**DB & DW**

|  |  |
| --- | --- |
| **审核人** |  |
| **重要性** | 中 |
| **紧迫性** | 中 |
| **拟制人** | 张包峰 |
| **提交日期** |  |

**作者：张包峰**

(版权所有,翻版必究)

**修改记录**

|  |  |  |
| --- | --- | --- |
| **更新时间** | **变更内容** | **变更理由** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

[**作者：张包峰** 1](#_Toc394787100)

[1.1 Parallel Database 4](#_Toc394787101)

[1.2 GreenPlum 4](#_Toc394787102)

[1.3 Vertica 4](#_Toc394787103)

[1.4 AsterData 4](#_Toc394787104)

[1.5 Paraccel 4](#_Toc394787105)

[1.6 TeraData 4](#_Toc394787106)

[1.7 Netezza 4](#_Toc394787107)

[1.8 列数据库 5](#_Toc394787108)

[1.8.1 InfoBright 5](#_Toc394787109)

[1.8.2 Sybase IQ 7](#_Toc394787110)

## Parallel Database

## Google Mesa (再读)

### 背景

Geo-Replicated, Near Real-Time, Scalable Data Warehousing

### 存储方面的设计

数据模型，

更新和查询，

数据版本管理，

多版本管理中，delta version的预处理是主要设计。delta compaction策略。

物理存储和索引，

物理存储形式比较简单，主要是要能节约存储空间。每张table有一个或多个索引，在选择压缩算法上，更看重压缩率和读的时候解压缩的速度，胜过于写的时候压缩的速度。

基本的索引查找是二分的方式。

### 系统架构

Mesa建立在BigTable和Colossus之上，是跨数据中心建立的。每个数据中心上一个实例。

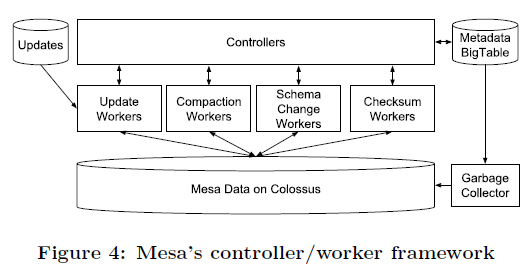
Mesa单个数据中心的实例由两个子系统组成：负责更新/维护的子系统和负责查询的子系统。

两套子系统是解耦的，方便各自的扩容。

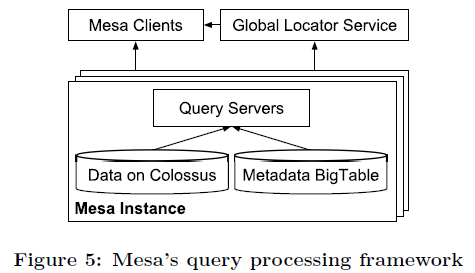
更新子系统

负责更新的子系统需要确保数据在该Mesa实例上的正确性，时效性以及为查询做好优化。后台需要处理很多不同的操作，包括载入更新数据，执行table合并，更改schema信息，给table做checksum。更新子系统主要由controller/workers框架实现。

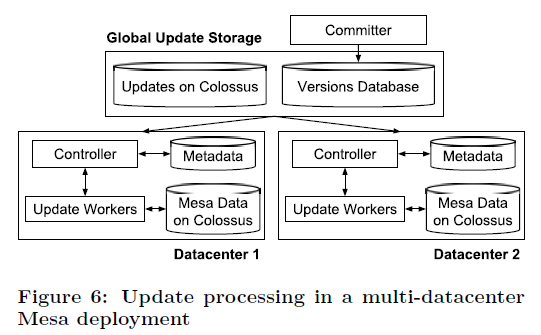
controller类似master节点，负责让worker执行工作，维护所有表的元数据。既充当了一个可扩展的table metadata cache的角色，又充电了work调度和work队列管理者的角色。



查询子系统



### 跨数据中心部署



一致性更新策略

Committer、版本数据库，如何做到查询和更新的搞吞吐量的。

新的Mesa实例

### 其他增强设计

### 经验教训

引用里有丰富的资源可以扩展阅读。

## GreenPlum

## Vertica

a DBMS Architecture Optimized for Data Warehousing

全名叫Vertica Analytic Database

### 介绍

[Vertica](http://www.vertica.com/)(属于HP公司)，是一个基于DBMS架构的数据库系统，适合读密集的分析型数据库应用，比如数据仓库，白皮书中全名称为Vertica Analytic Database。从命名中也可以看到，Vertica代表它数据存储是列式的，Analytic代表适合分析型需求，DB代表本身是数据库，支持SQL。

