字段，为后续的查询准备好中间表。此处使用别名可以简化 SQL 语句的输入，如财务数据中采集到的凭证表是两张表时，可连接两表引入摘要字段等。

3.3.5 对多个表的查询

若一个查询同时涉及两个以上的表，则称之为连接查询，包括等值连接查询、非等值连接查询、简单连接查询和复合连接查询等。

这种情况下必须对各表的对应关键字进行关联，一般情况下是对两张表的关联查询，当有更多张表要操作时可以每次对两张表操作后再和剩余的表关联操作，也可以扩展 SQL 语句同时对两个以上的表操作，以下只列出对两张表操作的情况。

（1）两张表的关联查询：

```
SELECT a.*,b.科目代码 FROM 对公活期存款明细表 AS a,活期存款账户动态表 AS b
    WHERE a.款项代码 = b.款项代码 and 科目代码 LIKE '201%';
```

该情况可看做对两张表连接的扩展，即根据 WHERE 所限定的条件来对两张表关联查询。

（2）两张表的嵌套查询，两层查询间要使用关键字 IN：

```
SELECT * INTO 频繁发生大额现金支取表 FROM 大额现金支取表
        WHERE 交易金额>=100000 and 款项代码 IN
        (
        SELECT 款项代码 FROM 款项代码发生次数_大额现金支取
                WHERE 次数 >=10
    )
  ORDER BY 款项代码,交易金额,记账日期;
```

这种情况也可理解为简单查询，不同的是其中 WHERE 所指定的某个条件是由另一张表的又一个查询所指定的，与单表查询的语句

```
SELECT * FROM 科目代码表 WHERE 科目代码 in ('101','102');
```

进行对比后发现，可将两张表的嵌套查询看做如上语句的扩展。如利用大额现金支取表和大额现金支取的款项代码发生次数表查询交易金额大且发生次数多的情况。再比如利用贷款表和股东表查询贷款表中含股东贷款的情况。

嵌套查询使我们可以用多个简单查询构成复杂查询，从而扩充 SQL 的查询能力。以多层嵌套的方式来构造程序，正是 SQL 中“结构化”的含义所在。

3.3.6 应用实例

1. 房产税征缴审计目标与流程

房产税征缴审计目标是检查地方税务机关房产税收入的真实性、完整性、合法性，以及征管控制的有效性。

通过关联纳税人基本信息表和税款征收数据表，分析纳税人房产税缴纳情况，以及比对

纳税人近几年缴纳房产税是否异常，审查地方税务机关房产税收入是否真实完整、合法合规，是否对纳税人及时足额缴纳房产税实施有效监控。房产税征缴审计逻辑关系见图 3-1。

