第 3 章

CHAPTER 3

从 0 到 1 设计系统

一个完整的系统项目从需求者心中的设想到实现通常需要经过项目立项阶段、需求分析阶段、系统详细设计阶段、开发与测试阶段、部署运维阶段、项目收尾阶段等多个阶段。

在项目立项阶段中，项目组根据客户提出的项目需求对项目目标进行明确，同时研判和制定项目在开发过程中的进度、成本、范围等计划书，并形成项目的总体计划；在需求分析阶段将对客户所提出的需求进行详细分析并形成需求文档，在这一步中会对系统的功能模块进行确定；系统详细设计阶段则根据需求文档对系统进行软硬件方面的设计，包括系统架构设计、界面设计、数据库设计等，UI 设计师会在此阶段根据需求及其原始模型制成设计图供客户确认，详细设计阶段是一个频繁跟客户进行沟通的过程；开发与测试阶段是项目开发组将功能需求实现的过程，在此阶段中前后端开发人员与测试人员要充分沟通并制定开发计划和测试计划，保证项目按时开发完成；部署运维阶段中的部署是项目进行上线的过程，运维则是日常维护系统以保证系统正常运行的过程；当项目经过以上众多阶段并正常运行无任何故障后，则进入了项目收尾阶段，至此整个项目结束。

在本章中，将针对需求分析阶段如何通过客户的不同需求去形成对应的功能模块进行探讨，并结合多种方式对数据库字段进行设计，有助于读者了解项目从 0 到 1 的设计过程。

3.1 功能模块是如何讨论出来的

系统的功能模块是项目组通过分析和讨论需求分析阶段形成的需求文档而得出的，而需求文档的初步形成需要经历多个阶段。

首先是销售经理与客户的初步沟通，销售经理会明确客户的大致需求，为什么说此刻的需求是大致的而不是完整的呢？因为客户通常并不具备信息化系统项目开发的相关知识，往往是基于某种可能影响企业工作效率或者成本方面的问题而形成的一个初步需求，例如全纸质化办公会增加耗材成本、查找资料需要花费大量时间成本、人工操作可能造成存档丢失等问题。在了解需求的过程中往往考验销售经理的技术功底，销售经理需及时对客户提出的问题给予口头上的解决方案，例如介绍适合项目体型的服务器参数、数据库管理系统等，所以在小企业中通常由项目经理兼任销售经理。

在了解大致的需求后，销售经理会与项目经理对项目进行初步评估，形成对项目的初步开发方案和预算表，并对客户进行反馈。在这一阶段中，开发公司内部会对项目进行可行性评估，如果评估报告中提到项目过于庞大复杂，则销售经理会对项目可能需要分包的部分准备招投标工作，从承包方的角度来讲就是所谓的二次分包。

在本节中，将解释系统从设想到立项的每个过程，并简要介绍市面上复杂的多端应用架构设计，同时读者可对常见的功能模块及其操作有一个简单的认识。

3.1.1 从设想到立项

立项是信息系统生命周期的初始阶段，销售经理申请项目立项，项目管理中心审查项目立项资料并根据项目大小程度任命项目经理，同时开发公司将进入需求调研阶段，对项目进行可行性评估、对需求进行分析并逐一明确、对用户开展需求调研、制定详细设计报告书，这是正式实现项目需求的前奏。

在需求分析阶段中，项目经理会与客户进行多次沟通，开发组成员会对需求进行充分调研，在保证项目最终目标不变的情况下引导客户或用户更加具体地描述系统应用场景。一个设想就像一个小点，需求则让小点不断地向外延伸，最终形成一个完整的圆。

1. 明确项目需求

在系统的复杂度上，如果以企业大小进行区分，则一个简单的企业系统可能只是单组织、多用户的形式，也就是适合小型企业内部使用的系统；如果是复杂的企业系统，例如有多个下属子公司的企业所使用的系统，就是多组织、多用户的形式，在详细设计时通常需要考虑不同子公司之间的协同合作和信息共享，同时对数据进行隔离。在面对多种不同权限用户的情况下，对功能模块的权限划分也变得尤为重要，项目经理必须详细了解客户所属企业的职能划分，保证系统内部的信息安全。面对复杂的项目情况，往往需要更多的时间成本才能明确每项的具体需求。

以生产型企业为例，项目经理需要充分了解企业从生产到售出成品过程中的每一步，例如生产原料的采购流程、采购订单的审核流程、原料制作的生产过程、成品的出入库及审核流程、订单管理的流程等。另外，生产型企业内部通常还有账单管理系统、客户关系管理系统(Customer Relationship Management，CRM)等。针对每个流程，项目经理都需要有大概的实现思路，如系统要面向的使用群体分为几类、系统的界面风格特点、某项流程可能有多少个功能模块等，还需要根据丰富的项目开发经验对开发周期进行估计。