### 优势

和传统关系型数据库系统以及其他列式数据(仓)库相比，Vertica存在下面三点最关键的优势。

**列存储**

Vertica对磁盘上的数据采用列式存储，显而易见，列存储可以在数据读取的时候避免不需要的列，减少IO带宽，而且列存储配合压缩算法可以节省磁盘存储量。Vertica的列存储在压缩方面还有其优化之处，见下文。

**Aggressive Compression**

这个姑且理解为侵略性压缩好了。其实在压缩方面，Vertica针对不同的数据类型，采用了多种不同的压缩方式，让原本磁盘上大量IO开销与CPU的压缩工作达到比较好的tradeoff。压缩率大致是高达90%。此外，Vertica 的写数据和读数据是分开进行的，读的数据以压缩的状态存在磁盘上，写的数据先缓存在内存里，异步合并到磁盘上，这个下文还会提到。

**多备份**

由于压缩比率很高(90%)，所以能够腾出足够的磁盘空间来做备份。这点很关键。首先，备份让Vertica具备容错性，且多机器上的备份还能提升ad-hoc查询性能。其次，能够通过备份来容错这点，区别了Vertica与传统数据库通过logging和二阶段提交这种方式来做容错的做法。而能做多备份的原因是由于其列存储压缩做的优化比较好，这点又是比其他列式存储的数据仓库有优势的地方。

table被拆分后存储的每个单元叫projection，每个projection按某个或某些attribute进行排序，而且不同的副本排序方式还会不同，所以这对查询又是有帮助的，这点下文也会提到。

当然，Vertica的压缩，面向列的存储以及table拆解后的存储对用户来说是透明的。Vertica对前端用户提供的是标准的SQL接口，而且兼容现有的ETL，reporting，BI工具，所以这点使得其他业务系统可以更方便迁移到Vertica上。

Vertica对硬件也没有特殊要求，可以跑在廉价的集群上，或是任何现成的linux机器上，使用本地磁盘做存储。

除了上面说的几点优势之外，Vertica还在性能，可扩展性，可用性以及使用便利性方面有以下优势。

**shared-nothing，grid-based 数据库架构**

Vertica可以高效的在普通的机器上扩容。事实上，在数据备份这块，Vertica提供k-safety的方式，k+1是备份数，可调，且每一份是完全的数据，后面会提到。

**混合数据存储**

这里的混合指的是内存和磁盘。一方面，新插入的数据是写入针对写做了优化的内存结构里，所以支持持续的，高性能的并发写入操作；另一方面，不影响实时的查询性能，因为读的数据来自于磁盘。事实上，写数据在内存里是乱序的，所以写吞吐会比较高，而读数据在磁盘上是有序的。

