FoxPro 2.5 For Windows

入門短平快

田志良　黃家新　杨　俊
陈　俊　王丽珍　刘唯一　编著

FOXPRO
2.5
FOR
WINDOWS

云南大学出版社

前　言

我国从七十年代末开始引进数据库管理系统，主要使用的数据库有 dBASE 系列和 Fox 系列。dBASE 是一个关系数据库管理系统，它由美国 Ashton－Tate 公司开发，先后推出 dBASED Ⅱ，dBASE Ⅲ、dBASE Ⅲ PLUS 和 dBASE Ⅳ 等几种版本，由于使用方便，性能优越而风靡一时，享有“大众数据库”的美誉。但 dBASE 也存在不少缺点，诸如：速度太慢，缺乏编译程序，人机界面差，命令和函数有限。随着 Ashton－Tate 公司被收购，昔日 dBASE 一统数据库天下的日子一去不复返了。而其它公司推出的兼容 dBASE 的数据库管理系统却异军突起，后来居上。其中的佼佼者有 Fox Software 公司的 Fox 系统数据库管理系统，该公司陆续推出的 FoxBASE 2.1 PLUS 具有伪编译能力，运行速度比 dBASE Ⅲ 快数倍，增加了不少命令和函数，能给出各种视角的 15 种二维图形和 32 种三维图形。但是仍采用圆点提示符这一极不友善的用户界面，也没有真正编译的工具。

此后，Fox Software 公司又相继推出 FoxPro 1.0、FoxPro 2.0，在 Fox Software 公司与 Microsoft 公司合并后，于 1993 年推出了 FoxPro 2.5，有了重大的提高，表现在：

第一，提供强有力的开发工具。诸如：屏幕生成器、报表生成器、菜单生成器、标签生成器和项目管理器。

第二，提供了一个真正的编译器，它可以脱离 FoxPro 环境，建立在 DOS 下运行的 EXE 文件。

第三，提供了 SQL 数据库标准查询语言。

第四，提供了 RQBE 举例相关查询。这是 FoxPro 的的一大创举，提供了一种新的方便快捷的查询方法。

第五，提供了 Rushmore 专利技术，极大地提高了数据库的查询速度。

第六，FoxPro 2.5 无论标准版还是扩展版都支持多用户。另外用 FoxPro 2.5 for DOS 建立的应用程序或数据不用改变，就可运行于 Windows、Macintosh 和 UNIX 操作系统下。

上述优点，使 FoxPro 2.5 越来越受到广大用户的青睐，实际上，它已成为微机上数据库管理系统的标准。

目　录

第一章　开发 MIS 的策略和方法

开发管理信息系统(MIS,Management Information System)是 FoxPro 的应用重点,目前国内几乎 90%以上的 MIS 是采用 dBASE、Foxbase 和 FoxPro 编写的,这三个系统都是一个家族的不同版本,其中 FoxPro 是目前开发 MIS 理想的数据库语言,要学习使用 FoxPro 开发 MIS,首先要对有关 MIS 的一些知识有所了解。为此,本章将详细地讨论有关 MIS 的四个问题:

(1)什么是 MIS;

(2)MIS 易于失败的原因是什么;

(3)怎样的 MIS 受到用户喜欢;

(4)如何成功地开发 MIS。

1.1　什么是 MIS

MIS 是一个由人和计算机组成的能进行管理信息的收集、传递、贮存、加工、维护和使用的系统。它由四个要素组成:

第一,现代经济管理理论;

第二,系统工程;

第三,数学方法;

第四,计算机。

其目的是通过自动化的事务处理方法来监视和控制企业的行为,通过对历史数据的科学统计提供可靠的预测数据,并通过一些优化方法的处理为决策提供依据。管理信息系统是生产力发展的阶段标志,是现代化管理的综合体现。

MIS 是一个高度创造性的系统,它具有如下特点:

第一,MIS 着眼于整体观点,强调管理职能的横向联系,使得它有利于分层管理;

第二,MIS 应用计算机和通讯技术,达到信息处理自动化;

第三,MIS 应用数学方法和系统方法来加工信息,为各层次提供决策依据;

第四,MIS 是一个人—机系统,它设置了若干人—机接口,以便充分发挥管理者的智慧和能动作用,使之更加适应环境改变情况下的管理信息处理。

1.2　开发 MIS 易于失败的原因在哪里

1.2.1　开发 MIS 所面临的挑战

MIS 用于企业,其业务处理是动态化的,表现在:

第一,企业经营的产品种类要有变化,否则失去竞争能力,仅仅是物料计划管理,其货源经常处于变化之中;另外,企业机构经常变化,中国尤其如此。

第二,虽然都是企业,但其性质大相径庭,如有生产型、管理型、经营型等企业,就生产型企业来讲,又可分为生产装配型、流程型等不同类型。正因为企业的千差万别,决定了以统一模式开发