除了与客户进行沟通外，项目小组内部则会展开对项目需求进行调研的环节。项目小组由项目总监、项目经理、UI设计师、前后端开发工程师、数据库工程师、运维工程师等人员组成，需求调研可以帮助项目小组更好地理解客户的需求，明确项目的功能模块，为后续的开发与测试阶段提供基础。在调研开始之前，项目总监主持召开小组调研会，明确调研的目的和范围、确定调研的内容及对象、设计调研的实施方案、筹备调研使用的工具材料等，形成需求调研计划。调研的方式有很多种，包括派出组内成员对客户所在企业的管理人员进行

访谈、研究与客户企业相同产业类型的系统功能、组织相关行业的专家开展流程分析，甚至派出小组成员进驻客户企业短期实习等多种方式。一份周密详细的需求调研计划可让客户感受到开发公司的专业，使其重视调研并全力配合，节省调研时间和提高调研效率，增进双方的熟悉度和好感度。调研可以更加明确客户提出的需求，把客户的需求抽象为具体的功能模块，为系统流程图的制作提供逻辑基础。

在满足客户的期望需求中，项目经理还可根据经验添加额外需求，给予客户出乎意料的惊喜，达到提高客户满意度的效果。同时项目经理也要预防可能出现的需求镀金和需求蔓延，需求镀金是当客户对某些功能需求表现出过于热衷的时候，出现忽略了其他功能重要性的情况，以生产型企业来讲就是过于偏重某个流程而忽略其他流程，这将会导致项目的各个模块在平衡度上出现偏差；需求蔓延则是当项目进入开发阶段中，客户在原定的需求列表上不断地提出新的需求或对原有需求进行功能扩展，这将造成项目的范围不断扩大，开发进度不断延期，开发成本也不断上升，最终可能导致项目无法在合同签订日期交付。针对这两种情况，通常可以采取敏捷开发、快速迭代等方式去满足客户的需求，但最好的方式还是准确理解客户的需求并制定完整的需求文档。

需求文档在整个开发过程中通常会经历多个版本的迭代，所以需求文档会在开头标注版本号及文档形成的日期、修改人、修改内容和审核人等信息，方便开发人员了解最新的需求变化。

2. 详细设计

仅有需求就像是画大饼，空有想法而没有实物，详细设计阶段则是把需求从虚变成实。系统详细设计就是在需求形成的前提下对各个功能模块进行实体化分析和设计，包括数据库设计、用户界面设计、功能接口设计、代码风格设计等，这是从设想向具体实现迈出的重要一步。

在已有需求的阶段上，数据库工程师首先需要根据项目的使用场景确定不同的字段及数据类型，例如关系数据库会采取 E-R 图的形式确定不同的对象实体、属性（字段）和关系；依据可能要存储的内容考虑数据库容量的大小，设计数据预留扩充和性能优化方案；对数据库的访问模式进行分析，对以写为主、以读为主或读写均衡的访问频率设计不同的数据处理方案；在数据安全性方面，设计数据加密、权限控制、身份验证等多种方案，防患于未然；在系统的可用性上，还需根据系统的稳定性、安全性、可靠性等因素设计数据定期备份方案、异地容灾方案、数据审计方案等。设计一个好的数据库模式能够最大限度地满足客户的应用需求。

由于用户界面是直接呈现在客户眼前的，故而对于开发公司来讲 UI 设计是真正的重中之重，相比之下客户看不到的数据逻辑处理反倒显得不那么重要。一个能让客户眼前一亮、轻松舒服的交互界面能减少客户在长时间使用过程中的疲劳感，提升客户的使用体验，给予客户有探索这个系统的欲望。在 UI 设计的过程中，需要反复地跟客户进行沟通，了解用户的使用习惯，例如一个面向企业高层的管理系统，色彩风格一定要简单、朴素，内容以清晰明了的可视化数据为主；如果是面向青年为主的应用系统，例如青年旅社预定系统、游戏社交平台等，色彩风格一定要鲜艳、多样，通常还具备动态效果的页面风格。UI 设计在明确

产品定位、受众人群后需要对整体风格进行把控，选择统一风格化的组件，可以让系统看起来更加协调，以基于 Vue 3 框架的 UI 库 Element Plus 为例，以蓝色为主色调，采用统一的颜色和图标，使整个应用的界面看起来具有一致性。

从交互的角度来考虑，UI 设计师需要尽可能地减少不必要的交互，同时对用户的操作提供足够的反馈。反馈包括控制反馈和页面反馈，完善的控制反馈可以让用户在与页面的交互过程中清晰地感知到自己在操作，而当操作后，页面反馈可以使页面元素清晰地展现当前状态。反馈也体现在用户操作之前，系统应给予用户充分的操作建议或安全提示，帮助用户做出正确的决策。

