图灵奖数据库大师Stonebraker师徒对数据库近20年发展与展望的2万字论文

图灵人工智能 2024年07月16日 00:01 北京

以下文章来源于云数据库技术 ，作者叶正盛

云数据库技术

云计算、数据库、大数据等技术分享与讨论

2024年6月，81岁的图灵奖数据库大师Michael Stonebraker（MIT）和他的学生Andrew Pavlo（CMU）联名在数据库顶级期刊SIGMOD发表了一篇名字奇怪的论文，对数据库近20年的发展做了总结与展望。

Michael Stonebraker

Andrew Pavlo

数据库近20年总结与展望“历史总是不断轮回”

两位作者都是数据库领域非常有影响力的人物。Stonebraker是图灵奖获得者，不老战神，也是PostgreSQL数据库前身Ingres的创始人；Andy在CMU任教，数据库界的网红，他在数据库优化领域有很多探索，他的Database of databases网站几乎收入了全球所有的数据库，并且经常带来很多数据库发展的思考与总结。

这篇文章表达了关系模型（RM）与SQL依然具备强壮的生命力，一直在吸收业界新的思想，包括文档数据库、图数据库、向量数据库等等，在系统架构方面看好OLAP领域的列式存储模型、云数据库，作者一直鄙视Hadoop架构，认为是历史的倒退，也完全不看好区块链数据库，最后也对AI大模型代替SQL进行评论，并表示当前还并不完备。

文章很多观点都是非常直接，2万字，比较长，适合慢慢品读。

由于论文涉及到几乎数据库所有的领域，很多内容不一定整理准确，还请谅解，更欢迎与我一起讨论。

以下内容来自论文的中文翻译与整理————————

What Goes Around Comes Around... And Around...

Michael Stonebraker
Massachusetts Institute of Technology
stonebraker@csail.mit.edu

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu

摘要

二十年前，我们中的一员（Stonebraker）共同撰写了一篇论文(What Goes Around and Comes Around)，评论了过去40年的数据建模研究和开发[188]。该论文表明，尽管有人试图替换它们，但关系模型（RM,Relational-Model）和SQL仍然是数据库管理系统（DBMS）的主流选择。相反，SQL吸收了这些替代方法中的最佳思想。

我们重新审视这个问题，并认为自2005年以来，这种演变一直在继续。再次有人反复尝试替换SQL或RM。但RM仍然是主导的数据模型，SQL已经扩展到捕捉来自其他人的好想法。因此，我们预计未来会有更多类似的情况，即SQL和关系型数据库管理系统（RDBMS）的持续发展。我们还讨论了DBMS的实现，并认为主要的进步在于RM系统，主要是由不断变化的硬件特性驱动的。

1 引言

2005年，作者之一（Stonebraker）参与撰写了红皮书的一章，标题为“What Goes Around and Comes Around”[188]。那篇论文检视了1960年代以来的主要数据建模运动：

- 层次结构（例如，IMS）：1960年代末和1970年代
- 网格结构（例如，CODASYL）：1970年代
- 关系型：1970年代和1980年代初
- 实体-关系：1970年代
- 扩展关系型：1980年代
- 语义：1970年代末和1980年代
- 面向对象：1980年代末和1990年代初
- 面向关系型：1980年代末和1990年代初
- 半结构化（例如，XML）：1990年代末和2000年代

我们的结论是，具有可扩展类型系统的关系模型（即，面向对象-关系型）已经统治了所有竞争者，市场上没有其他成功的系统。尽管2005年涵盖的许多非关系型DBMS今天仍然存在，但它们的供应商已经将它们归为遗留维护模式，没有人在它们上面构建新应用程序。这种持久性更多是数据“粘性”的证明，而不是这些系统的持久力量。换句话说，今天仍有许多IBM IMS数据库在运行，因为转换为使用现代DBMS既昂

贵又有风险。但是，没有一家初创公司会愿意选择在IMS上构建新应用程序。

自从我们2005年的调查以来，数据库领域发生了很多事情。在这段时间里，DBMS已经从它们在商业数据处理的根源扩展到几乎每种类型的数据。这导致了2010年代初的“大数据”时代，以及当前将机器学习（ML）与DBMS技术整合的趋势。

在本文中，我们分析了数据库中过去20年的数据模型和查询语言活动。我们将评论结构化为以下领域：

- (1) MapReduce系统
- (2) 键值存储
- (3) 文档数据库
- (4) 列族/宽列
- (5) 文本搜索引擎
- (6) 数组数据库
- (7) 向量数据库
- (8) 图形数据库

我们认为，大多数偏离SQL或RM(Relational-Model)的系统并没有主导DBMS格局，通常只服务于小众市场。许多最初以很大声势拒绝RM的系统（比如NoSQL）现在为RM数据库公开了一个类似SQL的接口。这样的系统现在正走向与RDBMS的融合。与此同时，SQL纳入了最好的查询语言思想，以扩大其对现代应用程序的支持并保持相关性。

尽管RM的基本原理没有太多变化，但在RM系统实现中发生了戏剧性的变化。本文的第二部分讨论了DBMS架构的进步，这些进步解决了现代应用程序和硬件的问题：(1) 列式系统，(2) 云数据库，(3) 数据湖/Lakehouses，(4) NewSQL系统，(5) 硬件加速器，以及 (6) 区块链数据库。其中一些是对DBMS实现的深刻变化，而另一些仅仅是基于错误前提的趋势。

我们对下一代DBMS的重要考虑因素的讨论作为结尾，附加上我们在科研和商业环境中对数据库未来希望的评论。

2 数据模型和查询语言

在我们的讨论中，我们将数据库中的数据模型和查询语言的研究和开发分为八个类别。

2.1 MapReduce 系统

Google在2003年构建了他们的MapReduce（MR）框架，作为处理其对互联网定期抓取的“关键解决方案”[122]。当时，Google在DBMS技术方面几乎没有专业知识，他们构建MR以满足他们的抓取需求。在数据库术语中，Map是一个用户定义的函数（UDF），执行计算或过滤，而Reduce是一个GROUP BY操作。大致上说，MR运行一个单一的查询：

```
SELECT map() FROM crawl_table GROUP BY reduce()
```

Google的MR方法并没有规定特定的数据模型或查询语言。相反，由在过程式MR程序中编写的Map和Reduce函数来解析数据文件的内容。

在2000年代末，其他公司对基于MR的系统非常感兴趣。Yahoo!（雅虎） 在2005年开发了一个名为

Hadoop的MR的开源版本[134]。它运行在一个分布式文件系统HDFS之上，这是Google文件系统的克隆[134]。几家初创公司成立来支持Hadoop的商业市场。我们将使用MR来指代Google的实现，Hadoop来指代开源版本，它们在功能上是相似的。

关于Hadoop与为OLAP工作负载设计的RDBMS的价值比较存在争议。这在2009年的一项研究中达到高潮，该研究表明数据仓库DBMS的性能超过了Hadoop[172]。这引发了Google和DBMS社区之间的辩论文章[123, 190]。Google认为，通过专业工程，MR系统将击败DBMS，用户不必在运行查询之前用模式加载数据。因此，MR更适合“一次性”任务，如文本处理和ETL操作。DBMS社区认为，MR由于其设计而存在性能问题，现有的并行DBMS已经解决了这些问题。此外，使用高级语言（SQL）在分区表上运行已被证明是一个好的编程模型[127]。