MIS 基本上是行不通的。

第三，MIS 开发者与用户所面对的计算机工业是一个瞬息万变、充满风险的行业，激烈的竞争使计算机厂家拼命采用先进技术，CPU 芯片以两年一代的速度进行更新，开发者刚选定一种"最新"的机型开展工作，功能更强、性能更好的更新机器随即出现，以致 MIS 开发者和用户感到无所适从。开发 MIS 所采用的软件工具和技术，虽然没有硬件变化快，但仍使开发者疲于跟随。刚刚熟悉的语言又被新语言代替。近几年来具有 SQL 的数据库、面向对象的方法以及客户机/服务器等相继出台，使开发人员眼花缭乱，不知所措，开发水平刚被喻为国内先进，随之就落后了。

第四，单纯行政命令或为了达标而硬性给企业规定必须使用计算机，并不能保证 MIS 开发成功，而是成为一项劳民伤财的做法，几乎没有一个企业在"达标"后真正把计算机用起来。管理信息系统的目的是为了从管理中要效益，只有它的内外部运行机制都按市场经济规律办事，它的管理与管理人员都达到一定水平，才会萌生出用计算机进行管理的需求。MIS 的作用不外乎经营管理和决策，对于决策而言，要有特大量的信息支持，决策支持系统要有一个完整的信息网络，能够提供内部和外部、纵向和横向、近期和历史多方面的数据，要有各式各样的预测算法和预测曲线，在经济还没有完全脱离行政干预的情况下，用计算机决策支持系统给领导提供决策依据是不现实的。

1.2.2 MIS 系统本质的特征和规律是什么

开发计算机软件一直被两大难题所困扰：一是怎样克服程序复杂性障碍；二是怎样将现实世界的模型在计算机中自然地表示出来。MIS 是一个典型复杂的应用问题。目前，国内 70%的计算机用于 MIS，开发的 MIS 不算少，但成功的不多。大部分 MIS 在刚刚开发完成时，功能很强，用户很满意，但随着时间的推移，系统变得十分脆弱，这样的事情比比皆是。因为，当用户要求发生变化时，MIS 不能适应，只好寿终正寝。为什么会出现这样的情况呢？就在于有的开发者忽视了 MIS 系统最本质的一些特征和规律：没有一个数据库应用系统适合任何环境，能够在百分之百时间内正确工作。这一平凡的结论，有些开发者没有认真加以研究，然而这一点在现实中确又十分重要。长期以来有的开发者认为数据库的值是时变的，而结构是时不变的，在这观点指导下，若干 MIS 当随着环境变化时，需要结构也相应变化，开发者就无所适从。下面是这些变化的情况：

①信息定义的改变。例如，一个教学管理系统，最初约定一个教师只教一门课，后来由于教员不够，允许一个教员教两到三门课，这样教员和课程关系一对一变为一对多关系。

②新的字段需要加到数据库中去。

③数据表示改变。例如，有的字段开始用代码表示，但在使用中，发现反而不方便，要求码编方式改换成非编码方式，或者反过来。

④次要关系性质的改变。例如，在数据库最初建立时并不重要的教学工作量关系，只要求打印出报表，可是在使用一段时间后，对该关系增加几个字段，可以用来作为教师考评的重要关系，必须对该关系增加一系列操作功能。

⑤关系系统和闭合性质。关系数据库系统的最大特点是：由查询过程得到的新关系又可以作为系统的基本关系加以确定，从而使用更方便。例如，学生成绩视图关系，本来作为一个查询结果，但将其确定下来作为对学生成绩的输入和修改，的确显得更为方便直观。

⑥关系的划分与合并。在使用过程中，发现如果将某个关系化分成两个或多个关系会更合适、更方便应用。或者，多个关系的查询会影响速度，如果将几个关系合并，速度可能大大提高。

凡此种种都要求数据库能够自动适应新的变化情况。上述情况可以概括为数据库结构的改变和数据库应用模式的改变，从而我们得到关系数据库系统两个最基本的性质：

第一，数据库结构的时变性；

第二，数据库应用模式的时变性。

正是这两个基本特征决定了 MIS 的基本规律——MIS 是一个不稳定系统。一些开发者对于这一点认识不清，相应的措施没有或者不力，这就是导致 MIS 易于失败的主要原因之一。

1.2.3 传统 MIS 方法的缺陷

1. 功能分解法

功能分解法是一种分而治之的方法，它把一个大系统分解为若干个功能，又把这些功能分解为若干子功能，每个子功能接口描述功能化。功能分解要求分析员根据用户所提供的文档和交流的信息，准确地将主要任务映射到功能和子功能上去，这种准确性在分析阶段是很难验证的，完全决定于分析员对问题的理解程度，并且这种理解是因人而异的，没有客观准则。

这种方法最大的缺点是对 MIS 的易变性不好掌握，另外，方法本身所提供的手段，如描述工具、具体步骤等都不够细致，是粗放、非规范的，换言之，方法的可操作性差。但有一点是肯定的，就是对 MIS 的分析，采用分而治之的策略是行之有效的。