在详细设计的过程中，功能接口的设计是最为复杂的，即设计系统业务的核心功能，如果没有设计好，则会为系统的可用性、安全性埋下隐患。首要考虑的是接口的安全性，后端工程师在设计中需要确保数据在交互过程中不会发生数据泄露，通常会通过提供 Token、动态路由、权限控制等保证接口的数据传输安全；其次是接口的性能设计，在面对海量数据的情况下，需针对不同的场景设计不同的算法处理查询操作，保证接口在面对高并发的情况下发挥良好性能，并对可能发生的极端情况预留 Plan B 接口；在后端设计接口的时候还需遵循一个安全原则，就是后端永远不能相信前端传过来的数据，对每个接口接收的参数都需经过过滤、清洗才能执行 CURD 操作，并参数化 SQL 查询，防止 SQL 注入、数字注入、字符注入等攻击，保证数据库安全；接口还应具备清晰的注释和反馈信息，需对可能出现的异常情况设计不同的状态码，方便程序员在开发时能快速地定位错误并解决问题，提高接口的容错性和可维护性；考虑到公司未来的发展可能性，接口还需进行可扩展性设计，预防当现有业务发生变化时，已有接口不能处理业务而导致有损公司收益的情况发生，通常需结合公司的战略发展去规划接口的可扩展性。

后端开发人员在初步设计接口后输出接口文档。接口文档顾名思义，是一份包括每个功能的实现接口、接口的请求方式、需要传入的参数及其限制说明，返回结果示例、可能出现的错误状态码及调用示例的文档。通常还有接口文档和接口规范文档之分，接口规范文档规定了接口的通用规则和标准，保证接口在不同功能模块之间的良好兼容性。通常在接口文档中以功能模块对所属接口进行区分，例如用户模块可能有更新用户信息接口、获取用户列表接口、获取用户具体信息接口等多个接口。一份有高质量的接口文档可以帮助前端开发人员更好地理解和使用接口，减少与后端开发人员的沟通成本，方便开发人员及时了解最新的接口变化。

在开发组内部，还需设计编程的代码风格(规范)。首先是代码的命名规范，命名要能够清晰地展示其用途，在前端通常会对模板的类名使用串行命名法、用全大写描述常量、使用驼峰命名法规定函数名和组件的命名，在后端则通常使用蛇形命名法对数据库的表名、字段名进行规定。在开发中还会使用插件工具对代码进行规范检测，如前端针对 JavaScript 语法的检测工具 ESLint 和格式化工具 Prettier；其次是代码的注释风格，面对只有代码没有注释的函数就像是钻入了充满迷雾的迷宫，需要花费大量的时间才能理解函数内部的实现逻辑，开发人员通常会在函数顶部使用多行注释描述函数功能、形参数量及其数据类型(主

流的代码编辑器(如 IntelliJ IDEA、WebStorm、VS Code 等)提供形参及数据类型提示功能),在函数内部使用单行注释描述语句的执行条件,在后端的函数处理文件中会在开头使用注解描述当前模块使用的所有参数。在设计代码风格时首先需要考虑代码的可读性,特别是在模块化开发过程中,可能存在多个组件引用的公共代码,开发人员应以方便小组其他成员阅读和理解的角度去写代码。良好的代码风格不仅能提高开发效率,在后期的维护中也能降低时间成本。

项目经理在详细设计阶段收尾工作需要出具《系统详细设计验收报告》,邀请客户共同确认验收报告并签字,这样客户就不会在后期随意更改项目需求,减少开发期间出现需求镀金和需求蔓延的概率。

3.1.2　客户端的多端设计

如果一个系统是可售的,即面向客户的,类似于 SaaS(Software as a Service,软件即服务)系统,则在最简单的情况下都会具备商家后台端和客户端,商家后台端通常用于设计客户可选的功能套餐和服务时长,当客户向商家支付费用后,可通过网站或软件登录系统使用购买的服务。

SaaS 系统开发商会根据大部分行业通用的流程去设计系统的基本功能。以生产行业为例,生产行业通常会具备采购原料的流程,在这一过程中客户要输入采购的原料信息,例如采购型号、原料单位、原料数量、采购时间等,那么这些字段就具备通用性,适合生产行业的不同产业,而当客户购买了系统服务后,还可在通用的字段上进一步修改成符合自身产业链的字段。在 SaaS 系统中,用户无须担心系统的运维问题,即开即用的特性使不少企业在面对非核心业务时会选择方便快捷的 SaaS 系统。

商家后台端和客户端虽然面对的是不同的对象,但从后端开发的角度来讲都同属于客户端(页面端)的范畴。在面对多个客户端的情况下,项目组就需要考虑如何设计业务逻辑和应用场景。

项目组在分析需求时就要确定系统所支持的设备或平台,如淘宝商城可在网站登录,也可以在手机端登录,这是物理意义上的多端,UI 设计人员需要考虑在不同设备和操作系统上的页面兼容性,以及在不同场景下使用人群的页面风格;后端工程师则需考虑以尽量简单的数据类型去表达场景所需内容,保证在不同类型的设备端上能够正常地进行数据交换和保持数据的同步性。