两篇论文中的许多讨论都集中在实现问题上（例如，索引，解析，查询处理，故障恢复）。从阅读这两篇论文可以得出一个合理的结论，即这两种系统都有其适用的地方。然而，技术世界的两个变化使这场辩论变得无关紧要。

第一个事件是Hadoop技术和服务市场在2010年代崩溃了。许多企业在Hadoop集群上花费了大量的资金，结果发现对这个功能几乎没有兴趣。开发者发现很难将他们的应用程序强行塞入受限的MR/Hadoop范式中。有相当大的努力在Hadoop之上提供SQL和RM接口，最值得注意的是Meta的Hive[30, 197]。

发生在CACM文章八个月后的下一个事件是Google宣布他们正在将他们的抓取处理从MR转移到BigTable[164]。原因是Google需要实时交互式更新其抓取数据库，但MR是一个批处理系统。Google最终在2014年宣布MR在他们的技术栈中没有位置，并将其淘汰[194]。

第一个事件让三家领先的Hadoop供应商（Cloudera, Hortonworks, MapR）没有可行的产品可卖。Cloudera将Hadoop重新定义为整个堆栈（Application、Hadoop、HDFS）。进一步的手法是，Cloudera在HDFS之上构建了一个RDBMS，Impala[150]，但没有使用Hadoop。他们意识到Hadoop在SQL DBMS的内部接口中没有位置，并且他们用直接构建在HDFS上的软件将其从他们的堆栈中移除出去。同样，MapR在HDFS上直接构建了Drill[22]，Meta创建了Presto[185]来取代Hive。

讨论：MR的缺陷如此之大，以至于尽管开发者社区的采用和热情，它也无法得救。Hadoop大约在十年前就死了，留下了企业中的HDFS集群遗产和一系列产品公司致力于从它们那里赚钱。目前，HDFS已经失去了它的光彩，因为企业意识到有更好的分布式存储替代品[124]。与此同时，分布式RDBMS正在蓬勃发展，特别是在云中。

MR系统实现的一些方面与分布式RDBMS的可扩展性，弹性和容错性有关。MR还带来了共享磁盘架构的复兴，随后产生了开放源代码文件格式和数据湖（见第3.3节）。Hadoop的限制为其他数据处理平台打开了大门，即Spark[201]和Flink[109]。两个系统最初都是作为MR的更好实现，具有过程式API，但此后已经增加了对SQL的支持[105]。

2.2 键/值存储（Key/Value Stores）

键/值（KV）数据模型是可能的最简单模型。它表示以下的二元关系：(key, value)

KV DBMS将数据集合表示为一个将键映射到值的关联数组。值通常是一个未标记的字节数组（如：一个blob），DBMS不知道其内容。由应用程序来维护模式并解析值到其相应的部分。大多数KV DBMS只提供对单个值的get/set/delete操作。

在2000年代，几家新兴的互联网公司为专注于特定应用的场景构建了各自的无共享（shared-nothing）、

分布式KV存储系统。如缓存会话数据。对于缓存，Memcached[131]是这种方法最著名的例子。Redis[67]自诩为Memcached替代品，提供了一个更强大的查询API，并支持检查点。对于持久化数据，亚马逊在2007年创建了Dynamo KV存储[125]。这些系统提供了比RDBMS更高和更可预测的性能，但功能更为有限。

第二个KV DBMS类别是设计为在与高级应用程序相同的地址空间中运行的嵌入式存储引擎。最早的独立嵌入式KV DBMS之一是90年代初的BerkeleyDB[170]。近期值得注意的包括Google的LevelDB[37]，Meta后来将其派生为RocksDB[68]。

讨论：键/值存储为开发者提供了一种快速的“开箱即用”的方式来存储数据，相比之下，配置RDBMS中的表则需要更多的工作。当然，在一个需要比简单的二元关系更复杂的应用程序中使用KV存储是危险的。如果应用程序需要记录多个字段，那么KV存储可能不是一个好方案。不仅应用程序必须解析记录字段，而且没有二级索引可以按值检索其他字段。同样，开发者必须在他们的应用程序中实现联接或多次获取操作。

为了解决这些问题，一些系统起初是一个KV存储，然后演变成一个更丰富的记录存储。这样的系统用半结构化的值替换了不透明的值（类似blob），例如JSON文档。这种转变的例子是亚马逊的DynamoDB[129]和Aerospike[9]。重新设计一个KV存储以支持复杂数据模型并不简单，而RDBMS可以轻松模拟KV存储而无需任何更改。如果应用程序需要一个嵌入式DBMS，今天有全功能的选项可用，包括SQLite[71]和DuckDB[180]。因此，即使是简单的应用程序，RDBMS也可能是更好的选择，因为如果应用程序的复杂性增加，RDBMS可以提供更好的扩展性。

过去20年的一种新架构趋势是使用嵌入式KV存储作为全功能DBMS的底层存储引擎。在此之前，构建一个新的DBMS需要工程师构建一个定制的存储引擎，该引擎内置集成在DBMS中。MySQL是第一个公开API的DBMS，允许开发者替换其默认的KV存储引擎（插件式存储引擎架构）。这个API使得Meta能够构建RocksDB来替换其大量MySQL数据库的InnoDB。类似地，MongoDB在2014年放弃了他们命运多舛的MMAP存储管理器，转而使用WiredTiger的KV存储[120, 138]。使用现有的KV存储允许开发者用更少的时间编写新的DBMS。

2.3 文档数据库（Document）

文档数据模型将数据库表示为记录对象的集合。每个文档包含字段/值对的层次结构，每个字段都由名称标识，字段的值可以是标量类型、值数组或另一个文档。以下JSON示例是一个包含嵌套的采购订单记录列表的客户文档，以及它们相应的订单项。

```
{
  "name": "First Last",
  "orders": [
    { "id": 123, "items": [...] },
    { "id": 456, "items": [...] }
  ]
}
```

文档数据模型几十年来一直是一个活跃的研究领域。这促成了像SGML[117]和XML[118]这样的数据格式

的出现。尽管在1990年代末XML数据库备受关注，我们在2005年正确预测它们不会取代RDBMS[188]。JSON自那以后已经超越了XML，成为基于Web的应用程序数据交换的标准。随着JavaScript的普及和随之而来的JSON普及，在2000年代诞生了几家公司，创建了本地存储JSON的面向文档的系统。

由于OLTP RDBMS在2000年代无法扩展，出现了几十个文档DBMS，它们使用NoSQL的口号进行市场推广[110]。这些系统的两个市场宣传与开发者产生了共鸣。首先，SQL和连接操作很慢，应该使用更“快速”的底层、单记录处理的接口。其次，现代应用程序不需要ACID事务，因此DBMS应该只提供较弱的事务概念（即NoSQL比较推崇的BASE[179]）。

由于这两个推动，NoSQL开始代表一个存储记录或文档为JSON的DBMS，支持低级API，并且支持较弱或不存在的事务。有几十种这样的系统，其中MongoDB[41]是最受欢迎的。