2. SASD 方法

SA 方法即结构化分析方法，其基本观点是把计算机看作将输入数据加工成输出数据的处理器。为了控制并简化软件结构的复杂性，将客观问题按功能需求进行自上而下的分解，得到若干子问题。所采用的主要工具是数据流程图(DFD)。这种软件分解具有层次结构，一直逐层求精直到满意为止。然后，进一步把 DFD 映射成模块结构及系统结构图。

这种方法是一个开环系统，没有闭环环节。当用户需求发生变化时，所画的 DFD 可能要作很大的修改，有的甚至要重新分析，可见这种方法也非常脆弱。其次，这种方法文档量太大，对稍大的 MIS，文档几乎难以完成。第三，DFD 描述手段虽能清楚地反映数据流程，但不能描述某些复杂情况，如实行并行，就连数据结构也不能清楚地描述出来。

3. 实体关系图(E－R)方法

该方法在逻辑数据库设计中被广泛采用。基本思想是在现实世界数据与逻辑数据库之间插入一个纯粹反映客观世界、与数据库具体设计无关的“企业模式”。该模式就是 E－R 图。E－R 图是客观世界的抽象，与用户的思维吻合，所以很容易被用户所接受，进而在开发者与用户之间找到一种共同交流的语言。一旦 E－R 图确定后，也很容易转换为逻辑数据库。

E－R 法最大的问题是，对一个简单的系统还可以画出 E－R 图，对一个稍大的系统，画 E－R 图就非常困难。

1.3 怎样的 MIS 才受到用户欢迎

一个成功的 MIS 需具备四个特点，才具有生命力，为用户所喜爱。

“快”，当用户提出需求后，一到两个月就可看到系统的局部实现，供用户试用，在试用中不断改进和完善，让用户建立对开发者的信任感。在半年到一年左右取代人工操作投入使用。

“活”，体现在三句话上“随动系统，在线维护，同步增长”。

MIS 要成为一个“随动系统”，这是针对 MIS 是一个不稳定系统而提出来的，其含义是 MIS 的数据库结构、数据库应用模式、处理算法、I/O 格式甚至系统结构本身都可随时更改，以提高 MIS 对于环境变迁的应变能力；“在线维护”含意是用户在使用过程中，根据需要随时要求修改系统不完

善的地方，并且不能长时间中断使用过程；“同步增长”含意是随着时间的推移，用户管理水平的提高，MIS 功能可随用户要求，通过修改同步提高，即 MIS 功能随时与用户管理要求保持一致。

“强”，系统操作简易，功能要强。

“低”，开发和维护 MIS 的代价要低，代价指时间和资金。

这些特点实质就是对用户要求最低，限制最少，时间和经费上最省，而得到效益最大且不冒失败的风险。

1.4 如何成功地开发 MIS

避免 MIS 建设失败要有正确的开发原则、开发策略，选择适用的开发方法、技术路线和管理模型。

1.4.1 开发原则

开发的原则是：规划长远，分步实施，着重当前，讲究实效。

1.4.2 开发策略

1. 开放性策略

一个系统，建立以后几年内不变化的情况是少有的，也就是说，人们要求系统扩充性好。在扩充过程中，要求原来的软件能在扩充部分中运行，这就是系统的可移植性，在扩充过程中，由于新技术不断涌现，扩充了的部分可能是性能更高的硬件和性能更好的软件，新扩充系统和原来系统之间就存在着是否可以互相操作的问题。几乎所有的硬件制造商和软件开发者都碰到上述问题并试图解决它们，不过各供应商所走的道路迥然不同，较大的供应商开始想在自己产品系列中做到可扩充、可移植与互操作，实质上是让用户走他们本公司的“开放”道路。众多中小供应商则考虑联合，走各种产品、各种系统之间可扩充、可移植与可互操作道路，用户马上发现走前一种开放性的道路，存在危险，即一旦该公司经营状况发生困难，用户无法从第三方寻求后续技术的支持。因而用户要求，不管硬件或软件，必须有多个供应商，这样，用户的投资才能得到保障。所以，目前公认的开放性标准是：可扩充性、可移植性、可互操作性和多供应渠道。

市场的需要就是动力。微机技术、小型分布式处理、客户机/服务器方式发展如此迅速，无一不是市场的因素作用。而市场因素归根到底是用户，用户希望自己的投资受到保护，希望自己的系统在新技术出现时能随之扩大，并把原来系统有机地包含在里面，还希望得到供应商连续不断的技术支持，这一切导致用户必然要走开放式系统的道路。

2. 小型分布处理的策略

企业各部门之间对计算机能力要求是不同的。设计部门往往要求提供计算能力和 CAD 方面的能力；生产部门需要数据处理的能力；销售部门需要通讯联网的能力；经理部门需要提供综合性数据，同时要求较多的办公自动化设备。这就决定了企业中很难用统一机型来满足各部门的不同要求，最好的办法是在不同计算机环境下建立不同的子系统，然后通过系统互连把企业的数据网建起来。企业各部门之间，为了共同目标有一个协同、协作关系，这已经成为当今企业文化的共同部分。反映到计算机方式上是各对等的机器间互连，或几乎是对等的网络之间的互连。小型分布式处理正适宜了这种发展的要求。

