第一章：简介

像MongoDB,CouchDB,Riak, Cassandra,或者HBase 这样的NoSQL数据库系统的主要优势在于它们的对现代应用的后台存储的方便性。这些数据库系统让应用能够更加灵活地存储和读取，无需事先定义一种模型就可以获取数据。因为移除了事先数据管理的操作，开发者们能够更加容易的获取并运行他们的应用程序而不用担心由于他们数据集合的范围、类型和依赖性可能带来的错误。

有人质疑这种“立即满足”型的应用开发方式可能会长期导致在代码管理、分享、性能等方面的问题。但是NoSQL方法的支持者们认为在发展响应迅速地独立数据集的过程中，仅仅是维持模型的开销的成本就太高。无论这两种方法孰优孰劣，在实践中我们发现使用键值和半结构化的数据形式的数量正在增长。而有策略的分析研究这些海量数据的重要性也在不断增加。

有一些数据库系统一开始就支持对原始的数据进行分析。例如MongoDB现在提供了一系列的数据分析集成工具；MapReduce也有分析数据的功能；而其他的NoSQL数据库系统则通过连接能够进行数据分析的开源计算框架（如：Hadoop，MapReduce，Apache Tez）来进行数据分析。

但是现在在使用NoSQL数据库系统的功能时存在非常明显的缺点。本地原始数据有时会和SQL的标准差很远，从而导致了很多第三方智能分析工具（SAP Business Objects, IBM Cognos, Microstrategy ,Tableau）不能使用。同时，由于Hadoop 提供了SQL接口用来存储数据（如 Hadapt，Hive，Impala）而这些功能要求用户在数据分析前首先通过SQL创建表。这也是NoSQL一开始就排除掉的主要原因。在有些实例中，模型可以容易的添加，但是其他情况中，数据必须首先要抽取-转换-加载 才能导入到有用的模型中。

传统数据库系统能够对多结构的数据（如：关系、键值或者其他半结构数据）进行标准的SQL查询。而这篇论文讨论的是如何设计一层关系使得用户能够在任何情况下都不需要建立数据模型。基本的想法就是告诉读者一个普遍的逻辑关系【13，15】，这个关系中对于存在独立的键列的逻辑表都在与之对应的独立数据集。而嵌套数据会被分解到不同的列中，因此可能导致逻辑表可能具有成百上千的列。

由于用物理方式存储这种数据是不切实际的，所以物理表示方法因逻辑表示方法的不同而改变。数据虽然存在关系数据库中，但是只有其中的一个子集真正的被定义成关系数据库中的一个列，而其他的列则是以连续的二进制列的方式存在与数据库系统中。当用户要使用时，这些数据通过内部解压的方式用这些连续的二进制列中提取出来。

我们主要的工作就是完成一个能够合理地完成上述操作的数据库系统的结构。这个结构包含以下内容：一个关系存储层；目录；表分析器；列具体化功能（column materializer），加载器； 查询重载器；转化数据索引。这篇论文描述了它的设计，启动还有各个元素之间的交互。

我们也设计了一个我们心目中的系统的原型，并把它和其他几种有多种存储方式和分析方式的系统进行了比较。（1. mongoDB 2. Postgre的JSON拓展 3. 能够存储键值对的EAV 模型的关系数据库）我们发现我们的原型在上述功能中表现更好而且还能提供更多的标准SQL接口。

第二章： 相关工作

分析性的系统并不要求用户定义一个表因为数据直接被系统以两种类型装载：
1.专门针对无联系数据模型的数据库系统，它没有专门的SQL接口
2.一般提供SQL或者类似SQL接口的数据库系统，它要求用户在查询前先定义潜在的数据的类型。

为了追求编程时的方便和灵活，第一种系统通常会要求使用者放弃一部分子集或者学习一种查询的语言。例如：JAQL使用迭代函数来提供了单元化数据的处理平台。不管它的拓展性，它不能使关系型的操作更加优化； MongoDB 虽然能够接受像JSON 这样的数据形式，它还是要求用户学习它的基于JS的查询语言，但是这种语言不支持类似“关系连接”（relational joins）这样的操作。

第二种系统允许用户任意的定义一个模型（schema）。大多数数据库系统通过外部的表支持这样的操作。从而把它读到的无联系的数据转化成有联系的数据。这些外部表还可以以索引的方式提高数据库系统的性能或者加载到数据库系统中。很多使用了Hadoop开源环境的SQL项目（Hive和 Impala）都用了这个相似的概念，也就是提供一个没有独立存储模型的关系查询引擎。数据存在

HDFS中，而用户通过SQL-on Hadoop的方法访问他们的数据。登陆了schema之后，用户可能输入查询命令，然后读取在HDFS上的数据然后利用SOH的方法完成他们的操作。

虽然这些系统比原来那些要在加载的时候预先定义一个模型的系统更加灵活，他们仍然有一定的限制。要完成SQL查询的命令，还是需要预定义一个目标模型来执行这些查询。而Sinew 则不需要这些要求，自动的展示数据的逻辑视图通过数据本身而不需要用户的输入。

Google Tenzing, Google Dremel 和Apache Drill 提供了无需定义schema就直接用SQL查询的方法。Tenzing， 类似Hive的SQL-on-MapReduce 系统通过潜在的数据推测关系模型 当且仅当 单调的数据类型能够匹配存在的关系模型。相对的，Dremel 和Drill 还有Sinew 支持嵌套数据。与Sinew不同的是，Dremel 和Drill只是查询执行引擎，只能单纯的提供分析。而Sinew 被设计为传统数据库系统的拓展，能够支持半结构化数据和支持其他存在于关系数据库中的键值数据。Sinew能够支持事务的更新，综合存储的接入控制，读写并发控制。Sinew 还可以继承原来关系数据库的优点：数据的统计和最优的查询能力。这个令Sinew 与Pathfinder类似，但是Sinew比它更加广泛的支持所有多结构数据，为数据提供更加清晰准确的SQL接口。

为了能够清楚的为多结构数据提供一个全特征的关系接口，Sinew使用了一种新的方法来在RDBMS存储多结构数据。一种普遍的方法是（曾被XML数据库使用过）是创建一个“shredder”来把文件传输进入RDBMS。通常依赖于一些边际模型的改变来把文件装入关系元组中。但是，即使是潜在的关系schema对于预测和选择等操作是最优的，系统的性能仍然被限制，因为即使重建一个最简单的记录也需要多次拼表。

最近，电子商务和其他网络应用使用一个row-per-object的数据映射方法。这检验了独立数据集对大数据的形成和存储的性能。这种方法要求对象能够整合进入每一个可能的键所在的元祖并且用具有很多列的宽表来存数据。一般来说，这种数据具有大量稀疏的键值，所以要求RDBMS的那些NULL的空数据不能对查询产生过度的影响或者造成性能的下降。面向列的RDBMS能够满足上述要求，但是它们在重建嵌套数据时存在困难。以为嵌套数据不是清楚的表示，所以只有用它们内部出现的键的集合来表示它们。我们将在3.1.1更加深入的讨论这些问题。