**物理数据库设计工具**

这说的是在使用Vertica的时候，提供了比较方便的工具，能帮助用户设计物理表，下文会提到。而且提供k-safety的机制保证容错。

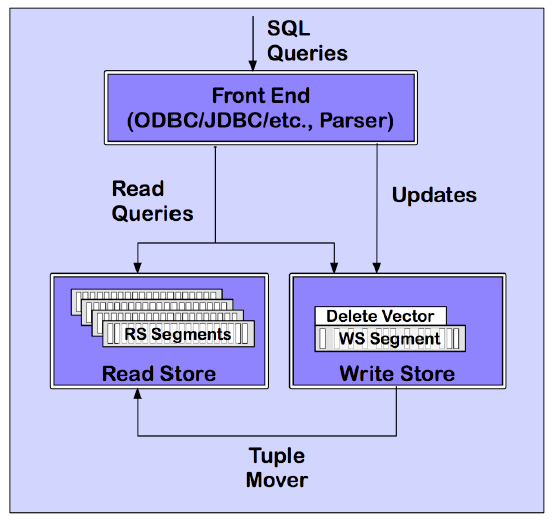
**高性能兼容ACID数据库**

有轻量级的事务和并发控制scheme，针对查询和数据载入。并且具备基于k-safety的失败恢复模型，而不是传统的基于日志的。

**方便的部署、监控、管理**

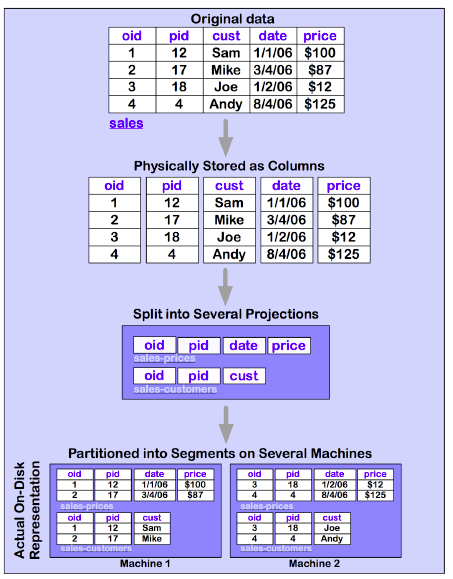
略

### 架构



上图为Vertica单个节点的架构，我们看到查询和更新是走的所谓的混合存储，即write-optimized store(以下简称WOS)和read-optimized store(以下简称ROS)。WOS在内存里，对应数据写入，里面的数据不排序也不压缩。ROS是被排序和压缩过了的数据，存在磁盘上，提供了支持高效查询的格式。Tuple Mover负责把WOS的数据迁移到ROS上，以批的形式把WOS内存里的数据排序和压缩后移到ROS，即磁盘上，也是保证高效的。在WOS和ROS内部，数据都是面向列存的。

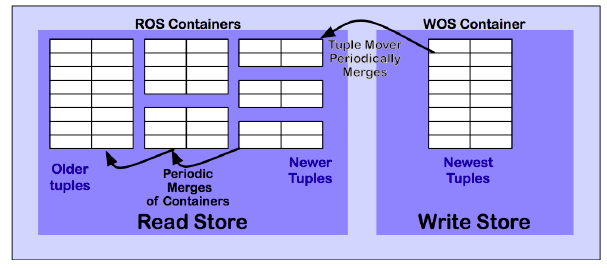
下面这张图展示了table存入到Vertica后，是怎么转变到物理存储的column形态的，即被切成了Projections。



在切分成物理存储这点上，Vertica的Database Designer会帮助选择一个合理的切分方式，这是上面提到的使用上的便利性的优势，下文还会简单提到如何做到。

看上去，对每个projection在多机上冗余备份存储是种浪费磁盘的做法，但是Vertica的侵略性压缩提供了90%的压缩率，所以不用担心。

下面展示的是从WOS到ROS，Tuple Mover如何做数据合并的大致流程。



具体不详细解读，大致是异步和批量的做merge，而且projection是被纵向切成segments，会方便高效的归并。

### 性能

这一节稍微深入分析和对比下Vertica的性能优势。

**列存储方面**

对于普通的查询来说，主要是磁盘访问和cpu cycles占据的时间，哪个时间长就是查询时间，或者说是瓶颈。那列式存储的话，通过压缩来充分使用cpu资源，减少io开销，在cpu和io上做到比较合理的tradeoff，这点是Vertica一个很重要的优势。

其他数据库系统会通过支持物化views或data cubes，来减少某些查询场景下的时间，类似于做一些预计算和预处理来优化查询速度。但是这点对查询的场景支持有限，并且仍然不及Vertica提供的查询性能。

**压缩方面**

Vertica除了提供RLE(run-length encoding)，还对连续的数据提供delta encoding和一个高效的Lempel Ziv实现，该实现很适合排过序的大多数值不相同的列数据，或未排序的数据。为浮点和时间数据提供特殊的压缩方式。

**ROS方面**

ROS本身的压缩和排序处理，使得磁盘上的数据查询比较高效。此外，ROS是dense packed不浪费disk pages。而传统数据库往往会让page留很多空，以便在不重新整理的前提下继续插入数据，磁盘空间利用率不是最高。ROS这部分工作是Tuple Mover异步做的，在归并阶段控制应该比较容易控制。在查询的时候，也会预读取查询量比较大的大块ROS。