纵观计算机技术的发展，可分为三个阶段。第一阶段是以 IBM 大型机为中心的集中分时处理

阶段；第二阶段是以资源共享为核心的网络计算机阶段；第三阶段是用小型分布式来描述的客户机/服务器阶段，小型分布式处理之所以成为一种趋势，主要还在于其开放性。小型分布式处理的趋势绝对不意味着仅仅是计算机方式的改变，它所带来社会的、哲学的、企业文化形态的变化，是不可忽视的。它还带来设计思想的变化。在集中分时处理的年代里，要开发 MIS 必须要有完整的方案，因为所有数据都在主机中，数据冗余被看成是不合理的，各部门要共享数据库，因此必须规定各种数据的严格格式，各数据库字段与字段长度，等等。在分布式处理模式中，处理能力不再是集中的；内外资源不再变得如此昂贵；数据冗余作为安全可靠或提高运行速度的保证而常被采用。分布处理把本来是局部的视为一个整体，这样考虑问题就简单多了，也容易实现。它们在更大范围内通过各种数据接口与其它部门的数据交换也容易实现。

3. 整体规划、分步开发的策略

把产品设计、生产管理、市场销售、办公管理、辅助决策等内容统一规划，综合设计，设计目标既要先进又要可行。总体设计应当支持有步骤分阶段的实施，每一阶段的目标是有限的，强调整体设计水平，而不强调具体软件的设计水平和个性，把效益明显、相对独立、领导关注的应用领域作为重点，开发完毕一个，应用一个，对系统逐渐扩大，使系统之间可以互连，并可随时采用先进技术。使用户以较低的价格和较高的技术风险获得最好的性能，使开放式小型分布系统的优越性得以体现，使目标系统源于现行系统，又高于现行系统，使开发应用的过程，变成企业管理完善和升华的过程。

4. 用户参与的策略

建立一个实用的、易于操作维护的、使其真正发挥作用的系统。在研制过程中，让用户参与开发，使系统设计的每一步都得到用户的理解和肯定，从而减少反复。同时开发的过程也是培训用户的过程。

1.4.3 正确选择开发 MIS 的方法

1.4.3.1 MIS 开发的方法和比较

开发 MIS 普遍采用的方法是生命周期法和原型法。

1. 生命周期法及其特点

生命周期法的开发过程如图 1.1 所示。

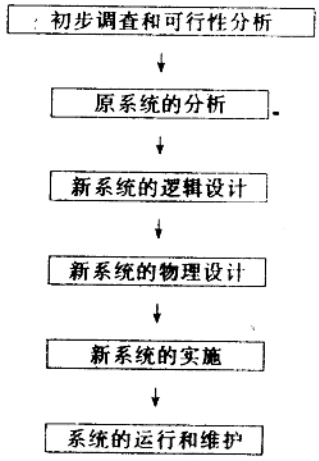

图 1.1 生命周期法开发过程

它把开发一个系统的全过程明确划分为若干阶段，在每个阶段指定了具体的工作任务和目标。每个阶段结束后，提交作为阶段工作结果的文档资料或软件。

因对各阶段指定了具体的工作内容和目标，所以即使各工作环节的联系十分复杂，也能对开发过程进行良好的管理和控制。因为对每一阶段的工作结束必须审议认可后，才能进行下一阶段的工作，所以可以最大限度地减少不必要的重复劳动。系统正式运行期间，一旦经营方式发生变化，可以通过查阅、修改各阶段的文档资料，对系统进行维护。生命周期法强调先分析、再设计、后实施的顺序，使系统方案产生于系统全面调查和分析之后，以确保实施后新系统的信息资源的共享。信息资源共享是高质量 MIS 的重要指标。