在功能需求设计中,首先需要注意的是功能是否具有统一开关模块,例如在开发商操作的前端页面可能存在用于控制某个功能是否可以使用的总功能模块,也就是说所有的功能模块都存在于这个模块的框架下,如果在这个模块内关闭了某个功能,则所有的子系统、客户系统都不能使用该功能,这样做的好处是当某个功能出现 Bug 时能第一时间停止此功能的使用,减少系统和客户的损失;其次是需要划分好功能的所处区域,如果是单个客户端的一体化系统,则功能会根据不同的模块进行划分,如果是多端系统,则功能可能会以不同的客户端进行划分。还是以购物网站为例,通常除了开发商的用于管控系统的端,还会有卖家

端和买家端,反馈功能在卖家端和买家端都有可能存在,但卖家端不能对商品进行购买,只能管理买家购物的订单,即买家端有购买商品功能,而卖家端有订单管理功能。

在如今的应用市场中,多端已经成为大多数软件的标配。在物理设备上,不少软件涵盖PC端、手机端、平板端、智能手表端等;在实际应用上,不少软件除了App还开发了能够在微信或支付宝小程序上使用的端。多端能够让用户在不同的设备上使用软件,在方便用户操作的同时还保证了数据的一致性,提高了使用效率,但同时多端的设计也带来了维护成本高的问题,当系统更新时,需要同步更新多个端的内容,并且需要考虑到不同端的兼容性问题,因此,在设计多端的系统或应用软件时,极大地考验了顶层设计的逻辑完整性,只有综合考虑所有的影响因素,才能实现系统在多端的最大效益。

3.1.3 常见功能模块及操作

各行各业的系统因为其所处的环境不同,导致其在核心业务上具有差异性,但是在功能模块上,通常具备以下几个功能及操作。

(1) 注册和登录功能:用于用户注册账号及登录系统的功能。

(2) 个人信息管理功能:用于设置用户的昵称、性别、年龄、职位、个人简介等。

(3) 用户管理功能:用于管理使用系统的用户,通常包括对用户信息进行修改,以及对账号进行冻结、解冻、封禁等操作。

(4) 权限管理功能:用于设置用户在系统内的操作权限,通常包括设置用户是否可访问某个模块,以及对模块内的数据是否可进行查看、修改、上传和下载等操作进行限制。

(5) 查询功能:用于对存在列表的模块进行查询操作,一般可分为精准查询和模糊查询两种查询方式。

(6) 数据统计和分析功能:信息化系统的特点之一,对系统内的数据进行统计和分析,如销售系统可展示近七天的销售额,并根据销售数量分析产品的市场接受率。

(7) 反馈功能:系统为了方便用户使用和完善系统功能,一般会开设反馈功能以供用户提出对系统的改善建议,同时也包括对用户的操作提供反馈信息。

(8) 信息推送功能:用于向用户发送企业内部公告或系统版本信息等。

(9) 系统设置功能:通常包括系统的基础信息设置和网站内容管理等。

以上这些功能模块是不同行业内部系统的通用功能,也是构成一个企业系统的基本功能。通常系统设计者会根据企业的组织结构、业务流程、生产要素等实际情况对上述基本功能进行选择和扩展,在丰富系统功能的同时满足最基本的业务逻辑。

3.2 如何设计数据库字段

在2.1.1节关于数据模型的关系模型介绍中,曾简单地提到过数据库的字段,它是数据表中某一列的列名,代表表中的一个属性或特征。在本节中,将从字段的命名、数据类型、约

束和功能判断的角度简单讲述如何设计数据的字段。

3.2.1　字段的命名

在数据表中，每个字段都有唯一的名字，在设计数据表时要根据字段所代表的意义设定字段名，以及字段的数据类型、长度、是否为主键等属性。

通常字段名在单词格式上会遵循下面几个规范：

(1) 应使字段名具备直观的描述性，尽可能地保持字段名简洁。

(2) 字段命名通常由小写单词构成，如需多个单词，则可通过下画线进行连接，也可使用驼峰命名法，一般不超过 3 个单词。

(3) 字段名一般采用名词或动宾短语，便于理解字段的内容和含义。

(4) 字段名禁止缩写，如将 object 写成 obj 或将 user_id 写成 uid 等。

(5) 字段名禁止与表名重复，并且不包含数据类型(如 string、varchar 等)的单词。

(6) 字段名在表达多个个体或数量时，单词应使用单数而不是复数。

在详细设计的过程中，项目开发组会对代码规范进行讨论并设计，在此期间也会对数据库的命名以简洁、明确和易于理解的原则进行规范，包括数据库命名、数据表命名及字段命名。开发人员在对数据库进行建模时应遵循公司命名规范，确保数据的一致性和可维护性。

3.2.2　字段的数据类型

在对字段的命名进行明确之后，开发人员会对字段的数据类型和长度进行选择和确认，那么应该如何选择字段的数据类型呢？

如果要存储的是数值，则要根据业务的需求确定字段的数据范围和精度，在 MySQL 的数值类型上，开发人员可以选择整数类型(包括 TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT)或浮点数类型(包括 FLOAT、DOUBLE、DECIMAL)，例如在面对交易金额、存储利率这类需要高精度和范围的数值时，通常会选择浮点数类型；在面对学生学号、身份证号、产品订单号等唯一标识符时，通常会选择整数类型。