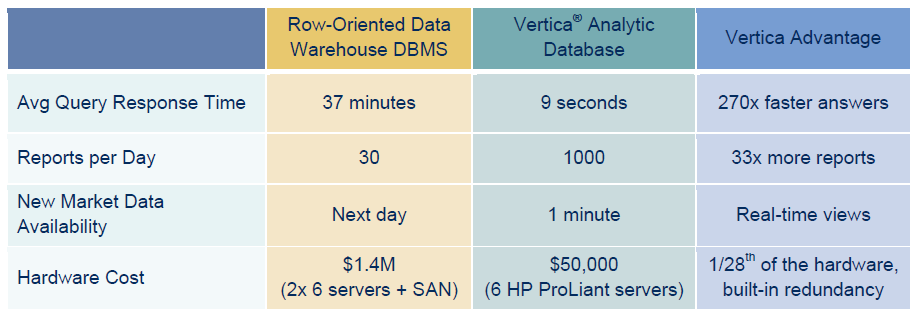
**排序方面**

如前面提到，Vertica通过多备份来做HA和失败恢复，有别于传统数据库基于日志的失败恢复方式，写入过程相比Vertica的话代价更大些。此外，额外不同的排序方式加快了查询速度，Vertica也会选择最优的方式来做排序。

**parallel shared-nothing on off-the-shelf hardware设计**

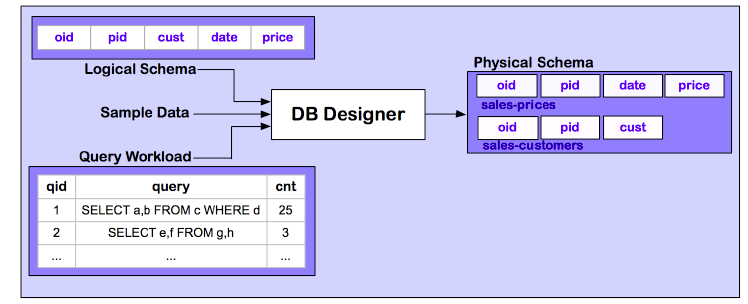
不多说了，方便线性扩容，对硬件没特殊要求。

Vertica的一个benckmark，如下图。



### 管理

之前说的DB Designer会帮助使用者切分表，做比较好的物理存储选择，原理如下，



它参考逻辑schema，样本数据和query语句三个东西来做决策的。而且DB Designer能够增量地为数据库进行重新设计物理schema。

关于失败恢复和k-safety的HA保障再提几点。商业和传统数据库通过日志记录和二阶段提交保证事务性的方式来做失败恢复，回滚之类的事情，Vertica通过备份k+1份实现，只要有一台没有挂掉，就可以从它那全份拷贝恢复，容错性是高的，而且基于压缩率高，我们也不担心磁盘因此吃紧。这一点还让Vertica能够提供hot-swapping(热替换)节点，即比较方便地移除节点和新增节点。

传统数据库是record-at-a-time或bulk loading的方式来插入新数据，Vertica与此不同的是能够提供持续载入功能，查询的节点往往是以snapshot isolation的模式进行的，所以某种意义上是read-only的，因此在写入的时候也不用上锁。而且WOS里的数据不需要排序也不压缩，批量写入吞吐是比较高的。

### 总结

Vertica与传统数据库系统和其他列式数据仓库系统相比的话，在性能上有比较明显的优势，在设计上有一些异同，比较适合ad-hoc查询，OLAP类型的作业。总的来说，Vertica通过列存储减少了io开销，再加上高效的压缩手段，极大节省了磁盘空间，基于此Vertica采用多备份来保证高可用性，并且多备份又能够增强查询性能。在使用和运维角度了，Vertica自带工具帮助用户做物理表的存储，能提供标准SQL接口，也兼容现有的BI、ETL工具方便作业往Vertica上迁移，而且Vertica部署对硬件没有特殊要求，能够线性扩展。

## AsterData

## Paraccel

## TeraData

## SAP

## SAS

## Netezza

## InfoBright

论文

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.5467&rep=rep1&type=pdf>

<http://bookshadow.com/weblog/2014/04/22/infobright-datawarehouse/>

Infobright是开源的MySQL数据仓库解决方案，引入了列存储方案，高强度的数据压缩，优化的统计计算(类似sum/avg/group by之类)，

infobright 是基于mysql的，但不装mysql亦可，因为它本身就自带了一个。mysql可以粗分为逻辑层和物理存储引擎，infobright主要实现的就是一个存储引擎，但因为它自身存储逻辑跟关系型数据库根本不同，所以，它不能像InnoDB那样直接作为插件挂接到mysql，它的逻辑层是mysql的逻辑层加上它自身的优化器。