这种方法的缺陷是：

①开发过程中后一步的工作依赖于前一步工作的结果，这种简单的继承，导致如果前面工作有差错而没有及时纠正，就会在后续工作中被积累放大。

②开发初期，开发者对系统目标不是很清楚，用户对计算机能做到什么程度也很模糊，

这种双重模糊性给最终软件留下若干考虑不周和错误隐患。

③MIS 系统开发周期长，时过境迁，若企业情况发生变化，按原计划建成的 MIS 可能就不适用了。

④生命周期法的开发方式为：程序＋文档。它把注意力淹没在程序和文档的编制中，而系统开发的主要任务是功能和需要的开发。

(2) 原型法及其特点

原型法开发过程如图 1.2 所示。

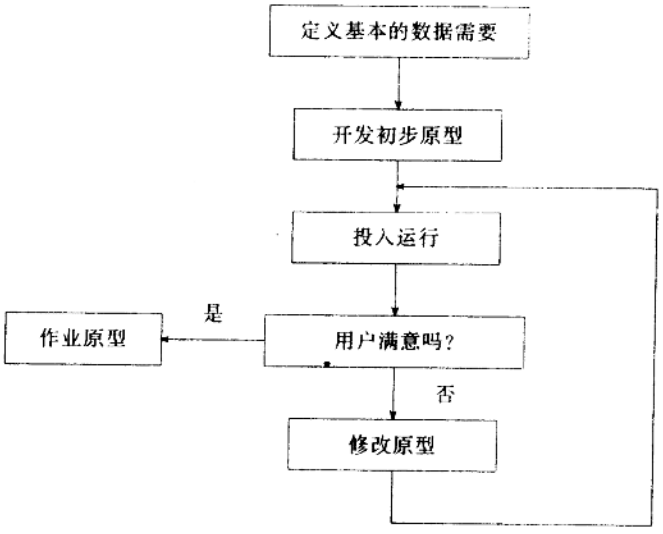

图 1.2　原型法开发过程

首先，根据用户基本信息要求编制程序，这程序叫做初始原型。通过原型的实际运行，逐步补充和明确系统的信息需求，修改和完善原型，把修改后的原型再投入运行，不断重复这一过程，直到用户满意为止，这时的原型称为作业原型。

使用原型法开发 MIS 时，因为在系统开发的初始阶段就提供了可运行的程序，就给用户以形

象的感觉，加强了用户对系统的理解。由于开发过程中不断征求用户意见，所以产生的作业原型能满足用户的需求。在系统的信息需求不明确时，或需求变化频繁时，使用原型法可减少开发工作的风险。所以原型法对于提高软件质量、缩短开发周期、降低开发成本和提高开发者工作效率都有积极意义。

这种方法存在的缺点是：

①具体怎样做不清楚。原型法的具体操作步骤仅仅是一些原则，具体怎样做不是很清楚。

②环境要求高。原型法要求在很短时间里构造出一个原型，这需要很高的环境支持，软件必须具有诸如生成器、屏幕绘图系统、功能语言等，要求开发者能够随时把用户意见及时转化为软件的规格说明，同时也要求用户具有一定的归纳能力。

③缺乏理论基础。至今为止还没有发现有人说出原型法的理论依据是什么。

综上所述，这些方法（当然还有其它方法）都是从实际中总结抽象得出来的，在长期的实践中，都起到了积极作用，同时也存在一定的问题，只有扬长避短，综合应用方能有效发挥它的作用。

1.4.3.2 选择 MIS 开发方法的依据和原则

在系统开发的初期，可按照图 1.3 所示的过程，根据系统自身和环境的特性、充分估计系统信息需求不确定性因素及影响程度；然后根据信息需求的不确定性，选取不同的保证策略；再根据选取的保证策略，确定系统的开发方法。

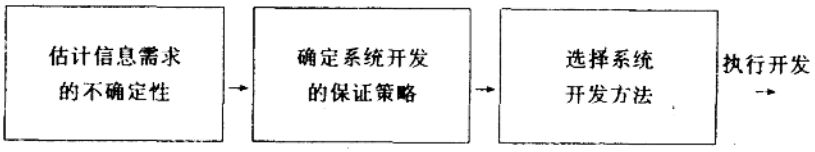

图 1.3 选择开发方法的过程

1. 估计系统信息不确定的方法

在开发一个系统的过程中，需要由用户描述原型的情况，系统分析员根据用户描述，设计新系统。描述原型和新系统所需的信息，叫做系统的信息需求。这些信息包括：系统的组织机构和经营管理的活动方式；系统具有的功能；需要向系统提供哪些数据；如何提供这些数据；系统怎样处理数据。若用户或分析员不能对系统的信息需求给出准确、全面的定义，就产生了信息需求的不确定性。在估计系统的信息不确定性程度时，主要考虑系统自身的特点、用户水平和分析员能力等因素。

(1)被开发系统自身的特点影响需求的不确定性。表现在：应用子系统规模较大，管理人员较多；系统经营活动方式不稳定；系统的组织机构变化频繁；原系统管理基础较差，缺乏提供被处理数据的渠道；系统的结构化程度低，数据处理方法模糊。

(2)用户水平较低造成需求的不确定性。表现在：用户不能全面准确地描述原系统的工作情况，因缺乏对 MIS 的认识，而不能提出对新系统的信息需求，或提出难以实现的需求。

(3)分析员能力较差，也会造成需求不确定性。表现在：缺少关于被开发系统经营活动方式的基础知识；理解用户意图能力较差；协调用户意见分歧的能力较差；不太熟悉系统开发方法或系统开发实践经验不足。总之，分析员在现代经济管理理论、软件工程、数学方法和计算机手段方面有所欠缺也是造成需求的不确定性因素之一。

上述各种影响程度越深，对系统信息需求的不确定性的影响越严重。

2. 确定系统开发的保证策略

根据信息需求不确定性程度由小到大的顺序，选择系统的保证策略依次有：接受保证、线性保证、重复保证和试验保证等策略。

接受保证策略：这种策略要求全面、严格地接受用户对需求的定义，并按此定义开发新系统。

当系统的信息需求完整、准确，而且在开发过程中用户需求基本不变，即系统的信息需求不确定性很小时，可采用接受保证策略。例如，机票预定系统、金融保险系统、图书检索系统等，这类系统的特点是业务处理方式基本不变。