讨论：文档DBMS本质上与1980年代的面向对象DBMS和1990年代末的XML DBMS相同。文档DBMS的支持者与他们的OO/XML前辈提出了相同的论点：以文档形式存储数据消除了应用程序OO代码与数据交互的方式和关系数据库存储它们之间的不匹配。他们还声称，将记录非规范化为嵌套结构对性能更好，因为它消除了分派多个查询以检索与给定对象相关的数据的需要（即，ORM中的“N+1问题”）。非规范化/预连接的问题是一个古老的话题，可以追溯到1970年代[116]：(1)如果联接不是一对多，那么就会有重复数据，(2)预连接并不一定比联接更快，(3)没有数据独立性。

尽管他们强烈抨击SQL很糟糕，但到2010年代末，几乎所有NoSQL DBMS都添加了SQL接口。直接的例子包括DynamoDB PartiQL[56]、Cassandra CQL[15]、Aerospike AQL[9]和Couchbase SQL++[72]。最后一个坚持的是MongoDB，但他们在2021年为他们的Atlas服务添加了SQL[42]。其支持SQL标准的DDL和DML操作，NoSQL供应商声称他们支持自己的专有查询语言，该语言源自或受SQL启发。对于大多数应用程序来说，这些区别是没有意义的。SQL和NoSQL衍生语言之间的任何语言差异主要是由于JSON扩展和维护操作。

许多剩余的NoSQL DBMS还添加了强一致性（ACID）事务（见第3.4节）。因此，NoSQL的信息已经从“Do not use SQL – it is too slow!”转变为“Not Only SQL”（SQL有时还是不错的）。

向NoSQL DBMS添加SQL和ACID降低了它们与RDBMS的功能差距。它们之间的主要区别似乎是JSON支持以及NoSQL供应商允许“schema later/Schemaless”数据库。但是，SQL标准在2016年添加了JSON数据类型和操作[165, 178]。并且随着RDBMS继续改善开发人员的“前五分钟”体验，我们相信这两种系统很快就会变得实际上无法区分。

高级语言几乎普遍优于逐条记录操作的方法，因为它们可以更少的代码并提供更大的数据独立性。尽管我们承认最初的SQL优化器很慢且效果不佳，但在过去50年里它们已经有了很大的改进。但优化器仍然是构建DBMS中最困难的部分。我们怀疑这种工程负担是NoSQL系统最初选择不支持SQL的一个因素。

2.4 列族数据库（Column-Family）

存在另一类使用称为列族（又称宽列）的数据模型的NoSQL系统。尽管它的名字，列族并不是一个列式数据模型。相反，它是对文档数据模型的简化，只支持一级嵌套而不是任意嵌套；它和关系型有点像，但每条记录可以有可选属性，单元格可以包含值数组。以下示例显示了从用户标识符键到包含每个用户不同配置文件信息的JSON文档的映射：

```
User1000 → { "name": "Alice",
```

```
                "accounts": [123, 456], "email": "xxx@xxx.edu" }
User1001 → { "name": "Bob",
                "email": ["yyy@yyy.org", "zzz@zzz.com"] }
```

第一个列族模型DBMS是Google在2004年推出的BigTable[111]。与采用SQL和出现的列式存储不同，Google使用这种数据模型和过程性客户端API。其他系统采用了列族模型，试图复制Google的定制实现。最值得注意的是Cassandra[14]和HBase[28]。它们也复制了BigTable的限制，包括缺乏联接和二级索引。

讨论：我们在第2.3节中关于文档模型的所有评论在这里也适用。在2010年代初期，谷歌在BigTable之上构建了RDBMS，包括MegaStore[99]和Spanner的第一个版本。从那以后，谷歌重写了Spanner，去除了BigTable的残余部分[98]，现在它是许多内部应用的主要数据库。一些NoSQL DBMS放弃了它们的专有API，转而支持SQL，但仍然保留了它们的非关系型架构。Cassandra用一种名为CQL[15]的类似SQL的语言替换了他们的Thrift-API，HBase现在推荐使用Phoenix SQL前端[57]。谷歌仍然提供BigTable作为云服务，但列族模型是一个独特的例外，具有与NoSQL DBMS相同的缺点。

2.5 文本搜索引擎

文本搜索引擎已经存在了很长时间，始于1960年代的开创性SMART系统[184]。SMART开创了信息检索和向量空间模型，在现代搜索引擎中几乎普遍使用，通过将文档标记化为“bag of words”，然后构建这些标记的全文索引（也称为倒排索引）来支持对它们内容的查询。该系统还意识到了噪音词（例如，“the”，“a”）、同义词（例如，“The Big Apple”是“New York City”的同义词）、关键词和距离（例如，“drought”经常出现在“climate change”附近）。

当今领先的文本搜索系统包括Elasticsearch[23]和Solr[70]，它们都使用Lucene[38]作为内部的搜索库。这些系统为存储和索引文本数据提供了良好的支持，但只提供有限的事务能力。这种限制意味着DBMS必须通过从头重建文档索引来从数据损坏中恢复，这导致显著的停机时间。

所有领先的RDBMS都支持全文搜索索引，包括Oracle[52]、Microsoft SQL Server[52]、MySQL[43]和PostgreSQL[62]。它们的搜索功能最近有所改进，通常与上述专用系统相当。它们还具有内置的事务支持的优势。但是，它们将搜索操作集成到SQL中通常很笨拙，并且在不同的DBMS之间有所不同。

讨论：文本数据本质上是无结构的，这意味着没有数据模型。相反，DBMS试图从文本中提取结构（即，元数据、索引），以避免“大海捞针”的顺序搜索。

2.6 数组数据库（Array Databases）

注：时序和空间数据库也可以理解为是数组数据库的一种场景

在许多计算领域，数组是显而易见的数据表示形式。我们使用术语“数组”来指代所有变体[182]：向量（一维 - 见第2.7节）、矩阵（二维）和张量（三维或更多维）。例如，针对地理区域的科学调查通常将数据表示为多维数组，存储使用基于位置/时间坐标的传感器测量

```
值：(纬度,经度,时间,[值向量])(纬度,经度,时间,[值向量])
```

其他一些数据集看起来像这样，包括基因组测序和计算流体动力学。数组也是大多数机器学习数据集的核心。

尽管基于数组的编程语言自20世纪60年代以来就已经存在（APL[142]），最初的数组数据库管理系统（DBMS）的工作始于20世纪80年代。PICDMS被认为是第一个使用数组数据模型的DBMS实现[114]。今天仍在开发的最古老的两个数组DBMS是Rasdaman[66, 103]和kdb+[34]。较新的数组DBMS包括SciDB[54, 191]和TileDB[76]。HDF5[29]和NetCDF[46]是科学数据的流行数组文件格式。

存储和查询现实世界数组数据集存在几个系统挑战。首先是数组数据并不总是规则的；例如，地理空间数据通常被分割成不规则形状。应用程序可以通过描述此映射的元数据将这样的网格映射到整数坐标[166]。因此，大多数应用程序在单个数据库中同时维护数组和非数组数据。

与基于行或列的DBMS不同，在任意维度上查询数组数据提出了独特的挑战。困难来自于将多维数组数据存储在像磁盘这样的线性物理存储介质上。为了克服这些挑战，数组DBMS必须采用索引和数据结构来支持数组维度上的高效遍历。