如果要存储字符串类型的数值，则 MySQL 中丰富的字符串类型可供多种场景选择，一个合适的字符串类型在运行中能够节省空间并提升性能。下面简要介绍几种常用的字符串类型。

如果存储一般的字符串数据，则通常会在 CHAR 类型和 VARCHAR 类型中进行选择，CHAR 类型用于存储长度固定的字符，而 VARCHAR 可存储可变长度的字符，在两者的比较中因为 VARCHAR 的可变特性，也使其比 CHAR 类型更节省空间；如果需要存储大量文本数据，例如新闻的稿件、科考论文、电影剧本等，则使用 TEXT 类型是个不错的选择，TEXT 类型是 MySQL 中支持存储大量文本数据的字符串类型，在 TEXT 类型家族中，还有 TINYTEXT、MDEIUMTEXT、LONGTEXT 三种类型，分别对应不同的长度；当存储音频、视频等媒体对象时，在后端通常会转换成二进制的格式进行存储，这时就需要使用

BLOB类型,这是一种用于存储二进制数据的字符串类型;在MySQL支持的字符串类型中,还有相对特殊的ENUM类型和SET类型,ENUM类型即枚举类型,用于存储预先定义好的字符串值列表,例如网页游戏中的装备数值、天气状态等,SET类型与ENUM类型类似,用于存储一组不重复且最多64个字符串值。

在面对需要进行单选的操作时,可使用布尔值进行判断,例如存储用户的性别、用户账号状态等,布尔类型包括BOOL类型和BIT类型。BOOL类型是MySQL中的标准布尔数据类型,是通过TINYINT类型实现的,当使用FALSE或TRUE作为布尔值时会被自动转换成0或1;BIT类型是一种二进制数据类型,可以存储一个或多个位的值,也可以为NULL值,当存储一个位时可表达为布尔类型,当存储多个位时可用来表达用户的权限,如使用三位二进制来分别代表读、写和删除操作。

如果使用Navicat创建字段,在选择完数据类型后,则通常会自动生成最适合该数据类型的长度。以VARCHAR类型为例,默认长度一般为255,代表能存储256个字符,将VARCHAR类型长度限制为255可以节省磁盘空间和内存,在对数据进行检索或操作时,较短的字段效率更高,因为减少了I/O操作和内存消耗,进而降低了查询复杂度和响应时间;如果为INT类型,则所表示数值的默认长度一般为11位,共4字节,代表能存储从−2147483648到2147483647的整数,在MySQL的早期版本中,默认所表示数值的长度为10位,但在某些情况下可能会出现数值过大而导致的溢出问题,所以所表示数值的默认长度被官方修改为11位。在本书的项目中,除对个别字段进行了长度修改外,其余字段皆使用默认长度,具体的长度限制会在后端代码中进行限制,这样在学习和使用数据库的同时,还可以减少读者在设计和维护数据库的负担。

3.2.3 约束

在设计数据库的字段时可根据使用场景定义不同的约束,约束是字段需要遵守的规则,主要用于保证数据的完整性和一致性。下面将介绍几种常用的约束及其使用场景。

(1) 主键约束(Primary Key):主键是数据表中用于标识记录唯一的字段,记录唯一也称为实体完整性,每张数据表只能有一个主键。常见的如学生表的学号、用户列表的手机号等都可设为主键,在一张数据表中,主键值是唯一的和不为空的,不为空也称为非空约束。

(2) 非空约束(Not Null):非空即字段的值不能为空,使用关键字Not Null来创建非空字段。还是以学生表为例,非空约束除了用于主键之外,学生的姓名、年龄同样应不为空,这对于确保数据的完整性和一致性非常重要。如果管理员在插入数据时未提供非空列的值,则将会引发数据库错误,这也是一种保证数据完整性的防范提醒。

(3) 唯一约束(Unique):唯一约束用于保证字段的值在表中具有唯一性。与主键不同,一张表可以有多个唯一约束。以用户表为例,除了用户自身的id,其所拥有的联系方式(如手机号、邮箱等)也都具有唯一性。在某些情况下会出现主键由多列组成的情况,这时可以选择用唯一约束来替代主键,以减少主键的复杂性。唯一约束还可加速查询操作,通常管理员会添加唯一索引值来快速定位满足条件的行。

(4) 外键约束(Foreign Key)：如果在当前表中的某个字段引用了其他表的主键，则称这个字段为外键，外键加强了两张表之间的一列或多列数据的联系。在主键约束中提到，主键是非空的，进而当前表中的外键，也一定存在并且是非空的，这就是外键约束。

(5) 检查约束(Check)：检查约束即限制字段的值范围，例如用户年龄范围、学生成绩范围。需要注意的是，在 MySQL 中不支持检查约束。

综上，在实际开发中，约束具有非常重要的作用，可以有效地保护数据的完整性、提高查询效率、方便多表数据联动和防止非法数据的输入，数据库开发人员应严格按照需求设计时确定的约束创建字段。