线性保证策略：这种保证策略要求在系统开发的每个阶段工作完成后，采用指定手段核实该阶段的工作与需求是否一致，一旦出现差别，只需修改本阶段的工作结果。当用户可以提出比较全面、准确的信息需求时，分析员只须稍加修改原方案就可满足用户信息要求，而且系统信息的不确定性较小，这时可以采用线性保证策略。

重复保证策略：在系统的信息需求定义不够全面、不够确切时或预先估计到在系统开发过程中需求会发生较大变动时，或分析员对某些设计环节缺少足够把握时，可采用这种保证策略，在开发过程中通过必要的顺序重复，获取对系统正确、完整的定义。

试验保证策略：当需求的不确定性程度较高时，在系统开发初期很难对系统的信息需求作出较满意的定义，只能首先建立粗略的原型，通过不断试验，丰富信息需求，识别原型不足之处，反复修改原型，逐步减少原型与系统实际信息需求间的差距，逼近系统的实际目标。这种保证方法是试验保证策略。

对于一个具体的应用系统，尽管很难在两种相近的保证策略之间作出确切的选择，但是选择保证策略仍是确定系统开发方法的依据和基础。

3. 根据保证策略选择开发方法的原则

采用接受保证策略的系统，因结构化程度高，对信息需求的定义准确，可以选取市场上销售的相应软件包，或者购买或移植同类企业已经开发成熟的应用系统，在不具备上述条件时，可以使用生命周期法开发新系统。

采用线性保证策略的系统，应当使用生命周期法进行开发。对于采用重复保证策略的系统，应当选用与原型相结合的生命周期法。在使用生命周期法开发系统的同时，辅以原型，以丰富和修改系统信息需求的定义，为分析员积累系统设计的实践经验，减少风险。比如：设计的数据处理方法是否恰当、速度是否快、数据通讯是否会出现通道拥挤现象等。

只有选择试验保证策略的系统，才适合使用原型法进行开发。

1.4.3.3 两种系统开发方法的适用对象

(1)使用生命周期法制订MIS的总体设计方案。在总体设计方案中，应当包括子系统的边界定义、系统信息资源的规划和管理、数据库系统的设计、计算机硬件的配置方案等等。只有在总体设计结束后，才能开发各个应用子系统。按照这个顺序，可以协调各子系统信息需求，实现企业信息资源的共享。尽管在开发各子系统时，可能修改总体设计方案中的某些细节，但采用基于原型法、权益于某些子系统信息需求的做法是不明智的。

(2)对于事务处理层次的子系统，结构化程度比较高，信息需求不确定性的影响因素少，可以采用接受保证或线性保证策略。比如：直接从已有的数据库或数据文件中产生的打印报表、财务记帐子系统、仓库存货子系统、订单管理子系统、劳资子系统等。

(3)对于作业控制层次的子系统，信息需求不确定的影响因素较少，可采用生命周期法为主、辅以原型。比如：财务成本核算子系统、原料订货子系统、生产调度子系统等。

(4)对于生产控制层次子系统。信息需求不确性因素较多，宜于选择重复保证策略，使用生命周期法并结合原型法开发该系统。比如：财务结算和分析子系统、库存控制子系统、劳动力调度等等。

(5)对于战略计划层次的各种决策支持子系统，结构化程度低，大量数据取自企业外部，系统信息需求变化很大，只能选择试验保证策略，可用原型法来开发这种子系统。比如：制定财务计划、产品推销战略、新产品成本/效益分析等。

1.5 建立MIS的条件

企业管理信息系统可促进企业事务处理自动化、决策科学化，是企业实现管理现代化的综合体现，建立企业管理信息系统是一项复杂的系统工程，要投入大量人力、物力和财力，必须采取慎重态度。要建立企业的MIS必须积极创造一系列条件，主要有以下几方面。

1.5.1 科学的管理基础

没有一个完善的手工管理系统，就不可能建立完善的计算机管理系统。只有在合理的管理体制、完善的规章制度、协调的组织结构、稳定的生产秩序、科学的管理方法和正确的统计数据的基础上，才可能建立计算机管理系统。具体说，要使管理业务标准化、信息流程程序化、定额精确化、实体编码化、报表统一化…等。企业的经营方式管理体制正在发生重大改革时，最好暂缓开发MIS。只有当企业的内、外运行机制都按市场经济规律办事，企业的管理与管理员工都达到一定水平，才是开发MIS的时机。

1.5.2 有一支专业队伍

专业队伍包括系统分析员、程序设计员、计算机维护员，此外还要对企业管理人员和业务人员进行MIS知识的普及教育，使他们对MIS有足够的理解和认识，他们才能有效地支持和参加MIS的开发工作。