讨论：数组DBMS是一个细分市场，只有在特定的垂直领域（我们在下一个部分讨论向量DBMS）看到了采用。例如，它们在基因组学领域有相当的吸引力。HDF5在卫星图像和其他网格科学数据中很受欢迎。但商业应用程序很少使用专用的数组DBMS，这对于任何产品的存活来说是必要的。没有主要的云服务提供商提供托管的数组DBMS服务，这意味着它们没有看到一个可观的市场。

数组DBMS供应商一直面临的挑战是，SQL包括对有序数组作为一等数据类型（尽管这违反了原始RM提议[115]）的支持。第一个提出将有序位图扩展到基于集合的无序RM的提议是在1993年[155]。早期的例子是Illustra的时间（一维）数据插件[31]。SQL:1999引入了对单维、固定长度数组数据类型的有限支持。SQL:2003扩展到支持没有预定义最大基数的嵌套数组。后来的加入者包括Oracle GeoRaster[4]和Teradata[73]。数据立方体是特殊用途的数组[135]，但由于它们的灵活性更好和工程成本更低，列式RDBMS已经取代了它们用于OLAP工作负载[113]。

最近，SQL:2023标准包括了对真正的多维数组（SQL/MDA）的支持，这在很大程度上受到了Rasdaman的RQL[166]的启发。这个更新允许SQL使用基于整数的坐标来表示具有任意维度的数组。实际上，这允许数据立方体存在于SQL框架中，但列式DBMS现在主导了这个市场。

2.7 向量数据库（Vector）

类似于列族模型是文档模型的简化，向量数据模型简化了数组数据模型为一维模型。鉴于向量DBMS目前从开发者和投资者那里吸引了最多的关注（类似于以前NoSQL的狂热），有必要单独讨论它们。引起这种兴趣的原因是开发者使用它们来存储由AI工具生成的单维度embedding。这些工具使用学习到的模型将记录的数据（例如，文本、图像）转换为表示其潜在语义的向量。例如，可以使用Google BERT将每篇维基百科文章转换为embedding，并将它们与额外的文章元数据一起存储在向量数据库中：

```
(标题,日期,作者,[嵌入向量])(标题,日期,作者,[嵌入向量])
```

这些嵌入向量的尺寸范围从简单变换器的几百维到高端模型的几千维；这些尺寸显然会随着更复杂的模型的发展而增长。

关键的区别在于向量和数组数据库的查询模式。向量用于相似性搜索，这些搜索找到的记录向量在高维空间中与给定输入向量距离最短。输入向量是使用相同的变换器生成的另一个embedding。与数组数据库不同，应用程序不使用向量数据库来搜索向量中的偏移，也不提取多个向量之间的切片。相反，主要用例是搜索相似性。

为了避免使用蛮力扫描来寻找最相似的记录，向量数据库构建索引以加速近似最近邻（ANN）搜索。应用程序发出查询，这些查询具有关于嵌入索引和非嵌入属性（即元数据）的谓词。然后，数据库管理系统选择是否在向量搜索之前（预过滤）或之后（后过滤）使用非嵌入谓词对记录进行过滤。

在这个新兴类别中有许多新的数据库管理系统，Pinecone [58]、Milvus [40] 和 Weaviate [84] 是领先的系统。文本搜索引擎，包括 Elasticsearch [23]、Solr [70] 和 Vespa [79]，扩展了它们的 API 以支持向量搜索。其他数据库管理系统重新品牌自己为向量数据库以加入这一潮流，例如 Kdb+ [34]。

向量数据库的一个引人注目的特点是，它们比关系数据库管理系统更好地与 AI 工具（例如 ChatGPT [16]、LangChain [36]）集成。这些系统在插入时本地支持将记录数据转换为embedding，使用这些工具，然后使用相同的转换将查询的输入参数转换为嵌入以执行 ANN 搜索；其他数据库管理系统要求应用程序在数据库外部执行这些转换。

讨论：与需要定制存储引擎和执行引擎以支持多维数据有效操作的数组数据库不同，向量数据库本质上是具有专门 ANN 索引的数据库管理系统。这些索引是一个特性，而不是一个新系统架构的基础。

在 2022 年底 ChatGPT 成为“主流”之后，不到一年，几个关系数据库管理系统就增加了自己的向量搜索扩展。在 2023 年，许多主要的关系数据库管理系统增加了向量索引，包括 Oracle [7]、SingleStore [137]、Rockset [8] 和 Clickhouse [157]。与此相比，RDBMS 中的 JSON 支持，NoSQL 系统如 MongoDB 和 CouchDB 在 2000 年代末变得流行，而 RDBMS 花了几年时间才增加对它的支持。

向量索引快速扩散有两个可能的解释。第一，通过嵌入进行相似性搜索是一个如此引人注目的用例，以至于每个数据库管理系统供应商都急忙推出了他们的版本并立即宣布。第二，引入一个新的索引数据结构的工程工作量足够小，以至于数据库管理系统供应商增加向量搜索并没有花费太多工作。他们中的大多数没有从头开始编写他们的向量索引，而是集成了一个开源库（例如，pgVector [145]、DiskANN [19]、FAISS [24]）。

我们预计，向量DBMS将经历与文档DBMS相同的演变过程，通过增加功能变得更像关系型数据库（例如，SQL、事务、可扩展性）。同时，现有的关系型数据库将在其已经很长的功能列表中添加向量索引，并继续关注下一个新兴趋势。

2.8 图数据库（Graph Databases）

在过去十年中，图数据库在学术界和工业界引起了很多兴趣[183]。许多应用程序使用知识图谱来模拟半结构化信息。社交媒体应用程序本质上包含面向图的关系（“点赞”，“朋友”）。关系设计工具为用户提供了他们数据库的实体关系（ER）模型。ER 图是一个图；因此，这个范式有明确的用例。

表示图的两种最普遍的方法是（1）资源描述框架（RDF）和（2）属性图[126]。使用属性图，数据库管理系统维护一个有向多图结构，支持节点和边缘的键/值标签。RDF 数据库（也称为 triplestores）只模拟一个有标签边的有向图。由于属性图更常见，并且是 RDF 的超集，我们将只讨论它们。我们考虑图数据库管理系统的两个用例，并讨论将限制它们被采用的问题。

第一类系统是用于操作性/OLTP 工作负载的：例如，应用程序通过以事务方式更新单个记录，在数据库中添加一个好友链接。Neo4j [44] 是最受欢迎的 OLTP 应用程序的图数据库管理系统。它使用指针支持边缘（就像 CODASYL 一样），但它不将节点与其“父”或“后代”聚集在一起。这种架构有利于遍历长的边缘链，因为它将进行指针追逐，而关系数据库管理系统必须通过联接来完成这一操作。但它们的潜在市场成功取决于是否有足够的“长链”场景，值得放弃关系数据库管理系统。

第二个用例是分析，它寻求从图中派生信息。一个例子是找出哪个用户在30岁以下有最多好友。值得注意的，如 Tigergraph [74] 和 JanusGraph [32]，专注于图数据库管理系统的查询语言和存储。其他系统，如 Giraph [26] 和 Turi [78]（前身为 Graphlab [27]），提供了一个计算框架，以支持面向图的程序的并行执行，通常由用户编写。