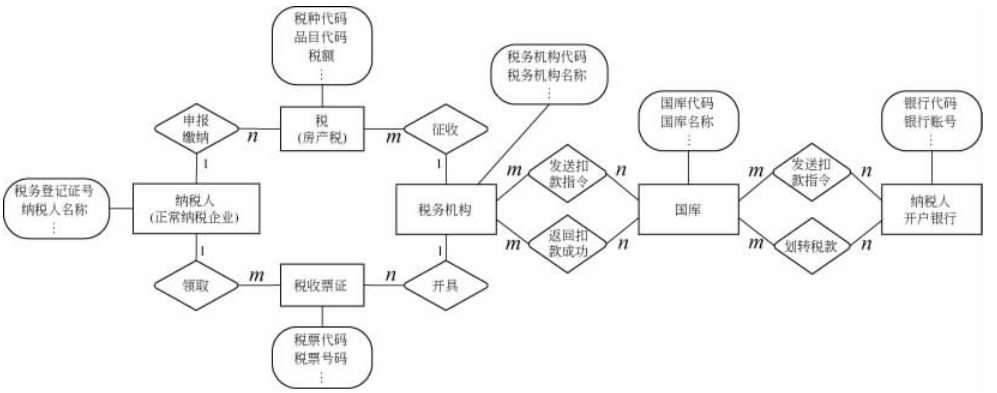

图 3-1　房产税征缴审计的 E-R 图

2. 建立缴纳房产税差额较大纳税人的查询模型

纳税金额较大的企业一般都有自己的固定房产，且缴纳的房产税较稳定，因此将正常缴纳税款但未缴纳房产税，或缴纳房产税较少的、近两年房产税纳税金额变化较大的重点税源户，确定为延伸审计对象；延伸时重点关注税务机关的征管措施，同时审查纳税人拥有的房产和变化情况以及相关涉税资料，检查是否及时、足额申报纳税。所以建立缴纳房产税差额较大纳税人的查询如下：

```
SELECT 税务登记证号,纳税人名称,sum(实纳税额)  AS 本年税款总额
    FROM 分析表_房产税税款征收数据表
    WHERE 税种代码 = '11' AND 冲负调整标志<>'9' AND 税款状态代码 = '08' AND year(税款状态改变日
期) = @审计年度
    GROUP BY 税务登记证号
JOIN
SELECT 税务登记证号,纳税人名称,sum(实纳税额)  AS 上年税款总额
    FROM 分析表_房产税税款征收数据表
    WHERE 税种代码 = '11' AND 冲负调整标志<>'9' AND 税款状态代码 = '08' AND year(税款状态改变日
期) = (@审计年度 - 1)
    GROUP BY 税务登记证号
```

3.4　软件测试方法

3.4.1　概述

软件测试按方法理论划分，有白盒测试和黑盒测试；按是否运行程序划分，有静态测试和动态测试；按阶段划分，有单元测试、集成测试、系统测试、验收测试、α 测试（内测）、β 测试（公测）、回归测试等；按测试手段划分，有手动测试和自动测试。

软件测试是信息系统审计的重要方法之一，本节主要介绍两种经典的软件测试方法，即黑盒测试和白盒测试，以及两种新发展的软件测试方法，即基于故障的软件测试方法和基于模型的软件测试方法。

3.4.2 黑盒测试

黑盒测试(black box testing)也称功能测试或数据驱动测试。信息系统审计师在已知企业的信息系统功能的条件下,通过测试来检测每个功能是否都能正常使用。

在测试时,把企业信息系统看做一个不能打开的黑盒子,在完全不考虑程序内部结构和内部特性的情况下,利用测试用例来对系统进行测试,以检查程序是否按照需求规格说明书的规定正常使用,程序是否能适当地接收输入数据而产生正确的输出信息,并且保持外部信息(如数据库或文件)的完整性。

黑盒测试方法着眼于信息系统外部功能,不考虑内部逻辑结构,针对企业信息系统的界面和功能进行测试,信息系统审计师在应用黑盒法时,手头只需有程序的功能说明书就够了。

黑盒测试方法主要有等价类划分、边界值分析、因果图分析、错误推测法等方法。

1. 等价类划分

等价类划分法的基本思想为,如果将输入数据的可能值分成若干个"等价类",就可以合理地假定:每一类的一个代表性的值在测试中的作用等价于这一类中的其他值,也就是说,如果某一类中的一个测试用例发现了错误,这一等价类中的其他测试用例也能发现同样的错误;反之,如果某一类中的一个测试用例没有发现错误,则这一类中的其他测试用例也不会查出错误(除非等价类中的某些测试用例又属于另一等价类,因为几个等价类是可能相交的)。

设计等价类的测试用例一般分为两步进行:

(1) 划分等价类并给出定义。

(2) 选择测试用例。

选择的原则是:有效等价类的测试用例尽量公用,以期进一步减少测试的次数;无效等价类必须每类一例,以防漏掉本来可能发现的错误。

划分等价类时,需要研究程序的功能说明,以确定输入数据的有效等价类和无效等价类。在确定输入数据的等价类时常常还需要分析输出数据的等价类,以便根据输出数据的等价类导出对应的输入数据等价类。

2. 边界值分析

经验表明,处理边界情况时程序最容易发生错误。例如,许多程序错误出现在下标、纯量、数据结构和循环等的边界附近。因此,设计使程序运行在边界情况附近的测试方案,暴露出错误的可能性更大一些。

使用边界值分析方法设计测试用例首先应该确定边界情况,这需要经验和创造性,通常输入等价类和输出等价类的边界,就是应该着重测试的程序边界情况。选取的测试数据应该刚好等于、小于和大于边界值。也就是说,按照边界值分析法,应该选取刚好等于、稍小于和稍大于等价类边界值的数据作为测试数据,而不是选取每个等价类内的典型值作为测试数据。

3. 因果图分析

因果图分析是为了解决边界值分析和等价划分的一个弱点,即未对输入条件的组合进行分析。因果图用一个系统的方法选择出此类高效的测试用例集,并且可以指出规格说明的不完整性和不明确之处。