1.5.3 企业领导的决心和支持是成功开发MIS的保证

MIS实现全企业的职能管理，它将改变传统的手工管理模式。信息系统中产生的一系列数据文件和工作命令，体现了各职能部门的管理权限。只有领导亲自挂帅的情况下，才能把这一切确定下来。基于这一点，企业领导班子第一把手最好是本行业的里手行家，能够统管整个企业。第二把手最好是了解国内外MIS发展动态，能够选择合适的平台进行开发或移植的MIS专家，有这样两个人的密切配合，就可以组织起队伍，就有可能开发出成功的MIS系统。

再者，一个成功的MIS，必须有足够的投资来作保证，只有领导重视的情况下才有资金的保障。

第三，MIS系统最终提供的是上层决策信息，它是企业领导作出决策的重要依据。领导对MIS不积极怎么会正确使用MIS提供的决策信息呢？

第二章　关系数据库基础知识

2.1　关系模型及数据操纵

2.1.1　关系数据模型概述

关系数据库是用二维表来描述客观世界的，表的每一列称为属性，属性名要唯一，属性的取值范围叫值域。属性名表构成表的框架，称为关系模式。表的每一行叫做元组，元组的集合构成模式下的具体的关系实例。FoxPro 中的记录和字段与这里的元组和属性是一回事。

表 2.1 是描述教师情况的一个关系。

表 2.1

姓名	性别	年龄	职称
张三	男	29	讲师
李四	女	42	副教授
…	…		

其中，姓名、性别、年龄、职称为属性名，它们构成关系模式。每个教师的具体情况组成表的一个元组。

归结起来，“关系”应该具有下面的性质：

(1)属性名不能相同，同一属性属同一值域；

(2)任意两行(即元组)不能全同；

(3)每一数值都是不可分的数据单位；

(4)行列次序无关。

2.1.2　关系操纵

人们要从二维表中选取所需的行和列就必须对二维表进行操纵，这里介绍“关系代数”用以对二维表实行操作。

关系代数可分为二类运算，一种是传统的关系运算，另一种是特殊关系运算。

1. 传统关系运算

(1)并(∪)

设 r_1、r_2 为模式 R 下的两个关系实例。$r_1 \cup r_2$ 是属于 r_1 或属于 r_2 或同时属于 r_1 和 r_2 的元组组成的集合。

(2)差(－)

设 r_1、r_2 为模式 R 上的关系实例。$r_1 - r_2$ 是由属于 r_1 而不属于 r_2 的元组组成的集合。注意 $r_1 - r_2 \neq r_2 - r_1$。

(3)笛卡尔积(×)

设 r_1 为模式 R 上的关系实例，r_3 为模式 S 上的关系实例。$r_1 \times r_3$ 是由 r_1 的第一个元组开始依次与 r_3 中所有元组组合，然后将 r_1 的第二个元组直至最后一个元组与 r_3 中所有元组组合，若 r_1 有 m 个元组，r_3 有 n 个元组，$r_1 \times r_3$ 则有 m·n 个元组。

(4)交(∩)

设 r_1、r_2 为模式 R 上的两个关系实例，$r_1 \cap r_2$ 是同时属于 r_1 和 r_2 的元组组成的集合。显然 $r_1 \cap r_2 = r_1 - (r_1 - r_2)$

表 2.2 中描述了上述几种运算。

表 2.2

r1

A	B	C
a	1	c
b	3	d
c	2	e

r2

A	B	C
r	3	t
c	2	e

r3

D	E
a	1
b	3

r1∪r2

A	B	C
a	1	c
b	3	d
c	2	e
r	3	t

r1－r2

A	B	C
a	1	c
b	3	d

r1∩r2

A	B	C
c	2	e

r1×r2

A	B	C	D	E
a	1	c	a	1
a	1	c	b	3
b	3	d	a	1
b	3	d	b	3
c	2	e	a	1
c	2	e	b	3

2. 特殊关系运算

(1)投影(∏)

设 A_1、A_2、…、A_m 为模式 R 中的属性，r_1 为 R 上的关系实例。$\prod_{A_1 A_2}$ 是从 r_1 中选择 A_1、A_2 两列构成一个新的关系。注意在新关系中可能有重复元组，此运算包括有消去重复元组。

(2)选择(σ)

设 θ 是一个条件表达式，r_1 是模式 R 上的关系实例。$\sigma_\theta(r_1)$是从 r_1 中选取满足条件 θ 的元组。

(3)联接(▷◁)

设 $R=(A_1,\cdots,A_m)$、$S=(B_1,\cdots,B_m)$是两个关系模式，r_1、r_2 分别为 R、S 上的实例。$r_1 \triangleright\triangleleft_{A_i \rho B_j} r_2$ 表示 $r_1 \times r_j$ 中满足 $A_i \rho B_j$ 条件的那些元组，其中 ρ 为比较运算符。当 ρ 为"="号时，称为等值联接。在联接运算中，有一种最重要的联接就是两个关系中共有属性的等值联接，称为自然联接，记为 $r_i \triangleright\triangleleft r_j$。