与关系分析中的查询不同，图分析查询包含如最短路径、cut set、clique determination等操作。算法选择和数据展示将决定DBMS的性能。这需要一个计算结构，允许开发者使用隐藏底层系统拓扑的抽象来编写自己的算法。然而，先前的研究表明，由于通信成本，分布式算法很少能胜过单节点实现[160]。一个更好的策略是将图压缩成一个空间高效的数据结构，使其适合单个节点的内存，然后针对这个数据结构运行查询。除了最大的图数据库外，所有图数据库可能最好都以这种方式处理。

讨论：无论图数据库管理系统针对 OLTP 还是 OLAP 工作负载，这些系统必须克服的关键挑战是，可以模拟图作为表的集合：

- 节点 (node_id, node_data)
- 边缘 (node_id_1, node_id_2, edge_data)

这意味着关系数据库管理系统始终是支持图的一个选项。但原生SQL 表达力不足以满足图查询，因此需要多个客户端-服务器往返来执行遍历操作。

一些关系数据库管理系统，包括 MsSQL [3] 和 Oracle [50]，提供了内置的 SQL 扩展，使存储和查询图数据更加容易。其他数据库管理系统使用关系之上的转换层来支持面向图的 API。Amazon Neptune [45] 是 Aurora MySQL 上的面向图的表层。Apache AGE 在 PostgreSQL [10] 之上提供了一个 OpenCypher 接口。

最近，SQL:2023 引入了属性图查询（SQL/PGQ），用于在关系数据库管理系统中定义和遍历图[196]。该语法建立在现有语言（例如，Neo4j 的 Cypher [49]、Oracle 的 PGQL [51] 和 TigerGraph 的 GSQL [75]）之上，并共享了新兴的 GQL 标准[126]的方面。因此，SQL/PGQ 进一步缩小了关系数据库管理系统和原生图数据库管理系统之间的功能差异。

问题是，图数据库管理系统供应商能否使他们的专业系统足够快，以克服上述缺点。已经有几项性能研究表明，关系数据库管理系统上的图模拟优于图数据库管理系统[130, 143]。最近的工作表明，DuckDB 中的 SQL/PGQ 比领先的图数据库管理系统性能高出多达 10 倍[196]。随着最坏情况优化连接[132, 168]和因子分解执行算法[100]在关系数据库管理系统中的图查询的进一步改进，这一趋势将继续。

2.9 Summary

从上述部分可以合理地得出结论，非 SQL、非关系型系统要么是小众市场，要么正在迅速成为 SQL/RM 系统。具体来说：

- MapReduce 系统：它们在几年前就已经消亡，现在最多是遗留技术。
- 键值存储：许多已经成熟为 RM 系统，或者只用于特定问题。这些通常可以被现代高性能 RDBMS

相等或超越。

- 文档数据库：这些 NoSQL 系统正与 RDBMS 相撞。这两种系统之间的区别随着时间的推移而减少，将来应该几乎无法区分。
- 列族系统：这些仍然是小众市场。如果没有 Google，本文就不会讨论这个类别。
- 文本搜索引擎：这些系统在多个存储架构中用于文本字段。如果 RDBMS 对搜索有更好的支持，这些就不必是单独的产品。
- 数组数据库：科学应用将继续忽略 RDBMS，而倾向于专门的数组系统。尽管有新的 SQL/MDA 增强功能，RDBMS 仍然不能有效地存储和分析数组，它们可能会变得更重要。
- 向量数据库：它们是单一用途的 DBMS，具有加速最近邻搜索的索引。RM DBMS 应该很快提供对这些数据结构和搜索方法的原生支持，使用它们可扩展的类型系统，这将使这种专门的数据库变得不必要。
- 图数据库：OLTP 图应用将主要由 RDBMS 服务。此外，分析图应用有独特的需求，最好在主内存中使用专门的数据结构来完成。RDBMS 将提供基于 SQL 或通过扩展的图中心 API。我们预计专门的图 DBMS 不会是一个大市场。
- 除了上述内容，还有一些提议将以前的数据模型重新品牌为某种新颖的东西。例如，图关系[158]与语义数据模型[202]相同。同样，文档关系是带有外键的文档模型[199]。其他人在 RDBMS 上提供了一个非 SQL 的外观（例如，PRQL[64]，Malloy[39]）。尽管这些语言处理了 SQL 的一些缺点，但它们不足以克服其根深蒂固的用户群和生态系统。

3 系统架构

在过去二十年中，DBMS架构出现了重大的新思想，这些思想反映了应用和硬件特性的变化。这些想法从非常好的到值得怀疑的都有，我们将依次讨论它们。

3.1 列式系统

为了理解列式DBMS的吸引力，我们需要解释数据仓库（OLAP）市场的起源。从20世纪90年代中期开始，企业开始收集他们面向客户的（通常是销售）数据。实体零售商（例如，沃尔玛）在构建历史销售数据库方面处于前列。这些公司通常发现，销售数据仓库在六个月内通过更好的库存订购和轮换决策就能收回成本。这种面向客户的数据库现在在企业中无处不在。

数据仓库应用有一些与OLTP工作负载不同的共同属性：

1. 它们本质上是历史的（即，它们定期加载后就是只读的）。
2. 只要他们负担得起存储，组织就会保留一切 - 想想从TB到PB。
3. 查询通常只访问表中的一小部分属性，并且是即席性(ad-hoc)的。

Ralph Kimball是数据仓库的星型模式数据建模的早期支持者[148, 149]。这个想法是构建一个包含项目级交易数据的事实表。经典的例子是一个事实表，其中包含零售企业中每次购买的项目记录。然后，围绕事实表构建维度表，包含从事实表中提取的共同信息以节省空间。同样，在零售环境中，这些维度表将包括有关客户、产品、商店和时间的信息。

通过按列而不是按行组织DBMS的存储有多个好处[87]。首先，压缩列式数据比基于行的数据更有效，因为数据块中通常有单一的值类型，经常有许多重复的字节。其次，Volcano风格的引擎每行执行一次操作符。相比之下，列式引擎有一个内循环，使用向量化指令处理整个列[106, 147]。最后，行存储每个记录都有一个大的头部（例如，20字节）来跟踪空值和版本元数据，而列存储每个记录的存储开销很小。

讨论：在过去二十年中，所有在数据仓库市场活跃的供应商都已经将他们的产品从行存储转换为列存储。这种转变带来了DBMS设计上的显著变化。此外，在过去二十年中，有几家新的供应商进入了市场，提供了列存储产品，例如亚马逊的Redshift[94]和谷歌的BigQuery[162]，以及独立公司的提供（例如，Snowflake[121]）。

总结来说，列存储是新的DBMS实现，具有专门的优化器、执行器和存储格式。它们已经接管了数据仓库市场，因为它们具有卓越的性能。

3.2 云数据库

2000年代末云平台的兴起也极大地影响了DBMS的实现（和销售模式）。最初的云DBMS提供将本地系统重新打包到带有直接附加存储的托管虚拟机中。但是在过去二十年中，网络带宽的增长速度远远超过了磁盘带宽，使得网络附加存储（NAS）作为一种替代附加存储的方案变得具有吸引力。这导致了人们对云环境下的DBMS架构进行了深刻的反思。