因果图是一种形式语言(有严格语法限制的语言),是将自然语言描述的规格说明转换为因果图。实质上,是一种数字逻辑电路(一个组合的逻辑网络),但没有使用标准的电子学符号,而是使用了稍微简单点的符号。借助因果图列出输入数据的各种组合与程序对应动作效果之间的阶段联系,构造判定表,由此设计测试用例是生成测试用例的有效办法。

4. 错误推测法

人们也可以通过经验或直觉推测程序中可能存在的各种错误,从而有针对性地编写检查这些错误的例子,这就是错误推测法。

错误推测法在很大程度上靠直觉和经验进行。它的基本想法是列举出程序中可能有的错误和容易发生错误的特殊情况,并且根据它们选择测试用例。对于程序中容易出错的情况也有一些经验总结出来,例如,输入数据为零或输出数据为零往往容易发生错误;如果输入或输出的数目允许变化(例如,被检索的或生成的表的项数),则输入或输出的数目为 0 和 1 的情况(例如,表为空或只有一项)是容易出错的情况。还应该仔细分析程序规格说明书,注意找出其中遗漏或省略的部分,以便设计相应的测试用例,检测程序员对这些部分的处理是否正确。

3.4.3 白盒测试

白盒测试(white box testing)也称结构测试或逻辑驱动测试,它是知道产品内部工作过程,可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行,按照程序内部的结构测试程序,检验程序中的每条通路是否都能按预定要求正确工作,而不顾它的功能。白盒测试的主要方法有逻辑驱动、基路测试等,主要用于软件验证。

白盒测试方法需要全面了解企业信息系统的内部流程。白盒法是穷举路径测试,测试者必须检查程序的内部结构,从检查程序的逻辑着手,得出测试数据。贯穿程序的独立路径数可能是天文数字,而且即使每条路径都测试了仍然可能有错误,因为穷举路径测试决不能查出程序违反了设计规范,即程序本身是个错误的程序;其次,穷举路径测试不可能查出程序中因遗漏路径而出错;最后穷举路径测试可能发现不了一些与数据相关的错误。对于信息系统审计师而言,测试时不需要穷举路径,只需要测试一些主要流程是否符合要求与规范。

白盒测试法的覆盖标准有逻辑覆盖、循环覆盖和基本路径测试。其中逻辑覆盖包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖。

1. 语句覆盖

为了暴露程序中的错误,至少每个语句应该执行一次。语句覆盖的含义是,选择足够多的测试数据,使被测试程序中的每个语句至少执行一次。

例如,图 3-2 是一个被测模块的流程图。它的源程序如下:

```
PROCEDURE  EXAMPLE (A,B:REAL; VAR X :REAL)
          BEGIN
              IF (A>1)AND (B=0)
                  THEN X := X/A
              IF (A=2) OR  (X>1)
                  THEN X := X+1
          END;
```

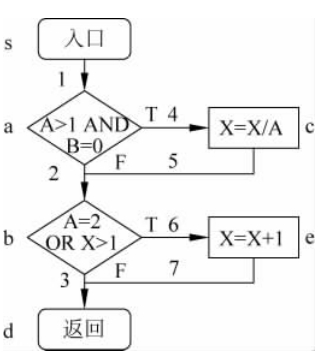

图 3-2 某段程序的流程图

为了使每个语句都执行一次，程序的执行路径应该是 sacbed，为此只需要输入下面的测试数据（实际上 X 可以是任意实数）：

A = 2,B = 0,X = 4

语句覆盖对程序的逻辑覆盖很少，在例子中两个判定条件都只测试了条件为真的情况，如果条件为假时处理有错误，显然不能发现。此外，语句覆盖只关心判定表达式的值，而没有分别测试判定表达式中每个条件取不同值时的情况。在上面的例子中，为了执行 sacbed 路径，以测试每个语句，只需两个判定表达式（A＞1）AND（B＝0）和（A＝2）OR（X＞1）都取真值，因此使用上述一组测试数据就够了。但是，如果程序中把第一个判定表达式中的逻辑运算符“AND”错写成“OR”，或把第二个判定式中的条件“X＞1”误写成“X＜1”，使用上面的测试数据并不能查出这些错误。