3.2.4　功能的判断

在设计数据表时，还需考虑添加用于判断状态的字段，这类字段通常具备初始值，一般为 NULL 值或自定义的默认值，如 0、1 等。

在 MySQL 中，NULL 值是表示空值的特殊值，在新建字段后如果没有定义初始值，则默认值为 NULL。程序员大多拥有自己的 Git 仓库，假设要将仓库的一个项目进行开源，就需要选择一款开源协议，如 Apache License 2.0、OpenSSL License、MIT 等，在没有选择开源协议之前，可以想象到开源协议在当前仓库的数据行中的值为 NULL。再例如在学生表中存在表达学科总成绩的字段，在没有得到所有学科的成绩前，可将值设为 NULL。如果当前字段的值只需获取一次性数据，则可以通过判断该值是否为 NULL 再决定是否插入值。

在实际开发中通常会使用 0 和 1 作为初始值去对某些功能进行判断。以回收站为例，这是计算机、手机都必备的一个软件，当删除某个文件时计算机首先会将文件保留在回收站内，只有当用户进入回收站进行真正删除操作时文件才会被销毁，在这一过程中，文件处于正常状态时可表示为 0，当文件在回收站时可表示为 1，通过状态的数值去表示文件是否被删除，即为一个简单的判断。同样的使用场景可见于新闻管理系统的稿件删除与否，对于重要的稿件资料，系统一般会设置二次判断功能，即初次删除为假删除，再次删除才为真删除，这其中的原理即通过 0 和 1 去实现。

在企业管理系统中，状态具有更加丰富的实现场景。下面简单介绍几种常用的场景。

(1) 登录状态：状态可用于判断用户是否处于登录或离线状态，并且系统可根据数据表中 0(假设 0 为登录状态)的数量统计当前登录人数，实现展现在线人数功能。

(2) 账号状态：可设置 0 和 1 代表正常账号或封禁账号的状态字段，在登录时检查账号状态是否正常再放行。

(3) 操作状态：在严格的系统中，还会使用字段记录用户的行为操作，当用户执行了某个敏感操作时，将字段置为 1，否则默认值为 0，这样可对一些敏感操作进行溯源处理，增强系统的安全性。

(4) 文件状态：如果企业系统是一个多组织的结构，则可以对文件列表中的普通文件添加状态 0，对共享事件添加状态 1，以此来判断当前存在的共享文件，还可使用 0 和 1 实现

文件正常状态和加密状态的判断。

(5) 事项状态：假设在高管的事件列表中存在相同量化等级的事项，如当天需要完成的重点事项有 3 个，那么系统应当有手动设置事项轻重缓急状态的功能，可在数据表中用数字 0、1、2 标记普通、重要和必要的状态，这样就有利于决策者做出更加准确的判断。

(6) 过滤状态：在实现搜索功能时，可根据目标在数据表中的状态作为附加搜索条件，如搜索身份为普通用户的账号，默认为搜索全部用户，但可增加单选框只选择正常状态的用户(状态为 0)或只选择封禁状态的用户(状态为 1)。

在合适的场景中正确使用 0 和 1 做判断，或者说使用布尔类型进行判断，可以直观地反映需求内部的逻辑关系，易于开发人员和后期维护人员理解，从而提高代码的可读性和可维护性；从后端直接返给前端的状态码也使前端程序员能够更加轻松地实现各种类型的条件判断，不需再进行额外的加工操作，减轻了前端程序员的负担；如场景(1)中登录人数的计算，也侧面说明了使用布尔类型可以更加方便地进行数据统计和分析，因此在项目的开发和数据库管理中，广泛地存在使用布尔类型实现功能判断的字段。

3.2.5 数据表的 id

在数据表中，通常会添加 id 对记录进行唯一标识，也就是主键。数据表中的 id 是一种自动增加的数字，在一般的查询场景中可通过 id 搜索表，也可以被用来修改和删除某个记录的索引，在复杂场景下可用以识别记录中的特殊值或链接(用 id 作为另一张表的外键使用)不同的表。

在创建数据表时，通常会将 id 设置为自增，也就是每次新增记录时，数据库会自动地为新记录分配一个在上条记录的 id 的基础上加 1 的 id，这有助于数据表中记录的连续性，也使排序和查询操作更加高效，例如以用户注册先后的顺序获取用户列表信息，就可以通过 id 的大小获取数据，当然也可以通过创建账号时间获取，但从检索速度上，通过简短无额外格式的 id 无疑比带有时间格式的速度更快。数据库自动分配给记录的 id 具有持久性，即 id 一旦分配给某条记录，即使该条记录被删除，id 也不会回收或重新分配给其他的记录，当这条记录被删除的时候，下条记录的 id 依旧是在被删除的记录的 id 的基础上进行自增，例如当条记录(同时为最后一条记录)的 id 为 23，那么上条记录的 id 为 22，如果把 id 为 23 的记录删除，并且新增一条记录，则表中的末尾两个 id 将变成 22、24。