### 优点

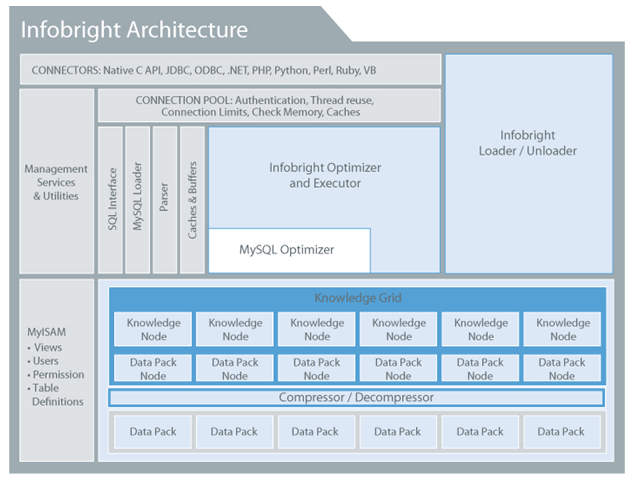
1、高压缩比率，平均压缩比可达10：1，甚至可以达到40:1，我用infobright把3.1G的数据存成不足300M。

2、列存储，即使数据量十分巨大，查询速度也很快。用于数据仓库，处理海量数据没一套可不行。

3、不需要建索引，就避免了维护索引及索引随着数据膨胀的问题。把每列数据分块压缩存放，每块有知识网格节点记录块内的统计信息，代替索引，加速搜索。

4、单台服务器可以高效地读写30T数据。具有可扩展性，这里是指对于同样的查询，当数据量是10T时，它耗费的时间不应该比1T数据量时慢太多，基本是一个数量级内。

下面是Infobright的架构图：



灰色部分是mysql原有的模块，白色与蓝色部分则是 infobright自身的。

### 系统结构分析

跟mysql一样的两层结构，上面的逻辑层处理查询逻辑，下面的是存储引擎。

逻辑层右端的loader与unloader是infobright的数据导入导出模块，也即处理SQL语句里LOAD DATA INFILE … 与SELECT … INTO FILE任务，由于infobright面向的是海量数据环境，所以这个数据导入导出模块是一个独立的服务，并非直接使用mysql的模块。

逻辑层的infobright优化器包在mysql查询优化器的外面，如下面将会提到的，因为它的存储层有一些特殊结构，所以查询优化方式也跟 mysql有很大差异。

存储层最底层是一个个的Data Pack（数据块）。每一个Pack装着某一列的64K个元素，所有数据按照这样的形式打包存储，每一个数据块进行类型相关的压缩（即根据不同数据类型采用不同的压缩算法），压缩比很高。它上层的压缩器与解压缩器就做了这个事情。

压缩层再向上就是infobright最重要的概念：Knowledge Grid（知识网格），这也是infobright放弃索引却能应用于大量数据查询的基础。它包含两类结点：每个Data Pack Node（知识节点）对应于一个Data Pack，存储该Data Pack的一些统计信息，如min, max, avg, null的个数，甚至不同值的量等等；Knowledge Node则存储了一些更高级的统计信息，以及与其它表的连接信息，这里面的信息有些是数据载入时已经算好的，有些是随着查询进行而计算的，所以说是具备一定的“智能”的。

### 论文Brighthouse

论文 Brighthouse: An Analytic Data Warehouse for Ad-hoc Queries，VLDB 2008

brighthouse是一个面向列的数据仓库，在列存储和压缩数据方面，数据压缩比达到10：1。其核心Knowledge Grid（知识网格）层，即一个能自动调节、所存出具特别小的元数据层，替代了索引的功能，提供了数据过滤、统计信息表达、实际数据位置信息等内容，让brighthouse可以作为一个分析型的数据仓库，，达到ad-hoc查询的速度。Knowledge Grid层介入的是query优化和执行阶段，减少数据读入量和解压缩开销。

