## 3.1 数据组织方式

数据管理系统功能的本质是数据的存储和查询，换言之，数据是DBMS系统操作的对象。关系数据模型是上个世纪80年代是研究的重点，也在之后的很长时间成为主流的数据模型。但是在同时，关系模型的局限也渐渐显露出来：在媒体、医疗、地理等领域中，关系模型的表现很差。因此，对象-关系模型和其他的模型逐渐被提出以弥补关系模型的不足。

按照不同的关系模型，科学界和工业界实现了不同的系统。在本节中，我们比较了数款常用的数据库系统，这些数据库系统包括：

* 分布式关系数据库Greenplum和BigTable
* 分布式文档存储系统MongoDB
* 分布式图形数据库Neo4j

在接下来的部分，我们将会详细说明这些系统是如何在逻辑和物理上组织数据的。除此之外，在本节的最后我们也添加了目前普遍使用的行列存储技术以及数据压缩技术的介绍。

### 3.1.1 Greenplum

Greenplum是Pivotal公司推出的分布式关系数据库，采用MPP架构，主要面对OLAP业务。在功能上，Greenplum支持传统的数据库技术，如索引、日志、锁等。除此之外，Greenplum还支持分区表、地理信息索引、json格式数据等。

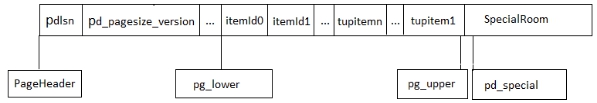
传统的关系型数据库主要采取行存储模式，这对于OLTP型事务是便捷的。行存储逻辑上如下表所示，表的每一行代表一个元组，按照关系型数据库的这种天然属性把数据按行来组织在一起。行存储模式下同一个元组的不同属性在存储上是连续的。如果要存储一个元组的某个属性值，首先读取元组的所有数据，然后进行投影操作，从整个元组中分离出需要的属性值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Album | Type | Singer | Comments |
| 后来的我们 | 五月天 | 流行 | 五月天 | 72842 |
| Lighters | Marshall Carter | Rope | Bruno Mars  Royce Da | 25427 |
| 十年 | 黑白灰 | 伤感 | 陈奕迅 | 56822 |
| 缘分 | 最红 | 怀旧/  影视原声 | 张国荣/  梅艳芳 | 410 |

除了行存储，Green plum还支持列存储模式，使用于OLAP场景。列存储模式下数据按照属性列来进行组织，如下表所示。表的不同元组的同种属性在存储上是连续的，不同的列独立存储。在Greenplum中，每一个属性列在操作系统中都是以单个文件的形式存放，根据实际需要读取对应的属性列。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | 后来的我们 | Lighters | 十年 | 缘分 |
| Album | 五月天 | Marshall Carter | 黑白灰 | 最红 |
| Type | 流行 | Rope | 伤感 | 怀旧/  影视原声 |
| Singer | 五月天 | Bruno Mars  Royce Da | 陈奕迅 | 张国荣/  梅艳芳 |
| Comments | 72842 | 25427 | 56822 | 410 |

Greenplum的物理存储模式如下图所示：一个文件页空间被逻辑地分割成三部分：页描述区（Page Header），元组数据空间（Tuple Item Space），以及特殊空间（Special Space）。页描述区记载了页的使用情况 , 包括元组数据空间和特殊空间的起始位置等。元组空间是实际记录元组数据的地方。每个被记录的元组称为一项 ,每项由描述id和元组数据构成 , 项描述 id 描述了元组存储位置 、大小以及一些状态标志。项描述id和项数据分别在元组数据空间的两头往中间靠拢存放 。



### 3.1.2 BigTable

BigTable是谷歌开发的分布式存储系统，可以用来管理半结构化数据。具有较强的扩展性，可以存储PB级的数据。谷歌网页的索引、Google Earth以及Google金融的数据都使用到了Big Table技术。

从功能上看，BigTable很像一个数据库，实现了很多数据库的策略，例如增删查改等。但BigTable提供了另一种组织数据的思路：

* 数据通过行和列名进行索引，这些名字可以是任意的字符串；
* 所有数据的类型都是无解释的字符串；
* 支持从内存或者硬盘中读取数据；

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Row Key | Time Stamp | Column Comments | Column Lable | | Column Singer |
| 怀旧 | 影视原声 |
| 缘份 | T5 | 410 | true | true | 张国荣 |
| T4 | 360 | true | true | 张国荣 |
| T3 | 310 | true | true | 张国荣 |
| T2 | 240 | true | true | 张国荣 |
| T1 | 210 | true | true | 张国荣 |