所有主要的云供应商都通过对象存储提供NAS（例如，亚马逊S3），并带有一些DBMS功能（例如，复制、过滤）。除了与直接附加存储相比具有更好的经济性外，对象存储还有几个优势，可以弥补增加的网络链接的成本。首先，由于计算节点与存储节点分离，系统可以提供每个查询的弹性；DBMS可以动态添加新的计算节点而无需重新 reshuffler 数据。它还允许DBMS为其存储节点使用与计算节点不同的硬件。其次，如果DBMS未充分利用，系统可以将计算节点重新分配给其他任务。另一方面，在shared-nothing DBMS中，节点必须始终在线以处理传入的查询请求。最后，将计算下推到存储节点是可行的（通常也是有利的）。这种执行策略被称为“将查询下推到数据”，和“将数据拉到查询”是相对的，这在DBMS中被很好地理解。

通常，前两个想法被称为“serverless computing”，由Snowflake[121]为云原生DBMS引入。其他供应商已经或正在将其云产品转移到 serverless 环境中。有效利用这种模型需要一个托管的多节点环境，在该环境中，多个DBMS客户被分组到具有多租户执行方案的相同节点上。

讨论：云数据库的出现是“历史轮回”的另一个例子。多节点共享磁盘DBMS是一个古老的想法，历史上往往效果不佳。然而，随着技术变革（更快的网络）和向云的迁移，它又重新流行起来。此外，在20世纪70年代，当计算机庞大且昂贵时，分时服务很受欢迎。云平台是大型分时服务，所以这个概念在几十年后又回来了。由于企业正在将一切可能的东西迁移到云端，我们预计这种共享磁盘将主导DBMS架构。因此，我们不认为未来会出现shared-nothing架构。

云计算对DBMS产生了深远的影响，导致它们被彻底重新架构。计算从本地迁移到云端为企业提供了一个千载难逢的机会，可以重构代码库并消除过去糟糕的技术决策。云环境还为供应商提供了一些在本地部署中不可能实现的优点。他们可以为所有客户跟踪趋势：他们可以监控意外行为、性能下降和使用模式。此外，他们可以在不中断服务的情况下推送增量更新和代码补丁。
从商业角度来看，开源DBMS面临着变得过于流行并被主要云提供商货币化的危险。亚马逊与MongoDB[153]和Elasticsearch[101]等ISV之间的公开争执就是显著的例子。

3.3 数据湖 / Lakehouses

云平台促进的另一个趋势是从单一的、专用的数据仓库转向由对象存储支持的数据湖。使用传统的数据仓库，组织将数据加载到DBMS中，系统将数据存储在具有专有格式的存储引擎中。供应商将他们的DBMS视为组织中与数据相关的所有事物的“守护者”。然而，这在过去的十年中，尤其是在科技公司中，并不是许多组织的模型。

采用数据湖架构，应用程序将文件上传到分布式对象存储，绕过了通过DBMS的传统途径[167]。然后，用户使用累积的文件执行查询和处理管道，使用lakehouse（湖仓一体，数据仓库和数据湖的混合词）执行引擎[93]。这些lakehouse系统提供了统一的基础设施，支持SQL和非SQL工作负载。这一点至关重要，因为过去十年表明，数据科学家和机器学习从业者通常使用基于Python的notebook，使用Pandas的DataFrame API[159]来访问数据，而不是SQL。几个项目利用DBMS方法来优化DataFrame处理，包括Dask[181]、Polars[61]、Modin[177]和Bodo[198]。

与使用DBMS特定的专有文件格式或低效的基于文本的文件（例如，CSV，JSON）不同，应用程序使用开源的、磁盘上的文件格式将数据写入数据湖[203]。两个最受欢迎的格式是Twitter/Cloudera的Parquet[55]和Meta的ORC[53, 140]。它们都借鉴了早期列式存储研究的技术，如PAX[90]、压缩[87]和嵌套数据（JSON）分解[121, 161]。Apache Arrow[11]是一个类似的二进制格式，用于在系统之间交换内存中的数据。用于读取/写入这些格式的开源库允许不同的应用程序创建数据文件，然后由其他系统解析和使用，从而增强了服务和业务单位之间的数据共享。

讨论：数据湖是2010年代早期"大数据"运动的继承者，部分由MR系统（第2.1节）和列存储（第3.1节）的普及推动。乍一看，数据湖对于组织来说似乎是一个糟糕的主意：允许任何应用程序将任意文件写入中央存储库，而没有任何治理，这会导致完整性、发现和版本控制问题[167]。湖仓为这些环境提供了急需的控制，有助于缓解许多与元数据、缓存和索引服务相关的问题[93]。额外的中间件，如Delta Lake[92]、Iceberg[6]和Hudi[5]，可以跟踪新数据并支持事务更新，使湖仓看起来更像传统的数据仓库。

数据湖给查询优化带来了新的挑战。数据库管理系统（DBMS）在获取数据的精确统计信息方面一直存在困难，导致查询计划选择不佳[154]。然而，数据湖系统可能完全缺乏对新摄入数据文件的统计信息。因此，在云中采用自适应查询处理策略势在必行，以使DBMS能够根据观察到的数据特征在执行过程中动态修改查询计划[97, 105, 163]。

所有主要的云供应商现在都提供某种形式的管理数据湖服务。由于基于对象存储的数据湖系统每千兆字节的成本比专有数据仓库要低得多，传统的OLAP供应商（例如Teradata、Vertica）已经扩展了他们的DBMS，以支持从对象存储中读取数据，以应对这种定价压力。此外，还有一些独立的系统也在这个领域，包括Databricks[105]、Dremio[21]、PrestoDB[63]和Trino[77]。

3.4 NewSQL 系统

在2000年代末，有多个分布式NoSQL数据库管理系统（DBMS）可供使用，它们被设计为水平扩展，以支持大量并发用户的在线应用程序[110]。然而，许多组织无法使用这些NoSQL系统，因为他们的应用程序不能放弃强事务性需求。但现有的关系型数据库管理系统（RDBMS）（尤其是开源的）无法（原生）跨多台机器扩展。作为回应，NewSQL系统在21世纪10年代初出现，寻求为在线事务处理（OLTP）工作负载提供NoSQL系统的可扩展性，同时仍支持SQL[95, 171]。换句话说，这些新系统试图实现21世纪00年代NoSQL DBMS的相同可扩展性，但仍保留20世纪90年代传统DBMS的关系模型（RM）和ACID事务。

NewSQL系统主要分为两类。第一类是内存DBMS，包括H-Store[144, 189]（商业化为VoltDB[83]）、

SingleStore[69]、Microsoft Hekaton[128]和HyPer[146]。其他初创公司提供的包括面向磁盘的分布式DBMS，如NuoDB[47]和Clustrix[17]。

讨论：NewSQL DBMS的采用尚未出现爆发式的增长[96]。这种兴趣不足的原因是现有的DBMS在当时已经足够好，这意味着组织不愿意承担将现有应用程序迁移到新技术的成本和风险。公司在更换OLTP DBMS时比OLAP更为担心风险。如果OLTP DBMS失败，公司将无法执行他们需要的交易来产生收入。相比之下，OLAP DBMS的故障可能仅限于暂时给分析师或数据科学家带来不便。

NewSQL DBMS还有其他限制，例如只支持标准SQL的子集或在多节点事务上表现不佳。一些NewSQL产品，如Microsoft的Hekaton，只能作为遗留DBMS的扩展使用，要求更快的引擎使用较慢DBMS的接口。