3.3 从 0 设计一张用户数据表

任何系统都是为用户服务的，而用户数据表则是这一切的基础。在设计用户数据表之前，作为即将成为全栈工程师的读者需要明白设计字段和设计表之前的差异，表是根据需求模块进行设计的，而字段是根据模块中的具体功能设计的。在本节中，将从用户数据表的重要性谈起，并从设计表的角度介绍数据表和模块、字段和功能的关系，最后通过 3.2 节的设

计理论从 0 创建一张用户数据表。

3.3.1　用户模块

在一个系统中，用户模块往往是最重要的模块之一，这是因为需求的提供者是用户，而系统的使用者也是用户。用户模块是系统中用户管理和处理用户信息的部分，提供了用户注册、登录、编辑用户信息、权限管理等功能。用户数据表是用于存储用户信息的数据库表，在表中通常会包含用户的基础信息，如账号、密码、用户名、性别、年龄、联系方式等，用户模块通过对用户数据表的增、删、改、查，实现了对用户信息的管理和操作。

在前端的设计中，用户模块的不同功能通常处于不同的页面，下面对用户模块常见的功能进行介绍并描述其常见场景。

1. 注册与登录功能

用于用户的注册与登录，其场景通常是系统的默认首页，即输入域名后进入的第 1 个页面，用户在输入账号和密码后登入系统内部。在后端设计注册及实现逻辑时需要对账号与密码的格式进行限制，如账号为大于 6 位和小于 12 位的纯数字、密码需结合数字和大小写字母等，并对登录次数进行限制；在前端也需对用户登录时输入的数据进行校验，达到前后端的双重校验。值得一提的是，随着安全隐患的不断增加，目前的登录除了校验账号和密码外还会结合滑动拼图块、校验登录 IP 地址、短信验证等进行多重校验，一些等级保护较高的系统还会通过 CA 认证去确认用户的真实身份。多重校验可以防止暴力破解、字典攻击、彩虹表攻击等，确保用户信息的真实性和合法性。

2. 管理用户功能

管理用户功能是用户模块的主体，在系统中通常以单模块的形式出现，以列表的方式对使用系统的用户进行展示。在用户表格中，展示了用户的基础信息，如账号、用户名、性别、年龄、联系方式、头像、部门、职位、创建时间和更新时间等。对用户的主要操作通常包括审核站外的注册人员（与之相对的是在系统内部创建的用户账号）、编辑用户的基础信息、设定用户的权限、对用户账号进行冻结和解冻等。

管理用户的重点在于权限，以新闻管理系统为例，通常会以三级的用户权限进行分层，即最顶层为超级管理员，中间层为负责各个模块（部门）的管理员，其余的为普通用户权限。超级管理员可对除其之外的所有用户进行管理，负责各个部门的管理员又可对其部门内的用户（员工）进行管理。普通用户一般只具备发布文章、浏览系统和搜索系统内容、发表评论和收藏文章等权限；对于中层的管理员来讲，除了普通用户的权限外，还兼具审核用户评论、审核和管理文章、修改普通用户账号信息等权限；超级管理员则具有最高权限，可以管理整个系统的所有用户账号。管理员在面对大量的用户数据时，想要对特定用户进行处理并不是一件简单的事情，所以在表格（用户模块抑或是其他的模块）中都会有搜索功能，便于管理员进行日常维护。

在一些的 B2C（Business to Customer，企业与消费者之间的电子商务模式）系统中，除

了系统的用户列表之外还有客户列表，客户列表除了客户的基础信息外还可双击查看每个客户的详细购物信息，如购买产品型号、购买次数、购买产品类别、支付价格区间、支付类型、消费场景和时间等，管理员在日常维护客户信息外还可对恶意购买的“黄牛”进行封禁账号处理。对于商家来讲，客户的购物信息具有非常重要的战略意义，结合客户列表对购物数据进行分析，可以帮助商家了解大多数客户对产品的喜好度、对产品的使用评价，进而优化产品设计、提高产品的吸引力，并结合分析的数据进一步指导产品的研发方向和拓展业务范围，不断提高产品的竞争能力。

3. 记录功能

在 3.2.3 节关于功能的判断描述中，提到了记录用户的操作行为字段，对于用户模块来讲，记录用户的操作行为是不可或缺的。

用户是业务的执行者，即使从手工记录的阶段转变为无纸化办公阶段，但仍然可能出现操作异常的情况。在 ERP 系统中通常涉及大量的业务数据，如销售订单、采购订单、出入库订单等，普通用户在面对大量数据时可能会出现录入错误，如填写单号的时候多写了一个数字、产品的型号填写错误等。对于审计部门的管理员(可以简单地理解为财务)来讲，需要经常对系统内保存的数据进行审计(对账)，以保证数据的正确性，如果 ERP 系统中没有记录用户的操作日志，则管理员对发生数据错误的时间节点、导致数据出错的操作人员无从找起，对系统来讲是一个极大的隐患。对于权限较高的管理员，当受到攻击时更有可能泄露系统保存的敏感信息，如客户信息、财务信息、企业负责人的联系方式等，当出现异常操作情况时，管理员可根据操作日志及时封禁出现敏感操作的高权限账号。