综上所述，可以看出语句覆盖是很弱的逻辑覆盖标准，为了更充分地测试程序，可以采用下述的逻辑覆盖标准。

2. 判定覆盖

判定覆盖的含义是，不仅每个语句必须至少执行一次，而且每个判定的可能的结果都应该至少执行一次，也就是每个判定的每个分支都至少执行一次。

对于上述例子来说，能够分别覆盖路径 sacbed 和 sabd 的两组测试数据，或者可以分别覆盖路径 sacbd 和 sabed 的两组测试数据，都满足判定覆盖标准。例如，用下面两组测试数据就可以做到判定覆盖：

① A＝3，B＝0，X＝3　（覆盖 sacbd）

② A＝2，B＝1，X＝1　（覆盖 sabed）

判定覆盖比语句覆盖强，但是对程序逻辑的覆盖程度仍然不高，例如，上面的测试数据只覆盖了程序全部路径的一半。

3. 条件覆盖

条件覆盖的含义是，不仅每个语句至少执行一次，而且是判定表达式中的每个条件都取到各种可能的结果。

图 3-2 的例子中共有两个判定表达式，每个表达式中有两个条件，为了做到条件覆盖，应该选取测试数据，使得在 a 点有下述各种结果出现：

A＞1，A≤1，B＝0，B≠0

在 b 点有下述各种结果出现：

A＝2，A≠2，X＞1，X≤1

只需要使用下面两组测试数据就可以达到上述覆盖标准：

（1）A＝2，B＝0，X＝4　（满足 A＞1，B＝0，A＝2 和 X＞1 的条件，执行路径 sacbed）

（2）A＝1，B＝1，X＝1　（满足 A≤1，B≠0，A≠2 和 X≤1 的条件，执行路径 sabd）

条件覆盖通常比判定覆盖强，因为它使判定表达式中每个条件都取到了两个不同的结果，判定覆盖却只关心整个判定表达式的值。例如，上面两组测试数据也同时满足判定覆盖标准。但是，也可能有相反的情况，虽然每个条件都取到了两个不同的结果，判定表达式却始终只取一个值。例如，如果使用下面两组测试数据，则只满足条件覆盖标准并不满足判定覆盖标准（第二个判定表达式的值总为真）：

(1) A=2,B=0,X=1 (满足 A>1,B=0,A=2 和 X≤1 的条件,执行路径 sacbed)

(2) A=1,B=1,X=2 (满足 A≤1,B≠0,A≠2 和 X>1 的条件,执行路径 sabed)

4. 判定/条件覆盖

既然判定覆盖不一定包含条件覆盖,条件覆盖也不一定包含判定覆盖,自然会提出一种能同时满足这两种覆盖标准的逻辑覆盖,这就是判定/条件覆盖,它的含义是,选取足够多的测试数据,使得判定表达式中的每个条件都取到各种可能的值,而且每个判定表达式也都取到各种可能的结果。

对于图 3-2 的例子而言,下述两组测试数据满足判定/条件覆盖标准:

(1) A=2,B=0,X=4

(2) A=1,B=2,X=1

但是,这两组测试数据也就是为了满足条件覆盖标准最初选取的两组数据,因此,有时判定/条件覆盖也并不比条件覆盖更强。

3.4.4 基于故障的测试

目前,市场上已有多个基于故障的软件测试系统。基于故障的测试的好处有如下几个方面。

(1) 针对性强。如果说某种故障是经常发生的,并且在被测软件中是存在的,则面向故障的测试可以检测出此类故障,不会像白盒测试和黑盒测试那样具有不确定性。

(2) 有些故障一次性测试是检测不出来的,这种故障用白盒测试和黑盒测试这两种方法是检测不出来的。例如,存储器泄露故障、空指针引用故障等。

(3) 可以避免其他测试方法对“小概率”检测效率比较低的情况。

故障检测的一般步骤是:在给定源代码的前提下,通过分析源代码,计算出和故障模型相匹配的程序代码,这个过程称为计算检查点(inspective points,IP)。当 IP 确定以后,接下来的工作是判断 IP 的性质。