NewSQL供应商还错误地预判过去十年内存DBMS的采用会更大。Flash供应商降低了成本，同时提高了存储密度、带宽和延迟。更高的DRAM成本和持久性内存（例如，Intel Optane）的崩溃意味着SSD将保持为OLTP DBMS的主导地位。

NewSQL的余波是一批新的分布式、事务性SQL RDBMS的出现。这些包括TiDB[141]、CockroachDB[195]、PlanetScale[60]（基于Vitess分片中间件[80]）和YugabyteDB[86]。在过去的十年中，主要的NoSQL供应商也在他们的系统中增加了事务，尽管他们之前曾强烈声称事务是不必要的。值得注意的DBMS包括MongoDB、Cassandra和DynamoDB。这当然是由于客户的请求，事实证明事务确实是必要的。Google在2012年放弃最终一致性，转而使用Spanner进行真实事务时，就明智地说过这一点[119]。

3.5 硬件加速器

在过去的50年里，人们一直在寻找一种经济高效的硬件加速器来用于数据库管理系统（DBMS）。其前景显而易见：为DBMS设计的专用硬件应该能够轻松超越传统CPU的性能。

在1980年代，供应商制造了定制硬件来加速DBMS，并将它们作为数据库机器[107]进行市场推广。Britton-Lee在1981年发布了第一个商业加速器产品(IDM/500)[192]，其中包含一个传统CPU和一个硬件加速器，用于卸载查询执行的部分。这个加速器针对执行路径的一小部分，并且成本效益不高。Teradata推出了自己的数据库机器，提供了用于在in-flight元组中排序的网络硬件(Y-net[1])，但它被一个纯软件解决方案所取代[85]。1980年代所有其他的定制硬件DBMS加速都失败了。

过去20年的趋势不是为数据库管理系统（DBMS）构建定制硬件，而是使用通用硬件（如现场可编程门阵列（FPGA）、图形处理器（GPU））来加速查询。这是一个诱人的想法：供应商可以在不花费制造硬件成本的情况下获得DBMS加速器的好处。

Netezza是最早的基于FPGA的DBMS之一，它在1990年代末作为PostgreSQL的一个分支开始。它使用FPGA来加速磁盘上页面的搜索，但最初不能搜索内存页面。Netezza在后来的版本中纠正了这个限制[2]。Swarm64试图销售一个FPGA加速器用于PostgreSQL，但在被收购之前转而使用没有FPGA的纯软件架构[91]。Vitesse的Deepgreen DB[81]是唯一可用的由ISV提供的FPGA增强型DBMS。

在GPU加速的DBMS市场上有更多的活动。值得注意的GPU DBMS包括Kinetica[35]、Sqream[35]、Brytlyt[13]和HeavyDB[48]。如果数据不适合GPU内存，那么查询执行就会因加载数据到设备而受到瓶颈，从而使硬件的并行化优势失效。

讨论：我们可以从上述分析中得出几个结论。首先，这些系统都专注于OLAP市场，只针对RDBMS；本

节讨论中基本上没有数据模型的含义。此外，OLAP工作负载将继续积极向云端转移，但专用硬件不太可能被接受，除非它是由云供应商构建的。

仅为数据库管理系统（DBMS）创建定制硬件对大多数公司来说并不具有成本效益。通用硬件避免了这个问题，但仍然存在将硬件集成到DBMS中的挑战。之所以GPU DBMS比FPGA系统更多的原因是，现有支持库可用于GPU（例如Nvidia CUDA[169]）。但由于规模经济，基于云的CPU计算资源非常便宜。任何加速器的成功可能仅限于本地数据库，但这个市场的增长速度与云数据库不同步。

即使某个加速器能够上市，并且显示出比现有技术数量级的改进，这也只解决了采用和成功所需问题的一半。一个仅生产硬件的公司必须找到某个数据库管理系统（DBMS）添加对其加速器的支持。如果加速器是DBMS的可选附加组件，那么采用率将会很低，因此DBMS供应商不会愿意花费工程时间来支持它。如果加速器是DBMS的关键组件，那么没有供应商会将如此重要部分的开发外包给外部供应商。

唯一能够成功使用定制硬件加速器的地方就是大型云供应商。他们可以在大规模下证明定制硬件5000万至1亿美元的研发成本是合理的。他们还控制着整个堆栈（硬件和软件），并可以在关键位置集成他们的硬件。亚马逊已经通过他们的Redshift AQUA加速器[102]做到了这一点。谷歌BigQuery在内存shuffle方面有定制组件[89]。

尽管胜算很小，但我们预测在未来二十年里，这个领域会有许多尝试。

3.6 区块链数据库

截至目前，区块链数据库作为一种逐渐衰退的数据库技术趋势。这些是去中心化的日志结构数据库（即账本），它们使用某种形式的Merkle树来维护递增的校验和。这些递增的校验和是区块链确保数据库日志记录不可变的方式：应用程序使用这些校验和来验证以前的数据库更新没有被更改。

区块链数据库的理想用例是点对点应用，其中无法信任任何人。没有中央权威来控制数据库更新的顺序。因此，区块链实现使用BFT提交协议来确定下一个应用于数据库的交易。

目前，加密货币（比特币）是区块链唯一的用例。此外，已经有尝试在区块链之上构建一个可用的DBMS，尤其是Fluree [25]、BigChainDB [12] 和 ResilientDB [136]。这些供应商（错误地）推广区块链，认为它提供了以前DBMS不可能实现的更好的安全性和可审计性。

讨论：在当今社会，我们需要信任多个实体。当一个人出售房屋时，他们信任产权公司来管理交易。唯一不需要现实世界信任的应用程序是暗网交易（例如洗钱）。合法企业不愿意支付性能代价（大约五个数量级）来使用区块链数据库管理系统（DBMS）。如果组织相互信任，他们可以更有效地运行共享的分布式DBMS，而无需浪费时间在区块链上。据我们所知，所有主要的加密货币交易所都是使用传统的RDBMS而不是区块链系统来运营他们的业务。

区块链支持者还提出了其他无意义的声明，称通过在对等网络环境中复制数据来实现数据弹性。没有理智的公司会依赖互联网上的随机参与者作为关键任务数据库的备份解决方案。

私人区块链数据库管理系统（DBMS）可能存在一个（较小的）市场。亚马逊在2018年发布的量子账本数据库（QLDB）[65]提供了与区块链相同的不可变和可验证的更新保证，但它不是去中心化的（即没有拜占庭容错提交协议）。亚马逊在发现完全去中心化的区块链DBMS没有令人信服的用例后构建了QLDB[108]。

3.7 总结

从数据库系统中的主要技术趋势中，我们可以得出以下要点：

- 列式系统：转向列式存储彻底改变了OLAP DBMS架构。
- 云数据库：云已经颠覆了构建可扩展DBMS的传统智慧。除了嵌入式数据库，任何不以云为基础创业的产品很可能会失败。
- 数据湖 / Lakehouses：基于云的对象存储使用开源格式将成为未来十年OLAP DBMS的原型。
- NewSQL系统：它们利用了新的思想，但还没有像列式和云DBMS那样产生同样的影响。它导致了新的分布式DBMS，支持更强的ACID语义，以对抗NoSQL的较弱BASE。
- 硬件加速器：我们没有看到除了主要云供应商之外的专业硬件的用例，尽管初创公司将继续尝试。
- 区块链数据库：一种寻找应用的低效技术。历史表明，这是系统开发的错误的途径。