表 2.3 描述了上述特殊关系运算。

表2.3

r1

A	B	C
1	3	5
2	3	5
4	2	1

r2

B	C	D
4	3	1
2	1	1
2	1	4

$\prod_{BC}(r_1)$

B	C
3	5
2	1

$\sigma_{B>2}(r_1)$

A	B	C
1	3	5
2	3	5

r1▷◁r2

A	B	C	D
4	2	1	1
4	2	1	4

2.2 问题的提出

属性间存在着复杂的相互关系，其中最基本也是最重要的关系称为函数依赖。函数依赖极为普遍地存在于现实生活中，描述一个学生的关系有学号(xh)、姓名(xm)、性别(xb)、出生年月(sr)等几个属性，由于一个学号只对应一个学生，一个学生只有一个年龄和性别。因而"学号"值确定以后，其它属性，诸如姓名、性别、出生年月等也被唯一的确定了。就象自变量 x 确定之后，相应的函数值 f(x)也就唯一确定了一样，我们说 xh 函数决定了 xm、xb、sr 等，或者说 xm、xb、sr 函数依赖于 xh，记为：xh→xm，xh→xb，xh→sr。

我们再分析一个例子，假如要建立一个数据库，对象有：学生(用学号 S# 描述)、系(用系名 SD 描述)、系负责人(用其姓名 MN 描述)、课程(用课程名 CN 描述)和成绩(G)。于是我们得到一组属性，记为：

$$U=\{S\#,SD,MN,CN,G\}$$

现实世界的已知事实告诉我们：

1. 一个系有若干学生，但一个学生只属于一个系；
2. 一个系只有一个负责人；
3. 一个学生可选多门课程，每门课程有若干学生选修；
4. 每个学生学习每一门课程有一个成绩。

于是得到属性集合 U 上的一组函数依赖，记为：

$F=\{S\#\rightarrow SD, SD\rightarrow MN, (S\#, CN)\rightarrow G\}$

这样我们得到一个关系模式：$S<U, F>$，但这个关系模式有三个“毛病”。

1. 如果一个系刚成立，尚无学生，或者虽然有了学生，但尚未安排课程。那么，我们就无法把这个系负责人的信息存入数据库。这叫插入异常。

2. 反过来，如果学生毕业了，我们在删除学生选修课程的信息的同时，就会把系名及其负责人的信息也丢掉了，这叫删除异常。

3. 冗余大，比如每个系负责人的姓名与该系每一个学生的每一门课程的成绩出现次数一样多。这样，一方面浪费存储，另一方面系统要付出很大的代价来维护数据库的完整性。比如，某系负责人更换后，就必须逐一修改有关这个系学生选修课程的每一个元组。

由于上述三个“毛病”，它是一个“不好”的数据库模式。一个“好”的数据库模式应当不会发生插入异常和删除异常，冗余应尽可能少。

为什么会发生插入异常和删除常呢？

这是因为这个模式中的函数依赖存在某些不好的性质。假若我们把这个单一的模式改造一下，分成三个关系模式：

$S<S\#, SD, S\#\rightarrow SD>$

$SG<S\#, SN, G, (S\#, CN)\rightarrow G>$

$DEPT<SD, MN, SD\rightarrow MN>$

这三个模式则不会发生插入异常、删除异常的毛病，数据的冗余也得到有效的控制。

一个模式的函数依赖会有哪些不好的性质，怎样将一个“不好”的数据库模式，改造成“好”的模式，这就是下一节规范化理论讨论的内容。

2.3 规范化

2.3.1 函数依赖

前面已经介绍了什么是函数依赖，下面介绍一些记号和术语：

- 若 $X\rightarrow Y$、$Y\rightarrow X$，则记作为 $X\longleftrightarrow Y$；
- 若 Y 不函数依赖于 X，则记作 $X\nrightarrow Y$；
- 若 $X\rightarrow Y$，并且对于 X 的任何一个真子集 X'，都没有 $X'\rightarrow Y$，则称 Y 对 X 完全函数依赖，记作：$X\xrightarrow{f}Y$；
- 若 $X\rightarrow Y$，但 Y 不完全函数依赖于 X，则称 Y 对 X 部分函数依赖，记作 $X\xrightarrow{p}Y$；
- 若 $X\rightarrow Y$、$Y\rightarrow X$、$Y\rightarrow Z$，则称 Z 对 X 传递函数依赖。加上条件 $Y\rightarrow X$ 是因为如果 $Y\rightarrow X$，则 $X\longleftrightarrow Y$，实际上 $X\rightarrow Z$ 是直接的，而不是传递函数依赖。例如：$U=\{$职工号(E#)，工资级别(SG)，应发工资(S)$\}$，$F=\{E\#\rightarrow SG, SG\rightarrow S\}$，则 S 传递依赖于 E#。

2.3.2 码

码也称为关键字，直观地，学生学号一旦确定，学生所有属性都被确定，我们说学号是学生这个模式的码。为了准确描述，我们给出下面形式化定义。

设 F 为模式 R 上的函数依赖集，U 为 R 上的属性集，K 为 R(U,F)中的属性或属性的组合，若