IP 的性质分为 3 类:故障 IP、正确的 IP 和不能确定的 IP。

(1) 故障 IP:该 IP 经过计算肯定是存在一个故障。

(2) 正确的 IP:该 IP 经过计算肯定不是一个故障。

(3) 不能确定的 IP:该 IP 经过计算无法确定是故障或者是正确的。

1. 计算检查点

计算检查点的过程可以用图 3-3 来描述,它的输入是源代码,输出的是由 IP 组成的数据库,数据库的每一个记录代表一个 IP。检查点计算算法有两个性能指标:一是计算复杂性;二是计算的精度。算法分析表明,对任何一类故障,计算 IP 的过程是线性复杂性。计算精度是指算法是否能计算出所有的 IP,分析表明,在控制流图上进行 IP 计算,其结果比较准确。

2. 计算检查点的性质

计算检查点的性质可以自动判断也可以人工判断。对某些语法规则比较简单的 IP 可以采用自动计算的办法,而对某些复杂 IP,目前还必须经过人工计算。在现阶段,给出一个判断所有 IP 性质的算法是比较困难的,大约只有 30% 的 IP 可以自动判断,一般的仍需要人工判断。

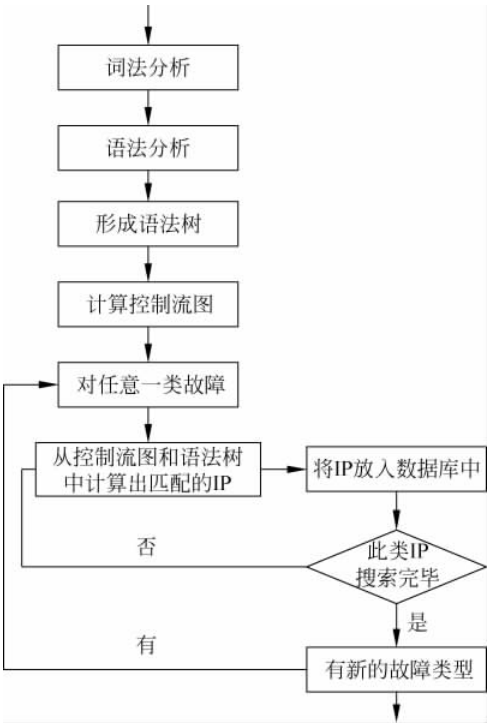

图 3-3 检查点的计算过程

面向故障的软件测试是今后软件测试发展的重要方向，因为只有对故障准确地认识才能实施有效的测试，所以面向故障测试今后所有解决的问题是高效率的 IP 计算方法和自动化的 IP 性质计算。

3.4.5 基于模型的测试

随着面向对象软件开发技术的广泛应用和软件测试自动化的要求，基于模型的软件测试(model-based software testing)逐渐得到重视。基于模型的测试最初应用于硬件测试，后来广泛应用于电信交换系统的测试，目前在软件测试中得到了一定的应用。

软件模型是对软件行为和软件结构的抽象描述。软件行为可以用系统输入序列、活动、条件、输出逻辑或者数据流进行描述，软件结构则使用组件图、部署图等进行描述。针对测试任务，通过对软件功能和结构进行抽象并用易于理解的方式进行描述，获得的模型就是对被测试软件系统精确的描述，可以用于软件测试。一般对软件不同行为要用不同模型进行描述。例如：控制流图、数据流图和程序依赖图表达了程序和代码结构间的行为关系，决策表和状态机则可以描述软件外部行为。

基于模型的软件测试可以根据软件行为模型和结构模型生成测试用例。当前随着软件规模的不断增长，使得基于程序的测试十分困难，而基于模型的软件测试则大大提高了测试的自动化水平，部分解决了测试失效的辨识问题，并可以应用成熟的理论和技术获得比较完善的分析结果。

基于模型的软件测试过程主要包括如下 5 个步骤。

1. 分析和理解被测试软件

基于模型的软件测试要求充分理解被测试软件。构造可以用于测试的模型的工作主要是根据测试目的确定测试对象和测试特征，针对被测试软件的相关属性建立相应模型。这个阶段的具体工作包括：