操作日志可以增加系统的安全性和可维护性，方便系统管理员追溯系统的历史操作记录，满足系统管理员对用户操作的审计要求，避免系统的信息出现泄露和滥用。

3.3.2 用户表字段

综合 3.3.1 节的注册与登录功能、管理用户功能和记录功能，对各个功能的操作对象进行抽象处理，为了方便开发，部分字段采取默认长度，具体如下。

(1) 登录功能：账号、密码。

(2) 管理用户功能：姓名、性别、年龄、头像、联系方式、部门、职位、状态。

(3) 记录功能：创建账号时间、更新账号信息时间。

通过以上抽象出来的字段并结合 3.2 节中关于字段的命名、数据类型、约束、功能判断和 id，可得到如下字段及其属性。

(1) id：主键，类型为 int，长度默认为 11，不为空且自增。

(2) account：账号，类型为 int，长度默认为 11。

(3) password：密码，类型为 varchar，长度默认为 255(此处长度并不是指密码的长度可以为 255，这是因为在实际的场景中数据表保存的密码皆经过加密，加密后的长度由加密中间件决定，故设为 255 更为保险，不会出现长度过小问题)。

(4) name：用户名(昵称)，类型为 varchar，长度默认为 255。

(5) sex：性别，类型为 varchar，长度默认为 255。

(6) age：年龄，类型为 int，长度默认为 11。

(7) image_url：用于存储头像在服务器中的地址，类型为 varchar，长度默认为 255。

(8) email：邮箱，即联系方式，类型为 varchar，长度默认为 255。

(9) department：部门，类型为 varchar，长度默认为 255。

(10) position：职位，类型为 varchar，长度默认为 255。

(11) create_time：创建时间，类型为 datetime，长度默认为 0(数据库系统会依据数据类型的规范自动分配默认长度，即动态分配)。

(12) update_time：更新账号信息时间，类型和长度同 create_time。

(13) status：账号状态，类型为 int。

3.3.3　创建用户数据表

1. 使用命令行创建用户数据表

创建名为 gbms(General Background Management System，通用后台管理系统)的数据库，并在数据库中新建名为 users 的数据表，命令如下：

```
#创建名为 gbms 的数据库
create database gbms;

#进入 gbms 数据库
use gbms

#新建名为 users 的数据表,并创建用户表字段
#primary key 为主键,auto_increment 为自增
create table users (
  id int primary key auto_increment,
  account int,
  password varchar(255),
  name varchar(255),
  sex varchar(255),
  age int,
  image_url varchar(255),
  email varchar(255),
  department varchar(255),
  position varchar(255),
  create_time datetime,
  update_time datetime,
  status int
);
```

2. 使用 Navicat 创建用户数据表

以 2.3.2 节的操作步骤为例，首先使用小皮面板创建名为 gbms 的数据库，如图 3-1 所示。

图 3-1　创建 gbms 数据库

接着打开 Navicat for MySQL 的连接，输入连接名、用户名和密码进行连接，如图 3-2 所示。

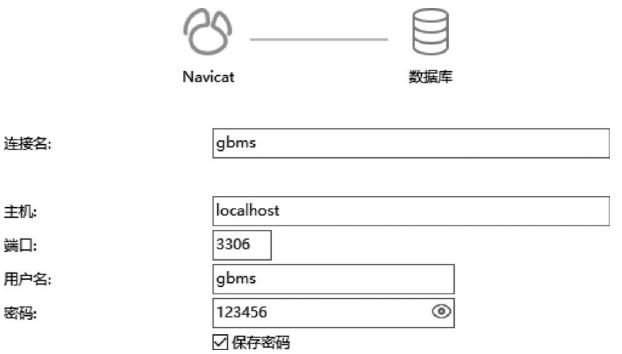

图 3-2　Navicat 连接数据库

在左边数据库列表中选择 gbms，并右击 gbms 中的表，单击“新建表”选项，进入建表页面并输入字段、数据类型、长度及其他选项。需要注意的是 id 为主键，不为空并且自增，如图 3-3 所示。

其余字段在类型和长度上遵循 3.3.2 节的设定，如图 3-4 所示。

最后单击左上角“保存”按钮，输入表名 users 并确认，至此 users 表就创建成功了，如图 3-5 所示。

图 3-3　设定 id 字段

名	类型	长度	小数点	不是 null	虚拟	键
id	int	11		☑	☐	1
account	int	11		☐	☐	
password	varchar	255		☐	☐	
name	varchar	255		☐	☐	
sex	varchar	255		☐	☐	
age	int	11		☐	☐	
image_url	varchar	255		☐	☐	
email	varchar	255		☐	☐	
department	varchar	255		☐	☐	
position	varchar	255		☐	☐	
create_time	datetime			☐	☐	
update_time	datetime			☐	☐	
status	int	11		☐	☐	

图 3-4　用户表字段

图 3-5　users 表