4 结语

我们对过去二十年数据库的分析有几点启示。不幸的是，其中一些是2005年论文中的警告的重复。

永远不要低估良好营销对糟糕产品的价值。数据库市场竞争激烈且利润丰厚。这种竞争促使供应商声称他们的新技术将解决各种问题，并改善开发者的生活。每个开发者以前都曾与数据库斗争过，因此他们特别容易接受这样的营销。劣质的DBMS产品通过强大的营销成功，尽管当时有更好的选择可用：Oracle在1980年代这样做了，MySQL在2000年代这样做了，MongoDB在2010年代这样做了。这些系统在早期获得了足够的关注，为他们赢得了时间来弥补他们早期积累的工程债务。

要警惕大型非专业数据库供应商的产品。过去十年数据库的一个有趣方面是，科技公司在内部构建DBMS，然后将它们作为开源项目分拆出来的趋势。所有这些系统最初都是作为科技公司的特定应用程序而开始的。然后，公司将DBMS作为开源项目发布（通常推向Apache基金会进行托管），希望从外部用户那里获得“免费”的开发。

有时它们来自能够负担开发新系统的大公司。著名的例子包括Meta（Hive [197], Presto [63], Cassandra [14], RocksDB [68]）和LinkedIn（Kafka [33], Pinot [59], Voldemort [82]）。其他系统来自初创公司构建数据密集型产品，他们觉得也需要构建一个数据库。最成功的例子是10gen (MongoDB) 和 PowerSet (HBase)，但也有很多失败的尝试。

这种避免“not invented here”软件的趋势部分是因为许多公司的晋升路径倾向于奖励那些制作新内部系统的工程师，即使现有工具已经足够。但这种扭曲导致许多没有数据库管理系统（DBMS）工程经验的团队开始构建新系统。当一家公司首次将其开源时，我们应该警惕这样的系统，因为它们几乎总是不成熟的技术。

不要忽视开箱即用的体验。许多非关系型DBMS的一个显著卖点是比关系型DBMS更好的“开箱即用”体验。大多数SQL系统需要用户首先创建一个数据库，然后定义他们的表，之后才能加载数据。这就是为什么数据科学家使用Python notebook快速分析数据文件的原因。因此，每个DBMS都应该让本地和云存储文件的原位处理变得容易。DuckDB日益流行的部分原因是它能够很好地做到这一点。

供应商还应该考虑客户在使用数据库时不可避免会面临的额外挑战，包括物理设计、参数调优、模式设计和查询调优。这是一个至关重要，可称之为“自动驾驶”DBMS的需求[173]。

开发者需要直接查询他们的数据库。在过去20年里创建的大多数在线事务处理（OLTP）应用程序主要通过抽象层与数据库交互，例如客户端API（例如REST、GraphQL）或对象关系映射器（ORM）库。这些层将应用程序的高级请求转换为数据库查询。ORM还自动处理维护任务，例如模式迁移。有人可能会争辩说，由于OLTP开发人员从未在他们的应用程序中编写原始SQL，所以他们的DBMS使用什么数据模型并不重要，因为这些层隐藏了它。

ORM是快速原型设计的重要工具。但它们通常牺牲将逻辑推送到DBMS的能力，以换取与多个DBMS的互操作性（兼容性）。开发者转而编写显式的数据库查询，以覆盖自动生成的低效查询。这就是为什么使用支持SQL的关系型数据库管理系统（RDBMS）是更好的选择。

人工智能/机器学习（AI/ML）对DBMS的影响将是巨大的。DBMS应该如何与现代AI/ML工具互动最近成为一个关键问题，特别是随着大型语言模型（LLM）（例如ChatGPT）的出现。虽然这个领域发展迅速，但我们提供一些初步意见。

由于LLM在将自然语言（NL）转换为查询代码（例如SQL）方面的进步[133]，使用NL查询数据库正在复兴。一些人甚至建议，这种AI驱动的查询界面将使SQL过时。NL界面是一个可以追溯到1970年代的老研究课题[139]，但历史上结果不佳，因此几乎没有广泛使用[88]。我们承认LLM在这项任务上有令人印象深刻的结果，但警告那些认为NL将取代SQL的人。没有人会使用NL编写OLTP应用程序，因为大多数使用ORM生成查询。对于在线分析处理（OLAP）数据库，NL可能在构建探索性分析的初始查询时有所帮助。然而，这些查询应该暴露给类似仪表板的工具，因为英语和其他NL充满了歧义和不精确性。

在企业内部，尤其是在处理财务数据时，人们不愿意依赖当前的LLM技术进行决策。最大的问题是LLM的输出对人类来说是无法解释的。其次，LLM系统需要的训练数据比"传统"ML系统（例如随机森林、贝叶斯模型）更多。公司通常不能将这些模型的训练数据创建外包给非技术人员。由于这些原因，企业数据采用LLM的速度将会谨慎缓慢。

最后，最近有大量关于使用AI/ML优化DBMS的研究[174]。例子包括面向ML的查询优化器[152，156]、配置调优[200，204]和访问方法[151，193]。虽然这种ML辅助优化是提高DBMS性能的强大工具，但它并不能消除对高质量系统工程的需求。

5 结论

我们预测，数据库的发展在未来几十年将继续循环往复。另一波开发者将声称SQL和关系模型（RM）已不足以应对新兴的应用领域。人们随后会提出新的查询语言和数据模型来解决这些问题。探索DBMS（数据库管理系统）的新思想和概念具有巨大的价值（这是我们为SQL获取新功能的地方）。正因为如此，数据库研究社区和市场变得更加健壮。然而，我们不认为这些新的数据模型会取代关系模型（RM）。

另一个担忧是很多新轮子在重新实现并且没有创新，在DBMS所必需的相同组件（例如，配置处理器、解析器、缓冲池）上的白白浪费。为了加速下一代DBMS的发展，社区应该促进开源可重用组件和服务的开发[112， 176]。有一些努力朝着这个目标前进，包括文件格式（见第3.3节）、查询优化（例如，Calcite [104]，Orca [186]）和执行引擎（例如，DataFusion [18]，Velox [175]）。我们认为数据库社区应该努力制定一个类似于POSIX的DBMS内部标准，以加速互操作性。

我们提醒开发者从历史中学习。换句话说，站在前人的肩膀上，而不是他们的脚趾上。我们中的一个很可能在二十年后仍然健在，因此非常期待在2044年撰写这篇论文的后续文章。

注：虽然Stonebraker已经81岁高龄，Andy出生于1981年，但还是特别期待20年后还能读到Stonebraker老爷子的文章。

由于论文涉及到几乎数据库所有的领域，很多内容不一定翻译准确，还请谅解，更欢迎一起讨论。论文参考资料太多，详细可以参考原文：

https://db.cs.cmu.edu/papers/2024/whatgoesaround-sigmodrec2024.pdf

整理人

叶正盛，NineData 创始人 &CEO，资深数据库专家，原阿里云数据库产品管理与解决方案部总经理。NineData（www.ninedata.cloud）是云原生数据管理平台，提供数据库 DevOps（SQL IDE、SQL 审核与发布、性能优化、数据安全管控）、数据复制（迁移、同步、ETL）、备份等功能，可以帮助用户更安全、高效使用数据。