数据管理系统的历史

研究工具的发展历史很有价值。历史向我们揭示了人们是如何将一个想法变成适合自己使用的工具的。这通常是一个曲折的过程，充斥着不同的甚至截然相反的见解、激烈的竞争、以及一次又一次的试错。学习历史可以帮助我们理解工具的本质，避免我们重蹈前人的覆辙。

数据管理系统的发展历史已经超过了半个世纪，里面有很多精彩的故事和发人深思的观点。但是，要挖掘和重现这段历史并非易事，需要收集大量的客观史料。笔者希望未来能有机会去整理这一段历史，因为这件事意义非凡。但目前我们只能根据已知的信息粗略介绍一下这项技术的发展历程。

从文件系统到关系数据库

上世纪60年代以前，计算机世界里能提供通用数据管理功能的只有文件系统。文件系统不区分数据的种类。对任何数据，无论是文档、照片还是可执行程序，文件系统都将它们视作字符串，记录在一个个的文件中。系统提供简单的文件读写接口，允许应用程序访问字符串中任意位置的内容。此外，文件系统还提供文件级别的持久性和安全性保障，确保文件不会轻易丢失或遭到破坏。至于文件中的数据应该如何组织和管理，文件系统将其全权交给了应用程序。

文件系统简单而通用的功能确实给应用开发带来了明显的便利。然而，人们在实践过程中逐渐发现，文件系统承担的数据管理功能粒度过粗，程序开发人员仍然需要花费大量时间去构建文件内部的数据组织和管理功能，而且其中的很多功能对大多数应用而言就是一样的。这就萌发了在文件系统的基础上构建数据管理系统的想法 - 能不能将应用程序对细粒度数据管理的共同需求抽象出来，再用通用的数据管理系统去满足这种需求？为了实现这个想法，首先需要统一应用程序对数据的组织方式。也就是说，我们首先需要为数据管理系统确定一个数据模型，用于定义数据的组织结构。有了数据模型之后，我们才能在其上定义数据管理功能的具体接口，并确定数据管理系统内部的实现方式。

在上世纪60年代，应用软件通常更重视数据的第二大功用，即将数据作为事实的凭据，用于实现人、财、物的管理。因此，当时的数据更多是记录关于人、财、物的信息。基于这种用途，人们很自然会想到将数据组织成“图”的形式。图中的节点代表人和物这样的实体，而图中的边代表人与人或人与物之间的联系。有了这样的图，人们可以沿着图的结构顺藤摸瓜，快速定位自己想要的信息。例如，张三和张三的银行账户都可被表示为图中的节点，节点中存放各自的相关信息，比如张三的节点存放了他的姓名、身份证号码、地址，银行账户节点则存放账号和余额等信息。同时，两个节点用边连接，代表张三的银行账户是属于张三的。这样，一旦我们定位了张三，就能够通过其节点上的边跳转到他的账户节点，获取他的账户信息，而后还可以从账户节点跳转到开户行节点，等等。早期的数据管理系统使用的“网状模型”和“层次模型”都是基于图的数据模型。

随着数据模型的确立，几款成熟的数据管理系统于60年代相继问世了。其中的代表包括：由GE公司研发的基于网状模型的IDS（Integrated Data Store）和由IBM公司研发的基于层次模型的IMS（Information Management System）。二者都在商业上取得了成功，让开发人员认识到了数据管理系统带来的便利。越来越多的应用软件开始被构建在数据管理系统之上。然而，正当大家顺理成章接受“图”作为数据管理的标准模型时，来自IBM的一位工程师大胆地提出了质疑，并从此改变了数据管理系统的历史轨迹。这位工程师名叫Edgar Frank Codd。

Codd认为，“网状模型”和“层次模型”的设计对软件工程是不利的；虽然基于这两种模型的数据管理系统能够承担一定的数据管理功能，但数据管理系统和应用程序之间的耦合度过高，导致软件整体难以维护。我们在前文阐述过，一个好的系统一定是一个好的模块。在Codd看来，像IDS和IMS这样的系统并不是好的模块，它们与应用程序之间的耦合度太高，违背了信息隐藏的原则。Codd的看法是中肯的。IDS和IMS将数据访问的详细过程交给了应用程序去实现。也就是说，应用程序需要告诉数据管理系统，如何在图上进行跳转，从而一步一步

逼近想要的信息。可想而知，一旦数据的组织结构发生变化，原有的数据访问过程就失效了，应用程序就必须做相应的修正。软件越复杂，数据管理系统和应用程序的相互制约就越严重。