这篇论文中的brighthouse就是商业数据仓库infobright。

#### 介绍

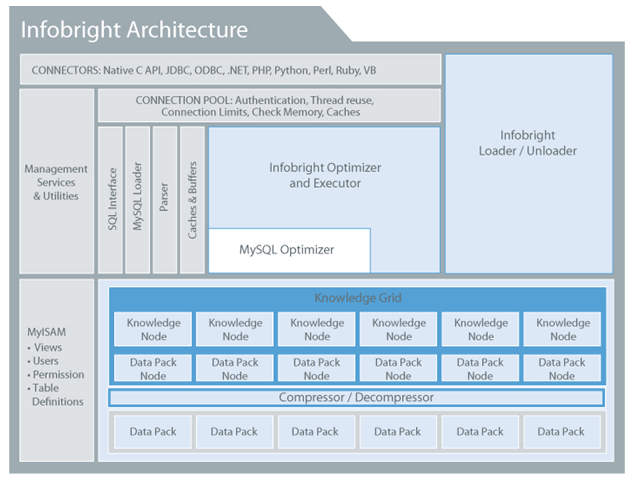
面向列的架构比较适合分析型数据仓库，面向行的架构比较适合OLTP系统。brighthouse是面向列的。

Knowledge Gird是一个data about data的思路，提供一些类似数据的统计信息，来帮助query优化和执行阶段取得想要的数据，这是brighthouse设计最核心的部分。从层次上说，介于query优化、执行层和数据(压缩)存储层之间。而且Knowledge Grid存储的元数据非常小，完全可以存在内存里。

Knowledge Grid由Knowledge Nodes组成，每个Node上记录了压缩数据的元数据信息，这些实际数据存储在Data Packs里，数据量比较庞大，是按列存的，不做分区(这部分信息由Knowledge Node维护)，且压缩过。所以Data Packs相当于代表了brighthouse的数据存储模型，而Knowledge Grid类似元数据层。

#### 架构和模块

架构图：



灰色部分是mysql原有的模块，白色与蓝色部分则是 infobright自身的。

跟mysql一样的两层结构，上面的逻辑层处理查询逻辑，下面的是存储引擎。

通过这张图主要说明几个部分，数据导入导出、Data Pack、Knowledge Grid、优化和执行。

逻辑层右端的loader与unloader是infobright的数据导入导出模块，是一个独立的服务。

存储层最底层是Data Pack。每一个Pack装着某一列的64K个元素，所有数据按照这样的形式打包存储，Data Pack根据不同数据类型采用不同的压缩算法，压缩比很高。

Knowledge Grid里面包含两类结点：

每个Data Pack Node对应一个Data Pack，存储一些统计信息，如min, max, avg, null的个数，总个数等；

Knowledge Node存储了一些更高级的统计信息，以及与其它表的连接信息，这里面的信息有些是数据载入时已经算好的，有些是随着查询进行而计算的，所以说是带自动化的。

Knowledge Grid里面还存了这样几种数据信息。

1. Histograms(HISTS)，为数字型的列创建的柱状图。以二进制的方式存。
2. Character Maps(CMAPS)，为字母型的列创建的信息，比如String里面各个字母出现的情况。
3. Pack-to-Packs，这部分是为join型的操作准备的，关联了两张table的某条件下两个column值。

应用方面，

HISTS适合between语句，因为柱状图表达了最大，最小，range内分别的信息。

CMAPS适合LIKE语句，因为是和字母相关的。

Pack-to-Packs适合join操作，为join操作提供适合条件的对应table的Row编号。

以上大致说明了几张数据统计信息表示和适合的场景。

query优化和执行方面，参考了粗糙集的思路来设计，把数据分为相关，不相关，怀疑三种类型，对应正向region，负向region和boundary region。

#### 总结

Infobright作为开源的MySQL数据仓库解决方案，引入了列存储方案，高强度的数据压缩，优化的统计计算等内容，本文是摘抄了infobright论文里最重要的设计点，Knowledge Gird是infobright设计上的核心。

## Sybase IQ

Sybase可以说是列式数据库的先驱，Sysbase IQ 15 就是Sybase 目前最新的列式数据库。它具有强大的功能，包括数据的快速加载、超高速的分析性能、强大的业务智能分析、领先的数据建模能力等等。