BigTable的具体数据组织方式如上图。

BigTable采用行键（row key）、列键（column key）和时间戳（timestamp）对图进行索引：

* 一个表中的行键，是任意的字符串。对每个行键下所包含的数据的读或写都是一个原子操作，（类似于行级别锁），以维护BigTable内数据的一致性。由于BigTable的存储是分布式的，行键还是数据划分的依据。
* 列键，也可以称为列簇，是一组相关属性的集合，通常都属于同一个数据类型。列键采用下面的语法命名：family:qualifier。例如上表中的Lable：怀旧，Lable:影视原声都属于同一个列簇，他们都表示了“缘分”这首歌曲的标签属性。在BigTable中，访问控制以及磁盘和内存审计是在列家族层面上进行的。
* 时间戳用来区分相同数据的不同版本，是一个64位的整型数。时间戳必须是唯一的，以避免数据的冲突。上表的T1-T5代表5个不同的时间戳“缘分”这首歌曲的信息，可以看出它们在不同的时间戳的评论数（Comments）列存在不同。在物理存储上，相同数据的不同版本根据时间戳进行降序存储。这样的好处是最新的版会被最先读取。

### 3.1.3 MongoDB

MongoDB是一个基于文档的数据库系统，有着灵活的存储模式。这种模式在MongoDB中被称为集合（Collections）。每一个集合都是一个数据库，类似于RDBMS中的表的概念；每个集合都可以有多个文档(Documents)，类似于RDBMS的一行。然而不同于RDBMS的是集合的模式并非预先定义好或者说是一种松散的模式。

MongoDB使用二进制json（Binary JSON，BSON）存储数据。BSON格式的数据(文档)存储有尺寸限制，最大为16M。因此对于较大的数据，MongoDB使用GridFS来辅助管理。GridFS使用两个集合（collection）存储文件：files集合用于存储文件的元数据；chunks集合用于存储文件内容的二进制数据。

BSON格式

BSON这种格式是专门为MongoDB而开发的，类似json的一种二进制格式。这种格式不一定比json存储的文件小，其优点是解释快。另外BSON增加了一些数据类型，如Date.

以歌曲的名称信息为例，一个具体的BSON示例如下：前面的四个字节表示文档的长度，采用小端码表示，表示文档的长度是20个字节。第5个字节（\x02）代表元素的类型，表示string类型。接下来的字节分表表示元素的名字、元素类型的长度以及元素的内容。最后一个字节（\x00）表示文档的结尾。

{"name": "缘分"}

文档长度

元素内容

元素名称



元

素

长度

元

素

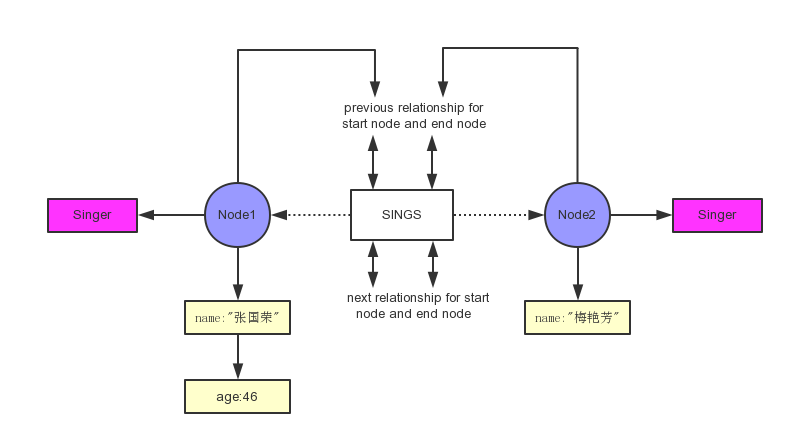
类

型

MongoDB 内部有预分配空间的机制，每个预分配的文件都用0 进行填充，由于有了这个机制,MongoDB 始终保持额外的空间和空余的数据文件。

### 3.1.4 Neo4j

Neo4j是一个高性能的NOSQL图形数据库，它将结构化数据存储在网络上而不是表中。下图描绘了Neo4j的存储模型。图中的顶点（Node)包含自身所拥有的属性结构（Property）和关系结构（Relationship ），这些结构都以链表的形式存储，Node分别保存着这些链表的头部指针。属性链表是一条单链表，为了读取节点的属性，我们从指向属性链表的头部指针出发依次向后读取；而关系链表是一条双向循环链表，同样遍历关系链表也能找到顶点所有的关系。



在Neo4j中,主要有4类节点，属性，关系等文件是以数组作为核心存储结构，每条记录都是定长的；同时对节点，属性，关系等类型的每个数据项都会分配一个唯一的ID，在存储时以该ID 为数组的下标。这样，在访问时通过其ID作为下标，实现快速定位。所以在图遍历等操作时，可以实现 free-index。这些类的具体结构可以参看下图。inUse是标志位，标志该条记录是否正在使用（是否可以进行回收）。nextRelID和nextPropld对应着前面提到的链表结构。Label是节点的标签信息；Extra是标志位。例如用来识别密集连接顶点（densely connected nodes）。同样的，在关系结构中，同样有inUse的标志位。跟随其后的是和关系有关联的两个定点的ID。其次还保存着关系类型的指针、两个端点在关系链表中的上一个/下一个关系的指针，以及当前关系是否位于关系链（relation chain）头部的一个标志位。