于是，Codd提出重新划分数据管理系统和应用程序之间的界限，将数据访问的过程尽可能交给数据管理系统去实现。在他看来，最好的情况是让应用程序只告诉数据管理系统自己想要什么，然后让数据管理系统自己决定如何获取数据。基于这种想法，Codd提出了关系模型。关系模型将数据组织成一张张的表，并且提供了一套结合了集合论和谓词逻辑的数据访问接口，让使用者可以清晰地表述自己的数据访问需求。（这一套访问接口后来演变成了SQL语言。）随着Codd对关系模型的竭力推广，关系模型逐渐得到学界和工业界的认可。70年代末，以Oracle为代表的关系数据库系统问世，逐渐取代了IDS和IMS，并走上了长盛不衰之路。

数据管理系统的多样化

关系数据库系统自一统江湖之后地位一直非常稳固。这得益于关系模型自身的优越性，但也与其他人为因素有关。自80年代起，Oracle成为了关系数据库的领跑者，并竭尽商业手段为自己构建了庞大的用户生态。整个商用数据管理系统的市场几乎完全被Oracle、IBM、Microsoft的关系数据库产品垄断。新的竞争者难以进入。同时，应用软件的生态变得越来越完善和成熟；SQL成为了数据访问的行业标准；围绕关系数据库的中间件和运维工具也越来越丰富；这些都进一步巩固了关系数据库的垄断地位。但这种垄断也限制了数据管理系统向多样化方向发展。80年代和90年代，科技人员陆续提出了若干数据管理系统的新想法，比如演绎数据库（Deductive Database）和面向对象数据库（Object Oriented Database）。但这些系统最终未能被市场接受。这些新系统并非缺乏优点，但它们实在难以和关系数据库的庞大用户生态和产品生态相抗衡。与此同时，各种新的功能被不断添加到关系数据库系统中，让系统变得越来越臃肿，复杂度到了无以复加的地步。

这样的局面直到21世纪之后才被逐渐打破。90年代末，互联网兴起。互联网公司通常都是小本起家，无力承担商用数据库的高额费用，因此大都选择MySQL这样的开源关系数据库产品。这也无形中推动了开源数据库的飞速发展。虽然起点低，但互联网应用的规模扩张是很快的。往往数个月时间，用户就可以遍布全世界。这样的扩张方式对关系数据库系统的性能扩展提出了极大挑战。互联网应用的开发者逐渐认识到，关系数据库并不是一把万能钥匙。过于丰富的功能反而变成了它的负担，限制了系统横向扩展能力。互联网带来的高额利润为互联网企业提供了充足的创新资本，利用在开源数据库上积累的研发经验，互联网企业开始谋求数据管理系统的革新。

2004年，Google推出了全新的数据管理系统Big Table，用来存放海量的互联网数据。两年后，Amazon推出了一款事务型数据管理系统Dynamo，用于承载大规模电子商务。这两款数据管理系统都选择了比关系数据库更简单的模型和功能，但提供强大的扩展能力，可以部署在由上千个节点组成的分布式系统上。 类似Big Table和Dynamo这样的系统最初只是互联网公司为自己的业务量身定做的。 但它们的优越性很快被广大的互联网企业所认识。对新型数据管理系统的需求因此开始迅速增长。几年后，各种新的开源数据管理系统像雨后春笋一样的出现了，它们被互联网应用广泛采纳，并逐渐向传统应用渗透。其中的部分数据管理系统被称为NoSQL数据库。言外之意是：传统关系数据库不再是数据管理的万灵药，每一个应用场景都有最适合它的数据管理系统。

从此，数据管理系统的形态开始朝多样化方向发展。除了关系数据库之外，我们现在能在市面上看到各种用于数据存储、数据管理和数据处理的系统和工具，包括文档数据库、图数据库、列式数据库、键值对存储、搜索引擎系统、时序数据库、流数据处理引擎、消息队列中间件、大数据分析平台等等。一个应用往往会将多个系统搭配使用，共同支撑起自己的业务。如前文所述，一个系统设计不可能在所有方面都做到极致，折衷是更理性的选择。如今，计算机的应用领域越来越广泛，应用对数据管理需求的差异也就越来越大。对不同的应用领域，我们不得不选择不同的折衷点。数据管理系统的多样化实属必然。

再次回到本书的写作初衷，对数据管理系统的学习不能再单纯以关系数据库系统为中心，而应该扩展到更广泛的系统。

上一页<< | **>>下一页**