（1）充分了解软件需求规范和设计文档、用户手册，和开发队伍充分交流。

（2）识别软件系统的用户，枚举每个用户的输入序列，研究每项输入的可能取值范围，包括合法值、边界值、非法值以及预期输出。这项工作往往需要工具支持。

（3）记录输入发生条件和响应发生条件。软件系统的响应是指用户能够得到的输出或可见的软件内部状态的改变。其目的是设计可以引发特定响应的测试例和评价测试结果。

（4）研究输入序列。例如，输入发生时刻，软件系统接收特定输入的条件，输入处理顺序。

（5）理解软件内部数据交换和计算过程，产生可能发现缺陷的测试数据。

2. 选择合适的测试模型

不同的模型适用于不同类型软件的测试，因此需要根据软件特点选择模型。模型的选择标准如下：

（1）了解可用的模型。不同的应用领域要使用不同的测试模型。

（2）根据模型特征进行选择。只有充分理解模型和软件系统，才能选择合适的模型对软件进行测试。

（3）人员、组织和工具对模型选择的影响。基于模型的软件测试对测试人员的知识结构和技术水平提出了一定要求，另外开发组织所使用的测试工具也影响模型的选择。

3. 构造测试模型

下面以基于状态机模型的测试为例说明如何构造测试模型。首先要抽象出软件系统状态，状态抽象一般要根据输入及输出条件进行，一般包括以下过程。

（1）生成一个输入序列并说明每个输入的适用条件，称做输入约束。例如，电话未摘机时，才允许有摘机动作发生。

（2）对每个输入要说明产生不同响应的上下文环境，称做响应约束。例如，电话摘机时，如果当前状态为振铃，则进行通话，否则为拨号音。

（3）根据输入序列、输入约束和响应约束构造相应状态机模型。

4. 生成和执行测试用例

测试用例的自动生成依赖于测试所使用的模型。以状态机模型为例，被遍历路径中弧的标记构成的序列就是测试用例。在构造满足测试准则的路径时，必须考虑约束条件，如路径长度限制。生成了满足特定的测试充分性准则的测试用例集合后就可以执行测试用例。以状态机模型为例，首先要写出仿真该软件系统的每个不同外界激励的脚本，称为仿真脚本，每个仿真脚本对应模型中一个不同的迁移，然后把测试用例集合翻译为测试脚本。测试用例的执行就是测试脚本的执行过程。

5. 收集测试结果并进行分析

基于模型的软件测试方法并没有解决测试失效的辨识问题，仍然要人工检查输出是否正确。但是通过状态验证可以部分解决测试失效的辨识问题，状态是内部数据的抽象，比较容易验证。另外与传统测试相比，基于模型的软件测试的优势之一就是可以根据测试结果分析软件的其他质量因素，如可靠性。

案例 1　安然公司破产——信息系统审计的转折点

【资料】

安然公司从 1990 年到 2000 年的 10 年间，销售收入从 59 亿美元上升到了 1008 亿美元，净利润从 2.02 亿美元上升到 9.79 亿美元，成为与通用、埃克森、美孚、壳牌等百年老店平起平坐的新一代商业巨擘。1999 年 11 月，面对美国网络经济的繁荣，安然公司创建了第一个基于互联网的电子商务平台——“安然在线”，提供从电和天然气现货到复杂的衍生品等 1500 多种商品交易。不到一年时间，即发展成为年交易规模近 2000 亿美元的全球最大的电子商务交易平台。高效率的电子商务交易平台成就了安然的辉煌，使其成为最具创新精神的公司。同时也埋下了安然崩塌的隐患，高效率的电子商务交易平台也使安然可以“方便、高效、快捷”地实现在关联企业之间进行“对倒”，通过“对倒”创造交易量，创造利润，创造“经营神话”。公司经历了 4 大步跨越，从名不见经传的一家普通天然气经销商，逐步发展成为世界上最大的天然气采购商和出售商、世界最大的电力交易商、世界领先的能源批发做市商、世界最大的电子商务交易平台，一步一个高潮，步步走向辉煌。

2001 年 12 月，在全球拥有 3000 多家子公司，掌控着美国 20％的电能和天然气交易的安然公司突然申请破产保护。

【思考题】

1. 审计师在审计企业的收入时，常常要通过函证等方法证实或证伪交易的真实性，从而核实其收入的真实性，但是电子商务的出现使得传统的审计方法面临挑战，审计师应该怎么办？

2. 信息系统审计师如何与注册会计师合作共同评估电子交易的真实性呢？

3. 分析电子商务是如何影响企业的内部控制和财务数据产生的，特别是对百度、阿里巴巴等完全依靠网络的新型企业而言，注册会计师和信息系统审计师应该如何面对？

4. 找几个典型行业，分析这些行业对信息系统的依赖程度，讨论信息系统的作用。